University of Science and Technology of Ha Noi
Department of Information and Communication Technology

Group Project Report

USTH Connect

Integrated app for university life assistant and student networking

by

Nguyen Thi Van	22BI13459
Chu Hoang Viet	22BI13462
Nguyen Hoai Anh	22BI13021
Nguyen Dang Nguyen	22BI13340
Do Minh Quang	22BI13379

Submission Date: December 31, 2024
Supervisors: Dr. Tran Giang Son

Contents

1 Introduction **6**

1.1 Context and Motivation . . . 6

1.2 Project Objectives . . . 6

1.3 Desired Outcomes . . . 7

1.4 Structure of Thesis . . . 7

1.5 Related works . . . 7

2 Requirement Analysis **8**

2.1 System requirements . . . 8

2.1.1 Functional Requirements . . . 8

2.1.2 Non-functional Requirements . . . 8

2.1.3 Desired Functionalities . . . 8

2.2 Use Case . . . 8

2.2.1 Use Cases Diagram . . . 8

2.2.2 Use Case Characteristics . . . 8

2.3 Use Case and Scenario Description . . . 8

2.3.1 Use case: Authenticate . . . 9

2.3.2 Use case: Student Schedule . . . 10

2.3.3 Use case: University Campus . . . 11

2.3.4 Use case: University Resource . . . 12

2.3.5 Use case: Student StudyBuddy . . . 13

2.3.6 Use case: Real Time Notification . . . 14

3 Methodologies **15**

3.1 System Architecture . . . 15

3.2 Database Design . . . 15

3.3 Use Case Implementation . . . 15

3.3.1 Sync Calendar Sequence Diagram . . . 16

3.3.2 Analysis Call and Message Technologies . . . 17

3.3.2.1 Session Initiation Protocol . . . 17

3.3.2.2 Linphone . . . 18

3.3.2.3 Transport Layer Security Protocol (TLS) . . . 18

3.3.2.4 Communication Handling . . . 19

3.3.2.4.1 Call Initialization . . . 19

3.3.2.4.2 Locating the callee . . . 20

3.3.2.4.3 One-on-One Instant Messaging . . . 22

4 AI Model Analysis and Training **23**

4.1 One-Hot Encode . . . 23

4.2 Dissimilarity Measure . . . 23

4.3 K-Mode Algorithm . . . 23

5 System Design and Implementation **23**

5.1 Hardware and Software Setup . . . 23

5.1.1 Hardware Setup . . . 24

5.1.2 Software Environment Configuration . . . 24

5.2 Application Software . . . 24

5.3 Initial Testing . . . 25

5.3.1 Machine Learning Model Integration and Model Training . . . 26

6 Results and Discussion **26**

6.1 Results . . . 26

6.1.1 Mobile App Results . . . 26

6.1.2 Machine Learning Results . . . 26

6.2 Discussion . . . 26

7 Conclusion & Future Work **26**

7.1 Conclusion . . . 26

7.2 Future Work . . . 26

LIST OF FIGURES

Figure 1: Authentication use case diagram 9
Figure 2: Schedule use case diagram 10
Figure 3: Campus use case diagram 11
Figure 4: StudyBuddy use case diagram 13
Figure 5: Real Time Notification use case diagram 14
Figure 6: Synchronize Calendar sequence diagram 16
Figure 7: SIP signaling and key components 20
Figure 8: SIP registration and locating callee 21
Figure 9: Simple instant messaging flow 22

LIST OF TABLES

Table 1: Actor Actions and System Actions for Authenticate 9
Table 2: Actor Actions and System Actions for Schedule 11
Table 3: Actor Actions and System Actions for Campus 12
Table 4: Actor Actions and System Actions for Resource 12
Table 5: Actor Actions and System Actions for StudyBuddy 14
Table 6: Actor Actions and System Actions for Real time Notification 15

1 Introduction

1.1 Context and Motivation

University life often presents a wide range of challenges for both students and administrators. At USTH, managing academic schedules, building social connections, and navigating campus resources can become an irritated tasks. The limitations of traditional systems, which are sometimes inconvenient, fragmented, and unable to kept track of, only add to the complexity. These systems find it hard to provide up-to-date information and lack of integration needed to offer a seamless experience for students, which leaves a feeling of overwhelmed and disconnected for student.

To overcome these challenges, we developed USTH Connect, an integrated University Life Assistant and Student Networking Application. USTH Connect combines advanced technologies like Spring Boot for backend services, Google Calendar for managing schedule, MapBox for campus navigation, and Moodle for accessing academic resources. Additionally, it employs Machine Learning to power its StudyBuddy feature, which allows students to connect with peers who share similar academic interests and goals.

USTH Connect is desgined to simplify academic and campus life. It combines all key resources into a user-friendly Android application, offering easy access to class schedules, course materials, and campus navigation tools. Apart from academic, the platform encourages more connections between students, especially the first year student, making it easier to build meaningful relationships and collaborate effectively. With StudyBuddy, students can find compatible study partners, chat, and even make audio calls through secure VoIP functionality powered by Linphone.

