Flight Management System

Abstract

This project encompasses a Flight management system that was programmed in Python programming language using the Streamlit framework. This system focuses on the management of the passengers, the flights bookings and the generation of boarding passes. Passengers and other relevant flight information are kept in an SQLite database and custom images of boarding passes are created by the PIL library. The most essential include such as the processes of registering new passengers, the process of displaying active flights, the process of ticketing, and the process of obtaining particular booking details for issuing a specific boarding pass. This system makes the process of booking and boarding a flight easy and hence It is beneficial for effective airline management.

Acknowledgement

We would like to express my special thanks to "Dr. Deepshubhra Guha Roy", for his able guidance and support in completing our project.

Soham Banerjee
(12022002016053)
B.Tech
V-Semester

_____

Soumyo Mallick
(12022002016054)
B.Tech
V-Semester

_____

Table of Contents

1. Introduction

The Flight Management System is a project designed to manage and streamline the booking of flights, passenger details, and the generation of boarding passes. This project leverages Streamlit for the front-end application interface and SQLite for managing data. The main objectives of the project include:

• Efficiently managing passenger information.
• Displaying available flights and enabling booking.
• Generating downloadable boarding passes.

2. System Architecture

The system architecture of the Flight Management System is composed of the following main components:

• Streamlit: Provides the front-end for user interaction.
• SQLite: Manages data persistence and storage for passengers, flights, and booking information.
• PIL (Python Imaging Library): Used to generate the boarding pass image dynamically based on booking details.

The system architecture for this Flight Management System application can be represented with the following key components and layers:

1. **User Interface Layer (Frontend)**

Streamlit Framework: The graphical interface of the system through which different users (e.g. participating airline staff and followers) engage with the system. It includes forms for passenger information, flight orders, options that add passengers or book a flight and generate boarding passes.

2. **Application Layer (Backend)**

Business Logic and Functions: Key operations within Handle Activities of the application such submit passengers, search applicable flights, book a flight and issue a boarding pass. Such functions are database oriented and work with data in accordance with user requests.

Add Passenger Function: Makes a cross reference whether it is necessary all fields are filled and uploads all details of passengers into the relevant container database.

Get Available Flights Function: Employs as CRUD the database so as to look for flights that are available such that the users seeking for flights are provided with a range of flights to choose from before booking a flight.

Book Ticket Function: Post a flight, this functionality is performed by attaching the ID of a passenger who intends to travel with the selected flight.

Generate Boarding Pass Function: How to book flights. Offers and collects details related to the booking and consequently formulates a fresh boarding pass which is then created into an image file so that it can then be downloaded.

3. **Database Layer Considered:** A unique layer of the system, it is reserved for SQLite Database: It contains the most important data of the system, which are the fundamental components of the system. The fundamental components of the system are the number of its data such as: Basic data for a passenger, a set of one or more services for the flight operated, a set of one or more services for the booking done. In the case of more complex systems, to achieve the desired result, these two data relationships must be achieved: Basic data for a passenger.

Tables are possible to define within one or more database objects:

Passenger Table: Provides a list of parameters related to the passengers. People may present their identity as (ID, name, age, and date of birth, country residential address and as an example, a telephone number).

Flights Table: Based on basic data identifying a flight (flight ID), Departure and Arrival Information and Times for Departures are integrated into the system.

Bookings Table: Cross-linking individual passengers with particular flights, must also include the information of when the passenger booked the flight and details required to print a ticket.

4. **File Storage Layer:** Specifically, the focal character and feature of the system that is designed systems generated for Bonuses Theater boarding passes is character boarding passes: Created using predefined templates, this allowed the creation of standard designs with specific formatting allowing users to view and download templates. Images of boarding passes or tickets can be created using the Pillow library. These images are created when a passenger reserves the appropriate use of liberty principles, horizontal space, and key image design.

5. **Reusable Libraries and Libraries:** PIL: Libraries that provide predefined parameters or relevant commands for a repeating graphic or text element. These are time savers whenever a design pass is created for a plane booking detail screen. Looking Pass A python program or image within a pass generated image using sequential, specific welcome pages or billboards detailing information such as boarding Pass, departure, destination country along with a picture of a plane with scheduling details embedded within it.

Datetime Module: Provides the paper with the time of a newspaper to be specific, every detail is consistent with the time when the booking and the paper used to be ordered occurred. The current time is always the time within the booking event and the order of ticket was made on.

