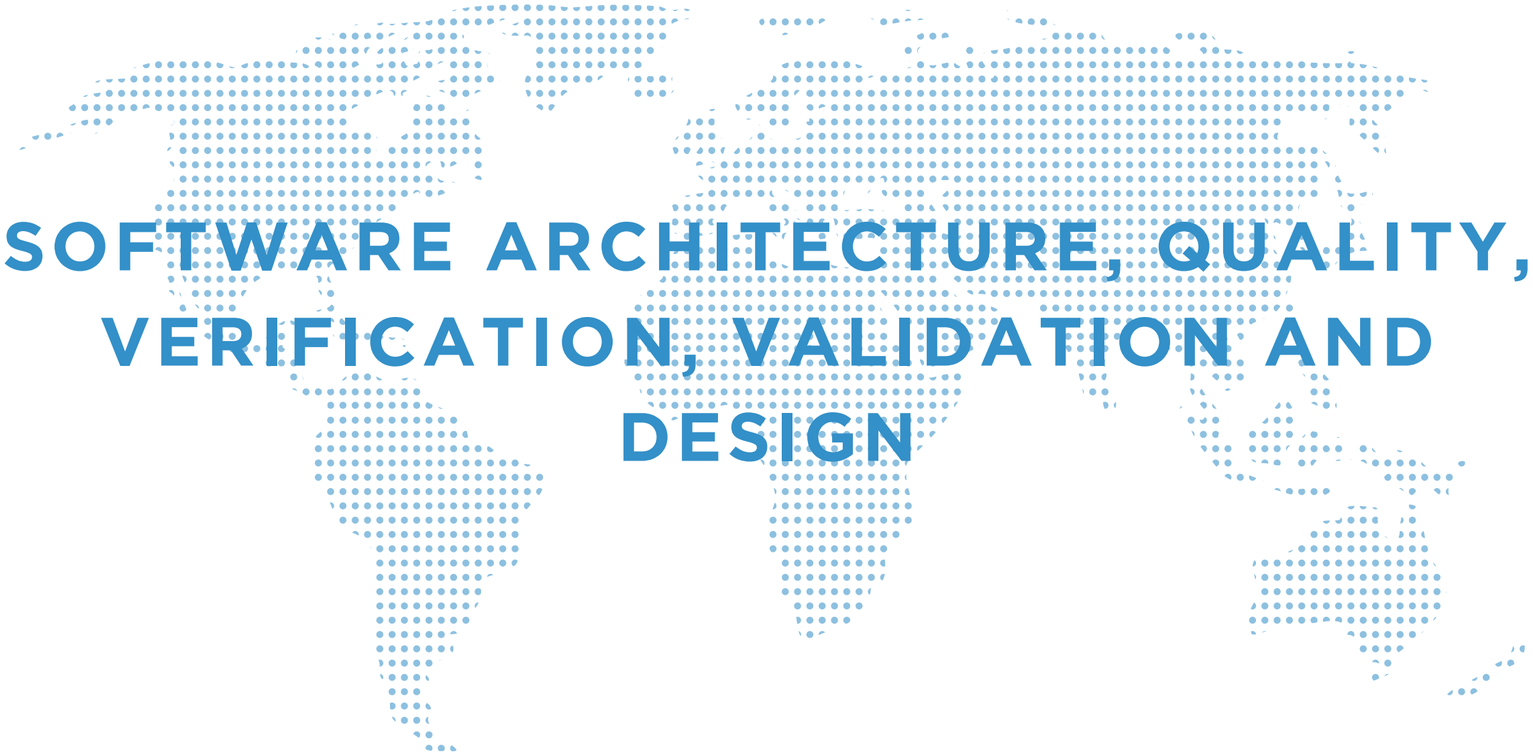
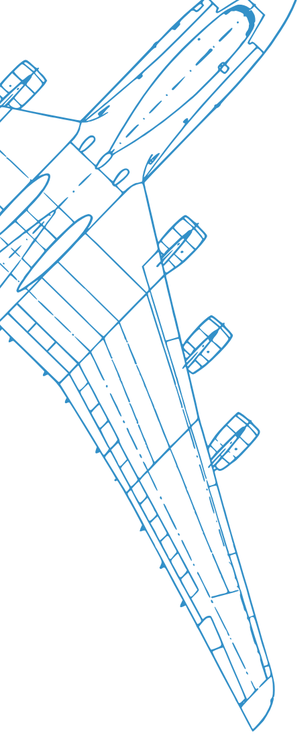
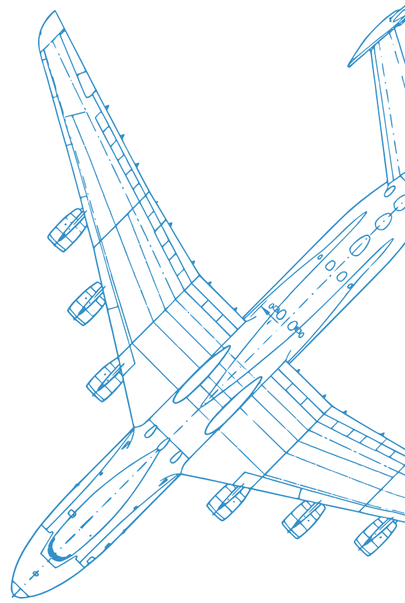
CONVENIENT FLIGHT BOOKING SYSTEM



