

Міністерство освіти і науки України
Національний університет «Запорізька Політехніка»

Кафедра програмних засобів

ЗВІТ

з самостійної роботи №2

з дисципліни «Системний аналіз» на тему:

«Автоматизована система купівлі авіаквитків»

Виконав:

Студент КНТ-122

О. А. Онищенко

Прийняли:

Викладач

В. М. Льовкін

2024

АВТОМАТИЗОВАНА СИСТЕМА КУПІВЛІ АВІАКВИТКІВ

1 Вступ

Ландшафт авіаперевезень суттєво змінився з появою автоматизованих систем продажу авіаквитків, які спростили процес бронювання та покращили якість обслуговування клієнтів. Незважаючи на значний прогрес, що призвів до ефективних процесів бронювання та оплати, залишаються проблеми з інтеграцією цих систем з динамічними моделями ціноутворення та багатоканальними мережами дистрибуції.

Провідні компанії в цій галузі, такі як Amadeus, Sabre і Travelport, продовжують впроваджувати інновації, спираючись на ідеї таких експертів галузі, як Алекс Кремер і Генрі Хартевельдт. Світові тенденції свідчать про перехід до більш персоналізованих і зручних інтерфейсів, використання штучного інтелекту для прогнозування поведінки споживачів та оптимізації продажів.

Актуальність цього дослідження полягає в тому, що воно може заповнити існуючі прогалини, пропонуючи систему, яка не тільки спрощує транзакції, але й пропонує предиктивну аналітику для управління запасами. Метою цієї роботи є розробка комплексного рішення, яке задовольнить потреби як авіакомпаній, так і мандрівників, і може бути застосоване на різних платформах і пристроях.

Ця робота ґрунтується на попередніх дослідженнях у цій галузі, спрямованих на синтез найкращих практик та впровадження нових підходів до дизайну та функціональності системи. Вона є свідченням постійного розвитку технологій авіаперевезень, спрямованих на значний прогрес у цій галузі.

2 Основна Частина

2.1 Проектування Системи

2.1.1 Вимоги до програмного забезпечення

Вимоги до програмного забезпечення є наступними:

- Середовище розробки: Visual Studio
- Мова програмування: C#
- Фреймворк: .NET для WinForms
- Файли даних: Текстові файли (flights.txt, tickets.txt, users.txt)

2.1.1.1 Функціональні вимоги до програмного забезпечення

Програма має виконувати наступні функції:

- вхід існуючих користувачів
- реєстрація нових користувачів
- різні повноваження доступу для адміністраторів при перегляді

квитків

- перегляд наявного списку квитків
- зміна параметрів квитків: додавання опціональних сервісів, зміна

місця, тощо

- замовлення квитків користувачем
- перегляд замовлених квитків
- вихід з облікового запису

2.1.2 Вхідні та вихідні дані

Вхідні та вихідні дані є наступними:

- Вхідні: Дані, введені користувачем через форми графічного інтерфейсу.
- Вихідні: Відображення даних у графічному інтерфейсі, оновлені текстові файли.

2.1.3 Спосіб роботи з даними

Спосіб роботи з даними є прямолінійним: користувач взаємодіє з графічним інтерфейсом, в залежності від виконаних дій до файлів даних надходять запити про створення об'єктів на основі класів даних. Після створення об'єкти заносяться у файли даних, а файли зберігаються. При потребі, в залежності від виконаних дій, з файлів даних робиться вибірка, а ці дані подаються через форми графічного інтерфейсу до користувача.

Організація даних виглядає наступним чином (Рис. 1.1):

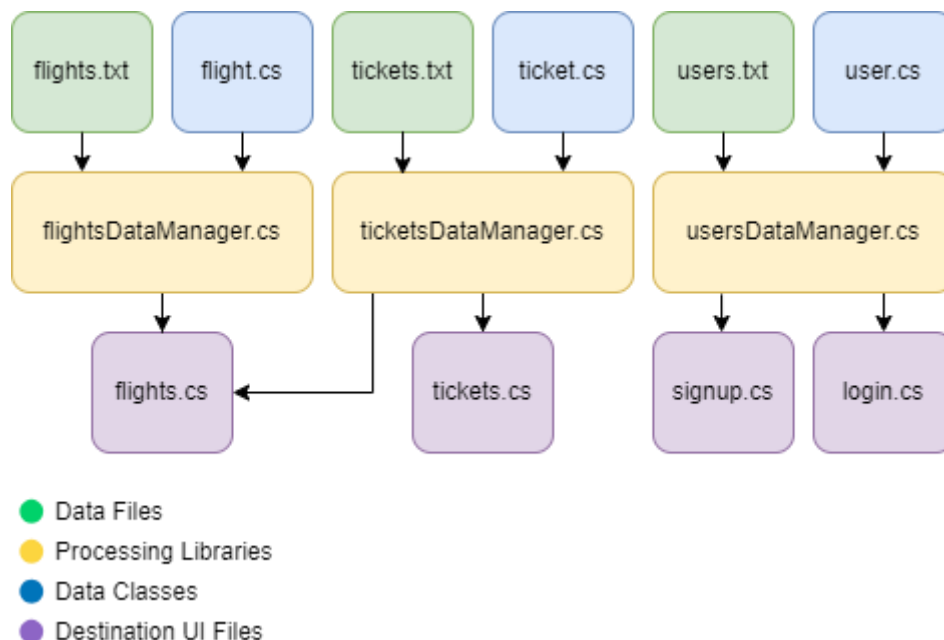


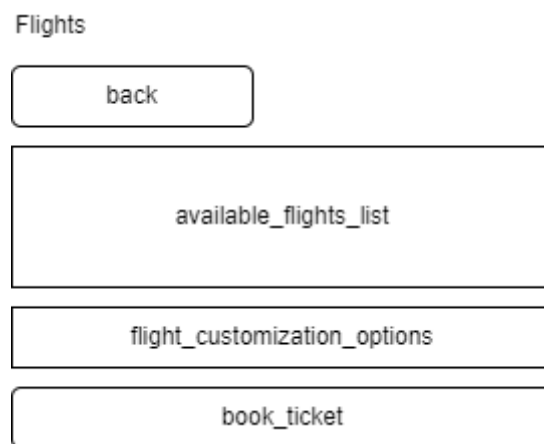
Рисунок 1.1 – Організація даних

2.1.4 Прототип графічного інтерфейсу користувача

Схематичний прототип графічного інтерфейсу користувача:


- Login Form
- Signup Form
- Menu Form
- Flights Form
- Tickets Form

Нижче наведено графічні форми прототипи графічного інтерфейсу користувача (Рис. 1.2-7):



The diagram shows a vertical stack of four rectangular boxes. The top box is labeled 'back'. The second box is labeled 'available_flights_list'. The third box is labeled 'flight_customization_options'. The bottom box is labeled 'book_ticket'.

Рисунок 1.2 – Прототип форми Flights



The diagram shows a vertical stack of four rectangular boxes. The first box is labeled 'user_name'. The second box is labeled 'user_password'. The third box is labeled 'login'. The bottom box is labeled 'sign up'.

Рисунок 1.3 – Прототип форми Login

Menu

tickets

flights

sign out

Рисунок 1.4 – Прототип форми Menu

Signup

user_name

user_password

☒ is_admin

sign up

login

Рисунок 1.5 – Прототип форми Signup

Tickets

back

tickets_data_table

Рисунок 1.6 – Прототип форми Tickets

- Form Name (plain text)
- Buttons (rounded corners)
- Data Fields or Inputs (square corners)

Рисунок 1.7 – Значення різних фігур на прототипах

2.1.5 Система об'єктів

Всі класи даних та обробники даних з їх інтерфейсами знаходяться в окремій бібліотеці класів `Class_Library`, з якої ми імпортуємо скомпільований `.dll` файл, аби потім використовувати їх в інтерфейсі нашої програми.

Система об'єктів, класів даних та обробників даних схематично виглядає наступним чином:

```
Class_Library // бібліотека класів даних та класів обробників даних
currentUser.cs // клас даних поточного користувача
    string name; // ім'я користувача
    string password; // пароль користувача
    bool isAdmin; // чи є користувач адміністратором
flight.cs // клас даних рейсу
    string name; // назва рейсу
    int price; // стандартна ціна рейсу
    string date; // дата рейсу
    int seats; // кількість місць
flightsDataManager.cs // клас обробник даних рейсів
    Interface IflightsDataManager; // інтерфейс
    List<flight> loadFlights(); // завантажує дані рейсів
    flight getFlight(string name); // повертає рейс за назвою
ticket.cs // клас даних квитка
    string userName; // ім'я користувача
    string flightName; // назва рейсу
    int price; // загальна ціна квитка
    string date; // дата рейсу
    int seatRow; // обраний ряд місця
    bool isMiddle; // чи місце у середньому ряду
    bool isWindow; // чи місце біля вікна
    bool isPrivate; // чи місце приватне
    bool isBaggage; // чи потрібне додаткове місце для багажу
    bool isMeal; // чи потрібне харчування на борту
ticketsDataManager.cs // клас обробник даних квитків
    Interface IticketsDataManager; // інтерфейс
    List<ticket> loadTickets(); // завантажує дані квитків
    ticket GetTicket(string flightName); // повертає квиток за назвою
    рейсу
    List<ticket> GetOwnTickets(string userName); // повертає квитки
    користувача
```

```

    void AddTicket(string userName, string flightName, int price,
string date, int seatRow, bool isMiddle, bool isWindow, bool isPrivate,
bool isBaggage, bool isMeal); // додає новий квиток за всіма даними
user.cs // клас даних користувача
    string name; // ім'я користувача
    string password; // пароль користувача
    bool isAdmin; // чи є користувач адміністратором
userManager.cs // клас обробник даних користувачів
    Interface IUserDataManager; // інтерфейс
    List<user> loadUsers(); // завантажує дані користувачів
    void addUser(string name, string password, bool isAdmin); // додає
нового користувача
    user getUser(string name); // повертає користувача за ім'ям
    bool isAdmin(string name); // повертає чи є користувач
адміністратором
    bool validateCredentials(string name, string password); // повертає
чи валідні дані користувача
App // містить форми користувацького інтерфейсу та файли даних
    flights.txt // дані рейсів
        flight_name,price,date,seats // назва рейсу, ціна, дата,
кількість місць
        // продовжується з нового рядку так само
    tickets.txt // дані квитків
        user_name,flight_name,price,date,seat_row,is_middle,is_window,is
_private,is_baggage,is_meal // ім'я користувача, назва рейсу, ціна,
дата, ряд місця, чи середній ряд, чи місце біля вікна, чи місце
приватне, чи потрібен додатковий багаж, чи потрібне харчування
        // продовжується з нового рядку так само
    users.txt // дані користувачів
        name,password,is_admin // ім'я користувача, пароль, чи є
адміністратором
        // продовжується з нового рядку так само
    flights.cs // форма перегляду рейсів
    login.cs // форма входу
    menu.cs // форма меню
    signup.cs // форма реєстрації
    tickets.cs // форма перегляду квитків

