UNIVERSITY OF SCIENCE AND TECHNOLOGY OF HANOI

DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY

GROUP PROJECT REPORT

USTH Connect

Integrated app for university life assistant and student networking

Group Members

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

Supervisor

Assoc. Prof. Tran Giang Son

January, 2025

TABLE OF CONTENTS

Acknowledgement

Our group would like to thank all individuals who have contributed to the completion of this project.

First and foremost, we would like to express our special thanks and gratitude to our professor Assoc. Prof. Tran Giang Son for guiding us throughout the project with his feedback and supervision.

We also appreciate the effort and the hard work of each individual members.

Finally, we are really grateful for the opportunity to work on this project and thanks for all the support.

LIST OF ABBREVIATIONS

API	Application Programming Interface
RBAC	Role-Based Access Control
JWT	JSON Web Token
VoIP	Voice over Internet Protocol
VPN	Virtual Private Network
K-modes	K-modes Clustering Algorithm
SIP	Session Initiation Protocol
TLS	Transport Layer Protocol
UAC	User Agent Client
UAS	User Agent Server
URI	Uniform Resource Identifier
DBI	Davies-Bouldin Index

LIST OF FIGURES

LIST OF TABLES

Abstract

University life especially for first-year students, involves several challenges, including managing academic timetables, finding study resources, locating buildings in the campus, and builing connections between students. In order to address these issues, USTH Connect, an integrated University Life Assistant and Student Networking App, was developed. This Android application utilizing the advanced technologies to improve access to academic calendars, study materials, and campus' buildings navigation, while also encouraging connections via the innovative StudyBuddy function. By utilizing the K-modes clustering model, the app will pair students with compatible study partners and enables communication through secure Linphone-powered VoIP audio calls and instant messaging.

In addition, we also use a centralized PostgreSQL database for data storage, data security. The software includes Google Calendar, MapBox, and Moodle, allowing for real-time updates, easier navigation, and fast resource access. USTH Connect aims to improve the university experience by simplifying academic tasks and encouraging contact between students. This report focuses on the application's design, development, and implementation, demonstrating how it solves issues while improving efficiency and connectivity in the academic environment.

Keywords: USTH Connect, K-modes clustering, Linphone, Google Calendar, MapBox, Moodle

I/ Introduction

1. Context and Motivation

During the academic journey, university students around the world face several challenges. Particularly, first-year students often struggle with managing academic schedules, finding their way around campus, accessing study materials, and forming meaningful social connections with their peers. In some universities, current traditional systems are able to serve their purposes well, but these systems are fragmented and difficult to maintain. Moreover, they often do not provide solutions for students to manage their schedules, access academic resources, and encourage interaction between students within a single platform. There are few platforms that help students expand their social networks or find suitable peers to connect with. However, these platforms lack other features which significantly improve their academic life.

Around the world, there are existing apps whose purpose is to solve some of these challenges above. For example, platforms like Google Calendar and Moodle already help students manage schedules and access course materials, while apps like Google Maps support in navigation around campus section. However, these apps do not provide a comprehensive solution for improvement of university life. Each of these has their own limitation. Google Calendar manages academic schedules but does not help students find compatible study partners or navigate campus. Moodle allows students to access lecture resources but does not contribute to the encouragement of social connections or provide tools for campus navigation. Similarly, while Google Maps is responsible for navigating place and locating study areas, it does not include the academic support and social connection assistance. Furthermore, their limitation make those apps less user-friendly, especially for students who need a combination of all these functionalities in one platform.

These drawbacks focus on how important a comprehensive solution which aims to meet all those specific needs is, especially the students at the University of Science and Technology of Hanoi. To overcome these challenges, we developed USTH Connect, an integrated University Life Assistant and Student Networking Application.

USTH Connect is built with the goal of supporting students' university life by combining modern yet simple technologies into an Android application, providing students with effortless access to the schedules, study materials and study location addresses. Not only does it support their studies, it also encourages more connections between students, supporting them in their social life. That helps them to create more quality and meaningful relationships in the future. With StudyBuddy, students can find compatible study partners, chat, and even make audio calls through secure VoIP functionality powered by Linphone. To support the StudyBuddy features, we apply the K-modes model to help students find compatible people in the university. USTH Connect uses a secure PostgreSQL database to manage data centrally, ensuring both protection and efficiency.

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 and administrators.

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. The platform will be accessible to students, making it easy for them to experience campus life, stay informed, and connect with friends.

3. Desired Outcomes

The USTH Connect is expected to enhance the university experience by streamlining academic and social interactions for students and administrators. To achieve that outcomes, we seamlessly integrate essential tools like schedules, course materials, and campus navigation. The StudyBuddy feature, supported by machine learning, is aimed to effectively connect students with compatible learning partners, encourage cooperation between students, and enhance the student social life. Linphone-powered chat and audio calls will facilitate seamless communication and strengthen social bonding among students. Real-time notifications for calendar updates and events together with calling and messaging will keep users informed, organized and connected. The integration of Google Calendar, MapBox, and Moodle promise a student-friendly experience across Android devices. In summary, USTH Connect is supposed to significantly improve the university experience for students and administrators by encouraging a more efficient and connected academic environment.

4. Structure of Report

The report will be structured as follows:

- **Part I: Introduction**
  Provide a general introduction to the report, 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: Machine Learning 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: System Design and Implementation**
  Overview of the system's design and implementation process, detailing the hardware and software setups, core application modules, and the initial testing phases to ensure seamless integration and functionality.

- **Part VI: 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 VII: Conclusion and Future Work**
  Reviews the project's successes in improving student life and fostering connections within the university. It also acknowledges the challenges encountered during development and identifies areas where the system could be further improved.

II/ Requirement Analysis

Ensuring and following the initial targets set for the project, we must be carefully analyzed the system architecture and developed a detailed work deployment plan. Below is a detailed description of the methodologies and procedures we implemented.

1. System requirements

This section outlines the essential requirements for the development and successful operation of USTH Connect, classified into functional, non-functional, and desired functionalities.

1.1 Functional Requirements

The functional requirements specify the primary operations and features of USTH Connect to meet student and system objectives:

- **Authenticate and Authorize:** Implement secure student authentication using JWT-based tokens and role-based access control (RBAC) for managing access permissions.
- **Integrate with Google Calendar:** Enable users to sync, view, and manage academic schedules in real-time within the application.
- **Match with StudyBuddy:** Allow students to find study partners based on shared academic interests, goals, and compatibility through machine learning algorithms.
- **Access Moodle Resources:** Provide seamless access to course materials, including slides, assignments, and reference documents, fetched directly from Moodle.
- **Send Real-time Notifications:** Notify users promptly about calendar updates, study partner incoming communication, and scheduled events.
- **Display Map and Navigation:** Integrate MapBox to display campus maps, building locations, and navigation features for easy campus exploration.
- **Enable Communication Features:** Facilitate text-based chatting and VoIP audio calls using Linphone for seamless interaction between users.
- **Provide Administrative Tools:** Offer reporting tools for administrators to monitor student engagement, system usage.

1.2 Non-functional Requirements

The non-functional requirements outline the system's essential performance, reliability, and usability standards. While official testing has not yet been carried out, the following testing are planned to validate each requirement:

- **Scalability:** The system should support a moderate increase in student numbers and data while maintaining acceptable performance levels. To perform test on the increase of users, we plan to use Apache JMeter or Locust to simulate growth usage, scaling up to around 130% of the current user load. These tests will focus on some important functionalities like schedule retrieval and display for each major, resources view, and StudyBuddy matching, in order to verify that performance of the application is stable and working well with a 30% increase in user numbers.

- **Security:** User data must be protected with basic encryption protocols, secure database practices (PostgreSQL). To test the security of the system, we plan to use OWASP ZAP to perform penetration testing and security audits. By taking advantages of the tools, we will simulate attacks, such as SQL injection and data interception, will be held to analyze and identify upcoming potential vulnerabilities to the system. These tests will focus on the robust security of sensitive information and how the system will act when facing against unauthorized access attempts.

- **Availability:** The system should aim for an uptime of at least 90%, ensuring minimal disruption to student access. To validate this, we plan to perform tests which involve of simulating a situation like a high user usage and resource demand during peak usage times. By using automated tools and performing a test around 30 days, we will check for the stability of the system behavior and uptime in these specific scenarios. These tests will ensure to meet the target uptime we mention above and to maintain the reliable access for users.

- **Performance:** Key functionalities, such as schedule retrieval and StudyBuddy matching, should respond within 5 seconds under normal conditions. To verify this, we plan to take advantages of libraries like Microbenchmark and Macrobenchmark to simulate the performance benchmarking. By dividing into different levels of concurrent user (e.g., 10, 50, and 100 users), we will apply the test for each levels to observe response time for these key features. The goal is to ensure that the response time will be around 5 seconds for at least 90% of all functionalities during typical and crowded conditions.

- **Usability:** The interface should be straightforward and functional, providing a smooth experience on most Android devices. To test the usability, we plan to perform with a sample group of students who will be asked perform common tasks, such as retrieving schedules, acessing resources, and finding StudyBuddies. Then through surveys and interviews, we can collect the feedback which will be used to identify areas for improvement. The goal is for at least 80% of participants to feel satisfied with the interface and find it functional and easy for them to use.

- **Maintainability:** Updates and feature additions should be feasible without causing significant downtime or compatibility issues. To test the maintainability of the system, we plan to simulate updates in a controlled development environment. This involve introducing new features or patches while performing all the above testing to ensure that all functionalities are working as expected. During the maintenance, we will also observe the update process for any potential errors, such as compatibility errorso or longer downtime. These tests will verify that the updates can be applied correctly with minimal interruption.

1.3 Desired Functionalities

These functionalities are not critical but aim to enhance the overall student experience and system capabilities:

- **Personalized Dashboard:** Provide a customizable home screen with quick access to frequently used features, such as schedules, notifications, and StudyBuddy matches.
- **Event Reminders:** Enable automated reminders for upcoming deadlines, meetings, and academic events.
- **Offline Mode:** Allow limited functionality, such as viewing previously synced schedule, campus location and resources link, when the student is offline.
- **User Feedback System:** Incorporate feedback and rating mechanisms for StudyBuddy matches and overall app experience.
- **Cross-platform Compatibility:** Develop future support for IOS devices and web browsers to expand student accessibility.

2. Use Case

Based on the use case diagram, this section will offer a general explanation of the system's features.

