MATCH MY UNI

**1. Problem Requirement Gathering**

**Problem Statement**  
Students aiming for higher education often face difficulties in finding suitable universities that match their academic achievements, budget, and desired field of study. Many are also unaware of scholarship opportunities based on personal achievements, such as being a *Hafiz-e-Quran* or winning medals in extracurricular activities.

Here’s how the **Login Page** requirement can be integrated into the list, including functional and non-functional aspects:

**2.Requirements**

1. **Login and Authentication System**
   * Develop a simple, secure login page where students can sign in using only their email and password.
   * Include basic functionality such as user registration and email-based password reset.
2. **User Profile Creation**
   * Allow students to create profiles by entering details such as:
     + Personal information (name, date of birth, location).
     + Academic scores (e.g., GPA, standardized test scores).
     + Extracurricular achievements (e.g., sports, competitions).
     + Preferences (countries, programs, budgets).
3. **University Matching Algorithm**
   * Implement a rule-based algorithm that analyzes user profiles to match them with suitable universities.
   * Incorporate filters for location, tuition fee range, program type, and language of instruction.
4. **Scholarship Indicator**
   * Provide tailored suggestions for scholarships based on personal, academic, and extracurricular achievements.
5. **University Exploration Tools**
   * Offer direct access to:
     + University websites.
     + Program details.
     + Dedicated scholarship pages for each institution.
6. **Alternative Program Suggestions**
   * Recommend similar programs to users based on their preferences, academic scores, and budget constraints.
7. **Favorites and Shortlist Functionality**
   * Enable users to mark and save universities or programs as favorites for easy comparison later.
8. **Real-Time Notifications**
   * Notify users about:
     + Application deadlines.
     + Scholarship updates.
     + New programs matching their preferences.
9. **Progress Tracker**
   * Allow students to monitor their application process, including submission status and deadlines for shortlisted universities.
10. **Feedback and Support System**

* Integrate a basic feedback form or email-based query system for user feedback and troubleshooting assistance.

**3. Objectives**

The objectives of *MatchMyUni* are to:

* Provide students with a personalized list of universities that align with their academic and financial profiles.
* Minimize the research burden by offering direct links to relevant university pages.
* Indicate potential scholarship opportunities based on academic and personal achievements.
* Encourage students to explore alternative programs based on their academic strengths and budget.

**4. Goals**

* **Primary Goal**: To develop an intuitive, user-friendly mobile application that matches students with universities and scholarship opportunities tailored to their profile.
* **Secondary Goals**:
  + Achieve at least a 90% satisfaction rate among users by simplifying the university selection process.
  + Expand the app’s university database across major study destinations (e.g., U.S., U.K., Canada).
  + Integrate minimal, scalable backend architecture to support thousands of users.

**5. Stakeholders**

* **Primary Stakeholders**:
  + **Students**: Primary users who benefit from university and scholarship matching.
  + **Educational Consultants**: Could use the app as an additional tool to advise students.
* **Secondary Stakeholders**:
  + **University Marketing Departments**: Potential collaboration to increase visibility for institutions.
  + **Scholarship Providers**: Can use the platform to promote scholarship7opportunities.

**6. Tools and Techniques**

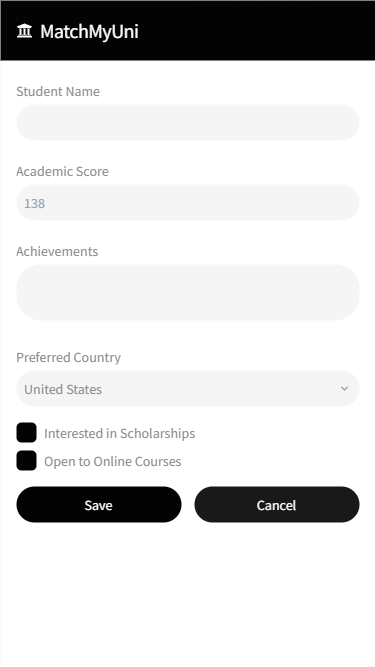
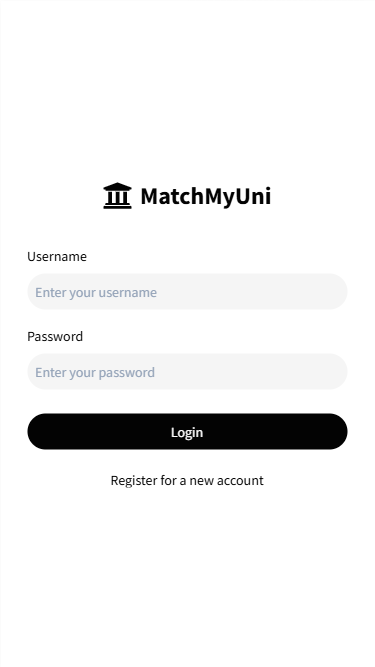
* **Development Tools**:
  + **Frontend**: React Native or Flutter, both provide cross-platform support (iOS and Android).
  + **Backend**: Node.js or Firebase for real-time data handling and minimal maintenance.
  + **Database**: Firestore (Firebase) or PostgreSQL, housing basic user profiles and university links.
* **Matching Algorithm**:
  + **Technique**: Implement a basic rule-based matching system, eventually enhanced with machine learning for personalized recommendations.
* **APIs & Integration**:
  + **Google Firebase**: For user authentication and data storage.
  + **University Websites (Web Scraping or APIs)**: Use approved APIs where possible to pull in university data.
* **User Interface (UI/UX) Tools**:
  + **Figma or Adobe XD**: For UI/UX design and prototyping.
  + **Analytics**: Google Analytics for Firebase or Mixpanel to monitor user interactions and satisfaction.

**7. Deployment**

* **Deployment Platforms**:
  + **Google Play Store and Apple App Store**: For distributing the app to Android and iOS users.
  + **Firebase Hosting**: For managing backend services and potentially a simple web portal.
* **Continuous Deployment**:
  + Use CI/CD tools like **GitHub Actions** or **Bitrise** to streamline app updates and deployment.
* **Beta Testing**:
  + Launch a beta version on platforms like **TestFlight** for iOS and **Google Play Console Beta** for Android, gathering early feedback and identifying bugs before a full launch.

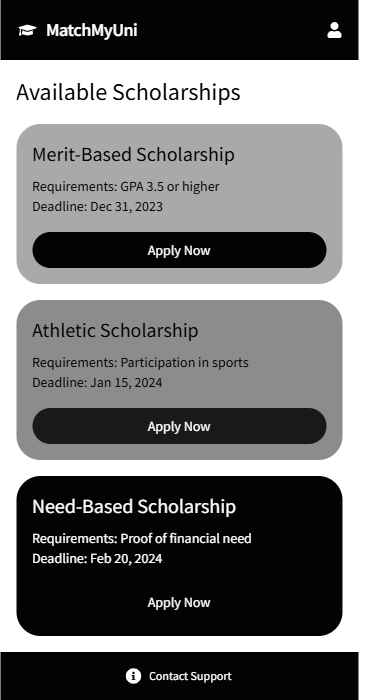
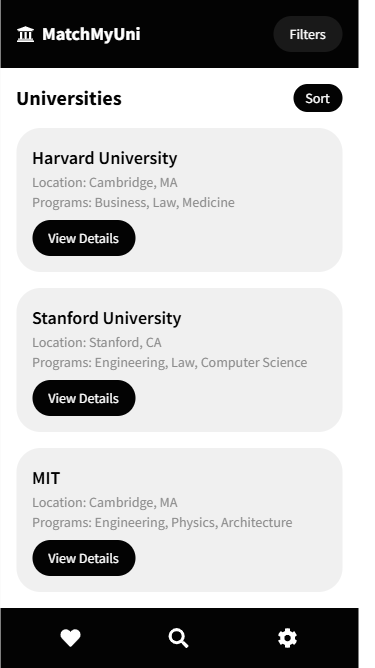
**PROTOTYPES:**

