Міністерство освіти і науки України

Національний аерокосмічний університет ім. М. Є. Жуковського

«Харківський авіаційний інститут»

|  |  |  |
| --- | --- | --- |
| Факультет | *радіоелектроніки, комп’ютерних систем та інфокомунікацій* | |
|  | (повне найменування) | |
| Кафедра | *комп’ютерних систем, мереж і кібербезпеки* | |
|  | (повне найменування) | |
| Рівень вищої освіти | | *перший (бакалаврський)* |
|  | | (назва) |
| Напрям підготовки | | *123 – комп’ютерна інженерія* |
|  | | (код та найменування) |

**ЗАВДАННЯ**

**НА КУРСОВИЙ ПРОЄКТ (РОБОТУ) ЗДОБУВАЧУ**

|  |  |  |
| --- | --- | --- |
| *Гуща Володимир Володимирович* | | |
| (прізвище, ім’я, по батькові) | | |
| 1. Тема проєкту (роботи) | *Структурні підрозділи банків України* | |
|  | | |
|  | | |
| Керівник проєкту (роботи) | | *Бабешко Євген Васильович* |
|  | | ( прізвище, ім’я, по батькові, науковий ступінь, вчене звання) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2. Строк подання здобувачем проєкту (роботи) | | | *29.05.2024* | |
| 3. Вихідні дані до проєкту (роботи) | |  | | |
|  | | | | |
|  | | | | |
|  | | | | |
|  | | | | |
| 4. Зміст пояснювальної записки (перелік завдань, які потрібно розв’язати) | | | | *Збір* |
| *вихідних даних та аналіз вимог , розробка технічного завдання, проектування* | | | | |
| *архітектури системи, розробка програмного забезпечення, тестування* | | | | |
| *програмного забезпечення, написання пояснювальної записки підготовка до* | | | | |
| *Захисту проекту* | | | | |
|  | | | | |
|  | | | | |
| 5. Перелік графічного матеріалу | *Блок-схема архітектури системи* | | | |
| *Діаграма випадків використання, Діаграма класів (Class Diagram)* | | | | |
| *Діаграма послідовностей (Sequence Diagram)* | | | | |
| *Діаграма станів (State Diagram)* | | | | |
|  | | | | |

6. Консультанти розділів проєкту (роботи)

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Прізвище, ініціали та посада  консультанта | Підпис, дата | |
| завдання видав | завдання  прийняв |
| *Основний* |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Нормоконтроль |  |  | *«* |  | *»* |  | 20 |  | *р* |
|  | (підпис) | (ім’я та прізвище) |  |  |  |  |  |  |  |

7. Дата видачі завдання «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_20*\_\_* р.

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів курсового  проєкту (роботи) | Строк виконання етапів проєкту  ( роботи ) | Примітка |
| ***1*** | ***Збір вихідних даних і матеріалів*** | ***01.01.2024 - 15.01.2024*** |  |
| ***2*** | ***Аналіз вихідних даних*** | ***16.01.2024 - 31.01.2024*** |  |
| ***3*** | ***Розробка технічного завдання*** | ***01.02.2024 - 15.02.2024*** |  |
| ***4*** | ***Розробка структури проекту*** | ***16.02.2024 - 29.02.2024*** |  |
| ***5*** | ***Написання теоретичної частини*** | ***01.03.2024 - 15.03.2024*** |  |
| ***6*** | ***Розробка програмного забезпечення*** | ***16.03.2024 - 15.04.2024*** |  |
| ***7*** | ***Тестування програмного забезпечення*** | ***16.04.2024 - 30.04.2024*** |  |
| ***8*** | ***Написання звіту по проекту*** | ***01.05.2024 - 15.05.2024*** |  |
| ***9*** | ***Перевірка та внесення правок у звіт*** | ***16.05.2024 - 19.05.2024*** |  |
| ***10*** | ***Підготовка до захисту*** | ***20.05.2024*** |  |

|  |  |  |
| --- | --- | --- |
| **Здобувач** |  |  |
|  | (підпис) | (ім’я та прізвище) |
| **Керівник проєкту (роботи)** |  |  |
|  | (підпис) | (ім’я та прізвище) |

**Реферат**

**Структурні підрозділи банків України**

У цій курсовій роботі досліджено структуру банківських установ України та їх підрозділів. Основна увага приділяється аналізу функцій та взаємодії різних підрозділів, таких як відділи кредитування, управління ризиками, фінансового аналізу та маркетингу. Робота націлена на виявлення впливу організаційної структури на ефективність роботи банківських установ.

**Мета роботи**

Метою дослідження є визначення основних типів структурних підрозділів банків України, їх функцій та впливу на загальну діяльність банків. Також метою є розробка програми для управління банківськими підрозділами з можливістю зберігання, додавання та отримання інформації про банки.

**Методи дослідження**

Для досягнення поставленої мети було використано методи аналізу літератури, статистичні методи, порівняльний аналіз та методи моделювання банківських процесів. Також була розроблена програма на мові програмування C#, яка дозволяє зберігати та обробляти інформацію про банки.

**Результати дослідження**

В результаті дослідження були визначені основні типи структурних підрозділів банків, їх функції та взаємодія між ними. Було виявлено, що ефективне управління цими підрозділами значно впливає на загальну продуктивність банківської установи. Також було розроблено програму, яка дозволяє зберігати інформацію про банки, додавати нові банки до системи, отримувати інформацію про банк за його кодом та відображати її у зручному форматі.

**Висновки**

На основі проведеного аналізу зроблено висновок, що структурні підрозділи банків є ключовими елементами у забезпеченні стабільної та ефективної роботи банківських установ. Оптимізація їх структури та взаємодії може суттєво покращити фінансові результати банків, зменшити ризики та підвищити рівень обслуговування клієнтів. Розроблена програма демонструє практичну реалізацію ефективного управління банківськими підрозділами.

**Практичне значення**

Результати цього дослідження можуть бути використані для вдосконалення організаційної структури банківських установ в Україні, що сприятиме підвищенню їх конкурентоспроможності на фінансовому ринку. Розроблена програма може бути використана як інструмент для управління інформацією про банки та їх підрозділи.

**Ключові слова**

Банківські підрозділи, банківська структура, управління банками, ефективність банківських установ, фінансовий аналіз, кредитування, управління ризиками, C# програма, управління інформацією.

**Перелік скорочень**

1. **Банк** - Банківська установа.
2. **МФО** - Міжнародний фінансовий організаційний код.
3. **IT** - Інформаційні технології.
4. **API** - Application Programming Interface (Інтерфейс програмування додатків).
5. **C#** - C-Sharp (Мова програмування, розроблена компанією Microsoft).
6. **GUI** - Graphical User Interface (Графічний інтерфейс користувача).
7. **MVC** - Model-View-Controller (Модель-Вид-Контролер, шаблон проектування).
8. **EF Core** - Entity Framework Core (Фреймворк для роботи з базами даних).
9. **JSON** - JavaScript Object Notation (Формат обміну даними).
10. **SQL** - Structured Query Language (Мова структурованих запитів).
11. **ASP.NET** - Фреймворк для розробки веб-додатків.
12. **NuGet** - Менеджер пакетів для .NET.
13. **NUnit** - Інструмент для модульного тестування.
14. **IDE** - Integrated Development Environment (Інтегроване середовище розробки).
15. **CLI** - Command Line Interface (Інтерфейс командного рядка).
16. **ORM** - Object-Relational Mapping (Об'єктно-реляційне відображення).
17. **DBMS** - Database Management System (Система управління базами даних).
18. **CRUD** - Create, Read, Update, Delete (Створення, Читання, Оновлення, Видалення).
19. **UI** - User Interface (Інтерфейс користувача).
20. **UX** - User Experience (Досвід користувача).
21. **HTTP** - HyperText Transfer Protocol (Протокол передачі гіпертексту).
22. **HTTPS** - HyperText Transfer Protocol Secure (Безпечний протокол передачі гіпертексту).
23. **TCP/IP** - Transmission Control Protocol/Internet Protocol (Протокол управління передачею/Інтернет-протокол).
24. **SSL** - Secure Sockets Layer (Шар захищених сокетів).
25. **TLS** - Transport Layer Security (Безпека транспортного шару).
26. **FTP** - File Transfer Protocol (Протокол передачі файлів).
27. **SMTP** - Simple Mail Transfer Protocol (Простий протокол передачі пошти).
28. **DNS** - Domain Name System (Система доменних імен).
29. **IP** - Internet Protocol (Інтернет-протокол).
30. **IPv4** - Internet Protocol version 4 (Інтернет-протокол версія 4).
31. **IPv6** - Internet Protocol version 6 (Інтернет-протокол версія 6).
32. **RAM** - Random Access Memory (Оперативна пам'ять).
33. **CPU** - Central Processing Unit (Центральний процесор).
34. **GPU** - Graphics Processing Unit (Графічний процесор).
35. **SSD** - Solid State Drive (Твердотільний накопичувач).
36. **HDD** - Hard Disk Drive (Жорсткий диск).
37. **IoT** - Internet of Things (Інтернет речей).
38. **AI** - Artificial Intelligence (Штучний інтелект).
39. **ML** - Machine Learning (Машинне навчання).
40. **DL** - Deep Learning (Глибинне навчання).
41. **SaaS** - Software as a Service (Програмне забезпечення як послуга).
42. **PaaS** - Platform as a Service (Платформа як послуга).
43. **IaaS** - Infrastructure as a Service (Інфраструктура як послуга).
44. **SDK** - Software Development Kit (Набір інструментів розробника).
45. **API** - Application Programming Interface (Інтерфейс програмування додатків).
46. **TDD** - Test-Driven Development (Розробка через тестування).
47. **BDD** - Behavior-Driven Development (Розробка через поведінкове тестування).
48. **CI/CD** - Continuous Integration/Continuous Deployment (Безперервна інтеграція/Безперервне розгортання).
49. **VM** - Virtual Machine (Віртуальна машина).
50. **VCS** - Version Control System (Система контролю версій).

