Hnoss - A Dating Application
Find Your Perfect Date

Contents

Overview

Hnoss is an innovative dating application designed to offer a range of features aimed at enhancing the user experience in the process of discovering and building meaningful connections. The platform incorporates cutting-edge technologies to provide users with a seamless and secure environment for finding matches and planning dates.

Purpose

The primary purpose of Hnoss is to go beyond traditional dating apps by integrating features such as profile matching, chat, interest-based recommendations, and AI-driven date planning. The application aims to facilitate diverse connections while ensuring user safety and satisfaction.

Audience

Hnoss is tailored for individuals aged 18 and above seeking various types of relationships, from casual to long-term commitments. The application targets a broad audience, taking into account different interests, preferences, and relationship goals.

Project Scope

The project includes the development of a feature-rich web application that encompasses profile matching, a user dashboard, chat functionality, interest-based recommendations, and an AI-driven date planning system. The scope also covers features such as a "For You" page, a chatbot, and relationship counselling.

Staff Characteristics

The development team should exhibit expertise in diverse areas, including matchmaking algorithms, real-time communication technologies, AI integration, and user experience design. Knowledge of relationship dynamics and counselling would be beneficial for the team involved in building and maintaining the application.

Existing Systems

While there are existing dating applications, Hnoss differentiates itself by offering a comprehensive set of features, including AI-driven date planning and relationship counselling, making it a unique and user-centric platform.

System Hardware Inventory and Overview

Hnoss will utilise cloud-based servers to ensure scalability. The system will be equipped with robust hardware configurations capable of handling chat interactions, and data processing efficiently.

Software Inventory and Platform Overview

Developed as a cross-platform application, Hnoss will be Web Application. The application will leverage modern development frameworks, ensuring a consistent and engaging user experience across different platforms.

Database Platform Overview

The application's database will be hosted securely, with a focus on efficient data retrieval and storage. The choice of database technology will be driven by factors such as scalability, performance, and data security.

Dependencies and Constraints

System Dependencies

Integration with third-party services for features such as location-based suggestions, and chat functionalities will be essential. Coordination with these services must be managed efficiently to maintain a seamless user experience.

Staffing Dependencies

The success of the project relies on the collaboration of a skilled and interdisciplinary team, including developers, designers, and relationship counsellors.

Network Topology

Hnoss will adopt a distributed network architecture to ensure high availability, efficient data transfer, and optimal communication between various components, especially during real-time interactions.

Constraints

The development timeline is constrained by market demands, and adherence to data protection regulations must be maintained. Compatibility with a variety of devices and screen sizes is also a critical constraint.

Requirements

Business Rules

- User profiles must include detailed interests and preferences to enhance matchmaking accuracy.
- AI-driven date suggestions should consider user preferences, location, and past interactions.

Functional Requirements

- Profile matching
- Chat functionalities.
- Post Sharing
- Friend Management
- Counseling Support

Non-Functional Requirements

- **Performance**
  The system should handle concurrent user connections and use application with minimal latency.

- **Security**
  User data, especially passwords, must be encrypted and secure. Access to premium features should be controlled based on subscription.

- **Reliability**
  The system should be available 99% of the time, with scheduled maintenance communicated in advance.

- **Scalability**

  Hnoss should be designed to scale to accommodate a growing user base.

Feasibility Requirements

Financial feasibility, considering revenue models such as subscription plans or in-app purchases such as date-booking.

Data Integrity Requirements

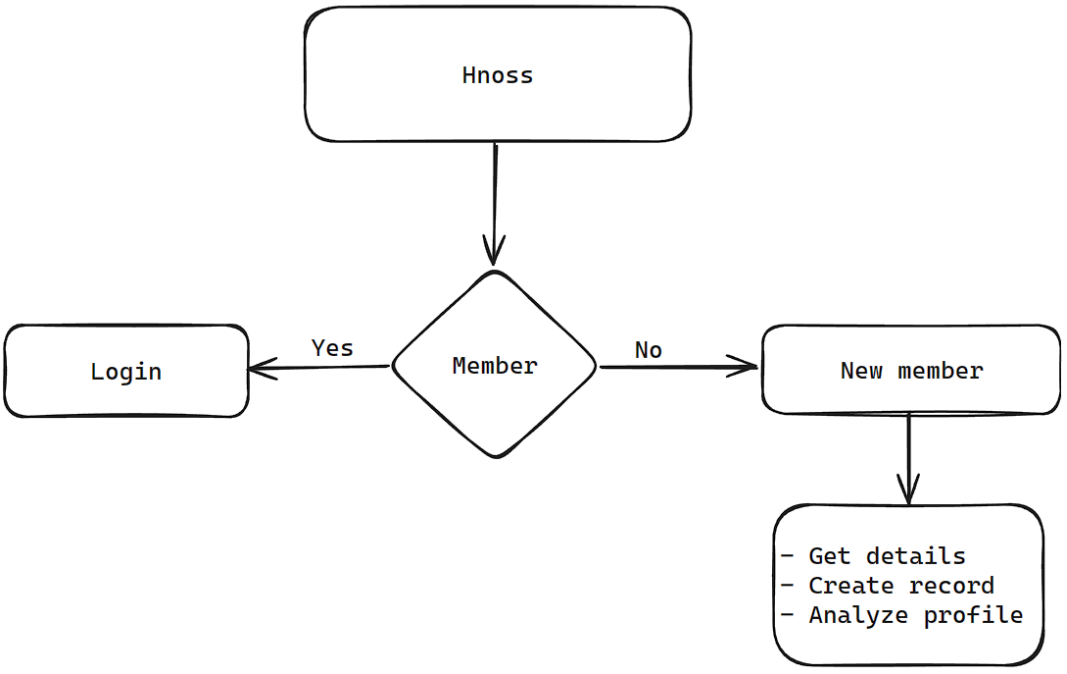
- Encryption of sensitive data during transmission and storage.
- Regular data backups and secure storage mechanisms.

