

**Project Documentation**

**Project Title:**

Social Media Platform

**Submitted By:**

Muhammad Abdullah Naeem Afzal

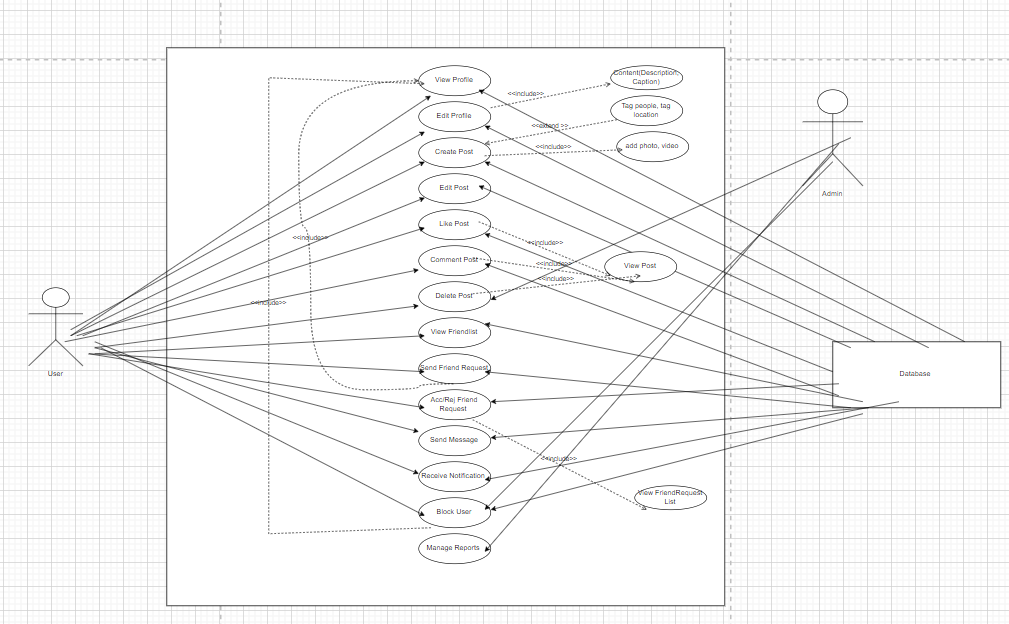
44722

**Submitted To:**

Sir Muhammad Usman Karim

**1.Use Case Diagram**

**Diagram:**

****

**Description:**

**UC-1:**

|  |  |
| --- | --- |
| **Use Case** | View Profile |
| **Actor** | User |
| **Stakeholders** | User, System |
| **Preconditions** | The user must be logged into the system. |
| **Postconditions** | The user's profile is displayed on the screen |
| **Main Flow** | 1. The **User** selects the option to view their profile. 2. The **System** retrieves the profile information from the **Database**. 3. The **System** displays the profile information (e.g., name, bio, profile picture). |
| **Alternate Flow** | If the user does not have a profile (e.g., they haven't set it up yet), the **System** will display a message prompting the user to complete their profile |
| **Exceptions** | If the **Database** fails to retrieve the profile data, an error message is displayed, and the user is asked to try again. |

**UC-2:**

|  |  |
| --- | --- |
| **Use Case** | Edit Profile |
| **Actor** | User |
| **Stakeholders** | User, System |
| **Preconditions** | The user must be logged into the system and must have a profile created. |
| **Postconditions** | The user's profile is updated in the **Database**. |
| **Main Flow** | 1. The **User** selects the option to edit their profile. 2. The **System** retrieves the current profile data from the **Database**. 3. The **System** displays the profile in an editable format (e.g., name, bio, profile picture). 4. The **User** makes changes and submits the updated profile. 5. The **System** validates the changes. 6. The **System** saves the updated profile to the **Database**. 7. The **System** confirms the successful update to the **User**. |
| **Alternate Flow** | If any validation errors occur (e.g., required fields are empty), the **System** prompts the **User** to fix the errors. |
| **Exceptions** | If the **Database** cannot save the changes (e.g., due to a connection error), an error message is displayed, and the user is asked to try again later. |

**UC-3:**

|  |  |
| --- | --- |
| **Use Case** | Create Post |
| **Actor** | User |
| **Stakeholders** | User, System |
| **Preconditions** | The user must be logged in and on the post creation page. |
| **Postconditions** | The **Post** is saved in the **Database** and displayed on the user's profile and feed. |
| **Main Flow** | 1. The **User** selects the option to create a post. 2. The **System** displays a post creation interface (e.g., text input, file upload). 3. The **User** enters content (e.g., text, image) for the post. 4. The **User** submits the post. 5. The **System** saves the post content to the **Database**. 6. The **System** displays the post in the user’s feed and profile. |
| **Alternate Flow** | If the post contains invalid content (e.g., prohibited language or unsupported media type), the **System** prompts the user to correct it. |
| **Exceptions** | If there is a **Database** error while saving the post, an error message is displayed, and the user is asked to try again. |

**UC-4:**

|  |  |
| --- | --- |
| **Use Case** | Edit Post |
| **Actor** | User |
| **Stakeholders** | User System |
| **Preconditions** | * The user must be logged into the system. * The user must have created the post they wish to edit (i.e., the post already exists). * The post must not be deleted or flagged as inappropriate. |
| **Postconditions** | * The post content is updated in the database and displayed with the changes to other users. * The **Updated** timestamp for the post is refreshed. |
| **Main Flow** | 1. **User selects a post to edit**: The regular user navigates to their profile or feed and chooses the post they want to modify. 2. **System displays the post in an editable format**: The system retrieves the post's content (text, images, etc.) and presents it in an editable view. The user can modify the content. 3. **User makes changes to the post**: The regular user updates the post, such as editing text, changing the image, or modifying any other content associated with the post. 4. **User submits the updated post**: Once the user is satisfied with the changes, they click on the "Save" or "Submit" button. 5. **System validates the changes**: The system checks if the changes are valid (e.g., no empty posts, no prohibited content, etc.). If the validation fails, the system prompts the user to correct any issues. 6. **System saves the updated post to the database**: After successful validation, the system saves the updated post content to the database, updating the original post with the new content. 7. **System confirms the successful update to the user**: The system displays a message confirming that the post has been successfully updated. The updated post is now visible to other users with the new content. |
| **Alternate Flow** | 1. **If the post contains invalid content** (e.g., inappropriate language, unsupported file type for images, or empty content):    * The system displays an error message and asks the user to correct the content before resubmitting. 2. **If the user cancels the edit**:    * The system returns the user to the post viewing screen without saving any changes, and no update is made to the post. |
| **Exceptions** | 1. **Database failure**:    * If there is a database error while saving the post, an error message is displayed, and the user is asked to try again later. 2. **Permission issues**:    * If the user attempts to edit a post they do not own (e.g., trying to edit a post made by another user), the system displays an error indicating they don't have permission to edit the post. 3. **Session Timeout**:    * If the user's session expires while they are editing the post, they are redirected to the login page with a message that their session has expired. The unsaved changes are lost. |

**UC-5:**

|  |  |
| --- | --- |
| **Use Case** | Delete Post |
| **Actor** | User, Admin |
| **Stakeholders** | User, Admin, System |
| **Preconditions** | * The **Admin** must be logged in and have sufficient privileges to delete posts. * The **User** should be logged in and should have posted a post which he desires to delete. |
| **Postconditions** | The post is permanently deleted from the **Database**. |
| **Main Flow** | 1. The **Admin/User** views the post they want to delete. 2. The **System** confirms that the user has the required privileges to delete the post. 3. The **Admin User** selects the option to delete the post. 4. The **System** deletes the post from the **Database**. 5. The **System** confirms that the post has been deleted. |
| **Alternate Flow** | If the post has already been deleted or does not exist, the **System** displays a message stating that the post could not be found. |
| **Exceptions** | If the **Database** fails to delete the post, an error message is displayed to the **Admin/ User**, and the deletion process is aborted. |

**UC-6:**

|  |  |
| --- | --- |
| **Use Case** | Comment on Post |
| **Actor** | User |
| **Stakeholders** | User, System |
| **Preconditions** | * The **User** must be logged in. * The **User** is viewing a post that allows comments. |
| **Postconditions** | * The comment is saved to the **Database**. * The comment is displayed under the post. |
| **Main Flow** | 1. The **User** views a post and selects the option to comment. 2. The **System** displays the comment input field. 3. The **User** enters their comment and submits it. 4. The **System** validates the comment (e.g., checks for empty text or inappropriate content). 5. If valid, the **System** saves the comment to the **Database** and displays it under the post. 6. The **System** updates the post’s comment count. |
| **Alternate Flow** | * **Empty Comment**: If the comment is empty, the **System** prompts the **User** to enter text. * **Inappropriate Content**: If the comment contains prohibited content, the **System** displays a warning and does not post the comment. |
| **Exceptions** | * **Database Error**: If saving the comment fails, the **System** displays an error message and asks the **User** to try again. * **Network Error**: If there’s a network issue, the **System** asks the **User** to retry later. |

**UC-7:**

|  |  |
| --- | --- |
| **Use Case** | Like Post |
| **Actor** | User |
| **Stakeholders** | User, System |
| **Preconditions** | The user must be logged in and viewing a post. |
| **Postconditions** | The post's like count is updated in the **Database**. |
| **Main Flow** | 1. The **User** selects the like button on a post. 2. The **System** updates the like count for the post in the **Database**. 3. The **System** displays the updated like count on the post. |
| **Alternate Flow** | If the user has already liked the post, the like is toggled off, and the like count is updated accordingly. |
| **Exceptions** | If there is an error when updating the like count in the **Database**, an error message is displayed. |

