

**Malaysian Institute of Information Technology**

**Software Requirements Specification**

**UniKL TutorFind**

**University Tutor Finder Application**

**Version 2.0**

**Prepared by**

**Alyssa Husna Binti Jamarizan (52213223155)**

**Alya Azwin Binti Zamri (52213223009)**

**6th September 2025**

\*Confidentiality Notice: This document contains confidential information intended for the project team and stakeholders mentioned herein. Unauthorized distribution is prohibited.

Table of Contents

[1.0 Introduction 1](#_Toc201332015)

[1.1 Purpose 1](#_Toc201332016)

[1.2 Document Conventions 1](#_Toc201332017)

[1.3 Intended Audience and Reading Suggestions 2](#_Toc201332018)

[1.4 Project Scope 2](#_Toc201332019)

[1.5 References 3](#_Toc201332020)

[2.0 Overall Description 4](#_Toc201332021)

[2.1 Product Perspective 4](#_Toc201332022)

[2.2 Product Features 4](#_Toc201332023)

[2.3 User Classes and Characteristics 4](#_Toc201332024)

[2.4 Operating Environment 5](#_Toc201332025)

[2.5 Design and Implementation Constraints 5](#_Toc201332026)

[2.6 Assumptions and Dependencies 5](#_Toc201332027)

[3.0 System Features 7](#_Toc201332028)

[3.1 System Feature: Login and Registration (LR) 7](#_Toc201332029)

[3.1.1 Description and Priority 7](#_Toc201332030)

[3.1.2 Stimulus/Response Sequences 7](#_Toc201332031)

[3.1.3 Functional Requirements 8](#_Toc201332032)

[3.2 System Feature: Profile Creation (PC) 9](#_Toc201332033)

[3.2.1 Description and Priority 9](#_Toc201332034)

[3.2.2 Stimulus/Response Sequences 9](#_Toc201332035)

[3.2.3 Functional Requirements 9](#_Toc201332036)

[3.3 System Feature: Profile Matching (PM) 10](#_Toc201332037)

[3.3.1 Description and Priority 10](#_Toc201332038)

[3.3.2 Stimulus/Response Sequences 10](#_Toc201332039)

[3.3.3 Functional Requirements 11](#_Toc201332040)

[3.4 System Feature: Search and Filters (SF) 11](#_Toc201332041)

[3.4.1 Description and Priority 11](#_Toc201332042)

[3.4.2 Stimulus/Response Sequences 11](#_Toc201332043)

[3.4.3 Functional Requirements 12](#_Toc201332044)

[3.5 System Feature: In-App Chatting (IC) 12](#_Toc201332045)

[3.5.1 Description and Priority 12](#_Toc201332046)

[3.5.2 Stimulus/Response Sequences 13](#_Toc201332047)

[3.5.3 Functional Requirements 13](#_Toc201332048)

[3.6 System Feature: Rate and Review (RR) 14](#_Toc201332049)

[3.6.1 Description and Priority 14](#_Toc201332050)

[3.6.2 Stimulus/Response Sequences 14](#_Toc201332051)

[3.6.3 Functional Requirements 14](#_Toc201332052)

[4.0 External Interface Requirements 16](#_Toc201332053)

[4.1 User Interfaces (UI Design for Web and Mobile) 16](#_Toc201332054)

[4.2 Hardware Interfaces 16](#_Toc201332055)

[4.3 Software Interfaces 17](#_Toc201332056)

[4.4 Communication Interfaces 18](#_Toc201332057)

[5.0 Use Case Models 19](#_Toc201332058)

[5.1 Use Case: Login and Registration (LR) 19](#_Toc201332059)

[5.2 Use Case: Profile Creation and Management (PC) 20](#_Toc201332060)

[5.3 Use Case: Search for Peers (SF) 20](#_Toc201332061)

[5.4 Use Case: Match with Peers (PM) 21](#_Toc201332062)

[5.5 Use Case: In-App Chatting (IC) 22](#_Toc201332063)

[5.6 Use Case: Rate and Review (RR) 22](#_Toc201332064)

[6.0 Non-Functional Requirements 24](#_Toc201332065)

[6.1 Performance 24](#_Toc201332066)

[6.2 Safety 24](#_Toc201332067)

[6.3 Security 25](#_Toc201332068)

[6.4 Software Quality Attributes (for both Web and Mobile platforms) 25](#_Toc201332069)

[7.0 Other Requirements 27](#_Toc201332070)

[7.1 Database Design (for both platforms) 27](#_Toc201332071)

[7.2 Testing Requirements 28](#_Toc201332072)

[7.3 Deployment Considerations 30](#_Toc201332073)

[Appendix A: Glossary 31](#_Toc201332074)

[Appendix B: Analysis Models (e.g., Entity-Relationship Diagrams) 33](#_Toc201332075)

[Appendix C: Issues List 34](#_Toc201332076)

# Introduction

## 1.1 Purpose

This Software Requirements Specification (**SRS**) document defines the software requirements for the **UniKL TutorFind** platform**, a peer-to-peer skill matching platform** designed for UniKL students. The platform enables students to connect with each other, facilitating the sharing and acquisition of skills in a collaborative environment.

The platform will be developed using **Flutter** for **cross-platform mobile and web application development**, and **Supabase** for **backend services, including user authentication, real-time database, and cloud storage**. It will be accessible on both Mobile and Web, ensuring broad accessibility among students.

The platform's development will proceed in three iterations, following the **Rapid Application Development (RAD)** methodology:

1. Iteration 1 (FYP1): Focuses on user login and registration and profile creation, ensuring basic authentication and profile functionalities.
2. Iteration 2 (FYP2): Implements the matching system and search and filters for peer-to-peer connections.
3. Iteration 3 (FYP2): Integrates the in-app chat system and reviews and ratings for communication and feedback.

Since this platform is a final year project (FYP), it is designed for **academic purposes only** and will not be deployed or maintained long-term after the project’s completion.

## 1.2 Document Conventions

This document follows the standard conventions for an SRS as outlined in the **SRS Template** given through UniKL VLE. Key conventions in this document include:

* **Bold**: Used for important terms, section headings, and key concepts.
* *Italics*: Used for terms defined in the glossary or for emphasis in examples.
* **Monospace font**: Used for code snippets, configuration files, or UI elements.
* Prioritization of Requirements:
  + **High**: Essential for the system’s core functionality.
  + **Medium**: Important but can be deferred if necessary for future iterations.
  + **Low**: Optional features for future development or enhancements.

This document adheres to the **RAD methodology**, emphasizing rapid prototyping, quick development cycles, and continuous user feedback.

## 1.3 Intended Audience and Reading Suggestions

This document is intended for the following stakeholders:

* **Project Developers**: To understand system functionalities, features, and architecture, and ensure they align with project objectives.
* **Project Managers**: For tracking the project's progress and ensuring that each iteration meets the outlined specifications.
* **Testers**: To define and execute test cases based on functional and non-functional requirements.
* **End-users (UniKL Students)**: To understand how they will interact with the platform and provide feedback.
* **Documentation Writers**: To assist in creating user manuals, help guides, and support documentation.