Diagram: System Architecture at a Glance

The diagram below describes the three structures and illustrates how the data flows between them. Imagine how the designer imagines the scope of the three-dimensional animations.

3. Database Design

The database design is structured around the entities required for a flight booking system. The core tables include:

• Passenger Table: Stores information such as Passenger ID, name, age, sex, address, contact, and email.

- **Fields**: pid (Primary Key), name, age, sex, address, contact, and email.
- **Purpose**: Stores passenger details, where each passenger has a unique ID (pid). The other fields capture demographic and contact information.
- **Constraints**: Each field must be filled out for data integrity.

• Flights Table: Contains flight details like flight ID, departure and arrival times, and destinations.

- **Fields**: flight_id (Primary Key), departure_place, departure_time, destination, and arrival_time.
- **Purpose**: Contains information on available flights, including unique identifiers (flight_id), locations, and times for departure and arrival.
- **Constraints**: Ensures that each flight entry is unique and that essential data like times and locations are filled out.

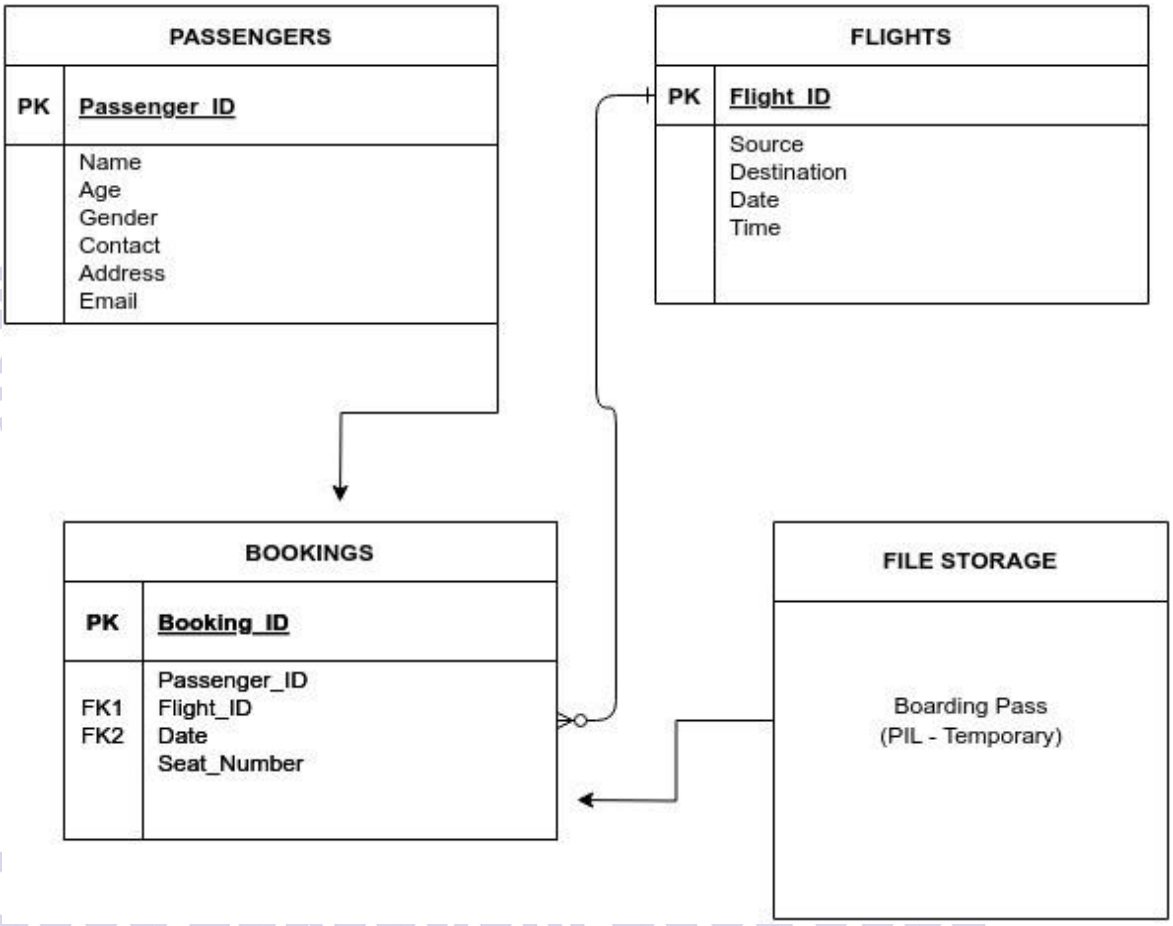
• Bookings Table: Records bookings linked to passengers and flights.

