Software Requirements Specification

for

UniMessenger

Version 1.1 approved

Prepared by Team:

PES1UG21CS523_SAMARTH HIREMATH
PES1UG21CS525_SAMARTH S BHAT
PES1UG21CS548_SEJAL VASUDEV
PES1UG21CS544_SATHWIK HJ

PES University

02-10-2023

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
UniMessenger SRS v1.1	15-11-23	Slight change in RTM	1.1

1. Introduction

Purpose

The purpose of this Software Requirements Specification (SRS) document is to define the scope, objectives, and functional requirements for the development of a University Messenger App. This app is intended to provide a robust and user-friendly communication platform specifically tailored to the needs of students, faculty, and staff within the university community. The University Messenger App aims to enhance communication, collaboration, and information sharing among its users while maintaining a secure and efficient environment.

Document Conventions

We will use the following conventions in this SRS document:

- Boldface text will be used for keywords and important terms.
- Italicized text will be used for emphasis.
- Bulleted lists will be used to present a list of items.
- Numbered lists will be used to present a list of items in a specific order.
- All requirements will be numbered sequentially.

Intended Audience and Reading Suggestions

This SRS document is intended for the following audiences:

- *Developers*: The SRS document serves as the main reference for those who are in charge of putting the software into practice. They will receive all of the specific specifications they require, including information on the user interface, data structures, algorithms, and the features and functioning of the software.
- *Managers of projects*: The SRS document will aid managers in organizing and supervising the software development process. It will notify them about the project's parameters, including its duration, budget, and hazards.
- *Marketing personnel*: The SRS document will assist marketing personnel in outlining the features and advantages of the programme to prospective clients. They can use it to produce marketing materials like brochures and website content.
- *Users*: This document will help you to understand how to use the software.
- *Testers:* The SRS document will help testers to test the software to ensure that it meets the requirements. It will provide them with information about the expected behavior of the software, as well as the test cases that they need to use.
- *Documentation writers:* The SRS document will help documentation writers to write the user documentation for the software. It will provide them with information about the features and

Product Scope

The personal wealth management software will track the following information:

- User Registration and Authentication: Users can create accounts and authenticate themselves securely.
- User Profiles: Users can create and manage their profiles, including profile pictures, contact information, and status updates.
- Messaging: Users can send and receive text messages, multimedia messages, and documents to individuals and groups.
- Notifications: Users will receive real-time notifications for messages, updates, and events.
- Group Chats: Users can create, join, and manage group chats for academic or extracurricular purposes.
- File Sharing: Users can share documents, images, videos, and other files within the app.
- Announcements: University administrators can broadcast important announcements to the entire university community.
- Event Calendar: Users can access and subscribe to the university's event calendar, view upcoming events, and receive event notifications.
- Privacy and Security: The app will prioritize user data security and privacy, with encryption and access control features.

References

the references that I used to write the above:

1. *IEEE Recommended Practice for Software Requirements Specifications: https://ieeexplore.ieee.org/document/88286*
2. *How to Write a Software Requirements Specification (SRS) Document: https://relevant.software/blog/software-requirements-specification-srs-document/*
3. *Writing a Software Requirements Specification Document: https://enisinanaj.medium.com/writing-a-software-requirements-specification-document-97d622805aef*
4. *What is SRS document as per IEEE standard?: https://www.studocu.com/in/document/galgotias-university/software-engineering-testing-methodologies/define-ieee-standards-for-srs/53729665*

2. Overall Description

2.1 Product Perspective

UniMessenger – is a cutting-edge communication and collaboration platform designed exclusively for the dynamic needs of the university community. It serves as a central hub for students, faculty, and staff, fostering seamless communication, information sharing, and engagement within the university ecosystem. With features such as real-time messaging, group chats, event notifications, and secure file sharing, this app empowers users to stay connected, informed, and engaged while prioritizing their data security and privacy. Say goodbye to scattered communication tools and hello to a unified, user-friendly, and efficient way of staying connected with peers, instructors, and university events. The University Messenger App is your key to a vibrant and connected academic experience.

2.2 Product Functions

User Registration and Authentication: Users can create accounts securely and log in with their credentials. Account authentication ensures that only authorized users can access the app.

User Profiles: Users can create and manage their profiles, including uploading profile pictures, updating contact information, and setting status messages to personalize their presence within the app.

Messaging: The app facilitates real-time one-on-one and group messaging, allowing users to send text messages, multimedia content (images, videos), and documents to individuals or multiple recipients.

