Полтавський університет економіки і торгівлі
Навчально-науковий інститут денної освіти
Форма навчання денна
Кафедра комп’ютерних наук та інформаційних технологій

Допускається до захисту
Завідувач кафедри
_____Олена ОЛЬХОВСЬКА

«_____»_____2026 р.

КВАЛІФІКАЦІЙНА РОБОТА
на тему
«РОЗРОБКА ІНФОРМАЦІЙНОГО ВЕБСАЙТУ ПО ГРІ WORLD OF TANKS»

зі спеціальності 122 «Комп’ютерні науки»
освітня програма «Комп’ютерні науки»
ступеня бакалавр

Виконавець роботи Климов Артур Володимирович
_____ «___»_____2026р.
(підпис)
Науковий керівник к.ф.-м.н., Олексійчук Юрій Федорович
_____ «___»_____2026р.
(підпис)

ПОЛТАВА 2026 р.

РЕФЕРАТ

Записка: 69 ст., 32 мал., 1 додаток (на 66 ст.), 1 таблиця, 24 джерел.

ВЕБСАЙТ, ІНТЕРФЕЙС, Firebase, HTML, CSS, JAVASCRIPT, REACT, КЛІЄНТ-СЕРВЕР.

Об’єктом розробки – інформаційний вебсайт, присвячений комп’ютерній грі World of Tanks, що реалізує базу даних і динамічний інтерфейс для перегляду, редагування та адміністрування інформаційних матеріалів.

Мета роботи – розроблення та програмна реалізація інформаційного вебресурсу з інтерактивним інтерфейсом і підключенням до бази даних Firebase, який забезпечує користувачам доступ до систематизованих матеріалів, пов’язаних із грою World of Tanks.

Методи дослідження – аналіз і синтез літературних джерел та існуючих сайтів, методи проєктування клієнт-серверних вебзастосунків, використання технологій HTML, CSS, JavaScript та бази даних Firebase для реалізації інтерфейсу.

У роботі здійснено інформаційний огляд сучасних вебтехнологій і платформ для створення сайтів, проведено аналіз наявних сайтів, присвячених грі World of Tanks, визначено їхні переваги та недоліки. Розроблено архітектуру сайту, описано його структуру та принцип роботи, створено базу даних у Firebase. Програмно реалізовано інтерфейс сайту та забезпечено взаємодію з базою даних. Проведено тестування розробленої системи, що підтвердило коректність роботи основного функціоналу.

Результати роботи можуть бути використані для створення подібних вебзастосунків з динамічним інтерфейсом та базою даних у середовищі Firebase.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

Умовні позначення, символи, скорочення, терміни	Пояснення умовних позначень, скорочень, символів
API	інтерфейс, що дозволяє програмам обмінюватися даними
HTML	мова розмітки гіпертексту для створення структури вебсторінок
CSS	мова стилів для оформлення зовнішнього вигляду вебсторінок
JavaScript	мова програмування для реалізації логіки та інтерактивності вебзастосунків
DOM	об’єктна модель документа, що представляє структуру HTML-сторінки
SPA	вебзастосунок, що працює на одній сторінці без перезавантажень
MMO	масова багатокористувацька онлайн-гра
PvP	ігровий режим, у якому гравці змагаються між собою
REST	стиль створення API з використанням стандартних HTTP-запитів
PvE	ігровий режим взаємодії гравця з ігровим середовищем

ВСТУП

Актуальність. Сьогодні значна частина повсякденного життя переноситься у мережу Інтернет, що сприяє розвитку інформаційного суспільства та подоланню просторових бар'єрів. Інформаційні вебресурси стають важливим інструментом у сфері освіти, дозвілля та професійної діяльності, забезпечуючи зручний доступ до даних, їх регулярне оновлення та інтерактивну взаємодію з користувачами у режимі реального часу. Окреме місце займають тематичні сайти, присвячені комп'ютерним іграм, адже вони дозволяють створювати віртуальні спільноти, обмінюватися досвідом, стратегіями та навчальними матеріалами. Гра World of Tanks як одна з найпопулярніших онлайн-ігор сучасності має широку аудиторію гравців, які активно користуються допоміжними інформаційними ресурсами. У зв'язку з цим актуальним завданням є розробка сучасного інформаційного сайту з динамічним інтерфейсом, інтеграцією з базою даних та доступом до мультимедійного й аналітичного контенту.

Мета роботи Проєктування та програмна реалізація інформаційного вебсайту, присвяченого грі World of Tanks, із використанням сучасних вебтехнологій і бази даних Firebase, що забезпечує інтерактивний інтерфейс та підтримку динамічного контенту.

Об'єктом розробки є процес розробки інформаційного вебсайту з інтерактивним інтерфейсом та динамічним контентом.

Предметом розробки є методи і засоби реалізації інтерфейсу та обміну даними між клієнтською частиною сайту і базою даних Firebase.

У роботі створено архітектуру та логіку роботи вебсайту, розроблено структуру бази даних, реалізовано інтерфейс і забезпечено інтеграцію з базою даних Firebase.

Новизна роботи полягає в тому, що у роботі запропоновано поєднання сучасних вебтехнологій (React, Firebase, HTML, CSS, JavaScript) та хмарної бази даних для створення динамічного інформаційного ресурсу з інтерактивними можливостями. Особливістю розробки є використання 3D-моделей танків безпосередньо у вебінтерфейсі, а також інтеграція з зовнішніми аналітичними

сервісами, що дозволяє відображати статистику та допоміжні дані з інших суміжних ресурсів. Це забезпечує користувачам не лише довідкову інформацію, а й розширений інструментарій для аналізу та взаємодії з контентом.

Практична цінність полягає в можливості використання розробленого сайту як інформаційного ресурсу для спільноти гравців, а також як навчального прикладу створення вебзастосунків із хмарним зберіганням даних.

Робота складається з чотирьох розділів. У першому розділі наведено постановку задачі та визначено мету і завдання розробки. У другому розділі проведено інформаційний огляд сучасних вебтехнологій, архітектури клієнт-серверних застосунків і аналіз існуючих сайтів, присвячених грі World of Tanks.

У третьому розділі подано теоретичні засади проєктування вебсайту, описано його архітектуру та базу даних. У четвертому розділі наведено практичну реалізацію вебсайту, описано основний функціонал, результати тестування і демонстрації роботи системи.

Обсяг пояснювальної записки: 69 сторінок, з них основна частина 61 сторінок, список використаних джерел 25 найменувань, додатки 1.

РОЗДІЛ 1. ПОСТАНОВКА ЗАВДАННЯ

Сучасний розвиток інтернет-технологій дає змогу створювати інтерактивні вебресурси, які швидко оновлюються та надають користувачам зручний доступ до інформації з будь-якої точки світу. Особливої популярності набули інформаційні сайти, що об’єднують навколо себе спільноти за інтересами. Однією з таких спільнот є гравці популярної гри World of Tanks, для яких важливим є доступ до актуальних матеріалів, гайдів, новин і аналітики, зібраних у зручному форматі. Незважаючи на велику кількість ресурсів у мережі, більшість з них або перевантажені зайвим контентом, або не дають можливості швидко додавати і редагувати дані, не мають зручної структури чи адаптивного інтерфейсу. Це створює потребу в розробці легкого, сучасного та функціонального інформаційного сайту, який забезпечить доступ до бази даних і дасть можливість швидко оновлювати інформацію.

Під час постановки задачі визначено початкові дані, що використовуватимуться для реалізації сайту:

- набір інформаційних матеріалів про гру World of Tanks (новини, гайди, довідкові дані);
- облікові записи користувачів (адміністратор-редактор, відвідувачі);
- структура бази даних, що зберігатиме інформацію у хмарному середовищі Firebase;
- технологічні засоби для створення інтерфейсу: HTML, CSS, JavaScript.

Завданням є створення вебсайту з такими основними можливостями:

- відображення даних з бази у вигляді структурованих розділів та карток;
- можливість додавання і редагування записів через інтерфейс сайту;
- збереження та оновлення даних у базі Firebase;
- адаптивний та зручний інтерфейс для різних пристроїв;
- мінімальний час завантаження сторінок і стабільна робота.

Очікуваним результатом є повністю працездатний прототип інформаційного сайту про гру World of Tanks, який має зрозумілу структуру, інтерактивний інтерфейс та взаємодіє з базою даних Firebase для зберігання і оновлення інформації.

Реалізація цього проєкту дасть змогу продемонструвати повний цикл розробки вебзастосунку, від аналізу потреб користувачів і проєктування архітектури до написання коду, інтеграції з базою даних та тестування готового продукту.

РОЗДІЛ 2. ІНФОРМАЦІЙНО-АНАЛІТИЧНИЙ ОГЛЯД

2.1 Історія розвитку та популярність гри World of Tanks

World of Tanks – це масова багатокористувацька онлайн-гра жанру MMO-екшену, присвячена бронетанковій техніці середини XX століття. Гра поєднує елементи аркадного екшену, тактичного планування та командної взаємодії, що забезпечило їй широку популярність серед гравців у всьому світі. Проєкт був розроблений та випущений компанією Wargaming, яка спеціалізується на створенні багатокористувацьких військово-історичних ігор [1].

Розробка World of Tanks розпочалася у другій половині 2000-х років у період, коли ринок онлайн-ігор активно зростав, а жанр MMO здебільшого асоціювався з фентезійними або науково-фантастичними світами. Команда Wargaming зробила ставку на відносно малозаповнену нішу, історичну бронетанкову тематику, поєднану з доступним ігровим процесом. Основною ідеєю стало створення динамічних командних боїв на танках, які б не вимагали від гравців глибоких симуляторних знань, але водночас зберігали відчуття автентичності техніки та тактики [2].

Офіційний реліз гри відбувся у 2010 році, після чого World of Tanks почала стрімко набирати популярність, насамперед у Європі та країнах пострадянського простору. Важливим чинником успіху стала модель free-to-play, яка дозволяла безкоштовно приєднатися до гри з можливістю добровільних внутрішніх покупок. Такий підхід значно розширив аудиторію та зробив гру доступною для користувачів із різним рівнем технічних можливостей і досвіду.

Ігровий процес World of Tanks базується на командних боях формату 15 на 15, де кожен гравець керує окремою бойовою машиною. У грі представлено кілька класів техніки, зокрема легкі, середні та важкі танки, протитанкові самохідні установки й артилерійські системи. Кожен клас має власну роль на полі бою, що стимулює командну взаємодію та тактичне мислення. Така структура боїв сприяла

формуванню глибокої ігрової мети, у якій успіх залежить не лише від індивідуальної майстерності, а й від злагоджених дій команди [3].

Значну увагу розробники приділили історичній складовій. У World of Tanks представлена бронетехніка різних країн, зокрема СРСР, США, Німеччини, Великої Британії, Франції, Японії, Китаю, Чехословаччини, Польщі, Швеції та Італії (мал. 1.1).

Мал. 1.1 – Скріншот з гри з гілками прокачки техніки

Хоча ігрові характеристики техніки адаптовані для балансу, зовнішній вигляд, назви та базові концепції машин ґрунтуються на реальних історичних зразках або інженерних проєктах. Це привернуло увагу не лише геймерів, а й шанувальників військової історії.

Протягом років існування проєкту World of Tanks зазнала суттєвого технічного та концептуального розвитку. Розробники регулярно випускали великі оновлення, які включали нові гілки техніки, карти, ігрові режими та графічні покращення. Одним із ключових етапів стало впровадження оновленого графічного рушія, що значно підвищив якість візуалізації та реалістичність ігрового середовища. Це дозволило грі зберегти конкурентоспроможність навіть через багато

років після релізу. Важливу роль у популяризації World of Tanks відіграла активна спільнота гравців та розвиток кіберспортивного напряму. Турніри, ліги та офіційні чемпіонати сприяли формуванню професійної сцени та підвищенню інтересу до гри з боку медіа. Крім того, Wargaming активно співпрацювала з контент-мейкерами, історичними музеями та освітніми проєктами, що посилювало культурну та пізнавальну цінність гри.

Одним із найяскравіших свідчень постійного розвитку та життєздатності гри стало вихід оновлення 2.0, яке розробники позиціонують як найбільше та найамбітніше за всю історію проєкту. Це масштабне оновлення не лише продовжило підтримувати зацікавленість існуючих гравців, але й привернуло увагу нових користувачів зі всього світу. У рамках цього патчу Wargaming впровадили комплексний ребаланс техніки, що охопив сотні машин усіх рівнів від початкових до найвищих, з метою покращення ігрового балансу, усунення застарілих механік та підвищення конкуренто-здатності кожного танка в бою [4].

Окрім цього, оновлення 2.0 принесло і кілька принципово нових елементів. Серед найважливіших додавання танків XI рівня, що означає розширення технологічного дерева та появу нових бойових можливостей для гравців.

Також суттєво було оновлено інтерфейс та гараж, який перетворився на своєрідний «танковий завод», де машини збираються та оснащуються, що позитивно вплинуло на зручність користування та загальне враження від гри (рис. 1.2).

Мал. 1.2 – Оновлений інтерфейс головного меню гри

До того ж у грі з’явився новий сюжетний PvE-режим з унікальною картою Nordskar, що дозволило гравцям випробовувати тактичні навички не лише в PvP-сутичках, але й у боротьбі з штучним інтелектом у спеціальних сценаріях. Новий матчмейкер, розроблений майже з нуля, оптимізує підбір команд, забезпечуючи більш збалансовані та динамічні бої, а оновлені карти, механіки та візуальні покращення лише підсилюють відчуття сучасності та якості проєкту.

Впровадження таких масштабних змін підтверджує, що World of Tanks не зупиняється на досягнутому, а продовжує розвиватися, модернізувати свій контент відповідно до очікувань спільноти та сучасних стандартів індустрії. Цей підхід дозволив грі залишатися актуальною навіть через багато років після її першого релізу, зберігаючи активну базу гравців та стабільний інтерес як ветеранів, так і новачків.

2.2 Архітектурні підходи до клієнт-серверних вебзастосунків

Сучасні клієнт-серверні вебзастосунки будуються на основі різноманітних архітектурних підходів. Архітектура визначає спосіб організації компонентів

застосунку, принципи їх взаємодії, а також розподіл відповідальності між клієнтською та серверною частинами. У процесі розвитку вебтехнологій сформувалося кілька ключових підходів, кожен з яких має власну сферу доцільного застосування, переваги та обмеження (рис. 1.3).

Одним із базових і найбільш поширених підходів є багатошарова або N-tier архітектура. Вона передбачає чіткий поділ застосунку на логічні шари, серед яких зазвичай виділяють рівень представлення, рівень бізнес-логіки та рівень збереження даних. Клієнтський інтерфейс відповідає за взаємодію з користувачем, серверна бізнес-логіка за оброблення запитів, правила та обчислення, а база даних за надійне зберігання інформації [5]. Такий поділ забезпечує кращу керованість коду, спрощує тестування та дозволяє незалежно масштабувати окремі частини системи. Водночас недоліком цього підходу є зростання складності взаємодії між шарами та потенційні затримки через багаторазові звернення між рівнями, особливо у великих розподілених системах.

Мал. 1.3 – Архітектурні підходи для створення клієнт-серверних систем

Подальшим етапом еволюції серверної частини стала мікросервісна архітектура, у якій бекенд не є єдиним монолітним застосунком, а складається з набору невеликих автономних сервісів. Кожен мікросервіс відповідає за окрему

функціональну область, має власну логіку, може використовувати окрему базу даних та розгортатися незалежно від інших. Взаємодія між сервісами зазвичай здійснюється через REST або GraphQL API [6]. Основною перевагою такого підходу є висока масштабованість, гнучкість у розвитку системи та можливість паралельної роботи різних команд. Разом із тим мікросервісна архітектура вимагає складнішої інфраструктури, ретельного управління мережевими викликами, безпекою та моніторингом, що може бути надмірним для проєктів середнього масштабу.

На клієнтському боці дедалі більшої популярності набули Single Page Application. SPA є вебзастосунками, у яких після початкового завантаження однієї HTML-сторінки подальша взаємодія з користувачем відбувається без повного перезавантаження сторінки. Оновлення контенту здійснюється динамічно за допомогою JavaScript, а обмін даними з сервером відбувається через асинхронні запити [7]. Такий підхід дозволяє досягти високої швидкості взаємодії, плавних переходів між розділами та відчуття роботи з повноцінним застосунком. Недоліком SPA вважається залежність від виконання JavaScript на стороні клієнта та складніша оптимізація для пошукових систем без додаткових механізмів.

Альтернативою SPA є підхід Server-Side Rendering, за якого повна HTML-сторінка формується на сервері та лише потім передається браузеру користувача. Це дозволяє значно покращити швидкість першого відображення контенту та забезпечує кращу індексацію сторінок пошуковими системами. SSR особливо ефективний для інформаційних ресурсів із великою кількістю текстового контенту [8]. Водночас серверне рендеринг-завантаження збільшує навантаження на сервер і ускладнює реалізацію складної клієнтської логіки, характерної для інтерактивних вебзастосунків.

Окрему нішу займають Progressive Web Apps, які поєднують можливості класичних вебсайтів і мобільних застосунків. PWA дозволяють працювати в офлайн-режимі, підтримують push-сповіщення та можуть встановлюватися на пристрій користувача без використання магазинів застосунків. Основною перевагою цього підходу є покращення залученості користувачів та розширення функціональності вебресурсу [9]. Проте реалізація PWA потребує додаткових

налаштувань і не завжди повністю підтримується всіма браузерами. Сучасним трендом є безсерверна архітектура, у межах якої розробник працює з окремими функціями, а вся інфраструктура керується хмарним провайдером, наприклад, AWS. У такому підході сервери як такі приховані від розробника, а масштабування відбувається автоматично. Це дозволяє зосередитися на бізнес-логіці та зменшити витрати на підтримку інфраструктури [10]. Проте безсерверні рішення мають обмеження щодо часу виконання функцій, складніші механізми налагодження та залежність від конкретного провайдера.

Подальшим розвитком ідей мікросервісів стала концепція micro frontends, у якій фронтенд застосунку складається з незалежних модулів, що можуть розроблятися та оновлюватися окремими командами. Такий підхід є доцільним для дуже великих систем, однак він суттєво ускладнює інтеграцію, збільшує обсяг початкових налаштувань і не є оптимальним для проєктів з обмеженим функціоналом.

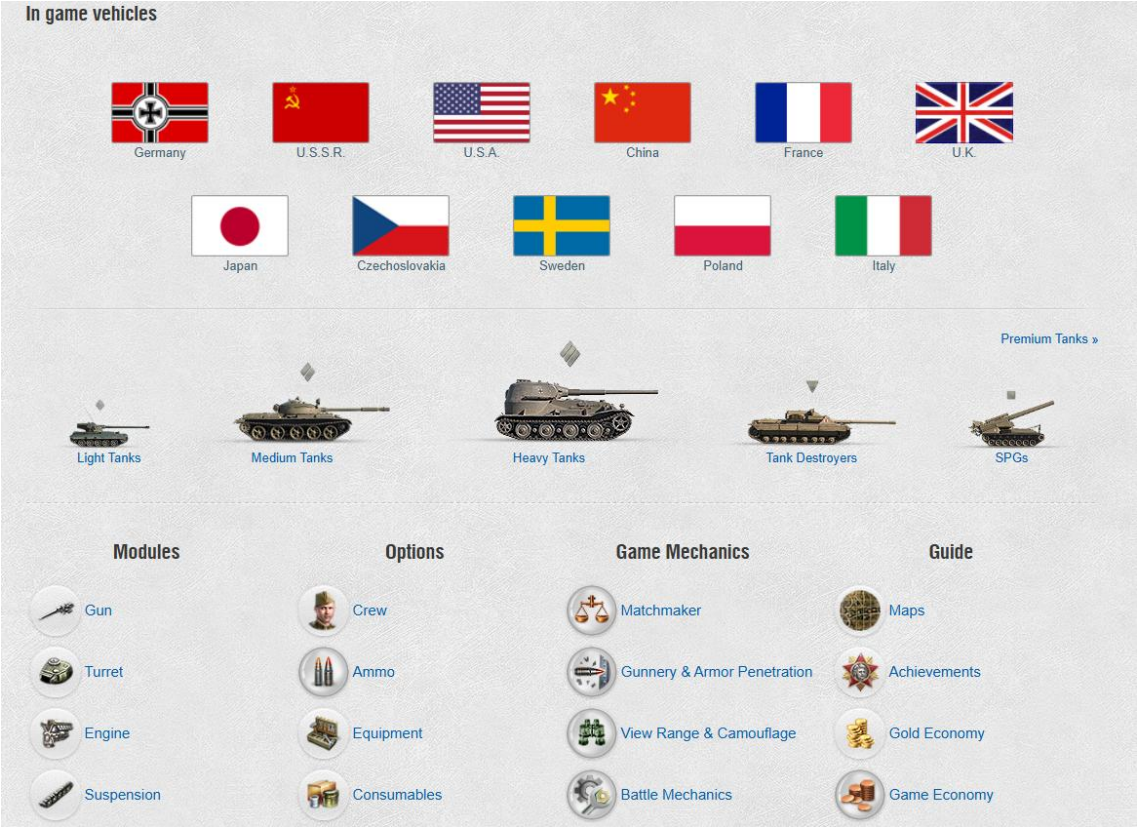
З огляду на специфіку інформаційного вебсайту, присвяченого грі World of Tanks, найбільш доцільним архітектурним вибором є Single Page Application. Такий сайт орієнтований на активну взаємодію користувача з контентом, швидку навігацію між розділами, динамічне завантаження даних про техніку, новини, оновлення та ігрові механіки. SPA дозволяє реалізувати плавний і сучасний інтерфейс без постійних перезавантажень сторінок, що особливо важливо для молодої та технічно підготовленої аудиторії гравців. Крім того, SPA добре інтегрується з хмарними сервісами та API, що дає змогу ефективно працювати з динамічними даними та масштабувати клієнтську частину без значних змін серверної логіки.

Застосування SPA-архітектури для вебсайту про World of Tanks є обґрунтованим з точки зору продуктивності, зручності користування та відповідності сучасним стандартам веброзробки, водночас залишаючи можливість подальшого розширення функціоналу без кардинальної зміни архітектурних рішень.

2.3 Огляд і порівняльний аналіз існуючих сайтів, присвячених грі World of Tanks

Для аналізу було обрано три популярні вебресурси, які активно використовуються спільнотою гравців для отримання довідкової інформації, аналітичних даних та візуалізації ігрових характеристик: World of Tanks Wiki, Tanks.gg та Tomato.gg. Кожен із зазначених сайтів має власну спеціалізацію, підхід до структурування контенту та цільову аудиторію, що дозволяє комплексно оцінити їхній потенціал як джерел для створення навчального інформаційного ресурсу в межах курсового проєкту.

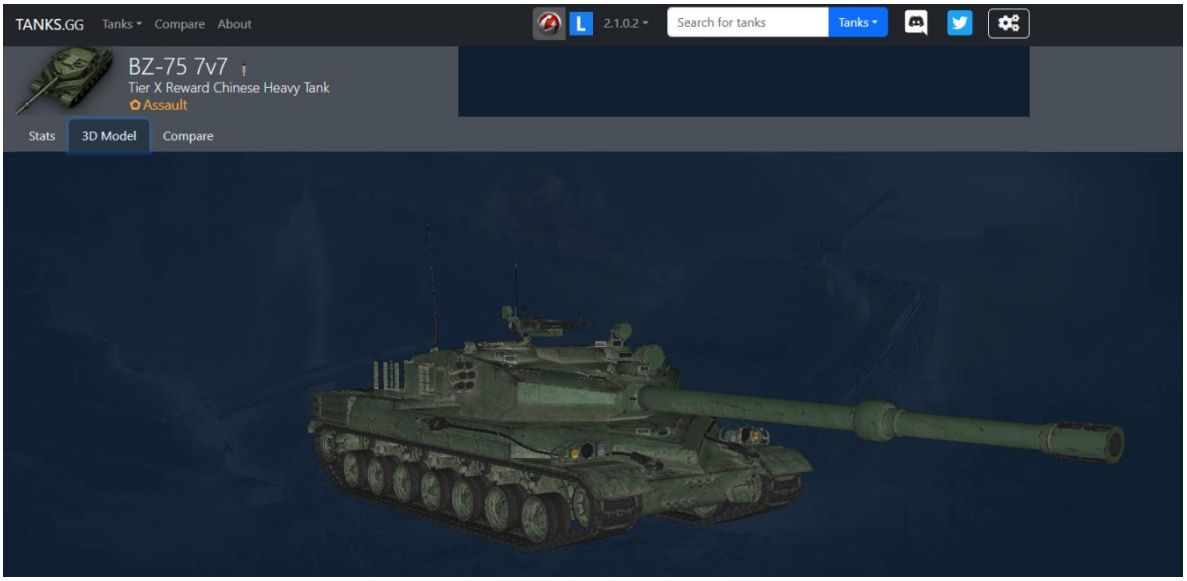
World of Tanks Wiki є офіційною енциклопедією гри, що створюється та підтримується компанією Wargaming у співпраці зі спільнотою гравців . Ресурс містить систематизовані та детальні описи бойової техніки, модулів, ігрових механік, режимів, карт і змін, які вносилися з кожним патчем (рис. 1.4). Важливою особливістю цього сайту є орієнтація на довідкову та навчальну інформацію, що робить його зручним для початкового ознайомлення з грою та вивчення її базових принципів [11].



Мал. 1.4 – Частина головної сторінки World of Tanks Wiki

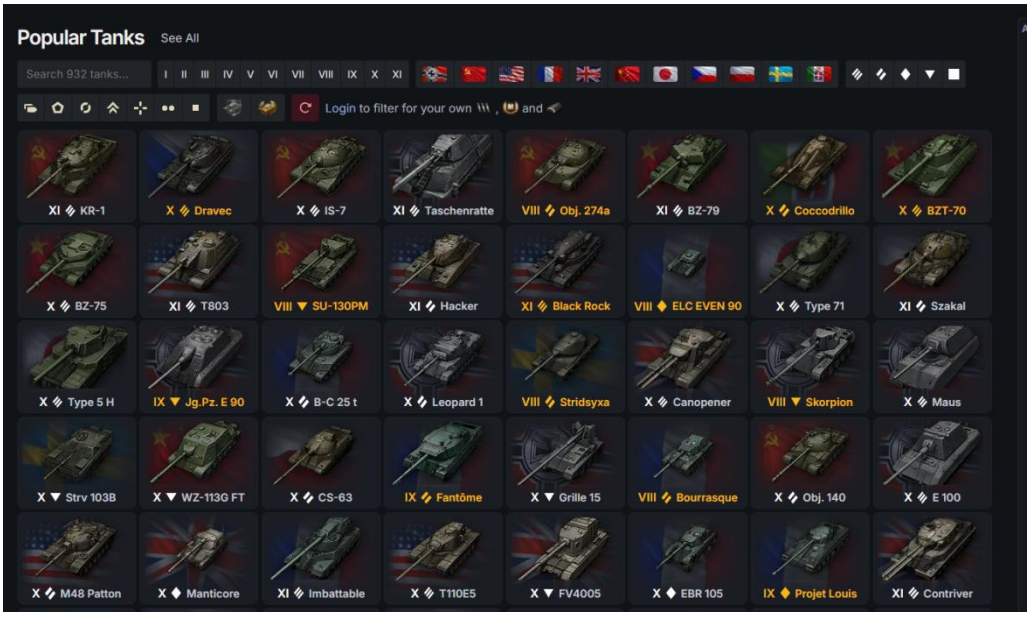
Разом із тим, через значний обсяг матеріалів і складність внутрішньої модерації, оновлення інформації після виходу великих патчів може відбуватися з певною затримкою, а сам контент майже не містить практичних рекомендацій щодо актуальної ігрової «мети» чи оптимальних тактичних рішень.

Tanks.gg є незалежним фанатським ресурсом, головною цінністю якого є потужні інструменти візуалізації. Сайт надає інтерактивні тривимірні моделі танків, схеми бронювання, розташування внутрішніх модулів і екіпажу, а також інструменти для порівняння техніки за різними характеристиками (рис. 1.5). Завдяки цьому Tanks.gg широко використовується гравцями для аналізу сильних і слабких сторін бойових машин, підбору обладнання та вивчення ефективних зон ураження [12]. Попри високу наочність і практичну корисність, ресурс не є офіційним, що зумовлює необхідність періодичної перевірки актуальності даних, особливо після масштабних ребалансів техніки.



Мал. 1.5 – 3D модель техніки з ресурсу Tanks.gg

Tomato.gg орієнтований насамперед на статистичний аналіз. Портал агрегує велику кількість даних щодо ефективності гравців і бойової техніки, зокрема показники WN8, відсоток перемог, середню шкоду, статистику по мапах та вимоги до отримання внутрішньоігрових відзнак (рис. 1.6) [13].



Мал. 1.6 – Статистика популярності техніки з ресурсу Tomato.gg

Такий підхід дозволяє використовувати сайт як інструмент для аналізу власного прогресу та порівняння техніки з позицій результативності. Водночас Tomato.gg майже не містить довідкових матеріалів або пояснень ігрових механік, а

актуальність даних напряму залежить від стабільності роботи API, що може спричиняти затримки в оновленні статистики.

Основні характеристики, переваги та обмеження розглянутих ресурсів узагальнено в таблиці 1.1.

Таблиця 1.1 – Порівняння сайтів-аналогів

Критерій	**World of Tanks Wiki**	**Tanks.gg**	**Tomato.gg**
Спрямованість	Офіційна енциклопедія гри	Візуалізація техніки та ТТХ	Аналітика та статистика
Тип контенту	Опис техніки, механік, патчів	3D-моделі, бронювання, порівняння	Рейтинги, WN8, статистика
Рівень достовірності	Високий (офіційні дані)	Середній–високий	Високий, залежний від API
Оновлюваність	Помірна	Висока	Залежить від API
Візуалізація даних	Обмежена	Дуже висока	Середня
Навчальна цінність	Висока для новачків	Практична, прикладна	Аналітична
Зручність навігації	Висока	Висока	Середня
Орієнтація на «мету»	Низька	Середня	Висока
Корисність для проєкту	Довідкові дані	3D-ілюстрації, порівняння	Статистика, графіки

Проведений аналіз свідчить, що кожен із розглянутих ресурсів вирішує власне коло завдань і не може бути повноцінно замінений іншим. Саме їх поєднання формує комплексне інформаційне середовище для гравців, що охоплює як теоретичні знання, так і практичні та аналітичні аспекти ігрового процесу.

Матеріали з цих сайтів можуть бути використані не лише як джерела інформації, а й як орієнтир для розширення функціональності майбутньої вебсистеми. Зокрема, інтеграція спрощених 3D-моделей танків за аналогією з Tanks.gg, візуалізація бронювання або додавання статистики мап і техніки на основі підходів Tomato.gg можуть значно підвищити практичну цінність розроблюваного ресурсу. Водночас довідкові описи та структурований контент World of Tanks Wiki можуть слугувати основою для формування навчального наповнення.

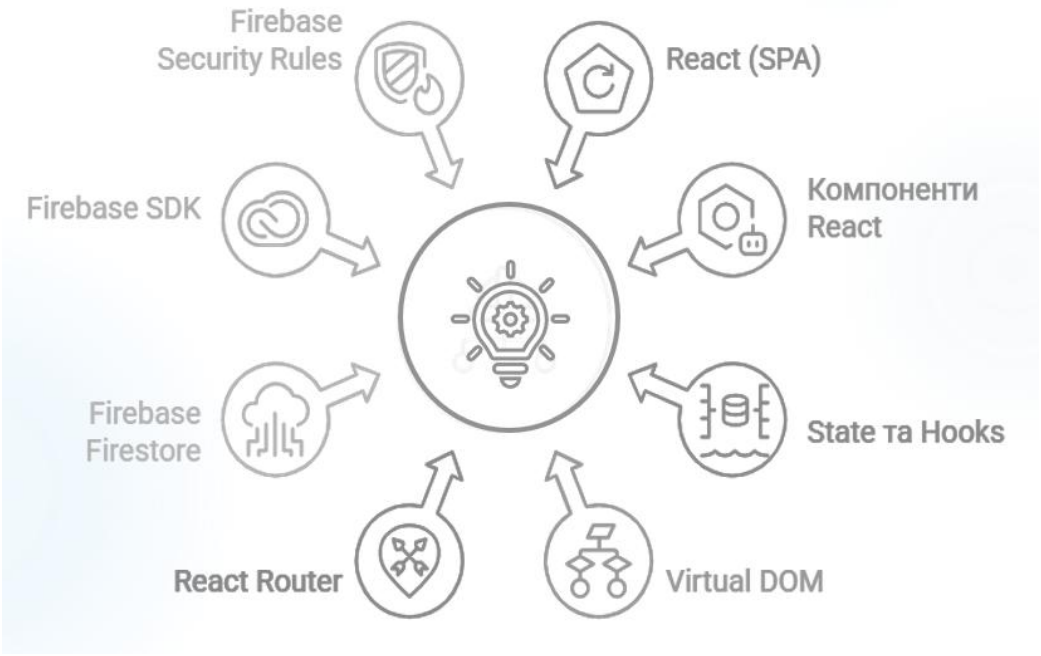
Таким чином, створюваний у межах кваліфікаційної роботи інформаційний сайт не має на меті прямої конкуренції з існуючими масштабними проєктами, які розробляються великими командами з використанням значних фінансових і

технічних ресурсів. Основним завданням є закріплення навичок веброзробки, проєктування архітектури, роботи з базами даних і реалізації інтерфейсів, а також демонстрація можливостей подальшого розширення системи функціоналом, орієнтованим на візуалізацію та аналітику даних.

РОЗДІЛ 3. ТЕОРЕТИКО-ПРОЄКТНА ЧАСТИНА

3.1. Загальна архітектура та принципи роботи сайту

Загальна архітектура розробленого вебсайту побудована на основі клієнт-серверного підходу з використанням технології React та моделі SPA (рис. 2.1). Обраний підхід дозволяє реалізувати сучасний інтерактивний інтерфейс, у якому взаємодія користувача з сайтом відбувається без повного перезавантаження сторінок. Усі основні переходи між розділами здійснюються на клієнтській стороні, що значно підвищує швидкість роботи застосунку та покращує загальний користувацький досвід.



Мал. 2.1 – Ключові компоненти архітектури системи

React у межах даного проєкту виконує роль основного інструмента формування інтерфейсу. Вся структура сайту поділена на окремі React-компоненти, кожен з яких відповідає за конкретний фрагмент інтерфейсу або логіки. До таких компонентів належать окремі сторінки сайту, інформаційні блоки, списки техніки, карт або новин, а також допоміжні елементи навігації. Компонентний підхід

забезпечує модульність архітектури, спрощує повторне використання коду та полегшує підтримку і розширення функціоналу сайту в майбутньому.

Керування даними всередині компонентів здійснюється за допомогою механізмів стану (state) та React Hooks. Стан компонентів використовується для збереження поточних даних, зокрема списків об’єктів, параметрів фільтрації, результатів запитів до бази даних та інших змінних, що впливають на відображення інтерфейсу [14]. Hooks дозволяють керувати життєвим циклом компонентів, обробляти асинхронні операції та реагувати на зміну даних. Такий підхід забезпечує чіткий зв’язок між логікою оброблення інформації та її візуальним представленням, що є важливим для динамічних інформаційних ресурсів.