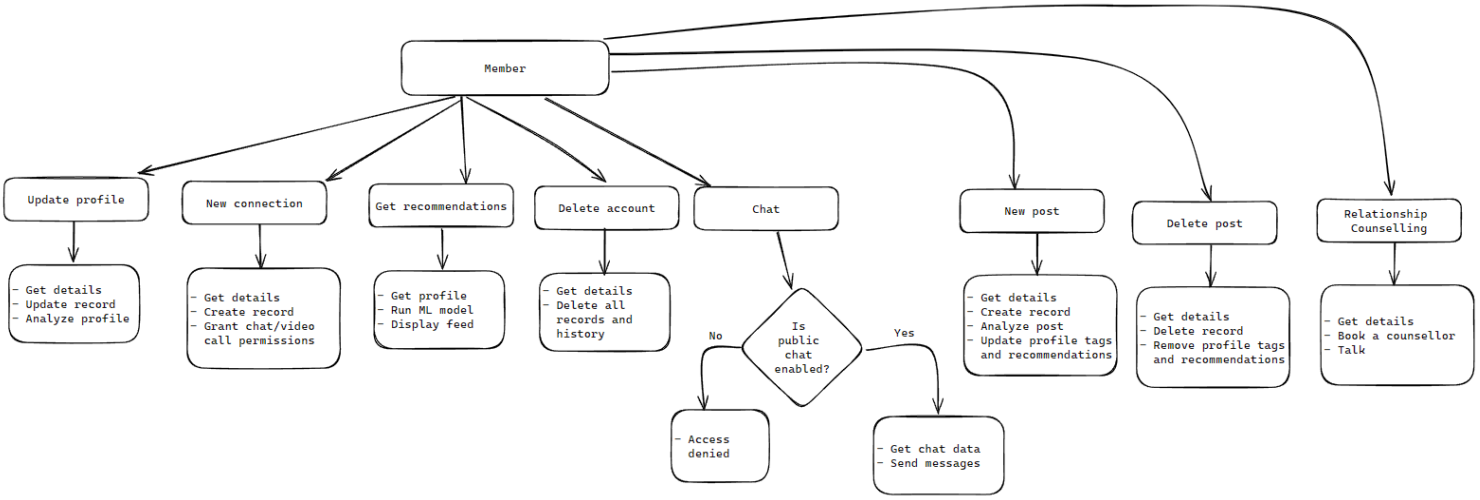
Security Requirements

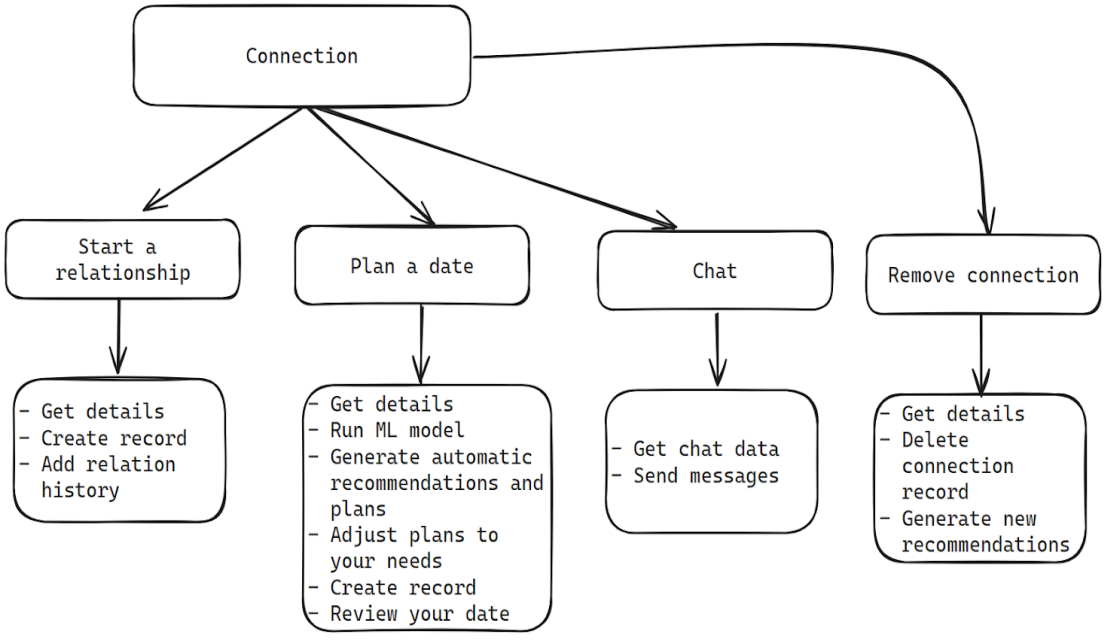
- Robust authentication mechanisms for user accounts.
- Regular security audits and updates to address vulnerabilities.

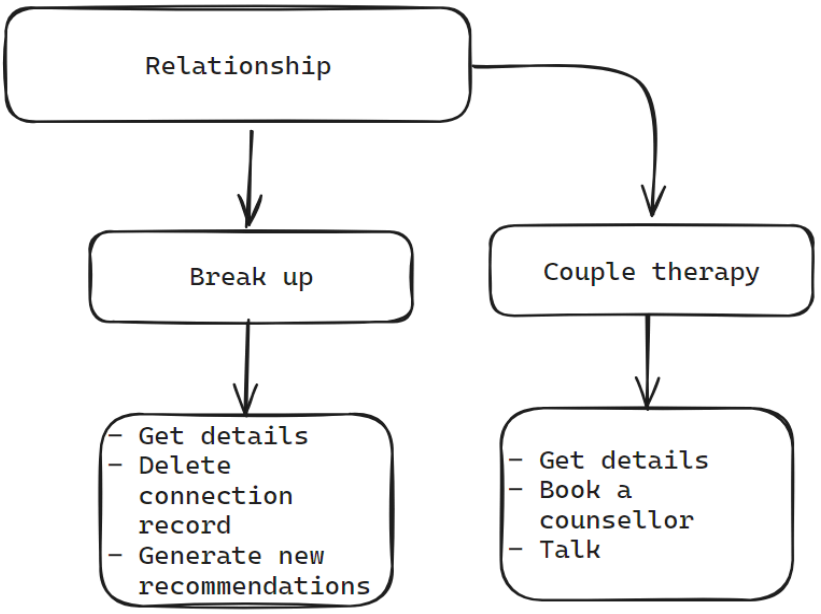
In conclusion, Hnoss is designed to be a comprehensive dating application, offering a wide array of features to cater to the diverse needs and preferences of its user base. The SRS provides a detailed roadmap for the development team to implement these features successfully.

Decision Tree




Member
Update profile
- Get details
- Update record
- Analyze profile
New connection
- Get details
- Create record
- Grant chat/video call permissions
Get recommendations
- Get profile
- Run ML model
- Display feed
Delete account
- Get details
- Delete all records and history
Chat
Is public chat enabled?
No
- Access denied
Yes
- Get chat data
- Send messages
New post
- Get details
- Create record
- Analyze post
- Update profile tags and recommendations
Delete post
- Get details
- Delete record
- Remove profile tags and recommendations
Relationship Counselling
- Get details
- Book a counsellor
- Talk


Connection
Start a relationship
- Get details
- Create record
- Add relation history
Plan a date
- Get details
- Run ML model
- Generate automatic recommendations and plans
- Adjust plans to your needs
- Create record
- Review your date
Chat
- Get chat data
- Send messages
Remove connection
- Get details
- Delete connection record
- Generate new recommendations



Decision Table

Conditions		
Login	**No**	**Yes**
Actions		
Access Hnoss	**-**	**X**
Register as New Member	**X**	**-**
Recommend	**-**	**X**
Post	**-**	**X**
Update Profile	**-**	**X**
Delete Post	**-**	**X**
Delete Account	**-**	**X**

Conditions			
Connection	**No**	**Yes**	**Yes**
Relation	**No**	**No**	**Yes**

Actions			
Chat	**-**	**X**	**X**
Public Chat	**X**	**X**	**X**
Plan a Date	**-**	**X**	**X**
Counselling	**X**	**X**	**X**
Break Up	**-**	**-**	**X**
Start a relationship	**-**	**X**	**-**

Algebraic Specifications

Users

Types:

defines users
uses boolean, double, string, user_element

user_element:
defined by username, password, photo, name, gender, dob, bio, address, phone, email, tags, post_id, connection_element, counsellor_id

connection_element:
defined by list[string]

Exceptions:

novalue, valueerror, notfound

Signature:

- **insert**:
  - users **X** user_element **->** boolean **+** {novalue} **+** {valueerror}
- **delete**:
  - users **X** user_element **X** string **X** string **->** boolean **+** {notfound}
- **find**:
  - users **X** string **->** user_element **+** {novalue} **+** {valueerror} **+** {notfound}
- **login**:
  - users **X** string **X** string **->** user_element **+** {notfound}
- **logout**:
  - users **X** user_element **->** boolean **+** {notfound}
- **update**:
  - users **X** user_element **X** user_element **->** boolean **+** {novalue} **+** {valueerror}
- **getconnections**:
  - users **X** user_element **->** connection_element
- **checkconnection**:
  - users **X** user_element **X** user_element **->** boolean
- **connect**:

    - users **X** user_element **X** user_element -> boolean + {notfound}

- **addrelationship**:
    - users **X** user_element **X** user_element -> boolean + {notfound}

- **getrelationship**:
    - users **X** user_element -> user_element + {notfound}

- **checkrelationship**:
    - users **X** user_element **X** user_element -> bool + {notfound}

- **breakup**:
    - users **X** user_element **X** user_element -> boolean + {notfound}

- **post**:
    - users **X** user_element **X** post_id **->** boolean **+** {novalue} **+** {valueerror}

- **delete_post**:
    - users **X** element **X** post_id **->** boolean **+** {notfound}

- **plandate**:
    - users **X** user_element **X** user_element **X** price **->** boolean

- **getCounsellor**:
    - users **X** user_element **X** counsellor_id-> boolean + {notfound}

- **evaluateProfile**:
    - users **X** user_element -> double

Equations:

find(u, username) = user_elem
delete(u, user, username, password) = delete if find(u, username) == user
plandate(u, user1, user2) = true if checkconnection(user1, user2) == true
addrelationship(u, user1, user2) = true if checkconnection(user1, user2) == true
breakup(u, user1, user2) = true if checkrelationship(user1, user2) == true

Chats

Types:

defines chat
uses string, timestamp

Exceptions:

novalue, valueerror, notfound, notauthorized

Signature:

- **insert**:
  - chats **X** chat **X** string **X** timestamp **->** chats + {notfound} + {notauthorized}

- **delete**:
  - chats **X** chat **X** string **->** chats + {notfound} + {notauthorized}

- **find**:
  - chats **X** chat **->** chat + {notfound} + {notauthorized}

Posts

Types:

post_element:
defined by postID, photo, title, description, location, date, likes, comments

Exceptions:

novalue, valueerror, notfound

Signature:

- **post**:
  - posts **X** post_element **->** posts **+** {novalue} **+** {valueerror}

- **delete_post**:
  - posts **X** postID **->** posts **+** {notfound}

- **find**:
  - posts **X** postID **->** post_element + {notfound}

Counsellor

Types:
uses boolean, double
defined by counsellor_id, username, chats

Exceptions:
novalue, valueerror, notfound

Signature:

- **accept**:
  - counsellor **X** username **->** boolean
- **reject**:
  - counsellor **X** username **->** boolean
- **setprice**:
  - counsellor **X** double **->** boolean + {notfound}