```

У графічному вигляді схема даних виглядає наступним чином (Рис. 1.8):

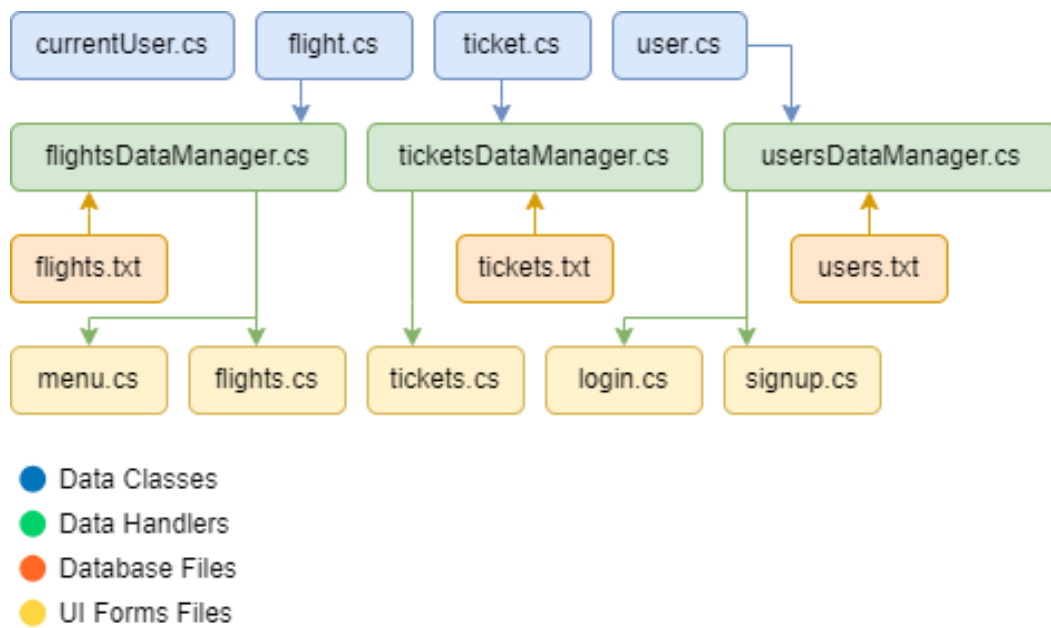


Рисунок 1.8 – Схема даних у графічному вигляді

2.1.6 Функціональні моделі процесів

Основні функціональні моделі подано у графічному вигляді нижче (Рис. 1.9-11)