Важливою складовою архітектури React є використання механізму Virtual DOM. Замість повного оновлення HTML-сторінки при кожній зміні даних React формує віртуальне представлення DOM-дерева та порівнює його з попереднім станом. У результаті в реальний DOM вносяться лише ті зміни, які дійсно необхідні [15]. Це дозволяє суттєво зменшити кількість операцій перерендерингу та забезпечує високу швидкодію застосунку навіть при роботі з великими обсягами динамічної інформації.

Навігація між сторінками сайту реалізована за допомогою бібліотеки React Router, яка забезпечує маршрутизацію на клієнтській стороні. Кожен розділ сайту має власний маршрут, що дозволяє зберігати логічну структуру вебресурсу та підтримувати зрозумілі URL-адреси [16]. При цьому перехід між сторінками не потребує звернення до сервера для завантаження нової HTML-сторінки, а відбувається миттєво в межах одного застосунку. Такий підхід є характерним для SPA-архітектури та добре підходить для інформаційних сайтів із великою кількістю розділів і взаємопов’язаного контенту.

Для зберігання та отримання динамічних даних у проєкті використовується хмарна база даних Firebase Firestore. Firestore є документоорієнтованою NoSQL-базою даних, що дозволяє зберігати структуровану інформацію у вигляді колекцій і документів [17]. Така модель добре підходить для роботи з даними про ігрову техніку, карти, новини або інші інформаційні об’єкти, оскільки не потребує жорстко

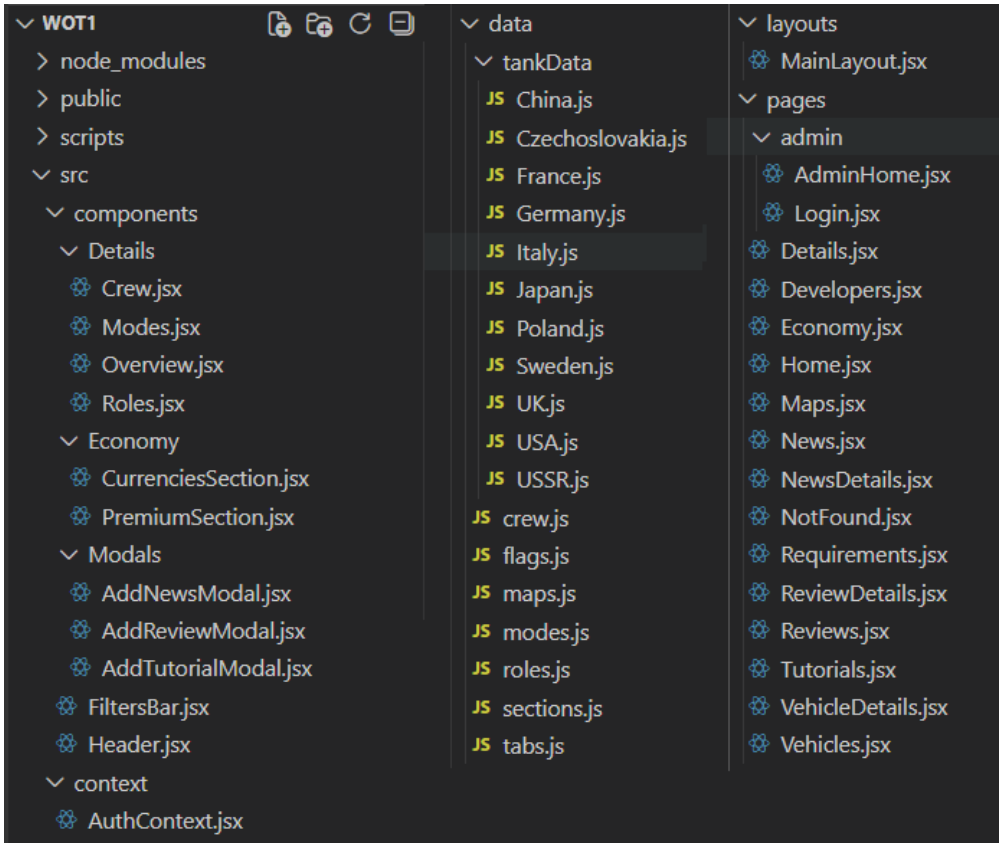
фіксованої схеми та легко масштабується. Отримання даних із Firestore відбувається в режимі реального часу, що дозволяє автоматично оновлювати інтерфейс сайту при зміні інформації в базі.

З’єднання клієнтського вебзастосунку з сервісами Firebase здійснюється за допомогою Firebase SDK. SDK надає програмні інтерфейси для роботи з базою даних, автентифікацією та іншими сервісами платформи [18]. Використання офіційного SDK спрощує інтеграцію, зменшує кількість низькорівневого коду та забезпечує стабільну взаємодію між фронтендом і хмарною інфраструктурою. Окрему увагу в архітектурі сайту приділено безпеці даних. Для цього застосовуються Firebase Security Rules, які визначають правила доступу до колекцій і документів бази даних. За допомогою цих правил обмежується можливість читання та запису інформації залежно від ролі користувача або типу запиту [19]. Такий підхід дозволяє захистити дані від несанкціонованого доступу та забезпечити контроль за коректністю взаємодії клієнтської частини з базою даних без необхідності реалізації окремого серверного бекенду.

У сукупності використані технології формують цілісну, масштабовану та зрозумілу архітектуру вебзастосунку. Поєднання React як клієнтської платформи, SPA-підходу, хмарної бази даних Firestore та вбудованих механізмів безпеки Firebase дозволяє реалізувати сучасний інформаційний сайт із високою швидкодією, гнучкою структурою та можливістю подальшого розширення функціоналу без суттєвих архітектурних змін.

3.2. Структура проєкту та логіка взаємодії компонентів

Структура проєкту вебсайту побудована з урахуванням принципів модульності, зрозумілості та зручності супроводу, що є характерним для сучасних SPA-застосунків на основі React. Усі основні робочі файли зосереджені в директорії src, яка містить логічно впорядковані підкаталоги, кожен з яких відповідає за окремий аспект функціонування сайту (рис. 2.2).



Мал. 2.2 – Структура проєкту та ключові компоеннти

Каталог components містить багаторазові React-компоненти, які використовуються на різних сторінках сайту. До цієї групи належать як загальні елементи інтерфейсу, так і спеціалізовані функціональні блоки. Зокрема, компонент Header.jsx реалізує шапку сайту з навігаційним меню та забезпечує швидкий перехід між основними розділами. Компоненти з підкаталогу Details відповідають за відображення деталізованої інформації про ігрові елементи, такі як ролі, режими гри, екіпаж та загальний огляд механік. Папка Economy містить компоненти, пов’язані з економічною системою гри, включно з описом внутрішньоігрових валют і преміум-механік. Окремо виділено папку Modals, у якій зосереджені модальні вікна для додавання новин, оглядів і навчальних матеріалів, що дозволяє реалізувати взаємодію з користувачем без переходу на окремі сторінки. Такий підхід підвищує зручність інтерфейсу та зменшує дублювання коду.

Каталог context використовується для зберігання глобальних контекстів стану застосунку. У файлі AuthContext.jsx реалізовано логіку автентифікації користувачів

із використанням сервісів Firebase. Контекст дозволяє централізовано зберігати інформацію про поточного користувача, його стан входу в систему та права доступу, а також надавати ці дані всім компонентам, які цього потребують. Завдяки використанню Context API усувається необхідність передавати параметри через велику кількість вкладених компонентів, що спрощує архітектуру та покращує читабельність коду.

У папці layouts зосереджені компоненти шаблонів сторінок. Основним із них є MainLayout.jsx, який визначає загальний каркас сайту, включно з хедером, областю для відображення контенту та спільними елементами оформлення. Використання layout-компонентів дозволяє забезпечити єдиний стиль і структуру для всіх сторінок, а також централізовано змінювати вигляд сайту без редагування кожної сторінки окремо.

Каталог pages містить набір сторінок, кожна з яких відповідає окремому розділу вебсайту та є логічною одиницею маршрутизації. Файл Home.jsx реалізує головну сторінку з базовою інформацією про гру та навігаційними елементами. Сторінка Developers.jsx присвячена інформації про студію-розробника Wargaming. У Economy.jsx подано матеріали щодо економіки гри, тоді як Maps.jsx використовується для огляду ігрових карт. Розділ новин реалізований за допомогою сторінок News.jsx та NewsDetails.jsx, які забезпечують перегляд списку публікацій і детального вмісту окремої новини. Аналогічний підхід застосовано для оглядів і навчальних матеріалів у файлах Reviews.jsx, ReviewDetails.jsx та Tutorials.jsx.

Окрему роль відіграє сторінка Vehicles.jsx, яка реалізує динамічну енциклопедію бойової техніки, а також VehicleDetails.jsx, що відповідає за відображення детальної інформації про конкретну одиницю техніки. Файл Details.jsx використовується для узагальненого відображення ключових ігрових аспектів, тоді як Requirements.jsx містить інформацію про системні вимоги до гри. Сторінка Login.jsx призначена для входу користувачів до адміністративної частини, а AdminHome.jsx для базового керування контентом. Файл NotFound.jsx реалізує сторінку помилки 404, яка відображається у випадку переходу за неіснуючим маршрутом.

Окремо в структурі проєкту присутній каталог data, у якому зберігаються підготовлені набори даних, що використовуються для наповнення сторінок сайту. Ці файли містять структуровану інформацію, яка підключається до компонентів для відображення контенту, однак безпосередньо в межах даного підрозділу логіка формування цих даних не розглядається.

Ключовими системними файлами проєкту є App.jsx та firebase.js. Файл App.jsx виступає головним компонентом застосунку, у якому налаштована маршрутизація між сторінками за допомогою клієнтського роутера, а також визначено, які layout-компоненти використовуються для різних груп сторінок. Файл firebase.js відповідає за ініціалізацію та налаштування підключення до сервісів Firebase, включно з базою даних і механізмами автентифікації, та використовується у всіх частинах застосунку, де потрібна взаємодія з хмарною інфраструктурою.

Запропонована структура проєкту забезпечує чіткий поділ відповідальності між компонентами, сторінками та сервісною логікою, спрощує навігацію в кодовій базі та створює основу для подальшого масштабування й розширення функціоналу сайту без порушення загальної архітектури.

3.3. Проєктування бази даних та збір ключової інформації

Архітектура даних будується на Firebase (Cloud Firestore), який працює зі структурою колекцій і документів, а не класичними реляційними таблицями. Для наочності модель подано у вигляді «таблиць» (рис. 2.3), але в реалізації це три колекції: reviews, news, beginners. Для адміністративних операцій застосовується Firebase Authentication (email/password), що надає вхід адміністратору і дозволяє змінювати вміст колекцій відповідно до правил доступу [20].

Колекція reviews призначена для зберігання оглядів гри та пов’язаного мультимедійного контенту. Кожен документ у цій колекції містить унікальний ідентифікатор, назву огляду, тип матеріалу (текстовий або відео), посилання на джерело, короткий опис для попереднього перегляду та часовий штамп створення.

Така структура дозволяє зручно сортувати огляди, відображати їх у вигляді списків або карток і забезпечувати швидкий доступ до повної версії матеріалу.

Мал. 2.3 – Структура колекцій бази даних

Колекція news використовується для публікації новин, пов’язаних із грою World of Tanks. Окрім ідентифікатора та заголовка, документи містять посилання на ілюстрацію, основний текст новини, джерело інформації та дату публікації. Зберігання зображень реалізовано через хмарне сховище Firebase Storage, а в самій колекції зберігаються лише URL-адреси, що зменшує обсяг даних у Firestore та підвищує ефективність роботи.

Колекція beginners орієнтована на розділ для новачків і містить навчальні матеріали різного типу. Поле category використовується для маршрутизації та фільтрації контенту між порадами, гайдами, FAQ та глосарієм. Це дозволяє формувати динамічні сторінки без дублювання логіки та забезпечує зручне масштабування навчального розділу в майбутньому.

Окрім динамічних даних, що зберігаються у Firestore, важливою частиною проєкту є інформація про бойову техніку, карти та інші ігрові елементи. Офіційне API гри World of Tanks на момент розробки проєкту не підтримується або не надає відкритого доступу для сторонніх навчальних ресурсів, у зв’язку з чим отримання цих даних було реалізовано шляхом парсингу HTML-сторінок офіційного сайту гри.

Збір даних здійснювався за допомогою середовища Node.js та бібліотеки Cheerio, яка дозволяє обробляти HTML-документи з використанням синтаксису, подібного до jQuery [21]. Алгоритм збору інформації складався з кількох послідовних етапів. На першому етапі HTML-код відповідної сторінки повністю зчитувався з локального файлу та завантажувався в парсер. Далі виконувалося вибіркове проходження DOM-дерева з метою вилучення лише тих елементів, які містили корисну інформацію, наприклад назви танків та зображення:

```
const html = fs.readFileSync("germany.html", "utf8");
const $ = cheerio.load(html);
$(".grid_item").each((i, el) => {
  const name = $(el).find(".grid_text").text().trim();
  let img = $(el).find("img.grid_image").attr("src");
  if (img && img.startsWith("//")) img = "https:" + img;
  tanks.push({ name, img });
});
```

На наступному етапі з тієї ж HTML-сторінки зчитувалися табличні дані з детальними характеристиками техніки. Алгоритм проходив по кожному рядку таблиці, вилучав тип техніки, рівень, основні числові параметри та ознаку преміум-статусу. Отримані значення очищувалися, нормалізувалися та приводилися до числового формату для подальшого використання в інтерфейсі:

```
$("a.table-view_row").each((i, el) => {
  const tier = $(el).find("span.table-view_text").first().text().trim();
  const params = $(el).find(".table-view_text").map((i, s) => $(s).text()).get();
  extra.push({ tier, params });
});
```

Після цього дані з різних етапів парсингу об’єднувалися за індексами, що відповідали порядку розташування елементів на сторінці. Таким чином формувався єдиний масив об’єктів, у якому кожен танк містив як базову інформацію, так і розширені характеристики. Результат зберігався у вигляді JavaScript-масиву з експортом, що дозволяло безпосередньо використовувати ці дані в React-компонентах без додаткових запитів до сервера.

Аналогічний підхід було застосовано і для оброблення даних про ігрові карти. Окремо формувалися масиви з основною інформацією та зображеннями мінікарт,

після чого вони синхронізувалися між собою за порядковими індексами та об’єднувалися в єдину структуру:

```
const updatedMaps = Maps.map((map, i) => ({
  ...map,
  miniMap: MiniMaps[i]?.miniMap || ""
}));
```

У підсумку було сформовано набір структурованих локальних даних, які використовуються для наповнення сайту інформацією про техніку та карти без прямої залежності від зовнішніх сервісів.

Слід зазначити, що описаний метод збору даних шляхом парсингу HTML-сторінок не є рекомендованим для промислових або комерційних проєктів, оскільки він залежить від структури сторінок стороннього ресурсу та може втратити працездатність після змін у верстці. Водночас у межах навчального проєкту цей підхід є виправданим і дозволяє продемонструвати практичні навички роботи з даними, алгоритмічної обробки HTML та підготовки інформації до використання у вебзастосунку. За потреби актуалізації інформації дані можуть бути оновлені шляхом повторного запуску розроблених алгоритмів, тим більше що масштабні оновлення характеристик техніки та карт виходять нечасто.

При цьому статистичні показники та 3D-моделі техніки у проєкті не зберігаються локально, а використовуються безпосередньо з ресурсів сайтів-аналогів, що дозволяє зменшити обсяг власної бази даних і зосередитися на навчальних та архітектурних аспектах розробки вебсистеми.

РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА

4.1. Інструменти програмної реалізації

Розробка вебсайту здійснювалася з використанням сучасного стеку вебтехнологій, які забезпечують створення інтерактивних, масштабованих та зручних у підтримці клієнт-серверних застосунків. Обрані інструменти охоплюють як базові мови веброзробки, так і спеціалізовані фреймворки та бібліотеки, що дозволяють значно підвищити продуктивність розробки та якість кінцевого результату.

Основою клієнтської частини вебзастосунку є мова розмітки HTML. Вона використовується для формування структури вебсторінок, визначення логічних блоків інтерфейсу, заголовків, текстового контенту, списків та мультимедійних елементів. HTML забезпечує семантичну організацію сторінок, що є важливим як для доступності ресурсу, так і для подальшої інтеграції зі стилями та сценаріями. У межах проєкту HTML-код формується переважно динамічно за допомогою React-компонентів, однак принципи семантичної розмітки зберігаються.

Для візуального оформлення сайту використовується каскадна таблиця стилів CSS. CSS відповідає за кольорову гаму, шрифти, розміщення елементів, адаптивність інтерфейсу та загальний зовнішній вигляд сторінок. Використання CSS дозволяє відокремити логіку та структуру сторінки від її презентаційного вигляду, що полегшує супровід і модифікацію дизайну. Особлива увага приділялася адаптивності, щоб сайт коректно відображався на різних розмірах екранів.

Ключовою мовою програмної логіки вебзастосунку є JavaScript. Вона використовується для оброблення подій користувача, роботи з даними, асинхронної взаємодії з базою даних та динамічного оновлення інтерфейсу. Саме JavaScript забезпечує інтерактивність сайту, дозволяючи змінювати вміст сторінок без їх перезавантаження, виконувати фільтрацію, сортування та керування станом компонентів. Крім клієнтської частини, JavaScript також застосовувався для допоміжних скриптів збору та підготовки даних.

На основі JavaScript реалізовано клієнтську архітектуру сайту з використанням бібліотеки React. React дозволяє будувати інтерфейс у вигляді незалежних компонентів, кожен з яких має власну логіку та стан. Такий підхід значно спрощує розробку складних інтерфейсів, підвищує повторне використання коду та полегшує масштабування проєкту. Завдяки концепції Virtual DOM React оптимізує оновлення інтерфейсу, що позитивно впливає на швидкодію застосунку.

Для стилізації інтерфейсу використовується CSS-фреймворк Tailwind CSS, який реалізує підхід utility-first. Замість створення великої кількості окремих CSS-класів, стилі застосовуються безпосередньо в розмітці у вигляді готових утилітарних класів [22]. Це дозволяє значно прискорити процес верстки, забезпечити єдиний стиль у межах усього сайту та зменшити обсяг кастомного CSS-коду. Tailwind CSS добре інтегрується з React і підходить для створення сучасних адаптивних інтерфейсів.

Для покращення взаємодії з користувачем і відображення системних повідомлень у проєкті використовується бібліотека SweetAlert2. Вона застосовується для відображення повідомлень про успішні дії, помилки, підтвердження операцій та інформаційні сповіщення. SweetAlert2 дозволяє замінити стандартні браузерні діалогові вікна на стилізовані модальні повідомлення, що підвищує зручність використання сайту та загальну естетику інтерфейсу [23].

Як середовище розробки було обрано Visual Studio Code, який є одним із найпопулярніших редакторів коду для веброзробки. VS Code забезпечує підтримку синтаксису HTML, CSS, JavaScript і React, має розвинену систему розширень, інструменти автодоповнення, підсвічування помилок і вбудований термінал. Використання цього середовища дозволило ефективно організувати робочий процес, спростити налагодження коду та підвищити загальну продуктивність розробки [24].

Поєднання базових вебмов (HTML, CSS, JavaScript) із сучасними інструментами та бібліотеками, такими як React, Tailwind CSS і SweetAlert2, а також використання Visual Studio Code як середовища розробки, забезпечило реалізацію функціонального, сучасного та зручного вебсайту, що відповідає вимогам навчального проєкту та сучасним стандартам веброзробки.

4.2. Опис основного функціоналу

На даному етапі розроблений вебсайт виконує функцію інформаційного та навчального посібника з гри World of Tanks. Його основне завдання полягає у наданні користувачам структурованого доступу до ключових відомостей про гру: описів ігрових механік, ролей техніки, режимів, екіпажу, карт, економіки, системних вимог та діяльності розробників. Бізнес-логіка реалізована в мінімальному обсязі, оскільки акцент зроблено на зручну навігацію, зрозумілу подачу інформації та модульну архітектуру компонентів.

Усі розділи сайту реалізовані у вигляді незалежних React-компонентів, що дозволяє легко підтримувати та розширювати функціонал без порушення загальної структури застосунку.

Навігація між сторінками реалізована в компоненті Header.jsx, який відображає верхню панель сайту. Меню описується у вигляді масиву об’єктів, де кожен елемент містить шлях маршруту та назву пункту меню:

```
const nav = [
  { to: "/details", label: "Про гру" },
  { to: "/vehicles", label: "Техніка" },
  { to: "/maps", label: "Мапи" },
  { to: "/news", label: "Новини" },
  { to: "/admin", label: "Адмін-панель" }
];
```

Такий підхід дозволяє динамічно формувати навігацію та спрощує її зміну. Для уникнення дублювання стилів використовується базовий набір класів, який застосовується до кожного пункту меню, а активний маршрут автоматично підсвічується:

```
const base =
  "text-sm sm:text-base font-medium underline-offset-4 transition-colors";
<NavLink
  to={i.to}
  className={({ isActive }) =>
    `${base} ${
      isActive
        ? "text-amber-700"
        : "text-zinc-700 hover:text-amber-600"
    }`
  }
```

```
>
  {i.label}
</NavLink>
```

Панель навігації зафіксована у верхній частині сторінки, що забезпечує постійну доступність меню під час прокрутки контенту.

Сторінка «Про гру» відкривається компонентом Overview.jsx. Тут подано базову інформацію про World of Tanks, поділену на логічні абзаци. Для візуального занурення користувача після вступного тексту інтегровано відеотрейлер з YouTube:

```
<div className="aspect-video max-w-3xl mx-auto my-6">
  <iframe
    src="https://www.youtube.com/embed/mWI-ltfKenw"
    allowFullScreen
    className="w-full h-full rounded-xl shadow"
  />
</div>
```

Інші текстові блоки пояснюють тактичну складову гри, систему розвитку техніки та роль спільноти, при цьому вся сторінка має єдину стилізацію.

Компонент Modes.jsx реалізує відображення ігрових режимів. Режими згруповані за категоріями, а перемикання між ними здійснюється за допомогою стану компонента. Додатково реалізовано фільтрацію за рівнем техніки:

```
const [tab, setTab] = useState("permanent");
const [filter, setFilter] = useState("all");

const filteredModes = MODES[tab].filter((m) =>
  filter === "all" ? true : m.level === filter
);
```

Кнопки вкладок і фільтрів формуються автоматично з масивів даних, а кожен режим відображається у вигляді окремої картки з іконкою, назвою та описом. Це дозволяє швидко орієнтуватися в доступних форматах гри.

Опис ролей техніки реалізовано в компоненті Roles.jsx. Вибір класу машин здійснюється через кнопки-перемикачі, які змінюють активний стан:

```
const [selected, setSelected] = useState(ROLES[0]);

<button
  onClick={() => setSelected(r)}
  className={
    selected.id === r.id
      ? "bg-orange-500 text-white"
      : "bg-zinc-100"
  }
```

```
>
  {r.name}
</button>
```

Після вибору ролі відображається відповідне зображення та детальний опис, що включає сильні та слабкі сторони, а також призначення цієї техніки в командному бою.

Компонент Crew.jsx відповідає за опис екіпажу танка. Дані про членів екіпажу зберігаються в масиві та автоматично відображаються у вигляді карток:

```
{CREW.map((c) => (
  <div key={c.name} className="flex gap-4 p-4 border rounded-xl">
    <img src={c.icon} alt={c.name} className="w-20 h-20" />
    <div>
      <p className="font-semibold text-orange-600">{c.name}</p>
      <p className="text-sm">{c.description}</p>
    </div>
  </div>
))}
```

Кожна картка містить інформацію про роль члена екіпажу, бонуси, які він надає, та наслідки його втрати, що дозволяє гравцям краще зрозуміти вплив екіпажу на ефективність техніки.

Сторінка Developers.jsx присвячена розробнику гри. Окрім текстового опису компанії Wargaming, тут використовується хронологія ключових подій, що відображається у вигляді горизонтальної шкали часу. Дані беруться з масиву подій і відображаються динамічно:

```
{items.map((m) => (
  <div key={m.year} className="flex flex-col items-center">
    <div className="font-semibold">{m.title}</div>
    <p className="text-sm">{m.text}</p>
  </div>
))}
```

Компонент Requirements.jsx відображає системні вимоги гри. Дані структуровані у вигляді карток із таблицями для мінімальних та рекомендованих параметрів, що спрощує порівняння та сприйняття інформації користувачем.

Сторінка економіки реалізована як окремий логічний модуль і призначена для пояснення внутрішньоігрових ресурсів, валют та механік прогресу в World of Tanks. Вона побудована за принципом вертикально структурованого контенту з боковою

навігацією, що дозволяє користувачеві швидко переходити між основними тематичними розділами без втрати контексту.

Основний компонент сторінки Economy.jsx відповідає за загальну компоновку, керування активним розділом та підсвічування пунктів навігації. Перелік розділів описано у вигляді масиву конфігурації:

```
const SECTIONS = [
  { id: "currencies", title: "Кредити, золото, бони" },
  { id: "premium", title: "Преміум акаунт" },
];
```

Цей масив використовується одночасно для формування меню в боковій панелі та для зв’язування навігації з відповідними секціями контенту. Активний розділ визначається за допомогою стану:

```
const [activeId, setActiveId] = useState(SECTIONS[0].id);
```

Для автоматичного визначення того, який розділ наразі знаходиться у фокусі користувача під час прокрутки, застосовується IntersectionObserver. Він відстежує появу секцій у зоні видимості та оновлює активний пункт меню:

```
const obs = new IntersectionObserver(
  (entries) => {
    const visible = entries
      .filter((e) => e.isIntersecting)
      .sort((a, b) => b.intersectionRatio - a.intersectionRatio)[0];
    if (visible?.target?.id) setActiveId(visible.target.id);
  },
  { rootMargin: "-20% 0px -60% 0px", threshold: [0.2, 0.4, 0.6, 0.8] }
);
```

Реалізовано зручну «живу» навігацію, яка підсвічує поточний розділ без потреби ручного перемикання.

Для стилізації пунктів меню використовується мемоізована функція, яка змінює вигляд активного та неактивного пунктів залежно від стану:

```
const linkClass = useMemo(
  () => (id) =>
    `block rounded-xl px-3 py-2 transition-colors ${
      activeId === id
        ? "bg-yellow-100 text-yellow-800"
        : "text-zinc-700 hover:bg-zinc-100"
    }`,
  [activeId]
);
```

Основний контент сторінки складається з двох незалежних секцій, які підключаються як окремі компоненти:

```
<main className="space-y-10">
  <CurrenciesSection />
  <PremiumSection />
</main>
```

Компонент CurrenciesSection.jsx призначений для пояснення базових внутрішньоігрових ресурсів. Логіка побудована навколо вкладок, які дозволяють перемикатися між різними типами валют. Активна вкладка зберігається у стані:

```
const [active, setActive] = useState("exp");
```

Кнопки вкладок формуються динамічно з масиву даних, що дозволяє легко додавати нові ресурси без зміни логіки компонента:

```
{TABS.map((tab) => (
  <button
    key={tab.id}
    onClick={() => setActive(tab.id)}
    className={active === tab.id ? "bg-amber-50" : "bg-white"}
  >
    <img src={tab.icon} alt={tab.title} />
  </button>
))}
```

Контент активної вкладки визначається пошуком відповідного елемента в масиві:

```
TABS.find((t) => t.id === active)
```

На основі цього об’єкта відображається назва ресурсу, опис, а також списки способів отримання та використання. Такий підхід забезпечує чітке розділення даних і логіки відображення, що є зручним для навчального контенту.

Компонент PremiumSection.jsx описує можливості преміум-акаунту та містить як статичні інформаційні блоки, так і інтерактивні елементи. Для збереження вибраної тривалості преміуму використовується стан:

```
const [days, setDays] = useState(30);
const [gold, setGold] = useState(PRICES[30]);
```

Ціна преміуму автоматично перераховується при зміні кількості днів:

```
useEffect(() => {
  if (PRICES[days]) {
    setGold(PRICES[days]);
  } else {
    setGold(0);
  }
}, [days]);
```

Орієнтовна вигода у вигляді бонусних кредитів обчислюється окремою функцією:

```
const calcCredits = () => {
  const bonus = (days / 7) * 750000;
  return bonus.toLocaleString("uk-UA");
};
```

Інтерфейс калькулятора дозволяє користувачеві змінювати кількість днів через випадаючий список, після чого миттєво оновлюється вартість у золоті та приблизний бонус.

Окремий блок присвячений конверсії валют і досвіду. Тут у текстовому вигляді пояснюються доступні курси обміну та умови переведення бойового досвіду у вільний, що доповнює загальну картину економічної системи гри. Сторінка економіки побудована за модульним принципом: кожен великий логічний блок винесений в окремий компонент, а вся навігація та взаємодія керуються через стан і спостерігачі. Масиви даних використовуються для формування вкладок, меню та інформаційних блоків, що зменшує дублювання коду та підвищує гнучкість системи.

Сторінка мап реалізована як окремий інформаційний розділ, призначений для огляду ігрових карт World of Tanks з поділом за типами місцевості. Основна мета сторінки надати користувачеві зручний інструмент для перегляду карт, їх описів, візуальних матеріалів та швидкого переходу до статистичних даних.

Усі карти завантажуються з локального масиву даних, який підключається на початку компонента:

```
import { Maps } from "../data/maps.js";
```

Для зручності відображення назви категорій перекладені в окремому об’єкті, який використовується як словник:

```
const typeLabels = {
  summer: "Літні карти",
  winter: "Зимові карти",
  desert: "Піщані карти",
  special: "Особливі карти",
};
```

Це дозволяє зберігати типи карт у даних у компактному вигляді, але відображати їх у зрозумілій для користувача формі.

Стан компонента використовується для збереження вибраної категорії, активної карти та карти, яка перебуває у фокусі прокрутки:

```
const [selectedType, setSelectedType] = useState("");
const [activeSlug, setActiveSlug] = useState("");
```

Перед відображенням карт виконується групування всього масиву за типами. Для цього застосовується метод reduce, який формує об’єкт, де ключем є тип карти, а значенням, масив відповідних мап:

```
const groupedMaps = Maps.reduce((acc, map) => {
  if (!acc[map.type]) acc[map.type] = [];
  acc[map.type].push(map);
  return acc;
}, {});
```

Такий підхід дозволяє легко будувати як фільтрацію, так і секційне відображення контенту без дублювання логіки.

У боковій панелі реалізовано селектор категорій, який змінює стан selectedType. Після вибору категорії користувач бачить список карт лише цього типу:

```
<select
  value={selectedType}
  onChange={(e) => setSelectedType(e.target.value)}
>
  <option value="">Всі категорії</option>
</select>
```

Для зручної навігації список карт у вибраній категорії відображається у вигляді кнопок. Клік по назві карти виконує плавну прокрутку до відповідного блоку в основній частині сторінки:

```
const handleCardClick = (slug) => {
  const el = document.getElementById(slug);
  if (el) {
    el.scrollIntoView({ behavior: "smooth", block: "start" });
```

```
      setActiveSlug(slug);
    }
  };
```

Додатково реалізовано автоматичне визначення активної карти під час прокрутки сторінки. Для цього використовується IntersectionObserver, який відстежує появу блоків карт у зоні видимості та оновлює стан activeSlug:

```
const obs = new IntersectionObserver(
  (entries) => {
    const visible = entries.find((e) => e.isIntersecting);
    if (visible?.target?.id) {
      setActiveSlug(visible.target.id);
    }
  },
  { rootMargin: "-40% 0px -40% 0px", threshold: 0.3 }
);
```

Завдяки цьому відповідна карта підсвічується як у списку, так і в основному контенті, що значно покращує орієнтацію на сторінці з великою кількістю інформації.

Основна частина сторінки будується шляхом проходу по згрупованих мапах. Для кожного типу формується окрема секція з заголовком:

```
{Object.entries(groupedMaps).map(([type, maps]) => (
  <section key={type}>
    <h2>{typeLabels[type]}</h2>
  </section>
))}
```

Кожна карта відображається у вигляді окремої інформаційної картки. У верхній частині показується назва карти, далі основне зображення та мінікарта, що дозволяє гравцю швидко оцінити загальну структуру локації:

```
<img src={map.img} alt={map.name} />
<img src={map.miniMap} alt={`${map.name} minimap`} />
```

Якщо для карти присутній текстовий опис, він виводиться окремим блоком. Це дозволяє поєднати візуальну інформацію з поясненням особливостей місцевості та геймплейних нюансів.

У нижній частині кожної картки розміщено кнопку переходу до зовнішнього ресурсу зі статистикою карти:

```
<a
  href={`https://tomato.gg/maps/EU/${map.slug}`}
```

```
  target="_blank"
  rel="noopener noreferrer"
>
  Показати статистику
</a>
```

Користувач може за потреби перейти до детальної аналітики ефективності карти, не перевантажуючи сам сайт складними статистичними обчисленнями.

Розділ «Новини» є одним із ключових динамічних елементів сайту, оскільки він працює безпосередньо з базою даних Firebase Firestore та підтримує як перегляд інформації звичайними користувачами, так і керування контентом з боку адміністратора. Архітектурно цей функціонал поділений на три основні частини: список новин, детальну сторінку новини та модальне вікно для додавання нових записів.

У компоненті News.jsx реалізовано завантаження новин з колекції news у Firestore. Дані отримуються один раз після монтування компонента за допомогою useEffect:

```
useEffect(() => {
  const fetchNews = async () => {
    const querySnapshot = await getDocs(collection(db, "news"));
    const data = querySnapshot.docs.map((doc) => ({
      id: doc.id,
      ...doc.data(),
    }));
    setNews(data);
  };
  fetchNews();
}, []);
```

Кожен документ бази даних перетворюється у JavaScript-об’єкт і зберігається у стані news. Такий підхід дозволяє надалі виконувати пошук, фільтрацію та сортування без додаткових запитів до бази даних.

Для зручності користувача реалізовано клієнтський пошук, фільтрацію за типом новини та сортування за датою. Усі ці операції виконуються над локальним масивом news:

```
const filteredNews = news
  .filter((n) => {
    const term = searchTerm.toLowerCase();
    const matchesSearch =
      n.title.toLowerCase().includes(term) ||
      (n.text?.toLowerCase() || "").includes(term);
    const matchesType =
```

```
      filterType === "all" ? true : n.type === filterType;
    return matchesSearch && matchesType;
  })
  .sort((a, b) => {
    return sortDate === "desc"
      ? new Date(b.date.split(".").reverse().join("-")) -
          new Date(a.date.split(".").reverse().join("-"))
      : new Date(a.date.split(".").reverse().join("-")) -
          new Date(b.date.split(".").reverse().join("-"));
  });
```

Таким чином користувач може швидко знайти потрібну новину, відфільтрувати лише патчі, івенти чи анонси, а також змінити порядок відображення записів.

