

**INFORMATION REGARDING DOCUMENT AND ITS APPROVAL**

**Version History**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version Number | Date | Amend By | Role in Project | Amendment reason |
| 1.0 | **05/05/2024** | **Alyssa Husna binti Jamarizan**  **Alya Azwin binti Zamri** | **Developer/Tester** | Initial Document Creation |
| 2.0 | 06/09/2025 | Alyssa Husna binti Jamarizan | Developer/Tester | Application Name and Database Technology Change |

**Document Approvals**

|  |  |  |  |
| --- | --- | --- | --- |
| Name of Approver | Role in Project | Signature or Electronic Approval | Date Approved |
| Mdm. Zanariah binti Abu Bakar | Supervisor |  |  |

**I. Software Test Plan Document Purpose:**

The purpose of this document is to define the scope, approach, resources, and schedule of the testing activities required to ensure the software meets its specified requirements and quality standards. It outlines various testing phases, test objectives, test strategies, and the criteria for entering and exiting each testing stage. It aims to provide a comprehensive guide for the testing team and a clear understanding of the overall testing process to all project stakeholders.

**II. Software Test Plan Contents:**

This document will give a full picture of how the software will be tested. It starts with the Test Policy, which lays out the basic rules. An Introduction to the Test Plan gives a brief look at the system, lists what we assume, and points to other documents. The main Test Strategy explains test goals, what features will be tested, how we will test over time, and rules for passing, failing, stopping, and starting tests. It also talks about possible problems. Then, Test Cases are shown with their Use Cases for each main feature. The Test Environment section lists the needed computers, software, and people. Lastly, the Test Schedule covers how testing will be set up, planned, timed, and how we will check progress, handle risks, deal with issues, and manage changes. A section for Definitions, Acronyms, and Abbreviations is also included.

**III. Software Test Plan Document in terms of topics to be covered:**

We cover topics starting with the Test Policy, including terms and rules, and explains the Rapid Application Development (RAD) method used. The Introduction provides a system overview. The Test Strategy covers test goals, the features to be tested, the testing process steps, rules for testing phases, and how to handle risks, using Black Box and Risk-Based Testing. Specific Test Cases are then shown. The Test Environment section lists needed items and team roles. Finally, the Test Schedule covers how testing is planned, timed, and managed, including checking progress, dealing with problems, and managing changes. A glossary for Definitions, Acronyms, and Abbreviations is also here.

**Table of Contents**

[1.0 Test Policy 1](#_Toc200910231)

[1.1 Software Testing Term Definition 1](#_Toc200910232)

[1.2 Advantages of Testing 2](#_Toc200910233)

[1.3 Testing Principles 3](#_Toc200910234)

[1.4 Test Missions & Objectives 4](#_Toc200910235)

[1.5 Communication Among Stakeholders 4](#_Toc200910236)

[1.6 SDLC Identification 5](#_Toc200910237)

[2.0 Introduction to Test Plan 7](#_Toc200910238)

[2.1 System Description 7](#_Toc200910239)

[3.0 Test Strategy 10](#_Toc200910240)

[3.1 Test Objective 10](#_Toc200910241)

[3.2 Scope 10](#_Toc200910242)

[3.3 Scope 11](#_Toc200910243)

[3.3.1 Features Tested 11](#_Toc200910244)

[3.4 Test Process 11](#_Toc200910245)

[3.5 Item Pass/Fail Criteria 13](#_Toc200910246)

[3.5.1 Pass Criteria 13](#_Toc200910247)

[3.5.2 Fail Criteria 13](#_Toc200910248)

[3.6 Suspension Criteria and Resumption Requirements 14](#_Toc200910249)

[3.6.1 Suspension Criteria 14](#_Toc200910250)

[3.6.2 Suspension Criteria 14](#_Toc200910251)

[3.7 Entry and Exit Criteria 15](#_Toc200910252)

[3.7.1 Entrance Criteria 15](#_Toc200910253)

[3.7.2 Exit Criteria 15](#_Toc200910254)

[3.8 Risks and Contingencies 15](#_Toc200910255)

[3.8.1 Product Risks 16](#_Toc200910256)

[3.8.2 Project Risks 16](#_Toc200910257)

[3.9 Test Level 18](#_Toc200910258)

[3.9.1 Black Box Testing 18](#_Toc200910259)

[3.9.2 Risk Based Testing 18](#_Toc200910260)

[4.0 Test Cases (with Use Cases) 23](#_Toc200910261)

[4.1 UniKL TutorFind – Overall Application 23](#_Toc200910262)

[4.1.1 Use Case Diagram 23](#_Toc200910263)

[4.2 System Feature 1: Login and Registration (LR) 24](#_Toc200910264)

[4.2.1 Use Case Diagrams 24](#_Toc200910265)

[4.2.2 Use Case Statements 27](#_Toc200910266)

[4.2.3 Test Cases 29](#_Toc200910267)

[4.3 System Feature 2: Profile Creation (PC) 36](#_Toc200910268)

[4.3.1 Use Case Diagram 36](#_Toc200910269)

[4.3.2 Use Case Statements 39](#_Toc200910270)

[4.3.3 Test Cases 41](#_Toc200910271)

[4.4 System Feature 3: Profile Matching (PM) 46](#_Toc200910272)

[4.4.1 Use Case Diagram 46](#_Toc200910273)

[4.4.3 Test Cases 49](#_Toc200910274)

[4.5 System Feature 4: Search and Filters (SF) 54](#_Toc200910275)

[4.5.1 Use Case Diagram 54](#_Toc200910276)

[4.5.2 Use Case Statements 55](#_Toc200910277)

[4.5.3 Test Cases 57](#_Toc200910278)

[4.6 System Feature 5: In-App Chatting (IC) 63](#_Toc200910279)

[4.6.1 Use Case Diagram 63](#_Toc200910280)

[4.6.2 Use Case Statements 64](#_Toc200910281)

[4.6.3 Test Cases 67](#_Toc200910282)

[4.7 System Feature 6: Rate and Review (RR) 73](#_Toc200910283)

[4.7.1 Use Case Diagram 73](#_Toc200910284)

[4.7.2 Use Case Statements 74](#_Toc200910285)

[4.7.3 Test Cases 77](#_Toc200910286)

[5.0 Test Environment 82](#_Toc200910287)

[5.1 Test Site 82](#_Toc200910288)

[5.1.1 Developer Site 82](#_Toc200910289)

[5.2 Facilities required 82](#_Toc200910290)

[5.2.1 Lab Space 82](#_Toc200910291)

[5.2.2 Furniture 82](#_Toc200910292)

[5.2.3 Power Infrastructure 82](#_Toc200910293)

[5.2.4 Communication Facilities 83](#_Toc200910294)

[5.2.5 Network Connectivity 83](#_Toc200910295)

[5.3 Required hardware and software specifications 83](#_Toc200910296)

[5.3.1 Hardware Requirement 83](#_Toc200910297)

[5.3.2 Software Requirements 84](#_Toc200910298)

[5.4 Testing Group – Tester Involved 85](#_Toc200910299)

[5.5 Preparation and training required of the test team 85](#_Toc200910300)

[6.0 Test Schedule 87](#_Toc200910301)

[6.1 Test Organization 87](#_Toc200910302)

[6.2 Test Planning and Estimation 89](#_Toc200910303)

[6.3 Test Schedule (Testing and Deployment Timeline) 89](#_Toc200910304)

[6.4 Iterative Testing Approach (Per Increment) 90](#_Toc200910305)

[6.4.1 Development Increment 1: Login and Registration & Profile Creation (Completed by early June 2025) 91](#_Toc200910306)

[6.4.2 Development Increment 2: Profile Matching & In-App Chatting (Completed by mid-November 2025) 91](#_Toc200910307)

[6.4.3 Development Increment 3: Search and Filters & Rate and Review (Completed by late December 2025) 92](#_Toc200910308)

[6.4.4 Final Functionality Testing (22/12/25 - 4/1/26) 92](#_Toc200910309)

[6.5 Test Progress Monitoring and Control 92](#_Toc200910310)

[6.6 Risk Management 93](#_Toc200910311)

[6.7 Incident Management 94](#_Toc200910312)

[6.8 Configuration Management 95](#_Toc200910313)

[7.0 Definitions, Acronyms and Abbreviation 97](#_Toc200910314)

[7.1 Definitions 97](#_Toc200910315)

[7.2 Abbreviations 100](#_Toc200910316)

**List of Tables**

[Table 1: Risk Classification for Prioritisation 22](#_Toc200909780)

[Table 2 UC\_001 Use Case Statement 28](#_Toc200909781)

[Table 3 UC002 Use Case Statement 29](#_Toc200909782)

[Table 4 TC\_001.1 Test Case 30](#_Toc200909783)

[Table 5 TC\_001.2 Test Case 31](#_Toc200909784)

[Table 6 TC\_001.3 Test Case 32](#_Toc200909785)

[Table 7 TC\_002.1 Test Case 33](#_Toc200909786)

[Table 8 TC\_002.2 Test Case 34](#_Toc200909787)

[Table 9 TC\_002.3 Test Case 35](#_Toc200909788)

[Table 10 TC\_002.4 Test Case 36](#_Toc200909789)

[Table 11 UC\_003 Use Case Statement 40](#_Toc200909790)

[Table 12 UC\_004 Use Case Statement 41](#_Toc200909791)

[Table 13 TC\_003.1 Test Case 42](#_Toc200909792)

[Table 14 TC\_003.2 Test Case 43](#_Toc200909793)

[Table 15 TC\_003.3 Test Case 44](#_Toc200909794)

[Table 16 TC\_003.4 Test Case 45](#_Toc200909795)

[Table 17 UC\_004 Use Case Statement 48](#_Toc200909796)

[Table 18 UC\_004.1 Use Case Statement 48](#_Toc200909797)

[Table 19 UC\_004.2 Use Case Statement 49](#_Toc200909798)

[Table 20 UC\_004.3 Use Case Statement 49](#_Toc200909799)

[Table 21 TC\_004.1 Test Case 51](#_Toc200909800)

[Table 22 TC\_004.2 Test Case 52](#_Toc200909801)

[Table 23 TC\_004.3 Test Case 53](#_Toc200909802)

[Table 24 TC\_004.4 Test Case 54](#_Toc200909803)

[Table 25 UC\_005 Use Case Statement 56](#_Toc200909804)

[Table 26 UC\_005.1 Use Case Statement 56](#_Toc200909805)

[Table 27UC\_005.2 Use Case Statement 57](#_Toc200909806)

[Table 28 UC\_005.3 Use Case Statement 57](#_Toc200909807)

[Table 29 TC\_005.1 Test Case 58](#_Toc200909808)

[Table 30 TC\_005.2 Test Case 59](#_Toc200909809)

[Table 31TC\_005.3 Test Case 60](#_Toc200909810)

[Table 32 TC\_005.4 Test Case 61](#_Toc200909811)

[Table 33 TC\_005.5 Test Case 62](#_Toc200909812)

[Table 34 UC\_006 Use Case Statement 65](#_Toc200909813)

[Table 35 UC\_006.1 Use Case Statement 65](#_Toc200909814)

[Table 36 UC\_006.1 Use Case Statement 66](#_Toc200909815)

[Table 37 UC\_006.3 Use Case Statement 66](#_Toc200909816)

[Table 38 UC\_006.4 Use Case Statement 67](#_Toc200909817)

[Table 39 TC\_006.1 Test Case 68](#_Toc200909818)

[Table 40 TC\_006.2 Test Case 69](#_Toc200909819)

[Table 41 TC\_006.3 Test Case 70](#_Toc200909820)

[Table 42 TC\_006.4 Test Case 71](#_Toc200909821)

[Table 43 TC\_006.5 Test Case 72](#_Toc200909822)

[Table 44 UC\_007 Use Case Statement 75](#_Toc200909823)

[Table 45 UC\_007.1 Use Case Statement 75](#_Toc200909824)

[Table 46 UC\_007.2 Use Case Statement 76](#_Toc200909825)

[Table 47 UC\_007.3 Use Case Statement 76](#_Toc200909826)

[Table 48 UC\_007.4 Use Case Statement 77](#_Toc200909827)

[Table 49 TC\_007.1 Test Case 78](#_Toc200909828)

[Table 50 TC\_007.2 Test Case 79](#_Toc200909829)

[Table 51 TC\_007.3 Test Case 80](#_Toc200909830)

[Table 52 TC\_007.4 Test Case 81](#_Toc200909831)

[Table 53 TC\_007.5 Test Case 82](#_Toc200909832)

[Table 54 Test Planning & Estimation 90](#_Toc200909833)

**List of Figures**

[Figure 1 Overall Application Use Case Diagram 23](#_Toc200797248)

[Figure 2 Overall Feature 1 Use Case Diagram 24](#_Toc200797249)

[Figure 3 User Registration Use Case Diagram 25](#_Toc200797250)

[Figure 4 User Login Use Case Diagram 26](#_Toc200797251)

[Figure 5 Overall Feature 2 Use Case Diagram 36](#_Toc200797252)

[Figure 6 Profile Creation Use Case Diagram 37](#_Toc200797253)

[Figure 7 Profile Update Use Case Diagram 38](#_Toc200797254)

[Figure 8 Feature 3 Profile Matching Use Case Diagram 46](#_Toc200797255)

[Figure 9 Feature 4 Search and Filters Use Case Diagram 54](#_Toc200797256)

[Figure 10 Feature 5 In-App Chat Use Case Diagram 63](#_Toc200797257)

[Figure 11 Feature 6 Rate and Review Use Case Diagram 73](#_Toc200797258)

# **1.0 Test Policy**

1.1 Software Testing Term Definition

Software testing is a systematic and dynamic process of executing a program or application with the explicit intent of identifying software defects (bugs, errors, faults). It encompasses all life cycle activities concerned with the planning, preparation, and evaluation of software products and related work products to determine if they satisfy specified requirements, demonstrate fitness for purpose, and detect defects.

Key terms related to software testing:

* **Error:** A human action that produces an incorrect result (e.g., a developer making a mistake in coding or a user performing an operation incorrectly).
* **Defect (Fault/Bug):** A flaw in a component or system that can cause the component or system to fail to perform its required function. It is a discrepancy between the actual behavior and the expected behavior (as per specifications or stated requirements). Defects are often introduced as a result of errors.
* **Failure:** The effect of a defect when executing a program (test object) which appears to the "outside". It is the deviation of the component or system from its expected delivery, service, or result. A failure is an event where the system does not perform as required.
* **Deficiency:** Non-fulfillment of a requirement related to an intended or specified use. It can refer to an impairment of usability or a failure to meet performance expectations.
* **Defect Masking:** An occurrence in which one defect prevents the detection of another defect.
* **Debugging:** The development activity that finds, analyzes, and removes the cause of a failure. Testing shows the presence of defects; debugging removes them.

1.2 Advantages of Testing

Implementing a comprehensive testing strategy for the UniKL TutorFind project offers several significant advantages:

* **Defect Prevention and Early Detection:** Testing allows for the early identification and resolution of defects. Defects introduced early in the development process (e.g., in requirements or design through static testing techniques like reviews) can cascade and multiply, leading to significantly higher costs and effort to fix if detected later.
* **Improved Software Quality:** Through rigorous and systematic testing, the overall quality, reliability, and stability of the UniKL TutorFind application are enhanced. This ensures that the product functions correctly, performs efficiently, is secure, and is usable, aligning with ISO/IEC 9126 quality attributes.
* **Enhanced User Satisfaction:** By verifying that the application meets user requirements and functions as expected, testing contributes directly to a positive user experience, fostering satisfaction among UniKL students utilizing the platform.
* **Cost Reduction:** Detecting and fixing defects early in the development cycle is significantly more cost-effective than addressing them after the application has been deployed or is in production. The cost of fixing a defect increases exponentially as it progresses through the software lifecycle.
* **Risk Mitigation:** Testing helps identify and mitigate potential product and project risks associated with software failures, data corruption, security vulnerabilities, or non-compliance with regulations. This proactive approach reduces the likelihood of negative impacts.
* **Information for Decision Making:** Test results provide objective information about the quality status of the software, enabling stakeholders (e.g., project managers, supervisors) to make informed decisions regarding release readiness, resource allocation, and project progression.
* **Ethical Considerations:** In many critical systems, software failures can cause harm to people, the environment, or significant financial loss to a company. Testing serves an ethical responsibility to minimize such risks.

## **1.3 Testing Principles**

The seven principles in testing are:

1. **Testing shows presence of defects, not their absence:** Testing can demonstrate that defects are present, but it cannot prove that there are no defects.
2. **Exhaustive testing is impossible:** Testing everything, including all combinations of inputs and preconditions, is not feasible except for trivial cases.
3. **Early testing:** To find defects early, testing activities should start as early as possible in the software development lifecycle.
4. **Defect clustering:** A small number of modules usually contain most of the defects discovered.
5. **Pesticide paradox:** If the same tests are repeated, eventually these tests will no longer find any new defects.
6. **Testing is context dependent:** Testing is done differently in different contexts. For example, safety-critical software requires different testing approaches than an e-commerce website.
7. **Absence-of-errors fallacy:** Finding and fixing many defects does not guarantee the success of a system if the system built is unusable or does not meet the user's needs and expectations.

