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
(підпис)

«___»_____202_ р.

КВАЛІФІКАЦІЙНА РОБОТА

на тему

«РОЗРОБКА ТА АНАЛІЗ ЕФЕКТИВНОСТІ РІЗНИХ АЛГОРИТМІВ ОБЛІКУ ВИТРАТ ТА ДОХОДІВ У ПРОСТОМУ ЗАСТОСУНКУ ДЛЯ КЕРУВАННЯ ОСОБИСТИМИ ФІНАНСАМИ»

зі спеціальності 122 Комп'ютерні науки
освітня програма «Комп'ютерні науки»
ступеня магістра

Виконавець роботи Задворкін Максим Олександрович
_____«____»_____ 202_ р.
(підпис)

Науковий керівник к. ф.-м. н., доцент, Парфьонова Тетяна Олександрівна
_____«____»_____ 202_ р.
(підпис)

Рецензент _____

ПОЛТАВА 2025

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

Умовні позначення, символи, скорочення, терміни	Пояснення умовних позначень, скорочень, символів
clearFields()	Ця функція призначена для повного очищення всіх елементів введення на формі, щоб користувач міг почати роботу з чистими полями. Після виклику цієї функції усі текстові поля TextBox, випадаючі списки ComboBox, перемикачі RadioButton, прапорці CheckBox та інші елементи управління скидаються у початковий стан.
displayCategoryList()	Використовується для візуалізації і показу усіх наявних категорій, що зберігаються у програмі або базі даних.
refreshData()	Це виклик методу, який відповідає за оновлення або повторне завантаження даних у програмі.
SqlConnection	Це клас у середовищі .NET C#, який використовується для встановлення з’єднання з базою даних SQL Server. Він є частиною простору імен System.Data.SqlClient і є основним елементом для взаємодії між програмою та базою даних.

ВСТУП

У сучасному світі питання ефективного управління особистими фінансами стає дедалі актуальнішим. Збільшення кількості електронних транзакцій, різноманітність джерел доходів і витрат, а також необхідність швидкого прийняття фінансових рішень вимагають створення зручних інструментів для контролю коштів. Одним із таких рішень є застосунки для обліку особистих фінансів. Їх ключовим елементом виступають алгоритми, які дозволяють організовувати, аналізувати та оптимізувати дані про доходи й витрати.

Особистий фінансовий облік - це систематизація даних про рух коштів окремої особи чи сім'ї. Основними завданнями є: фіксація джерел доходу та категорій витрат; аналіз фінансових потоків; формування звітів і прогнозів; оптимізація витрат і накопичень. Для реалізації цих завдань у програмному забезпеченні застосовуються алгоритми різного рівня складності - від простих структурованих таблиць до машинного навчання та статистичних моделей.

Особисті фінансові застосунки допомагають людині бачити повну картину власних грошових потоків, формувати індивідуальні бюджети, встановлювати фінансові цілі та приймати більш обґрунтовані рішення. Водночас ефективність таких систем визначається не лише зручністю інтерфейсу чи привабливістю дизайну, але й передусім алгоритмами, які лежать в їхній основі. Саме алгоритмічні рішення дозволяють організувати дані, аналізувати їх, робити прогнози та пропонувати рекомендації користувачеві.[1]

Мета кваліфікаційної роботи – розробка та дослідження ефективності різних алгоритмів обліку витрат і доходів у межах створення простого програмного застосунку для керування особистими фінансами, який забезпечує зручний інтерфейс та високу надійність зберігання даних.

Об’єкт розробки – система програмного забезпечення для ведення та організації особистих фінансів, що забезпечує загальні функції контролю доходів і витрат.

Предмет розробки – методи, алгоритми та засоби реалізації процесів обліку, зберігання, обробки й відображення фінансових даних у межах програмної системи, спрямовані на забезпечення зручного та ефективного керування особистими фінансами.
Методи дослідження – аналітичні, порівняльні та програмно-експериментальні.[2]

Актуальність розробки – у сучасних умовах постійного зростання цін, нестабільності економіки та збільшення обсягу фінансових операцій питання раціонального управління особистими фінансами набуває особливого значення. Багато користувачів стикаються з проблемою неефективного контролю витрат, відсутності систематизованих записів і неможливості швидко оцінити власний фінансовий стан. Саме тому розробка програмного застосунку, який дозволяє автоматизувати облік доходів і витрат, є надзвичайно актуальною роботою на сьогодення.

Кваліфікаційна робота складається з чотирьох розділів. У першому розділі складено постановку задачі, створення застосунку для керування особистими фінансами. У другому розділі оглянуто застосунок Monefy для керування особистими фінансами, а також оглянуто схожий застосунок Moneygraph. У третьому розділі розглянуто покрокову логіку роботи коду застосунку і алгоритмів обліку фінансів, UML схему роботи застосунку, UML схему роботи коду застосунку. У четвертому розділі розглянуто процес програмної реалізації застосунку управління фінансами, а також опис роботи програмного забезпечення. Обсяг пояснювальної записки: 45 стор., в т.ч. основна частина – 41 стор., джерела – 20 назв.

1. ПОСТАНОВКА ЗАДАЧ

Метою роботи є створення програмного застосунку для обліку особистих фінансів, який дозволяє користувачу вести систематизований облік витрат і доходів та отримувати аналітичну інформацію про свій фінансовий стан. Для досягнення цієї мети необхідно вирішити ряд задач:

1. Аналіз предметної області та існуючих рішень, вивчення існуючих програм для обліку фінансів та їх алгоритмів обробки даних.
2. Виявлення переваг і недоліків цих рішень, визначення критеріїв ефективності алгоритмів обліку.
3. Проєктування архітектури застосунку, визначення основних модулів системи: модуль введення даних, модуль збереження, модуль аналітики та звітності.
4. Розробка структури бази даних для зберігання записів про доходи та витрати, категорій і користувацьких налаштувань.
5. Вибір технологій та інструментів розробки C#, .NET, SQL Server, Windows Forms.
6. Реалізація алгоритмів обліку, ведення обліку отриманих доходів користувача.
7. Збереження інформації у локальній базі даних.
8. Робота з категоріями доходів, що зберігаються у таблиці.
9. З виконання CRUD-операцій над таблицею доходів і витрат.
10. Забезпечення точності та надійності обробки даних, уникнення дублювання записів та помилок введення.
11. Розробка користувацького інтерфейсу, створення зручного та інтуїтивного інтерфейсу для введення доходів і витрат, перегляду статистики.
12. Проведення детального тестування для перевірки правильності роботи застосунку.

2. ІНФОРМАЦІЙНИЙ ОГЛЯД

2.1. Огляд застосунку Monefy для керування особистими фінансами

Як просто стежити за своїми фінансами та розуміти, куди зникає кожна гривня? Monefy - особистий фінансовий менеджер і трекер бюджету, який зробить цей процес набагато зручнішим. [3]

Щоразу, коли користувач купує каву, оплачує рахунок чи заправляє авто, достатньо просто додати нову витрату - і готово! Додаток автоматично зберігатиме всі витрати та згодом відображатиме їх у зручному застосунку.

Користувачу відкривається головний екран на якому відображено графік прибутків та витрат. (рис. 2.1).

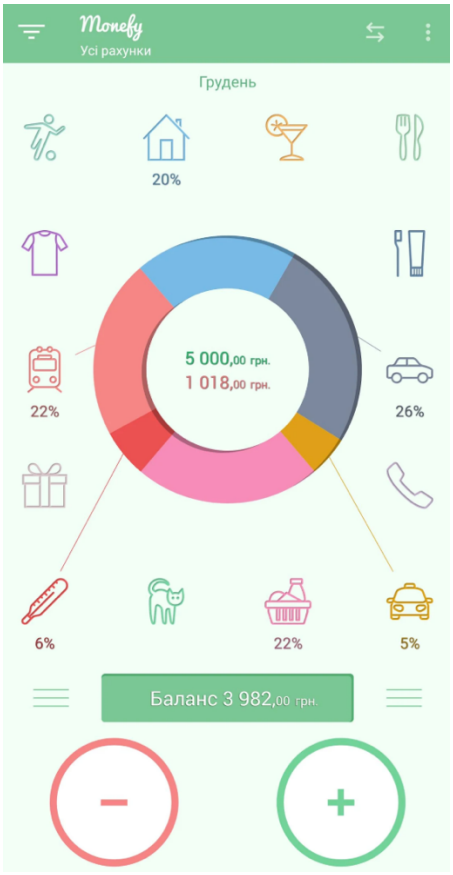

Рисунок 2.1 – Головний екран застосунку

У застосунку є можливість перемкнутися з графіку на режим списку, щоб детальніше переглянути інформацію про бюджет. (рис. 2.2). Додавання витрат або прибутків здійснюється по натисканню круглих кнопок знизу. (рис. 2.3).

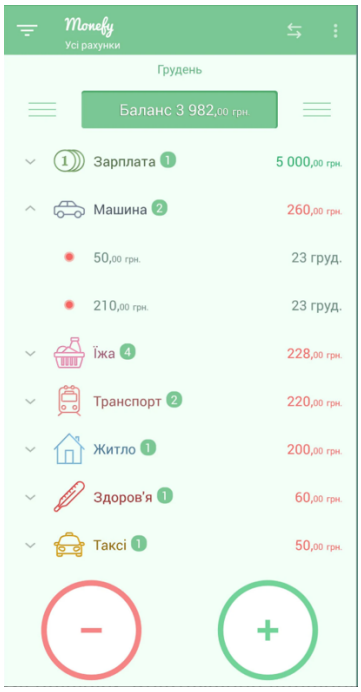

Рисунок 2.2 – Режим списку

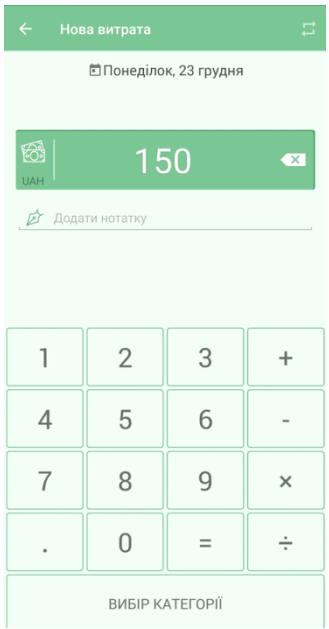

Рисунок 2.3 – Додавання витрати або прибутку

Застосунок має відмінні кольорові іконки, які чудово вписуються в загальний дизайн та надають зручне інтуїтивно зрозуміле керування категоріями. (рис. 2.4).

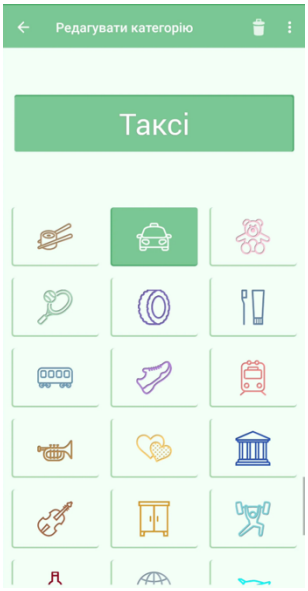

Рисунок 2.4 – Огляд іконок категорій застосунку

Також у застосунку є чудова та зручна можливість синхронізації даних через Dropbox або Google drive. (рис. 2.5).

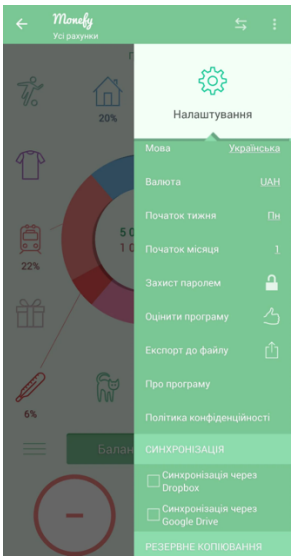

Рисунок 2.5 – Синхронізація даних

За потреби користувач може перемкнути оформлення застосунку на темний колір. (рис. 2.6).

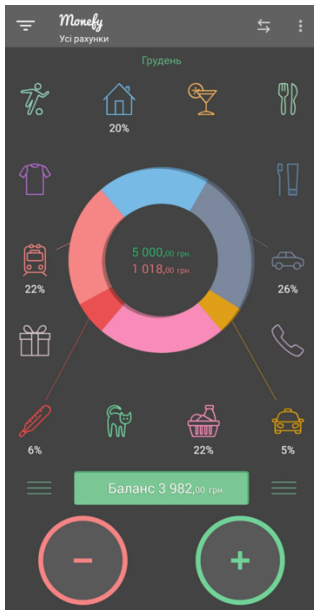

Рисунок 2.6 – Огляд темного оформлення застосунку

Для зручності користувача, додатково є можливість розмістити віджети на робочому столі. (рис. 2.7).

