IT314 - Software Engineering
Lab-6

Group 4 Members:

1. 202001002 - Suyash Vyas
2. 202001005 - Mann Joshi
3. 202001015 - Om Vaghani
4. 202001017 - Mihir Patel
5. 202001019 - Ruchi Detroja
6. 202001022 - Meet Shrimali
7. 202001025 - Nilav Shah
8. 202001030 - Ritik Mahyavanshi
9. 202001048 - Dakshveer Singh Chauhan

A domain analysis model is a method of visualizing and organizing the various concepts and interactions within a specific topic or industry. For News Aggregator project we can capture different aspects using the following domain analysis model

User Categories:

- Registered User : User on the platform who has created an account and has access to all the basic features such as Posting , Commenting , Voting or Reporting a post. They can also search posts on the basis of their interest in any specific topics also.

User Needs:

- ❖ Registered User:
  - ➢ Easy registration process.
  - ➢ Clear and easy-to-use interface for posting, commenting, voting or reporting a post.
  - ➢ Ability to personalize their searches and feed based on their interests
  - ➢ Ability to follow or see specific users and their profiles.

Functionalities:

- User Management: Creating, editing or Deleting user profiles.
- Content Management: Creating and deleting posts and comments.
- Voting: Allowing users to vote on any post or comments and determining their popularity on the basis of votes.
- Search: enabling user to search user profiles and posts for any specific categories.

Relationships:

- Users can create and manage their own profiles, as well as their post contents and participate in discussions.
- Voting and ranking algorithms determine the visibility and popularity of content within the system.

Design goals:

- User-friendly interface for browsing and posting content
- Efficient and secure processing of user interactions, such as upvoting, commenting, and moderating
- Accurate tracking and sorting of posts and comments by category and popularity
- Flexibility and customizability for moderators to manage and customize the community.

Boundary Objects:

1. User interface: This is the interface through which users can interact with the News Aggregator. It could include elements such as a homepage, post creation page, comment creation page, login/registration page, and settings page.
2. API endpoints: These are the endpoints through which external applications can interact with the News Aggregator, for example, to retrieve posts or comments.
3. Database schema: This represents the structure of the database used to store the News Aggregator data. It could include tables for users, posts, comments, and votes.

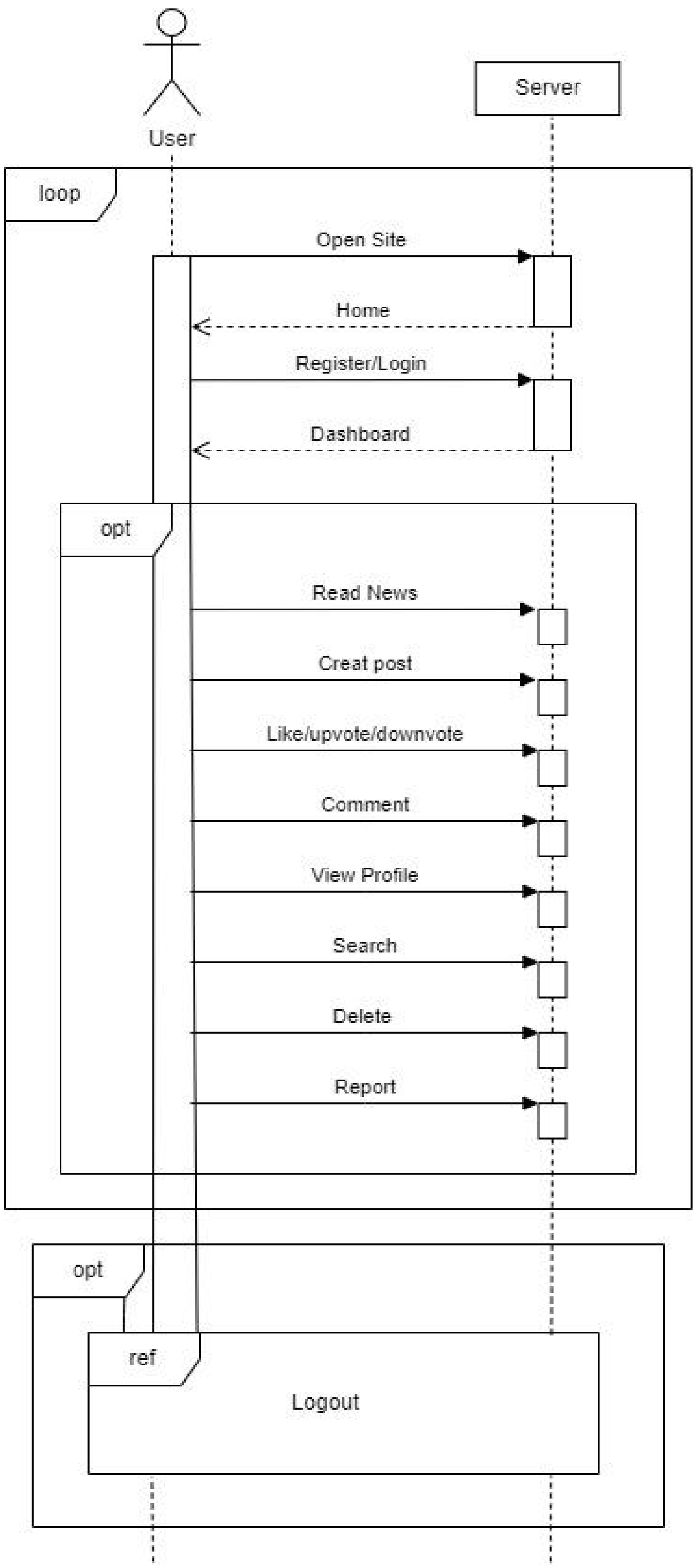
Controller Objects:

1. Post controller: This is responsible for handling user requests related to creating, retrieving, updating, and deleting posts.
2. Comment controller: This is responsible for handling user requests related to creating, retrieving, updating, and deleting comments.
3. User controller: This is responsible for handling user requests related to creating, retrieving, updating, and deleting user accounts.

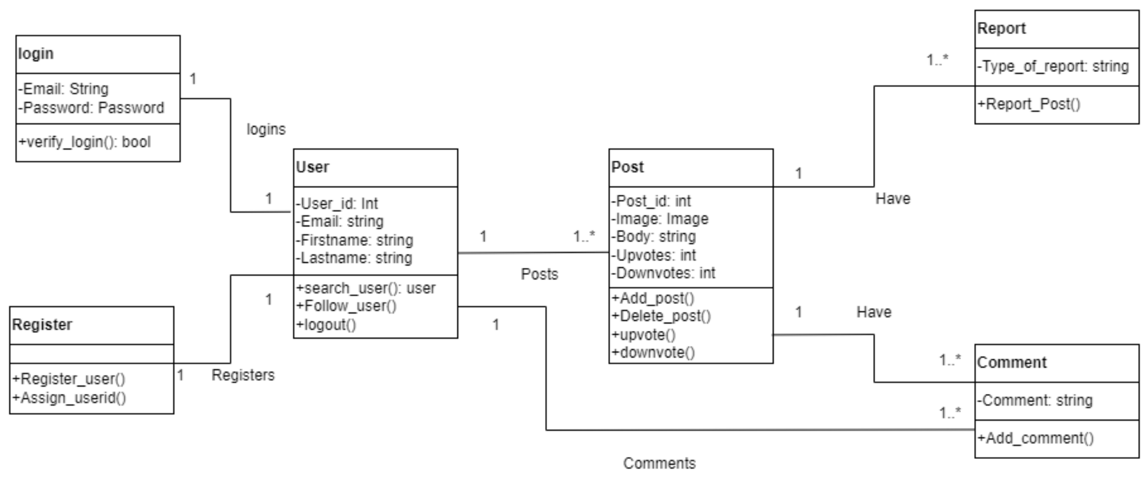
Entity Objects:

1. User entity: This represents a user in the system, and could include properties such as username, email, password, and account creation date.
2. Post entity: This represents a post in the system, and could include properties such as title, content, author, creation date, and upvotes/downvotes.
3. Comment entity: This represents a comment in the system, and could include properties such as content, author, parent post, creation date, and upvotes/downvotes.

Sequence Diagram



Class Diagram



High Level System Design

The high level system design could be based on the client-server architecture, with the following subsystems:

1. Client subsystem: This subsystem includes the user interface components that are presented to the user, such as web pages or mobile apps. The client subsystem communicates with the server subsystem to retrieve and submit data.

2. Server subsystem: This subsystem includes the server components that handle the business logic and data management for the system. It is responsible for receiving requests from the client subsystem, processing them, and returning responses. The server subsystem includes the following subsystems.

a) Web server: Handles incoming HTTP requests and serves static content (e.g. HTML, CSS, JavaScript files).
b) . Application server: Executes application logic (e.g. user authentication, post creation, comment moderation) and communicates with the database server.
c) . Database server: Stores and manages the system data, such as user accounts, posts, comments, and categories.

3. Third-party subsystem: This subsystem includes any external services or APIs that the system uses, such as payment gateways or email services.