Наявність адміністративних прав визначається локально через localStorage. Якщо користувач є адміністратором, для нього відкривається додатковий функціонал:

```
const [isAdmin] = useState(localStorage.getItem("isAdmin") === "true");
```

Для адміністратора відображається кнопка додавання новини, а також кнопки видалення для кожного запису.

Процес видалення новини супроводжується підтвердженням дії через бібліотеку SweetAlert, що зменшує ризик випадкової втрати даних:

```
const handleDelete = async (id, title) => {
  const confirm = await Swal.fire({
    title: "Видалити новину?",
    text: `Точно видалити: "${title}" ?`,
    icon: "warning",
    showCancelButton: true,
  });
  if (confirm.isConfirmed) {
    await deleteDoc(doc(db, "news", id));
    setNews((prev) => prev.filter((n) => n.id !== id));
  }
};
```

Після успішного видалення стан оновлюється без перезавантаження сторінки, що відповідає принципам SPA.

Компонент NewsDetails.jsx відповідає за відображення повного тексту окремої новини. Ідентифікатор запису отримується з URL за допомогою useParams, після чого виконується запит до Firestore:

```
const { id } = useParams();
useEffect(() => {
```

```
  const fetchNews = async () => {
    const docRef = doc(db, "news", id);
    const docSnap = await getDoc(docRef);
    if (docSnap.exists()) {
      setNewsItem({ id: docSnap.id, ...docSnap.data() });
    }
  };
  fetchNews();
}, [id]);
```

Цей підхід дозволяє реалізувати класичну модель «список → деталі» без дублювання даних у пам’яті застосунку.

Компонент AddNewsModal.jsx реалізує форму додавання новин для адміністратора. Усі поля керуються локальним станом:

```
const [title, setTitle] = useState("");
const [type, setType] = useState("Новина");
const [text, setText] = useState("");
const [source, setSource] = useState("");
```

Перед відправленням даних виконується базова валідація, після чого новина зберігається у Firestore:

```
const newsData = {
  title,
  type,
  text,
  source: source || "—",
  date: getCurrentDate(),
};

const docRef = await addDoc(collection(db, "news"), newsData);
```

Після успішного додавання модальне вікно закривається, а нова новина одразу додається до загального списку без повторного запиту до бази даних.

Сторінки «Огляди» та «Новачкам» побудовані на тій самій архітектурній основі, що й сторінка новин: використовується Firebase Firestore як джерело даних, React-стан для керування інтерфейсом та SPA-маршрутизація для переходу між списком і детальним переглядом матеріалів. Тому базові механізми отримання, додавання та видалення записів є аналогічними і не потребують повторного детального опису. Водночас ці розділи мають низку специфічних особливостей, пов’язаних зі структурою контенту та логікою його відображення.

Головною відмінністю оглядів є підтримка двох різних типів матеріалів: відеооглядів і текстових оглядів. Тип огляду визначається на етапі створення матеріалу й зберігається в полі type документа Firestore:

```
const [type, setType] = useState("video");
```

Залежно від вибраного типу, адміністратору відображається різний набір полів форми. Для відеооглядів обов'язковими є посилання та короткий опис, тоді як для текстових оглядів використовується розширена структура з кількох логічних секцій.

При збереженні текстового огляду його вміст форматується в один текстовий блок зі спеціальними маркерами розділів:

```
reviewData.text =
`#INTRO\n${intro}\n\n#BODY\n${body}\n\n#IMPORTANT\n${important}\n\n#CONCLUSION\n${con
clusion}`;
```

Такий підхід дозволяє зберігати складну статтю в одному полі бази даних, не ускладнюючи структуру колекції.

На сторінці детального перегляду огляду ці маркери використовуються для парсингу тексту і подальшого виведення секцій у різному візуальному оформленні:

```
const sections = text.split(/#(INTRO|BODY|IMPORTANT|CONCLUSION)/);
```

У результаті вступ, основна частина, важливі зауваження та висновок відображаються як окремі блоки з різними стилями, що покращує читабельність і сприйняття матеріалу. Таким чином, розділ «Огляди» відрізняється від новин більш складною внутрішньою структурою контенту та підтримкою різних форматів матеріалів.

Сторінка «Новачкам» орієнтована на навчальний контент і поєднує одразу кілька типів матеріалів в межах однієї сторінки. На відміну від новин, тут використовується категоріальна модель, де всі матеріали зберігаються в одній колекції tutorials, але мають різні значення поля category.

```
const [category, setCategory] = useState("Поради та гайди");
```

Залежно від обраної категорії адміністратор заповнює різні поля:

– для «Порад та гайдів»: назву, текст і, за потреби, відео;

– для FAQ: питання та відповідь;

– для словника термінів: термін і його визначення.

Логіка формування об’єкта перед збереженням динамічно змінюється залежно від категорії:

```
if (category === "FAQ") {
  newItem = { ...newItem, question, answer };
}
```

На стороні відображення матеріали групуються за категоріями та виводяться в окремих секціях однієї сторінки. Для довгих текстів у гайдах реалізовано механізм розгортання і згортання вмісту, що дозволяє не перевантажувати інтерфейс:

```
const toggleExpand = (id) => {
  setExpandedIds((prev) =>
    prev.includes(id) ? prev.filter((x) => x !== id) : [...prev, id]
  );
};
```

Для FAQ та словника термінів застосовується інший формат подачі: чітке візуальне розділення питання і відповіді або терміна і пояснення, що підкреслює довідковий характер цих матеріалів.

Сторінки «Огляди» та «Новачкам» використовують ті самі базові механізми роботи з Firestore, що й сторінка новин, але відрізняються структурою даних і логікою відображення контенту. В оглядах ключовою особливістю є підтримка різних форматів матеріалів і парсинг структурованого тексту, тоді як розділ «Новачкам» реалізує багатокатегорійну навчальну систему з адаптивними формами введення та різними сценаріями візуалізації. Це дозволяє використовувати спільну архітектурну основу, водночас гнучко підлаштовуючи інтерфейс під конкретний тип інформації.

Сторінка «Техніка» є одним із найбільш функціонально насичених розділів інформаційного вебсайту та реалізує інтерактивну енциклопедію бойових машин гри World of Tanks. Її основне призначення полягає у наданні користувачеві можливості швидкого пошуку, фільтрації, сортування та детального аналізу ігрової техніки за різними характеристиками. Дані про техніку зберігаються у вигляді статичних JavaScript-масивів, структурованих за націями. Для кожної держави

використовується окремий модуль, що містить перелік танків з їхніми характеристиками (назва, тип, ранг, технічні параметри, іконки, зображення тощо). На рівні сторінки всі ці масиви об’єднуються в єдину колекцію:

```
const allTanks = [
  ...China, ...USSR, ...USA, ...France, ...Germany,
  ...Italy, ...Japan, ...Poland, ...Sweden, ...UK, ...Czechoslovakia,
];
```

Такий підхід дозволяє зберегти логічну структурованість даних і водночас забезпечити зручну подальшу обробку в межах одного компонента.

Для зберігання параметрів взаємодії користувача використовується механізм React state. Окремі змінні стану відповідають за обрані нації, типи техніки, рівень (ранг), преміум-статус, пошуковий запит і параметри сортування. З метою покращення користувацького досвіду всі вибрані фільтри автоматично зберігаються у localStorage браузера:

```
useEffect(() => {
  localStorage.setItem("tankFilters", JSON.stringify({ ... }));
}, [selectedNations, selectedTypes, selectedTier, premiumOnly, searchQuery]);
```

Завдяки цьому при повторному відкритті сторінки користувач отримує вже налаштований інтерфейс із попередньо застосованими фільтрами, що є важливим елементом сучасних SPA-застосунків.

Основна логіка фільтрації реалізована з використанням хука useMemo, що дозволяє оптимізувати обчислення та уникнути зайвих перерендерів. Кожен танк перевіряється на відповідність обраним критеріям: нація, тип, ранг, преміум-статус, а також текстовий пошук. Пошук здійснюється не лише за назвою машини, але й за числовими характеристиками (ХП, урон, швидкість), що значно розширює можливості аналізу.

```
const filteredTanks = useMemo(() => {
  return allTanks.filter((t) => { ... });
}, [selectedNations, selectedTypes, selectedTier, premiumOnly, searchQuery]);
```

Після фільтрації застосовується механізм сортування, який дозволяє впорядковувати техніку за будь-якою з ключових характеристик. Стан сортування зберігає назву поля та напрям (зростання або спадання). Для числових значень

виконується попереднє приведення типів, що гарантує коректність порівнянь. Інтерфейс таблиці візуально відображає активний параметр сортування за допомогою стрілок та зміни стилю заголовка стовпця.

Окремий компонент FiltersSidebar реалізує бокову панель фільтрації, яка динамічно генерується на основі доступних даних. Панель містить фільтри за націями (з відображенням прапорів), типами техніки (іконки класів), рангами та преміум-статусом. Додатково в заголовку панелі відображається кількість знайдених машин, що надає користувачеві миттєвий зворотний зв’язок щодо результатів фільтрації.

Основна частина сторінки реалізована у вигляді таблиці, де кожен рядок відповідає окремому танку. Рядок містить візуальні елементи (прапор, іконку типу, значок преміуму) та ключові характеристики. Клік по рядку таблиці ініціює навігацію до сторінки детального перегляду техніки з використанням React Router:

```
onClick={() => navigate(`/vehicle/${encodeURIComponent(tank.name)}`)}
```

Компонент VehicleDetails відповідає за відображення повної інформації про конкретну бойову машину. Дані вибираються з загального масиву за назвою, переданою через параметри маршруту. Сторінка містить велике зображення техніки, інформацію про націю, тип і ранг, а також розширений набір технічних характеристик, поданих у вигляді адаптивної сітки з іконками.

Окремою функціональною особливістю є можливість підключення 3D-моделі танка через вбудований iframe із зовнішнього ресурсу tanks.gg. Формування URL відбувається динамічно на основі назви техніки, що дозволяє автоматизувати інтеграцію без ручного налаштування для кожної одиниці.

Сторінка «Техніка» поєднує складну логіку обробки даних із високим рівнем інтерактивності інтерфейсу. Використання React hooks, мемоізації, клієнтської фільтрації та сортування, а також поділ функціональності на логічні компоненти забезпечує масштабованість, зручність підтримки та відповідність сучасним вимогам до вебзастосунків інформаційного типу.

Для керування вмістом інформаційного вебсайту реалізовано механізм авторизації адміністратора, який обмежує доступ до адміністративних функцій лише уповноваженим користувачам. Вхід до адміністративної частини здійснюється через окрему сторінку авторизації, де перевіряються облікові дані користувача.

Перевірка логіна та пароля виконується на клієнтському рівні. У разі успішної авторизації в локальному сховищі браузера зберігається спеціальний прапорець, що позначає користувача як адміністратора:

```
if (email === "wotadmin@gmail.com" && password === "wotadming") {
  localStorage.setItem("isAdmin", "true");
  navigate("/admin");
}
```

Використання localStorage дозволяє зберігати інформацію про статус адміністратора між переходами сторінками без повторної авторизації в межах одного сеансу роботи з сайтом.

Для захисту адміністративних сторінок реалізовано механізм перевірки прав доступу. Під час відкриття адміністративної панелі система перевіряє наявність відповідного прапорця, і в разі його відсутності виконується автоматичне перенаправлення на сторінку входу:

```
useEffect(() => {
  const isAdmin = localStorage.getItem("isAdmin") === "true";
  if (!isAdmin) {
    navigate("/login");
  }
}, []);
```

Такий підхід забезпечує базовий контроль доступу та унеможливлює відкриття адміністративних розділів неавторизованими користувачами навіть при прямому переході за URL-адресою.

Після успішної авторизації адміністратор отримує доступ до адміністративної панелі, яка виступає центральним елементом керування контентом сайту. В межах своїх повноважень адміністратор може:

- керувати інформаційним наповненням розділу новин;
- модерувати та оновлювати огляди гри;
- редагувати та актуалізувати матеріали розділу «Новачкам»;

– виконувати видалення або приховування застарілого контенту.

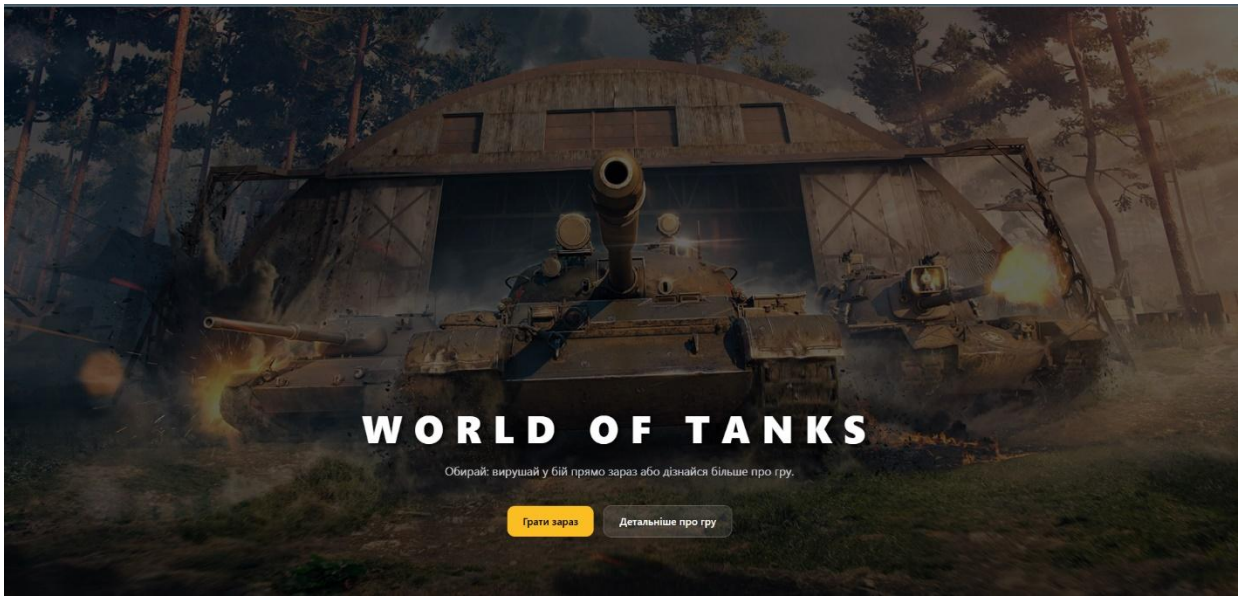
Інтерфейс адміністративної панелі інформує користувача про доступні можливості та роль адміністратора, що підтверджує його розширені права доступу:

```
<p>
  Тут ви можете керувати контентом сайту: додавати та редагувати
  огляди, новини і розділ «Новачкам».
</p>
```

Реалізований механізм авторизації адміністратора забезпечує логічне розмежування ролей користувачів, підвищує безпеку вебсайту та дозволяє централізовано керувати інформаційним наповненням без втручання у клієнтську частину основного інтерфейсу.

4.3. Тестування та демонстрація роботи сайту

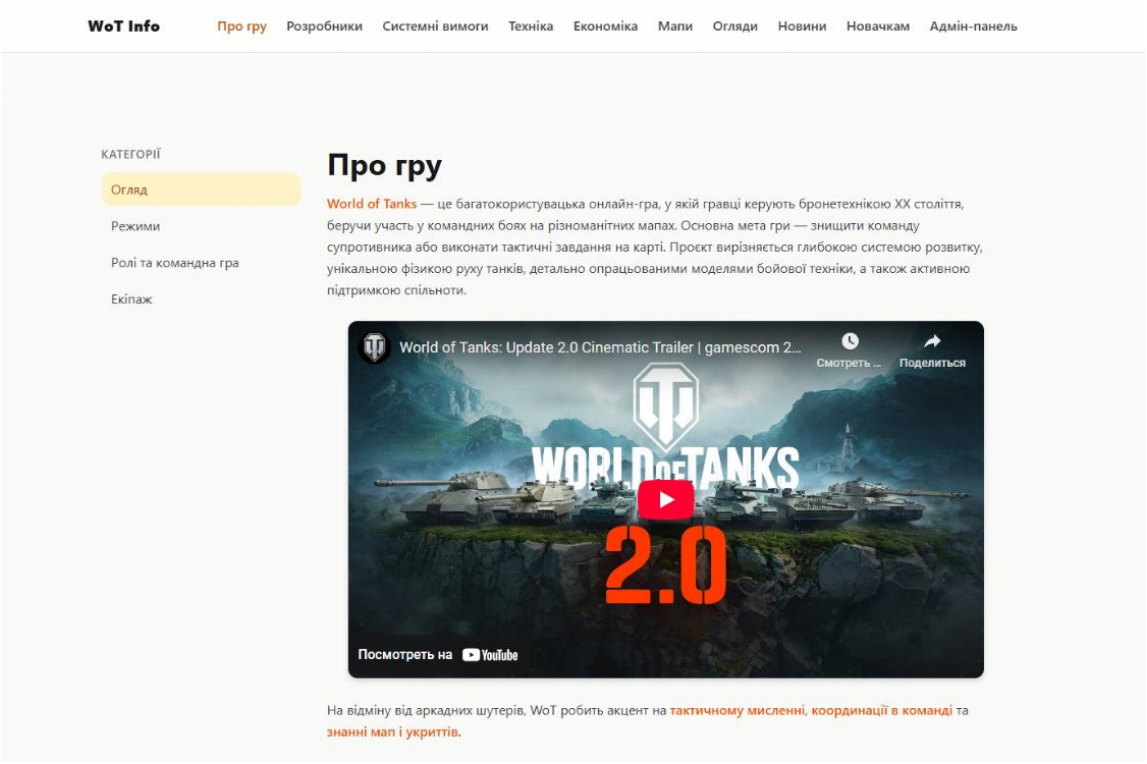
Демонстрація роботи вебсайту здійснювалася на локальному сервері з використанням збережених компонентів React. Для наочності результатів наведено серію скріншотів, які показують основні сторінки та функціонал системи. Кожен етап супроводжується коротким описом, що дозволяє оцінити коректність відображення інтерфейсу та перевірити відповідність поставленим вимогам. Першою демонструється головна сторінка сайту (рис. 4.1). Вона містить фонове зображення з бойовою технікою, центральний заголовок World of Tanks та дві кнопки: «Грати зараз» і «Детальніше про гру».



Мал. 4.1 – Перша сторінка сайту

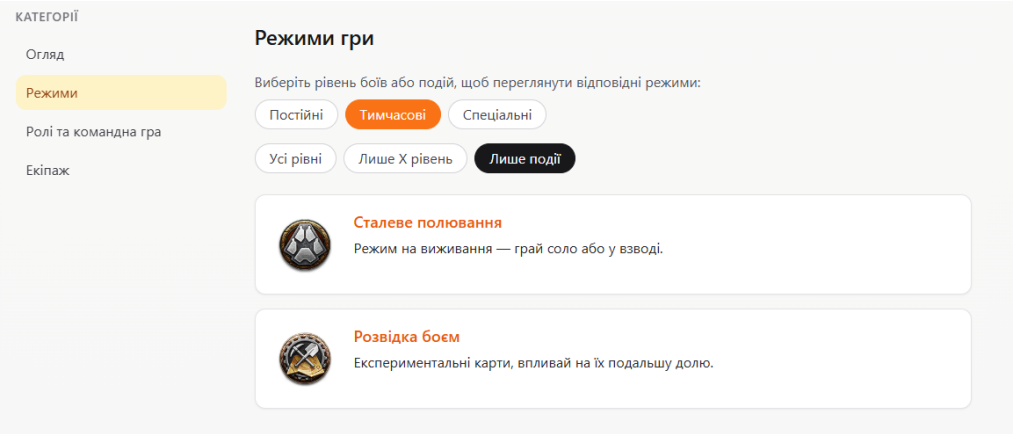
Візуальне оформлення відповідає сучасним вимогам: використано чіткий контраст між текстом та зображенням, а також яскраві акценти для кнопок. Усі елементи розташовані логічно, що спрощує навігацію для користувача.

На другій сторінці (мал. 4.2) відображається розділ «Про гру». Тут користувач може ознайомитися з базовою інформацією про проект.



Мал. 4.2 – Розділ про гру

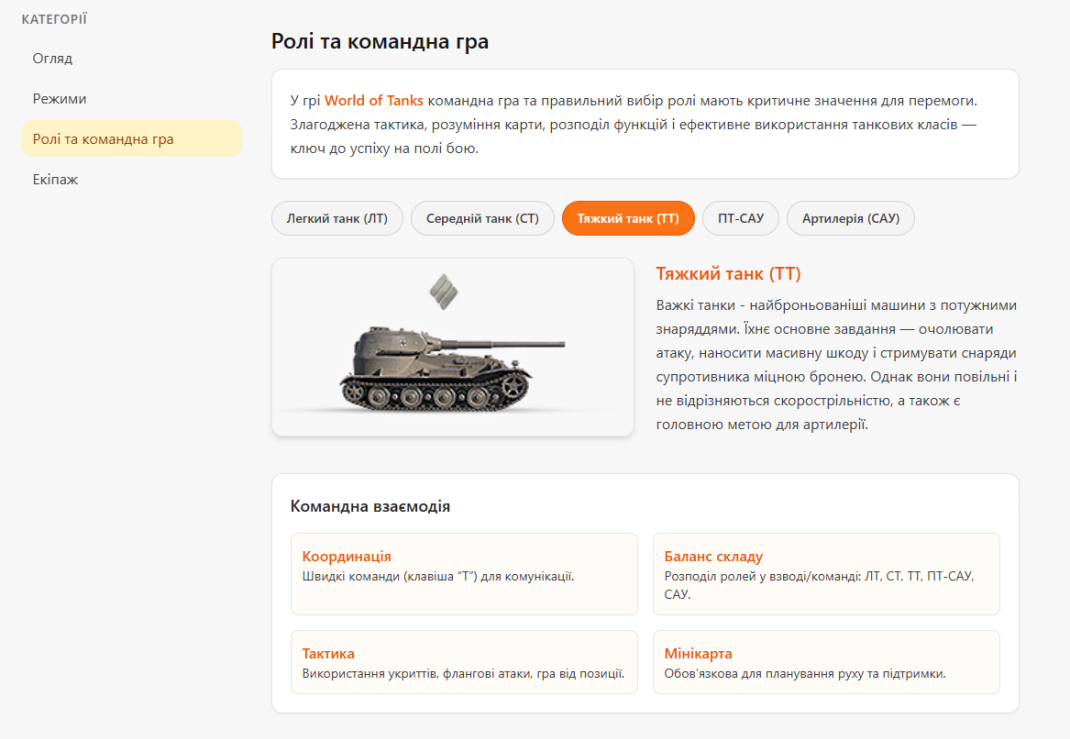
Текстові блоки подані структуровано, ключові слова підсвічені кольором. У центрі інтегровано трейлер з YouTube, який дозволяє одразу отримати уявлення про атмосферу гри. Дизайн стриманий і витриманий у єдиному стилі з головною сторінкою. Далі наведено категорію «Режими гри» (рис. 4.3).



Мал. 4.3 – Категорія режими гри

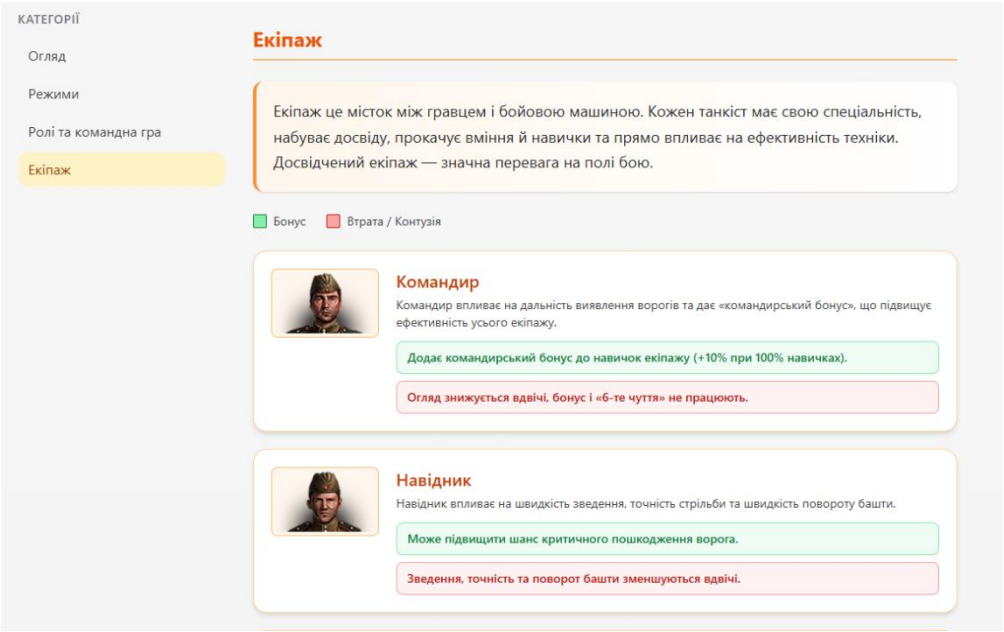
Вона містить набір перемикачів для вибору постійних, тимчасових чи спеціальних форматів боїв. Також доступний фільтр за рівнями. Нижче відображаються картки з іконкою, назвою та описом режиму. Тестування підтвердило, що перемикання вкладок і фільтрів працює коректно, відображаючи лише релевантні дані.

В категорії «Ролі та командна гра» (рис. 4.4) реалізовано вибір типу техніки: легкі, середні, важкі танки, ПТ-САУ та артилерія. Після вибору відображається зображення конкретного класу та текстовий опис його особливостей. Нижче подано блок «Командна взаємодія», де узагальнено ключові аспекти командної гри: координація, тактика, баланс складу. Усі елементи розташовані логічно, взаємодія з кнопками працює стабільно.



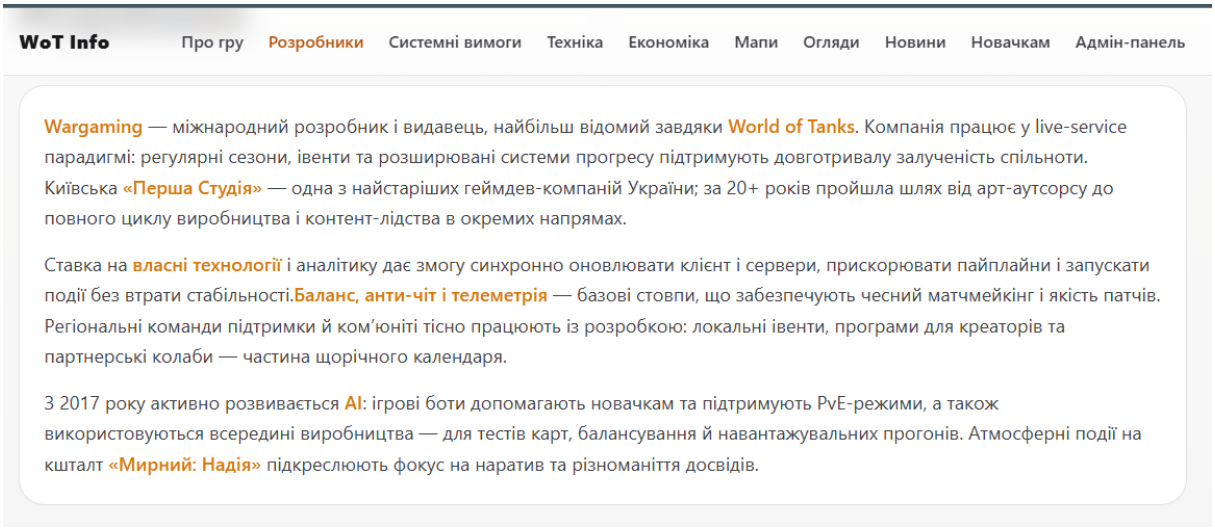
Мал. 4.4 – Категорія ролі та команда гра

Останній екран в цьому розділі демонструє категорію «Екіпаж» (рис. 4.5). Тут відображено картки членів екіпажу з фотографіями, описом, бонусами та наслідками їхньої втрати.



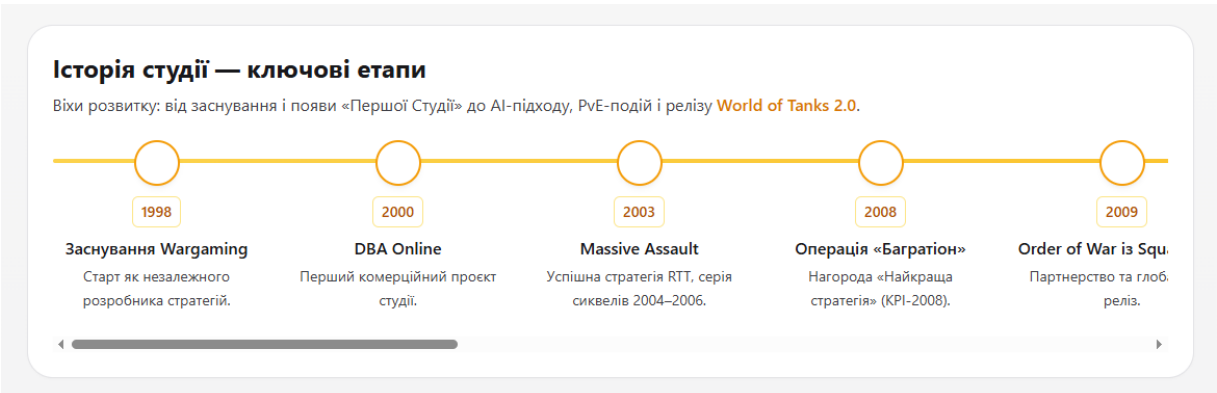
Мал. 4.5 – Категорія екіпаж

Використано кольорове кодування: зелений для бонусів та червоний для втрат. Такий підхід значно спрощує сприйняття інформації. Тестування підтвердило, що картки відображаються коректно, а візуальне розділення інформації відповідає вимогам зручності інтерфейсу. Наступна сторінка «Розробники», у верхній частині якої розташований блок із текстовою інформацією про компанію Wargaming та її діяльність (рис. 4.6).



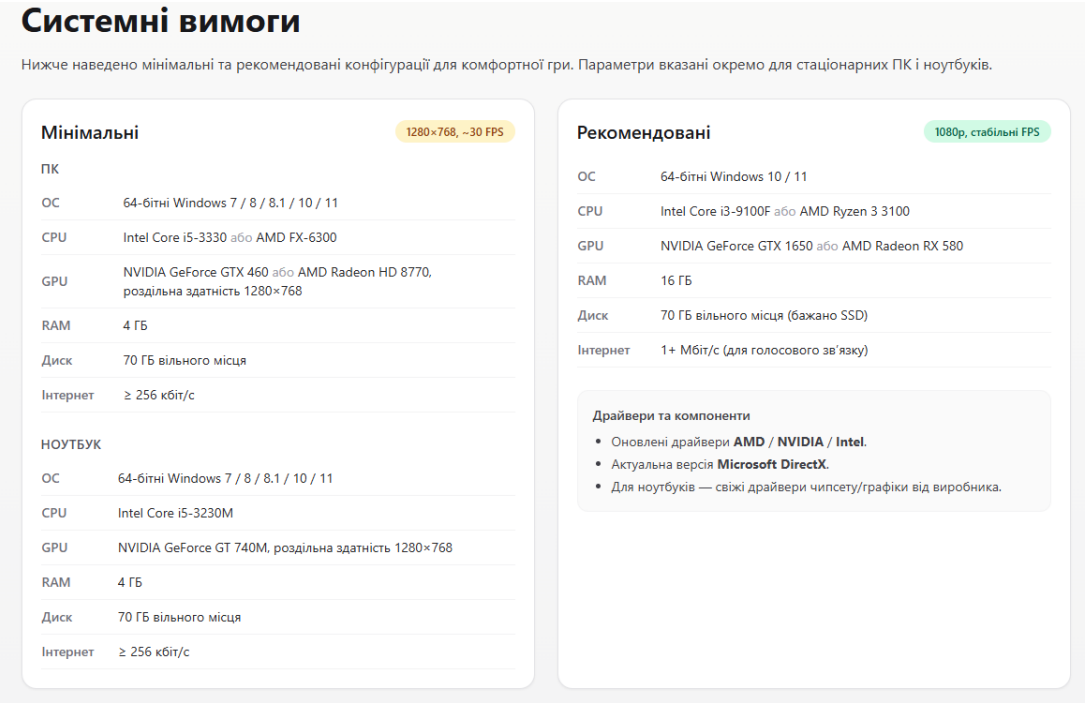
Мал. 4.6 – Сторінка розробники та інформація про Wargaming

У нижній частині сторінки розміщено інтерактивний таймлайн (рис. 4.7), який візуалізує основні віхи розвитку компанії: від заснування у 1998 році до сучасних етапів на кшталт релізу World of Tanks 2.0. Елементи таймлайну відображені у вигляді горизонтальної шкали з подіями, що дає можливість швидко оцінити історію компанії.



Мал. 4.7 – Сторінка розробники, таймлайн Wargaming

На сторінці «Системні вимоги» (рис. 4.8) наведено мінімальні та рекомендовані параметри для ПК і ноутбуків.

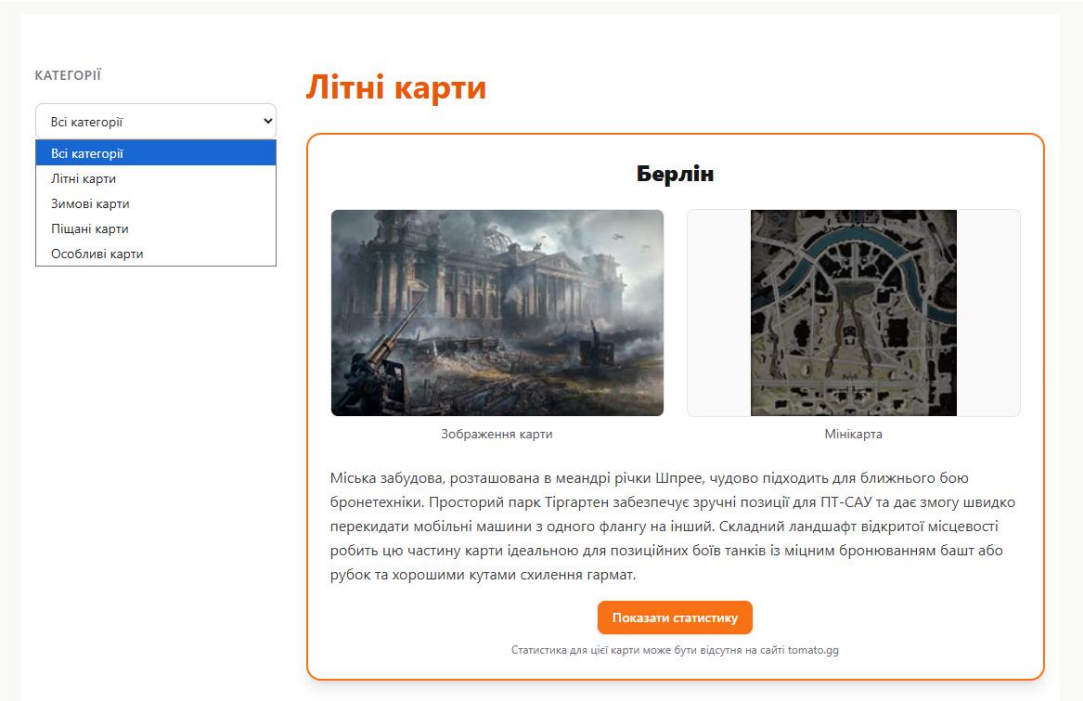


Мал. 4.8 – Сторінка системних вимог

Дані подані у вигляді таблиць із чітким поділом, що дозволяє швидко оцінити вимоги до обладнання. Блок із драйверами та компонентами розміщений у нижній частині, що підкреслює його важливість для стабільної роботи гри. Дизайн сторінки побудований на контрасті світлого фону та кольорових акцентів, що виділяють ключові параметри.