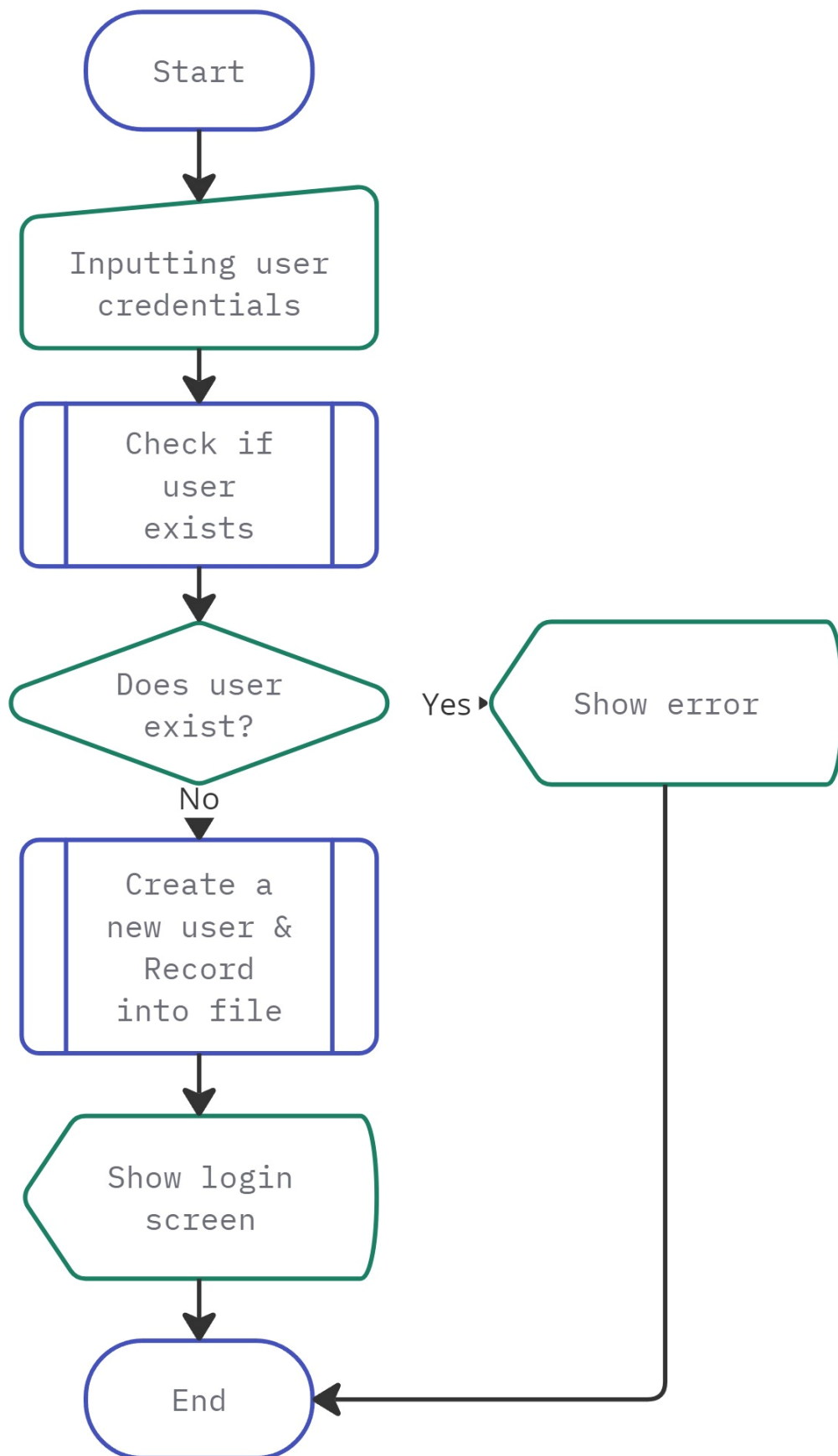


Рисунок 1.9 – Процес реєстрації нового користувача

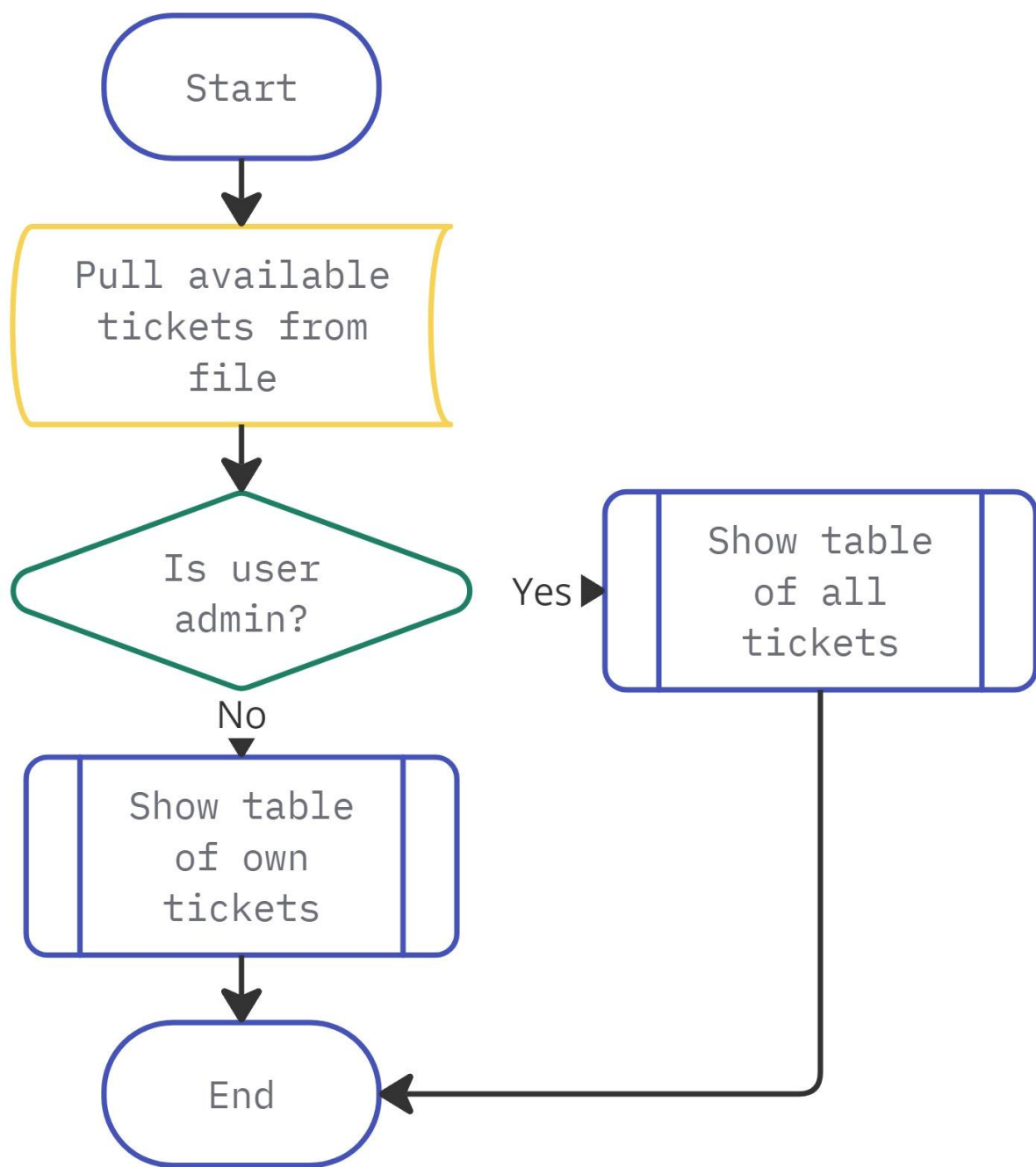


Рисунок 1.10 – Процес перегляду квитків

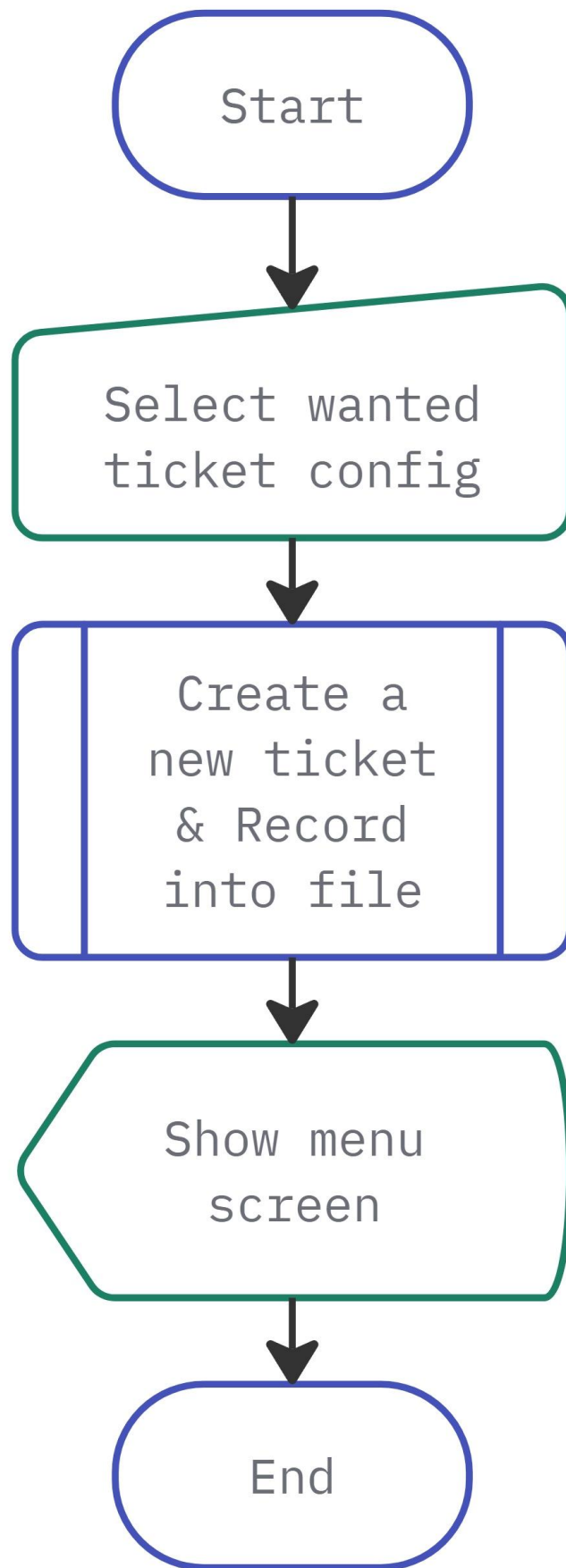


Рисунок 1.11 – Процес замовлення квитку

2.2 Опис Програми

2.2.1 Структурна схема

Структурна схема програмного забезпечення має наступний вигляд (Рис. 2.2):

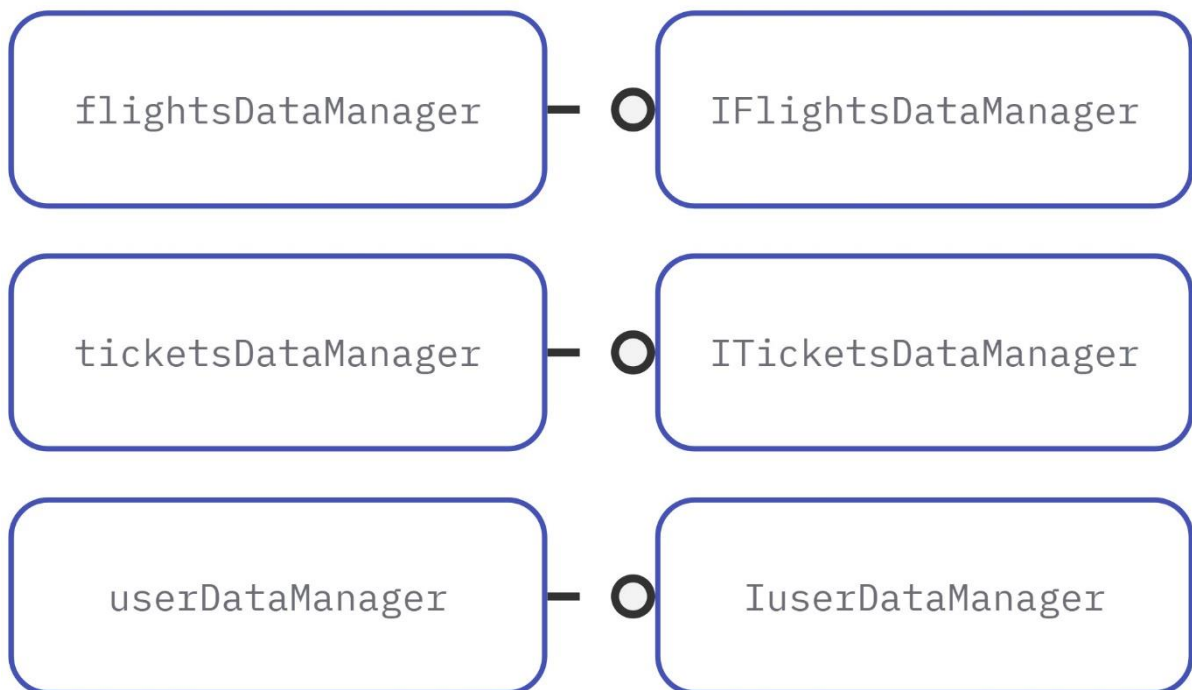


Рисунок 2.2 – Структурна схема програмного забезпечення

2.2.2 Опис модулів

Структуру проєкту можна умовно поділити на чотири секції:

- обробники даних
- класи даних
- файли даних
- форми інтерфейсу користувача

Де обробники даних та класи даних належать до окремої бібліотеки `Class_Library`, а файли даних та форми користувацького інтерфейсу лежать в основній теці `App`.

2.2.2.1 Обробники даних

Класи обробники даних, що містяться у програмі, описані нижче (пункти 2.2.2.1.1-3).

2.2.2.1.1 *ticketsDataManager.cs*

Обробник даних квитків містить наступні методи:

- завантажує список усіх квитків: `List<ticket> loadTickets();`
- повертає квиток за назвою рейсу: `ticket GetTicket(string flightName);`
- повертає квиток за ім'ям користувача: `List<ticket> GetOwnTickets(string userName);`
- створює та додає новий квиток: `void AddTicket(string userName, string flightName, int price, string date, int seatRow, bool isMiddle, bool isWindow, bool isPrivate, bool isBaggage, bool isMeal);`

2.2.2.1.2 *userDataManager.cs*

Обробник даних користувачів містить наступні методи:

- повертає список усіх користувачів: `List<user> loadUsers();`
- створює та додає нового користувача: `void addUser(string name, string password, bool isAdmin);`
- повертає користувача за ім'ям: `user getUser(string name);`
- визначає чи є переданий користувач адміністратором: `bool isAdmin(string name);`

- визначає чи є передані облікові дані вірними, тобто чи існує запис з такими даними у системі: `bool validateCredentials(string name, string password);`

2.2.2.1.3 flightsDataManager.cs

Обробник даних польотів або рейсів містить наступні методи:

- повертає усі польоти: `List<flight> loadFlights();`
- повертає польот за назвою: `flight getFlight(string name);`

2.2.2.2 Класи даних

Класи даних, що містяться у програмі описані нижче (пункти 2.2.2.2.1-4).

