PRACTICAL FILE

SESSION: 2023-24

Software Engineering Lab
(CIC 357)

III Year, V Sem

Submitted to:

Name: Dr Archana Sharma
Designation: Associate Professor

Submitted by:

Name: Adityan Verma
Enrollment No. 07818002721

Department of Computer Science and Engineering
Delhi Technical Campus, Greater Noida

INDEX

S.NO.	PROGRAM NAME	DATE OF EXPERIMENT	DATE OF SUBMISSION	SIGN.
1	Write down the problem statement for a suggested system of relevance.			
2	Do requirement analysis and develop Software Requirement Specification Sheet (SRS) for suggested system.			
3	To perform the function-oriented diagram: Data Flow Diagram (DFD) and Structured chart			
4	To perform the user's view analysis for the suggested system: Use case diagram.			
5	To draw the structural view diagram for the system: Class diagram, object diagram.			
6	To draw the behavioural view diagram: State-chart diagram, Activity diagram.			
7	To perform the behavioural view diagram for the suggested system: Sequence diagram, Collaboration diagram.			
8	To perform the implementation view diagram: Component diagram for the system.			
9	Perform Estimation of effort using FP Estimation for chosen system.			
10	To prepare time line chart/Gantt Chart/PERT Chart for selected software project.			

EXPERIMENT 1

AIM: Write down the problem statement for a suggested system of relevance.

REAL-TIME CHAT APPLICATION

➤ **Problem Statement:**

The absence of a robust and user-friendly Real-time Chat Application limits efficient and effective communication, hindering swift information exchange, teamwork, and real-time decision-making processes among individuals and groups

In real-time chat applications, users often experience difficulty finding relevant conversations and messages amidst a large volume of chats. This can lead to frustration, inefficiency, and a lack of engagement. To address this issue, a system is needed that can effectively identify and present relevant content to users based on their interests, context, and preferences.

Contextualizing the Problem:

The existing landscape of real-time chat applications lacks efficiency and user-friendliness, impeding smooth information exchange, collaborative efforts, and timely decision-making processes for both individuals and groups.

Describing the Issue:

Users encounter difficulties in navigating through a vast volume of chats, leading to inefficiencies, frustration, and a lack of engagement. Relevant conversations and messages become challenging to identify within the cluttered interface, hampering the overall communication experience.

Relevance of the Problem:

This issue significantly impacts productivity, teamwork, and user satisfaction. Without an effective solution, users remain burdened by the overwhelming volume of messages, hindering their ability to locate and engage with pertinent content swiftly

Name: Adityan Verma   **Enrolment No.** 07818002721
.

Suggested System:

System of relevance for a real-time chat application should incorporate the following features:

1. **Content-based relevance:** Analyse the content of messages to identify keywords, topics, and entities that align with user interests.

2. **Context-based relevance:** Consider the context of conversations, including participants, timestamps, and chat channels, to determine the relevance of messages.

3. **User-based relevance:** Personalize the relevance algorithm based on user preferences, such as frequently used keywords, saved messages, and followed users.

4. **Real-time relevance:** Continuously evaluate the relevance of messages as new information is added to the chat application.

5. **Adaptive relevance:** Continuously refine the relevance algorithm based on user feedback and engagement metrics.

By incorporating these features, a suggested system of relevance can significantly improve the user experience in real-time chat applications, enabling users to find and engage with the most relevant content effortlessly.

Objectives of the Research:

The primary research objectives entail:

- Designing a real-time chat application that addresses the challenge of identifying and presenting relevant content to users.

- Incorporating features such as content-based, context-based, and user-based relevance algorithms to streamline message identification.

- Implementing real-time and adaptive relevance mechanisms to continuously enhance user experience and engagement.

➢ **The 5 'W's:**

- **Who:** Individuals and groups relying on real-time chat applications for communication and collaboration.

- **What:** The issue revolves around the inefficiency in identifying relevant content within the chat interface.

- **When:** The issue occurs whenever users engage in communication and information exchange through the application.

- **Where:** The issue affects the usability and effectiveness of chat applications across various devices and platforms.

- **Why:** It's crucial to solve this problem to enhance productivity, streamline communication, and boost user engagement within real-time chat applications.

EXPERIMENT 2

AIM: Do requirement analysis and develop Software Requirement Specification Sheet (SRS) for suggested system.

Software Requirement Specification Sheet (SRS)

Project: Real-time Chat Application

1.   INTRODUCTION

1.1   PURPOSE OF THE DOCUMENT

The primary objective of this document is to establish a clear and comprehensive set of guidelines and specifications for the development of a Real-time Chat Application. It aims to provide a detailed outline of the system's requirements, functionalities, and constraints to ensure a common understanding among all stakeholders involved in the project.

1.2   SCOPE OF THE DOCUMENT

This document delineates the scope of the Real-time Chat Application, encompassing its functionalities, user classes, system interfaces, and operational constraints. It serves as a reference point for the development team, aiding in the creation of a robust and user-friendly communication platform.

1.3   ACRONYMS

- **SRS:** Software Requirement Specification
- **FRS:** Functional Requirement Specification

1.4   INTENDED AUDIENCE AND READING SUGGESTIONS

The intended audience for this document includes the development team, stakeholders, project managers, and quality assurance teams involved in the creation and validation of the Real-time Chat Application.

- **Development Team:** To guide the development process.

- **Stakeholders:** To understand the project's scope and requirements.

- **Project Managers:** To ensure alignment with project objectives.

- **Quality Assurance Teams:** To validate system compliance.

- **Documentation Writers:** Involved in creating user manuals, technical documentation, and guides.

- **Marketing Staff:** Gaining an understanding of the application's features and capabilities for promotional purposes.

- **Users (End-Users):** Understanding the functionalities and capabilities of the chat application.

1.5   DOCUMENT OVERVIEW

This document is structured to provide a comprehensive understanding of the Real-time Chat Application. It is divided into sections that cover various aspects of the system, starting with an overall description, external interfaces, functional and non-functional requirements, use cases, and other essential details crucial for successful system development and deployment.

2.   OVERALL DESCRIPTION

2.1   PRODUCT PERSPECTIVE

The Real-time Chat Application is designed as a standalone, robust communication platform that allows users to engage in instantaneous messaging, multimedia sharing, and group discussions. It operates as an independent system, leveraging modern technologies for real-time communication while maintaining compatibility across various devices and platforms.

2.2   PRODUCT FUNCTIONS

The application serves as a multifunctional platform with the following core features:

1. **Registration and Login:** Allows users to create accounts and authenticate securely.

2. **Profile Page:** Enables users to manage credentials and avatars.

3. **User Classes:** Supports various user roles, including Students, Professors, Super Admin, and Moderators.

4. **Chat Rooms:** Provides dedicated spaces for group discussions and interactions.

5. **Online Status Feature:** Indicates users' online presence for real-time communication.

6. **Search Functionality:** Facilitates quick and efficient search within chats and messages.

7. **Emoji Pickers and Reaction System:** Allows users to express reactions using emoji during conversations.

8. **Light & Dark Mode:** Offers interface customization for user preference.

2.3   USER CLASSES AND CHARACTERISTICS

The Real-time Chat Application caters to distinct user classes within the college ecosystem, each with specific roles and characteristics:

Students:

- Engage in academic discussions related to courses, projects, and extracurricular activities.
- Access dedicated chat rooms for specific classes, interest groups, or clubs.
- Collaborate on group assignments, share resources, and seek academic support.

Professors:

- Facilitate teaching, mentorship, and academic discussions.
- Create and moderate chat rooms for individual courses or study groups.
- Provide guidance, share resources, and interact with students outside of class hours.

Super Admin:

- Oversee administrative functions within the application, ensuring smooth operations.
- Manage user roles, permissions, and system configurations.
- Address technical issues and provide support as needed.

Moderators:

- Assist in maintaining decorum and relevance within specific chat rooms.
- Monitor discussions, ensuring adherence to college guidelines and academic integrity.
- Support users, resolve conflicts, and escalate issues when necessary.

The Real-time Chat Application is tailored to enhance communication and collaboration within the college community, fostering an environment conducive to academic engagement, resource sharing, and cohesive interaction among students, professors, and administrative staff.

2.4   OPERATING ENVIRONMENT

The application is built using modern web technologies such as React, Tailwind, and utilizes Firebase as a Backend-as-a-Service (BaaS). The backend is implemented using Node/Express and MongoDB, with specific data models for chats, chat lines, messages, seen, reactions, and file relevance.

2.5   DESIGN AND IMPLEMENTATION CONSTRAINTS

The design focuses on a responsive and intuitive user interface to ensure ease of use across various devices and screen sizes. Additionally, the system's scalability and performance are key considerations in its design and implementation.

2.6   USER DOCUMENTATION

Comprehensive user documentation will be provided to guide users through the application's functionalities and features, ensuring an optimal user experience.

2.7   ASSUMPTIONS AND DEPENDENCIES

The successful deployment and operation of the Real-time Chat Application are dependent on stable internet connectivity and adherence to modern web standards across supported browsers and devices.

3.   EXTERNAL INTERFACE REQUIREMENTS

3.1   USER INTERFACES

The Real-time Chat Application is designed as a standalone, robust communication platform that allows users to engage in instantaneous messaging, multimedia sharing, and group discussions. It operates as an independent system, leveraging modern technologies for real-time communication while maintaining compatibility across various devices and platforms.

3.2   HARDWARE INTERFACES

The Real-time Chat Application does not necessitate specific hardware requirements beyond standard computing devices, ensuring accessibility across a wide range of devices, including computers, laptops, tablets, and smartphones.

3.3   SOFTWARE INTERFACES

The application integrates the following software components and interfaces:

- **Frontend Technologies:** Utilizes React and Tailwind for the web and mobile interfaces.

- **Backend Framework:** Implemented with Node.js/Express.js for server-side operations.

- **Database Management:** Relies on MongoDB for data storage, utilizing two distinct models: chatRoom and chatMessage.

- **Real-time Communication:** Employs Socket.IO for real-time data exchange and communication.

3.4   COMMUNICATION INTERFACES

The Real-time Chat Application requires stable internet connectivity to facilitate real-time communication between users. It operates over standard HTTP/HTTPS protocols for secure data transmission and Web-Socket connections for real-time messaging functionality.

4.   FUNCTIONAL REQUIREMENT SPECIFICATIONS (FRS)

4.1   SYSTEM FEATURES

The Real-time Chat Application encompasses the following key features:

4.1.1 Registration and Login

- **Description:** Users can create accounts and log in securely to access the application.
- **Requirements:**
    - User registration with valid credentials.
    - Secure login mechanism with authentication protocols.