USTH Connect places a strong emphasis on data security and system reliability. It utilizes a secure PostgreSQL database to manage data centrally, ensuring both protection and efficiency. Additionally, communication between the backend and mobile devices is secured through the Tailscale VPN, providing strong protection for student privacy and uniterrupted access to the platform's features.

This report explores the design and development of USTH Connect, detailing the project's workflow, main features, and the challenges faced during its implementation. It also highlights the creative solutions used to address these challenges, demonstrating how USTH Connect enhances the university experience for students, faculty, and administrators.

1.2 Project Objectives

The primary objective of this project is to design, develop and implement an integrated system that enhances university life and fosters student networking through advanced technological solutions. The goal is to create a platform that seamlessly intergrates personal academic data, event annoucements, and campus resources. By using advanced frameworks and APIs, together with recommendation system, we aim to simplify the administrative tasks, improve campus accessibility,

and encourage more interaction among students. This platform should be user-friendly and accessible through all devices, making it easier for students to experience campus life, stay informed, and connect with their peers.

1.3 Desired Outcomes

1.4 Structure of Thesis

The thesis will be structured as follows:

- **Part I: Introduction**
  Provide a general introduction to the thesis, including an overview of the project, its objectives, and the scope of the work.
- **Part II: Requirement Analysis**
  Lists all the tools, techniques, and system requirements used in the project. It includes both functional and non-functional requirements, as well as desired functionalities.
- **Part III: Methodologies**
  System architecture, database design, and implementation details of various features, illustrated with sequence diagrams.
- **Part IV: AI Model Analysis and Training**
  Analysis and training of AI models for recommend system for study buddy matchmaking, including datasets and model development, with (Model Name) integration.
- **Part V: Results and Discussions**
  Summarizes the implementations and achievements of the system. It reflects on how the objectives were met and provides a summary of the project's outcomes.
- **Part VI: Conclusion and Future Work**

1.5 Related works

In this part we will cite some related works/papers that we used mainly for this project. We also summarize the content of these resources.

2 Requirement Analysis

2.1 System requirements

2.1.1 Functional Requirements

2.1.2 Non-functional Requirements

2.1.3 Desired Functionalities

2.2 Use Case

2.2.1 Use Cases Diagram

The below diagram is to demonstrate the interaction between users and system:

2.2.2 Use Case Characteristics

The system in Figure above has two roles: **Admin** and **Student**:

- **Admin:** Admins have access to all students' functionalities. Additionally, they can manage schedules, campus, resources and create and manage student accounts.
- **Student:** Students can login, reset their password. They have permission to check their schedule, university campus and resources. Additionally, they can connect to other students, who share the same interests, subjects, hobbies, by using StudyBuddy.

2.3 Use Case and Scenario Description

This section provides a detailed description of the various use cases and scenarios within the USTH Connect application. Each use case outlines the interactions between students and the system, describing how the system responds to specific student actions.

2.3.1 Use case: Authenticate

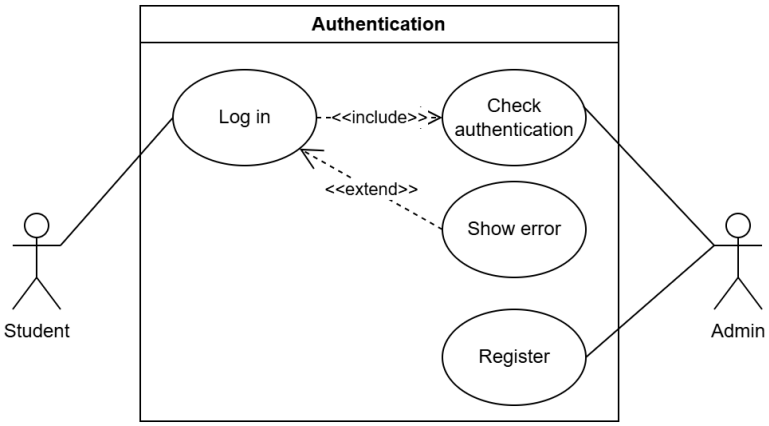

Figure 1: Authentication use case diagram

Description: Students can login and reset their passwords in case they forget them. Admin has to create accounts for students.

Pre-conditions: Before this use case begins, students must ensure that all system packages are fully installed to avoid errors.

Post-conditions:

- **Success:** Students can access the main screen.
- **Failure:** Not displaying the main screen, error message is shown in the system.

