Flight Booking System Report

1 Overview

The Flight Booking System is designed to offer a streamlined experience for users looking to book flights. This system allows users to search for flights based on departure and arrival airports, enter their details, select meal options, and choose additional services. The system is built with a focus on Object-Oriented Programming (OOP) principles and adheres to SOLID design principles to ensure it is modular, maintainable, and flexible.

2 Design and Architecture

The architecture of the Flight Booking System is organized into distinct modules, each handling a specific aspect of the booking process:

- **Airport Module**: Manages airport information, such as codes and names.
- **Airline Module**: Contains details about airlines and their flights.
- **Flight Module**: Represents the details of individual flights, including timings, prices, and seat availability.
- **Customer Module**: Stores customer information including names, contact details, and passport numbers.
- **Booking Module**: Handles the entire booking process, from calculating total costs to storing booking details.
- **Services Module**: Manages additional services, such as meal options and other extras.
- **UI Module**: Implements the graphical user interface (GUI) using Tkinter, providing a user-friendly experience for interacting with the system.

3 Application of OOP Concepts

The Flight Booking System utilizes several key OOP concepts to ensure a well-organized and maintainable codebase:

- **Encapsulation**: Each class is designed to handle specific data and functionality. For example, the `Airport` class manages airport details, while the `BookingService` class deals with booking logistics. This separation of concerns hides internal details from other parts of the system, making each component more manageable.

- **Inheritance**: This principle is applied to allow new features to extend existing classes without modifying their core functionality. For instance, if there were a base `Service` class, new service types like `MealOption` and `AdditionalService` could inherit from it, facilitating the addition of new services.
- **Polymorphism**: Different service types can be handled through a common interface or base class. This allows the system to process various services (meals, extra baggage) uniformly, without needing to differentiate between them in the booking logic.
- **Abstraction**: The system uses abstraction to provide clear interfaces for interacting with different components. For example, the `FlightSearch` class exposes a method for searching flights but hides the details of how flights are retrieved or managed.

4 Application of SOLID Principles

The system adheres to SOLID principles to ensure a robust and adaptable design:

- **Single Responsibility Principle (SRP)**: Each class in the system is focused on a single responsibility. For instance, the `Customer` class is solely responsible for handling customer details, while the `BookingService` class manages the booking process. This clear separation helps in maintaining and evolving the system.
- **Open/Closed Principle (OCP)**: The system is designed to be open for extension but closed for modification. New functionalities, such as additional services or airlines, can be added without altering existing classes. This design allows the system to evolve without compromising existing functionality.
- **Liskov Substitution Principle (LSP)**: The system ensures that subclasses can replace their parent classes without affecting the system's behavior. For example, different types of services (like meals and extra baggage) can be substituted in the booking process without disrupting the overall functionality.
- **Interface Segregation Principle (ISP)**: The system employs specific interfaces to avoid forcing classes to depend on methods they do not use. For instance, the `FlightSearch` and `BookingService` classes have focused interfaces that cater to their specific needs, preventing unnecessary dependencies.
- **Dependency Inversion Principle (DIP)**: High-level modules depend on abstractions rather than concrete implementations. For example, the `BookingService` class relies on abstract representations of flights and services, promoting flexibility and reducing tight coupling.

5 GUI Implementation

The graphical user interface (GUI) is developed using Tkinter to provide an interactive and user-friendly experience:

5.1 Key Features of the GUI

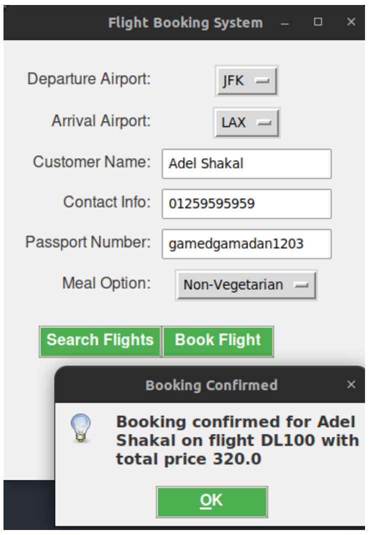
- **Flight Search**: Users can select departure and arrival airports from dropdown menus and search for available flights.
- **Customer Information**: Input fields are available for users to enter their details.
- **Meal and Additional Services**: Users can select meal options and add other services.
- **Buttons**: Includes buttons for searching flights and booking.

5.2 Design Choices

- **Modern and Colorful**: The GUI features a modern look with custom styles applied via `ttk.Style`. This includes thoughtful use of colors, fonts, and padding to enhance visual appeal and usability.
- **Responsive Design**: The layout is designed to adapt to different window sizes, ensuring a consistent user experience across various screen resolutions.

5.3 Screenshots and Code

- **Screenshots**: The following screenshot shows the main interface of the GUI.



Flight Booking System

Departure Airport: JFK
Arrival Airport: LAX
Customer Name: Adel Shakal
Contact Info: 01259595959
Passport Number: gamedgamadan1203
Meal Option: Non-Vegetarian

Booking Confirmed

Booking confirmed for Adel Shakal on flight DL100 with total price 320.0

Figure 1: Main Interface of the Flight Booking System GUI

- **Code**: The GUI code is contained in `ui/main_ui.py`, detailing the creation of widgets, styling, and event handling to deliver a polished user interface.

6 Conclusion

The Flight Booking System is a well-structured application that effectively utilizes OOP and SOLID principles to provide a modular and maintainable solution for flight booking. The design ensures clear responsibilities for each class and allows the system to be easily extended and modified. The GUI offers a modern, user-friendly experience, making the booking process intuitive and engaging.