4.1.2 Profile Page

- **Description:** Users can manage their credentials and avatars within their profiles.
- **Requirements:**
    - Profile editing options for credentials.
    - Avatar uploading and customization features.

4.1.3 User Classes

- **Description:** The application supports various user roles with distinct privileges.
- **Requirements:**
    - Role-based access control for Students, Professors, Super Admin, and Moderators.

4.1.4 Chat Rooms

- **Description:** Dedicated spaces for group discussions and interactions.
- **Requirements:**
    - Creation and management of chat rooms.
    - Access control and membership management within rooms.

4.1.5 Online Status Feature

- **Description:** Indicates users' online presence for real-time communication.
- **Requirements:**
    - Online/offline status indicators for users.

4.1.6 Search Functionality

- **Description:** Facilitates quick and efficient search within chats and messages.
- **Requirements:**
    - Search feature across chat rooms and messages.

4.1.7 Emoji Pickers and Reaction System

- **Description:** Allows users to express reactions using emoji during conversations.
- **Requirements:**
    - Emoji picker for message reactions.
    - Reaction system implementation within chats.

4.1.8 Light & Dark Mode
- **Description:** Offers interface customization for light and dark mode preferences.
- **Requirements:**
    - Toggle between light and dark modes for user interface.

4.2   FUNCTIONAL REQUIREMENTS

4.2.1 Front end (Storefront) Requirements

- Detailed functional requirements for the user interface, ensuring usability and functionality across platforms.

4.2.2 Back end (Administrative Tools) Requirements

- Specific backend functional requirements, including server-side operations, database interactions, and real-time communication protocols.

4.2   USE CASES

4.3.1 Front end (Storefront)

- Illustrative use cases detailing user interactions and functionalities from a frontend perspective.

4.3.2 Back end (Administrative Tools)

- Use cases outlining backend functionalities and operations for administrative and system-level tasks.

5.   NON-FUNCTIONAL REQUIREMENT

5.1   USABILITY REQUIREMENTS

The Real-time Chat Application must ensure usability in the following aspects:

- **Intuitiveness:** The user interface should be intuitive and easy to navigate.

- **Accessibility:** Accessibility standards should be met for users with disabilities.

- **Consistency:** Ensure consistent user experience across different devices and platforms.

- **Learning Curve:** Users should be able to quickly adapt to the application's functionalities.

5.2   PERFORMANCE REQUIREMENTS

The application should meet the following performance criteria:

- **Responsiveness:** Messages and chat room loading should be near-instantaneous.

- **Scalability:** The system should handle a growing user base without performance degradation.

- **Resource Utilization:** Optimize resource usage to ensure efficient operation across devices.

- **Minimum Downtime:** Ensure high availability with minimal system downtimes.

5.3   COMPATIBILITY REQUIREMENTS

The Real-time Chat Application should exhibit compatibility with various platforms and environments:

- **Cross-Platform Compatibility:** Ensure seamless operation across different operating systems and devices.

- **Browser Compatibility:** Support major web browsers and versions without functional discrepancies.

- **Device Adaptability:** Adapt the application interface to different screen sizes and resolutions.

5.4   USABILITY REQUIREMENTS

The application must adhere to robust security measures to safeguard user data and privacy:

- **Data Encryption:** Implement end-to-end encryption for secure message transmission.

- **Authentication and Authorization:** Employ secure authentication mechanisms and role-based access control.

- **Secure Storage:** Ensure encryption and secure storage of user data and credentials.

- **Protection against Threats:** Implement measures to mitigate risks from cyber threats like XSS, CSRF, etc.

5.5   RELIABILITY REQUIREMENTS

The application should exhibit reliability in terms of:

- **Stability:** Ensure consistent and stable operation without frequent crashes or errors.

- **Fault Tolerance:** Ability to recover gracefully from unexpected failures or system interruptions.

- **Data Integrity:** Maintain data integrity and consistency across the platform.

6.   OTHER REQUIREMENT

6.1   LEGAL AND COMPLIANCE

- **Data Protection Regulations:** Ensure compliance with relevant data protection laws and regulations.

- **Intellectual Property:** Respect intellectual property rights regarding content shared within the application.

- **Terms of Service:** Specify user guidelines and terms of service within the application.

6.2   DOCUMENTATION AND TRAINING

- **User Manuals:** Provide comprehensive user documentation explaining application features and functionalities.

- **Training Materials:** Develop training resources for new users to familiarize themselves with the application.

6.3   MAINTENANCE AND SUPPORT

- **Maintenance Schedule:** Outline regular maintenance plans and update schedules for the application.

- **Technical Support:** Offer user support channels and response times for technical issues.

6.4   PERFORMANCE METRICS AND MONITORING

- **Monitoring Tools:** Implement tools to monitor application performance and user interactions.

- **Performance Metrics:** Define key performance indicators (KPIs) to assess application performance.

6.5   BACKUP AND RECOVERY

- **Data Backup:** Ensure regular backup of user data to prevent loss in case of system failures.

- **Recovery Protocols:** Establish recovery procedures to restore system functionality in case of data loss.

6.6   USER FEEDBACK AND IMPROVEMENT

- **Feedback Mechanism:** Incorporate a user feedback system to gather suggestions and improvements.

- **Iterative Development:** Plan for iterative updates and improvements based on user feedback.

Name: Adityan Verma   **Enrolment No.** 07818002721

EXPERIMENT 3

AIM: To perform the function-oriented diagram: Data Flow Diagram (DFD) and Structured chart

Data Flow Diagram (DFD):

Level 0 DFD (Context Diagram)

The Level 0 DFD gives an overview of the system, showing interactions between the system and external entities such as users and databases.

- **Processes:**
    - Main system (Chat Application)

- **Entities:**
    - Users
    - Chat Rooms
    - Databases
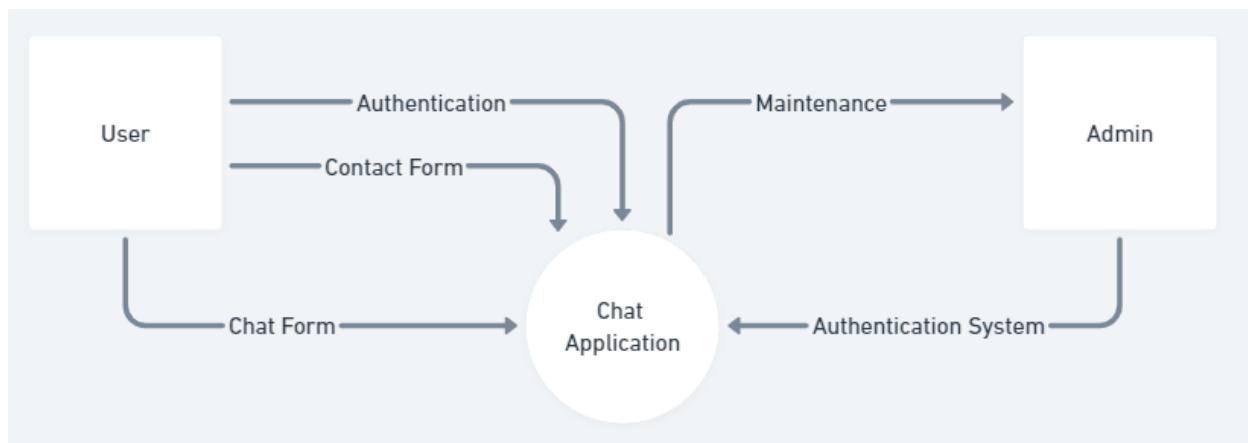
Figure: *Real-time Chat Application level 0 Data Flow Diagram*

Level 1 DFD (Detailed Diagram)

The Level 1 DFD delves deeper into the system, breaking down processes and data flow within the system.

- **Processes:**
  - User Login/Logout
  - Profile Management
  - Chat Room Creation
  - Message Sending/Receiving
  - Search Functionality
  - Emoji Reactions

- **Data Stores:**
  - User Profiles
  - Chat Logs

- **Data Flows:**
  - User login details flow to authentication process
  - Messages flow from sender to receiver through the chat system

Structured Chart

A Structured Chart provides a hierarchical breakdown of system modules or functions.

- ➢ Main Module: Real-time Chat Application

  - Submodules:

    - User Authentication
    - Profile Management
    - Chat Room Management
    - Message Handling
    - Search Functionality
    - Reaction System

Each submodule can further break down into smaller functions or processes.

Name: Adityan Verma   **Enrolment No.** 07818002721
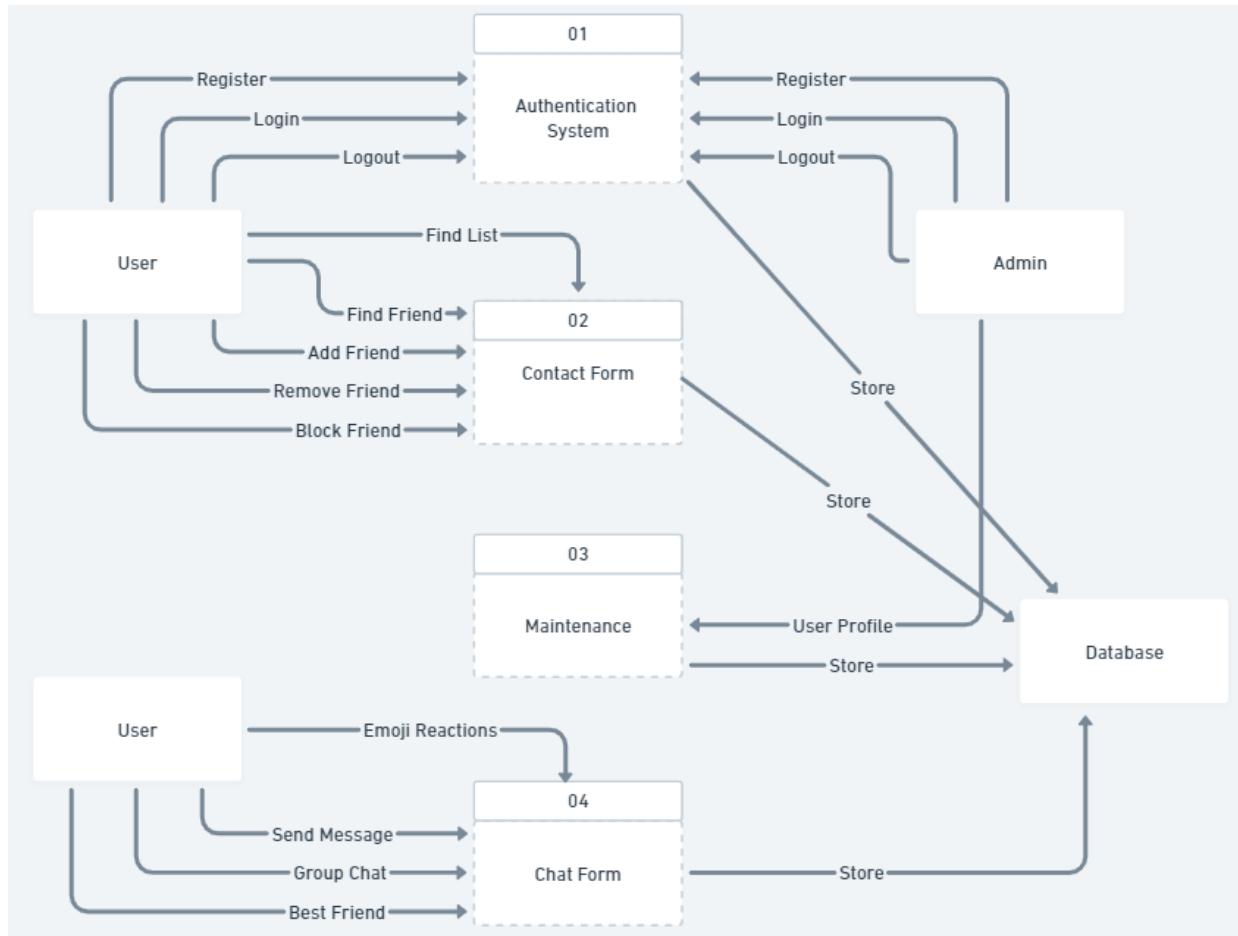
Figure: Real-time Chat Application level 1 Data Flow Diagram