Table 1: Actor Actions and System Actions for Authenticate

Actor Action	System Action
Login to the system (User)	1. Students enter studentID and password given by Admin
	2. System verifies studentID and password against the stored database
	3. If both studentID and password are correct, the system grants access to the student and displays the main screen
	4. If it is incorrect, the system displays an error message and asks students to re-enter
Change Password (User)	1. Student enters studentID and old password
	2. System verifies studentID and password against the stored database
	3. If the information entered is correct, students can change their password
	4. System updates the new password in the database
	5. Students enter the new password to login

2.3.2 Use case: Student Schedule

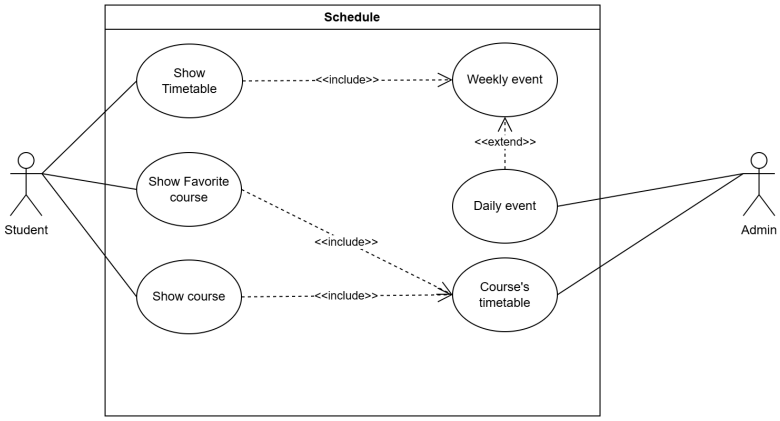

Figure 2: Schedule use case diagram

Description: This use case enables students to view their academic timetable. The system sync schedule every 10 minutes to make sure no daily event is missing.

Pre-conditions:

- Before this use case begins, students must be logged into the system.
- The system up-to-date course details: time, locations, lecturer.

Post-conditions:

- **Success:**
  - Students can see their daily events.
  - System sends notification if there is a new class or a canceled class.
- **Failure:** Not display the event daily and show error messages.

Table 2: Actor Actions and System Actions for Schedule

Actor Action	System Action
User click button "Timetable"	1. The system shows a calendar
	2. Students can change the day, week, month format in the calendar
	3. After choosing a day, students can see daily events
User click button "Favorite Course"	1. The system shows a list of favorite courses which are added by Students
	2. Students can see the timetable of their favorite course after clicking it
	3. Un-click the favorite icon, the system will delete that course out of the list of favorite courses
User click button "Course"	1. The system shows all the courses which match with the student's major and year
	2. Students can see the course class timetable after clicking the course
	3. Click the favorite icon, the system will add that course to the list of favorite courses and show them in Favorite Course

2.3.3 Use case: University Campus

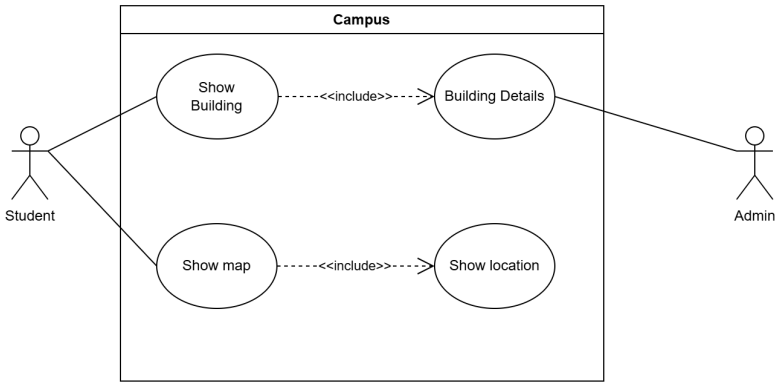

Figure 3: Campus use case diagram

Description: The system displays a list of campus buildings and their details. The system can locate each building in a list on a map.

Pre-conditions:

- Before this use case begins, students must be logged into the system.
- Building data must be fetched from the database.

Post-conditions:

- **Success:**

  - Students can search for buildings on campus and each details.
  - System shows a map where the student and building are located.

- **Failure:** The system displays an error message or map fails to load.

Table 3: Actor Actions and System Actions for Campus

Actor Action	**System Action**
User click button "Building"	1. The system shows a list of buildings which students study in
	2. After clicking a building, the system returns its details
User click button "Map"	1. System loads a map to locate your location and each building

2.3.4 Use case: University Resource

Description: Students can access lectures, slides, exercises and homework from all bachelor's programs, which the system displays.

Pre-conditions:

- Before this use case begins, students must be logged into the system.
- Lectures, slides, exes and homeworks must be fetched successfully.

Post-conditions:

- **Success:** Students can access and download all resources from all bachelor's programs.
- **Failure:** The system displays an error message or resource fail to open or fail to load data.