Notifications: Users receive instant notifications for incoming messages, updates, and important events, ensuring they stay informed and can respond promptly.

Group Chats: Users can create, join, and manage group chats for various purposes, such as academic discussions, project collaboration, or extracurricular activities.

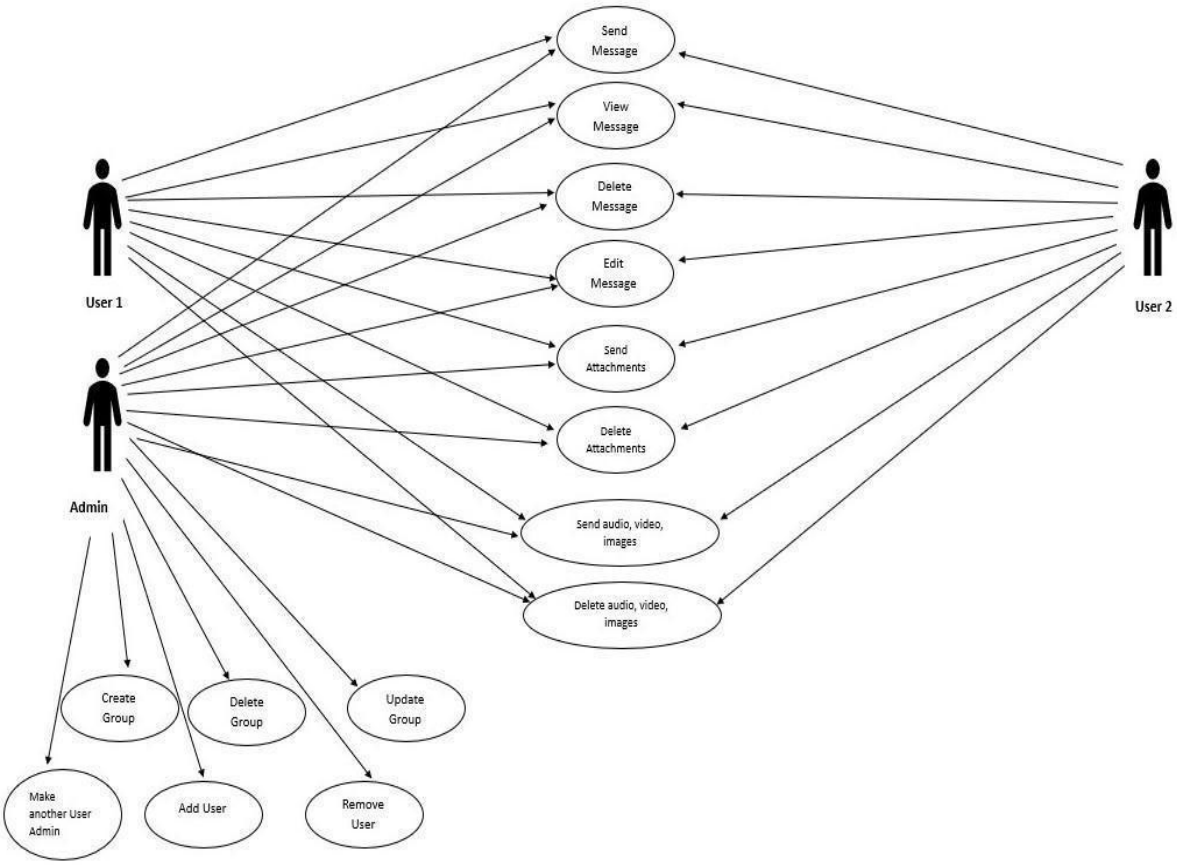
File Sharing: The app enables users to share documents, images, videos, and other file types within conversations or group chats, facilitating seamless information exchange.
Announcements: University administrators or authorized users can broadcast important announcements to the entire university community, ensuring that critical information reaches all users promptly.

Event Calendar: Users can access and subscribe to the university's event calendar, view upcoming events, and receive event notifications, enhancing their engagement with university activities and programs.

Privacy and Security: The app prioritizes user data security and privacy, employing encryption and access control features to protect sensitive information and maintain a safe digital environment.

2.3 User Classes and Characteristics

***Figure 1**: Context diagram*



2.4 Operating Environment

***OE-1**:* The University Messenger App will be compatible with the following web browsers: Microsoft Edge and Internet Explorer versions 11 and above.