**UC-8:**

|  |  |
| --- | --- |
| **Use Case** | Send Friend Request |
| **Actor** | User |
| **Stakeholders** | User, System |
| **Preconditions** | The user must be logged in and viewing another user’s profile. |
| **Postconditions** | A **FriendRequest** is saved in the **Database** with a "Pending" status. |
| **Main Flow** | 1. The **User** selects the option to send a friend request. 2. The **System** creates a **FriendRequest** and saves it in the **Database**. 3. The **System** notifies the recipient (via notification) that a friend request has been sent. |
| **Alternate Flow** | If the recipient has already been sent a friend request or the users are already friends, the system will display a message indicating the request cannot be sent. |
| **Exceptions** | If there’s a **Database** issue, an error message is displayed, and the user is prompted to try again. |

**UC-9:**

|  |  |
| --- | --- |
| **Use Case** | Accept/Reject Friend Request |
| **Actor** | User |
| **Stakeholders** | User, System |
| **Preconditions** | The user must be logged in and have received a friend request. |
| **Postconditions** | The **FriendRequest** status is updated to "Accepted" or "Rejected" in the **Database**. |
| **Main Flow** | 1. The **User** views the received friend request. 2. The **System** presents options to accept or reject the request. 3. The **User** selects an option (accept or reject). 4. The **System** updates the **FriendRequest** status in the **Database** accordingly. 5. The **System** sends a notification to the other user regarding the acceptance or rejection. |
| **Alternate Flow** | If the user changes their mind or wants to reconsider, they can cancel the action before confirming. |
| **Exceptions** | If the **Database** fails to update the friend request status, an error message is displayed, and the user is prompted to try again. |

**UC-10:**

|  |  |
| --- | --- |
| **Use Case** | Block User |
| **Actor** | User, Admin |
| **Stakeholders** | User, Admin, System |
| **Preconditions** | The **Admin/User** must be logged in and **Admin** must have the appropriate permissions to block users. |
| **Postconditions** | The blocked user is restricted from interacting with the platform or contacting the **User**. |
| **Main Flow** | 1. The **Admin/User** selects the user they wish to block. 2. The **System** verifies the **Admin/User**’s authorization to block the user. 3. The **Admin/User** selects the option to block the user. 4. The **System** updates the user’s status in the **Database** to "Blocked". 5. The **System** notifies the **Admin/User** and the **Blocked User** about the action. |
| **Alternate Flow** | If the **Admin/User** attempts to block their own account (by mistake), the **System** will display a warning and prevent the action. |
| **Exceptions** | If the **Database** fails to update the user status, an error message is displayed, and the user is not blocked. |

**UC-11:**

|  |  |
| --- | --- |
| **Use Case** | Manage Reports |
| **Actor** | Admin |
| **Stakeholders** | Admin, System |
| **Preconditions** | * The **Admin** must be logged in and have permissions to manage reports. * There must be content that has been reported by users. |
| **Postconditions** | The **System** updates the status of the reported content, which may involve taking action such as removing or moderating content. |
| **Main Flow** | 1. The **Admin** views the list of reported content. 2. The **System** displays each report with context (e.g., the post, comment, user who reported). 3. The **Admin** reviews the reported content and decides on an action (e.g., delete the content, issue a warning). 4. The **System** executes the action selected by the **Admin** (e.g., removes content, sends a warning message). 5. The **System** logs the action taken and notifies the **User** (if necessary). |
| **Alternate Flow** | If the **Admin User** determines that the content is not in violation, they may dismiss the report. |
| **Exceptions** | If there is a **Database** issue when processing the report or deleting content, the **System** displays an error message and prompts the **Admin** to try again. |

**UC-12:**

|  |  |
| --- | --- |
| **Use Case** | View Friend List |
| **Actor** | User |
| **Stakeholders** | User, System |
| **Preconditions** | * The **User** must be logged into the system. * The **User** must have at least one friend in the system. |
| **Postconditions** | The **System** displays the **User's** friend list. |
| **Main Flow** | 1. The **User** selects the option to view their friend list from their profile or the main menu. 2. The **System** retrieves the **User's** friend list from the **Database**. 3. The **System** displays the list of friends, showing each friend's name and profile picture. 4. The **User** can scroll through the list and view more details if needed (e.g., profile page). |
| **Alternate Flow** | **No Friends**: If the **User** has no friends, the **System** displays a message indicating that the **User** has no friends yet and suggests sending friend requests. |
| **Exceptions** | **Database Error**: If there’s an issue retrieving the friend list from the **Database**, the **System** displays an error message and prompts the **User** to try again later. |

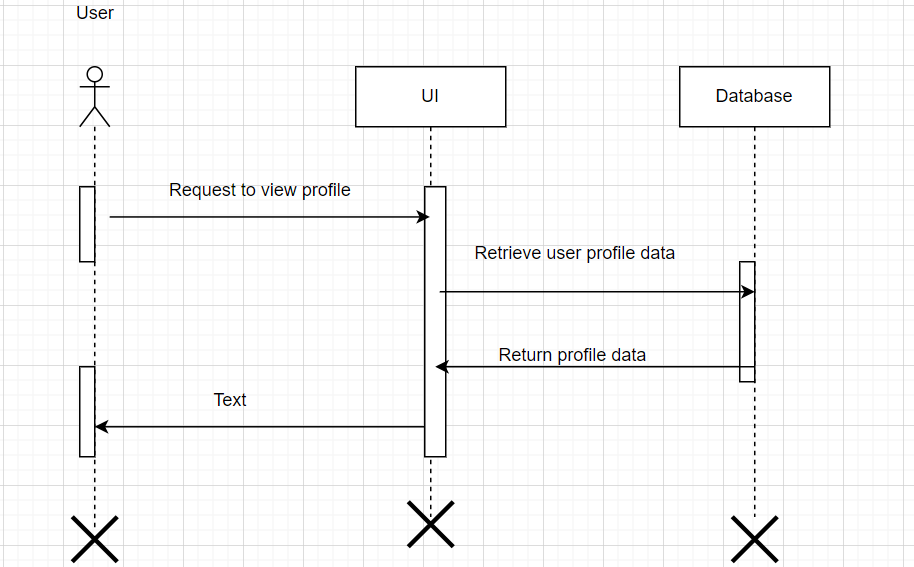
**UC-13:**

|  |  |
| --- | --- |
| **Use Case** | Receive Notification |
| **Actor** | User |
| **Stakeholders** | User, System |
| **Preconditions** | The user must be logged in and have events triggering notifications. |
| **Postconditions** | The **Notification** is displayed to the user. |
| **Main Flow** | 1. The **System** detects an event that triggers a notification (e.g., friend request, like). 2. The **System** creates a notification in the **Database**. 3. The **System** displays the notification to the **User**. |
| **Alternate Flow** | If there are multiple notifications, the **System** can display them in a list. |
| **Exceptions** | None. |

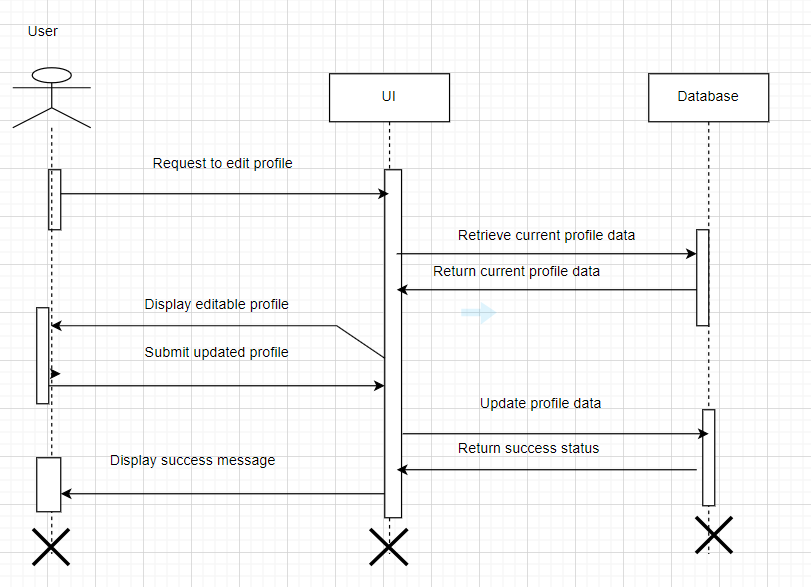
**UC-14:**

|  |  |
| --- | --- |
| **Use Case** | Send Message |
| **Actor** | User |
| **Stakeholders** | User, System |
| **Preconditions** | The user must be logged in and viewing the recipient's profile or message interface. |
| **Postconditions** | The **Message** is saved in the **Database** and sent to the recipient. |
| **Main Flow** | 1. The **User** selects the option to send a message. 2. The **System** opens the message interface. 3. The **User** enters the message content. 4. The **System** sends the message to the **Database**. 5. The **System** delivers the message to the recipient and updates the conversation history. |
| **Alternate Flow** | If the message contains inappropriate content (e.g., profanity), the **System** will display a warning and prevent submission. |
| **Exceptions** | If there’s a **Database** failure while sending the message, an error message is displayed. |