Table 4: Actor Actions and System Actions for Resource

Actor Action	**System Action**
User click button "Show Resource"	1. The system displays a list of bachelor's programs
	2. After choosing bachelor's programs, the screen shows majors, then, the system shows the subject of the major and lectures, slides, ... of a subject of major

2.3.5 Use case: Student StudyBuddy

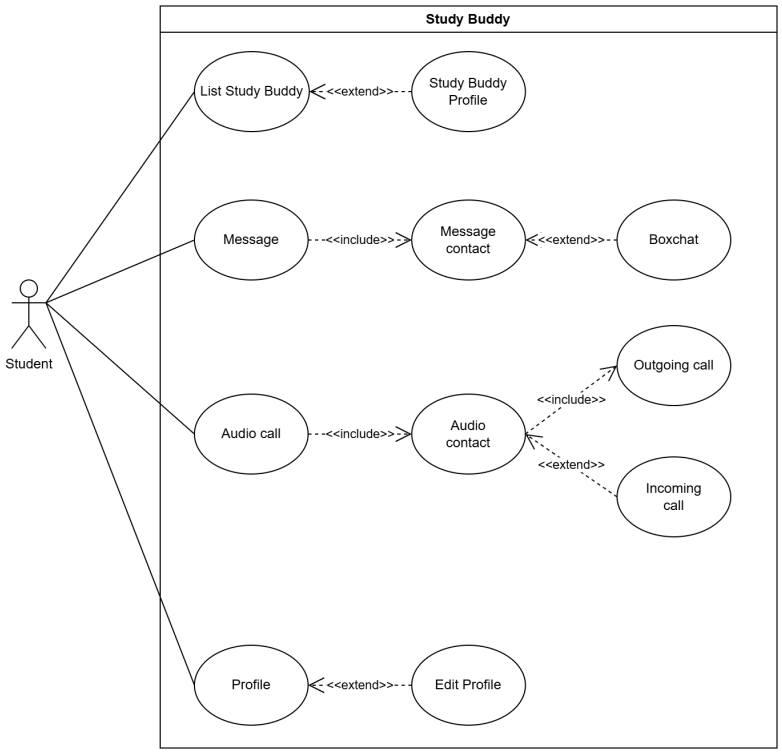

Figure 4: StudyBuddy use case diagram

Description: The system allows students to connect and find each other students, who have the same things with the other. Students can view their profile, interest, ...

Pre-conditions:

- Before this use case begins, students must do a small survey and be logged into the system.
- The system must fetch successfully with other students' profiles.

Post-conditions:

- **Success:**
  - User can view a list of suggestion friends; send and receive text message, audio call; view and update profile.
  - Admins can refresh and reload the suggestion matching list.
- **Failure:**
  - The system fails to fetch student data.
  - System fails to displayprofile of match recommendation.
  - System display an error message.

Table 5: Actor Actions and System Actions for StudyBuddy

Actor Action	System Action
User click button "StudyBuddy"	1. The system displays a list of Students who are in the list of recommendations
	2. After clicking the "Yes" button, add that Student to Chat and Contact
	3. After clicking the "No" button, remove that Student from the list
	4. After clicking the "Refresh" button, the system will refresh the list of recommended students
User click button "Message"	1. The system displays students who Students have matched with
	2. After clicking the box chat, start a chat with another student
User click button "Contact"	1. The system displays students who Students have matched with
	2. After clicking the contact, start an audio call with another student
User click button "Profile"	1. The system fetches the StudyBuddy profile of Student from the database
	2. After clicking "Edit Profile", Students can change information about their StudyBuddy profile after the system is verified

2.3.6 Use case: Real Time Notification

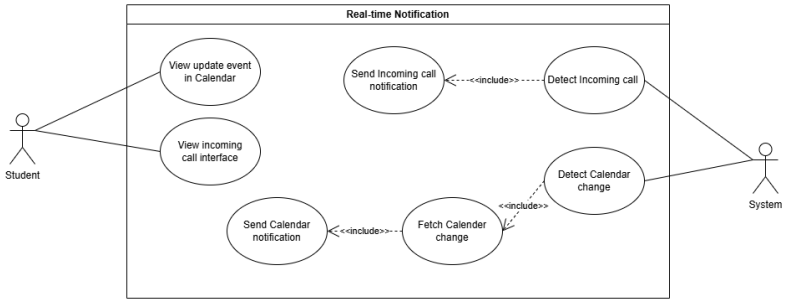

Figure 5: Real Time Notification use case diagram

Description: The push notification system alerts the user whenever there is a change in their calendar (a new event or modification of an existing one) or when they receive a call. Notifications are delivered in real-time, ensuring that the user stays updated on any important events or incoming calls.

Pre-conditions:

- The user's device must be connected to the internet.
- The user must have granted the application permission to access the calendar and incoming notifications.

Post-conditions:

- The real time notification will pop up on the user's device to attract the user's attention.
- The notifications will contain:
  - A bolded title with context about the notification's content.
  - The application icon to show where the notification is coming from.
  - A subtext for describing more information about the notification.