SOFTWARE ARCHITECTURE, QUALITY, VERIFICATION, VALIDATION AND DESIGN



We present our innovative project aimed at creating a user-friendly flight booking system. Let's delve into the problems we address, how we propose to solve them, and the design patterns we employ to achieve our goals.

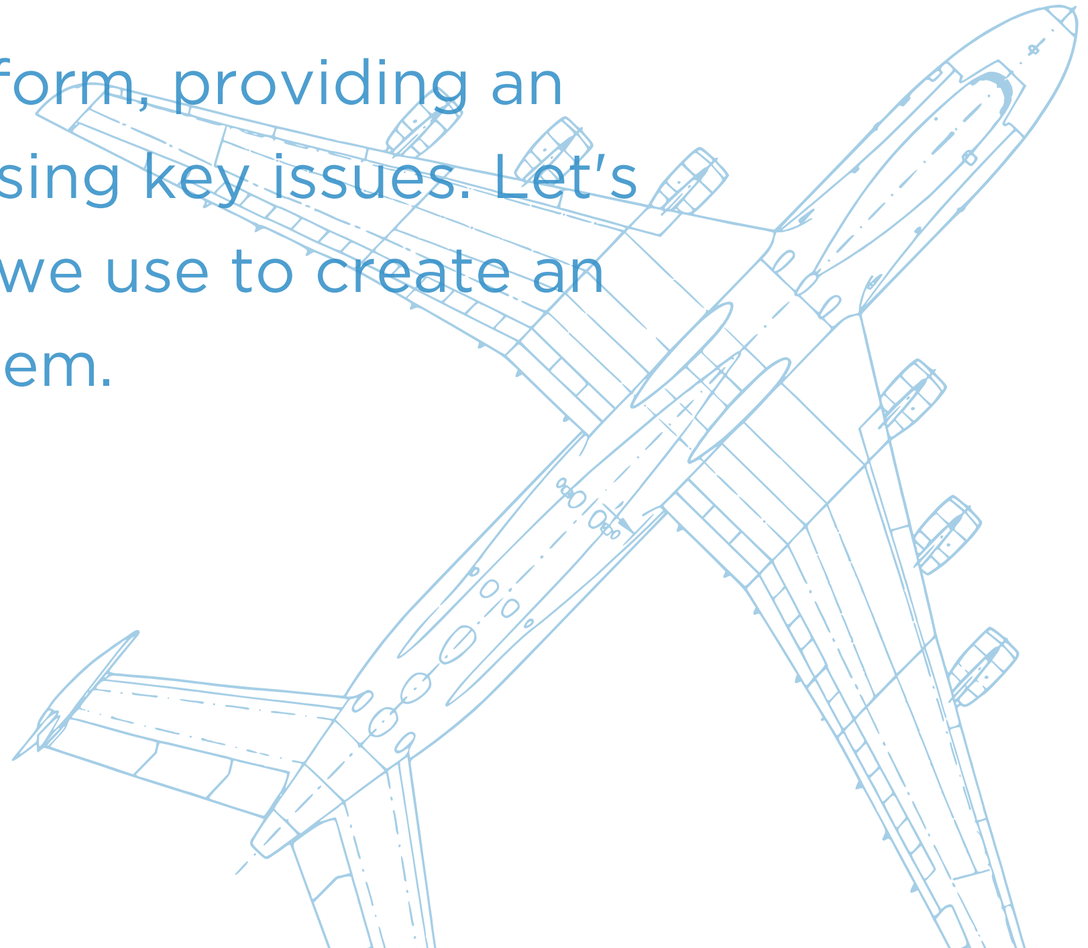


PROBLEM STATEMENT

The process of purchasing flight tickets often involves difficulties, demanding significant time and effort from passengers. Lack of centralized information, long queues, and limited seat choices create inconveniences for travelers. Our goal is to simplify and optimize this process.

