CECS 575 OBJECT ORIENTED ANALYSIS

ASSIGNMENT - 1

GROUP – 2

BASSAPA RUSHIL (029336955)

STEPHEN JAKKU (029348161)

GATTINENI ABHISHEK (029363371)

SAI VIJETA BODDAPATI (029348980)

CHANDUVARDHAN KANDAM (029325294)

VISHITH SAI GOUD JAKKU (029336175)

SAI TEJA POLAMPALLY (029336305)

CNF - Community Network Forum

CNF is a place where different communities of people can share their thoughts and involve in a conversation. In this context, community can be a single user or a group of users with similar interests.

- The community can share content by creating posts.
- Comments on posts will provide a discussion forum.
- Comments & posts can be upvoted or downvoted.

This is a web application built using ReactJS as frontend and java spring boot, MySQL as backend. Both backend and frontend will have separate servers (microservices architecture). We plan to deploy the application in Heroku or Google Cloud Platform. We will make use of Github actions for CI/CD process.

Github Repo Link

https://github.com/VishithSai/cecs575_group2

Use Cases

Use Case 1 & 2 – Fully Dressed Format

Use Case UC1: Create a community

Scope: CNF Web Application
Level: user-goal
Primary Actor: User (Registered member of CNF)
Stakeholders and Interests:

- User: User wants a new community to be created without any errors like duplication of communities.
- CNF Web Application: Wants to validate the new community and create one if there are no issues.
- Other Registered Users: Wants to be able to view and post in the newly created community.
- Developers: Wants everything to work as expected.

Preconditions: User is identified and authenticated.
Success Guarantee: A new community is successfully created without duplicating existing communities. Other users are able to view and post in the newly created community.
Main Success Scenario:

1. Registered User is identified and authenticated.
2. User clicks on create new Community.
3. User fills the form with all the required details.
4. User submits the form.
5. The data is validated and if everything is as expected a new community is created.
6. All the users can view and post in the community.

Extensions:

- Duplicate community (Failure case):
    1. Registered user is identified and authenticated.
    2. User clicks on create new Community.
    3. User fills the form with all the required details.
    4. User submits the form.
    5. Submission fails as the same community already exists.
    6. User either changes the community's name or stops the process.

Special Requirements:

- User authentication must be completed within 30 seconds 90% of the times.
- Web application need to have high availability and low failover.
- Response time needs to be adequate during large number of requests.

Frequency of Occurrence: Nearly continuous based on user's requirement to create community.

Use Case UC2: Create a post (Under a community)

Scope: CNF Web Application
Level: user-goal
Primary Actor: User (Registered member of CNF)
Stakeholders and Interests:

- User: User wants a new post to be created under a community.
- CNF Web Application: Wants to validate the new post and create one if there are no issues.
- Other Registered Users: Wants to be able to view, comment, upvote and downvote in the newly created post.
- Developers: Wants everything to work as expected.

Preconditions: User is identified and authenticated.
Success Guarantee: A new post is successfully created under a community. Other users can view, comment, upvote and downvote the newly created post.
Main Success Scenario:

1. Registered User is identified and authenticated.
2. User selects the community where he needs to post.
3. User clicks on create new post.
4. User fills the form with all the required details.
5. User submits the form.
6. The data is validated and if everything is as expected a new post is created under a community.
7. All the users can view, comment, upvote and downvote the post in that community.

Special Requirements:

- User authentication must be completed within 30 seconds 90% of the times.
- Web application need to have high availability and low failover.
- Response time needs to be adequate during large number of requests.

Frequency of Occurrence: Nearly continuous based on user's requirement to create a post.

Use Case 3 & 4 – Brief Format

Use Case UC3: Comment on a post

User logs in to CNF. The web application identifies and authenticates the user. User views the community posts and clicks on comment to share his thoughts about the post. User types the comment and submits the comment. CNF application stores the comment in the database and makes it visible to all the other users. Other users can upvote, downvote and reply to the newly created comment.