***OE-2**:* The University Messenger App will allow user access from within the university's network (Intranet) and, if authorized for external access through the university's firewall, from an Internet connection at the user's location, including their home.

2.5 Design and Implementation Constraints

***CO-1**:* The Process Impact Intranet Development Standard, Version 1.3 [2] must be followed in the design, coding, and maintenance documentation of the system.
***CO-2**:* Implement robust security measures and adhere to data privacy regulation.
***CO-3**:* All HTML code shall conform to the HTML 4.0 standard.
***CO-4**:* All scripts shall be written using HTML.

2.6 User Documentation

***UD-1**:* Aims to help discover peers from various years and departments. Also assist users in discovering the best academic, extracurricular, and campus resources to maximize their educational and personal development.
***UD-2**:* The system shall provide an online tutorial for the user to become used to the

system's operation the first time a new user enters the system and on user demand thereafter.

2.7 Assumptions and Dependencies

***AS-1**:* One common assumption in developing a university app is assuming that all users have reliable and continuous access to the internet and compatible devices for accessing the app.

***DE-1**:* **UniMessenger** relies on a specific DBMS (e.g., MySQL, PostgreSQL, MongoDB) to store and manage user data, academic information, events, and other relevant content.

***DE-2**:* **UniMessenger** depends on a web server to serve web pages, handle incoming requests from users, and communicate with the application's back-end logic and database.

3. External Interface Requirements

3.1 User Interfaces

User Registration and Login: Users should be able to register for an account using email, phone number, or social media accounts. Users should be able to log in securely with their credentials.

User Profiles: Users can create and edit their profiles with a profile picture, username, status, etc. Users can set privacy settings for their profiles.

Chat Interface: Intuitive and user-friendly chat interface with real-time message updates. Support for text, emojis, images, files, and multimedia messages. Options for sending voice messages and making video calls.

Contact Management: Users can add contacts/friends by searching, importing from phone contacts, or connecting social media accounts. Users can organize contacts into groups or favorites. Presence indicators (online, offline, last seen) for contacts.

Notifications: Push notifications for new messages, friend requests, mentions, etc. Users can customize notification preferences (sound, vibration, etc.).

Settings: Users can customize app settings (theme, font size, notifications, privacy settings). Account settings for changing passwords, email, or phone number.

3.2 Hardware Interfaces

Device Compatibility: Support for various devices (smartphones, tablets, desktops) and operating systems (iOS, Android, Windows, macOS). Ensure the application's responsiveness on different screen sizes and resolutions.

3.3 External System Interfaces:

Authentication and Authorization: Integration with external authentication systems (e.g., OAuth) for secure login. Authorization mechanisms to control user access levels within the application.

Database System: Interaction with a database system to store user profiles, chat history, and multimedia files. Databases should be optimized for fast read and write operations.

Cloud Storage: Integration with cloud storage services (e.g., AWS S3, Google Cloud Storage) for storing multimedia files. Secure and efficient file upload and download mechanisms.

Third-Party APIs: Integration with third-party APIs for features like location sharing, weather updates, or language translation. The specific features that are included in the software interface will depend on our target audience and the features that they need.

3.4 Communications Interfaces

Real-Time Communication:

Use of WebSocket or similar protocols to enable real-time message delivery.

Encryption and security protocols to ensure secure communication.

Data Protection and Privacy:

Compliance with data protection laws (such as GDPR) regarding user data storage and processing.

Privacy policy and terms of service accessible to users.

Reporting and Moderation:

Reporting mechanisms for users to report inappropriate content or users.

Moderation tools to handle reported content and users' violations of community guidelines.

4. System Features

4.1 User Registration and Authentication

- Description: This feature allows users to create accounts securely and log in with their credentials. Account authentication ensures that only authorized users can access the app.
- Requirements:
    - Users can register using email, phone number, or social media accounts.
    - Users can reset their passwords securely.
    - Passwords are securely hashed and stored.

4.2 Messaging

- Description: The app facilitates real-time one-on-one and group messaging, allowing users to send text messages, multimedia content (images, videos), and documents to individuals or multiple recipients.
- Requirements:
    - Users can send text messages, emojis, images, and files.
    - Messages are delivered in real-time.
    - Users can create and participate in group chats.

4.3 Notifications

- Description: Users receive instant notifications for incoming messages, updates, and important events, ensuring they stay informed and can respond promptly.
- Requirements:
    - Users receive push notifications for new messages, friend requests, mentions, etc.
    - Users can customize notification preferences.