2.1 Use Cases Diagram

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

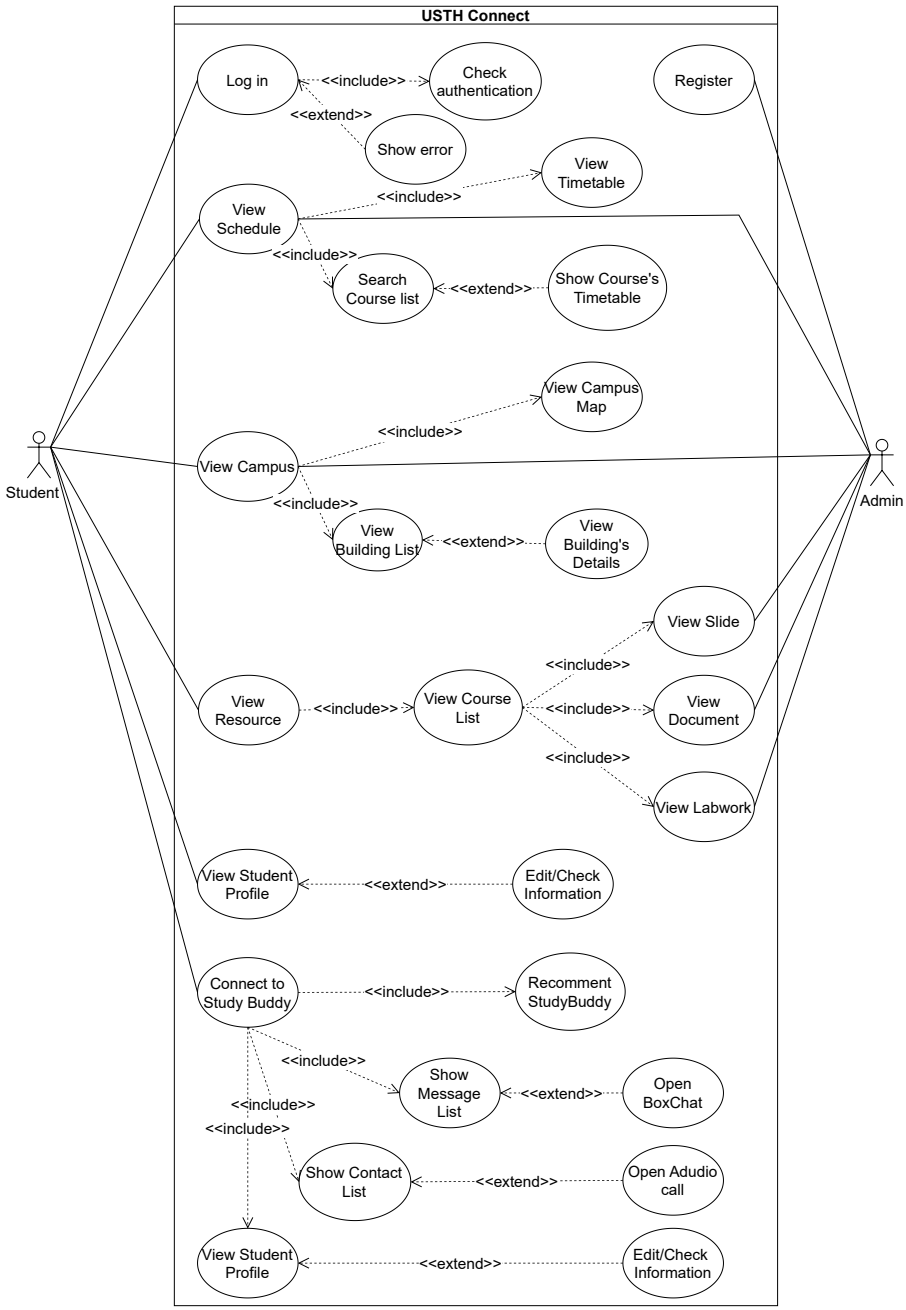


Figure 1: Use case diagram

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.

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.

Use case: Authenticate

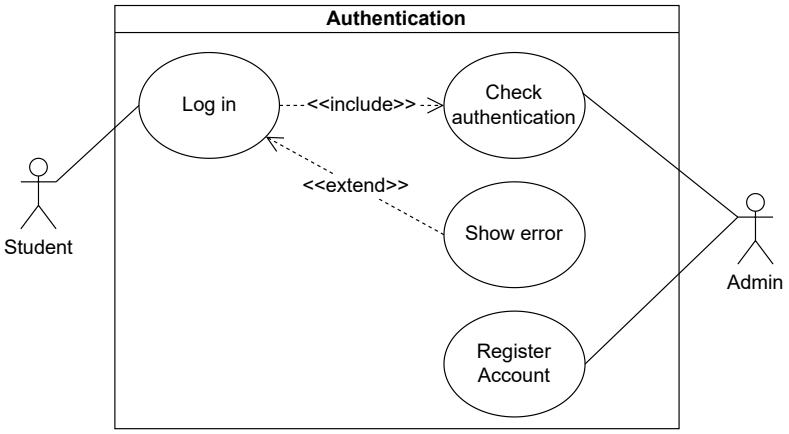


Figure 2: Authentication use case diagram

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.

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

Table 1: Actor Actions and System Actions for Authenticate

Use case: Student Schedule

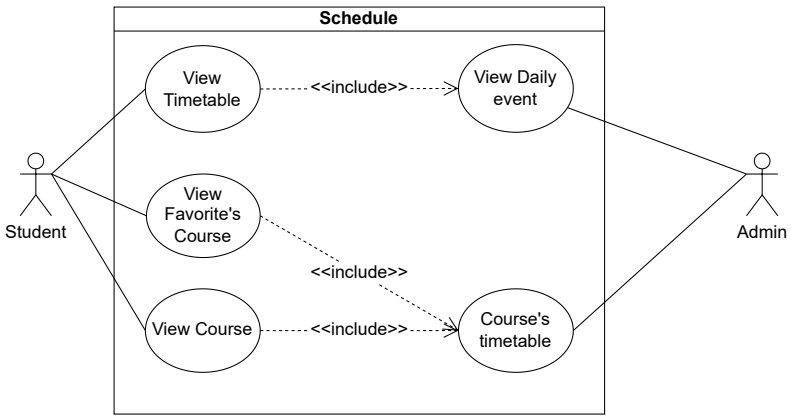


Figure 3: Schedule use case diagram

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 is up-to-date with 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 cancelled class.
- **Failure:** Not display the event daily and show error messages.

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

Table 2: Actor Actions and System Actions for Schedule

Use case: University Campus

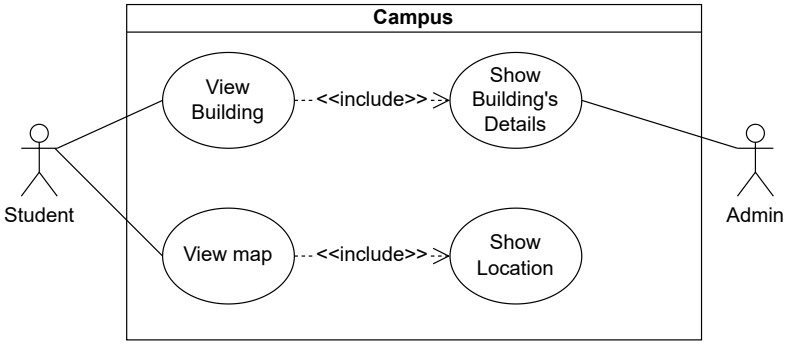


Figure 4: Campus use case diagram

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.

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

Table 3: Actor Actions and System Actions for Campus

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.

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

Table 4: Actor Actions and System Actions for Resource

Use case: Student StudyBuddy

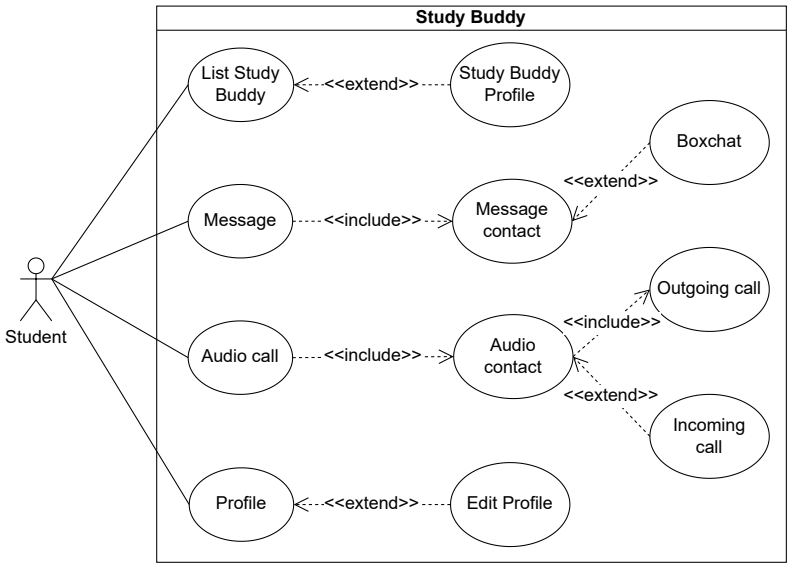


Figure 5: StudyBuddy use case diagram

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 display profile of match recommendation.
  - System display an error message.

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

Table 5: Actor Actions and System Actions for StudyBuddy

Use case: Real Time Notification

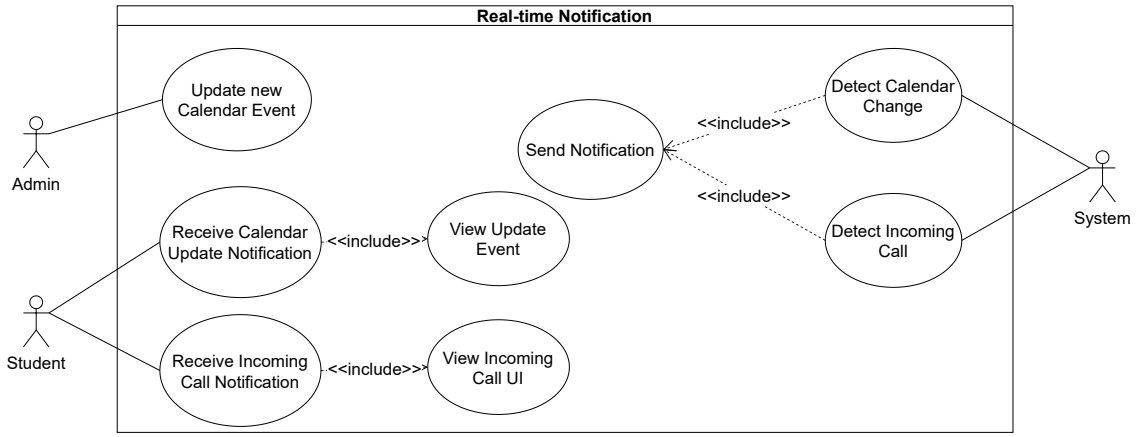


Figure 6: Real Time Notification use case diagram

Description: The push notification system alerts the student 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 student stays updated on any important events or incoming calls.

Pre-conditions:

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

Post-conditions:

- The real time notification will pop up on the student's device to attract the student'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.

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 student to handle the incoming call (accept or decline)

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

III/ Methodologies

This section offers an overview of the methodologies used throughout the system development after examining the objectives of the USTH Connect. It ensures that essential actions are executed in a systematic and well-planned manner.

1. Use Case Implementation

This section provides an overview of the implementation of use cases in the system, ensuring that all critical steps are carried out and described in detail.

1.1 Sync Calendar Sequence Diagram

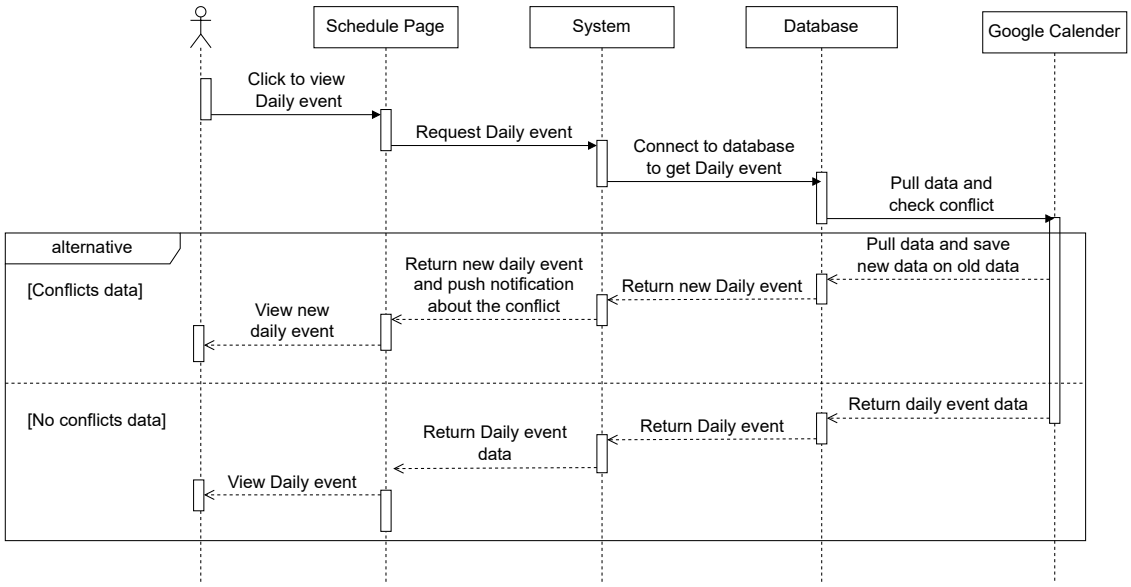


Figure 7: Synchronize Calendar sequence diagram

Description: This sequence diagram describes how students view their daily events from the scheduling system. First, the system gets data from the database and checks it with Google Calendar. Then, when a lecturer or university updates a new daily event, which is in conflict with its database, the system gets conflict data and saves it to the database. Finally, the students see the new event and send notification about the conflicts.