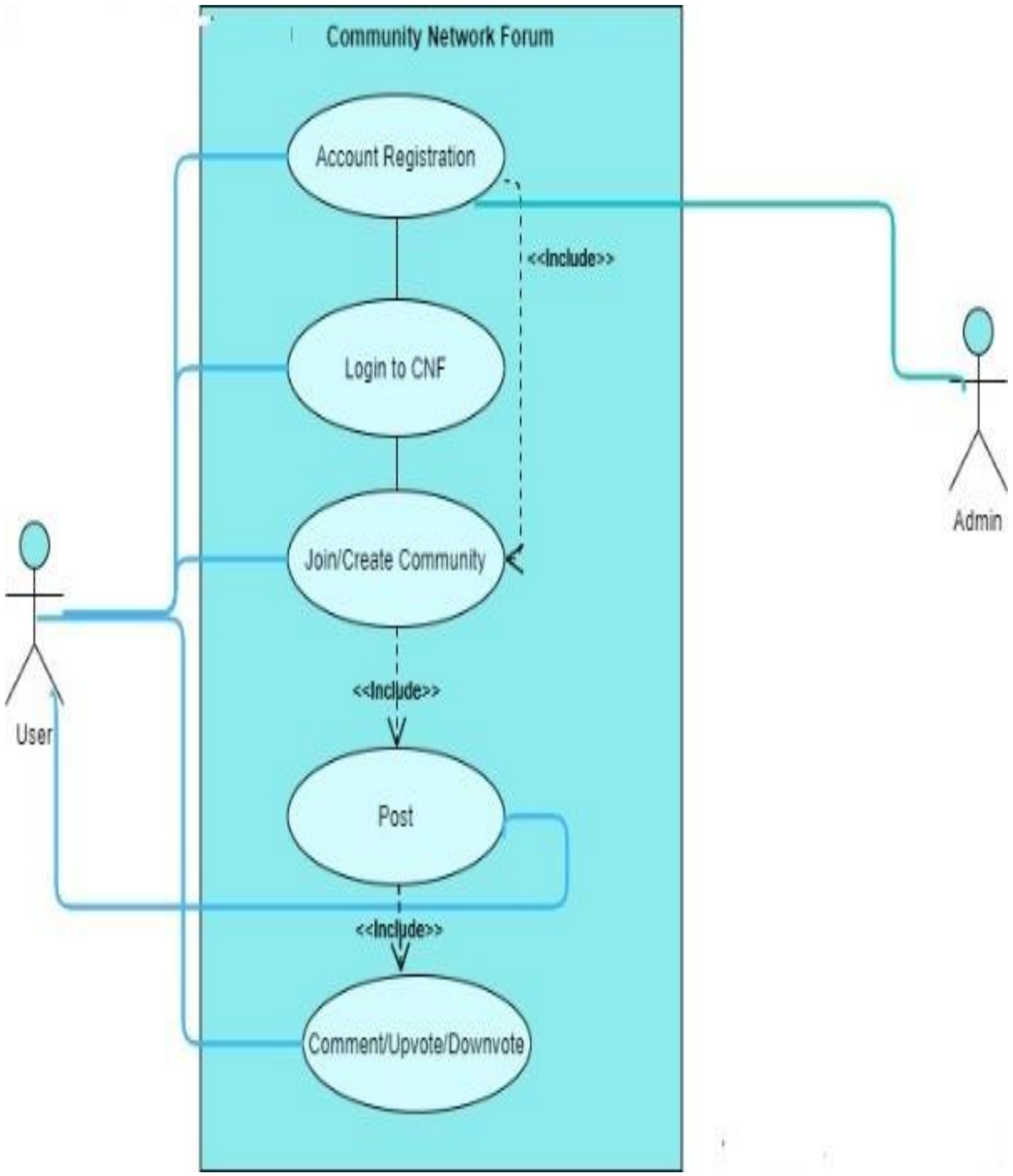
Use Case UC4: UpVote/DownVote on a project

User logs in to CNF. The web application identifies and authenticates the user. User views the community posts and can UpVote/DownVote based on his view on the post. CNF application stores the Vote data in the database and makes it visible to all the other users. Other users can view the number of UpVotes/DownVotes on the posts.

