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

Group Project Report

USTH Connect

Integrated app for university life assistant and student networking

by

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

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

Contents

List of Figures

List of Tables

1 Introduction

1.1 Context and Motivation

College life throws a lot of curveballs at students and those who run the show. At USTH, for example, juggling classes, making friends, and finding your way around campus can get pretty overwhelming. Traditional systems often feel clunky, with a lot of manual work, things scattered everywhere, and outdated information. This makes it tough for students to stay on top of things, find the resources they need, and connect with other students.

To overcome these challenges, we developed ULASNA – the University Life Assistant and Student Networking Application. ULASNA brings together some powerful tools like Spring Boot, Google Calendar, MapBox, and Moodle, along with a bit of machine learning magic. This platform makes academic life smoother by giving students easy access to their schedules, course materials, and all the important campus info. But ULASNA isn't just about academics – it helps students connect with each other too. The "StudyBuddy" feature, powered by machine learning, matches students with similar interests and goals, making it easier to find study partners and build friendships. We designed ULASNA with the user in mind, creating a user-friendly mobile app for Android devices. And we take data security seriously, using a robust PostgreSQL database and a secure VPN (Tailscale) to keep everything safe.

This report dives deep into the design and development of ULASNA, highlighting the challenges we faced along the way and how we overcame them. Ultimately, we believe ULASNA has the potential to significantly improve the college experience for both students and faculty

1.2 Process Flow

This section outlines the comprehensive process flow of University Life Assistant and Student Networking Application, detailing the progress from system construction to feature integration, including authentication with Spring Boot, Google Calendar integration, MapBox integration, Moodle resource fetching, real-time notifications, and the StudyBuddy matching system with machine learning algorithms.

1.2.1 System Construction

This section details the process of building the system. It covers the setup of hardware components, the configuration of software environment, the creation of the application's software components, and initial testing to make sure everything works smoothly.

1.2.1.1 Hardware Setup

The hardware components required for the system include:

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

1.2.1.2 Software Environment Configuration

The hardware components required for the system include:

- **Operating System:** Window was used for hosting the backend services and database, while Android was the main platform for the app.
- **Database:** PostgreSQL was installed and configured as the relational database management system to store user information, calendar events, location data, and StudyBuddy profiles.
- **Backend Framework:** Spring Boot was deployed to manage REST API endpoints and handle authentication, authorization, and data synchronization.
- **Mobile Development Tools:** Android Studio served as the primary IDE for developing the Android application, integrating Java and libraries such as Retrofit and MapBox SDK.

1.2.1.3 Application Software

The application software consists of several key modules:

- **Authentication and Authorization Module:** Implements JWT-based authentication and Role-Based Access Control (RBAC) to manage access permissions for ADMIN and USER roles.
- **Google Calendar Integration Module:** Fetches event data from Google Calendar, detects changes, and delivers notifications to the user through the mobile application.
- **MapBox Integration Module:** Stores and serves latitude and longitude coordinates of campus locations to dynamically render maps within the application.
- **Moodle Resource Module:** Interacts with Moodle APIs to retrieve course-related resources such as slides, source code, and PDFs.
- **StudyBuddy Matching Module:**
  - Collects user profile data (e.g., interests, personality).
  - Utilizes a machine learning recommendation system to suggest suitable matches based on shared interests and compatibility.

  - Incorporates a chat feature for text-based communication between matched users.
  - Enables audio calls between users through the Linphone library, which provides SIP-based VoIP functionality for real-time communication.

- **Notification System:** Facilitates real-time notifications for calendar event updates, and received call.

1.2.1.4 Initial Testing

Initial testing was performed to verify the functionality and integration of all components:

- **Hardware Testing:** Verified the correct installation and operation of servers, development machines, and networking equipment, utilizing a Tailscale VPN for secure connections.
- **Software Testing:** Ensured proper configuration and performance of the operating system, PostgreSQL database, and Spring Boot services.
- **Integration Testing:** Validated the seamless interaction between backend APIs and mobile app features, including:
  - Google Calendar synchronization.
  - MapBox map rendering.
  - Moodle resource retrieval.
  - StudyBuddy matching functionality.
  - Linphone-based audio calling capabilities.