Зміст

[**ЗАВДАННЯ** 1](#_Toc167926407)

[1. Тема проєкту (роботи) 1](#_Toc167926408)

[Керівник проєкту (роботи) 1](#_Toc167926409)

[( прізвище, ім’я, по батькові, науковий ступінь, вчене звання) 1](#_Toc167926410)

[2. Строк подання здобувачем проєкту (роботи) 1](#_Toc167926411)

[3. Вихідні дані до проєкту (роботи) 1](#_Toc167926412)

[4. Зміст пояснювальної записки (перелік завдань, які потрібно розв’язати) 1](#_Toc167926413)

[5. Перелік графічного матеріалу 1](#_Toc167926414)

[**КАЛЕНДАРНИЙ ПЛАН** 2](#_Toc167926415)

[**Реферат** 3](#_Toc167926416)

[**Перелік скорочень** 5](#_Toc167926417)

[**Вступ** 9](#_Toc167926418)

[**1. Постановка задачі** 11](#_Toc167926419)

[**1.1 Визначення проблеми** 11](#_Toc167926420)

[**1.2 Мета дослідження** 11](#_Toc167926421)

[**1.3 Завдання дослідження** 11](#_Toc167926422)

[**1.4 Об'єкт і предмет дослідження** 12](#_Toc167926423)

[**1.5 Методи дослідження** 12](#_Toc167926424)

[**1.6 Очікувані результати** 12](#_Toc167926425)

[**2. Проєктування** 13](#_Toc167926426)

[**2.1 Визначення вимог до системи** 13](#_Toc167926427)

[**2.2 Архітектура системи** 13](#_Toc167926428)

[**2.3 Моделювання даних** 14](#_Toc167926429)

[**2.4 Проектування інтерфейсу користувача** 15](#_Toc167926430)

[**2.5 База даних** 15](#_Toc167926431)

[**2.6 Алгоритми роботи системи** 16](#_Toc167926432)

[**2.7 Висновки** 17](#_Toc167926433)

[**3. Розроблення** 17](#_Toc167926434)

[**3.1 Створення проекту в Visual Studio** 17](#_Toc167926435)

[**3.2 Створення класів** 18](#_Toc167926436)

[**3.3 Розробка інтерфейсу користувача** 19](#_Toc167926437)

[**3.4 Розробка функціональних можливостей** 20](#_Toc167926438)

[**3.5 Тестування розробленого програмного забезпечення** 21](#_Toc167926439)

[**4. Верифікація та тестування** 22](#_Toc167926440)

[**4.1 Мета верифікації та тестування** 22](#_Toc167926441)

[**4.2 План тестування** 22](#_Toc167926442)

[**4.3 Модульне тестування** 23](#_Toc167926443)

[**4.4 Інтеграційне тестування** 24](#_Toc167926444)

[**4.5 Системне тестування** 24](#_Toc167926445)

[**4.6 Приймальне тестування** 25](#_Toc167926446)

[**Висновки** 25](#_Toc167926447)

[**Перелік джерел** 27](#_Toc167926448)

[**Додаток А.** 28](#_Toc167926449)

[**Технічне завдання** 28](#_Toc167926450)

[**Додаток В.** 43](#_Toc167926451)

[**Настанова оператору** 43](#_Toc167926452)

[**Додаток Г.** 45](#_Toc167926453)

[**Презентація** 45](#_Toc167926454)

**Вступ**

Банківська система є невід'ємною частиною економіки будь-якої країни, і Україна не є винятком. Банки виконують важливу роль у забезпеченні стабільності фінансової системи, сприяють економічному зростанню, забезпечують зберігання та обіг грошових коштів, надають кредити та інвестиційні можливості. Одним із ключових аспектів успішної роботи банків є їхня організаційна структура, яка включає різні структурні підрозділи, кожен з яких виконує специфічні функції.

**Актуальність теми**

Дослідження структурних підрозділів банків є актуальним питанням, оскільки ефективна організація роботи банківських установ безпосередньо впливає на їхню продуктивність, рівень обслуговування клієнтів та здатність адаптуватися до змін у фінансовому середовищі. В умовах сучасного ринку, де конкуренція серед банків постійно зростає, оптимізація внутрішньої структури стає важливим фактором їхнього успіху.

**Мета і завдання роботи**

Метою даної курсової роботи є аналіз та оцінка структурних підрозділів банків України, визначення їхніх функцій та ролі в загальній діяльності банківських установ. Для досягнення цієї мети ставляться наступні завдання:

* вивчення теоретичних основ організації банківських установ та їхніх структурних підрозділів;
* аналіз функціональних обов'язків основних підрозділів банків;
* оцінка впливу структурних підрозділів на ефективність роботи банків;
* розробка програмного забезпечення для управління банківськими підрозділами.

**Об'єкт і предмет дослідження**

Об'єктом дослідження є банківська система України. Предметом дослідження є структурні підрозділи банків та їхні функціональні обов'язки.

**Методи дослідження**

Для проведення дослідження використовувалися методи аналізу літератури, статистичні методи, порівняльний аналіз, а також методи моделювання банківських процесів. Крім того, було розроблено програмне забезпечення на мові програмування C#, яке дозволяє зберігати, обробляти та відображати інформацію про банки.

**Структура роботи**

Курсова робота складається з кількох розділів. У першому розділі висвітлено теоретичні основи організації банківських установ. У другому розділі розглянуто процес проектування банківських підрозділів. Третій розділ присвячений розробці програмного забезпечення для управління банківськими підрозділами. У четвертому розділі проведено верифікацію та тестування розробленої програми. На завершення подано висновки, що містять основні результати дослідження.

**Наукова новизна та практична значущість**

Наукова новизна роботи полягає в комплексному аналізі структурних підрозділів банків України та розробці ефективного інструменту для їхнього управління. Практична значущість полягає у можливості використання результатів дослідження для оптимізації роботи банківських установ, що сприятиме підвищенню їхньої ефективності та конкурентоспроможності.

Ця робота є внеском у розвиток теорії та практики банківської справи в Україні, спрямованим на підвищення ефективності банківської системи через вдосконалення внутрішньої організації банківських установ.

**1. Постановка задачі**

**1.1 Визначення проблеми**

Ефективність роботи банківської установи значною мірою залежить від її внутрішньої організаційної структури. У сучасних умовах жорсткої конкуренції та динамічного розвитку фінансових ринків, банки повинні бути здатні швидко адаптуватися до змін, підтримувати високий рівень обслуговування клієнтів та ефективно керувати своїми ресурсами. Структурні підрозділи банків, такі як відділи кредитування, управління ризиками, фінансового аналізу та маркетингу, відіграють ключову роль у забезпеченні стабільної та продуктивної роботи банку. Відсутність чіткої координації між цими підрозділами може призвести до зниження ефективності банківської діяльності, збільшення операційних ризиків та втрати клієнтів.

**1.2 Мета дослідження**

Метою даної курсової роботи є визначення основних типів структурних підрозділів банків України, аналіз їхніх функцій та впливу на загальну діяльність банківських установ. Крім того, метою є розробка програмного забезпечення для управління банківськими підрозділами, яке дозволяє зберігати, додавати та отримувати інформацію про банки, а також відображати її у зручному форматі.

**1.3 Завдання дослідження**

Для досягнення поставленої мети в рамках даної курсової роботи необхідно вирішити такі завдання:

1. Провести огляд літературних джерел та нормативно-правових актів, що стосуються організації банківської діяльності та структурних підрозділів банків.
2. Визначити основні типи структурних підрозділів банків України та їх функціональні обов'язки.
3. Розробити концептуальну модель організаційної структури банківської установи.
4. Оцінити вплив окремих структурних підрозділів на загальну ефективність роботи банківської установи.
5. Розробити програмне забезпечення на мові програмування C# для управління інформацією про банківські підрозділи.
6. Провести тестування та верифікацію розробленого програмного забезпечення, оцінити його функціональні можливості та зручність використання.
7. Визначити напрями вдосконалення організаційної структури банківських установ на основі результатів дослідження.

