60004210210

Amartya Mishra

COMPS-c31

SE Experiment 3

60004210210
Amartya Mishra
COMPS3 - C31

SE EXPERIMENT - 3

Aim: Identify Scenario & develop UML use case & class Diagram.

Theory:

Abstraction & Simplification:
class Diagrams abstract Complex Systems into managable components focusing on class & relation ship.

communication:
They provide a common language for Stakeholders to understand & discuss the system Structure & behavior.

BluePrint & implementation:
class diag. guide developers in writing code, defining class hierarchies & Establishing Relationships.

Analysis & design:
They are in early stage & analysis & design by identifying key classes & relation ship facilitating iterative refining.

Modularity & reusability:
Organize classes that coherent units promotes modularity & reusability facilitating maintenance

·Documentation:
class diagram serves as a valuable document offering a ~~v~~ visual representation of system's architecture for reference throughout the development life cycle.

Experiment 3

Darshit Sarda 60004210208

Amartya Mishra 60004210210

Gaurang Bhogle 60004210192

Shubham Mehta 60004210191

Aim: Identify scenarios & develop UML Use case and Class Diagram for the project

Theory:

A class diagram in software engineering serves as a foundational tool for visualizing the structure and relationships within a system. It is a graphical representation of the classes, attributes, operations, and relationships among objects that make up the software system. Here's a breakdown of its importance and roles:

1. Abstraction and Simplification: Class diagrams help in abstracting complex systems into manageable components. By representing classes and their relationships visually, developers can focus on high-level design decisions without getting bogged down in implementation details.

2. Communication: Class diagrams serve as a common language between stakeholders, including developers, designers, and clients. They provide a clear and concise way to communicate the system's structure and behavior, facilitating collaboration and understanding among team members.

3. Blueprint for Implementation: Class diagrams provide a blueprint for the implementation phase of software development. Developers use them as a guide to write code, define class hierarchies, and establish relationships between objects.

4. Analysis and Design: During the early stages of software development, class diagrams aid in analysis and design. They help identify the key classes and their relationships, allowing developers to model the system's structure before writing any code. This iterative process enables refining the design based on feedback and changing requirements.

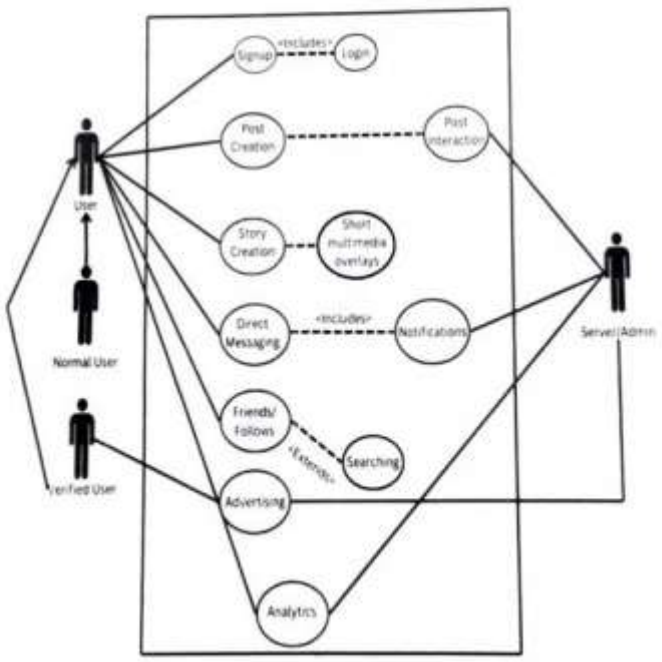
5. Modularity and Reusability: Class diagrams promote modularity and reusability by organizing classes into coherent units. Well-designed class hierarchies and relationships allow developers to encapsulate functionality within classes, making it easier to maintain and extend the system over time. Additionally, reusable classes can be leveraged across multiple projects, reducing development time and effort.