У цілому сторінка виглядає сучасно, є інформативним і дає змогу швидко порівняти технічні можливості системи з вимогами гри.

На сторінці «Мапи» (рис. 4.9) реалізовано зручний інтерфейс для перегляду і вивчення ігрових локацій World of Tanks. У лівій частині сторінки розміщено панель категорій, яка дозволяє обрати тип мап: усі категорії, літні, зимові, піщані та особливі. Такий підхід спрощує навігацію та дає змогу швидко відфільтрувати потрібний набір локацій залежно від умов бою та стилю гри користувача.



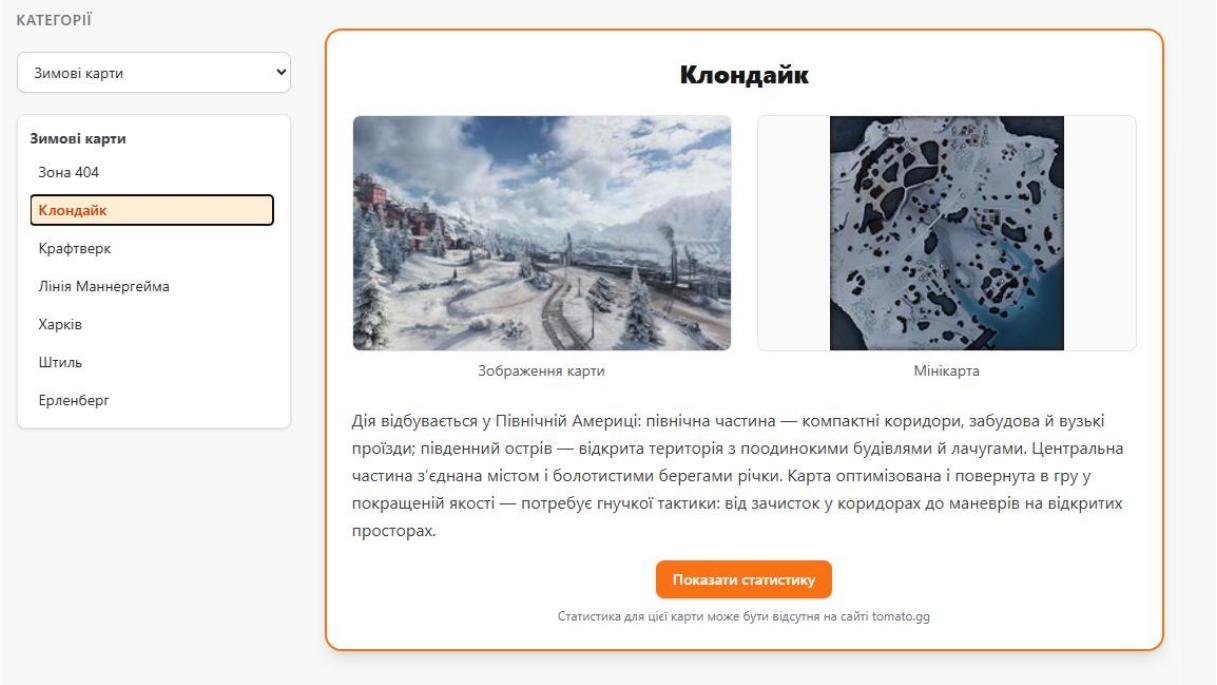
Мал. 4.9 – Сторінка «Мапи»

Центральна частина сторінки відведена під відображення детальної інформації про вибрану мапу. У верхній частині картки виводиться назва локації, нижче два візуальні блоки: основне зображення мапи та її мінімапа. Основне зображення дає загальне уявлення про атмосферу, рельєф і тип забудови, тоді як мінікарта дозволяє оцінити планування, ключові напрямки руху та стратегічні точки.

Під зображеннями подано текстовий опис мапи, у якому стисло охарактеризовано особливості місцевості, рекомендовані тактичні підходи та переваги для різних типів техніки.

Загалом сторінка «Мапи» виконує роль довідкового та аналітичного розділу, поєднуючи візуальну та текстову інформацію. Тестування підтвердило, що всі елементи сторінки відображаються коректно, а структура інтерфейсу є зрозумілою навіть для нових користувачів.

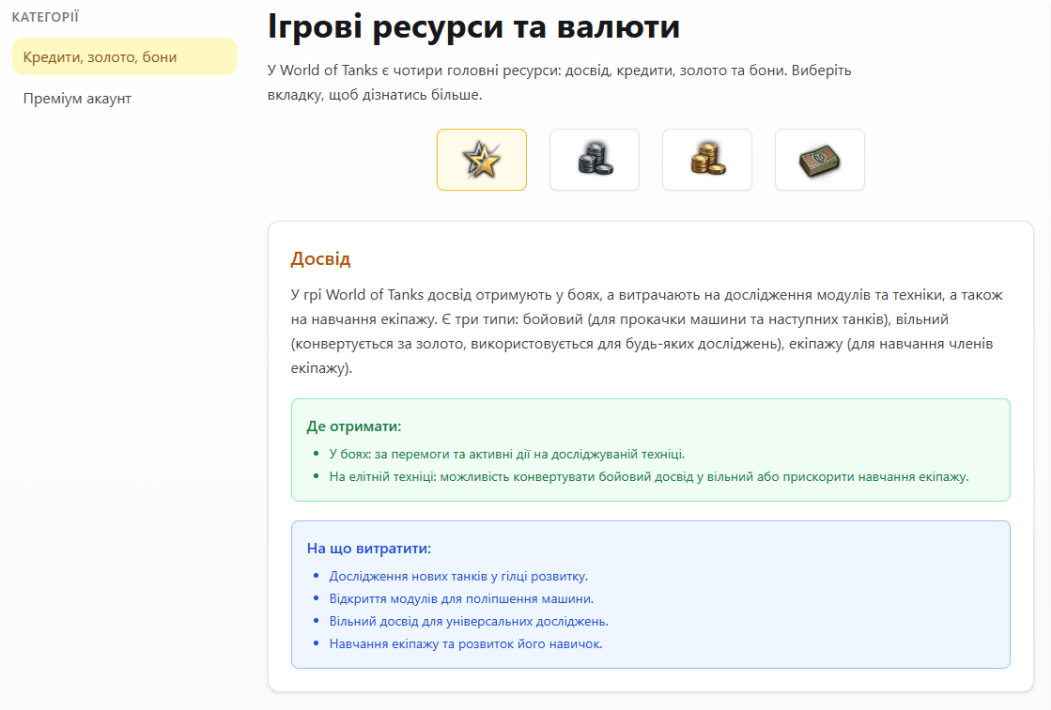
На рисунку 4.10 продемонстровано роботу активного фільтра мап. Після вибору конкретної категорії, наприклад «Зимові карти», інтерфейс автоматично оновлюється без перезавантаження сторінки. У лівій панелі відображається список лише тих мап, які належать до обраної категорії, а в основному блоці детальна картка поточно активної локації.



Мал. 4.10 – Сторінка «Мапи», робота фільтрів

Активний елемент у списку підсвічується кольором, що дозволяє користувачу чітко розуміти, яка мапа наразі вибрана. При натисканні на іншу назву зі списку відбувається плавна прокрутка до відповідного блоку з описом, а візуальне виділення змінюється автоматично. Така поведінка реалізована з використанням механізмів відстеження видимості елементів і значно підвищує зручність роботи зі сторінкою.

На сторінці «Економіка» (рис. 4.11) реалізовано інформативний розділ, присвячений основним ігровим ресурсам та валютам у World of Tanks. У лівій частині інтерфейсу розміщено вертикальне меню навігації з категоріями, що дозволяє швидко перемикатися між підрозділами. Активна категорія «Кредити, золото, бони» підсвічується кольором, що забезпечує зрозумілу візуальну ієрархію.

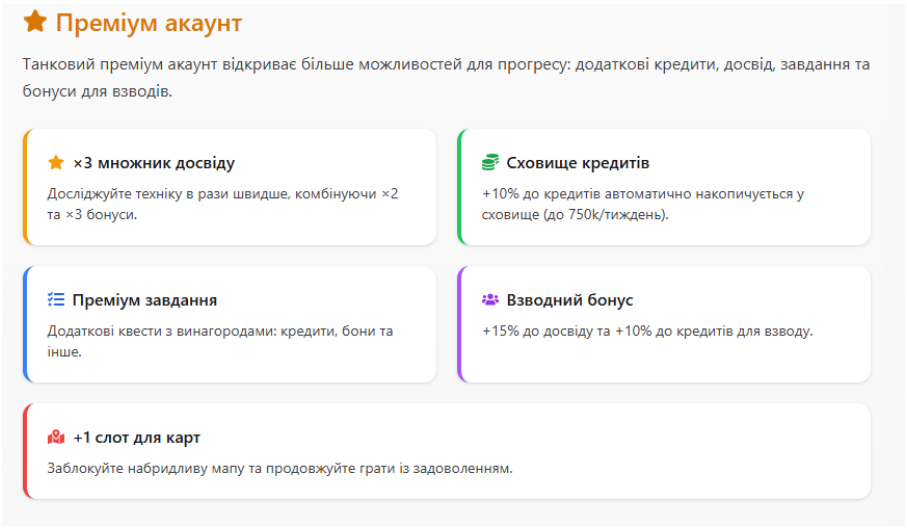


Мал. 4.11 – Сторінка «Економіка»

Центральна частина сторінки містить заголовок «Ігрові ресурси та валюти» та короткий опис, який пояснює роль досвіду, кредитів, золота і бонів у прогресі гравця. Нижче розташовано інтерактивні вкладки з іконками валют, що дозволяють перемикатися між окремими типами ресурсів без перезавантаження сторінки.

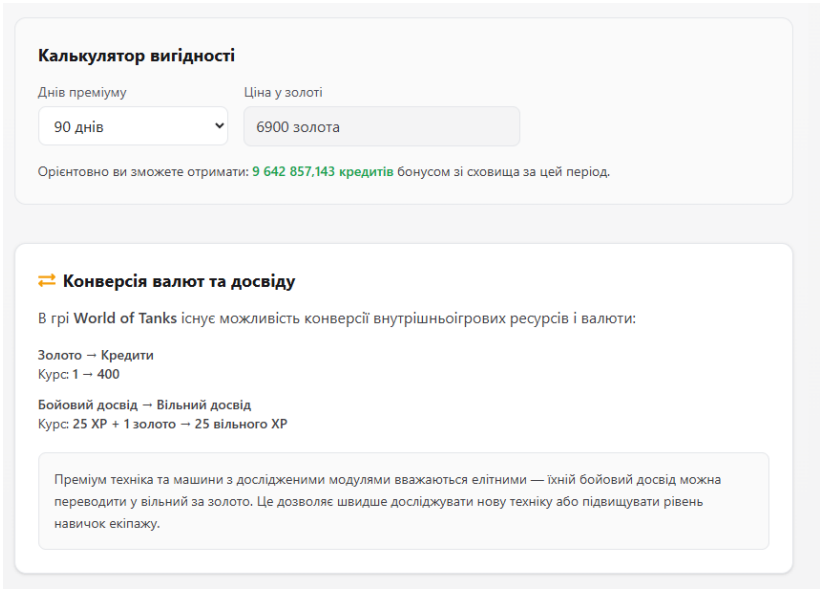
Для кожного ресурсу відображається структурований опис, у якому пояснюється його призначення та особливості використання.

Наступний підрозділ сторінки «Економіка» присвячений преміум акаунту (мал. 4.12). У верхній частині розміщено заголовок із піктограмою зірки та короткий пояснювальний текст, який узагальнює основні переваги танкового преміум акаунту.



Мал. 4.12 – Сторінка «Економіка», розділ «Преміум акаунт»

Основний контент подано у вигляді окремих інформаційних карток, кожна з яких описує конкретний бонус. В нижній частині сторінки реалізовано блок «Калькулятор вигідності» (рис. 4.13), який дозволяє наочно оцінити економічну доцільність використання преміум акаунту.



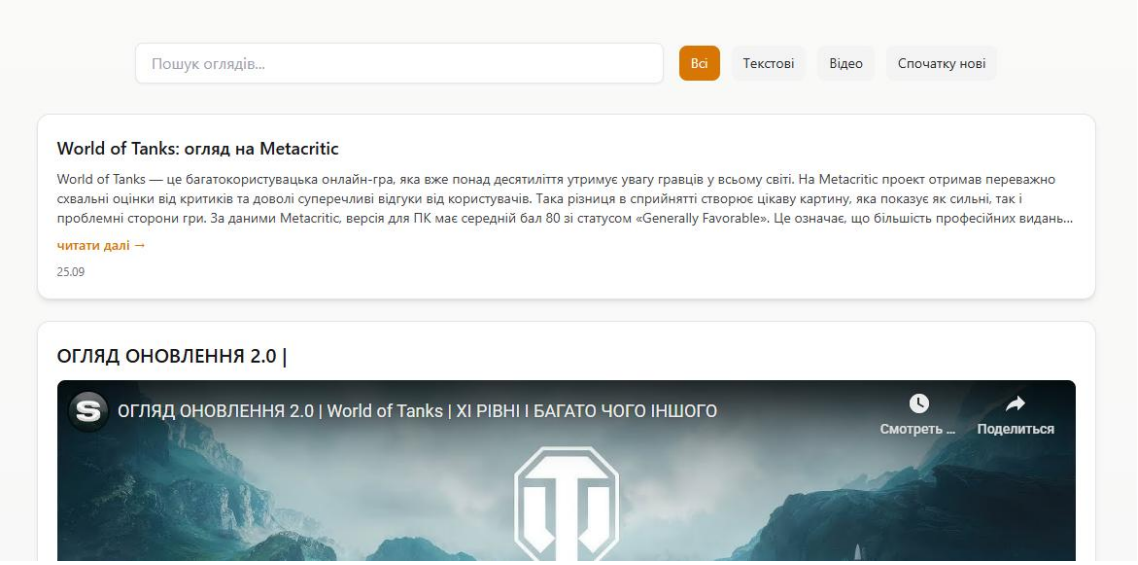
Мал. 4.13 – Сторінка «Економіка», розділ «Преміум акаунт»

Користувач може обрати кількість днів преміуму зі спадного списку, після чого система автоматично відображає вартість у золоті та орієнтовну кількість кредитів, які можна отримати бонусом зі сховища за вибраний період. Результати розрахунку виводяться у текстовому форматі з акцентом на ключових числових

значеннях, що забезпечує зрозумілість і прозорість обчислень. Тестування підтвердило, що зміна параметрів миттєво оновлює результат без затримок.

Окремим інформаційним блоком подано розділ, присвячений конверсії внутрішньоігрових ресурсів і досвіду. Тут пояснюються доступні курси обміну, зокрема перетворення золота у кредити та бойового досвіду у вільний досвід.

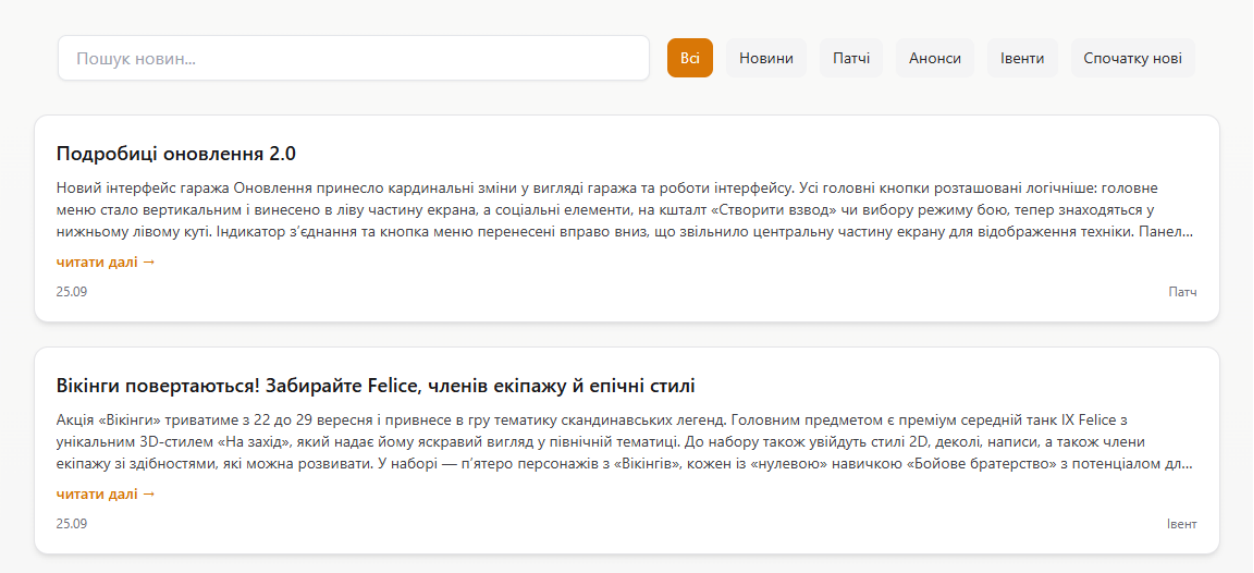
На малюнку 4.14 зображено сторінку «Огляди». У верхній частині розміщено поле пошуку та кнопки фільтрації за типом контенту (усі, текстові, відео), а також перемикач сортування за датою. Це дозволяє користувачеві швидко знайти потрібний матеріал.



Мал. 4.14 – Сторінка «Огляди»

Нижче відображаються картки оглядів. Текстові огляди містять заголовок, короткий опис і посилання «читати далі», тоді як відеоогляди одразу інтегрують плеєр YouTube. Такий підхід забезпечує єдиний формат подачі різних типів контенту, зберігаючи при цьому їхню специфіку.

На малюнку 4.15 показано сторінку «Новини», яка за структурою схожа на сторінку оглядів, але має додаткову класифікацію за типами: новини, патчі, анонси та івенти. У верхній частині також розміщено поле пошуку та фільтри, що дозволяють швидко відсортувати стрічку.

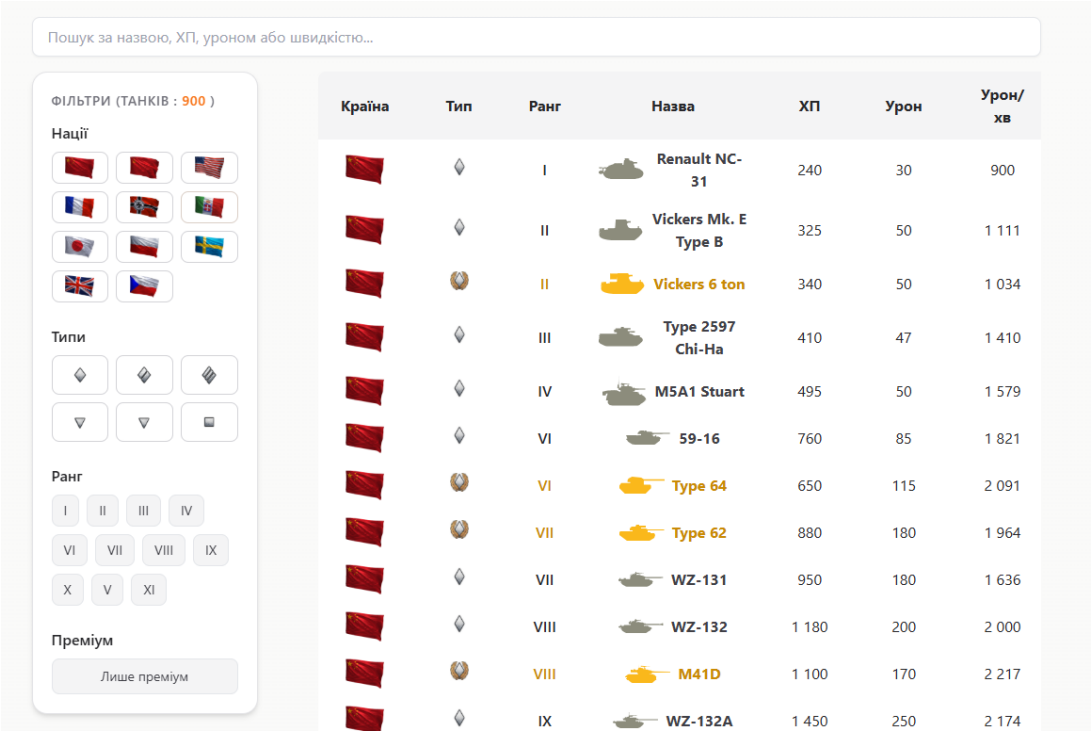


Мал. 4.15 – Сторінка «Новини»

Кожна новина представлена у вигляді окремої картки з заголовком, фрагментом тексту, датою публікації та типом матеріалу. Посилання «читати далі» веде на окрему сторінку з повним текстом новини. Тестування підтвердило коректну роботу фільтрів, пошуку та переходів між сторінками.

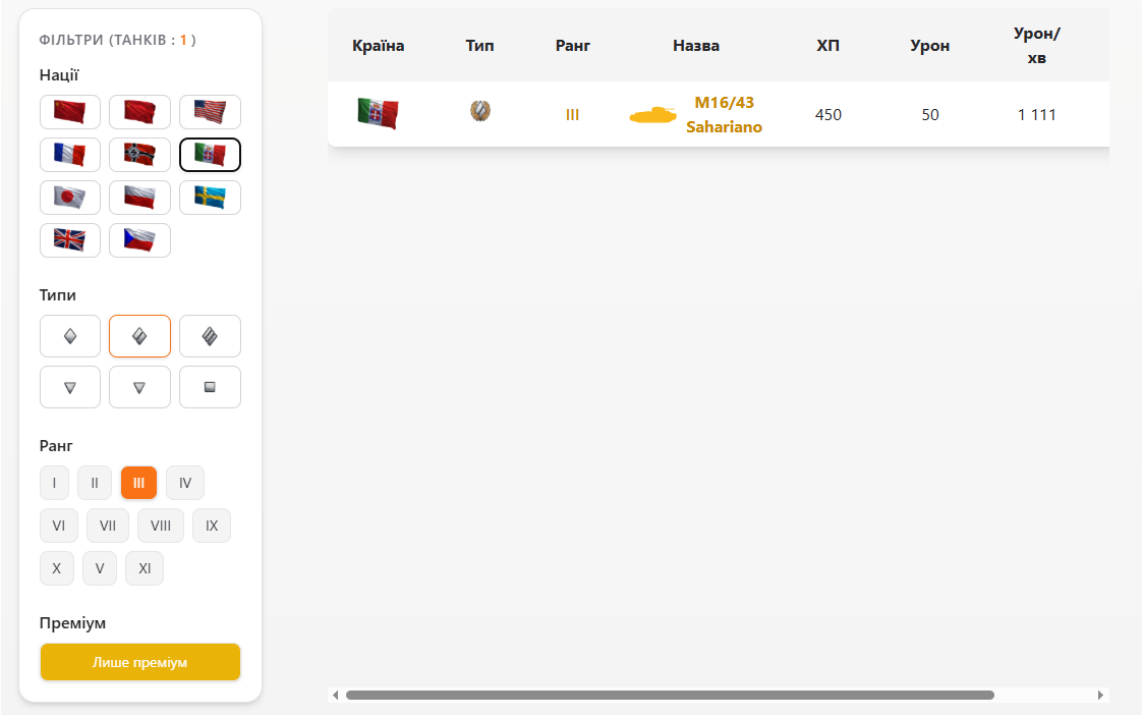
На сторінці «Техніка» (мал. 4.16) реалізовано повноцінний каталог бойових машин World of Tanks у вигляді таблиці з розширеними можливостями фільтрації та пошуку. У верхній частині сторінки розміщене універсальне поле пошуку, яке дозволяє знаходити техніку за назвою, кількістю очок міцності, середнім уроном або швидкістю. Основна частина інтерфейсу поділена на дві зони: бокову панель фільтрів та таблицю з результатами.

Таблиця містить ключові характеристики кожної машини, зокрема країну, тип, ранг, назву, ХП, урон та урон за хвилину. Такий формат подання забезпечує швидке порівняння техніки між собою та зручний аналіз ігрових параметрів.



Мал. 4.16 – Сторінка «Техніка»

На мал. 4.17 продемонстровано роботу активних фільтрів.

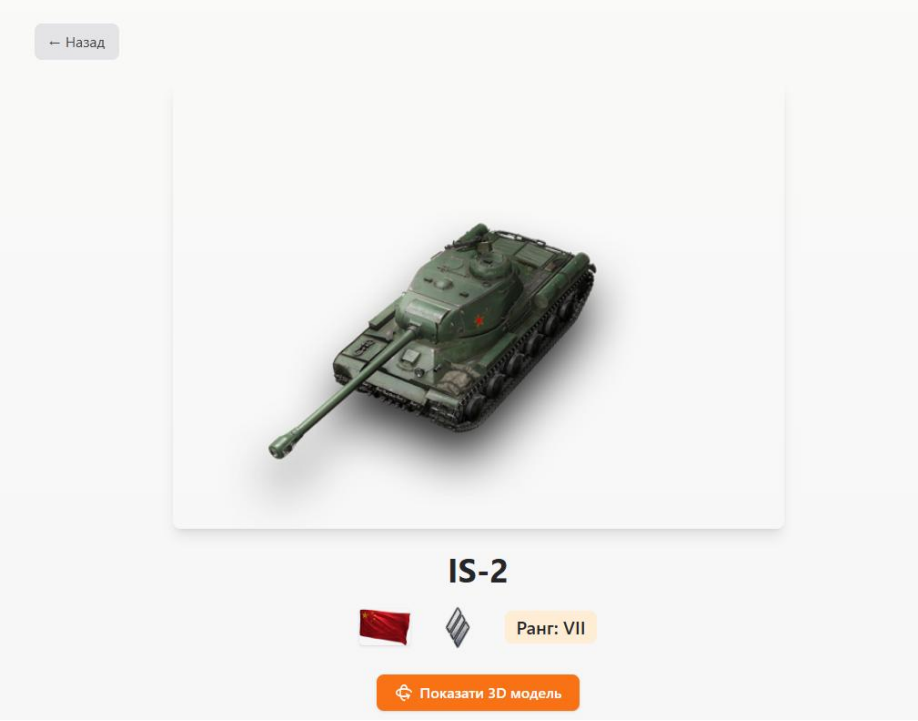


Мал. 4.17 – Сторінка «Техніка», активні фільтри

Користувач може комбінувати декілька критеріїв одночасно: вибір нації, типу техніки, рівня (рангу) та преміум-статусу. Активні елементи фільтрації візуально підсвічуються, а кількість знайдених машин автоматично оновлюється.

Тестування показало, що фільтри працюють коректно навіть при складних комбінаціях умов, забезпечуючи миттєве оновлення таблиці без перезавантаження сторінки. Це підвищує зручність навігації та дозволяє швидко знаходити потрібну техніку.

При натисканні на окрему одиницю техніки користувач переходить на сторінку з детальною інформацією про вибраний танк (мал. 4.18).

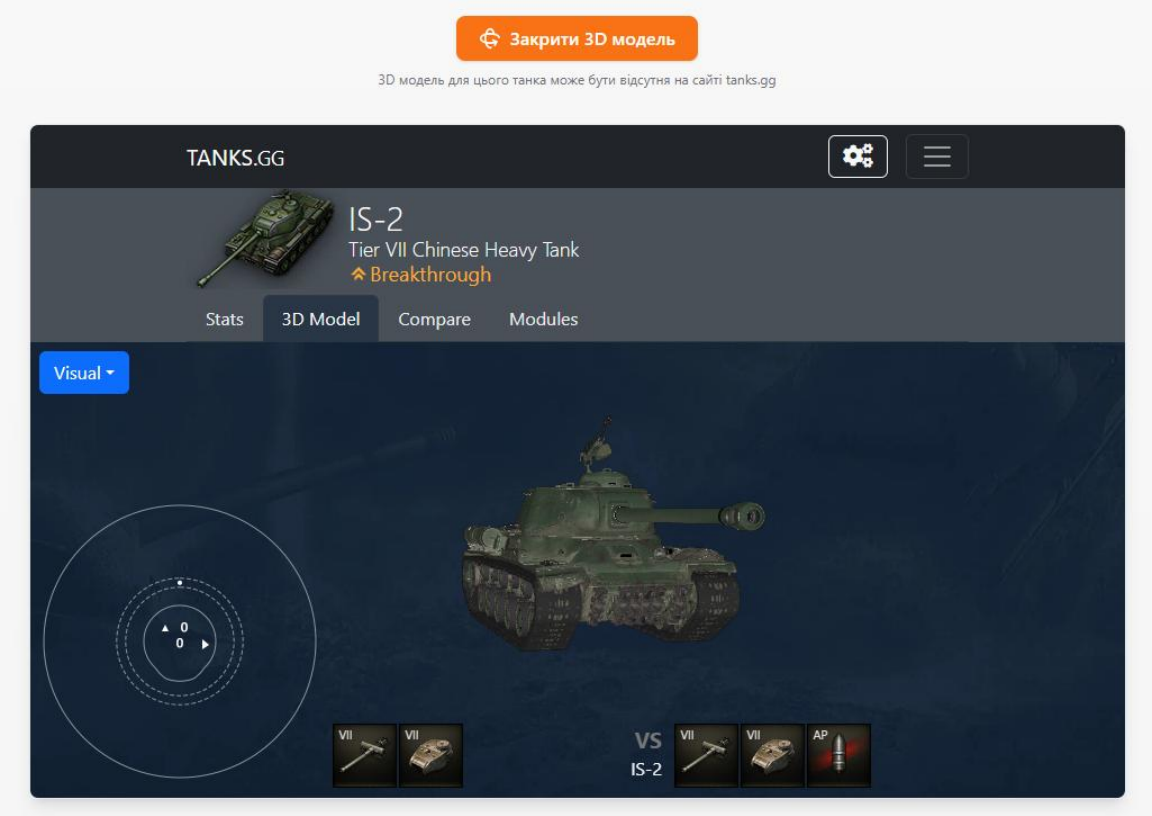


Мал. 4.18 – Сторінка детальної інформації про техніку

У верхній частині відображається велике зображення машини, її назва, нація, тип та ранг. Таке представлення дозволяє одразу ідентифікувати техніку та її місце в ігровій ієрархії.

Інтерфейс сторінки витриманий у єдиному стилі з каталогом, що забезпечує візуальну цілісність і зрозумілість для користувача.

Нижче на сторінці доступна кнопка для відкриття інтерактивної 3D-моделі танка (мал. 4.19).



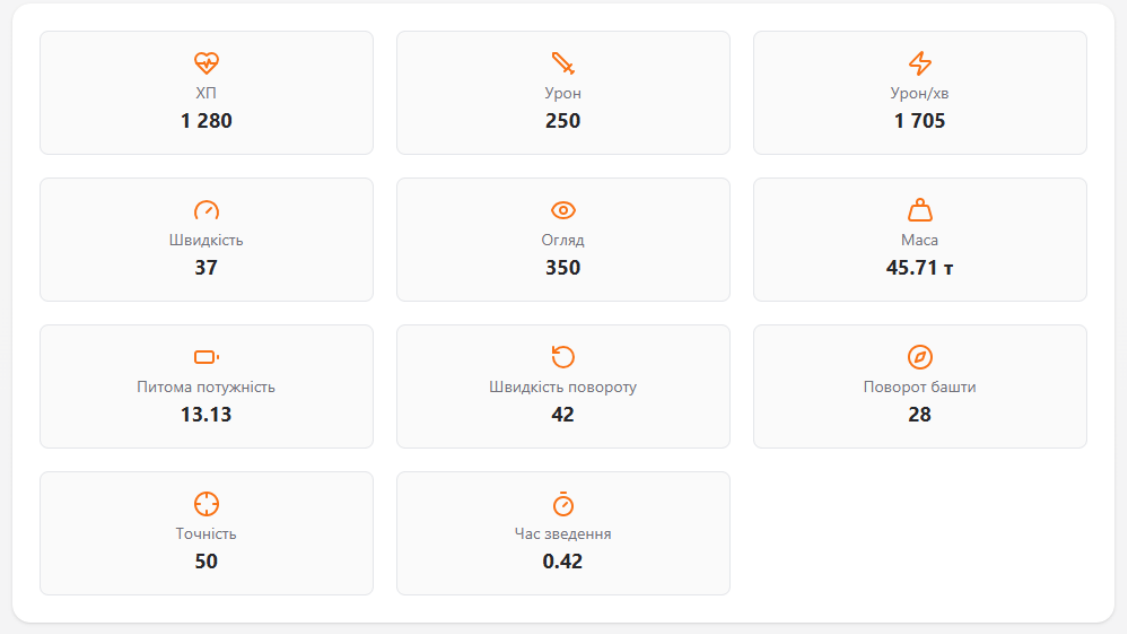
Мал. 4.19 – 3D-модель танку

Модель завантажується з зовнішнього ресурсу tanks.gg за умови її наявності. Користувач може обертати модель, змінювати ракурс огляду та детально розглянути конструкцію машини.

У разі відсутності 3D-моделі система коректно інформує користувача відповідним повідомленням, що свідчить про продуману обробку виняткових ситуацій.

У нижній частині сторінки розміщена панель з основними характеристиками техніки (мал. 4.20). Параметри подані у вигляді окремих інформаційних блоків з іконками, що відображають очки міцності, урон, урон за хвилину, швидкість, огляд, масу, питому потужність, швидкість повороту, точність та час зведення. Використання карткового формату та піктограм значно полегшує сприйняття

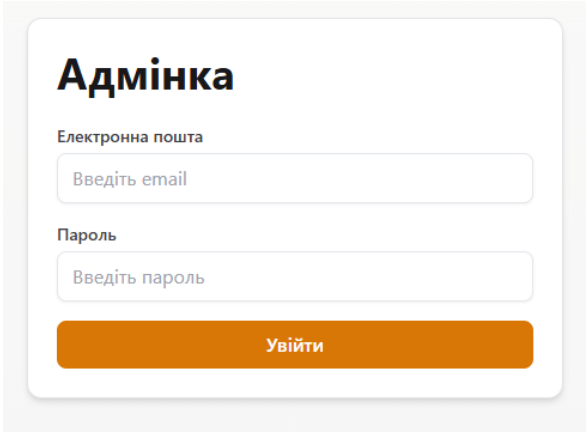
числових даних і дозволяє швидко оцінити сильні та слабкі сторони конкретної машини.



Мал. 4.20 – Характеристики техніки

Загалом сторінка «Техніка» реалізована як інформативний та функціонально завершений модуль, що поєднує аналітичні можливості з наочним візуальним представленням даних.