Pre-conditions:

- Students are logged in.
- Students have access to the Schedule Page.
- System connects to the database and Google Calendar.

Post-conditions:

- System connects to the database and Google Calendar.
- System sends a notification of an update daily event.
- The database is updated with the new data from Google Calendar.

1.2 StudyBuddy Sequence Diagram

Description: This sequence diagram describes how students connect with others. When students access the Study Buddy page, they see a list of recommended study buddies, which is generated by a model based on their database. Students can communicate with the others by using 2 features: message and calling.

Pre-conditions:

- Students are logged in.
- Students have access to the Study Buddy page.
- Students are logged in to an SIP account.
- Both students are connected to each other.

Post-conditions:

- Students view a list of recommended study buddies.
- Students view and edit their profiles.
- Students can contact each other while using the message and call of the application.

1.3 Real-Time Notification Sequence Diagram

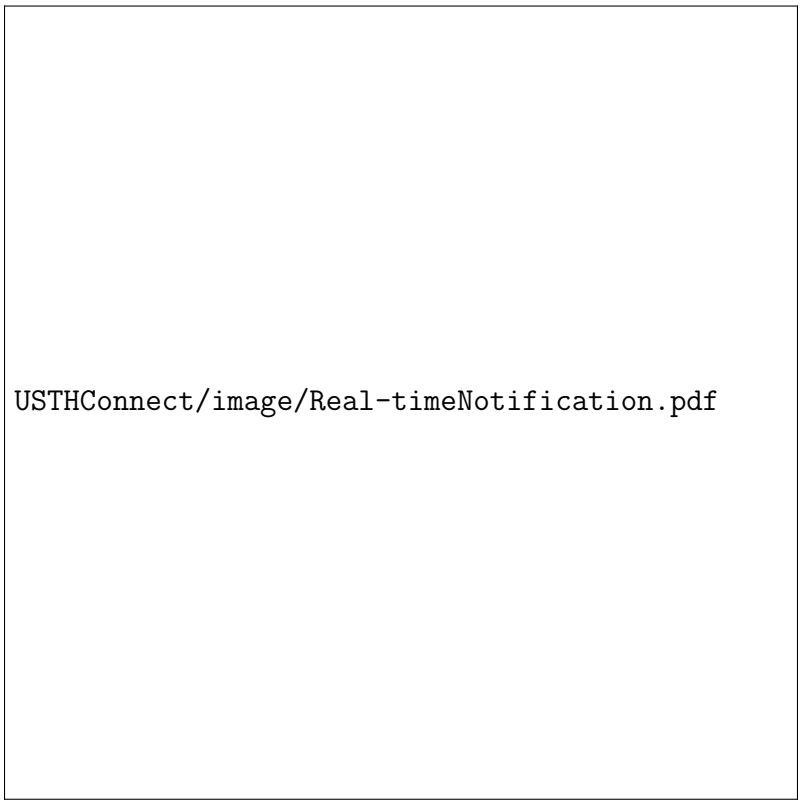


Figure 8: Real Time Notification sequence diagram

Description: This sequence diagram describes how systems send notifications to students to view update events in the calendar and incoming calls from other students. First, the system connects to Google Calendar and SIP Server. Then, if the calendar changes, systems detect the calendar, which is updated by the lecturer or university, fetch the calendar and send notification. Second, the system connects to the SIP Server. When a student calls another, the system detects that and sends notification to the student.

Pre-conditions:

- Students are logged in.
- System connected to both SIP Server and Google Calendar.

Post-conditions:

- Students receive notification about the update calendar.
- Students receive a notification and can access the incoming call interface.

1.4 SIP-Based Communication Analysis

This section below provides how the contact between users works utilising the help of SIP-based service Linphone.

1.4.1 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).

1.4.2 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.

1.4.3 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.

1.4.4 Communication Handling

1.4.4.1 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.

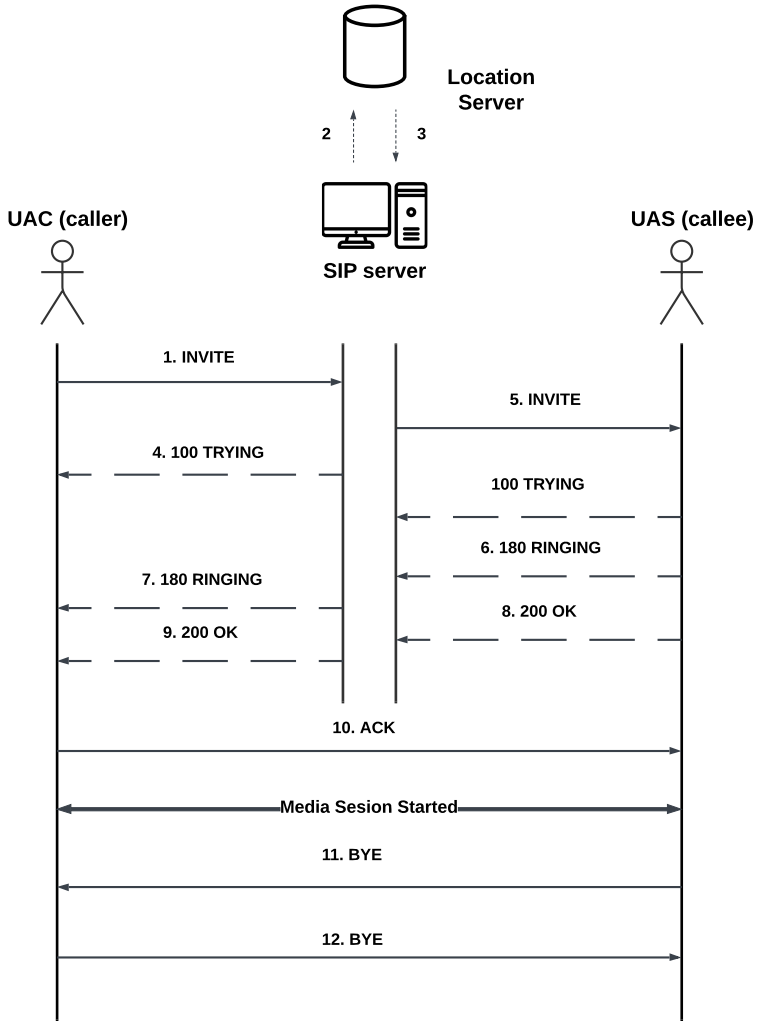


Figure 9: SIP signaling and key components

1.4.4.2 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.

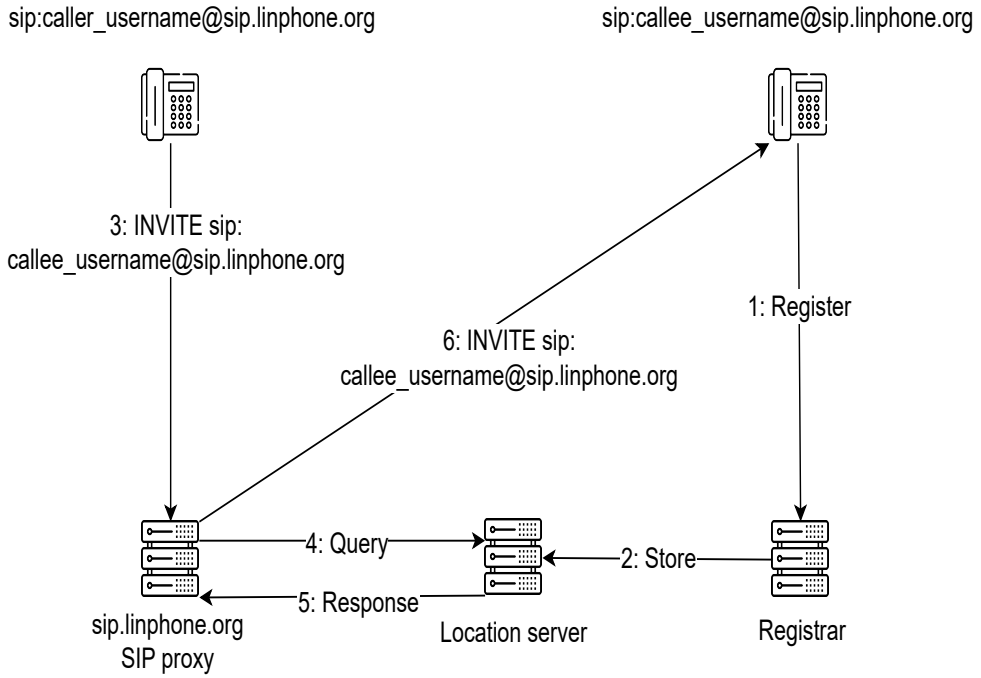


Figure 10: SIP registration and locating callee

1.4.4.3 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.

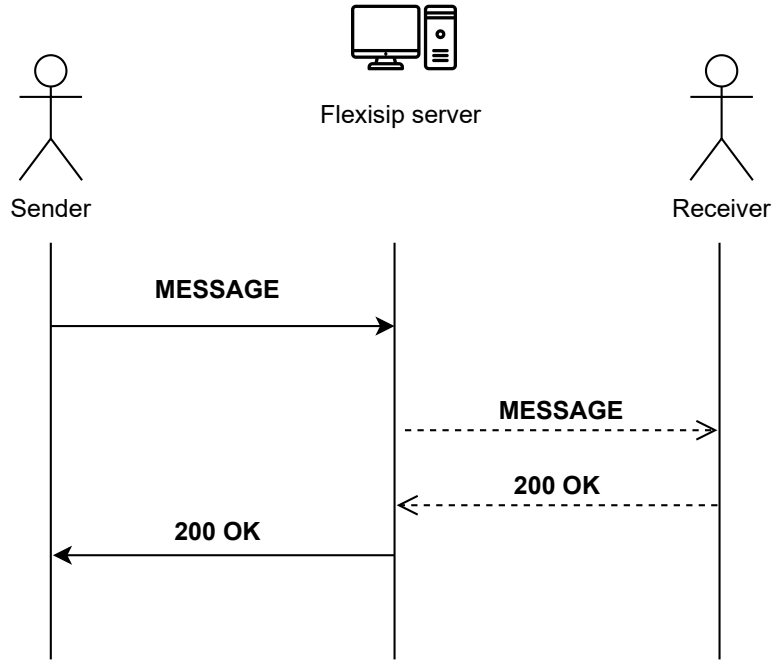


Figure 11: Simple instant messaging flow

2. System Architecture

This section provides an overview of the technologies and components used in the system. It focuses on the roles and interactions of various systems and services that work together in a layered architecture with a client-server relation, creating a seamless, integrated solution for university life assistance and student networking.