1.2.2 Machine Learning Model Integration and Model Training

1.3 Project Objectives

The primary objective of this project is to design, develop and implement an integrated system that enhances university life and fosters student networking through advanced technological solutions. The goal is to create a platform that seamlessly intergrates personal academic data, event annoucements, and campus resources. By using advanced frameworks and APIs, together with recommendation system, we aim to simplify the administrative tasks, improve campus accessibility, and encourage more interaction among students. This platform should be user-friendly and accessible through all devices, making it easier for students to experience campus life, stay informed, and connect with their peers.

1.4 Desired Outcomes

1.5 Structure of Thesis

The thesis will be structured as follows:

- **Part I: Introduction**

  Provide a general introduction to the thesis, including an overview of the project, its objectives, and the scope of the work.

- **Part II: Requirement Analysis**

  Lists all the tools, techniques, and system requirements used in the project. It includes both functional and non-functional requirements, as well as desired functionalities.

- **Part III: Methodologies**

  System architecture, database design, and implementation details of various features, illustrated with sequence diagrams.

- **Part IV: AI Model Analysis and Training**

  Analysis and training of AI models for recommend system for study buddy matchmaking, including datasets and model development, with (Model Name) integration.

- **Part V: Results and Discussions**

  Summarizes the implementations and achievements of the system. It reflects on how the objectives were met and provides a summary of the project's outcomes.

- **Part VI: Conclusion and Future Work**

1.6 Related works

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

2 Requirement Analysis

2.1 System requirements

2.1.1 Functional Requirements

2.1.2 Non-functional Requirements

2.1.3 Desired Functionalities

2.2 Use Case

2.2.1 Use Cases Diagram

2.2.2 Use Case Characteristics

2.3 Use Case and Scenario Description

3 Methodologies

3.1 System Architecture

3.2 Database Design

3.3 Use Case Implementation

3.3.1 Analysis Call and Message Technologies

3.3.1.1 Session Initiation Protocol

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

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

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

SIP architecture consists of:

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

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

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

3.3.1.2 Linphone

Linphone is an open-source VoIP (Voice over Internet Protocol) application that utilizes SIP-based user agent. It features all basic SIP-related services, such as audio and video calls, call management, call transfer, audio conferencing, and instant messages. The free Linphone SIP service is released with an open-source license; and the SIP server software powering this service is called Flexisip. Linphone uses its open-source library as its core, called liblinphone. The library is a SIP-based SDK5 for video and audio over IP and is written in C/C++. The application is available on Linux, Windows, MacOS, iOS, and Android.

The combination of Linphone and Flexisip SIP proxy provides secure end-user registration and call setup. More precisely, Linphone client establishes and maintains a SIP TLS connection to the Flexisip server. The Linphone client verifies the SIP server's identity based on the X.509 digital certificate of the server (a list of trusted root authorities is provided at compilation time). In this way, message and entity authentication, as well as confidentiality, of the information exchanged between the Linphone client and the Flexisip server is ensured.

3.3.1.3 Transport Layer Security Protocol (TLS)

Transport Layer Security, or TLS, is a widely adopted security protocol designed to facilitate privacy and data security for communications over the Internet. TLS was derived from a security protocol called Secure Socket Layer (SSL). A primary use case of TLS is encrypting the communication between applications and servers. TLS is based on symmetric encryption and is a client - server model, where the client is able to authenticate the server and, optionally, the server is also able to authenticate the client.

When using SIP over TLS, the whole SIP signalling is encrypted. However it holds only on the segments of the communication which actually use TLS. Therefore, TLS will be automatically set to each end of the user by default in this project.