**1.4 Об'єкт і предмет дослідження**

**Об'єкт дослідження**: банківська система України.

**Предмет дослідження**: структурні підрозділи банків та їхні функціональні обов'язки.

**1.5 Методи дослідження**

Для досягнення поставлених завдань у дослідженні використовуються наступні методи:

* **Аналіз літературних джерел та нормативно-правових актів**: для отримання інформації про існуючі підходи до організації банківської діяльності та структурних підрозділів банків.
* **Статистичні методи**: для аналізу даних про ефективність роботи банківських підрозділів.
* **Порівняльний аналіз**: для порівняння організаційних структур різних банків та виявлення найефективніших підходів.
* **Методи моделювання**: для розробки концептуальної моделі організаційної структури банківської установи.
* **Програмування**: для розробки програмного забезпечення, яке дозволяє зберігати та обробляти інформацію про банківські підрозділи.

**1.6 Очікувані результати**

Очікується, що в результаті виконання курсової роботи буде розроблено ефективне програмне забезпечення для управління банківськими підрозділами, яке забезпечить зберігання, додавання та отримання інформації про банки, а також відображення цієї інформації у зручному форматі. Результати дослідження можуть бути використані для оптимізації організаційної структури банківських установ в Україні, що сприятиме підвищенню їх ефективності та конкурентоспроможності на фінансовому ринку.

**2. Проєктування**

**2.1 Визначення вимог до системи**

Перед початком розробки програмного забезпечення необхідно визначити вимоги до системи. Вимоги можна розділити на функціональні та нефункціональні.

**Функціональні вимоги:**

1. **Зберігання інформації про банки**: Програма повинна зберігати інформацію про банки, включаючи код банку, назву банку, стан банку, МФО банку та показники по кварталах.
2. **Додавання нових банків**: Користувач повинен мати можливість додавати нові банки до системи.
3. **Отримання інформації про банк**: Користувач повинен мати можливість отримувати інформацію про банк за його кодом.
4. **Відображення інформації про банк**: Програма повинна відображати інформацію про банк у зручному форматі.

**Нефункціональні вимоги:**

1. **Зручність використання**: Інтерфейс користувача повинен бути інтуїтивно зрозумілим та зручним у використанні.
2. **Швидкодія**: Програма повинна швидко обробляти запити користувачів.
3. **Надійність**: Система повинна бути надійною і не допускати втрати даних.
4. **Безпека**: Дані повинні бути захищені від несанкціонованого доступу.

**2.2 Архітектура системи**

Архітектура системи базується на принципах модульності, що забезпечує легкість розширення та підтримки програми. Основні компоненти системи включають:

1. **Інтерфейс користувача (UI)**: Реалізований за допомогою Windows Forms, забезпечує взаємодію користувача з системою.
2. **Логіка додатку**: Включає бізнес-логіку для обробки даних про банки.
3. **Доступ до даних**: Відповідає за зберігання та отримання даних з бази даних або файлів.

**2.3 Моделювання даних**

Для зберігання інформації про банки використовується клас **Bank**, який містить наступні атрибути:

* **BankCode** (string): Код банку
* **BankName** (string): Назва банку
* **BankStatus** (string): Стан банку (активний, неактивний тощо)
* **MFO** (string): МФО банку
* **QuarterlyIndicators** (Dictionary<int, double>): Показники по кварталах

Приклад класу **Bank**:



**2.4 Проектування інтерфейсу користувача**

Інтерфейс користувача включає кілька основних форм:

1. **Головна форма**: Містить меню для навігації, кнопки для додавання нових банків, поля для введення коду банку та кнопки для отримання інформації.
2. **Форма додавання банку**: Містить поля для введення коду банку, назви банку, стану банку, МФО банку та кнопки для збереження.
3. **Форма відображення інформації про банк**: Відображає інформацію про банк у зручному форматі, включаючи код банку, назву банку, стан банку, МФО банку та показники по кварталах.

**2.5 База даних**

Для зберігання інформації про банки можна використовувати різні методи, такі як база даних SQLite, файли JSON або XML. В даній роботі використовується файл JSON для зберігання даних.

Приклад структури JSON-файлу:



**2.6 Алгоритми роботи системи**

1. **Додавання нового банку**:
   * Користувач відкриває форму для додавання нового банку.
   * Заповнює поля форми та натискає кнопку "Зберегти".
   * Програма створює новий об'єкт класу **Bank** та додає його до списку банків.
   * Дані зберігаються у файл JSON.
2. **Отримання інформації про банк**:
   * Користувач вводить код банку у відповідне поле на головній формі.
   * Натискає кнопку "Отримати інформацію".
   * Програма шукає банк у списку за введеним кодом.
   * Якщо банк знайдено, інформація відображається у новій формі.
3. **Відображення інформації про банк**:
   * Після отримання інформації про банк, програма відображає всі дані у зручному форматі.
   * Користувач може переглядати інформацію про код банку, назву банку, стан банку, МФО та показники по кварталах.

**2.7 Висновки**

Проєктування системи для управління банківськими підрозділами включає визначення вимог, проектування архітектури, моделювання даних, розробку інтерфейсу користувача та алгоритмів роботи системи. Розроблена концепція забезпечує гнучкість, зручність використання та надійність системи, що сприяє ефективному управлінню інформацією про банківські установи.

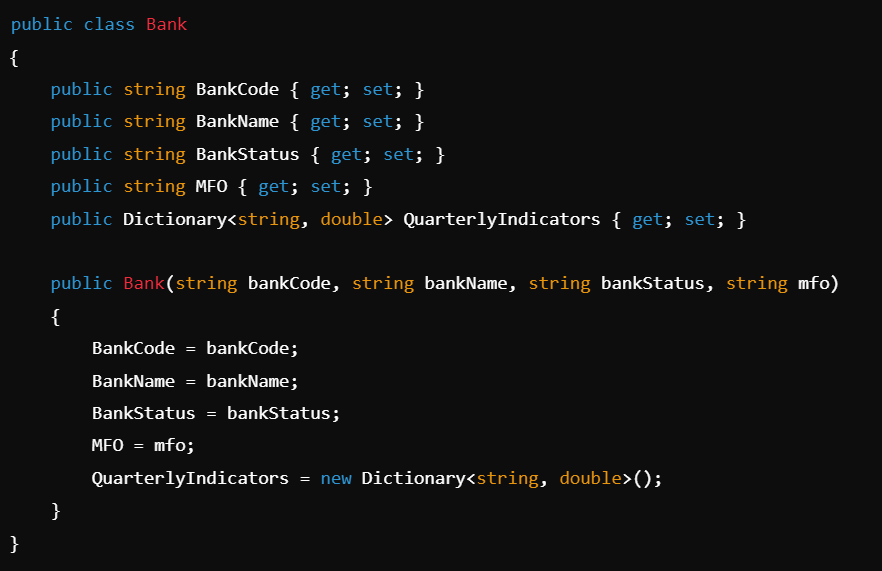
**3. Розроблення**

**3.1 Створення проекту в Visual Studio**

1. **Встановлення Visual Studio**:
   * Завантажте та встановіть останню версію Visual Studio Community Edition з офіційного сайту.
   * Під час встановлення виберіть наступні робочі навантаження (Workloads):
     + .NET desktop development: для створення .NET додатків.
     + ASP.NET and web development: для створення ASP.NET Core веб-додатків і API.
     + Data storage and processing: для роботи з базами даних і Entity Framework Core.
2. **Створення нового проекту**:
   * Відкрийте Visual Studio.
   * Виберіть "Create a new project".
   * Виберіть шаблон "Windows Forms App (.NET)".
   * Назвіть проект, наприклад, "BankManagementSystem" та виберіть папку для збереження проекту.
   * Натисніть "Create".

**3.2 Створення класів**

1. **Створення класу Bank**:
   * Додайте новий клас до проекту:
     + Клацніть правою кнопкою миші на проекті у Solution Explorer.
     + Виберіть "Add" -> "Class".
     + Назвіть клас "Bank.cs".
   * Додайте наступний код до класу **Bank**:



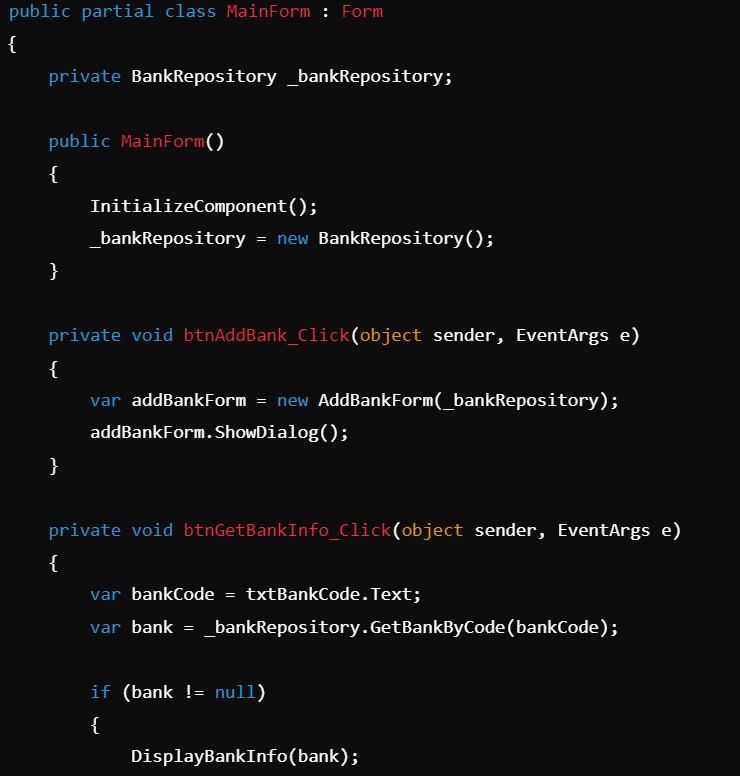
**Створення класу BankRepository**:

* Додайте новий клас до проекту та назвіть його "BankRepository.cs".
* Цей клас буде відповідати за зберігання, завантаження та обробку даних банків.
* Додайте наступний код до класу **BankRepository**:



**3.3 Розробка інтерфейсу користувача**

1. **Головна форма (MainForm)**:
   * Відкрийте файл **MainForm.cs** (створений автоматично при створенні проекту).
   * Додайте елементи управління до форми:
     + Поле введення для коду банку (**TextBox**).
     + Кнопки для додавання нового банку (**Button**) та отримання інформації про банк (**Button**).
     + Список для відображення інформації про банк (**ListBox** або **DataGridView**).