Table 6: Actor Actions and System Actions for Real time Notification

Actor Action	**System Action**
Admin update or add event to the user's calendar	1. The system detects the calendar changes and triggers the notification
User receives an incoming call	1. The system identifies incoming calls and sends a push notification to inform users in real time
User click on the notification	1. **Calendar notification:** The system will open the application and display the updated event on the calendar
	2. **Incoming call notification:** The system will open the application and allow the user to handle the incoming call (accept or decline)

3 Methodologies

3.1 System Architecture

3.2 Database Design

3.3 Use Case Implementation

3.3.1 Sync Calendar Sequence Diagram

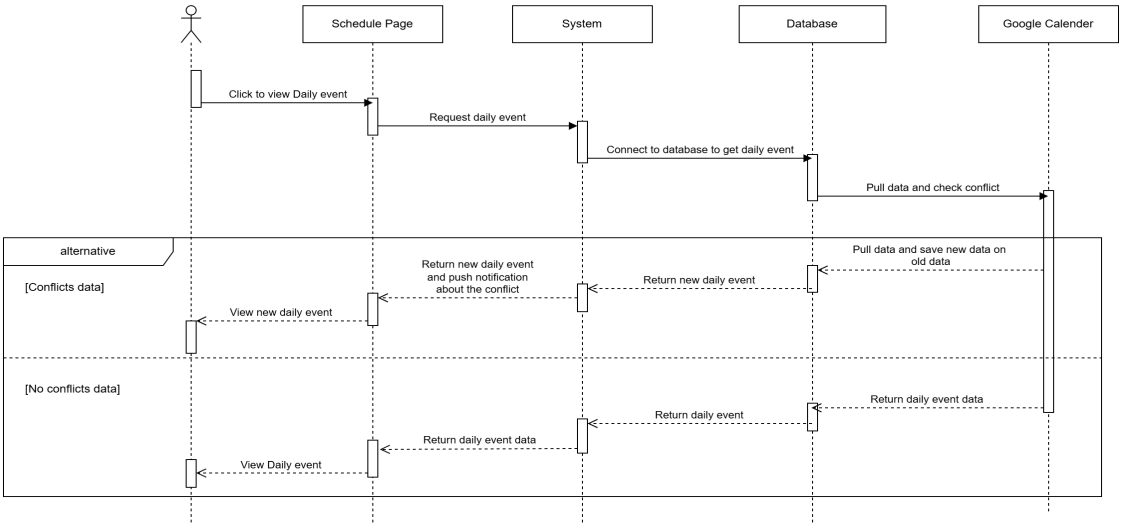

Figure 6: Synchronize Calendar sequence diagram

3.3.2 Analysis Call and Message Technologies

3.3.2.1 Session Initiation Protocol

In order to facilitate data exchange between two parties, establishing a session is essential. However, identifying and connecting with the second participant can be challenging. To address this, specialized protocols have been designed specifically for such scenarios.

The Session Initiation Protocol, commonly known as SIP, is a signaling protocol operating at the application layer, designed for Internet telephony, IP-based telephone systems, as well as mobile phone calling over LTE. SIP's primary purpose is in VoIP systems, where it serves as a support protocol for registering and locating users, and for call set up and management. It can initiate, maintain, and terminate communication sessions that include voice, video and messaging applications. SIP supports five facets of establishing and terminating multimedia communications: user location, user availability, user capabilities, session setup, and session management.

SIP functions as a standalone application but relies on other protocols to ensure the overall architecture operates effectively. At the transport layer, it is transported by using the Transmission Control Protocol (TCP), the User Datagram Protocol (UDP), or the Transport Layer Security (TLS) depending on specific circumstances. In this project, TLS is used (more on TLS in Section 1.3).

SIP architecture consists of:

- **User Agent (UA):** an end point of the network, able to send requests (also known as User Agent Client - UAC) and receive responses (User Agent Server - UAS). User Agent usually acts as both client and server and some examples are - IP phone, softphone, and camera. The caller's phone acts as a client and the callee's phone acts as a server.
- **The Proxy Server:** takes a request from a user agent and forwards it to another user (i.e., an INVITE message).
- **The Registrar Server:** which is responsible for registering users to the network. It accepts registration requests from user agents and helps users to authenticate themselves within the network. It stores the URI and the location of users in a database to help other SIP servers within the same domain.
- **The Redirect Server:** receives requests and looks up the intended recipient of the request in the location database created by the registrar.
- **The Location Server:** provides information regarding the caller's possible locations to the Redirect and Proxy Servers.

In SIP, Each user is identified with a unique address, called SIP Uniform Resource Identifier (SIP URI). It is an address that contains information for establishing a session with the other end. The

SIP URI resembles an E-Mail address and is written in the syntax below with the following URI parameters: SIP - URI = sip:x@y:Port where x = username and y = host (domain or IP).

3.3.2.2 Linphone

Linphone is an open-source VoIP (Voice over Internet Protocol) application that utilizes SIP-based user agent. VoIP messages and calls are made over an IP network rather than over traditional public switched telephone networks (PSTN). Linphone features all basic SIP-related services, such as audio and video calls, call management, call transfer, audio conferencing, and instant messages. The free Linphone SIP service is released with an open-source license; and the SIP server software powering this service is called Flexisip. Linphone uses its open-source library as its core, called liblinphone. The library is a SIP-based SDK5 for video and audio over IP and is written in C/C++. The application is available on Linux, Windows, MacOS, iOS, and Android.
The combination of Linphone and Flexisip SIP proxy provides secure end-user registration and call setup. More precisely, Linphone client establishes and maintains a SIP TLS connection to the Flexisip server. The Linphone client verifies the SIP server's identity based on the X.509 digital certificate of the server (a list of trusted root authorities is provided at compilation time). In this way, message and entity authentication, as well as confidentiality, of the information exchanged between the Linphone client and the Flexisip server is ensured.