Date

Types:
uses boolean, double, string, integer
defined by date_id, username1, username2, place, time, price, review, status

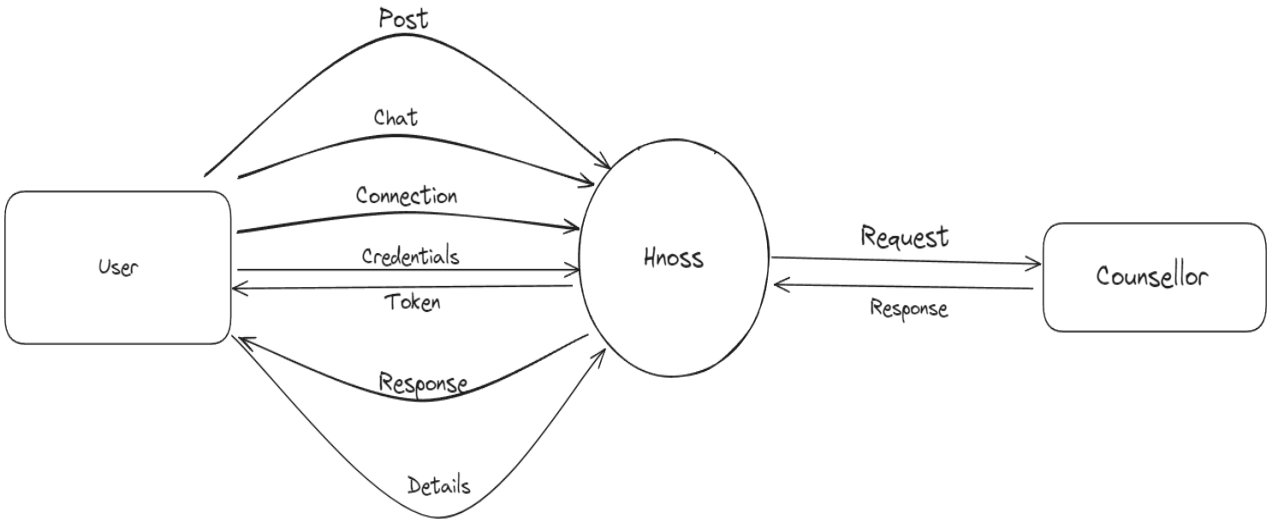
Exceptions:
novalue, valueerror, notfound

Signature:

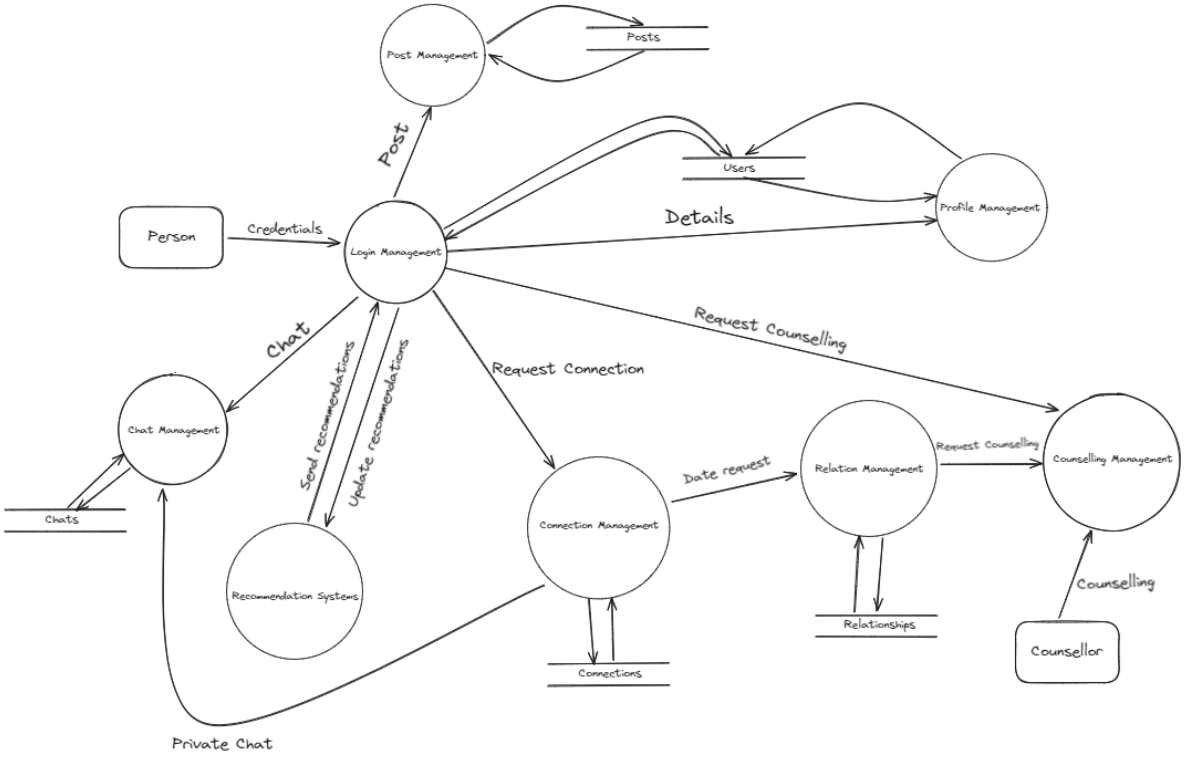
- **accept**:
  - date_id **X** username **->** boolean
- **reject**:
  - date_id **X** username **->** boolean
- **review**:
  - date_id **X** string **->** boolean + {notfound}
- **getstatus**:
  - date_id **->** string + {notfound}
- **setstatus**:
  - date_id **X** integer **->** boolean + {notfound}