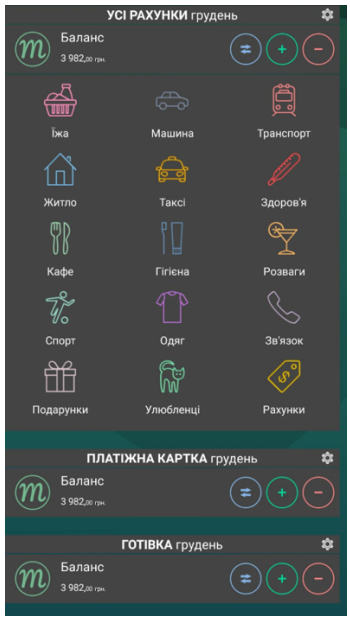

Рисунок 2.7 – Огляд віджету

Monefy - Розпорядник бюджету має переваги та недоліки.

Переваги:

1. Програма має просте та дружнє оформлення. Всі основні функції доступні буквально в кілька натискань - можна миттєво додати витрату або дохід, вибрати категорію та суму.
2. Щоб занести нову покупку, достатньо кількох секунд. Це зручно під час щоденного використання - наприклад, після покупки кави або проїзду в транспорті.
3. Витрати й доходи відображаються у вигляді яскравих кругових діаграм і графіків, що допомагає миттєво побачити, куди саме йде більша частина бюджету.
4. Monefy дозволяє створювати різні рахунки (готівка, банківська картка, заощадження) і переміщати гроші між ними, що допомагає бачити повну картину фінансів.
5. Застосунок корисний для тих, хто має доходи або витрати в різних валютах - можна легко налаштувати облік у гривнях, доларах, євро тощо.
6. Це зручно, якщо ви користуєтеся кількома пристроями або хочете мати резервну копію даних.
7. Користувач може встановити PIN-код або біометричний захист для входу в застосунок, що запобігає несанкціонованому доступу.
8. Monefy дозволяє аналізувати витрати за категоріями, періодами (день, тиждень, місяць) і бачити тенденції - де можна зекономити або які звички потребують змін.

Недоліки:

1. Усі витрати потрібно вносити вручну. Для деяких користувачів це може бути незручно, особливо якщо операцій багато.

2. Безкоштовна версія має обмежений функціонал
3. Деякі можливості, як-от синхронізація чи створення додаткових категорій, доступні лише у платній версії Pro.
4. Застосунок більше підходить для фіксації фактів витрат, ніж для детального фінансового планування чи прогнозування.
5. Користувач може змінювати або створювати категорії, але гнучкість у цьому питанні менша, ніж у деяких конкурентів (наприклад, Wallet чи Money Lover).

Monefy - чудовий вибір для тих, хто хоче швидко й просто контролювати свої щоденні витрати без складних налаштувань. Його переваги - це зручність, візуальна наочність і швидкість роботи. Проте для користувачів, яким потрібна глибока аналітика, автоматичне підключення банків або спільний бюджет, функціоналу може не вистачити.

2.2. Огляд застосунку Moneygraph для контролю витрат та прибутків

Moneygraph - це сучасний застосунок для обліку доходів і витрат, створений для того, щоб допомогти користувачам ефективно керувати власними фінансами. Його головна мета - показати повну картину руху грошей і зробити процес контролю бюджету простим, наочним та зручним. (рис. 2.8).

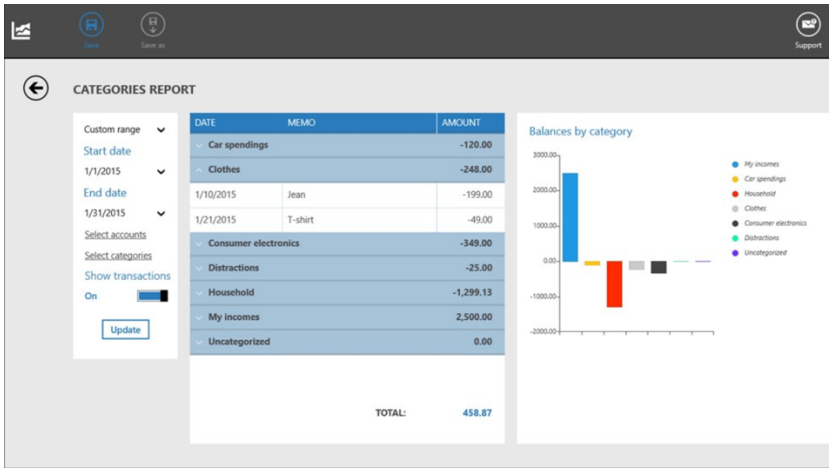

Рисунок 2.8 – Головний екран застосунку

У застосунку користувач може додавати усі свої доходи та витрати, створювати власні категорії, вести облік за днями, тижнями чи місяцями. (рис. 2.9). Також є режим редагування, це чудовий інструмент для тих, хто прагне контролювати свої фінанси, планувати заощадження та формувати фінансову дисципліну. (рис. 2.10).

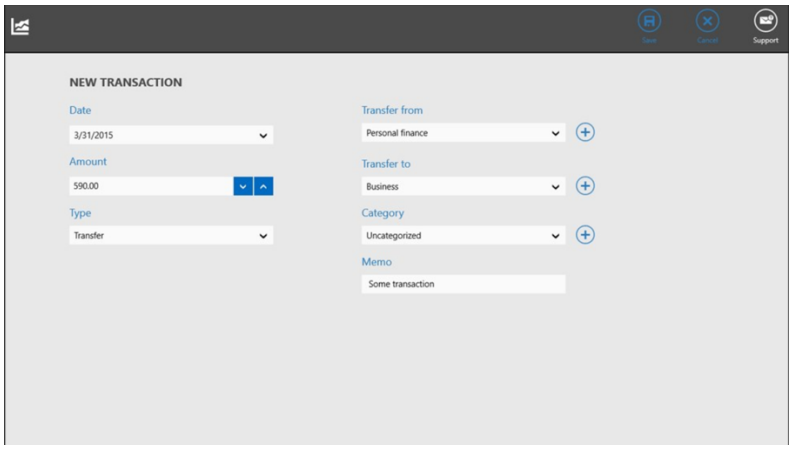

Рисунок 2.9 – Додавання нової транзакції

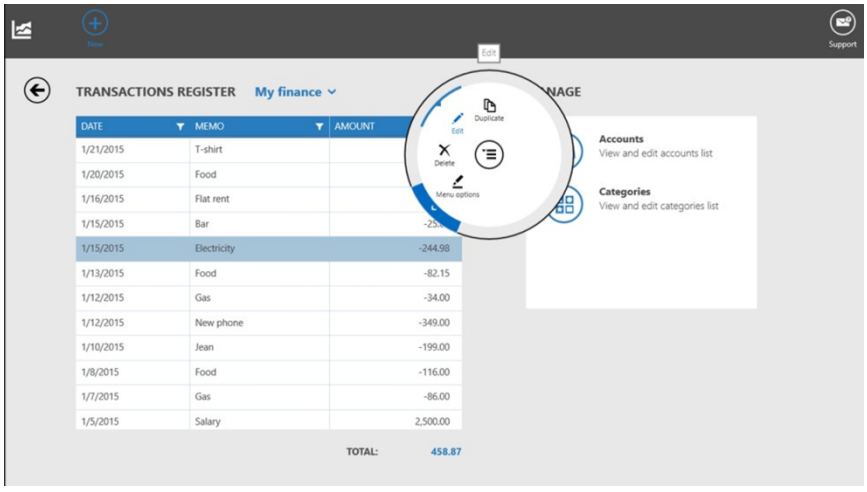

Рисунок 2.10 – Режим редагування

У застосунку Moneygraph передбачена можливість створення та відновлення резервної копії даних, що дозволяє користувачам не втратити інформацію про свої доходи, витрати та категорії у разі видалення програми, зміни пристрою або системного збою. (рис. 2.11).

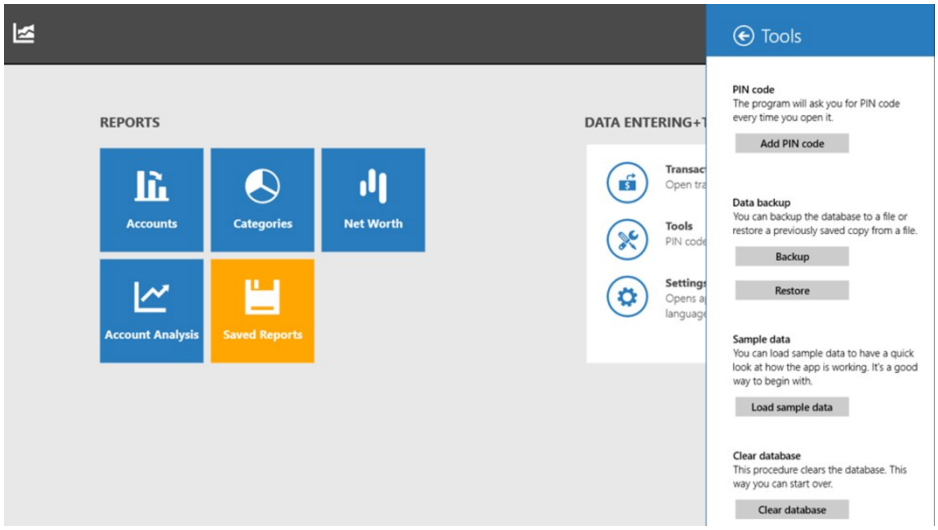

Рисунок 2.11 – Резервна копія

У застосунку Moneygraph передбачена функція встановлення PIN-коду, яка забезпечує додатковий рівень безпеки та конфіденційності фінансових даних користувача. (рис. 2.12).

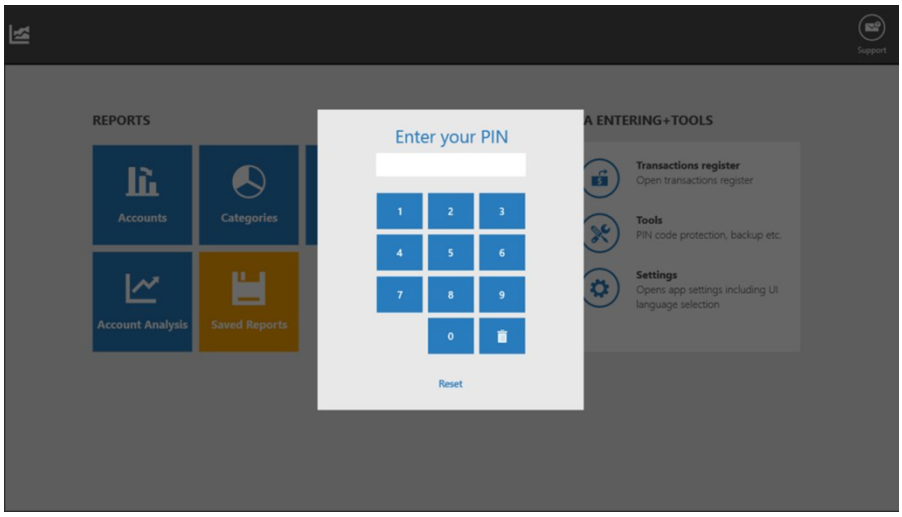

Рисунок 2.12 – Пін код

Загальний огляд переваг і недоліків застосунку Moneygraph.

Переваги:

1. Дозволяє вести облік доходів і витрат у зручному форматі - створювати категорії витрат, вести кілька рахунків, аналізувати рух коштів.
2. Наявність графіків і звітів дає можливість візуалізувати фінанси - це допомагає краще усвідомлювати, на що йдуть гроші.
3. Підтримка функцій, як-от резервна копія і PIN-код, додають рівень безпеки та збереження даних.
4. Проста у користуванні інтерфейс-ідея, що підходить тим, хто не хоче надто складних фінансових інструментів, але бажає контролю.

