

Assignment

Design Patterns

# Implementation of Notification System

This implementation uses the **Factory Design Pattern** to create different types of notifications based on the channel of notification (Email/SMS), the **Singleton Pattern** to ensure that only one instance of the notification system is created, and the **Adapter pattern** to convert the interface of the notification system into an interface that can be used by different channels (Email/SMS). It also uses the **Facade pattern** to provide a simple interface for sending notifications and the **Strategy pattern** to define different strategies for sending notifications. It also uses Observer Pattern is used to notify multiple observers (subscribers) when a new notification is sent.

Below is a detailed implementation of the notification system:

1. **Factory pattern**: The factory pattern is used in the NotificationFactory class to create different types of notifications based on the channel of notification (Email/SMS). The createNotification method takes the channel and recipient as input and returns an instance of the appropriate notification class (EmailNotification or SMSNotification).

The factory pattern will help you create objects without exposing the instantiation logic to the client.

1. **Singleton pattern**: The Singleton pattern is used in the NotificationSystem class to ensure that only one instance of the notification system is created. The getInstance method returns the existing instance if it exists, or creates a new instance if it doesn’t.

The singleton pattern will help to avoid creating multiple instances of the notification system.

1. **Adapter pattern**: The adapter pattern is used to adapt the notification system to new channels of notification. The Adapter pattern is used in the EmailNotification and SMSNotification classes to convert the interface of the notification system into an interface that can be used by different channels (Email/SMS). Both classes implement the Notification interface and provide their own implementation of the send method.

The adapter pattern will help to add new channels of notification without modifying the existing code.

1. **Facade pattern**: The Facade pattern is used in the NotificationSystem class to provide a simple interface for sending notifications or for the client to interact with the notification system. The sendNotification method takes the channel, recipient, subject, and message as input and uses the NotificationFactory to create an instance of the appropriate notification class. It then calls the send method on that instance to send the notification.

The facade pattern will help to hide the complexity of the notification system from the client.

1. **Strategy pattern**: The strategy pattern is used to define different strategies for sending notifications based on the channel of notification (Email/SMS). Here I have used the Strategy pattern in the NotificationContext class to limit the length of the subject. The strategy pattern will help to change the behaviour of the notification system at runtime.
2. **Observer / PubSub pattern**: The observer / PubSub pattern is used to notify users who have subscribed for notifications. The observer / PubSub pattern will help to notify users without tightly coupling them with the notification system.