**Login page: Student information:**



Forget Password

**University selection:** **Scholarships:**

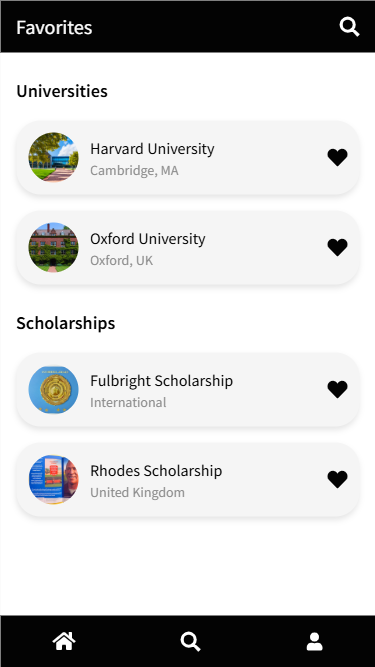


**Feedback**

Other Program

Notification

**Favorites page:**

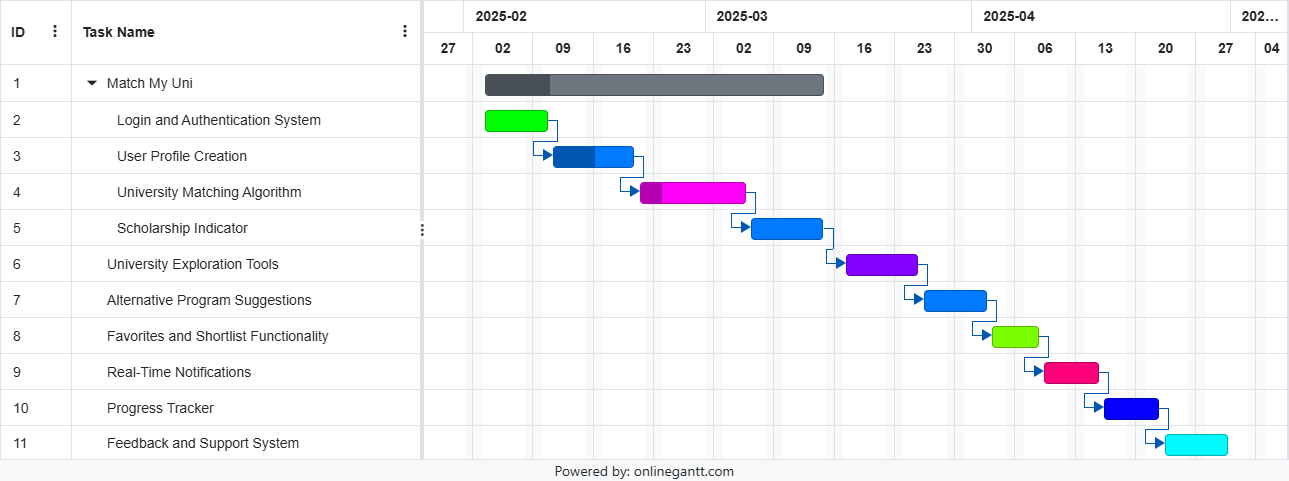


Progress

**WBS (word breakdown system):**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Level** | **WBS** | **Phase/Task Description** | **Start Date** | **End Date** | **Notes** |
| 1 | 1.0 | Login and Authentication System | 01-Feb-2025 | 10-Feb-2025 | Includes secure login, user registration, password reset, and re-login. |
| 2 | 1.1 | User Registration Implementation | 01-Feb-2025 | 04-Feb-2025 | Focuses on creating new accounts. |
| 2 | 1.2 | Login Functionality | 05-Feb-2025 | 06-Feb-2025 | Development of login features. |
| 2 | 1.3 | Password Reset | 07-Feb-2025 | 08-Feb-2025 | Includes email-based password recovery. |
| 2 | 1.4 | Re-login Mechanism | 09-Feb-2025 | 10-Feb-2025 | Ensure users can return and log in to their accounts seamlessly. |
| 1 | 2.0 | User Profile Creation | 11-Feb-2025 | 20-Feb-2025 | Profile setup with personal, academic, and preference details. |
| 2 | 2.1 | Personal Information Input | 11-Feb-2025 | 13-Feb-2025 | Include fields like name, DOB, location, and education type. |
| 2 | 2.2 | Academic and Achievement Details Input | 14-Feb-2025 | 16-Feb-2025 | GPA, test scores, and extracurricular activities. |
| 2 | 2.3 | Preference Setup | 17-Feb-2025 | 20-Feb-2025 | Selection of countries, programs, and budget. |
| 1 | 3.0 | University Matching Algorithm | 21-Feb-2025 | 05-Mar-2025 | Rule-based algorithm for university matching. |
| 2 | 3.1 | Filters for Location | 21-Feb-2025 | 23-Feb-2025 | Add location-based matching criteria. |
| 2 | 3.2 | Tuition Fee Range Filters | 24-Feb-2025 | 26-Feb-2025 | Add fee range to the algorithm. |
| 2 | 3.3 | Program Type Filters | 27-Feb-2025 | 02-Mar-2025 | Integrate program-based filters. |
| 2 | 3.4 | Algorithm Testing | 03-Mar-2025 | 05-Mar-2025 | Test functionality and accuracy. |
| 1 | 4.0 | Scholarship Indicator | 06-Mar-2025 | 15-Mar-2025 | Provide tailored scholarship suggestions. |
| 2 | 4.1 | Scholarship Rule Development | 06-Mar-2025 | 10-Mar-2025 | Create scholarship matching rules. |
| 2 | 4.2 | Integration with User Profile | 11-Mar-2025 | 13-Mar-2025 | Link scholarships to user details. |
| 2 | 4.3 | Scholarship Interface | 14-Mar-2025 | 15-Mar-2025 | Develop UI for displaying scholarships. |
| 1 | 5.0 | University Exploration Tools | 16-Mar-2025 | 25-Mar-2025 | Direct access to websites, programs, and scholarship pages. |
| 2 | 5.1 | University Website Linking | 16-Mar-2025 | 18-Mar-2025 | Enable direct access to university websites. |
| 2 | 5.2 | Program Details Display | 19-Mar-2025 | 22-Mar-2025 | Create detailed program pages. |
| 2 | 5.3 | Scholarship Page Access | 23-Mar-2025 | 25-Mar-2025 | Add links to institution scholarship pages. |
| 1 | 6.0 | Alternative Program Suggestions | 26-Mar-2025 | 02-Apr-2025 | Recommend similar programs based on preferences. |
| 1 | 7.0 | Favorites and Shortlist Functionality | 03-Apr-2025 | 08-Apr-2025 | Allow users to mark and save universities for easy comparison. |
| 1 | 8.0 | Real-Time Notifications | 09-Apr-2025 | 15-Apr-2025 | Notify users about deadlines, updates, and new programs. |
| 1 | 9.0 | Progress Tracker | 16-Apr-2025 | 22-Apr-2025 | Enable users to monitor application progress. |
| 1 | 10.0 | Feedback and Support System | 23-Apr-2025 | 30-Apr-2025 | Basic feedback form or email-based support system. |

**Gantt chart:**

****

**Requirements:**

**Functional requirements:**

1. **Login and Authentication System**
   * Develop a simple, secure login page where students can sign in using only their email and password.
   * Include basic functionality such as user registration and email-based password reset.
   * Include sign-in if the student again comes to use the account after again using.
   * Include forget password if the user forget its password.
2. **User Profile Creation**
   * Allow students to create profiles by entering details such as:
     + Personal information (name, date of birth, location, which education(graduation/masters)).
     + Academic scores (e.g., GPA, standardized test scores).
     + Extracurricular achievements (e.g., sports, competitions).
     + Preferences (countries, programs, budgets).
3. **University Matching Algorithm**
   * Implement a rule-based algorithm that analyzes user profiles to match them with suitable universities.
   * Incorporate filters for location, tuition fee range, program type.
4. **Scholarship Indicator**
   * Provide tailored suggestions for scholarships based on personal, academic, and extracurricular achievements.
5. **University Exploration Tools**
   * Offer direct access to:
     + University websites.
     + Program details.
     + Dedicated scholarship pages for each institution.
6. **Alternative Program Suggestions**
   * Recommend similar programs to users based on their preferences, academic scores, and budget constraints.
7. **Favorites and Shortlist Functionality**
   * Enable users to mark and save universities or programs as favorites for easy comparison later.
8. **Real-Time Notifications**
   * Notify users about:
     + Application deadlines.
     + Scholarship updates.
     + New programs matching their preferences.
9. **Progress Tracker**
   * Allow students to monitor their application process, including submission status and deadlines for shortlisted universities.
10. **Feedback and Support System**

* Integrate a basic feedback form or email-based query system for user feedback and troubleshooting assistance.

**Non-Functional Requirements**

1. **Scalability**:

The app must handle up to 10,000 simultaneous users and support a database with 1 million user profiles.

1. **Response Time**:

University matches should load within 2 seconds for most queries.

1. **Availability**:

Ensure 99.9% uptime to support global users.

1. **Security**:

Use end-to-end encryption (HTTPS/TLS) and store passwords securely with hashing (e.g., bcrypt).

1. **Data Privacy**:

Comply with GDPR/CCPA, ensuring user data protection and offering data deletion upon request.

1. **Usability**:

Provide an intuitive, user-friendly interface with minimal navigation steps for completing key tasks.

1. **Accessibility**:

Adhere to accessibility standards (WCAG 2.1) to support users with disabilities.

1. **Device Compatibility**:

Ensure smooth functionality on Android (6.0+) and iOS (13.0+) devices.

1. **Backup and Recovery**:

Implement automated backups with recovery capabilities within 4 hours of a major failure.

1. **Third-Party Integration**:

Support seamless integration with APIs for university and scholarship data retrieval.

**List of Actors:**

**Primary Actors:**

1. **Student:**
   * Creates and manages their profile.
   * Searches for universities and programs.
   * Applies to universities and tracks applications.
   * Receives notifications and updates.
   * Provides feedback and support requests.