## 

## **1.4 Test Missions & Objectives**

The common objectives of testing include:

* **Evaluate Work Products:** To assess the quality of work products such as requirements, design, and code for correctness, completeness, and adherence to standards.
* **Verify Requirements:** To ensure that all specified requirements (functional and non-functional) are accurately implemented in the UniKL TutorFind application.
* **Validate Fitness for Use:** To confirm that the UniKL TutorFind application is fit for its intended purpose and meets the real-world needs of UniKL students for skill exchange.
* **Build Confidence:** To build confidence in the level of quality of the software, thereby reducing the level of risk of failure in production.
* **Find Defects:** The primary and most recognized objective: to find as many defects as possible, enabling their removal before deployment.
* **Prevent Defects:** Through early involvement of testers in the requirements and design phases (e.g., reviews), and the use of static testing, defects can be prevented from being introduced.
* **Provide Information for Decision-Making:** To provide sufficient information to stakeholders to allow them to make informed decisions about the release or academic submission of the software.

1.5 Communication Among Stakeholders

Effective communication and a collaborative, non-blaming culture are crucial for successful testing and defect resolution, as highlighted in the "Psychology of Testing" concept in class:

* **Constructive Mindset:** Testers should adopt a professional, constructive mindset, aiming to collaborate with developers to improve quality rather than focusing on personal blame.
* **Independent Testing (Benefits):** While a small academic project may not have fully independent test teams, fostering an independent mindset (e.g., peer reviews by those not coding a specific module) can improve defect detection effectiveness. Independent testers are less biased by their own assumptions and code.
* **Objective Reporting:** Incident reports should be objective, factual, and focus on the problem (the defect and its observed failure), not on the individual who made the error. This facilitates efficient defect analysis and resolution.
* **Feedback Loop:** Clear and timely communication between testers, developers, and project supervisors is essential for a continuous feedback loop that drives quality improvements.

1.6 SDLC Identification

The UniKL TutorFind App utilizes the Rapid Application Development (RAD) model. RAD is a type of iterative-incremental development model. In this approach, software development occurs in small, manageable steps rather than as a single, large effort. This means testing is not a single, isolated phase at the end, but an ongoing activity throughout the development cycles. The integration of testing into the SDLC ensures that quality is built in, rather than "tested in" at the last minute. This approach aligns with the principle of "Early Testing" and emphasizes continuous verification and validation within each increment.

Testing within the RAD model is adapted to this iterative sequence:

* Test Planning: For each increment or intermediate delivery, reusable tests are planned.
* Test Reuse and Expansion: Existing tests can be reused in subsequent iterations, and new tests are developed for newly added functionality.
* Multi-Level Testing: Different testing levels (e.g., component, integration, system, acceptance testing) can be performed for each increment within an iteration.
* Continuous Testing: Continuous integration testing and regression testing are essential to ensure that new changes do not negatively impact existing functionalities.
* Verification and Validation: Both verification (checking if the product is built correctly) and validation (checking if the correct product is built) are conducted for each increment.

# **2.0 Introduction to Test Plan**

## **2.1 System Description**

This Acceptance Test Plan is for UniKL TutorFind: A Peer-to-Peer Student Learning Exchange Platform. The platform aims to address the challenge faced by UniKL students who possess valuable skills (such as coding, video editing, or public speaking) but lack a centralized means to share and acquire these skills with peers. By creating this online platform, UniKL TutorFind facilitates connections among students for collaborative learning, fostering knowledge exchange, skill development, and community building within UniKL. It also provides students with opportunities to gain valuable teaching and networking experience.

Key modules of the UniKL TutorFind platform include:

* Registration (UniKL Only Access): Ensures that only UniKL students with a valid @unikl.edu.my email address can register and log into the platform.
* User Profiles: Allows students to create, update, and manage their profiles, listing skills they can teach and skills they wish to learn.
* Matching System: Facilitates matching students based on complementary skills, availability, and specified preferences (e.g., learning method, location).
* Search & Filters: Enables students to search for peers based on various criteria such as skills, availability, location, and learning preferences.
* In-App Chat: Provides a real-time communication feature for students to coordinate learning sessions and exchange information directly within the application.
* Reviews & Ratings: Allows students to rate and review their learning sessions, providing constructive feedback and contributing to building trust within the platform.

**2.2 Assumptions**

The development and testing of the UniKL TutorFind platform assume 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.

**2.3 References**

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

* **Flutter Documentation:** Official documentation for mobile and web development.
* **Supabase Documentation:** Official documentation for Supabase database, authentication, and real-time features.
* **SRS Template:** Standard template used for structuring the Software Requirements Specification document.
* **UniKL TutorFind Project Proposal:** The proposal outlining the project's objectives, features, and scope (April 2025).
* **UniKL TutorFind Software Requirements Specification (SRS) Draft (01.05.2025 FYP UniKL TutorFind SRS Draft.pdf)**
* **UniKL TutorFind Final Software Test Plan (STP) (02.06.2025 FinalSTP UniKL TutorFind (Azwin & Alyssa).pdf)**

**2.4 Document Overview**

This document serves as the Acceptance Test Plan for the UniKL TutorFind platform. Its primary purpose is to define the scope, approach, resources, and schedule of the testing activities required to ensure the software meets its specified requirements and quality standards. This plan outlines various testing phases, test objectives, test strategies, and the criteria for entering and exiting each testing stage. It aims to provide a comprehensive guide for the testing team and a clear understanding of the overall testing process to all project stakeholders.

# **3.0 Test Strategy**

## 3.1 Test Objective

The primary objectives of testing for the UniKL TutorFind platform are to:

* **Find Defects:** Identify and document as many software defects as possible to ensure the quality and stability of the application.
* **Gain Confidence in Quality:** Build confidence in the quality level of the software, confirming that it meets its specified requirements and performs as expected.
* **Provide Information for Decision-Making:** Deliver accurate and timely information about the quality status of the software to stakeholders, enabling informed decisions regarding release readiness.
* **Prevent Defects:** Implement testing activities early in the software development lifecycle to prevent defects from being introduced and propagating through later stages.

## 3.2 Scope

The testing scope for the UniKL TutorFind platform encompasses both the web and mobile applications, focusing on verifying the functionality and performance of key features as defined in the Software Requirements Specification (SRS). This plan primarily focuses on **Acceptance Testing**, which will validate that the system meets the user's needs and business requirements.

The testing will cover all critical modules and their interactions, ensuring that the peer-to-peer skill matching application functions correctly in its intended operating environments.

## 

## 3.3 Scope

### 3.3.1 Features Tested

The following key features of the UniKL TutorFind platform will be thoroughly tested:

* **Login and Registration (LR):** Verification of user authentication, including successful registration with UniKL email addresses, login functionality, and handling of invalid/duplicate credentials.
* **Profile Creation (PC):** Validation of user profile creation, updates, and management, including listing skills to teach and skills to learn, and setting availability.
* **Profile Matching (PM):** Verification of the system's ability to match students based on complementary skills, availability, and specified preferences.
* **Search and Filters (SF):** Testing the functionality allowing users to search for peers using various criteria (skills, availability, location) and apply filters to refine results.
* **In-App Chatting (IC):** Validation of real-time communication between matched peers, including sending/receiving messages and chat history storage.
* **Rate and Review (RR):** Verification of the feature allowing users to rate learning sessions (1-5 stars) and provide written feedback.

## 3.4 Test Process

* The UniKL TutorFind application utilizes the **Rapid Application Development (RAD)** model, an iterative-incremental development model. Testing within this RAD model is adapted to the following sequence, integrating principles from the ISTQB syllabus regarding testing in various software development models (e.g., Iterative-Incremental, V-Model, W-Model concepts from Chapter 2 - Lifecycle):
* **Iterative Test Planning and Design:** For each increment or intermediate delivery, test planning occurs. Reusable test cases are identified and refined, and new tests are designed to cover newly added functionality. Test analysis and design begin in parallel with corresponding development stages ("Test Early" principle).
* **Multi-Level Testing per Increment:** Different test levels are performed for each increment within an iteration. This ensures comprehensive validation at various stages of development:
* **Component Testing (Unit Testing):** The first level of testing, focusing on individual software components (modules, units, or classes) in isolation. Objectives include ensuring correct and complete implementation of specified functionality, testing robustness, and evaluating internal quality attributes. Component tests are often carried out by developers, often using test harnesses (drivers and stubs) to simulate calling and called components.
* **Integration Testing:** The second level, where individual components or groups of components are combined and tested to verify their interfaces and interactions. Objectives include detecting defects in interfaces and the interaction between components. Various integration strategies (e.g., top-down, bottom-up, big-bang) can be considered, though incremental approaches are preferred in RAD.
* **System Testing:** The third level of testing, performed on a complete and integrated system to evaluate the system’s compliance with the specified requirements. This level examines the system from the customer's and end-user's perspective, validating if both functional and non-functional requirements are fully implemented.
* **Acceptance Testing (UAT in Academic Context):** The final stage of testing, often involving end-users or stakeholders, to determine if the system satisfies its acceptance criteria and to enable the customer (or in this academic context, the supervisor/academic panel) to determine whether to accept the system. This often involves user-centric validation.
* **Continuous Testing & Regression:** Continuous integration testing and regression testing are essential within this iterative model. **Regression testing** verifies that changes (e.g., bug fixes, new features) have not adversely affected existing, previously tested functionalities. This addresses the "Pesticide Paradox" by ensuring old defects don't reappear while new ones are sought.
* **Verification and Validation (V&V):** Both verification ("Are we building the product right?" – checking against specifications) and validation ("Are we building the right product?" – checking against user needs and fitness for purpose) are conducted for each increment, ensuring that the system is both technically sound and meets user expectations.

## 3.5 Item Pass/Fail Criteria

### 3.5.1 Pass Criteria

* All identified critical and high-priority test cases for a given feature or iteration are executed and pass without any unresolved defects.
* The software functionality performs exactly as described in the requirements and use case statements.
* No blocking or critical defects remain open.
* All reported defects for the current iteration are either resolved, deferred with stakeholder agreement, or identified as non-blocking.

### 3.5.2 Fail Criteria

* Any critical or high-priority test case fails to meet its expected outcome.
* A blocking defect is discovered that prevents further testing of a major functionality.
* The software crashes, hangs, or exhibits severe data corruption.
* The number of unresolved defects exceeds a predefined threshold.
* The software does not meet the functional or non-functional requirements specified in the SRS.

## 3.6 Suspension Criteria and Resumption Requirements

Testing activities for a specific module or test cycle will be suspended if the following criteria are met:

### 3.6.1 Suspension Criteria

* More than 50% of critical or high-priority test cases fail due to a single, underlying defect or a cluster of related defects.
* A "showstopper" or "blocking" defect (Class 1) is identified that prevents further execution of a significant portion of the test cases or prevents access to critical system functionalities.
* The test environment becomes unstable or unavailable, making reliable testing impossible.
* Key dependencies (e.g., database, external services) are non-functional or consistently produce errors.

### 3.6.2 Suspension Criteria

Testing will only resume when the following conditions are met:

* All blocking or showstopper defects identified as suspension criteria have been fixed and verified by the development team.
* The test environment is stable, configured correctly, and validated for readiness.
* A new build containing the fixes is deployed and confirmed to be stable.
* A retest of the previously failed critical test cases is performed and passes successfully.

## 3.7 Entry and Exit Criteria

### 3.7.1 Entrance Criteria

Testing for an iteration or phase can begin when:

* The Software Requirements Specification (SRS) is finalized and approved.
* The test plan and test cases are reviewed, approved, and readily available.
* The test environment is set up, configured, and stable.
* All necessary test data is prepared and loaded into the system.
* The development build or release candidate is available and has passed basic smoke tests.
* Required testing tools and resources are in place and accessible.

### 3.7.2 Exit Criteria

Testing for an iteration or phase can be considered complete when:

* All planned test cases for the current scope have been executed.
* All critical and high-priority defects have been resolved, retested, and closed.
* The number of open medium and low-priority defects is within acceptable limits, as agreed upon by stakeholders.
* Test coverage metrics (e.g., requirement coverage, test case coverage) meet the defined targets.
* Regression tests for previously stable functionalities have been executed and passed.
* A final test summary report is prepared and approved by relevant stakeholders.

## 3.8 Risks and Contingencies

This section identifies potential risks that could impact the successful completion of the UniKL TutorFind project and its testing activities, along with contingency plans to mitigate them. Risks are categorized as product risks (related to the software itself) and project risks (related to project management).

### 3.8.1 Product Risks

* **Risk:** Software delivered with critical functional defects (e.g., matching algorithm errors, chat failures).
  + **Contingency:** Implement thorough Unit, Integration, and System testing; prioritize critical path test cases; conduct comprehensive Acceptance Testing.
* **Risk:** Performance issues (slow response times, high resource usage) due to large user base or complex operations.
  + **Contingency:** Conduct performance testing; optimize database queries and algorithms; consider server scaling options.
* **Risk:** Security vulnerabilities (unauthorized access, data breaches).
  + **Contingency:** Implement secure coding practices; conduct security testing; regularly review Supabase security rules; ensure proper authentication and authorization.
* **Risk:** Usability problems (confusing interface, difficult navigation).
  + **Contingency:** Conduct usability testing (informal reviews, walkthroughs); gather early user feedback through prototypes; adhere to UI/UX design guidelines.

### 3.8.2 Project Risks

* **Risk:** Unrealistic Schedule/Timeline or development delays.
  + **Contingency:** Prioritize test cases based on criticality; escalate delays to project management/supervisor; consider reducing scope of low-priority features if necessary; establish clear milestones and monitor progress.
* **Risk:** Insufficient Test Data, leading to incomplete test coverage.
  + **Contingency:** Develop a detailed test data generation strategy; utilize data masking tools for sensitive data; create scripts for automated data generation; ensure diverse data sets.
* **Risk:** Resource Constraints (e.g., limited time for testing, lack of specific hardware/software).
  + **Contingency:** Cross-train team members; explore temporary resource augmentation; prioritize tasks based on impact; optimize test execution time; leverage cloud-based environments.
* **Risk:** Changes in Requirements During Testing, causing rework.
  + **Contingency:** Implement a strict change control process; conduct impact analysis for every change request; communicate changes effectively to the test team; ensure flexibility in test plan to accommodate minor changes; formalize requirements at increment start.
* **Risk:** Technical Issues in Test Environment (e.g., server downtime, tool incompatibilities).
  + **Contingency:** Establish a dedicated environment support strategy; implement robust environment monitoring; maintain clear communication channels with technical support for quick issue resolution; use stable, well-maintained tools.
* **Risk:** Critical Defects Discovered Late in Cycle, increasing cost of fix.
  + **Contingency:** Emphasize early testing ("shift-left"); conduct regular reviews; implement thorough unit and integration testing; encourage static analysis.
* **Risk:** Critical Defects Discovered Late in Cycle
  + **Contingency:** Emphasize early testing (shift-left); conduct regular reviews; implement thorough unit and integration testing.

## 3.9 Test Level

The testing approach for UniKL TutorFind will involve multiple levels, including component, integration, system, and acceptance testing, to ensure comprehensive validation of the application. Within these levels, specific testing types will be applied:

### 3.9.1 Black Box Testing

Black Box Testing will be extensively used for the UniKL TutorFind platform. This technique focuses on the functionality of the application without regard for its internal structure or code. Testers will interact with the system through its user interface, providing inputs and observing outputs to verify that the software behaves according to the specified requirements. This approach is ideal for verifying all the user-facing features such as Login and Registration, Profile Creation, Matching System, Search and Filters, In-App Chat, and Rate and Review. The primary goal is to ensure that the application functions correctly from an end-user perspective.

### 3.9.2 Risk Based Testing

Risk-Based Testing will be a core part of the test strategy for UniKL TutorFind. This approach prioritizes testing efforts based on the potential risks associated with different functionalities. Features with higher business impact, complexity, or a history of defects will receive more rigorous testing. The STP incorporates "Risk Number" (e.g., 1 (Lowest) to 4 (Critical)) in its test cases to indicate the criticality of the feature being tested. This ensures that critical functionalities and potential failure points are identified early and thoroughly validated, maximizing the effectiveness of the testing process within project constraints.

#### 3.9.2.1 Risk Analysis.

This section presents a detailed risk analysis for the UniKL TutorFind platform. The analysis adopts a quantitative approach, assessing the likelihood of a defect leading to a failure and the potential impact of such a failure. This approach aligns with Risk-Based Testing principles, ensuring that critical functionalities receive prioritized testing attention.