PROPOSED SOLUTION

Our project is a digital platform, providing an intuitive interface and addressing key issues. Let's explore the design patterns we use to create an efficient system.



1. OBSERVER PATTERN

- Role: SeatObserver, SeatSubject
- Description: The "Observer" pattern is used to track changes in seat status.

2. STRATEGY PATTERN

- Role: PricingStrategy, BasePricingStrategy, IncreasingPricingStrategy
- Description: The "Strategy" pattern is used to define various pricing strategies.

3. FACTORY METHOD PATTERN

- Role: SeatFactory, BasePricingSeatFactory, IncreasingPricingSeatFactory
- Description: The "Factory Method" pattern is employed for creating objects without explicitly specifying their class.

4. FACADE PATTERN

- Role: HomePageFacade
- Description: The Facade provides a simple interface for interacting with a more complex subsystem.

Why we choose Strategy Pattern?

Monitoring Changes {

The Observer pattern is used here to notify interested objects about changes in the state of seats on the plane. The "Observer" pattern efficiently implements the mechanism for notifying changes, providing timely updates to the user interface. SeatObserver is an interface that defines the update method, which is called when the seat status changes. SeatSubject is an object that is monitored by various observers (users objects), and it notifies them of changes.

}

Why we choose Strategy Pattern?

Flexibility in Pricing {

Utilizing various pricing strategies is essential for us. The "Strategy" pattern offers a flexible mechanism for defining, encapsulating, and swapping different price calculation algorithms. This allows adaptation to various pricing scenarios. The PricingStrategy defines the method of calculating the ticket price (calculate Price). BasePricingStrategy and IncreasingPricingStrategy implement different methods for calculating prices.

}

Why we choose Factory Method Pattern?

Flexibility in Object Creation {

We strive for flexibility in creating objects, such as seats with different pricing strategies. The "Factory Method" pattern provides an abstract interface for creating objects without specifying their concrete classes. This facilitates the easy addition of new strategies or seat types in the future.The SeatFactory defines an interface for creating seats, and the BasePricingSeatFactory and the IncreasingPricingSeatFactory implement methods for creating seats with different pricing strategies.

}

Why we choose Facade Pattern?

Simplified Interaction {

Addresses the need for simplifying the interaction with a complex subsystem. The HomePage Facade class can be viewed as a facade, providing a simple interface for users to interact with the more intricate subsystem, such as buttons, seat selection, etc., within the user interface. This enhances user experience.

}

SUMMARY

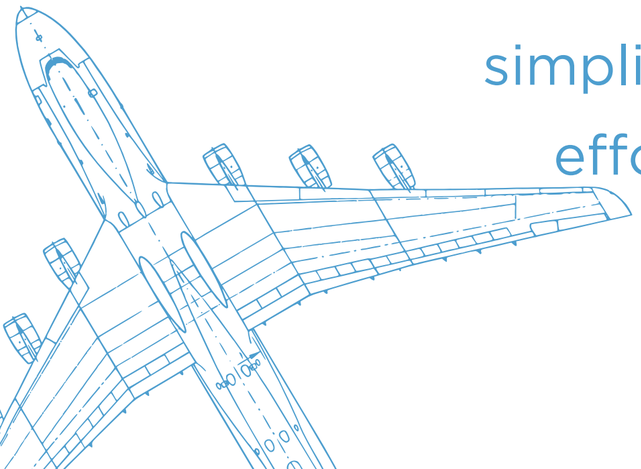
The choice of these patterns is driven by a commitment to flexibility, code readability, and ease of maintenance. The Observer ensures timely UI updates, the Strategy provides flexibility in pricing management, and the Factory Method simplifies the addition of new elements to the system the inclusion of the Facade pattern further enhances our project by simplifying user interaction with the graphical user interface. Together, they create an efficient and scalable architecture for our project, aligning with the goal of optimizing the flight booking process.

IMPLEMENTATION

- We aim to create an intuitively understandable interface for selecting flights, dates, and routes, as well as to provide flexible pricing strategies. Our digital platform will enable users to easily choose preferred seats, explore dynamic pricing options, and quickly book their flight tickets.

RESULTS

- We anticipate that implementing our solution will significantly simplify the ticket purchasing process, reducing the time and effort required. We strive to providing a convenient and efficient platform for ticket transactions.



In conclusion, our project not only addresses current challenges in the flight booking process but also offers innovative solutions using design patterns. We believe our digital platform will make the flight booking process more convenient and efficient for all users.

With Respect!

If you have any questions:
+7 706 418 81 47