DataFlow Diagram

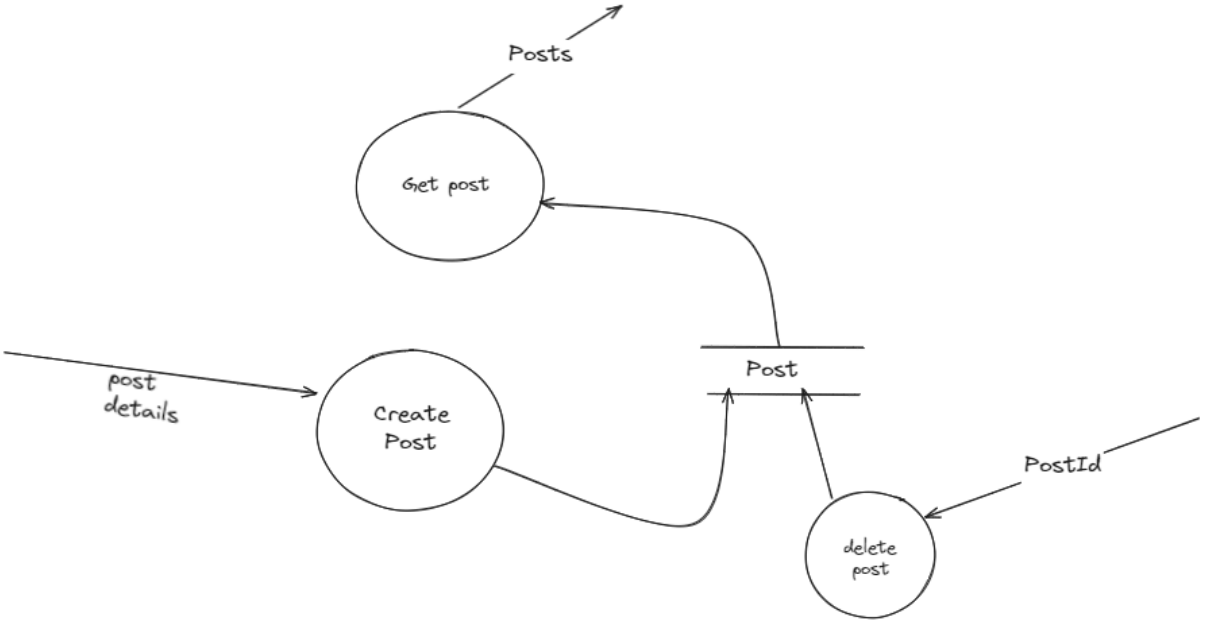
- **Level 0:**

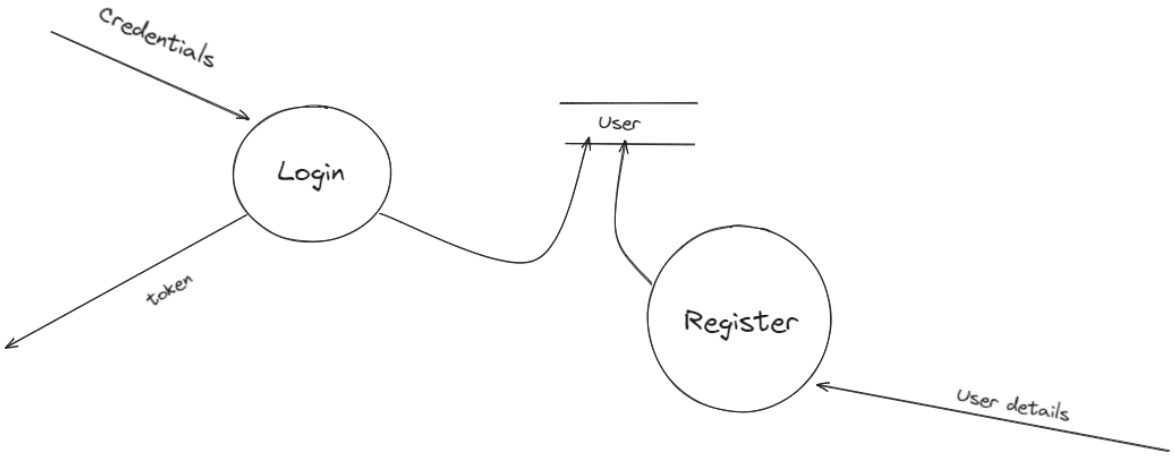


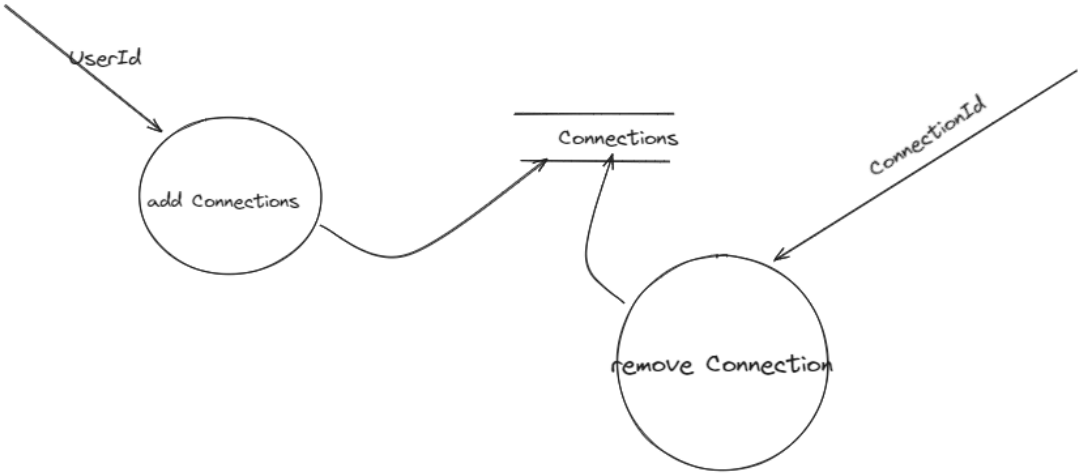
- **Level 1:**

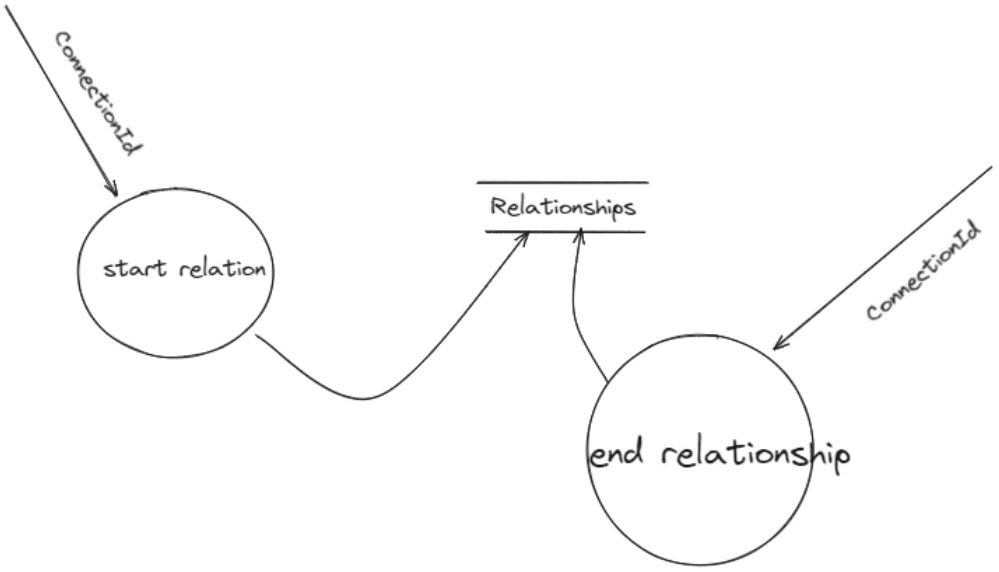


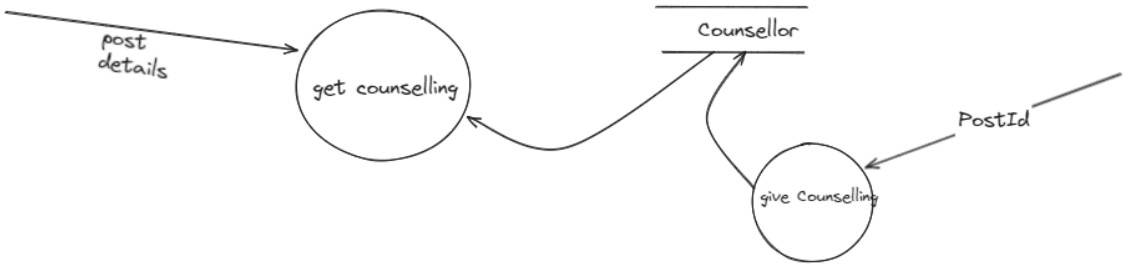
- **Level 2:**

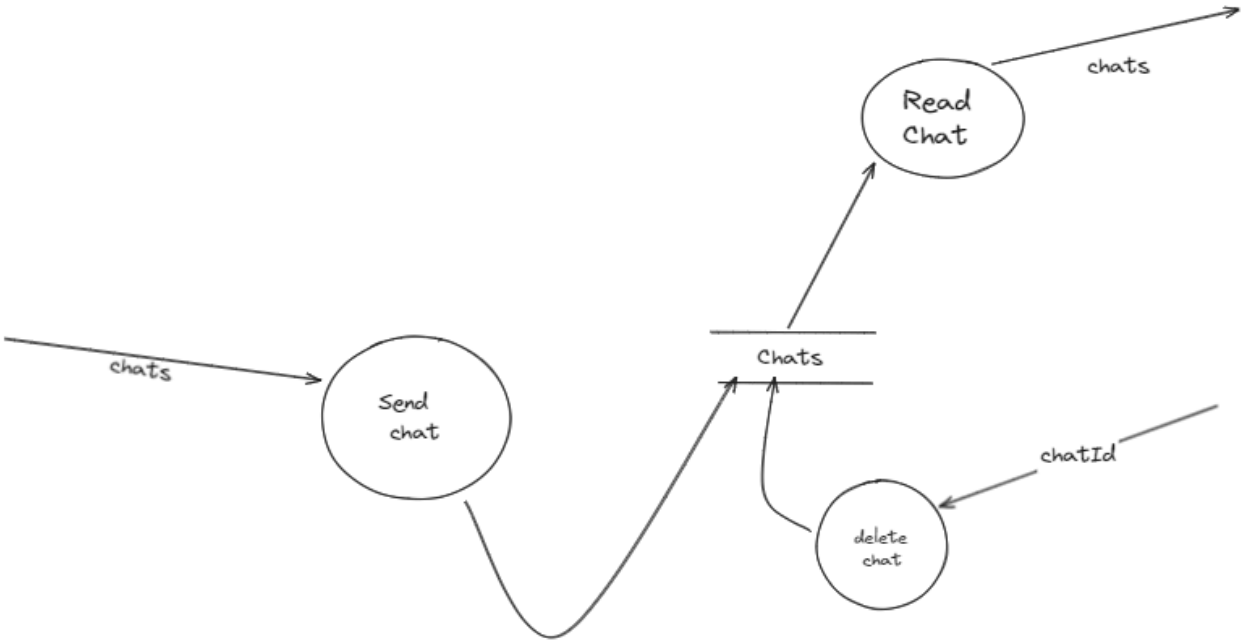

Posts
Get post
Post
post details
Create Post
PostId
delete post


Credentials
Login
User
xjoker
Register