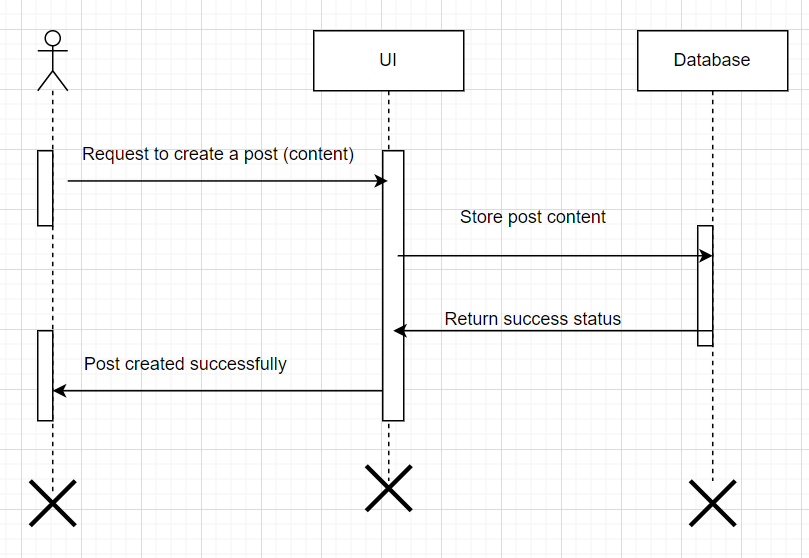
**2.Sequence Diagrams**

**SD-1: (View Profile)**

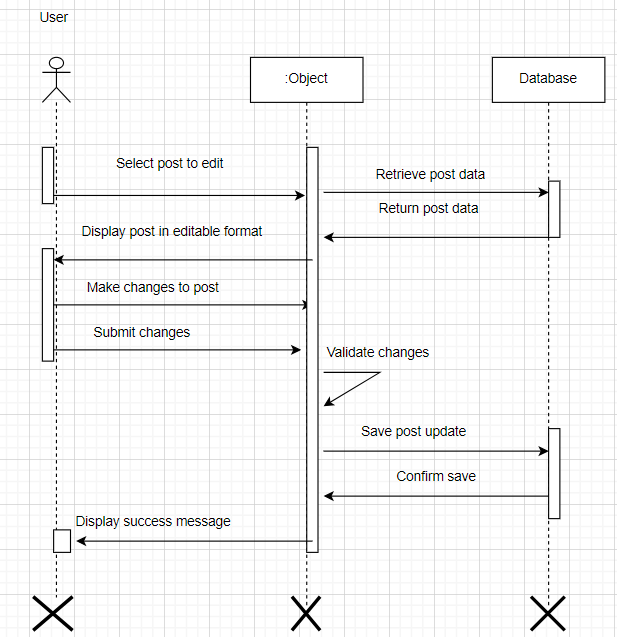
**SD-2: (Edit Profile)**

****

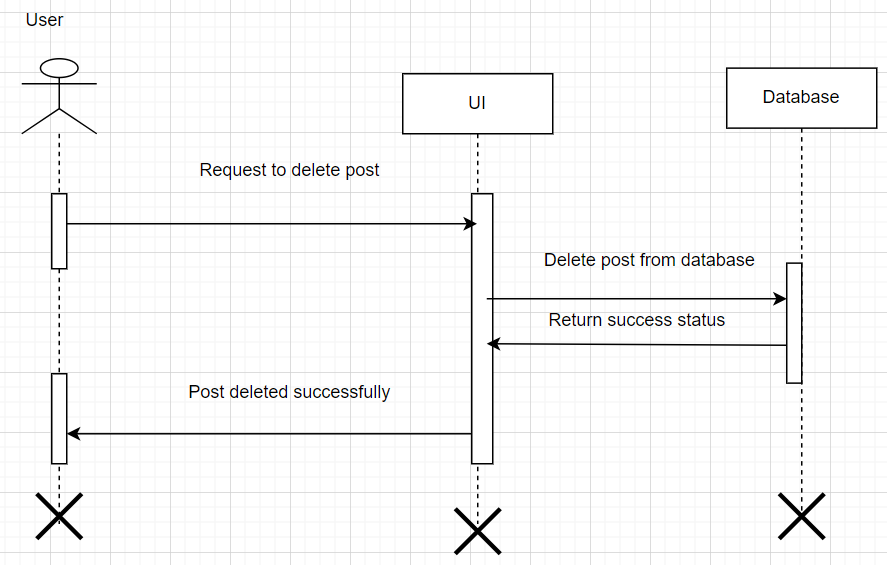
**SD-3: (Create Post)**

****

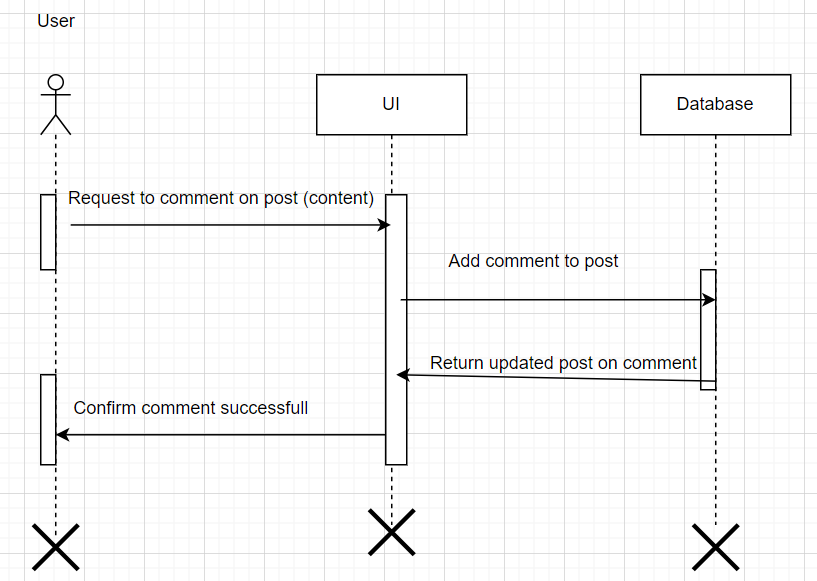
**SD-4: (Edit Post)**

****

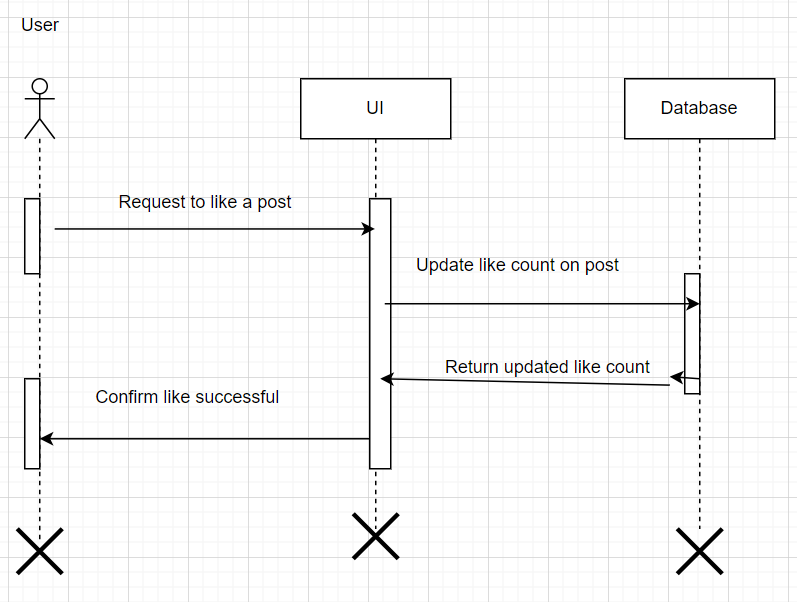