Reading Sequence:

* **Section 1**: Introduction to the project and document's purpose.
* **Section 2**: High-level overview of system features and scope.
* **Section 3**: Detailed functional and non-functional requirements.
* **Section 4**: Describes interactions with external systems.
* **Section 5**: Use cases for web and mobile applications.
* **Section 6**: Non-functional requirements (performance, security, etc.).
* **Section 7**: Additional requirements (database design, testing, deployment considerations).
* **Appendices**: Glossary, analysis models, and issues list.

## 1.4 Project Scope

The **UniKL TutorFind** platform facilitates peer-to-peer learning among **UniKL students**, enabling them to share skills and connect based on learning goals and complementary expertise. The platform will be developed using **Flutter** and **Supabase**, providing a seamless user experience across **Mobile** and **Web**.

This SRS outlines the platform's development, which will proceed in three iterations:

* **Iteration 1**: Focuses on **user login and registration** and **profile creation**, including Supabase Authentication and profile management.
* **Iteration 2**: Implements the **matching system** and **search and filters** for student connections.
* **Iteration 3**: Integrates the **in-app chat** system and **reviews and ratings** for communication and feedback.

Since this platform is a **final year project (FYP)**, it will be deployed only for the duration of the project and will not be maintained long-term after its completion. The primary goal is to deliver functional software that meets academic objectives.

Key features include:

* **User Login and Registration:** Verification through Supabase Authentication using @unikl.edu.my email.
* **User Profiles**: Students can create, update, and manage profiles showcasing skills they can teach and want to learn.
* **Matching System**: Matches students based on complementary skills, availability, and preferences.
* **Search and Filters**: A powerful search system that allows students to find peers based on skills, availability, location, and learning preferences.
* **Messaging System**: Real-time chat through **Supabase Realtime Database** for communication between matched peers.
* **Reviews and Ratings**: A feedback mechanism that allows students to rate their sessions and provide constructive feedback.

## 1.5 References

The following documents and resources were used in the preparation of this SRS:

* **Flutter Documentation**: Official documentation for mobile and web development. [Flutter Docs](https://flutter.dev/docs)
* **Supabase Documentation**: Official documentation for Supabase database, authentication, and real-time features. [Supabase Docs](https://firebase.google.com/docs)
* **SRS Template**: Standard template used for structuring this document.
* **UniKL TutorFind Project Proposal**: The proposal outlining the project’s objectives, features, and scope. (April 2025)

# 2.0 Overall Description

## 2.1 Product Perspective

The **UniKL TutorFind** platform is a **final year project** (FYP) designed to facilitate **peer-to-peer skill matching** among **UniKL students**. It is a self-contained product developed to meet the academic objectives of the FYP and will not be deployed for long-term use. The platform helps students connect based on **complementary skills**, using **Flutter** for cross-platform development and **Supabase** for backend services like authentication and data storage.

The platform will be accessible on both **Mobile** and **Web**, ensuring a broad user base. It will serve as a **proof of concept** for a functional **peer-to-peer skill matching platform**.

## 2.2 Product Features

The **UniKL TutorFind** platform includes the following key features:

* **User Login and Registration**: Students can register using their **@unikl.edu.my** email as authentication and access the platform.
* **User Profiles**: Students can create and manage profiles, showcasing skills they can teach and want to learn.
* **Matching System**: The system matches students based on complementary skills, availability, and preferences.
* **Search and Filters**: Students can search, and filter peers based on skills, availability, and location.
* **Messaging System**: Real-time chat functionality for communication between matched peers.
* **Reviews and Ratings**: Students can provide feedback and ratings after each learning session.

Since this is an academic project, the platform will not be maintained beyond the FYP and will not be deployed for long-term use.

## 2.3 User Classes and Characteristics

The primary users of the **UniKL TutorFind** platform are **UniKL students**, who will use the platform to either share their skills or find peers to learn from. The platform will also have **administrators** (faculty or project team members) who will oversee the system’s functionality, though their role is minimal.

Since this is a **final year project (FYP)**, the platform is intended for short-term use, primarily for academic purposes, with limited user engagement after the project's completion. The platform is not intended to support large-scale or long-term user engagement beyond its academic purpose.

## 2.4 Operating Environment

The **UniKL TutorFind** platform will operate in both **mobile** and **web** environments:

* **Web Environment**: Optimized for the latest versions of major browsers, including **Google Chrome** and **Safari**.
* **Mobile Environment**: The mobile app will be available for **Android** and **iOS** users, using **Flutter** to create a consistent cross-platform experience.

Since the platform is for academic demonstration, deployment and support will be limited to the project phase, with no long-term infrastructure maintained after completion.

## 2.5 Design and Implementation Constraints

The development of the **UniKL TutorFind** platform is constrained by the following:

* **Cross-Platform Development**: The platform must be developed using **Flutter** for both Mobile and Web, limiting the use of other native technologies.
* **Supabase Dependency**: The platform depends on **Supabase** for authentication, real-time database, and cloud storage, which limits flexibility and scalability.
* **FYP Scope**: The project is intended for academic purposes and will not be maintained or deployed for long-term use after completion.

## 2.6 Assumptions and Dependencies

The development of the **UniKL TutorFind** platform assumes the following:

* **Access to Supabase and Flutter**: The platform will rely on **Supabase** for user authentication and data storage, and **Flutter** for mobile and web development.
* **UniKL Student Access**: The platform assumes that all users will have access to a **@unikl.edu.my** email address for user login and registration, and that they are UniKL students.
* **Stable Internet Connectivity**: Since the platform relies on **real-time data synchronization** through Supabase, stable internet connectivity is assumed for users to engage with the system.
* **Academic Timeline**: The platform’s development and deployment are constrained by the academic schedule, meaning that it will not be supported or maintained beyond the FYP phase.

# 3.0 System Features

**High Level Overview** of System Features:

|  |  |  |
| --- | --- | --- |
|  | Feature | Feature ID |
| 1 | Login and Registration | LR |
| 2 | Profile Creation | PC |
| 3 | Profile Matching | PM |
| 4 | Search and Filter | SF |
| 5 | In-App Chatting | IC |
| 6 | Rate and Review | RR |

## 3.1 System Feature: Login and Registration (LR)

### 3.1.1 Description and Priority

* **Feature Name**: Login and Registration
* Priority: High
* **Description**: This feature enables **UniKL students** to register and log in using their **@unikl.edu.my email**. It will employ **Supabase Authentication** for secure access to the platform, ensuring that only authorized users can interact with the system.