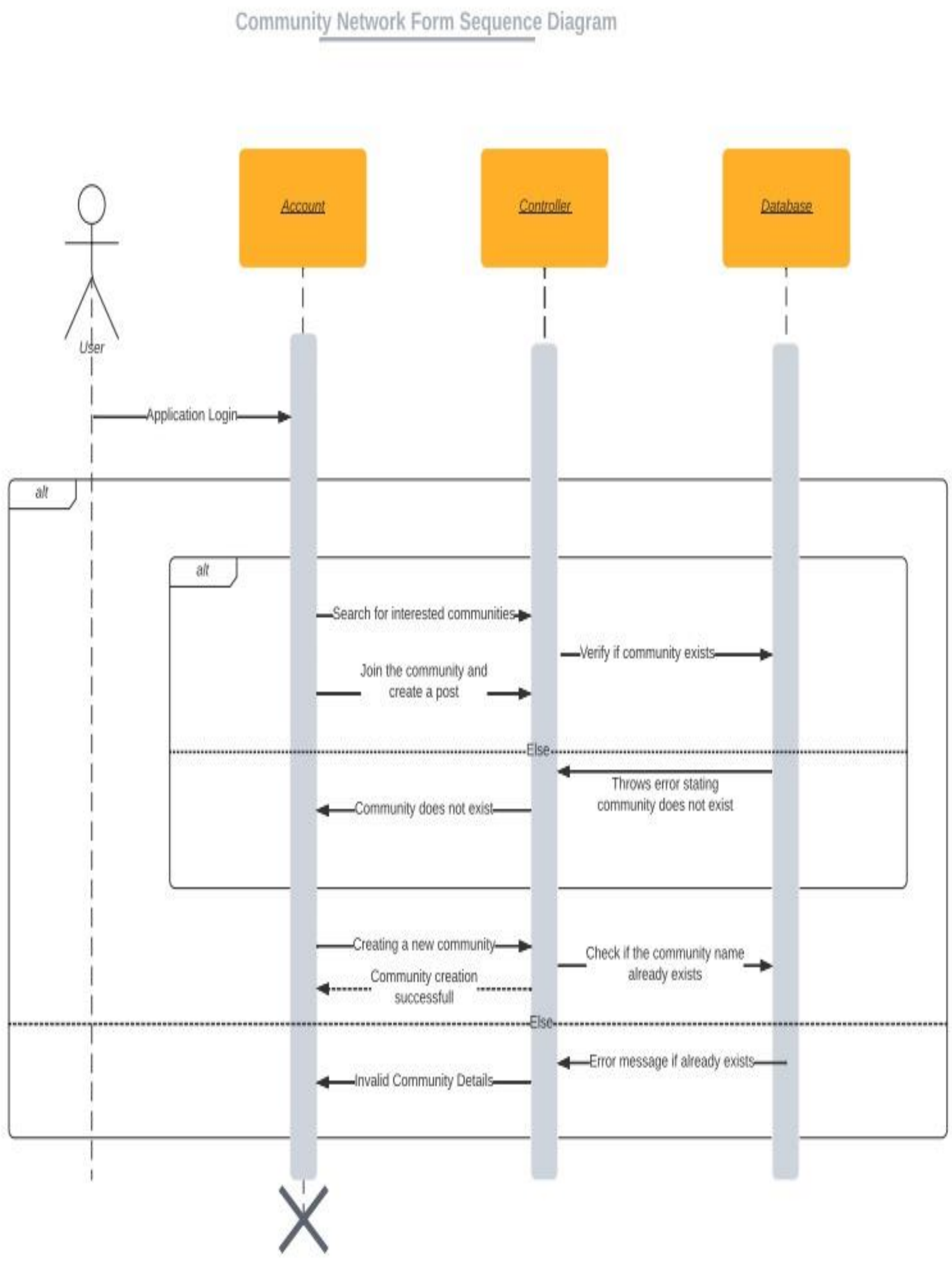
Non-Functional Requirements

FURPS	Functionality	Community Creation / Post Creation / Comment, UpVote or DownVote a post
	Usability	User interface must be consistent
	Reliability	System should be available 99% of the time
	Performance	User authentication must be completed within 30 seconds.
	Supportability	Web application is monitored frequently
+	Security	Unauthorized users has no access