**SD-5: (Delete Post)**

****

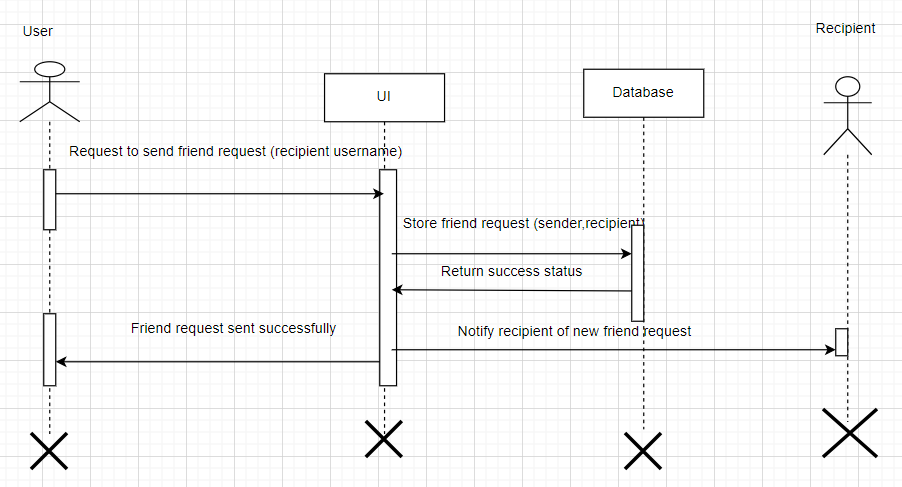
**SD-6: (Comment on Post)**

****

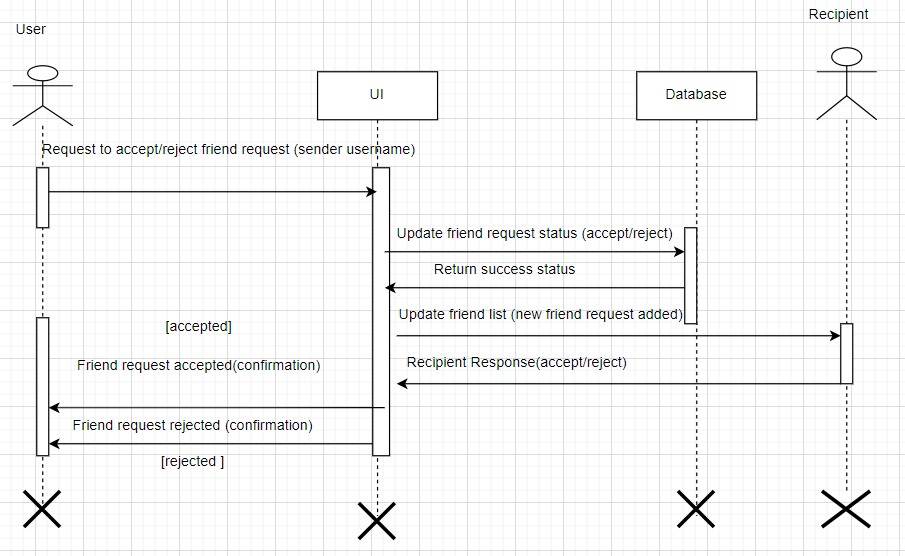
**SD-7: (Like Post)**

****

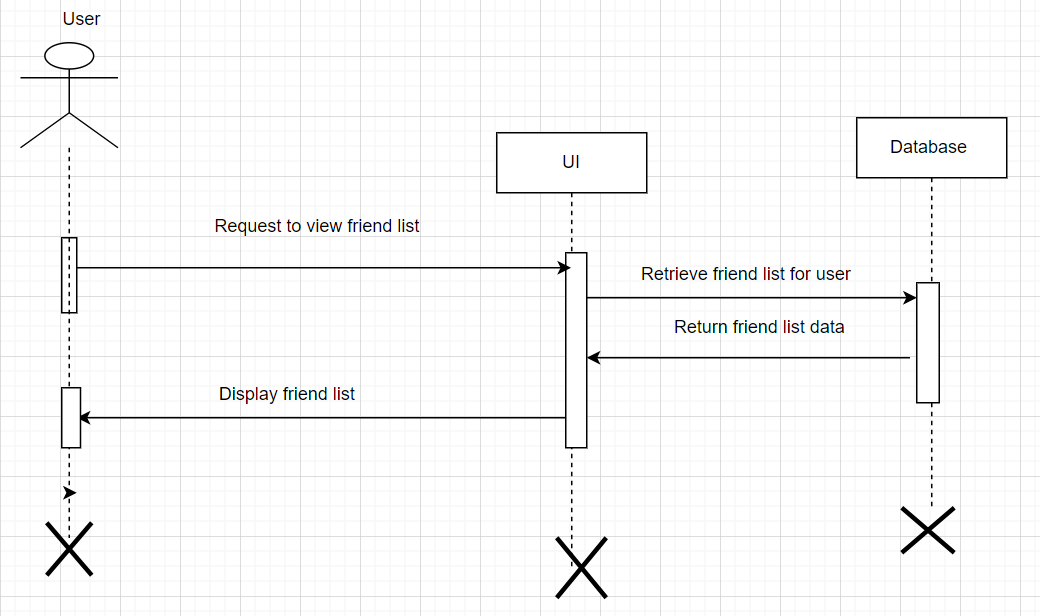
**SD-8: (Send friend Request)**

****

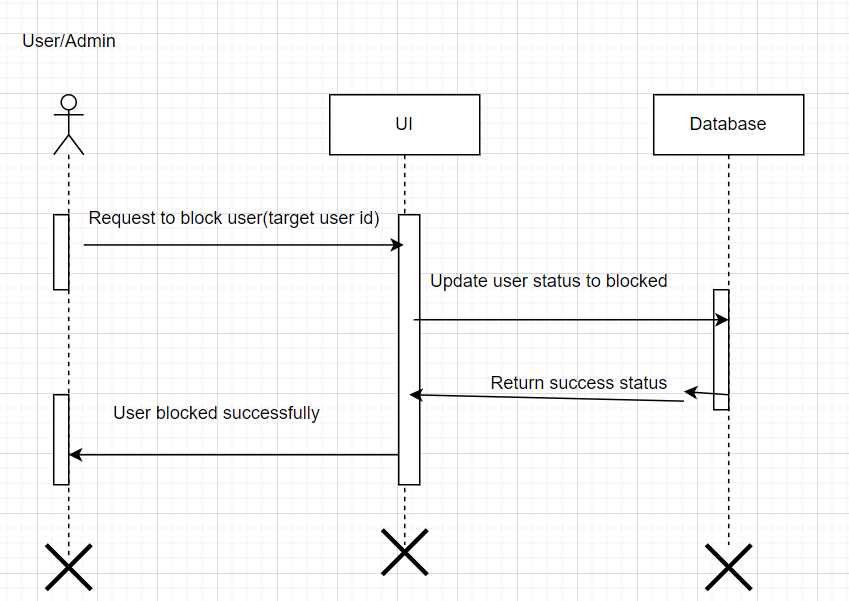
**SD-9: (Accept/Reject Friend Request)**