3.3.1.4 Communication Handling

3.3.1.4.1 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.

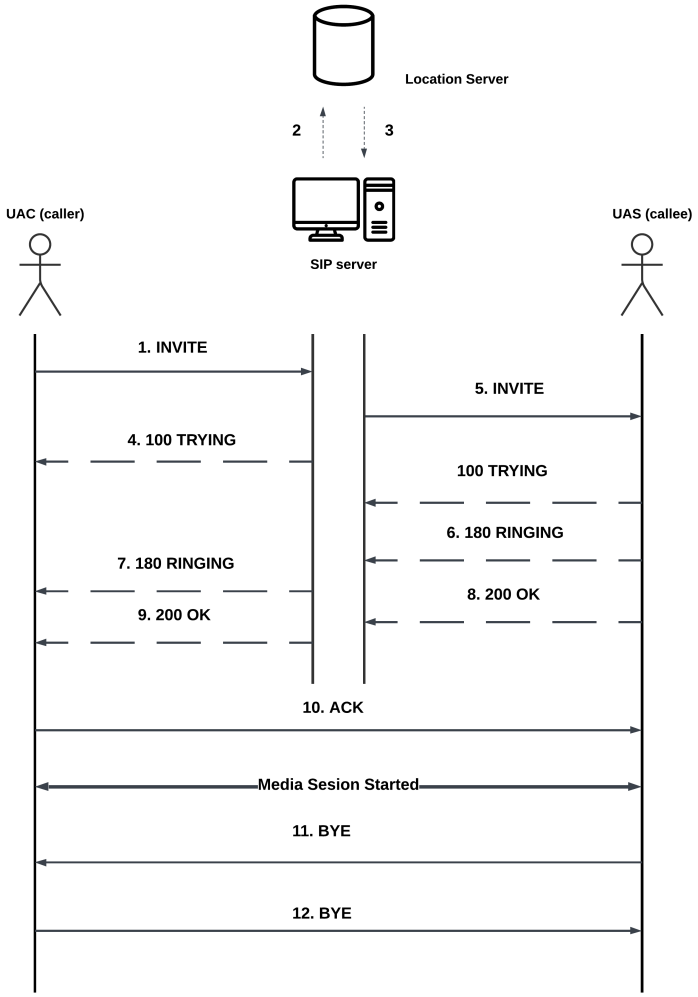

Figure 1: SIP signaling and key components

3.3.1.4.2 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.

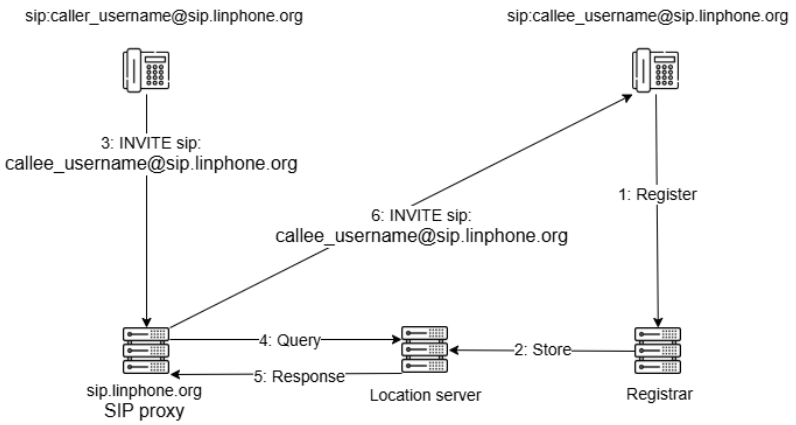

Figure 2: SIP registration and locating callee

4 AI Model Analysis and Training

5 Results and Discussion

5.1 Results

5.1.1 Mobile App Results

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

5.1.2 Machine Learning Results

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

5.2 Discussion

6 Conclusion & Future Work

6.1 Conclusion

6.2 Future Work