Недоліки:

1. PIN-код і резервне копіювання, ці функції входять лише в преміум-версію або частково обмежені - тобто безкоштовна версія може бути обмеженою.
2. Застосунок сам по собі не замінює професійного фінансового радника або бухгалтерію - він допомагає вести облік, але не вирішує складні фінансові стратегії чи податкові питання.
3. Якщо дані вводити вручну (що часто буває у таких застосунках), то витрати часу на ведення записів можуть зменшити мотивацію користування.
4. Залежність від користувача: для коректного аналізу потрібно регулярно вносити дані, категоризувати витрати. Якщо це не робити - користі буде менше.

3. ТЕОРЕТИЧНА ЧАСТИНА

3.1. Покрокова логіка роботи коду застосунку і алгоритмів обліку фінансів

Запуск головної форми, коли користувач успішно входить у систему через Form1 вікно логіну, програма відкриває MainForm. Далі викликається конструктор класу MainForm(). У конструкторі викликається метод InitializeComponent(), він створює елементи інтерфейсу форми. Далі викликається метод displayUsername() - щоб показати ім’я користувача на екрані.

Крок 2. Відображення імені користувача.

Метод displayUsername() зчитує ім’я користувача з класу Form1 через змінну Form1.username. Потім форматує його, першу літеру робить великою, інші залишає без змін.

getUsername.Substring(0, 1).ToUpper() + getUsername.Substring(1)

Виводить привітання у текстовому полі greetUser.

Крок 3. Кнопка закриття програми.

Подія close_Click виконується при натисканні кнопки або іконки виходу, викликає Application.Exit() - повністю завершує роботу програми.

Крок 4. Кнопка “Logout” вихід із системи.

Подія logout_btn_Click, показує повідомлення з підтвердженням:

Ви впевнені, що хочете вийти з системи?

Якщо користувач натискає так:

Створюється новий об’єкт Form1 форма входу, відображається форма логіну loginForm.Show(). Поточна форма MainForm ховається. Якщо ні - дія скасовується.

Крок 5. Перемикання між вкладками/розділами.

Головна форма має кілька вкладених форм-контролів:

dashboardForm1

categoryForm1

incomeForm1

expensesForm1

Кожна кнопка відповідає за показ потрібної форми.

Кнопка Dashboard. Подія dashboard_btn_Click робить видимою форму dashboardForm1 і приховує всі інші. Викликає метод refreshData() у DashboardForm, щоб оновити дані і баланси.

Кнопка “Add Category”, подія addCategoty_btn_Click робить активною CategoryForm. Приховує решту вкладок, потім викликає cForm.refreshData() - оновлює список категорій.

Кнопка “Income”, подія income_btn_Click, показує форму IncomeForm1 і приховує всі інші вкладки, викликає iForm.refreshData() - оновлює дані про доходи.

Кнопка “Expenses”, подія expenses_btn_Click показує форму ExpensesForm, приховує інші, а також викликає eForm.refreshData() - оновлює дані про витрати.

Крок 6. Завершення роботи.

Програма завершується після натискання кнопки “Close”. При “Logout” - користувач повертається на форму входу. Усі дані в основних формах можуть оновлюватися через метод refreshData().

Стислий псевдокод алгоритму:

Start MainForm

→ InitializeComponent()

→ displayUsername()

  → get username from Form1

  → display “Welcome, Username”

When close button clicked:

  → Exit application

When logout button clicked:

  → Ask confirmation

→ If Yes → Show Form1 (login) and hide MainForm

When dashboard button clicked:

→ Show dashboardForm1

→ Hide others

→ dashboardForm1.refreshData()

When addCategory button clicked:

→ Show categoryForm1

→ Hide others

→ categoryForm1.refreshData()

When income button clicked:

→ Show incomeForm1

→ Hide others

→ incomeForm1.refreshData()

When expenses button clicked:

→ Show expensesForm1

→ Hide others

→ expensesForm1.refreshData()

End

Алгоритм роботи програми обліку фінансів реалізує повний цикл управління даними про витрати користувача - збереження, відображення, оновлення та видалення інформації у базі даних.

Після запуску форми програма ініціалізує компоненти інтерфейсу та викликає два головні методи - displayCategoryList() і displayExpenseData(). Перший підключається до бази даних і вибирає усі категорії типу «Витрата» зі статусом «Пасивний», додаючи їх у випадаючий список expense_category. Це дозволяє користувачеві вибрати відповідну категорію витрат під час заповнення форми. Другий метод завантажує список усіх наявних витрат з таблиці expenses через об’єкт ExpensesData і відображає їх у таблиці dataGridView1, де кожен рядок містить деталі - категорію, назву витрати, суму, опис і дату.

Коли користувач заповнює форму й натискає кнопку «Додати», спрацьовує обробник події expense_addBtn_Click. Він перевіряє, чи заповнені всі поля. Якщо якісь поля порожні, виводиться повідомлення про помилку. Якщо ж усі поля заповнені, відкривається з'єднання з базою даних expence.mdf, формується SQL-запит INSERT INTO expenses (...) VALUES (...), у який передаються параметри: категорія, назва товару або послуги, вартість, опис, дата витрати та поточна дата додавання. Після успішного виконання запиту програма очищує поля форми, повідомляє користувача про додавання запису та оновлює таблицю витрат на екрані.

Для очищення введених даних використовується метод clearFields(), який повертає елементи введення до початкового стану - скидає вибрану категорію, текстові поля та опис. Цей метод також викликається при натисканні кнопки «Очистити» (expense_clearBtn_Click).

Коли користувач натискає на певний рядок у таблиці (dataGridView1_CellClick), програма зчитує всі значення вибраного запису та заповнює ними форму. Одночасно зберігається у змінну getID ідентифікатор цього запису, щоб програма знала, з яким саме рядком працювати далі.

Для оновлення інформації про витрату використовується обробник expense_updateBtn_Click. Спочатку він перевіряє, чи обрано запис і чи поля не порожні. Далі відкривається з'єднання з базою даних і готується SQL-запит UPDATE expenses SET ... WHERE id = @id. Перед оновленням користувач отримує запит підтвердження. Якщо він погоджується, програма оновлює дані в таблиці expenses, очищує поля форми, повідомляє про успіх та перезавантажує список записів у таблиці.

Видалення запису відбувається через подію expense_deleteBtn_Click. Аналогічно до оновлення, програма перевіряє вибрані поля, показує повідомлення із запитом підтвердження, і якщо користувач обирає «Так», виконує SQL-команду DELETE FROM expenses WHERE id = @id. Після цього запис видаляється з бази, форма очищується, а таблиця оновлюється.

Для динамічного оновлення даних без перезапуску форми існує метод refreshData(), який при потребі викликається асинхронно через Invoke, щоб оновити список категорій і витрат у реальному часі.

Таким чином, алгоритм обліку фінансів у цій програмі можна описати як послідовність взаємодій між користувачем, формою інтерфейсу та базою даних, де:

1. Користувач вводить дані про витрати.
2. Програма перевіряє правильність введення.
3. Дані зберігаються, оновлюються або видаляються у базі SQL.
4. Відображення таблиці автоматично оновлюється для відображення актуальної інформації.

Загалом програма реалізує класичну модель CRUD (Create, Read, Update, Delete), що дозволяє вести повний контроль над усіма витратами користувача в системі обліку фінансів.

3.2. Загальна UML діаграма роботи застосунку для керування особистими фінансами

Загальна UML діаграма застосунку (рис. 3.1)

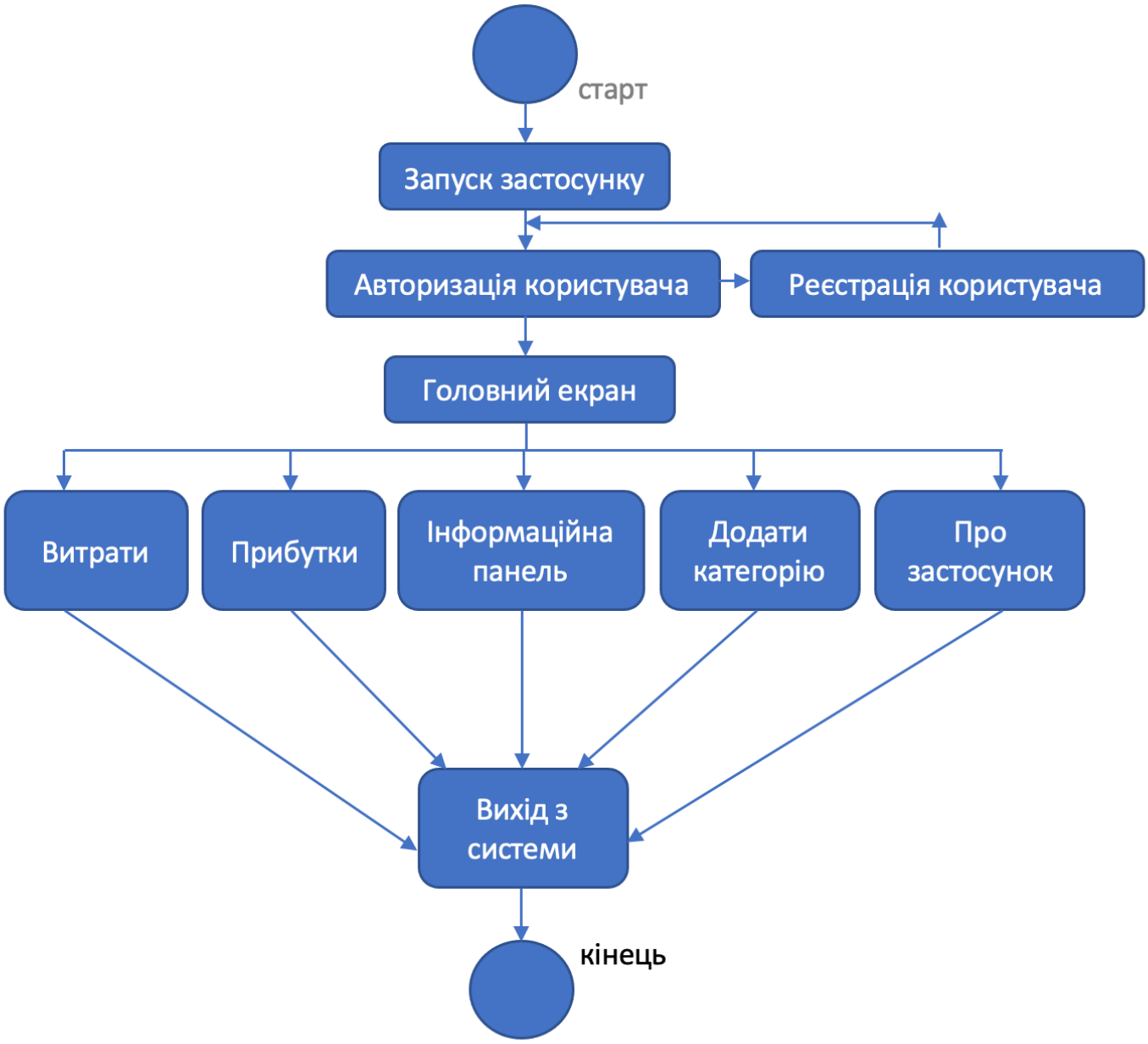

Рисунок 3.1 – Загальна UML діаграма застосунку

3.3. UML діаграма роботи коду застосунку

UML діаграма роботи коду застосунку (рис. 3.2)