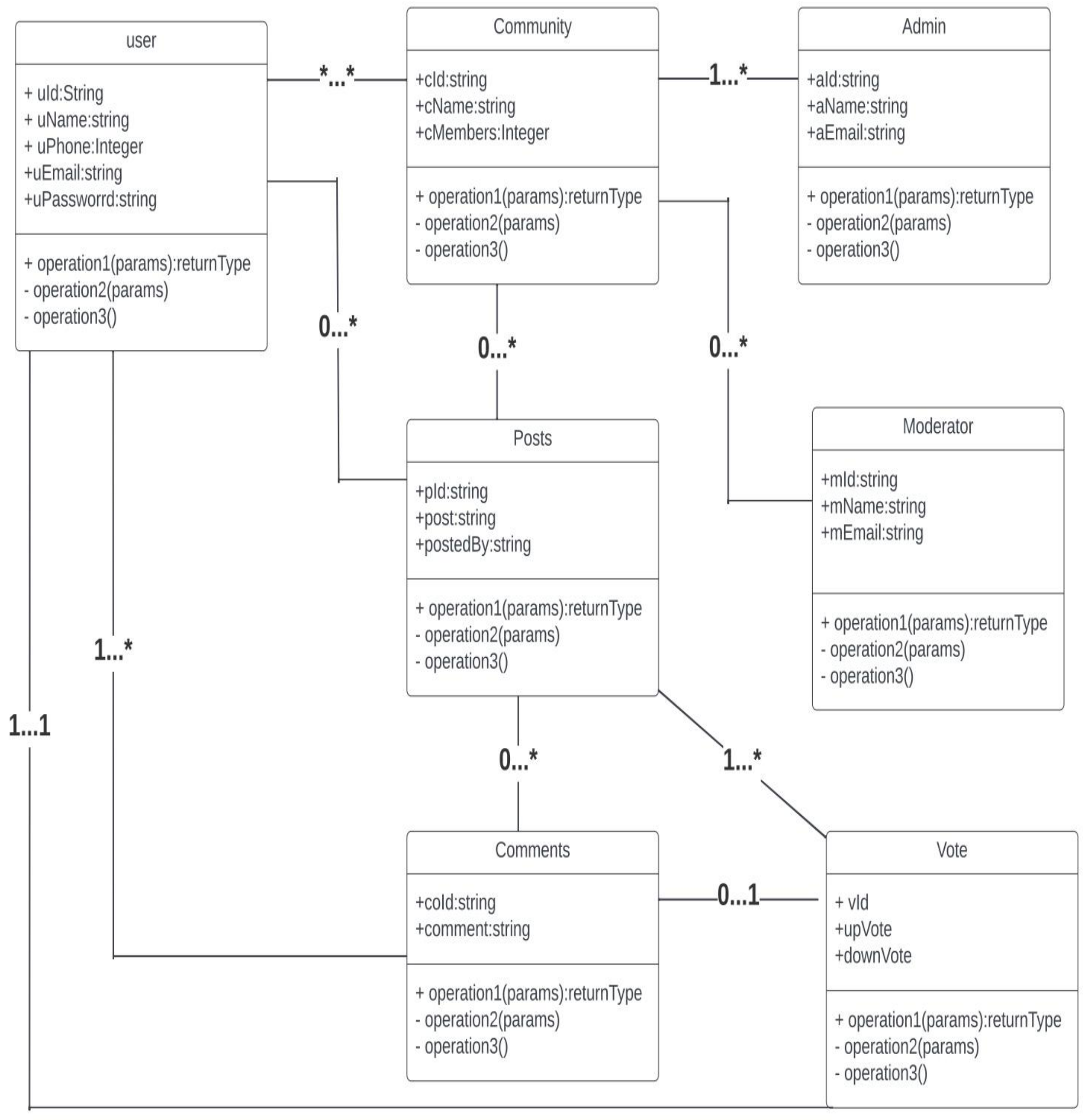
UML Use Case Diagram



UML SSD


Community Network Form Sequence Diagram
Account
Controller
Database
User
Application Login
alt
alt
Search for interested communities
Verify if community exists
Join the community and create a post
Else
Throws error stating community does not exist
Community does not exist
Creating a new community
Check if the community name already exists
Community creation successfull
Else
Error message if already exists
Invalid Community Details

UML Domain model



