Software Requirements Specification

for

The Posting Hub

Version 1.0 approved

Group 2

Sharjeel Ali 22L-6721
Noor ul Ain 22L-6693
Anas Rahim 22L-6595
Eemaan Ahmad 22L-6744

16/03/2025

Table of Contents

1. Introduction

1.1 Purpose

This document defines the functional, non-functional, and external interface requirements for The Posting Hub, a web-based blogging platform. It covers the software's initial release (Version 1.0). The scope of this SRS encompasses the entire system, including both user-facing functionalities and administrative capabilities. The document serves as a guide for developers, testers, and stakeholders, ensuring a shared understanding of the system's capabilities and constraints. It also aims to provide a clear and structured foundation for software development, minimizing ambiguities and aligning with the project's business goals.

1.2 Document Conventions

This SRS document adheres to the following conventions:

- **Requirement Prioritization**: Each requirement statement is assigned a priority level (High, Medium, or Low) to indicate its importance for the functionality of the system.
- **Formatting**:
  - ○ Bold is used for section headings and key terms.
  - ○ Italics are used for emphasis.
  - ○ Monospace is used for technical terms, code, or system responses.
- **Document Structure**: The document follows a structured format with sections organized hierarchically to provide a systematic overview of the software requirements.
- **Requirement Numbering:**All requirements are uniquely numbered using the format **REQ-XX**, where **XX** represents the specific requirement number.
- **Change Management**: TBD (To Be Determined) is used as a placeholder for incomplete or pending information.

1.3 Product Scope

Description:

The Posting Hub is a user-friendly, interactive blogging platform designed to simplify content creation and engagement for users. The platform will support a seamless blogging experience, allowing users to write, edit, and publish posts with rich formatting options. It will also incorporate social engagement features such as liking, commenting, and sharing posts.

Purpose:

The purpose of The Posting Hub is to address the limitations of existing blogging platforms by offering a simple, customizable, and user-friendly solution for content creation and community engagement. The platform will serve as a medium for knowledge exchange, discussion, and content discovery for users who wish to share their thoughts, expertise, and ideas online.

Benefits:

Convenience: Easy-to-use platform for creating, managing, and engaging with blog posts.

Efficiency: Streamlined content creation with rich text editing and categorization.

Accuracy: Advanced search and filtering for precise content discovery.

Objectives and Goals:

Increase User Engagement: Foster interactions through likes, comments, and notifications.

Enhance User Experience: Provide a customizable and intuitive interface for seamless blogging.

Optimize Content Management: Enable efficient organization of posts using tags, categories, and filters.

---

2. Overall Description

2.1 Product Perspective

The Posting Hub is a standalone web-based application designed to facilitate seamless content creation, sharing, and engagement. It is not part of any existing product family but is built to address the limitations of current blogging platforms by offering simplicity, customization, and robust engagement features.. While The Posting Hub operates independently, it may interact with external systems such as third-party APIs for email/phone verification and notifications, as well as a database for storing user data, blog posts, and engagement metrics.

2.2 Product Functions

The major functions of The Posting Hub include:

- User Authentication and Management
- Blog Posting and Editing
- Content Formatting and Organization
- Post Engagement
- Search and Filtering
- Notifications and Bookmarking
- Reporting and Feedback

2.3 User Classes and Characteristics

User Classes:

- Regular Users (Bloggers and Readers): Create, manage, and engage with blog posts.

User Characteristics:

- Regular users:

➔ Range from casual bloggers to experienced content creators.

➔ Engage frequently with the platform for posting, reading, and community interaction.

➔ Come from diverse backgrounds with varying levels of familiarity with blogging tools.

2.4 Operating Environment

The Posting Hub will operate in a web-based environment. The following operating environment is required:

Hardware Platform:

Servers: Standard web servers

Client Devices: Compatible with desktops, laptops, tablets, and mobile devices.

Operating System Compatibility:

Compatible with major operating systems, including Windows, macOS, Linux, iOS, and Android.

Software Components:

Frontend: Built using HTML, CSS, and JavaScript for a responsive and interactive user interface.

Backend: Powered by Django (Python-based framework) for robust server-side logic and database management.

Web Browsers: Compatible with modern web browsers such as Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge.

2.4 Design and Implementation Constraints

- **Technology Stack**: The BlogPost system will be implemented as a web application using HTML, CSS, JavaScript, Python (Django) for backend, and a relational database.
- **Hardware Compatibility**: The system will be accessible across browsers
- **Security Considerations**: User authentication and authorization will follow industry standards, with strong password enforcement.
- **Data Integrity**: The system must ensure that blog posts, comments, likes, and other interactions remain consistent and free from corruption.
- **User Interface**: The UI must be responsive and follow accessibility guidelines to ensure usability for all types of users.

2.5 User Documentation

To ensure a smooth user experience, the following documentation will be provided:

User Manuals:

End User Manual: A comprehensive guide for users on how to create accounts, post blogs, engage with content, and use search/filter features.

Online Help:

Contextual Help: Tooltips and inline help text within the application to assist users in real-time.

FAQ Section: A dedicated FAQ page addressing common user queries and issues.

Video Tutorials: Step-by-step video guides for key functionalities like account creation, blog posting, and profile management.

2.6 Assumptions and Dependencies

Assumed factors:

- Users will have **basic digital literacy** to interact with blogging and social engagement features.
- Internet connectivity is required for accessing and using *The Posting Hub*.
- Users will adhere to **community guidelines** and content moderation policies.
- The system will receive **regular maintenance and updates** to ensure stability.

Dependencies:

- **Django Framework** – The backend relies on Django for user authentication, data handling, and API integration.
- **Database System** – A relational database (MySQL) is required for data storage and retrieval.
- **Hosting Services** – Deployment on a reliable cloud hosting platform (e.g., AWS, Heroku).

3.External Interface Requirements

3.1 User Interfaces

The logical characteristics of the user interface include:

- A **user-friendly web-based interface** for regular users, ensuring seamless navigation and accessibility.
- **Sample screen images** will be provided in a separate user interface specification to illustrate the platform’s layout, user flows, and key functionalities.
- **GUI standards** will follow modern web design principles, ensuring a clean, responsive, and intuitive user experience.
- **Screen layout constraints** will be based on a responsive design approach, ensuring compatibility across various devices, including desktops, tablets, and smartphones.
- **Standard buttons and functions** such as "Create Post," "Edit Profile," "Like," "Comment," "Bookmark," "Report," and "Delete" will be available to enhance user interaction.

- **Navigation menus** will provide access to major sections.
- **Post creation and editing tools** will support text formatting options such as Bold, Italics, Lists, and media insertion (images, links).
- **Search and filter options** will allow users to find content based on keywords, tags, categories, and user preferences.
- **Notification system** will allow real-time alerts for actions such as new comments, likes, and connection requests.
- **Error messages and validation feedback** will follow standard display conventions, providing clear and actionable responses to users in case of incorrect inputs, failed actions, or unauthorized access attempts.
- **Security measures** such as CAPTCHA verification for user login and signup will prevent spam and automated attacks.

3.2 Hardware Interfaces

The system will interact with the following hardware components:

- **Client-Side (User Devices):** Desktops, laptops, Android/iOS mobile devices.
- **Server-Side Requirements:** Web server with Django and MySQL support.
- **Peripheral Devices:** Keyboard, mouse, touchscreen for mobile users.

3.3 Software Interfaces

The software product will interface with the following software components:

- **Database Management System:** MySQL will be used for storing and retrieving user data, blog posts, comments, likes, and other relevant information efficiently.
- **Authentication and Authorization System:** Django's built-in authentication framework will handle basic user authentication, supplemented by third-party APIs for email/phone verification and notifications.
- **Web Browsers:** Compatible with modern web browsers such as Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge for seamless user interaction.
- **Operating Systems:** Compatible with major operating systems, including Windows, macOS, Linux, iOS, and Android, for platform compatibility.

3.4 Communications Interfaces

The software product will require the following communication functions:

- **Web Browser for User Interaction:** The Posting Hub will be accessed through a web browser, enabling users to interact with the platform using a responsive and intuitive interface.
- **Network Server Communication Protocols:** The application will use HTTP/HTTPS protocols to ensure reliable and secure data transmission between the client-side (browser) and server-side (Django backend).
- **Electronic Forms for User Input and Submission:** Users will interact with the system through various forms, such as sign-up, login, blog post creation, comments, and reporting. These forms will validate inputs before submission to ensure data integrity.
- **Communication Security and Encryption:** Secure communication will be ensured using HTTPS with SSL/TLS encryption to protect user data, including login credentials, personal information, and blog content.

---

4 System Features

4.1 Registration

- **Description and Priority:** Enables new users to sign up using email or phone verification. High priority.
- **Stimulus/Response:** User enters credentials -> System validates -> Account is created.
- **Functional Requirements:**
  - Email/phone-based signup with verification.
  - Password strength validation.
  - Automatic basic profile creation.

Use Case: User Sign Up

Use Case ID	**UC-01**

Name	User Sign Up
Actors	User, System
Precondition	User is not registered
Description	User signs up using email or phone number
Flow of Events	1. User enters email/phone and password 2. System validates input 3. User re-enters password for verification 4. System validates password's strength 5. System verifies code and creates account
Postcondition	Account is created
Exceptions	Invalid email/phone, weak password, account already exist

4.2 Login

- **Description and Priority**: Allows users to log in via email, phone, or username. High priority.
- **Functional Requirements**:
  - Multi-method authentication (email/phone/username).
  - Secure password storage and authentication.
  - Password reset functionality.

Use Case: User Login

Use Case ID	**UC-02**
Name	User Login
Actors	User, System
Precondition	User is registered
Description	User logs in using email, phone, or username

Flow of Events	1. User enters email or phone number 2. System verifies email or phone number 3. User enter password 4. System verifies credentials 5. User is authenticated
Postcondition	User is logged in
Exceptions	Incorrect password, account does not exist

Use Case: Forget Password

Use Case ID	**UC-03**
Name	Forget Password
Actors	User, System
Precondition	User is registered and logged in
Description	User resets password
Flow of Events	1. User requests password reset 2. System sends reset link 3. User enters new password 4. System updates password
Postcondition	Password is changed
Exceptions	Invalid reset link, same password entered

4.3 Profile Management

- **Description and Priority**: Allows users to update their personal information. Medium priority.
- **Functional Requirements**:
  - Update profile picture, email, phone, and password.
  - Manage connected social media accounts.
  - Delete account functionality.

Use Case: Delete Account

Use Case ID	UC-04
Name	Delete Account
Actors	User, System
Precondition	User is logged in
Description	User deletes their account
Flow of Events	1. User requests account deletion 2. System confirms action by requesting password 3. User enters password 4. System verifies the password 5. System removes account
Postcondition	Account is deleted
Exceptions	Incorrect Password

Use Case: Edit Profile

Use Case ID	UC-05
Name	Profile Management
Actors	User, System
Precondition	User is logged in
Description	User manages their profile
Flow of Events	1. User updates profile details 2. System validates the input 3. User presses the save button 4. System saves changes
Postcondition	Profile is updated
Exceptions	Invalid input

Use Case: View User's Profile

Use Case ID	**UC-06**
Name	View User's Profile
Actors	User, System
Precondition	User is logged in
Description	Allows users to view another user's profile
Flow of Events	1. User navigates to profile 2. System retrieves and displays profile details
Postcondition	User views another user's profile
Exceptions	Profile not found → Display error message

4.4 Blog Posting

- **Description and Priority**: Users can create, edit, delete, and format blog posts. High priority.
- **Functional Requirements**:
  - Create posts with tags, images, and links.
  - Save posts as drafts before publishing.
  - Delete posts and associated comments.