2.1 System Architecture Diagram

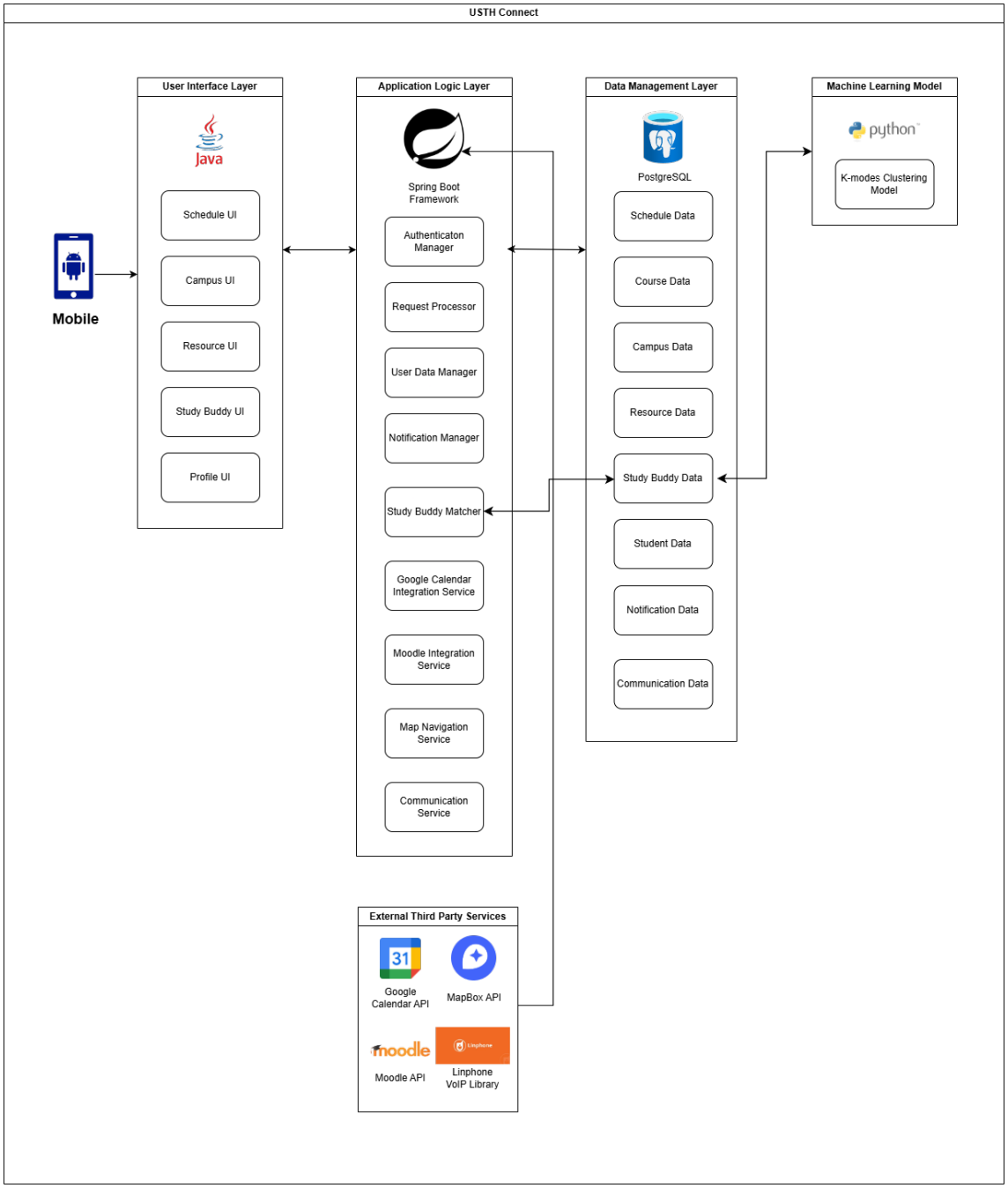


Figure 12: USTH Connect System Architecture Diagram

2.2 Backend - Spring Boot

The backend of the application is configured using Spring Boot, a well-known and powerful Java framework for its ability to simplify the backend development. Spring Boot handles the core logic of the application, which includes student authentication, processing data from third-party services, and providing APIs for the mobile frontend. It configures a seemless integration with various services such as Google Calendar, Mapbox, and Moodle, ensuring that schedule events, campus locations, and lecture resouces are always up-to-date and easy to access. The backend is responsible for processing and storing student data, managing roles with JWT for secure access, and applying RBAC to differentiate between ADMIN and USER roles. This security configuration ensures that students can only access the data and features which are allowed according to their given roles, enhancing the security and functionality of the system.

2.3 Mobile Frontend - Java (Android)

The mobile app is developed using Java for Android, which acts as the UI for the system. This frontend enables students to interact with various features, such as viewing events from Google Calendar, accessing campus location through Mapbox, and browsing academic resources coming from Moodle. The app fetches data from the backend through APIs, ensuring that it always displays the latest information that the backend system can provide. For the Google Calendar implementation, the app periodically retrieves and displays these followings events, which makes it easier for students to stay updated on their study schedules. Mapbox provides an overview map of a campus, allowing students to locate buildings and study halls using their latitude and longitude data. Additionally, the app integrates the Study-Buddy feature, where students can find potential study partners based on their personal description, interests and habits. Moreover, it supports real-time audio and message communication through the Linphone VoIP library.

The frontend UI service is necessary in our system architecture because it acts as the bridge between the user and the system's backend service. The UI provides students with an user-friendly interface to access the application features such as schedule view, campus navigation, resource access, and academic collboration. Without the frontend UI, the system would lack a method for students to communicate to the backend service, therefore the application is considered unusable. The frontend ensures that the students have a smooth and comfortable experience while using our platform, which helps them improving their academic life and connecting with their peers seamlessly.

2.4 Machine Learning - K-modes Clustering

The StudyBuddy feature of the system using a Python-based machine learning model that uses the K-modes clusteting algorithm to match students based on their personal interests and study preferences. When students enter details about their personality traits, hobbies, learning style and academic goals, this data is sent to the backend, which processes it using the K-modes algorithm. After receiving data, the algorithm groups students with similar characteristics, enabling the system to send back the proper recommendation to the students. Then, the potential study partners will be displayed in the mobile app, allowing students to connect to each others who have compatible interests. This feature is implemented using Python's machine learning libraries, which provide the necessary methods for clustering and pattern recognition in student matching.

There are many advantages for why Python was chosen over Java for the implementation of the machine learning model in our system architecture. First, the libraries which are designed specifically for machine learning configuration are offered by Python, such as (Add used libraries). These libraries provide powerful and optimized tools for implementing our algorithm which is K-modes clustering, fostering faster development and experimentation. Regarding Java, it lacks various of tools which directly support the implementation process and require more complex and custom implementations. Additionally, Python provides direct support for the K-modes clustering algorithm through the kmodes library, which is a key reason why we choose Python over Java in the implementation of the machine learning model. As regards to the process of prototyping and experimentation, which is significant for fine-tuning machine learning models, Python allows a faster and short duration for this process. Finally, Python integrates seamlessly with the backend services, ensuring easy communication with other system components through APIs. In conclusion, Python is considered as the most suitable choice for implementing the required machine Learning model which plays a crucial role in the StudyBuddy feature.

2.5 Communication - Linphone VoIP

To integrate real-time communication between matched students, the system implements Linphone, an open-source VoIP library. Linphone allows students to initiate audio calls, enabling seamless communication for students who wish a collaboration on projects or study-together session. After the StudyBuddy algorithm matches two studens, they can contact and communicate each other through the app through the app using the Linphone VoIP integration. Therefore, that students can interact with their matches without relying on external messaging or calling services. Linphone provides a reliable and efficient platform for voice communication, contributing to the goal of encouraging more interaction and collaboration between students in the university.

2.6 How does it work together ?

This system works by seamlessly integrating various components. The Spring Boot backend serves as the brain of the operation, managing user accounts, data storage, and communication with external services like Google Calendar, Mapbox, Moodle, and Linphone VoIP library. The mobile app, built with Java, communicates with the backend to access real-time information. To connect students with similar interests, the system employs a smart matching algorithm (using Python). Once matched, students can easily make voice calls within the app thanks to Linphone's built-in calling feature. Each part of this system depends on the others. The app relies on the backend for accurate information, while the backend utilizes external services and the database to operate smoothly. The backend also processes the matching results and ensures secure data delivery to the app. In summary, this integrated system creates a comprehensive platform that enhances the university experience for students by supporting their academic and encouraging more communication between them.

3. Database Design

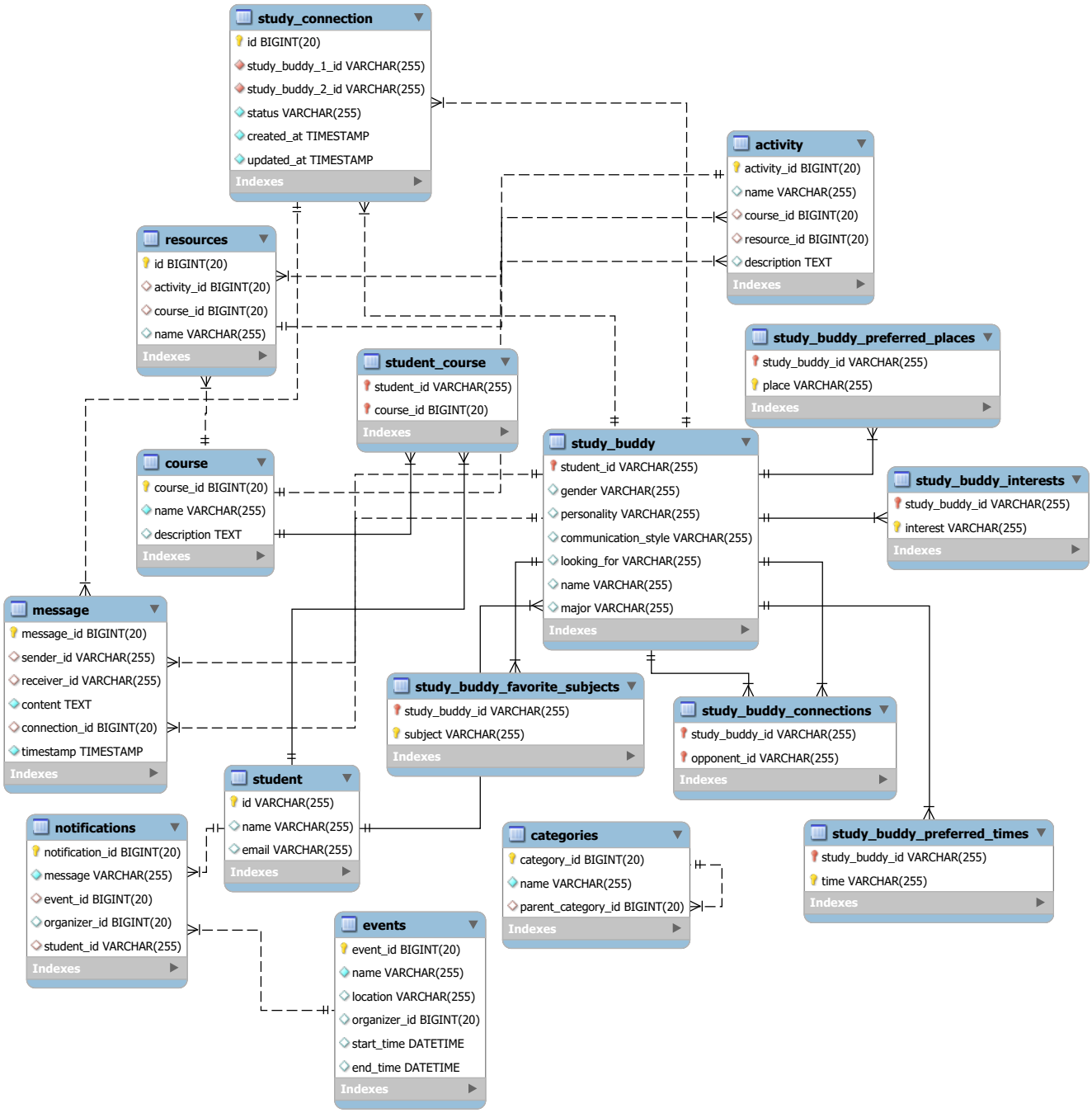
This section provides a detailed description of the database design for the USTH Connect application. It highlights the key requirements, objectives, and the structure that support the system's functionality. Each table and its relationships are thoroughly designed to handle different types of data, such as student profiles, schedule events, course resources, campus naviation, and StudyBuddy connections. This section also includes an Entity-Relationship diagram that visually represents the relationships between various entities.

3.1 Database Requirements and Objectives

The database for USTH Connect is designed to manage a wide range of information, including student profiles, schedules, study buddy connections, and course resources. The main objectives of the database design are:

- To provide a reliable and centralized system for storing academic and social data.
- To ensure data integrity and maintain relationships between tables for accurate reporting and queries.
- To support seamless integration with external services, such as Google Calendar and Moodle.
- To enable efficient data retrieval for features like StudyBuddy matching, schedule updates, and resource access.
- To maintain scalability and adaptability for increasing student data as the system grows.

3.2 Entity-Relationship Diagram


study_connection
id BIGINT(20)
study_buddy_1_id VARCHAR(255)
study_buddy_2_id VARCHAR(255)
status VARCHAR(255)
created_at TIMESTAMP
updated_at TIMESTAMP
Indexes
activity
activity_id BIGINT(20)
name VARCHAR(255)
course_id BIGINT(20)
resource_id BIGINT(20)
description TEXT
Indexes
resources
id BIGINT(20)
activity_id BIGINT(20)
course_id BIGINT(20)
name VARCHAR(255)
Indexes
study_buddy_preferred_places
study_buddy_id VARCHAR(255)
place VARCHAR(255)
Indexes
student_course
student_id VARCHAR(255)
course_id BIGINT(20)
Indexes
course
course_id BIGINT(20)
name VARCHAR(255)
description TEXT
Indexes
study_buddy
student_id VARCHAR(255)
gender VARCHAR(255)
personality VARCHAR(255)
communication_style VARCHAR(255)
looking_for VARCHAR(255)
name VARCHAR(255)
major VARCHAR(255)
Indexes
study_buddy_interests
study_buddy_id VARCHAR(255)
interest VARCHAR(255)
Indexes
message
message_id BIGINT(20)
sender_id VARCHAR(255)
receiver_id VARCHAR(255)
content TEXT
connection_id BIGINT(20)
timestamp TIMESTAMP
Indexes
study_buddy_favorite_subjects
study_buddy_id VARCHAR(255)
subject VARCHAR(255)
Indexes
study_buddy_connections
study_buddy_id VARCHAR(255)
opponent_id VARCHAR(255)
Indexes
student
id VARCHAR(255)
name VARCHAR(255)
email VARCHAR(255)
Indexes
notifications
notification_id BIGINT(20)
message VARCHAR(255)
event_id BIGINT(20)
organizer_id BIGINT(20)
student_id VARCHAR(255)
Indexes
categories
category_id BIGINT(20)
name VARCHAR(255)
parent_category_id BIGINT(20)
Indexes
study_buddy_preferred_times
study_buddy_id VARCHAR(255)
time VARCHAR(255)
Indexes
events
event_id BIGINT(20)
name VARCHAR(255)
location VARCHAR(255)
organizer_id BIGINT(20)
start_time DATETIME
end_time DATETIME
Indexes

3.3 Table and Relationships

This section provides a detailed description of the tables within the USTH Connect database and their relationship.

Table: Activity

Attributes and Data Types

Column Name	Data type	Description	Constraints
activity_id	bigserial	Unique identifier for the activity	Primary Key, Not Null
activity_name	varchar(255)	Name of the activity	
activity_type	varchar(255)	Type of activity	
due_date	timestamp	Deadline for the activity	
completion_status	varchar(255)	Status of the activity	
course_id	bigint	ID of the course associated with the activity	Foreign Key
resource_id	bigint	ID of a resource associated with the activity	Foreign Key

Table 7: Attributes and data types for the `Activity` table

Constraints

- **Primary Key**: `activity_id`
- **Foreign Key** (`course_id`): References `course(course_id)`.
- **Foreign Key** (`resource_id`): References `resources(id)`.

Relationships

- Each `activity` belongs to one `course`.
- Each `activity` may have one associated `resource`.

Table: Course

Attributes and Data Types

Column Name	Data type	Description	Constraints
course_id	bigserial	Unique identifier for the course	Primary Key, Not Null
short_name	varchar(255)	Short Name of the course	Not Null
full_name	varchar(255)	Full Name of the course	Not Null
description	varchar(255)	Description of the course	
visibility	boolean	The current visibility status of the course	

Table 8: Attributes and data types for the `Course` table

Constraints

- **Primary Key**: `course_id`
- **Foreign Key** (`course_id`): In the `student_course` table, `course_id` references `course(course_id)`.

Relationships

- Each `course` can be associated with multiple `students` through the `student_course` table.

Table: Events

Attributes and Data Types

Column Name	Data type	Description	Constraints
event_id	serial	Unique identifier for the events	Primary Key, Not Null
event_name	text	Name of the event	
event_description	text	Description of the event	
event_start	timestamp	Starting time of the event	
event_end	timestamp	Ending time of the event	
location	varchar(255)	Location of the Event	
organizer_id	int	Unique identifier for the organizer	
google_event_id	varchar(255)	Unique identifier generated by the google	

Table 9: Attributes and data types for the `Events` table

Constraints

- **Primary Key**: `event_id`.
- **Foreign Key** (`organizer_id`): References `organizer(id)`.
- **Unique Key** (`google_event_id`): Ensures uniqueness of the Google Event ID.

Relationships

- Each `event` is organized by one `organizer`.
- Each `event` is associated with one `location` from `maps`.

Table: Maps

Attributes and Data Types

Column Name	Data type	Description	Constraints
id	serial	Unique identifier for the map location	Primary Key, Not Null
location	varchar(255)	The name of the location	Default: "No Location Provided"
location_value	varchar(255)	Value the location	
Latitude	double precision	Latitude of the location	
Longitude	double precision	Longitude of the location	

Table 10: Attributes and data types for the `Maps` table

Constraints

- **Primary Key**: `id`.
- **Unique Key**: `location`.

Relationships

- Each `location` in `maps` can be referenced in `events`.

Table: Message

Attributes and Data Types

Column Name	Data type	Description	Constraints
id	bigserial	Unique identifier for the message	Primary Key, Not Null
connection_id	bigint	Reference to the connection in the `Study Connection` table	Foreign Key, Not Null
sender_id	varchar(255)	Identifier of the sender	Foreign Key, Not Null
receiver_id	varchar(255)	Identifier of the receiver	Foreign Key, Not Null
content	varchar(255)	Text content of the message	Not Null
created_at	timestamp	Timestamp when the message was created	Not Null, Default: `now()`
is_read	boolean	Indicates if the message has been read	Not Null, Default: `false`

Table 11: Attributes and data types for the `Message` table

Constraints

- **Primary Key**: `id`.
- **Foreign Key**:
  - `connection_id` references `id` in the `Study Connection` table, with `ON DELETE NO ACTION`.
  - `receiver_id` references `student_id` in the `Study Buddy` table, with `ON DELETE NO ACTION`.
  - `sender_id` references `student_id` in the `Study Buddy` table, with `ON DELETE NO ACTION`.

Indices

- `idx_connection_id`: Index on `connection_id` for optimized query performance.
- `idx_receiver_id`: Index on `receiver_id` for optimized query performance.
- `idx_sender_id`: Index on `sender_id` for optimized query performance.

Relationships

- Each `Message` is associated with a specific `Connection`.
- `sender_id` and `receiver_id` reference students in the `Study Buddy` table.

Table: Notifications

Attributes and Data Types

Column Name	Data type	Description	Constraints
notification_id	serial	Unique identifier for the notification	Primary Key, Not Null
student_id	varchar(255)	Identifier of the student receiving the notification	Foreign Key, Not Null
message	text	The content of the notification	Not Null
created_at	timestamp	Timestamp when the notification was created	Not Null, Default: `CURRENT_TIMESTAMP`
is_read	boolean	Indicates if the notification has been read	Not Null, Default: `false`
event_id	integer	Reference to the event related to the notification	Foreign Key
organizer_id	integer	Reference to the organizer of the event	Foreign Key

Table 12: Attributes and data types for the `Notifications` table

Constraints

- **Primary Key**: `notification_id`.
- **Foreign Key**:
  - `student_id` references `id` in the `Student` table, with `ON DELETE NO ACTION`.
  - `event_id` references `event_id` in the `Events` table, with `ON DELETE NO ACTION`.
  - `organizer_id` references `id` in the `Organizers` table, with `ON DELETE NO ACTION`.

Relationships

- Each `Notification` is associated with a specific `Student`.
- `event_id` references the `Events` table, linking notifications to events.
- `organizer_id` references the `Organizers` table, linking notifications to the organizer of the event.

Table: Organizers

Attributes and Data Types

Column Name	Data type	Description	Constraints
id	serial	Unique identifier for the organizer	Primary Key, Not Null
email	varchar(255)	Email of the organizer	Not Null, Unique
name	varchar(255)	Name of the organizer	

Table 13: Attributes and data types for the `Organizers` table

Constraints

- **Primary Key**: `id`.
- **Unique Key**: `email`.

Relationships

- `Organizers` table contains information about the organizers who manage events or resources.

Table: Resources

Attributes and Data Types

Column Name	Data type	Description	Constraints
id	bigserial	Unique identifier for the resource	Primary Key, Not Null
resource_type	varchar(255)	Type of the resource (e.g., video, document, etc.)	
resource_name	varchar(255)	Name of the resource	
course_id	bigint	Reference to the course related to the resource	Foreign Key
activity_id	bigint	Reference to the activity related to the resource	Foreign Key
file_url	varchar(255)	URL of the resource file	
resource_id	bigint	Auto-generated identifier for the resource	Not Null

Table 14: Attributes and data types for the `Resources` table

Constraints

- **Primary Key**: `id`.
- **Foreign Key**:
  - `activity_id` references `activity_id` in the `Activity` table, with `ON DELETE NO ACTION`.
  - `course_id` references `course_id` in the `Course` table, with `ON DELETE NO ACTION`.

Relationships

- Each `Resource` is associated with a specific `Activity` and a `Course`.

Table: Student

Attributes and Data Types

Column Name	Data type	Description	Constraints
id	varchar(255)	Unique identifier for the student	Primary Key, Not Null
fullname	varchar(255)	Full name of the student	
gender	smallint	Gender of the student (e.g., Male, Female)	
major	varchar(255)	Major field of study	
dob	timestamp(6)	Date of birth of the student	
study_year	smallint	The academic year of study	
full_name	varchar(255)	Full name of the student	Not Null
password	varchar(255)	Password for student account	
role	varchar(255)	Role of the student (e.g., student, admin)	
phone	varchar(255)	Contact phone number	
email	varchar(255)	Email address of the student	
sip_username	varchar(255)	SIP username for communication	
sip_password	varchar(255)	SIP password for communication	

Table 15: Attributes and data types for the `Student` table

Constraints

- **Primary Key**: `id`.

Relationships

- The `Student` table contains information about individual students.
- Each student can be enrolled in one or more courses, as represented in the `Student Course` table. This is a many-to-many relationship between the `Student` table and the `Course` table.

Table: Study Buddy

Attributes and Data Types

Column Name	Data type	Description	Constraints
student_id	varchar(255)	Unique identifier for the student	Primary Key, Not Null, Unique
gender	varchar(255)	Gender of the student	
personality	varchar(255)	Personality description of the student	
communication_style	varchar(255)	Preferred communication style	
looking_for	varchar(255)	What the student is looking for in a study buddy	
name	varchar(255)	Full name of the student	
major	varchar(255)	Major field of study of the student	

Table 16: Attributes and data types for the `study_buddy` table

Constraints

- **Primary Key**: `student_id`.
- **Unique**: `student_id` must be unique.
- **Foreign Key**:
  - `student_id` references `id` in the `student` table, with `ON DELETE NO ACTION`.

Relationships

- Each `Study Buddy` is associated with a student through the `student_id`.