Name: Adityan Verma

Enrolment No. 07818002721

ER Diagram for Real-time Chat Application:
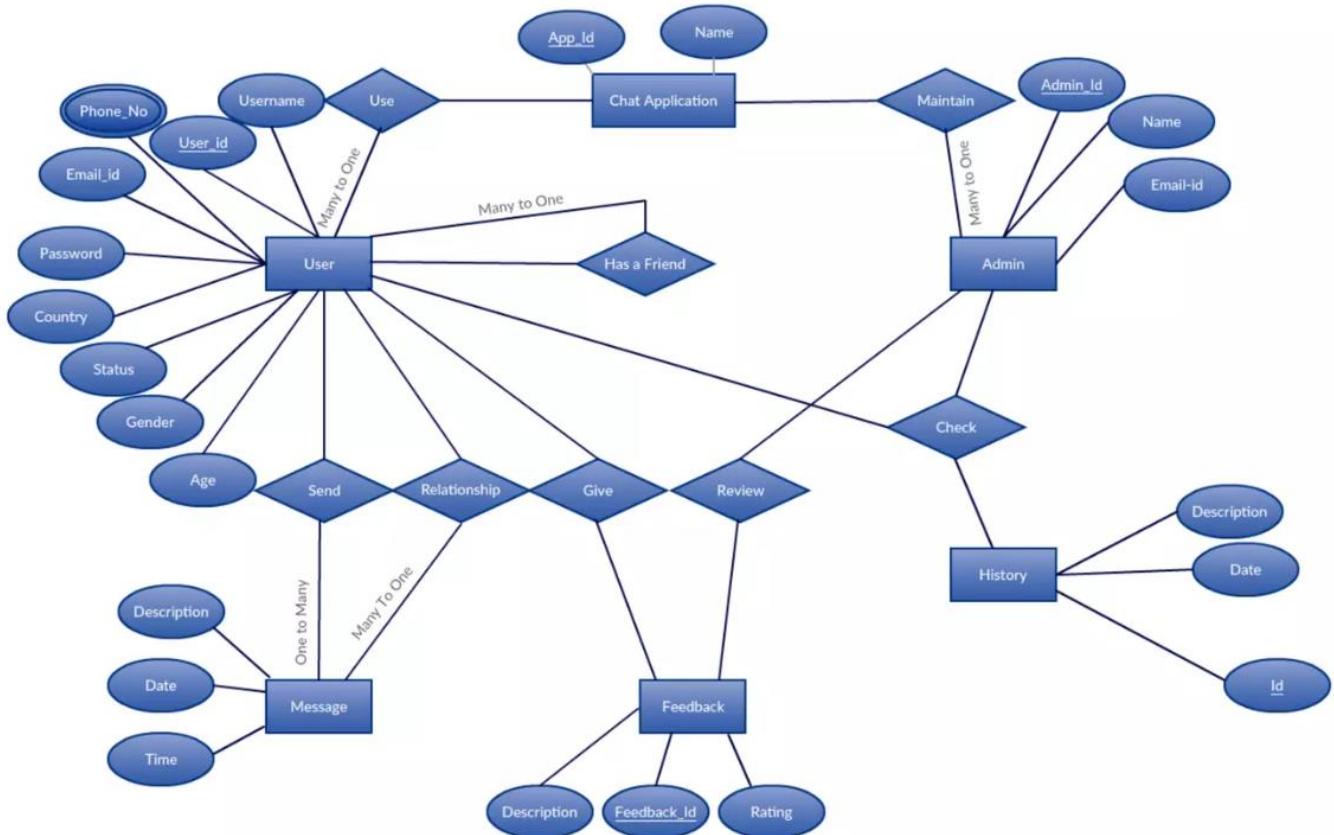
Figure: Real-time Chat Application Entity Relationship (ER) Diagram

Conclusion:

1. Level 0 and Level 1 Data Flow Diagram (DFD) for Real-time Chat Application have been drawn Successfully.
2. Structure Chart for Real-time Chat Application have been Drawn Successfully

EXPERIMENT 4

AIM: To perform the user's view analysis for the suggested system: Use case diagram.

Use Case Diagram:

1. Use Case Table

Figure: *Use Case table for Real-time Chat Application*
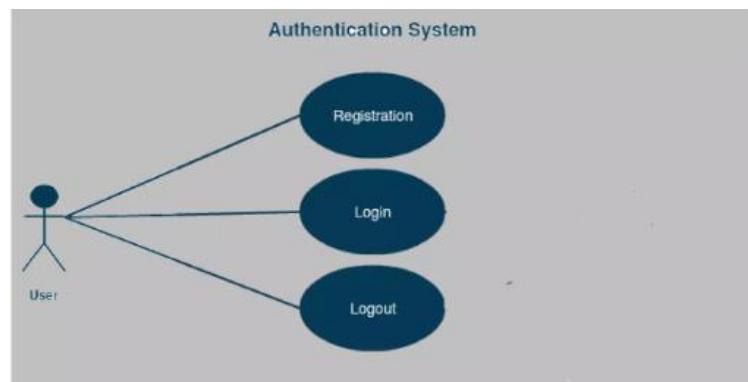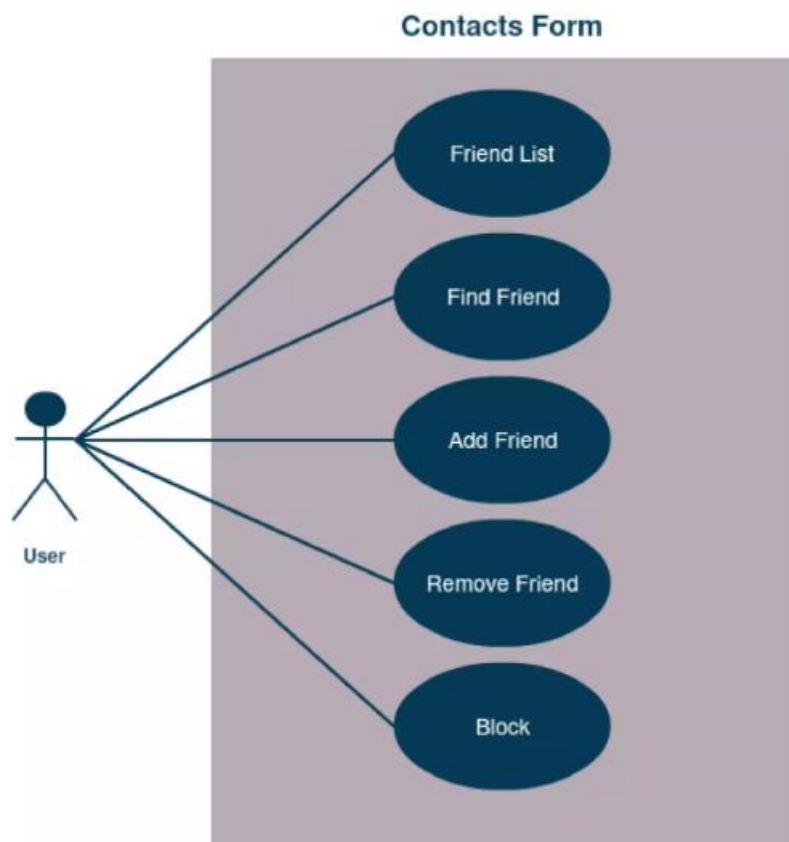
4. Authentication System

5. Contacts Form
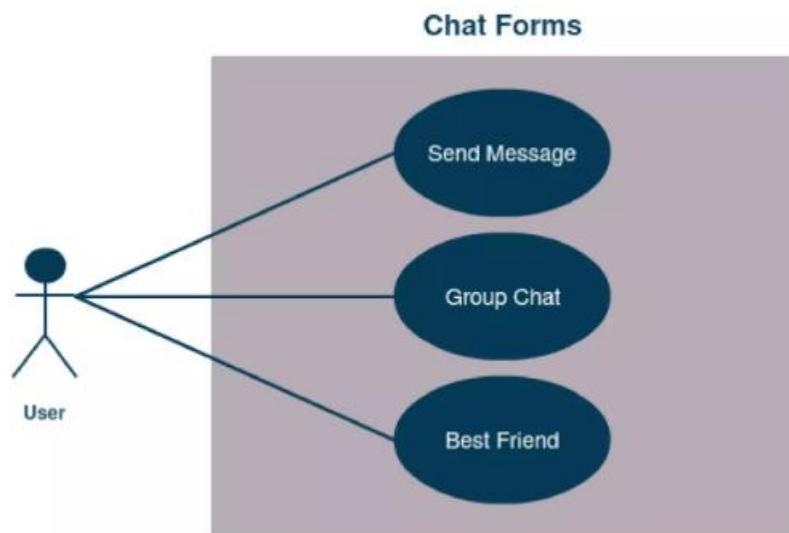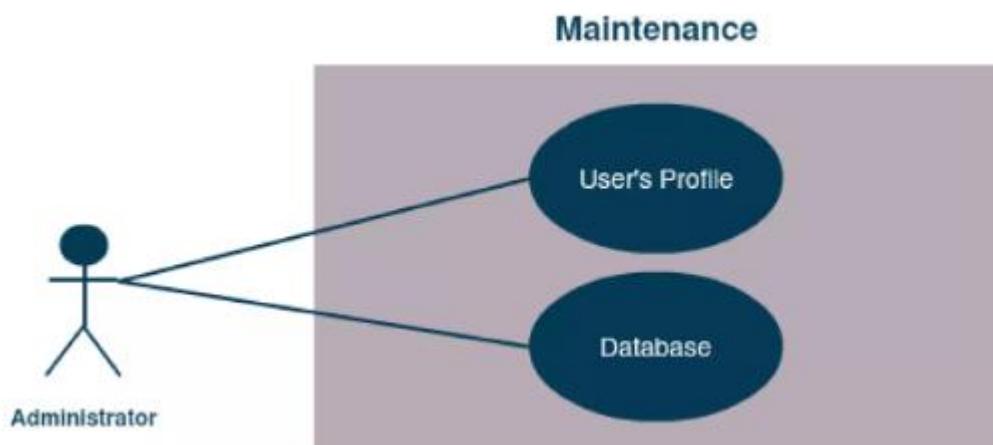
Name: Adityan Verma   **Enrolment No.** 07818002721