**Secondary Actors:**

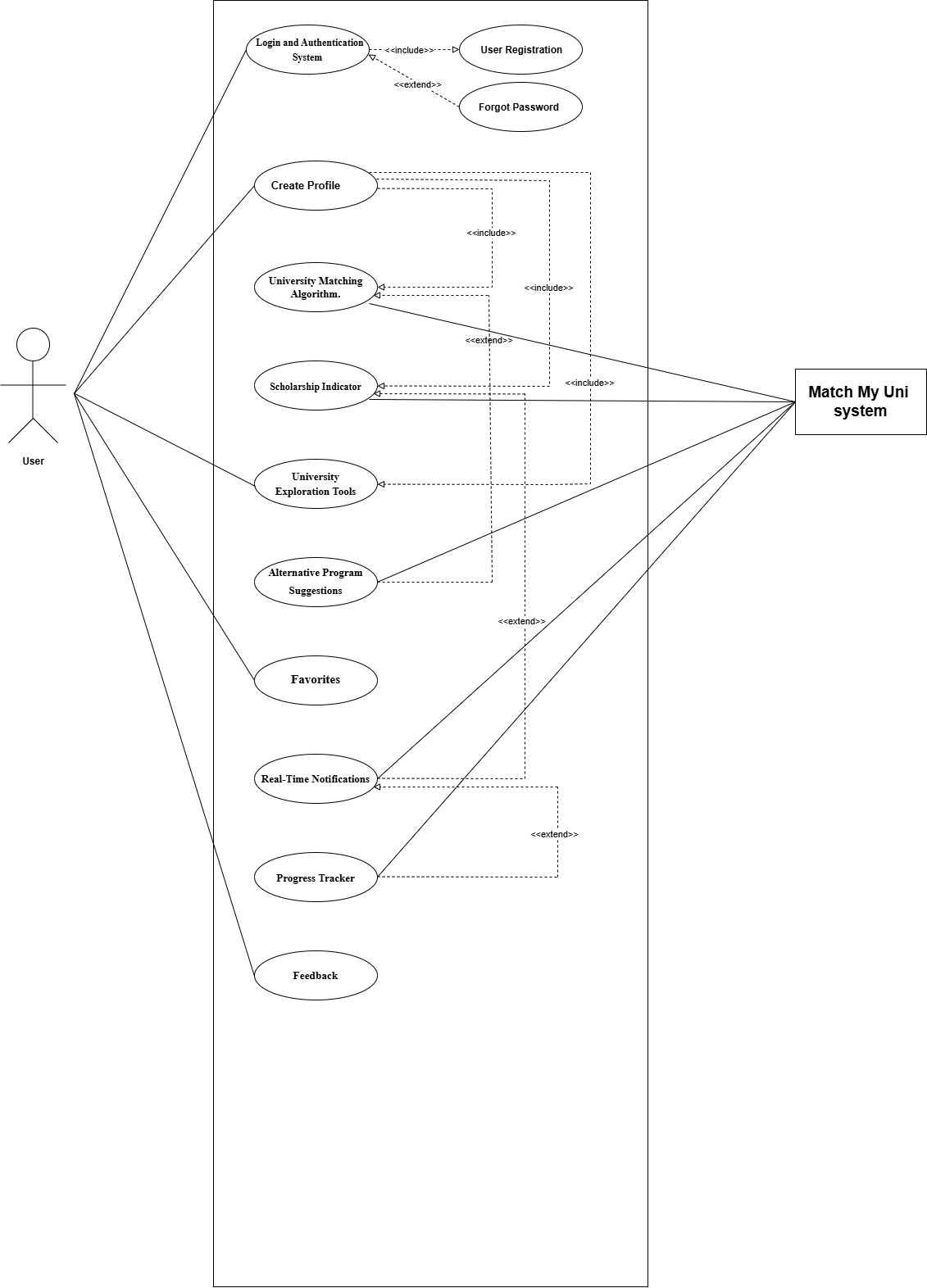
1. **University:**
   * (Indirectly) Provides information about their programs, deadlines, and scholarship opportunities.
   * (Indirectly) May be updated in the system's database.
2. **System Administrator:**
   * Manages the system's operations, data, and security.
   * Updates university and program information.
   * Troubleshoots issues and provides technical support.

**Tertiary Actors:**

1. **Third-Party Services:**
   * Provide additional services like email notifications, or data analytics.

**List of use cases:**

1. **User Registration:**
   * Create a new user account.
   * Verify email address.
   * Set a password.
   * Forget password.
2. **Profile Creation:**
   * Provide personal information (name, DOB, location).
   * Input academic details (GPA, test scores).
   * List extracurricular activities and achievements.
   * Specify preferences (countries, programs, budget).
3. **University Search:**
   * Search for universities based on location, program, tuition fee, language, or other criteria.
   * Filter results based on user preferences.
4. **University Matching:**
   * Utilize the matching algorithm to suggest suitable universities.
   * Receive personalized recommendations based on profile data.
5. **Scholarship Search:**
   * Find scholarships based on academic merit, financial need, or specific criteria.
   * Receive tailored scholarship recommendations.
6. **University Exploration:**
   * Access university websites directly.
   * View program details and course offerings.
   * Explore scholarship opportunities for specific universities.
7. **Favorite and Shortlist:**
   * Save universities and programs for future reference.
   * Create a personalized shortlist for comparison.
8. **Notification and Alerts:**
   * Receive notifications for application deadlines, scholarship updates, and personalized recommendations.
9. **Application Tracking:**
   * Monitor application status and deadlines.
   * Receive reminders for required documents and application fees.
10. **Feedback and Support:**
    * Provide feedback on the system's performance and features.
    * Contact support for assistance with issues or querie

**Main use case:  
**

**Login:**

Match My uni System

User

**Forget Password:**

Match My uni System

User

**Use Case Table for Login Process**

|  |  |
| --- | --- |
| **Field** | **Details** |
| **Use Case Name** | Login |
| **Created By** | khadija Asif |
| **Date Created** | 24-05-24 |
| **Last Updated By** | 03-06-24 |
| **Last Revision Date** | 24-06-24 |
| **Actors** | **Primary**: User, **Secondary**: System |
| **Description** | This use case allows a user to register by creating a new account with an email and password, log in using their credentials (email and password), and reset their password if forgotten. The system handles login authentication by verifying the user's email and password, and facilitates password recovery when necessary. |
| **Trigger** | |  | | --- | |  |  |  | | --- | | The user attempts to log in, register, or reset their password | |
| **Preconditions** | 1. The user must have a valid email for registration. 2. The user must be logged in or attempt to reset their password if they forgot it. |
| **Post conditions** | 1. The user is successfully logged in or redirected to the password reset page. 2. If password reset is triggered, the user receives a password reset link. |
| **Normal Flow** | 1. The user navigates to the login page. 2. The user enters their email and password. 3. The system verifies the credentials (email and password). 4. If credentials are valid, the user is granted access to the dashboard. 5. If credentials are invalid, the system displays an error message prompting re-entry. |
| **Alternative Flows** | 1. **Forgot Password**: The user clicks "Forgot Password", enters their email, and receives a reset link. 2. **Re-Login**: If the user logs out and returns later, they enter their email and password to log back in. |
| **Exceptions** | 1. If the email/password is invalid, the system displays an error message. 2. If the email is not registered, the system prompts the user to check their credentials or register. |
| **Includes** | |  | | --- | |  |  |  | | --- | | 1. Register Account (for first-time users). 2. Forgot Password (if user cannot remember password) | |
| **Frequency of Use** | |  | | --- | | High (frequent use by students logging in and out of the system). |  |  | | --- | |  | |
| **Special Requirements** | 1. Password must meet security criteria (e.g., minimum length, complexity). 2. Email must be unique for each registration. |
| **Assumptions** | 1. The user has internet access. 2. The email entered by the user is valid. |
| **Notes and Issues** | 1. The system must securely hash passwords before storing them. 2. Ensure email verification for registration and password reset to avoid fraudulent activity. |
| **Use Case ID** | UC-1 |

**Profile setup:**

Match My uni System

User

**Use Case Table for Profile setup:**

|  |  |
| --- | --- |
| **Field** | **Details** |
| **Use Case Name** | Login |
| **Created By** | khadija Asif |
| **Date Created** | 24-05-24 |
| **Last Updated By** | 03-06-24 |
| **Last Revision Date** | 24-06-24 |
| **Actors** | **Primary**: User, **Secondary**: System |
| **Description** | |  | | --- | | This use case allows a student to create a profile by entering personal information, academic scores, extracurricular achievements, and preferences (countries, programs, budgets). | |
| **Trigger** | |  |  | | --- | --- | |  | The user initiates the profile creation process by signing up or accessing their profile section. | |
| **Preconditions** | |  |  | | --- | --- | |  | 1. The user must be logged in. 2. The user must have access to the profile creation section. | |
| **Post conditions** | |  |  | | --- | --- | |  | 1. The user's profile is successfully created with all the entered details. 2. The profile data is stored securely in the system. | |
| **Normal Flow** | |  |  | | --- | --- | |  | 1. The user navigates to the profile creation page. 2. The user enters personal information (name, date of birth, location, education level). 3. The user provides academic scores (GPA, standardized test scores). 4. The user adds extracurricular achievements (sports, competitions). 5. The user sets preferences (countries, programs, budgets). 6. The system validates and stores the data. 7. A success message is displayed upon completion. | |
| **Alternative Flows** | |  |  | | --- | --- | |  | 1. If the user misses any required field, the system prompts them to fill in the missing information. 2. If the user decides to skip a section (e.g., extracurricular achievements), the system saves the data entered up to that point and allows the user to complete it later. | |
| **Exceptions** | |  |  | | --- | --- | |  | 1. Invalid or incomplete data (e.g., incorrect GPA format or missing name) leads to an error message and prompts the user to correct the information. 2. If the user is not logged in or doesn’t have access, the system redirects them to the login page or shows an access error. | |
| **Includes** | |  |  | | --- | --- | | **Includes** | 1. Profile Update (if the user wants to edit their profile later). | |
| **Frequency of Use** | |  |  | | --- | --- | |  | Medium (primarily used during profile creation, but may be accessed for updates). | |
| **Special Requirements** | |  |  | | --- | --- | |  | 1. The data entered by the user should be validated for format (e.g., GPA, dates). 2. Ensure that personal data is stored securely and in compliance with privacy regulations (e.g., GDPR). | |
| **Assumptions** | |  |  | | --- | --- | |  | 1. The user has access to the internet. 2. The user will provide accurate information. | |
| **Notes and Issues** | |  |  | | --- | --- | |  | 1. Ensure that no sensitive information (e.g., password) is stored unencrypted. 2. There should be a mechanism to handle large input data (e.g., for extracurricular achievements). | |
| **Use Case ID** | UC-2 |