Table: Study Buddy Connections

Attributes and Data Types

Column Name	Data type	Description	Constraints
study_buddy_id	varchar(255)	ID of the first study buddy	Primary Key, Not Null
opponent_id	varchar(255)	ID of the second study buddy (opponent)	Primary Key, Not Null

Table 17: Attributes and data types for the `study_buddy_connections` table

Constraints

- **Primary Key**: `study_buddy_id`, `opponent_id`.
- **Foreign Key**:
  - `study_buddy_id` references `student_id` in the `study_buddy` table, with `ON DELETE NO ACTION`.
  - `opponent_id` references `student_id` in the `study_buddy` table, with `ON DELETE NO ACTION`.

Relationships

- Each `Study Buddy Connection` is associated with two `Study Buddy` entities (study_buddy_id and opponent_id).

Table: Study Buddy Favorite Subjects

Attributes and Data Types

Column Name	Data type	Description	Constraints
study_buddy_id	varchar(255)	ID of the study buddy	Primary Key, Not Null
subject	varchar(255)	Name of the favorite subject	Primary Key, Not Null

Table 18: Attributes and data types for the `study_buddy_favorite_subjects` table

Constraints

- **Primary Key**: `study_buddy_id`, `subject`.
- **Foreign Key**:
  - `study_buddy_id` references `student_id` in the `study_buddy` table, with `ON DELETE NO ACTION`.

Relationships

- Each `Study Buddy Favorite Subject` is associated with a specific `Study Buddy`.

Table: Study Buddy Interests

Attributes and Data Types

Column Name	Data type	Description	Constraints
study_buddy_id	varchar(255)	ID of the study buddy	Primary Key, Not Null
interest	varchar(255)	Interest of the study buddy	Primary Key, Not Null

Table 19: Attributes and data types for the `study_buddy_interests` table

Constraints

- **Primary Key**: `study_buddy_id`, `interest`.
- **Foreign Key**:
  - `study_buddy_id` references `student_id` in the `study_buddy` table, with `ON DELETE NO ACTION`.

Relationships

- Each `Study Buddy Interest` is associated with a specific `Study Buddy`.

Table: Study Buddy Preferred Places

Attributes and Data Types

Column Name	Data type	Description	Constraints
study_buddy_id	varchar(255)	ID of the study buddy	Primary Key, Not Null
place	varchar(255)	Name of the preferred place	Primary Key, Not Null

Table 20: Attributes and data types for the `study_buddy_preferred_places` table

Constraints

- **Primary Key**: `study_buddy_id`, `place`.
- **Foreign Key**:
  - `study_buddy_id` references `student_id` in the `study_buddy` table, with `ON DELETE NO ACTION`.

Relationships

- Each `Study Buddy Preferred Place` is associated with a specific `Study Buddy`.

Table: Study Buddy Preferred Times

Attributes and Data Types

Column Name	Data type	Description	Constraints
study_buddy_id	varchar(255)	ID of the study buddy	Primary Key, Not Null
time	varchar(255)	Preferred time of the study buddy	Primary Key, Not Null

Table 21: Attributes and data types for the `study_buddy_preferred_times` table

Constraints

- **Primary Key**: `study_buddy_id`, `time`.
- **Foreign Key**:
  - `study_buddy_id` references `student_id` in the `study_buddy` table, with `ON DELETE NO ACTION`.

Relationships

- Each `Study Buddy Preferred Time` is associated with a specific `Study Buddy`.

Table: Study Connection

Attributes and Data Types

Column Name	Data type	Description	Constraints
id	bigint	Unique identifier for the connection	Primary Key, Not Null
study_buddy_1_id	varchar(255)	ID of the first study buddy	Foreign Key, Not Null
study_buddy_2_id	varchar(255)	ID of the second study buddy	Foreign Key, Not Null
status	varchar(255)	Status of the connection	Not NullI
created_at	timestamp	Timestamp when the connection was created	Not Null
updated_at	timestamp	Timestamp when the connection was last updated	Not Null

Table 22: Attributes and data types for the `study_connection` table

Constraints

- **Primary Key**: `id`.
- **Foreign Key**:
  - `study_buddy_1_id` references `student_id` in the `study_buddy` table, with `ON DELETE NO ACTION`.
  - `study_buddy_2_id` references `student_id` in the `study_buddy` table, with `ON DELETE NO ACTION`.

Relationships

- Each `Study Connection` is associated with two `Study Buddy` entities (study_buddy_1_id and study_buddy_2_id).

IV/ Clustering Model for Recommendation

1. K-Mode Algorithm

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

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

Let $\{Q_1, Q_2, ..., Q_k\}$ be the modes of clusters $\{C_1, C_2, ..., C_k\}$. The cost function can be defined by taking the total dissimilarities of each sample and its assigned cluster's mode(**?**):

$$E = \sum_{l=1}^{k} \sum_{i=1}^{n} y_{i,l} d(X_i, Q_l)$$

where $y_{i,l}$ is a binary indicator variable (1 if X_i refers to cluster l, 0 otherwise), k is the number of clusters, n is the total number of samples, $d(X_i, Q_l)$ is the dissimilarity between sample X_i and the mode Q_l. The objective is to minimize the value of the cost function by following the following steps:

1. Randomly select K initial modes as K clusters.
2. Calculate the dissimilarity between data points and the mode of each cluster. Then 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 Steps 2 and 3 until every cluster has no changes after testing the whole dataset.

2. One-Hot Encoding

One-hot Encoding is a data preprocess method to change text format to numerical data type so that the machine can understand and process the data. It transforms

a variable with n distinct categories into n separate columns as new features. Each column is represented by two numbers: 0 and 1. The value 1 indicates the presence of a category, while the value 0 shows its absence.(**?**)

3. Evaluation Metrics

This part covers the intrinsic evaluation metrics that be used to evaluate model performance.

3.1 Silhouette Coefficient

Choosing the optimal number of k value is crucial for the K-mode algorithm. To study the optimal value of k for a clustering algorithm, various methods have been proposed. While the ground truth is not available in a data set, intrinsic methods should be used to evaluate the clustering quality. These methods will examining how well the clusters are separated or compacted together (**?**). Silhouette Coefficient is one of the most commonly used intrinsic methods, which measures the quality of a cluster. The highest silhouette score calculated by the mean silhouette coefficient of all samples is considered the optimal number of clusters (**?**).

With a dataset D of n objects, in which D is partitioned into k clusters, $\{C_1, C_2, ..., C_k\}$. For each object $o \in D$, let $a(o)$ be the average distance of o to all other objects in the same cluster. Similarly, let $b(o)$ be the minimum average distance of o to other clusters. Suppose $o \in C_i (1 \leq i \leq k)$, then (**?**):

$$a(o) = \frac{\sum_{o' \in C_i, o \neq o'} dist(o, o')}{|C_i| - 1}$$

and

$$b(o) = \min_{C_j : 1 \leq j \leq k, j \neq i} \frac{\sum_{o' \in C_j} dist(o, o')}{|C_j|}$$

The silhouette coefficient of object o is defined as:

$$s(o) = \frac{b(o) - a(o)}{\max\{a(o), b(o)\}}$$

The silhouette coefficient is in $[-1, 1]$. The $a(o)$ value illustrated the compactness of the cluster which o belongs. Smaller $a(o)$ means the cluster is more compact. The $b(o)$ value shows how well-separated the cluster o is from other clusters. Larger $b(o)$ means the cluster is more separated. When the silhouette coefficient is close to 1, it indicates that the object o is compact and distance to others. On the other hand, if the silhouette coefficient is close to -1 (i.e., $a(o) > b(o)$), means the object o is far from its cluster and close to other clusters. This is the reason why a larger silhouette coefficient is considered better in terms of clustering quality.

3.2 Davies-Bouldin Index

Another evaluation metric that can be used to evaluate the clustering quality is the Davies-Bouldin Index (DBI). The DBI is calculated by the average similarity between each cluster and its most similar cluster. The lower the DBI value, the better the clustering quality. The DBI is defined as follows (**?**):

$$DBI = \frac{1}{n} \sum_{i=1}^{n} max_{j \neq i} \frac{s_i + s_j}{d(c_i, c_j)}$$

and

$$s_i = \frac{1}{n_i} \sum_{j=1}^{n_i} d(x_j, c_i)$$

In which n is the number of clusters, s_i is the average distance between each data point belonging to cluster i and the centroid of cluster i, $d(c_i, c_j)$ indicated the distance between the centroids of cluster i and j.

4. Model Implementation

4.1 Data Preprocessing

The input data is processed by applying one-hot encoding method. In the beginning, the student's data has been converted into a dense binary array before being used as input for the machine learning model. Moreover, since each student has to provide their information before having permission to use this application (choose at least one category for each attribute *(Interests, Favorite Subject,...)*), there are no missing data in our dataset. These tables below show the example for the data preprocessing part:

Major	**Interests**	...	**Favorite Subject**
DS	V-pop, Volleyball, Board games	...	Calculus, Linear Algebra, Artificial Intelligence
ICT	EDM, Board games, DIY, Movie, Netflix	...	Calculus, Chemistry, Programming
ICT	Football, Cooking, Food tour, Travel, Rap	...	Programming

Table 23: Data before preprocessing

DS	**ICT**	**V-pop**	**Volleyball**	**Board games**	**EDM**	**...**	**Programming**
1	0	1	1	1	0	...	0
0	1	0	0	1	1	...	1
0	1	0	0	0	0	...	1

Table 24: Data after preprocessing

4.2 Model Training

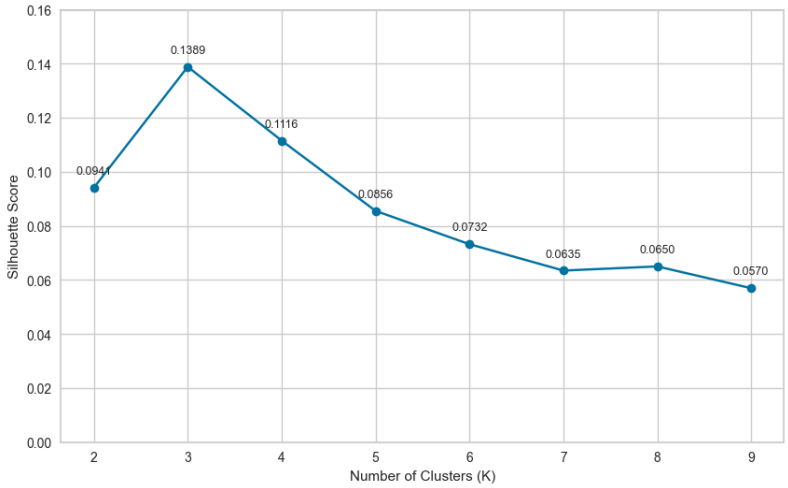


Figure 13: Optimal clusters K

To determine the optimal number of clusters K, the K-Modes clustering model is trained with different K values varying from 2 to 9 and select the best model according to the cost function. For each training iteration, the K-Modes algorithm is executed multiple times (15 in our problem) with different initial centroids seeds. The final result is the best initialized centroids in terms of cost function.(**?**)
After the training is completed, each student is assigned to their predicted cluster, which will be used for the recommendation part.

4.3 Student Recommendation

Firstly, the information of new student is encoded by one-hot encoding and fed into the model. The output returned is the predicted cluster to which the new student belongs. Depending on this prediction, students within the same cluster are chosen to recommend.

V/ System Design and Implementation

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.

1. Hardware and Software Setup

The design and implementation of the system started by setting up the hardware and software environment. Regarding the hardware setup, we have development machines for hosting backend services and the PostgreSQL database, as well as user devices which are mostly Android smartphones or virtual Android devices for accessing features like Schedule view, Campus navigation, and Moodle resources.

On the software side, the operating system used for hosting backend services and the database is Window, while Android acted as the primary platform for the mobile application. For the relational database management system, we choosed PostgreSQL to handle student data, calendar events, location details, and data for StudyBuddy feature. The backend framework used Spring Boot for managing REST API endpoints, authentication, authorization, as well as data synchronization. To potentially support the development of the application, we used Android Studio as the main environment for building the application using Java language, as it contains of libraries like Retrofit and MapBox SDK.