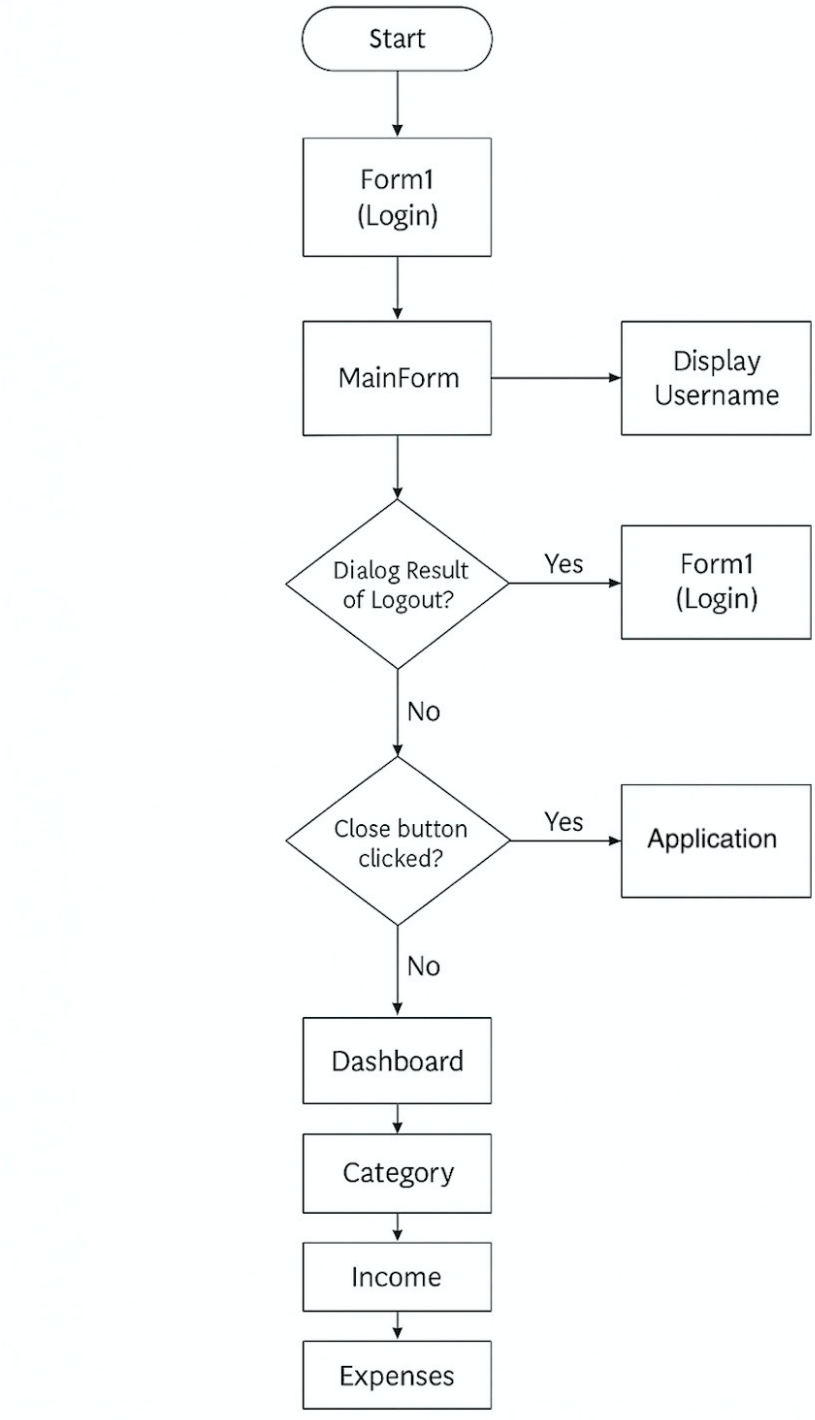

Рисунок 3.2 – UML діаграма роботи коду застосунку

4. ПРАКТИЧНА ЧАСТИНА

4.1. Обґрунтування вибору програмних засобів

Вибір мови програмування C# та середовища розробки Microsoft Visual Studio для створення програмного продукту керування особистими фінансами є цілком обґрунтованим як з технічної, так і з методологічної точки зору. Мова C# належить до сімейства сучасних об’єктно-орієнтованих мов і поєднує у собі високу продуктивність, безпечність виконання, простоту синтаксису та широкі можливості для створення застосунків будь-якої складності. Саме ці характеристики визначили доцільність її використання у процесі розробки системи обліку доходів і витрат.

По-перше, C# забезпечує повну підтримку об’єктно-орієнтованого програмування (ООП), що є надзвичайно важливим для структурованої, масштабованої та зрозумілої розробки. Використання класів, об’єктів, наслідування та інкапсуляції дало змогу розділити програму на логічні модулі - головну форму, форми обліку доходів, витрат і категорій. Такий підхід значно спрощує подальшу підтримку, налагодження та розширення функціоналу програми.

По-друге, мова C# працює в екосистемі .NET Framework, що надає потужну базу готових бібліотек, інструментів та API для розробки настільних застосунків. Завдяки використанню технології Windows Forms стало можливим створення інтуїтивного графічного інтерфейсу користувача з високим рівнем інтерактивності. Компоненти Windows Forms забезпечують просте керування елементами віконного інтерфейсу - кнопками, текстовими полями, таблицями, списками тощо. Це дозволило реалізувати зрозумілу структуру системи з чіткою логікою навігації між модулями.

По-третє, C# відзначається високим рівнем безпеки виконання коду завдяки керованому середовищу виконання (CLR - Common Language

Runtime). CLR контролює доступ до пам’яті, обробляє виняткові ситуації та запобігає типових помилок, які можуть призводити до збоїв або витоків даних. У контексті фінансового застосунку, де важливо забезпечити стабільність і точність обробки інформації, ця властивість є принципово важливою.

Крім того, Visual Studio як середовище розробки надає зручні засоби для налагодження, тестування та візуального проектування інтерфейсу. Інтегрований редактор форм дозволив створити інтерфейс швидко та ефективно, без потреби писати великий обсяг низькорівневого коду. Середовище також підтримує автоматичну генерацію подій, що значно прискорює реалізацію логіки програми.

Не менш важливою перевагою вибору C# є його кросплатформеність і масштабованість. Завдяки сучасним версіям .NET (зокрема .NET 6 і .NET 8) розроблений застосунок може бути адаптований не лише для Windows, але й для інших операційних систем, якщо в майбутньому виникне потреба в розширенні сфери його застосування.

Ще однією перевагою C# є зручність роботи з базами даних. Мова має потужні механізми доступу до SQL-баз через ADO.NET, LINQ або Entity Framework. Це дозволяє ефективно реалізовувати збереження, пошук і оновлення фінансових даних користувача. Таким чином, програмне середовище C# надає оптимальний баланс між зручністю розробки, продуктивністю і безпечністю роботи з інформацією.

Також варто відзначити, що C# має високу популярність у професійній спільноті, активно підтримується Microsoft і має велику базу навчальних матеріалів, прикладів та бібліотек. Це робить його перспективним вибором для навчальних і науково-дослідних проєктів, зокрема для магістерських робіт.

4.2. Опис програмної реалізації

Застосунок керування особистими фінансами розроблено на основі мові програмування C# [9, 10], з використанням редактору коду Visual Studio. Перший крок розробки – створення нового проекту. (рис. 4.1)

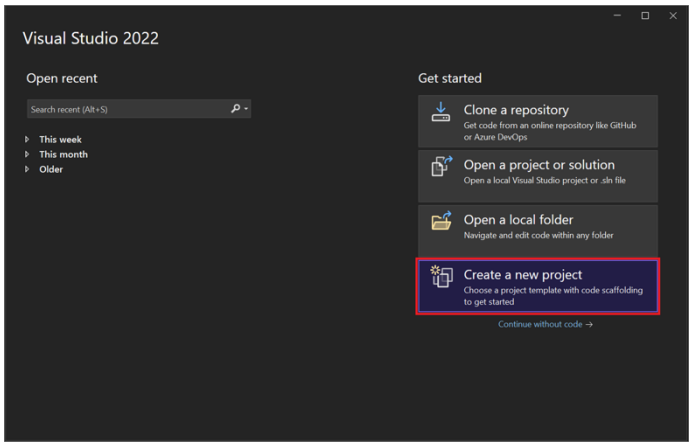

Рисунок 4.1 – Новий проект

Потім розробляється зовнішній вигляд вікна входу у систему. За основу використаний мінімалістичний стиль вікна. (рис. 4.2)

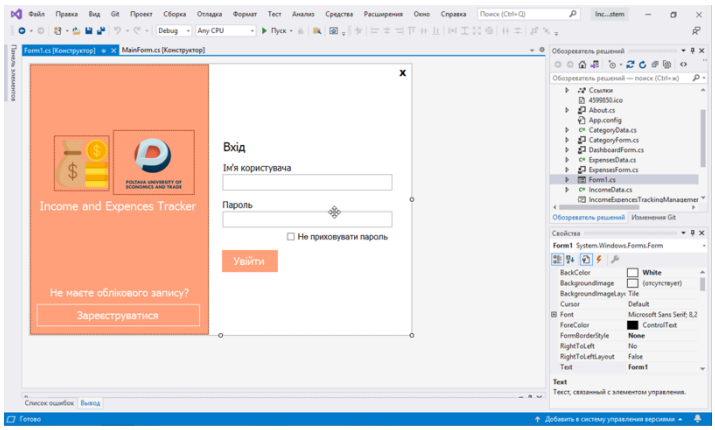

Рисунок 4.2 – Мінімалістичний дизайн вікна входу

Після вікна входу, було розроблено вікно реєстрації користувача. (рис. 4.3)

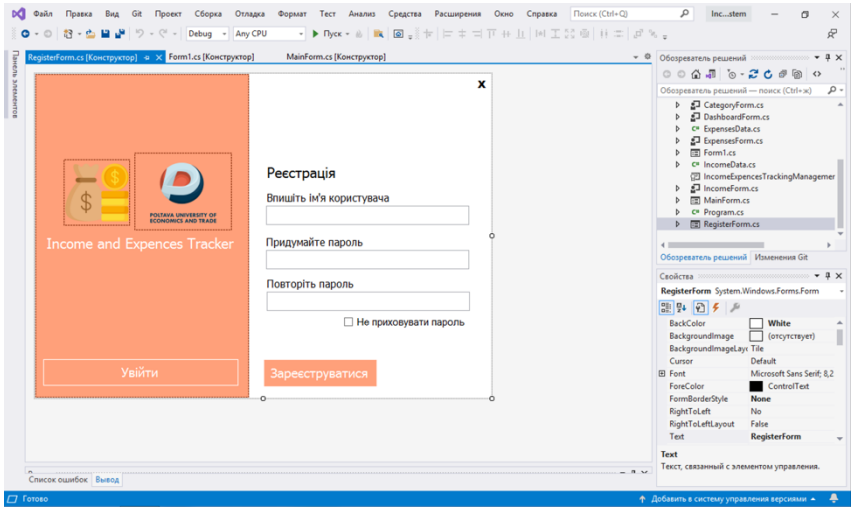

Рисунок 4.3 – Розробка вікна реєстрації

Далі розроблено головний екран застосунку керування особистими фінансами з відображенням прибутків та витрат, а також боковою панелькою керування. (рис. 4.4)

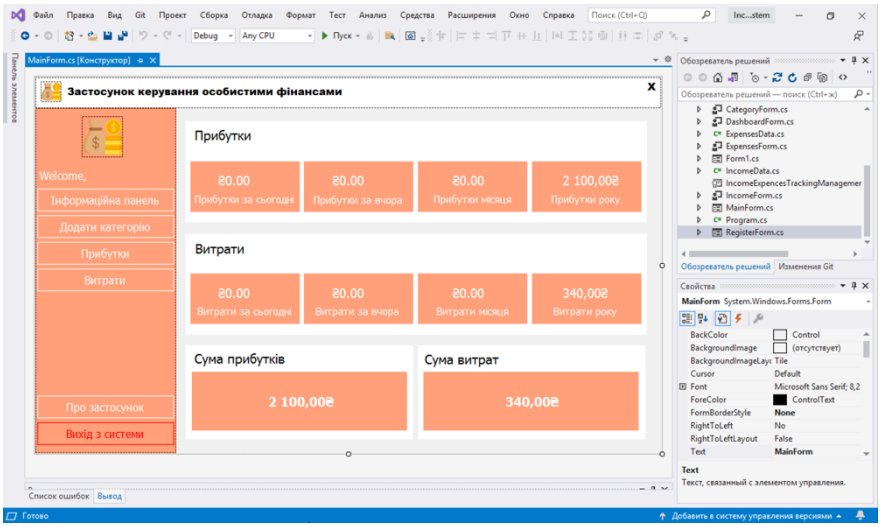

Рисунок 4.4 – Головний екран застосунку

Вікно для додавання категорій програмується окремо, всі вікна які користувач буде запускати з бокової панельки – будуть вкладені всередину головного вікна. (рис. 4.5)