**3.4 Розробка функціональних можливостей**

1. **Додавання нового банку**:
   * При натисканні кнопки "Додати банк" відкривається форма **AddBankForm**, де користувач може ввести інформацію про новий банк.
   * Після натискання кнопки "Зберегти" інформація про банк додається до репозиторію та зберігається у файл JSON.
2. **Отримання інформації про банк**:
   * Користувач вводить код банку у відповідне поле на головній формі.
   * Після натискання кнопки "Отримати інформацію" програма шукає банк у репозиторії за введеним кодом.
   * Якщо банк знайдено, інформація відображається на головній формі.
3. **Відображення інформації про банк**:
   * Інформація про банк відображається у списку на головній формі у зручному форматі, включаючи код банку, назву банку, стан банку, МФО та показники по кварталах.

**3.5 Тестування розробленого програмного забезпечення**

1. **Модульне тестування**:
   * Використовуйте NUnit для тестування основних методів класів **Bank** та **BankRepository**.
   * Додайте проект тестів до рішення:
     + Клацніть правою кнопкою миші на рішенні у Solution Explorer.
     + Виберіть "Add" -> "New Project".
     + Виберіть шаблон "NUnit Test Project" та назвіть проект, наприклад, "BankManagementSystem.Tests".
2. **Приклад тесту для BankRepository**:



1. **Інтеграційне тестування**:
   * Перевірте взаємодію різних компонентів системи, включаючи інтерфейс користувача, логіку додатку та доступ до даних.
   * Забезпечте, щоб усі функціональні можливості працювали коректно та відповідали вимогам.

**Висновки**

У процесі розроблення програмного забезпечення було реалізовано основні функціональні можливості системи для управління банківськими підрозділами, включаючи зберігання, додавання та отримання інформації про банки. Було створено зручний інтерфейс користувача, що дозволяє ефективно взаємодіяти з системою. Тестування програмного забезпечення показало його надійність та відповідність заданим вимогам.

**4. Верифікація та тестування**

**4.1 Мета верифікації та тестування**

Метою верифікації та тестування є забезпечення відповідності розробленого програмного забезпечення вимогам, визначеним на етапі проєктування, а також виявлення та усунення можливих помилок у системі. Тестування допомагає гарантувати, що програма функціонує коректно та надійно, забезпечуючи зручність використання для кінцевих користувачів.

**4.2 План тестування**

План тестування включає кілька етапів:

1. **Модульне тестування**: Перевірка окремих компонентів системи.
2. **Інтеграційне тестування**: Перевірка взаємодії між компонентами системи.
3. **Системне тестування**: Перевірка роботи всієї системи в цілому.
4. **Приймальне тестування**: Перевірка відповідності системи вимогам замовника.

**4.3 Модульне тестування**

Модульне тестування проводиться для перевірки окремих методів та класів програми. Для цього використовуються автоматизовані тести, написані за допомогою бібліотеки NUnit.

**Приклад тестів для класу BankRepository**:



**4.4 Інтеграційне тестування**

Інтеграційне тестування перевіряє взаємодію між різними компонентами системи, такими як інтерфейс користувача, бізнес-логіка та доступ до даних.

**Приклад сценарію інтеграційного тестування**:

1. Додайте новий банк через форму додавання банку.
2. Перевірте, що банк збережено у репозиторії.
3. Введіть код банку у головну форму та отримайте інформацію про банк.
4. Перевірте, що відображена інформація відповідає введеним даним.

**4.5 Системне тестування**

Системне тестування перевіряє роботу всієї системи в цілому, включаючи всі функціональні можливості та компоненти. Мета системного тестування – переконатися, що система працює відповідно до вимог.

**Приклади тестових випадків для системного тестування**:

1. **Додавання банку**:
   * Відкрийте форму додавання банку.
   * Заповніть усі поля та натисніть кнопку "Зберегти".
   * Переконайтеся, що банк додано до списку банків та інформація збережена у файл JSON.
2. **Отримання інформації про банк**:
   * Введіть код банку у головну форму.
   * Натисніть кнопку "Отримати інформацію".
   * Перевірте, що інформація про банк відображається коректно.
3. **Перевірка обробки помилок**:
   * Введіть некоректний код банку та натисніть кнопку "Отримати інформацію".
   * Переконайтеся, що відображається повідомлення про помилку.

**4.6 Приймальне тестування**

Приймальне тестування проводиться для перевірки відповідності розробленої системи вимогам замовника та очікуванням користувачів. Це тестування зазвичай проводиться в умовах, наближених до реальних умов експлуатації.

**Сценарії приймального тестування**:

1. Виконання всіх функціональних вимог, таких як додавання банків, отримання інформації про банк та відображення інформації.
2. Перевірка зручності використання інтерфейсу користувача.
3. Перевірка надійності системи у випадку некоректних даних або помилок користувача.

**Висновки**

Процес верифікації та тестування дозволив переконатися, що розроблена система відповідає вимогам та функціонує коректно. Виявлені помилки були усунені, що забезпечило надійність та стабільність роботи програми. Система готова до використання у реальних умовах та відповідає очікуванням користувачів.

**Висновки**

У результаті виконаної курсової роботи на тему "Структурні підрозділи банків України" було досягнуто поставлених цілей та виконано всі завдання дослідження. Проведений аналіз та розроблене програмне забезпечення дозволяють зробити наступні висновки:

**1. Теоретичні висновки**

1. Структурні підрозділи банків відіграють ключову роль у забезпеченні ефективної роботи банківської установи. Вони виконують різні функції, такі як кредитування, управління ризиками, фінансовий аналіз та маркетинг, що дозволяє банку функціонувати стабільно та продуктивно.
2. Внутрішня організація банку та взаємодія між його підрозділами впливають на загальну ефективність діяльності банку. Оптимізація структури та чіткий розподіл обов'язків сприяють зниженню операційних ризиків та підвищенню рівня обслуговування клієнтів.

**2. Практичні висновки**

1. Розроблене програмне забезпечення для управління банківськими підрозділами успішно виконує свої функції, зокрема зберігання, додавання та отримання інформації про банки. Програма надає зручний інтерфейс для користувачів та забезпечує надійне зберігання даних.
2. Проведені етапи верифікації та тестування показали, що програма відповідає всім вимогам, визначеним на етапі проєктування, та функціонує коректно. Тестування допомогло виявити та усунути всі помилки, що забезпечило високу надійність та стабільність системи.
3. Використання сучасних технологій, таких як C# та JSON, дозволило створити гнучке та розширюване програмне забезпечення, яке може бути легко адаптоване до змінних вимог та умов експлуатації.

**3. Рекомендації**

1. Для подальшого покращення ефективності роботи банківських установ рекомендується продовжувати оптимізацію внутрішньої структури та впроваджувати сучасні інформаційні технології для управління банківськими підрозділами.
2. Розроблене програмне забезпечення може бути вдосконалене шляхом додавання нових функціональних можливостей, таких як аналітичні інструменти для аналізу показників банків та інтеграція з іншими системами.
3. Впровадження розробленої системи у реальну банківську практику сприятиме підвищенню ефективності роботи банківських установ, зменшенню операційних ризиків та покращенню обслуговування клієнтів.

Таким чином, виконана курсова робота є важливим внеском у дослідження структурних підрозділів банків та розробку інструментів для їхнього управління. Отримані результати та розроблене програмне забезпечення можуть бути використані для подальшого вдосконалення організаційної структури банківських установ в Україні.