GRASP Patterns

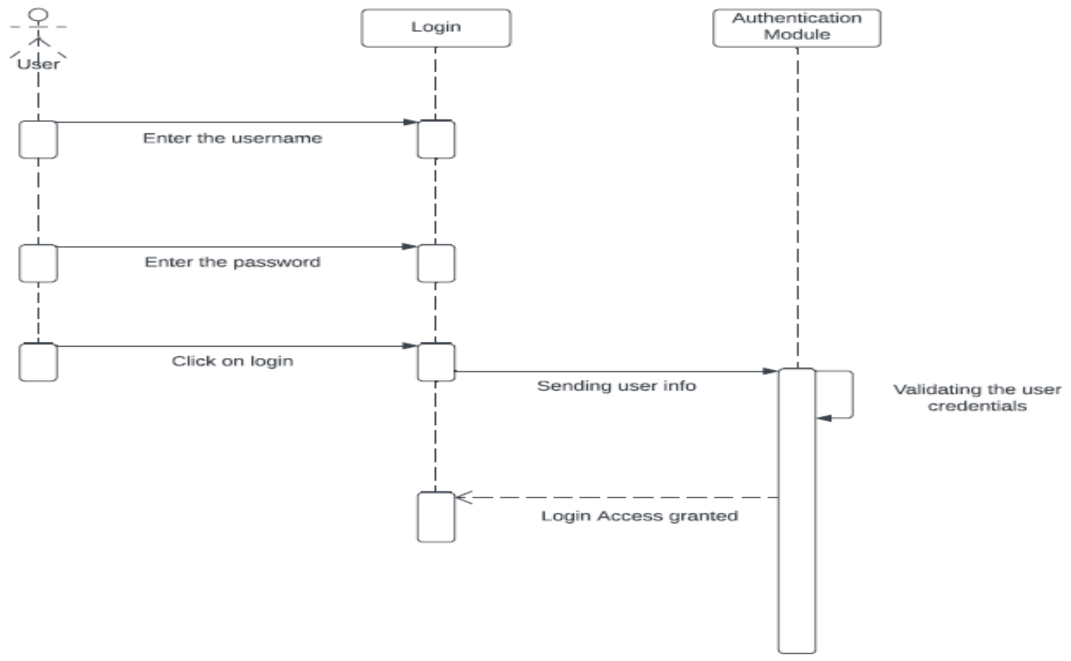
A) Controller

In our Project, the user will logon to the forum through a login interface. When the user presses the login button the first action that should be triggered is the Authenticator which will validate the user credentials and allows the user, access to the forum.

As per the definition of the Controller, the first object which receives the command from UI in this case is Authentication Module. Thus, Authentication Module is the Controller.