Use Case: Read Post

Use Case ID	**UC-07**
Name	Read Post
Actors	User, System
Precondition	User is logged in
Description	Allows users to read available posts
Flow of Events	1. User clicks on the post 2. System request and displays posts
Postcondition	User has viewed the post
Exceptions	None

Use Case: Add Post

Use Case ID	**UC-08**
Name	Blog Posting
Actors	User, System
Precondition	User is logged in
Description	User creates and manages blog posts
Flow of Events	1. User creates a post 2. User adds tags, images, and links 3. System saves and uploads the post
Postcondition	Post is created
Exceptions	Invalid content

Use Case: Delete Post

Use Case ID	**UC-9**
Name	Delete Post
Actors	User, System
Precondition	User is logged in and owns the post
Description	Allows users to delete their own posts
Flow of Events	1. User selects post 2. System confirms deletion 3. System removes post
Postcondition	Post is deleted
Exceptions	none

4.5 Post Engagement

- **Description and Priority**: Users can like, comment, and share posts. High priority.
- **Functional Requirements**:
  - Like/unlike posts.
  - Comment and delete your own comments.
  - Share posts on external platforms.

.

Use Case: Like

Use Case ID	**UC-10**
Name	Like Post
Actors	User, System
Precondition	User is reading the post
Description	Allows users to like a post
Flow of Events	1. User clicks the like button 2. System registers the like and updates the post
Postcondition	Post reflects the like count
Exceptions	None

Use Case: Comment

Use Case ID	**UC-11**
Name	Comment on Post
Actors	User, System
Precondition	User is reading the post
Description	Allows users to add a comment on a post

Flow of Events	1. User clicks the comment button 2. System ask for the input 3. User types and presses the submit button 4. System validates and adds the comment to the post
Postcondition	Comment is successfully posted
Exceptions	none

Use Case: Delete Comment

Use Case ID	**UC-13**
Name	Delete Comment
Actors	User, System
Precondition	User is logged in and owns the comment
Description	Allows users to delete their own comments
Flow of Events	1. User selects the comment and clicks delete 2. System confirms and removes the comment from the post
Postcondition	Comment is deleted
Exceptions	None

Use Case: Save Post

Use Case ID	**UC-14**
Name	Save Post
Actors	User, System
Precondition	User is logged in
Description	Allows users to save a post for later
Flow of Events	1. User clicks save 2. System adds the post to the saved posts list

Postcondition	Post is successfully saved
Exceptions	None

4.6 Search and Filter

- **Description and Priority**: Users can search for posts by keywords, tags, and categories. High priority.
- **Functional Requirements**:
  - Keyword-based search.
  - Filter by date, category, and engagement metrics.
  - Sort posts based on relevance.

Use Case: Search Option

Use Case ID	**UC-15**
Name	Search Option
Actors	User, System
Precondition	User is logged in
Description	User searches content
Flow of Events	1. User enters searches post by heading or based on tags used 2. System fetches results 3. User views results
Postcondition	Results are displayed
Exceptions	No matching results

4.7 Connect with Users

- **Description and Priority**: Allows users to establish connections with others. Medium priority.
- **Functional Requirements**:
  - Send and accept/reject connection requests.
  - Manage connection lists.
  - Disconnect from users.

Use Case: Connect with User

Use Case ID	UC-16
Name	Connect with User
Actors	User, System
Precondition	User is logged in
Description	Users connect with each other
Flow of Events	1. User sends connection request 2. Recipient accepts/rejects 3. System updates connection status
Postcondition	Connection is established
Exceptions	None

4.8 Notifications

- Description and Priority: Users receive alerts for interactions. Medium priority.
- Functional Requirements:
  - Notifications for likes, comments, and connection requests.
  - Email and push notifications.

Use Case: Notifications

Use Case ID	UC-17
Name	Notifications
Actors	System
Precondition	Trigger event occurs
Description	System sends notifications

Flow of Events	1. Trigger event occurs 2. System sends notification to respective recipients
Postcondition	Notification is sent
Exceptions	None

4.9 Report Issues

- **Description and Priority**: Allows users to report inappropriate content. Medium priority.
- **Functional Requirements:**
  - Report posts, comments, and user behavior.
  - Admin review system for flagged content.

Use Case: Report Object (Object can be a Post, Comment, or User)

Use Case ID	**UC-18**
Name	Report Issues
Actors	User, System
Precondition	User is logged in
Description	User reports an issue
Flow of Events	1. User reports post 2. System request for reasoning 3. System records report
Postcondition	Report is recorded
Exceptions	None

5.Other Nonfunctional Requirements

5.1 Performance Requirements

- System should respond to user actions within 2 seconds under normal load.
- Handle up to 1000 concurrent users without performance degradation.
- Database queries should be optimized for quick retrieval.

5.2 Safety Requirements

- Secure user data storage with encryption.
- Regular backups to prevent data loss.

5.3 Security Requirements

- Enforce strong password policies.
- Implement user identity verification mechanisms.
- Protect against common web vulnerabilities (SQL injection)

5.4 Software Quality Attributes

- Usability: Intuitive UI with accessibility features.
- Reliability: 99.9% uptime for availability.
- Maintainability: Modular design for easy updates.

5.5 Business Rules

- Only registered users can post, like, and comment.
- Admins have full control over content moderation.
- Users can report inappropriate content, leading to review and potential removal.

6.Other Requirements

- **Database Requirements**

- The system should utilize a relational database management system (RDBMS) to store user data, booking information, and system configurations.
- The database should be designed to efficiently handle concurrent read and write operations during peak usage times.

● Internationalization Requirements

- The system should support multiple languages to cater to a diverse user base, with English as the default language.
- Date and time formats should be customizable based on user preferences and regional standards.

● Legal Requirements

- The system should comply with all relevant laws and regulations pertaining to online payment processing, data protection, and consumer rights in the regions where it operates.
- Terms of service and privacy policy documents should be displayed and agreed to by users during account creation and booking processes.

● Reuse Objectives

- The system should prioritize the reuse of existing software components and libraries to minimize development effort and ensure consistency in functionality.
- Code reusability and modular design principles should be followed to facilitate future expansion and maintenance.
- Design patterns such as MVC (Model-View-Controller) should be implemented to promote component-based development and reusability.
- Common functionalities such as user authentication, session management, and database access should be encapsulated into reusable components.

Appendix A: Glossary

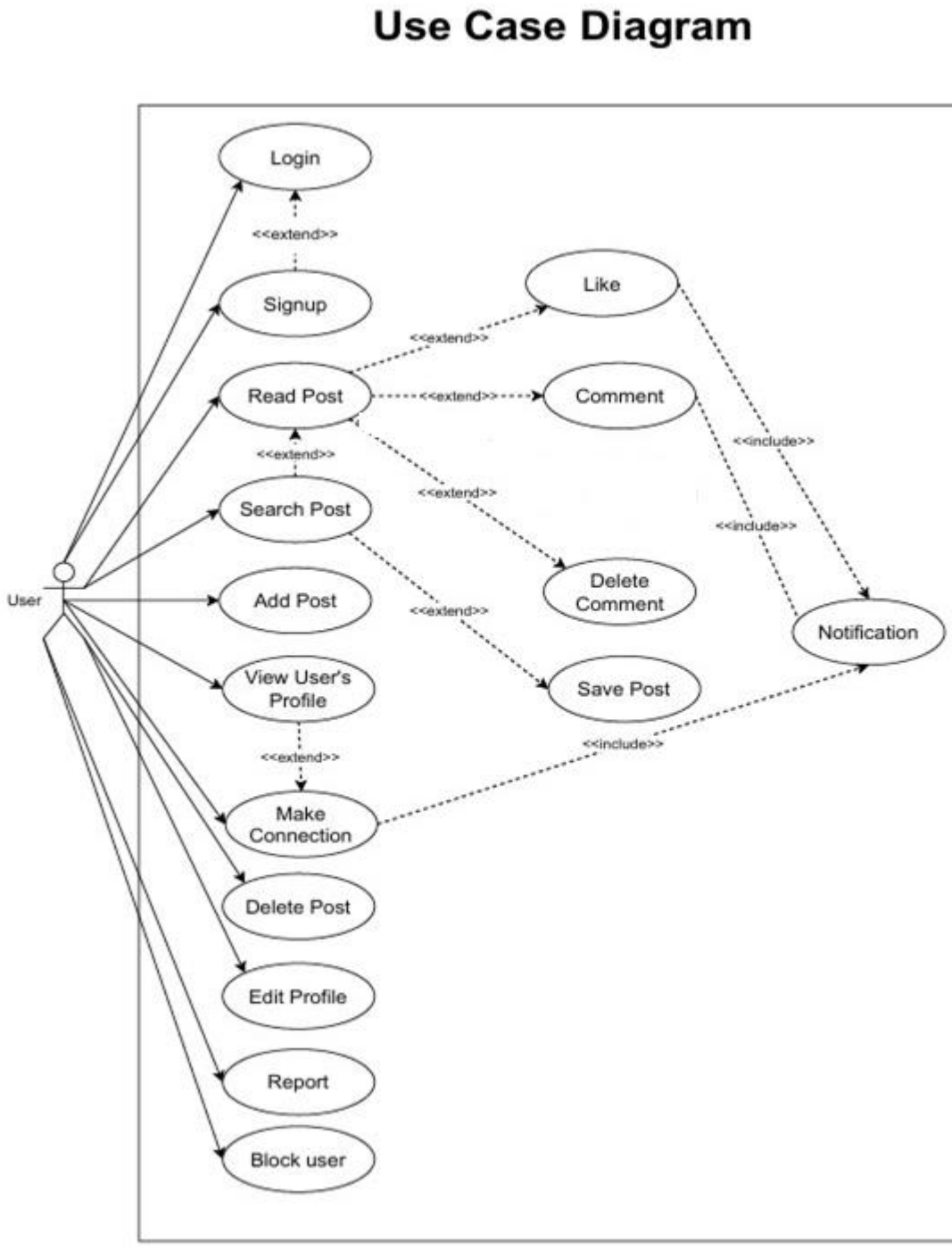
RDBMS - Relational Database Management System

API - Application Programming Interface

GUI - Graphical User Interface

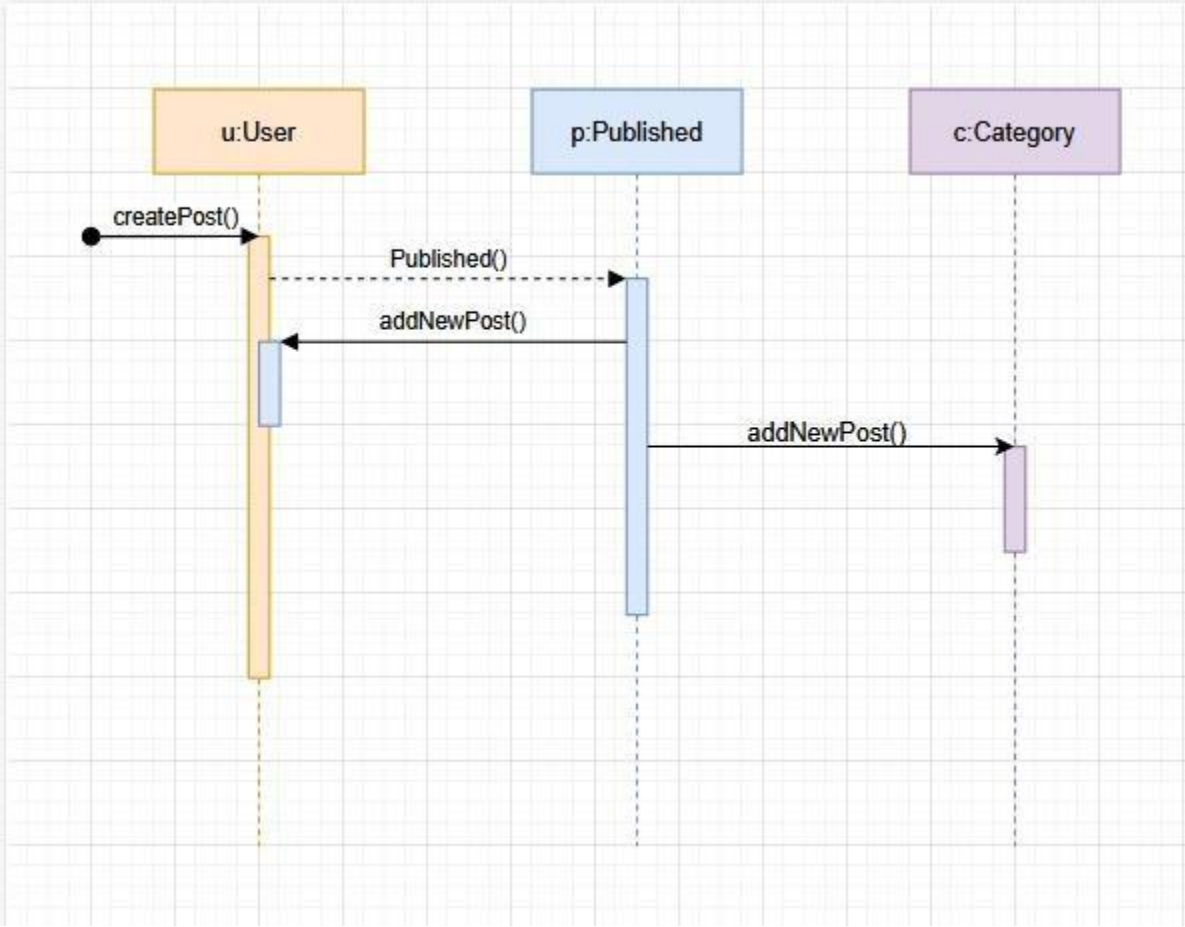
Appendix B: Analysis Models

Use Case Diagram:

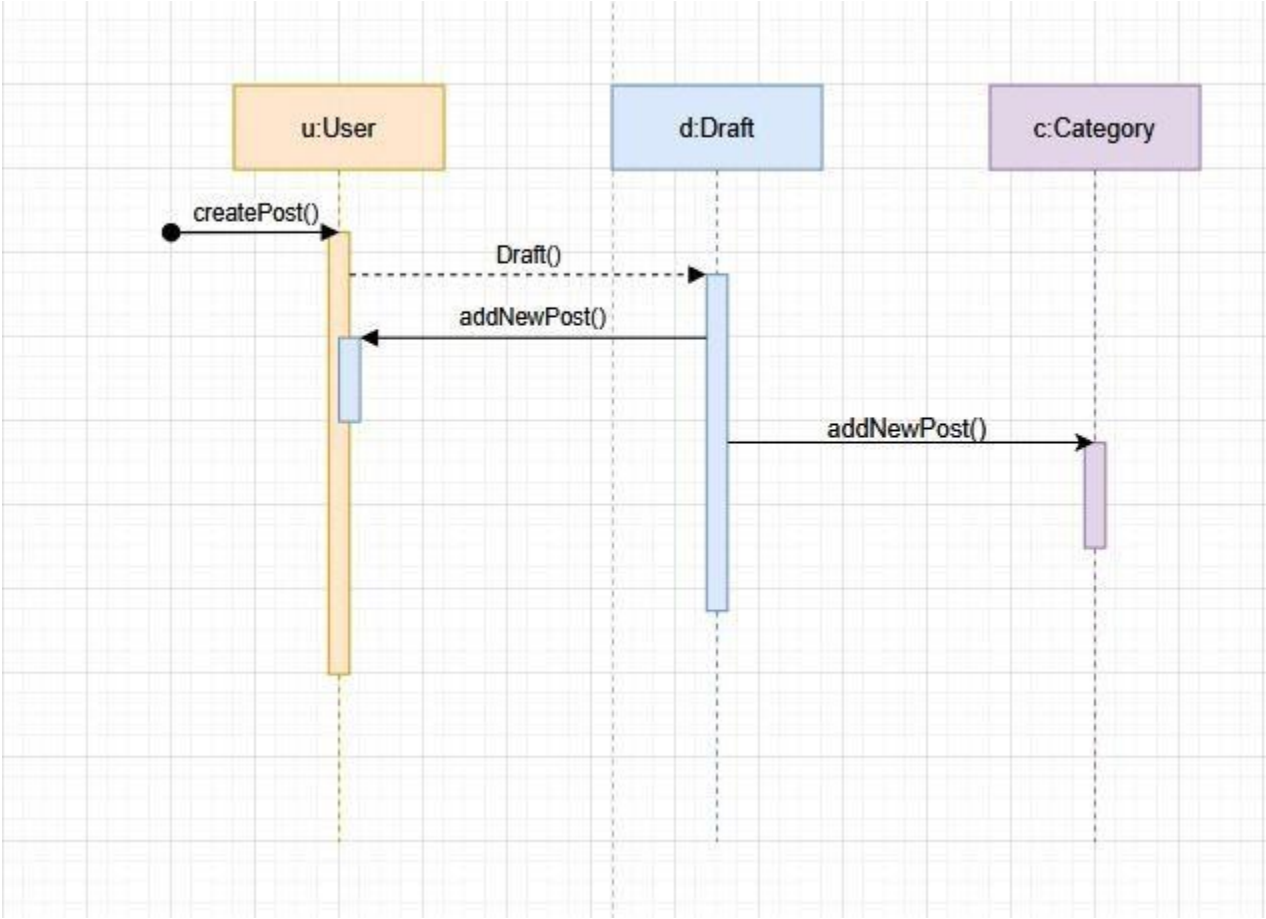


Sequence Diagrams

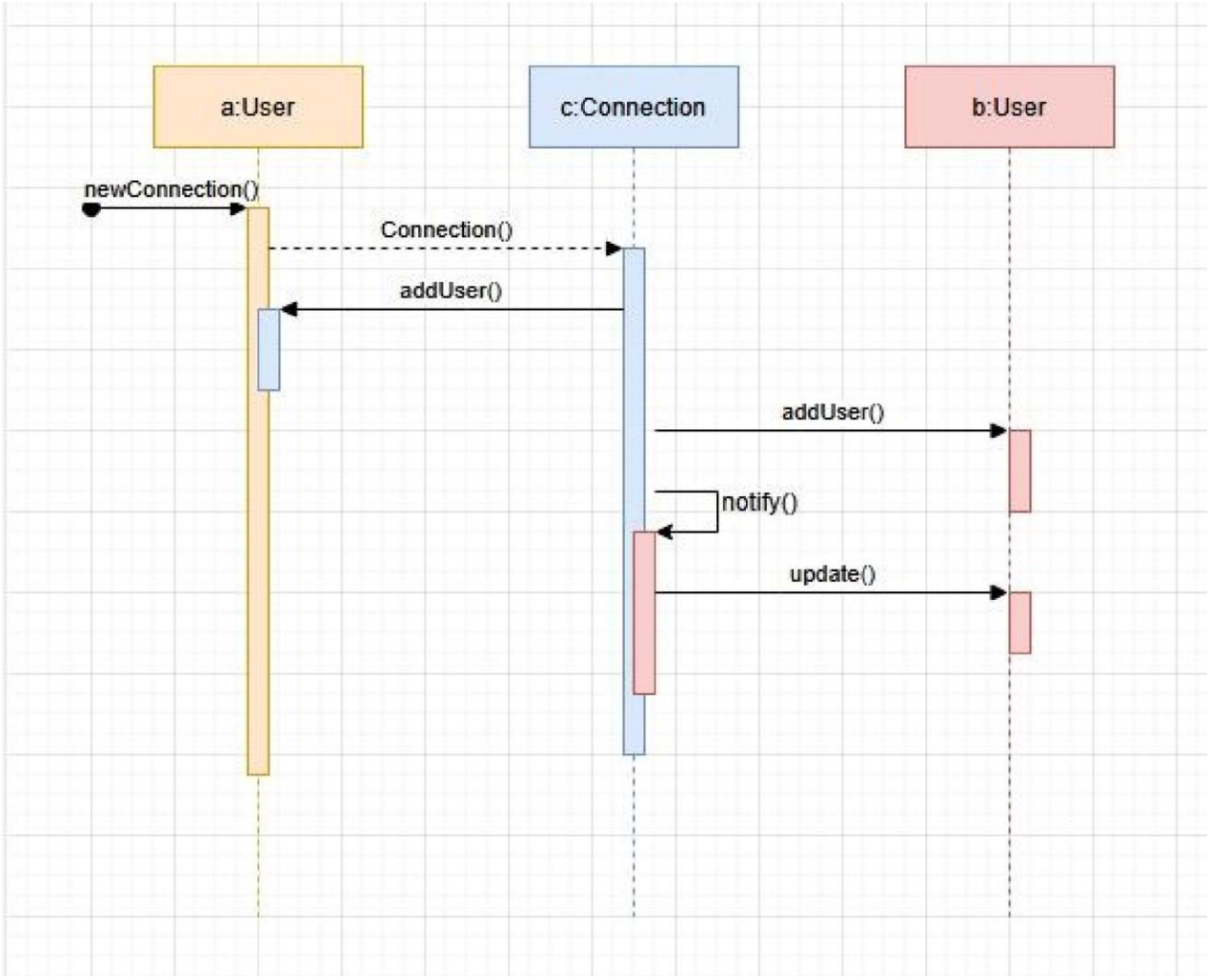
1. Create/Publish Post



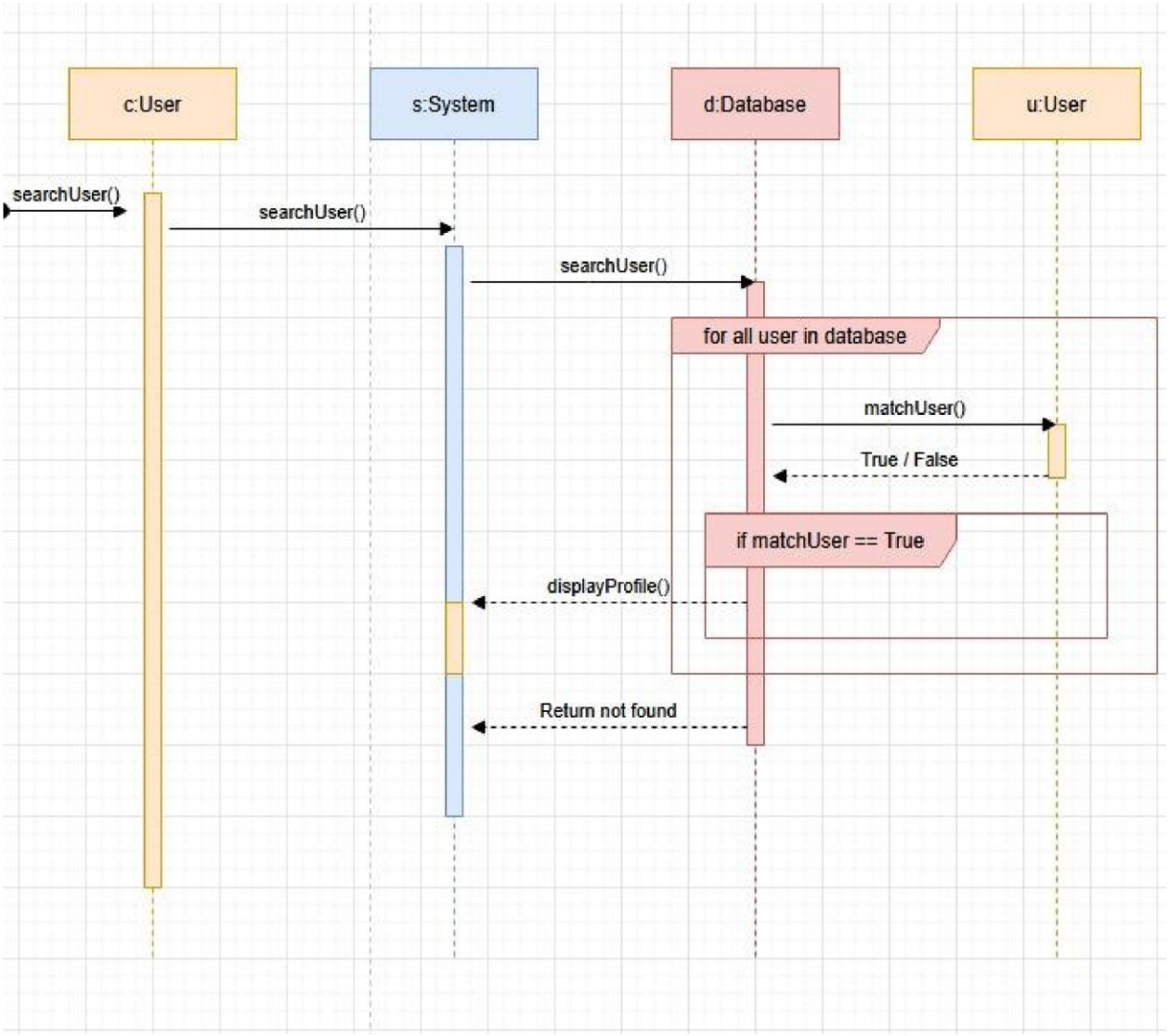
2. Save Post as Draft



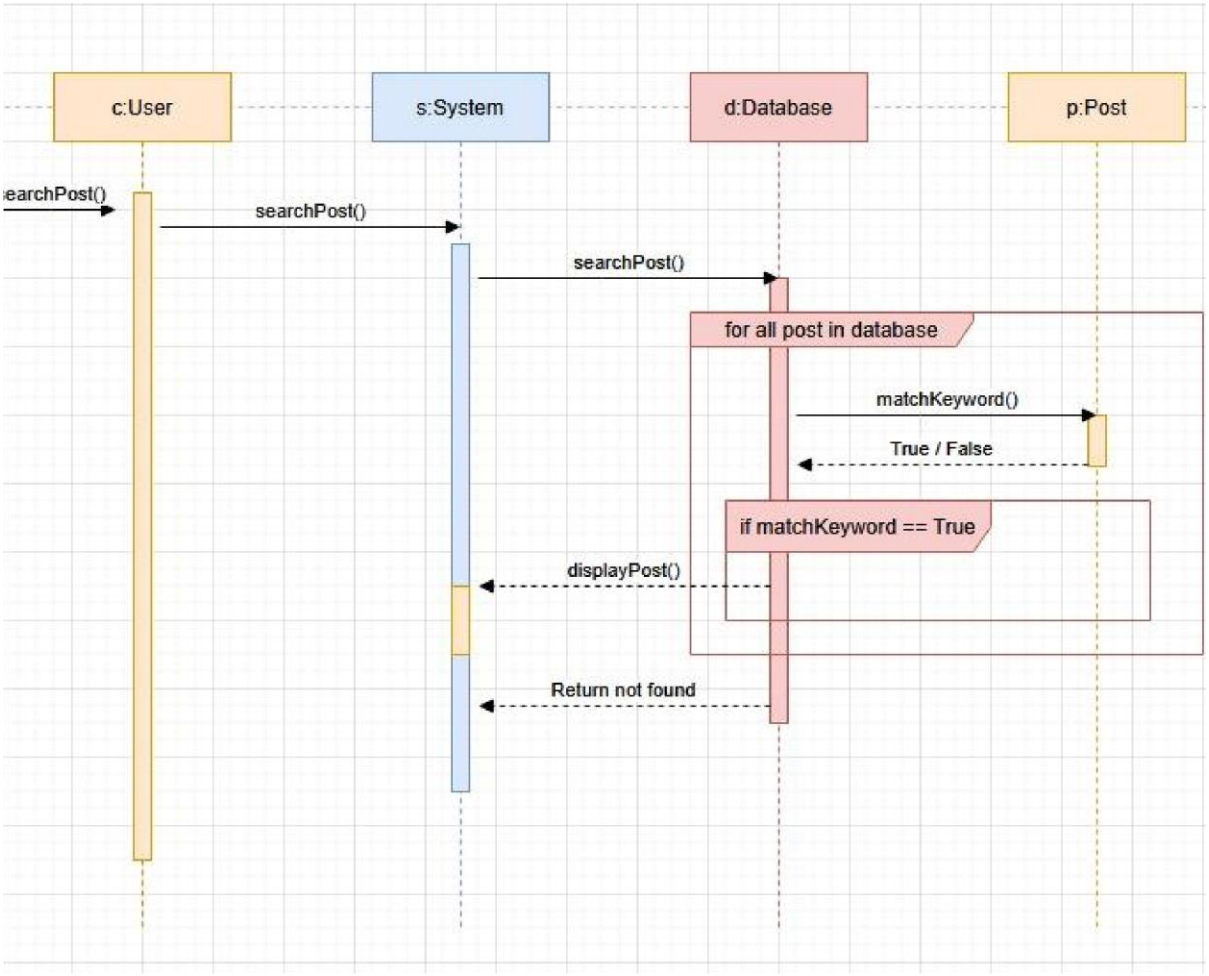