Formula used in calculation:

**Total Risk (R=L×I)**

*Likelihood (L): Scaled from 1 (negligible) to 9 (very high)*

*Impact (I): Non-linear monetary scale, reflecting potential harm or disruption (e.g., 1, 10, 50, 100, 500, 1000)*

**Risk Classification for Prioritization:**

* R<3000: Priority 1 (Lowest)
* R≥3000: Priority 2 (Low)
* R≥4000: Priority 3 (Medium)
* R≥8000: Priority 4 (High)
* R≥9000: Priority 5 (Highest)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Feature (Test Case Name)** | **Likelihood**  **(L) (1-9)** | **Impact (I)** | **Total Risk (R=L×I)** | **Priority Number (1-5)** |
| **System Feature 1: Login and Registration (LR)** |  |  |  |  |
| Functionality Test Case for Registration (TC\_001.1) | 1 | 500 | 500 | 1 |
| Invalid Email Format Test Case for Registration (TC\_001.2) | 6 | 500 | 3000 | 2 |
| Duplicate Email Registration Test Case (TC\_001.3) | 8 | 500 | 4000 | 3 |
| System Error Handling during Login (TC\_002.4) | 9 | 1000 | 9000 | 4 |
| Functionality Test Case for Login (TC\_002.1) | 1 | 500 | 500 | 1 |
| Incorrect Credentials Login Test Case (TC\_002.2) | 6 | 500 | 3000 | 2 |
| Unverified Account Login Test Case (TC\_002.3) | 8 | 500 | 4000 | 3 |
| **System Feature 2: Profile Creation (PC)** |  |  |  |  |
| Profile Creation Functionality Test Case (TC\_003.1) | 1 | 500 | 500 | 1 |
| Incomplete Profile Fields Test Case (TC\_003.2) | 6 | 500 | 3000 | 2 |
| Profile Update Functionality Test Case (TC\_003.3) | 8 | 500 | 4000 | 3 |
| System Error Handling during Profile Creation (TC\_003.4) | 9 | 1000 | 9000 | 4 |
| **System Feature 3: Profile Matching (PM)** |  |  |  |  |
| Functionality Test Case for Profile Matching (TC\_004.1) | 1 | 500 | 500 | 1 |
| No Matching Results Test Case (TC\_004.2) | 6 | 500 | 3000 | 2 |
| No Availability Matching Test Case (TC\_004.3) | 8 | 500 | 4000 | 3 |
| System Error During Matching Test Case (TC\_004.4) | 9 | 1000 | 9000 | 4 |
| **System Feature 4: Search and Filters (SF)** |  |  |  |  |
| Search and Filters Functionality Test (TC\_005.1) | 1 | 500 | 500 | 1 |
| No Search Results Test Case (TC\_005.2) | 6 | 500 | 3000 | 2 |
| No Results After Filter Application Test Case (TC\_005.3) | 8 | 500 | 4000 | 3 |
| System Error During Search Test Case (TC\_005.4) | 9 | 1000 | 9000 | 4 |
| Sorting Search Results Test Case (TC\_005.5) | 6 | 500 | 3000 | 2 |
| **System Feature 5: In-App Chatting (IC)** |  |  |  |  |
| In-App Chatting Functionality Test (TC\_006.1) | 1 | 500 | 500 | 1 |
| No Internet Connection During Messaging Test Case (TC\_006.2) | 8 | 500 | 4000 | 3 |
| Chat Window Timeout Test Case (TC\_006.3) | 6 | 500 | 3000 | 2 |
| System Error During Message Sending Test Case (TC\_006.4) | 9 | 1000 | 9000 | 4 |
| View Chat History Test Case (TC\_006.5) | 6 | 500 | 3000 | 2 |
| **System Feature 6: Rate and Review (RR)** |  |  |  |  |
| Rate and Review Functionality Test (TC\_007.1) | 1 | 500 | 500 | 1 |
| No Written Feedback Submission Test Case (TC\_007.2) | 6 | 500 | 3000 | 2 |
| Edit/Delete Review Test Case (TC\_007.3) | 8 | 500 | 4000 | 3 |
| System Error During Review Submission Test Case (TC\_007.4) | 9 | 1000 | 9000 | 4 |

Table 1: Risk Classification for Prioritisation

# 4.0 Test Cases (with Use Cases)

4.1 UniKL TutorFind – Overall Application

4.1.1 Use Case Diagram

**A diagram of a diagram

AI-generated content may be incorrect.**

Figure Overall Application Use Case Diagram

4.2 System Feature 1: Login and Registration (LR)

4.2.1 Use Case Diagrams

**A diagram of a flowchart

AI-generated content may be incorrect.**

Figure Overall Feature 1 Use Case Diagram

A diagram of a user registration

AI-generated content may be incorrect.

Figure User Registration Use Case Diagram

**A diagram of a user interface

AI-generated content may be incorrect.**

Figure User Login Use Case Diagram

4.2.2 Use Case Statements

UC\_001

|  |  |
| --- | --- |
| **Use Case ID** | UC\_001 |
| **Use Case Name** | Login and Registration |
| **Description** | |  | | --- | |  |  |  | | --- | | Allows users to register a new account or log into the system using their **@s.unikl.edu.my** email for authentication. | |
| **Primary Actor** | User |
| **Include Use Cases** | Email Format Validation, Email Duplication Check, Verification Email |
| **Scenario: Main Flow** | |
| **Step** | **Action** |
| **Pre-condition** | User does not have an existing account and is on the registration page. |
| **1.1** | User clicks the "Sign Up" button. |
| **1.2** | The system displays a registration form requesting email, name, and password. |
| **1.3** | User enters email, name, and password in the registration form. |
| **1.4** | The system validates that the email follows the @unikl.edu.my format. |
| **1.5** | The system checks for duplicate entries to prevent multiple accounts with the same email. |
| **1.6** | The system sends a verification email to the provided @unikl.edu.my email address. |
| **1.7** | User clicks the verification link in the email to confirm registration. |
| **1.8** | The system activates the account and allows the user to log in with the provided email and password. |
| **Post-condition** | User is successfully registered, verified, and can now log in to the platform. |
| **Scenario: Alternate Flow 1 - Invalid Email Format** | |
| **Pre-condition** | User is on the registration page and enters an invalid email format. |
| **2.1** | User enters an email address in an incorrect format (e.g., [alyssa@gmail.com](mailto:alyssa@gmail.com)). |
| **2.2** | The system checks the email format. |
| **2.3** | The system displays an error message indicating the email must be in the @unikl.edu.my format. |
| **2.4** | User is prompted to re-enter the email with the correct format. |
| **Post-condition** | Registration is not completed until the user enters a valid @unikl.edu.my email. |
| **Scenario: Alternate Flow 2 - Duplicate Email** | |
| **Pre-condition** | User is on the registration page and enters an email that is already registered. |
| **2.1** | User enters a duplicate email (e.g., [alyssa@unikl.edu.my](mailto:alyssa@unikl.edu.my)). |
| **2.2** | The system checks for duplicates in the email database. |
| **2.3** | The system displays an error message indicating that the email is already in use. |
| **2.4** | User is prompted to log in or use another email address. |
| **Post-condition** | User is not registered, and the system prompts the user to log in or enter a different email. |
| **Scenario: Robust Flow** | |
| **Pre-condition** | User is on the registration page and enters data, but the system experiences an issue. |
| **3.1** | User enters a valid email address (e.g., [alyssa@unikl.edu.my](mailto:alyssa@unikl.edu.my)) and clicks submit. |
| **3.2** | The system attempts to send the verification email but encounters a server issue (e.g., Supabase service down). |
| **3.3** | The system displays an error message like "There was an issue sending the verification email. Please try again later." |
| **3.4** | User is prompted to try again later or contact support. |
| **Post-condition** | The registration process does not complete, and the user is informed of the system issue. |

Table UC\_001 Use Case Statement

Use Case ID: UC002

|  |  |
| --- | --- |
| **Use Case ID** | UC\_002 |
| **Use Case Name** | Log in User |
| **Description** | |  | | --- | |  |  |  | | --- | | Allows UniKL Users to log in using their registered @s.unikl.edu.my email and password. | |
| **Primary Actor** | User |
| **Include Use Cases** | Supabase Authentication |
| **Scenario: Main Flow** | |
| **Step** | **Action** |
| **Pre-condition** | User has registered and verified their account. |
| **1.1** | User enters their email and password on the login screen. |
| **1.2** | User clicks the "Log In" button. |
| **1.3** | The system authenticates the email and password via Supabase Authentication. |
| **1.4** | If the credentials are correct, the system grants access to the platform. |
| **Post-condition** | User is logged in and has access to the platform. |
| **Scenario: Alternate Flow 1 - Incorrect Credentials** | |
| **Pre-condition** | User enters an incorrect email or password. |
| **2.1** | User enters incorrect email or password. |
| **2.2** | User clicks "Log In". |
| **2.3** | The system displays an error message indicating incorrect credentials. |
| **Post-condition** | User is not logged in and is prompted to correct the credentials. |
| **Scenario: Alternate Flow 2 – Unverified Account** | |
| **Pre-condition** | User has registered but has not verified their email. |
| **2.1** | User attempts to log in using their registered email and password. |
| **2.2** | The system checks if the email is verified. |
| **2.3** | The system denies access and prompts the user to verify their email. |
| **Post-condition** | User cannot log in until they verify their email. |
| **Scenario: Robust Flow** | |
| **Pre-condition** | User attempts to log in but experiences a system error (e.g., Supabase authentication service failure). |
| **3.1** | User enters valid email and password. |
| **3.2** | The system fails to authenticate the user due to a service issue. |
| **3.3** | The system displays an error message like "Service unavailable. Please try again later." |
| **3.4** | User is informed of the error and asked to retry. |
| **Post-condition** | User is not logged in and is informed of the system error. |

Table UC002 Use Case Statement

### 4.2.3 Test Cases

Test Case for UC\_001 - Register New User (TC\_001.1)

Scenario: Main Flow

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tested By** | Placeholder Name | | **Test Level** | | User Acceptance Testing | |
| **Test Type** | Black Box Testing – Functional Testing | | | | | |
| **Test Design Technique** | Use Case Testing | | | | | |
| **Risk Number** | 1 (Lowest) | | | | | |
| **Test Case Number** | TC\_001.1 | | | | | |
| **Test Case Name** | Functionality Test Case for Registration | | | | | |
| **Requirement Traceability** | LR-1 (Login and Registration Feature - Registration) | | | | | |
| **Test Case Description** | To verify the functionality of the Registration module with valid inputs. | | | | | |
| **Item to be tested** | | | | | | |
| 1 | Registration form fields, navigation to login page, and success message. | | | | | |
| **Pre-condition** | User is not registered and is on the Registration page. | | | | | |
| **Test Procedure Specification** | | | | | | |
| **Input** | **Test Data** | **Expected Output / Result** | | **Actual Result** | | **Pass / Fail** |
| All registration fields filled with valid details. | Name: Alyssa Husna,  Email: alyssa.jamarizan@s.unikl.edu.my,  Password: Zxcvb12345,  Gender: Female,  Date of Birth: 14/11/2003 | The system verifies the user input, displays a success message, and redirects to the login page. | |  | |  |
| **Post-condition** | If the registration is successful, the system stores the data and redirects the user to the login page. | | | | | |
|  |  | | | | | |
|  |  | | | | | |

Table TC\_001.1 Test Case

Test Case for UC\_001 - Register New User (TC\_001.2)

Scenario: Alternate Flow 1 - Invalid Email Format

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Placeholder Name | | **Test Level** | | User Acceptance Testing | |
| **Test Type** | Black Box Testing – Functional Testing | | | | | |
| **Test Design Technique** | Use Case Testing | | | | | |
| **Risk Number** | 2 (Medium) | | | | | |
| **Test Case Number** | TC\_001.2 | | | | | |
| **Test Case Name** | Invalid Email Format Test Case for Registration | | | | | |
| **Requirement Traceability** | LR-1-1 (Login and Registration Feature - Email Validation) | | | | | |
| **Test Case Description** | To verify that the system rejects registration if the email is not in the correct format. | | | | | |
| **Item to be tested** | | | | | | |
| 1 | Email format validation. | | | | | |
| **Pre-condition** | User is on the registration page. | | | | | |
| **Test Procedure Specification** | | | | | | |
| **Input** | **Test Data** | **Expected Output / Result** | | **Actual Result** | | **Pass / Fail** |
| Email is not in @s.unikl.edu.my format. | Name: Alyssa Husna,  Email: alyssa.jamarizan@gmail.com,  Password: Zxcvb12345,  Gender: Female,  Date of Birth: 14/11/2003 | The system shows an error message indicating the email must be in the @s.unikl.edu.my format. | |  | |  |
| **Post-condition** | Registration is not completed, and the user is prompted to enter a valid email. | | | | | |
|  |  | | | | | |
|  |  | | | | | |

Table TC\_001.2 Test Case

Test Case for UC\_001 - Register New User (TC\_001.3)

Scenario: Alternate Flow 2 - Duplicate Email

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tested By** | Placeholder Name | | **Test Level** | | User Acceptance Testing | |
| **Test Type** | Black Box Testing – Functional Testing | | | | | |
| **Test Design Technique** | Use Case Testing | | | | | |
| **Risk Number** | 3 (High) | | | | | |
| **Test Case Number** | TC\_001.3 | | | | | |
| **Test Case Name** | Duplicate Email Registration Test Case | | | | | |
| **Requirement Traceability** | LR-1-2 (Login and Registration Feature - Duplicate Email Check) | | | | | |
| **Test Case Description** | To verify that the system prevents registration if the email is already registered. | | | | | |
| **Item to be tested** | | | | | | |
| 1 | Email duplication check | | | | | |
| **Pre-condition** | User is on the registration page. | | | | | |
| **Test Procedure Specification** | | | | | | |
| **Input** | **Test Data** | **Expected Output / Result** | | **Actual Result** | | **Pass / Fail** |
| The email is already registered in the system. | Name: Alyssa Husna,  Email: alyssa.jamarizan@s.unikl.edu.my,  Password: Zxcvb12345,  Gender: Female,  Date of Birth: 14/11/2003 | The system shows an error message indicating that the email is already in use. | |  | |  |
| **Post-condition** | Registration is not completed, and the user is prompted to log in or enter a different email. | | | | | |
|  |  | | | | | |
|  |  | | | | | |

Table TC\_001.3 Test Case

Test Case for UC\_002 - Log in User (TC\_002.1)

Scenario: Main Flow

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tested By** | Placeholder Name | | **Test Level** | | User Acceptance Testing | |
| **Test Type** | Black Box Testing – Functional Testing | | | | | |
| **Test Design Technique** | Use Case Testing | | | | | |
| **Risk Number** | 1 (Lowest) | | | | | |
| **Test Case Number** | TC\_002.1 | | | | | |
| **Test Case Name** | Functionality Test Case for Login | | | | | |
| **Requirement Traceability** | LR-4 (Login and Registration Feature - Login) | | | | | |
| **Test Case Description** | To verify that a user can successfully log in with valid credentials. | | | | | |
| **Item to be tested** | | | | | | |
| 1 | Email format validation | | | | | |
| **Pre-condition** | User is on the registration page. | | | | | |
| **Test Procedure Specification** | | | | | | |
| **Input** | **Test Data** | **Expected Output / Result** | | **Actual Result** | | **Pass / Fail** |
| Correct login credentials (valid email and password). | Email: alyssa.jamarizan@s.unikl.edu.my,  Password: Zxcvb12345 | The system verifies the credentials and grants access to the platform. | |  | |  |
| **Post-condition** | User is logged in and has access to the platform. | | | | | |
|  |  | | | | | |
|  |  | | | | | |

Table TC\_002.1 Test Case

Test Case for UC\_002 - Log in User (TC\_002.2)

Scenario: Alternate Flow 1 - Incorrect Credentials

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tested By** | Placeholder Name | | **Test Level** | | User Acceptance Testing | |
| **Test Type** | Black Box Testing – Functional Testing | | | | | |
| **Test Design Technique** | Use Case Testing | | | | | |
| **Risk Number** | 2 (Medium) | | | | | |
| **Test Case Number** | TC\_002.2 | | | | | |
| **Test Case Name** | Incorrect Credentials Login Test Case | | | | | |
| **Requirement Traceability** | LR-5-1 (Login and Registration Feature - Error Message for Invalid Login) | | | | | |
| **Test Case Description** | To verify that the system rejects login attempts with incorrect credentials. | | | | | |
| **Item to be tested** | | | | | | |
| 1 | Email and password mismatch. | | | | | |
| **Pre-condition** | User is registered but enters incorrect login details. | | | | | |
| **Test Procedure Specification** | | | | | | |
| **Input** | **Test Data** | **Expected Output / Result** | | **Actual Result** | | **Pass / Fail** |
| Incorrect password. | Email: alyssa.jamarizan@s.unikl.edu.my,  Password: WrongPassword123 | The system displays an error message: "Incorrect email or password." | |  | |  |
| **Post-condition** | User is not logged in, and they are prompted to correct the credentials. | | | | | |
|  |  | | | | | |
|  |  | | | | | |

Table TC\_002.2 Test Case

Test Case for UC\_002 - Log in User (TC\_002.3)

