Flight Reservation Desktop Application

A simple and user-friendly flight reservation system built with Python, Tkinter, and SQLite. This desktop application allows users to book, view, update, and delete flight reservations through an intuitive graphical interface.

Features

- **Book Flights**: Create new flight reservations with passenger details
- **View Reservations**: Display all bookings in a sortable table format
- **Edit Reservations**: Update existing booking information
- **Delete Reservations**: Remove unwanted reservations with confirmation
- **Data Persistence**: SQLite database for reliable data storage
- **User-Friendly Interface**: Clean and intuitive GUI using Tkinter

Screenshots

Home Page

The main interface with navigation options to book flights or view existing reservations.

Booking Page

Form to enter flight details including passenger name, flight number, departure/destination cities, date, and seat number.

Reservations List

Table view showing all booked flights with options to edit or delete individual reservations.

Installation & Setup

Prerequisites

- Python 3.7 or higher
- No additional packages required (uses Python standard library)

Running from Source Code

1. **Clone or Download** the project files to your computer
2. **Navigate** to the project directory
3. **Run** the application:

```bash
python main.py
```

File Structure

```
flight_reservation_app/
├── main.py              # Main application entry point
├── database.py          # SQLite database operations
├── home.py              # Home page UI
├── booking.py           # Flight booking form
├── reservations.py      # Reservation list view
├── edit_reservation.py  # Edit/delete functionality
├── requirements.txt     # Dependencies list
├── README.md            # This documentation
└── flights.db           # SQLite database (created automatically)
```

Usage Guide

Booking a New Flight

1. Click "📝 **Book Flight**" from the home page
2. Fill in all required fields:
   - **Name**: Passenger's full name
   - **Flight Number**: Airline flight identifier (e.g., "AA123")
   - **Departure**: Origin city or airport
   - **Destination**: Destination city or airport
   - **Date**: Flight date in YYYY-MM-DD format (e.g., "2024-12-25")
   - **Seat Number**: Assigned seat (e.g., "12A")
3. Click "✈️ **Book Flight**" to save the reservation
4. Success confirmation will show the reservation ID

Viewing Reservations

1. Click "📋 **View Reservations**" from the home page
2. All bookings are displayed in a table with sortable columns
3. Use "🔄 **Refresh**" to reload the latest data
4. Double-click any row to edit that reservation

Editing a Reservation

1. From the reservations list, select a booking and click "✏️ **Edit Selected**"
   - Or double-click directly on the reservation row
2. Modify any field values in the form
3. Click "💾 **Update Reservation**" to save changes
4. Use "🔄 **Reset Form**" to restore original values
5. Click "❌ **Cancel**" to return without saving

Deleting a Reservation

1. **From Reservations List**: Select a booking and click "🗑️ **Delete Selected**"
2. **From Edit Page**: Click "🗑️ **Delete Reservation**" while editing
3. Confirm the deletion in the popup dialog
4. The reservation will be permanently removed

Database Schema

The application uses SQLite with the following table structure:

```sql
CREATE TABLE reservations (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    flight_number TEXT NOT NULL,
    departure TEXT NOT NULL,
    destination TEXT NOT NULL,
    date TEXT NOT NULL,
    seat_number TEXT NOT NULL
);
```

Building Executable

To create a standalone Windows executable:

1. **Install PyInstaller**:

   ```bash
   pip install pyinstaller
   ```

2. **Build the executable**:

```bash
pyinstaller --onefile --windowed --name "Flight Reservation System" main.py
```

3. **Find the executable** in the `dist/` folder

Executable Options

- `--onefile`: Creates a single executable file
- `--windowed`: Hides the console window (GUI app)
- `--name`: Sets the executable name
- `--icon=icon.ico`: Adds a custom icon (if available)

Technical Details

Architecture

- **Frontend**: Tkinter (Python's built-in GUI framework)
- **Backend**: SQLite (embedded database)
- **Language**: Python 3.7+
- **Design Pattern**: Modular page-based architecture

Key Components

- **Database Class**: Handles all SQLite operations (CRUD)
- **Page Classes**: Separate UI components for each screen
- **Main App Class**: Coordinates navigation and manages application state

Data Validation

- **Required Fields**: All form fields must be completed
- **Date Format**: Validates YYYY-MM-DD format
- **Error Handling**: User-friendly error messages
- **Confirmation Dialogs**: Prevents accidental data loss

Troubleshooting

Common Issues

"Database locked" error:

- Close any other instances of the application
- Ensure the database file isn't open in another program

Date validation error:

- Use the exact format: YYYY-MM-DD
- Example: 2024-12-25 (not 12/25/2024)

Application won't start:

- Ensure Python 3.7+ is installed
- Check that all files are in the same directory
- Run from command line to see error messages

Missing reservations:

- Click the "🔄 Refresh" button
- Check if the `flights.db` file exists in the same directory

Getting Help

If you encounter issues:

1. Check that all required files are present
2. Ensure Python is properly installed
3. Try running from the command line to see detailed error messages

Version History

v1.0.0 - Initial Release

- Complete CRUD functionality
- Multi-page navigation
- SQLite database integration
- Form validation
- User-friendly interface

License

This project is provided as-is for educational and personal use.

Contributing

This is a self-contained educational project. Feel free to modify and enhance the code for your own learning purposes.

Flight Reservation System v1.0

Built with Python, Tkinter & SQLite