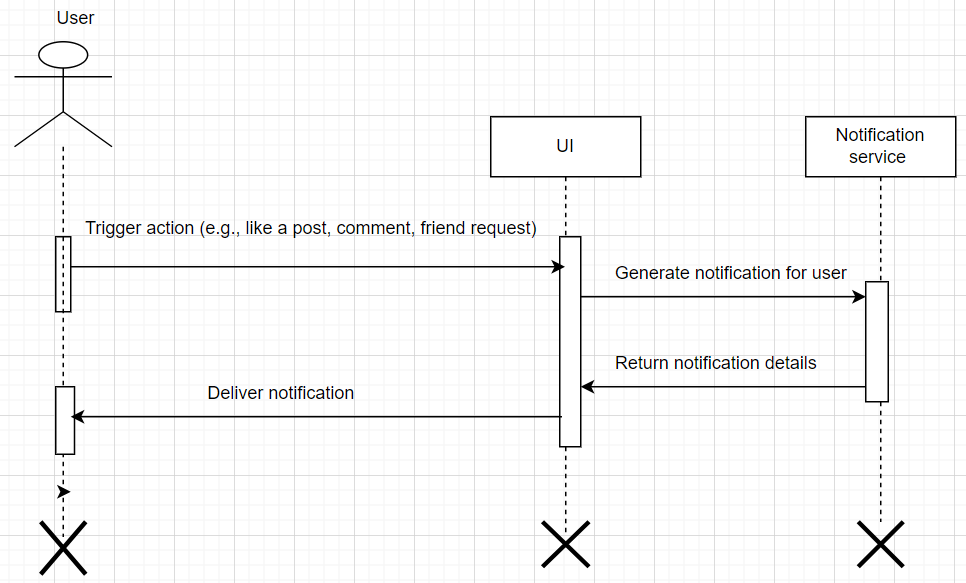
**SD-10: (View Friend List)**

****

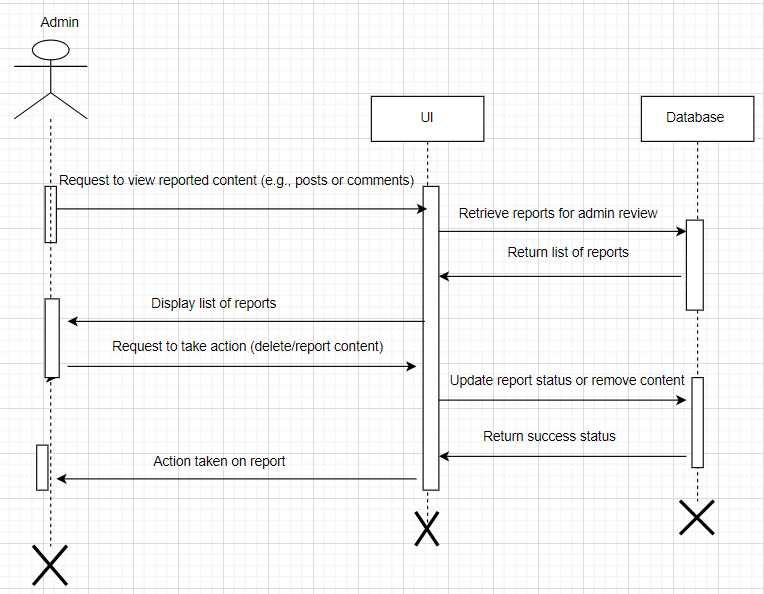
**SD-11: (Block User)**

****

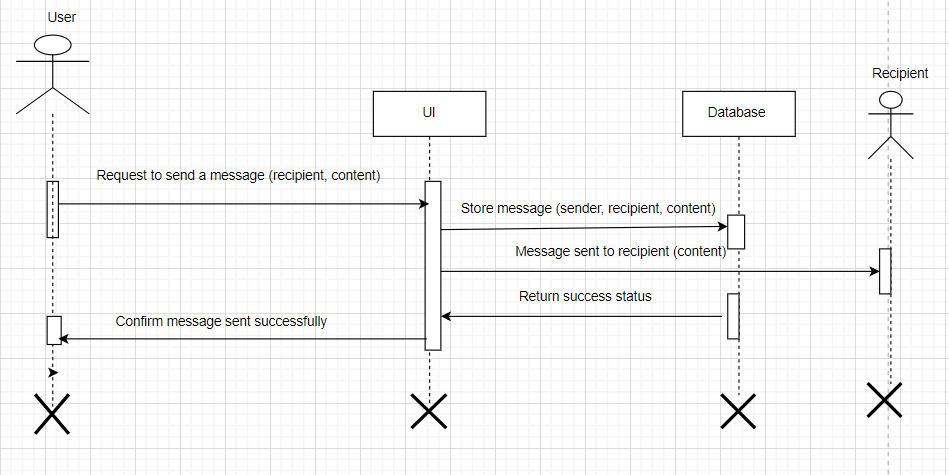
**SD-12: (Receive Notifications)**

****

**SD-13: (Manage Reports)**

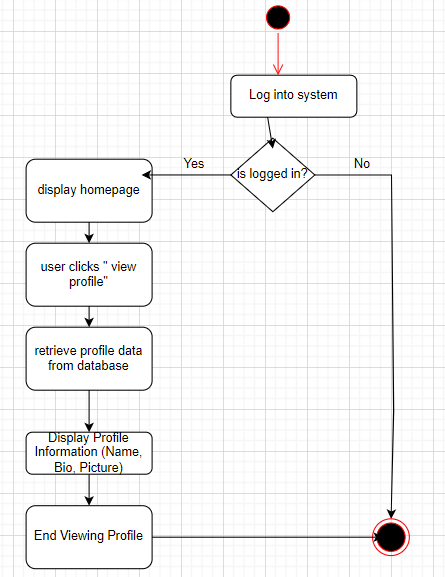
****

**SD-14: (Send Message)**

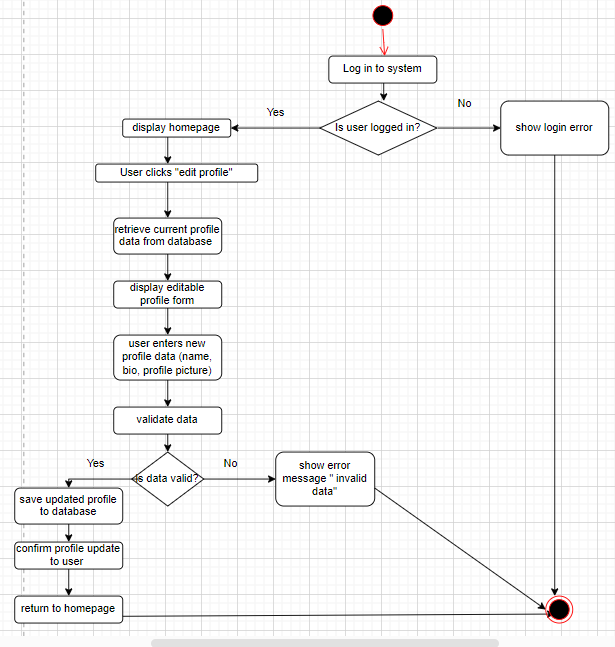
****

**3.Activity Diagrams**

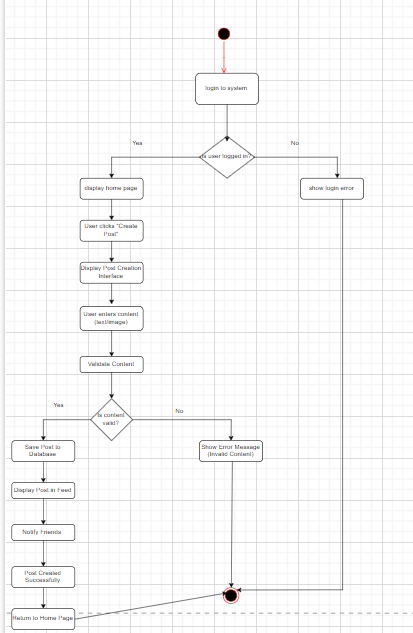
**AD-1: (View Profile)**

****

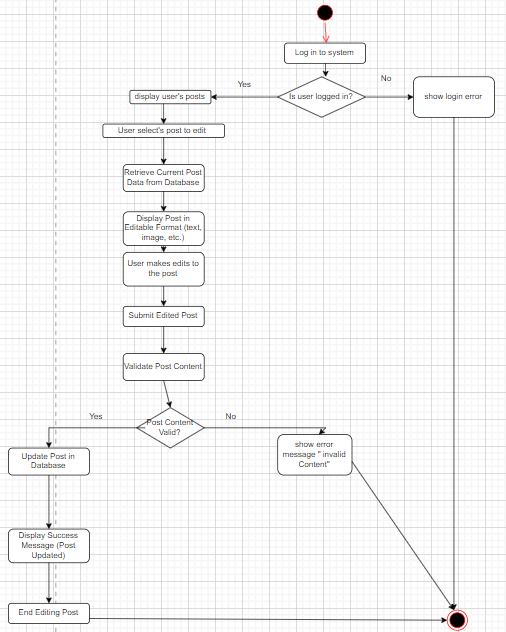