**University Matching Algorithm:**

Match My uni System

user

**Use Case Table for matching algorithm:**

|  |  |
| --- | --- |
| **Field** | **Details** |
| **Use Case Name** | Login |
| **Created By** | khadija Asif |
| **Date Created** | 24-05-24 |
| **Last Updated By** | 03-06-24 |
| **Last Revision Date** | 24-06-24 |
| **Actors** | **Primary**: User, **Secondary**: System |
| **Description** | |  | | --- | | This use case allows a student to input their preferences and profile data to match with suitable universities based on location, tuition fees, program type, and language of instruction. The system uses a rule-based algorithm to process the data and provide a list of matching universities. | |
| **Trigger** | |  |  | | --- | --- | |  | The user submits their profile and preferences to the system for matching. | |
| **Preconditions** | |  | | --- | | 1. The user has successfully created a profile. 2. The user has selected preferences for location, program type, language, and budget. | |
| **Post conditions** | |  | | --- | | 1. The system has successfully matched the user to suitable universities. 2. A list of universities is displayed based on the user's profile and preferences. | |
| **Normal Flow** | |  |  | | --- | --- | |  | 1. The user enters their profile information, including preferences for location, program type, budget, and language of instruction. 2. The system applies a rule-based algorithm to match universities based on these preferences. 3. The system filters the universities based on the selected criteria (e.g., tuition fee range, program type, language of instruction). 4. The system displays the list of universities that match the user's profile. | |
| **Alternative Flows** | |  |  | | --- | --- | |  | 1. **No Matches Found**: If no universities match the user's preferences, the system notifies the user and suggests modifying their preferences. 2. **Partial Match**: If the user’s preferences partially match, the system may show a list of universities that are close matches and provide alternative suggestions. | |
| **Exceptions** | |  |  | | --- | --- | |  | 1. If the user does not enter enough information (e.g., no preferences set), the system prompts the user to fill in missing details. 2. If the system fails to retrieve data or encounters a processing error, an error message is shown to the user. | |
| **Includes** | |  |  | | --- | --- | |  | 1. **Profile Information Validation** (to ensure data is correct before matching). 2. **Alternative Program Suggestions** (if no direct match is found). | |
| **Frequency of Use** | |  |  | | --- | --- | |  | High (as this feature will be accessed regularly after profile creation and preference updates). | |
| **Special Requirements** | |  |  | | --- | --- | |  | 1. The matching algorithm must be able to handle complex queries involving multiple criteria. 2. The system should consider a range of variables like tuition fees, program types, and geographical preferences. 3. The system should be optimized for performance to provide results quickly, even with a large database of universities. | |
| **Assumptions** | |  |  | | --- | --- | |  | 1. The user will provide accurate and up-to-date preferences. 2. University data (including fees, programs, and locations) is regularly updated. | |
| **Notes and Issues** | |  |  | | --- | --- | |  | 1. The algorithm should be flexible enough to adapt to new types of user preferences (e.g., new program types or regions). 2. Ensure that the system can handle scenarios where universities may have missing or incomplete data for certain criteria. | |
| **Use Case ID** | UC-3 |

**Scholarship Indicator:**

Match My uni System

User

**Use Case Table for Scholarship Indicator:**

|  |  |
| --- | --- |
| **Field** | **Details** |
| **Use Case Name** | Login |
| **Created By** | khadija Asif |
| **Date Created** | 24-05-24 |
| **Last Updated By** | 03-06-24 |
| **Last Revision Date** | 24-06-24 |
| **Actors** | **Primary**: User, **Secondary**: System |
| **Description** | |  | | --- | | This use case allows a student to receive tailored scholarship suggestions based on their personal, academic, and extracurricular achievements. The system analyzes the user's profile and matches them with relevant scholarships. | |
| **Trigger** | |  | | --- | | The user inputs their profile data, and the system processes this information to match scholarships. | |
| **Preconditions** | |  | | --- | | 1. The user has entered their profile data (personal, academic, and extracurricular achievements). 2. The system has access to a database of available scholarships. | |
| **Post conditions** | |  | | --- | | 1. The user receives a list of scholarships that match their profile. 2. The user can access details for each scholarship suggestion. | |
| **Normal Flow** | |  |  | | --- | --- | |  | 1. The user navigates to the scholarship section of the app. 2. The user inputs or updates their personal, academic, and extracurricular information. 3. The system processes the data and matches it with available scholarships. 4. The system displays a list of scholarships based on the user’s achievements. | |
| **Alternative Flows** | |  |  | | --- | --- | |  | 1. **No Match Found**: If the system doesn’t find any matching scholarships, it informs the user and suggests reviewing their profile information. 2. **Partial Match**: If there is no perfect match, the system may suggest scholarships that are close or offer partial eligibility. | |
| **Exceptions** | |  |  | | --- | --- | |  | 1. If the user has incomplete or invalid data in their profile (e.g., missing GPA or achievements), the system prompts the user to complete or correct the information. 2. If there is an error in processing, the system displays an error message to the user. | |
| **Includes** | |  |  | | --- | --- | |  | 1. **Profile Data Validation** (ensuring that all necessary data is entered for accurate matching). | |
| **Frequency of Use** | |  |  | | --- | --- | |  | Medium (used when the user is actively looking for scholarship opportunities). | |
| **Special Requirements** | |  |  | | --- | --- | |  | 1. The system should have an up-to-date scholarship database with eligibility criteria. 2. The system must prioritize scholarships that match the user's achievements. | |
| **Assumptions** | |  |  | | --- | --- | |  | 1. The user has access to the internet. 2. The scholarships database is regularly updated with new opportunities. | |
| **Notes and Issues** | |  |  | | --- | --- | |  | 1. The system should consider various scholarship types (e.g., merit-based, need-based, sports-related) when matching. 2. Ensure that the user’s data (e.g., GPA, extracurricular activities) is accurately assessed when making suggestions. | |
| **Use Case ID** | UC-4 |

**University Exploration Tools:**

Match My uni System

User

**Use Case Table for matching algorithm:**

|  |  |
| --- | --- |
| **Field** | **Details** |
| **Use Case Name** | Login |
| **Created By** | khadija Asif |
| **Date Created** | 24-05-24 |
| **Last Updated By** | 03-06-24 |
| **Last Revision Date** | 24-06-24 |
| **Actors** | **Primary**: User, **Secondary**: System |
| **Description** | |  | | --- | | This use case allows a student to access detailed information about universities, including their official websites, program offerings, and dedicated scholarship pages. The system facilitates exploration by providing links to external pages for further information. | |
| **Trigger** | |  | | --- | | The user selects a university and wants to explore its website, programs, and scholarships. | |
| **Preconditions** | |  | | --- | | 1. The user has selected a university from the list of matched universities. 2. The system has access to external links (university websites, program details, scholarship pages). | |
| **Post conditions** | |  |  | | --- | --- | |  | 1. The user is directed to the university’s official website. 2. The user is able to view detailed information about programs and scholarships. | |
| **Normal Flow** | |  | | --- | | 1. The user selects a university from the list of matched results. 2. The system provides links to the university’s official website, program details, and scholarship pages. 3. The user clicks on the appropriate link to view more detailed information about the university, programs, or scholarships. | |
| **Alternative Flows** | |  |  | | --- | --- | |  | 1. **No Program Details Available**: If no program details are available for a specific university, the system notifies the user and provides an alternative option. 2. **No Scholarship Information**: If the scholarship page is unavailable, the system may suggest contacting the university directly or searching for external scholarship resources. | |
| **Exceptions** | |  | | --- | | 1. If the external links (university websites, program pages, scholarship pages) are broken or unavailable, the system shows an error message and provides alternative suggestions. 2. If the user is not connected to the internet, the system will notify the user of the need for an internet connection to access external links. | |
| **Includes** | |  | | --- | | 1. **Displaying University Details**: Display the university’s website, program information, and scholarship links. 2. **Navigating to External Links**: The system ensures smooth navigation to external sites. | |
| **Frequency of Use** | |  |  | | --- | --- | |  | Medium (used when a user is interested in further exploring a university or its offerings). | |
| **Special Requirements** | |  |  | | --- | --- | |  | 1. The system must ensure that external links are up-to-date and functional. 2. The system should be able to handle large amounts of data (university websites, program details, etc.) without performance degradation. | |
| **Assumptions** | |  | | --- | | 1. The user has internet access to visit the university websites. 2. The universities maintain up-to-date websites and scholarship pages. | |
| **Notes and Issues** | |  |  | | --- | --- | |  | 1. Some universities may not have detailed information on scholarships or programs available online. 2. Consider integrating APIs from universities to pull real-time data on programs and scholarships. | |
| **Use Case ID** | UC-5 |

**Alternative Program Suggestions**:

Match My uni System

User

**Use Case Table for Alternative Program Suggestions:**

|  |  |
| --- | --- |
| **Field** | **Details** |
| **Use Case Name** | Login |
| **Created By** | khadija Asif |
| **Date Created** | 24-05-24 |
| **Last Updated By** | 03-06-24 |
| **Last Revision Date** | 24-06-24 |
| **Actors** | **Primary**: User, **Secondary**: System |
| **Description** | |  | | --- | | This use case allows the system to recommend similar programs to the user based on their academic scores, preferences (such as program type, location, budget), and financial constraints. The system processes the user’s profile data and provides a list of suggested programs. | |
| **Trigger** | |  |  | | --- | --- | |  | The user provides their academic scores, preferences, and budget information, and the system processes this data to recommend similar programs. | |
| **Preconditions** | |  |  | | --- | --- | |  | 1. The user has entered their academic scores, preferences, and budget. 2. The system is capable of processing this data to generate recommendations. | |
| **Post conditions** | |  |  | | --- | --- | |  | 1. The user receives a list of similar programs based on their profile. 2. The user can review and explore these suggested programs. | |
| **Normal Flow** | |  |  | | --- | --- | |  | 1. The user inputs their academic scores, preferences (program type, location), and budget. 2. The system processes the data and matches the user with suitable programs. 3. The system displays the recommended programs to the user. | |
| **Alternative Flows** | |  |  | | --- | --- | |  | 1. **No Similar Programs**: If no programs match, the system notifies the user and suggests adjusting their profile information (e.g., expanding budget, location). 2. **Partial Match**: The system recommends programs that partially meet the user’s criteria, explaining the areas where the programs don't fully match. | |
| **Exceptions** | |  |  | | --- | --- | |  | 1. If the user’s profile is incomplete or invalid (e.g., missing academic scores), the system prompts the user to provide the necessary information. 2. If the system experiences a failure in processing the recommendations, it displays an error message to the user. | |
| **Includes** | |  |  | | --- | --- | |  | 1. **Profile Validation**: The system validates the provided academic scores, preferences, and budget. 2. **Program Matching**: The system uses the entered data to match programs based on the user’s criteria. | |
| **Frequency of Use** | |  |  | | --- | --- | |  | Medium (used when the user is seeking alternative program options). | |
| **Special Requirements** | |  | | --- | | 1. The system must have access to up-to-date and comprehensive program data. 2. The system should provide filters and sorting options to help the user further narrow down their recommendations. | |
| **Assumptions** | |  |  | | --- | --- | |  | 1. The user has an internet connection to view recommended programs. 2. The system is connected to a database containing program information that includes the program type, tuition fees, location, and other relevant criteria. | |
| **Notes and Issues** | |  |  | | --- | --- | |  | 1. If no programs match the user’s criteria, the system should provide suggestions for adjusting preferences (e.g., increasing the budget, considering other locations). 2. Consider offering options for users to set alerts for new matching programs or updates. | |
| **Use Case ID** | UC-6 |