2.2.2.2.1 currentUser.cs

Клас даних поточного користувача містить наступні поля:

- ім'я поточного користувача: `string name;`
- пароль поточного користувача: `string password;`
- чи є поточний користувач адміністратором: `bool isAdmin;`

2.2.2.2.2 user.cs

Клас даних користувача містить наступні поля, аналогічні полям поточного користувача:

- ім'я користувача: `string name;`
- пароль користувача: `string password;`
- чи є користувач адміністратором: `bool isAdmin;`

2.2.2.2.3 *flight.cs*

Клас даних польоту містить наступні поля:

- назва польоту: `string name;`
- стандартна ціна квитка: `int price;`
- дата польоту: `string date;`
- кількість місць на борту: `int seats;`

2.2.2.2.4 *ticket.cs*

Клас даних квитку містить наступні поля:

- ім'я користувача, що придбав квиток: `string userName;`
- назва польоту: `string flightName;`
- загальна ціна квитка: `int price;`
- дата польоту: `string date;`
- обраний ряд сидіння: `int seatRow;`
- чи обраний середній ряд: `bool isMiddle;`
- чи обране сидіння біля вікна: `bool isWindow;`
- чи приватне сидіння: `bool isPrivate;`
- чи потрібне додаткове місце для багажу: `bool isBaggage;`
- чи потрібне харчування на борту: `bool isMeal;`

2.2.2.3 Файли даних

Файли даних, що містяться у системі, визначені нижче (пункти 2.2.2.3.1-3)

2.2.2.3.1 *flights.txt*

Кожен рядок файлу польотів містить таку схему запису об'єкту:

```
flight_name,price,date,seats
```

2.2.2.3.2 *tickets.txt*

Кожен рядок файлу квитків містить наступну схему запису:

```
user_name,flight_name,price,date,seat_row,is_middle,is_window,is  
_private,is_baggage,is_meal
```

2.2.2.3.3 *users.txt*

Кожен рядок файлу користувачів містить наступну схему запису:

```
name,password,is_admin
```

2.2.2.4 Форми користувацького інтерфейсу

Форми користувацького інтерфейсу, що містяться в системі, описані нижче (пункти 2.2.2.4.1-5)

2.2.2.4.1 *flights.cs*

Форма відображення наявних польотів містить наступні глобальні поля:

- фіксована висота нерозгорнутого вікна: `const int collapsedHeight = 190;`
- висота розгорнутого вікна: `const int expandedHeight = 560;`
- поточний обраний рейс: `flight selectedFlight;`
- сумарна ціна квитку: `int totalPrice;`
- ціна обраних опцій сидіння: `int seatTypePrice = 0;`

- ціна обраних опцій харчування: `int mealPrice = 0;`
- ціна обраних опцій багажу: `int luggagePrice = 0;`
- ціна обраних опцій приватного місця: `int privatePrice = 0;`

Форма також містить наступні методи:

- конструктор форми, згортає висоту форми, підвантажує польоти:

```
public flights();
```

- оновлює сумарну вартість квитку: `void updateTotal();`

- виводить список наявних рейсів: `void`

```
renderFlightNames(List<flight> flights);
```

- подія натискання кнопки назад: `void menuButton_Click(object sender, EventArgs e);`

- обробляє вибір польоту зі списку: `void`

```
flightsList_SelectedValueChanged(object sender, EventArgs e);
```

- подія натискання радіо кнопки: `void`

```
middleRadio_CheckedChanged(object sender, EventArgs e);
```

- подія натискання радіо кнопки: `void`

```
randomRadio_CheckedChanged(object sender, EventArgs e);
```

- подія натискання радіо кнопки: `void`

```
windowRadio_CheckedChanged(object sender, EventArgs e);
```

- подія натискання кнопки пташки: `void`

```
mealBox_CheckedChanged(object sender, EventArgs e);
```

- подія натискання кнопки пташки: `void`

```
luggageBox_CheckedChanged(object sender, EventArgs e);
```

- подія натискання кнопки пташки: `void`

```
privateBox_CheckedChanged(object sender, EventArgs e);
```

- подія натискання кнопки замовлення: `void`

```
bookButton_Click(object sender, EventArgs e);
```

2.2.2.4.2 login.cs

Форма логіну містить наступні методи:

- конструктор: `public login();`

- подія натискання кнопки вхід, перевіряє облікові дані: `void`

```
loginButton_Click(object sender, EventArgs e);
```

- подія натискання кнопки реєстрації, відкриває вікно реєстрації:

```
void registerButton_Click(object sender, EventArgs e);
```

2.2.2.4.3 menu.cs

Форма меню містить наступні методи:

- конструктор: `public menu();`

- подія натискання кнопки виходу: `void signoutButton_Click(object sender, EventArgs e);`

- подія натискання кнопки польотів: `void flightsButton_Click(object sender, EventArgs e);`

- подія натискання кнопки квитків: `void ticketsButton_Click(object sender, EventArgs e);`

2.2.2.4.4 signup.cs

Форма реєстрації містить наступні методи:

- конструктор: `public signup();`

- відкриває форму логіну, закриває поточну: `void loginButton_Click(object sender, EventArgs e);`

- створює нового користувача, переходить до форми логіну: `void signupButton_Click(object sender, EventArgs e);`

2.2.2.4.5 tickets.cs

Форма квитків містить наступні методи:

- підвантажує список квитків: `public tickets();`

- завантажує список квитків та виводить їх: `void loadTickets();`

- закриває цю форму, відкриває меню: `void`

```
backButton_Click(object sender, EventArgs e);
```

2.2.3 Опис роботи

Робота з програмою завжди починається з вікна входу (Рис. 3.1):

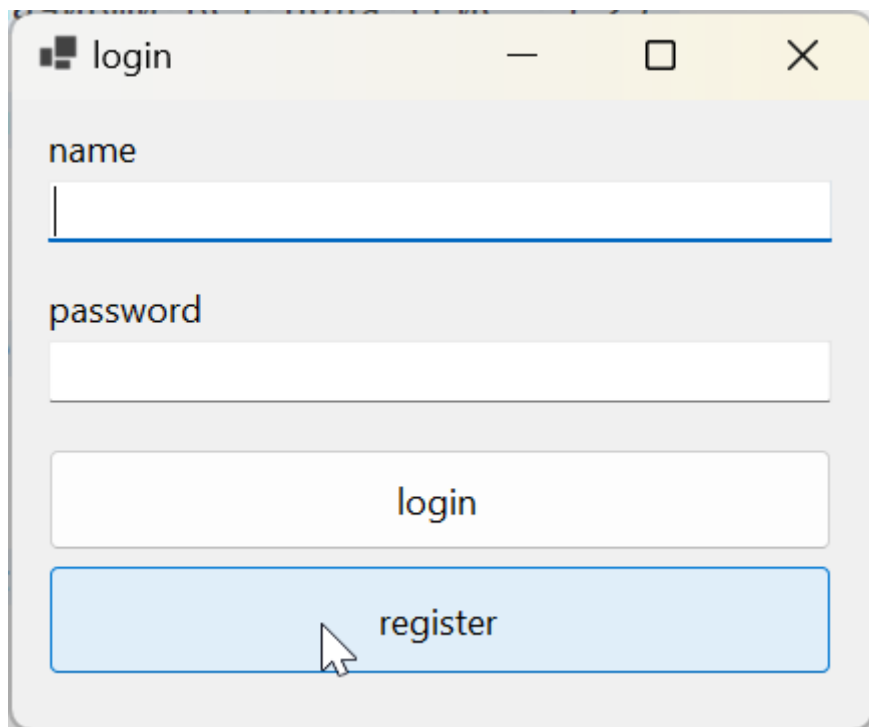
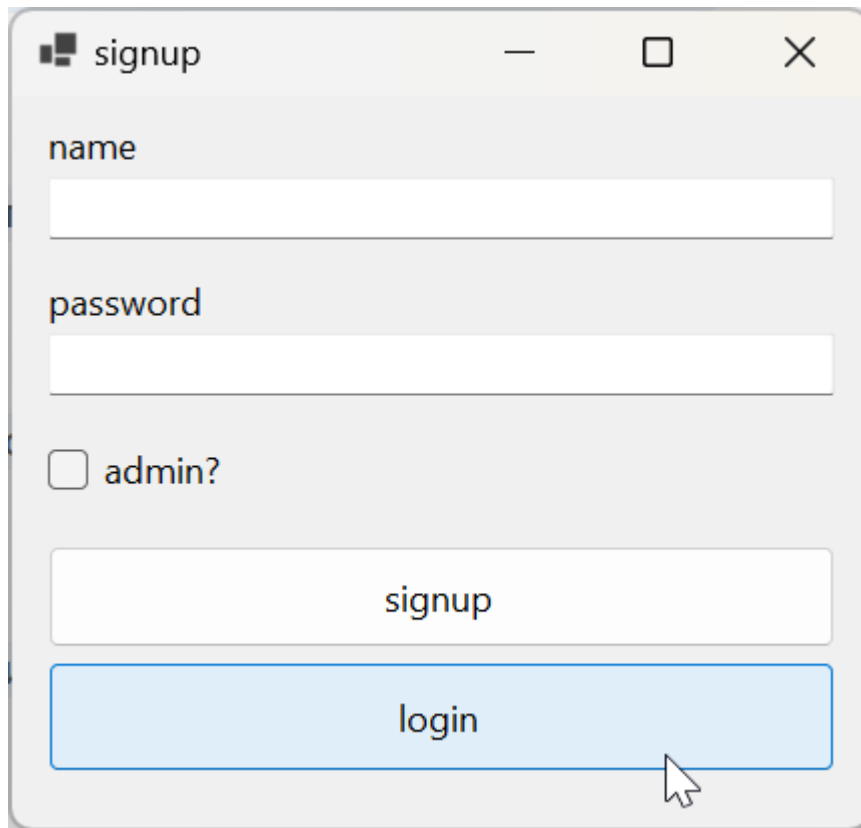
The image shows a standard Windows-style application window titled "login". The window has a light gray background and rounded corners. At the top, there is a title bar with the text "login" and standard minimize, maximize, and close buttons. Below the title bar, the window contains two text input fields. The first field is labeled "name" and the second is labeled "password". Below the "password" field, there are two buttons. The first button is white with the text "login" in black. The second button is blue with the text "register" in white. A mouse cursor is pointing at the "register" button.