3.3.2.3 Transport Layer Security Protocol (TLS)

Transport Layer Security, or TLS, is a widely adopted security protocol designed to facilitate privacy and data security for communications over the Internet. TLS was derived from a security protocol called Secure Socket Layer (SSL). A primary use case of TLS is encrypting the communication between applications and servers. TLS is based on symmetric encryption and is a client - server model, where the client is able to authenticate the server and, optionally, the server is also able to authenticate the client.
When using SIP over TLS, the whole SIP signalling is encrypted. However it holds only on the segments of the communication which actually use TLS. Therefore, TLS will be automatically set to each end of the user by default in this project.

3.3.2.4 Communication Handling

3.3.2.4.1 Call Initialization

A session is initiated with an INVITE method which is the request from a UA client (caller). INVITE has attributes containing source address, destination address, and information about the session from the caller. The SIP client creates an INVITE message for callee, which is normally sent to a proxy server.

If the OK message takes over 200ms to deliver, the progress and status (TRYING) are sent to the caller. The three-way handshake occurs when the OK message confirms the connection to the caller and the ACK message confirms the existence of connection to the callee. Then the media transport, which is logically separated from session initiation will be established. When there is a BYE message being sent, the session will be terminated.

The most common SIP messages explain for the above figure:

- **INVITE:** Initiate a dialog for establishing a call. The request is sent by a user agent client to a user agent server.
- **OK:** confirmation of a request.
- **ACK:** confirmation of the connection form caller to callee.
- **BYE:** Signal termination for a session and end a call.

Below is the illustration of a simple SIP operations:

- **Step 1:** SIP client creates INVITE message for callee, which is normally sent to a proxy server. The proxy server will then obtain the IP address of SIP server that handles requests for the requested domain.
- **Step 2:** The proxy server will reference a location server to identify the next hop server.
- **Step 3:** The location server (non - SIP server) stores information about the next hop server and will return the IP address of callee.
- **Step 4:** Trying message will be sent to the caller when the session initialization is in process.
- **Step 5:** When the IP address is achieved from the step 3, the proxy server will forward the INVITE message to callee machine.
- **Step 6 and 7:** The callee device will ring and the proxy server will forward the RINGING message from the callee to the caller.
- **Step 8 and 9:** When successfully reaching the callee, if the callee accepts the call, the OK response will be sent back from the callee to the caller through the proxy server.

- **Step 10:** An ACK confirmation will then be sent. After that a full-fledged media session is initiated between UAC and UAS.
- **Step 11 and 12:** The session will then be terminated when one of two components sends the BYE message.

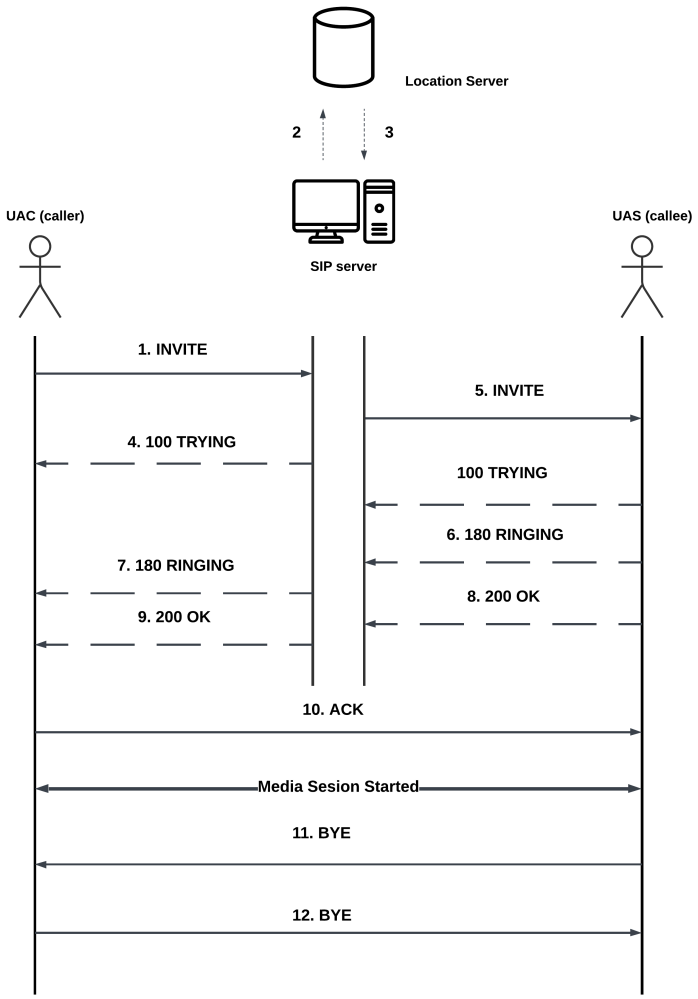

Figure 7: SIP signaling and key components

3.3.2.4.2 Locating the callee

SIP relies on supporting network services like DNS to enable the discovery of proxies and user endpoints. Each user is assigned a unique SIP URI, which is essential for identification and communication. To establish a call using a free SIP service, the following process is typically involved:

- **Account Registration:** A user registers an account with the free SIP service, which provides a unique SIP URI (e.g., sip: username@sip.linphone.org). This registration associates the user with a proxy server, making them reachable for SIP signaling.