**Favorites and Shortlist Functionality**:

Match My uni System

User

**Remove from favorites** :

Match My uni System

User

**Use Case Table for Favorites and Shortlist Functionality:**

|  |  |
| --- | --- |
| **Field** | **Details** |
| **Use Case Name** | Login |
| **Created By** | khadija Asif |
| **Date Created** | 24-05-24 |
| **Last Updated By** | 03-06-24 |
| **Last Revision Date** | 24-06-24 |
| **Actors** | **Primary**: User, **Secondary**: System |
| **Description** | |  | | --- | | This use case allows the user to click the "Add to Favorites" button to save a university or program to their shortlist, which can later be viewed and compared. | |
| **Trigger** | |  |  | | --- | --- | |  | The user clicks the "Add to Favorites" button next to a program or university. | |
| **Preconditions** | |  |  | | --- | --- | |  | 1. The user is logged into the system. 2. The user is browsing through available universities or programs. | |
| **Post conditions** | |  |  | | --- | --- | |  | 1. The selected university or program is added to the user’s shortlist. 2. The user can view and compare the programs/universities in the shortlist. | |
| **Normal Flow** | |  | | --- | | 1. The user clicks the "Add to Favorites" button next to a university or program. 2. The system adds the selected program or university to the user’s shortlist. 3. The user can view the shortlisted programs/universities later. | |
| **Alternative Flows** | |  | | --- | | 1. **Removing from Shortlist**: The user can click on "Remove from Shortlist" to delete a program/university from their list. 2. **Viewing the Shortlist**: The user can view their shortlist at any time by navigating to a dedicated section. | |
| **Exceptions** | |  |  | | --- | --- | |  | 1. If there’s an error in saving the program/university to the shortlist (e.g., database issue), the user is notified with an error message. 2. If the user is logged out while adding to the favorites, they are prompted to log in again before proceeding. | |
| **Includes** | |  |  | | --- | --- | |  | 1. **Data Validation**: The system validates that the program/university is successfully added to the shortlist. 2. **View Shortlist**: The user can access their list of saved universities or programs from a dedicated page. | |
| **Frequency of Use** | |  | | --- | | High (used frequently as users explore and save programs or universities they are interested in). | |
| **Special Requirements** | |  | | --- | | 1. The system must allow users to access their favorites even after logging out. 2. The system should provide an intuitive interface to add or remove universities or programs from the shortlist. | |
| **Assumptions** | |  |  | | --- | --- | |  | 1. The user has a valid internet connection. 2. The system is working properly without database issues or login errors. | |
| **Notes and Issues** | |  | | --- | | 1. Consider allowing users to categorize their shortlist or add custom labels to the programs/universities (e.g., "Interested," "Favorite"). 2. The system should offer an easy way to compare shortlisted programs. | |
| **Use Case ID** | UC-7 |

**Real-Time Notifications**:

Match My uni System

System

**Use Case Table for real time notifications:**

|  |  |
| --- | --- |
| **Field** | **Details** |
| **Use Case Name** | Login |
| **Created By** | khadija Asif |
| **Date Created** | 24-05-24 |
| **Last Updated By** | 03-06-24 |
| **Last Revision Date** | 24-06-24 |
| **Actors** | **Primary**: User, **Secondary**: System |
| **Description** | |  | | --- | | The system sends real-time notifications to users about application deadlines, scholarship updates, and new programs matching their preferences. | |
| **Trigger** | |  |  | | --- | --- | |  | The system detects updates or relevant events (e.g., approaching deadlines, new scholarships, new programs). | |
| **Preconditions** | |  |  | | --- | --- | |  | 1. The user is registered and logged into the system. 2. The user has set preferences for receiving notifications. 3. The system is connected to the necessary data sources for deadlines, scholarships, and programs. | |
| **Post conditions** | |  |  | | --- | --- | |  | 1. The user receives the notification. 2. The system records that the notification was delivered to avoid duplicates. | |
| **Normal Flow** | |  |  | | --- | --- | |  | 1. The system checks for updates on application deadlines, scholarships, and new programs. 2. When a relevant event occurs, the system generates a notification. 3. The system delivers the notification to the user via the app or email. 4. The user views the notification and may take action (e.g., apply to a program, check a scholarship). | |
| **Alternative Flows** | |  |  | | --- | --- | |  | 1. **Disable Notifications**: If the user disables notifications, the system stops sending updates. 2. **Delayed Notifications**: If the user is offline, the system queues notifications for later delivery. | |
| **Exceptions** | |  |  | | --- | --- | |  | 1. If the system cannot fetch updates (e.g., API failure), notifications are delayed. 2. If the user has no preferences set, the system does not send notifications. | |
| **Includes** | |  |  | | --- | --- | |  | 1. **Preference Management**: The user sets preferences for types of notifications they want to receive. 2. **Data Validation**: The system verifies the accuracy of event data before sending notifications. | |
| **Frequency of Use** | |  | | --- | | Frequent; triggered whenever there are updates. | |
| **Special Requirements** | |  |  | | --- | --- | |  | 1. Notifications should be real-time and delivered without significant delays. 2. The user should have an option to customize notification preferences. | |
| **Assumptions** | |  |  | | --- | --- | |  | 1. The system has access to reliable and up-to-date data. 2. The user’s device is capable of receiving notifications. | |
| **Notes and Issues** | |  |  | | --- | --- | |  | 1. Ensure that notifications are not repetitive or irrelevant. 2. Provide an option for users to view missed notifications in the app. | |
| **Use Case ID** | UC-8 |

**Progress Tracker**:

Match My uni System

User

**Use Case Table for progress tracker:**

|  |  |
| --- | --- |
| **Field** | **Details** |
| **Use Case Name** | Login |
| **Created By** | khadija Asif |
| **Date Created** | 24-05-24 |
| **Last Updated By** | 03-06-24 |
| **Last Revision Date** | 24-06-24 |
| **Actors** | **Primary**: User, **Secondary**: System |
| **Description** | |  | | --- | | This use case allows students to monitor the progress of their university applications, including submission status and approaching deadlines for shortlisted universities. | |
| **Trigger** | |  |  | | --- | --- | |  | The user views the progress tracker page or receives a notification about an update or deadline. | |
| **Preconditions** | |  |  | | --- | --- | |  | 1. The user is logged into the system. 2. The user has shortlisted universities or submitted applications. | |
| **Post conditions** | |  |  | | --- | --- | |  | 1. The user can view the current status of applications. 2. The system logs any updates made to application statuses. | |
| **Normal Flow** | |  |  | | --- | --- | |  | 1. The user accesses the progress tracker. 2. The system displays a list of shortlisted universities and their application statuses. 3. The user can view detailed information about submission statuses and deadlines. 4. The user updates the status if needed (e.g., marking an application as submitted). | |
| **Alternative Flows** | |  |  | | --- | --- | |  | 1. **No Shortlisted Universities**: If the user has not shortlisted universities, the system displays a message prompting the user to add universities. 2. **No Updates Available**: If there are no changes in application statuses, the system displays the last updated statuses. | |
| **Exceptions** | |  |  | | --- | --- | |  | 1. If the system cannot fetch data (e.g., due to a network issue), the user is notified to try again later. 2. If the user attempts to update the status for a university not in their shortlist, the system blocks the action. | |
| **Includes** | |  |  | | --- | --- | |  | 1. **Notification Management**: The system alerts the user to approaching deadlines. 2. **Data Synchronization**: The system synchronizes updated statuses with the server. | |
| **Frequency of Use** | |  |  | | --- | --- | |  | Frequent; users may check progress regularly during application periods. | |
| **Special Requirements** | |  |  | | --- | --- | |  | 1. The progress tracker must be visually intuitive, showing clear status indicators. 2. Notifications for deadlines should be sent promptly. | |
| **Assumptions** | |  |  | | --- | --- | |  | 1. The system has access to accurate status data for each application. 2. Users update their application statuses as needed. | |
| **Notes and Issues** | |  |  | | --- | --- | |  | 1. Ensure the tracker supports multiple universities and their respective deadlines. 2. Provide an option to export or print the progress for offline use. | |
| **Use Case ID** | UC-9 |