Рисунок 3.1 – Вікно входу до програми

У вікні входу користувач може зайти у свій обліковий запис. Якщо користувач облікового запису не має, він може перейти до вікна реєстрації (Рис. 3.2)



The image shows a web browser window with the title 'signup'. The window has a standard macOS-style title bar with a red close button, a yellow maximize button, and a green window control button. The main content area is a light gray form. It contains two text input fields, one labeled 'name' and one labeled 'password'. Below the 'password' field is a checkbox labeled 'admin?'. At the bottom of the form are two buttons: a white button labeled 'signup' and a blue button labeled 'login'. A mouse cursor is hovering over the 'login' button.

Рисунок 3.2 – Вікно реєстрації нового користувача

У вікні реєстрації користувач може створити обліковий запис, зазначивши всі поля (Рис. 3.2)

Після входу або реєстрації, користувач потрапляє до головного меню (Рис. 3.3)

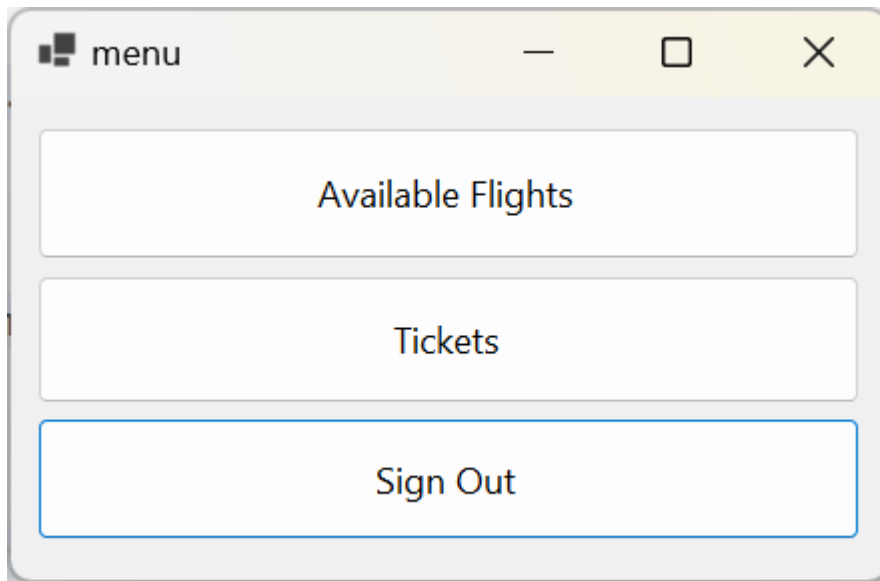


Рисунок 3.3 – Головне меню програми

У головному меню користувач може вийти з облікового запису, що перенесе його назад до вікна входу (Рис. 3.4)

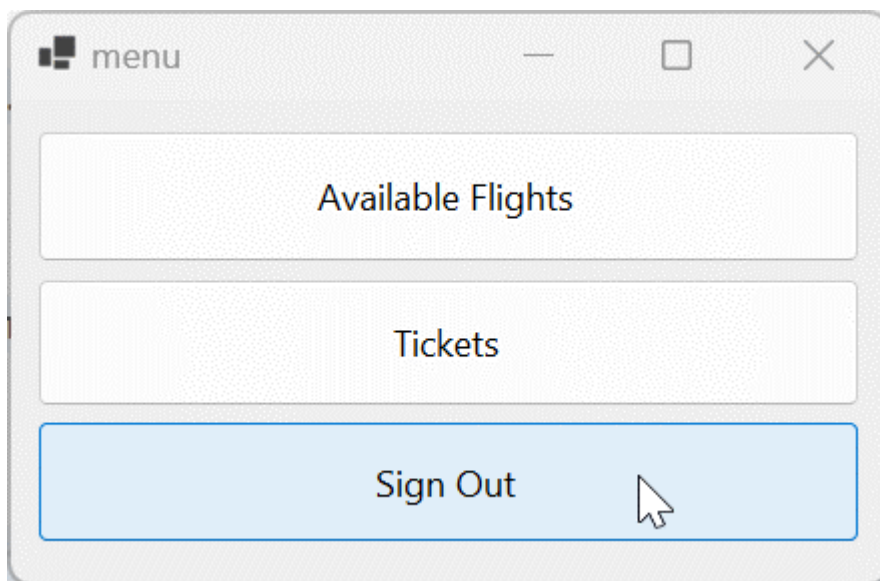


Рисунок 3.4 – Кнопка виходу з облікового запису

У головному меню користувач може переглянути список польотів (Рис. 3.5)

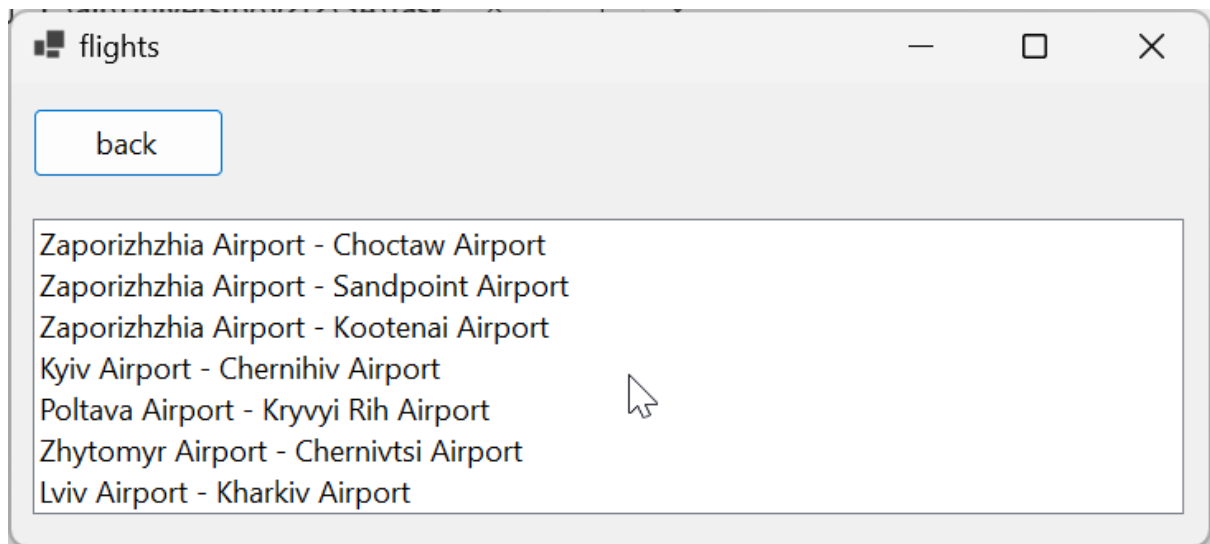


Рисунок 3.5 – Список доступних польотів

У списку польотів користувач може обрати підходящий польот, змінити параметри та замовити квиток (Рис. 3.6)