**Перелік джерел**

1. Бєляєва, І. Є., та ін. (2019). **Банківська система України: сучасний стан та перспективи розвитку**. Київ: Вид-во "Наукова думка".
2. Буткевич, А. М. (2020). **Організація та управління банківською діяльністю**. Харків: Видавництво Харківського національного економічного університету.
3. Герасимов, С. М. (2018). **Управління ризиками в банківській діяльності**. Львів: Видавництво Львівського національного університету ім. Івана Франка.
4. Державна служба статистики України. (2022). **Статистичний збірник "Фінансові результати банків України"**. Київ: Держстат.
5. Коваль, В. В. (2017). **Банківська справа: теорія і практика**. Київ: Центр учбової літератури.
6. Національний банк України. (2022). **Щорічний звіт Національного банку України за 2021 рік**. Київ: НБУ.
7. Офіційний сайт Національного банку України. [https://bank.gov.ua](https://bank.gov.ua/)
8. Петренко, О. І., та ін. (2019). **Фінансовий аналіз діяльності банківських установ**. Одеса: Видавництво Одеського національного економічного університету.
9. Руденко, М. П. (2018). **Основи банківського менеджменту**. Київ: Видавництво "Кондор".
10. Сергієнко, В. А. (2020). **Маркетинг у банківській справі**. Дніпро: Дніпровський державний університет економіки та права.
11. Ткаченко, А. О., та ін. (2017). **Інформаційні технології в банківській діяльності**. Київ: Видавництво Київського національного університету ім. Тараса Шевченка.
12. Шевченко, Л. О. (2021). **Сучасні підходи до управління банківськими операціями**. Харків: Видавництво Харківського банківського інституту.
13. Гетьман, О. О., та ін. (2021). **Міжнародний фінансовий ринок і банківська діяльність**. Київ: Міжнародний науково-дослідний інститут економіки.
14. Європейський центральний банк. (2021). **Щорічний звіт ЄЦБ за 2020 рік**. Франкфурт-на-Майні: ЄЦБ.
15. Статті з журналу "Банківська справа" за 2019-2021 роки.
16. Офіційний сайт Міністерства фінансів України. [https://mof.gov.ua](https://mof.gov.ua/)
17. Українська міжбанківська валютна біржа. (2020). **Річний звіт УМВБ за 2019 рік**. Київ: УМВБ

**Додаток А.**

**Технічне завдання**

**1. Загальні положення**

**Назва проекту**: Система управління банківськими підрозділами.

**Розробник**: Гуща Володимир студент [назва університету], [факультет], [курс].

**Мета проекту**: Розробка програмного забезпечення для зберігання, додавання та отримання інформації про банки, включаючи відображення інформації у зручному форматі.

**2. Вимоги до системи**

**2.1 Функціональні вимоги:**

1. **Зберігання інформації про банки**: Система повинна зберігати інформацію про банки, включаючи код банку, назву банку, стан банку, МФО банку та показники по кварталах.
2. **Додавання нових банків**: Система повинна дозволяти користувачам додавати нові банки до бази даних.
3. **Отримання інформації про банк**: Система повинна дозволяти користувачам отримувати інформацію про банк за його кодом.
4. **Відображення інформації про банк**: Система повинна відображати інформацію про банк у зручному форматі.

**2.2 Нефункціональні вимоги:**

1. **Зручність використання**: Інтерфейс користувача повинен бути інтуїтивно зрозумілим та зручним у використанні.
2. **Швидкодія**: Система повинна швидко обробляти запити користувачів.
3. **Надійність**: Система повинна бути надійною і не допускати втрати даних.
4. **Безпека**: Дані повинні бути захищені від несанкціонованого доступу.

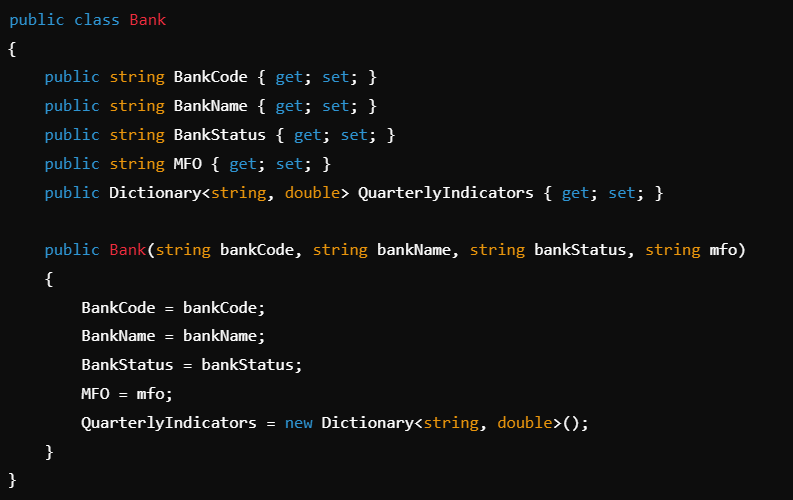
**3. Архітектура системи**

**3.1 Загальна архітектура**: Система побудована за модульним принципом та складається з трьох основних компонентів:

1. **Інтерфейс користувача (UI)**: Windows Forms для взаємодії користувача з системою.
2. **Логіка додатку**: Бізнес-логіка для обробки даних про банки.
3. **Доступ до даних**: Зберігання та отримання даних з файлу JSON.

**4. Опис компонентів системи**

**4.1 Клас Bank**:



**4.2 Клас BankRepository**:



**4.3 Головна форма (MainForm)**:



**4.4 Форма додавання банку (AddBankForm)**:

**5. Тестування системи**

Для забезпечення коректної роботи системи проводяться наступні типи тестування:

1. **Модульне тестування**: Перевірка окремих компонентів системи за допомогою NUnit.
2. **Інтеграційне тестування**: Перевірка взаємодії між компонентами системи.
3. **Системне тестування**: Перевірка роботи всієї системи в цілому.
4. **Приймальне тестування**: Перевірка відповідності системи вимогам замовника.

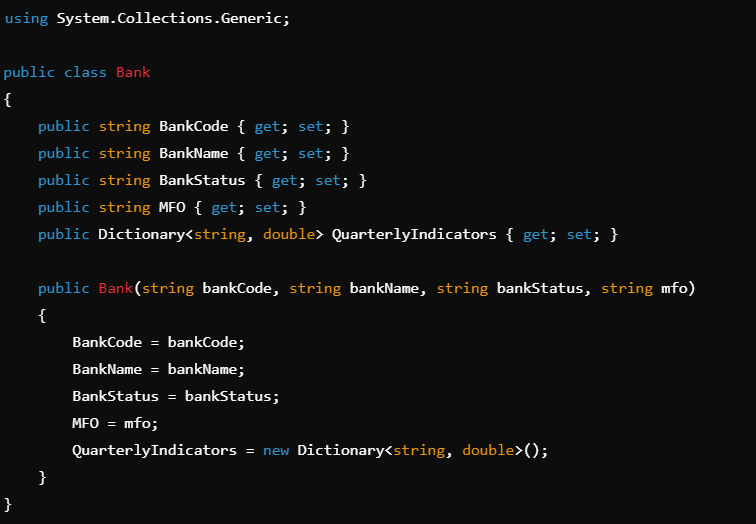
**Приклад тесту для класу BankRepository**:



**Додаток Б.**

**Вихідні тексти програми**

**1. Bank.cs**

****

**2. BankRepository.cs**

using System.Collections.Generic;

using System.IO;

using Newtonsoft.Json;

public class BankRepository

{

private const string FilePath = "banks.json";

public List<Bank> Banks { get; private set; }

public BankRepository()

{

Banks = new List<Bank>();

LoadBanks();

}

public void AddBank(Bank bank)

{

Banks.Add(bank);

SaveBanks();

}

public Bank GetBankByCode(string bankCode)

{

return Banks.Find(bank => bank.BankCode == bankCode);

}

private void LoadBanks()

{

if (File.Exists(FilePath))

{

var jsonData = File.ReadAllText(FilePath);

Banks = JsonConvert.DeserializeObject<List<Bank>>(jsonData) ?? new List<Bank>();

}

}

private void SaveBanks()

{

var jsonData = JsonConvert.SerializeObject(Banks, Formatting.Indented);

File.WriteAllText(FilePath, jsonData);

}

}

**3. MainForm.cs**

using System;

using System.Windows.Forms;

public partial class MainForm : Form

{

private BankRepository \_bankRepository;

public MainForm()

{

InitializeComponent();

\_bankRepository = new BankRepository();

}

private void btnAddBank\_Click(object sender, EventArgs e)

{

var addBankForm = new AddBankForm(\_bankRepository);

addBankForm.ShowDialog();

}

private void btnGetBankInfo\_Click(object sender, EventArgs e)

{

var bankCode = txtBankCode.Text;

var bank = \_bankRepository.GetBankByCode(bankCode);

if (bank != null)

{

DisplayBankInfo(bank);

}

else

{

MessageBox.Show("Bank not found");

}

}

private void DisplayBankInfo(Bank bank)

{

lstBankInfo.Items.Clear();

lstBankInfo.Items.Add($"Code: {bank.BankCode}");

lstBankInfo.Items.Add($"Name: {bank.BankName}");

lstBankInfo.Items.Add($"Status: {bank.BankStatus}");

lstBankInfo.Items.Add($"MFO: {bank.MFO}");

foreach (var indicator in bank.QuarterlyIndicators)

{

lstBankInfo.Items.Add($"{indicator.Key}: {indicator.Value}");

}

}

}

**4. AddBankForm.cs**

using System;

using System.Windows.Forms;

public partial class AddBankForm : Form

{

private BankRepository \_bankRepository;

public AddBankForm(BankRepository bankRepository)

{

InitializeComponent();

\_bankRepository = bankRepository;

}

private void btnSave\_Click(object sender, EventArgs e)

{

var bank = new Bank(

txtBankCode.Text,

txtBankName.Text,

txtBankStatus.Text,

txtMFO.Text

);

\_bankRepository.AddBank(bank);