- **Call Setup:** To initiate a call, the SIP client uses the provided SIP URI and sends an INVITE message to the proxy server. The proxy server relies on DNS to locate the callee's proxy server and forwards the INVITE message accordingly.
- **DNS Role in Call Routing:** DNS resolves the domain portion of the SIP URI (e.g., sipserver.com) to the corresponding SIP server's IP address. This process ensures that the caller's proxy server can locate the appropriate next-hop server to route the call.
- **Communication Establishment:** Once the proxy servers successfully exchange SIP signaling messages, the callee's user agent receives the call request, and the session is established.

The figure below shows the SIP registration process and how to detect the location of the callee.

- **Step 1:** One user agent with SIP URI `sip:callee_username@sip.linphone.org` register to linphone free sip service registrar server. This server stores the user's URI and current IP address in its location service database. This makes sure that the user is reachable within the `sip.linphone.org` domain.
- **Step 2:** Linphone's registrar updates its location service with the association of `sip:callee_username@sip.linphone.org`.
- **Step 3:** Another user agent (the caller), with the SIP URI `sip:caller_username@sip.linphone.org`, sends an INVITE message to their local proxy server. The destination address in the INVITE is `sip:caller_username@sip.linphone.org`.
- **Step 4 and 5:** The SIP server will then query for its location service and receive the register address of the callee user agent.
- **Step 6:** The proxy server will then forward the INVITE method to the callee's device.

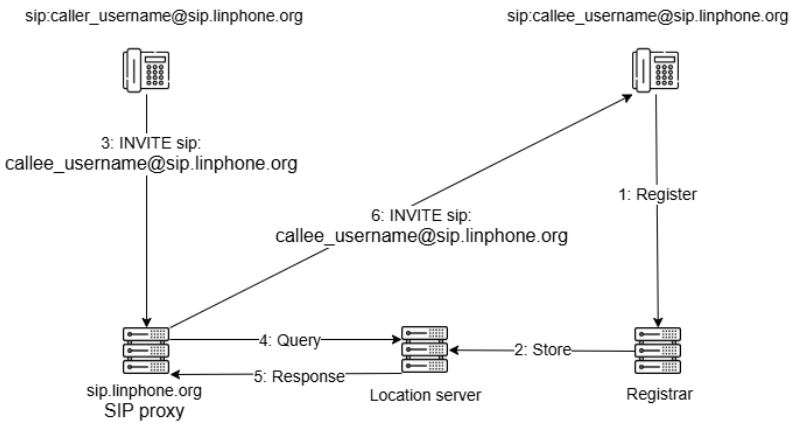

Figure 8: SIP registration and locating callee

3.3.2.4.3 One-on-One Instant Messaging

Instant Messaging is the process of transfering of messages between users in near real-time. The back and forth process to transfer messages have to be fast enough for participants to sustain an interactive dialogue. MESSAGE method - an extension to SIP is used to deliver messages between users. The SIP MESSAGE method is ideal for asynchronous communication as it operates indepently from audio call session. In order to send and receive messages, client - server architecture is used.

One-to-one messaging is a direct exchange messages between two users. After users' credentials is being checked, that information will be sent to the Flexisip server of Linphone. The Flexisip server will then create a temporary file which includes user contact list information. Users can select one of the receiver among their contact lists to send a message. Message will then be sent to the receiver following these steps:

- **Initiating a message:** The sender types a message and then send it. The message will be formated into a SIP MESSAGE request.
- **Message Transmission:** SIP Server will process the received SIP MESSAGE request from the sender and then forward it to the receiver.
- **Receiving the message:** When the receiver get the request, Linphone will then parse the message and display it in the chat interface. In the case that the receiver is offline, the Flexisip server will queue the message for later delivery. A 200 OK response will be deliver to the sender to confirm sucessful exchanging messages.

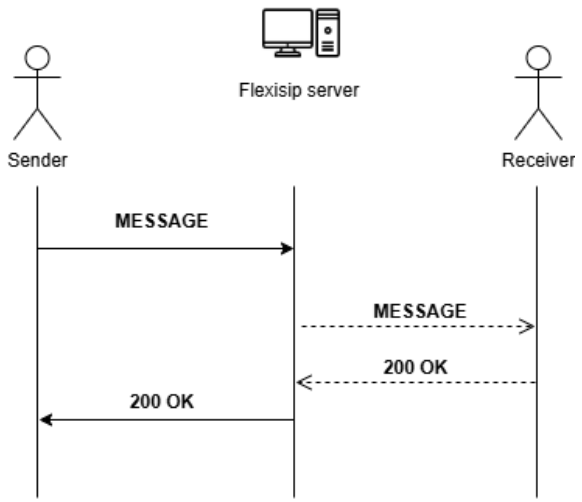

Figure 9: Simple instant messaging flow

4 AI Model Analysis and Training

4.1 One-Hot Encode

One-hot Encoding is a data preprocessing technique that transforms a single categorical variable with n distinct categories into n separate columns. Each column is represented by one number 0 or 1, where the value 1 presents the presence of that category and the value 0 indicates its absence. This method helps machine learning models, which typically require numerical input, better understand and handle the data.