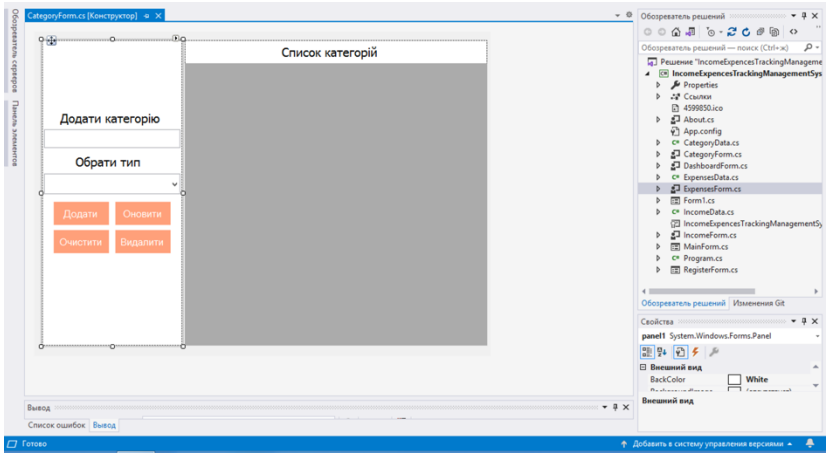

Рисунок 4.5 – Вікно додавання категорій

Після вікна додавання категорій, розробляється вікно з списком прибутків, тут у користувача є можливість відразу обрати категорію та вписати необхідні деталі прибутків. Також додано функцію копіювання записів. (рис. 4.6)

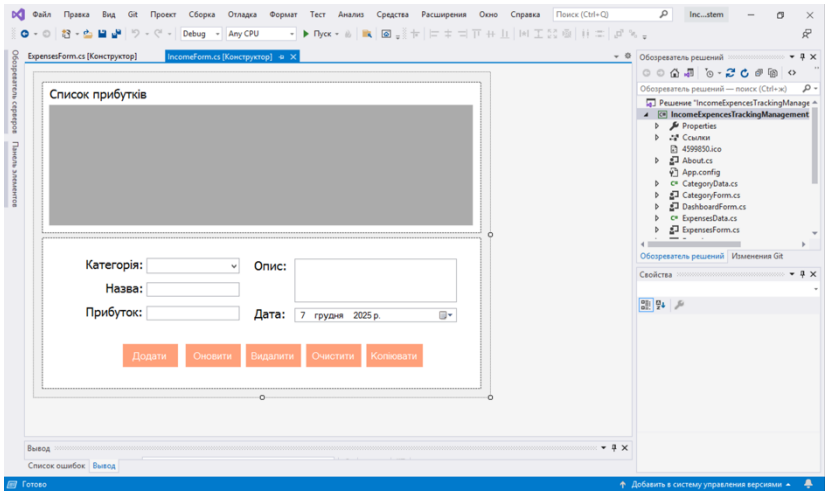

Рисунок 4.6 – Вікно з списком прибутків

Потім було розроблено вікно з витратами, тут користувач також обирає потрібну категорію та заповнює деталі витрати. Для зручності всі витрати також відображаються у вигляді списку. Тут теж додано копіювання записів. (рис. 4.7)

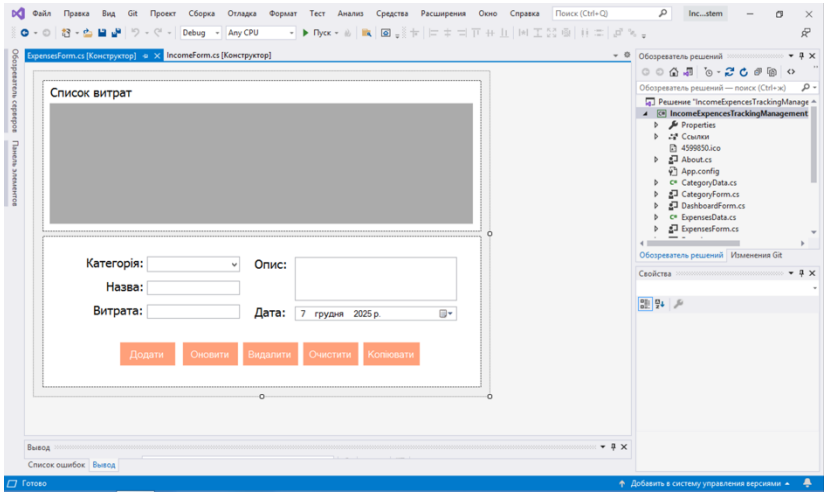

Рисунок 4.7 – Вікно з витратами

Додатково створено інформаційне вікно у такому ж мінімалістичному єдиному стилі програмного забезпечення. (рис. 4.8)

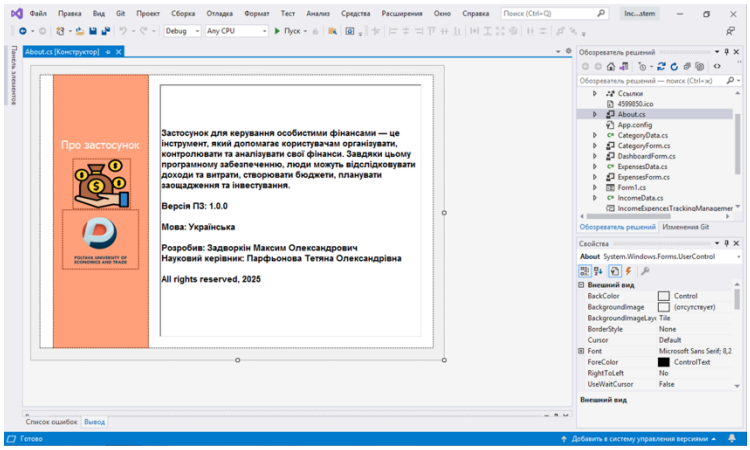

Рисунок 4.8 – Інформаційне вікно

Для зберігання даних користувачів, категорій, витрат та прибутків використовується база даних SQL, було створено декілька таблиць. (рис. 4.9)

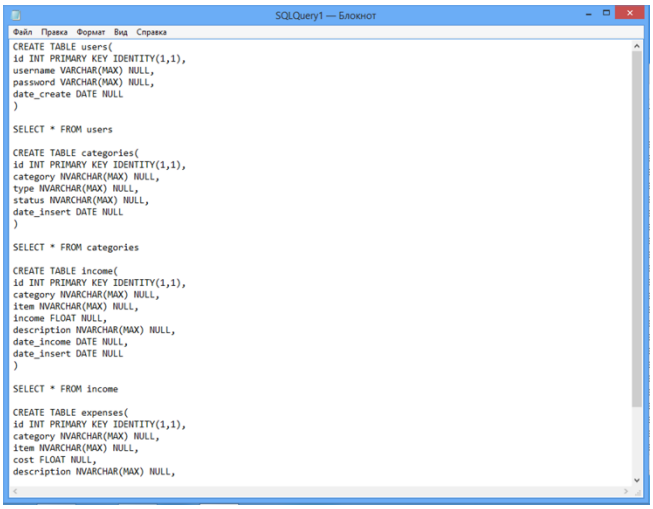

Рисунок 4.9 – Створення таблиць SQL

Для правильності обробки вхідних даних користувача було окремо створено клас для вікна з прибутками та витратами. Цей клас обробляє вхідні дані та передає їх до бази даних SQL. (рис. 4.10)

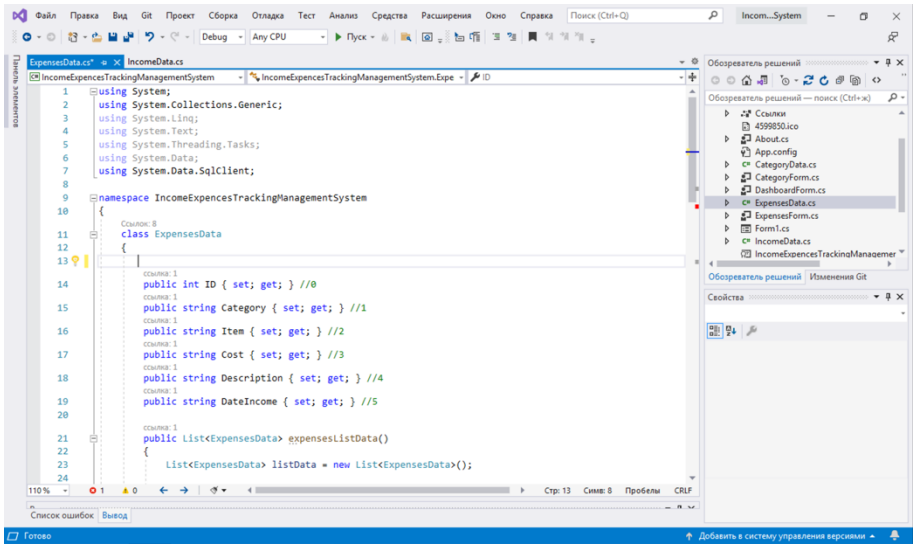

Рисунок 4.10 – Клас обробки даних вікна прибутків і витрат

Наступний важливий крок – прописування коду для правильної роботи всіх компонентів застосунку та відображення інформації у списках та на головній панелі. (рис. 4.11)

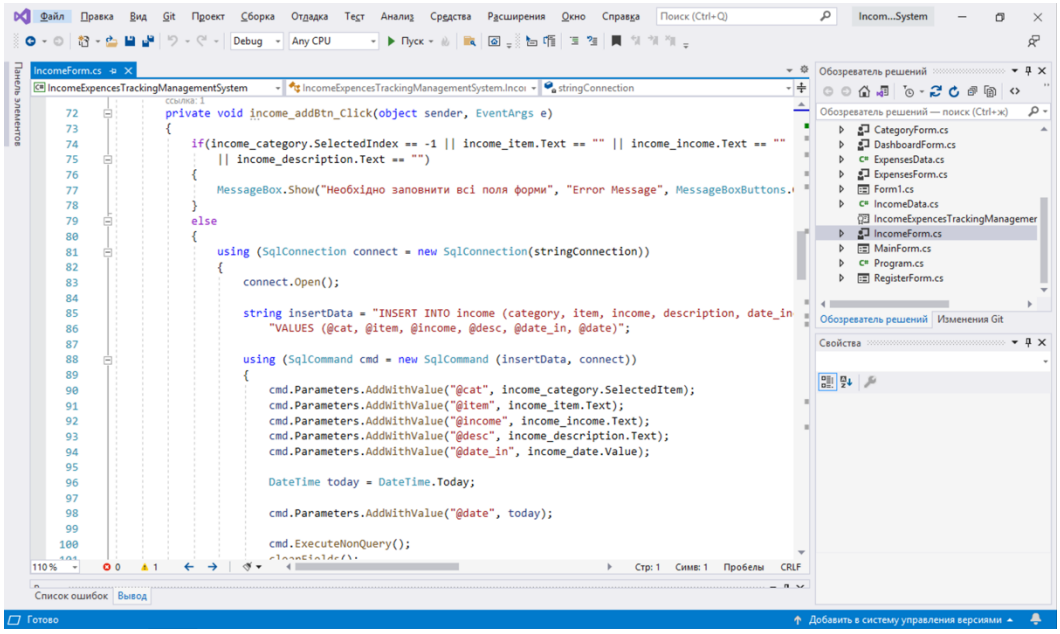

Рисунок 4.11 – Процес програмування застосунку

Код було написано за допомогою середовища розробки Microsoft Visual Studio 2019 із використанням мови програмування C# та технології Windows Forms. Visual Studio 2022 надала інструменти для графічного проєктування інтерфейсу, автоматичної генерації коду елементів форми та зручної прив’язки подій. У програмі використано стандартні бібліотеки .NET Framework, зокрема System.Data.SqlClient для роботи з базою даних SQL Server LocalDB, що забезпечило можливість реалізувати додавання, редагування, видалення та відображення даних у застосунку.

