Activity - Design Pattern Implementation

Implement A Design Pattern Of Choice

Design Patterns can be a very key part of software solutions that we use to solve system creation and extension related problems. They are broadly divided into 3 categories:

1. Creational Patterns - Patterns that are related to object creation. Example: *Singleton Pattern.*
2. Structural Patterns - Patterns related to the structure of these objects and classes. Example: *Façade Pattern.*
3. Behavioral Patterns - Patterns that are related to algorithms and inner-working of classes. Example: *Observer Pattern.*

Tasks

Repository for extending

```
https://github.com/Sidx-sys/Reservation-System-Starter
```

Choose design patterns of your choice to implement and enhance this system. *This also includes fixing existing inextensible code*. Mention clearly the design used and how you extended into your coding practice.

💡 Example

Use of the **Factory Method** design pattern, to add flexibility the system of creating planes (In `flight.reservation.plane`). In the current scenario, if we would want to extend the `plane` model, to add an additional attribute such as `air_hostesses_required`, one would have to add that attribute individually for all the `CASE` statements, making it inextensible.

Task 1

Explain using *code snippets,* that what could be done to solve the issue mentioned in the Example above. It could involve the classes that need to be created, how one could use these classes, etc. Mention how the code change solved the issue.

Task 2

Find another such issue in the code, and **extend it to follow a design pattern**, in the same way as done in Task 1.

Future Tasks

This activity would be extended to the next assignment, so it would be advisable to **fork** this repository and work on it to extend it. Further instructions are provided in the repository.