2. Chat Form

3. Maintenance
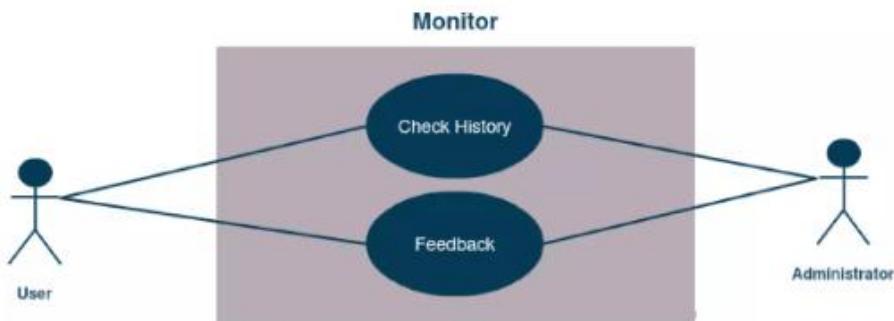
Name: Adityan Verma   **Enrolment No.** 07818002721

6. Monitor

Conclusion:

Use case diagram for Real-time Chat Application has been drawn successfully.

Name: Adityan Verma   **Enrolment No.** 07818002721

EXPERIMENT 5

AIM: To draw the structural view diagram for the system: Class diagram, object diagram.

Class Diagram:
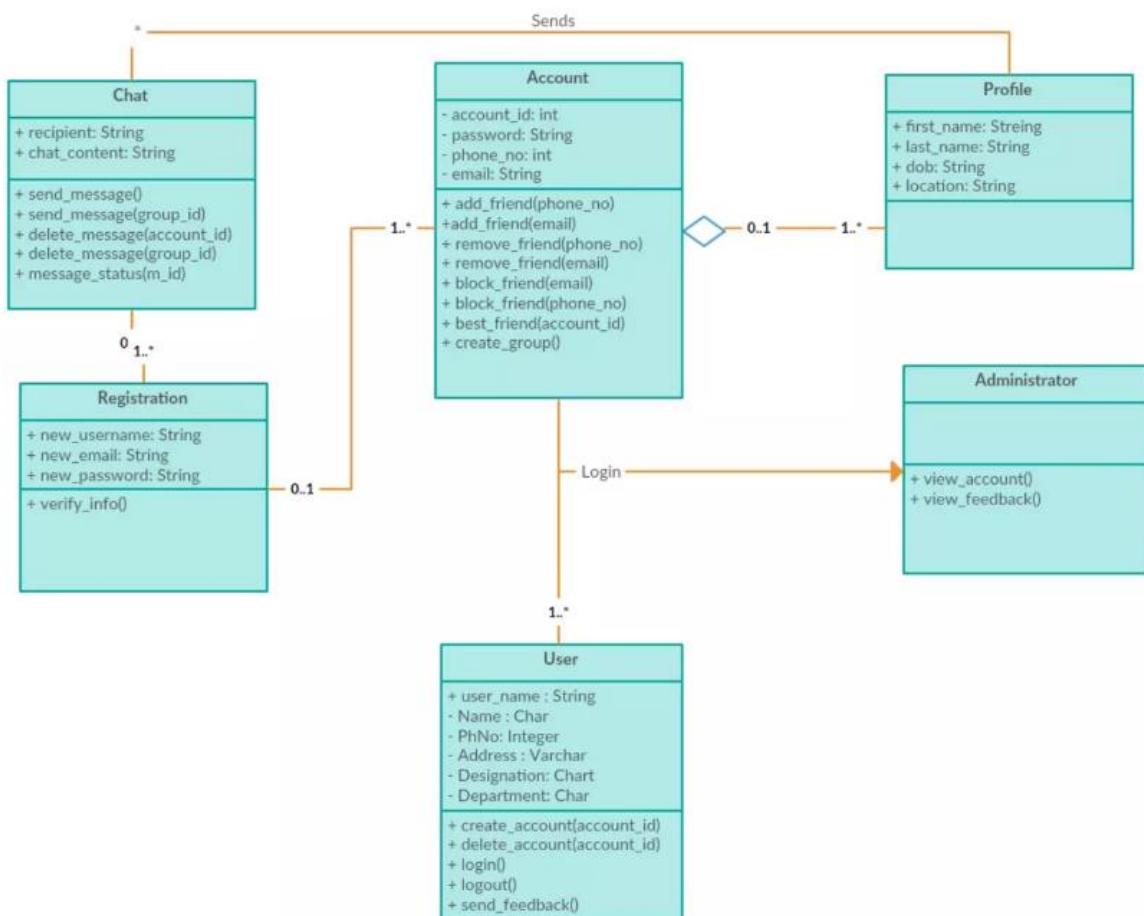
Figure: *Class Diagram for Real-time Chat Application*

Name: Adityan Verma   **Enrolment No.** 07818002721

Object Diagram:
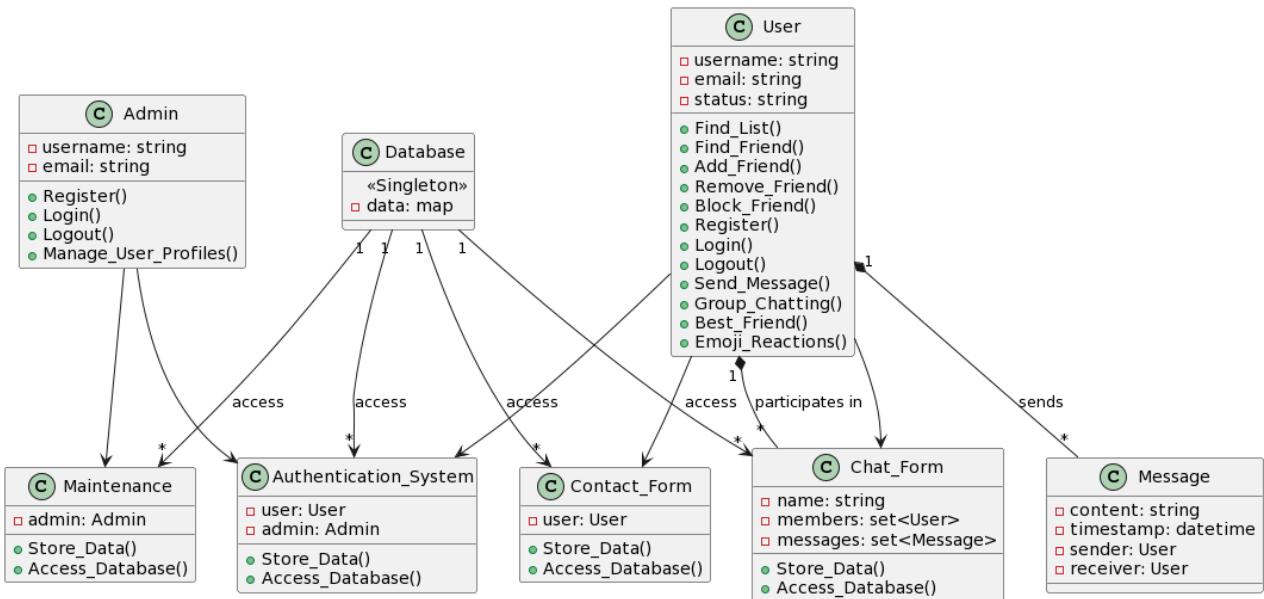
Figure: *Object Diagram for Real-time Chat Application*

Conclusion:

Class diagram and object Diagram for Real-time Chat Application has been done successfully.

Name: Adityan Verma   **Enrolment No.** 07818002721

EXPERIMENT 6

AIM: To draw the behavioural view diagram: State-chart diagram, Activity diagram.