**Feedback and Support System**:

Match My uni System

User

**Use Case Table for feedback and support:**

|  |  |
| --- | --- |
| **Field** | **Details** |
| **Use Case Name** | Login |
| **Created By** | khadija Asif |
| **Date Created** | 24-05-24 |
| **Last Updated By** | 03-06-24 |
| **Last Revision Date** | 24-06-24 |
| **Actors** | **Primary**: User, **Secondary**: System |
| **Description** | |  | | --- | | This use case allows students to submit feedback about the app or request support for issues encountered. The system acknowledges submissions and forwards them for review or resolution. | |
| **Trigger** | |  |  | | --- | --- | |  | The user accesses the feedback or support section of the application. | |
| **Preconditions** | |  | | --- | | 1. The user is logged into the application. 2. The system is operational and capable of processing requests. | |
| **Post conditions** | |  |  | | --- | --- | |  | 1. The system logs feedback or queries. 2. A confirmation message is sent to the user. 3. Support queries are forwarded to the relevant team for troubleshooting. | |
| **Normal Flow** | |  |  | | --- | --- | |  | 1. The user navigates to the feedback/support section. 2. The user fills out a form with details (e.g., feedback or issue description). 3. The user submits the form. 4. The system acknowledges receipt of the submission. 5. The system logs the submission and forwards support requests as needed. | |
| **Alternative Flows** | |  |  | | --- | --- | |  | 1. **Feedback without Login**: If the user is not logged in, they can submit anonymous feedback. 2. **Support Offline**: If the system cannot forward the request immediately, it queues the request for later. | |
| **Exceptions** | |  |  | | --- | --- | |  | 1. If the form submission fails (e.g., due to network issues), the user is prompted to try again. 2. If the user provides incomplete information, the system requests additional details before accepting the submission. | |
| **Includes** | |  |  | | --- | --- | |  | 1. **Form Validation**: Ensures that the user submits complete and valid information. 2. **Acknowledgment System**: Sends a confirmation message upon successful submission. | |
| **Frequency of Use** | |  |  | | --- | --- | |  | Infrequent; typically used when users encounter issues or wish to provide feedback. | |
| **Special Requirements** | |  |  | | --- | --- | |  | 1. The form must be simple and intuitive to encourage user participation. 2. Provide options for attaching screenshots or documents for troubleshooting. | |
| **Assumptions** | |  |  | | --- | --- | |  | 1. The system has a reliable mechanism to store and forward feedback and queries. 2. The user has access to the internet while submitting feedback or support requests. | |
| **Notes and Issues** | |  |  | | --- | --- | |  | 1. Ensure the privacy of feedback and queries, especially for sensitive issues. 2. Provide a tracking mechanism for support requests if possible. | |
| **Use Case ID** | UC-10 |

**Extend is on next page.**

**Extend use case:**

<<extend>>

<<extend>>

<<extend>>

<<extend>>

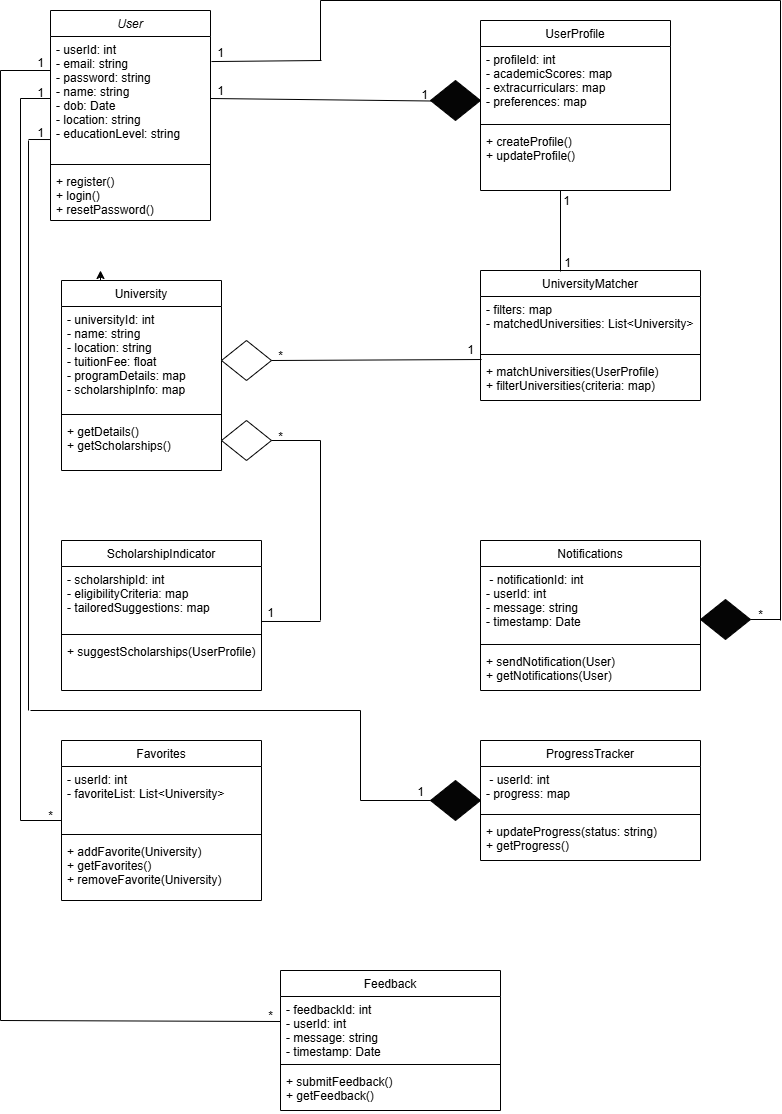
<<extend>>

User

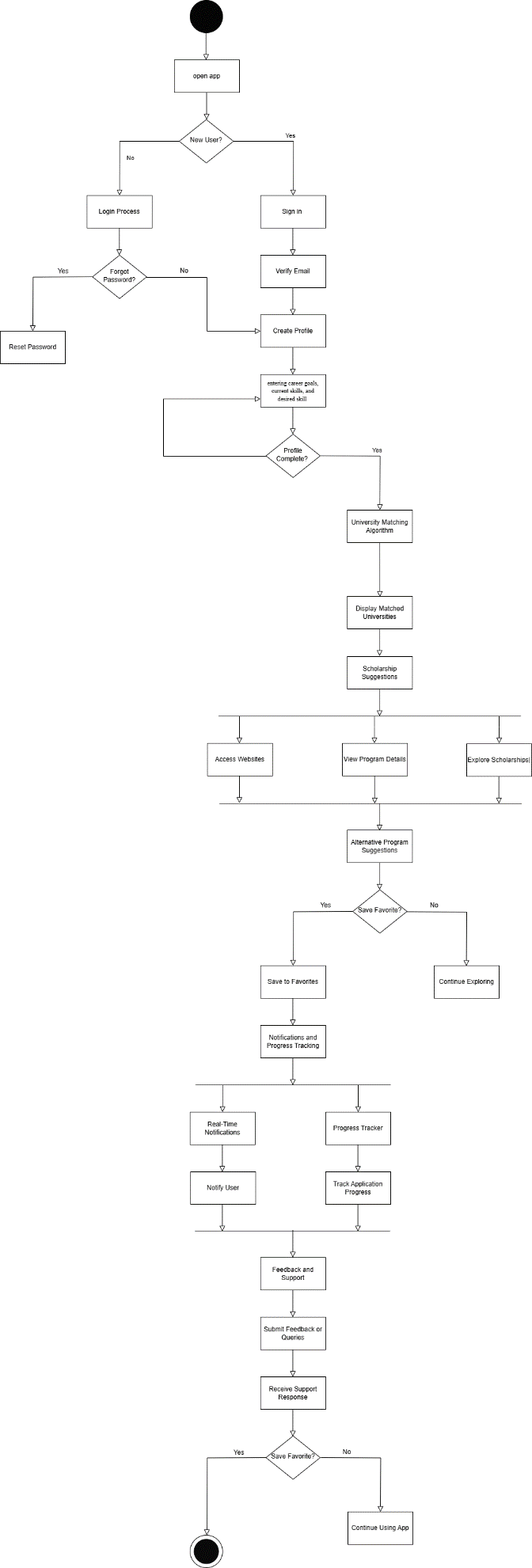
<<extend>>

<<extend>>

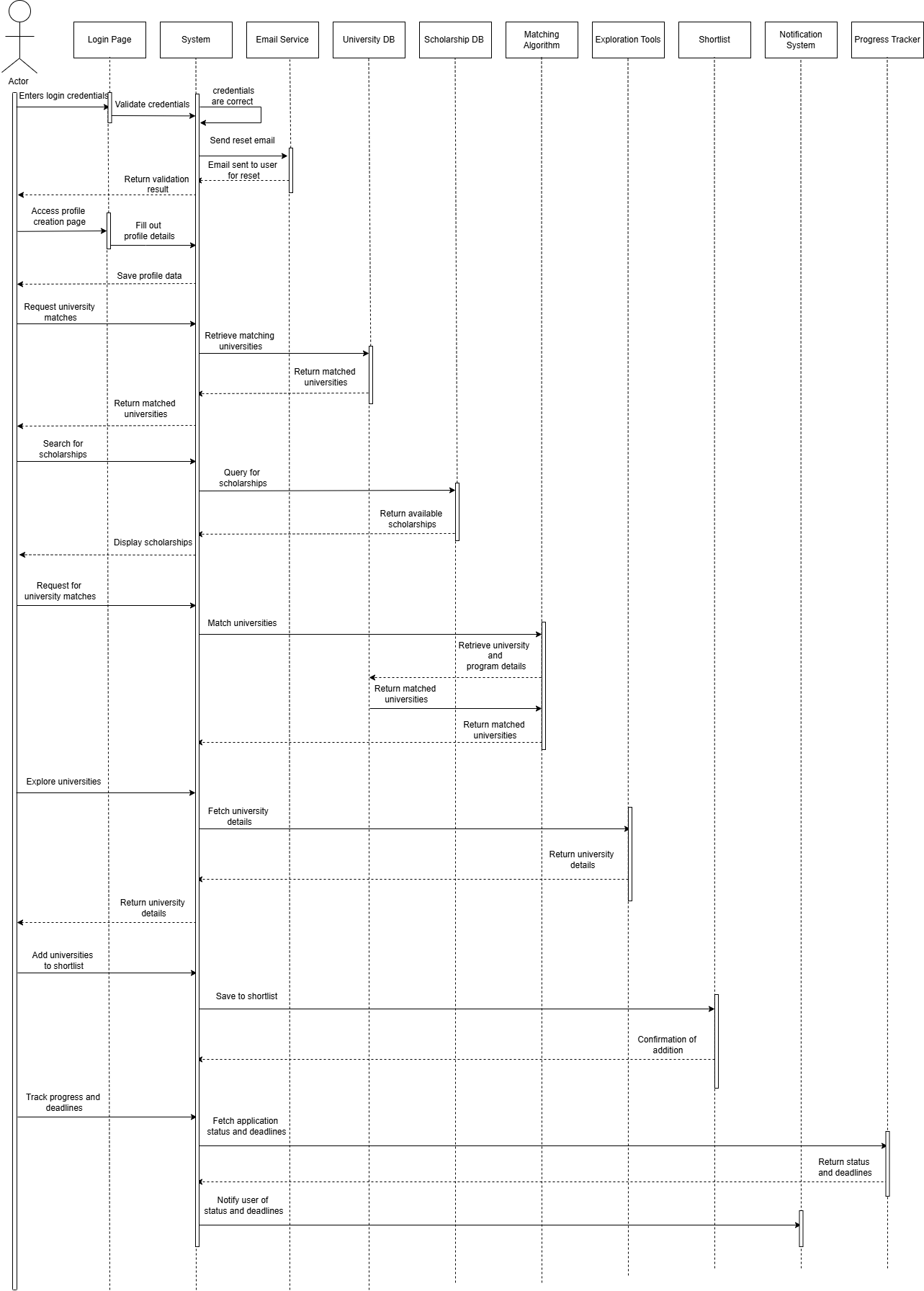
**Class diagram:**



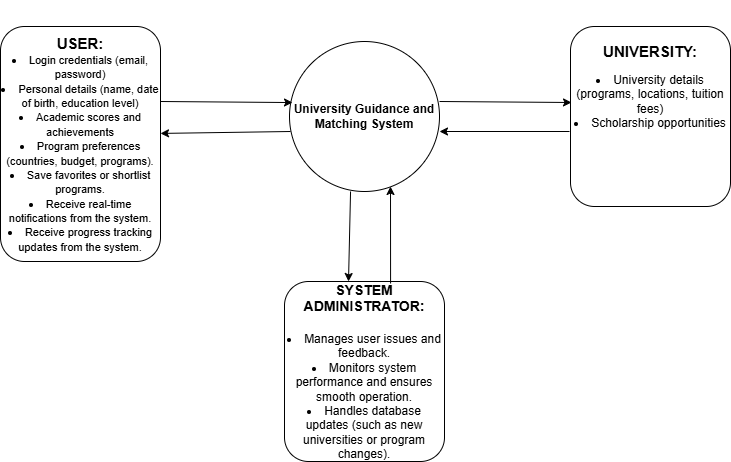
**Activity Diagram**

****

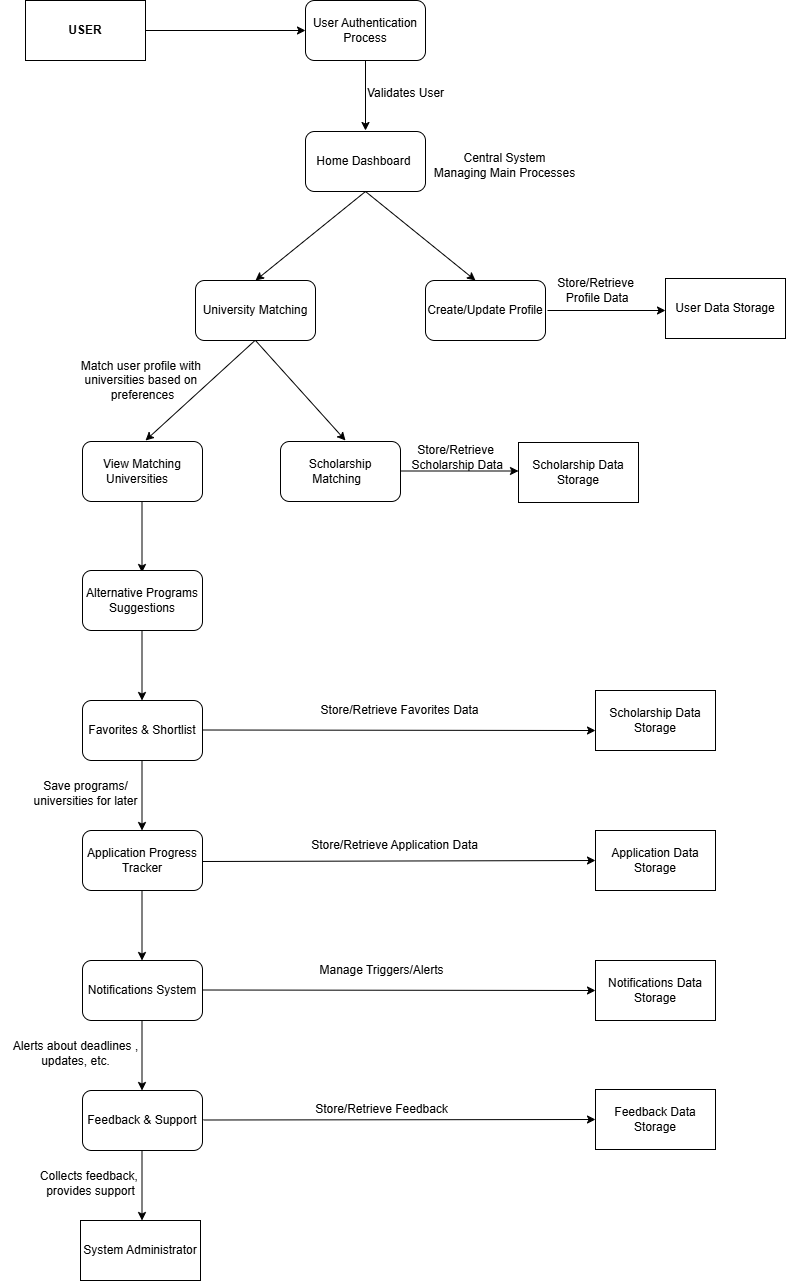
**Sequence diagram:**

****

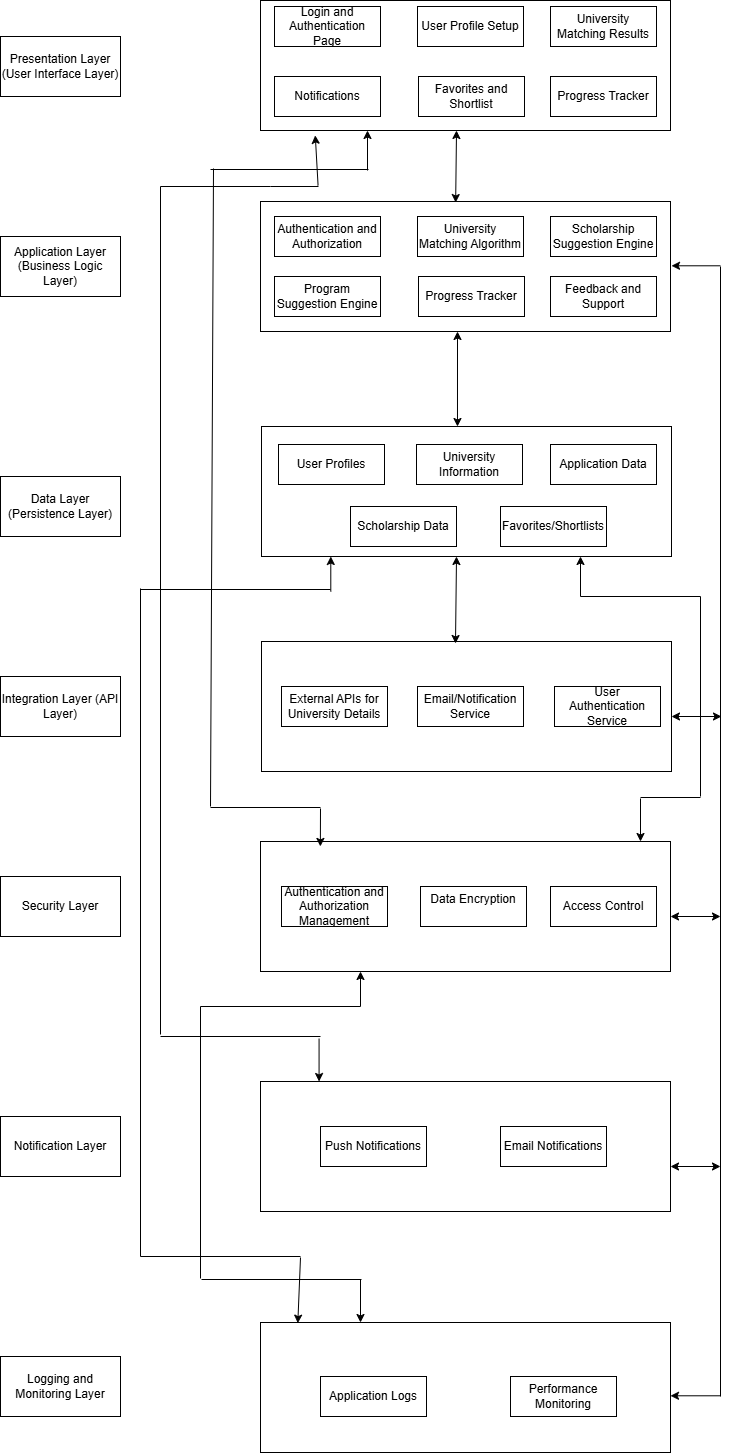
**DFD-Level 0:**

****

**DFD-Level 1:**

****

**Architectural Design:**

****

**Testing table :  
1-login and authentication:**

|  |  |
| --- | --- |
| **Module Name** | **login and authentication** |
| **Test Case ID** | 01 |
| **Test Case Description** | Verify the functionality of login, registration, password reset, and secure data storage. |
| **Prerequisites** | Application must be deployed with a database connection. User account exists for login and password reset scenarios. |
| **Tester Name** | [Tester Name] |
| **Environmental Information** | [Environment Details: e.g., OS, browser, version] |
| **Test Scenario** | 1. Verify that a student can register for the first time with valid email and password. 2. Verify that a registered user can log in with valid credentials. 3. Verify the "Forgot Password" functionality allows password reset via email. 4. Verify that login fails when incorrect credentials are provided. 5. Verify that users can log in again after a session ends. 6. Verify that passwords are securely stored in the database (e.g., encrypted). |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Test Steps** | **Expected Results** | **Actual Results** | **Status** | **Comments** |
| 1.1 | 1. Open login page.  2. Enter valid email and password.  3. Click "Login." | User should be logged in successfully. | User logged in successfully. | Pass | Login process works as expected. Ensure session persistence. |
| 1.2 | 1. Click "Forgot Password."  2. Enter registered email.  3. Submit request. | A password reset email should be sent to the registered email. | Password reset email sent. | Pass | Check email content and reset link functionality. |
| 1.3 | 1. Open login page.  2. Enter incorrect email or password.  3. Click "Login." | Error message should appear for invalid credentials. | Error message displayed. | Pass | Ensure error message is clear and not overly technical. |