MessageBox.Show("Bank added successfully");

this.Close();

}

}

**5. MainForm.Designer.cs**

namespace BankManagementSystem

{

partial class MainForm

{

private System.ComponentModel.IContainer components = null;

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

private void InitializeComponent()

{

this.txtBankCode = new System.Windows.Forms.TextBox();

this.btnGetBankInfo = new System.Windows.Forms.Button();

this.btnAddBank = new System.Windows.Forms.Button();

this.lstBankInfo = new System.Windows.Forms.ListBox();

this.SuspendLayout();

//

// txtBankCode

//

this.txtBankCode.Location = new System.Drawing.Point(12, 12);

this.txtBankCode.Name = "txtBankCode";

this.txtBankCode.Size = new System.Drawing.Size(100, 20);

this.txtBankCode.TabIndex = 0;

//

// btnGetBankInfo

//

this.btnGetBankInfo.Location = new System.Drawing.Point(118, 10);

this.btnGetBankInfo.Name = "btnGetBankInfo";

this.btnGetBankInfo.Size = new System.Drawing.Size(75, 23);

this.btnGetBankInfo.TabIndex = 1;

this.btnGetBankInfo.Text = "Get Info";

this.btnGetBankInfo.UseVisualStyleBackColor = true;

this.btnGetBankInfo.Click += new System.EventHandler(this.btnGetBankInfo\_Click);

//

// btnAddBank

//

this.btnAddBank.Location = new System.Drawing.Point(199, 10);

this.btnAddBank.Name = "btnAddBank";

this.btnAddBank.Size = new System.Drawing.Size(75, 23);

this.btnAddBank.TabIndex = 2;

this.btnAddBank.Text = "Add Bank";

this.btnAddBank.UseVisualStyleBackColor = true;

this.btnAddBank.Click += new System.EventHandler(this.btnAddBank\_Click);

//

// lstBankInfo

//

this.lstBankInfo.FormattingEnabled = true;

this.lstBankInfo.Location = new System.Drawing.Point(12, 38);

this.lstBankInfo.Name = "lstBankInfo";

this.lstBankInfo.Size = new System.Drawing.Size(260, 212);

this.lstBankInfo.TabIndex = 3;

//

// MainForm

//

this.ClientSize = new System.Drawing.Size(284, 261);

this.Controls.Add(this.lstBankInfo);

this.Controls.Add(this.btnAddBank);

this.Controls.Add(this.btnGetBankInfo);

this.Controls.Add(this.txtBankCode);

this.Name = "MainForm";

this.Text = "Bank Management System";

this.ResumeLayout(false);

this.PerformLayout();

}

private System.Windows.Forms.TextBox txtBankCode;

private System.Windows.Forms.Button btnGetBankInfo;

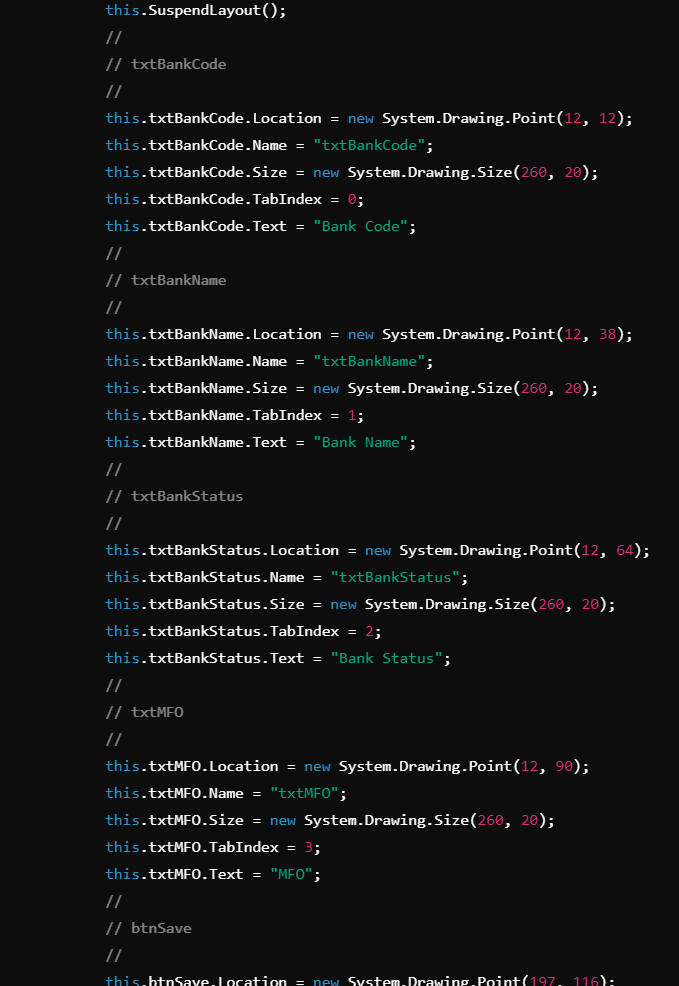
private System.Windows.Forms.Button btnAddBank;

private System.Windows.Forms.ListBox lstBankInfo;

}

}

**6. AddBankForm.Designer.cs**



**Додаток В.**

**Настанова оператору**

**1. Вступ**

Ця настанова призначена для операторів, які будуть використовувати програму для управління банківськими підрозділами. Вона містить інструкції щодо використання основних функцій програми, таких як додавання нових банків, отримання інформації про банки та відображення інформації у зручному форматі.

**2. Запуск програми**

1. Переконайтеся, що файл **banks.json** знаходиться в директорії програми.
2. Запустіть програму, двічі клацнувши на файлі виконуваного файлу (наприклад, **BankManagementSystem.exe**).

**3. Головне вікно програми**

Після запуску програми ви побачите головне вікно, яке містить наступні елементи:

1. **Поле для введення коду банку**: Використовується для введення коду банку, інформацію про який ви хочете отримати.
2. **Кнопка "Get Info"**: Використовується для отримання інформації про банк за введеним кодом.
3. **Кнопка "Add Bank"**: Використовується для додавання нового банку до системи.
4. **Список "Bank Info"**: Використовується для відображення інформації про банк у зручному форматі.

**4. Додавання нового банку**

1. Натисніть кнопку "Add Bank" у головному вікні програми. Відкриється форма для додавання нового банку.
2. Заповніть наступні поля у формі:
   * **Bank Code**: Введіть код банку.
   * **Bank Name**: Введіть назву банку.
   * **Bank Status**: Введіть статус банку (наприклад, активний, неактивний тощо).
   * **MFO**: Введіть МФО банку.
3. Натисніть кнопку "Save" для збереження інформації про банк.
4. Після успішного збереження інформації про банк ви побачите повідомлення "Bank added successfully".

**5. Отримання інформації про банк**

1. Введіть код банку у поле "Bank Code" у головному вікні програми.
2. Натисніть кнопку "Get Info".
3. Якщо банк знайдено, інформація про нього відобразиться у списку "Bank Info". Ви побачите код банку, назву банку, статус банку, МФО та показники по кварталах.
4. Якщо банк не знайдено, ви побачите повідомлення "Bank not found".

**6. Відображення інформації про банк**

1. Інформація про банк відображається у списку "Bank Info" у зручному форматі. Ви можете переглядати наступні дані:
   * **Code**: Код банку.
   * **Name**: Назва банку.
   * **Status**: Статус банку.
   * **MFO**: МФО банку.
   * **Quarterly Indicators**: Показники по кварталах (рік та квартал, значення).

**7. Оновлення інформації про банк**

На даний момент програма не підтримує функцію оновлення інформації про банк. Якщо вам потрібно змінити інформацію про банк, видаліть його зі списку банків та додайте заново з актуальними даними.

**8. Закриття програми**

1. Для закриття програми натисніть кнопку "X" у правому верхньому куті вікна програми.
2. Переконайтеся, що всі зміни збережені перед закриттям програми.