- **Fields**: pid (Foreign Key to Passenger table), flight_id (Foreign Key to Flights table), and booking_date.
- **Purpose**: Links passengers to flights they have booked, forming a many-to-one relationship (many bookings per passenger).
- **Constraints**: Uses foreign keys to enforce relationships with the Passenger and Flights tables, ensuring that bookings only refer to existing passengers and flights.

*Relationships

- **One-to-Many Relationship**:
  - Each **Passenger** can have multiple entries in the **Bookings** table (i.e., multiple flights can be booked by the same passenger).
  - Each **Flight** can also be linked to multiple passengers through the **Bookings** table.

- **Normalization**: The database follows normalization principles, reducing redundancy and optimizing data retrieval for booking and passenger queries.



4. Application Features

A. Passenger Management

- **Add New Passenger**: Allows users to input and save passenger details, including name, age, gender, address, contact, and email.
- **Data Validation**: Ensures all required fields are filled before saving, improving data accuracy and completeness.

B. Flight Availability

- **View Available Flights**: Displays available flights with details such as flight ID, departure place and time, destination, and arrival time.
- **Selection for Booking**: Users can select from the available flights, making the booking process straightforward and user-friendly.

C. Flight Booking

- **Book a Flight**: Associates a passenger with a selected flight and saves the booking information with the current date and time.
- **Booking Confirmation**: Confirms successful booking with a message, ensuring users know their booking was saved.

D. Boarding Pass Generation

- **Retrieve Booking Details**: Fetches a passenger's booking details to populate the boarding pass.
- **Generate and Download Boarding Pass**: Creates a customized image with boarding pass details (name, departure, destination, flight info, date, and time), which can be viewed and downloaded by the user.

E. User-Friendly Interface

- **Streamlit Frontend**: Provides a straightforward, interactive web-based interface, making the application accessible and easy to use without needing extensive technical knowledge.
- **Error Handling**: Displays messages for missing data or unselected options, guiding users through correct usage of the application.

These features combine to create a comprehensive system that efficiently manages flight booking processes from passenger data entry to boarding pass generation that results in an ultimate customer satisfiable experience.

5. Code Implementation

Code Implementation for the Flight Management System-Integration of several python libraries along with a defined workflow :-

- **Interacting with the Database (SQLite)**

Database Functions: Core functions involving passenger addition, retrieval of flights, booking of tickets, and fetching booking details. These are done through SQL queries, along with the use of sqlite3. Modifications are saved to ensure data consistency.

- **Streamlit Interface**

User Interface: Streamlit framework is the frontend application with a user-friendly web interface. Functions such as st.text_input, st.number_input, st.selectbox, and buttons input passenger information, available flight details, and booking the ticket.

Data Validation, Error Messages: The interface has checks for data entered to be complete; after that, it displays error or success messages to assist a user.

- **Flight Booking Algorithm**

Add Passenger: Collects all details as required and stores them in a database if all the detail is complete.

Flight Selection and Booking: The system fetches available flights, displays them in a drop-down list, and stores bookings pertaining to passenger ID for easy record-keeping and retrieval.

- **Boarding Pass Generation (PIL)**

Image Creation: Utilizing the PIL library, the application generates a customized boarding pass with all flight and passenger information. This boarding pass is drawn on a blank image with specified font, color, and layout and then displayed and saved for download.

- **Interactivity and Instant Feedback in Real-Time**

Success/Error Notifications: Streamlit flashes feedback to the user depending on the success or failure of executing the functions.

Download: The application enables the downloading of the boarding pass by the user.

This is achievable since the modular code design and use of Python libraries leaves only a minimal room for error in managing the booking process and handling passenger information on the system, which results in being friendlier to the user and data consistency throughout the entire process.

6. Conclusion

It is a Streamlit-based application that, through a SQLite database, manages the data of the airline. Functions include the following:

Database Interaction : The application uses a SQLite database referred to as airline_50.db for storing and retrieval of data.

Passenger Management: The system has an add passenger information function. The icon, when clicked, will prompt the user to fill all the fields to insert.

Flight Recovery: A process fetches the available flights, likely showing choices from which one would choose an appropriate one.

The code provides an efficient interface for airline data entry and retrieval that are mainly passenger- and flight-oriented, with Streamlit built on database management behind a simple, interactive web user interface.