User details

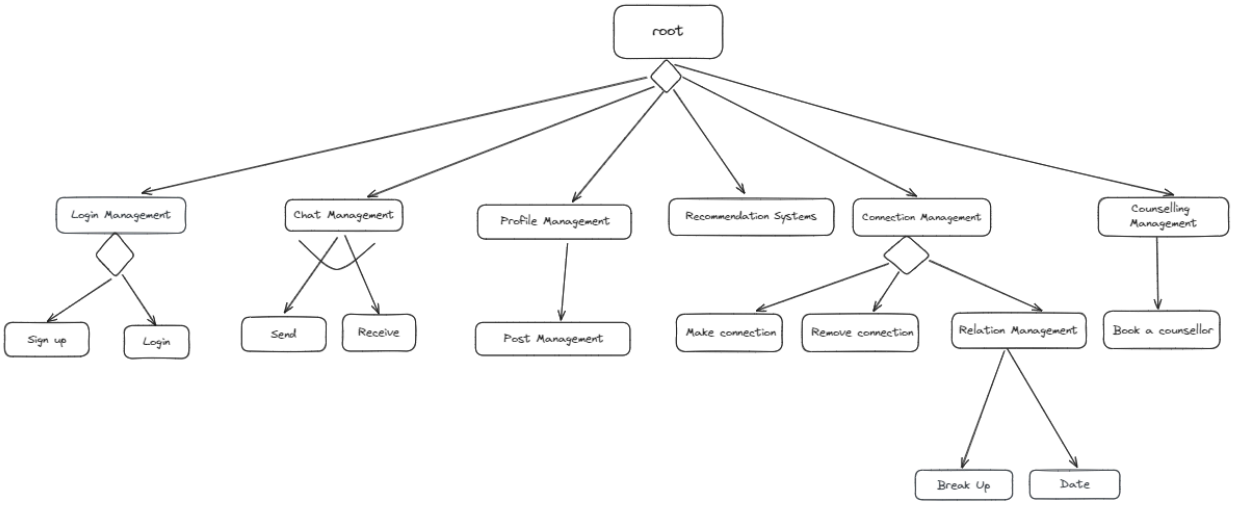

UserId
add Connections
Connections
ConnectionId
remove Connection


ConnectionId
start relation
Relationships
ConnectionId
end relationship

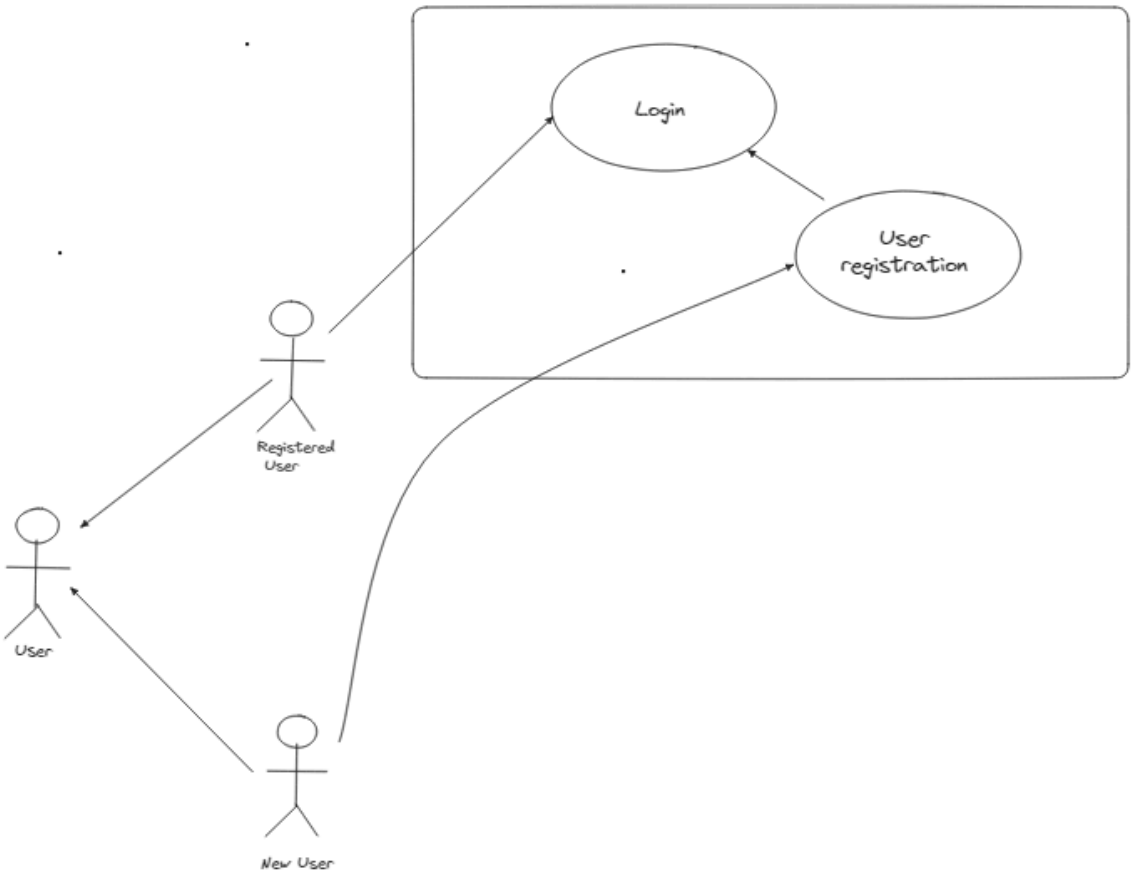

post
details
get counselling
Counsellor
PostId
give Counselling

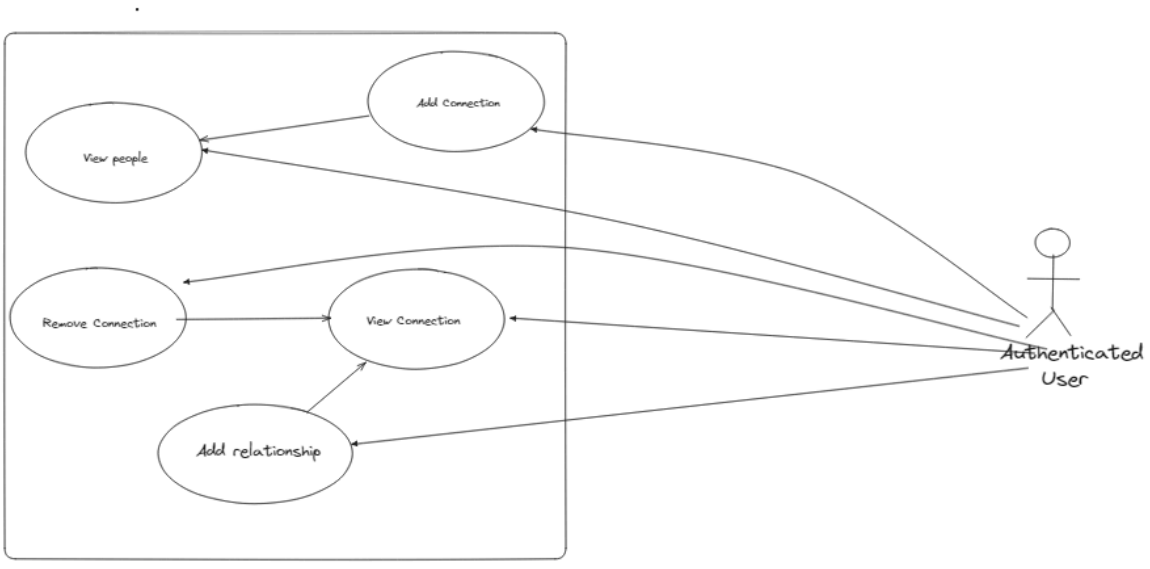

chats
Read
Chat
chats
Send
chat
Chats
chatId
delete
chat