State-Chart Diagram:
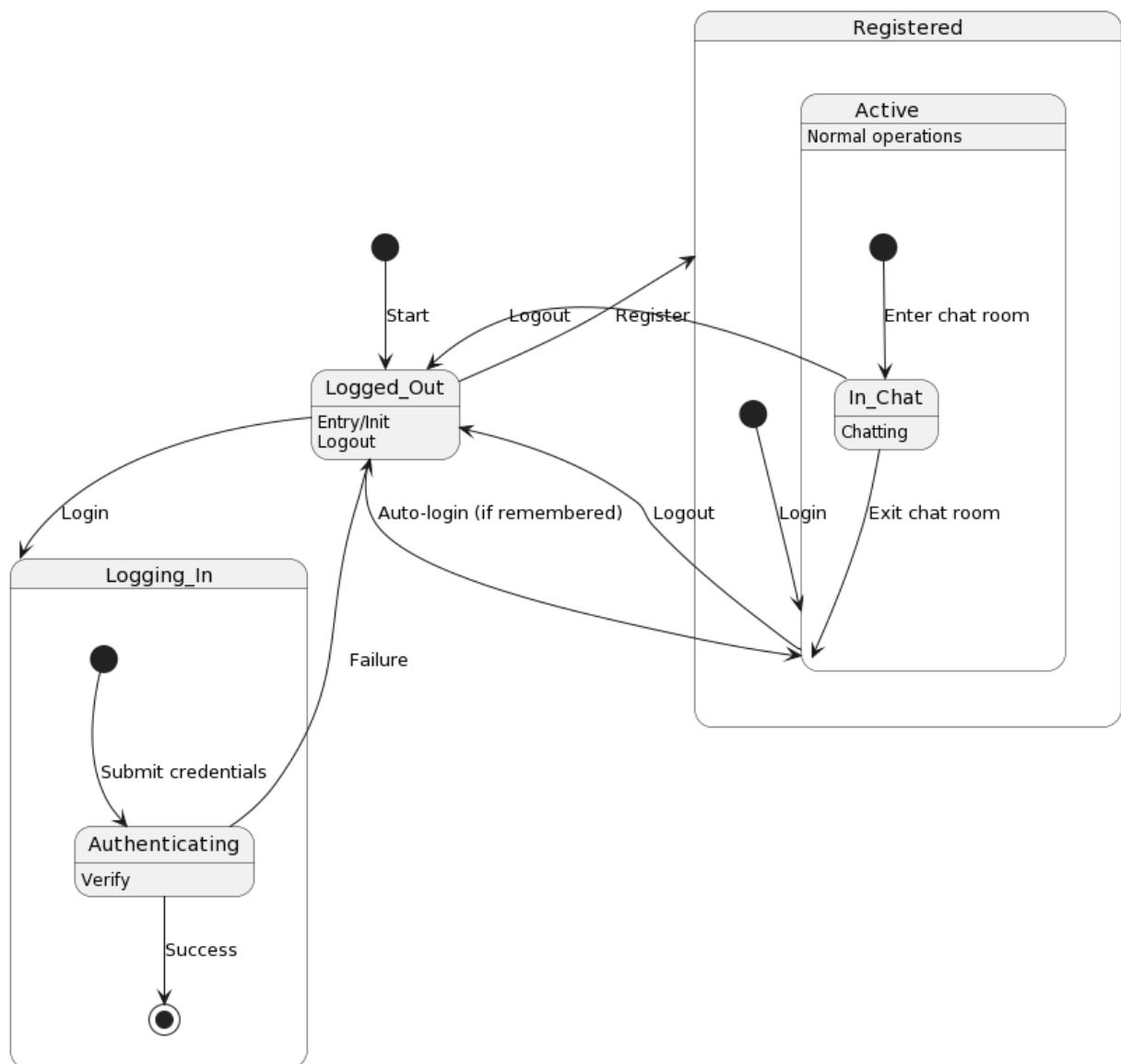
Figure: *State-Chart Diagram for Authentication System, Maintenance, Monitor*

Name: Adityan Verma   **Enrolment No.** 07818002721

Activity Diagram:

1. Authentication System, Maintenance, Monitor
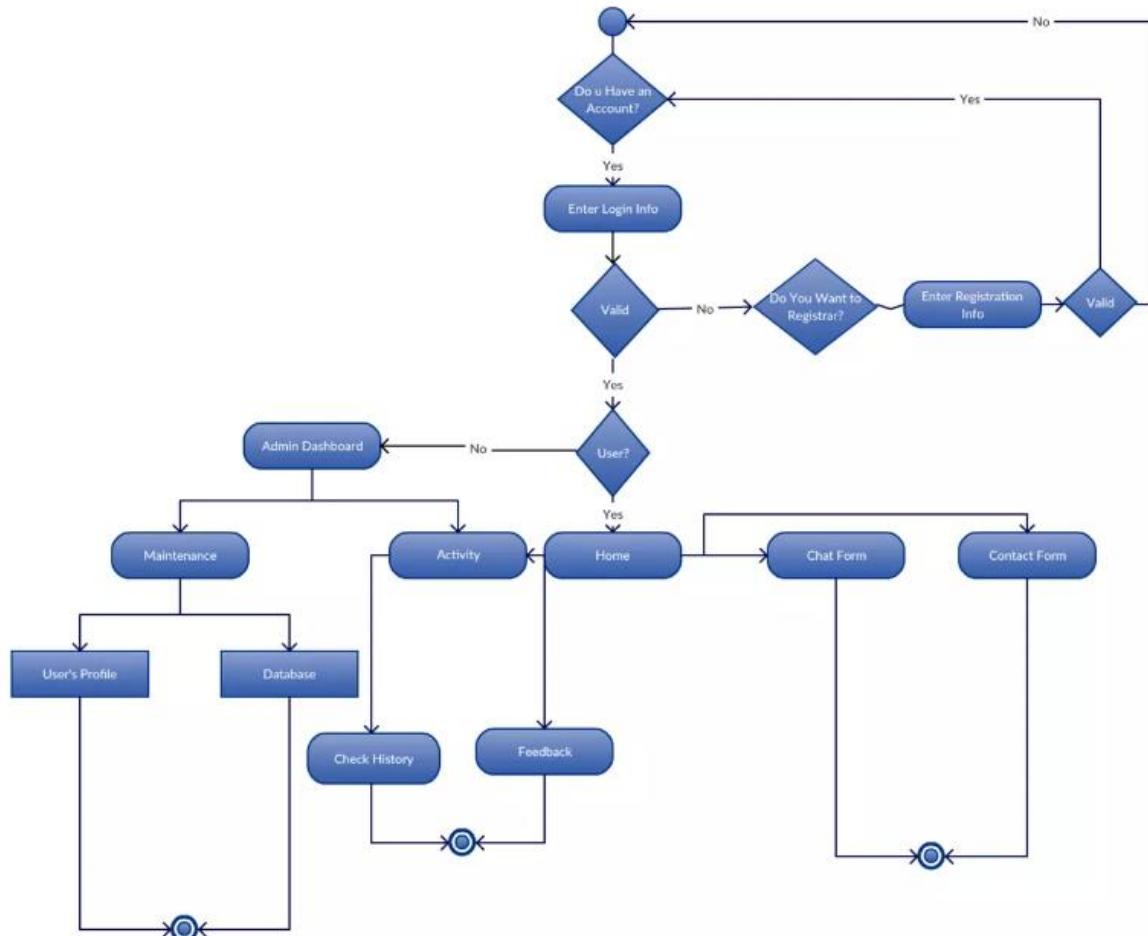
Figure: Activity Diagram for Authentication System, Maintenance, Monitor

2. Contact Form, Chat Form
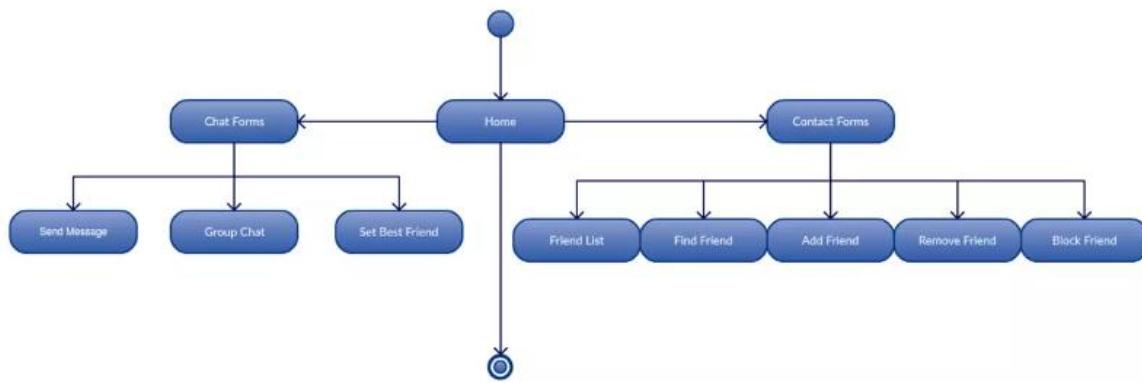
Figure: *Activity Diagram for Contact Form, Chat Form*

Conclusion:

State-Chart diagram and Activity diagram for Real-time Chat Application has been done successfully.

Name: Adityan Verma   **Enrolment No.** 07818002721

EXPERIMENT 7

AIM: To perform the behavioural view diagram for the suggested system: Sequence diagram, Collaboration diagram.

Sequence Diagram:
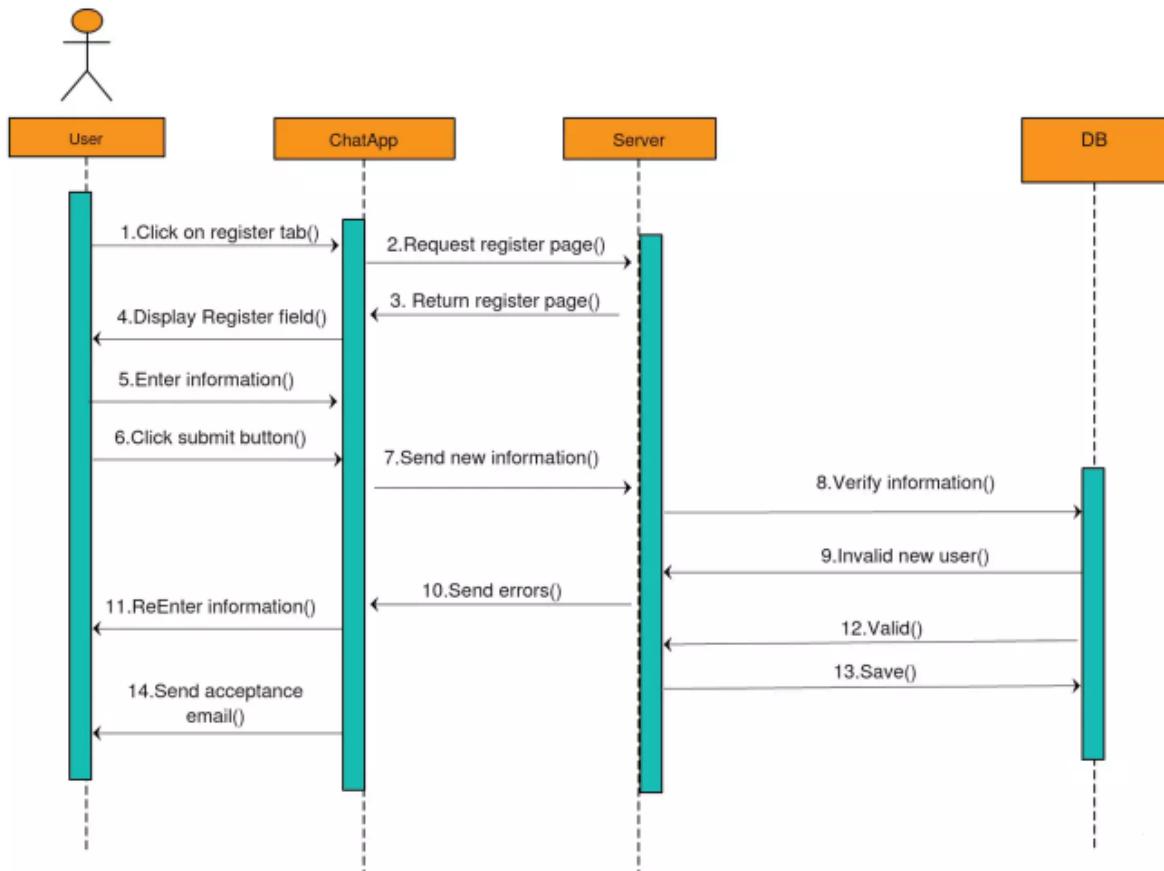
Figure: Sequence Diagram for Registration Functionality