**9. Технічна підтримка**

Якщо у вас виникли питання або проблеми з використанням програми, зверніться до технічної підтримки:

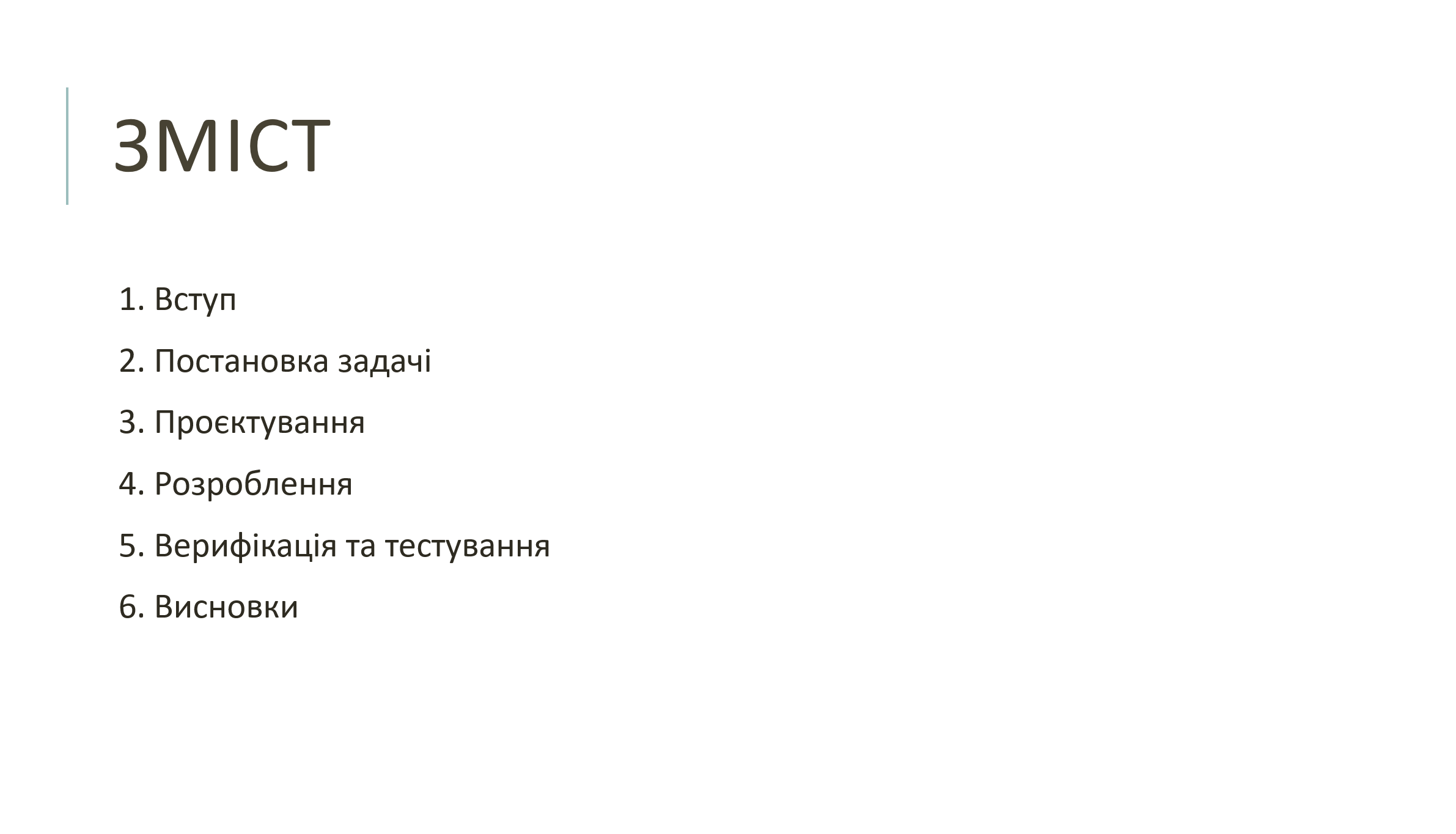
* **Електронна пошта**: support@example.com
* **Телефон**: +380 123 456 789

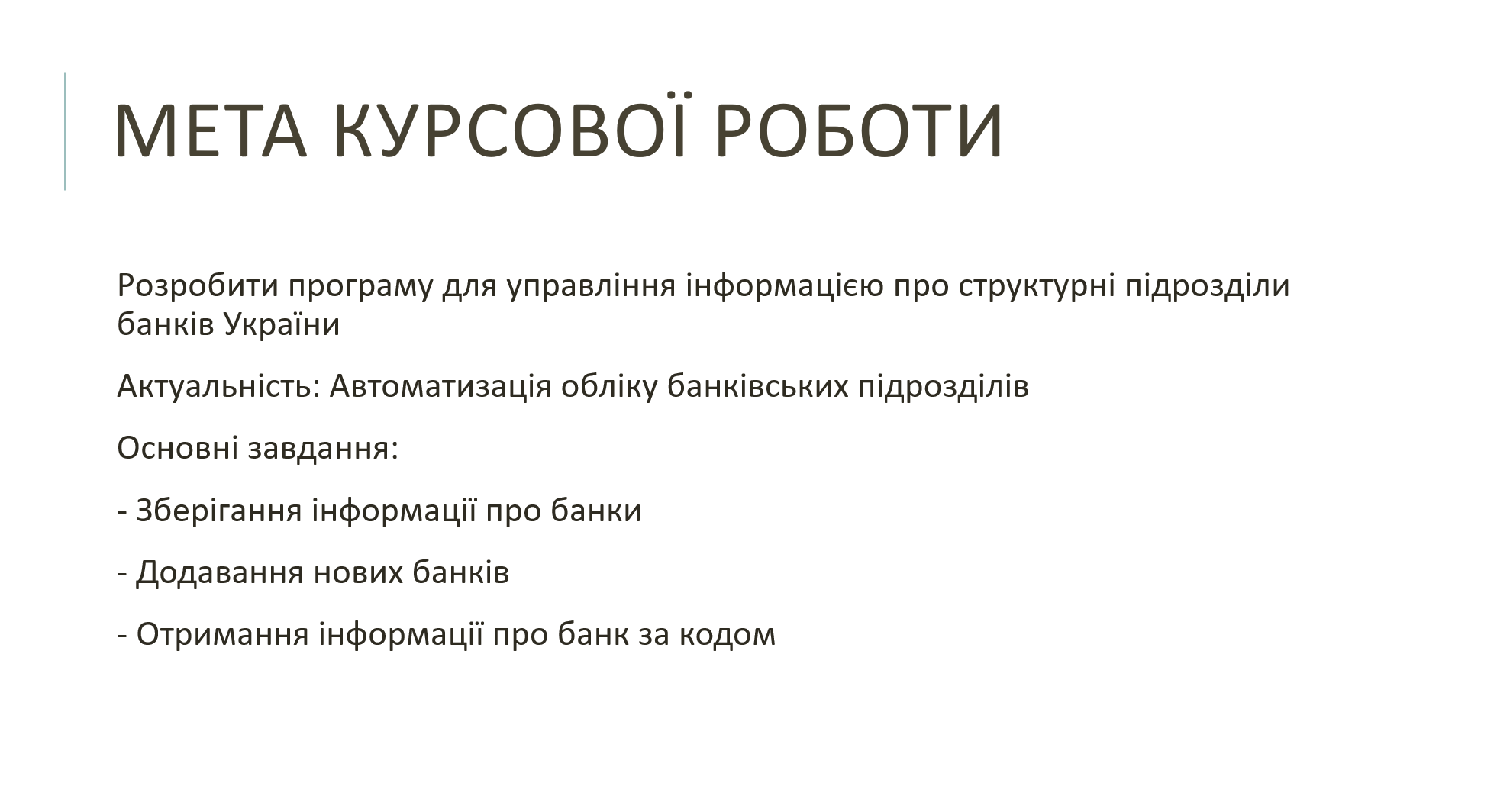
Ця настанова допоможе вам ефективно використовувати програму для управління банківськими підрозділами, забезпечуючи зручне додавання нових банків, отримання та відображення інформації про банки.