Class	Responsibility
Login	Responsible for knowing username and password.
Authentication module	Responsible for verifying login credentials of the user.

Sequence Diagram



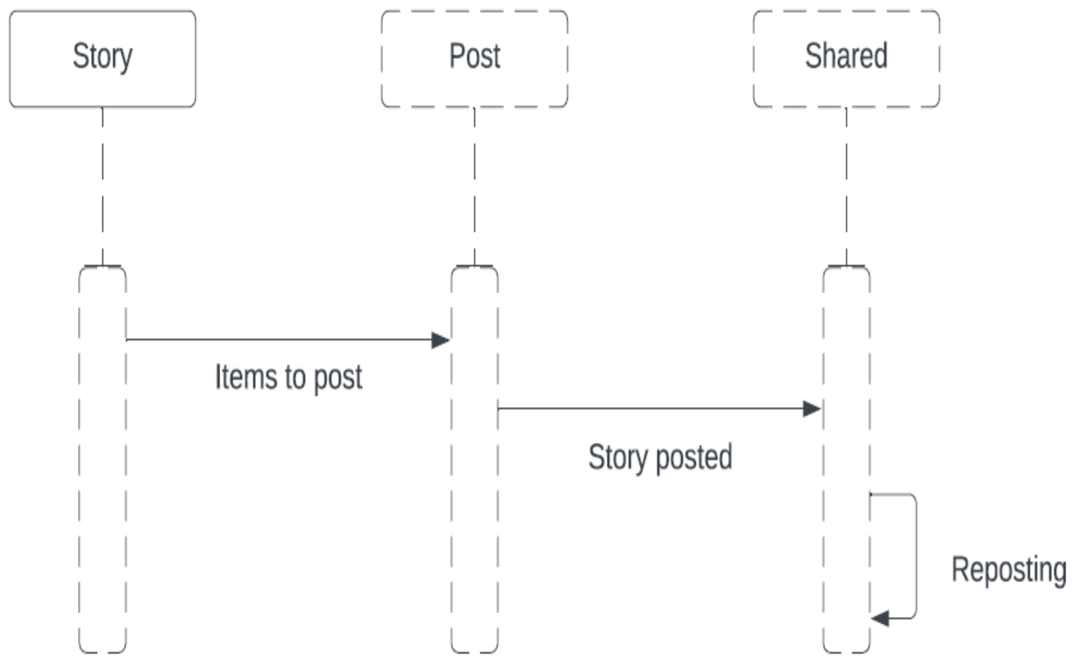
B) Creator:

In this case, ‘shared’ is an object which is instantiated by the class ‘Post’. The ‘Post’ class is given the responsibility of creating an instance of ‘shared’.

The post class is a Creator because it is responsible for creating a condition (instance) Shared. (Sharing a post). It is not possible to share, if there is no post.

Class	Responsibility
Post	Responsible for posting content on the feed.
Shared	Responsible for content sharing.