**AD-2: (Edit Profile)**

****

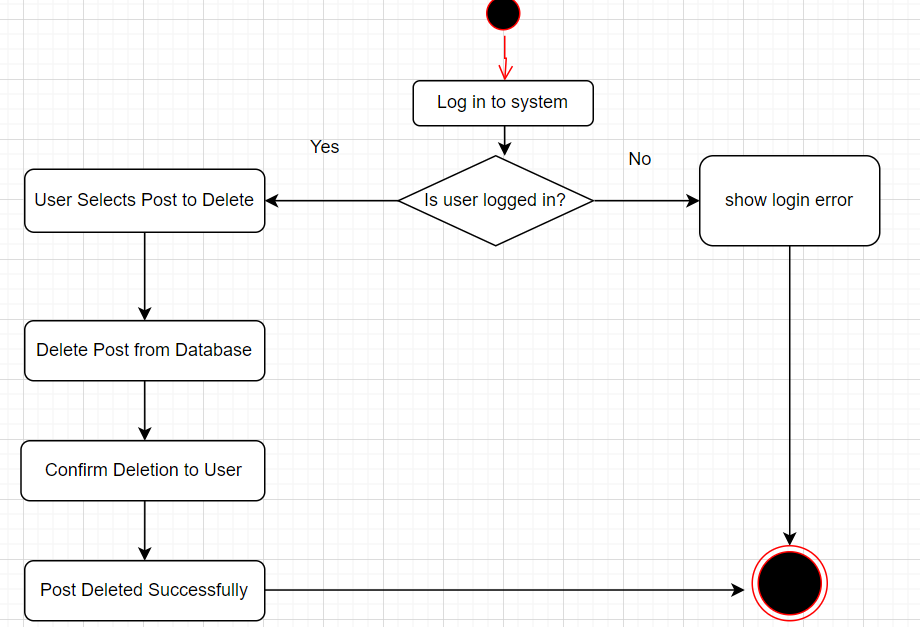
**AD-3: (Create Post)**

****

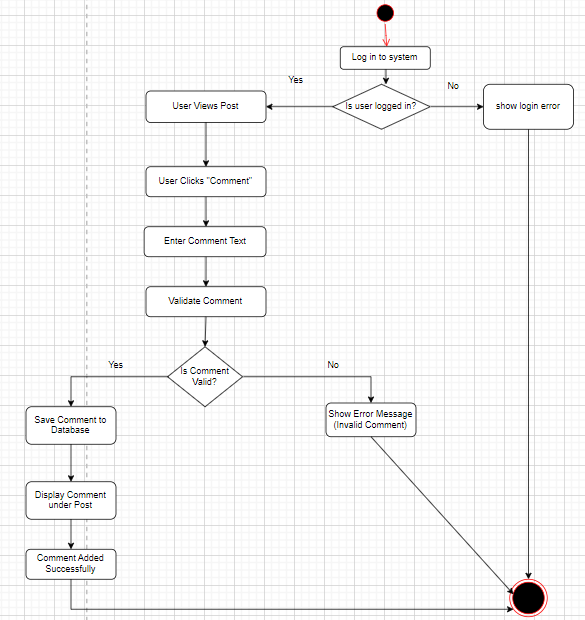
**AD-4: (Edit Post)**

****

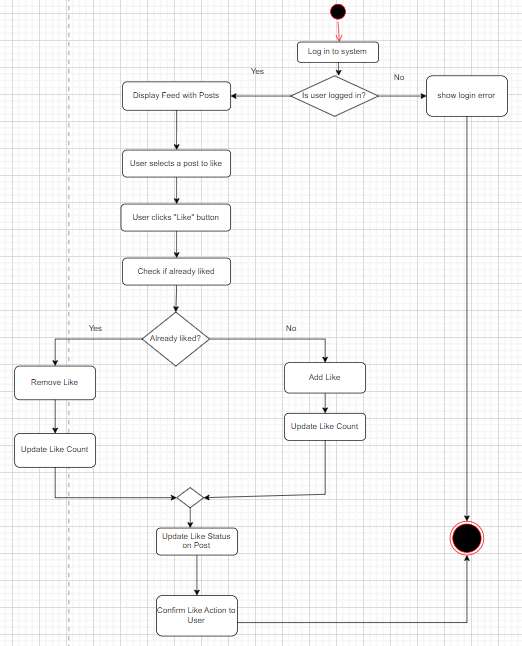
**AD-5: (Delete Post)**

****

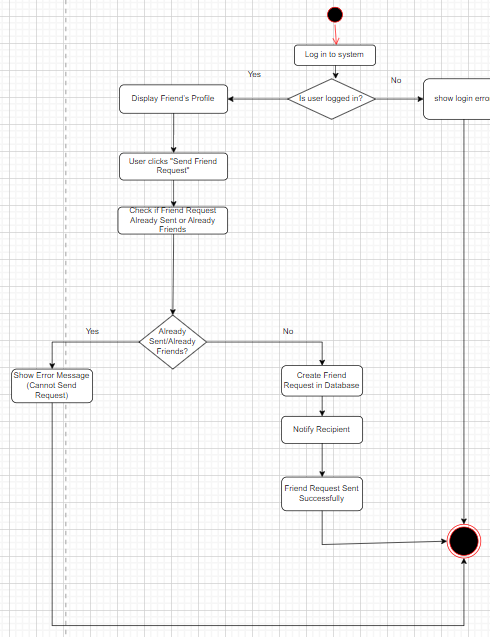
**AD-6: (Comment on Post)**

****

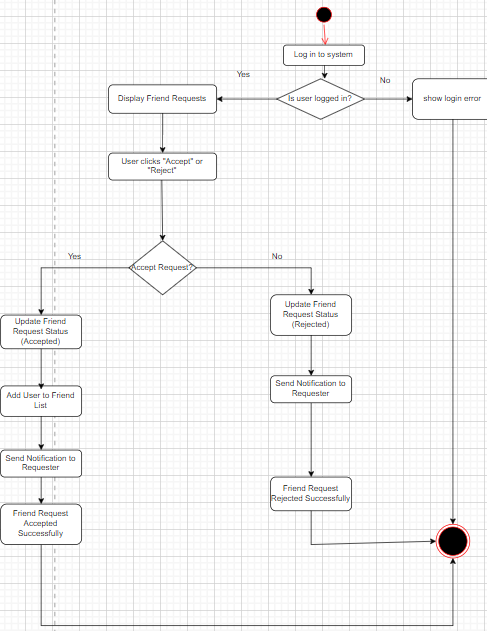
**AD-7: (Like Post)**

****

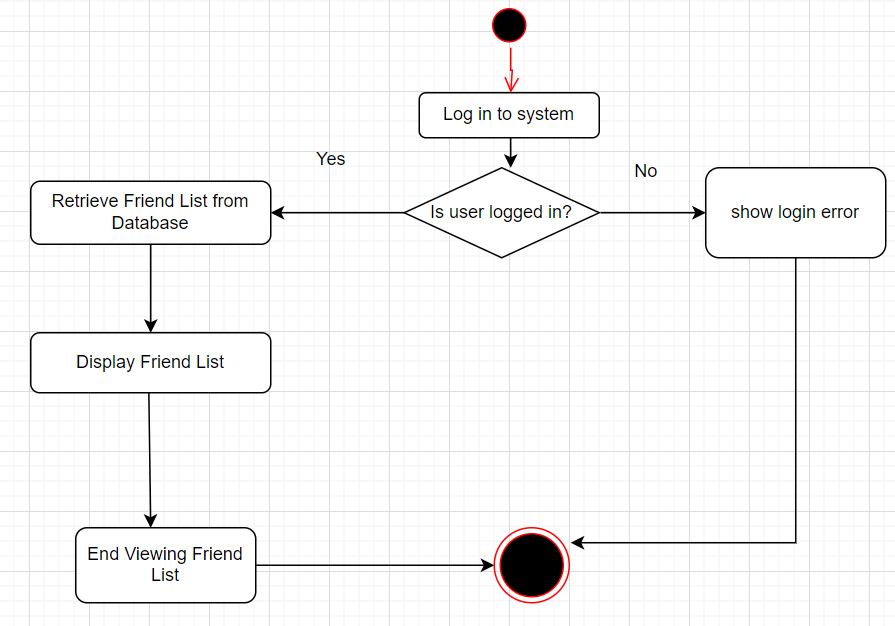
**AD-8: (Send Friend Request)**

****

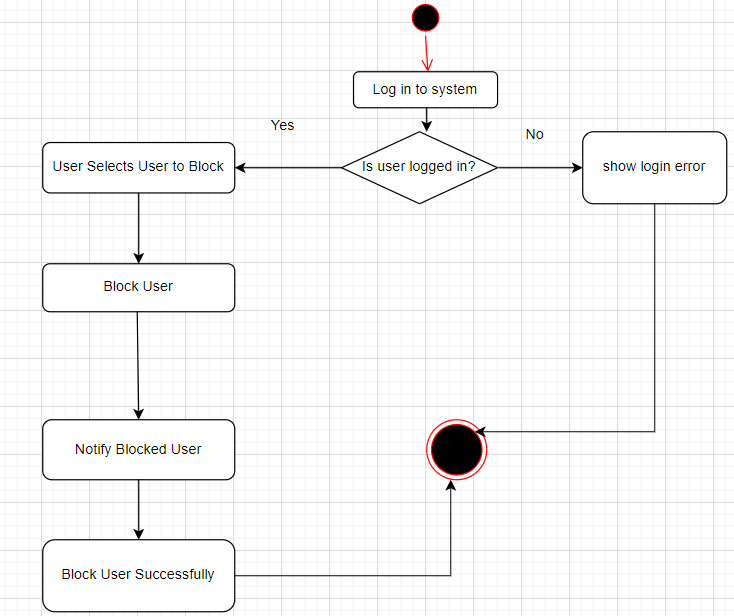
**AD-9: (Accept/Reject Friend Request)**