Name: Adityan Verma   **Enrolment No.** 07818002721
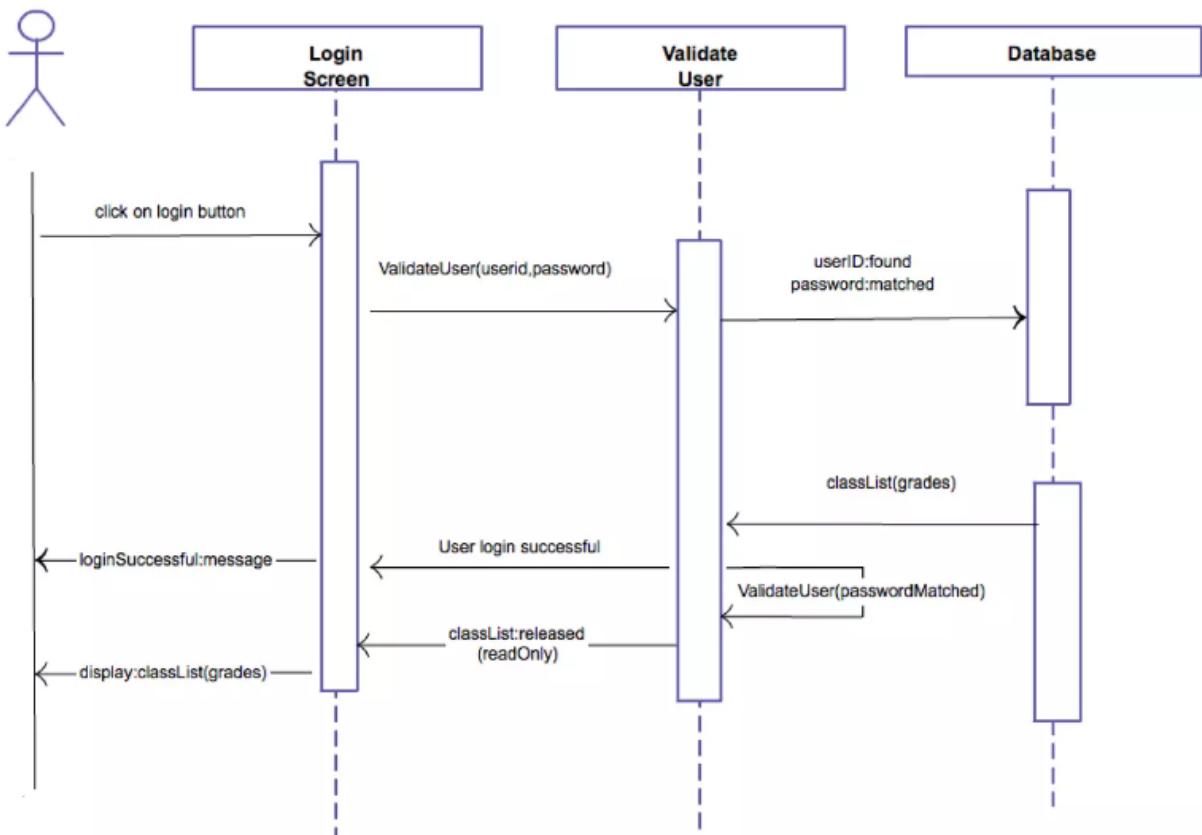
Figure: *Sequence Diagram for Login Functionality*
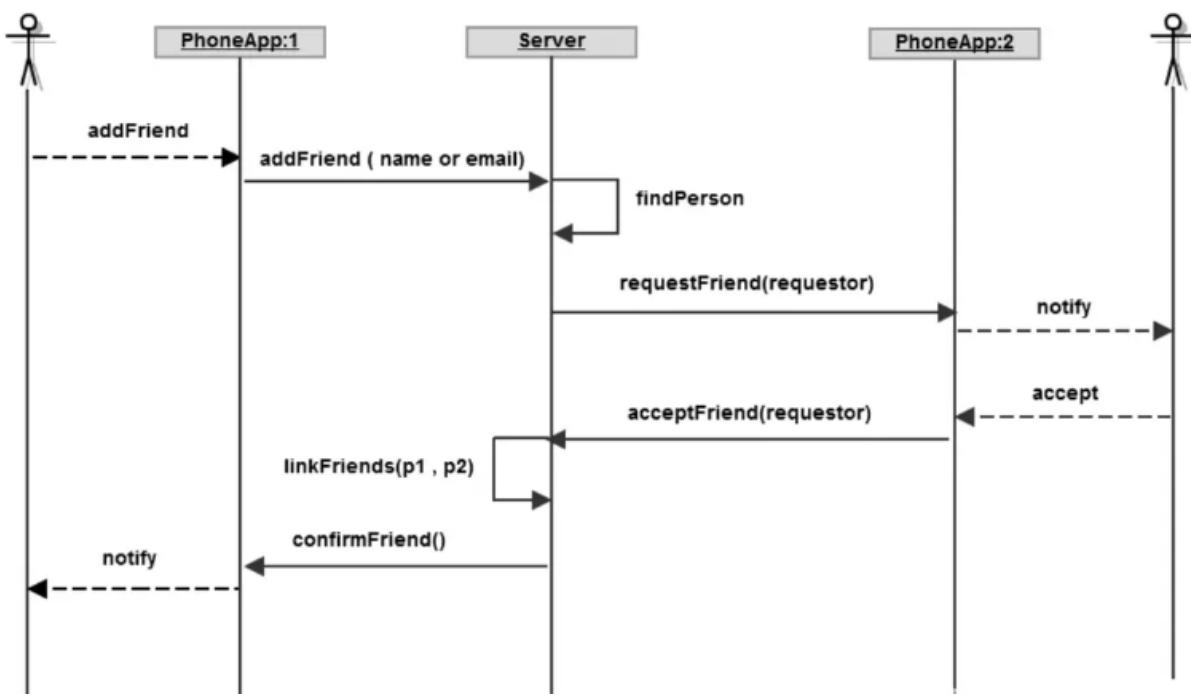
Figure: Sequence Diagram for Add Friend Functionality

Name: Adityan Verma   **Enrolment No.** 07818002721
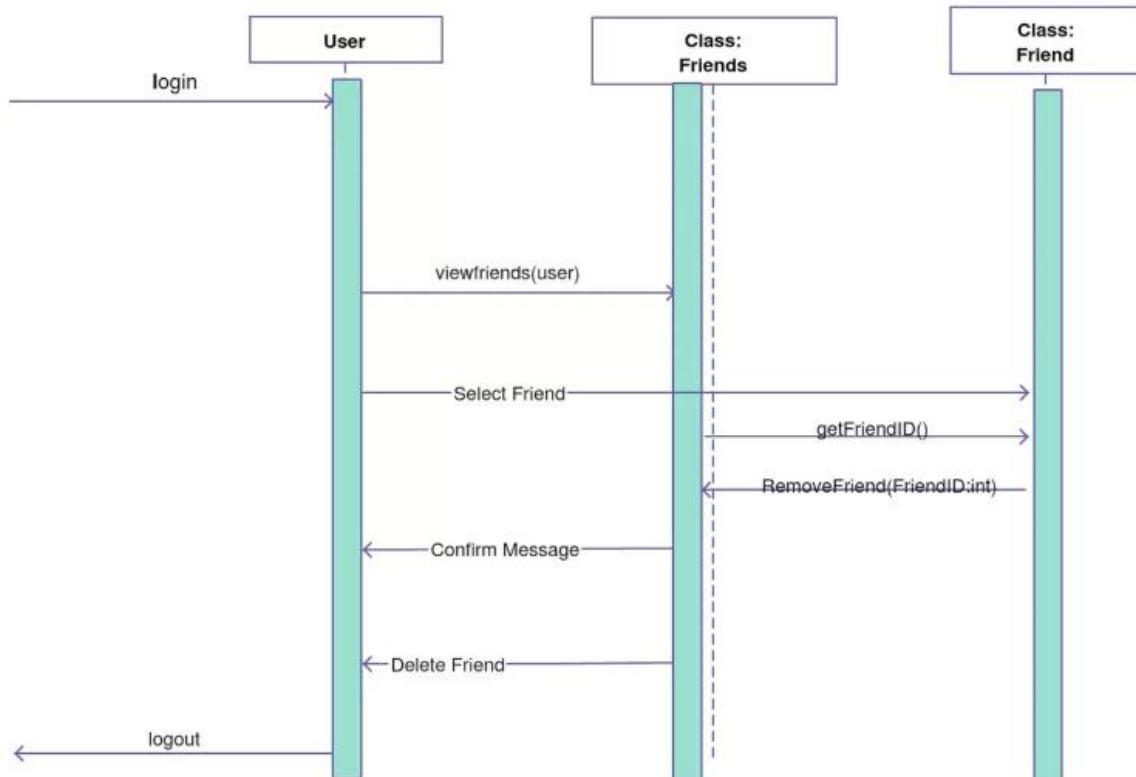
Figure: *Sequence Diagram for Remove Friend Functionality*
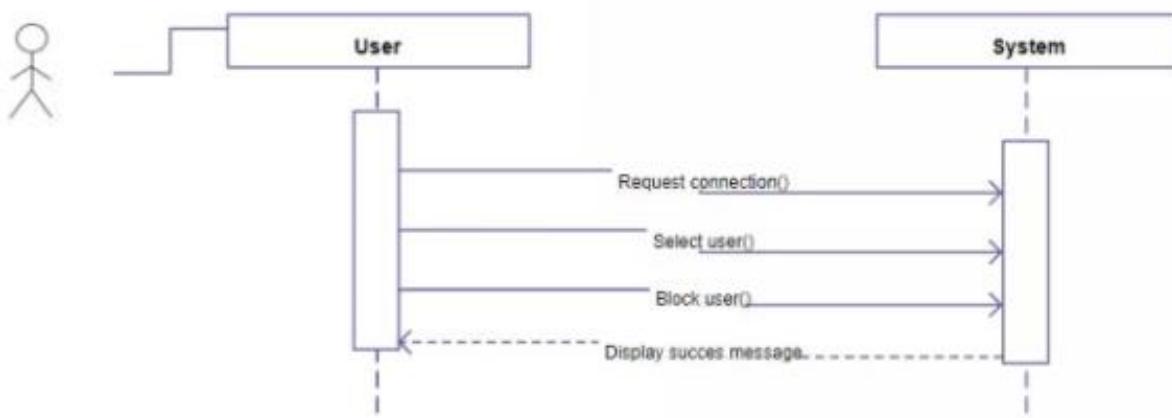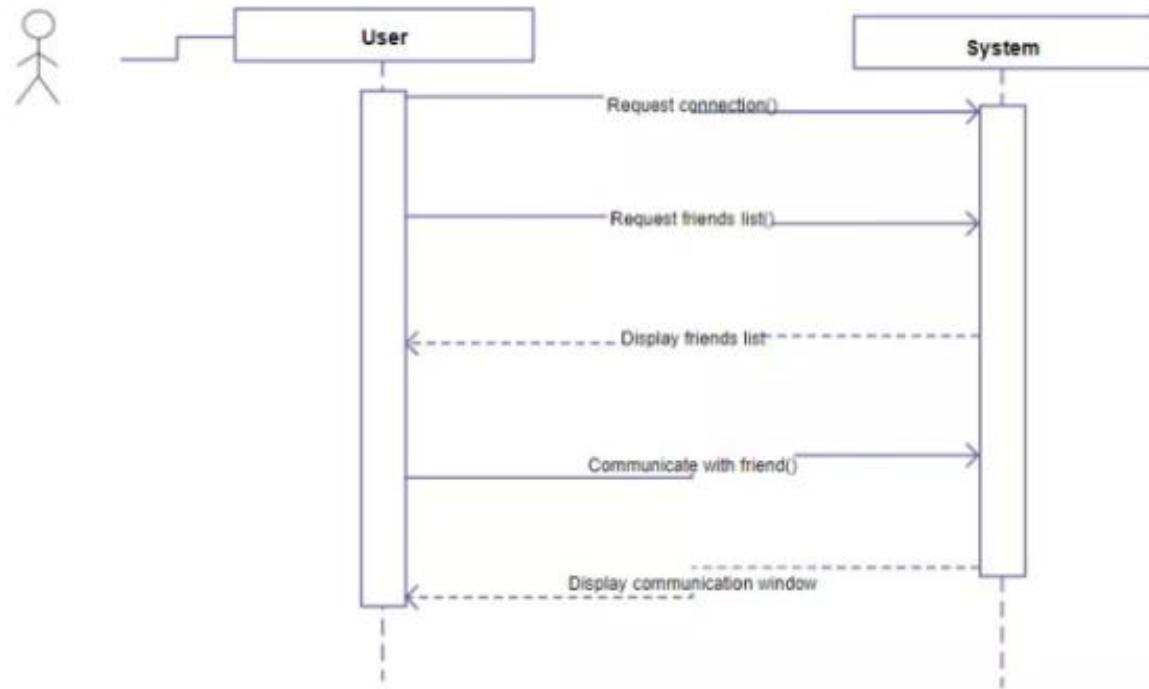
Figure: Sequence Diagram for Block Friend Functionality