### 3.1.2 Stimulus/Response Sequences

|  |  |  |
| --- | --- | --- |
| Step | Stimulus (User Action) | System Response |
| 1 | User clicks "Sign Up" button. | The system displays a registration form requesting email, name, and password. |
| 2 | User enters email, name, and password in the registration form. | The system validates that the email follows the @unikl.edu.my format and checks for duplicate entries. |
| 3 | User clicks "Submit" after filling out the registration form. | The system sends a verification email to the @unikl.edu.my email address. |
| 4 | User clicks the verification link in the email to confirm the registration. | The system activates the account and allows the user to log in with the provided email and password. |
| 5 | User attempts to log in with their email and password. | The system authenticates the user via Supabase Authentication and grants access to the platform. |
| 6 | User enters incorrect email/password and tries to log in. | The system displays an error message like “Incorrect email or password.” |

### 3.1.3 Functional Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Functional Requirement | Priority | Description |
| LR-1 | The system must allow users to register using a valid @unikl.edu.my email address. | High | Ensures that only valid UniKL students can register. |
| LR-1-1 | The system must validate that the email follows the @unikl.edu.my format. | High | Ensures that only UniKL students with valid email formats can register. |
| LR-1-2 | The system must check for duplicate entries to prevent multiple accounts with the same email. | High | Prevents account duplication, ensuring each user has a unique registration. |
| LR-2 | The system must authenticate users using Supabase Authentication when they log in. | High | Verifies user credentials securely and prevents unauthorized access. |
| LR-2-1 | The system must hash passwords and store them securely. | High | Ensures that user passwords are encrypted and stored securely. |
| LR-3 | The system must send a verification email after registration to ensure the email address is valid. | Medium | Confirms user identity and ensures the registration process is legitimate. |
| LR-3-1 | The system must trigger the verification email upon successful registration. | Medium | Confirms user registration by sending an email with a unique verification link. |
| LR-4 | The system must allow users to log in using their @unikl.edu.my email and password. | High | Allows users to authenticate their sessions and gain access to their accounts. |
| LR-4-1 | The system must authenticate users by comparing the entered password with the stored hash. | High | Ensures that users can only access their account after correct password authentication. |
| LR-5 | The system must display error messages when registration or login fails (e.g., incorrect email/password). | High | Provides feedback to users when they attempt to register or log in with invalid credentials. |
| LR-5-1 | The system must show an error message for incorrect email/password during login attempts. | High | Helps users troubleshoot login issues by displaying relevant error messages. |
| LR-5-2 | The system must show an error message if the registration email is already in use. | Medium | Informs users of existing accounts associated with the email to avoid duplication. |

## 3.2 System Feature: Profile Creation (PC)

### 3.2.1 Description and Priority

* **Feature Name**: Profile Creation
* Priority: High
* **Description**: This feature allows users to create, update, and manage their profiles, including adding skills, learning preferences, and availability. It plays a key role in enabling the **Profile Matching** functionality of the platform.

### 3.2.2 Stimulus/Response Sequences

|  |  |  |
| --- | --- | --- |
| Step | Stimulus (User Action) | System Response |
| 1 | User clicks "Create Profile" after registration. | The system displays the profile creation form. |
| 2 | User enters name, skills they can teach, skills they want to learn, and availability. | The system validates that all fields are completed before allowing profile submission. |
| 3 | User clicks "Save Profile". | The system saves the profile data in the database and provides a confirmation message. |
| 4 | User edits their profile (e.g., adds new skills). | The system updates the profile and displays a confirmation message. |

### 3.2.3 Functional Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Functional Requirement | Priority | Description |
| PC-1 | The system must allow users to create a profile after registration. | High | Ensures that users can create personalized profiles for learning and teaching. |
| PC-1-1 | The system must prompt users to input name, skills they can teach, skills they want to learn, and availability. | High | Ensures complete profile information for matchmaking. |
| PC-2 | The system must allow users to add skills they can teach and skills they want to learn. | High | Enables students to indicate what they can teach and what they wish to learn, forming the basis for the matching system. |
| PC-2-1 | The system must allow users to update or delete existing skills. | Medium | Allows users to keep their profiles dynamic and up-to-date with their current teaching and learning preferences. |
| PC-3 | The system must allow users to update their profiles with new skills, availability, and preferences as necessary. | Medium | Supports profile modifications and ensures the platform adapts to changes in students' needs. |
| PC-4 | The system must store user profiles securely in the database. | High | Ensures data security and privacy of user information. |

## 3.3 System Feature: Profile Matching (PM)

### 3.3.1 Description and Priority

* **Feature Name**: Profile Matching
* Priority: High
* **Description**: This feature allows the system to match students based on **complementary skills**, **availability**, and **learning preferences**. It forms the core functionality of the platform and ensures that students are connected with the right peers.

### 3.3.2 Stimulus/Response Sequences

|  |  |  |
| --- | --- | --- |
| Step | Stimulus (User Action) | System Response |
| 1 | User clicks "Find a Peer" or enters a search term based on their learning needs. | The system searches for potential matches based on skills and availability. |
| 2 | The system matches students based on complementary skills and learning availability. | The system displays the matched students in a list. |
| 3 | The user selects a match from the list. | The system notifies both users of the match and opens a chat interface for them to connect. |

### 3.3.3 Functional Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Functional Requirement | Priority | Description |
| PM-1 | The system must match students based on complementary skills (e.g., programming, language). | High | Matches students with complementary learning goals, enabling peer-to-peer learning. |
| PM-1-1 | The system must filter matches by skills, ensuring that students are matched with others who can teach or learn similar subjects. | High | Guarantees that the match aligns with the skills of both students. |
| PM-2 | The system must consider availability when matching students. | Medium | Ensures that students are matched based on when they are free to participate in learning sessions. |
| PM-3 | The system must notify users when a match has been found. | Medium | Notifies users promptly so they can engage with their matched peers. |
| PM-4 | The system must provide the match quality score based on the alignment of skills, availability, and preferences. | Low | Provides transparency by ranking the match quality for students. |

## 3.4 System Feature: Search and Filters (SF)

### 3.4.1 Description and Priority

* **Feature Name**: Search and Filters
* Priority: High
* **Description**: This feature allows students to search for peers based on specific criteria like skills, availability, and location. The filter system makes it easier for users to narrow down search results and find the most relevant matches for learning sessions.

### 3.4.2 Stimulus/Response Sequences

|  |  |  |
| --- | --- | --- |
| Step | Stimulus (User Action) | System Response |
| 1 | User enters a search keyword in the search bar (e.g., *"Python"*). | The system displays a list of users who match the specified skill or keyword. |
| 2 | User applies filters such as skill category, availability, and location. | The system updates the search results by filtering users based on the applied criteria. |
| 3 | The system displays a filtered list of peers based on the selected criteria. | Users can scroll through the results and select the most suitable peer for learning. |
| 4 | User selects a peer from the search results. | The system displays the selected peer’s profile with further interaction options (e.g., message, connect). |

### 3.4.3 Functional Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Functional Requirement | Priority | Description |
| SF-1 | The system must allow users to search for peers by skills. | High | Provides students with an easy way to find peers based on specific skills they want to learn. |
| SF-1-1 | The system must match search keywords to skills listed in user profiles. | High | Ensures that the system returns accurate results matching the search criteria. |
| SF-2 | The system must allow users to filter search results by skill category, availability, and location. | High | Enables users to narrow their search and find peers who fit specific learning needs. |
| SF-2-1 | The system must allow filtering by availability, allowing users to see peers who are free at specific times. | Medium | Filters results based on time to help students find peers available when they are. |
| SF-2-2 | The system must allow filtering by location, allowing students to find peers nearby for in-person sessions. | Medium | Supports location-based filtering to facilitate offline learning sessions if desired. |
| SF-3 | The system must display search results in a clear and intuitive way. | High | Improves user experience by providing an easy-to-read list of search results. |
| SF-3-1 | The system must sort search results based on relevance to the user’s search criteria. | Medium | Ensures that more relevant matches appear at the top of the search results, improving user experience. |

## 3.5 System Feature: In-App Chatting (IC)

### 3.5.1 Description and Priority

* **Feature Name**: In-App Chatting
* **Priority**: Medium
* **Description**: This feature enables students to communicate with their peers directly within the platform, using real-time messaging. Supabase Realtime Database ensures messages are delivered instantly and allows students to coordinate learning sessions.