2. Application Software

The application was divided into several modules, as each of them serve a specific feature. The Authentication and Authorization Module implemented JWT-based authentication and managed RBAC for ADMIN and USER roles. The Google Calendar Integration Module was responsible for fetching event data, detecting updates, and notifying students about changes. The MapBox Integration Module enabled storage of campus's study places coordinates and overall campus map viewing.

Regarding Moodle Resource Module, we set up a communication with Moodle APIs to retrieve course materials such as slides, source code, and PDFs. About StudyBuddy Matching Module, we implemented a machine learning recommendation system to suggest matches based on shared interests, collecting student profile data and offering text-based chat and SIP-based VoIP audio call using the Linphone library. The Notification System provided real-time alerts for calendar updates and incoming calls.

3. Initial Testing

To ensure all components were functioning correctly and fully compatible, we conducted initial testing divided into hardware, software, and integration testing. Hardware testing involved verifying the performance and the reliability of development machines, servers, as well as networking environment. Regarding software testing,

we focused on verifying proper installation, configuration, and performance. The Windows operating system installed on the backend machines was tested for resource allocation, security settings, and reliability. PostgreSQL was initialized with schemas and sample data, testin with CRUD operations to ensure smooth data handling. Spring Boot REST API endpoints were tested individually using Postman to confirm proper functionality, including following the JWT authentication with token.

Integration testing validated interactions between the various system components. For Google Calendar synchronization, we created dummy events and updated with dummy data to verify proper fetching and real-time notifications. MapBox integration was tested by checking stored campus location data with real-world coordinates, ensuring the accuracy and functionality for campus navigation on multiple devices, including virtual devices. Moodle API communication was verified by simulating the process of course material upload and confirmationo of successful retrieval of slides, source code, and PDFs. Additionally, the StudyBuddy matching feature was tested using both dummy and real profiles to validate the recommendation system, ensuring the high accuracy matches based on profile information. Chat functionality was tested for text-based messaging and SIP-based VoIP audio calls, with the Linphone library ensuring seamless connection and clear audio quality. Lastly, the notification system was tested by simulating calendar dummy events updates and incoming calls to confirm notification prompt message. This testing phase ensured a reliable and fully functional system, ready for deployment.

4. Application Software

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

- **Authentication and Authorization Module:**
  - Implements JWT-based authentication.
  - Manages RBAC for ADMIN and USER roles.
- **Google Calendar Integration Module:**
  - Fetches event data from Google Calendar.
  - Detects event updates and sends notifications to the student 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 student 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. 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.

VI/ Results and Discussion

This section focus on the results and discuss the outcome of the USTH Connect project, focusing on key aspects such as model performance, application development progress, and challenges we encounter during the development.

1. Results

1.1 Mobile App Results

In this part we will show the result of the application developement, mainly about what we have developed so far and some specific results of the application.

Application Features Progress

The USTH Connect mobile application has made significant progress in its development, results in successful implementation of several features. Apart from the configuration of login and registration system using JWT authentication and RBAC, we manage to develop a functions for synchronizing calendar events with the school's Google Calendar. In our application, we have displayed the schedule according to the student's major and study year to save student effort finding the correct schedule. Moreover, students can view all the lectures events for a specific subject. Also, the system periodically fetchs events and detects event changes to notify the students about their schedule. We have configured a map feature using MapBox SDK to display campus maps with location markers for each buildings and facilities which supports student efficiently navigate within the university. Regarding lecture resources, we have successfully linked the application to Moodle, allowing students to retrieve and view course materials directly from the app. For the main feature StudyBuddy, we have developed a recommendation system using machine learning model. By retrieving student's data, we can successfully return and display a list of compatible partners for students to enrich their social life. The app provides a chat platform and audio call using Linphone VoIP library for real-time communication between matched students. Especially, the audio call has been improved with real-time connection and notifications. However, the chat platform yet does not have a notification for incoming messages and more appealing UI for better application experience. Also, we have not figure out an efficient way to store the image to support Study Buddy feature in displaying images for each users.

Mobile-Specific Results

We tested across a variety of devices (emulator and Android) and the performance was consistent, with the two main factors: App Launch Time and Network Usage. The average app launch time was 2 seconds on Android devices and 4 seconds on Emulator devices. The app also optimized data usage by caching schedule events, map location and course materials, which partly reduce a dependency on continuous internet access for those features. By combining many features of other applications

into one, USTH Connect offer students an efficient tool. This eliminates the need for multiple applications, saving students time and enhancing their overall experience.

1.2 Machine Learning Results

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

Model Performance

Model	Silhouette Score	Davies-Bouldin Score
K-Modes	0.1389	3.4489

Table 25: Model Performance

Table **??** presents the values of the Silhouette and Davies-Bouldin score. The silhouette score indicates how close the data point is to its cluster compared to the others, while the Davies-Bouldin score shows how high the overall quality of the clustering model. With a Silhouette score of 0.1389 and a Davies-Bouldin score of 3.4489, the centroids of each cluster are relatively close to each other, and the boundaries between clusters may not be ideal.

The same pre-processing method and K-Modes model is also applied to the Zoo dataset (with true label) and gives the result: 0.5362 of Silhouette Score and 1.21 of Davies-Bouldin Score. The Kuhn-Munkres(**?**) algorithm is used to match predicted clusters with the true types of animals to maximize accuracy. This performance is notably better than the results obtained in our dataset. The possible reason for this may be the higher dimensionality of the input data in our dataset, which can increase noise and complexity for clustering.

Similar System

The same pre-preocessing method and K-Modes model is applied to the Zoo dataset (with true label) and gives the result: 0.5362 of Silhouette Score and 1.21 of Davies-Bouldin Score. The Kuhn-Munkres(**?**) algorithm is used to match the predicted clusters with the true types of animals to maximize accuracy.

2. Discussion

2.1 Intepretation of Results

The development of USTH Connect has show significant progress in creating a comprehensive mobile application designed to improve university life. The app efficiently combines various features, including Google Calendar synchronization, campus naviation, access to Moodle resources, and the StudyBuddy recommendation system.

Although the clustering algorithm provides valuable insights into study partner compatibility, evaluation metrics indicates that the need of improvement in cluster separation and cohesion. The low Silhouette score and high Davies-Bouldin score show the need for better feature selection or model tuning to enhance clustering performance.

Regarding the mobile application, fetching and caching data like schedules and course materials has dramatically enhanced usability, enabling partial offline access. However, the lack of chat message notifications and a more optimized image storage system reduce the overall user experience.

2.2 Comparison to Similar System

Unlike traditional university assistance apps that mostly focus on course and student management or navigation, USTH Connect combines multiple features into a single application, implementing academic tools with a social networking developed by machine learning. In contrast to other platforms which may recommend compatible groups or clubs, USTH Connect's StudyBuddy feature offers a more personalized study partner recommendation system. In terms of user experience, many similar systems lack seamless communication methods, such as real-time audio calling with real-time notification, which is one of the features that USTH Connect offers. However, the need of improvements in UI design and notification system to surpass other platforms.

2.3 Challenges and Limitations

Despite the progress made, the project faced serveral challenges and limitations. First, the current system for storing and retrieving student images in the StudyBuddy feature is inefficient, resulting in a poor user experience and diminishing the visual appeal of the interface. Additionally, the performance of the clustering algorithm indicates the need for further research in feature selection, parameter tuning, or exploring alternative algorithms to optimize its effectiveness. While notifications for calendar events and VoIP calls are functioning as expected, the system lacks of notifications for incoming chat messages and matched partners in the StudyBuddy feature, which needs to be addressed. Furthermore, certain sections of the app, such as the chat interface and the StudyBuddy partner profile view, require improved visual design and user-friendly features to enhance the overall user experience. About the cross-device testing problem, although the application has been tested on emulators and Android devices, broader cross-device testing is necessary to ensure compatibility across different screen sizes and operating system versions. Another critical aspect is the Moodle integration, where the app currently relies on a custom-built Moodle system rather than integrating with the official Moodle platform provided by the school. This limitation prevents synchronization with official course materials, restricting students' seamless access to academic resources and course content. Full integration with the official Moodle platform is essential to enhance the app's functionality and provide students with comprehensive support for their academic needs.

VII/ Conclusion & Future Work

1. Conclusion

The USTH Connect mobile application has achieved significant progress in implementing essential university functions into a single platform. Features like calendar synchronization, campus naviation, Moodle integration, and the StudyBuddy recommendation system help simplify and improve student life. Along with obtained successes, some challenges are still there, especially in improving chat features, refining StudyBuddy features, and optimizing the clustering algorithm. The StudyBuddy feature's machine learning model indicates the need of improvement, especially the clustering algorithm's performance. The Silhouette and Davies-Bouldin scores suggest that the current model's cluster separation is not efficient as expected, which requires the further improving of the feature.

On the other hand, the app's ability to fetch, manage and cache data for offline access, combined with real-time communication features such as audio calling and notifications, which improves the overall user experience.

In summary, USTH connect has shown potential in becoming a vital tools for students at the University of Science and Technology of Hanoi, enhancing their academic and social experiences. Despite some challenges, its development is positive, and with future improvement, it can become a fully integrated, user-friendly application.

2. Future Work

To contribute to the current achievements of USTH Connect and address its existing limitations, future development will focus on several key areas. Enhancing the StudyBuddy is our first priority, as it is our key feature of the application, involving the improvement of the machine learning model used for recommending study partners. We will find an alternative clustering algorithms, refining the feature selection process, and optimizing the image storage system for user profiles to improve the user experience to the full extent.

Improving the application's user interface is another significant section that we need to focus on. Sections like the chat interface, the StudyBuddy profile views, and other areas will be refined to improve the overall visual and make user more engaged into our application. To declare the problems about our user interface, user feedback play a vital role in creating these enhancement suggestion to verify a design that match the student expectations and preferences. Additionally, optimizing the notification system will be a key improvement as it associates with many key functionalities like schedule retrieval and StudyBuddy communication features. We will focus to enable a notification for chat features, as well as notify users about new StudyBuddy matches and importance calendar events such as Exam day or Meetings.

Regarding the clusteing algorithm, the current one for StudyBuddy feature will need further optimization through testing alternative methods, fine-tuning parameters, and combining additional relevant features to improve the accuracy of the recommendation. Moreover, the model performance could be improved by reducing the dimensionality of the input data and integrating deep learning for features extraction. With this integration, the model can reduce noise features while still retain the characteristics of user. After the application is running, user behaviors such as the number of likes, time spent on recommending part, and the number of connected friends will be collected for further analysis and better user experience understanding. Similarly, the platform compatibility and cross-device testing will be improved to ensure the application performs consistently across various screen sizes and operating systems. This work will help deliver a more unique and unified experience for all users.

Moreover, full integration with the university's Moodle system is necessary to provide students with up-to-date course materials and academic resources. Finally, future versions of the application will introduce more features based on our user feedback, enabling USTH Connect to address their needs and contribute more to the improvement of USTH student's university life.

By addressing these key areas, USTH Connect can evolve to meet the growing needs of students, providing even greater benefits to both users and the university community. The continuous development of the app will ensures the valuable and vital tools for students. Through future improvements, USTH Connect has the potential to become a necessary platform that supports both academic and social activities, as well as encourage a more connected and engaged academic environment.