Scenario: Alternate Flow 2 - Unverified Account

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tested By** | Placeholder Name | | **Test Level** | | User Acceptance Testing | |
| **Test Type** | Black Box Testing – Functional Testing | | | | | |
| **Test Design Technique** | Use Case Testing | | | | | |
| **Risk Number** | 3 (High) | | | | | |
| **Test Case Number** | TC\_002.3 | | | | | |
| **Test Case Name** | Unverified Account Login Test Case | | | | | |
| **Requirement Traceability** | LR-4-1 (Login and Registration Feature - Account Authentication with Hashing) | | | | | |
| **Test Case Description** | To verify that unverified users cannot log in. | | | | | |
| **Item to be tested** | | | | | | |
| 1 | Email verification check. | | | | | |
| **Pre-condition** | User is registered but enters incorrect login details. | | | | | |
| **Test Procedure Specification** | | | | | | |
| **Input** | **Test Data** | **Expected Output / Result** | | **Actual Result** | | **Pass / Fail** |
| Correct login credentials, but email is unverified. | Email: alyssa.jamarizan@s.unikl.edu.my,  Password: Zxcvb12345 | The system denies access and prompts the user to verify their email. | |  | |  |
| **Post-condition** | User cannot log in until email verification is completed. | | | | | |
|  |  | | | | | |
|  |  | | | | | |

Table TC\_002.3 Test Case

Test Case for UC\_002 - Log in User (TC\_002.4)

Scenario: Robust Flow

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tested By** | Placeholder Name | | **Test Level** | | User Acceptance Testing | |
| **Test Type** | Black Box Testing – Functional Testing | | | | | |
| **Test Design Technique** | Use Case Testing | | | | | |
| **Risk Number** | 4 (Critical) | | | | | |
| **Test Case Number** | TC\_002.4 | | | | | |
| **Test Case Name** | System Error Handling during Login | | | | | |
| **Requirement Traceability** | LR-4 (Login and Registration Feature - Login) | | | | | |
| **Test Case Description** | To verify that the system handles errors during login, such as Supabase service issues. | | | | | |
| **Item to be tested** | | | | | | |
| 1 | Error handling for login services. | | | | | |
| **Pre-condition** | User attempts to log in but experiences a system error (e.g., Supabase authentication service failure). | | | | | |
| **Test Procedure Specification** | | | | | | |
| **Input** | **Test Data** | **Expected Output / Result** | | **Actual Result** | | **Pass / Fail** |
| A system service issue prevents authentication (e.g., Supabase authentication service failure). | Email: alyssa.jamarizan@s.unikl.edu.my,  Password: Zxcvb12345 | The system displays an error message like "Service unavailable. Please try again later." | |  | |  |
| **Post-condition** | User is not logged in and is informed of the system error. | | | | | |
|  |  | | | | | |
|  |  | | | | | |

Table TC\_002.4 Test Case

4.3 System Feature 2: Profile Creation (PC)

4.3.1 Use Case Diagram

**A diagram of a process

AI-generated content may be incorrect.**

Figure Overall Feature 2 Use Case Diagram

**A diagram of a profile creation system

AI-generated content may be incorrect.**

Figure Profile Creation Use Case Diagram

**A diagram of a profile updating system

AI-generated content may be incorrect.**

Figure Profile Update Use Case Diagram

4.3.2 Use Case Statements

Use Case UC\_003

|  |  |
| --- | --- |
| **Use case ID** | UC\_003 |
| **Use Case Name** | Profile Creation |
| **Description** | Allows users to create, update, and manage their profiles, including adding skills, learning preferences, and availability. This feature plays a key role in enabling the Profile Matching functionality of the platform. |
| **Primary Actor** | User |
| **Include Use Cases** | Profile Validation, Profile Update |
| **Scenario: Main Flow** | |
| **Step** | **Action** |
| **Pre-condition** | User has completed the registration process and is on the profile creation page. |
| **1.1** | User clicks the "Create Profile" button after registration. |
| **1.2** | The system displays the profile creation form. |
| **1.3** | User enters name, skills they can teach, skills they want to learn, and availability. |
| **1.4** | The system validates that all fields are completed before allowing profile submission. |
| **1.5** | User clicks "Save Profile". |
| **1.6** | The system saves the profile data in the database and provides a confirmation message. |
| **Post-condition** | User's profile is created, and the system redirects them to their profile page. |
| **Scenario: Alternate Flow 1 – Incomplete Profile Fields** | |
| **Pre-condition** | User is on the profile creation page and leaves required fields incomplete. |
| **2.1** | User clicks "Save Profile" without completing all fields. |
| **2.2** | The system detects incomplete fields. |
| **2.3** | The system prompts the user to complete all required fields. |
| **Post-condition** | User cannot submit the profile until all required fields are completed. |
| **Scenario: Alternate Flow 2 - Profile Update** | |
| **Pre-condition** | User has already created a profile and wants to update it. |
| **2.1** | User clicks "Edit Profile". |
| **2.2** | The system displays the current profile details. |
| **2.3** | User updates the profile with new skills, availability, or preferences. |
| **2.4** | User clicks "Save Profile". |
| **2.5** | The system updates the profile in the database and displays a confirmation message. |
| **Post-condition** | User's profile is updated successfully. |
| **Scenario: Robust Flow** | |
| **Pre-condition** | User has entered profile data, but the system encounters an issue saving it. |
| **3.1** | User clicks "Save Profile" after entering their profile data. |
| **3.2** | The system attempts to save the profile but encounters a database error. |
| **3.3** | The system displays an error message: "There was an issue saving your profile. Please try again later." |
| **Post-condition** | User is prompted to retry or contact support, and the profile is not saved. |

Table UC\_003 Use Case Statement

Use Case UC\_004

|  |  |
| --- | --- |
| **Use case ID** | UC\_004 |
| **Use Case Name** | Update Profile |
| **Description** | Allows users to update their profiles by adding new skills, updating their availability, and editing preferences. |
| **Primary Actor** | User |
| **Include Use Cases** | Profile Validation, Profile Update |
| **Scenario: Main Flow** | |
| **Step** | **Action** |
| **Pre-condition** | User has already created a profile and is on the profile page. |
| **1.1** | User clicks the "Edit Profile" button. |
| **1.2** | The system displays the profile editing form. |
| **1.3** | User updates the profile details (e.g., adding new skills, updating availability). |
| **1.4** | User clicks "Save Profile". |
| **1.5** | The system saves the updated profile in the database and displays a confirmation message. |
| **Post-condition** | User's profile is updated successfully and saved in the system. |
| **Scenario: Alternate Flow 1 – Invalid Input Format** | |
| **Pre-condition** | User enters invalid input (e.g., incorrect date format or empty field) while editing their profile. |
| **2.1** | User attempts to save the profile with invalid data. |
| **2.2** | The system detects the invalid input. |
| **2.3** | The system displays an error message indicating the invalid field and prompts the user to correct it. |
| **Post-condition** | User cannot save the profile until the invalid input is corrected. |
| **Scenario: Robust Flow - Database Error during Profile Update** | |
| **Pre-condition** | User attempts to update their profile, but the system encounters a database issue. |
| **3.1** | User clicks "Save Profile" after updating their details. |
| **3.2** | The system attempts to save the updated profile data but encounters a database issue. |
| **3.3** | The system displays an error message: "There was an issue saving your profile. Please try again later." |
| **Post-condition** | User is prompted to try again or contact support, and the profile is not updated. |

Table UC\_004 Use Case Statement

4.3.3 Test Cases

Test Case for UC\_003 - Profile Creation (TC\_003.1)

Scenario: Main Flow

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tested By** | Placeholder Name | | **Test Level** | | User Acceptance Testing | |
| **Test Type** | Black Box Testing – Functional Testing | | | | | |
| **Test Design Technique** | Use Case Testing | | | | | |
| **Risk Number** | 1 (Lowest) | | | | | |
| **Test Case Number** | TC\_003.1 | | | | | |
| **Test Case Name** | Profile Creation Functionality Test Case | | | | | |
| **Requirement Traceability** | REQ\_PC-1 (Profile Creation - Profile Creation) | | | | | |
| **Test Case Description** | To verify the functionality of the Profile Creation module with valid inputs. | | | | | |
| **Item to be tested** | | | | | | |
| 1 | Profile form fields, profile creation success message, and profile saving. | | | | | |
| **Pre-condition** | User has completed the registration process and is on the profile creation page. | | | | | |
| **Test Procedure Specification** | | | | | | |
| **Input** | **Test Data** | **Expected Output / Result** | | **Actual Result** | | **Pass / Fail** |
| All required fields filled with valid information. | Name: Alyssa Husna,  Email: alyssa.jamarizan@s.unikl.edu.my,  Skills to Teach: "Python Programming",  Skills to Learn: "Machine Learning",  Availability: "Weekdays, 3 PM - 5 PM" | The system saves the profile data and provides a confirmation message. | |  | |  |
| **Post-condition** | User's profile is created successfully, and they are redirected to the profile page. | | | | | |

Table TC\_003.1 Test Case

Test Case for UC\_003 - Profile Creation (TC\_003.2)

Scenario: Alternate Flow 1 - Incomplete Profile Fields

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tested By** | Placeholder Name | | **Test Level** | | User Acceptance Testing | |
| **Test Type** | Black Box Testing – Functional Testing | | | | | |
| **Test Design Technique** | Use Case Testing | | | | | |
| **Risk Number** | 2 (Medium) | | | | | |
| **Test Case Number** | TC\_003.2 | | | | | |
| **Test Case Name** | Incomplete Profile Fields Test Case | | | | | |
| **Requirement Traceability** | REQ\_PC-1-1 (Profile Creation - Field Validation) | | | | | |
| **Test Case Description** | To verify that the system prompts the user to complete all required fields before submitting the profile. | | | | | |
| **Item to be tested** | | | | | | |
| 1 | Profile Field Validation | | | | | |
| **Pre-condition** | User is on the profile creation page and leaves required fields incomplete. | | | | | |
| **Test Procedure Specification** | | | | | | |
| **Input** | **Test Data** | **Expected Output / Result** | | **Actual Result** | | **Pass / Fail** |
| Missing the "Skills to Learn" field. | Name: Alyssa Husna,  Email: alyssa.jamarizan@s.unikl.edu.my,  Skills to Teach: "Python Programming",  Availability: "Weekdays, 3 PM - 5 PM" | The system prompts the user to fill in the "Skills to Learn" field before submitting the profile. | |  | |  |
| **Post-condition** | The user cannot submit the profile until all required fields are filled. | | | | | |

Table TC\_003.2 Test Case

Test Case for UC\_003 - Profile Creation (TC\_003.3)

Scenario: Alternate Flow 2 – Profiule Update

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tested By** | Placeholder Name | | **Test Level** | | User Acceptance Testing | |
| **Test Type** | Black Box Testing – Functional Testing | | | | | |
| **Test Design Technique** | Use Case Testing | | | | | |
| **Risk Number** | 3 (High) | | | | | |
| **Test Case Number** | TC\_003.3 | | | | | |
| **Test Case Name** | Profile Update Functionality Test Case | | | | | |
| **Requirement Traceability** | REQ\_PC-2 (Profile Creation - Profile Update) | | | | | |
| **Test Case Description** | To verify that the system allows users to update their profiles successfully. | | | | | |
| **Item to be tested** | | | | | | |
| 1 | Profile update functionality. | | | | | |
| **Pre-condition** | User has created a profile and wants to update it. | | | | | |
| **Test Procedure Specification** | | | | | | |
| **Input** | **Test Data** | **Expected Output / Result** | | **Actual Result** | | **Pass / Fail** |
| The user updates the profile with new skills (e.g., changing "Python Programming" to "Data Science"). | Name: Alyssa Husna,  Email: alyssa.jamarizan@s.unikl.edu.my,  Skills to Teach: "Python Programming",  Skills to Learn: "Machine Learning",  Availability: "Weekdays, 3 PM - 5 PM" (initial profile),  Updated Skills to Teach: "Data Science" | The system prompts the user to fill in the "Skills to Learn" field before submitting the profile. | |  | |  |
| **Post-condition** | The user's profile is updated successfully. | | | | | |

Table TC\_003.3 Test Case

Test Case for UC\_003 - Profile Creation (TC\_003.4)

Scenario: Robust Flow - System Error

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tested By** | Placeholder Name | | **Test Level** | | User Acceptance Testing | |
| **Test Type** | Black Box Testing – Functional Testing | | | | | |
| **Test Design Technique** | Use Case Testing | | | | | |
| **Risk Number** | 4 (Critical) | | | | | |
| **Test Case Number** | TC\_003.4 | | | | | |
| **Test Case Name** | System Error Handling during Profile Creation | | | | | |
| **Requirement Traceability** | REQ\_PC-4 (Profile Creation - Data Security) | | | | | |
| **Test Case Description** | To verify that the system handles errors during profile creation, such as database issues. | | | | | |
| **Item to be tested** | | | | | | |
| 1 | Error handling during profile creation. | | | | | |
| **Pre-condition** | User has entered all profile data, but the system encounters an issue saving it. | | | | | |
| **Test Procedure Specification** | | | | | | |
| **Input** | **Test Data** | **Expected Output / Result** | | **Actual Result** | | **Pass / Fail** |
| All fields filled, but the system encounters a database issue when attempting to save the profile. | Name: Alyssa Husna,  Email: alyssa.jamarizan@s.unikl.edu.my,  Skills to Teach: "Python Programming",  Skills to Learn: "Machine Learning",  Availability: "Weekdays, 3 PM - 5 PM" | The system displays an error message: "There was an issue saving your profile. Please try again later." | |  | |  |
| **Post-condition** | The user's profile is not saved, and they are prompted to try again later. | | | | | |

Table TC\_003.4 Test Case

4.4 System Feature 3: Profile Matching (PM)

4.4.1 Use Case Diagram

**A diagram of a diagram

AI-generated content may be incorrect.**

Figure Feature 3 Profile Matching Use Case Diagram

***4.4.2 Use Case Statements***

UC\_004

|  |  |
| --- | --- |
| **Use case ID** | UC\_004 |
| **Use Case Name** | Profile Matching |
| **Description** | Allows the system to match students based on complementary skills, availability, and learning preferences, ensuring that users are connected with the right peers for collaborative learning. |
| **Primary Actor** | User |
| **Include Use Cases** | Skill Matching, Availability Matching, Chat Interface |
| **Scenario: Main Flow** | |
| **Step** | **Action** |
| **Pre-condition** | The user is logged in and is on the profile matching page. |
| **1.1** | The user clicks "Find a Peer" or enters a search term based on their learning needs. |
| **1.2** | The system searches for potential matches based on complementary skills and availability. |
| **1.3** | The system displays the matched students in a list. |
| **1.4** | The user selects a match from the list. |
| **1.5** | The system notifies both students of the match and opens a chat interface for them to connect. |
| **Post-condition** | The user and matched peer can now communicate and arrange a learning session. |
| **Scenario: Alternate Flow 1 – No Matching Results** | |
| **Pre-condition** | The user clicks "Find a Peer" or enters a search term, but no matches are found. |
| **2.1** | The system fails to find any matches based on the search criteria. |
| **2.2** | The system displays a message saying "No matches found based on your search criteria." |
| **2.3** | The user is prompted to modify their search criteria or try again later. |
| **Post-condition** | The user does not find a match and is prompted to adjust their search. |
| **Scenario: Alternate Flow 2 – No Availability** | |
| **Pre-condition** | The system finds matches based on skills, but there are no matching availability slots. |
| **2.1** | The system identifies students with complementary skills but with no overlapping availability. |
| **2.2** | The system displays a message saying "No peers available based on your current availability." |
| **2.3** | The user is prompted to adjust their availability or search again later. |
| **Post-condition** | The user does not find a match due to availability constraints. |
| **Scenario: Robust Flow – System Error** | |
| **Pre-condition** | The user is on the profile matching page and enters their search criteria. |
| **3.1** | The system encounters an error while searching for matches (e.g., connectivity issue or database error). |
| **3.2** | The system displays an error message: "There was an issue while searching for peers. Please try again later." |
| **3.3** | The user is prompted to retry the search or contact support for further assistance. |
| **Post-condition** | The user is unable to find a match due to the system error, and the search is not completed. |

Table UC\_004 Use Case Statement

UC\_004.1

|  |  |
| --- | --- |
| **Use case ID** | UC\_004.1 |
| **Use Case Name** | Match based on skills. |
| **Description** | Matches students based on complementary skills (e.g., programming, language), ensuring that they can learn from and teach each other effectively. |
| **Primary Actor** | System |
| **Include Use Cases** | Skill Matching |
| **Scenario: Main Flow** | |
| **Step** | **Action** |
| **Pre-condition** | The user has entered their learning needs and is looking for a peer with complementary skills. |
| **1.1** | The system searches for peers based on the skills entered by The user (e.g., programming or language). |
| **1.2** | The system filters match based on skill compatibility, ensuring students with complementary skills are matched. |
| **1.3** | The system displays the matched peers who have the necessary skills. |
| **Post-condition** | The user receives a list of peers with matching skills, allowing them to connect for learning. |