4.2 Dissimilarity Measure

The dissimilarity measure between two categorical objects X and Y (with m attributes) can be defined by the total mismatches of their corresponding categorical attributes. The two objects X and Y are considered to be more similar when the value of a mismatched number gets smaller.

$$d(X,Y) = \sum_{i=1}^{m} \delta(x_i, y_i)$$

where

$$\delta(x_i, y_i) = \begin{cases} 0 & (x_i = y_i) \\ 1 & (x_i \neq y_i) \end{cases}$$

4.3 K-Mode Algorithm

1. Randomly select K initial modes as K clusters.
2. Assign objects to the cluster whose mode is closest to it (which has the smallest dissimilarity). Update the cluster's mode by selecting the highest frequency of each category within the cluster. If there are more than 2 categories of an attribute that have the same highest frequency, we randomly select one of them to serve as mode.
3. Recalculate the dissimilarity of objects with the current modes and reassign the object to the cluster with closest mode and update the new modes for clusters.
4. Repeat Step 2 and 3 until every cluster has no changes after testing the whole dataset.

5 System Design and Implementation

5.1 Hardware and Software Setup

This section describes the detailed process of setting up the system, covering hardware and software configurations, as well as initial testing to ensure all components work seamlessly.

5.1.1 Hardware Setup

The essential hardware components for the system include:

- **Development Machines:** Computers used for developing and hosting backend services and databases (Spring Boot and PostgreSQL).
- **User Devices:** Smartphones or virtual devices running the Android app to access features such as Google Calendar, MapBox maps, and Moodle resources.
- **Networking Equipment:** Tailscale VPN for secure and reliable communication between user devices and the backend server.

5.1.2 Software Environment Configuration

The software setup involves configuring the necessary tools and platforms to support the development and operation of the system:

- **Operating System:** Windows was used to host backend services and the database, while Android was the primary platform for the mobile application.
- **Database:** PostgreSQL was installed and configured as the relational database management system for managing user data, calendar events, location details, and StudyBuddy profiles.
- **Backend Framework:** Spring Boot was implemented to manage REST API endpoints, handle authentication and authorization, and synchronize data between components.
- **Mobile Development Tools:** Android Studio served as the primary integrated development environment (IDE) for creating the Android application, leveraging Java and libraries such as Retrofit and MapBox SDK.

5.2 Application Software

The application software comprises several core modules, each designed to handle specific functionalities:

- **Authentication and Authorization Module:**
  - Implements JWT-based authentication.
  - Manages Role-Based Access Control (RBAC) for ADMIN and USER roles.
- **Google Calendar Integration Module:**
  - Fetches event data from Google Calendar.
  - Detects event updates and sends notifications to the user via the mobile application.

- **MapBox Integration Module:**
  - Stores campus location coordinates (latitude and longitude).
  - Dynamically renders campus maps within the app.
- **Moodle Resource Module:**
  - Communicates with Moodle APIs to retrieve course resources, such as slides, source code, and PDFs.
- **StudyBuddy Matching Module:**
  - Collects user profile data (e.g., interests, personality).
  - Uses a machine learning recommendation system to suggest matches based on shared interests.
  - Features include:
    * Chat functionality for text-based communication.
    * Audio calling capabilities powered by the Linphone library, which provides SIP-based VoIP functionality.
- **Notification System:**
  - Delivers real-time notifications for calendar updates and incoming calls.

5.3 Initial Testing

Extensive testing ensured all components functioned correctly and integrated seamlessly:

- **Hardware Testing:**
  - Verified the setup and operation of development machines, servers, and networking equipment using Tailscale VPN for secure communication.
- **Software Testing:**
  - Tested the configuration and performance of the operating system, PostgreSQL database, and Spring Boot services.
- **Integration Testing:**
  - Validated the interaction between backend APIs and mobile app features, ensuring functionality for:
    * Google Calendar synchronization.
    * MapBox map rendering.

* Moodle resource retrieval.
* StudyBuddy matching and chat capabilities.
* Linphone-based audio calling.

5.3.1 Machine Learning Model Integration and Model Training

This comprehensive setup and testing process ensured that the system was ready for deployment and met the project's objectives effectively.

6 Results and Discussion

6.1 Results

6.1.1 Mobile App Results

In this part we can have the demo for each feature of the app.

6.1.2 Machine Learning Results

In this part we will show the result of the clustering algorithm, using the evaluation metrics that we mentioned in the previous section.

6.2 Discussion

7 Conclusion & Future Work

7.1 Conclusion

7.2 Future Work