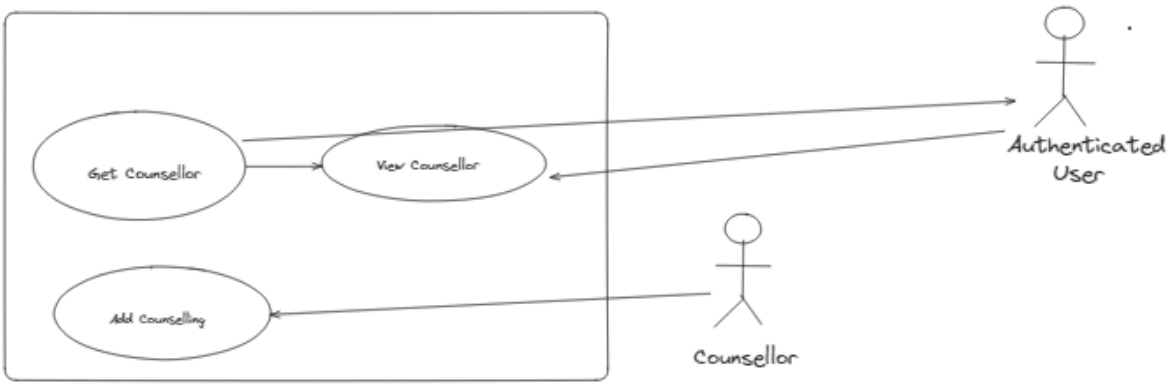
Structure Diagram

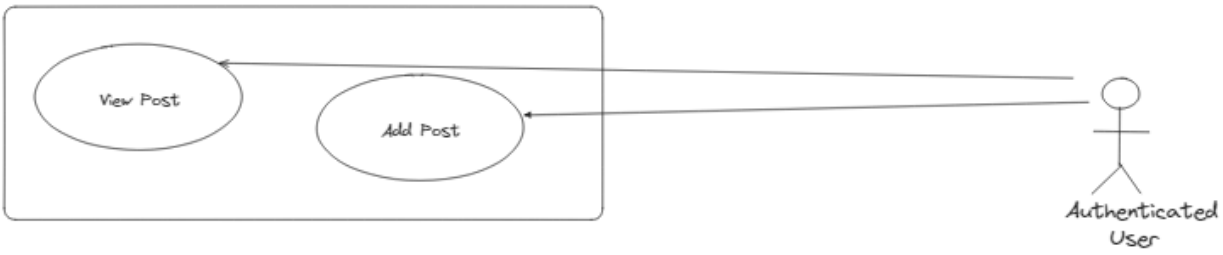


Use Case Diagram

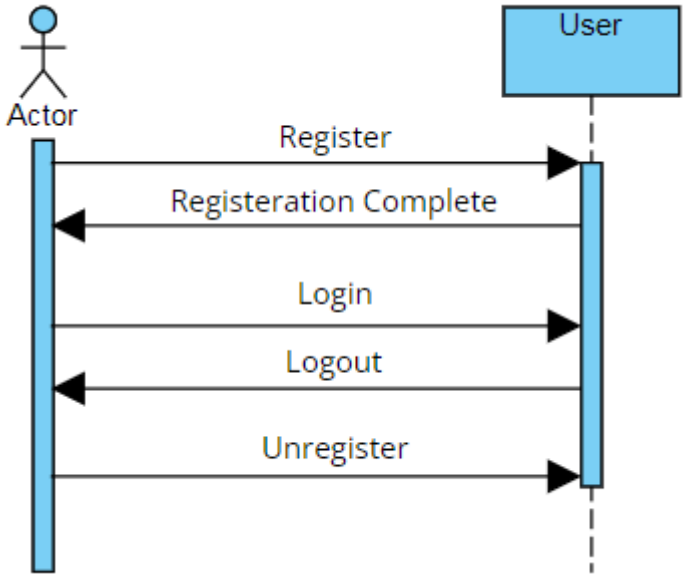


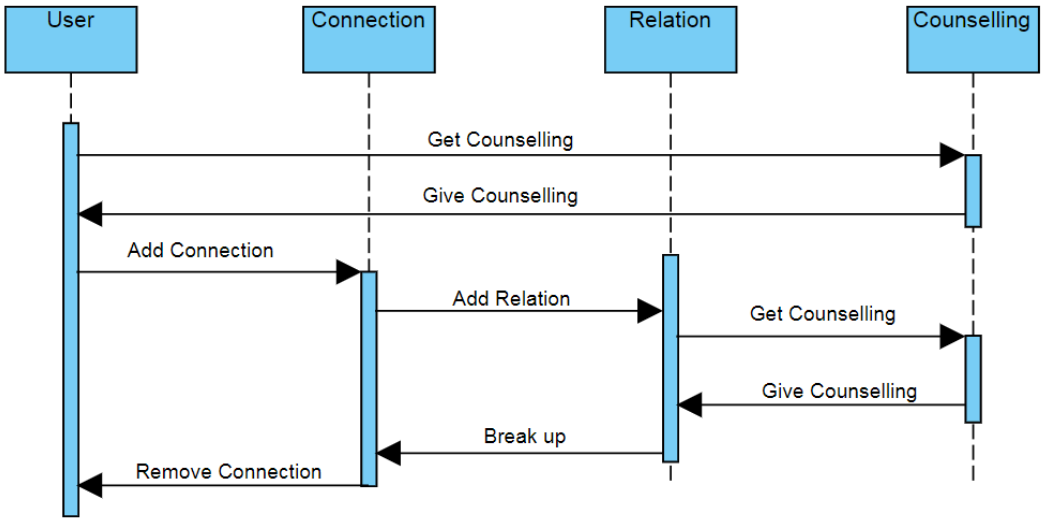


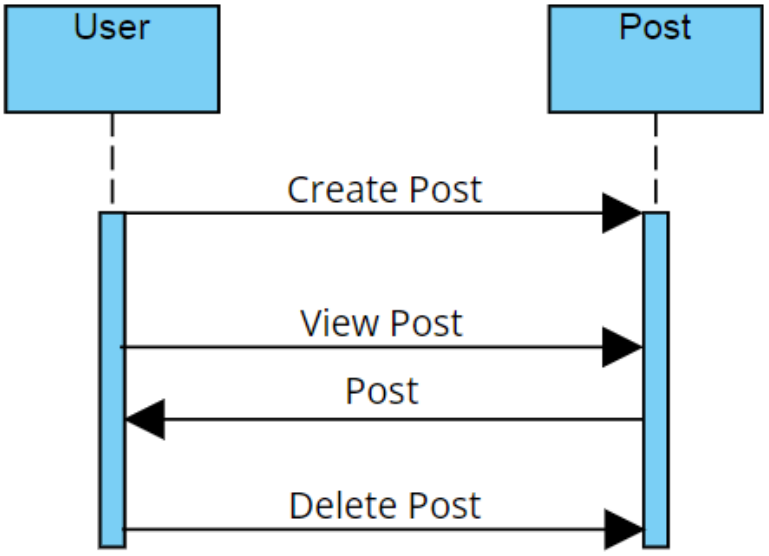

Get Counsellor
View Counsellor
Add Counselling
Authenticated User
Counsellor


View Post
Add Post
Authenticated User

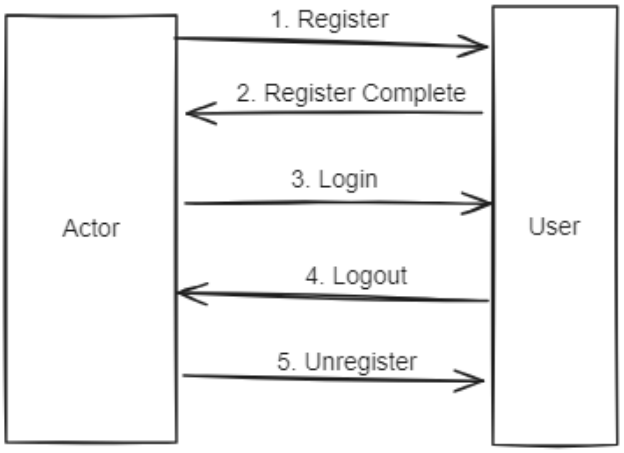
Sequence Diagram