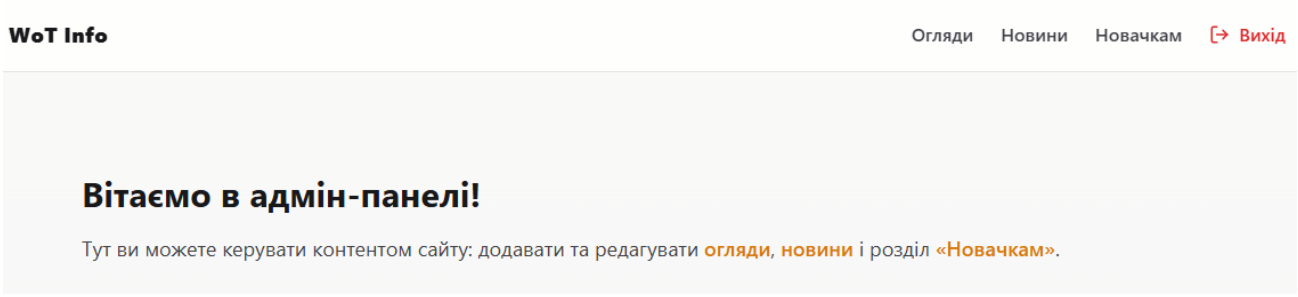
На мал. 4.21 представлено форму входу до адміністративної панелі вебсайту.



Мал. 4.21 – Форма входу адміністратора

Вона містить два обов’язкові поля для введення електронної пошти та пароля, а також кнопку «Увійти». Інтерфейс виконано у стриманому стилі з чіткою візуальною ієрархією елементів, що забезпечує зручність користування. Авторизація здійснюється шляхом перевірки облікових даних адміністратора, після чого користувач автоматично перенаправляється до головної сторінки адмін-панелі. У разі введення некоректних даних доступ до системи блокується, що запобігає несанкціонованому керуванню контентом.

Після успішної авторизації відкривається головна сторінка адміністративної панелі (мал. 4.22).



Мал. 4.22 – Головна сторінка в панелі адміністратора

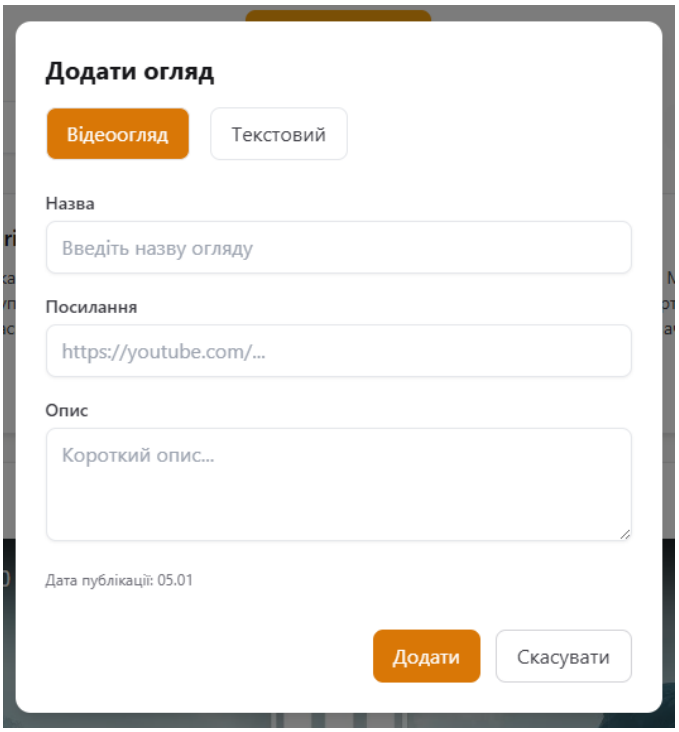
У верхній частині відображається навігаційне меню з доступом до основних розділів: Огляди, Новини, Новачкам, а також кнопка виходу з облікового запису.

У центральній частині сторінки розміщено інформаційний блок із привітальним повідомленням та коротким описом можливостей адміністратора. Звідси адміністратор може керувати наповненням сайту, контролюючи актуальність і коректність публікацій.

Сторінки «Огляди», «Новини» та «Новачкам» для адміністратора мають ідентичну логіку побудови та інтерфейс. Усі елементи контенту відображаються у вигляді списку, де кожен запис супроводжується кнопкою видалення (у вигляді хрестика). Це дозволяє швидко керувати матеріалами без переходу на окремі сторінки редагування.

У верхній частині кожної сторінки розташована кнопка «Додати», яка відкриває форму створення нового елемента відповідного типу.

На мал. 4.23 наведено приклад форми додавання огляду.



Мал. 4.23 – Форма додавання огляду

Адміністратор може обрати тип матеріалу: відеоогляд або текстовий огляд. Форма містить поля для введення назви, посилання (для відео), короткого опису, а також автоматично фіксує дату публікації. Інтерфейс реалізований у вигляді модального вікна, що дозволяє додавати контент без перезавантаження сторінки. Кнопки «Додати» та «Скасувати» забезпечують контроль над процесом створення матеріалу. Після підтвердження новий запис зберігається у базі даних і відразу відображається на відповідній публічній сторінці сайту.

У ході проведення комплексного тестування розробленого інформаційного вебсайту, присвяченого грі World of Tanks, було перевірено коректність роботи основних модулів системи. Тестування здійснювалося в умовах локального серверного середовища з використанням компонентної архітектури React.

За результатами перевірки встановлено, що навігаційна структура сайту реалізована коректно: усі сторінки відкриваються відповідно до заданої маршрутизації, переходи між розділами відбуваються без затримок і помилок.

Інтерфейсні елементи відображаються стабільно, зберігаючи єдиний стиль оформлення та логічне розташування компонентів. Функціональні можливості користувацької частини системи, зокрема фільтрація даних, пошук, сортування, перегляд детальної інформації та інтеграція зовнішніх ресурсів (3D-моделі техніки), працюють відповідно до технічних вимог. Усі інтерактивні елементи реагують коректно, забезпечуючи зручну та інтуїтивно зрозумілу взаємодію з користувачем.

Загалом результати тестування свідчать про те, що розроблена система є стабільною, функціонально повною та відповідає поставленим вимогам. Архітектура вебсайту дозволяє легко масштабувати проєкт, розширювати функціональність і наповнювати систему новими даними, що робить її придатною для подальшої експлуатації та розвитку.

ВИСНОВКИ

В межах виконання кваліфікаційної роботи було спроєктовано та реалізовано інформаційний вебсайт, присвячений грі World of Tanks, який забезпечує структурований доступ до довідкової, аналітичної та оглядової інформації для широкого кола користувачів. У процесі роботи виконано повний цикл розроблення програмного продукту: від аналізу предметної області та формування функціональних вимог до реалізації, тестування та оцінювання результатів.

У ході дослідження було проаналізовано існуючі інформаційні ресурси подібного призначення та визначено їхні основні недоліки, зокрема перевантаженість інтерфейсу, відсутність локалізації та обмежені можливості інтерактивної взаємодії. На основі отриманих результатів сформовано концепцію вебсайту з акцентом на зручність користування, модульну архітектуру та сучасні підходи до побудови інтерфейсу.

Реалізація проєкту виконана з використанням бібліотеки React у форматі односторінкового застосунку (SPA), що забезпечило високу швидкодію та плавну навігацію між розділами без перезавантаження сторінок. Для зберігання та керування даними використано хмарні сервіси Firebase, що дозволило реалізувати динамічне оновлення контенту та базові механізми безпеки. У результаті створено функціональні розділи сайту, зокрема інформаційні сторінки, довідник техніки, систему фільтрації та пошуку, інтеграцію зовнішніх сервісів і панель адміністратора для керування контентом.

Проведене тестування підтвердило коректність роботи всіх основних модулів системи, стабільність інтерфейсу та відповідність розробленого вебсайту поставленим вимогам. Отриманий програмний продукт може використовуватися як інформаційний ресурс для гравців та зацікавлених користувачів, а також як основа для подальшого розвитку.

Перспективами подальшого вдосконалення системи є розширення функціоналу за рахунок підключення офіційних API гри, впровадження авторизації користувачів із персоналізацією контенту, реалізація системи коментарів та

рейтингів, а також оптимізація адаптивності для мобільних пристроїв. Додатково доцільним є впровадження багатомовної підтримки та аналітичних інструментів для відстеження активності користувачів.

Поставлену мету кваліфікаційної роботи досягнуто, а отримані результати підтверджують практичну цінність і перспективність розробленої інформаційної системи.

СПИСОК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Official World of Tanks Website (EU). URL: https://worldoftanks.eu/uk/
2. IgroDrom. World of Tanks — огляд гри. URL: https://igrodrom.net/ua/world-of-tanks/
3. World of Tanks. Загальний гід для гравців. URL: https://worldoftanks.eu/uk/content/guide/general/
4. World of Tanks. Оновлення 2.0: ребаланс гри. URL: https://worldoftanks.eu/uk/news/general-news/update-2-0-rebalance/
5. Guru99. N-tier Architecture: System Concepts and Tips. URL: https://www.guru99.com/uk/n-tier-architecture-system-concepts-tips.html
6. Zmerzlyi I. Microservices Architecture. Medium. URL: https://medium.com/@IvanZmerzlyi/microservices-architecture-461687045b3d
7. DOU. Обговорення сучасних вебархітектур. URL: https://dou.ua/forums/topic/50894/
8. RALABS. Рендеринг: різновиди, застосування, порівняльна характеристика. Medium. URL: https://medium.com/@ralabs.ua/рендеринг-різновиди-застосування-порівняльна-характеристика-cc12e3a6707d
9. Wezom. Progressive Web Apps (PWA): можливості та переваги. URL: https://wezom.com.ua/ua/blog/pwa-prilozheniya-web-progressive-app
10. DOU. Обговорення SPA та сучасних frontend-рішень. URL: https://dou.ua/forums/topic/49698/
11. Wargaming Wiki. World of Tanks. URL: https://wiki.wargaming.net/en/World_of_Tanks
12. Tanks.gg — аналітика та 3D-моделі техніки World of Tanks. URL: https://tanks.gg/
13. Tomato.gg — статистика та аналітика гравців World of Tanks. URL: https://tomato.gg/
14. React Documentation. Using the Effect Hook. URL: https://uk.legacy.reactjs.org/docs/hooks-effect.html

15. React Documentation. FAQ: Internals of React. URL: https://uk.legacy.reactjs.org/docs/faq-internals.html

16. React Router. Official Documentation. URL: https://reactrouter.com/

17. Firebase Documentation. Cloud Firestore. URL: https://firebase.google.com/docs/firestore

18. Firebase Documentation. Firestore Client Libraries. URL: https://firebase.google.com/docs/firestore/client/libraries

19. Firebase Documentation. Security Rules. URL: https://firebase.google.com/docs/rules

20. Firebase Documentation. Authentication. URL: https://firebase.google.com/docs/auth

21. Cheerio.js. Official Documentation. URL: https://cheerio.js.org/

22. Tailwind CSS. Official Documentation. URL: https://tailwindcss.com/

23. SweetAlert2. Official Website. URL: https://sweetalert2.github.io/

24. Visual Studio Code. Official Website. URL: https://code.visualstudio.com/

ДОДАТКИ

Додаток А. Лістинг реалізованих компонентів сайту

App.jsx

```
import { StrictMode } from "react";
import { BrowserRouter, Routes, Route } from "react-router-dom";
import { AuthProvider } from "./context/AuthContext";

import Home from "./pages/Home";
import Details from "./pages/Details";
import Developers from "./pages/Developers";
import Requirements from "./pages/Requirements";
import Reviews from "./pages/Reviews";
import News from "./pages/News";
import Vehicles from "./pages/Vehicles";
import MainLayout from "./layouts/MainLayout";
import NotFound from "./pages/NotFound";
import Tutorials from "./pages/Tutorials"
import Maps from "./pages/Maps";
import Economy from "./pages/Economy";
import Login from "./pages/admin/Login";
import AdminHome from "./pages/admin/AdminHome";
import ReviewDetails from "./pages/ReviewDetails";
import NewsDetails from "./pages/NewsDetails";
import VehicleDetails from "./pages/VehicleDetails";
function AppContent() {
  return (
    <Routes>
      {/* Головна без хедера */}
      <Route path="/" element={<Home />} />

      {/* Усі інформаційні сторінки під єдиним лейаутом з хедером */}
      <Route element={<MainLayout />}>
        <Route path="/details" element={<Details />} />
        <Route path="/developers" element={<Developers />} />
        <Route path="/requirements" element={<Requirements />} />
        <Route path="/reviews" element={<Reviews />} />
        <Route path="/news" element={<News />} />
        <Route path="/vehicles" element={<Vehicles />} />
        <Route path="/tutorials" element={<Tutorials />} />
        <Route path="/maps" element={<Maps />} />
        <Route path="/economy" element={<Economy />} />
         <Route path="/login" element={<Login />} />
          <Route path="/admin" element={<AdminHome />} />
          <Route path="/reviews/:id" element={<ReviewDetails />} />
            <Route path="/news/:id" element={<NewsDetails  />} />
             <Route path="/vehicle/:name" element={<VehicleDetails />} />

      </Route>


        <Route path="*" element={<NotFound />} />
    </Routes>
  );
}

export default function App() {
  return (
```

```
      <StrictMode>
        <AuthProvider>
          <BrowserRouter>
            <AppContent />
          </BrowserRouter>
        </AuthProvider>
      </StrictMode>
    );
  }
```

AuthContext.jsx

```
  import { createContext, useContext, useEffect, useState } from "react";
  import { auth } from "../firebase";
  import { onAuthStateChanged } from "firebase/auth";

  const AuthContext = createContext();

  export const useAuth = () => useContext(AuthContext);

  export const AuthProvider = ({ children }) => {
    const [user, setUser] = useState(null);
    const [loading, setLoading] = useState(true);

    useEffect(() => {
      const unsub = onAuthStateChanged(auth, (user) => {
        setUser(user);
        setLoading(false);
      });

      return () => unsub();
    }, []);

    return (
      <AuthContext.Provider value={{ user }}>
        {!loading && children}
      </AuthContext.Provider>
    );
  };
```

MainLayout.jsx

```
  import React from "react";
  import { Outlet } from "react-router-dom";
  import Header from "../components/Header";

  export default function MainLayout() {
    return (
      <div className="min-h-[100svh] bg-gradient-to-b from-stone-50 to-zinc-
100 text-zinc-800">
        <Header />
        <div className="mx-auto max-w-6xl px-4 py-10 sm:py-14">
          <Outlet />
        </div>
      </div>
    );
  }
```

Header.jsx

```
import React from "react";
import { Link, NavLink, useNavigate } from "react-router-dom";
import { LogOut } from "lucide-react";

export default function Header() {
  const navigate = useNavigate();
  const isAdmin = localStorage.getItem("isAdmin") === "true";

  const base =
    "text-sm sm:text-base font-medium underline-offset-4 transition-colors
duration-200";

  const navUser = [
    { to: "/details", label: "Про гру" },
    { to: "/developers", label: "Розробники" },
    { to: "/requirements", label: "Системні вимоги" },
    { to: "/vehicles", label: "Техніка" },
    { to: "/economy", label: "Економіка" },
    { to: "/maps", label: "Мапи" },
    { to: "/reviews", label: "Огляди" },
    { to: "/news", label: "Новини" },
    { to: "/tutorials", label: "Новачкам" },
    { to: "/login", label: "Адмін-панель" },
  ];

  const navAdmin = [
    { to: "/reviews", label: "Огляди" },
    { to: "/news", label: "Новини" },
    { to: "/tutorials", label: "Новачкам" },
  ];

  const handleLogout = () => {
    localStorage.removeItem("isAdmin");
    navigate("/details");
  };

  return (
    <header  className="sticky  top-0  z-50  backdrop-blur-md  bg-white/70
border-b border-zinc-200">
      <div className="mx-auto max-w-6xl px-4">
        <div className="h-16 flex items-center justify-between gap-4">

          <Link
            to="/"
            className="font-extrabold text-lg tracking-wide text-zinc-900
hover:text-amber-600 transition"
            title="Повернутися на головну"
          >
            WoT Info
          </Link>


          <div className="flex items-center gap-6">
            <nav className="flex items-center gap-4 sm:gap-6">
              {(isAdmin ? navAdmin : navUser).map((i) => (
                <NavLink
                  key={i.to}
                  to={i.to}
                  className={({ isActive }) =>
                    `${base} ${
                      isActive
                        ? "text-amber-700"
                        : "text-zinc-700 hover:text-amber-600"
                    }`
                  }
```

```
                    >
                      {i.label}
                    </NavLink>
                  ))}
                </nav>


                {isAdmin && (
                  <button
                    onClick={handleLogout}
                    className="flex items-center gap-2 text-sm sm:text-base
font-medium text-red-600 hover:text-red-700 transition"
                    title="Вийти з адмінки"
                  >
                    <LogOut size={18} />
                    Вихід
                  </button>
                )}
              </div>
            </div>
          </div>
        </header>
      );
    }
```

FiltersBar.jsx

```
    import React, { useMemo } from "react";
    import { nationFlags, typeIcons } from "../data/flags";

    // словник назв націй (підказки)
    const nationNames = {
      china: "Китай",
      ussr: "СРСР",
      usa: "США",
      france: "Франція",
      germany: "Німеччина",
      italy: "Італія",
      japan: "Японія",
      poland: "Польща",
      sweden: "Швеція",
      uk: "Велика Британія",
      czechoslovakia: "Чехословаччина",
    };

    // словник назв типів (підказки)
    const typeNames = {
      lighttank: "Легкі танки",
      mediumtank: "Середні танки",
      heavytank: "Важкі танки",
      atspg: "ПТ-САУ",
      "at-spg": "ПТ-САУ",
      spg: "Артилерія",
    };

    export default function FiltersSidebar({
      tanks,
      filteredTanks, // ✅ тепер отримуємо
      selectedNations,
      setSelectedNations,
      selectedTypes,
      setSelectedTypes,
      selectedTier,
```

```
        setSelectedTier,
        premiumOnly,
        setPremiumOnly,
      }) {
        // унікальні нації
        const uniqueNations = useMemo(() => {
          const seen = new Set();
          return tanks.reduce((acc, t) => {
            if (!seen.has(t.nation)) {
              seen.add(t.nation);
              const     flagKey     =     t.nation.charAt(0).toUpperCase()     +
t.nation.slice(1);
              acc.push({
                code: t.nation,
                name: nationNames[t.nation] || t.nation,
                flag: nationFlags[flagKey] || null,
              });
            }
            return acc;
          }, []);
        }, [tanks]);

        // унікальні ранги
        const uniqueTiers = useMemo(() => {
          return [...new Set(tanks.map((t) => t.tier))].sort((a, b) => a - b);
        }, [tanks]);

        // тоглери
        const toggleNation = (code) => {
          setSelectedNations(
            selectedNations.includes(code)
              ? selectedNations.filter((c) => c !== code)
              : [...selectedNations, code]
          );
        };

        const toggleType = (type) => {
          setSelectedTypes(
            selectedTypes.includes(type)
              ? selectedTypes.filter((t) => t !== type)
              : [...selectedTypes, type]
          );
        };

        return (
          <aside  className="sticky  top-20  h-max  w-60  bg-white  rounded-2xl
shadow-md p-5 border border-zinc-200">
            {/* Заголовок з кількістю знайдених */}
            <h4  className="mb-3  text-sm  font-semibold  text-zinc-500  uppercase
tracking-wider">
              Фільтри
                (Танків            :<span            className="text-orange-500">
{filteredTanks.length}   </span>)

            </h4>

            {/* Нації */}
            <div className="mb-6">
              <h5 className="font-semibold mb-2">Нації</h5>
              <div className="grid grid-cols-3 gap-2">
                {uniqueNations.map((n) => (
                  <button
                    key={n.code}
                    onClick={() => toggleNation(n.code)}
                    title={n.name}
```

```
                    className={`flex items-center justify-center p-1 rounded-lg
border transition ${
                      selectedNations.includes(n.code)
                        ? "border-orange-500 shadow"
                        : "border-zinc-300 hover:border-orange-300"
                    }`}
                  >
                    {n.flag && (
                      <img
                        src={n.flag}
                        alt={n.name}
                        className="w-8 h-6 object-contain"
                      />
                    )}
                  </button>
                ))}
              </div>
            </div>

            {/* Типи */}
            <div className="mb-6">
              <h5 className="font-semibold mb-2">Типи</h5>
              <div className="grid grid-cols-3 gap-2">
                {Object.keys(typeNames).map((key) => (
                  <button
                    key={key}
                    onClick={() => toggleType(key)}
                    title={typeNames[key]}
                    className={`flex items-center justify-center p-1 rounded-lg
border transition ${
                      selectedTypes.includes(key)
                        ? "border-orange-500 shadow"
                        : "border-zinc-300 hover:border-orange-300"
                    }`}
                  >
                    <img
                      src={typeIcons[key]}
                      alt={typeNames[key]}
                      className="w-8 h-8 object-contain"
                    />
                  </button>
                ))}
              </div>
            </div>

            {/* Ранги */}
            <div className="mb-6">
              <h5 className="font-semibold mb-2">Ранг</h5>
              <div className="flex flex-wrap gap-2">
                {uniqueTiers.map((tier) => (
                  <button
                    key={tier}
                    onClick={() =>
                      setSelectedTier(selectedTier === tier ? null : tier)
                    }
                    className={`px-3 py-1.5 rounded-lg border text-sm transition
${
                      selectedTier === tier
                        ? "bg-orange-500 text-white shadow"
                        : "bg-zinc-100 text-zinc-700 hover:bg-zinc-200"
                    }`}
                  >
                    {tier}
                  </button>
                ))}
```

```
            </div>
          </div>

          {/* Преміум */}
          <div>
            <h5 className="font-semibold mb-2">Преміум</h5>
            <button
              onClick={() => setPremiumOnly(!premiumOnly)}
              className={`w-full px-3 py-2 rounded-lg border text-sm
transition ${
                premiumOnly
                  ? "bg-yellow-500 text-white shadow"
                  : "bg-zinc-100 text-zinc-700 hover:bg-zinc-200"
              }`}
            >
              Лише преміум
            </button>
          </div>
        </aside>
      );
    }
```

Overview.jsx

```
    import React from "react";

    export default function Overview() {
      return (
        <section id="overview" className="scroll-mt-24">
          <h2 className="text-3xl sm:text-4xl font-bold text-zinc-900">Про
гру</h2>

          {/* Перший абзац + трейлер */}
          <p className="mt-4 text-zinc-700 leading-relaxed">
            <span className="text-orange-600 font-semibold">World of
Tanks</span> — це багатокористувацька онлайн-гра, у якій гравці керують
бронетехнікою XX століття, беручи участь у командних боях на різноманітних
мапах. Основна мета гри — знищити команду супротивника або виконати тактичні
завдання на карті. Проєкт вирізняється глибокою системою розвитку, унікальною
фізикою руху танків, детально опрацьованими моделями бойової техніки, а також
активною підтримкою спільноти.
          </p>

          <div className="mt-6 max-w-3xl mx-auto aspect-video">
            <iframe
              width="100%"
              height="100%"
              src="https://www.youtube.com/embed/mWI-ltfKenw"
              title="World of Tanks Trailer"
              frameBorder="0"
              allow="accelerometer; autoplay; clipboard-write; encrypted-
media; gyroscope; picture-in-picture"
              allowFullScreen
              className="rounded-xl shadow-md border"
            ></iframe>
          </div>

          {/* Наступні абзаци */}
          <p className="mt-6 text-zinc-700 leading-relaxed">
            На відміну від аркадних шутерів, WoT робить акцент на{" "}
            <span className="text-orange-600 font-semibold">тактичному
мисленні</span>,{" "}
```

```
            <span   className="text-orange-600   font-semibold">координації   в
команді</span> та{" "}
            <span   className="text-orange-600   font-semibold">знанні   мап   і
укриттів</span>.
          </p>

          <p className="mt-6 text-zinc-700 leading-relaxed">
            У грі реалізовано понад{" "}
            <span    className="text-orange-600    font-semibold">600    одиниць
техніки</span> з більш ніж десяти націй. Кожен танк має власні сильні та слабкі
сторони, тому гравцям пропонується велике поле для експериментів і стратегії.
          </p>

          <p className="mt-6 text-zinc-700 leading-relaxed">
            Однією з ключових особливостей WoT є система{" "}
            <span   className="text-orange-600   font-semibold">дослідження   та
розвитку</span>, що заохочує до поступового прогресу.
          </p>

          <p className="mt-6 text-zinc-700 leading-relaxed">
            Велика увага приділена{" "}
            <span    className="text-orange-600    font-semibold">реалістичному
дизайну</span> — моделі танків створені на основі архівних креслень.
          </p>

          <p className="mt-6 text-zinc-700 leading-relaxed">
            WoT має активну{" "}
            <span      className="text-orange-600      font-semibold">спільноту
гравців</span> — регулярні оновлення, івенти, кланові війни, сезонні нагороди та
турніри.
          </p>

          <div className="mt-6 rounded-2xl border border-zinc-200 bg-white p-5
shadow-sm">
            <ul className="list-disc pl-6 space-y-2 text-zinc-700">
              <li>
                <span   className="text-orange-600   font-semibold">Понад   600
танків</span> — від легких до надважких.
              </li>
              <li>
                <span     className="text-orange-600     font-semibold">Система
прокачки</span> техніки з досвідом і модулями.
              </li>
              <li>
                <span className="text-orange-600 font-semibold">Щоденні місії,
події та акції</span> підтримують динамічність гри.
              </li>
            </ul>
          </div>
        </section>
      );
    }
```

Roles.jsx

```
    import React from "react";
    import { ROLES } from "../../data/roles";

    export default function Roles({ selected, setSelected }) {
      return (
        <section id="roles" className="scroll-mt-24">
      <h3   className="text-2xl   font-semibold   text-zinc-900   mb-4">Ролі   та
командна гра</h3>
```

```
        {/* Короткий вступ */}
        <div  className="mb-6  rounded-xl  border  border-zinc-200  bg-white  p-5
shadow-sm text-zinc-700 leading-relaxed">
          У  грі  <span  className="text-orange-600  font-semibold">World  of
Tanks</span> командна гра та
          правильний вибір ролі мають критичне значення для перемоги. Злагоджена
тактика, розуміння
          карти, розподіл функцій і ефективне використання танкових класів —
ключ до успіху на полі бою.
        </div>

        {/* Перемикачі */}
        <div className="flex flex-wrap gap-2 mb-6">
          {ROLES.map((role) => (
            <button
              key={role.id}
              onClick={() => setSelected(role)}
              className={`px-4  py-2  rounded-full  text-sm  font-medium  border
transition-all
                ${
                  selected.id === role.id
                    ? "bg-orange-500 text-white border-orange-600"
                    :  "bg-zinc-100  text-zinc-700  hover:bg-orange-100  border-
zinc-300"
                }`}
            >
              {role.name}
            </button>
          ))}
        </div>

        {/* Картка вибраної ролі */}
        <div className="grid grid-cols-1 md:grid-cols-2 gap-6 items-center">
          <div  className="overflow-hidden  rounded-xl  shadow-md  border  border-
zinc-200">
            <img src={selected.img} alt={selected.name} className="w-full h-auto
object-cover" />
          </div>
          <div>
            <h4    className="text-xl    font-semibold    text-orange-600    mb-
2">{selected.name}</h4>
            <p className="text-zinc-700 leading-relaxed text-base">
              {selected.desc}
            </p>
          </div>
        </div>

        {/* Командна взаємодія */}
        <div  className="mt-10  rounded-xl  border  border-zinc-200  bg-white  p-5
shadow-sm space-y-4">
          <h4     className="text-lg     font-semibold     text-zinc-800">Командна
взаємодія</h4>
          <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
            {[
              {
                label: "Координація",
                desc: "Швидкі команди (клавіша “T”) для комунікації."
              },
              {
                label: "Баланс складу",
                desc:  "Розподіл ролей у взводі/команді: ЛТ,  СТ,  ТТ,  ПТ-САУ,
САУ."
              },
              {
```

```
              label: "Тактика",
              desc: "Використання укриттів, флангові атаки, гра від позиції."
            },
            {
              label: "Мінікарта",
              desc: "Обов’язкова для планування руху та підтримки."
            }
          ].map((item, idx) => (
            <div key={idx} className="border rounded-lg p-3 bg-orange-50/50">
              <p className="font-semibold text-orange-600">{item.label}</p>
              <p className="text-sm text-zinc-700">{item.desc}</p>
            </div>
          ))}
        </div>
      </div>
    </section>
      );
    }
```

Crew.jsx

```
    import React from "react";
    import { CREW } from "../../data/crew";

    export default function Crew() {
      return (
     <section id="progression" className="scroll-mt-24">
          <h3 className="text-2xl font-bold text-orange-600 mb-6 border-b-2
border-orange-300 pb-2">
            Екіпаж
          </h3>

          <div className="space-y-6">
            {/* Вступ */}
            <div className="bg-gradient-to-r from-orange-50 to-white border-l-
4 border-orange-400 p-5 rounded-xl shadow">
              <p className="text-zinc-800 text-lg leading-relaxed">
                Екіпаж це місток між гравцем і бойовою машиною. Кожен танкіст
має
                свою спеціальність, набуває досвіду, прокачує вміння й навички
та
                прямо впливає на ефективність техніки. Досвідчений екіпаж —
значна
                перевага на полі бою.
              </p>
            </div>
    <div className="flex items-center gap-6 mb-4">
      <div className="flex items-center gap-2">
        <span className="w-4 h-4 bg-green-300 rounded-sm border border-green-
700"></span>
        <span className="text-sm text-zinc-700">Бонус</span>
      </div>
      <div className="flex items-center gap-2">
        <span  className="w-4  h-4  bg-red-300  rounded-sm  border  border-red-
700"></span>
        <span className="text-sm text-zinc-700">Втрата / Контузія</span>
      </div>
    </div>
            {/* Картки екіпажу */}
            <div className="space-y-5">
              {CREW.map((member, idx) => (
                <div
                  key={idx}
```

```
                    className="flex items-start gap-5 p-5 bg-white rounded-2xl
shadow-md border border-orange-200 hover:shadow-lg transition"
                  >
                    {/* Іконка */}
                    <div className="flex-shrink-0">
                      <img
                        src={member.icon}
                        alt={member.name}
                        className="w-25 h-20 rounded-lg border border-orange-300
bg-orange-50 p-1"
                      />
                    </div>

                    {/* Інфо */}
                    <div className="flex-grow">
                      <p className="text-xl font-semibold text-orange-700 mb-1">
                        {member.name}
                      </p>
                      <p className="text-sm text-zinc-700 mb-3">
                        {member.description}
                      </p>

                      {/* Бонус і Втрата */}
                      <div className="flex flex-col gap-2">
                        <div   className="bg-green-50   border   border-green-300
text-green-700 px-3 py-2 rounded-lg text-sm font-medium shadow-sm">
                          {member.bonus}
                        </div>
                        <div  className="bg-red-50  border  border-red-300  text-
red-700 px-3 py-2 rounded-lg text-sm font-medium shadow-sm">
                          {member.loss}
                        </div>
                      </div>
                    </div>
                  </div>
                ))}
              </div>

              {/* Навички та перки */}
              <div>
                <h4 className="text-lg font-semibold text-orange-600 mb-3">
                  Навички та перки
                </h4>
                <div className="grid md:grid-cols-2 gap-4">
                  {[
                    {
                      title: "Бойове братство",
                      desc:  "+5%  до  всіх  умінь,  якщо  вивчене  у  всіх  членів
екіпажу."
                    },
                    {
                      title: "Наставник",
                      desc: "Командир збільшує досвід інших танкістів на 10%."
                    },
                    {
                      title: "Майстер на всі руки",
                      desc: "Командир частково компенсує втрату іншого екіпажу."
                    },
                    {
                      title: "Прогрес навичок",
                      desc: "Кожне нове вміння потребує вдвічі більше досвіду."
                    }
                  ].map((perk, idx) => (
                    <div
                      key={idx}
```

```
                    className="bg-orange-50 border border-orange-200 p-4
rounded-lg shadow-sm"
                  >
                    <p          className="font-semibold          text-orange-
700">{perk.title}</p>
                    <p className="text-sm text-zinc-700">{perk.desc}</p>
                  </div>
                ))}
              </div>
            </div>

            {/* Бонуси / штрафи */}
            <div>
              <h4 className="text-lg font-semibold text-orange-600 mb-3">
                Бонуси та штрафи
              </h4>
              <div  className="bg-white  border  border-zinc-200  p-5  rounded-xl
shadow-sm">
                <ul  className="list-disc  pl-6  space-y-2  text-sm  text-zinc-
700">
                  <li>
                    <span              className="font-semibold">Командирський
бонус:</span> до
                    +10% до навичок екіпажу.
                  </li>
                  <li>
                    <span         className="font-semibold">Обладнання:</span>
вентиляція дає
                    +5% до всіх навичок.
                  </li>
                  <li>
                    <span  className="font-semibold">Спецпаї:</span>  +10%  до
всіх
                    умінь (шоколад, доппаек, тощо).
                  </li>
                  <li>
                    <span className="font-semibold">Без перенавчання:</span> -
25% або
                    -50% ефективності залежно від техніки.
                  </li>
                  <li>
                    <span  className="font-semibold">Контузія:</span>  -50%
                    ефективності  та  втрата  всіх  перків,  крім  «Бойового
братства».
                  </li>
                </ul>
              </div>
            </div>

            {/* Звання */}
            <div  className="bg-gradient-to-r  from-zinc-50  to-white  border-l-4
border-zinc-300 p-4 rounded-lg text-sm shadow-sm">
              <p>
                <span      className="font-semibold      text-zinc-800">Звання
екіпажу:</span>{" "}
                підвищуються  з  досвідом.  Максимальне  звання  —  майор
(командир),
                капітан       (наводчик/водій),       старший       лейтенант
(радист/заряджаючий).
              </p>
            </div>
          </div>
        </section>
      );
    }
```