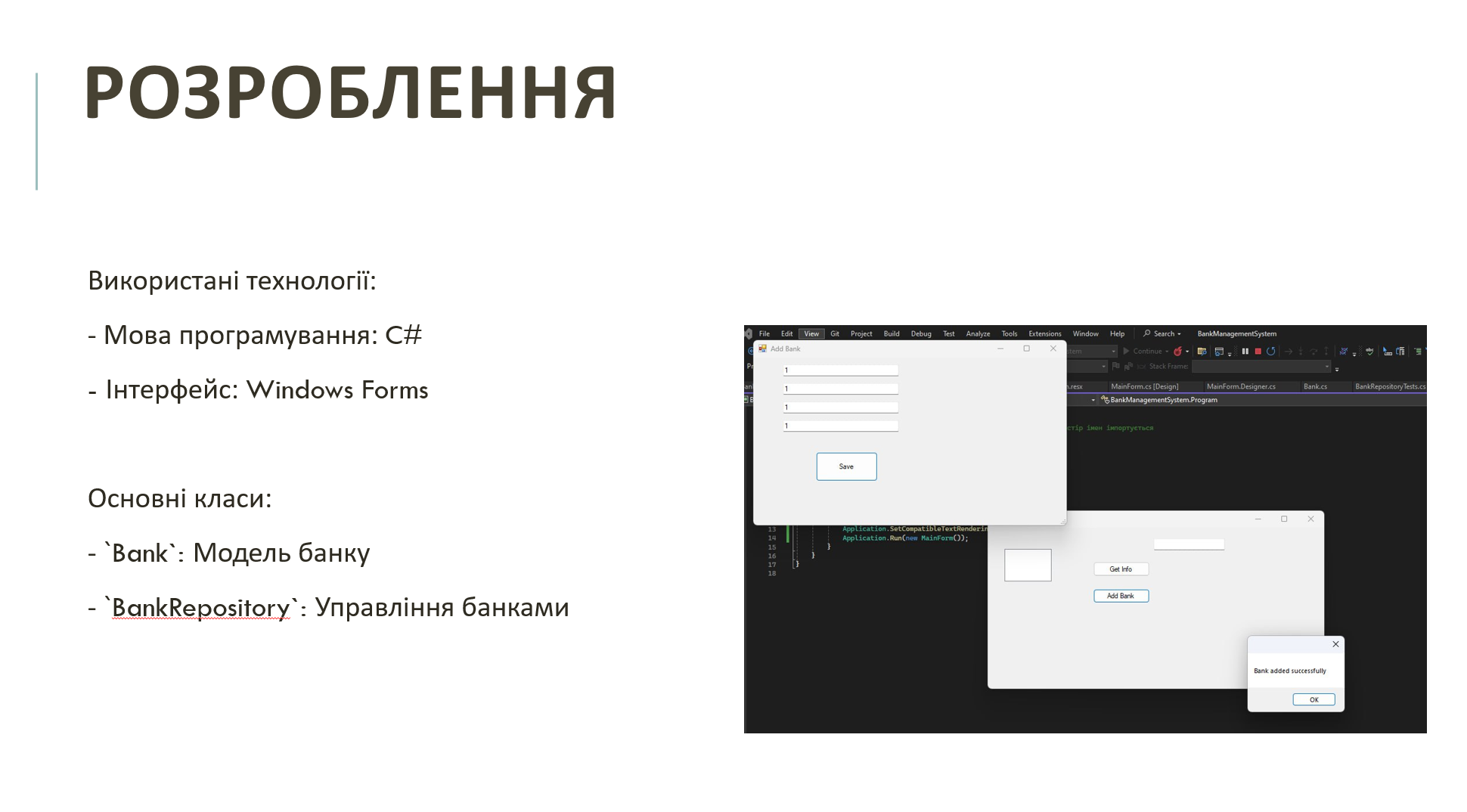
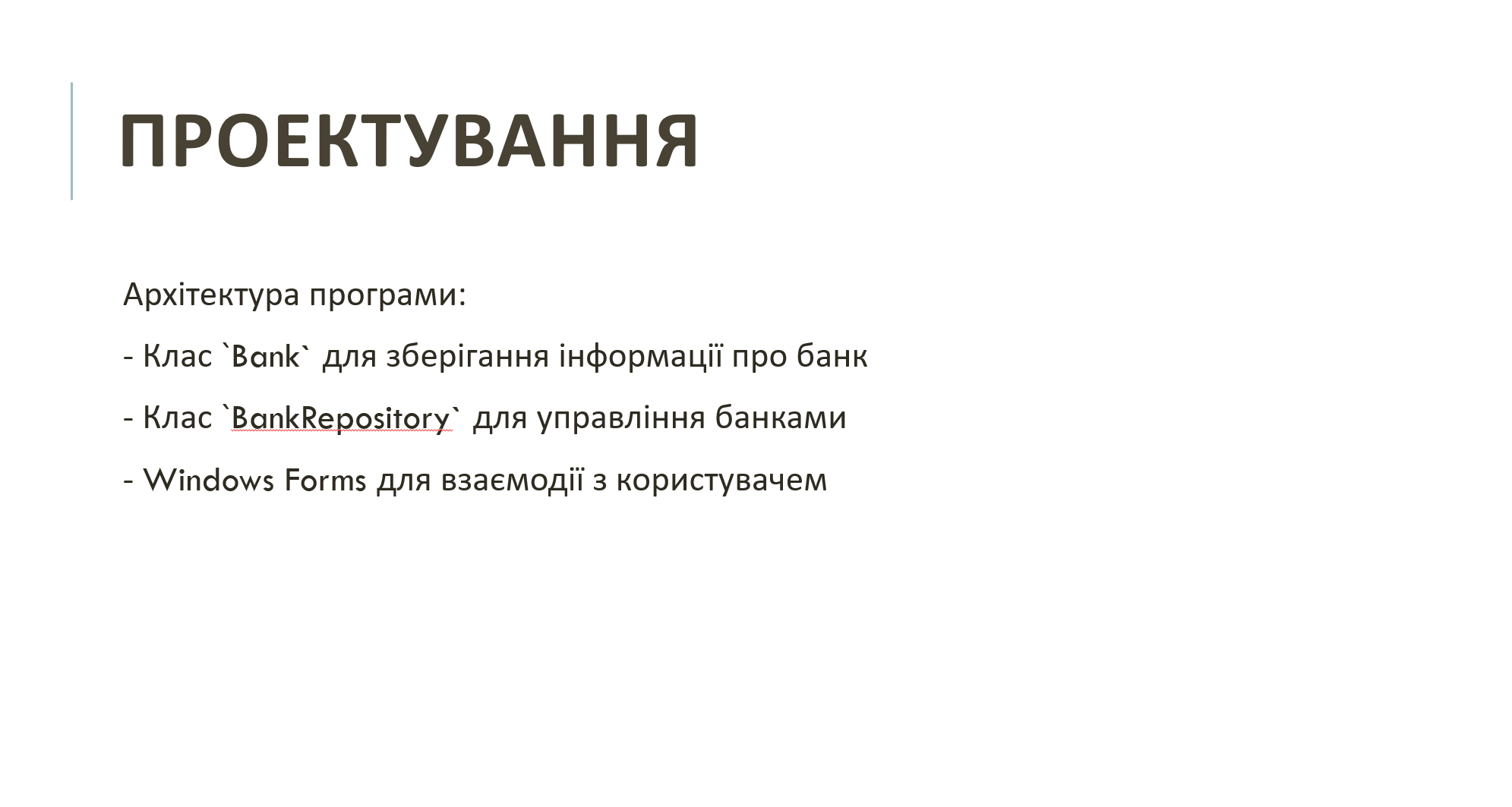
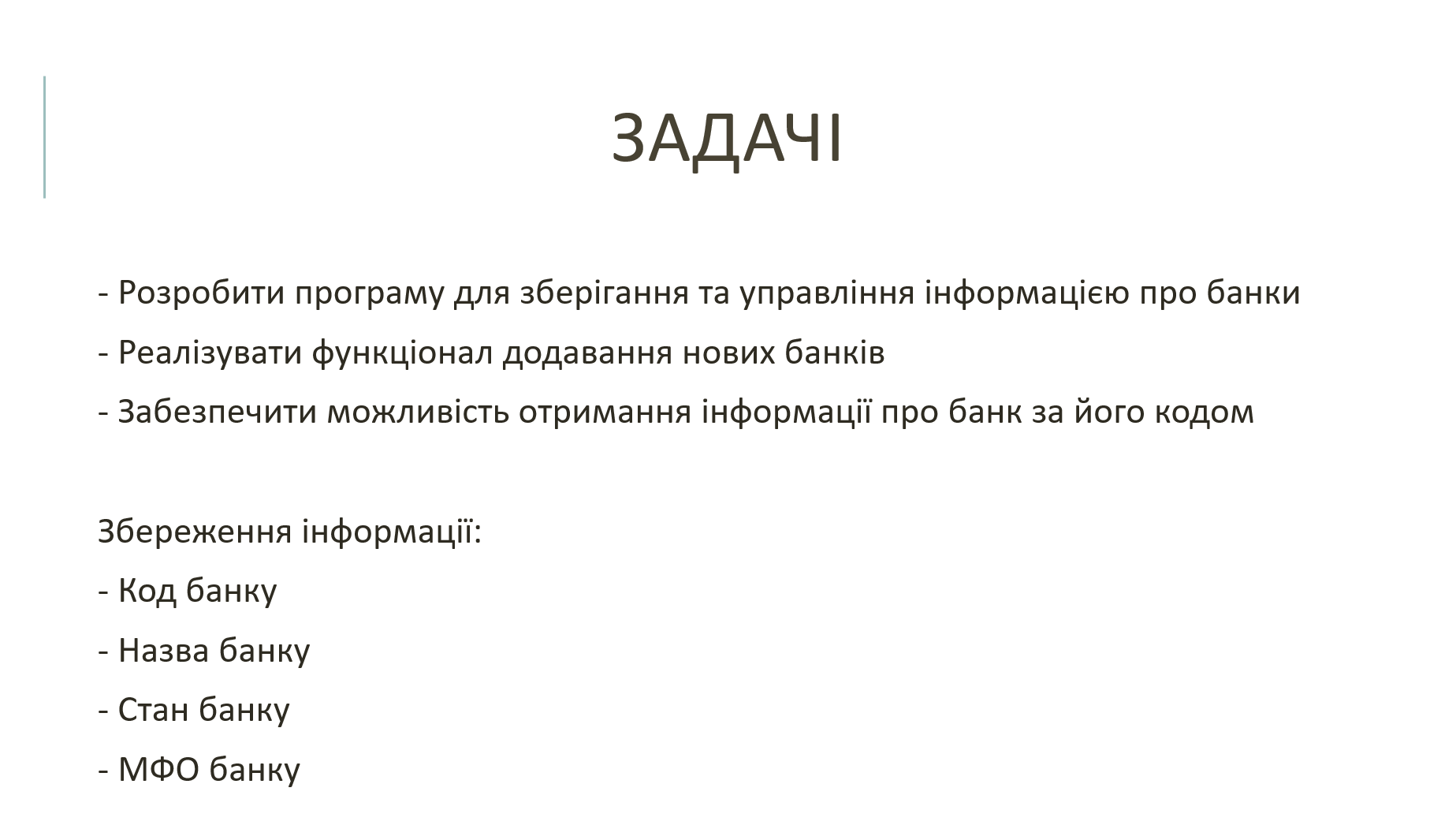
**Додаток Г.**

**Презентація**

****

****

****

****