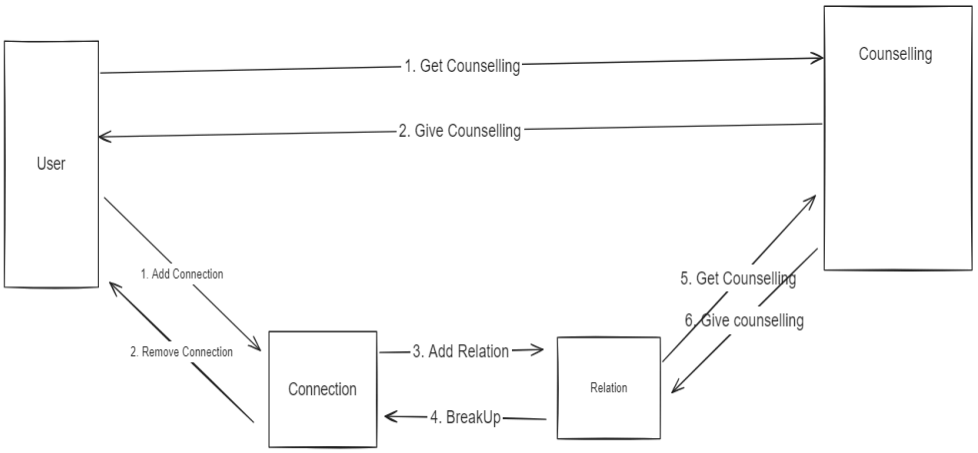


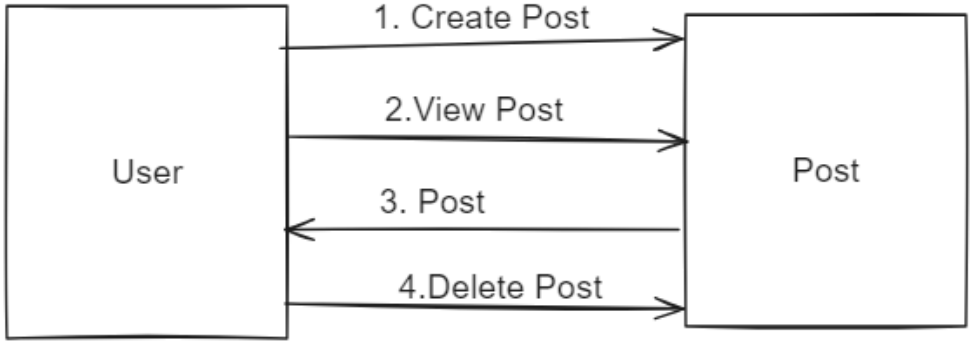




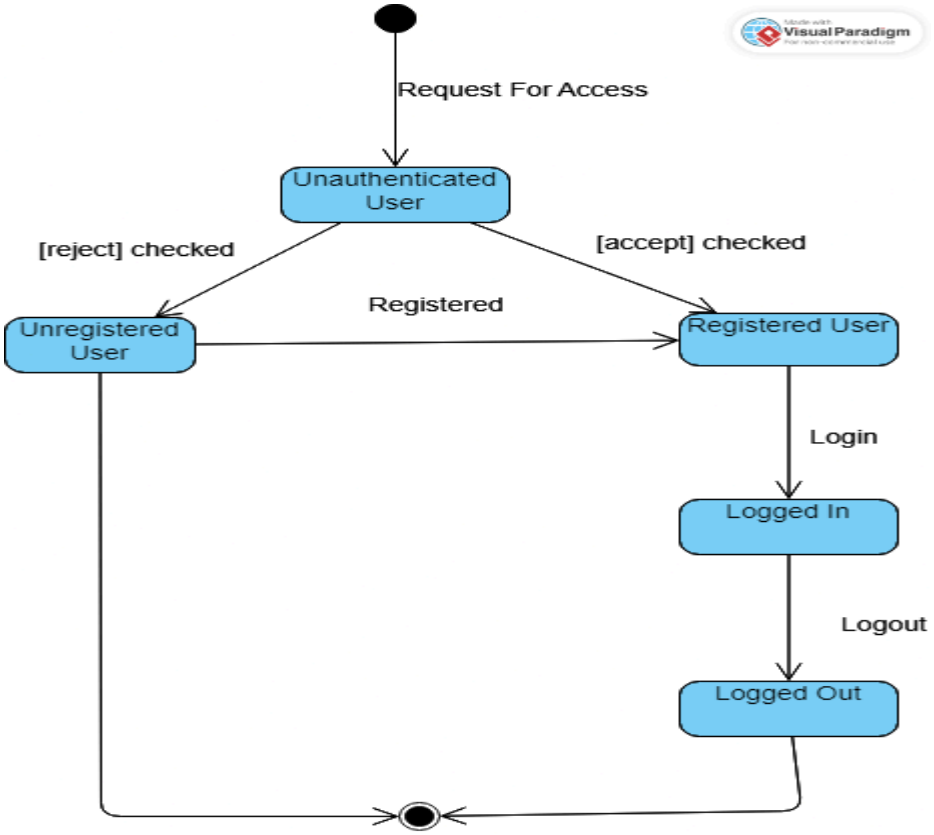
Collaboration Diagram

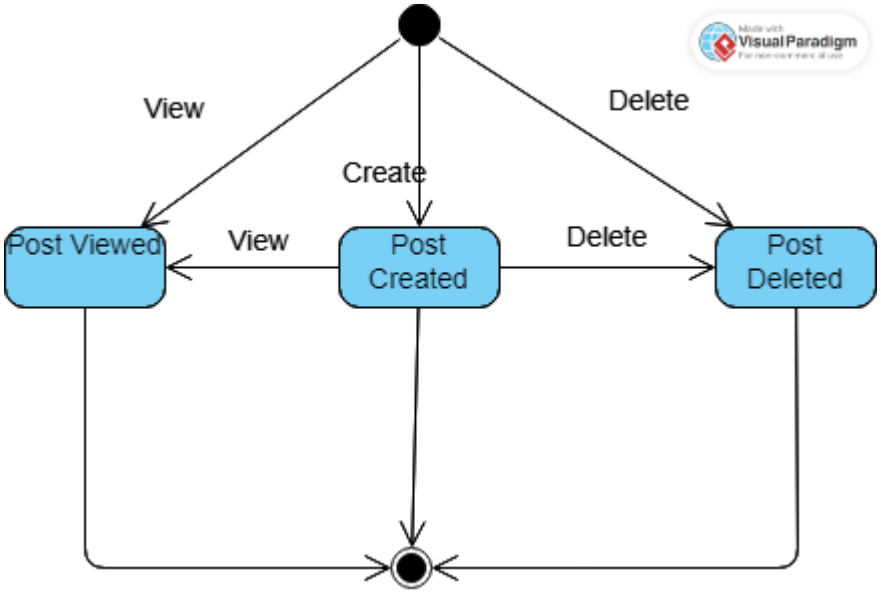


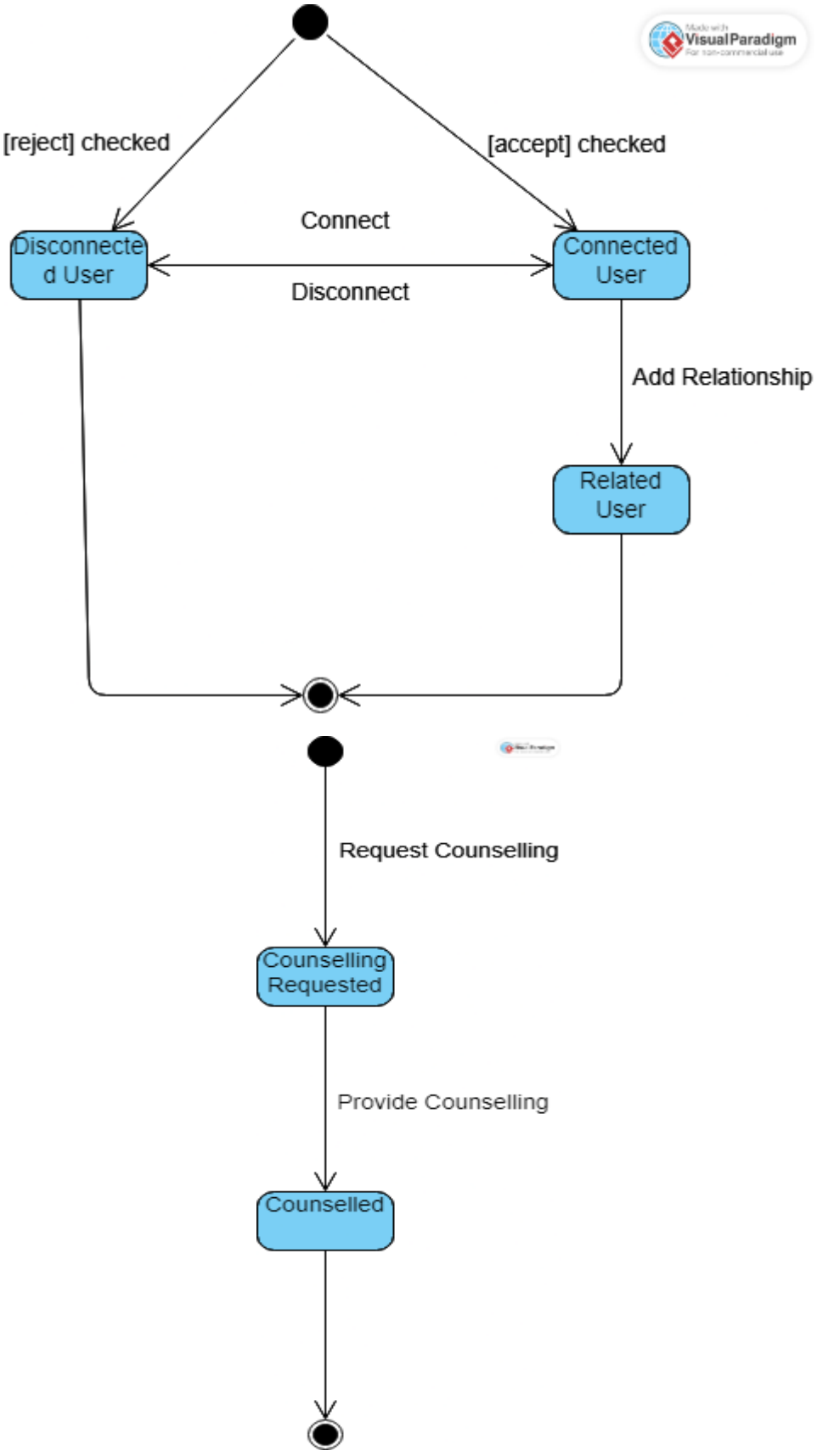

User
Counselling
1. Get Counselling
2. Give Counselling
1. Add Connection
2. Remove Connection
Connection
3. Add Relation
4. BreakUp
Relation
5. Get Counselling
6. Give counselling