Table UC\_004.1 Use Case Statement

UC\_004.2

|  |  |
| --- | --- |
| **Use case ID** | UC\_004.2 |
| **Use Case Name** | Match based on availability. |
| **Description** | Matches students based on their availability for learning sessions, ensuring that both students are free to participate at the same time. |
| **Primary Actor** | System |
| **Include Use Cases** | Availability Matching |
| **Scenario: Main Flow** | |
| **Step** | **Action** |
| **Pre-condition** | The user has entered their availability preferences (e.g., times they are free for learning). |
| **1.1** | The system searches for peers based on the entered availability times. |
| **1.2** | The system filters matches based on availability, ensuring students are matched with peers available at the same times. |
| **1.3** | The system displays the matched peers based on available time slots. |
| **Post-condition** | The user receives a list of peers with compatible availability for collaborative learning. |

Table UC\_004.2 Use Case Statement

UC\_004.3

|  |  |
| --- | --- |
| **Use case ID** | UC\_004.3 |
| **Use Case Name** | Notification on Match |
| **Description** | Sends a notification to user when peer match is found. |
| **Primary Actor** | System |
| **Include Use Cases** | Notification System |
| **Scenario: Main Flow** | |
| **Step** | **Action** |
| **Pre-condition** | The user has found a match based on skills and availability. |
| **1.1** | The system sends a notification to both students notifying them of the match. |
| **1.2** | The system includes a prompt for The users to open the chat interface and connect. |
| **Post-condition** | The users are notified of their match and can begin communication to arrange a learning session. |

Table UC\_004.3 Use Case Statement

4.4.3 Test Cases

Test Case for UC\_004 - Profile Matching (TC\_004.1)

Scenario: Main Flow

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tested By** | Placeholder Name | | **Test Level** | | User Acceptance Testing | |
| **Test Type** | Black Box Testing – Functional Testing | | | | | |
| **Test Design Technique** | Use Case Testing | | | | | |
| **Risk Number** | 1 (Lowest) | | | | | |
| **Test Case Number** | TC\_004.1 | | | | | |
| **Test Case Name** | Functionality Test Case for Profile Matching | | | | | |
| **Requirement Traceability** | PM-1 (Profile Matching Feature - Skill and Availability Matching) | | | | | |
| **Test Case Description** | To verify the functionality of the Profile Matching module by matching students based on complementary skills and availability. | | | | | |
| **Item to be tested** | | | | | | |
| 1 | Matching process, user notification, and chat interface. | | | | | |
| **Pre-condition** | User has completed their profile and is on the profile matching page. | | | | | |
| **Test Procedure Specification** | | | | | | |
| **Input** | **Test Data** | **Expected Output / Result** | | **Actual Result** | | **Pass / Fail** |
| User enters a search term for the skills and availability they need. | Skills to Learn: "programming"Availability: "available at 4 PM" | The system matches students based on the entered skills and availability and displays the list of matched students. The system sends a notification to both the student and the matched peer. The chat interface is opened for further communication. | |  | |  |
| **Post-condition** | The student and matched peer can now communicate and arrange a learning session. | | | | | |

Table TC\_004.1 Test Case

Test Case for UC\_004 - Profile Matching (TC\_004.2)  
Scenario: Alternate Flow 1 - No Matching Results

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tested By** | Placeholder Name | | **Test Level** | | User Acceptance Testing | |
| **Test Type** | Black Box Testing – Functional Testing | | | | | |
| **Test Design Technique** | Use Case Testing | | | | | |
| **Risk Number** | 2 (Medium) | | | | | |
| **Test Case Number** | TC\_004.2 | | | | | |
| **Test Case Name** | No Matching Results Test Case | | | | | |
| **Requirement Traceability** | REQ\_PM-2 (Profile Matching Feature - No Matches Found) | | | | | |
| **Test Case Description** | To verify that the system handles the case where no matches are found based on the search criteria. | | | | | |
| **Item to be tested** | | | | | | |
| 1 | Search criteria handling, system response to no matches. | | | | | |
| **Pre-condition** |  | | | | | |
| **Test Procedure Specification** | | | | | | |
| **Input** | **Test Data** | **Expected Output / Result** | | **Actual Result** | | **Pass / Fail** |
| User enters search criteria, but there are no peers that match the skills or availability. | Skills to Learn: "cooking" Availability: "available at 4 PM" | The system displays a message saying, "No matches found based on your search criteria." The student is prompted to modify the search criteria. | |  | |  |
| **Post-condition** | The student does not find a match and is prompted to adjust their search. | | | | | |

Table TC\_004.2 Test Case

Test Case for UC\_004 - Profile Matching (TC\_004.3)

Scenario: Alternate Flow 2 - No Availability

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tested By** | Placeholder Name | | **Test Level** | | User Acceptance Testing | |
| **Test Type** | Black Box Testing – Functional Testing | | | | | |
| **Test Design Technique** | Use Case Testing | | | | | |
| **Risk Number** | 3 (High) | | | | | |
| **Test Case Number** | TC\_004.3 | | | | | |
| **Test Case Name** | No Availability Matching Test Case | | | | | |
| **Requirement Traceability** | REQ\_PM-3 (Profile Matching Feature - Availability Matching) | | | | | |
| **Test Case Description** | To verify that the system properly handles the case when matches are found, but there is no overlapping availability. | | | | | |
| **Item to be tested** | | | | | | |
| 1 | Availability matching, system response to no overlapping availability. | | | | | |
| **Pre-condition** |  | | | | | |
| **Test Procedure Specification** | | | | | | |
| **Input** | **Test Data** | **Expected Output / Result** | | **Actual Result** | | **Pass / Fail** |
| User enters a search term with complementary skills, but no peers have availability that matches the user's free times. | Skills to Learn: "programming" Availability: "available at 3 AM" | The system displays a message: "No peers available based on your current availability." The student is prompted to adjust their availability or search criteria. | |  | |  |
| **Post-condition** | The student is prompted to adjust their availability or search again. | | | | | |

Table TC\_004.3 Test Case

Test Case for UC\_004 - Profile Matching (TC\_004.4)

Scenario: Robust Flow - System Error

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tested By** | Placeholder Name | | **Test Level** | | User Acceptance Testing | |
| **Test Type** | Black Box Testing – Functional Testing | | | | | |
| **Test Design Technique** | Use Case Testing | | | | | |
| **Risk Number** | 4 (Critical) | | | | | |
| **Test Case Number** | TC\_004.4 | | | | | |
| **Test Case Name** | System Error During Matching Test Case | | | | | |
| **Requirement Traceability** | REQ\_PM-4 (Profile Matching Feature - System Error Handling) | | | | | |
| **Test Case Description** | To verify that the system correctly handles errors during the matching process. | | | | | |
| **Item to be tested** | | | | | | |
| 1 | System error handling, notification system during errors. | | | | | |
| **Pre-condition** |  | | | | | |
| **Test Procedure Specification** | | | | | | |
| **Input** | **Test Data** | **Expected Output / Result** | | **Actual Result** | | **Pass / Fail** |
| User enters search criteria, but the system encounters an error (e.g., database connectivity issue). | Skills to Learn: "programming" Availability: "available at 3 AM" | The system displays an error message: "There was an issue while searching for peers. Please try again later." The user is prompted to retry or contact support. | |  | |  |
| **Post-condition** | The student is unable to complete the match search due to the system error, and the process is halted. | | | | | |

Table TC\_004.4 Test Case

4.5 System Feature 4: Search and Filters (SF)

4.5.1 Use Case Diagram

**A diagram of a diagram

AI-generated content may be incorrect.**

Figure Feature 4 Search and Filters Use Case Diagram

4.5.2 Use Case Statements

UC\_005

|  |  |
| --- | --- |
| **Use case ID** | UC\_005 |
| **Use Case Name** | Search and Filter Peers |
| **Description** | Allows students to search for peers based on specific criteria such as skills, availability, and location, and filter the results to find the most relevant matches for learning sessions. |
| **Primary Actor** | User |
| **Include Use Cases** | Search by Skills, Apply Filters, Display Search Results |
| **Scenario: Main Flow** | |
| **Step** | **Action** |
| **Pre-condition** | The user is logged in and is on the peer search page. |
| **1.1** | User enters a search keyword in the search bar (e.g., "Python"). |
| **1.2** | The system displays a list of users who match the specified skill or keyword. |
| **1.3** | User applies filters such as skill category, availability, and location. |
| **1.4** | The system updates the search results by filtering users based on the applied criteria. |
| **1.5** | The system displays a filtered list of peers based on the selected criteria. |
| **1.6** | Users can scroll through the results and select the most suitable peer for learning. |
| **1.7** | User selects a peer from the search results. |
| **1.8** | The system displays the selected peer’s profile with further interaction options (e.g., message, connect). |
| **Post-condition** | The user has successfully selected a peer and can interact with them to arrange a learning session. |
| **Scenario: Alternate Flow 1 – No Search Results** | |
| **Pre-condition** | The user searches for a skill or keyword that has no matching peers. |
| **2.1** | User enters a search keyword that returns no results (e.g., rare skill or no matching profiles). |
| **2.2** | The system displays a message indicating "No matches found." |
| **2.3** | User is prompted to modify the search term or apply different filters. |
| **Post-condition** | The user is informed that no peers match the search criteria. |
| **Scenario: Alternate Flow 2 – No Matching Results After Filter Applied** | |
| **Pre-condition** | The user has applied filters, but no matching results are found based on the selected criteria. |
| **2.1** | User applies filters such as availability and location. |
| **2.2** | The system finds no results based on the selected filters (e.g., no one is available at the selected times or location). |
| **2.3** | The system displays a message indicating "No matches found based on your selected filters." |
| **2.4** | User is prompted to modify the filters or search criteria. |
| **Post-condition** | The user is informed that no matches are found with the selected filters. |
| **Scenario: Robust Flow – System Error** | |
| **Pre-condition** | The user enters a search term, but the system encounters an error while retrieving the results. |
| **3.1** | User enters a search keyword (e.g., "Python") or applies filters. |
| **3.2** | The system attempts to process the search, but encounters a system or database error. |
| **3.3** | The system displays an error message: "There was an issue retrieving search results. Please try again later." |
| **3.4** | User is prompted to try the search again later or contact support. |
| **Post-condition** | The search process fails, and the user is informed of the issue. |

Table UC\_005 Use Case Statement

UC\_005.1

|  |  |
| --- | --- |
| **Use case ID** | UC\_005.1 |
| **Use Case Name** | Search by Skills |
| **Description** | Allows The user to search for peers based on a specific skill or keyword (e.g., "Python"). |
| **Primary Actor** | User |
| **Include Use Cases** | Match keywords to skill. |
| **Scenario: Main Flow** | |
| **Step** | **Action** |
| **Pre-condition** | The user is on the search page and has entered a search keyword in the search bar. |
| **1.1** | User enters a search keyword related to a skill (e.g., "Python"). |
| **1.2** | The system matches the keyword with the skills listed in peer profiles. |
| **1.3** | The system displays a list of peers who have the entered skill in their profiles. |
| **Post-condition** | The user sees a list of peers who match the search keyword. |

Table UC\_005.1 Use Case Statement

UC\_005.2

|  |  |
| --- | --- |
| **Use case ID** | UC\_005.2 |
| **Use Case Name** | Apply Filters |
| **Description** | Allows The user to apply filters based on skill category, availability, and location to narrow search results. |
| **Primary Actor** | Student |
| **Include Use Cases** | Filter by Availability, Filter by Location |
| **Scenario: Main Flow** | |
| **Step** | **Action** |
| **Pre-condition** | The user has entered a search keyword and is now ready to apply filters. |
| **1.1** | User applies filters such as skill category, availability, and location. |
| **1.2** | The system processes the selected filters and updates the search results. |
| **Post-condition** | The search results are updated based on the selected filters, showing a more refined list of peers. |

Table UC\_005.2 Use Case Statement

UC\_005.3

|  |  |
| --- | --- |
| **Use case ID** | UC\_005.3 |
| **Use Case Name** | Display Search Results |
| **Description** | Displays the search results in a clear and intuitive way, sorted by relevance to the user's search criteria. |
| **Primary Actor** | System |
| **Include Use Cases** | - |
| **Scenario: Main Flow** | |
| **Step** | **Action** |
| **Pre-condition** | The system has processed the search and filter criteria. |
| **1.1** | The system sorts the search results based on relevance to the search keyword and applied filters. |
| **1.2** | The system displays the search results in a list, showing matching peers. |
| **Post-condition** | The user is presented with a list of relevant matches that meet their search criteria. |

Table UC\_005.3 Use Case Statement

### 4.5.3 Test Cases

Test Case for UC\_005 - Search and Filters (TC\_005.1)

Scenario: Main Flow

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tested By** | Placeholder Name | | **Test Level** | | User Acceptance Testing | |
| **Test Type** | Black Box Testing – Functional Testing | | | | | |
| **Test Design Technique** | Use Case Testing | | | | | |
| **Risk Number** | 1 (Lowest) | | | | | |
| **Test Case Number** | TC\_005.1 | | | | | |
| **Test Case Name** | Search and Filters Functionality Test | | | | | |
| **Requirement Traceability** | REQ\_SF-1 (Search and Filters Feature - Search by Skills) | | | | | |
| **Test Case Description** | To verify the functionality of the Search and Filters module. | | | | | |
| **Item to be tested** | | | | | | |
| 1 | Search bar, skill match, filter application, and displaying results. | | | | | |
| **Pre-condition** |  | | | | | |
| **Test Procedure Specification** | | | | | | |
| **Input** | **Test Data** | **Expected Output / Result** | | **Actual Result** | | **Pass / Fail** |
| User enters skill in the search bar and applies filters for availability and location. | Skills to Learn: "Python" | The system matches peers with the skill "Python" and displays results based on the applied filters (e.g., peers available at the specified time and location). The results are displayed in a list, sorted by relevance. | |  | |  |
| **Post-condition** | The student views relevant peers and can choose one for interaction. | | | | | |

Table TC\_005.1 Test Case

Test Case for UC\_005 - Search and Filters (TC\_005.2)

Scenario: Alternate Flow 1 - No Search Results

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tested By** | Placeholder Name | | **Test Level** | | User Acceptance Testing | |
| **Test Type** | Black Box Testing – Functional Testing | | | | | |
| **Test Design Technique** | Use Case Testing | | | | | |
| **Risk Number** | 2 (Medium) | | | | | |
| **Test Case Number** | TC\_005.2 | | | | | |
| **Test Case Name** | No Search Results Test Case | | | | | |
| **Requirement Traceability** | REQ\_SF-1-1 (Search and Filters Feature - No Search Results) | | | | | |
| **Test Case Description** | To verify the system behavior when no matches are found for the search. | | | | | |
| **Item to be tested** | | | | | | |
| 1 | Search functionality and no match display. | | | | | |
| **Pre-condition** |  | | | | | |
| **Test Procedure Specification** | | | | | | |
| **Input** | **Test Data** | **Expected Output / Result** | | **Actual Result** | | **Pass / Fail** |
| User enters unlisted skills in the search bar (no peers have this skill listed). | Skills to Learn: "Quantum Computing" | The system displays a message: "No matches found based on your search criteria." | |  | |  |
| **Post-condition** | The user is informed that no results match their search, and they can modify their criteria. | | | | | |

Table TC\_005.2 Test Case

Test Case for UC\_005 - Search and Filters (TC\_005.3)

Scenario: Alternate Flow 2 - No Matching Results After Filters

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tested By** | Placeholder Name | | **Test Level** | | User Acceptance Testing | |
| **Test Type** | Black Box Testing – Functional Testing | | | | | |
| **Test Design Technique** | Use Case Testing | | | | | |
| **Risk Number** | 3 (High) | | | | | |
| **Test Case Number** | TC\_005.3 | | | | | |
| **Test Case Name** | No Results After Filter Application Test Case | | | | | |
| **Requirement Traceability** | REQ\_SF-2 (Search and Filters Feature - Filters Application) | | | | | |
| **Test Case Description** | To verify the system behavior when no results are returned after applying filters. | | | | | |
| **Item to be tested** | | | | | | |
| 1 | Filter application and handling no results after applying filters. | | | | | |
| **Pre-condition** |  | | | | | |
| **Test Procedure Specification** | | | | | | |
| **Input** | **Test Data** | **Expected Output / Result** | | **Actual Result** | | **Pass / Fail** |
| User enters Skill in the search bar and applies filters for availability at selected time and selected location. No peers match these criteria. | Skills to Learn: "Python" Availability: "5 PM" Location: "New York" | The system displays a message: "No matches found based on your selected filters." | |  | |  |
| **Post-condition** | The user is informed that no results match the selected filters and is prompted to modify them. | | | | | |

Table TC\_005.3 Test Case

Test Case for UC\_005 - Search and Filters (TC\_005.4)

Scenario: Robust Flow - System Error During Search

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tested By** | Placeholder Name | | **Test Level** | | User Acceptance Testing | |
| **Test Type** | Black Box Testing – Functional Testing | | | | | |
| **Test Design Technique** | Use Case Testing | | | | | |
| **Risk Number** | 4 (Critical) | | | | | |
| **Test Case Number** | TC\_005.4 | | | | | |
| **Test Case Name** | System Error During Search Test Case | | | | | |
| **Requirement Traceability** | REQ\_SF-3 (Search and Filters Feature - System Error Handling) | | | | | |
| **Test Case Description** | To verify the system behavior when an error occurs during the search or filter process. | | | | | |
| **Item to be tested** | | | | | | |
| 1 | Error handling and user notification during system failure. | | | | | |
| **Pre-condition** |  | | | | | |
| **Test Procedure Specification** | | | | | | |
| **Input** | **Test Data** | **Expected Output / Result** | | **Actual Result** | | **Pass / Fail** |
| User enters test data in the search bar, but the system encounters a database issue while processing the search. | Skills to Learn: "Python" | The system displays an error message: "There was an issue retrieving search results. Please try again later." | |  | |  |
| **Post-condition** | The search process fails, and the user is informed about the system issue. | | | | | |

Table TC\_005.4 Test Case

Test Case for UC\_005 - Search and Filters (TC\_005.5)

Scenario: Main Flow - Sorting Search Results

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tested By** | Placeholder Name | | **Test Level** | | User Acceptance Testing | |
| **Test Type** | Black Box Testing – Functional Testing | | | | | |
| **Test Design Technique** | Use Case Testing | | | | | |
| **Risk Number** | 2 (Medium) | | | | | |
| **Test Case Number** | TC\_005.5 | | | | | |
| **Test Case Name** | Sorting Search Results Test Case | | | | | |
| **Requirement Traceability** | REQ\_SF-3-1 (Search and Filters Feature - Sorting Results) | | | | | |
| **Test Case Description** | To verify that the system sorts search results based on relevance to the user's criteria. | | | | | |
| **Item to be tested** | | | | | | |
| 1 | Sorting functionality and result relevance. | | | | | |
| **Pre-condition** |  | | | | | |
| **Test Procedure Specification** | | | | | | |
| **Input** | **Test Data** | **Expected Output / Result** | | **Actual Result** | | **Pass / Fail** |
| User enters "Python" and applies the "Skill Category: Programming" filter. Results include 10 peers with varying levels of relevance. | Skills to Learn: "Python" Availability: "available at 4 PM" | The system displays results sorted by relevance, with the most relevant peers appearing at the top. | |  | |  |
| **Post-condition** | The user is presented with sorted search results based on relevance. | | | | | |

Table TC\_005.5 Test Case

4.6 System Feature 5: In-App Chatting (IC)

4.6.1 Use Case Diagram

**A diagram of a chat conversation

AI-generated content may be incorrect.**

Figure Feature 5 In-App Chat Use Case Diagram

4.6.2 Use Case Statements

UC\_006

|  |  |
| --- | --- |
| **Use case ID** | UC\_006 |
| **Use Case Name** | In-App Chatting |
| **Description** | Allows students to communicate with their peers directly within the platform using real-time messaging, powered by Supabase Realtime Database for instant delivery. This feature supports the coordination of learning sessions between peers. |
| **Primary Actor** | User |
| **Include Use Cases** | Open Chat Window, Send Message, Receive Message, Chat History |
| **Scenario: Main Flow** | |
| **Step** | **Action** |
| **Pre-condition** | The user has already matched with a peer and is on the peer's profile page. |
| **1.1** | User clicks the "Chat" button next to a matched peer. |
| **1.2** | The system opens a real-time chat window between the two students. |
| **1.3** | User types and sends a message in the chat window. |
| **1.4** | The system updates the chat window in real-time and sends a notification to the recipient. |
| **1.5** | The recipient receives a notification of the new message. |
| **1.6** | The system updates the recipient's chat interface to show the newly received message. |
| **1.7** | The recipient reads the message and replies. |
| **1.8** | The system updates the chat window in real-time for both users. |
| **Post-condition** | The two users can continue messaging back and forth in real-time through the chat window. |
| **Scenario: Alternate Flow 1 – No Internet Connection** | |
| **Pre-condition** | The user is on the chat window but has no active internet connection. |
| **2.1** | User types and sends a message, but the system detects no internet connection. |
| **2.2** | The system displays an error message indicating that the message could not be sent due to connectivity issues. |
| **2.3** | The system provides an option to try sending the message again once the internet connection is restored. |
| **Post-condition** | The message is not delivered until the connection is re-established, and The user is informed. |
| **Scenario: Alternate Flow 2 – Chat Window Timeout** | |
| **Pre-condition** | The user is inactive in the chat window for a prolonged period. |
| **2.1** | The system detects inactivity and prompts the user with a message saying, "Session will expire soon due to inactivity." |
| **2.2** | User continues the conversation or the chat window automatically closes if no further actions are made. |
| **2.3** | If the user resumes activity, the system updates the chat interface to restore it. |
| **Post-condition** | The chat session may expire if the user doesn't respond, or it continues after the user interacts again. |
| **Scenario: Robust Flow – System Error** | |
| **Pre-condition** | The user is sending a message, but the system encounters an error (e.g., Supabase service failure). |
| **3.1** | User types and sends a message, but the system encounters a Supabase error while processing the message. |
| **3.2** | The system displays an error message indicating "There was an issue sending your message. Please try again later." |
| **3.3** | The system prompts the user to retry sending the message later or contact support if the issue persists. |
| **Post-condition** | The message is not delivered, and the user is notified of the system issue. |

Table UC\_006 Use Case Statement

UC\_006.1

|  |  |
| --- | --- |
| **Use case ID** | UC\_006.1 |
| **Use Case Name** | Open Chat Window |
| **Description** | The system opens a real-time chat window for users to communicate with each other once they click the "Chat" button on the peer's profile. |
| **Primary Actor** | User |
| **Include Use Cases** | - |
| **Scenario: Main Flow** | |
| **Step** | **Action** |
| **Pre-condition** | The user is logged in and has a matched peer. |
| **1.1** | User clicks the "Chat" button next to a matched peer. |
| **1.2** | The system opens a real-time chat window between the two students. |
| **Post-condition** | A real-time chat window is opened for communication between the two matched peers. |

Table UC\_006.1 Use Case Statement

UC\_006.2

|  |  |
| --- | --- |
| **Use case ID** | UC\_006.2 |
| **Use Case Name** | Send Message |
| **Description** | Allow students to send each other messages in real-time through the in-app chat window. |
| **Primary Actor** | User |
| **Include Use Cases** | Update Chat Window |
| **Scenario: Main Flow** | |
| **Step** | **Action** |
| **Pre-condition** | The user is in an active chat window with a peer. |
| **1.1** | User types and sends a message in the chat window. |
| **1.2** | The system updates the chat window in real-time and sends a notification to the recipient. |
| **Post-condition** | The message is sent and visible in both users' chat windows, with real-time updates. |

Table UC\_006.1 Use Case Statement

UC\_006.3

|  |  |
| --- | --- |
| **Use case ID** | UC\_006.3 |
| **Use Case Name** | Receive Message |
| **Description** | Allows students to receive messages in real-time and view them instantly in the chat window. |
| **Primary Actor** | User |
| **Include Use Cases** | Update Chat Window |
| **Scenario: Main Flow** | |
| **Step** | **Action** |
| **Pre-condition** | The user is logged into the chat window and another student has sent a message. |
| **1.1** | The recipient receives a notification of the new message. |
| **1.2** | The system updates the recipient's chat interface to show the newly received message. |
| **Post-condition** | The recipient can view the message instantly in their chat window. |

Table UC\_006.3 Use Case Statement

UC\_006.4

|  |  |
| --- | --- |
| **Use case ID** | UC\_006.4 |
| **Use Case Name** | Store Chat History |
| **Description** | The system must store chat histories for users to scroll through. |
| **Primary Actor** | System |
| **Include Use Cases** | - |
| **Scenario: Main Flow** | |
| **Step** | **Action** |
| **Pre-condition** | The user has exchanged multiple messages with a peer. |
| **1.1** | User navigates to the chat window to view past messages. |
| **1.2** | The system loads and displays the chat history for the user to scroll through. |
| **Post-condition** | The user can scroll through and read past messages in the chat window. |

Table UC\_006.4 Use Case Statement

### 4.6.3 Test Cases

Test Case for UC\_006 - In-App Chatting (TC\_006.1)

Scenario: Main Flow

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tested By** | Placeholder Name | | **Test Level** | | User Acceptance Testing | |
| **Test Type** | Black Box Testing – Functional Testing | | | | | |
| **Test Design Technique** | Use Case Testing | | | | | |
| **Risk Number** | 1 (Lowest) | | | | | |
| **Test Case Number** | TC\_006.1 | | | | | |
| **Test Case Name** | In-App Chatting Functionality Test | | | | | |
| **Requirement Traceability** | REQ\_IC-1 (In-App Chatting Feature - Real-time Messaging) | | | | | |
| **Test Case Description** | To verify that the chat window opens, messages are sent and received in real-time, and chat history is displayed correctly. | | | | | |
| **Item to be tested** | | | | | | |
| 1 | Chat window opening, real-time messaging, chat history display. | | | | | |
| **Pre-condition** |  | | | | | |
| **Test Procedure Specification** | | | | | | |
| **Input** | **Test Data** | **Expected Output / Result** | | **Actual Result** | | **Pass / Fail** |
| User clicks the "Chat" button, types a message, and sends it to a matched peer. The recipient replies. | TestText123 | The system opens the chat window, messages are exchanged in real-time, and both users see the conversation instantly. Chat history is stored and accessible. | |  | |  |
| **Post-condition** | The two users can continue messaging back and forth in real-time, and both can view the full chat history. | | | | | |

Table TC\_006.1 Test Case

Test Case for UC\_006 - In-App Chatting (TC\_006.2)

Scenario: Alternate Flow 1 - No Internet Connection

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tested By** | Placeholder Name | | **Test Level** | | User Acceptance Testing | |
| **Test Type** | Black Box Testing – Functional Testing | | | | | |
| **Test Design Technique** | Use Case Testing | | | | | |
| **Risk Number** | 3 (High) | | | | | |
| **Test Case Number** | TC\_006.2 | | | | | |
| **Test Case Name** | No Internet Connection During Messaging Test Case | | | | | |
| **Requirement Traceability** | REQ\_IC-1-1 (In-App Chatting Feature - Real-time Messaging) | | | | | |
| **Test Case Description** | To verify the system behavior when there is no internet connection while messaging. | | | | | |
| **Item to be tested** | | | | | | |
| 1 | Internet connectivity, real-time messaging failure, error handling. | | | | | |
| **Pre-condition** |  | | | | | |
| **Test Procedure Specification** | | | | | | |
| **Input** | **Test Data** | **Expected Output / Result** | | **Actual Result** | | **Pass / Fail** |
| User types a message, but the system detects no internet connection and attempts to send the message. | TestText123 | The system displays an error message indicating that the message could not be sent due to connectivity issues and prompts the user to try again later. | |  | |  |
| **Post-condition** | The message is not delivered until the connection is re-established, and the student is informed. | | | | | |

Table TC\_006.2 Test Case

Test Case for UC\_006 - In-App Chatting (TC\_006.3)

Scenario: Alternate Flow 2 - Chat Window Timeout

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tested By** | Placeholder Name | | **Test Level** | | User Acceptance Testing | |
| **Test Type** | Black Box Testing – Functional Testing | | | | | |
| **Test Design Technique** | Use Case Testing | | | | | |
| **Risk Number** | 2 (Medium) | | | | | |
| **Test Case Number** | TC\_006.3 | | | | | |
| **Test Case Name** | Chat Window Timeout Test Case | | | | | |
| **Requirement Traceability** | REQ\_IC-2 (In-App Chatting Feature - Session Timeout) | | | | | |
| **Test Case Description** | To verify that the system handles inactivity in the chat window by timing out and informing the user. | | | | | |
| **Item to be tested** | | | | | | |
| 1 | Inactivity timeout, user prompt for session expiration, chat window handling. | | | | | |
| **Pre-condition** |  | | | | | |
| **Test Procedure Specification** | | | | | | |
| **Input** | **Test Data** | **Expected Output / Result** | | **Actual Result** | | **Pass / Fail** |
| User remains inactive in the chat window for a prolonged period. | - | The system detects inactivity and displays a message: "Session will expire soon due to inactivity." If no further interaction occurs, the chat window closes or the user is prompted to continue. | |  | |  |
| **Post-condition** | The chat session is either automatically closed or remains open if the user interacts again. | | | | | |

Table TC\_006.3 Test Case

Test Case for UC\_006 - In-App Chatting (TC\_006.4)

Scenario: Robust Flow - System Error During Message Sending

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tested By** | Placeholder Name | | **Test Level** | | User Acceptance Testing | |
| **Test Type** | Black Box Testing – Functional Testing | | | | | |
| **Test Design Technique** | Use Case Testing | | | | | |
| **Risk Number** | 4 (Critical) | | | | | |
| **Test Case Number** | TC\_006.4 | | | | | |
| **Test Case Name** | System Error During Message Sending Test Case | | | | | |
| **Requirement Traceability** | REQ\_IC-3 (In-App Chatting Feature - System Error Handling) | | | | | |
| **Test Case Description** | To verify the system behavior when an error occurs while sending a message (e.g., database or Supabase issue). | | | | | |
| **Item to be tested** | | | | | | |
| 1 | Error handling, message failure, and notification system. | | | | | |
| **Pre-condition** |  | | | | | |
| **Test Procedure Specification** | | | | | | |
| **Input** | **Test Data** | **Expected Output / Result** | | **Actual Result** | | **Pass / Fail** |
| User types and sends a message, but the system encounters an error while processing the message. | TestMessage123 | The system displays an error message: "There was an issue sending your message. Please try again later." and prompts the user to retry sending the message or contact support. | |  | |  |
| **Post-condition** | The message is not delivered, and the user is notified of the system issue. | | | | | |

Table TC\_006.4 Test Case

Test Case for UC\_006 - In-App Chatting (TC\_006.5)

Scenario: Main Flow - View Chat History

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tested By** | Placeholder Name | | **Test Level** | | User Acceptance Testing | |
| **Test Type** | Black Box Testing – Functional Testing | | | | | |
| **Test Design Technique** | Use Case Testing | | | | | |
| **Risk Number** | 2 (Medium) | | | | | |
| **Test Case Number** | TC\_006.5 | | | | | |
| **Test Case Name** | View Chat History Test Case | | | | | |
| **Requirement Traceability** | REQ\_IC-2 (In-App Chatting Feature - Chat History) | | | | | |
| **Test Case Description** | To verify that the system properly stores and displays the chat history for users to scroll through. | | | | | |
| **Item to be tested** | | | | | | |
| 1 | Chat history storage, scrolling functionality. | | | | | |
| **Pre-condition** |  | | | | | |
| **Test Procedure Specification** | | | | | | |
| **Input** | **Test Data** | **Expected Output / Result** | | **Actual Result** | | **Pass / Fail** |
| User scrolls through past messages in the chat window to view the chat history. | - | The system displays the chat history in the window, allowing the user to scroll through previous messages. | |  | |  |
| **Post-condition** | The user can view all previous messages in the chat window. | | | | | |

Table TC\_006.5 Test Case

4.7 System Feature 6: Rate and Review (RR)

4.7.1 Use Case Diagram

**A diagram of a system

AI-generated content may be incorrect.**

Figure Feature 6 Rate and Review Use Case Diagram

4.7.2 Use Case Statements

UC\_007

|  |  |
| --- | --- |
| **Use case ID** | UC\_007 |
| **Use Case Name** | Rate and Review |
| **Description** | Allows students to leave feedback after completing a learning session, including a star rating (1-5 stars) and written comments. The system calculates and updates the average rating for the reviewed peer, helping build trust within the platform. |
| **Primary Actor** | User |
| **Include Use Cases** | Rate Peer, Write Feedback, Update Average Rating, Review History |
| **Scenario: Main Flow** | |
| **Step** | **Action** |
| **Pre-condition** | The user has completed a learning session and is on the peer’s profile page. |
| **1.1** | User clicks the "Rate and Review" button after completing the session. |
| **1.2** | The system displays a rating interface with options for 1-5 stars and a text box for written feedback. |
| **1.3** | User selects a star rating and writes feedback. |
| **1.4** | The system stores the feedback and updates the average rating for the reviewed peer. |
| **1.5** | The system calculates and updates the average rating for the peer being reviewed. |
| **1.6** | The updated rating is displayed on the peer’s profile. |
| **1.7** | User reviews the feedback they have written in their review history. |
| **1.8** | The system displays the user’s previously submitted ratings and reviews for all sessions. |
| **Post-condition** | The rating is stored, the peer’s profile is updated with the new average rating, and the user can view their review history. |
| **Scenario: Alternate Flow 1 – Missing Feedback** | |
| **Pre-condition** | The user attempts to submit a rating without writing any feedback in the text box. |
| **2.1** | User selects a star rating but leaves the text box empty. |
| **2.2** | The system detects that no feedback has been written. |
| **2.3** | The system prompts the user to enter feedback, with an option to skip the feedback and submit only the star rating. |
| **Post-condition** | The system ensures that users can leave feedback, even if they decide not to write a review, but encourages feedback submission. |
| **Scenario: Alternate Flow 2 – Edit or Deleting Review** | |
| **Pre-condition** | The user has submitted a review and wishes to modify or delete it within the allowed time frame. |
| **2.1** | User clicks "Edit" or "Delete" next to their previously submitted review. |
| **2.2** | The system allows the user to either edit the text or delete the review entirely. |
| **2.3** | The system updates the peer's profile with the edited review or removes the review if deleted. |
| **Post-condition** | The peer’s profile is updated with the new review or no review is displayed if the user deletes it. |
| **Scenario: Robust Flow – System Error** | |
| **Pre-condition** | The user has selected a rating and written feedback but the system encounters an error during submission. |
| **3.1** | User submits the rating and feedback, but the system encounters a server or database error. |
| **3.2** | The system displays an error message: "There was an issue submitting your rating. Please try again later." |
| **3.3** | The system prompts the user to try again later or contact support if the issue persists. |
| **Post-condition** | The rating is not submitted, and The user is notified of the system error. |