4.3. Опис роботи застосунку для керування особистими фінансами

Після запуску застосунку для керування особистими фінансами користувача зустрічає вікно входу в систему. Користувачу необхідно вписати ім’я та пароль, якщо облікового запису ще немає – необхідно зареєструвати. (рис. 4.12)

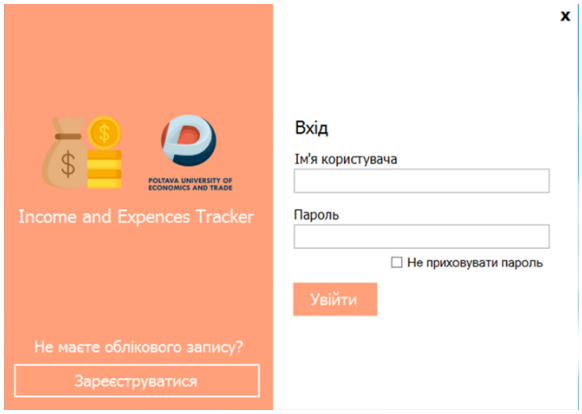

Рисунок 4.12 – Вікно входу

Якщо натиснути кнопку реєстрації, користувачу відкриється вікно реєстрації, потім необхідно вписати ім’я користувача та придумати пароль. (рис. 4.13)

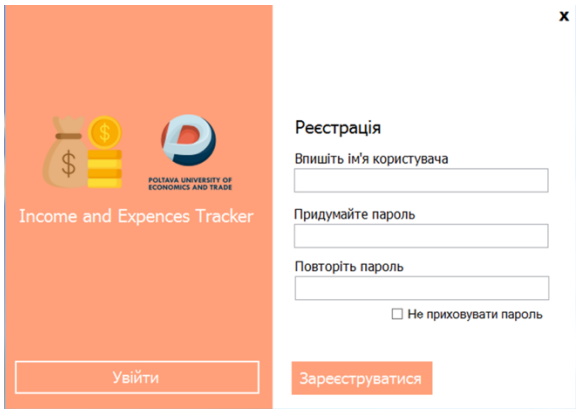

Рисунок 4.13 – Вікно реєстрації

Після успішного входу в систему, користувача зустрічає інформаційна панель, на якій виведено прибутки і витрати за сьогодні, вчора, прибутки і витрати за місяць, а також за рік. Окремо виведено суму усіх прибутків та витрат. (рис. 4.14)

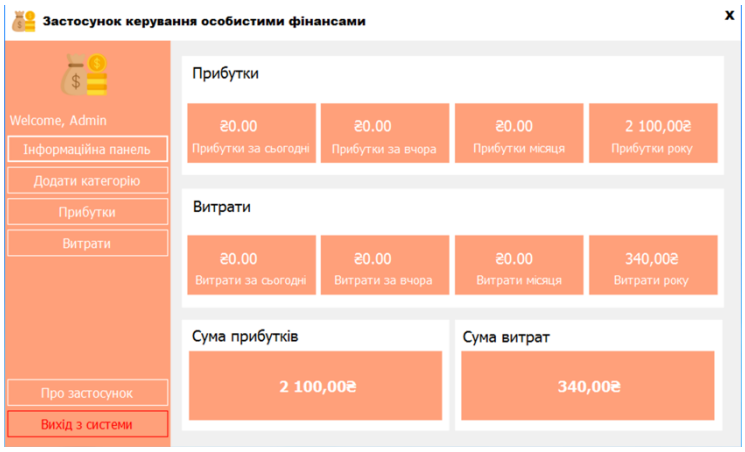

Рисунок 4.14 – Інформаційна панель

Якщо користувач у лівій панелі кнопок натисне кнопку «Про застосунок», користувачу висвітиться вкладене вікно з інформацією про розроблений застосунок. (рис. 4.15)

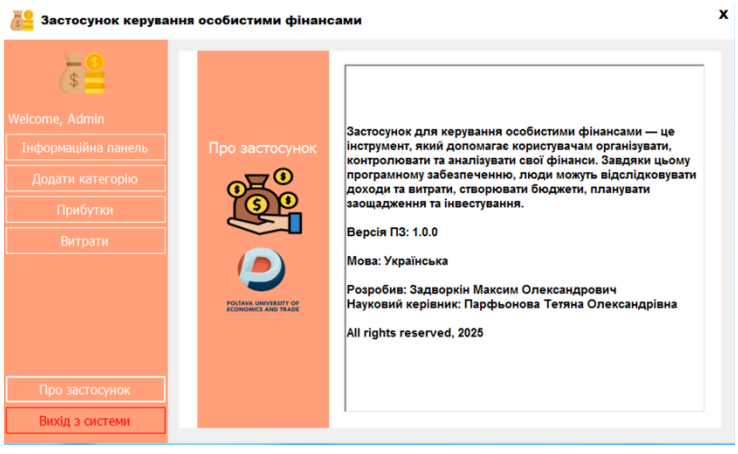

Рисунок 4.15 – Інформаційне вікно застосунку

Якщо користувач у лівому блоці кнопок натисне кнопку «Додати категорію», висвітиться вкладене вікно з можливістю вписати назву категорії і обрати тип. Після заповнення користувач може додати цей запис і він додається до списку, також можна оновити запис з списку, видалити запис з списку або очистити поля заповнення категорії. (рис. 4.16)

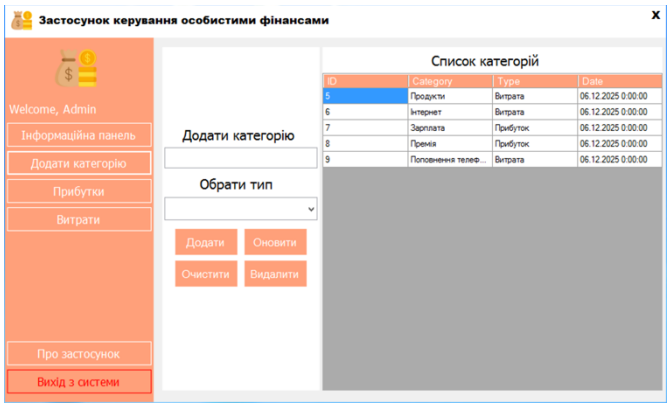

Рисунок 4.16 – Список категорій

При кожній взаємодії з кнопками списку категорій, користувача супроводжують інформаційні вікна для додаткового інформування або підказок. (рис. 4.17)

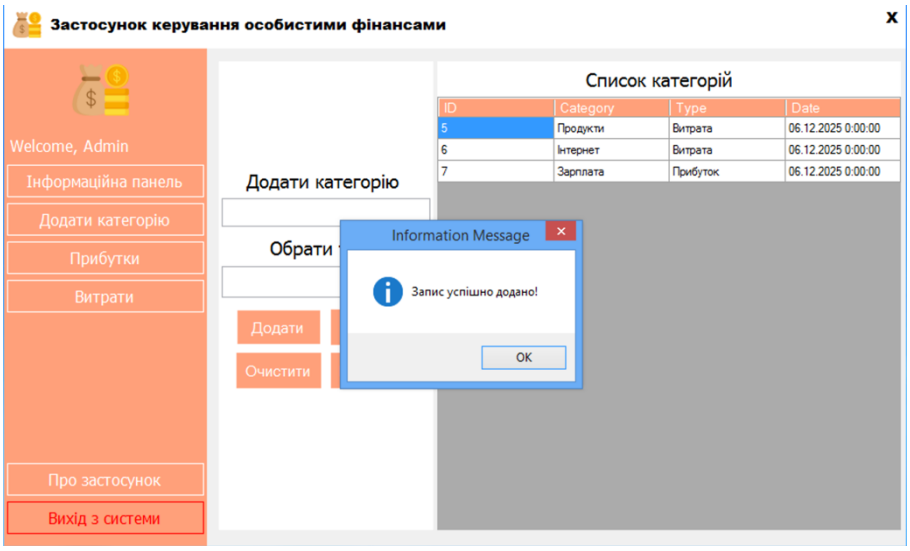

Рисунок 4.17 – Інформаційне вікно

Якщо користувач у лівому блоці кнопок натисне кнопку «Прибутки», висвітиться вкладене вікно з списком прибутків. Тут користувач може обрати раніше вписану категорію, вписати назву і суму прибутку, написати додатковий опис, заповнити дату. Також є можливість додати, оновити, видалити записи, додатково очистити поля. (рис. 4.18)

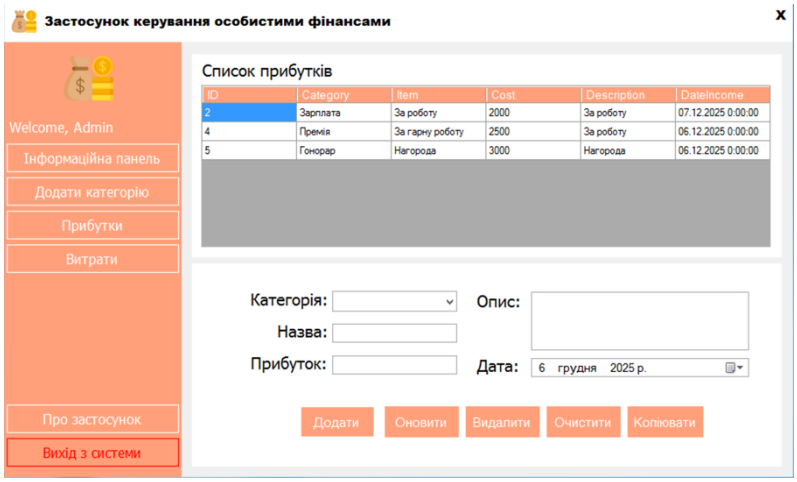

Рисунок 4.18 – Вікно з списком прибутків

Якщо користувач у лівому блоці кнопок натисне кнопку «Витрати», висвітиться вкладене вікно з списком витрат. Тут користувач аналогічно може обрати раніше вписану категорію витрат, вписати назву і суму витрати, написати додатковий опис, заповнити дату. Тут теж є можливість додати, оновити, видалити записи, додатково очистити поля. (рис. 4.19)

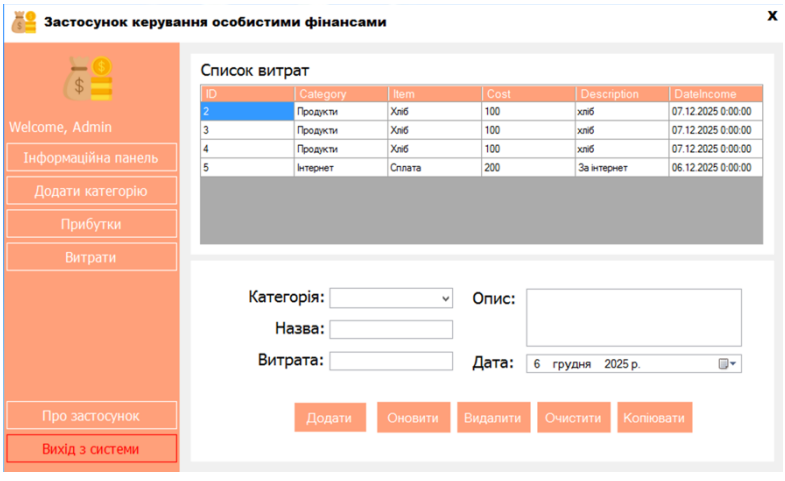

Рисунок 4.19 – Вікно з списком витрат

При кожній взаємодії з кнопками списку прибутків або витрат, користувача супроводжують інформаційні вікна для додаткового інформування або підказок. (рис. 4.20)

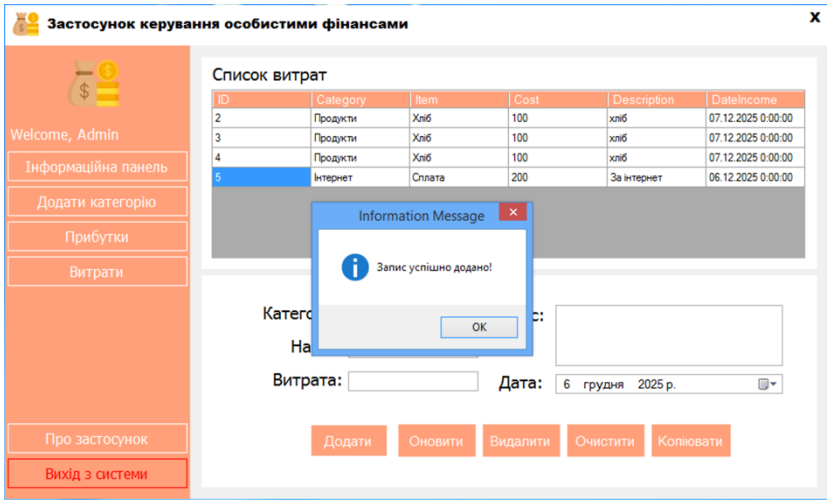

Рисунок 4.20 – Інформаційне вікно додавання запису до списку

При оновленні запису необхідно обрати з списку потрібний запис та редагувати його у полях, потім натиснути кнопку “Оновити”, спливаюче вікно як зворотній відгук успіху операції. (рис. 4.21)