flights

back

Zaporizhzhia Airport - Choctaw Airport

Zaporizhzhia Airport - Sandpoint Airport

Zaporizhzhia Airport - Kootenai Airport

Kyiv Airport - Chernihiv Airport

Poltava Airport - Kryvyi Rih Airport

Zhytomyr Airport - Chernivtsi Airport

Lviv Airport - Kharkiv Airport

Zaporizhzhia Airport - Kootenai Airport

Ticket price: \$30

Flight date: 13/05/2024

Available seats: 77

Select seat

Select seat row

7

Select seat type

☐ Random seat | +\$0

☒ Middle row | +\$10

☐ Window seat | +\$30

Select optional extra services

☐ Meal on board | +\$15

☐ Extra luggage space | +\$30

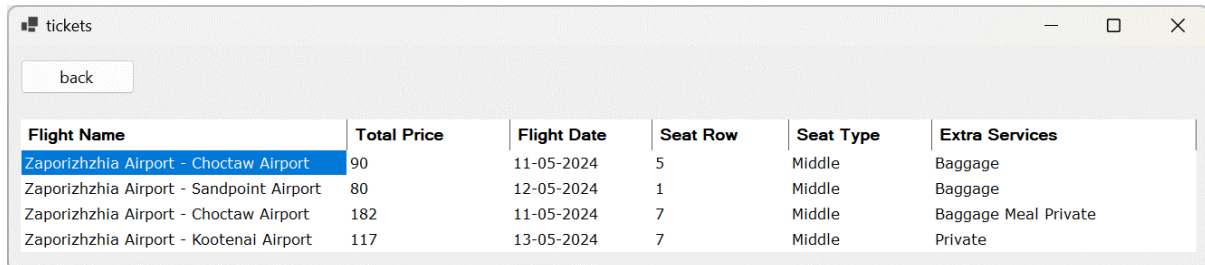
☒ Private compartment | +\$77

In total: \$117

Book Ticket

Рисунок 3.6 – Кнопка замовлення квитка

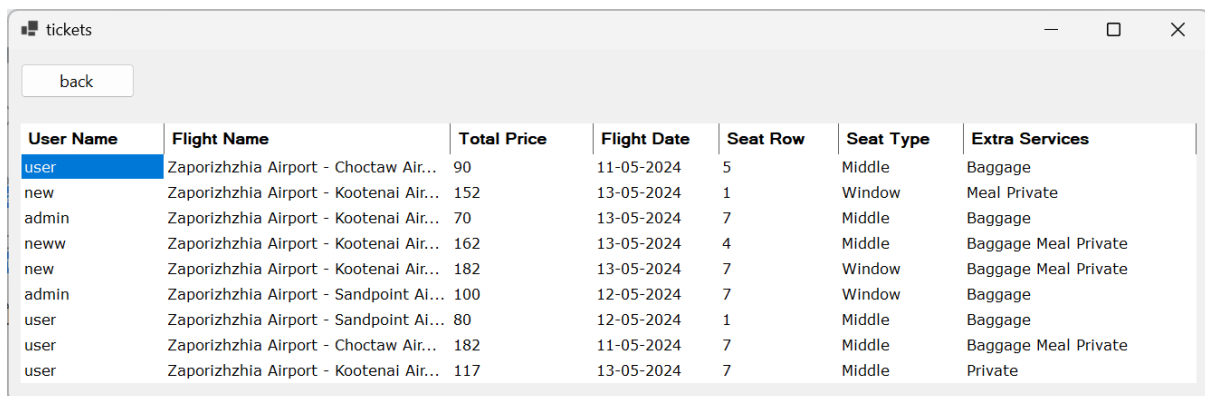
Після замовлення квитка користувача переносить до головного меню, де він може переглянути свої квитки (Рис. 3.7)



Flight Name	Total Price	Flight Date	Seat Row	Seat Type	Extra Services
Zaporizhzhia Airport - Choctaw Airport	90	11-05-2024	5	Middle	Baggage
Zaporizhzhia Airport - Sandpoint Airport	80	12-05-2024	1	Middle	Baggage
Zaporizhzhia Airport - Choctaw Airport	182	11-05-2024	7	Middle	Baggage Meal Private
Zaporizhzhia Airport - Kootenai Airport	117	13-05-2024	7	Middle	Private

Рисунок 3.7 – Список квитків користувача

Якщо користувач є адміністратором, йому будуть доступні квитки всіх користувачів (Рис. 3.8)



User Name	Flight Name	Total Price	Flight Date	Seat Row	Seat Type	Extra Services
user	Zaporizhzhia Airport - Choctaw Air...	90	11-05-2024	5	Middle	Baggage
new	Zaporizhzhia Airport - Kootenai Air...	152	13-05-2024	1	Window	Meal Private
admin	Zaporizhzhia Airport - Kootenai Air...	70	13-05-2024	7	Middle	Baggage
neww	Zaporizhzhia Airport - Kootenai Air...	162	13-05-2024	4	Middle	Baggage Meal Private
new	Zaporizhzhia Airport - Kootenai Air...	182	13-05-2024	7	Window	Baggage Meal Private
admin	Zaporizhzhia Airport - Sandpoint Ai...	100	12-05-2024	7	Window	Baggage
user	Zaporizhzhia Airport - Sandpoint Ai...	80	12-05-2024	1	Middle	Baggage
user	Zaporizhzhia Airport - Choctaw Air...	182	11-05-2024	7	Middle	Baggage Meal Private
user	Zaporizhzhia Airport - Kootenai Air...	117	13-05-2024	7	Middle	Private

Рисунок 3.8 – Список квитків для адміністратора

2.2.4 Опис повідомлень

Програма містить наступні повідомлення (Рис. 4.1-4):

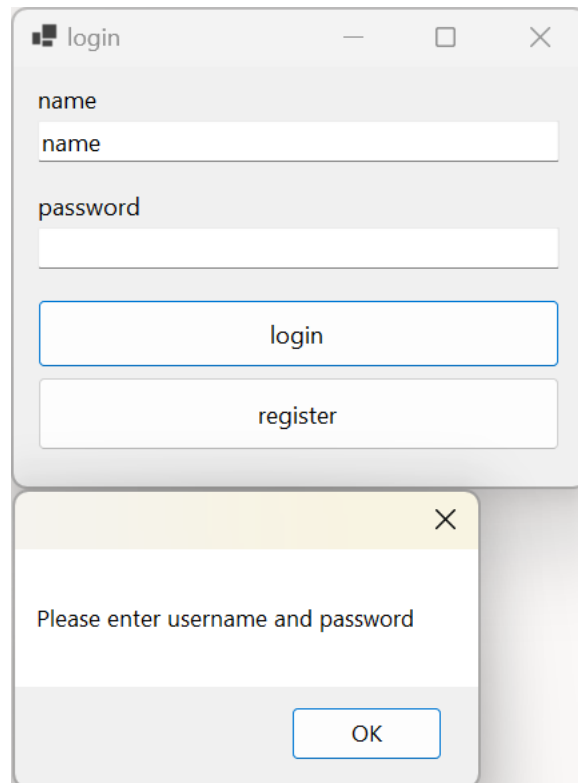


Рисунок 4.1 – Помилка Відсутній пароль або ім'я

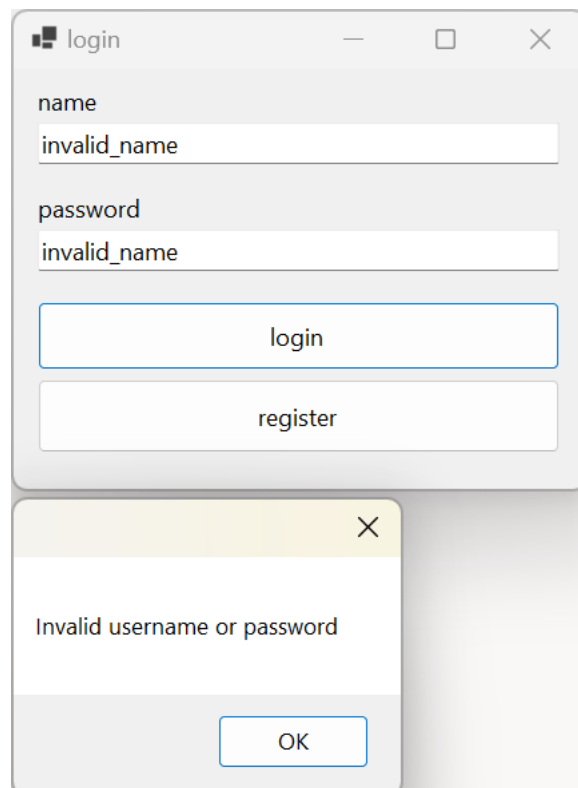


Рисунок 4.2 – Помилка Некоректні облікові дані

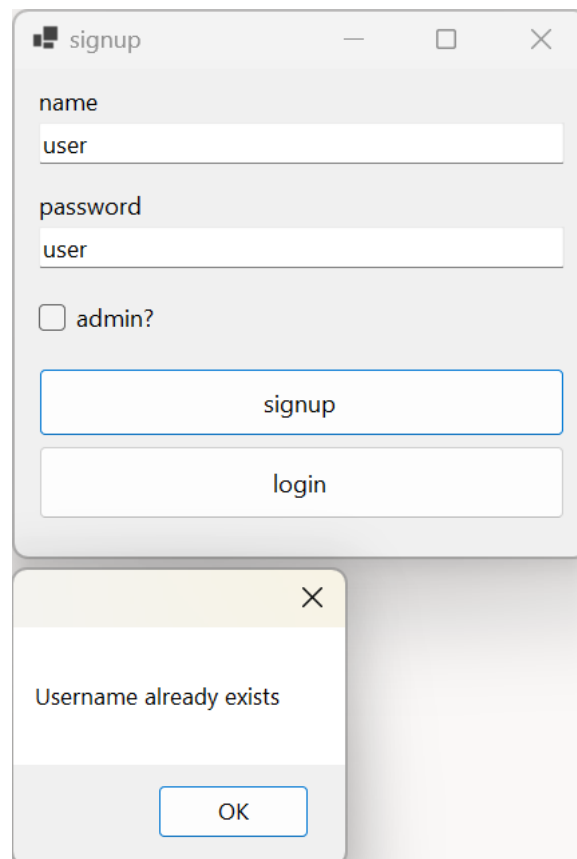


Рисунок 4.3 – Помилка Користувач вже існує

flights

back

Zaporizhzhia Airport - Choctaw Airport

Zaporizhzhia Airport - Sandpoint Airport

Zaporizhzhia Airport - Kootenai Airport

Kyiv Airport - Chernihiv Airport

Poltava Airport - Kryvyi Rih Airport

Zhytomyr Airport - Chernivtsi Airport

Lviv Airport - Kharkiv Airport

Zaporizhzhia Airport - Choctaw Airport

Ticket price: \$50

Flight date: 11/05/2024

Available seats: 77

Select seat

Select seat row

7

Select seat type

Random seat | +\$0

Middle row | +\$10

Window seat | +\$30

Select optional extra services

Meal on board | +\$15

Extra luggage space | +\$30

Private compartment | +\$77

In total: \$182

Book Ticket

×

Ticket booked successfully

OK

Рисунок 4.4 – Підтвердження покупки квитка

2.3 Управління Ризиками

2.3.1 Виявлення ризиків

Виявимо ризики у проєкті:

- Корупція файлів даних:
 - Опис: Файли даних, що зберігають інформацію про рейси та квитки, може бути пошкоджена через апаратні збої, програмні помилки або інші проблеми.
 - Сфери впливу: Цілісність даних, доступність системи.
- Зміни обсягу робіт:
 - Опис: Зміни у вимогах або обсязі проєкту можуть призвести до додаткової роботи, затримок або невідповідності початковим цілям.
 - Сфери впливу: План-графік проєкту, зусилля з розробки.
- Економічна нестабільність:
 - Опис: Економічні коливання (наприклад, інфляція, девальвація валюти) можуть вплинути на витрати та фінансування проєкту.
 - Сфери впливу: Бюджет проєкту, фінансова стабільність.