Sequence diagram



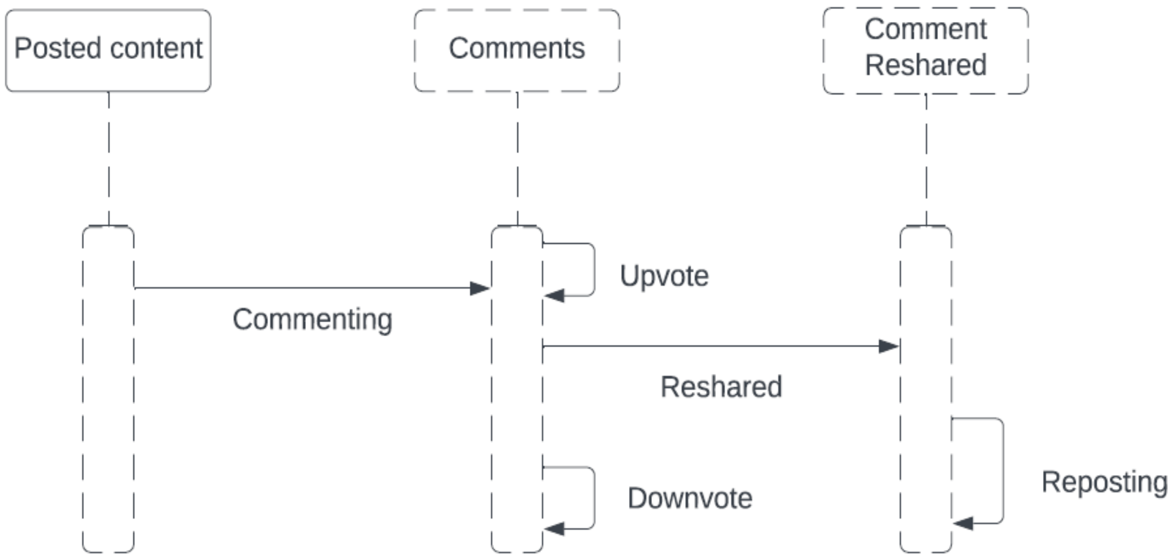
C) Low coupling:

Different kinds of reactions on the post like comments, upvote, downvote and reshare will have minimal impact on the actual post (There can be a post without reactions). Also, the reactions like comments, upvote etc. are dependent on the post itself (as there can't be reactions without a post) and sometimes help to support the post.

This indicates that Comments class is dependent on Post class but change in one class will have lower impact on other class. Both classes have lower dependency and are examples of Low Coupling.

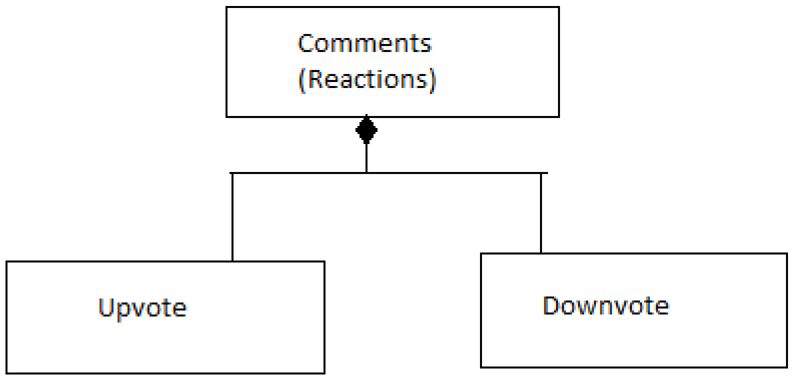
Class	Responsibility
Comments	Responsible for reactions on the Post
Post	Responsible for posting content on the feed