4.4 Group Chats

- Description: Users can create, join, and manage group chats for various purposes, such as academic discussions, project collaboration, or extracurricular activities.
- Requirements:
    - Users can create group chats.
    - Users can invite others to join group chats.
    - Group chat administrators can manage chat settings.

4.5 Privacy and Security

- Description: The app prioritizes user data security and privacy, employing encryption and access control features to protect sensitive information and maintain a safe digital environment.
- Requirements:
    - User data is encrypted in transit and at rest.
    - Access controls are in place to protect user data.

4.6 File Sharing

- Description: The app enables users to share documents, images, videos, and other file types within conversations or group chats, facilitating seamless information exchange.
- Requirements:
    - Users can upload and download files.
    - Supported file formats include documents, images, and videos.

4.7 Event Calendar

- Description: Users can access and subscribe to the university's event calendar, view upcoming events, and receive event notifications, enhancing their engagement with university activities and programs.
- Requirements:
  - Users can view and subscribe to the university's event calendar.
  - Users receive event notifications for subscribed events.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- **Response Time for User Interactions:** The messaging app should respond to user interactions (e.g., sending/receiving messages, viewing profiles) within 1 second to ensure a seamless and responsive user experience.

- **Real-time Messaging:** Messages should be delivered to recipients in real-time, with a latency of no more than 2 seconds to maintain a conversational flow.

- **Scalability:** The system should be able to support a scalable number of users and messages without compromising performance. It should handle at least 500 concurrent users.

- **Security Measures**

a)Authentication and Authorization Latency: Authentication and authorization processes should be completed within 300 milliseconds to ensure secure access to user messages and information.

b)Data Encryption: All data, including messages and user profiles, should be encrypted both in transit and at rest to maintain the privacy and security of user information.

- **Notification Delivery**

Notification Delivery Time: Notifications (e.g., new messages, group invitations) should be delivered to users in real-time or near-real-time, within 5 seconds of the triggering event.
Data Retrieval

- **Data Retrieval Time**

Fetching historical messages or user data should be accomplished in less than 2 seconds to avoid delays in displaying conversations and profiles.
Concurrent Message Handling

- **Concurrent Message Handling:**

The system should be capable of handling a minimum of 200 concurrent messages without affecting the responsiveness of the platform.
Monitoring and Alerting

- **Automated Monitoring and Alerting**

Monitoring systems should be set up to track system performance and trigger alerts in case of performance degradation or breaches of predefined thresholds, ensuring the system's reliability.

- **Device Compatibility**

a)**Cross-Platform Support:** The app should be compatible with various devices and screen sizes, with response times optimized for mobile phones, tablets, and desktops to provide a consistent and user-friendly experience.

b)**Offline Mode:** The app should support offline message delivery and synchronization, allowing users to access their messages even when not connected to the internet.

- **User Reporting**

Reporting and Activity Tracking: The app should provide users with the ability to track message delivery status, read receipts, and activity history, ensuring transparency and user control over their communication.

- **Compliance**

Compliance with Privacy Regulations: Ensure compliance with university data privacy regulations and standards, including data retention policies and user consent for data processing.

5.2 Safety Requirements

1. **Data Security:**

- Requirement: Protect user data from unauthorized access, breaches, or cyberattacks.
- Safeguards: Implement robust encryption, access controls, and intrusion detection systems.
- Actions to Prevent: Regularly update security protocols and conduct security audits.
- External Policies/Regulations: Comply with university data protection policies and relevant regulations (e.g., FERPA).
- Safety Certifications: Comply with university security standards.

2. **User Privacy:**

- Requirement: Ensure the confidentiality of user information.
- Safeguards: Maintain strict confidentiality agreements and access controls.
- Actions to Prevent: Educate app users on privacy policies and enforce strict data access protocols.
- External Policies/Regulations: Adhere to university data privacy policies and applicable laws (e.g., FERPA).
- Safety Certifications: Compliance with university privacy standards.

3. **Moderation:**

- Requirement: Detect and prevent the dissemination of inappropriate or harmful content within the university community.
- Safeguards: Implement content moderation algorithms and reporting mechanisms.
- Actions to Prevent: Regularly update content moderation algorithms to adapt to new threats.
- External Policies/Regulations: Comply with university guidelines for acceptable use of the app.