Modes.jsx

```
      import React from "react";

      export default function Modes({ tab, setTab, filter, setFilter,
filteredModes }) {
        return (
          <section id="modes" className="scroll-mt-24">
        <h3   className="text-2xl   font-semibold   text-zinc-900   mb-6">Режими
гри</h3>
       <p className="text-base text-zinc-600 mb-2">
          Виберіть рівень боїв або подій, щоб переглянути відповідні режими:
        </p>
        {/* Вкладки */}
        <div className="flex gap-2 mb-4 flex-wrap">
          {[
            { id: "permanent", label: "Постійні" },
            { id: "temporary", label: "Тимчасові" },
            { id: "special", label: "Спеціальні" },
          ].map(({ id, label }) => (
            <button
              key={id}
              onClick={() => setTab(id)}
              className={`px-4 py-1 rounded-full border ${
                tab === id
                  ? "bg-orange-500 text-white border-orange-500"
                  : "bg-white text-zinc-700 border-zinc-300 hover:bg-orange-50"
              } transition`}
            >
              {label}
            </button>
          ))}
        </div>
        {/* Фільтр по рівню */}
        <div className="flex gap-2 mb-6 flex-wrap">
          {[
            { id: "all", label: "Усі рівні" },
            { id: "x-only", label: "Лише X рівень" },
            { id: "event", label: "Лише події" },
          ].map(({ id, label }) => (
            <button
              key={id}
              onClick={() => setFilter(id)}
              className={`px-4 py-1 rounded-full border ${
                filter === id
                  ? "bg-zinc-900 text-white border-zinc-900"
                  : "bg-white text-zinc-700 border-zinc-300 hover:bg-zinc-100"
              } transition`}
            >
              {label}
            </button>
          ))}
        </div>
        {/* Список режимів */}
        <div className="grid gap-4">
          {filteredModes.map((mode, idx) => (
            <div
              key={idx}
              className="flex items-start gap-4 rounded-xl bg-white shadow-sm
border border-zinc-200 hover:shadow-md p-4 transition"
            >
              <img
```

```
                src={mode.icon}
                alt={mode.title}
                className="w-20 h-20 object-contain"
              />
              <div>
                <h5      className="text-lg      font-semibold      text-orange-
600">{mode.title}</h5>
                <p className="text-zinc-800 text-base mt-1">{mode.desc}</p>
              </div>
            </div>
          ))}
        </div>
      </section>
        );
      }
```

CurrenciesSection.jsx

```
      import React, { useState } from "react";
      import { TABS } from "../../data/tabs";

      export default function CurrenciesSection() {
        const [active, setActive] = useState("exp");

        return (
          <section id="currencies" className="scroll-mt-24">
            <h2 className="text-3xl sm:text-4xl font-bold text-zinc-900">
              Ігрові ресурси та валюти
            </h2>
            <p className="mt-4 text-zinc-700 leading-relaxed max-w-2xl">
              У World of Tanks є чотири головні ресурси: досвід, кредити, золото
та
              бони. Виберіть вкладку, щоб дізнатись більше.
            </p>

            {/* Tabs (тільки іконки по центру) */}
            <div className="mt-6 flex justify-center gap-6 flex-wrap">
              {TABS.map((tab) => (
                <button
                  key={tab.id}
                  onClick={() => setActive(tab.id)}
                  className={`flex  flex-col  items-center  px-4  py-2  rounded-lg
border shadow-sm transition w-24 ${
                    active === tab.id
                      ? "bg-amber-50 border-amber-400"
                      : "bg-white hover:bg-zinc-50 border-zinc-200"
                  }`}
                >
                  <img src={tab.icon} alt={tab.title} className="w-12 h-12" />
                </button>
              ))}
            </div>

            {/* Active content */}
            <div  className="mt-8  rounded-xl  border  border-zinc-200  bg-white
shadow p-6">
              <h3 className="text-xl font-semibold text-amber-700 mb-3">
                {TABS.find((t) => t.id === active)?.title}
              </h3>
              <p className="text-zinc-700 leading-relaxed mb-5">
                {TABS.find((t) => t.id === active)?.description}
              </p>
```

```
              {/* Де отримати */}
              <div  className="mb-5  p-4  rounded-lg  bg-green-50  border  border-
green-300">
                <h4     className="text-green-700     font-semibold     mb-2">Де
отримати:</h4>
                <ul className="list-disc pl-6 space-y-1 text-sm text-green-700">
                  {TABS.find((t) => t.id === active)?.earn.map((item, i) => (
                    <li key={i}>{item}</li>
                  ))}
                </ul>
              </div>

              {/* На що витратити */}
              <div className="p-4 rounded-lg bg-blue-50 border border-blue-300">
                <h4    className="text-blue-700    font-semibold    mb-2">На    що
витратити:</h4>
                <ul className="list-disc pl-6 space-y-1 text-sm text-blue-700">
                  {TABS.find((t) => t.id === active)?.spend.map((item, i) => (
                    <li key={i}>{item}</li>
                  ))}
                </ul>
              </div>
            </div>
          </section>
        );
      }
```

PremiumSection.jsx

```
      import React, { useState, useEffect } from "react";
      import { FaCoins, FaStar, FaTasks, FaUsers, FaMapMarkedAlt, FaExchangeAlt
} from "react-icons/fa";

      // таблиця цін у золоті
      const PRICES = {
        360: 20500,
        180: 11900,
        90: 6900,
        30: 2500,
        14: 1800,
        7: 1250,
        3: 650,
        1: 250,
      };

      export default function PremiumSection() {
        const [days, setDays] = useState(30);
        const [gold, setGold] = useState(PRICES[30]);

        const calcCredits = () => {
          const bonus = (days / 7) * 750000;
          return bonus.toLocaleString("uk-UA");
        };

        useEffect(() => {
          if (PRICES[days]) {
            setGold(PRICES[days]);
          } else {
            setGold(0);
          }
        }, [days]);

        return (
```

```
          <section id="premium" className="scroll-mt-24">
            <h3 className="text-2xl font-semibold text-amber-600 flex items-
center gap-2">
              <FaStar /> Преміум акаунт
            </h3>
            <p className="mt-3 text-zinc-700 leading-relaxed">
              Танковий преміум акаунт відкриває більше можливостей для прогресу:
              додаткові кредити, досвід, завдання та бонуси для взводів.
            </p>

            {/* Картки бонусів */}
            <div className="mt-6 grid sm:grid-cols-2 gap-5">
              <div className="p-5 rounded-xl bg-white shadow border-l-4 border-
amber-500">
                <h4 className="font-semibold text-zinc-900 flex items-center
gap-2">
                  <FaStar className="text-amber-500" /> ×3 множник досвіду
                </h4>
                <p className="text-sm mt-2 text-zinc-700">
                  Досліджуйте техніку в рази швидше, комбінуючи ×2 та ×3 бонуси.
                </p>
              </div>

              <div className="p-5 rounded-xl bg-white shadow border-l-4 border-
green-500">
                <h4 className="font-semibold text-zinc-900 flex items-center
gap-2">
                  <FaCoins className="text-green-600" /> Сховище кредитів
                </h4>
                <p className="text-sm mt-2 text-zinc-700">
                  +10% до кредитів автоматично накопичується у сховище (до
750k/тиждень).
                </p>
              </div>

              <div className="p-5 rounded-xl bg-white shadow border-l-4 border-
blue-500">
                <h4 className="font-semibold text-zinc-900 flex items-center
gap-2">
                  <FaTasks className="text-blue-600" /> Преміум завдання
                </h4>
                <p className="text-sm mt-2 text-zinc-700">
                  Додаткові квести з винагородами: кредити, бони та інше.
                </p>
              </div>

              <div className="p-5 rounded-xl bg-white shadow border-l-4 border-
purple-500">
                <h4 className="font-semibold text-zinc-900 flex items-center
gap-2">
                  <FaUsers className="text-purple-600" /> Взводний бонус
                </h4>
                <p className="text-sm mt-2 text-zinc-700">
                  +15% до досвіду та +10% до кредитів для взводу.
                </p>
              </div>

              <div className="p-5 rounded-xl bg-white shadow border-l-4 border-
red-500 sm:col-span-2">
                <h4 className="font-semibold text-zinc-900 flex items-center
gap-2">
                  <FaMapMarkedAlt className="text-red-500" /> +1 слот для карт
                </h4>
                <p className="text-sm mt-2 text-zinc-700">
```

```
                Заблокуйте набридливу мапу та продовжуйте грати із
задоволенням.
              </p>
            </div>
          </div>

          {/* Калькулятор вигідності */}
          <div className="mt-8 rounded-xl bg-zinc-50 border p-6">
            <h4 className="text-lg font-bold text-zinc-900 mb-3">Калькулятор
вигідності</h4>
            <div className="flex flex-col sm:flex-row gap-4">
              <div>
                <label className="text-sm text-zinc-600">Днів преміуму</label>
                <select
                  value={days}
                  onChange={(e) => setDays(Number(e.target.value))}
                  className="w-full border rounded-lg px-3 py-2 mt-1"
                >
                  {Object.keys(PRICES).map((d) => (
                    <option key={d} value={d}>
                      {d} днів
                    </option>
                  ))}
                </select>
              </div>
              <div>
                <label className="text-sm text-zinc-600">Ціна у золоті</label>
                <input
                  type="text"
                  value={`${gold} золота`}
                  disabled
                  className="w-full border rounded-lg px-3 py-2 mt-1 bg-zinc-
100 text-zinc-700"
                />
              </div>
            </div>
            <p className="mt-4 text-sm text-zinc-700">
              Орієнтовно ви зможете отримати:{" "}
              <span className="font-semibold text-green-600">{calcCredits()}
кредитів</span>{" "}
              бонусом зі сховища за цей період.
            </p>
          </div>

          {/* Конверсія валют та досвіду */}
          <div className="mt-10 rounded-xl bg-white border p-6 shadow">
            <h4 className="text-lg font-bold text-zinc-900 flex items-center
gap-2">
              <FaExchangeAlt className="text-amber-500" /> Конверсія валют та
досвіду
            </h4>
            <p className="mt-3 text-zinc-700 leading-relaxed">
              В грі <span className="font-semibold">World of Tanks</span>
існує можливість
              конверсії внутрішньоігрових ресурсів і валюти:
            </p>

            <ul className="mt-4 space-y-3 text-sm text-zinc-700">
              <li>
                <span className="font-semibold">Золото → Кредити</span> <br />
                Курс: <span className="font-semibold">1 → 400</span>
              </li>
              <li>
                <span  className="font-semibold">Бойовий  досвід  →  Вільний
досвід</span> <br />
```

```
                Курс: <span className="font-semibold">25 XP + 1 золото → 25
вільного XP</span>
              </li>
            </ul>

            <div className="mt-5 bg-zinc-50 border rounded-lg p-4">
              <p className="text-sm text-zinc-700 leading-relaxed">
                Преміум техніка та машини з дослідженими модулями вважаються
елітними —
                їхній бойовий досвід можна переводити у вільний за золото.
                Це дозволяє швидше досліджувати нову техніку або підвищувати
                рівень навичок екіпажу.
              </p>
            </div>
          </div>
        </section>
      );
    }
```

AddNewsModal.jsx

```
    import React, { useState } from "react";
    import { db } from "../../firebase";
    import { collection, addDoc } from "firebase/firestore";
    import Swal from "sweetalert2";

    export default function AddNewsModal({ onClose, onAdded }) {
      const [title, setTitle] = useState("");
      const [type, setType] = useState("Новина");
      const [text, setText] = useState("");
      const [source, setSource] = useState("");
      const [error, setError] = useState("");

      // Дата у форматі ДД.ММ
      const getCurrentDate = () => {
        const now = new Date();
        return `${String(now.getDate()).padStart(2, "0")}.${String(
          now.getMonth() + 1
        ).padStart(2, "0")}`;
      };

      const resetForm = () => {
        setTitle("");
        setType("Новина");
        setText("");
        setSource("");
        setError("");
      };

      const handleSubmit = async (e) => {
        e.preventDefault();

        if (!title.trim()) {
          setError("Введіть заголовок!");
          return;
        }
        if (!text.trim()) {
          setError("Введіть текст новини!");
          return;
        }

        const newsData = {
          title,
```

```
            type,
            text,
            source: source || "—", // якщо пусто, ставимо тире
            date: getCurrentDate(),
          };

          try {
            const docRef = await addDoc(collection(db, "news"), newsData);
            console.log("Новина додана:", newsData);

            Swal.fire({
              icon: "success",
              title: "Новина додана!",
              showConfirmButton: false,
              timer: 1000,
            });

            if (onAdded) {
              onAdded({ id: docRef.id, ...newsData });
            }

            resetForm();
            onClose();
          } catch (error) {
            console.error("Помилка при додаванні новини:", error);
            setError("Не вдалося додати новину. Спробуйте ще раз.");
          }
        };

        return (
          <div  className="fixed  inset-0  bg-black/50  flex  items-center  justify-
center z-50">
            <div  className="bg-white  rounded-xl  shadow-lg  p-6  w-full  max-w-lg">
              <h3   className="text-xl   font-bold   text-zinc-900   mb-4">Додати
новину</h3>

              <form onSubmit={handleSubmit} className="space-y-4">
                {/* Заголовок */}
                <div>
                  <label className="block text-sm font-medium text-zinc-700">
                    Заголовок
                  </label>
                  <input
                    type="text"
                    value={title}
                    onChange={(e) => setTitle(e.target.value)}
                    className="mt-1 w-full border rounded-lg px-3 py-2 shadow-sm
focus:ring-2 focus:ring-amber-500 focus:outline-none"
                    placeholder="Введіть заголовок"
                    required
                  />
                </div>

                {/* Тип */}
                <div>
                  <label className="block text-sm font-medium text-zinc-700">
                    Тип
                  </label>
                  <select
                    value={type}
                    onChange={(e) => setType(e.target.value)}
                    className="mt-1 w-full border rounded-lg px-3 py-2 shadow-sm
focus:ring-2 focus:ring-amber-500 focus:outline-none"
                  >
                    <option>Новина</option>
```

```
                    <option>Патч</option>
                    <option>Анонс</option>
                    <option>Івент</option>
                  </select>
                </div>

                {/* Текст */}
                <div>
                  <label className="block text-sm font-medium text-zinc-700">
                    Текст новини
                  </label>
                  <textarea
                    rows="4"
                    value={text}
                    onChange={(e) => setText(e.target.value)}
                    className="mt-1 w-full border rounded-lg px-3 py-2 shadow-sm
focus:ring-2 focus:ring-amber-500 focus:outline-none"
                    placeholder="Введіть текст новини..."
                    required
                  ></textarea>
                </div>

                {/* Джерело */}
                <div>
                  <label className="block text-sm font-medium text-zinc-700">
                    Джерело (необов’язково)
                  </label>
                  <input
                    type="text"
                    value={source}
                    onChange={(e) => setSource(e.target.value)}
                    className="mt-1 w-full border rounded-lg px-3 py-2 shadow-sm
focus:ring-2 focus:ring-amber-500 focus:outline-none"
                    placeholder="Наприклад: офіційний сайт"
                  />
                </div>

                {/* Дата */}
                <p className="text-xs text-zinc-500">
                  Дата публікації: {getCurrentDate()}
                </p>

                {/* Повідомлення про помилку */}
                {error && <p className="text-sm text-red-600">{error}</p>}

                {/* Кнопки */}
                <div className="flex justify-end gap-3 pt-4">
                  <button
                    type="submit"
                    className="px-4  py-2  rounded-lg  bg-amber-600  text-white
hover:bg-amber-700"
                  >
                    Додати
                  </button>
                  <button
                    type="button"
                    onClick={onClose}
                    className="px-4 py-2 rounded-lg border border-zinc-300 text-
zinc-700 bg-white hover:bg-red-500 hover:text-white transition"
                  >
                    Скасувати
                  </button>
                </div>
              </form>
            </div>
```

```
        </div>
      );
    }
```

AddReviewModal.jsx

```
    import React, { useState } from "react";
    import { db } from "../../firebase";
    import { collection, addDoc } from "firebase/firestore";
    import Swal from "sweetalert2";
    export default function AddReviewModal({ onClose, onAdded }) {
      const [type, setType] = useState("video");
      const [title, setTitle] = useState("");
      const [url, setUrl] = useState("");
      const [description, setDescription] = useState("");
      const [intro, setIntro] = useState("");
    const [body, setBody] = useState("");
    const [important, setImportant] = useState("");
    const [conclusion, setConclusion] = useState("");
      const [author, setAuthor] = useState("");
      const [error, setError] = useState("");

      const getCurrentDate = () => {
        const now = new Date();
        return `${String(now.getDate()).padStart(2, "0")}.${String(
          now.getMonth() + 1
        ).padStart(2, "0")}`;
      };

    const resetForm = () => {
      setTitle("");
      setUrl("");
      setDescription("");
      setIntro("");
      setBody("");
      setImportant("");
      setConclusion("");
      setAuthor("");
      setError("");
    };

      const handleSubmit = async (e) => {
        e.preventDefault();

        if (!title.trim()) {
          setError("Введіть назву!");
          return;
        }
        if (type === "video" && (!url.trim() || !description.trim())) {
          setError("Заповніть всі поля для відеоогляду!");
          return;
        }
    if (type === "text" && (!intro.trim() && !body.trim() && !important.trim()
&& !conclusion.trim())) {
      setError("Заповніть хоча б одну секцію текстового огляду!");
      return;
    }

        const reviewData = {
          type,
          title,
          date: getCurrentDate(),
        };
```

```
          if (type === "video") {
            reviewData.url = url;
            reviewData.description = description;
          }
          if (type === "text") {
        reviewData.text =

`#INTRO\n${intro}\n\n#BODY\n${body}\n\n#IMPORTANT\n${important}\n\n#CONCLUSION\n
${conclusion}`;
        reviewData.author = author;
      }

       try {
        const docRef = await addDoc(collection(db, "reviews"), reviewData);
        console.log("Огляд додано:", reviewData);

        Swal.fire({
          icon: "success",
          title: "Огляд додано!",
          showConfirmButton: false,
          timer: 1000,
        });

        if (onAdded) {
          onAdded({ id: docRef.id, ...reviewData });
        }
        resetForm();
        onClose();
      } catch (error) {
        console.error("Помилка при додаванні огляду:", error);
        setError("Не вдалося додати огляд. Спробуйте ще раз.");
      }

        };

        return (
          <div  className="fixed  inset-0  bg-black/50  flex  items-center  justify-
center z-50">
            <div  className="bg-white  rounded-xl  shadow-lg  p-6  w-full  max-w-lg">
              <h3   className="text-xl   font-bold   text-zinc-900   mb-4">Додати
огляд</h3>

              {/* Перемикач */}
              <div  className="flex  gap-4  mb-6">
                <button
                  type="button"
                  onClick={() => setType("video")}
                  className={`px-4 py-2 rounded-lg border ${
                    type === "video"
                      ? "bg-amber-600 text-white"
                      : "bg-white text-zinc-700 hover:bg-zinc-100"
                  }`}
                >
                  Відеоогляд
                </button>
                <button
                  type="button"
                  onClick={() => setType("text")}
                  className={`px-4 py-2 rounded-lg border ${
                    type === "text"
                      ? "bg-amber-600 text-white"
                      : "bg-white text-zinc-700 hover:bg-zinc-100"
                  }`}
                >
```

```
                Текстовий
              </button>
            </div>

            {/* Форма */}
            <form onSubmit={handleSubmit} className="space-y-4">
              {/* Назва */}
              <div>
                <label className="block text-sm font-medium text-zinc-700">
                  Назва
                </label>
                <input
                  type="text"
                  value={title}
                  onChange={(e) => setTitle(e.target.value)}
                  className="mt-1 w-full border rounded-lg px-3 py-2 shadow-sm
focus:ring-2 focus:ring-amber-500 focus:outline-none"
                  placeholder="Введіть назву огляду"
                  required
                />
              </div>

              {/* Якщо відео */}
              {type === "video" && (
                <>
                  <div>
                    <label  className="block  text-sm  font-medium  text-zinc-
700">
                      Посилання
                    </label>
                    <input
                      type="url"
                      value={url}
                      onChange={(e) => setUrl(e.target.value)}
                      className="mt-1  w-full  border  rounded-lg  px-3  py-2
shadow-sm focus:ring-2 focus:ring-amber-500 focus:outline-none"
                      placeholder="https://youtube.com/..."
                      required
                    />
                  </div>

                  <div>
                    <label  className="block  text-sm  font-medium  text-zinc-
700">
                      Опис
                    </label>
                    <textarea
                      rows="3"
                      value={description}
                      onChange={(e) => setDescription(e.target.value)}
                      className="mt-1  w-full  border  rounded-lg  px-3  py-2
shadow-sm focus:ring-2 focus:ring-amber-500 focus:outline-none"
                      placeholder="Короткий опис..."
                      required
                    ></textarea>
                  </div>
                </>
              )}

              {/* Якщо текстовий */}
              {type === "text" && (
                <>
          <div>
            <label className="block text-sm font-medium text-zinc-700">
              Вступ
```

```
            </label>
            <textarea
              rows="2"
              value={intro}
              onChange={(e) => setIntro(e.target.value)}
              className="mt-1 w-full border rounded-lg px-3 py-2 shadow-sm
focus:ring-2 focus:ring-amber-500 focus:outline-none"
              placeholder="Короткий вступ..."
            ></textarea>
          </div>

          <div>
            <label className="block text-sm font-medium text-zinc-700">
              Основна частина
            </label>
            <textarea
              rows="4"
              value={body}
              onChange={(e) => setBody(e.target.value)}
              className="mt-1 w-full border rounded-lg px-3 py-2 shadow-sm
focus:ring-2 focus:ring-amber-500 focus:outline-none"
              placeholder="Основний текст..."
            ></textarea>
          </div>

          <div>
            <label className="block text-sm font-medium text-zinc-700">
              Важливі речі
            </label>
            <textarea
              rows="3"
              value={important}
              onChange={(e) => setImportant(e.target.value)}
              className="mt-1 w-full border rounded-lg px-3 py-2 shadow-sm
focus:ring-2 focus:ring-amber-500 focus:outline-none"
              placeholder="Що треба особливо підкреслити..."
            ></textarea>
          </div>

          <div>
            <label className="block text-sm font-medium text-zinc-700">
              Висновок
            </label>
            <textarea
              rows="3"
              value={conclusion}
              onChange={(e) => setConclusion(e.target.value)}
              className="mt-1 w-full border rounded-lg px-3 py-2 shadow-sm
focus:ring-2 focus:ring-amber-500 focus:outline-none"
              placeholder="Фінальний висновок..."
            ></textarea>
          </div>

                    <div>
                      <label  className="block  text-sm  font-medium  text-zinc-
700">
                        Джерело / Автор
                      </label>
                      <input
                        type="text"
                        value={author}
                        onChange={(e) => setAuthor(e.target.value)}
                        className="mt-1 w-full border rounded-lg px-3 py-2
shadow-sm focus:ring-2 focus:ring-amber-500 focus:outline-none"
                        placeholder="IGN, PC Gamer, Автор..."
```

```
                    />
                  </div>
                </>
              )}

              <p className="text-xs text-zinc-500">
                Дата публікації: {getCurrentDate()}
              </p>

              {/* Повідомлення про помилку */}
              {error && <p className="text-sm text-red-600">{error}</p>}

              {/* Кнопки */}
              <div className="flex justify-end gap-3 pt-4">
                <button
                  type="submit"
                  className="px-4  py-2  rounded-lg  bg-amber-600  text-white
hover:bg-amber-700"
                >
                  Додати
                </button>
                <button
                  type="button"
                  onClick={onClose}
                  className="px-4 py-2 rounded-lg border border-zinc-300 text-
zinc-700 bg-white hover:bg-red-500 hover:text-white transition"
                >
                  Скасувати
                </button>
              </div>
            </form>
          </div>
        </div>
      );
    }
```

AddTutorialModal.jsx

```
    import React, { useState } from "react";
    import { db } from "../../firebase"; // ⚡ поправ шлях під свій проект
    import { collection, addDoc } from "firebase/firestore";
    import Swal from "sweetalert2";

    export default function AddTutorialModal({ onClose, onAdded }) {
      const [category, setCategory] = useState("Поради та гайди");
      const [title, setTitle] = useState("");
      const [text, setText] = useState("");
      const [videoUrl, setVideoUrl] = useState(""); // нове поле
      const [question, setQuestion] = useState("");
      const [answer, setAnswer] = useState("");
      const [term, setTerm] = useState("");
      const [definition, setDefinition] = useState("");
      const [error, setError] = useState("");

      const getCurrentDate = () => {
        const now = new Date();
        return `${String(now.getDate()).padStart(2, "0")}.${String(
          now.getMonth() + 1
        ).padStart(2, "0")}`;
      };

      const resetForm = () => {
        setCategory("Поради та гайди");
```

```
    setTitle("");
    setText("");
    setVideoUrl("");
    setQuestion("");
    setAnswer("");
    setTerm("");
    setDefinition("");
    setError("");
  };

  const handleSubmit = async (e) => {
    e.preventDefault();

    let newItem = {
      category,
      date: getCurrentDate(),
    };

    if (category === "Поради та гайди") {
      if (!title.trim() || !text.trim()) {
        setError("Заповніть назву і текст!");
        return;
      }
      newItem = { ...newItem, title, text };
      if (videoUrl.trim()) {
        newItem.videoUrl = videoUrl;
      }
    } else if (category === "FAQ") {
      if (!question.trim() || !answer.trim()) {
        setError("Заповніть питання і відповідь!");
        return;
      }
      newItem = { ...newItem, question, answer };
    } else if (category === "Словник термінів") {
      if (!term.trim() || !definition.trim()) {
        setError("Заповніть термін і пояснення!");
        return;
      }
      newItem = { ...newItem, term, definition };
    }

    try {
      const docRef = await addDoc(collection(db, "tutorials"), newItem);
      console.log("Матеріал додано:", newItem);

      Swal.fire({
        icon: "success",
        title: "Матеріал додано!",
        showConfirmButton: false,
        timer: 1000,
      });

      if (onAdded) {
        onAdded({ id: docRef.id, ...newItem });
      }

      resetForm();
      onClose();
    } catch (error) {
      console.error("Помилка при додаванні матеріалу:", error);
      setError("Не вдалося додати матеріал. Спробуйте ще раз.");
    }
  };

  return (
```

```
          <div className="fixed inset-0 bg-black/50 flex items-center justify-
center z-50">
            <div className="bg-white rounded-xl shadow-lg p-6 w-full max-w-lg">
              <h3   className="text-xl   font-bold   text-zinc-900   mb-4">Додати
матеріал</h3>

              <form onSubmit={handleSubmit} className="space-y-4">
                {/* Категорія */}
                <div>
                  <label className="block text-sm font-medium text-zinc-700">
                    Категорія
                  </label>
                  <select
                    value={category}
                    onChange={(e) => setCategory(e.target.value)}
                    className="mt-1 w-full border rounded-lg px-3 py-2 shadow-sm
focus:ring-2 focus:ring-amber-500 focus:outline-none"
                  >
                    <option>Поради та гайди</option>
                    <option>FAQ</option>
                    <option>Словник термінів</option>
                  </select>
                </div>

                {/* Поля залежно від категорії */}
                {category === "Поради та гайди" && (
                  <>
                    <div>
                      <label  className="block  text-sm  font-medium  text-zinc-
700">
                        Назва
                      </label>
                      <input
                        type="text"
                        value={title}
                        onChange={(e) => setTitle(e.target.value)}
                        className="mt-1  w-full  border  rounded-lg  px-3  py-2
shadow-sm focus:ring-2 focus:ring-amber-500 focus:outline-none"
                        placeholder="Введіть назву матеріалу"
                        required
                      />
                    </div>
                    <div>
                      <label  className="block  text-sm  font-medium  text-zinc-
700">
                        Текст
                      </label>
                      <textarea
                        rows="3"
                        value={text}
                        onChange={(e) => setText(e.target.value)}
                        className="mt-1  w-full  border  rounded-lg  px-3  py-2
shadow-sm focus:ring-2 focus:ring-amber-500 focus:outline-none"
                        placeholder="Введіть текст..."
                        required
                      ></textarea>
                    </div>
                    <div>
                      <label  className="block  text-sm  font-medium  text-zinc-
700">
                        Посилання на відео (опціонально)
                      </label>
                      <input
                        type="url"
                        value={videoUrl}
```

```
                        onChange={(e) => setVideoUrl(e.target.value)}
                        className="mt-1 w-full border rounded-lg px-3 py-2
shadow-sm focus:ring-2 focus:ring-amber-500 focus:outline-none"
                        placeholder="https://youtube.com/..."
                      />
                    </div>
                  </>
                )}

                {category === "FAQ" && (
                  <>
                    <div>
                      <label className="block text-sm font-medium text-zinc-
700">
                        Питання
                      </label>
                      <input
                        type="text"
                        value={question}
                        onChange={(e) => setQuestion(e.target.value)}
                        className="mt-1 w-full border rounded-lg px-3 py-2
shadow-sm focus:ring-2 focus:ring-amber-500 focus:outline-none"
                        placeholder="Введіть питання"
                        required
                      />
                    </div>
                    <div>
                      <label className="block text-sm font-medium text-zinc-
700">
                        Відповідь
                      </label>
                      <textarea
                        rows="3"
                        value={answer}
                        onChange={(e) => setAnswer(e.target.value)}
                        className="mt-1 w-full border rounded-lg px-3 py-2
shadow-sm focus:ring-2 focus:ring-amber-500 focus:outline-none"
                        placeholder="Введіть відповідь"
                        required
                      ></textarea>
                    </div>
                  </>
                )}

                {category === "Словник термінів" && (
                  <>
                    <div>
                      <label className="block text-sm font-medium text-zinc-
700">
                        Термін
                      </label>
                      <input
                        type="text"
                        value={term}
                        onChange={(e) => setTerm(e.target.value)}
                        className="mt-1 w-full border rounded-lg px-3 py-2
shadow-sm focus:ring-2 focus:ring-amber-500 focus:outline-none"
                        placeholder="Введіть термін"
                        required
                      />
                    </div>
                    <div>
                      <label className="block text-sm font-medium text-zinc-
700">
                        Пояснення
```

```
                    </label>
                    <textarea
                      rows="3"
                      value={definition}
                      onChange={(e) => setDefinition(e.target.value)}
                      className="mt-1 w-full border rounded-lg px-3 py-2
shadow-sm focus:ring-2 focus:ring-amber-500 focus:outline-none"
                      placeholder="Введіть пояснення"
                      required
                    ></textarea>
                  </div>
                </>
              )}

              {/* Повідомлення про помилку */}
              {error && <p className="text-sm text-red-600">{error}</p>}

              {/* Кнопки */}
              <div className="flex justify-end gap-3 pt-4">
                <button
                  type="submit"
                  className="px-4 py-2 rounded-lg bg-amber-600 text-white
hover:bg-amber-700"
                >
                  Додати
                </button>
                <button
                  type="button"
                  onClick={onClose}
                  className="px-4 py-2 rounded-lg border border-zinc-300 text-
zinc-700 bg-white hover:bg-red-500 hover:text-white transition"
                >
                  Скасувати
                </button>
              </div>
            </form>
          </div>
        </div>
      );
    }
```

Home.jsx

```
    import React from "react";
    import { Link } from "react-router-dom";

    const BG_IMAGE = "https://www.certainaffinity.com/wp-
content/uploads/2024/02/wot-10.jpg";
    const PLAY_URL = "https://worldoftanks.eu/";

    export default function Home() {
      return (
        <main className="relative h-screen w-full overflow-hidden">
          {/* Фон-картинка */}
          <div
            className="absolute inset-0 -z-10 bg-center bg-cover"
            style={{ backgroundImage: `url(${BG_IMAGE})` }}
          >
            <div className="absolute inset-0 bg-black/60" />
          </div>

          {/* Контент — нижче екрана */}
```

```
            <div  className="relative  h-full  flex  flex-col  justify-end  items-
center text-center px-4 pb-24">
              <h1
                className="text-white   text-5xl   sm:text-6xl   font-extrabold
uppercase tracking-[0.3em]
                           drop-shadow-[3px_3px_4px_rgba(0,0,0,0.8)]"
              >
                World of Tanks
              </h1>

              <p  className="mt-6  max-w-2xl  text-lg  text-gray-200/90  drop-
shadow">
                Обирай: вирушай у бій прямо зараз або дізнайся більше про гру.
              </p>

              <div className="mt-10 flex flex-col sm:flex-row gap-4">
                <a
                  href={PLAY_URL}
                  target="_blank"
                  rel="noreferrer"
                  className="px-6 py-3 rounded-xl bg-amber-400 text-black font-
semibold shadow-md hover:bg-amber-300 transition"
                >
                  Грати зараз
                </a>

                <Link
                  to="/details"
                  className="px-6  py-3  rounded-xl  bg-white/10  text-white  font-
semibold ring-1 ring-white/30 shadow-md hover:bg-white/15 transition"
                >
                  Детальніше про гру
                </Link>
              </div>
            </div>
          </main>
        );
      }
```

Details.jsx

```
import React, { useEffect, useMemo, useState } from "react";
import { SECTIONS } from "../data/sections";
import { ROLES } from "../data/roles";
import { MODES } from "../data/modes";

import Overview from "../components/Details/Overview";
import Modes from "../components/Details/Modes";
import Roles from "../components/Details/Roles";
import Crew from "../components/Details/Crew";

export default function Details() {
  const [activeId, setActiveId] = useState(SECTIONS[0].id);
  const [selected, setSelected] = useState(ROLES[0]);
  const [tab, setTab] = useState("permanent");
  const [filter, setFilter] = useState("all");

  const filteredModes = MODES[tab].filter((mode) => {
    if (filter === "all") return true;
    if (filter === "x-only") return mode.level === "x-only";
    if (filter === "event") return mode.level === "event";
    return true;
  });
```

```
        useEffect(() => {
          const obs = new IntersectionObserver(
            (entries) => {
              const visible = entries
                .filter((e) => e.isIntersecting)
                .sort((a, b) => b.intersectionRatio - a.intersectionRatio)[0];
              if (visible?.target?.id) setActiveId(visible.target.id);
            },
            { rootMargin: "-20% 0px -60% 0px", threshold: [0.2, 0.4, 0.6, 0.8] }
          );

          SECTIONS.forEach((s) => {
            const el = document.getElementById(s.id);
            if (el) obs.observe(el);
          });
          return () => obs.disconnect();
        }, []);

        const linkClass = useMemo(
          () => (id) =>
            `block rounded-xl px-3 py-2 text-sm sm:text-base transition-colors
${
              activeId === id
                ? "bg-amber-100 text-amber-800"
                : "text-zinc-700 hover:bg-zinc-100"
            }`,
          [activeId]
        );

        return (
          <div className="min-h-[100svh] bg-gradient-to-b from-stone-50 to-zinc-
100 text-zinc-800">
            <div className="mx-auto max-w-6xl px-4 py-10 sm:py-14">
              <div        className="grid        grid-cols-1        md:grid-cols-
[240px_minmax(0,1fr)] gap-8">
                <aside className="md:sticky md:top-20 h-max">
                  <h4   className="mb-3   text-sm   font-semibold   text-zinc-500
uppercase tracking-wider">
                    Категорії
                  </h4>
                  <nav className="space-y-1">
                    {SECTIONS.map((s) => (
                      <a
                        key={s.id}
                        href={`#${s.id}`}
                        onClick={() => setActiveId(s.id)}
                        className={linkClass(s.id)}
                      >
                        {s.title}
                      </a>
                    ))}
                  </nav>
                </aside>
                <main className="scroll-smooth space-y-10">
                  <Overview />
                  <Modes
                    tab={tab}
                    setTab={setTab}
                    filter={filter}
                    setFilter={setFilter}
                    filteredModes={filteredModes}
                  />
                  <Roles selected={selected} setSelected={setSelected} />
                  <Crew />
```

```
            </main>
          </div>
        </div>
      </div>
    );
  }
```