Name: Adityan Verma   **Enrolment No.** 07818002721
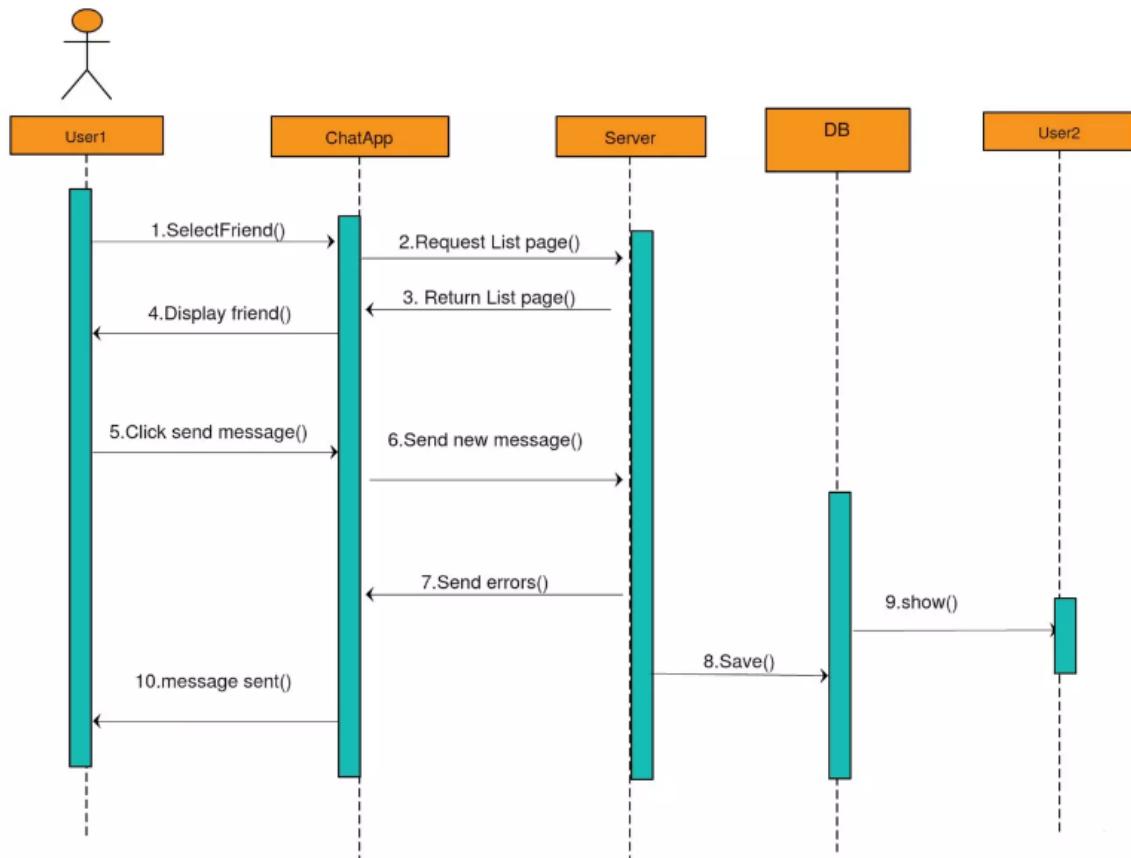
Figure: Sequence Diagram for Send Message Functionality

Name: Adityan Verma
Enrolment No. 07818002721
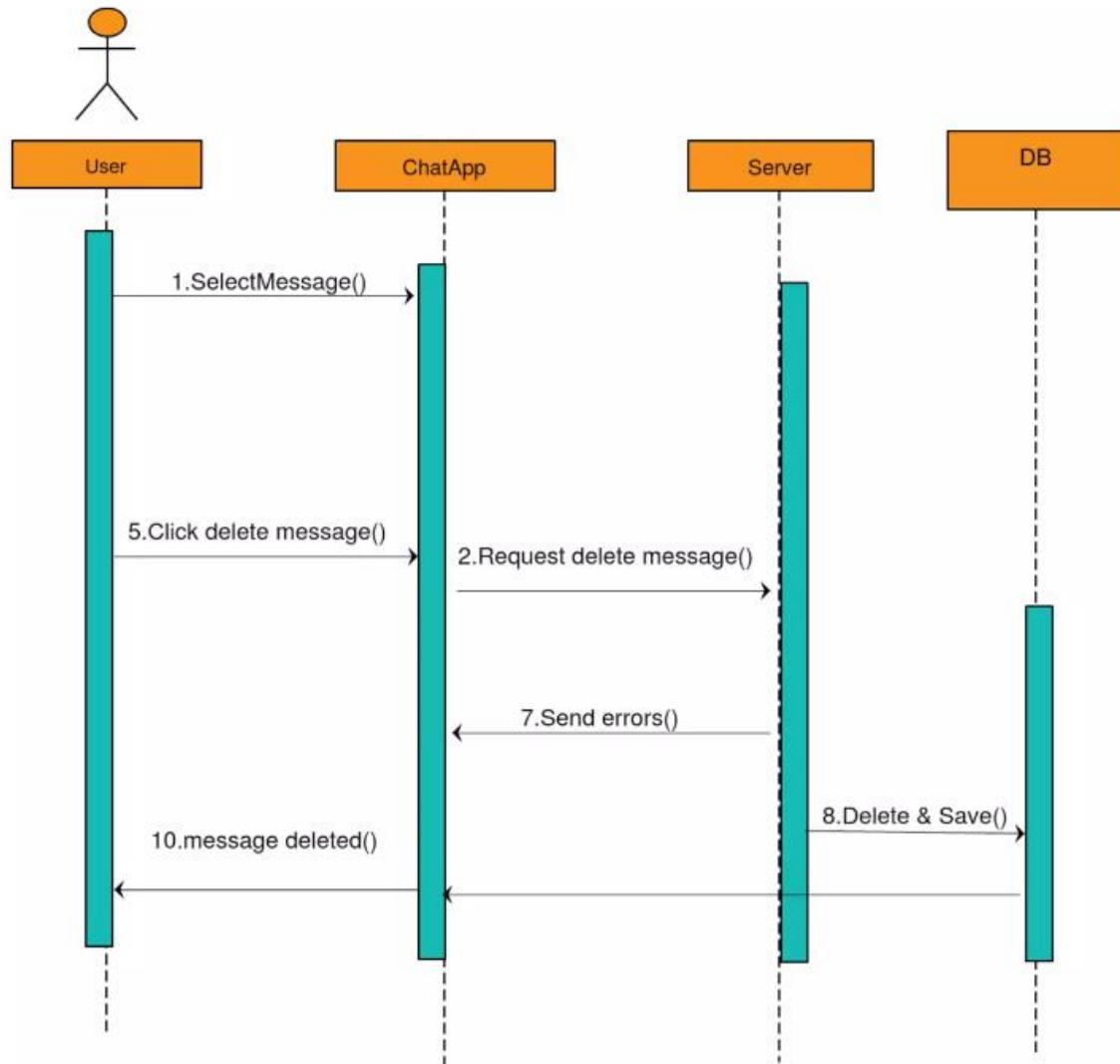
Figure: Sequence Diagram for Delete Message Functionality