5.3 Security Requirements

1. **Data Encryption:**

- Requirement: Encrypt user messages and sensitive data to protect against unauthorized access.
- User Identity Authentication: Implement secure user authentication methods.
- External Policies/Regulations: Comply with university data protection policies.
- Security Certifications: Comply with relevant university security standards.

2. **Access Control:**

- Requirement: Ensure that only authorized university members have access to sensitive data.
- User Identity Authentication: Implement access controls based on user roles and enforce strong password policies.
- External Policies/Regulations: Adhere to university access control policies.
  Security Certifications: Comply with university security controls.

3. **Incident Detection and Response:**

- Requirement: Implement systems to detect and respond to security incidents, such as unauthorized access or data breaches.
- User Identity Authentication: Monitor user activities for suspicious behavior.
- External Policies/Regulations: Comply with university incident response procedures.
- Security Certifications: Certified Information Systems Security Professional (CISSP) for security professionals.

4. **Data Backup and Recovery:**

- Requirement: Regularly backup user data and implement a disaster recovery plan to minimize data loss.
- User Identity Authentication: Secure backups with access controls and encryption.
- External Policies/Regulations: Follow university best practices for data backup and recovery.
- Security Certifications: Comply with relevant university data management standards.

5.4 Software Quality Attributes

- **Security:**

Requirement: Ensure the app is secure to protect user data and privacy within the university community.
Verifiable: Conduct regular security audits and penetration testing, aiming for compliance with university security standards.
Relative Preference: Security is a top priority to safeguard user information.

- **Reliability:**

Requirement: The app should be reliable and available to users 24/7 with minimal downtime.
Verifiable: Monitor uptime and track system availability, striving for at least 99.9% uptime.
Relative Preference: High reliability is essential to maintain user trust and ensure uninterrupted communication.

- **Maintainability:**

Requirement: The app should be easily maintainable to promptly address issues and implement updates.
Verifiable: Measure the time to resolve issues and track the frequency of software updates.
Relative Preference: Ease of maintenance is crucial to respond swiftly to changing user needs.

- **Usability:**

Requirement: The app should be user-friendly and intuitive for students, faculty, and staff.
Verifiable: Conduct user satisfaction surveys and usability testing to achieve high user satisfaction scores.
Relative Preference: Usability should be balanced with functionality to enhance user experience.

- **Interoperability:**

Requirement: Ensure compatibility with various platforms and devices used by university members.
Verifiable: Verify successful communication with different operating systems and browsers.
Relative Preference: High interoperability is vital for seamless communication across the university community.

- **Flexibility:**

Requirement: The app should be flexible to adapt to changing university communication needs.
Verifiable: Measure the time and effort required to implement new features or updates.
Relative Preference: Flexibility is crucial to accommodate evolving communication requirements.

- **Scalability:**

Requirement: The app should be scalable to handle a growing number of users and messages.
Verifiable: Monitor system performance as the user base expands, ensuring smooth scaling.
Relative Preference: Scalability is important to accommodate the university's growing community.

- **Portability:**

Requirement: Ensure the app's functionality across different platforms and devices.
Verifiable: Test the app's performance on various operating systems and devices commonly used by the university community.
Relative Preference: Portability enhances accessibility for all university members.

- **Availability**:

Requirement: Maintain high availability with minimal planned downtime for maintenance.
Verifiable: Monitor uptime and schedule maintenance activities to minimize disruption.
Relative Preference: High availability is essential to ensure uninterrupted communication for university members.

5.5 Business Rules

- User Authentication and Authorization:

Only registered and authenticated university members can access the messaging app.
Administrators have the authority to assign specific roles or permissions to users, such as student, faculty, or staff privileges.

- **Privacy and Data Protection:** User data, including messages and personal details, is kept confidential and secure. Access to sensitive information is restricted to authorized university personnel only.
- **Content Moderation:** Implement content moderation mechanisms to prevent the dissemination of inappropriate or harmful content within the university community.
- **Compliance with University Policies:** Ensure that all app activities and communications comply with university policies, including acceptable use policies and codes of conduct.
- **Notification of Important Events:** University members should receive notifications of important events, such as class updates, campus announcements, and university news.
- **Message Retention:** Implement message retention policies in compliance with university data retention guidelines.

