ErasmusGo

A project on behalf of IPCA for the course Applied Project

Bastien De Meulenaere (50997)

Lucas Viaene (51003)

Paul Blankenhorn (51005)

0. Table of Contents

1. Introduction

1.1. Contextualization, Objectives, and Motivation

ErasmusGo is designed to support incoming Erasmus students in navigating the initial challenges of settling into a new academic and cultural environment.
The application aims to simplify the onboarding process for international students by centralizing essential resources such as account setup, campus navigation, event management, and communication tools.

Objectives:

- Provide a streamlined registration process for new Erasmus students.
- Simplify first-time setup for campus life, including activating student accounts.
- Facilitate communication with professors and peers.
- Offer tools like campus maps, academic calendars, and transportation information to ease student transitions.
- Provide language support and cultural integration resources.

Motivation:
The Erasmus exchange experience can be overwhelming for international students unfamiliar with their new surroundings. ErasmusGo seeks to empower students with confidence and independence as they adapt to their academic and cultural settings.

2. Team Organization

To ensure clarification on roles and responsibilities, we divided our team into three roles:

- **Lucas Viaene: Project Manager/Coordinator:**
  - Manages the timeline, organizes sprints, and ensures everyone is on track.
  - Oversees documentation and merges everyone's work into a single report.

- **Paul Blankenhorn: Developer/Tech Lead:**
  - Focuses on the mobile app's coding and technical aspects.
  - Prepares diagrams like sequence diagrams and models closely tied to development.

- **Bastien De Meulenaere: UX/UI and Documentation Specialist:**
  - Creates wireframes/mockups, diagrams, and user stories.
  - Supports documentation, testing, and any needed refinements.

3. Development of the System Proposal

3.1. Stakeholders

The key stakeholders for ErasmusGo are:

- **Erasmus Students:** The users, benefiting from the app's features to facilitate their stay.
- **University Staff and Professors:** For providing data and managing student-related interactions.

3.2. Functional Requirements

The app must meet the following functional requirements:

- **Registration and Login:**
  - Students can register accounts using email.
  - Password recovery functionality for forgotten credentials.
- **Profile Management:**
  - Students can update their profiles with personal details such as their profile picture.
- **Campus Navigation:**
  - Provide maps for building and classroom locations.
  - Steps for activating university accounts.
- **Communication Features:**
  - Contact details for professors.
  - Peer-finding tools based on nationality.
- **Event Management:**
  - View upcoming events and activities.
- **Academic Calendar and Transportation Information:**
  - Provide key dates and holidays.
  - Display public transport routes and schedules.
- **Language and Cultural Support:**
  - Basic Portuguese learning resources.
  - Suggestions for external Portuguese language courses.

3.3. Non-Functional Requirements

The app must also satisfy the following non-functional criteria:

- **Usability:**
  - Interface must be intuitive and simple to navigate.
- **Performance:**
  - Handle concurrent logins efficiently.
  - Respond to user queries within 2 seconds.
- **Security:**
  - Require strong password protocols during registration.
- **Scalability:**
  - Ensure the app accommodates an increasing number of users and events.
- **Availability:**
  - Maintain uptime of 99.9% during academic sessions.

3.4 Hardware and Software Requirements

3.4.1. Hardware Requirements

ErasmusGo is a lightweight application designed to run efficiently on standard devices:

- **For Students (Mobile App Users):**
  - **Device Type:** Smartphone or Tablet
  - **Operating System:** Android 6.0+ or iOS 12.0+
  - **RAM:** 1 GB or more
  - **Processor:** Dual-core (1.0 GHz or higher)
  - **Storage Space:** Minimum 20 MB for app installation
  - **Internet Connection:** Wi-Fi or mobile data (2G or higher)
- **For Backend Hosting:**
  - **Server Specifications (Cloud or Local):**
    - **Processor:** Dual-core CPU (2.0 GHz or higher)
    - **RAM:** 4 GB
    - **Storage:** 10 GB available for database and app resources
    - **Network Bandwidth:** 10 Mbps or higher

3.4.2. Software Requirements

- **For App Development:**
  - **Frontend:** Kotlin (Jetpack Compose)
  - **Backend:** Firebase for authentication and cloud storage
  - **Database:** SQLite (local storage)
  - **Development Tools:** Android Studio (latest version)
- **For App Users:**
  - **App Compatibility:**
    - Android 6.0+ or iOS 12.0+
    - Firebase Authentication for secure login
- **For Server Hosting:**
  - **Cloud Service Provider:** Firebase (no dedicated server required for small-scale operations)

This setup ensures that ErasmusGo operates smoothly on common school devices without requiring advanced hardware or software infrastructure.

4. System Architecture

The system architecture is designed to accommodate ease of use, scalability, and modularity. Below is an overview of the app's structure and functionality:

4.1. Landing Page

- **Login Button:** Redirects users to their accounts.
- **Register Button:** Leads to the registration screen.
- **Forgot Password Button:** Initiates the password recovery process.

4.2. Main Menu

- **Profile Section:**
  - View and edit personal information.
- **Communication:**
  - **Contact Teachers:** List and search professor contact details.
  - **Find Peers:** Identify other students from the same nationality.
- **Campus Information:**
  - **Account Activation:** Steps for activating a school account.
  - **Academic Calendar:** Key dates and holidays.
  - **Campus Map:** Interactive maps of classrooms and buildings.
- **Transportation & Discounts:**
  - **Public Transport Information:** Routes and schedules.
  - **Student Discounts:** Categorized discount opportunities.
- **Learning & Events:**
  - **Learn Portuguese:** Basic phrases and language resources.
  - **Upcoming Events:** List and details of planned activities.

4.3. Technical Infrastructure

- **Frontend:**
  - Developed using Kotlin Jetpack-Compose for a cross-platform mobile experience.
  - Intuitive UI/UX to ensure a seamless experience.
- **Backend:**
  - Local SQLite for communication between the app and database and Firebase for external API call.
- **Database:**
  - SQLite Database to securely store user data and application resources.

5. User Stories

To correctly define our work, we created user stories for the different stakeholders that could possibly use our application. Using the MoSCoW approach, we prioritize these user stories within our project scope:

5.1. Must have: Essential to launch the app.

- As a new Erasmus student, I want to register an account using my email so that I can access the app's features.
- As a new Erasmus student, I want to be able to login to my personal account.
- As a new Erasmus student, I want to recover my password if I forget it so that I can regain access.
- As a new Erasmus student, I want to complete my profile with my name, photo, email, and nationality so that I can identify myself within the application.

5.2. Should have: Useful features.

- As a new Erasmus student, I want to be guided through the first steps I need to complete on campus, such as activating my personal school account, so that I can settle in quickly.
- As a new Erasmus student, I want to get in contact with my professors before attending the first lecture, so that I can clarify any doubts about the course.
- As a student, I want to access a calendar with holidays and important dates, so that I stay organized.
- As a non-native Portuguese-speaking student, I want to learn some basic Portuguese to survive and to receive information on how to learn more Portuguese.
- As a student, I want to find out how I can get to school using public transport, so I can plan my commute.

5.3. Could have: Nice-to-have features.

- As a new Erasmus student, I want to locate classrooms on a map, so that I can find my classes easily.
- As a student, I want to view a list of upcoming events, so I can participate and meet others.
- As a student from another country, I want to find others from my country, so that I can connect with them.
- As a student, I want to know what discounts I qualify for by being a student, so that I can save money.

5.4. Won't have (for now): Features you defer to later iterations.

- /

6. Use Cases

Here's a list of use cases, defined in detail and prioritized by the MoSCoW structure.

6.1. Must Have (Essential to launch the app):

6.1.1. Use Case: Registration and Authentication

- **Priority:** Must
- **Actor:** New Erasmus Student
- **Goal:** Enable users to register an account and log in securely.
- **Preconditions:**
  - User has installed the app.
  - User has a valid email address.
- **Main Flow:**
  1. User selects the "Register" option on the landing page.
  2. User enters their email, creates a password, and fills out initial profile details (name, photo, nationality).
  3. App validates the input and creates the account.
  4. User logs in using their credentials.
- **Postconditions:**
  - User can access their personal account.
- **Alternative Path:**
  - If the email is invalid or already registered, display an error message.

6.1.2. Use Case: Password Recovery

- **Priority:** Must
- **Actor:** Student
- **Goal:** Allow users to recover their password.
- **Preconditions:** User has an existing account.
- **Main Flow:**
  1. User selects "Forgot Password" on the login screen.
  2. App sends a password recovery email.
  3. User follows the link to reset their password.
- **Postconditions:**
  - User successfully resets their password and regains access to their account.

6.1.3. Use Case: Profile Completion

- **Priority:** Must
- **Actor:** New Erasmus Student
- **Goal:** Allow users to complete their profile with relevant details.
- **Preconditions:** User has successfully registered an account.
- **Main Flow:**
  1. User navigates to the "Profile" section after logging in.
  2. User uploads a photo and fills in personal details (name, email, nationality).
  3. User saves the changes.
- **Postconditions:**
  - User's profile is updated and visible in their account.
- **Alternative Path:**
  - If required fields are incomplete, app prompts user to provide missing information.

6.2. Should Have (Useful features):

6.2.1. Use Case: Activate Student Account

- **Priority:** Should
- **Actor:** New Erasmus Student
- **Goal:** Guide users through activating their school account.
- **Preconditions**: Student has installed the app.
- **Main Flow**:
  1. Student selects "How do I activate my student account?" under "Campus Information".
  2. App guides the student through creating or activating their account.
- **Postconditions:**
  - User completes their onboarding process.

6.2.2. Use Case: Teacher Communication

- **Priority:** Should
- **Actor:** Student
- **Goal:** Allow students to contact their professors.
- **Preconditions:**
  - App has a list of professors and their contact details.
- **Main Flow:**
  1. Student navigates to the "Contact Teachers" section.
  2. App displays a searchable list of professors with their contact details.
  3. Student clicks on a professor's name to view their full contact information.
- **Postconditions:**
  - Student retrieves the professor's contact details.

6.2.3. Use Case: Calendar Management

- **Priority:** Should
- **Actor:** Student
- **Goal:** Provide an academic calendar with important dates.
- **Main Flow:**
  1. Student opens the "Academic Calendar" section.
  2. App displays a calendar view with marked holidays and events.
  3. Student taps a date to view detailed information.
- **Postconditions:**
  - Student accesses calendar details.

6.2.4. Use Case: Learning Portuguese

- **Priority:** Should
- **Actor:** Non-native Portuguese-speaking Student
- **Goal:** Provide basic Portuguese phrases and external learning resources.
- **Preconditions:**
  - App is connected to external language learning platforms.
- **Main Flow:**
  1. User accesses the "Learn Portuguese" section.
  2. App displays basic phrases and links to additional resources.
- **Postconditions:**
  - User learns basic Portuguese and accesses external resources.

6.2.5. Use Case: Public Transportation

- **Priority:** Should
- **Actor:** Student
- **Goal:** Help students find public transport options to campus.
- **Preconditions:** App contains local transport data.
- **Main Flow:**
  1. Student navigates to the "Public Transport" section.
  2. App displays routes, schedules, and commute tips.
- **Postconditions:**
  - Student plans their commute to campus.

6.3. Could Have (Nice-to-have features):

6.3.1. Use Case: Campus Map

- **Priority:** Could
- **Actor:** Student
- **Goal:** Enable students to find classrooms and buildings on campus.
- **Preconditions:** App has campus map data.
- **Main Flow:**
    1. Student opens the "Campus Map" section.
    2. User searches for a specific room or building.
    3. App displays the location on a map with directions.
- **Postconditions:**
    - Student identifies the desired location.

6.3.2. Use Case: Upcoming Events

- **Priority:** Could
- **Actor:** Student
- **Goal:** Provide a list of upcoming events and allow registration.
- **Main Flow:**
    1. Student opens the "Upcoming Events" section.
    2. App displays event details (name, date, description).
    3. Student registers for an event.
- **Postconditions:**
    - Student successfully registers for the event.

6.3.3. Use Case: Peer Connections

- **Priority:** Could
- **Actor:** Student
- **Goal:** Connect students from the same country.
- **Preconditions:** Student profile includes nationality.
- **Main Flow:**
    1. Student navigates to the "Find Peers" section.
    2. App displays a list of students with the same nationality.
    3. Student selects a peer to view their profile.
- **Postconditions:**
    - Student connects with another peer.

6.3.4. Use Case: Student Discounts

- **Priority:** Could
- **Actor:** Student
- **Goal:** Provide a categorized list of student discounts.
- **Preconditions:** App contains discount data.
- **Main Flow:**
    1. Student accesses the “Student Discounts” section.
    2. App displays a categorized list (e.g., food, transport).
- **Postconditions:**
    - Student views and utilizes the discounts.

7. Functional Wireframes and Workflow

We created functional wireframes using Figma to visualize and represent how the ErasmusGo app operates. The wireframes include one main workflow:

7.1. Erasmus Student Workflow:

- This set of wireframes outlines the app's functionality from the perspective of a general Erasmus student.
- It demonstrates key features such as registration, login, profile management, campus navigation, event access, and communication tools.

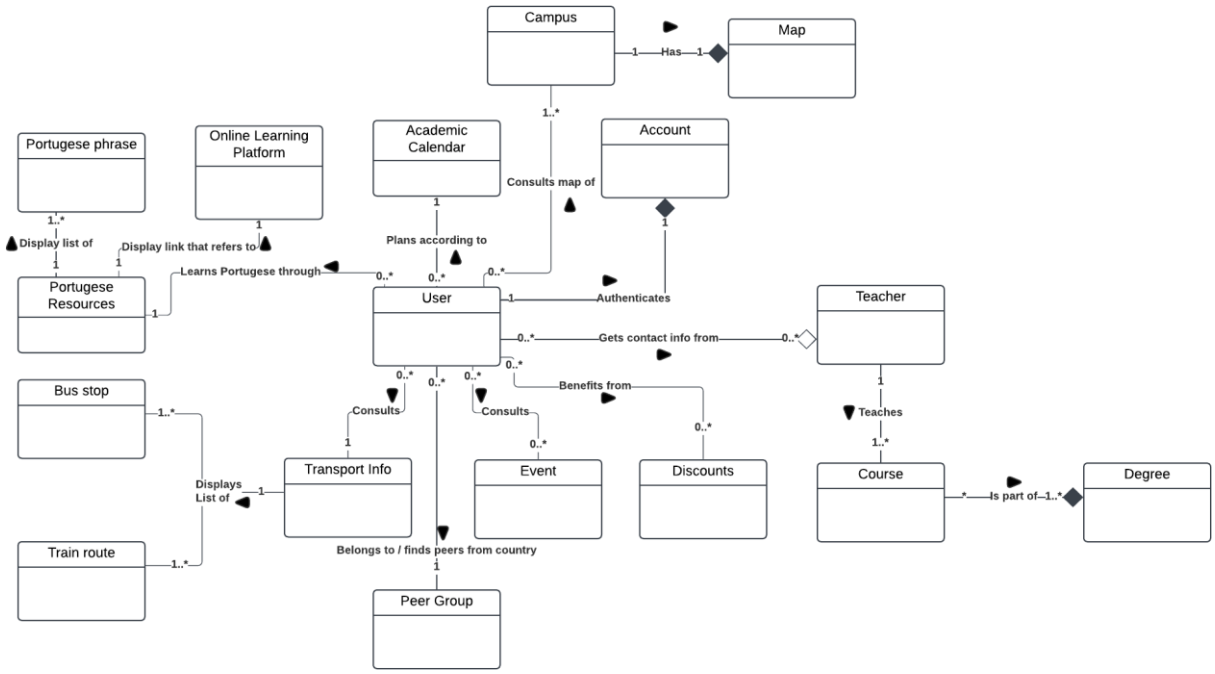
These wireframes provide a clear, step-by-step depiction of the app's interface and interactions, ensuring the workflows align with user needs and functional requirements.

8. Diagrams and Models

To ensure a clear and structured representation of the system, we created the following diagrams:

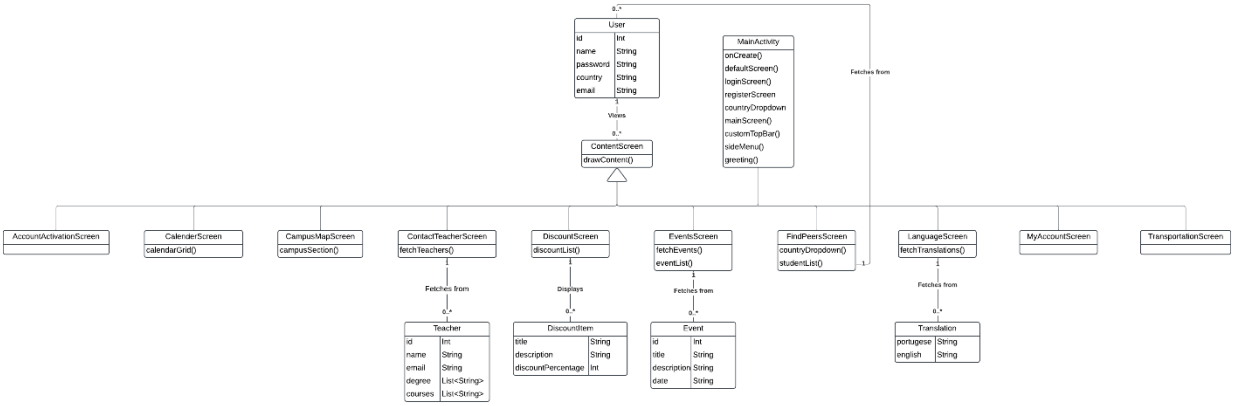
8.1. Domain Model

The domain model provides an abstract view of the main concepts and entities within the ErasmusGo application and their relationships. It highlights how the core elements interact within the system's domain.



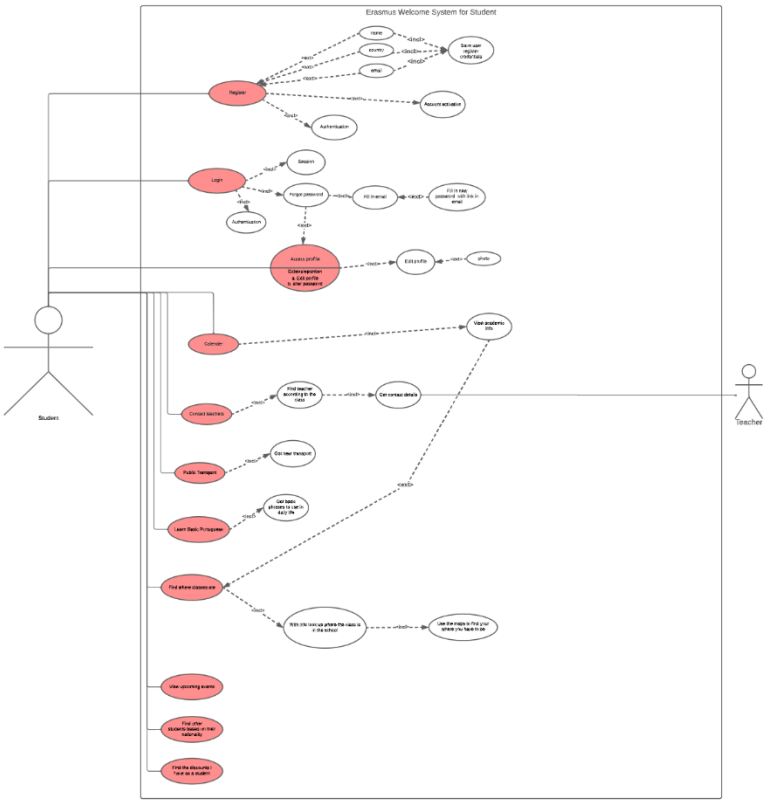
8.2. Class Diagram

The class diagram provides a detailed representation of the application's structure, illustrating the classes, their attributes, methods, and the relationships between them.



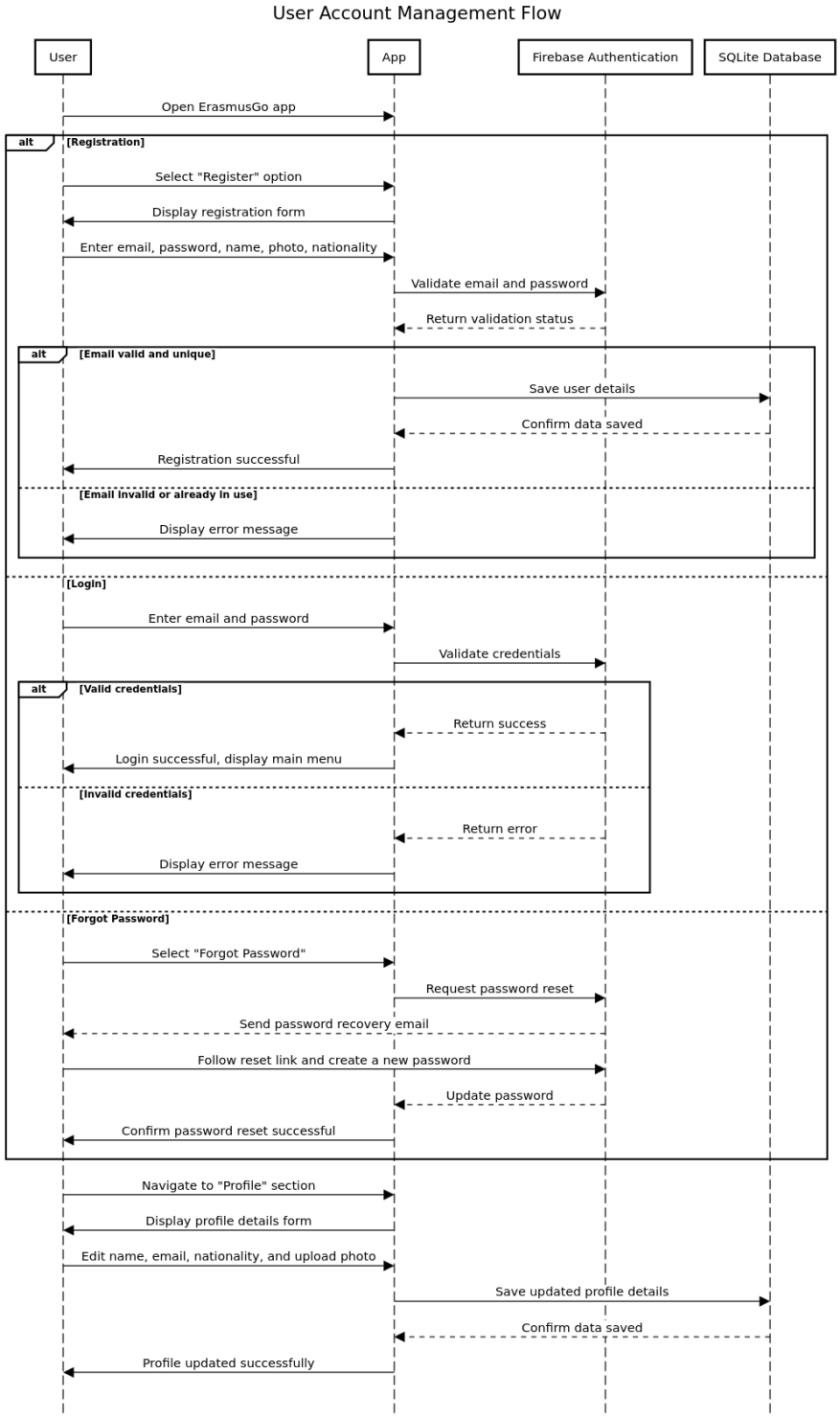
8.3. Use Case Diagram

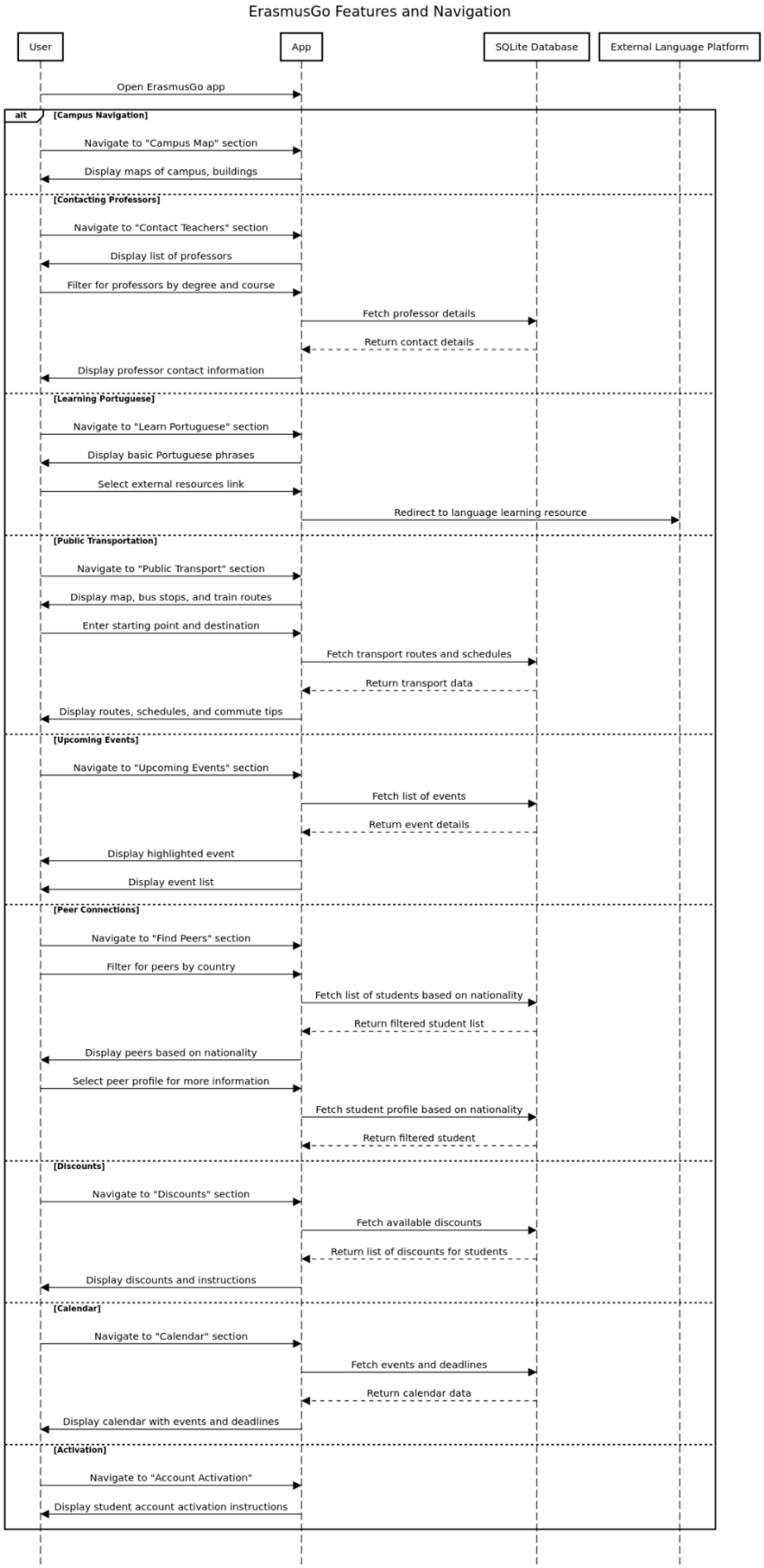
The use case diagram visually represents the system's functionality by showing the actors and their interactions with the system. It captures the core functionality required to meet user goals. You can find the Legend **here**.



8.4. Sequence Diagram

The sequence diagram illustrates the dynamic interaction between system components during specific processes. It focuses on the order of messages exchanged to complete a use case. You can also find the **Account Management Flow** and the **Features and Navigation** diagrams in the "/doc" folder in our solution on GitHub.




ErasmusGo Features and Navigation
User
App
SQLite Database
External Language Platform
Open ErasmusGo app
alt
[Campus Navigation]
Navigate to "Campus Map" section
Display maps of campus, buildings
[Contacting Professors]
Navigate to "Contact Teachers" section
Display list of professors
Filter for professors by degree and course
Fetch professor details
Return contact details
Display professor contact information
[Learning Portuguese]
Navigate to "Learn Portuguese" section
Display basic Portuguese phrases
Select external resources link
Redirect to language learning resource
[Public Transportation]
Navigate to "Public Transport" section
Display map, bus stops, and train routes
Enter starting point and destination
Fetch transport routes and schedules
Return transport data
Display routes, schedules, and commute tips
[Upcoming Events]
Navigate to "Upcoming Events" section
Fetch list of events
Return event details
Display highlighted event
Display event list
[Peer Connections]
Navigate to "Find Peers" section
Filter for peers by country
Fetch list of students based on nationality
Return filtered student list
Display peers based on nationality
Select peer profile for more information
Fetch student profile based on nationality
Return filtered student
[Discounts]
Navigate to "Discounts" section
Fetch available discounts
Return list of discounts for students
Display discounts and instructions
[Calendar]
Navigate to "Calendar" section
Fetch events and deadlines
Return calendar data
Display calendar with events and deadlines
[Activation]
Navigate to "Account Activation"
Display student account activation instructions

9. Tests Carried Out

During the development process, we did not perform formal unit tests. Instead, we focused on rigorous debugging and functionality testing using multiple Android devices and simulations. This ensured that the application operates smoothly across various scenarios and device configurations. Our testing approach allowed us to identify and resolve issues effectively, contributing to the overall stability and reliability of the application.

10.Mobile Application User Manual

To make the application usage straightforward and accessible to all users, we created a comprehensive **Mobile Application User Manual**. This document provides clear step-by-step guidance on all functionalities, ensuring that users can easily navigate and utilize the application to its fullest potential. You can access the full user manual at the following link: ErasmusGo User Manual.

11. Aims and Objectives (Fulfilled)

The primary aim of ErasmusGo is to simplify the onboarding and campus integration process for Erasmus students. Below are the objectives and their fulfillment status:

- **Provide a streamlined registration process for new Erasmus students.**
  ✅ **Fulfilled**: The app allows easy account creation with a secure registration and login system.
- **Simplify first-time setup for campus life, including activating student accounts.**
  ✅ **Fulfilled**: A detailed guide to activating university accounts is integrated within the app.
- **Facilitate communication with professors and peers.**
  ✅ **Fulfilled**: Features include professor contact details and a peer-finding tool based on nationality.
- **Offer tools like campus maps, academic calendars, and transportation information to ease student transitions.**
  ✅ **Fulfilled**: Interactive campus maps, academic calendars, and public transport details are available in the app.
- **Provide language support and cultural integration resources.**
  ✅ **Partially Fulfilled**: Basic Portuguese learning resources are implemented; cultural integration tools are deferred for future updates.

12. Warnings

While ErasmusGo aims to be a comprehensive onboarding tool, users should consider the following limitations:

- **Data Accuracy:**
  - Information such as academic calendars, professor contact details, and transportation schedules are subject to change. Ensure you verify critical details with official university resources.
- **Dependency on Internet Connection:**
  - Many app features, such as contact search and public transport information, require an active internet connection.
- **Limited Language Support:**
  - The app currently focuses on Portuguese basics. Students requiring advanced support are encouraged to explore external courses.
- **Personal Data Security:**
  - Users must ensure the privacy of their login credentials. The app employs secure authentication methods, but users are responsible for their account safety.

13.Glossary

To ensure clarity, here are definitions for key terms used in this document:

- **Erasmus:** A European Union program that enables students to study or work abroad in participating countries.
- **Onboarding:** The process of acclimating new users (students) to their academic and cultural environment.
- **Firebase:** A platform developed by Google for building mobile and web applications, used for authentication and cloud storage in ErasmusGo.
- **Jetpack Compose:** A modern UI toolkit for building native Android applications.
- **SQLite:** A lightweight database used for storing app-related data locally.
- **MoSCoW:** A prioritization technique to classify project requirements into Must-have, Should-have, Could-have, and Won't-have categories.

14.Conclusion

ErasmusGo addresses the critical challenges faced by incoming Erasmus students by offering a centralized, intuitive platform for essential resources. Its features streamline the registration process, facilitate campus integration, and support effective communication with professors and peers. Although the app has fulfilled most of its primary objectives, areas like advanced language resources and cultural integration remain opportunities for future enhancement.

Through its seamless design, ErasmusGo empowers students to confidently navigate their academic and cultural transitions, setting a strong foundation for a successful Erasmus experience.

15. Bibliography

- Google Firebase Documentation: https://firebase.google.com/docs
- Jetpack Compose Overview: https://developer.android.com/jetpack/compose
- SQLite Official Website: https://www.sqlite.org/
- Erasmus+ Program Guide: https://erasmus-plus.ec.europa.eu/
- Android Studio User Guide: https://developer.android.com/studio