**Design Patterns Assignment**

Implemented design patterns:

• Factory Pattern: Produces items without disclosing the reasoning involved in their creation.

• Singleton Pattern: Ensures that a class has a single instance and offers a single point of access to it globally.

• Adapter Pattern: Allows interfaces that are incompatible to coexist.

• Facade Pattern: Offers a simplification to a complex subsystem's interface.

• Observer/PubSub Pattern: Creates a dependency between objects to alert users to changes.

• Strategy Pattern: Identifies and facilitates the interchangeability of a family of algorithms.

Factory Pattern 🡪

It is Creational pattern which means its used to create objects without revealing the details of class modules. This was used to create instances of all the notification channels/strategies without revealing the logic that they were created. SmsNotificationFactory and EmailNotificationFactory creates instances of their own notification strategies.

Singleton Pattern 🡪

The NotificationService class is implemented using the singleton pattern. Throughout the application, it makes sure that only one instance of the NotificationService class is produced. The class' private constructor avoids new instances of the class from being created by outside classes.This design pattern guarantees that a NotificationService which used to issue notifications class has a single instance and offers a single point of access to that instance globally.

Adapter Pattern 🡪

The INotification interface gives a standard contract for various notification kinds.

Implementing this interface allows the EmailNotification and SmsNotification classes to provide particular implementations for sending email and SMS notifications.

Facade pattern 🡪

The NotificationSystem class serves as a face for sending notifications. The difficulty of interacting with notification strategies gets solved here. The SendNotification function makes the process of sending notifications simpler without even knowing the details of the services and strategies.

Observer Pattern 🡪

The IUser interface defines the terms of the contract for users who can subscribe to notifications.

This interface is implemented by the User class to send notifications as soon as they are created or subscribed to. Whenever one user changes its state, all dependent objects are automatically notified and changed.

Strategy Pattern 🡪

EmailNotificationStrategy and SmsNotificationStrategy implement a common INotificationStrategy to offer several notification-sending methods. This enables us to switch between different notification methods. And this encapsulates the same type of algorithms together.