Table UC\_007 Use Case Statement

UC\_007.1

|  |  |
| --- | --- |
| **Use case ID** | UC\_007.1 |
| **Use Case Name** | Rate Peer |
| **Description** | Allows The users to select a star rating (1-5) to provide feedback about a learning session. |
| **Primary Actor** | User |
| **Include Use Cases** | - |
| **Scenario: Main Flow** | |
| **Step** | **Action** |
| **Pre-condition** | The user is on the peer’s profile page after completing a learning session. |
| **1.1** | User clicks the "Rate and Review" button to start rating the peer. |
| **1.2** | The system displays the rating interface with a 1-5 star selection. |
| **1.3** | User selects a star rating based on the session’s quality. |
| **Post-condition** | The selected star rating is recorded and stored in the system. |

Table UC\_007.1 Use Case Statement

UC\_007.2

|  |  |
| --- | --- |
| **Use case ID** | UC\_007.2 |
| **Use Case Name** | Write Feedback |
| **Description** | Allows The users to write detailed feedback about the learning session, providing constructive criticism or praise. |
| **Primary Actor** | User |
| **Include Use Cases** | - |
| **Scenario: Main Flow** | |
| **Step** | **Action** |
| **Pre-condition** | The user has selected a star rating for the session. |
| **1.1** | The system displays a text box for The user to write their feedback. |
| **1.2** | User writes feedback in the provided text box. |
| **Post-condition** | The written feedback is stored along with the star rating and can be updated or deleted if needed. |

Table UC\_007.2 Use Case Statement

UC\_007.3

|  |  |
| --- | --- |
| **Use case ID** | UC\_007.3 |
| **Use Case Name** | Update Average Rating |
| **Description** | The system calculates and update average rating for a peer after receiving a new review. |
| **Primary Actor** | System |
| **Include Use Cases** | - |
| **Scenario: Main Flow** | |
| **Step** | **Action** |
| **Pre-condition** | The system has received a new rating for a peer from a student. |
| **1.1** | The system calculates the new average rating based on all reviews received for the peer. |
| **1.2** | The system updates the peer’s profile page with the newly calculated average rating. |
| **Post-condition** | The peer’s profile reflects the updated average rating. |

Table UC\_007.3 Use Case Statement

UC\_007.4

|  |  |
| --- | --- |
| **Use case ID** | UC\_007.4 |
| **Use Case Name** | Display Average Rating |
| **Description** | Displays the average rating of a peer on their profile page, providing transparency for other students. |
| **Primary Actor** | System |
| **Include Use Cases** | - |
| **Scenario: Main Flow** | |
| **Step** | **Action** |
| **Pre-condition** | The peer has received at least one rating. |
| **1.1** | The system calculates and displays the average rating on the peer’s profile page. |
| **Post-condition** | The user can view the peer’s average rating when considering them as a potential learning partner. |

Table UC\_007.4 Use Case Statement

### 4.7.3 Test Cases

Test Case for UC\_007 - Rate and Review (TC\_007.1)

Scenario: Main Flow

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tested By** | Placeholder Name | | **Test Level** | | User Acceptance Testing | |
| **Test Type** | Black Box Testing – Functional Testing | | | | | |
| **Test Design Technique** | Use Case Testing | | | | | |
| **Risk Number** | 1 (Lowest) | | | | | |
| **Test Case Number** | TC\_007.1 | | | | | |
| **Test Case Name** | Rate and Review Functionality Test | | | | | |
| **Requirement Traceability** | REQ\_RR-1 (Rate and Review Feature - Rating and Review Submission) | | | | | |
| **Test Case Description** | To verify the functionality of the rating and review process (selecting stars, writing feedback, and submitting). | | | | | |
| **Item to be tested** | | | | | | |
| 1 | Rate selection, feedback submission, profile update. | | | | | |
| **Pre-condition** |  | | | | | |
| **Test Procedure Specification** | | | | | | |
| **Input** | **Test Data** | **Expected Output / Result** | | **Actual Result** | | **Pass / Fail** |
| User clicks the "Rate and Review" button, selects a 4-star rating, and writes a comment in the feedback box. | 4 Stars  “TestComment” | The system records the star rating and feedback, calculates the updated average rating for the peer, and displays the rating on the peer’s profile. | |  | |  |
| **Post-condition** | The peer’s profile reflects the updated rating, and the user can see their submitted review in the review history. | | | | | |

Table TC\_007.1 Test Case

Test Case for UC\_007 - Rate and Review (TC\_007.2)

Scenario: Alternate Flow 1 - No Written Feedback

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tested By** | Placeholder Name | | **Test Level** | | User Acceptance Testing | |
| **Test Type** | Black Box Testing – Functional Testing | | | | | |
| **Test Design Technique** | Use Case Testing | | | | | |
| **Risk Number** | 2 (Medium) | | | | | |
| **Test Case Number** | TC\_007.2 | | | | | |
| **Test Case Name** | No Written Feedback Submission Test Case | | | | | |
| **Requirement Traceability** | REQ\_RR-1 (Rate and Review Feature - Rating and Review Submission) | | | | | |
| **Test Case Description** | To verify that the system allows users to submit a rating without written feedback. | | | | | |
| **Item to be tested** | | | | | | |
| 1 | Rating functionality without feedback, system handling of no feedback. | | | | | |
| **Pre-condition** |  | | | | | |
| **Test Procedure Specification** | | | | | | |
| **Input** | **Test Data** | **Expected Output / Result** | | **Actual Result** | | **Pass / Fail** |
| User clicks the "Rate and Review" button, selects a 5-star rating, and leaves the feedback text box empty. | 4 Stars | The system records the star rating and allows the submission without written feedback. | |  | |  |
| **Post-condition** | The peer’s profile reflects the rating without written feedback. | | | | | |

Table TC\_007.2 Test Case

Test Case for UC\_007 - Rate and Review (TC\_007.3)

Scenario: Alternate Flow 2 - Edit/Delete Review

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tested By** | Placeholder Name | | **Test Level** | | User Acceptance Testing | |
| **Test Type** | Black Box Testing – Functional Testing | | | | | |
| **Test Design Technique** | Use Case Testing | | | | | |
| **Risk Number** | 3 (High) | | | | | |
| **Test Case Number** | TC\_007.3 | | | | | |
| **Test Case Name** | Edit/Delete Review Test Case | | | | | |
| **Requirement Traceability** | REQ\_RR-2-1 (Rate and Review Feature - Edit or Delete Review) | | | | | |
| **Test Case Description** | To verify that the system allows users to edit or delete their review within the allowed time frame. | | | | | |
| **Item to be tested** | | | | | | |
| 1 | Review editing and deletion functionality. | | | | | |
| **Pre-condition** |  | | | | | |
| **Test Procedure Specification** | | | | | | |
| **Input** | **Test Data** | **Expected Output / Result** | | **Actual Result** | | **Pass / Fail** |
| User submits a 4-star rating and feedback, then decides to edit the review. User changes the rating to 5 stars and modifies the text. | 5 Stars  “TestComment2" | The system updates the peer’s profile with the edited rating and feedback. | |  | |  |
| **Post-condition** | The peer’s profile reflects the updated review, and the user’s review history shows the modified review. | | | | | |

Table TC\_007.3 Test Case

Test Case for UC\_007 - Rate and Review (TC\_007.4)

Scenario: Robust Flow - System Error During Review Submission

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tested By** | Placeholder Name | | **Test Level** | | User Acceptance Testing | |
| **Test Type** | Black Box Testing – Functional Testing | | | | | |
| **Test Design Technique** | Use Case Testing | | | | | |
| **Risk Number** | 4 (Critical) | | | | | |
| **Test Case Number** | TC\_007.4 | | | | | |
| **Test Case Name** | System Error During Review Submission Test Case | | | | | |
| **Requirement Traceability** | REQ\_RR-3 (Rate and Review Feature - Error Handling) | | | | | |
| **Test Case Description** | To verify the system behavior when an error occurs during the review submission process (e.g., database error). | | | | | |
| **Item to be tested** | | | | | | |
| 1 | Error handling, user notification during review submission failure. | | | | | |
| **Pre-condition** |  | | | | | |
| **Test Procedure Specification** | | | | | | |
| **Input** | **Test Data** | **Expected Output / Result** | | **Actual Result** | | **Pass / Fail** |
| User submits a 3-star rating and feedback, but the system encounters a database error while processing the review. | - | The system displays an error message: "There was an issue submitting your rating. Please try again later." | |  | |  |
| **Post-condition** | The rating is not submitted, and the user is notified about the system issue. | | | | | |

Table TC\_007.4 Test Case

Test Case for UC\_007 - Rate and Review (TC\_007.5)

Scenario: Main Flow - Display Average Rating

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tested By** | Placeholder Name | | **Test Level** | | User Acceptance Testing | |
| **Test Type** | Black Box Testing – Functional Testing | | | | | |
| **Test Design Technique** | Use Case Testing | | | | | |
| **Risk Number** | 2 (Medium) | | | | | |
| **Test Case Number** | TC\_007.5 | | | | | |
| **Test Case Name** | Display Average Rating Test Case | | | | | |
| **Requirement Traceability** | REQ\_RR-3-1 (Rate and Review Feature - Display Average Rating) | | | | | |
| **Test Case Description** | To verify that the average rating is displayed on the peer’s profile after a review is submitted. | | | | | |
| **Item to be tested** | | | | | | |
| 1 | Average rating calculation, profile update. | | | | | |
| **Pre-condition** |  | | | | | |
| **Test Procedure Specification** | | | | | | |
| **Input** | **Test Data** | **Expected Output / Result** | | **Actual Result** | | **Pass / Fail** |
| User submits a 4-star rating and feedback for a peer. | 4 Stars | The system updates the peer’s profile with the new average rating based on the submitted review. | |  | |  |
| **Post-condition** | The peer’s profile shows the updated average rating. | | | | | |

Table TC\_007.5 Test Case

# 5.0 Test Environment

## 5.1 Test Site

### 5.1.1 Developer Site

The primary test site for the UniKL TutorFind project will be the **Developer Site**. This refers to the individual development environments configured and maintained by the project team members. While these may be personal workstations, a standardized setup and consistent configuration will be adhered to for testing purposes to ensure replicability of results.

## 5.2 Facilities required

### 5.2.1 Lab Space

A dedicated **Lab Space** (or a consistent, designated working area) is required for focused testing activities. This environment should minimize external distractions and allow for the stable setup of all necessary hardware and software components used for testing the application.

### 5.2.2 Furniture

Appropriate **Furniture**, including ergonomically suitable desks and chairs, will be utilized to support the extended periods of testing and documentation required for comprehensive quality assurance.

### 5.2.3 Power Infrastructure

A stable and reliable **Power Infrastructure** is essential to ensure continuous operation of all testing hardware, including laptops, mobile devices, and any supplementary equipment. This mitigates risks of power interruptions during critical testing phases.

### 5.2.4 Communication Facilities

Essential **Communication Facilities** are vital for seamless collaboration within the project team and for reporting findings to the project supervisor/assessor. This includes reliance on institutional email, virtual conferencing platforms (e.g., Google Meet, Microsoft Teams), and real-time chat applications for immediate dissemination of test results and defect reports.

### 5.2.5 Network Connectivity

Consistent and robust **Network Connectivity** (both wired and wireless, as applicable) is critical, particularly given the UniKL TutorFind platform's dependency on Supabase for real-time data synchronization. Reliable internet access is fundamental for testing cloud-based functionalities, user authentication, and real-time communication features across both web and mobile interfaces.

## 5.3 Required hardware and software specifications

The UniKL TutorFind platform is developed using Flutter for cross-platform mobile and web application development, and Supabase for backend services. The hardware and software specifications are designed to accommodate these core technologies.

### 5.3.1 Hardware Requirement

**For Web Application Testing:**

* **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).
* **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:**

* **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.

### 5.3.2 Software Requirements

**Operating Systems:**

* Windows 10 Pro (64-bit) or Windows 11 Pro (64-bit) for desktop development and web application testing.
* 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):**

* 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).
* Android Studio (latest version, including Android SDK and build tools).
* Xcode (latest version, for iOS development and simulators).
* Supabase CLI and associated SDKs for backend service interaction.
* PHP (version compatible with Laravel, e.g., PHP 8.x).
* Composer (PHP dependency manager).
* Node.js and npm/yarn (for front-end asset compilation if used).
* MySQL Database Server (latest stable version).

**Testing and Documentation Tools:**

* Version Control System (Git, via GitHub).
* Integrated Development Environments (IDEs) such as VS Code, Android Studio, Xcode.
* PDF Reader for reviewing test plans and requirements documents.
* Microsoft Office Suite or equivalent (for documentation, test reports).

## 5.4 Testing Group – Tester Involved

The testing group for the UniKL TutorFind project primarily comprises the student developers themselves:

* **Alyssa Husna binti Jamarizan (52213223155)**
* **Alya Azwin binti Zamri (52213223009)**

As developers, they will assume the dual role of testers, responsible for designing, executing, and documenting test cases, as well as reporting and tracking defects. This integrated approach leverages their understanding of the system's internal workings for more effective defect identification within the project's academic scope.

## 5.5 Preparation and training required of the test team

Given the nature of an academic project where developers often double as testers, preparation and training are crucial for ensuring the quality of the testing process. This includes:

* **Comprehensive Project Understanding:** A thorough understanding of the entire project scope, system requirements (functional and non-functional), and architectural design.
* **Self-Paced Learning and Research:** Continuous self-training on best practices in software testing, particularly for mobile and web applications, and familiarization with specific testing methodologies (e.g., Black Box, Risk-Based). This includes referencing external resources like the ISTQB syllabus.
* **Adherence to Documentation Standards:** Strict adherence to the University's guidelines and templates for test plan documentation, test case writing, and defect reporting (e.g., IEEE 829 for incident reports).
* **Tool Proficiency:** Ensuring complete proficiency with all specified development, testing, and communication tools.
* **Internal Knowledge Transfer:** Regular discussions and peer reviews within the small team to share knowledge, align testing strategies, and ensure consistency in test execution and defect reporting.
* **Supervisor Consultation:** Leveraging guidance from the project supervisor for validation of testing strategies and problem resolution.

# 6.0 Test Schedule

This section details the management aspects of the testing process for UniKL TutorFind, including test organization, planning, progress monitoring, risk management, and incident management.

## 6.1 Test Organization

Test organization defines how testing is structured within the project, including roles, responsibilities, and the degree of independence.

* **Test Team Roles:** For the UniKL TutorFind project, the student developers assume both test management and tester roles.
* **Test Manager (Lead Tester):** In this academic context, the student developers jointly assume the responsibilities of a test manager. Their tasks include:
* Providing and updating the Test Policy and Test Strategy.
* Planning and coordinating testing activities with project managers (supervisors).
* Distributing test effort and setting objectives.
* Selecting and applying testing techniques, and setting coverage degrees.
* Estimating time, effort, and cost of testing.
* Identifying and managing resources.
* Defining test levels, iterations, entry and exit criteria.
* Planning incident management.
* Monitoring and controlling test execution, updating schedules, and writing test reports.
* Setting up configuration management for testware.
* Selecting and gathering metrics.
* Selecting and organizing training for test tools (if applicable).
* Deciding on test harness implementation.
* **Tester (Test Designer, Test Automator, Test Administrator):** The student developers also perform these operational roles. Their tasks include:
* Analyzing, reviewing, and assessing specifications and models for testability.
* Contributing to test schedule creation.
* Creating test specifications and implementing test harnesses (e.g., drivers and stubs for component/integration testing).
* Preparing and acquiring test data.
* Executing tests at all levels.
* Logging and evaluating test results, documenting incidents.
* Using test administration and monitoring tools.
* Automating tests where feasible.
* Measuring performance (if required).
* Reviewing test cases developed by others.
* **Testers' Soft Skills:** Beyond technical skills, effective testers need: team-mindedness, diplomatic skills, willingness to question assumptions, assertiveness, confidence, accuracy, creativity, and the capability to quickly understand complex software and domains.
* **Independence of Testing:** While full organizational independence (separate test team) may not be feasible in a small academic project, fostering a degree of independence is important.
* **Possible Approaches (in increasing order of independence):**

1. Developers test their own code.
2. Developers test each other's code (peer review).
3. One member of the development team is exclusively responsible for testing ("module tester").
4. Independent test teams within the organization report to project management.
5. Independent testers from technical or IT departments, or user groups.
6. External testing service providers.