### 3.5.2 Stimulus/Response Sequences

|  |  |  |
| --- | --- | --- |
| Step | Stimulus (User Action) | System Response |
| 1 | User clicks the "Chat" button next to a matched peer. | The system opens a real-time chat window between the two students. |
| 2 | User types and sends a message. | The system updates the chat window in real-time and sends a notification to the recipient. |
| 3 | The recipient receives a notification of the new message. | The system updates the recipient’s chat interface to show the newly received message. |
| 4 | The recipient reads the message and replies. | The system updates the chat window in real-time for both users. |

### 3.5.3 Functional Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Functional Requirement | Priority | Description |
| IC-1 | The system must provide real-time messaging for matched peers. | High | Ensures that students can communicate instantly with their peers. |
| IC-1-1 | The system must update the chat window in real-time as new messages are sent or received. | High | Provides instant messaging with no delays, improving communication for session coordination. |
| IC-2 | The system must store chat histories for future reference. | Medium | Allows users to review previous messages, improving learning by recalling prior discussions. |
| IC-2-1 | The system must allow users to scroll through chat history. | Medium | Improves usability by enabling students to revisit earlier conversations. |
| IC-3 | The system must send notifications to users when new messages are received. | High | Keeps users informed of new messages they receive from peers. |
| IC-3-1 | The system must send both in-app notifications and mobile push notifications for new messages. | Medium | Ensures that users are notified in real-time, even if they are not actively using the app. |

## 3.6 System Feature: Rate and Review (RR)

### 3.6.1 Description and Priority

* **Feature Name**: Rate and Review
* **Priority**: Medium
* **Description**: This feature allows students to leave feedback after completing a learning session. Ratings (1-5 stars) and review comments help build trust within the platform, allowing future users to make informed decisions when selecting learning partners.

### 3.6.2 Stimulus/Response Sequences

|  |  |  |
| --- | --- | --- |
| Step | Stimulus (User Action) | System Response |
| 1 | User clicks the "Rate and Review" button after completing a session. | The system displays a rating interface with options for 1-5 stars and a text box for written feedback. |
| 2 | User selects a star rating and writes feedback. | The system stores the feedback and updates the average rating for the reviewed peer. |
| 3 | The system calculates and updates the average rating for the peer being reviewed. | The updated rating is displayed on the peer’s profile. |
| 4 | User reviews the feedback they have written in their review history. | The system displays the user’s previously submitted ratings and reviews for all sessions. |

### 3.6.3 Functional Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Functional Requirement | Priority | Description |
| RR-1 | The system must allow users to rate their peers after completing a learning session. | High | Ensures that students can provide feedback about their learning experiences. |
| RR-1-1 | The system must allow ratings from 1 to 5 stars. | High | Provides a clear, simple method for rating the quality of the learning experience. |
| RR-2 | The system must allow users to leave detailed written feedback after a session. | Medium | Helps users provide constructive feedback, improving future learning sessions. |
| RR-2-1 | The system must allow users to edit or delete their feedback within a specified time frame after submission. | Low | Enables users to update their reviews if they need to amend any feedback. |
| RR-3 | The system must calculate and update the average rating for a user after every review. | Medium | Provides transparency to users regarding the quality of their peers, fostering trust and accountability. |
| RR-3-1 | The system must display the average rating on the user’s profile page. | Medium | Displays the peer's rating score for others to review before engaging with them. |

# 4.0 External Interface Requirements

This section describes the interfaces that the UniKL TutorFind system will interact with, including user interfaces, hardware, software, and communication protocols.

## 4.1 User Interfaces (UI Design for Web and Mobile)

The UniKL TutorFind platform will feature a consistent and intuitive user interface across both its web and mobile (Android and iOS) applications, developed using Flutter.

* **Design Principles:** The UI will prioritize simplicity, ease of navigation, and clear visual hierarchy to ensure a positive user experience. A clean and modern aesthetic will be employed, aligning with contemporary mobile and web design trends.
* **Responsive Design:** The UI will be fully responsive, adapting seamlessly to various screen sizes and orientations, from small mobile devices to large desktop monitors. This ensures optimal usability regardless of the access device.
* **Interactive Elements:** All interactive elements (buttons, input fields, links) will be clearly distinguishable, provide visual feedback upon interaction, and have adequate touch target sizes for mobile users.
* **Feedback Mechanisms:** The UI will provide clear feedback for user actions, including success messages, error notifications, and loading indicators, to guide users through interactions.
* **Accessibility:** Efforts will be made to ensure basic accessibility standards are met to allow a broader range of users to interact with the platform.
* **Key Screens (Examples):**
* Login/Registration Screen
* Profile Creation/Editing Screen
* Dashboard/Home Screen (displaying matched peers, suggested skills)
* Search and Filter Results Screen
* Peer Profile View Screen
* In-App Chat Interface
* Rate and Review Submission Interface

## 4.2 Hardware Interfaces

The UniKL TutorFind application itself does not directly interface with specialized hardware. However, the development, testing, and operational environments require specific hardware specifications to ensure optimal performance and functionality.

* **For Web Application Testing/Usage:**
* **Processor:** Intel Core i5 (8th Generation or newer) / AMD Ryzen 5 (2000 series or newer) or equivalent.
* **RAM:** Minimum 8 GB (16 GB recommended for optimal performance during extensive testing and development).
* **Storage:** 256 GB SSD (Solid State Drive) or higher, with sufficient free space for operating system, development tools, test builds, and test data.
* **For Mobile Application Testing/Usage:**
* **Physical Android Devices:** Android 10 or newer.
* **Physical iOS Devices:** iOS 14 or newer.
* **Emulators:** Android Studio Emulator for initial development testing and convenience, minimizing reliance on physical devices for every test iteration.

## 4.3 Software Interfaces

The UniKL TutorFind platform heavily relies on various software interfaces and external services for its functionality.