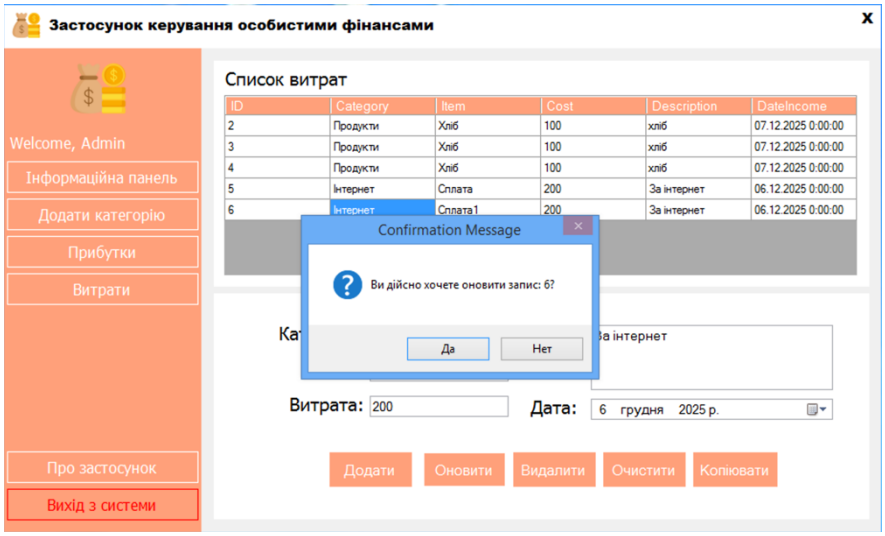

Рисунок 4.21 – Сповіщення про оновлення запису

Для видалення запису з списків витрат або прибутків, користувачу необхідно спочатку обрати потрібний запис, потім редагувати його і натиснути кнопку “Видалити”, висвітиться спливаюче вікно з підтвердженням видалення запису. (рис. 4.22)

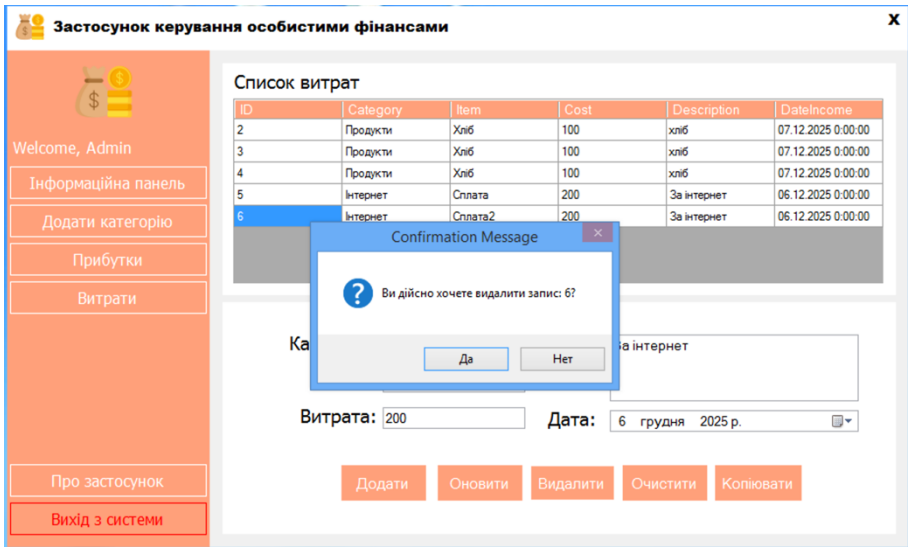

Рисунок 4.22 – Інформаційне вікно видалення запису

Якщо користувач погодився видалити запис, йому додатково висвітиться інформаційне вікно про успішне видалення запису. (рис. 4.23)

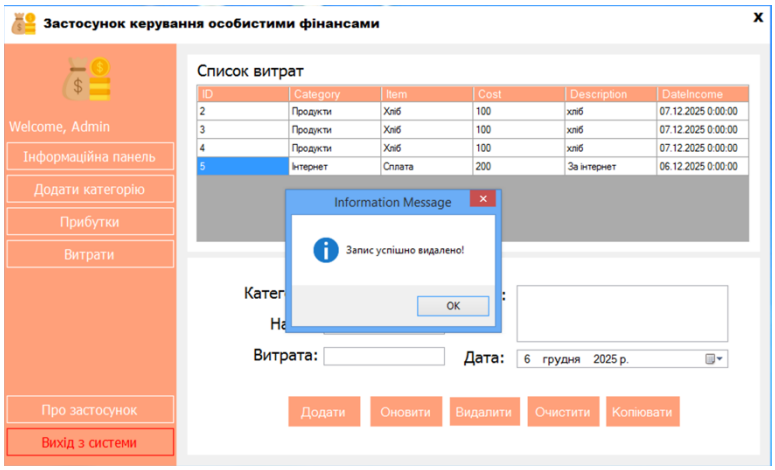

Рисунок 4.23 – Інформаційне вікно успішного видалення запису

Якщо користувач у лівому меню головного вікна натисне кнопку “Вихід з системи”, йому додатково висвітиться вікно з підтвердженням. (рис. 4.24)

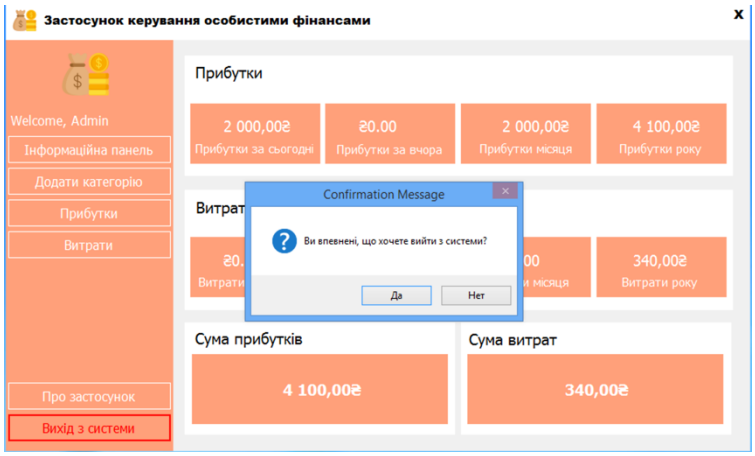

Рисунок 4.24 – Підтвердження виходу з системи

Також у застосунку оброблено багато помилок, якщо у вікні логіну помилитись з паролем – система виведе повідомлення. (рис. 4.25)

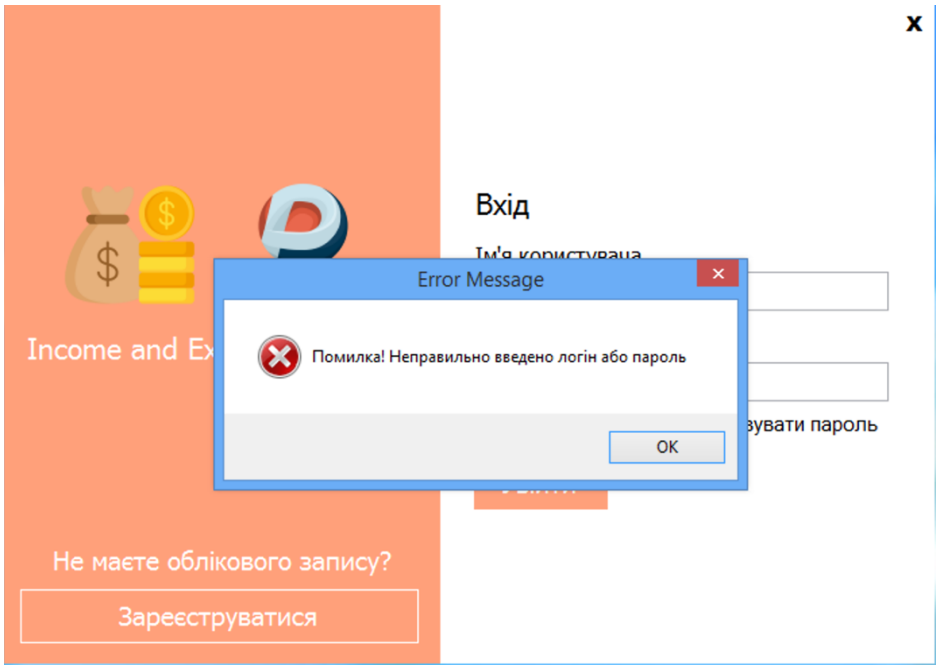

Рисунок 4.25 – Обробка помилки введення логіну або паролю

Якщо користувач не заповнив якесь поле або всі поля у будь якому вікні – система виведе помилку пустих полів. (рис. 4.26)

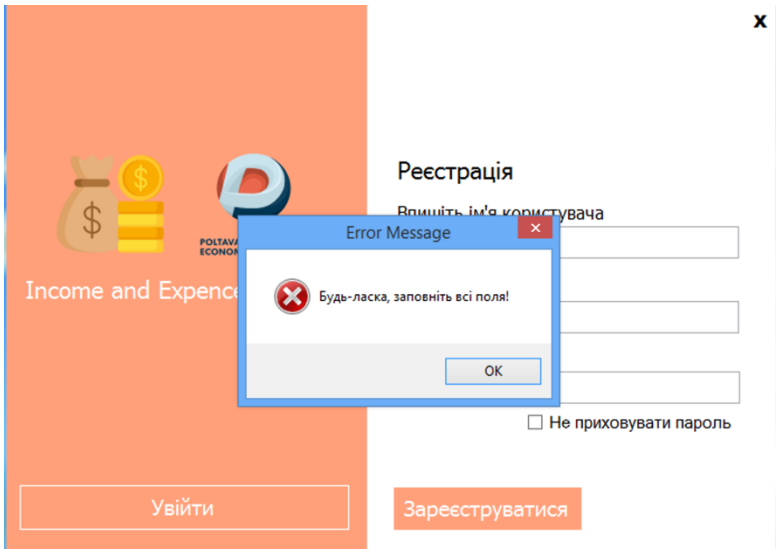

Рисунок 4.26 – Помилка пусті поля

Якщо користувач не заповнив поля і не обрав відповідь у випадаючому списку у вікні категорій, він отримає помилку. (рис. 4.27)

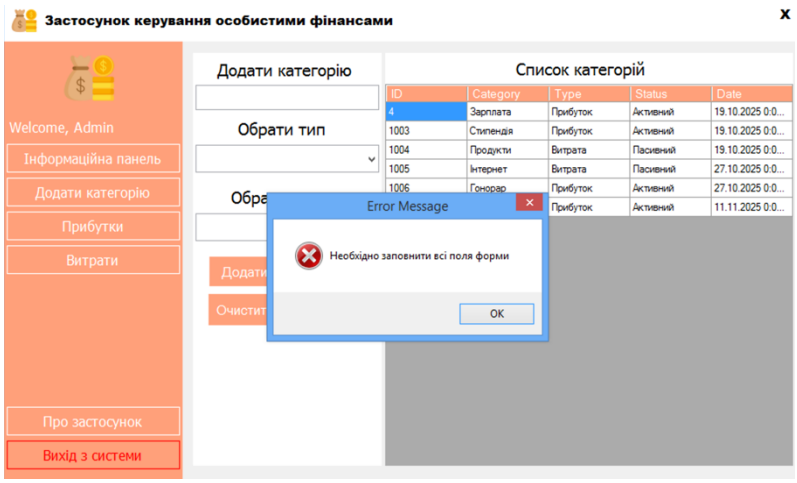

Рисунок 4.27 – Помилка пустих полів вікна категорій

Якщо користувач не обрав запис з списку категорій і натиснув кнопки “оновити” або “видалити”, він отримає помилку і підказку (рис. 4.28)