Appendices

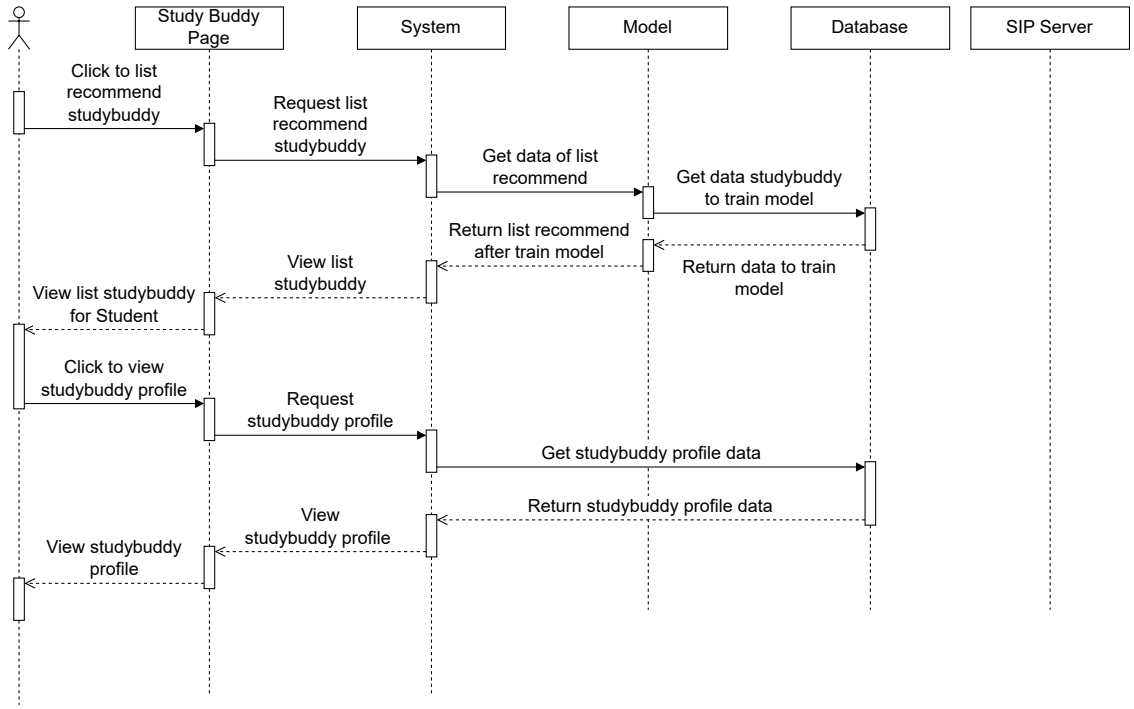


Figure 14: Study Buddy recommend list sequence diagram

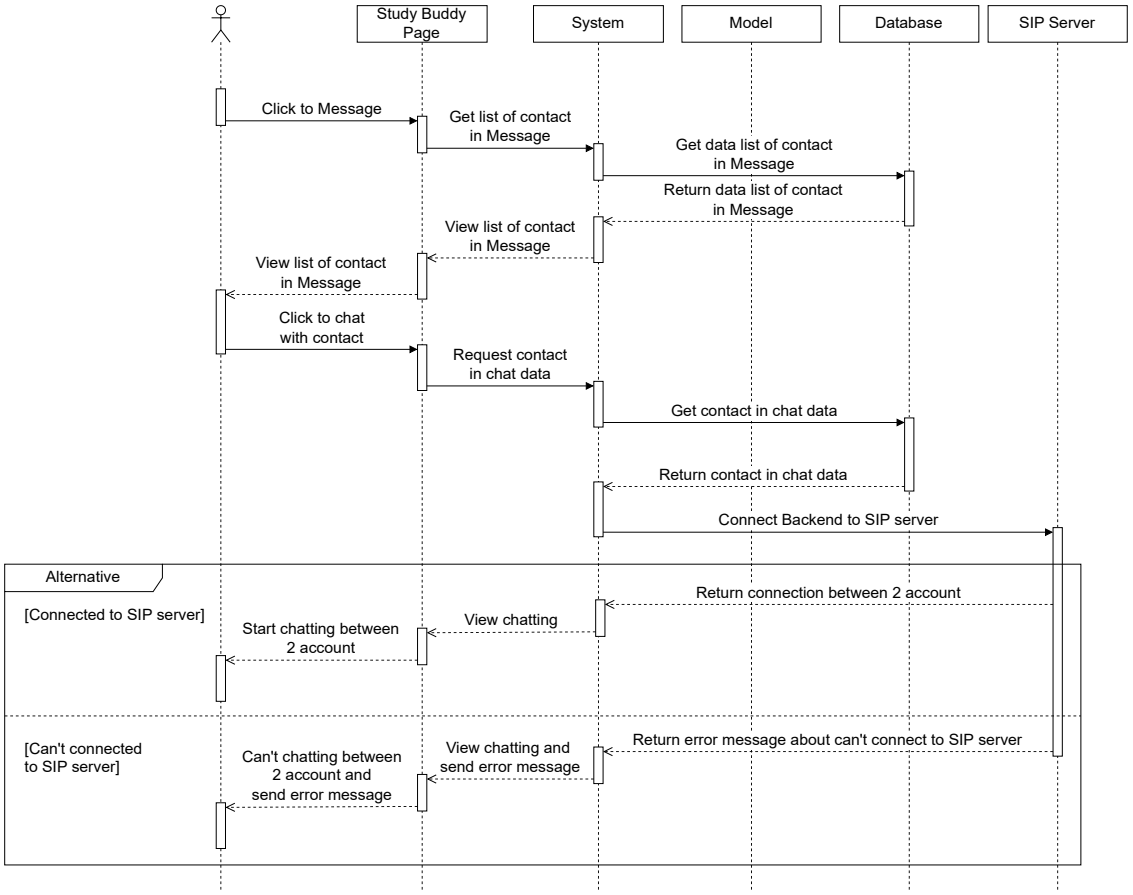


Figure 15: Study Buddy chat message sequence diagram

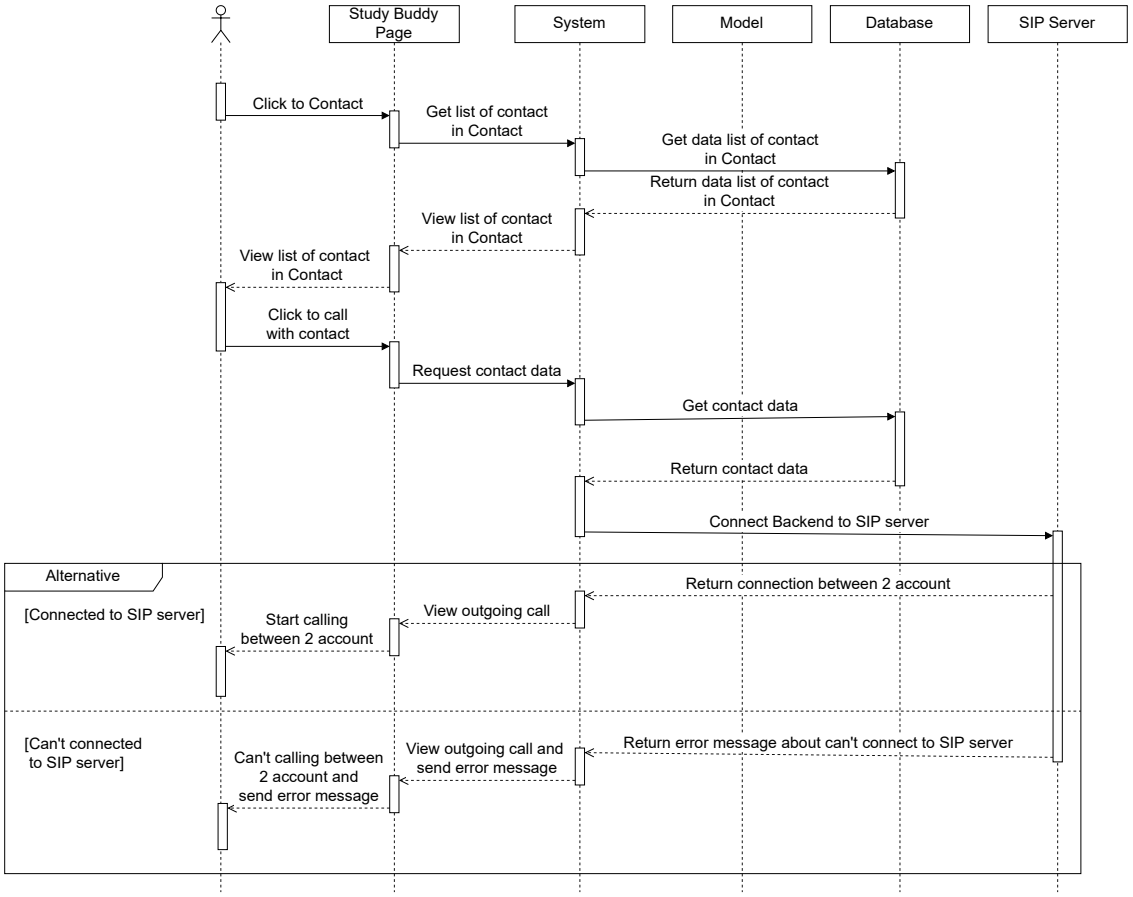


Figure 16: Study Buddy audio call sequence diagram

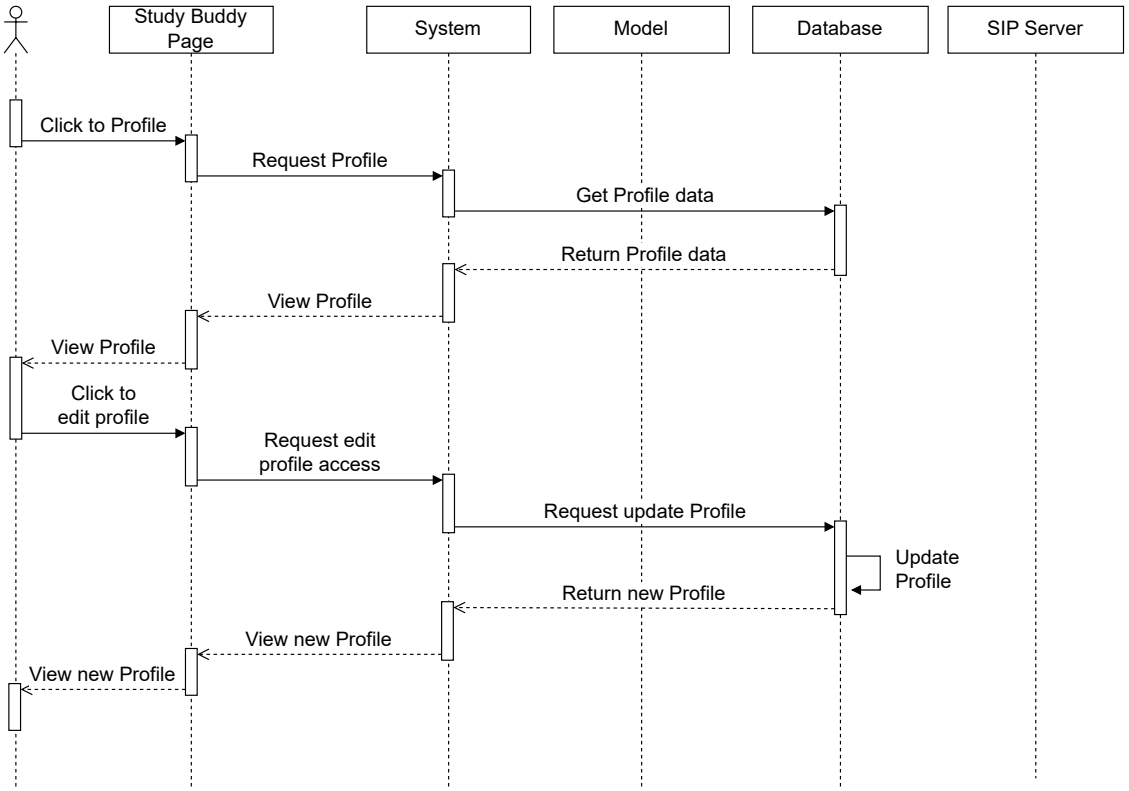


Figure 17: Study Buddy profile sequence diagram