* **Operating Systems (Development & Testing):**
* Windows 10 Pro (64-bit) or Windows 11 Pro (64-bit).
* macOS Catalina (10.15) or newer (for iOS development and testing).
* Latest stable versions of Android OS on physical devices and emulators.
* Latest stable versions of iOS on physical devices and simulators.
* **Web Browsers (for Web Application Testing/Usage):**
* Google Chrome (latest stable version).
* Mozilla Firefox (latest stable version).
* Apple Safari (latest stable version, on macOS).
* Microsoft Edge (latest stable version).
* **Development and Runtime Environments:**
* **Flutter SDK:** Latest stable release for cross-platform development.
* **Android Studio:** Latest version, including Android SDK and build tools, for Android development and emulation.
* **Xcode:** Latest version, for iOS development and simulators.
* **Supabase CLI and Associated SDKs:** For interaction with Supabase backend services (Authentication, Firestore, Realtime Database, Cloud Functions if needed).
* **PHP (version compatible with Laravel, e.g., PHP 8.x):** If any backend APIs are built outside Supabase.
* **Composer (PHP dependency manager):** For managing PHP dependencies.
* **Node.js and npm/yarn:** For front-end asset compilation (if used) or other JavaScript-based tooling.
* **MySQL Database Server (latest stable version):** If any local database or external SQL database is integrated beyond Supabase.
* **Testing and Documentation Tools:**
* **Version Control System:** Git, primarily managed via GitHub for collaborative development and code versioning.
* **Integrated Development Environments (IDEs):** VS Code, Android Studio, Xcode.
* **PDF Reader:** For reviewing test plans and requirements documents.
* **Microsoft Office Suite or equivalent:** For documentation, test reports, and project management.

## 4.4 Communication Interfaces

The UniKL TutorFind platform relies on standard internet communication protocols for its core functionalities.

* **HTTPS/TCP/IP:** All network communication will occur over HTTPS (Hypertext Transfer Protocol Secure) utilizing TCP/IP for secure data transmission between the client applications (web/mobile) and Supabase backend services.
* **Supabase Realtime Database API:** This API serves as the primary communication interface for real-time messaging (in-app chat) and dynamic updates to user profiles and matching data.
* **Supabase Authentication API:** This API handles secure user registration, login, and session management.
* **Email Services:** For sending account verification emails during the registration process. This will interface with a Supabase-integrated or external email service.
* **Notifications (Push Notifications):** If mobile push notifications are implemented (as per functional requirements for chat), the system will interface with Supabase Cloud Messaging (FCM) or platform-specific notification services (e.g., Apple Push Notification Service, Google Cloud Messaging).
* **Stable Internet Connectivity:** Consistent and robust network connectivity (both wired and wireless) is critical for the application's functionality, especially for real-time features like chat and data synchronization with Supabase.

# 5.0 Use Case Models

This section presents the detailed Use Case Models for the UniKL TutorFind platform, illustrating the interactions between users (Students) and the system's various functionalities. The use case statements provide comprehensive textual descriptions of each interaction, and their corresponding diagrams are referenced in the companion Software Test Plan.

* **Note:** While the diagrams themselves are not directly embedded in this Software Requirement Specification, their content and details are fully represented in the accompanying Use Case Statements and were instrumental in the development and testing phases as illustrated in the Software Test Plan.

## 5.1 Use Case: Login and Registration (LR)

This use case describes how a UniKL student registers for a new account and securely logs into the UniKL TutorFind platform.

* **Actor:** Student
* **Description:** The student provides their @unikl.edu.my email address and a password to create a new account or uses existing credentials to log in. The system authenticates the user and grants access.
* **Preconditions:** None for registration; an existing account for login.
* **Postconditions:** User is logged in to the system.
* **Flow of Events:**

1. The student accesses the UniKL TutorFind application.
2. The system presents the login/registration interface.
3. To Register:
   1. The student selects the "Sign Up" option.
   2. The student enters a valid @unikl.edu.my email, name, and desired password.
   3. The system validates the email format and checks for uniqueness.
   4. The system sends a verification email to the provided UniKL email address.
   5. The student clicks the verification link in the email.
   6. The system activates the student's account.
   7. The system directs the student to the login page.
4. To Login:
   1. The student enters their registered @unikl.edu.my email and password.
   2. The system authenticates the credentials against Supabase Authentication.
   3. If authentication is successful, the student is granted access to their dashboard.
   4. If authentication fails, the system displays an error message (e.g., "Invalid credentials").

## 5.2 Use Case: Profile Creation and Management (PC)

This use case describes how a registered UniKL student creates and manages their personal profile, detailing their skills and availability.

* **Actor:** Student
* **Description:** After successful registration, the student creates their profile, which includes essential information like skills they can teach, skills they want to learn, and their availability for sessions. They can also update this information at any time.
* **Preconditions:** Student is logged in to the UniKL TutorFind platform.
* **Postconditions:** Student's profile is created or updated in the system.
* **Flow of Events:**

1. The student logs into the UniKL TutorFind platform.
2. The system identifies if the student has an existing profile.
3. To Create Profile:
   1. If no profile exists, the system prompts the student to create one.
   2. The student inputs their name, selects skills they can teach, skills they want to learn, and specifies their availability.
   3. The student saves the profile.
   4. The system stores the profile data in Supabase Firestore and confirms the creation.
4. To Update Profile:
   1. If a profile exists, the student navigates to their profile page.
   2. The student selects the "Edit Profile" option.
   3. The student modifies existing information (e.g., adds new skills, changes availability, updates location).
   4. The student saves the changes.
   5. The system updates the profile data in Supabase Firestore and confirms the update.

## 5.3 Use Case: Search for Peers (SF)

This use case describes how a student actively searches for other students based on specific skills or filters to find potential matches.

* **Actor:** Student
* **Description:** The student uses a search bar and applies various filters (e.g., skill category, availability, location) to find other students who meet their specific learning or teaching needs.
* **Preconditions:** Student is logged in to the UniKL TutorFind platform and has a created profile.
* **Postconditions:** A list of filtered peer profiles is displayed.
* **Flow of Events:**

1. The student logs into the UniKL TutorFind platform.
2. The student navigates to the "Search" or "Find Peers" section.
3. The system displays a search bar and available filter options.
4. To Search by Keyword:
   1. The student enters a keyword (e.g., "Python programming") into the search bar.
   2. The system retrieves and displays a list of students whose profiles match the keyword.
5. To Apply Filters:
   1. The student selects one or more filter criteria (e.g., "Available on Weekends," "Skills to Teach: Web Development," "Location: UniKL MIIT").
   2. The system refines the displayed list of students based on the applied filters.
   3. The student can then view detailed profiles of the displayed peers.

## 5.4 Use Case: Match with Peers (PM)

This use case describes how the system intelligently identifies and presents compatible peers to a student based on their profile information.

* **Actor:** System (initiates the matching), Student (receives suggestions)
* **Description:** The system continually analyzes student profiles to identify complementary skill sets and mutual availability, suggesting potential learning partners. Students can also initiate a search that triggers a matching process.
* **Preconditions:** Students involved have created and updated profiles.
* **Postconditions:** The system presents a list of matched peers; optional notification to matched students.
* **Flow of Events:**

1. The system performs a background process to identify potential matches based on:
   1. skillsToTeach matching with another student's skillsToLearn.
   2. availability overlaps between potential matches.
   3. (Optional) Other preferences like location.
2. The system displays these suggested matches on the student's dashboard or a dedicated "Matches" section.
3. Alternatively (User-Initiated Matching):
   1. The student performs a search (as per Use Case 5.3).
   2. The system immediately processes the search query to find the most relevant matches based on the student's stated needs and the available profiles.
   3. The system presents the matched peer profiles, often with a "Connect" or "Message" option.
   4. (Optional) The system sends an in-app notification to the matched students indicating a potential connection.