3. Connect with User



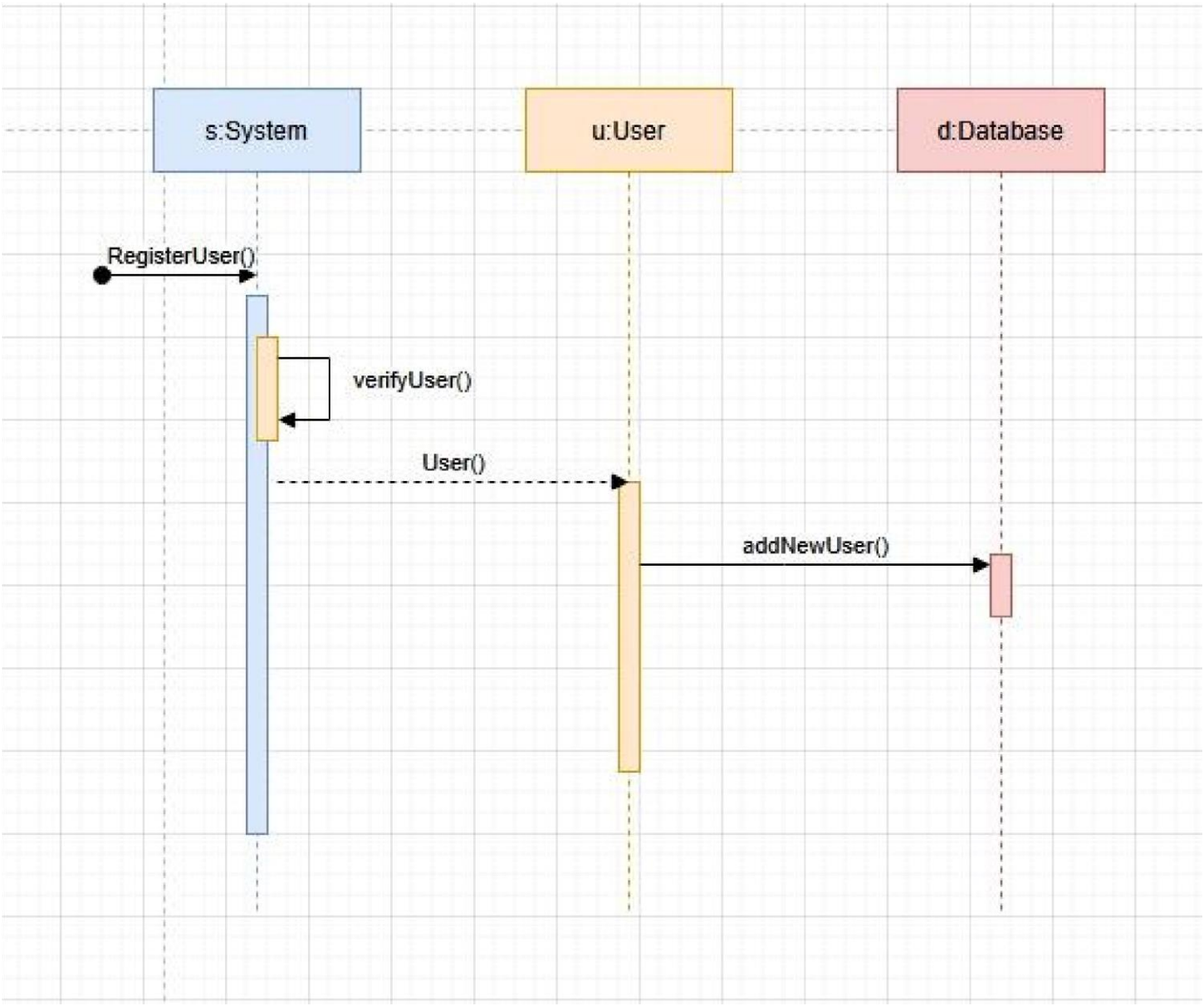
4. Search


c:User
s:System
d:Database
u:User
searchUser()
searchUser()
searchUser()
for all user in database
matchUser()
True / False
if matchUser == True
displayProfile()
Return not found