Developers.jsx

```
  import React from "react";

  const MILESTONES = [
    { year: "1998", title: "Заснування Wargaming", text: "Старт як незалежного розробника стратегій." },
    { year: "2000", title: "DBA Online", text: "Перший комерційний проєкт студії." },
    { year: "2003", title: "Massive Assault", text: "Успішна стратегія RTT, серія сиквелів 2004–2006." },
    { year: "2008", title: "Операція «Багратіон»", text: "Нагорода «Найкраща стратегія» (KPI-2008)." },
    { year: "2009", title: "Order of War із Square Enix", text: "Партнерство та глобальний реліз." },
    { year: "2010", title: "World of Tanks", text: "Реліз, що сформував live-service парадигму." },
    { year: "2011", title: "«Перша Студія» (Kyiv)", text: "Приєднання: контент для WoT, повний цикл розробки." },
    { year: "2013–2014", title: "Консолі та мобайл", text: "WoT: Xbox 360 Edition, запуск WoT Blitz." },
    { year: "2017", title: "AI-ініціативи", text: "Боти для навчання, PvE, внутрішніх тестів." },
    { year: "2021", title: "Івенти «Мирний: Надія»", text: "Атмосферні PvE-події, креативні колаби." },
    { year: "2022", title: "Реорганізація бізнесу", text: "Вихід з РФ/РБ, фокус на інших регіонах." },
    { year: "2023", title: "WargamingUnited", text: "Благодійні ініціативи на підтримку України." },
    { year: "2025", title: "World of Tanks 2.0", text: "Поколіннєве оновлення систем і технологій." },
  ];

  const HorizontalTimeline = ({ items }) => {
    return (
      <div className="w-full overflow-x-auto py-6">
        <div className="relative flex items-start gap-8 min-w-max">
          <div className="absolute top-4 left-0 right-0 h-1 bg-gradient-to-r from-amber-300 via-amber-400 to-amber-500 rounded-full">
            <div className="absolute right-[-10px] top-1/2 -translate-y-1/2 w-0 h-0 border-l-[12px] border-l-amber-500 border-y-[8px] border-y-transparent" />
          </div>

          {items.map((m) => (
            <div
              key={m.year}
              className="basis-[200px] shrink-0 flex flex-col items-center text-center relative"
            >
              <div className="relative z-10 flex items-center justify-center w-10 h-10 rounded-full bg-white ring-2 ring-amber-500 shadow-md">
                <span className="text-amber-600">{m.icon}</span>
              </div>
```

```
                  <div className="mt-3 px-2 py-1 rounded-md border border-amber-
200 text-amber-700 text-sm font-semibold">
                    {m.year}
                  </div>
                  <div      className="mt-2      font-semibold      text-zinc-
900">{m.title}</div>
                  <p    className="mt-1    text-sm    text-zinc-700    leading-
relaxed">{m.text}</p>
                </div>
              ))}
            </div>
          </div>
        );
      };

      export default function Developers() {
        return (
          <div className="space-y-12">
            <header className="mb-2">
              <h2 className="text-3xl sm:text-4xl font-extrabold text-zinc-900
tracking-tight">
                Розробники
              </h2>
            </header>

            <section className="rounded-3xl p-6 ring-1 ring-zinc-200 bg-white
shadow-sm">
              <p className="text-lg leading-relaxed text-zinc-700">
                <span  className="text-amber-600  font-semibold">Wargaming</span>
—
                міжнародний розробник і видавець, найбільш відомий завдяки{" "}
                <span    className="text-amber-600    font-semibold">World    of
Tanks</span>.
                Компанія працює у live-service парадигмі: регулярні сезони,
івенти та
                розширювані системи прогресу підтримують довготривалу
залученість
                спільноти. Київська{" "}
                <span       className="text-amber-600       font-semibold">«Перша
Студія»</span> —
                одна з найстаріших геймдев-компаній України; за 20+ років
пройшла шлях
                від арт-аутсорсу до повного циклу виробництва і контент-лідства
в
                окремих напрямах.
              </p>

              <p className="mt-4 text-lg leading-relaxed text-zinc-700">
                Ставка на{" "}
                <span className="text-amber-600 font-semibold">
                  власні технології
                </span>{" "}
                і аналітику дає змогу синхронно оновлювати клієнт і сервери,
                прискорювати   пайплайни   і   запускати   події   без   втрати
стабільності.
                <span className="text-amber-600 font-semibold">
                  Баланс, анти-чіт і телеметрія
                </span>{" "}
                — базові стовпи, що забезпечують чесний матчмейкінг і якість
патчів.
                Регіональні команди підтримки й ком’юніті тісно працюють із
розробкою:
                локальні івенти, програми для креаторів та партнерські колаби —
                частина щорічного календаря.
              </p>
```

```
            <p className="mt-4 text-lg leading-relaxed text-zinc-700">
              З 2017 року активно розвивається{" "}
              <span className="text-amber-600 font-semibold">AI</span>: ігрові
боти
              допомагають новачкам та підтримують PvE-режими, а також
              використовуються всередині виробництва — для тестів карт,
балансування
              й навантажувальних прогонів. Атмосферні події на кшталт{" "}
              <span      className="text-amber-600      font-semibold">«Мирний:
Надія»</span>{" "}
              підкреслюють фокус на наратив та різноманіття досвідів.
            </p>
          </section>
          <section  className="rounded-3xl  p-6  ring-1  ring-zinc-200  bg-white
shadow-sm">
            <h3 className="text-2xl font-bold text-zinc-900">
              Історія студії — ключові етапи
            </h3>
            <p className="mt-2 text-zinc-700">
              Віхи розвитку: від заснування і появи «Першої Студії» до AI-
підходу,
              PvE-подій і релізу{" "}
              <span className="text-amber-600 font-semibold">
                World of Tanks 2.0
              </span>
              .
            </p>
            <HorizontalTimeline items={MILESTONES} />
          </section>
        </div>
      );
    }
```

Requirements.jsx

```
    import React from "react";

    export default function Requirements() {
      return (
        <>
          <h2    className="text-3xl    sm:text-4xl    font-bold    text-zinc-
900">Системні вимоги</h2>
          <p className="mt-4 text-zinc-700 leading-relaxed">
            Нижче  наведено  мінімальні  та  рекомендовані  конфігурації  для
комфортної гри.
            Параметри вказані окремо для стаціонарних ПК і ноутбуків.
          </p>

          {/* Карти з вимогами */}
          <div className="mt-6 grid sm:grid-cols-2 gap-6">
            {/* Мінімальні */}
            <div  className="rounded-2xl  border  border-zinc-200  bg-white  p-5
shadow-sm">
              <div className="flex items-center justify-between gap-3">
                <h3      className="font-semibold      text-zinc-900      text-
xl">Мінімальні</h3>
                <span  className="inline-flex  items-center  rounded-full  bg-
amber-100 text-amber-800 px-3 py-1 text-xs font-medium">
                  1280×768, ~30 FPS
                </span>
              </div>
```

```
                {/* ПК */}
                <div className="mt-4">
                  <h4 className="text-sm font-semibold text-zinc-600 uppercase
tracking-wide">ПК</h4>
                  <div className="mt-2 overflow-x-auto">
                    <table className="w-full text-sm">
                      <tbody className="[&_tr]:border-b [&_tr]:border-zinc-100">
                        <tr>
                          <th className="text-left py-2 pr-3 text-zinc-500 font-
medium">ОС</th>
                          <td className="py-2">64-бітні Windows 7 / 8 / 8.1 / 10
/ 11</td>
                        </tr>
                        <tr>
                          <th className="text-left py-2 pr-3 text-zinc-500 font-
medium">CPU</th>
                          <td   className="py-2">Intel   Core   i5-3330   <span
className="text-zinc-400">або</span> AMD FX-6300</td>
                        </tr>
                        <tr>
                          <th className="text-left py-2 pr-3 text-zinc-500 font-
medium">GPU</th>
                          <td className="py-2">
                            NVIDIA GeForce GTX 460 <span className="text-zinc-
400">або</span> AMD Radeon HD 8770,<br />
                            роздільна здатність 1280×768
                          </td>
                        </tr>
                        <tr>
                          <th className="text-left py-2 pr-3 text-zinc-500 font-
medium">RAM</th>
                          <td className="py-2">4 ГБ</td>
                        </tr>
                        <tr>
                          <th className="text-left py-2 pr-3 text-zinc-500 font-
medium">Диск</th>
                          <td className="py-2">70 ГБ вільного місця</td>
                        </tr>
                        <tr>
                          <th className="text-left py-2 pr-3 text-zinc-500 font-
medium">Інтернет</th>
                          <td className="py-2">≥ 256 кбіт/с</td>
                        </tr>
                      </tbody>
                    </table>
                  </div>
                </div>

                {/* Ноутбук */}
                <div className="mt-6">
                  <h4 className="text-sm font-semibold text-zinc-600 uppercase
tracking-wide">Ноутбук</h4>
                  <div className="mt-2 overflow-x-auto">
                    <table className="w-full text-sm">
                      <tbody className="[&_tr]:border-b [&_tr]:border-zinc-100">
                        <tr>
                          <th className="text-left py-2 pr-3 text-zinc-500 font-
medium">ОС</th>
                          <td className="py-2">64-бітні Windows 7 / 8 / 8.1 / 10
/ 11</td>
                        </tr>
                        <tr>
                          <th className="text-left py-2 pr-3 text-zinc-500 font-
medium">CPU</th>
                          <td className="py-2">Intel Core i5-3230M</td>
```

```
                        </tr>
                        <tr>
                          <th className="text-left py-2 pr-3 text-zinc-500 font-
medium">GPU</th>
                          <td className="py-2">
                            NVIDIA GeForce GT 740M, роздільна здатність 1280×768
                          </td>
                        </tr>
                        <tr>
                          <th className="text-left py-2 pr-3 text-zinc-500 font-
medium">RAM</th>
                          <td className="py-2">4 ГБ</td>
                        </tr>
                        <tr>
                          <th className="text-left py-2 pr-3 text-zinc-500 font-
medium">Диск</th>
                          <td className="py-2">70 ГБ вільного місця</td>
                        </tr>
                        <tr>
                          <th className="text-left py-2 pr-3 text-zinc-500 font-
medium">Інтернет</th>
                          <td className="py-2">≥ 256 кбіт/с</td>
                        </tr>
                      </tbody>
                    </table>
                  </div>
                </div>
              </div>

              {/* Рекомендовані */}
              <div  className="rounded-2xl  border  border-zinc-200  bg-white  p-5
shadow-sm">
                <div className="flex items-center justify-between gap-3">
                  <h3      className="font-semibold      text-zinc-900      text-
xl">Рекомендовані</h3>
                  <span  className="inline-flex  items-center  rounded-full  bg-
emerald-100 text-emerald-800 px-3 py-1 text-xs font-medium">
                    1080p, стабільні FPS
                  </span>
                </div>

                {/* (єдині вимоги; ноутбуки зазвичай наближені по класу) */}
                <div className="mt-4">
                  <div className="mt-2 overflow-x-auto">
                    <table className="w-full text-sm">
                      <tbody className="[&_tr]:border-b [&_tr]:border-zinc-100">
                        <tr>
                          <th className="text-left py-2 pr-3 text-zinc-500 font-
medium">ОС</th>
                          <td className="py-2">64-бітні Windows 10 / 11</td>
                        </tr>
                        <tr>
                          <th className="text-left py-2 pr-3 text-zinc-500 font-
medium">CPU</th>
                          <td className="py-2">
                            Intel  Core  i3-9100F  <span  className="text-zinc-
400">або</span> AMD Ryzen 3 3100
                          </td>
                        </tr>
                        <tr>
                          <th className="text-left py-2 pr-3 text-zinc-500 font-
medium">GPU</th>
                          <td className="py-2">
                            NVIDIA GeForce GTX 1650 <span className="text-zinc-
400">або</span> AMD Radeon RX 580
```

```
                        </td>
                      </tr>
                      <tr>
                        <th className="text-left py-2 pr-3 text-zinc-500 font-
medium">RAM</th>
                        <td className="py-2">16 ГБ</td>
                      </tr>
                      <tr>
                        <th className="text-left py-2 pr-3 text-zinc-500 font-
medium">Диск</th>
                        <td  className="py-2">70  ГБ  вільного  місця  (бажано
SSD)</td>
                      </tr>
                      <tr>
                        <th className="text-left py-2 pr-3 text-zinc-500 font-
medium">Інтернет</th>
                        <td   className="py-2">1+   Мбіт/с   (для   голосового
зв’язку)</td>
                      </tr>
                    </tbody>
                  </table>
                </div>
              </div>

              {/* Драйвери / Примітки */}
              <div className="mt-6 rounded-xl border border-zinc-100 bg-zinc-
50 p-4">
                <h4  className="text-sm  font-semibold  text-zinc-700">Драйвери
та компоненти</h4>
                <ul  className="mt-2  text-sm  text-zinc-700  list-disc  pl-5
space-y-1">
                  <li>Оновлені      драйвери      <strong>AMD</strong>      /
<strong>NVIDIA</strong> / <strong>Intel</strong>.</li>
                  <li>Актуальна            версія            <strong>Microsoft
DirectX</strong>.</li>
                  <li>Для  ноутбуків  —  свіжі  драйвери  чипсету/графіки  від
виробника.</li>
                </ul>
              </div>
            </div>
          </div>

          {/* Дрібні примітки */}
          <p className="mt-6 text-xs text-zinc-500">
            Примітка: продуктивність залежить від налаштувань графіки, типу
носія (SSD/HDD),
            фонових програм та стану драйверів.
          </p>
        </>
      );
    }
```

Maps.jsx

```
import React, { useEffect, useState } from "react";
import { Maps } from "../data/maps.js"; // масив карт

// переклад назв категорій
const typeLabels = {
  summer: "Літні карти",
  winter: "Зимові карти",
  desert: "Піщані карти",
  special: "Особливі карти",
```

```
};

export default function MapsPage() {
  const [selectedType, setSelectedType] = useState("");
  const [activeSlug, setActiveSlug] = useState("");
  const [openStats, setOpenStats] = useState(null); // slug карти для
статистики

  // групування карт по типах
  const groupedMaps = Maps.reduce((acc, map) => {
    if (!acc[map.type]) acc[map.type] = [];
    acc[map.type].push(map);
    return acc;
  }, {});

  const handleCardClick = (slug) => {
    const el = document.getElementById(slug);
    if (el) {
      el.scrollIntoView({ behavior: "smooth", block: "start" });
      setActiveSlug(slug);
    }
  };

  // IntersectionObserver для підсвітки активної карти при скролі
  useEffect(() => {
    const obs = new IntersectionObserver(
      (entries) => {
        const visible = entries.find((e) => e.isIntersecting);
        if (visible?.target?.id) {
          setActiveSlug(visible.target.id);
        }
      },
      { rootMargin: "-40% 0px -40% 0px", threshold: 0.3 }
    );

    Maps.forEach((map) => {
      const el = document.getElementById(map.slug);
      if (el) obs.observe(el);
    });

    return () => obs.disconnect();
  }, []);

  return (
    <div className="min-h-[100svh] bg-gradient-to-b from-white to-zinc-100
text-zinc-800">
      <div className="mx-auto max-w-7xl px-4 py-10 sm:py-14">
        <div       className="grid       grid-cols-1       md:grid-cols-
[260px_minmax(0,1fr)] gap-8">
          {/* Sidebar */}
          <aside className="md:sticky md:top-20 h-max space-y-5">
            <h4  className="text-sm  font-semibold  text-zinc-500  uppercase
tracking-wider">
              Категорії
            </h4>
            <select
              value={selectedType}
              onChange={(e) => setSelectedType(e.target.value)}
              className="w-full rounded-lg border border-zinc-300 px-3 py-
2 text-sm bg-white shadow-sm"
            >
              <option value="">Всі категорії</option>
              {Object.keys(groupedMaps).map((type) => (
                <option key={type} value={type}>
                  {typeLabels[type] || type}
```

```
                    </option>
                  ))}
                </select>

                {/* Список карт обраної категорії */}
      {selectedType && groupedMaps[selectedType]?.length > 0 && (
        <div className="bg-white border rounded-lg shadow p-3">
          <h5 className="text-sm font-bold text-zinc-700 mb-2">
            {typeLabels[selectedType]}
          </h5>
          <div className="space-y-2 max-h-[300px] overflow-y-auto pr-1">
            {groupedMaps[selectedType].map((map) => (
              <button
                key={map.slug}
                onClick={() => handleCardClick(map.slug)}
                className={`block w-full text-left px-2 py-1 text-sm rounded
transition ${
                  activeSlug === map.slug
                    ? "bg-orange-100 text-orange-700 font-semibold"
                    : "hover:bg-zinc-100"
                }`}
              >
                {map.name}
              </button>
            ))}
          </div>
        </div>
      )}
              </aside>

              {/* Main */}
              <main className="scroll-smooth space-y-12">
                {Object.entries(groupedMaps).map(([type, maps]) =>
                  maps.length > 0 ? (
                    <section key={type} id={type} className="scroll-mt-24">
                      <h2 className="text-3xl sm:text-4xl font-bold text-
orange-600 mb-8">
                        {typeLabels[type] || type}
                      </h2>

                      <div className="space-y-10 ">
                        {maps.map((map) => (
                          <div
                            id={map.slug}
                            key={map.slug}
                            className={`bg-white rounded-2xl shadow border
transition p-6 ${
                              activeSlug === map.slug
                                ? "border-2 border-orange-500 shadow-lg"
                                : "border-zinc-200 hover:shadow-md"
                            }`}
                          >
                            {/* Назва */}
                            <h3 className="text-2xl font-extrabold text-zinc-
900 mb-6 text-center">
                              {map.name}
                            </h3>

                            {/* Зображення + мінікарта */}
                            <div className="grid md:grid-cols-2 gap-6 mb-6">
                              <div className="space-y-2">
                                <img
                                  src={map.img}
                                  alt={map.name}
```

```
                                    className="w-full h-56 object-cover rounded-
lg border"
                                  />
                                  <p  className="text-center  text-sm  text-zinc-
600">
                                    Зображення карти
                                  </p>
                                </div>
                                <div className="space-y-2">
                                  <img
                                    src={map.miniMap}
                                    alt={`${map.name} minimap`}
                                    className="w-full    h-56    object-contain
rounded-lg border bg-zinc-50"
                                  />
                                  <p  className="text-center  text-sm  text-zinc-
600">
                                    Мінікарта
                                  </p>
                                </div>
                              </div>

                              {/* Опис */}
                              {map.description && (
                                <p  className="text-base  text-zinc-700  leading-
relaxed mb-4">
                                  {map.description}
                                </p>
                              )}

                              {/* Кнопка показати статистику */}
                              <div className="text-center">
                                <a
                                  href={`https://tomato.gg/maps/EU/${map.slug}`}
                                  target="_blank"
                                  rel="noopener noreferrer"
                                  className="inline-flex items-center gap-2 px-4
py-2 bg-orange-500 hover:bg-orange-600 text-white text-sm font-semibold rounded-
lg shadow transition"
                                >
                                  Показати статистику
                                </a>

                                <p  className="mt-2  text-xs  text-zinc-500  text-
center">
                                  Статистика для цієї карти може бути відсутня
на
                                  сайті tomato.gg
                                </p>
                              </div>
                            </div>
                          ))}
                        </div>
                      </section>
                    ) : null
                  )}
                </main>
              </div>
            </div>
          </div>
        );
      }
```

Economy.jsx

```
import React, { useEffect, useMemo, useState } from "react";
import CurrenciesSection from "../components/Economy/CurrenciesSection";
import PremiumSection from "../components/Economy/PremiumSection";

const SECTIONS = [
  { id: "currencies", title: "Кредити, золото, бони" },
  { id: "premium", title: "Преміум акаунт" },
];

export default function Economy() {
  const [activeId, setActiveId] = useState(SECTIONS[0].id);

  useEffect(() => {
    const obs = new IntersectionObserver(
      (entries) => {
        const visible = entries
          .filter((e) => e.isIntersecting)
          .sort((a, b) => b.intersectionRatio - a.intersectionRatio)[0];
        if (visible?.target?.id) setActiveId(visible.target.id);
      },
      { rootMargin: "-20% 0px -60% 0px", threshold: [0.2, 0.4, 0.6, 0.8] }
    );

    SECTIONS.forEach((s) => {
      const el = document.getElementById(s.id);
      if (el) obs.observe(el);
    });
    return () => obs.disconnect();
  }, []);

  const linkClass = useMemo(
    () => (id) =>
      `block rounded-xl px-3 py-2 text-sm sm:text-base transition-colors
${
        activeId === id
          ? "bg-yellow-100 text-yellow-800"
          : "text-zinc-700 hover:bg-zinc-100"
      }`,
    [activeId]
  );

  return (
    <div className="min-h-[100svh] bg-gradient-to-b from-white to-zinc-100
text-zinc-800">
      <div className="mx-auto max-w-6xl px-4 py-10 sm:py-14">
        <div       className="grid       grid-cols-1       md:grid-cols-
[240px_minmax(0,1fr)] gap-8">
          {/* Sidebar */}
          <aside className="md:sticky md:top-20 h-max">
            <h4   className="mb-3   text-sm   font-semibold   text-zinc-500
uppercase tracking-wider">
              Категорії
            </h4>
            <nav className="space-y-1">
              {SECTIONS.map((s) => (
                <a              key={s.id}              href={`#${s.id}`}
className={linkClass(s.id)}>
                  {s.title}
                </a>
              ))}
            </nav>
          </aside>

          {/* Content */}
```

```
            <main className="scroll-smooth space-y-10">
              <CurrenciesSection />
              <PremiumSection />
            </main>
          </div>
        </div>
      </div>
    );
  }
```

Vehicles.jsx

```
import React, { useState, useEffect, useMemo } from "react";

import { China } from "../data/tankData/China";
import { USSR } from "../data/tankData/USSR";
import { USA } from "../data/tankData/USA";
import { France } from "../data/tankData/France";
import { Germany } from "../data/tankData/Germany";
import { Italy } from "../data/tankData/Italy";
import { Japan } from "../data/tankData/Japan";
import { Poland } from "../data/tankData/Poland";
import { Sweden } from "../data/tankData/Sweden";
import { UK } from "../data/tankData/UK";
import { Czechoslovakia } from "../data/tankData/Czechoslovakia";
import FiltersSidebar from "../components/FiltersBar";
import { typeIcons, premiumBadge } from "../data/flags";
import { useNavigate } from "react-router-dom";
// Стрілочки для сортування
const ArrowUp = () => <span className="inline-block ml-1">↑</span>;
const ArrowDown = () => <span className="inline-block ml-1">↓</span>;

// ☞ Статичний масив усіх танків
const allTanks = [
  ...China,
  ...USSR,
  ...USA,
  ...France,
  ...Germany,
  ...Italy,
  ...Japan,
  ...Poland,
  ...Sweden,
  ...UK,
  ...Czechoslovakia,
];

export default function Vehicles() {
  const [sortConfig, setSortConfig] = useState({ key: null, direction:
null });
const navigate = useNavigate();
  // state для фільтрів
const savedFilters = JSON.parse(localStorage.getItem("tankFilters") ||
"{}");

const [selectedNations, setSelectedNations] =
useState(savedFilters.selectedNations || []);
const [selectedTypes, setSelectedTypes] =
useState(savedFilters.selectedTypes || []);
const [selectedTier, setSelectedTier] = useState(savedFilters.selectedTier
|| null);
const [premiumOnly, setPremiumOnly] = useState(savedFilters.premiumOnly ||
false);
```

```
      const [searchQuery, setSearchQuery] = useState(savedFilters.searchQuery ||
"");

        // ▾ Збереження фільтрів у localStorage при зміні
      useEffect(() => {
        localStorage.setItem(
          "tankFilters",
          JSON.stringify({
            selectedNations,
            selectedTypes,
            selectedTier,
            premiumOnly,
            searchQuery
          })
        );
      },    [selectedNations,    selectedTypes,    selectedTier,    premiumOnly,
searchQuery]);

      // ▾ Відновлення при першому завантаженні
      useEffect(() => {
        const saved = localStorage.getItem("tankFilters");
        if (saved) {
          const parsed = JSON.parse(saved);
          setSelectedNations(parsed.selectedNations || []);
          setSelectedTypes(parsed.selectedTypes || []);
          setSelectedTier(parsed.selectedTier || null);
          setPremiumOnly(parsed.premiumOnly || false);
          setSearchQuery(parsed.searchQuery || "");
        }
      }, []);


        // 🔎 Фільтрація
        const filteredTanks = useMemo(() => {
          return allTanks.filter((t) => {
            const matchNation =
              selectedNations.length             ===             0             ||
selectedNations.includes(t.nation);
            const matchType =
              selectedTypes.length === 0 ||
              selectedTypes.includes(t.type.replace("-prem", ""));
            const matchTier = !selectedTier || t.tier === selectedTier;
            const matchPremium =
              !premiumOnly ||
              t.premium === 1 ||
              (t.type && t.type.includes("prem"));

            // Пошук по назві або характеристикам
            const query = searchQuery.toLowerCase();
            const matchSearch =
              query === "" ||
              t.name.toLowerCase().includes(query) ||
              (t.hp && t.hp.toString().includes(query)) ||
              (t.dmg && t.dmg.toString().includes(query)) ||
              (t.speed && t.speed.toString().includes(query));

            return  matchNation  &&  matchType  &&  matchTier  &&  matchPremium  &&
matchSearch;
          });
        },    [selectedNations,    selectedTypes,    selectedTier,    premiumOnly,
searchQuery]);

        // ▾ Сортування після фільтрації
      const sortedTanks = useMemo(() => {
```

```
        if (!sortConfig.key || !sortConfig.direction) return filteredTanks;

        return [...filteredTanks].sort((a, b) => {
          let aVal = a[sortConfig.key];
          let bVal = b[sortConfig.key];

          // якщо числові поля → кастимо у числа
          if (["hp", "dmg", "dpm", "speed", "tier"].includes(sortConfig.key)) {
            aVal = parseInt(aVal.toString().replace(/\s/g, ""), 10) || 0;
            bVal = parseInt(bVal.toString().replace(/\s/g, ""), 10) || 0;
          } else {
            // для рядків → нормалізуємо до lowerCase
            aVal = aVal?.toString().toLowerCase();
            bVal = bVal?.toString().toLowerCase();
          }

          if (aVal < bVal) return sortConfig.direction === "asc" ? -1 : 1;
          if (aVal > bVal) return sortConfig.direction === "asc" ? 1 : -1;
          return 0;
        });
      }, [filteredTanks, sortConfig]);

        // 📌 Логіка перемикання сортування
        const requestSort = (key) => {
          let direction = "asc";
          if (sortConfig.key === key && sortConfig.direction === "asc") {
            direction = "desc";
          } else if (sortConfig.key === key && sortConfig.direction === "desc")
{
            direction = null;
          }
          setSortConfig({ key, direction });
        };

        // 📝 Відмалювання заголовку таблиці
        const renderHeader = (label, key) => {
          const isActive = sortConfig.key === key;
          const direction = sortConfig.direction;

          return (
            <th
              className={`px-6  py-3  text-center  cursor-pointer  select-none
transition ${
                isActive ? "text-orange-500 font-bold" : "hover:text-orange-400"
              }`}
              onClick={() => requestSort(key)}
            >
              <span className="inline-flex items-center justify-center">
                {label}
                {isActive &&
                  (direction === "asc" ? <ArrowUp /> : direction === "desc" ?
<ArrowDown /> : null)}
              </span>
            </th>
          );
        };

        return (
          <div className="p-6">
            <h3 className="text-2xl font-bold text-zinc-800 uppercase mb-6">
              Танки
            </h3>

            {/* 🔎 Поле пошуку */}
```

```
            <div className="mb-4">
              <input
                type="text"
                placeholder="Пошук за назвою, ХП, уроном або швидкістю..."
                value={searchQuery}
                onChange={(e) => setSearchQuery(e.target.value)}
                className="w-full px-4 py-2 border rounded-lg shadow-sm
focus:outline-none focus:ring-2 focus:ring-orange-400"
              />
            </div>

            <div className="grid grid-cols-1 md:grid-cols-[280px_minmax(0,1fr)]
gap-6">
              {/* 🔽 Бокове меню фільтрів */}
              <FiltersSidebar
                tanks={allTanks}
                filteredTanks={filteredTanks}
                selectedNations={selectedNations}
                setSelectedNations={setSelectedNations}
                selectedTypes={selectedTypes}
                setSelectedTypes={setSelectedTypes}
                selectedTier={selectedTier}
                setSelectedTier={setSelectedTier}
                premiumOnly={premiumOnly}
                setPremiumOnly={setPremiumOnly}
              />

              {/* 🔽 Таблиця танків */}
              <div className="overflow-x-auto">
                <table className="min-w-full bg-white shadow-lg rounded-lg
overflow-hidden text-base">
                  <thead className="bg-zinc-100 text-zinc-800">
                    <tr>
                      {renderHeader("Країна", "nation")}
                      {renderHeader("Тип", "type")}
                      {renderHeader("Ранг", "tier")}
                      {renderHeader("Назва", "name")}
                      {renderHeader("ХП", "hp")}
                      {renderHeader("Урон", "dmg")}
                      {renderHeader("Урон/хв", "dpm")}
                      {renderHeader("Швидкість", "speed")}
                    </tr>
                  </thead>
                  <tbody>
                    {sortedTanks.map((tank, i) => {
                      const isPremium =
                        tank.premium === 1 || (tank.type &&
tank.type.includes("prem"));

                      return (
                        <tr
                          key={i}
                           onClick={() =>
navigate(`/vehicle/${encodeURIComponent(tank.name)}`)}
                          className="hover:bg-zinc-100 hover:shadow-md
transition-all text-center"
                        >
                          {/* Країна */}
                          <td className="px-4 py-2 min-w-[100px]">
                            <img
                              src={tank.flag}
                              alt="flag"
                              className="w-12 h-8 object-contain mx-auto
rounded"
```

```
                        />
                      </td>

                      {/* Тип */}
                      <td className="px-4 py-2 min-w-[100px]">
                        <div  className="relative  inline-flex  items-center
justify-center">
                          <img
                            src={typeIcons[tank.type]}
                            alt={tank.type}
                            className="w-7 h-7"
                          />
                          {isPremium && (
                            <img
                              src={premiumBadge}
                              alt="premium"
                              className="absolute w-5 h-5 opacity-90"
                            />
                          )}
                        </div>
                      </td>

                      {/* Ранг */}
                      <td
                        className={`px-4 py-2 font-semibold min-w-[80px] ${
                          isPremium ? "text-yellow-600" : "text-zinc-800"
                        }`}
                      >
                        {tank.tier}
                      </td>

                      {/* Назва */}
                      <td className="px-4 py-2 min-w-[160px]">
                        <div  className="flex  items-center  gap-2  justify-
center">
                          <img
                            src={tank.icon}
                            alt="tank-icon"
                            className="w-12 h-8 object-contain"
                          />
                          <span
                            className={`font-bold ${
                              isPremium  ?  "text-yellow-600"  :  "text-zinc-
700"

                            }`}
                          >
                            {tank.name}
                          </span>
                        </div>
                      </td>

                      {/* Характеристики */}
                      <td className="px-4 py-2 min-w-[100px]">{tank.hp}</td>
                      <td         className="px-4         py-2         min-w-
[100px]">{tank.dmg}</td>
                      <td         className="px-4         py-2         min-w-
[110px]">{tank.dpm}</td>
                      <td         className="px-4         py-2         min-w-
[110px]">{tank.speed}</td>
                    </tr>
                  );
                })}
              </tbody>
            </table>
          </div>
```

```
      </div>
    </div>
  );
}
```

VehicleDetails.jsx

```
import React, { useState } from "react";
import { useParams, useNavigate } from "react-router-dom";
import { China } from "../data/tankData/China";
import { USSR } from "../data/tankData/USSR";
import { USA } from "../data/tankData/USA";
import { France } from "../data/tankData/France";
import { Germany } from "../data/tankData/Germany";
import { Italy } from "../data/tankData/Italy";
import { Japan } from "../data/tankData/Japan";
import { Poland } from "../data/tankData/Poland";
import { Sweden } from "../data/tankData/Sweden";
import { UK } from "../data/tankData/UK";
import { Czechoslovakia } from "../data/tankData/Czechoslovakia";

import { typeIcons, premiumBadge } from "../data/flags";
import {
  HeartPulse,
  Sword,
  Zap,
  Gauge,
  Eye,
  Weight,
  Battery,
  RotateCcw,
  Compass,
  Crosshair,
  Timer,
  Rotate3D,
} from "lucide-react";

const allTanks = [
  ...China, ...USSR, ...USA, ...France, ...Germany,
  ...Italy, ...Japan, ...Poland, ...Sweden, ...UK, ...Czechoslovakia,
];

const statIcons = {
  hp: HeartPulse,
  dmg: Sword,
  dpm: Zap,
  speed: Gauge,
  viewRange: Eye,
  weight: Weight,
  specificPower: Battery,
  traverseSpeed: RotateCcw,
  turretTraverse: Compass,
  acr: Crosshair,
  aim: Timer,
};

export default function VehicleDetails() {
  const { name } = useParams();
  const navigate = useNavigate();
  const [show3D, setShow3D] = useState(false);

  const tank = allTanks.find((t) => t.name === decodeURIComponent(name));
```

```
        if (!tank) {
          return (
            <div className="p-6 text-center">
              <h2 className="text-2xl font-bold">Танк не знайдено</h2>
              <button
                onClick={() => navigate("/vehicles")}
                className="mt-4 px-4 py-2 bg-orange-500 text-white rounded-lg
shadow hover:bg-orange-600"
              >
                Назад
              </button>
            </div>
          );
        }

        const isPremium = tank.premium === 1 || (tank.type &&
tank.type.includes("prem"));

        // генеруємо id танка для tanks.gg
      const tankSlug = tank.name
        .toLowerCase()
        .replace(/[().]/g, "")
        .replace(/\s+/g, "-")
        .replace(/-+/g, "-")
        .replace(/^-|-$/g, "");
        const modelUrl = `https://tanks.gg/tank/${tankSlug}/model?vm=visual`;

        return (
          <div className="p-6 max-w-5xl mx-auto">
            {/* Назад */}
            <button
              onClick={() => navigate("/vehicles")}
              className="mb-6 px-4 py-2 bg-zinc-200 text-zinc-700 rounded-lg
hover:bg-orange-100 transition"
            >
              ← Назад
            </button>

            {/* Заголовок */}
            <div className="flex flex-col items-center mb-8">
              <img
                src={tank.img}
                alt={tank.name}
                className="w-full max-w-2xl h-auto object-contain rounded-lg
shadow-lg mb-6"
              />
              <h2 className="text-4xl font-bold text-zinc-800">{tank.name}</h2>
              <div className="flex items-center gap-6 mt-4">
                <img src={tank.flag} alt="flag" className="w-14 h-10 rounded
shadow" />
                <div className="relative w-14 h-14 flex items-center justify-
center">
                  <img src={typeIcons[tank.type]} alt={tank.type} className="w-
14 h-14" />
                  {isPremium && (
                    <img
                      src={premiumBadge}
                      alt="premium"
                      className="absolute inset-0 w-8 h-8 m-auto opacity-90"
                    />
                  )}
                </div>
                <span className="text-xl font-semibold bg-orange-100 px-3 py-1
rounded-lg">
                  Ранг: {tank.tier}
```

```
              </span>
            </div>

            {/* Кнопка 3D */}
            <button
              onClick={() => setShow3D(!show3D)}
              className="mt-6 flex items-center gap-2 px-5 py-2 bg-orange-500
text-white font-semibold rounded-lg shadow hover:bg-orange-600 transition"
            >
              <Rotate3D className="w-5 h-5" />
              {show3D ? "Закрити 3D модель" : "Показати 3D модель"}
            </button>
            <p className="mt-2 text-xs text-zinc-500 text-center">
       3D модель для цього танка  може бути відсутня на сайті tanks.gg
      </p>
          </div>

          {/* 3D модель */}
          {show3D && (
            <div className="mb-10">
              <iframe
                src={modelUrl}
                title={`${tank.name} 3D model`}
                className="w-full h-[600px] rounded-lg shadow-lg border"
              ></iframe>
            </div>
          )}

          {/* Характеристики */}
          <div  className="grid  grid-cols-2  md:grid-cols-3  gap-5  bg-white
shadow rounded-2xl p-6">
            <Info Icon={statIcons.hp} label="ХП" value={tank.hp} />
            <Info Icon={statIcons.dmg} label="Урон" value={tank.dmg} />
            <Info Icon={statIcons.dpm} label="Урон/хв" value={tank.dpm} />
            <Info  Icon={statIcons.speed}  label="Швидкість"  value={tank.speed}
/>
            <Info           Icon={statIcons.viewRange}           label="Огляд"
value={tank.viewRange} />
            <Info  Icon={statIcons.weight}  label="Маса"  value={`${tank.weight}
т`} />
            <Info   Icon={statIcons.specificPower}   label="Питома   потужність"
value={tank.specificPower} />
            <Info  Icon={statIcons.traverseSpeed}  label="Швидкість  повороту"
value={tank.traverseSpeed} />
            <Info    Icon={statIcons.turretTraverse}    label="Поворот    башти"
value={tank.turretTraverse} />
            <Info Icon={statIcons.acr} label="Точність" value={tank.acr} />
            <Info  Icon={statIcons.aim}  label="Час  зведення"  value={tank.aim}
/>
          </div>
        </div>
      );
    }

    function Info({ Icon, label, value }) {
      return (
        <div  className="flex  flex-col  items-center  bg-zinc-50  p-4  rounded-lg
border hover:shadow-md transition">
          <Icon className="w-6 h-6 text-orange-500 mb-1" />
          <span className="text-sm text-zinc-500">{label}</span>
          <span className="text-lg font-bold text-zinc-800">{value}</span>
        </div>
      );
    }
```