6. Documentation: Class diagrams serve as valuable documentation for the software system. They provide a visual representation of the system's architecture, which can be referenced by developers, testers, and other stakeholders throughout the software development

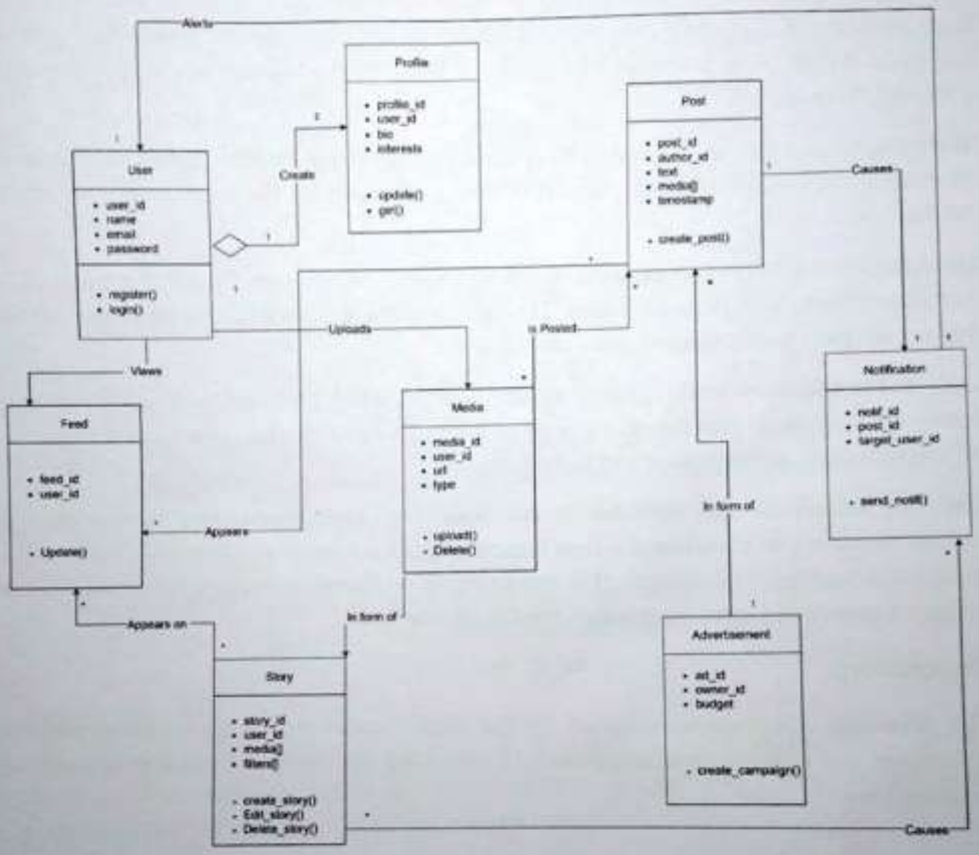
lifecycle. Additionally, class diagrams can be supplemented with additional documentation, such as class descriptions and usage guidelines, to further enhance understanding.

In essence, class diagrams play a crucial role in software engineering by facilitating communication, guiding implementation, promoting modularity and reusability, and serving as documentation for the system. They are indispensable tools for designing and understanding complex software systems, helping teams build robust and maintainable software products.

Use case Diagram:



Class Diagram:



User and Profile: The User class represents the core entity of the system, encompassing user accounts and authentication. Each User has an associated Profile, which stores additional information about the user, such as their bio and interests. This relationship is a composition, where a Profile object cannot exist without a User object.

User and Post: A User can create multiple Posts, which can contain text, media, and other content. This relationship is a one-to-many association, where one User can be associated with multiple Post objects.

Post and Media: A Post can be composed of one or more Media objects, which represent images, videos, or other multimedia content. This is a composition relationship, where Media objects are owned by and exist within the context of a Post.

Feed andl Post: The Feed class is an aggregation of multiple Post objects. A Feed represents the collection of posts that a User will see in their personalized content feed. This relationship allows for efficient retrieval and organization of posts for each user.

User and Story: A User can create multiple Stories, which are ephemeral multimedia content similar to Instagram or Snapchat stories. This is another one-to-many association between User and Story objects.

Story and Media: Similar to Posts, a Story can be composed of multiple Media objects, such as images and videos. This composition relationship allows for the creation of rich, multi-media stories.

User and Media: Users can upload and manage Media objects directly, without necessarily associating them with Posts or Stories. This one-to-many relationship enables users to store and access their media content independently.

User and Notification: Users can receive Notifications, which are triggered by various events within the system, such as new posts or interactions. This one-to-many relationship allows for efficient delivery of notifications to individual users.

User and Advertisement: The Advertisement class represents promotional content created by business users or advertisers. A User (specifically, a business user) can create and manage multiple Advertisement objects. This one-to-many relationship enables businesses to run targeted advertising campaigns within the Circlify platform.

Conclusion:

The proposed class analysis diagram for the Circlify social media application provides a structured and object-oriented approach to modelling the system's components and their relationships.