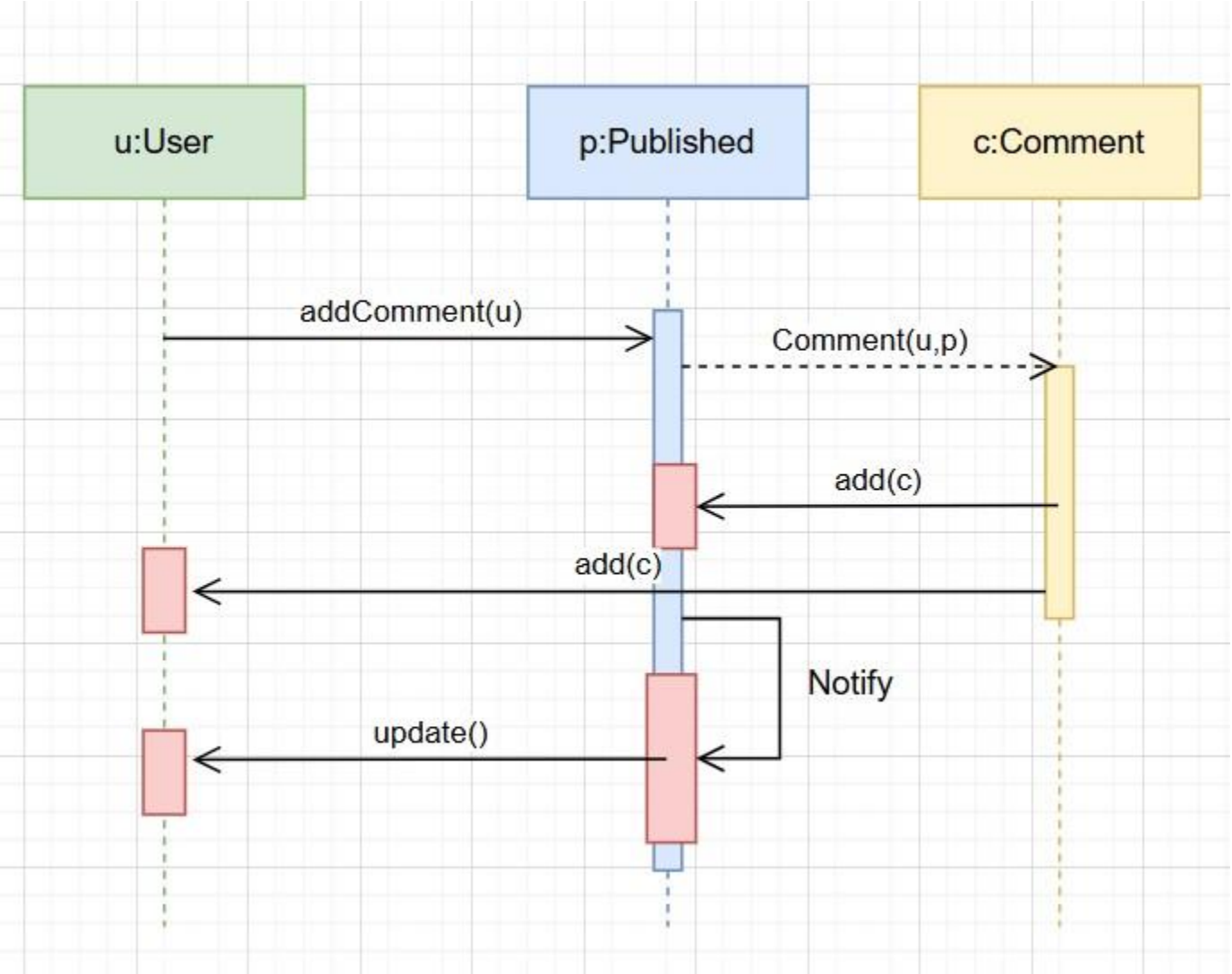
5. Search Post Using Keywords



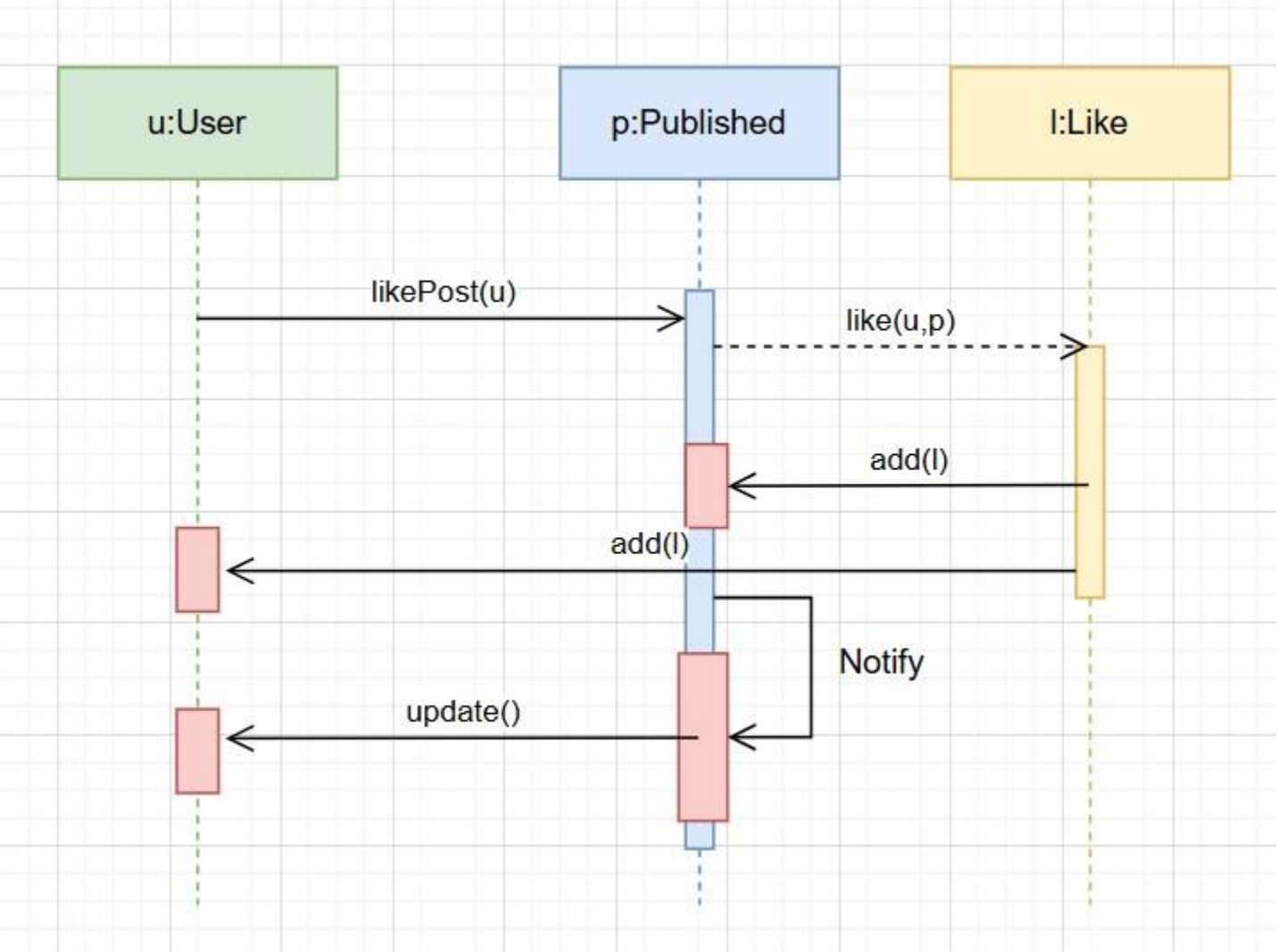
6. Register User



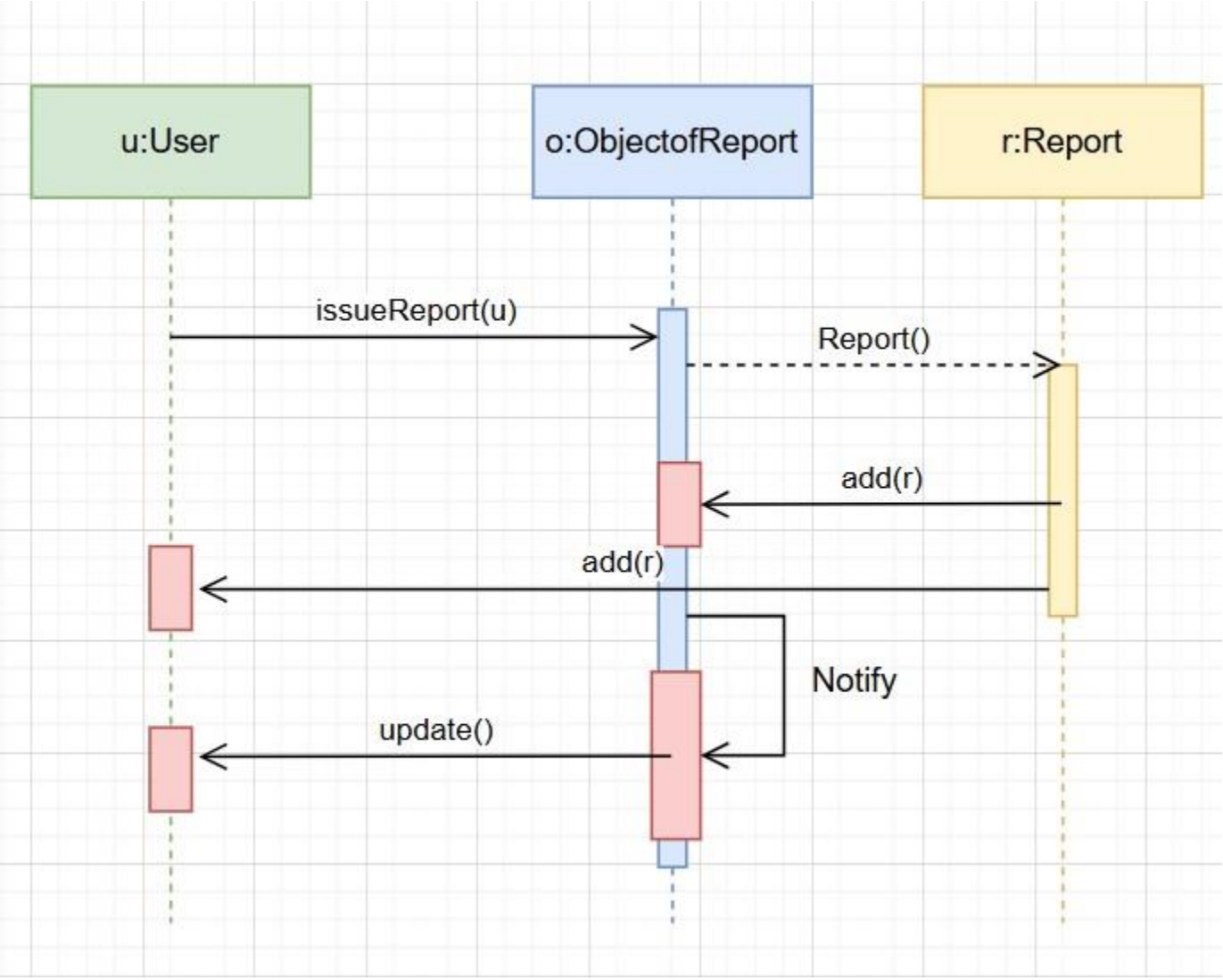
7. Add Comment



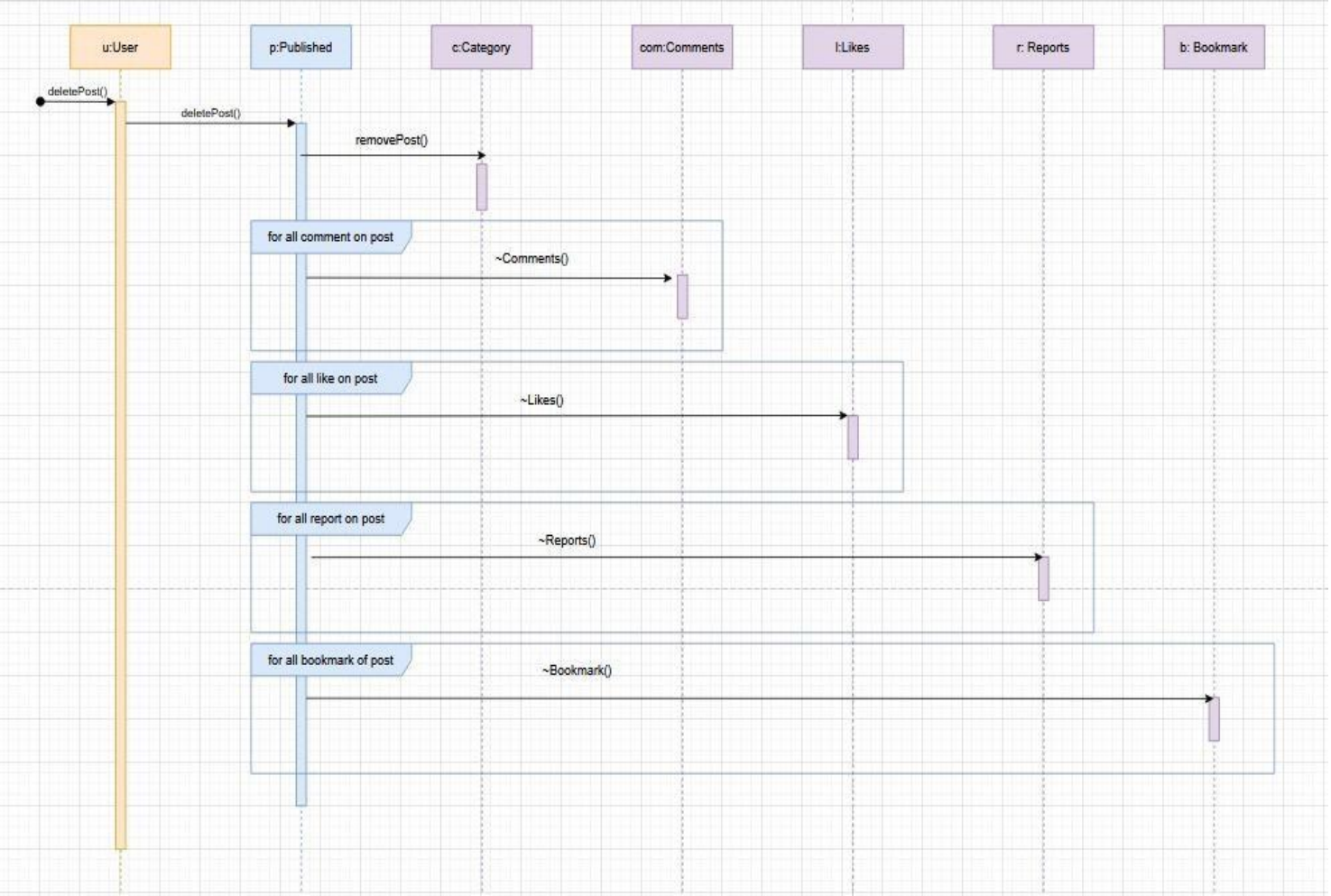
8. Like Post


u:User
p:Published
l:Like
likePost(u)
like(u,p)
add(l)
add(l)
Notify
update()