Reviews.jsx

```
import React, { useEffect, useState } from "react";
import { db } from "../firebase";
import { collection, getDocs, deleteDoc, doc } from "firebase/firestore";
import AddReviewModal from "../components/Modals/AddReviewModal";
import { Link } from "react-router-dom";
import Swal from "sweetalert2";
export default function Reviews() {
  const [isModalOpen, setIsModalOpen] = useState(false);
  const [reviews, setReviews] = useState([]);
  const isAdmin = localStorage.getItem("isAdmin") === "true";
  const [searchTerm, setSearchTerm] = useState("");
  const [filterType, setFilterType] = useState("all");
  const [sortDate, setSortDate] = useState("desc");
  useEffect(() => {
    const fetchReviews = async () => {
      const querySnapshot = await getDocs(collection(db, "reviews"));
      const data = querySnapshot.docs.map((doc) => ({
        id: doc.id,
        ...doc.data(),
      }));
      setReviews(data);
    };
    fetchReviews();
  }, []);
  const filteredReviews = reviews
    .filter((r) => {
      const term = searchTerm.toLowerCase();
      const matchesSearch =
        r.title.toLowerCase().includes(term) ||
        (r.description?.toLowerCase() || "").includes(term) ||
        (r.text?.toLowerCase() || "").includes(term);

      const matchesType = filterType === "all" ? true : r.type ===
filterType;

      return matchesSearch && matchesType;
    })
    .sort((a, b) => {
      if (sortDate === "desc") {
        return (
          new Date(b.date.split(".").reverse().join("-")) -
          new Date(a.date.split(".").reverse().join("-"))
        );
      } else {
        return (
          new Date(a.date.split(".").reverse().join("-")) -
          new Date(b.date.split(".").reverse().join("-"))
        );
      }
    });
  const handleReviewAdded = (newReview) => {
    setReviews((prev) => [...prev, newReview]);
  };
  const handleDelete = async (id, title) => {
    const confirm = await Swal.fire({
      title: "Видалити огляд?",
      text: `Точно видалити: "${title}" ?`,
      icon: "warning",
      showCancelButton: true,
      confirmButtonColor: "#d33",
      cancelButtonColor: "#f19f0b",
```

```
            confirmButtonText: "Видалити",
            cancelButtonText: "Скасувати",
          });

          if (confirm.isConfirmed) {
            try {
              await deleteDoc(doc(db, "reviews", id));
              setReviews((prev) => prev.filter((r) => r.id !== id));

              Swal.fire({
                icon: "success",
                title: "Видалено!",
                text: `Огляд "${title}" успішно видалено.`,
                timer: 1000,
                showConfirmButton: false,
              });
            } catch (error) {
              Swal.fire("Помилка!", "Не вдалося видалити огляд", "error");
              console.error("Помилка при видаленні:", error);
            }
          }
        };
        function cleanText(text) {
          if (!text) return "";
          return            text.replace(/#(INTRO|BODY|IMPORTANT|CONCLUSION)/gi,
"").trim();
        }
        return (
          <div className="max-w-6xl mx-auto px-4 py-8">
            {isAdmin && (
              <div className="mt-6 flex justify-center">
                <button
                  onClick={() => setIsModalOpen(true)}
                  className="px-6   py-3   bg-amber-500   text-white   rounded-lg
hover:bg-amber-700 shadow-md"
                >
                  Додати огляд
                </button>
              </div>
            )}
            <div  className="mt-6  flex  flex-col  sm:flex-row  gap-4  items-center
justify-center">
              <input
                type="text"
                value={searchTerm}
                onChange={(e) => setSearchTerm(e.target.value)}
                placeholder="Пошук оглядів..."
                className="w-full sm:w-1/2 border rounded-lg px-4 py-2 shadow-sm
focus:ring-2 focus:ring-amber-500 focus:outline-none"
              />
              <div className="flex gap-3">
                <button
                  onClick={() => setFilterType("all")}
                  className={`px-3 py-2 rounded-lg text-sm ${
                    filterType === "all" ? "bg-amber-600 text-white" : "bg-zinc-
100"
                  }`}
                >
                  Всі
                </button>
                <button
                  onClick={() => setFilterType("text")}
                  className={`px-3 py-2 rounded-lg text-sm ${
                    filterType  ===  "text"  ?  "bg-amber-600  text-white"  :  "bg-
zinc-100"
```

```
                }`}
              >
                Текстові
              </button>
              <button
                onClick={() => setFilterType("video")}
                className={`px-3 py-2 rounded-lg text-sm ${
                  filterType === "video" ? "bg-amber-600 text-white" : "bg-
zinc-100"
                }`}
              >
                Відео
              </button>
              <button
                onClick={() => setSortDate(sortDate === "desc" ? "asc" :
"desc")}
                className="px-3 py-2 rounded-lg text-sm bg-zinc-100 hover:bg-
zinc-200"
              >
                {sortDate === "desc" ? "Спочатку нові" : "Спочатку старі"}
              </button>
            </div>
          </div>
          <section className="mt-8 grid gap-6">
            {filteredReviews.length === 0 ? (
              <p className="text-zinc-700">Оглядів не знайдено...</p>
            ) : (
              filteredReviews.map((r) =>
                r.type === "video" ? (
                  <div
                    key={r.id}
                    className="relative bg-white rounded-xl shadow-md p-5
ring-1 ring-zinc-200"
                  >
                    {isAdmin && (
                      <button
                        onClick={() => handleDelete(r.id, r.title)}
                        className="absolute top-3 right-3 text-red-600
hover:text-red-800 text-3xl font-bold"
                        title="Видалити"
                      >
                        ×
                      </button>
                    )}

                    <h3 className="text-xl font-semibold text-zinc-900">
                      {r.title}
                    </h3>
                    <div className="mt-3 aspect-video rounded-lg overflow-
hidden">
                      <iframe
                        width="100%"
                        height="100%"
                        src={r.url.replace("watch?v=", "embed/")}
                        title={r.title}
                        frameBorder="0"
                        allow="accelerometer; autoplay; clipboard-write;
encrypted-media; gyroscope; picture-in-picture"
                        allowFullScreen
                        className="w-full h-full"
                      ></iframe>
                    </div>
                    <p className="mt-3 text-sm text-zinc-
700">{r.description}</p>
                    <p className="mt-2 text-xs text-zinc-500">{r.date}</p>
```

```
                  </div>
                ) : (
                  <div
                    key={r.id}
                    className="relative bg-white rounded-xl shadow-md p-5
ring-1 ring-zinc-200"
                  >
                    {isAdmin && (
                      <button
                        onClick={() => handleDelete(r.id, r.title)}
                        className="absolute top-3 right-3 text-red-600
hover:text-red-800 text-3xl font-bold"
                        title="Видалити"
                      >
                        ×
                      </button>
                    )}

                    <h3 className="text-lg font-semibold text-zinc-900">
                      {r.title}
                    </h3>
                    <p className="mt-2 text-sm text-zinc-700 line-clamp-3">
                      {cleanText(r.text)}
                    </p>
                    <Link
                      to={`/reviews/${r.id}`}
                      className="mt-2 inline-block text-sm font-medium text-
amber-600 hover:underline"
                    >
                      читати далі →
                    </Link>
                    <p className="mt-2 text-xs text-zinc-500">{r.date}</p>
                  </div>
                )
              )
            )}
          </section>

          {isModalOpen && (
            <AddReviewModal
              onClose={() => setIsModalOpen(false)}
              onAdded={handleReviewAdded}
            />
          )}
        </div>
      );
    }
```

ReviewDetails.jsx

```
    import React, { useEffect, useState } from "react";
    import { useParams, Link } from "react-router-dom";
    import { db } from "../firebase";
    import { doc, getDoc } from "firebase/firestore";

    export default function ReviewDetails() {
      const { id } = useParams();
      const [review, setReview] = useState(null);
      const [loading, setLoading] = useState(true);

      useEffect(() => {
        const fetchReview = async () => {
          try {
```

```
            const docRef = doc(db, "reviews", id);
            const docSnap = await getDoc(docRef);
            if (docSnap.exists()) {
              setReview({ id: docSnap.id, ...docSnap.data() });
            } else {
              setReview(null);
            }
          } catch (error) {
            console.error("Помилка завантаження огляду:", error);
          } finally {
            setLoading(false);
          }
        };
        fetchReview();
      }, [id]);

      if (loading) {
        return      <p      className="text-center      text-zinc-600      mt-
10">Завантаження...</p>;
      }

      if (!review) {
        return   <p   className="text-center   text-red-600   mt-10">Огляд   не
знайдено</p>;
      }

      // 🔽 Функція парсингу
      function parseArticle(text) {
        const parts = {
          intro: "",
          body: "",
          important: "",
          conclusion: ""
        };

        if (!text) return parts;

        const sections = text.split(/#(INTRO|BODY|IMPORTANT|CONCLUSION)/);
        for (let i = 1; i < sections.length; i += 2) {
          const key = sections[i].toLowerCase();
          const value = sections[i + 1]?.trim() || "";
          if (parts[key] !== undefined) parts[key] = value;
        }
        return parts;
      }

      const parts = parseArticle(review.text);

      return (
        <main className="max-w-3xl mx-auto px-4 py-10">
          {/* Назва */}
          <h1 className="text-3xl sm:text-4xl font-bold text-zinc-900 mb-4">
            {review.title}
          </h1>

          {/* Метадані */}
          <p className="text-sm text-zinc-500 mb-6">
            Дата {review.date}{" "}
            {review.author && (
              <span className="ml-4">
                {review.author.startsWith("http") ? (
                  <a
                    href={review.author}
                    target="_blank"
```

```
                    rel="noopener noreferrer"
                    className="text-amber-600 hover:underline"
                  >
                    Джерело
                  </a>
                ) : (
                  <>Автор: {review.author}</>
                )}
              </span>
            )}
          </p>

          {/* 🔽 Відображення статті */}
      <article className="prose prose-zinc max-w-none space-y-8">
        {/* Вступ */}
        {parts.intro && (
          <p  className="text-lg  text-zinc-800  leading-relaxed  whitespace-pre-
line">
            {parts.intro}
          </p>
        )}

        {/* Основна частина */}
        {parts.body && (
          <div  className="bg-zinc-50  border  border-zinc-200  rounded-xl  p-5
shadow-sm">
            <p  className="text-base  text-zinc-700  leading-relaxed  whitespace-
pre-line">
              {parts.body}
            </p>
          </div>
        )}

        {/* Важливі речі */}
        {parts.important && (
          <div className="flex items-start gap-3 bg-yellow-50 border-l-4 border-
amber-500 p-5 rounded-lg shadow-sm">
            <span className="text-amber-600 text-xl">❗</span>
            <p   className="text-lg   text-zinc-800   whitespace-pre-line   font-
medium">
              {parts.important}
            </p>
          </div>
        )}

        {/* Висновок */}
        {parts.conclusion && (
          <div  className="flex  items-start  gap-3  bg-orange-50  border  border-
orange-200 p-6 rounded-xl shadow-md">
            <span className="text-amber-700 text-xl">🧠</span>
            <p className="text-lg text-zinc-700 whitespace-pre-line italic">
              {parts.conclusion}
            </p>
          </div>
        )}
      </article>


          {/* Кнопка назад */}
          <div className="mt-10">
            <Link
              to="/reviews"
              className="px-5   py-2   rounded-lg   bg-amber-600   text-white
hover:bg-amber-700 transition"
```

```
            >
              ← Назад до оглядів
            </Link>
          </div>
        </main>
      );
    }
```

News.jsx

```
    import React, { useState, useEffect } from "react";
    import { db } from "../firebase";
    import { collection, getDocs, deleteDoc, doc } from "firebase/firestore";
    import AddNewsModal from "../components/Modals/AddNewsModal";
    import { Link } from "react-router-dom";
    import Swal from "sweetalert2";

    export default function News() {
      const [isAdmin] = useState(localStorage.getItem("isAdmin") === "true");
      const [isModalOpen, setIsModalOpen] = useState(false);
      const [news, setNews] = useState([]);
      const [searchTerm, setSearchTerm] = useState("");
      const [filterType, setFilterType] = useState("all");
      const [sortDate, setSortDate] = useState("desc");

      useEffect(() => {
        const fetchNews = async () => {
          const querySnapshot = await getDocs(collection(db, "news"));
          const data = querySnapshot.docs.map((doc) => ({
            id: doc.id,
            ...doc.data(),
          }));
          setNews(data);
        };
        fetchNews();
      }, []);

      const filteredNews = news
        .filter((n) => {
          const term = searchTerm.toLowerCase();
          const matchesSearch =
            n.title.toLowerCase().includes(term) ||
            (n.text?.toLowerCase() || "").includes(term) ||
            (n.source?.toLowerCase() || "").includes(term);

          const matchesType = filterType === "all" ? true : n.type ===
filterType;

          return matchesSearch && matchesType;
        })
        .sort((a, b) => {
          if (sortDate === "desc") {
            return (
              new Date(b.date.split(".").reverse().join("-")) -
              new Date(a.date.split(".").reverse().join("-"))
            );
          } else {
            return (
              new Date(a.date.split(".").reverse().join("-")) -
              new Date(b.date.split(".").reverse().join("-"))
            );
          }
        });
```

```
        const handleNewsAdded = (newItem) => {
          setNews((prev) => [...prev, newItem]);
        };

        const handleDelete = async (id, title) => {
          const confirm = await Swal.fire({
            title: "Видалити новину?",
            text: `Точно видалити: "${title}" ?`,
            icon: "warning",
            showCancelButton: true,
            confirmButtonColor: "#d33",
            cancelButtonColor: "#f19f0b",
            confirmButtonText: "Видалити",
            cancelButtonText: "Скасувати",
          });

          if (confirm.isConfirmed) {
            try {
              await deleteDoc(doc(db, "news", id));
              setNews((prev) => prev.filter((n) => n.id !== id));

              Swal.fire({
                icon: "success",
                title: "Видалено!",
                text: `Новину "${title}" успішно видалено.`,
                timer: 1000,
                showConfirmButton: false,
              });
            } catch (error) {
              Swal.fire("Помилка!", "Не вдалося видалити новину", "error");
              console.error("Помилка при видаленні:", error);
            }
          }
        };

        return (
          <div className="max-w-6xl mx-auto px-4 py-8">
            {/* Кнопка додати */}
            {isAdmin && (
              <div className="mt-6 flex justify-center">
                <button
                  onClick={() => setIsModalOpen(true)}
                  className="px-6 py-3 bg-amber-500 text-white rounded-lg
hover:bg-amber-700 shadow-md"
                >
                  Додати новину
                </button>
              </div>
            )}

            {/* Пошук + фільтри */}
            <div className="mt-6 flex flex-col sm:flex-row gap-4 items-center
justify-center">
              <input
                type="text"
                value={searchTerm}
                onChange={(e) => setSearchTerm(e.target.value)}
                placeholder="Пошук новин..."
                className="w-full sm:w-1/2 border rounded-lg px-4 py-2 shadow-sm
focus:ring-2 focus:ring-amber-500 focus:outline-none"
              />
              <div className="flex gap-3">
                <button
                  onClick={() => setFilterType("all")}
```

```
                  className={`px-3 py-2 rounded-lg text-sm ${
                    filterType === "all" ? "bg-amber-600 text-white" : "bg-zinc-
100"
                  }`}
                >
                  Всі
                </button>
                <button
                  onClick={() => setFilterType("Новина")}
                  className={`px-3 py-2 rounded-lg text-sm ${
                    filterType === "Новина" ? "bg-amber-600 text-white" : "bg-
zinc-100"
                  }`}
                >
                  Новини
                </button>
                <button
                  onClick={() => setFilterType("Патч")}
                  className={`px-3 py-2 rounded-lg text-sm ${
                    filterType === "Патч" ? "bg-amber-600 text-white" : "bg-
zinc-100"
                  }`}
                >
                  Патчі
                </button>
                <button
                  onClick={() => setFilterType("Анонс")}
                  className={`px-3 py-2 rounded-lg text-sm ${
                    filterType === "Анонс" ? "bg-amber-600 text-white" : "bg-
zinc-100"
                  }`}
                >
                  Анонси
                </button>
                <button
                  onClick={() => setFilterType("Івент")}
                  className={`px-3 py-2 rounded-lg text-sm ${
                    filterType === "Івент" ? "bg-amber-600 text-white" : "bg-
zinc-100"
                  }`}
                >
                  Івенти
                </button>
                <button
                  onClick={() => setSortDate(sortDate === "desc" ? "asc" :
"desc")}
                  className="px-3 py-2 rounded-lg text-sm bg-zinc-100 hover:bg-
zinc-200"
                >
                  {sortDate === "desc" ? "Спочатку нові" : "Спочатку старі"}
                </button>
              </div>
            </div>

            {/* Стрічка новин */}
            <section className="mt-8 grid gap-6">
              {filteredNews.length === 0 ? (
                <p className="text-zinc-700">Новин не знайдено...</p>
              ) : (
                filteredNews.map((n) => (
                  <div
                    key={n.id}
                    className="relative bg-white rounded-xl shadow-md p-5 ring-1
ring-zinc-200"
                  >
```

```
                    {isAdmin && (
                      <button
                        onClick={() => handleDelete(n.id, n.title)}
                        className="absolute    top-3    right-3    text-red-600
hover:text-red-800 text-3xl font-bold"
                        title="Видалити"
                      >
                        ×
                      </button>
                    )}

                    <h3     className="text-lg     font-semibold     text-zinc-
900">{n.title}</h3>
                    <p className="mt-2 text-sm text-zinc-700 line-clamp-3">
                      {n.text}
                    </p>
                    <Link
                      to={`/news/${n.id}`}
                      className="mt-2  inline-block  text-sm  font-medium  text-
amber-600 hover:underline"
                    >
                      читати далі →
                    </Link>
                    <div className="mt-2 flex items-center justify-between text-
xs text-zinc-500">
                      <span>{n.date}</span>
                      <span>{n.type}</span>
                    </div>
                  </div>
                ))
              )}
            </section>

            {/* Модалка */}
            {isModalOpen && (
              <AddNewsModal
                onClose={() => setIsModalOpen(false)}
                onAdded={handleNewsAdded}
              />
            )}
          </div>
        );
      }
```

NewsDetails.jsx

```
      import React, { useEffect, useState } from "react";
      import { useParams, Link } from "react-router-dom";
      import { db } from "../firebase";
      import { doc, getDoc } from "firebase/firestore";

      export default function NewsDetails() {
        const { id } = useParams();
        const [newsItem, setNewsItem] = useState(null);
        const [loading, setLoading] = useState(true);

        useEffect(() => {
          const fetchNews = async () => {
            try {
              const docRef = doc(db, "news", id);
              const docSnap = await getDoc(docRef);
              if (docSnap.exists()) {
                setNewsItem({ id: docSnap.id, ...docSnap.data() });
```

```
            } else {
              setNewsItem(null);
            }
          } catch (error) {
            console.error("Помилка завантаження новини:", error);
          } finally {
            setLoading(false);
          }
        };
        fetchNews();
      }, [id]);

      if (loading) {
        return      <p      className="text-center      text-zinc-600      mt-
10">Завантаження...</p>;
      }

      if (!newsItem) {
        return   <p   className="text-center   text-red-600   mt-10">Новину   не
знайдено</p>;
      }

      return (
        <main className="max-w-3xl mx-auto px-4 py-10">
          {/* Заголовок */}
          <h1 className="text-3xl sm:text-4xl font-bold text-zinc-900 mb-4">
            {newsItem.title}
          </h1>

          {/* Метадані */}
          <div className="flex flex-wrap items-center gap-3 text-sm text-zinc-
500 mb-6">
            <span  className="px-3  py-1  rounded-full  bg-amber-100  text-amber-
700 font-medium">
              {newsItem.type}
            </span>
            <span> {newsItem.date}</span>
            {newsItem.source && newsItem.source !== "—" && (
              <span>
                {newsItem.source.startsWith("http") ? (
                  <a
                    href={newsItem.source}
                    target="_blank"
                    rel="noopener noreferrer"
                    className="text-amber-600 hover:underline"
                  >
                    Джерело
                  </a>
                ) : (
                  <>Джерело: {newsItem.source}</>
                )}
              </span>
            )}
          </div>

          <article className="prose prose-zinc max-w-none">
            <p  className="text-lg  leading-relaxed  text-zinc-800  whitespace-
pre-line">
              {newsItem.text}
            </p>
          </article>

          <div className="mt-10">
            <Link
              to="/news"
```

```
                className="px-5   py-2   rounded-lg   bg-amber-600   text-white
hover:bg-amber-700 transition"
              >
                ← Назад до новин
              </Link>
            </div>
          </main>
        );
      }
```

Tutorials.jsx

```
      import React, { useEffect, useMemo, useState } from "react";
      import { db } from "../firebase";
      import { collection, getDocs, deleteDoc, doc } from "firebase/firestore";
      import AddTutorialModal from "../components/Modals/AddTutorialModal";
      import Swal from "sweetalert2";

      const SECTIONS = [
        { id: "tutorials", title: "Поради та гайди" },
        { id: "faq", title: "FAQ" },
        { id: "glossary", title: "Словник термінів" },
      ];

      export default function Tutorials() {
        const [activeId, setActiveId] = useState(SECTIONS[0].id);
        const [isModalOpen, setIsModalOpen] = useState(false);
        const [tutorials, setTutorials] = useState([]);
        const [expandedIds, setExpandedIds] = useState([]);

        const [isAdmin] = useState(localStorage.getItem("isAdmin") === "true");

        useEffect(() => {
          const obs = new IntersectionObserver(
            (entries) => {
              const visible = entries
                .filter((e) => e.isIntersecting)
                .sort((a, b) => b.intersectionRatio - a.intersectionRatio)[0];
              if (visible?.target?.id) setActiveId(visible.target.id);
            },
            { rootMargin: "-20% 0px -60% 0px", threshold: [0.2, 0.4, 0.6, 0.8] }
          );

          SECTIONS.forEach((s) => {
            const el = document.getElementById(s.id);
            if (el) obs.observe(el);
          });
          return () => obs.disconnect();
        }, []);

        useEffect(() => {
          const fetchTutorials = async () => {
            const querySnapshot = await getDocs(collection(db, "tutorials"));
            const data = querySnapshot.docs.map((doc) => ({
              id: doc.id,
              ...doc.data(),
            }));
            setTutorials(data);
          };
          fetchTutorials();
        }, []);

        const handleDelete = async (id, title) => {
```

```
      const confirm = await Swal.fire({
        title: "Видалити матеріал?",
        text: `Точно видалити: "${title || "елемент"}"?`,
        icon: "warning",
        showCancelButton: true,
        confirmButtonColor: "#d33",
        cancelButtonColor: "#f19f0b",
        confirmButtonText: "Видалити",
        cancelButtonText: "Скасувати",
      });

      if (confirm.isConfirmed) {
        try {
          await deleteDoc(doc(db, "tutorials", id));
          setTutorials((prev) => prev.filter((t) => t.id !== id));
          Swal.fire({
            icon: "success",
            title: "Видалено!",
            timer: 1000,
            showConfirmButton: false,
          });
        } catch (error) {
          Swal.fire("Помилка!", "Не вдалося видалити матеріал", "error");
          console.error("Помилка при видаленні:", error);
        }
      }
    };

    const toggleExpand = (id) => {
      setExpandedIds((prev) =>
        prev.includes(id) ? prev.filter((x) => x !== id) : [...prev, id]
      );
    };

    const linkClass = useMemo(
      () => (id) =>
        `block rounded-xl px-3 py-2 text-sm sm:text-base transition-colors
${
          activeId === id
            ? "bg-yellow-100 text-yellow-800"
            : "text-zinc-700 hover:bg-zinc-100"
        }`,
      [activeId]
    );

    const renderTutorialCard = (t) => {
      if (t.category === "Поради та гайди") {
        const expanded = expandedIds.includes(t.id);
        return (
          <div
            key={t.id}
            className="relative bg-white rounded-xl shadow-md p-5 ring-1
ring-zinc-200"
          >
            {isAdmin && (
              <button
                onClick={() => handleDelete(t.id, t.title)}
                className="absolute top-3 right-3 text-red-600 hover:text-
red-800 text-2xl font-bold"
              >
                ×
              </button>
            )}
```

```
                <h4       className="text-lg       font-semibold       text-zinc-
900">{t.title}</h4>

                {t.videoUrl ? (
                  <div  className="mt-3  aspect-video  rounded-lg  overflow-hidden
shadow">
                    <iframe
                      src={t.videoUrl.replace("watch?v=", "embed/")}
                      title={t.title}
                      frameBorder="0"
                      allow="accelerometer;      autoplay;      clipboard-write;
encrypted-media; gyroscope; picture-in-picture"
                      allowFullScreen
                      className="w-full h-full"
                    ></iframe>
                  </div>
                ) : (
                  <p className="mt-3 text-sm text-zinc-700">
                    {expanded ? t.text : `${t.text.slice(0, 150)}...`}
                  </p>
                )}

                <div  className="mt-3  flex  items-center  justify-between  text-xs
text-zinc-500">
                  {!t.videoUrl && (
                    <button
                      onClick={() => toggleExpand(t.id)}
                      className="text-amber-600 hover:underline text-sm"
                    >
                      {expanded ? "Сховати" : "Показати все"}
                    </button>
                  )}
                </div>
              </div>
            );
          }

      if (t.category === "FAQ") {
        return (
          <div
            key={t.id}
            className="relative  bg-white  rounded-2xl  shadow-md  p-6  border
border-amber-100 hover:shadow-lg transition"
          >
            {isAdmin && (
              <button
                onClick={() => handleDelete(t.id, t.question)}
                className="absolute  top-3  right-3  text-red-600  hover:text-red-
800 text-2xl font-bold"
                title="Видалити"
              >
                ×
              </button>
            )}

            {/* Питання */}
            <div className="flex items-start gap-3">
              <span className="text-amber-600 text-lg mt-1"></span>
              <p        className="text-lg        font-semibold        text-zinc-
900">{t.question}</p>
            </div>

            {/* Відповідь */}
            <div className="mt-3 ml-6 border-l-4 border-amber-200 pl-4">
```

```
              <p        className="text-sm        text-zinc-700        leading-
relaxed">{t.answer}</p>
            </div>
          </div>
        );
      }


        if (t.category === "Словник термінів") {
        return (
          <div
            key={t.id}
            className="relative  bg-white  rounded-2xl  shadow-md  p-6  border
border-blue-100 hover:shadow-lg transition"
          >
            {isAdmin && (
              <button
                onClick={() => handleDelete(t.id, t.term)}
                className="absolute  top-3  right-3  text-red-600  hover:text-red-
800 text-2xl font-bold"
                title="Видалити"
              >
                ×
              </button>
            )}

            {/* Термін */}
            <div className="flex items-start gap-3">
              <span className="text-blue-600 text-lg mt-1"></span>
              <p className="text-lg font-semibold text-zinc-900">{t.term}</p>
            </div>

            {/* Пояснення */}
            <div className="mt-3 ml-6 border-l-4 border-blue-200 pl-4">
              <p        className="text-sm        text-zinc-700        leading-
relaxed">{t.definition}</p>
            </div>
          </div>
        );
      }

        };

        return (
          <div className="min-h-[100svh] bg-gradient-to-b from-white to-zinc-100
text-zinc-800">
            <div className="mx-auto max-w-6xl px-4 py-10 sm:py-14">
              {isAdmin && (
                <div className="mb-6 flex justify-center">
                  <button
                    onClick={() => setIsModalOpen(true)}
                    className="px-5  py-2  bg-amber-500  text-white  rounded-lg
hover:bg-amber-700 shadow"
                  >
                    Додати матеріал
                  </button>
                </div>
              )}

              <div        className="grid        grid-cols-1        md:grid-cols-
[240px_minmax(0,1fr)] gap-8">
                {/* Sidebar */}
                <aside className="md:sticky md:top-20 h-max">
                  <h4   className="mb-3   text-sm   font-semibold   text-zinc-500
uppercase tracking-wider">
```

```
                  Категорії
                </h4>
                <nav className="space-y-1">
                  {SECTIONS.map((s) => (
                    <a              key={s.id}              href={`#${s.id}`}
className={linkClass(s.id)}>
                      {s.title}
                    </a>
                  ))}
                </nav>
              </aside>

              {/* Content */}
              <main className="scroll-smooth space-y-10">
                <section id="tutorials" className="scroll-mt-24">
                  <h2  className="text-3xl  sm:text-4xl  font-bold  text-zinc-
900">
                    Поради та гайди
                  </h2>
                  <div className="mt-4 grid gap-4">
                    {tutorials
                      .filter((t) => t.category === "Поради та гайди")
                      .map(renderTutorialCard)}
                  </div>
                </section>

                <section id="faq" className="scroll-mt-24">
                  <h3     className="text-2xl     font-semibold     text-zinc-
900">FAQ</h3>
                  <div className="mt-4 grid gap-4">
                    {tutorials
                      .filter((t) => t.category === "FAQ")
                      .map(renderTutorialCard)}
                  </div>
                </section>

                <section id="glossary" className="scroll-mt-24">
                  <h3 className="text-2xl font-semibold text-zinc-900">
                    Словник термінів
                  </h3>
                  <div className="mt-4 grid gap-4">
                    {tutorials
                      .filter((t) => t.category === "Словник термінів")
                      .map(renderTutorialCard)}
                  </div>
                </section>
              </main>
            </div>
          </div>

        {isModalOpen && (
        <AddTutorialModal
          onClose={() => setIsModalOpen(false)}
          onAdded={(newItem) =>
            setTutorials((prev) => [newItem, ...prev])
          }
        />
      )}
          </div>
        );
      }
```

Login.jsx

```
import React, { useState } from "react";
import { useNavigate } from "react-router-dom";

export default function Login() {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [error, setError] = useState("");
  const navigate = useNavigate();

  const handleLogin = (e) => {
    e.preventDefault();
    if (email === "wotadmin@gmail.com" && password === "wotadming") {
      localStorage.setItem("isAdmin", "true"); // зберігаємо прапорець
      navigate("/admin"); // перенаправлення
    } else {
      setError("Невірна пошта або пароль");
    }
  };

  return (
    <div className="max-w-md mx-auto mt-12 p-6 bg-white rounded-xl shadow-
md border border-zinc-200">
      <h2 className="text-3xl sm:text-4xl font-bold text-zinc-900 mb-6">
        Адмінка
      </h2>

      <form onSubmit={handleLogin} className="space-y-4">
        {/* Поле email */}
        <div>
          <label className="block text-sm font-medium text-zinc-700">
            Електронна пошта
          </label>
          <input
            type="email"
            value={email}
            onChange={(e) => setEmail(e.target.value)}
            className="mt-1 w-full border rounded-lg px-3 py-2 shadow-sm
focus:ring-2 focus:ring-amber-500 focus:outline-none"
            placeholder="Введіть email"
            required
          />
        </div>

        {/* Поле пароль */}
        <div>
          <label className="block text-sm font-medium text-zinc-700">
            Пароль
          </label>
          <input
            type="password"
            value={password}
            onChange={(e) => setPassword(e.target.value)}
            className="mt-1 w-full border rounded-lg px-3 py-2 shadow-sm
focus:ring-2 focus:ring-amber-500 focus:outline-none"
            placeholder="Введіть пароль"
            required
          />
        </div>

        {/* Помилка */}
        {error && <p className="text-red-600 text-sm">{error}</p>}

        {/* Кнопка */}
        <button
```

```
                type="submit"
                className="w-full bg-amber-600 text-white py-2 px-4 rounded-lg
font-medium hover:bg-amber-700 transition"
              >
                Увійти
              </button>
            </form>
          </div>
        );
      }
```

AdminHome.jsx

```
      import React, { useEffect } from "react";
      import { useNavigate } from "react-router-dom";

      export default function Admin_Home() {
        const navigate = useNavigate();

        useEffect(() => {
          const isAdmin = localStorage.getItem("isAdmin") === "true";
          if (!isAdmin) {
            navigate("/login"); // якщо користувач не адмін → редирект
          }
        }, [navigate]);

        return (
          <main className="max-w-5xl mx-auto px-4 py-8">
            <h2 className="text-3xl font-bold text-zinc-900 mb-4">
              Вітаємо в адмін-панелі!
            </h2>
            <p className="text-lg text-zinc-700 leading-relaxed">
              Тут ви можете керувати контентом сайту: додавати та редагувати{"
"}
              <span className="font-semibold text-amber-600">огляди</span>,{" "}
              <span  className="font-semibold  text-amber-600">новини</span>  і
розділ{" "}
              <span className="font-semibold text-amber-600">«Новачкам»</span>.
            </p>
          </main>
        );
      }
```