## 5.5 Use Case: In-App Chatting (IC)

This use case details how students communicate directly with their matched peers within the application.

* **Actor:** Student
* **Description:** Students can engage in real-time text-based conversations with peers they have matched with, facilitating the coordination of learning sessions, sharing resources, and general communication.
* **Preconditions:** Student is logged in, and a match with another peer has been established.
* **Postconditions:** Messages are exchanged and stored in chat history.
* **Flow of Events:**

1. The student navigates to their "Matches" or "Messages" section.
2. The student selects a matched peer to initiate or continue a chat.
3. The system opens the chat interface for the selected conversation.
4. The student types a message into the input field.
5. The student sends the message.
6. The system instantly sends the message to the recipient via Supabase Realtime Database.
7. The system displays the sent message in the sender's chat history.
8. The recipient receives the message in real-time, and their chat interface updates.
9. The system sends an in-app and/or mobile push notification to the recipient for new messages.
10. Both students can scroll through the chat history to review past conversations.

## 5.6 Use Case: Rate and Review (RR)

This use case describes how a student provides feedback on a completed learning session with a peer.

* **Actor:** Student
* **Description:** After a learning session, the student can rate their peer (1-5 stars) and provide optional written feedback. This contributes to the peer's overall rating and helps other students make informed decisions.
* **Preconditions:** Student has completed a learning session with a peer.
* **Postconditions:** The peer's rating is updated, and the review is recorded.
* **Flow of Events:**

1. The student completes a learning session with a peer.
2. The system prompts the student to rate and review the session (e.g., via a notification or a "Past Sessions" log).
3. The student navigates to the "Rate and Review" interface for that specific session/peer.
4. The system displays a star rating selector (1-5 stars) and a text area for comments.
5. The student selects a star rating.
6. The student optionally types a written comment.
7. The student submits the review.
8. The system stores the review in Supabase Firestore.
9. The system recalculates the average rating for the reviewed peer.
10. The system updates and displays the new average rating on the reviewed peer's profile.
11. The student can later view their submitted reviews in a "My Reviews" section.

# 6.0 Non-Functional Requirements

Non-functional requirements define the quality attributes of the system and constraints on its operation. Meeting these requirements is crucial for the overall user experience and system reliability, especially given the academic scope.

## 6.1 Performance

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Non-Functional Requirement** | **Priority** | **Description** |
| NFR-P1 | The system must ensure login and registration processes complete within 3 seconds under normal network conditions. | High | Ensures a quick and efficient onboarding experience for users. |
| NFR-P2 | The system must ensure profile creation and updates save and display confirmation within 2 seconds. | High | Provides immediate feedback to users on profile modifications. |
| NFR-P3 | The system must return search queries and filter applications results within 3 seconds for up to 100 concurrent users. | High | Guarantees timely search results, even with moderate user load. |
| NFR-P4 | The system must deliver and display real-time chat messages within 1 second. | High | Ensures a responsive and fluid communication experience. |
| NFR-P5 | The system must process rate and review submissions within 2 seconds. | Medium | Provides quick confirmation for user feedback. |
| NFR-P6 | The system must support up to 100 concurrent active users without significant degradation in response time. | Medium | Ensures stability and performance for the expected academic user base. |
| NFR-P7 | The application should gracefully handle typical usage spikes expected from a university-level student population. | Low | Prevents system crashes or severe slowdowns during peak academic usage. |

## 6.2 Safety

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Non-Functional Requirement** | **Priority** | **Description** |
| NFR-S1 | The system must ensure the integrity of user profiles, chat histories, and ratings/reviews. | High | Prevents data corruption or loss through robust database operations and Supabase's inherent data consistency. |
| NFR-S2 | The system must minimize risks of harm or negative impact to users or their data. | High | Addresses the overall safety of user interaction and data. |
| NFR-S3 | The platform should include basic measures to prevent misuse. | Medium | Incorporates features like the review and rating system to provide transparency and potential for reporting inappropriate content. |
| NFR-S4 | The system design will adhere to ethical principles by ensuring data privacy, transparent matching algorithms, and mechanisms for users to control their profile visibility and interactions. | High | Ensures responsible handling of user data and interactions. |

## 6.3 Security

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Non-Functional Requirement** | **Priority** | **Description** |
| NFR-SC1 | The system must use Supabase Authentication to securely verify user identities based on their @unikl.edu.my email addresses and hashed passwords. | High | Ensures secure access and prevents unauthorized user impersonation. |
| NFR-SC2 | Passwords must be securely hashed and never stored in plain text. | High | Protects user credentials from exposure in case of a data breach. |
| NFR-SC3 | Email verification is a mandatory step for account activation to prevent unauthorized access. | High | Confirms user identity and legitimacy of registration. |
| NFR-SC4 | Users must only be able to access and modify their own profiles and chat histories. | High | Enforces strict data access control based on user identity. |
| NFR-SC5 | Access to administrative functions (if any implemented) must be restricted to authorized personnel. | High | Protects sensitive system management features. |
| NFR-SC6 | Supabase Security Rules will be configured to enforce data access permissions. | High | Utilizes Supabase's native security features to control data read/write operations. |
| NFR-SC7 | All data transmitted between the client (web/mobile) and the Supabase backend must be encrypted using HTTPS. | High | Protects data in transit from interception and tampering. |
| NFR-SC8 | Sensitive user data stored in Supabase Firestore/Realtime Database will be protected by Supabase's inherent security features and security rules. | High | Ensures data at rest is secure within the Supabase environment. |
| NFR-SC9 | Secure coding practices will be followed to prevent common web and mobile vulnerabilities. | Medium | Reduces the risk of common security flaws like SQL injection or cross-site scripting. |
| NFR-SC10 | Regular review of Supabase security rules will be conducted to ensure they are robust and prevent unauthorized data access or modification. | Medium | Maintains a strong security posture through continuous vigilance. |

## 6.4 Software Quality Attributes (for both Web and Mobile platforms)

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Non-Functional Requirement** | **Priority** | **Description** |
| NFR-QA1 | The user interface (UI) on both web and mobile platforms must be intuitive, easy to learn, and efficient to use. | High | Ensures a positive and productive user experience for all students. |
| NFR-QA2 | Navigation should be clear and consistent across the application. | High | Helps users find their way around the app without confusion. |
| NFR-QA3 | Error messages and system feedback should be clear, concise, and helpful. | Medium | Guides users effectively when issues arise. |
| NFR-QA4 | The platform should minimize user effort to achieve common tasks (e.g., finding a peer, sending a message). | High | Promotes efficiency and user satisfaction. |
| NFR-QA5 | The system should operate without critical failures (crashes, data loss) during normal operation. | High | Ensures the application is reliable and dependable. |
| NFR-QA6 | Error handling mechanisms will be implemented to ensure the system remains stable even when encountering unexpected conditions. | High | Improves system robustness in the face of network issues or external service failures. |
| NFR-QA7 | The system should accurately perform its core functions (login, profile management, matching, search, chat, rate/review) as specified. | High | Guarantees that the main features work as intended. |
| NFR-QA8 | The codebase will be well-structured, modular, and documented to facilitate future modifications, bug fixes, and feature enhancements. | Medium | Supports easier maintenance and future development. |
| NFR-QA9 | Consistent coding standards and practices will be adhered to. | Medium | Ensures code readability and maintainability. |
| NFR-QA10 | Leveraging Flutter and Supabase contributes to maintainability by providing a unified codebase and managed backend services. | Medium | Highlights architectural choices that enhance maintainability. |
| NFR-QA11 | The use of Flutter ensures high portability across Android, iOS, and Web platforms from a single codebase. | High | Allows the application to run seamlessly on multiple platforms. |
| NFR-QA12 | The system's design will facilitate effective testing. | High | Simplifies the testing process and helps ensure quality. |
| NFR-QA13 | Comprehensive test cases will be developed and executed to ensure functional and non-functional requirements are met. | High | Verifies that all requirements, including quality attributes, are satisfied. |

# 7.0 Other Requirements

This section covers additional requirements pertinent to the UniKL TutorFind project, including architectural considerations, testing methodologies, and deployment strategy.