6. Other Requirements

1. **Frontend (React):**
    - React: A JavaScript library for building the user interface and components.
    - React Router: For managing navigation within the application.
    - Redux: For state management, especially for complex applications.
    - Axios: For making HTTP requests to the backend.
    - Material-UI, Bootstrap and CSS: for styling and layout.
2. **Backend (Node.js, Express.js):**
    - Node.js: A JavaScript runtime environment for running server-side code.
    - Express.js: A web application framework for handling HTTP requests and routing.
    - Socket.io (optional): For real-time communication between the server and clients (important for a chat application).
    - JWT (JSON Web Tokens) for authentication and authorization.
3. **Database (MySQL):**
    - MySQL: A relational database management system for storing application data.
    - MySQL driver for Node.js: Use a suitable npm package like mysql or mysql2 to connect to the MySQL database from your Node.js backend.
4. **Other Tools and Libraries:**
    - Git: Version control system for tracking changes in your codebase.
    - npm (Node Package Manager): To manage dependencies and packages for your project.
    - IDE (Integrated Development Environment): Choose a suitable IDE like Visual Studio Code, WebStorm, or Atom for development.
    - Postman: For testing API endpoints during development.
5. **Deployment:**
    - Choose a suitable hosting provider for deploying your app, such as AWS, Heroku, DigitalOcean, or others.

    - Configure your server, domain, and database for production use.
6. **Testing:**
    - Jest, Mocha, or Jasmine: Testing frameworks for unit and integration testing.
    - Supertest: For testing HTTP requests and endpoints.
7. **Security:**
    - Implement secure authentication and authorization mechanisms using JWT.
    - Input validation and sanitation to prevent SQL injection and other vulnerabilities.
    - HTTPS: Ensure secure communication between the client and server.

Appendix A: Glossary

- **SRS (Software Requirements Specification):** A comprehensive document that outlines the requirements and specifications for a software project, including its features, functionalities, and constraints.
- **User Authentication:** The process by which a user proves their identity to access a system, typically through a username and password.
- **API (Application Programming Interface):** A set of rules and protocols that allow different software applications to communicate and interact with each other.
- **GDPR (General Data Protection Regulation):** European Union regulations governing the privacy and protection of personal data.
- **DBMS (Database Management System):** A software application that manages and organizes data in a database.
- **HTTPS (Hypertext Transfer Protocol Secure):** A secure version of the HTTP protocol used for secure communication over the internet.

Appendix B: Analysis Models

1. Intent Recognition:
Objective: To understand the intent behind user messages.
Use Case: Intent recognition helps in routing messages to appropriate channels or customer support agents.
Implementation: Implement Natural Language Understanding (NLU) models, such as pre-trained chatbot frameworks or create custom models using machine learning techniques.

2. Spam Detection:
Objective: To detect and filter out spam messages.
Use Case: Spam detection ensures that users receive relevant and genuine messages.
Implementation: Use machine learning algorithms to identify patterns in spam messages, implement blacklists, and utilize regular expressions for known spam patterns.

3. Language Translation:
Objective: To translate messages from one language to another.
Use Case: Language translation facilitates communication between users who speak different languages.
Implementation: Integrate translation APIs like Google Translate or utilize pre-trained translation models based on Neural Machine Translation (NMT) techniques.

4. Keyword Extraction:
Objective: To identify important keywords from messages.
Use Case: Keyword extraction helps in understanding the main topics of conversation.
Implementation: Utilize NLP techniques to extract significant words or phrases from messages. This can aid in

categorizing messages or triggering specific actions based on keywords.

5. User Profiling:
Objective: To create user profiles based on chat interactions.
Use Case: User profiling helps in offering personalized recommendations and services.
Implementation: Collect user data from chat interactions and apply machine learning algorithms to create user profiles. Consider factors like frequently used words, conversation history, and preferences.

6. Security and Anomaly Detection:
Objective: To detect security threats and unusual activities.
Use Case: Security and anomaly detection prevent unauthorized access and protect user data.
Implementation: Utilize machine learning models to detect patterns of malicious activities, such as account breaches, phishing attempts, or abnormal usage patterns.

7.. Conversation Flow Optimization:
Objective: To optimize the flow of conversations for better user experience.
Use Case: Optimized conversation flow ensures smooth interactions and user satisfaction.
Implementation: Analyze conversation data to identify bottlenecks, repeated queries, or confusing interactions. Use this data to optimize chatbot responses and conversation paths.

Appendix C: Requirement Traceability Matrix

Sl NO	Requirement ID	Brief Description of Requirement	Design Reference	Code File Reference	Test Case ID	System Test Case ID