2.3.2 Ймовірність ризиків

Оцінимо ймовірність кожного ризику:

- Корупція файлів даних:
 - Якісний: Помірний (завдяки регулярному резервному копіюванню та надійному управлінню взаємодії з файлами даних).
- Зміни обсягу робіт:
 - Якісний: Високий (зміни обсягу робіт є поширеним явищем у програмних проєктах).
- Економічна нестабільність:

- Якісний: Дуже високий (зовнішні фактори поза контролем людей).

2.3.3 Вплив ризиків

Оцінимо вплив на аспекти проєкту:

- Корупція файлів даних:
 - Якісний: Помірний (зусилля з відновлення даних, простої системи).
- Зміни обсягу робіт:
 - Якісний: Високий (доопрацювання, затримки).
- Економічна нестабільність:
 - Якісний: Високий (коригування бюджету, розподіл ресурсів).

2.3.4 Оцінка до управління

Поєднаємо ймовірності та вплив:

- Корупція файлів даних: Важливий (помірна ймовірність, помірний вплив).
- Зміни обсягу робіт: Критичний (висока ймовірність, високий вплив).
- Економічна нестабільність: Критичний (дуже висока ймовірність, високий вплив).

2.3.5 Оцінка після управління

Оцінимо ризики після управлінню:

- Корупція файлів даних: Мінімізований (стратегії резервного копіювання, моніторинг).
- Зміни обсягу робіт: Заходи в разі непередбачуваних ситуацій (процес управління змінами).
- Економічна нестабільність: Моніторинг та адаптація (фінансові резерви).

3 Висновки

*Бо заплата за гріх смерть, а благодатний дар
Божий вічне життя в Христі Ісусі, Господі нашім
([Римляни 6:23](#))*

На закінчення, розробка передової автоматизованої системи продажу квитків являє собою значний стрибок вперед в індустрії авіаперевезень. Система, розроблена в цій роботі, використовуючи описану структуру проекту, пропонує надійне рішення, яке вирішує поточні проблеми у сфері бронювання та ціноутворення, забезпечуючи при цьому предиктивну аналітику для управління запасами.

Інтеграція зручного інтерфейсу з можливостями прогнозування на основі штучного інтелекту гарантує, що авіакомпанії зможуть пропонувати персоналізований досвід для мандрівників, оптимізуючи продажі та задоволеність клієнтів. Наукова та соціальна значущість цієї роботи полягає в тому, що вона може революціонізувати спосіб взаємодії авіакомпаній та мандрівників, зробивши авіаперевезення більш доступними та ефективними.

Якісні та кількісні показники успіху цієї системи включають покращення простоти транзакцій, підвищення точності управління запасами та покращення користувацького досвіду. Надійність цих

результатів ґрунтується на комплексному підході, застосованому при розробці системи, що відображено в структурі проекту.

Рекомендації щодо впровадження включають поетапне розгортання системи на різних платформах з постійним моніторингом та механізмами зворотного зв'язку для забезпечення адаптивності до мінливих потреб ринку. Ця робота не тільки сприяє розширенню знань у галузі технологій повітряних перевезень, але й створює прецедент для майбутніх інновацій, спрямованих на покращення глобального досвіду подорожей.

ДОДАТОК А ПРОГРАМНИЙ КОД

A1 – Current_User.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Library{
    public class currentUser{
        public static string name { get; set; }
        public static string password { get; set; }
        public static bool isAdmin { get; set; }
    }
}
```

A2 – Flight.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Library{
    public class flight{
        public string name { get; set; }
        public int price { get; set; }
        public string date { get; set; }
        public int seats { get; set; }
    }
}
```



```
}
```

A3 – Flights_Data_Manager.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Library{
    public interface IFlightsDataManager{
        List<flight> loadFlights();
        flight getFlight(string name);
    }
    public class flightsDataManager : IFlightsDataManager{
        public string
flightsFilePath="E:\\all\\University\\y2t2\\SA\\tasks\\ct2\\dev\\App\\fli
ghts.txt";
        public List<flight> loadFlights(){
            List<flight> flights = new List<flight>();
            if (File.Exists(flightsFilePath)){
                string[] strings = File.ReadAllLines(flightsFilePath);
                foreach (string line in strings){
                    string[] parts = line.Split(',');
                    if (parts.Length == 4){
                        flight flight = new flight{
                            name = parts[0],
                            price = int.Parse(parts[1]),
                            date = parts[2],
                            seats = int.Parse(parts[3])
                        };
                        flights.Add(flight);
                    }
                }
            }
            return flights;
        }
        public flight getFlight(string name){
            List<flight> flights = loadFlights();
            foreach (flight flight in flights){
                if (flight.name == name){
                    return flight;
                }
            }
            return null;
        }
    }
}
```

```
}
```

A4 – Ticket.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Library{
    public class ticket{
        public string userName { get; set; }
        public string flightName { get; set; }
        public int price { get; set; }
        public string date { get; set; }
        public int seatRow { get; set; }
        public bool isMiddle { get; set; }
        public bool isWindow { get; set; }
        public bool isPrivate { get; set; }
        public bool isBaggage { get; set; }
        public bool isMeal { get; set; }
    }
}
```

A5 – Ticket_Data_Manager.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Library{
    public interface ITicketsDataManager{
        List<ticket> loadTickets();
        ticket GetTicket(string flightName);
        List<ticket> GetOwnTickets(string userName);
        void AddTicket(string userName, string flightName, int price,
string date, int seatRow, bool isMiddle, bool isWindow, bool isPrivate,
bool isBaggage, bool isMeal);
    }

    public class ticketsDataManager : ITicketsDataManager{
        public string ticketsFilePath =
"E:\\all\\University\\y2t2\\SA\\tasks\\ct2\\dev\\App\\tickets.txt";
        public List<ticket> loadTickets(){
            List<ticket> tickets = new List<ticket>();
        }
    }
}
```

```

        if (File.Exists(ticketsFilePath)){
            string[] strings = File.ReadAllLines(ticketsFilePath);
            foreach (string line in strings){
                string[] parts = line.Split(',');
                if (parts.Length == 10){
                    ticket ticket = new ticket{
                        userName = parts[0],
                        flightName = parts[1],
                        price = int.Parse(parts[2]),
                        date = parts[3],
                        seatRow = int.Parse(parts[4]),
                        isMiddle = bool.Parse(parts[5]),
                        isWindow = bool.Parse(parts[6]),
                        isPrivate = bool.Parse(parts[7]),
                        isBaggage = bool.Parse(parts[8]),
                        isMeal = bool.Parse(parts[9]),
                    };
                    tickets.Add(ticket);
                }
            }
        }
        return tickets;
    }

    public ticket GetTicket(string flightName){
        List<ticket> tickets = loadTickets();
        foreach (ticket ticket in tickets){
            if (ticket.flightName == flightName){
                return ticket;
            }
        }
        return null;
    }

    public List<ticket> GetOwnTickets(string userName){
        List<ticket> tickets = loadTickets();
        List<ticket> ownTickets = new List<ticket>();
        foreach (ticket ticket in tickets){
            if (ticket.userName == userName){
                ownTickets.Add(ticket);
            }
        }
        return ownTickets;
    }

    public void AddTicket(string userName, string flightName, int
price, string date, int seatRow, bool isMiddle, bool isWindow, bool
isPrivate, bool isBaggage, bool isMeal){
        File.AppendAllText(ticketsFilePath,
        $"{\n{userName},{flightName},{price},{date},{seatRow},{isMiddle.ToString()

```

```

        .ToLower()}, {isWindow.ToString().ToLower()}, {isPrivate.ToString().ToLower()
    }}, {isBaggage.ToString().ToLower()}, {isMeal.ToString().ToLower()}");
        Console.WriteLine("Ticket added successfully");
    }
}
}

```

A6 – User.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Library{
    public class user{
        public string name { get; set; }
        public string password { get; set; }
        public bool isAdmin { get; set; }
    }
}

```

A7 – User_Data_Manager.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Library{
    public interface IUserDataManager{
        List<user> loadUsers();
        void addUser(string name, string password, bool isAdmin);
        user getUser(string name);
        bool isAdmin(string name);
        bool validateCredentials(string name, string password);
    }

    public class userDataManager : IUserDataManager{
        public string usersFilePath =
"E:\\all\\University\\y2t2\\SA\\tasks\\ct2\\dev\\App\\users.txt";
        public List<user> loadUsers(){
            List<user> users = new List<user>();
            if (File.Exists(usersFilePath)){
                string[] lines = File.ReadAllLines(usersFilePath);
                foreach (string line in lines){

```

```

        string[] parts = line.Split(',');
        if (parts.Length == 3){
            user user = new user{
                name = parts[0],
                password = parts[1],
                isAdmin = bool.Parse(parts[2]),
            };
            users.Add(user);
        }
    }
}

return users;
}

public void addUser(string name, string password, bool isAdmin){
    File.AppendAllText(usersFilePath,
        $"\n{name},{password},{isAdmin.ToString().ToLower()}\n");
    Console.WriteLine("User added successfully");
}

public user getUser(string name){
    List<user> users = loadUsers();
    foreach (user user in users){
        if (user.name == name){
            return user;
        }
    }
    return null;
}

public bool isAdmin(string name){
    user user = getUser(name);
    if (user != null) { return user.isAdmin; }
    return false;
}

public bool validateCredentials(string name, string password){
    user user = getUser(name);
    if (user != null && user.password == password) { return true; }

    return false;
}
}
}
}

```

A8 – Flights.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;

```

```

using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Class_Library;

namespace App{
    public partial class flights : Form{
        public flight selectedFlight;
        public const int collapsedHeight = 190;
        public const int expandedHeight = 560;

        public int totalPrice;
        public int seatTypePrice = 0;
        public int mealPrice = 0;
        public int luggagePrice = 0;
        public int privatePrice = 0;
        public flights(){
            InitializeComponent();
            infoPanel.Visible = false;
            optionsPanel.Visible = false;
            this.Height = collapsedHeight;

            flightsDataManager Manager = new flightsDataManager();
            List<flight> flights = Manager.loadFlights();
            renderFlightNames(flights);
        }
        public void updateTotal(){
            totalPrice = selectedFlight.price + seatTypePrice + mealPrice
+ luggagePrice + privatePrice;
            totalPriceLabel.Text = $"In total: ${totalPrice}";
        }
        public void renderFlightNames(List<flight> flights){
            flightsList.Items.Clear();
            foreach (flight flight in flights){
                flightsList.Items.Add(flight.name);
            }
        }
        private void menuButton_Click(object sender, EventArgs e){
            this.Hide();
            menu menu = new menu();
            menu.Show();
        }
        private void flightsList_SelectedValueChanged(object sender,
EventArgs e){
            flightsDataManager Manager = new flightsDataManager();
            selectedFlight =
Manager.getFlight(flightsList.SelectedItem.ToString());
            infoPanel.Visible = true;

```

```

optionsPanel.Visible = true;
this.Height = expandedHeight;

selectedLabel.Text = selectedFlight.name;
priceLabel.Text = $"Ticket price: ${selectedFlight.price}";
dateLabel.Text = $"Flight date:
{selectedFlight.date.Replace("-", "/)}";
seatsLabel.Text = $"Available seats: {selectedFlight.seats}";

int seatRows = (int)Math.Floor((double)selectedFlight.seats /
10);

rowSelect.Maximum = seatRows;

totalPrice = selectedFlight.price;
updateTotal();
}
private void middleRadio_CheckedChanged(object sender, EventArgs
e){
    if (middleRadio.Checked){
        randomRadio.Checked = false;
        windowRadio.Checked = false;
        seatTypePrice = 10;
        updateTotal();
    }
}
private void randomRadio_CheckedChanged(object sender, EventArgs
e){
    if (randomRadio.Checked){
        windowRadio.Checked = false;
        middleRadio.Checked = false;
        seatTypePrice = 0;
        updateTotal();
    }
}
private void windowRadio_CheckedChanged(object sender, EventArgs
e){
    if (windowRadio.Checked){
        randomRadio.Checked = false;
        middleRadio.Checked = false;
        seatTypePrice = 30;
        updateTotal();
    }
}
private void mealBox_CheckedChanged(object sender, EventArgs e){
    mealPrice = mealBox.Checked ? 15 : 0;
    updateTotal();
}

```

```

        private void luggageBox_CheckedChanged(object sender, EventArgs
e){
            luggagePrice = luggageBox.Checked ? 30 : 0;
            updateTotal();
        }
        private void privateBox_CheckedChanged(object sender, EventArgs
e){
            privatePrice = privateBox.Checked ? 77 : 0;
            updateTotal();
        }
        private void bookButton_Click(object sender, EventArgs e){
            ticketsDataManager Manager = new ticketsDataManager();
            Manager.AddTicket(currentUser.name, selectedFlight.name,
totalPrice, selectedFlight.date, (int)rowSelect.Value,
middleRadio.Checked, windowRadio.Checked, privateBox.Checked,
luggageBox.Checked, mealBox.Checked);
            MessageBox.Show("Ticket booked successfully");
            this.Hide();
            menu menu = new menu();
            menu.Show();
        }
    }
}

```

A9 – Flights.txt

```

Zaporizhzhia Airport - Choctaw Airport,50,11-05-2024,77
Zaporizhzhia Airport - Sandpoint Airport,40,12-05-2024,77
Zaporizhzhia Airport - Kootenai Airport,30,13-05-2024,77
Kyiv Airport - Chernihiv Airport,60,14-05-2024,77
Poltava Airport - Kryvyi Rih Airport,77,15-05-2024,77
Zhytomyr Airport - Chernivtsi Airport,70,16-05-2024,77
Lviv Airport - Kharkiv Airport,67,17-05-2024,77

```

A10 – Login.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Class_Library;

```



```

namespace App{
    public partial class login : Form{
        public login(){
            InitializeComponent();
        }
        public void loginButton_Click(object sender, EventArgs e){
            string name = nameInput.Text;
            string password = passwordInput.Text;

            if (name == "" || password == ""){
                MessageBox.Show("Please enter username and password");
                nameInput.Text = "";
                passwordInput.Text = "";
                return;
            }

            userDataManager Manager = new userDataManager();
            if (Manager.validateCredentials(name, password)){
                currentUser.name = name;
                currentUser.password = password;
                currentUser.isAdmin = Manager.isAdmin(name);

                this.Hide();
                menu menu = new menu();
                menu.Show();
            }
            else{
                MessageBox.Show("Invalid username or password");
                nameInput.Text = "";
                passwordInput.Text = "";
            }
        }
        private void registerButton_Click(object sender, EventArgs e){
            this.Hide();
            signup signup = new signup();
            signup.Show();
        }
    }
}

```

A11 – Menu.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;

```

```

using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Class_Library;

namespace App{
    public partial class menu : Form{
        public menu(){
            InitializeComponent();

            if (currentUser.isAdmin){
                ticketsButton.Text = "All Tickets";
            }
        }
        private void signoutButton_Click(object sender, EventArgs e){
            this.Hide();
            login login = new login();
            login.Show();
        }
        private void flightsButton_Click(object sender, EventArgs e){
            this.Hide();
            flights flights = new flights();
            flights.Show();
        }
        private void ticketsButton_Click(object sender, EventArgs e){
            this.Hide();
            tickets tickets = new tickets();
            tickets.Show();
        }
    }
}

```

A12 – Sign_Up.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Class_Library;

namespace App{
    public partial class signup : Form{

```

```

public signup(){
    InitializeComponent();
}
private void loginButton_Click(object sender, EventArgs e){
    this.Hide();
    login login = new login();
    login.Show();
}
private void signupButton_Click(object sender, EventArgs e){
    string name = nameInput.Text;
    string password = passwordInput.Text;
    bool isAdmin = adminCheck.Checked;

    userDataManager Manager = new userDataManager();
    user user = Manager.getUser(name);
    if (user != null){
        MessageBox.Show("Username already exists");
        nameInput.Text = "";
        passwordInput.Text = "";
        return;
    }

    Manager.addUser(name, password, isAdmin);
    this.Hide();
    login login = new login();
    login.Show();
}
}
}

```

A13 – Tickets.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Class_Library;

namespace App{
    public partial class tickets : Form{
        public tickets(){
            InitializeComponent();
            if (!currentUser.isAdmin){

```

```

        ticketsGrid.Columns[0].Visible = false;
    }
    loadTickets();
}
public void loadTickets(){
    ticketsDataManager Manager = new ticketsDataManager();
    ticketsGrid.Rows.Clear();
    List<ticket> tickets;
    if (currentUser.isAdmin){
        tickets = Manager.loadTickets();
    }
    else{
        tickets = Manager.GetOwnTickets(currentUser.name);
    }
    for (int i = 0; i < tickets.Count; i++){
        ticket ticket = tickets[i];
        string seatType = ticket.isMiddle ? "Middle" :
(ticket.isWindow ? "Window" : "Normal");
        string extraServices = "";
        extraServices += ticket.isBaggage ? "Baggage " : "";
        extraServices += ticket.isMeal ? "Meal " : "";
        extraServices += ticket.isPrivate ? "Private " : "";
        ticketsGrid.Rows.Add(ticket.userName, ticket.flightName,
ticket.price, ticket.date, ticket.seatRow, seatType, extraServices);
    }
}
private void backButton_Click(object sender, EventArgs e){
    this.Hide();
    menu menu = new menu();
    menu.Show();
}
}
}
}

```

A14 – Tickets.txt

```

user,Zaporizhzhia Airport - Choctaw Airport,90,11-05-
2024,5,true,false,false,true,false
new,Zaporizhzhia Airport - Kootenai Airport,152,13-05-
2024,1,false,true,true,false,true
admin,Zaporizhzhia Airport - Kootenai Airport,70,13-05-
2024,7,true,false,false,true,false
neww,Zaporizhzhia Airport - Kootenai Airport,162,13-05-
2024,4,true,false,true,true,true
new,Zaporizhzhia Airport - Kootenai Airport,182,13-05-
2024,7,false,true,true,true,true

```

```
admin,Zaporizhzhia Airport - Sandpoint Airport,100,12-05-  
2024,7,false,true,false,true,false  
user,Zaporizhzhia Airport - Sandpoint Airport,80,12-05-  
2024,1,true,false,false,true,false  
user,Zaporizhzhia Airport - Choctaw Airport,182,11-05-  
2024,7,true,false,true,true,true  
user,Zaporizhzhia Airport - Kootenai Airport,117,13-05-  
2024,7,true,false,true,false,false
```

A15 – Users.txt

```
admin,admin,true  
user,user,false  
new,new,false  
neww,neww,true
```