## 7.1 Database Design

The UniKL TutorFind platform will primarily leverage **Supabase Firestore** as its NoSQL cloud-hosted database for storing and synchronizing user data across web and mobile platforms. This choice facilitates real-time data updates, simplified backend development, and scalability inherent to Supabase.

* **Firestore Collections:**
* **users Collection:** Stores individual user profiles, including fields such as:
* userId (Document ID, derived from Supabase Authentication UID)
* name
* email (@unikl.edu.my format)
* skillsToTeach (Array of strings)
* skillsToLearn (Array of strings)
* availability (e.g., Array of objects, each with day, startTime, endTime)
* location (String, e.g., "Kuala Lumpur", "UniKL MIIT")
* averageRating (Numeric, calculated)
* totalReviews (Numeric)
* createdAt (Timestamp)
* updatedAt (Timestamp)
* **chats Collection:** Stores individual chat messages within conversations.
* chatId (Document ID, representing a unique conversation between two users)
* participants (Array of userIds)
* messages (Sub-collection within chatId, each document representing a message)
* messageId (Document ID)
* senderId
* receiverId
* text
* timestamp
* readStatus
* **reviews Collection:** Stores individual ratings and reviews.
* reviewId (Document ID)
* reviewerId
* reviewedUserId
* rating (Integer, 1-5)
* comment (String, optional)
* timestamp
* **Data Serialization:** For complex data structures or nested arrays that exceed Firestore's direct capabilities, data will be serialized to JSON strings using JSON.stringify() before saving and parsed using JSON.parse() upon retrieval.
* **Supabase Realtime Database:** May be used in conjunction with Firestore specifically for the real-time chat feature due to its lower latency for frequent, small data updates.
* **Security Rules:** Comprehensive Supabase Security Rules will be implemented to ensure that users can only read/write their own private data and access public data as intended.

## 7.2 Testing Requirements

Comprehensive testing is critical to ensure the quality and reliability of the UniKL TutorFind platform. The testing strategy follows the principles outlined in the Software Test Plan.

* **Test Objectives:**
* Identify and document as many software defects as possible.
* Gain confidence in the quality level of the software, confirming it meets specified requirements.
* Provide accurate and timely information to stakeholders for release decisions.
* Prevent defects by implementing testing activities early in the SDLC.
* **Scope of Testing:**
* All key features of the UniKL TutorFind platform will be thoroughly tested: Login and Registration, Profile Creation, Profile Matching, Search and Filters, In-App Chatting, and Rate and Review.
* Testing will cover both web and mobile (Android and iOS) applications.
* Both functional and non-functional requirements (performance, security, usability) will be validated.
* **Test Process (Iterative RAD Model):**
* **Iterative Test Planning and Design:** Test planning occurs for each development increment. Reusable test cases are identified, and new tests are designed for new functionality.
* **Multi-Level Testing per Increment:**
* **Component Testing (Unit Testing):** Focuses on individual software components in isolation (often by developers).
* **Integration Testing:** Verifies interfaces and interactions when combining components.
* **System Testing:** Evaluates the complete, integrated system against specified functional and non-functional requirements.
* **Acceptance Testing (UAT in Academic Context):** Final stage involving end-users/supervisors to confirm the system satisfies acceptance criteria and user needs.
* **Continuous Testing & Regression:** Continuous integration testing and regression testing will be performed to ensure new changes do not adversely affect existing functionalities.
* **Verification and Validation (V&V):** Both verification ("Are we building the product right?") and validation ("Are we building the right product?") will be conducted for each increment.
* **Test Levels and Techniques:**
* **Black Box Testing:** Extensively used to verify application functionality through the user interface, based on specified requirements, without considering internal code structure. This is ideal for all user-facing features.
* **Risk-Based Testing:** Prioritizes testing efforts based on the potential risks (likelihood and impact) associated with different functionalities. Critical functionalities will receive more rigorous testing, ensuring efficient resource allocation within project constraints. Test cases will include a "Risk Number" (e.g., 1-5, Lowest to Highest).
* **Pass/Fail Criteria:**
* **Pass Criteria:** All critical and high-priority test cases pass, software functions as described, no blocking/critical defects remain open, and all reported defects for the current iteration are resolved/deferred.
* **Fail Criteria:** Any critical/high-priority test case fails, a blocking defect is found preventing further testing, software crashes/hangs, unresolved defects exceed threshold, or software fails to meet SRS requirements.
* **Suspension and Resumption Criteria:** Clearly defined criteria exist for suspending testing (e.g., >50% critical test case failures, showstopper defects, unstable environment) and for resuming testing (all blocking defects fixed, stable environment, new build deployed, retest passes).
* **Entry and Exit Criteria:** Formal criteria for beginning and completing each test iteration or phase (e.g., SRS finalized, test plan approved, environment set up, all planned test cases executed, critical defects closed, test summary report approved).
* **Test Progress Monitoring and Control:**
* **Test Metrics:** Defect-based metrics (number found/fixed, severity, priority), test case-based metrics (execution percentage, pass/fail rates), and test object-based metrics (requirement coverage) will be collected.
* **Test Reporting:** Regular status reports (e.g., weekly) and a final Test Summary Report will be produced to communicate progress, issues, and risks to stakeholders.
* **Test Control:** Corrective actions (adjusting schedules, reallocating resources, re-prioritizing tests) will be taken based on monitoring results.
* **Incident Management:** A systematic process for reporting, classifying, tracking, and resolving deviations (defects). Incident reports will be objective, factual, and include identification, description, impact (severity/urgency), status, and history.
* **Configuration Management:** Ensures the integrity and traceability of all work products (software builds, test plans, test cases, test data, documentation) throughout the project lifecycle, crucial for regression testing and defect replication.

## 7.3 Deployment Considerations

As a Final Year Project (FYP), the deployment of UniKL TutorFind will be limited in scope and duration, focusing on demonstration and academic assessment.

