Part 4 of report

Question one :

1)Availability & Cost : For this project, one of the most important requirements is availability. The system should be available for users to post, comment, or browse items for sharing. Furthermore, it should be available for users to chatting and shipping decisions. This will increase the cost of the system's components such as servers.

2)Scalability & Performance : Increasing scalability within this system will affect its performance. This system must handle more than 3000 user per hour. If the number of users increase, while uploading pictures on posts and chats, this might affect the system performance or responding time.

Question 2:

LOC for posting functions(Create, Update, Delete, GetPost)= 140 line

LOC for login function (including style codes)= 219 line

LOC for chatting(create message, mark as read, update last time)=92 line

LOC for adding group (create group, update group, delete group)=141

CCM for posting function= 11

CCM for login function= 3

CCM for chatting= 9

CCM for adding group= 6

Question 3:

1) WMC : 1) User class = 27 method

2)Post class = 13 method

3) Payment class= 4

4)Comment class = 4

5) Community class = 9

6) Feedback class = 9

7) Profile class = 16

8) Local centers class = 7

9) Volunteer class = 8

10) Chat class = 8

11) Notification class =5

12) Employee class =8

13)Monthly report class = 4

14)Platform group class =10

15)Group admin class= 5

2) DIT :

1 For Employee abstract class.

1 For User class

3) NOC :

3 For Employee abstract class

2 For User class

4) CBO : There are no couple classes.

5) RFC : There are no classes that calls methods from other methods.

6) LCOM : No Subclasses use different methods.

The rest of part two of report:

c)

Join Local Group Group

User interface :

Listing Page / Join Request Form

User type: User

Post an Item

User interface:New Post Creation Page

User type: User

Comment on Post

User interface:Post Detail Page with Comments

User type: User

Report Inappropriate Post

User interface:Report Button + Flagging Form

User type:User

Deliver Item Delivery

User interface: Confirmation Interface

User type: Exchanger / User

Review Join Requests

User interface: Admin Dashboard – Join Requests

User type:Group Admin

User interface: Shipping & Payment Page

User type :User (Receiver)

h) Strategy to Implement Use Cases

Strategy Used: Model-View-Service (MVS)

Explanation:

Model: Holds system data and business logic (e.g., User, Group, Post, Report).

View: Presents the UI (Web pages, input forms, etc.).

Service: Connects the View and Model, coordinates business logic, validates input, and executes system operations.

Advantages:

1.Encourages modular design and clear separation of concerns.

2.Easy to scale and test components individually.3.Views can be reused with different services.

3.Services act like Use Case controllers, improving readability and traceability.

Disadvantages:

1.Slightly more initial setup than a centralized controller.

2.For very small applications, MVS might feel over-engineered.

j) Three Design Patterns Applied (within MVS Architecture)

We applied the following design patterns in the context of the Model–View–Service (MVS) architecture:

1. Factory Pattern (used in the Service layer)

Problem it solves: Creating different types of reports or notifications (e.g., MonthlyVolunteerReport, ExchangeReport) without exposing creation logic to the caller.

Application: The Service layer calls a ReportFactory to create appropriate report objects depending on the report type.

Impact on design:

Improves maintainability and separation of concerns.

Reduces tight coupling between Services and concrete Models.

Follows MVS by keeping logic inside Service layer.

2. Observer Pattern (used between Model and View)

Problem it solves: Keeping user interfaces updated in real time (e.g., notifications when a post is flagged or a request is accepted).

Application: The UserInterface observes the Model (or NotificationManager). When the Model changes, the View is notified automatically.

Impact on design:

Enhances responsiveness of the View.

Maintains loose coupling between Model and View.

Fits MVS well by automating UI updates.

3. Singleton Pattern (used in the Model layer)

Problem it solves: Preventing multiple instances of shared resources like Database connection or Configuration manager.

Application: A DatabaseConnector class is implemented as a Singleton to ensure a single shared connection instance.

Impact on design:

Ensures data consistency across the system.

Reduces resource usage.

Supports centralized access in the Model layer.

n) Forks or Cascades in Interaction Diagrams

Strategy Used: Forks

Explanation:

In our Sequence Diagrams, we used Forks when the System sends requests to multiple lifelines in parallel, without chaining.

Example: In “Review Group Join Requests”:

System → Database: getRequests()

System → GroupAdmin: notifyPendingRequests()

Advantages:

1.Centralized logic remains in the system.

2.Easier to debug and test independently.

3.Better control and validation inside the Service layer.

Disadvantages:

1.Slightly tighter coupling between the system and its dependencies.

2.May require more coordination logic in the controller/service layer