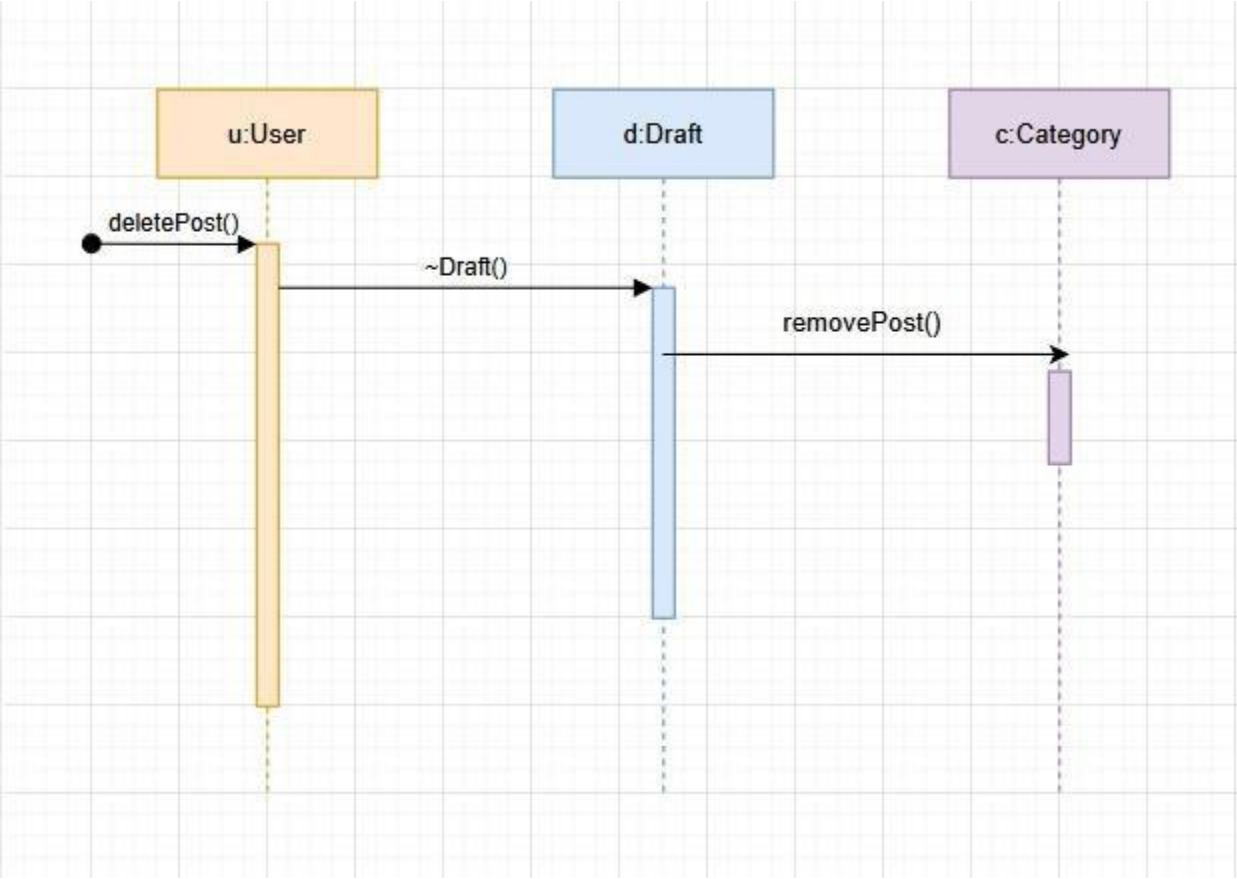
9.Report Issue



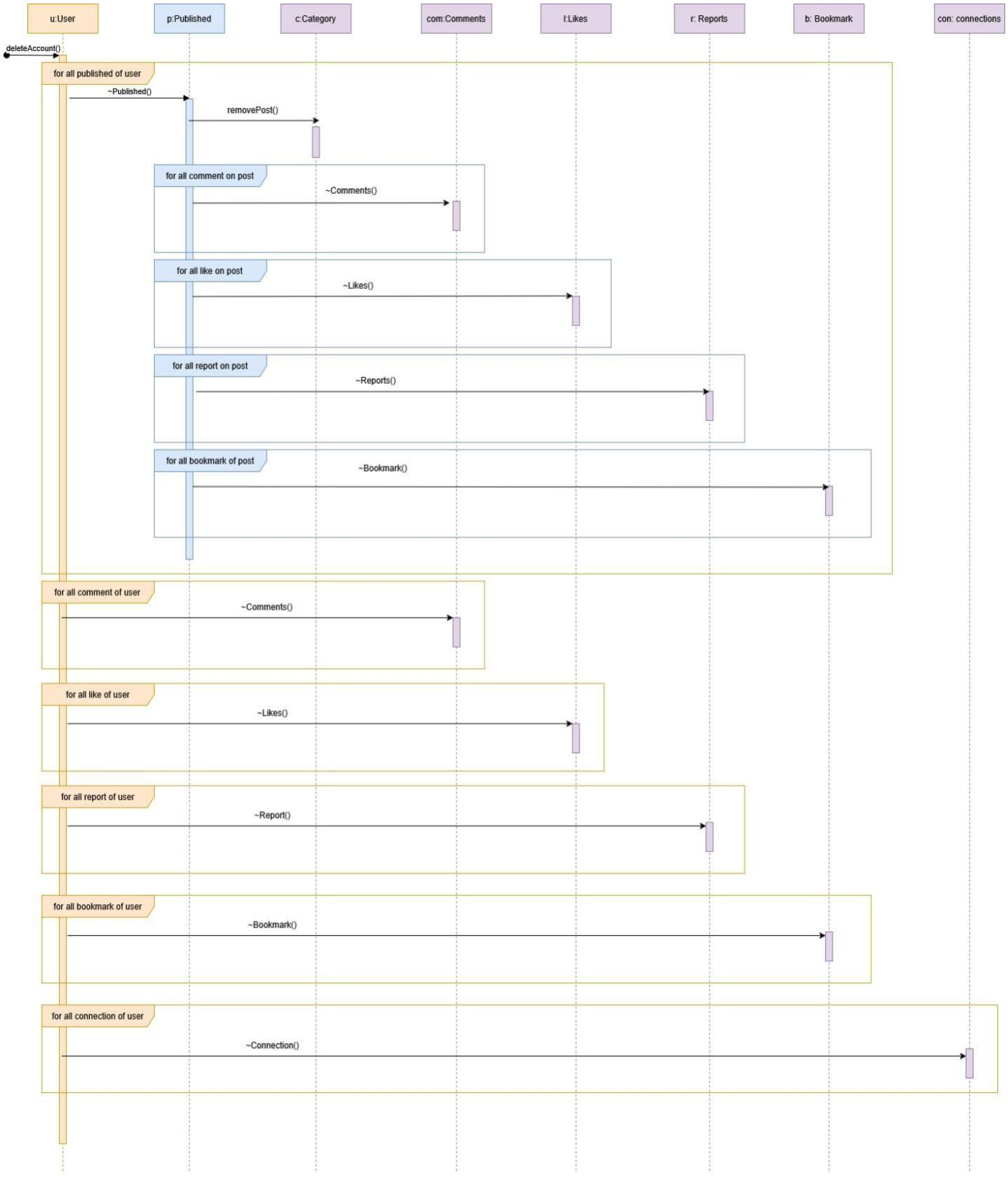
10. Delete Published Posts



11. Delete Saved Drafts

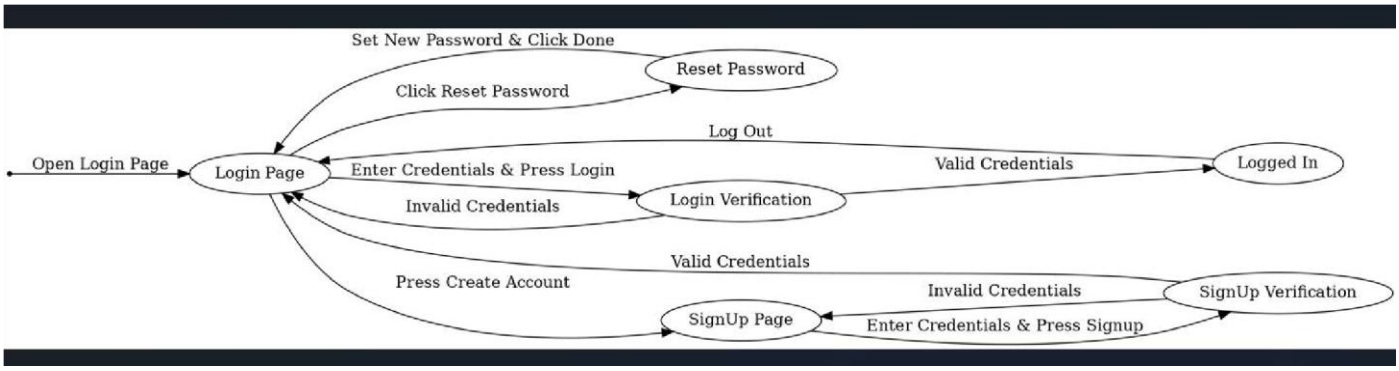


12. Delete User

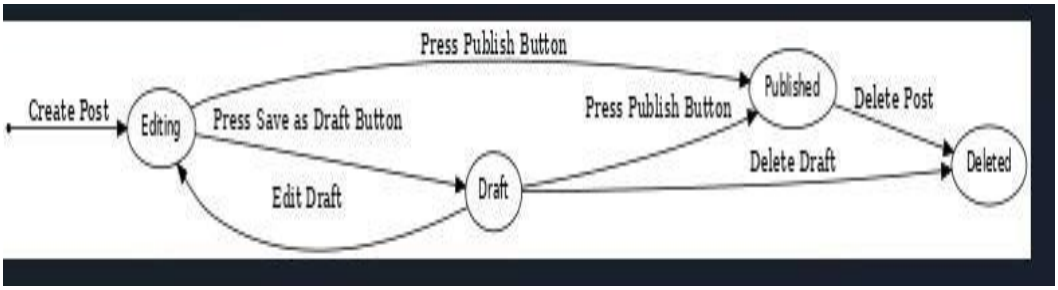


State Diagrams:

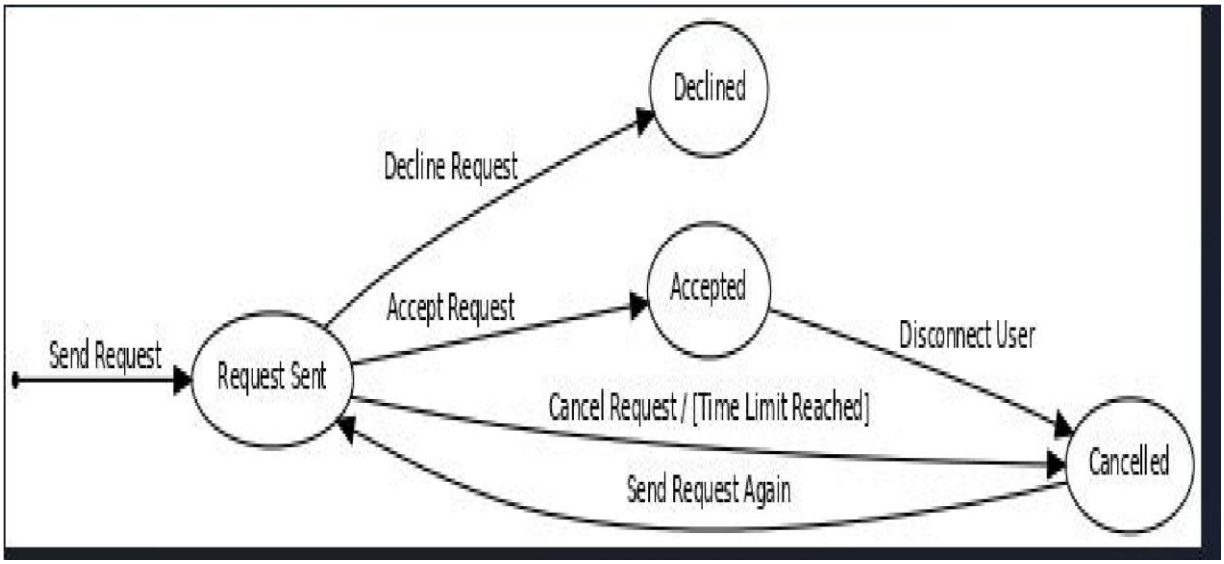
Login SignUp SystemReset Password



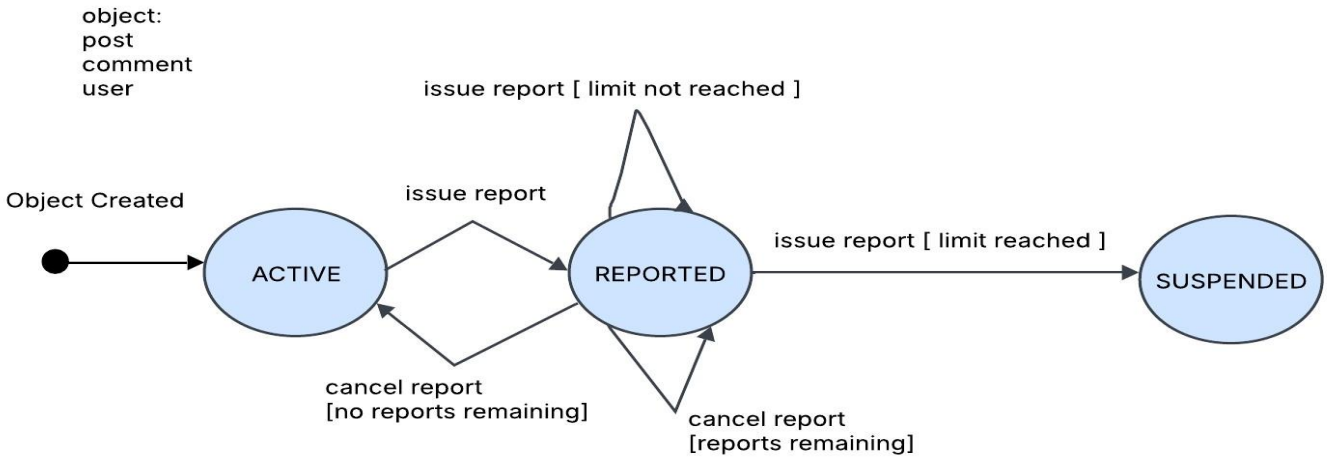
Post Creation:



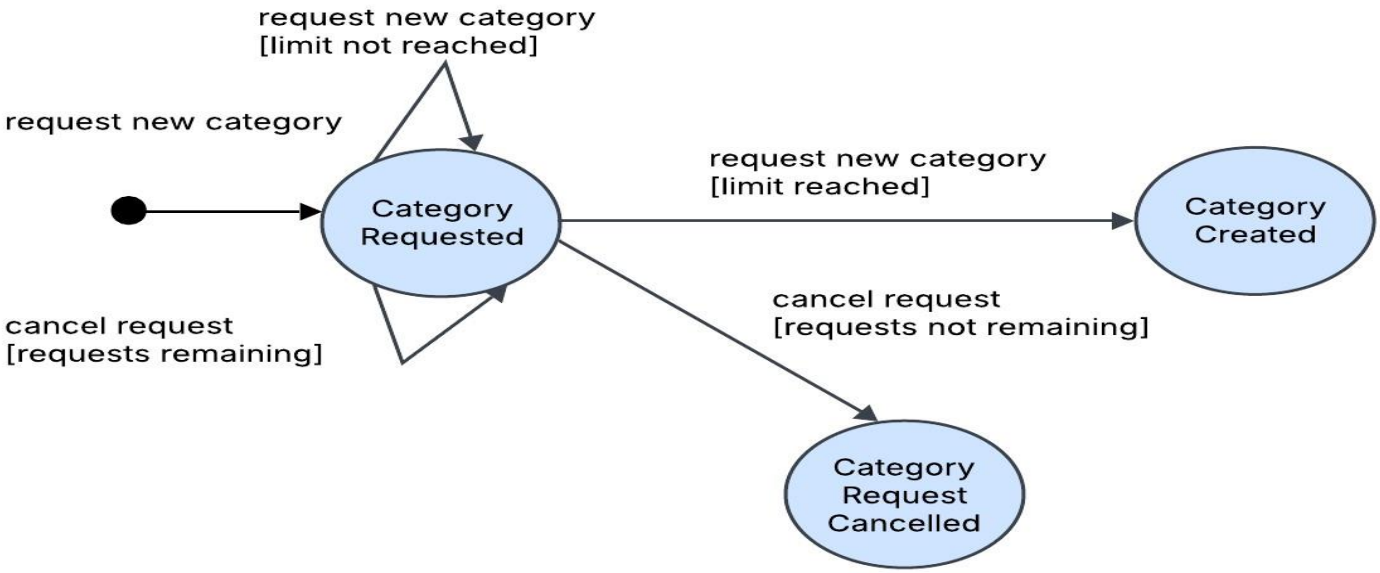
Post:



Report Object:

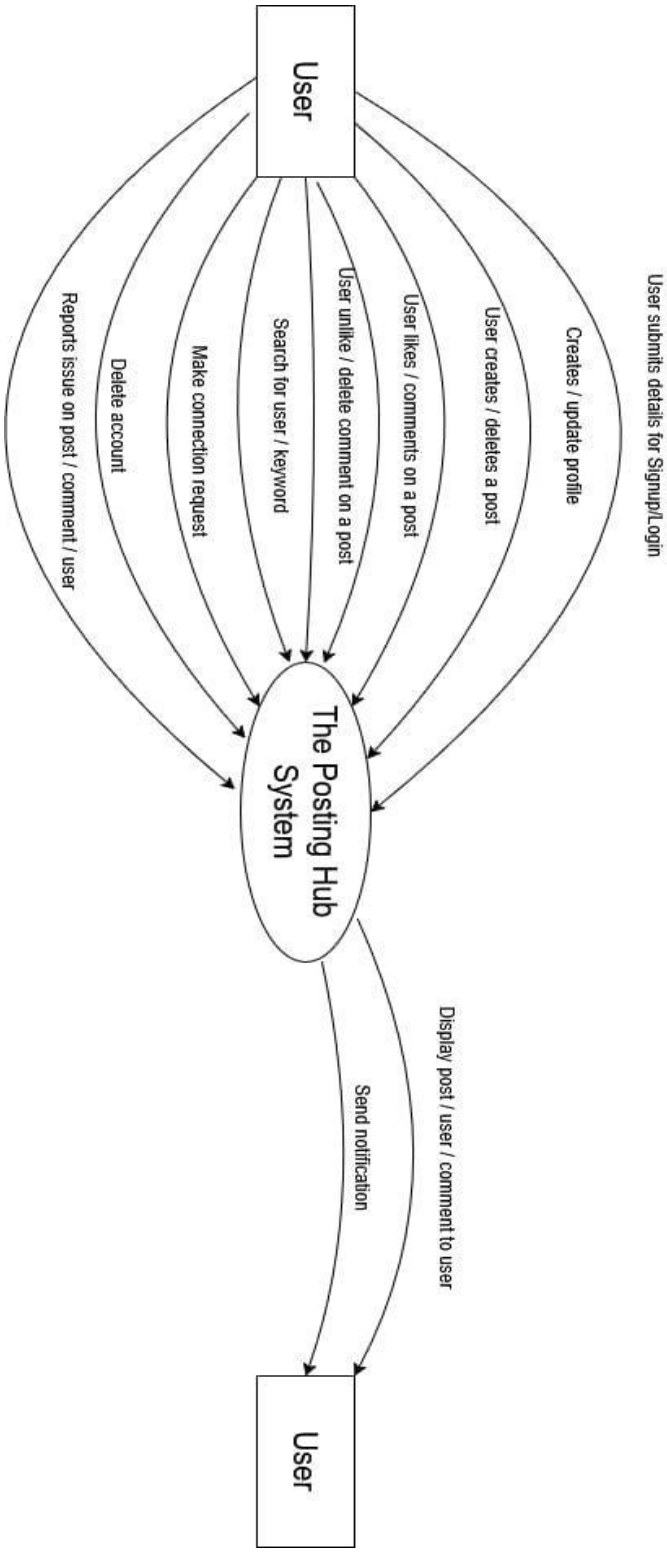


Category Request:

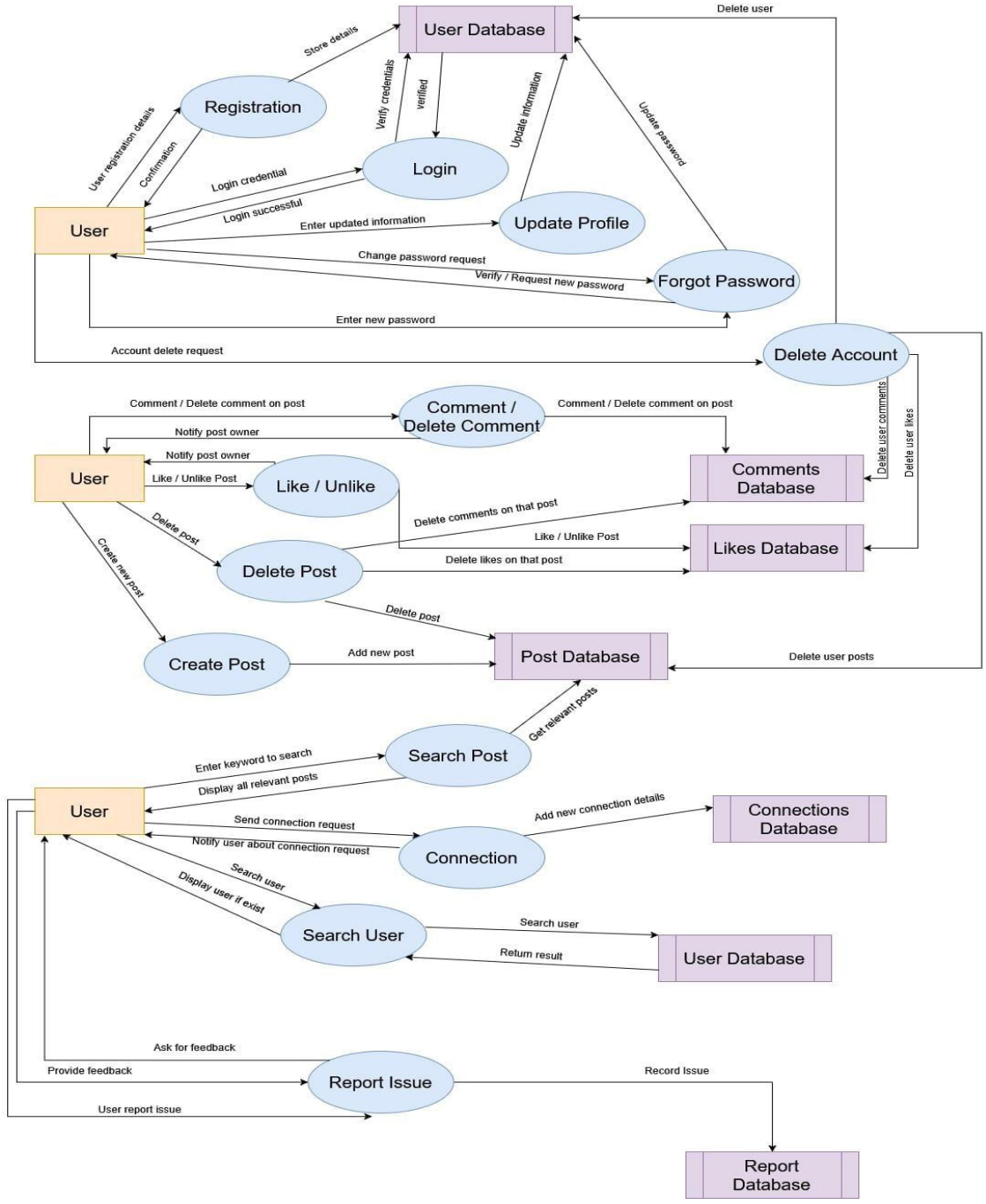


Data Flow Diagrams:

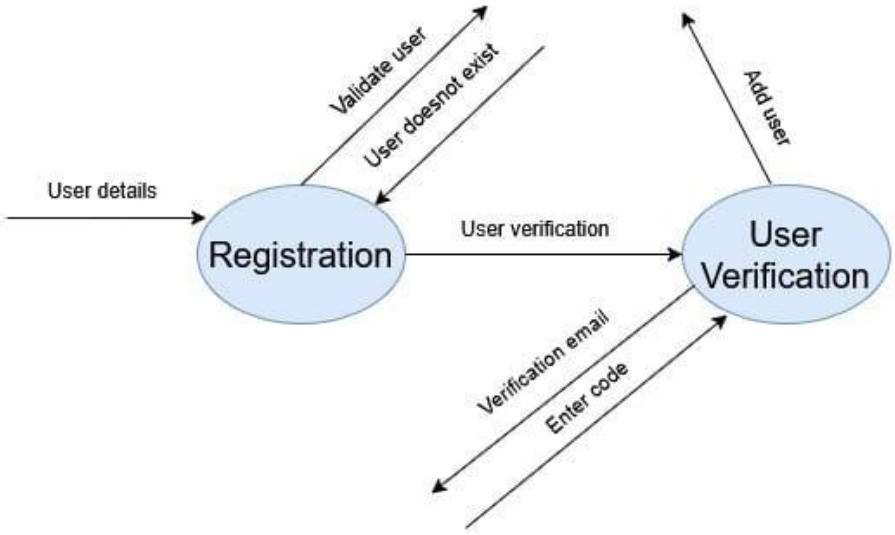
Level 0 Data Flow Diagram:

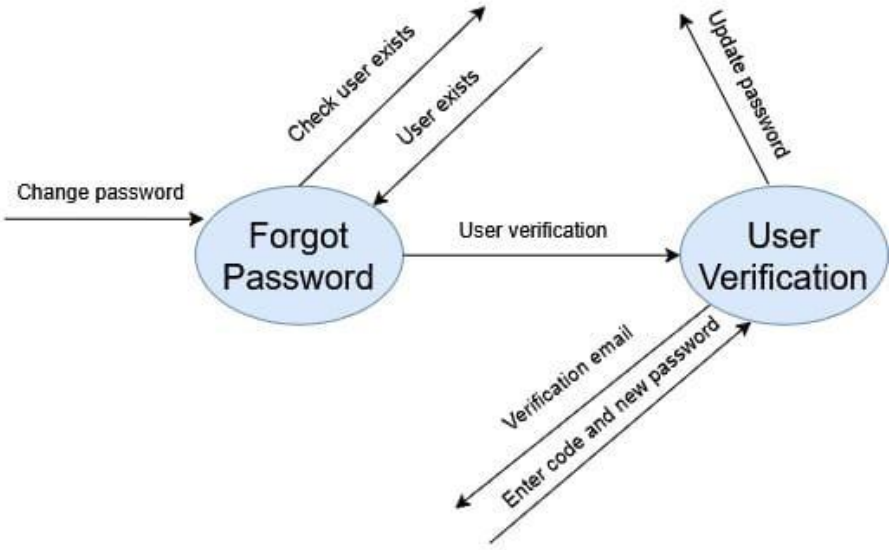


Level 1 Data Flow D



Level 2 Data Flow Diagram:





Class Diagrams

Simple

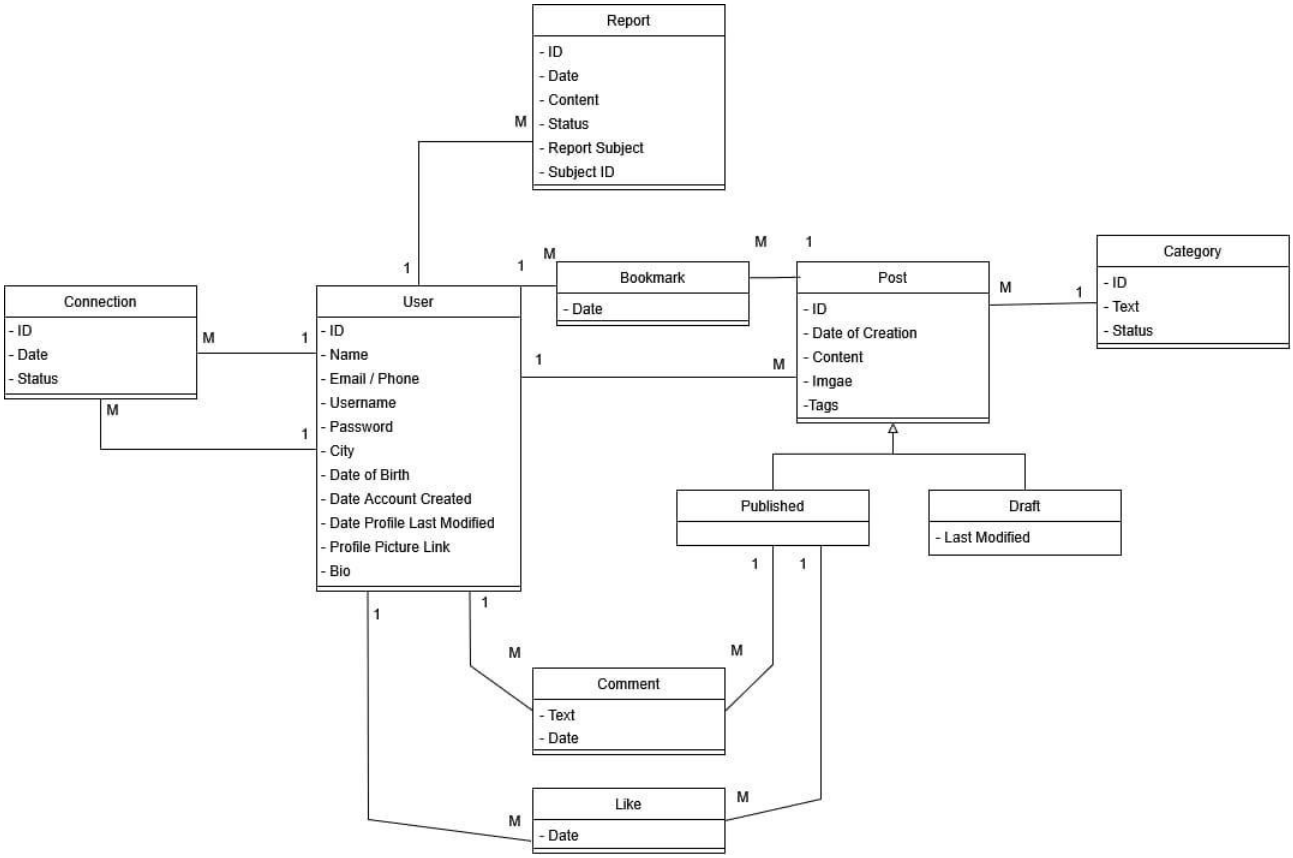
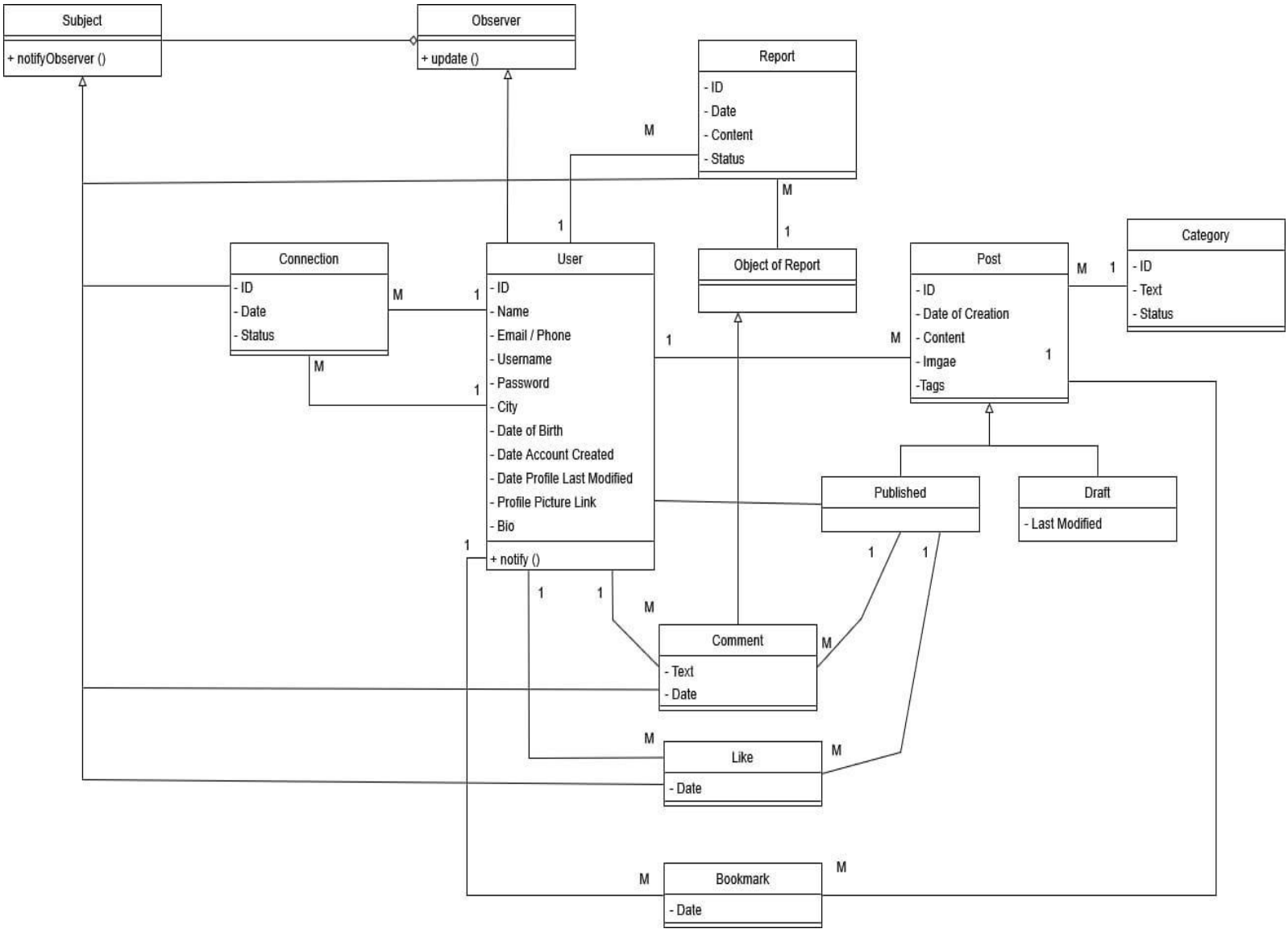


Diagram with Design Patterns]:



Appendix C: To Be Determined List

1. TBD -
2. TBD -
3. TBD -
4. TBD -