User
1. Create Post
2.View Post
3. Post
4.Delete Post
Post

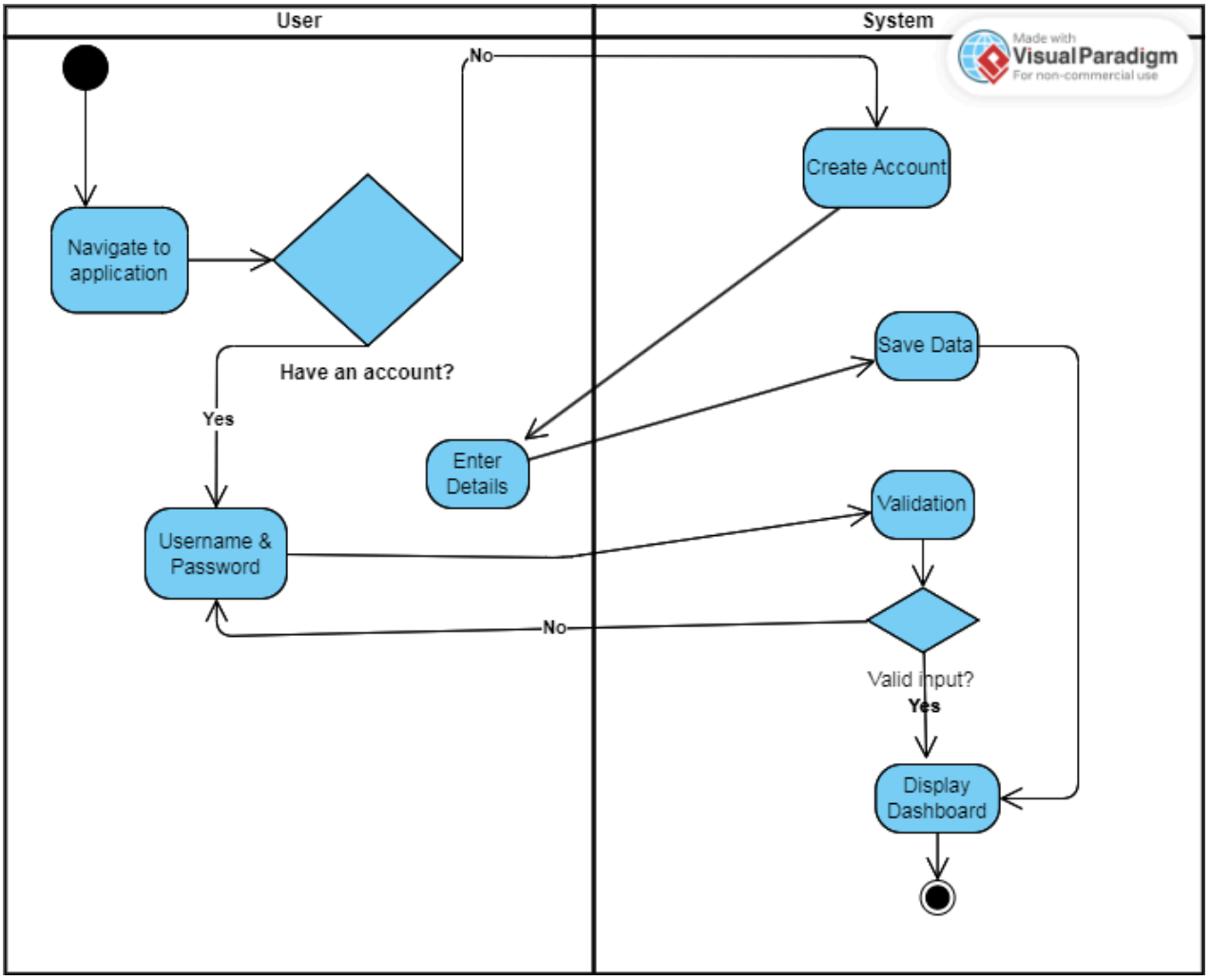
State Diagram

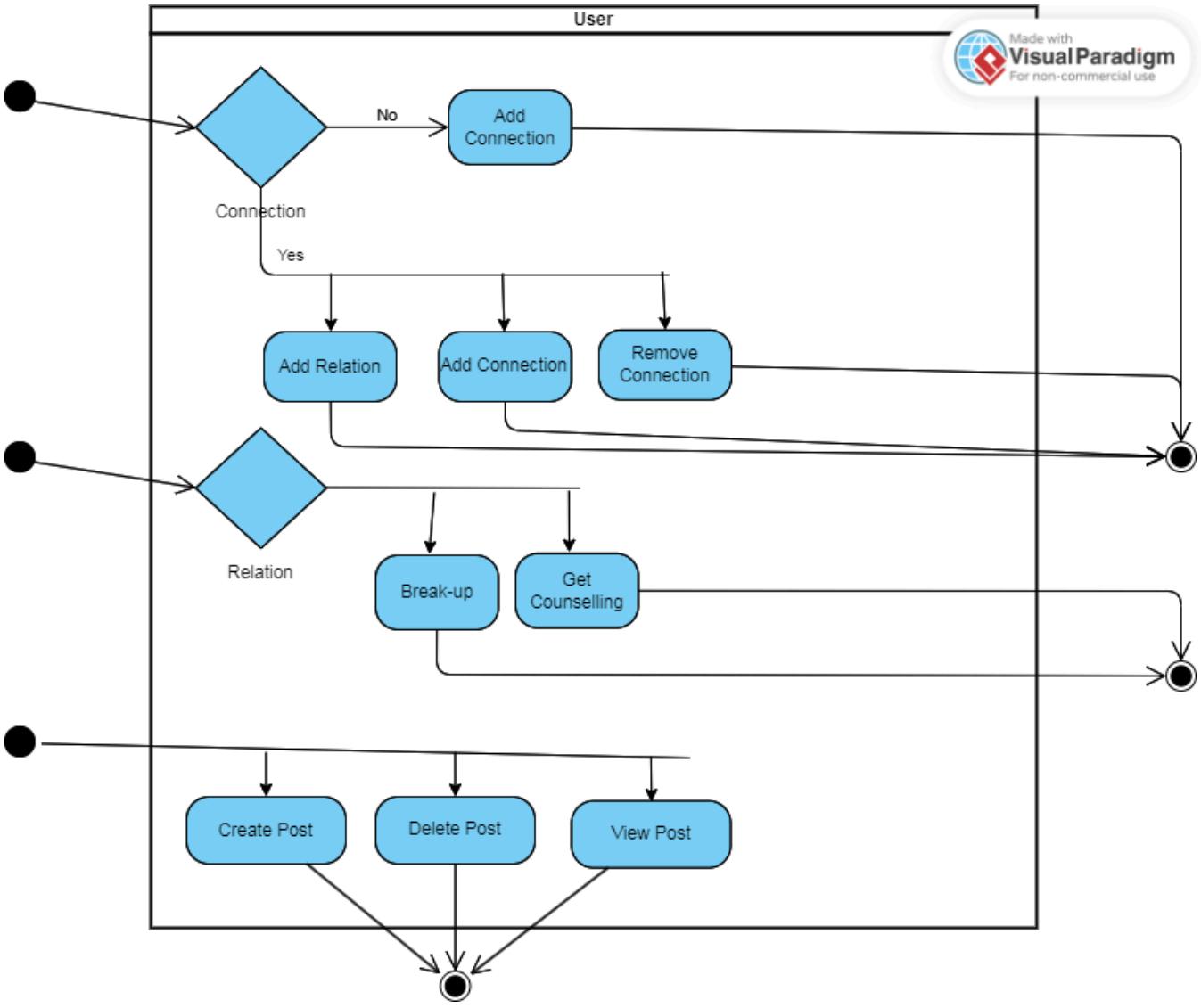





[reject] checked
[accept] checked
Connect
Disconnecte d User
Connected User
Disconnect
Add Relationship
Related User
Request Counselling
Counselling Requested
Provide Counselling
Counselled

Activity Diagram




User
No
Add Connection
Connection
Yes
Add Relation
Add Connection
Remove Connection
Relation
Break-up
Get Counselling
Create Post
Delete Post
View Post
Made with Visual Paradigm
For non-commercial use

Component Diagram