* **Deployment Environment:** The application will be deployed to Supabase Hosting for the web application and distributed via development builds (e.g., APK for Android, TestFlight for iOS) for mobile, rather than public app stores.
* **Deployment Schedule:** The final deployment for academic demonstration is scheduled between 5/1/2026 and 18/1/2026, following the completion of all functionality testing.
* **Limited Maintenance:** Post-completion of the FYP, the application will not receive ongoing maintenance, updates, or long-term support. Its primary goal is to serve as a proof of concept and meet academic objectives.
* **Scalability (Academic Context):** While Supabase offers inherent scalability, the deployment for this project will not be configured for large-scale production usage, as the anticipated user base is limited to the UniKL student community for academic purposes.
* **Monitoring (Academic Context):** Basic monitoring of application performance and error logs will be conducted during the project phase to identify critical issues, but not as part of a continuous operational support model.

# Appendix A: Glossary

This appendix provides definitions for key terms used throughout this Software Requirements Specification.

* **Acceptance Testing (UAT):** A formal testing process to determine if a system satisfies its acceptance criteria and to enable the customer (in this context, the supervisor/academic panel) to determine whether to accept the system.
* **Black Box Testing:** A method of software testing that examines the functionality of an application without peering into its internal structures or workings, focusing on inputs and outputs based on specifications.
* **Component Testing:** The first level of testing, focusing on individual, smallest testable units or modules of software in isolation.
* **Deployment:** The process of making the software application available for use, typically on a server or hosting environment, allowing users to access it.
* **Defect (Fault/Bug):** A flaw in a component or system that can cause the component or system to fail to perform its required function.
* **Error:** A human action that produces an incorrect result; a mistake made by a person.
* **Failure:** The deviation of the component or system from its expected delivery, service, or result, often caused by a defect.
* **Supabase:** An open-source, Backend-as-a-Service (BaaS) platform that provides developers with a suite of tools to build applications, acting as an alternative to Firebase.
* **Flutter:** An open-source UI software development kit created by Google, used for developing natively compiled cross-platform applications from a single codebase.
* **Functional Requirement (FR):** A requirement that specifies a function that a component or system must perform.
* **FYP:** Final Year Project.
* **In-App Chatting (IC):** A system feature allowing real-time text-based communication between users within the application.
* **Incident:** Any event occurring during testing that requires investigation (e.g., a query, a problem, an observation, a suggestion for improvement). Often refers to a defect.
* **Integration Testing:** A phase in software testing in which individual software modules are combined and tested as a group to verify their interfaces and interactions.
* **ISTQB:** International Software Testing Qualifications Board.
* **Login and Registration (LR):** A system feature enabling user account creation and authentication.
* **Non-Functional Requirement (NFR):** A requirement that specifies a criterion that can be used to judge the operation of a system, rather than specific behaviors (e.g., performance, security, usability).
* **Peer-to-Peer (P2P):** A decentralized communications model where each party has equivalent capabilities. In UniKL TutorFind, it refers to students directly connecting to exchange skills.
* **Profile Creation (PC):** A system feature allowing users to create and manage their personal profiles with skills and availability.
* **Profile Matching (PM):** A system feature that connects users based on complementary skills and availability.
* **Quality Assurance (QA):** The application of planned and systematic quality activities to ensure that quality requirements are met.
* **Rapid Application Development (RAD):** An agile software development methodology emphasizing rapid prototyping, iterative development, and continuous feedback.
* **Rate and Review (RR):** A system feature allowing users to provide ratings and written feedback for their peers after a session.
* **Regression Testing:** A type of software testing that verifies that recent program or code changes have not adversely affected existing features or introduced new defects into previously tested software.
* **Reliability:** The capability of the software product to maintain a specified level of performance under given conditions and for a specified time period.
* **Risk-Based Testing:** A type of software testing that prioritizes testing efforts based on the potential risks (likelihood and impact) associated with different functionalities.
* **Search and Filters (SF):** A system feature enabling users to search for peers by criteria and refine results.
* **Software Requirements Specification (SRS):** A document that describes the intended purpose, features, and behavior of a software system.
* **Software Test Plan (STP):** A document detailing the scope, approach, resources, and schedule of intended test activities.
* **System Testing:** A level of testing that tests a complete and integrated system to evaluate the system’s compliance with the specified requirements, from a holistic perspective.
* **Usability:** The effort required to use, and the individual assessment of such use by a specific or implied group of users.
* **Verification:** Confirmation by examination and through the provision of objective evidence that specified requirements have been fulfilled ("Did we implement the system right?").
* **Validation:** Confirmation by examination and through the provision of objective evidence that the requirements for a specific intended use or application have been fulfilled ("Did we implement the right system?").

# Appendix B: Analysis Models

This appendix includes visual analysis models that further describe the system's structure and data relationships.

**Entity-Relationship Diagrams (ERDs):**

A screenshot of a computer

AI-generated content may be incorrect.

**Hierarchical representation of Firestore database structure:**

/collections

|-- /users

| |-- {userId} (document)

| | |-- email: string

| | |-- password: string

| | |-- name: string

| | |-- is\_verified: boolean

| | |

| | |-- /user\_skills (subcollection)

| | | |-- {skillId} (document)

| | | | |-- skill\_id: string (reference to /skills/{skillId})

| | | | |-- skill\_type: string

| | |

| | |-- /availability (subcollection)

| | | |-- {availabilityId} (document)

| | | | |-- day\_of\_week: string

| | | | |-- start\_time: time

| | | | |-- end\_time: time

| | | | |-- ...

| | |

| | |-- /ratings (subcollection)

| | | |-- {ratingId} (document)

| | | | |-- reviewer\_id: string (reference to /users/{reviewerId})

| | | | |-- stars: number

| | | | |-- comment: string

|

|-- /profiles

| |-- {userId} (document, same ID as user)

| | |-- bio: string

| | |-- profile\_picture\_url: string

| | |-- location: string

| | |-- average\_rating: number

|

|-- /skills

| |-- {skillId} (document)

| | |-- name: string

| | |-- category: string

|

|-- /chat\_rooms

| |-- {chatRoomId} (document)

| | |-- user1\_id: string

| | |-- user2\_id: string

| | |-- user1\_name: string (denormalized data)

| | |-- user2\_name: string (denormalized data)

| | |

| | |-- /messages (subcollection)

| | | |-- {messageId} (document)

| | | | |-- sender\_id: string

| | | | |-- message\_text: string

| | | | |-- sent\_at: datetime

| | | | |-- sender\_name: string (denormalized data)

| | | | |-- is\_read: boolean

|

|-- /reviews

| |-- {reviewId} (document)

| | |-- reviewer\_id: string

| | |-- reviewee\_id: string

| | |-- rating: number

| | |-- comment: string

|

|-- /ratings (Top-level collection for many-to-many ratings)

| |-- {ratingId} (document)

| | |-- reviewer\_id: string

| | |-- reviewee\_id: string

| | |-- stars: number

| | |-- comment: string

# Appendix C: Issues List

This appendix serves as a list of identified issues, defects, or open questions that arose during the requirements elicitation, analysis, or specification process.

**Current Status:**

As of the current development phase, no specific issues, defects, or open questions have been formally identified. The mobile application is currently in the planning stages of development, while the web application has been partially completed. This section will be updated as new issues arise during the ongoing development and testing of both applications.