Name: Adityan Verma   **Enrolment No.** 07818002721
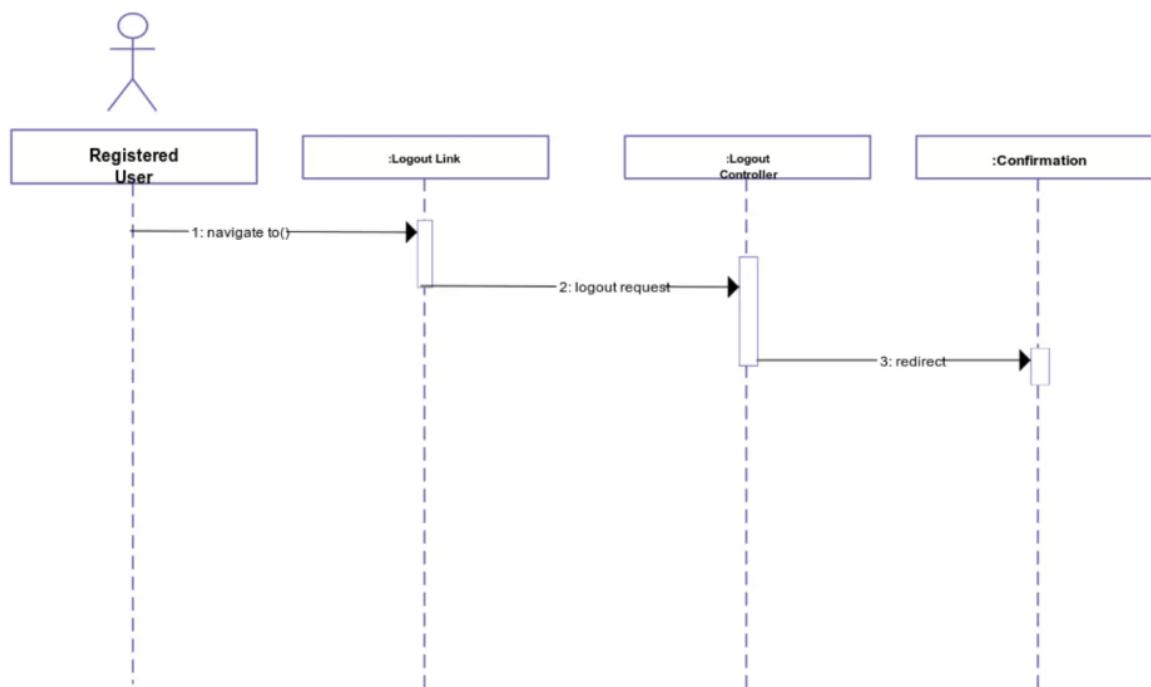
Figure: *Sequence Diagram for Logout Functionality*

Conclusion:

Sequence diagram for Real-time Chat Application has been done successfully.

EXPERIMENT 8

AIM: To perform the implementation view diagram: Component diagram for the system.

Components:

- **User Interface:** Handles graphical interface elements visible to users, facilitating user interaction with the application.

- **Frontend Components:** Encompasses GUI logic and user interaction handling, translating user actions into backend requests.

- **Authentication:** Manages user login and verification processes, ensuring secure access to the application.

- **Chat Management:** Handles chat functionalities, facilitating message sending, receiving, and other chat-related features.

- **Database:** Stores user accounts, chat logs, and other application-related data securely.

- **Notification Service:** Responsible for sending alerts and notifications to users for various events within the application.

- **External Services:** Integrates with external APIs or services, enabling additional functionalities or data exchange beyond the application's core features.
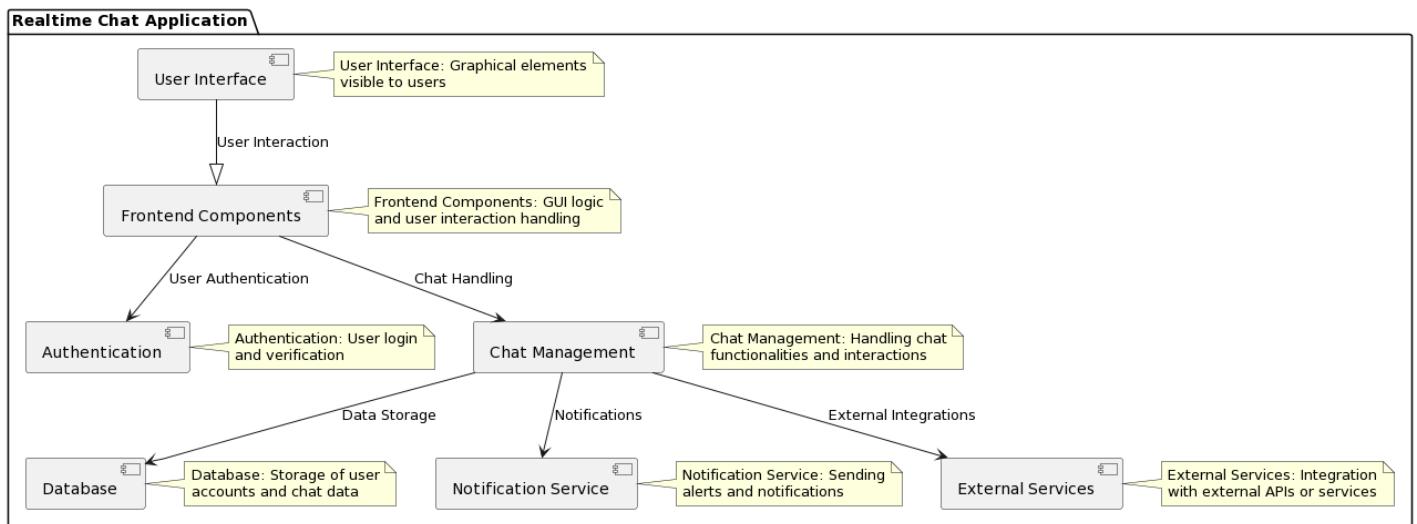
Name: Adityan Verma

Enrolment No. 07818002721

Component Diagram:

Conclusion:

Component diagram for Real-time Chat Application has been done successfully.

EXPERIMENT 9

AIM: Perform Estimation of effort using FP Estimation for chosen system.

Function Point (FP) estimation is a method used for estimating the size of software projects based on functionalities or functional components. For a Real-time Chat Application, we can estimate the effort using the Function Point Analysis approach.

Here are the basic steps for FP estimation:

1. Identify Functional Components:

- External Inputs (EI): User logins, message sending, profile edits.

- External Outputs (EO): Displaying chat messages, notifications.

- External Inquiries (EQ): Search functionality, user profile view.

- Internal Logical Files (ILF): User accounts, chat logs, message database.

- External Interface Files (EIF): Integration with external services, APIs.

2. Assign Complexity Weights:

Assign complexity weights based on the complexity of each functional component. For example:

- Low Complexity: Simple user login.

- Medium Complexity: Chat message sending/receiving.

- High Complexity: Complex search functionalities across multiple parameters.

3. Calculate Function Points:

- For each functional component, use complexity weights and count the occurrences of each.

- Calculate Unadjusted Function Points (UFP) using a weighted sum of the components.

- Apply complexity factors (such as Technical Complexity, Environmental Complexity, etc.) to calculate Adjusted Function Points (AFP).

4. Effort Estimation:

- Use historical data or industry norms to convert Adjusted Function Points (AFP) into person-months or person-hours based on your team's productivity.

Name: Adityan Verma   **Enrolment No.** 07818002721

Calculating Function Points (FP) involves a detailed analysis of the functionalities, their complexities, and assigning weights to them. Since we don't have detailed information on each functional component and its complexity in your Real-time Chat Application, I'll provide a basic hypothetical estimation:

Complexity Levels:

- **Low Complexity:** Authentication System, Contact Form

- **Medium Complexity:** Maintenance

- **High Complexity:** Chat Form

FP Estimation:

Given:

- **EI (External Inputs):** 10

- **EO (External Outputs):** 8

- **EQ (External Inquiries):** 5

- **ILF (Internal Logical Files):** 4

- **EIF (External Interface Files):** 2

Weights:

- **Low Complexity (3 points)**

- **Medium Complexity (4 points)**

- **High Complexity (6 points)**

Calculation:

UFP = (EI×EI weight) + (EO×EO weight) + (EQ×EQ weight) + (ILF×ILF weight) + (EIF×EIF weight)

Assuming weights:

- EI weight = 3

- EO weight = 3

- EQ weight = 4

- ILF weight = 4

- EIF weight = 4

UFP = (10×3) +(8×3) +(5×4) +(4×4) +(2×4) = 30+24+20+16+8 = 98

Assuming a complexity factor of 1.1:

AFP = UFP × Complexity Factor = 98×1.1 = 107.8 ≈ 108

This is a hypothetical estimation based on assigned complexity levels. For a more accurate estimation, a detailed analysis considering specific complexities within each functionality would be necessary.

Name: Adityan Verma　　　　　　　　　　　　　**Enrolment No.** 07818002721

EXPERIMENT 10

AIM: To prepare time line chart/Gantt Chart/PERT Chart for selected software project.

Steps to Create Gantt Chart for Real-time Chat App:

1. **Task Identification:**

   - List all project tasks such as design, development, testing, deployment, etc.

   - Break down tasks into granular activities for each phase.

2. **Task Dependencies:**

   - Define dependencies between tasks. Some tasks may need to be completed before others can start.

3. **Estimate Duration:**

   - Estimate the time required for each task or activity. Use historical data or expert judgment.

4. **Sequence Tasks:**

   - Arrange tasks in chronological order considering dependencies.

5. **Use Gantt Chart Software:**

   - Use specialized software like Microsoft Project, Asana, or even online Gantt chart generators.

   - Enter tasks, durations, dependencies, and start dates to create the Gantt chart.

6. **Visual Representation:**

   - The Gantt chart visually represents tasks over time, showing start and end dates, durations, and dependencies.

7. **Review and Update:**

   - Regularly review and update the Gantt chart as the project progresses.

Name: Adityan Verma   **Enrolment No.** 07818002721

Time Line Chart:

| Project_Starts | Requirement_Gathering | Design_UI_UX | Backend_Development | Frontend_Development | Database_Setup | Integration_with_Sockets | Testing_Bug_Fixes | Deployment |

Project Starts
on 2023/11/01

Requirement Gathering
from 2023/11/01 to 2023/11/10

Design UI/UX
from 2023/11/11 to 2023/11/25

Backend Development
from 2023/11/26 to 2023/12/05

Frontend Development
from 2023/12/06 to 2023/12/11

Database Setup
from 2023/12/11 to 2023/12/17

Integration with Sockets
from 2023/12/18 to 2023/12/25

Testing & Bug Fixes
from 2023/12/26 to 2024/01/06

Deployment
from 2024/01/07 to 2024/01/10

| Project_Starts | Requirement_Gathering | Design_UI_UX | Backend_Development | Frontend_Development | Database_Setup | Integration_with_Sockets | Testing_Bug_Fixes | Deployment |

Gantt Chart:

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55

Requirement Gathering
Design UI/UX
Backend Development
Frontend Development
Database Setup
Integration with Sockets
Testing & Bug Fixes
Deployment

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55

Name: Adityan Verma   **Enrolment No.** 07818002721

PERT Chart: