Sprints

Simple Flight Reservation Desktop App Using Tkinter and SQLite

1. General Description of the Task

This project involves developing a **basic flight reservation system** as a desktop application using **Tkinter for the GUI** and **SQLite for database management**. Users should be able to **book, view, update, and delete reservations** using a simple and user-friendly interface.

Objectives:

1. Build a **GUI** using Tkinter to manage flight reservations.
2. Implement **CRUD (Create, Read, Update, Delete)** operations with **SQLite**.
3. Structure the app into **multiple pages (views)** for a better user experience.
4. Upload the final project to **GitHub** with documentation.

Tools to be Used:

- **Python**
- **Tkinter** (for GUI)
- **SQLite** (for database)
- **GitHub** (for version control)

---

2. Requirements

2.1 GUI Development with Tkinter

- **Details/Steps:**
    1. Create a Tkinter window with a title and proper dimensions.
    2. Implement navigation between different pages.
    3. Include buttons for booking, updating, and deleting reservations.
- **Deliverable:** A functional Tkinter window with navigation and UI elements.

2.2 Database Setup (SQLite)

- **Details/Steps:**
    1. Create an SQLite database named `flights.db`.
    2. Design a table for storing reservation details:
        - `id` (Primary Key, Auto Increment)

        - `name` (Passenger Name)
        - `flight_number`
        - `departure`
        - `destination`
        - `date`
        - `seat_number`
    3. Connect the database with the application.
- **Deliverable:** A properly structured SQLite database with tables.

2.3 Implement CRUD Operations

- **Details/Steps:**
    1. **Create**: Allow users to book a flight and store the data in the database.
    2. **Read**: Display a list of all reservations.
    3. **Update**: Allow users to modify existing reservations.
    4. **Delete**: Provide an option to remove a reservation.
- **Deliverable:** A fully functional database integration supporting CRUD operations.

2.4 Upload the Project to GitHub

- **Details/Steps:**
    1. Initialize a Git repository in the project folder (`git init`).
    2. Add a `README.md` file explaining how to run the app.
    3. Push the project to GitHub.
- **Deliverable:** A public GitHub repository containing the complete project.

2.5. Export the App into exe

- **Details/Steps:**
    - **Preparation**: Ensure your Python application is complete and functioning correctly, with all dependencies clearly listed.
    - **Tool Setup**: Install PyInstaller using pip (`pip install pyinstaller`) and set up your configuration according to your application's needs.
    - **Compilation**: Use PyInstaller to build the executable by navigating to your project directory in the command line and running `pyinstaller --onefile your_script_name.py`.
    - **Testing and Distribution**: Test the `.exe` file on different systems to ensure it operates as expected, then distribute the executable along with a README file for instructions.
- **Delievable:** A fully running Windows executable file.

Sprints

3. Page Structure & Navigation

3.1 Home Page (Main Window)

- Display options: "Book Flight", "View Reservations".
- Buttons for navigation.

3.2 Flight Booking Page

- Input fields:
  - Name
  - Flight Number
  - Departure
  - Destination
  - Date
  - Seat Number
- Submit button to store the reservation.

3.3 Reservation List Page

- Show all reservations in a table.
- Edit and delete buttons for each entry.

3.4 Edit Reservation Page

- Pre-filled form with existing reservation details.
- Update button to modify the data.

---

4. File Structure

```
/flight_reservation_app
  ├── main.py              # Main application file
  ├── database.py          # SQLite database connection and setup
  ├── home.py              # Home page UI
  ├── booking.py           # Flight booking form
  ├── reservations.py      # View all reservations
  ├── edit_reservation.py  # Update/Delete functionality
  ├── flights.db           # SQLite database file
  ├── requirements.txt     # Required Python libraries
  ├── README.md            # Project documentation
├── .gitignore              # Git ignore file
```

---

5. Sample UI:

https://flighty-reserve-mate.lovable.app/

Sprints

6. Final Deliverables Checklist

1. Source Code Files

✅ `main.py` – Main application entry point
✅ `database.py` – SQLite database connection and setup
✅ `home.py` – Home page UI
✅ `booking.py` – Flight booking form UI
✅ `reservations.py` – Reservation list UI
✅ `edit_reservation.py` – Edit and delete reservation functionality
✅ `requirements.txt` – List of required Python libraries

2. Database File

✅ `flights.db` – SQLite database storing reservation records

3. Executable File

✅ `dist/main.exe` – Standalone executable file for Windows

4. Documentation

✅ `README.md` – Instructions on how to install and use the application
✅ `user_guide.pdf` (Optional) – Detailed guide for end users

5. Export

✅ `icon.ico` – Custom application icon (if applied)
✅ Installer `.exe` – If created for easier installation

6. Final Packaging for Submission

✅ **Zip File** containing:

- All source code files
- SQLite database
- `.exe` file
- Documentation
- Optional installer