Sequence Diagram

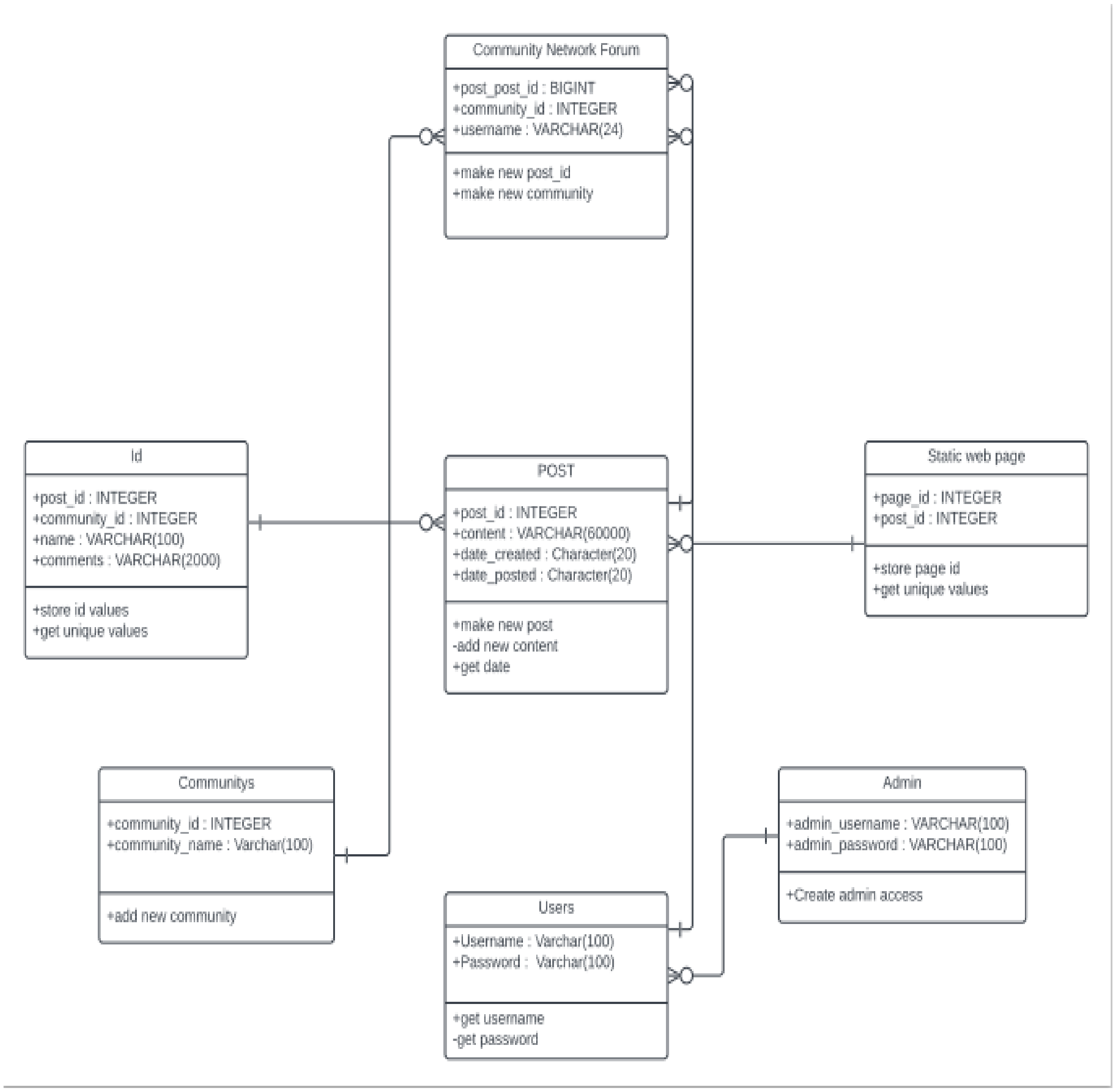


D) Polymorphism

In the Comments class, the vote task can be performed in two ways, Upvote and downvote. The same thing performed in different ways is known as Polymorphism. It helps us in handling new variations.



Design Class Diagram


Community Network Forum
+post_post_id : BIGINT
+community_id : INTEGER
+username : VARCHAR(24)
+make new post_id
+make new community
Id
+post_id : INTEGER
+community_id : INTEGER
+name : VARCHAR(100)
+comments : VARCHAR(2000)
+store id values
+get unique values
POST
+post_id : INTEGER
+content : VARCHAR(60000)
+date_created : Character(20)
+date_posted : Character(20)
+make new post
-add new content
+get date
Static web page
+page_id : INTEGER
+post_id : INTEGER
+store page id
+get unique values
Communitys
+community_id : INTEGER
+community_name : Varchar(100)
+add new community
Admin
+admin_username : VARCHAR(100)
+admin_password : VARCHAR(100)
+Create admin access
Users
+Username : Varchar(100)
+Password : Varchar(100)
+get username
-get password