* **Advantages of Independence:** Impartiality, fresh perspective, ability to question assumptions, discovery of different types of defects.
* **Disadvantages of Independence (less relevant for small projects):** Higher communication effort, potential bottleneck if not planned well, risk of developers abdicating quality responsibility. For UniKL TutorFind, developers are the testers, so internal peer review and supervisor feedback are the primary mechanisms for achieving a degree of independent perspective.

## 6.2 Test Planning and Estimation

Test planning is a continuous activity within the RAD model, detailing the scope, approach, resources, and schedule.

* **Test Planning Objectives:** To define the objectives of testing for the project, identify the items to be tested, specify the testing tasks, personnel responsible, and the overall schedule. It supports effective test control.
* **Content of Test Plan:** This document (Software Test Plan) summarizes the purpose and content of the test planning, aligning with standards like IEEE 829.
* **Test Approaches:** For UniKL TutorFind, the chosen test approaches are primarily **model-based** (iterative-incremental RAD), **methodical** (functional, structural testing), and **analytical** (risk-based testing).
* **Test Effort Estimation:** For an academic project, effort estimation is integrated into the overall project timeline. Factors influencing effort include:
* **Product characteristics:** Quality of test basis (SRS), size, complexity, testability, required reliability/security/performance levels.
* **Development process characteristics:** Maturity of processes, tools used, stability of environment, skills of the team.
* **Test results:** Number and severity of defects found, rework required.
* **Estimation Techniques:** While formal metrics-based (e.g., historical data, industry benchmarks) and expert-based (e.g., Wideband Delphi) approaches exist for larger projects, for UniKL TutorFind, estimation relies on expert judgment (student team, supervisor) integrated with the academic timeline.

## 6.3 Test Schedule (Testing and Deployment Timeline)

The primary testing activities, focusing on **Functionality Testing**, are scheduled as follows, based on the provided Gantt chart:

|  |  |  |  |
| --- | --- | --- | --- |
| **Task Title** | **Start Date** | **End Date** | **Duration (Weeks)** |
| Functionality Testing | 22/12/25 | 4/1/26 | 2 |
| Deployment | 5/1/26 | 18/1/26 | 2 |

Table Test Planning & Estimation

**Notes on the Testing Schedule:**

* **Functionality Testing:** This phase is dedicated to rigorously testing all core functionalities and system features identified in the Test Plan (Chapter 3.3.1). It aims to ensure that the UniKL TutorFind application performs according to its specified requirements and user expectations across both web and mobile platforms.
* **Integrated Approach:** While a dedicated "Functionality Testing" phase is noted, it's important to understand that testing activities are implicitly conducted throughout the development increments. The listed "Functionality Testing" phase represents a comprehensive system-level and acceptance testing effort prior to final deployment.
* **Deployment:** Following successful functionality testing and resolution of critical defects, the application will proceed to the deployment phase. This involves making the application accessible for final review and demonstration.

## 6.4 Iterative Testing Approach (Per Increment)

In line with the RAD methodology, various levels of testing, including User Acceptance Testing (UAT) or its academic equivalent, will be performed at the end of each major development increment. This ensures continuous feedback, early defect detection, and validation against project requirements. This iterative approach ensures that quality is built into the product throughout the development cycle, rather than being a single, isolated activity at the very end.

### 6.4.1 Development Increment 1: Login and Registration & Profile Creation (Completed by early June 2025)

* **Features:** Authentication System & Database, System Modules 1 & 2 (Login, Registration, Profile Creation).
* **Testing Focus:** Immediately following the completion of these modules (ending around 25/5/25, with System Demo from 26/5/25 to 1/6/25), the team will conduct **Unit Testing** and **Integration Testing** to verify individual components and their interactions. This will be followed by **System Testing** to ensure these foundational features work correctly as a whole.
* **User Acceptance (Academic Context):** An informal **User Acceptance Review** will be conducted, likely coinciding with the "System Demo" phase. This will involve the project team demonstrating the implemented features to the supervisor(s) or selected peer reviewers. The objective is to gather early feedback on usability, adherence to initial requirements, and overall user experience for the core authentication and profile management functionalities. Feedback received will inform necessary adjustments for subsequent iterations.

### 6.4.2 Development Increment 2: Profile Matching & In-App Chatting (Completed by mid-November 2025)

* **Features:** Profile Matching Algorithm, Messaging System.
* **Testing Focus:** Upon the completion of these features (Profile Matching Algorithm ending 2/11/25, Messaging System ending 16/11/25), **Integration Testing** will be crucial to ensure seamless interaction between newly developed modules and existing core functionalities. **System Testing** will then verify the end-to-end flow of profile matching and real-time communication.
* **User Acceptance (Academic Context):** A more structured **User Acceptance Walkthrough** will be performed with the supervisor(s) following the completion of these modules. This session will specifically validate that the matching algorithms produce accurate results and that the in-app chat provides a functional and intuitive communication channel, meeting the user stories defined in the Software Requirements Specification. User feedback at this stage is critical for refining the interactive aspects of the platform.

### 6.4.3 Development Increment 3: Search and Filters & Rate and Review (Completed by late December 2025)

* **Features:** Search and Filter Algorithm, Feedback System.
* **Testing Focus:** As these features are developed and integrated (Search and Filter Algorithm ending 7/12/25, Feedback System ending 21/12/25), continuous **System Testing** will occur to ensure their proper functioning within the overall application.
* **User Acceptance (Academic Context):** Specific **User Feedback Sessions** will be held for these features. This will ensure that the search results are accurate and efficient, and that the rating and review mechanism effectively captures and displays user feedback, contributing to the platform's overall trustworthiness and utility.

### 6.4.4 Final Functionality Testing (22/12/25 - 4/1/26)

* This dedicated phase represents the formal **System Testing** and **Acceptance Testing** of the then-complete version of the application. It ensures that all integrated modules function cohesively and meet all functional and non-functional requirements before the final project submission. This is the primary UAT stage where the application's readiness for its academic demonstration is formally assessed.

## 6.5 Test Progress Monitoring and Control

Test progress monitoring and control involves systematically collecting and analyzing test metrics to assess test progress against the plan, and taking corrective actions as needed (ISTQB Chapter 6 - Test Management).

* **Test Metrics:** Measurable characteristics used to monitor test progress and evaluate test completion. For UniKL TutorFind, relevant metrics include:
* **Defect-based metrics:** Number of defects found and fixed per test cycle or module, defect density (defects per unit of code/feature), defect severity and priority distribution, re-test results.
* **Test case-based metrics:** Percentage of planned test cases executed, percentage of test cases passed/failed, number of test cases run per day.
* **Test object-based metrics:** Test coverage (e.g., requirement coverage, functional area coverage, potentially code coverage for specific critical modules from white-box testing).
* **Test Reporting:** Regular reporting provides transparency on test status.
* **Test Status Reports:** Produced regularly (e.g., weekly or at the end of each increment) to communicate current test progress, issues, and risks.
* **Test Summary Reports (IEEE 829):** A final report issued at the end of major test phases or the entire project, summarizing test activities, results, deviations from plan, and providing information for release decisions.
* **Test Control:** Based on monitoring, corrective actions are taken. This includes adjusting schedules, reallocating resources, re-prioritizing tests, or escalating issues to management.
* **Exit Criteria Evaluation:** Continuously evaluate if the defined exit criteria (from Chapter 3.7.2) are being met, providing objective data to decide when a test phase or the overall testing can be considered complete.

## 6.6 Risk Management

Risk management is the systematic application of procedures and practices to identify, analyze, prioritize, and control risks to avoid potential failures. Testing plays a crucial role in managing both product and project risks.

* **Activities:**
* **Risk Identification:** Continuously identifying potential risks throughout the project lifecycle (e.g., unclear requirements, complex new technology, resource limitations).
* **Risk Analysis and Prioritization:** Assessing the **likelihood** (probability of a risk occurring) and **impact** (harm if the risk materializes) of identified risks. This allows for prioritizing risks (e.g., using a matrix: Total Risk = Likelihood \* Impact).
* **Risk Monitoring and Control:** Tracking identified risks, monitoring their status, and taking proactive measures to keep them within acceptable levels.
* **Risk Mitigation:** Implementing actions to reduce the likelihood or impact of identified risks.
* **Product Risks:** Risks related to the actual quality or fitness for purpose of the UniKL TutorFind application (e.g., software failing, performance issues, security flaws). Testing helps mitigate these by actively finding defects.
* **Project Risks:** Risks related to the project management and execution, impacting budget, schedule, or resources (e.g., delays, resource unavailability). Testing informs these risks by providing progress updates and defect data.
* **Risk-Based Testing:** As described in Chapter 3.9.4, this approach ensures that testing effort is focused proportionally to the level of risk, prioritizing tests for critical functionalities or areas with high uncertainty.

## 6.7 Incident Management

Incident management (also known as defect management or bug tracking) is the process of reporting, classifying, tracking, and resolving deviations observed during testing. A well-functioning incident management procedure is crucial for defect resolution.

* **Incident (Defect) Reporting:** When a tester observes a deviation from expected behavior, an incident report is created. It's important to use the term "incident" to avoid immediate blame.
* **Content of an Incident Report (based on IEEE 829):**
* **Identification:** Unique ID, date/time of detection, identity of tested software/document (version, configuration), activity where deviation was observed, identity of test case (if applicable).
* **Description:** Clear and accurate description of the deviation, steps to reproduce, actual result, expected result.
* **Impact:** Severity (impact on system/stakeholders: e.g., Class 1-5, from system crash to minor blemish).
* **Urgency:** Priority (urgency of correction: e.g., Patch, Next Version, Occasionally, Open).
* **Status:** Current state of the incident (e.g., Open, Assigned, Fixed, Retest, Closed, Deferred, Duplicate, Rejected).
* **History:** Record of changes made to the incident report, including actions taken by developers and testers.
* **Conclusions/Recommendations:** Insights into the root cause or suggestions for prevention.
* **Incident Database:** A central tool (e.g., JIRA, Bugzilla, or even a shared spreadsheet for a small project) is used to collect, administer, sort, and track all incidents. This facilitates workflow, allocation of tasks, and provides an up-to-date view of incident status.
* **Defect Life Cycle:** Incidents typically go through a lifecycle (e.g., New -> Open -> Assigned -> Fixed -> Retest -> Closed). Testers are responsible for creating reports and retesting fixes; developers are responsible for analyzing and fixing defects.
* **Test Log:** A chronological record of relevant details about the execution of tests, used to identify mismatches and inform incident reports.

## 6.8 Configuration Management

Configuration management (CM) is a critical supporting activity for testing, ensuring the integrity and traceability of testware and software (ISTQB Chapter 6 - Test Management).

* **Purpose:** To establish and maintain the integrity of work products (software, testware, documentation) throughout the project lifecycle.
* **Support for Testing:**
* **Version Control:** Ensures that the correct versions of test objects (application builds) and testware (test plans, test cases, test data, test scripts) are used and maintained. This is crucial for regression testing and replicating defects.
* **Traceability:** Allows for linking testware to specific versions of the software under test and to the requirements it validates.

**Environment Management:** Helps manage different hardware and software configurations used for testing, especially relevant for cross-platform applications like UniKL TutorFind (web, Android, iOS).

# 7.0 Definitions, Acronyms and Abbreviation

## 7.1 Definitions

* **Acceptance Testing:** A formal testing process to determine if a system satisfies its acceptance criteria and to enable the customer to determine whether to accept the system. In an academic context, this often involves demonstration to and approval by the project supervisor or academic panel.
* **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.
* **Branch Coverage (C1-coverage):** A White Box testing criterion that requires that all decision outcomes (branches) in a control flow graph are executed at least once. 100% decision coverage implies 100% branch coverage.
* **Component Testing:** The first level of testing, focusing on individual, smallest testable units or modules of software in isolation.
* **Coverage:** The degree, expressed as a percentage, to which a specified coverage item (e.g., statements, decisions, requirements) has been exercised by a test suite.
* **Debugging:** The development activity that finds, analyzes, and removes the cause of a failure.
* **Defect (Fault/Bug):** A flaw in a component or system that can cause the component or system to fail to perform its required function.
* **Defect Masking:** An occurrence in which one defect prevents the detection of another defect.
* **Deficiency:** Non-fulfillment of a requirement related to an intended or specified use, which can include impairment of usability or failure to meet performance.
* **Deployment:** The process of making the software application available for use, typically on a server or hosting environment, allowing users to access it.
* **Decision (Program Point):** A program point at which the control flow has two or more alternative routes (e.g., an if statement or a loop).
* **Dynamic Testing:** Testing that involves the execution of the software of the component or system.
* **Error:** A human action that produces an incorrect result; a mistake made by a person.
* **Exit Criteria:** The set of generic and specific conditions, agreed upon with the stakeholders, for permitting a process (e.g., a test phase) to be officially completed.
* **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 cross-platform applications from a single codebase.
* **Functionality Testing:** A type of software testing that verifies whether the software system performs as per the functional requirements specification.
* **Incident:** Any event occurring that requires investigation (e.g., a query, a problem, an observation, a suggestion for improvement). Often refers to a defect.
* **Incident Management:** The process of recognizing, investigating, taking action to resolve incidents (defects).
* **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.
* **Laravel:** A free, open-source PHP web framework, intended for the development of web applications following the model–view–controller (MVC) architectural pattern.
* **MySQL:** An open-source relational database management system (RDBMS) that uses Structured Query Language (SQL).
* **Peer-to-Peer (P2P):** A decentralized communications model in which each party has equivalent capabilities and can initiate a communication session. In UniKL TutorFind, it refers to students directly connecting to exchange skills.
* **Rapid Application Development (RAD):** An agile software development methodology that emphasizes rapid prototyping, iterative development, and continuous feedback to deliver high-quality systems quickly.
* **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 (ISO 9126).
* **Requirements Traceability:** The ability to identify related items in documentation and software, such as requirements with associated tests, allowing for impact analysis.
* **Review:** A manual static testing technique involving human examination of documents (e.g., requirements, design, code) to find and remove defects.
* **Risk:** A possible problem that would threaten the achievement of one or more stakeholders' project objectives. Defined by likelihood and impact.
* **Risk-Based Testing:** A type of software testing that is based on the prioritization of tests by risk, focusing effort on higher-risk items.
* **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.
* **Static Analysis:** Automated testing against predefined rules, typically applied to formal work products like source code, to identify defects without executing the code.
* **Static Testing:** Testing of a component or system at requirements or implementation level without execution of any software, e.g., reviews or static code analysis.
* **Statement Coverage (C0-coverage):** A White Box testing criterion that requires all executable statements of the test object to be executed at least once.
* **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.
* **Test Case:** A set of preconditions, inputs, actions (where applicable), expected results and postconditions, developed to verify a particular objective or test condition.
* **Test Environment:** An environment containing hardware, instrumentation, simulators, software tools, and other support elements needed to conduct a test.
* **Test Harness:** A test environment comprised of stubs and drivers needed to execute tests, especially at lower test levels.
* **Test Level:** A group of test activities that are organized and managed together, with specific objectives, typical test objects, test bases, and responsibilities (e.g., component, integration, system, acceptance).
* **Test Procedure Specification (Test Script):** A document specifying a sequence of actions for the execution of a test.
* **Test Suite:** A set of several test cases for a component or system under test.
* **Unit Testing:** (See Component Testing)
* **Usability:** The effort required to use, and the individual assessment of such use by a specific or implied group of users (ISO 9126).
* **Validation:** Confirmation by examination and through provision of objective evidence that the requirements for a specific intended use or application have been fulfilled ("Did we implement the right system?").
* **Verification:** Confirmation by examination and through the provision of objective evidence that specified requirements have been fulfilled ("Did we implement the system right?").
* **White Box Testing (Structure-based testing):** A dynamic test design technique that uses information about the internal structure of the component or system to derive test cases and measure coverage.

## 7.2 Abbreviations

* **CM:** Configuration Management
* **FYP:** Final Year Project
* **IC:** In-App Chatting (System Feature)
* **IEEE:** Institute of Electrical and Electronics Engineers
* **ISO:** International Organization for Standardization
* **ISTQB:** International Software Testing Qualifications Board
* **LR:** Login and Registration (System Feature)
* **MVC:** Model-View-Controller
* **P2P:** Peer-to-Peer
* **PC:** Profile Creation (System Feature)
* **PM:** Profile Matching (System Feature)
* **QA:** Quality Assurance
* **RAD:** Rapid Application Development
* **RR:** Rate and Review (System Feature)
* **SDLC:** Software Development Lifecycle
* **SF:** Search and Filters (System Feature)
* **SQL:** Structured Query Language
* **SRS:** Software Requirements Specification
* **SSD:** Solid State Drive
* **STP:** Software Test Plan
* **UI:** User Interface
* **UniKL:** Universiti Kuala Lumpur
* **UAT:** User Acceptance Testing
* **UX:** User Experience