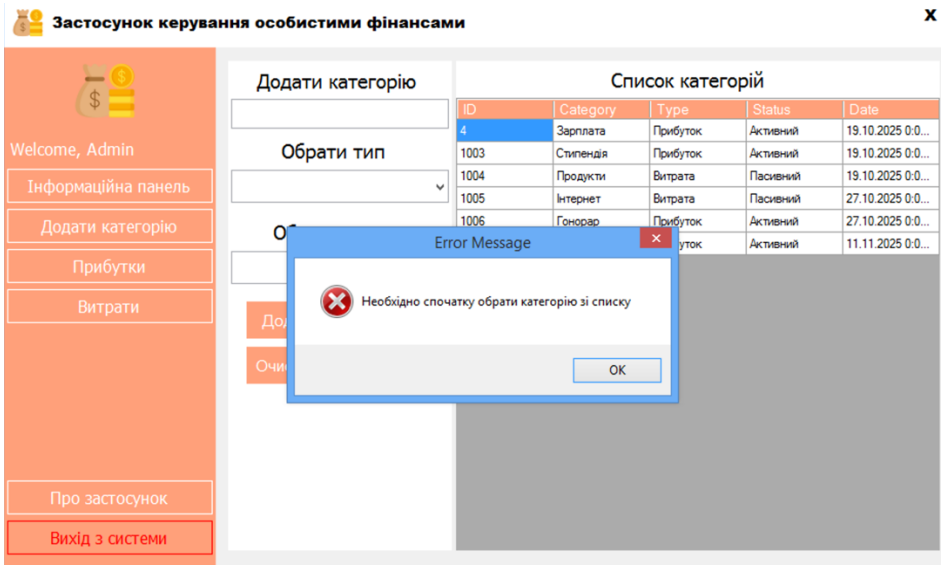

Рисунок 4.28 – Вивід помилки і підказки

Якщо користувач не заповнив поля у вікні прибутків, система виведе помилку. (рис. 4.29)

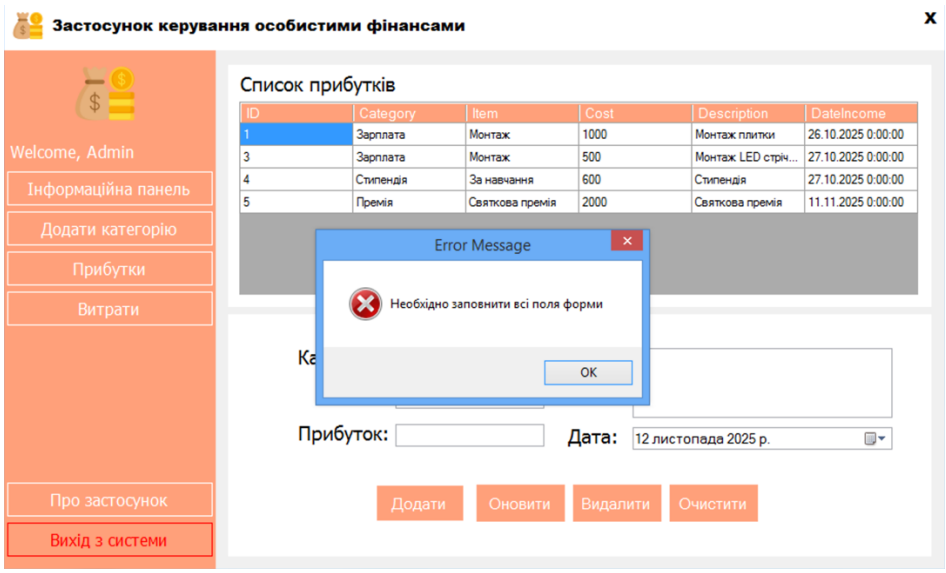

Рисунок 4.29 – Виведення помилки

Якщо користувачеві потрібно скопіювати декілька записів – така можливість є, вибравши запис з списку і натиснувши кнопку “Копіювати”. (рис. 4.30)

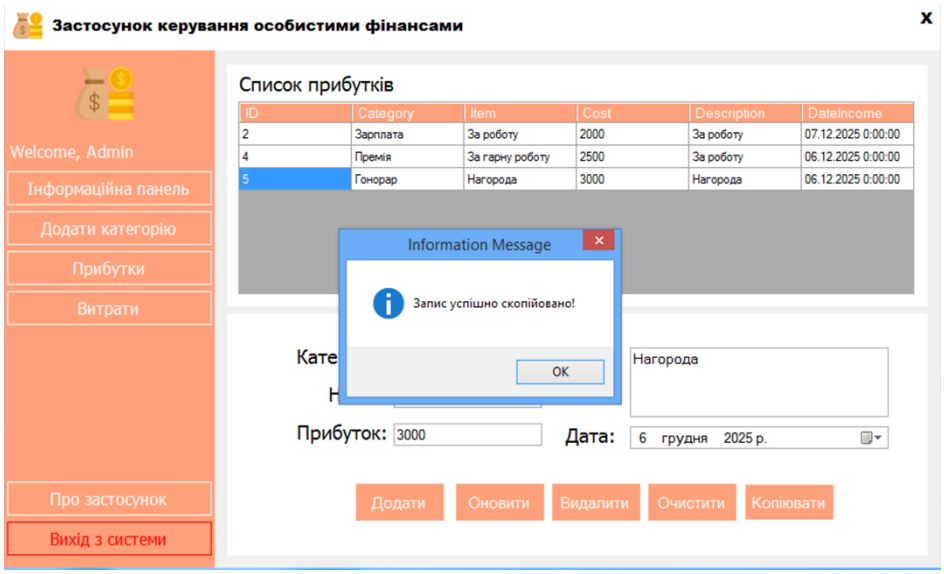

Рисунок 4.30 – Копіювання записів

Аналогічно додано кнопку “Копіювати” і до форми з витратами, потрібно лише обрати запис для копіювання. (рис. 4.31)

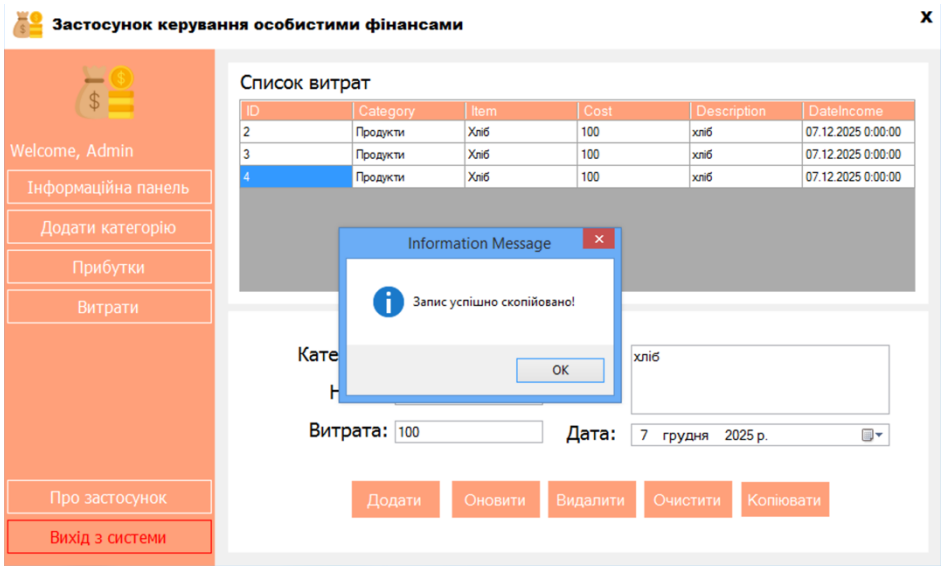

Рисунок 4.31 – Копіювання записів витрат

Для роботи застосунку, обов’язково треба мати встановлене програмне забезпечення SQL local database. (рис. 4.29)

Рисунок 4.29 – SQL local database

Загалом, підводячи підсумки огляду, інтерфейс програми характеризується простотою, логічністю та зручністю, а функціонал - повнотою і практичною ефективністю. Програма дозволяє виконувати всі необхідні операції з управління особистими фінансами, що робить її зручною у щоденному використанні. Поєднання продуманого дизайну, чіткої логіки обробки подій та стабільної роботи забезпечує високу якість користувацького досвіду та підтверджує успішність реалізованої системи.

ВИСНОВКИ

Отже, у результаті розробки та аналізу застосунку для керування особистих фінансів було створено повнофункціональний програмний продукт для обліку доходів і витрат користувача, який забезпечує зручну, наочну та ефективну взаємодію із фінансовими даними. Програму реалізовано мовою C# із використанням середовища Windows Forms, що дозволило поєднати логіку обробки даних з інтуїтивним графічним інтерфейсом користувача.

Розроблена система передбачає модульну структуру, що складається з кількох основних форм - головний екран, категорії, прибутку, витрати. Такий підхід забезпечує логічне розмежування функціоналу, простоту навігації та масштабованість проєкту. Головна форма виконує роль центрального елемента керування, з якого користувач може переходити між різними модулями, здійснювати вхід, вихід із системи або завершення роботи програми.

Під час розробки реалізовано:

1. Механізм авторизації користувача та відображення його імені в інтерфейсі;
2. Систему обробки подій, натискання кнопок, підтвердження дій, вихід з облікового запису;
3. Оновлення даних у кожному модулі через виклик спеціалізованих методів, що забезпечує актуальність інформації;
4. Зручну систему керування вкладками, де користувач одночасно бачить лише один активний розділ, що підвищує зручність роботи та запобігає перевантаженню інтерфейсу.

Практична цінність розробленої системи полягає в тому, що вона може бути використана як особистий фінансовий менеджер для обліку витрат та доходів, аналізу бюджету та контролю фінансової поведінки

користувача. Вона може бути розширена під потреби малого бізнесу або адаптована для корпоративного використання.

Створена програма демонструє застосування об’єктно-орієнтованого підходу, ефективне використання подійної моделі Windows Forms і забезпечує повноцінну реалізацію системи управління фінансами. Отримані результати підтверджують доцільність обраних методів, технологій і структурних рішень, а сама розробка може бути основою для подальшого вдосконалення та впровадження у практичні проєкти.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ольховська О.В., Черненко О. О. методичні рекомендації до виконання кваліфікаційної роботи для студентів спеціальності 122 Комп'ютерні науки освітня програма «Комп'ютерні науки» ступеня бакалавра / О. О. Черненко., Ольховська О.В. – Полтава : ПУЕТ, 2025. – 58 с.
2. Вступні іспити в магістратуру [Електронний ресурс]. – Режим доступу: https://osvita.ua/master/umovy/64981/
3. Огляд застосунку керування особистими фінансами Monefy [Електронний ресурс]. – Режим доступу: https://play.google.com/store/apps/details?id=com.monefy.app.lite&hl=ua&gl=US
4. Огляд застосунку Moneygraph+ [Електронний ресурс]. – Режим доступу: https://apps.microsoft.com/detail/9nblgggz5815?hl=uk-ua&gl=UA
5. Розробка програмного забезпечення [Електронний ресурс]. – Режим доступу: https://studfile.net/preview/5118185
6. Системне програмне забезпечення [Електронний ресурс]. – Режим доступу: https://studfile.net/preview/2426774/page:10/
7. C# Підручник [Електронний ресурс]. – Режим доступу: https://www.google.com/url?sa=t&source
8. Категорія: Мова програмування C# [Електронний ресурс]. – Режим доступу: https://abitap.com/category/c/#google_vignette
9. What is C# and use cases of C#? [Електронний ресурс]. – Режим доступу: https://www.devopsschool.com/blog/what-is-c#-net-and-use-cases-of-c#-net/
10. Створення програми в Visual Studio 2019 [Електронний ресурс]. – Режим доступу: https://learn.ztu.edu.ua/mod/page/view.php?id=9976
11. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання: ДСТУ 7.1-2006. – [Чинний від 2007-07-01]. – К. : Держспоживстандарт України, 2007. – 47 с.

12. Microsoft Visual Studio [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Microsoft_Visual_Studio
13. development with Visual Studio [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/visualstudio/get-started/csharp/?view=vs-2022
14. Get Started [Електронний ресурс]. – Режим доступу: https://www.w3schools.com/cs/cs_getstarted.php
15. Visual Studio [Електронний ресурс]. – Режим доступу: https://www.halvorsen.blog/documents/programming/csharp/csharp.php
16. Створення програми в Visual Studio 2019 [Електронний ресурс]. – Режим доступу: https://learn.ztu.edu.ua/mod/page/view.php?id=9976
17. Microsoft Visual Studio [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Microsoft_Visual_Studio
18. Навіщо потрібні програми для обліку [Електронний ресурс]. – Режим доступу: https://smartoblik.com/nav%D1%96shcho_potr%D1%96bn%D1%96_prohramy_dlya_obliku/
19. Об'єктно-орієнтоване програмування (Advanced Encryption Standard instructions) [Електронний ресурс]. – Режим доступу: https://vseosvita.ua/test/elementy-teorii-obiektno-oriientovanoho.
20. Методологія та організація ведення обліку фінансів [Електронний ресурс]. – Режим доступу: https://magazine.faaf.org.ua/metodologiya-ta-organizaciya-vedennya-obliku-v-umovah-avtomatizacii.html

ДОДАТОК А. КОД ПРОГРАМИ