****

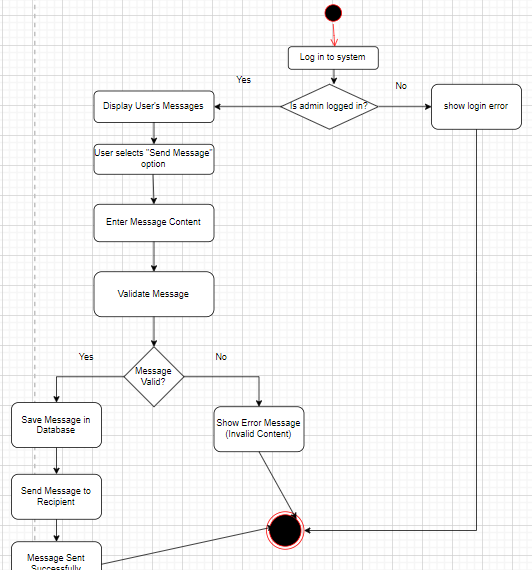
**AD-10: (View Friend List)**

****

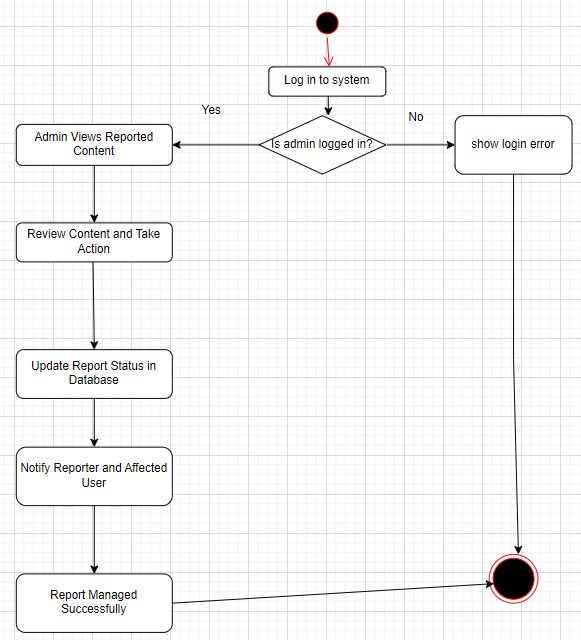
**AD-11: (Block User)**

****

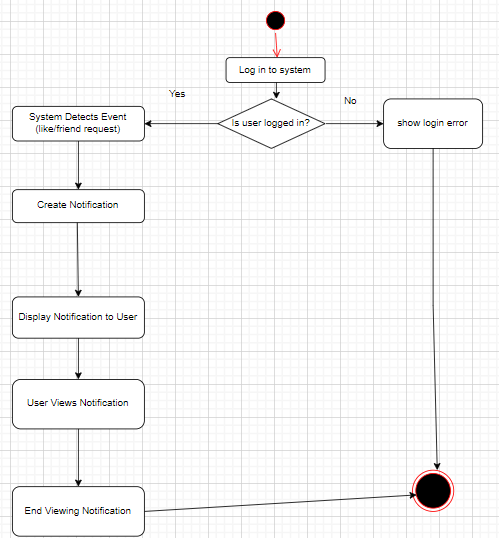
**AD-12: (Send Message)**

****

**AD-13: (Manage Reports)**

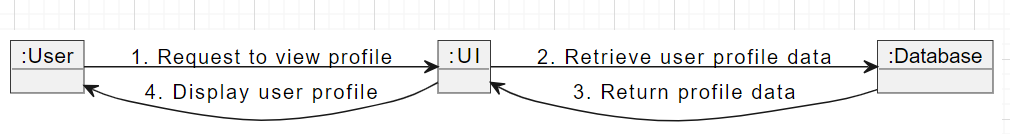
****

**AD-14: (Receive Notifications)**

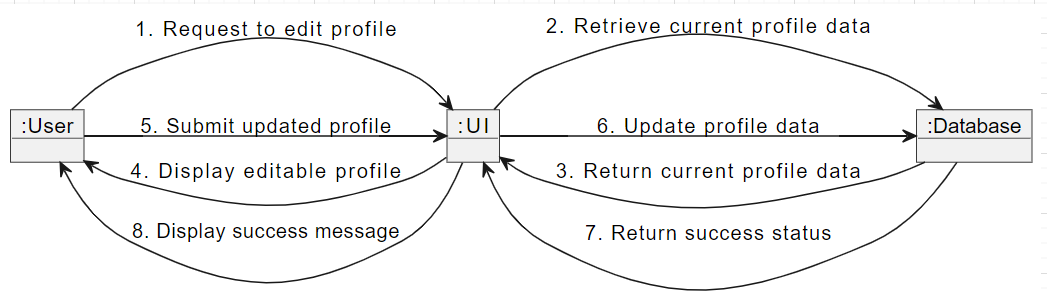
****

**4.Collaboration Diagrams**

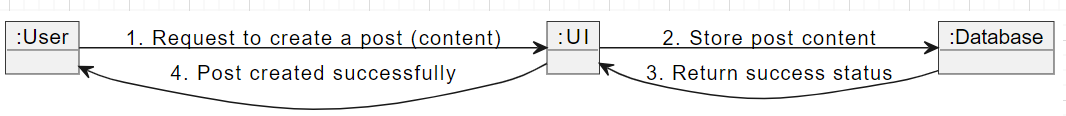
**CD-1: (View Profile)**

****

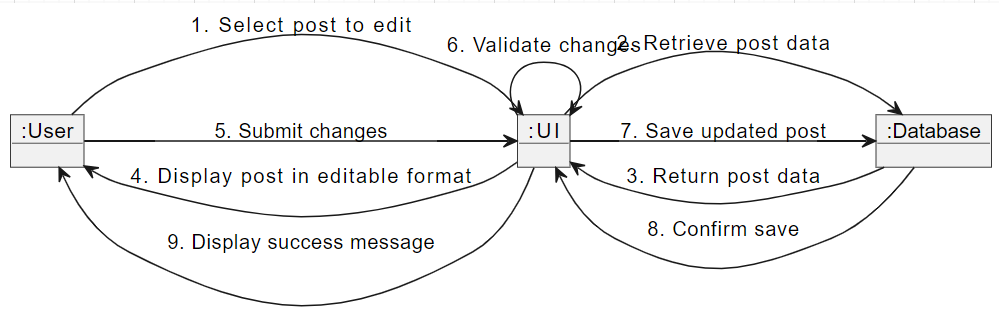
**CD-2: (Edit Profile)**

****

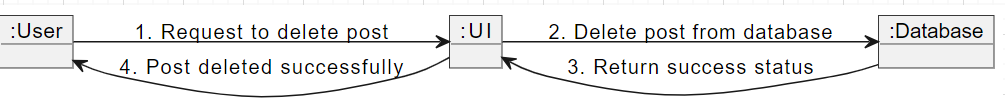
**CD-3: (Create Post)**

****

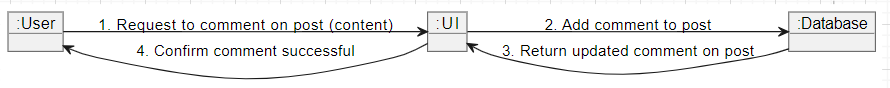
**CD-4: (Edit Post)**

****

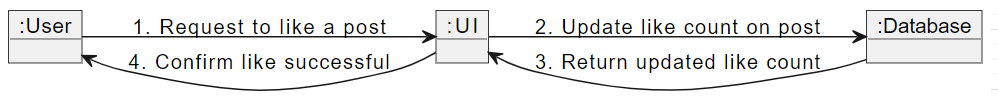
**CD-5: (Delete Post)**

****

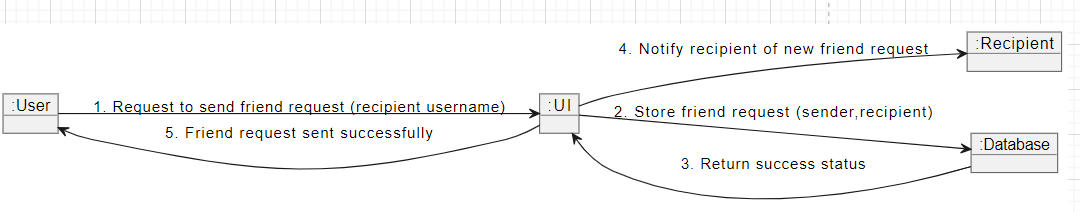
**CD-6: (Comment Post)**

****

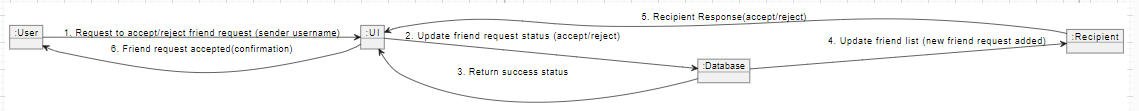
**CD-7: (Like Post)**

****

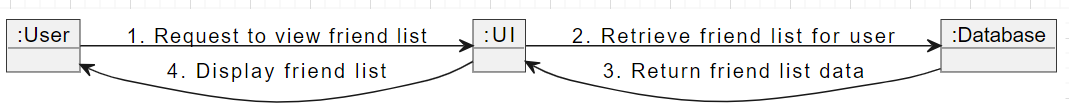
**CD-8: (Send Friend Request)**

****

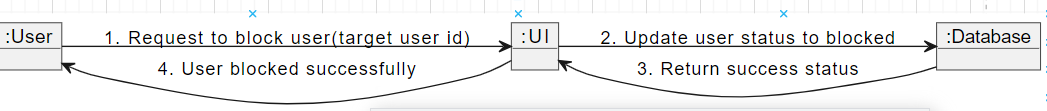
**CD-9: (Accept/Reject Friend Request)**

****

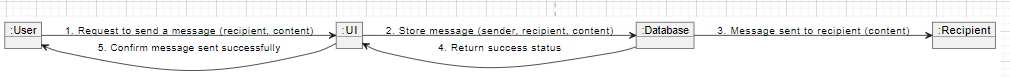
**CD-10: (View Friend List)**

****

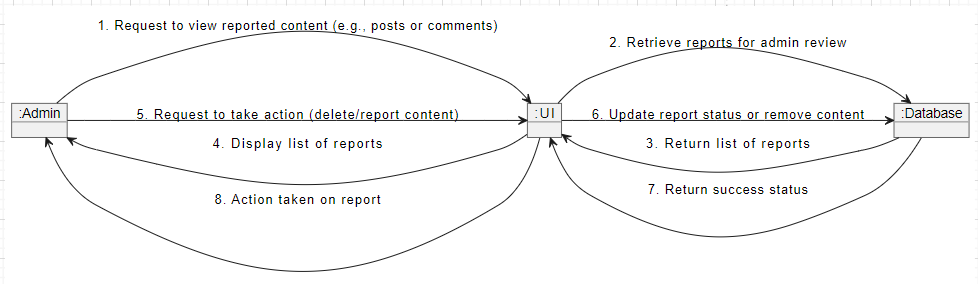
**CD-11: (Block User)**

****

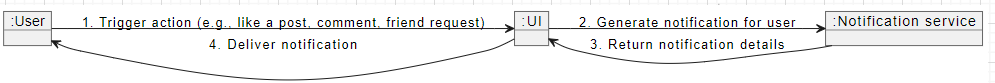
**CD-12: (Send Message)**

****

**CD-13: (Manage Reports)**

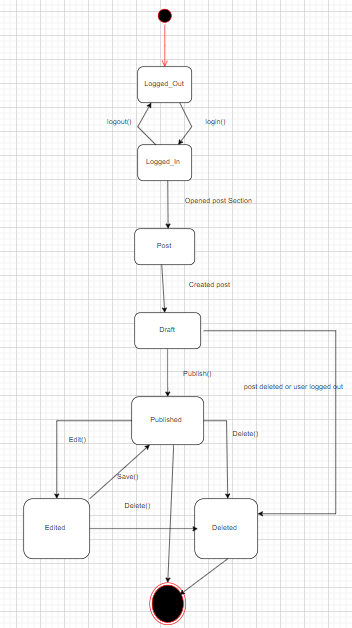
****

**CD-14: (Receive Notifications)**

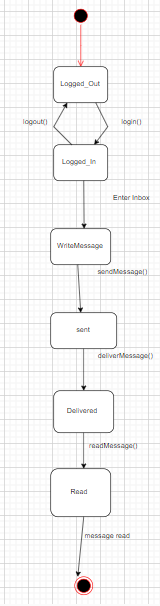
****

**5.State Chart Diagrams**

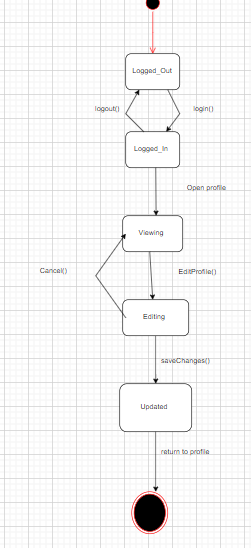
**SCD-1: (Post)**

****

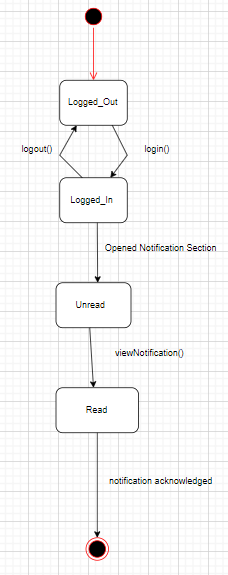
**SCD-2: (Send Message)**

****

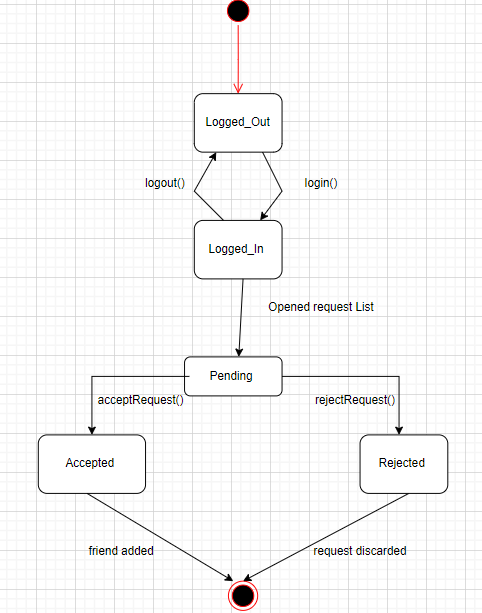
**SCD-3: (User Profile)**

****

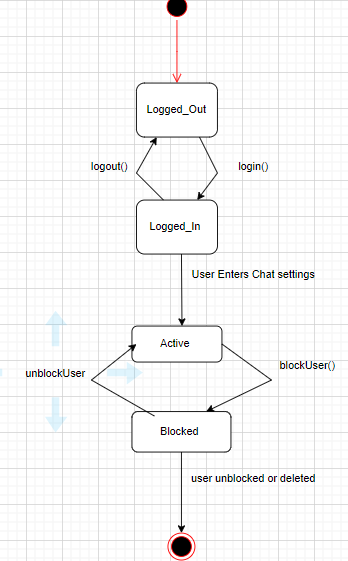
**SCD-4: (Notifications)**

****

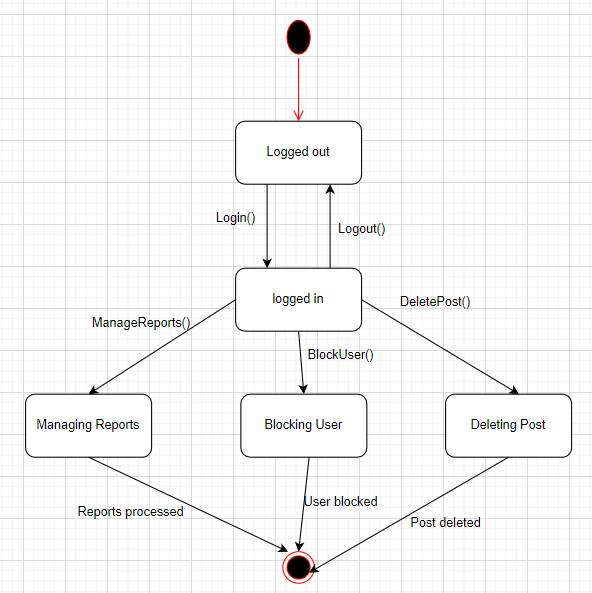
**SCD-5: (Friend Request Acc/Rej)**

****

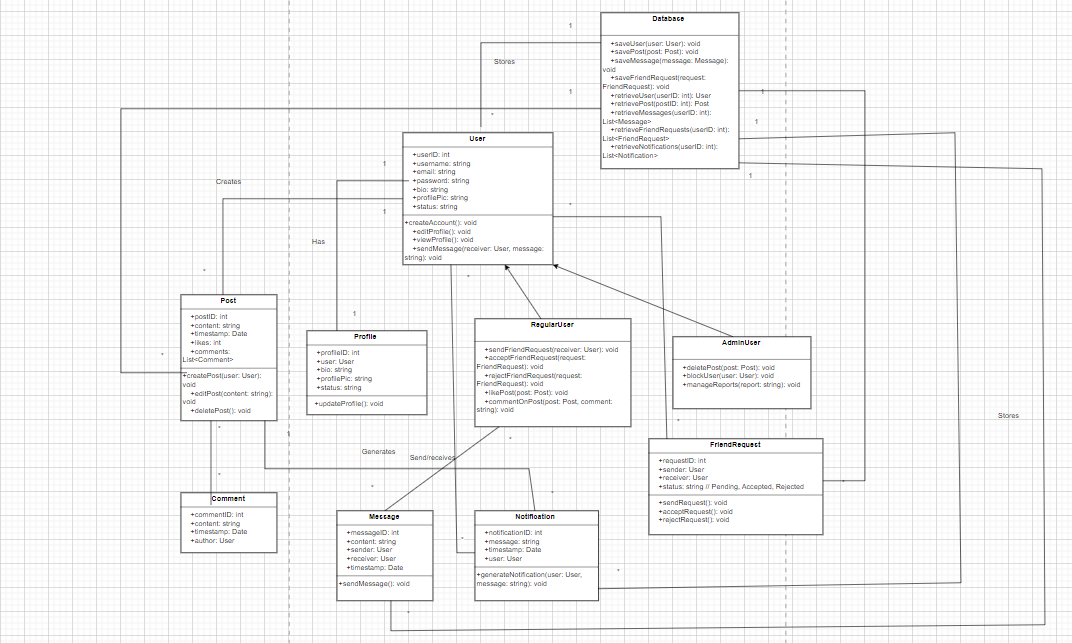
**SCD-6: (Block User)**

****

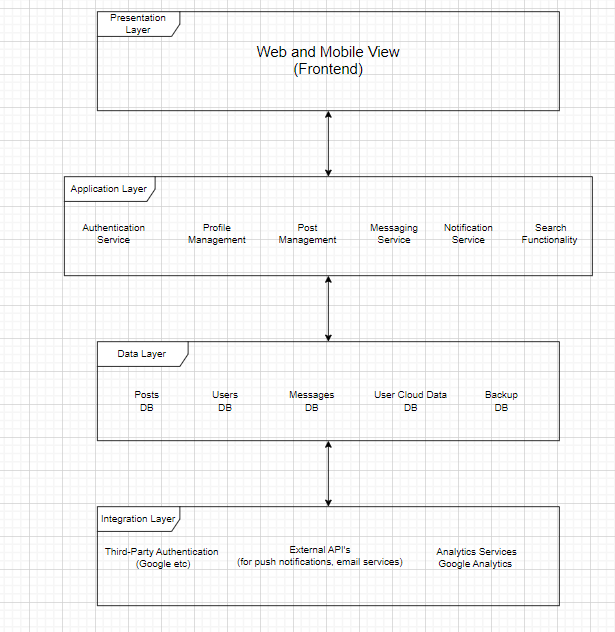
**SCD-7: (Admin)**

****

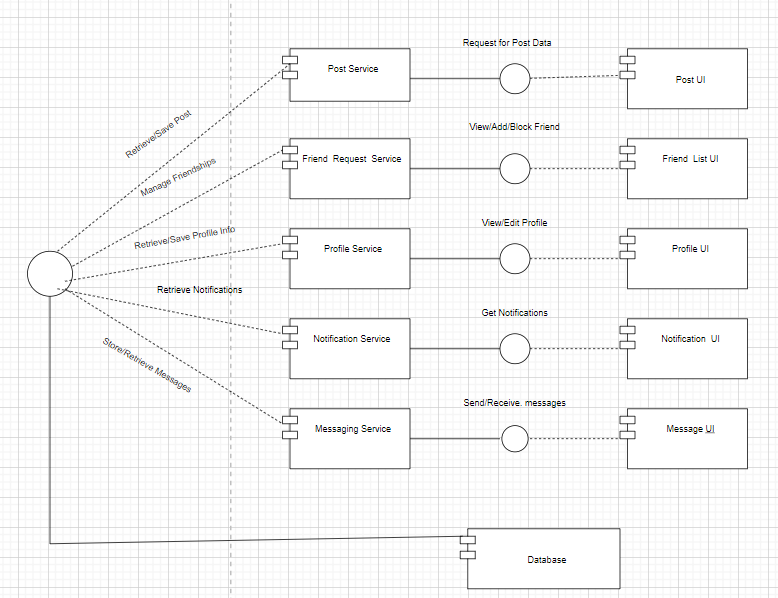
**6.Class Diagram**

****

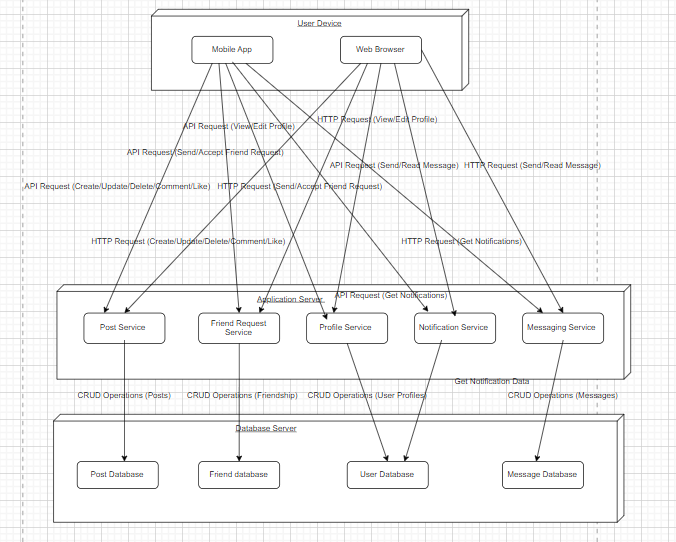
**7.Layered Architecture Diagram**

****

**8.Component Diagram**

****

**9.Deployment Diagram**

****