**2.User Profile Creation**

|  |  |
| --- | --- |
| **Module Name** | **User Profile Creation** |
| **Test Case ID** | 02 |
| **Test Case Description** | Verify the functionality of user profile creation, including personal information, academic scores, achievements, and preferences. |
| **Prerequisites** | |  | | --- | | The application must be deployed and allow user registration. The student must be able to access the profile creation page. |  |  | | --- | |  | |
| **Tester Name** | [Tester Name] |
| **Environmental Information** | [Environment Details: e.g., OS, browser, version] |
| **Test Scenario** | 1. Verify that a student can enter personal information (name, date of birth, location, education level). 2. Verify that a student can input academic scores (e.g., GPA, standardized test scores). 3. Verify that a student can enter extracurricular achievements (e.g.,sports,competitions). 4Verify that a student can input preferences (e.g., countries, programs, budgets). 5. Verify that all entered data is saved correctly to the profile and can be edited later. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Test Case ID** | | |  | | --- | | **Test Steps** | | |  |  |  | | --- | --- | --- | | |  | | --- | | **Expected Results** |  |  | | --- | |  | | | |  | | --- | | **Actual Results** | | |  | | --- | | **Status** | | |  | | --- | | **Comments** | |
| 2.1 | |  | | --- | | 1. Open profile creation page.  2. Enter name, date of birth, location, and education. | | |  | | --- | | Personal information should be correctly saved. | | |  |  |  | | --- | --- | --- | | |  | | --- | | Personal information saved. |  |  | | --- | |  | | | |  | | --- | | Pass | | |  |  |  | | --- | --- | --- | | |  | | --- | | Verify all fields accept input in the correct format. |  |  | | --- | |  | | |
| 2.2 | 1. Enter GPA and standardized test scores. | |  | | --- | | Academic data should be correctly saved. |  |  | | --- | |  | | |  | | --- | | GPA and test scores saved. |  |  | | --- | |  | | |  | | --- | | Pass | | Validate that GPA is numeric and test scores are within a valid range. |
| 2.3 | |  | | --- | | 1. Enter extracurricular achievements (e.g., sports, competitions). |  |  | | --- | |  | | |  | | --- | | Achievements should be correctly saved. |  |  | | --- | |  | | Achievements saved. | |  | | --- | | Pass | | Make sure the input form allows multiple entries if needed. |
| 2.4 | |  | | --- | | 1. Enter preferences (countries, programs, budgets). |  |  | | --- | |  | | |  | | --- | | Preferences should be correctly saved. |  |  | | --- | |  | | |  | | --- | | Preferences saved. |  |  | | --- | |  | | |  | | --- | | Pass | | Validate that preferences are stored and displayed correctly. |

**3. University Matching Algorithm**:

|  |  |
| --- | --- |
| **Module Name** | **University Matching Algorithm** |
| **Test Case ID** | 03 |
| **Test Case Description** | Verify the functionality of the university matching algorithm, including filters for location, tuition fee range, program type, and language of instruction. |
| **Prerequisites** | The application must have user profiles with relevant data. University data must be available for matching. |
| **Tester Name** | [Tester Name] |
| **Environmental Information** | [Environment Details: e.g., OS, browser, version] |
| **Test Scenario** | 1. Verify that the algorithm matches universities based on the location preference. 2. Verify that the algorithm matches universities within the specified tuition fee range. 3. Verify that the algorithm matches universities offering the preferred program type (e.g., undergraduate/graduate). 4. Verify that the algorithm matches universities where the language of instruction matches the user's preference. 5. Verify that universities are returned in the correct order according to the matching criteria and filters. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Test Case ID** | | |  | | --- | | **Test Steps** | | |  |  |  | | --- | --- | --- | | |  | | --- | | **Expected Results** |  |  | | --- | |  | | | |  | | --- | | **Actual Results** | | |  | | --- | | **Status** | | |  | | --- | | **Comments** | |
| 3.1 | 1. Enter academic details, budget, and preferences.  2. Run matching algorithm. | A list of matching universities should be displayed. | Matching universities displayed. | Pass | Ensure algorithm properly matches based on academic and budget details. |
| 3.2 | 1. Apply filters for location and program type.  2. Run matching algorithm. | Matching universities should filter based on location and program. | Filtered matching universities. | Pass | Test different combinations of location and program filters. |

**Scholarship Indicator:**

|  |  |
| --- | --- |
| **Module Name** | **Scholarship Indicator** |
| **Test Case ID** | 04 |
| **Test Case Description** | |  | | --- | | Verify that the scholarship indicator provides tailored suggestions based on personal, academic, and extracurricular achievements. | |
| **Prerequisites** | |  | | --- | | The application must have user profiles with relevant data (personal, academic, and extracurricular achievements). | |
| **Tester Name** | [Tester Name] |
| **Environmental Information** | [Environment Details: e.g., OS, browser, version] |
| **Test Scenario** | |  | | --- | | 1. Verify that scholarships are suggested based on personal achievements (e.g., volunteer work, leadership roles). |  |  |  | | --- | --- | |  | 2. Verify that scholarships are suggested based on academic achievements (e.g., GPA, standardized test scores). |  |  |  | | --- | --- | |  | 3. Verify that scholarships are suggested based on extracurricular achievements (e.g., sports, competitions, community service). |  |  |  | | --- | --- | |  | 4. Verify that the system prioritizes scholarship suggestions based on relevance to the user’s achievements. |  |  |  | | --- | --- | |  | 5. Verify that when no scholarships match the criteria, an appropriate message or notification is displayed. |  |  |  | | --- | --- | |  | 6. Verify that scholarships displayed have accurate eligibility criteria based on the user's profile data. | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Test Case ID** | | |  | | --- | | **Test Steps** | | |  |  |  | | --- | --- | --- | | |  | | --- | | **Expected Results** |  |  | | --- | |  | | | |  | | --- | | **Actual Results** | | |  | | --- | | **Status** | | |  | | --- | | **Comments** | |
| 4.1 | 1. Enter personal, academic, and extracurricular data.  2. View scholarship suggestions. | Scholarships matching user profile should be suggested. | Scholarships suggested. | Pass | Check if the suggested scholarships are relevant and accurate. |
| 4.2 | 1. Enter academic data only.  2. View scholarship suggestions. | Scholarships relevant to academic achievements should be suggested. | Academic-based scholarships suggested. | Pass | Ensure that only academic scholarships appear when academic data is entered. |

**University Exploration Tools:**

|  |  |
| --- | --- |
| **Module Name** | **University Exploration Tools** |
| **Test Case ID** | 05 |
| **Test Case Description** | |  | | --- | | Verify the functionality of university exploration tools, providing direct access to university websites, program details, and scholarship pages. | |
| **Prerequisites** | |  | | --- | | The application must include links to u | |
| **Tester Name** | [Tester Name] |
| **Environmental Information** | [Environment Details: e.g., OS, browser, version] |
| **Test Scenario** | |  | | --- | | 1. Verify that users can access university websites directly from the app. |  |  |  | | --- | --- | |  | 2. Verify that users can view program details for each university. |  |  |  | | --- | --- | |  | 3. Verify that users can access dedicated scholarship pages for each institution. |  |  |  | | --- | --- | |  | 4. Verify that all links to university websites, program details, and scholarship pages are working correctly and load within a reasonable time. |  |  |  | | --- | --- | |  | 5. Verify that when there is no available information (e.g., missing program or scholarship pages), an appropriate message or notification is displayed. |  |  |  | | --- | --- | |  | 6. Verify that links open in the correct browser window or within the app’s web-view, depending on design specifications. | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Test Steps** | **Expected Results** | **Actual Results** | **Status** | **Comments** |
| 5.1 | 1. Open university exploration section.  2. Click on a university link. | The university website should open in a new tab or in-app browser. | University website opened. | Pass | |  | | --- | | Check for broken links and ensure browser compatibility. |  |  | | --- | |  | |
| 5.2 | 1. View program details for a selected university. | Program details should be displayed clearly. | Program details displayed. | Pass | Test if program details are correct and well-structured. |
| 5.3 | 1. Access the scholarship page for a university. | |  | | --- | | Scholarship page should load correctly. |  |  | | --- | |  | | Scholarship page loaded. | Pass | Verify page loads promptly and contains accurate information. |

**Alternative Program Suggestions:**

|  |  |
| --- | --- |
| **Module Name** | **Alternative Program Suggestions** |
| **Test Case ID** | 06 |
| **Test Case Description** | |  | | --- | | Verify the functionality of alternative program suggestions based on user preferences, academic scores, and budget constraints. | |
| **Prerequisites** | |  |  | | --- | --- | |  | The application must have user profiles with preferences, academic scores, and budget data. Universities and programs must be available for comparison. | |
| **Tester Name** | [Tester Name] |
| **Environmental Information** | [Environment Details: e.g., OS, browser, version] |
| **Test Scenario** | |  | | --- | | 1. Verify that the system suggests alternative programs based on the user’s academic scores. |  |  |  | | --- | --- | |  | 2. Verify that the system suggests alternative programs based on the user’s budget constraints. |  |  |  | | --- | --- | |  | 3. Verify that the system suggests alternative programs based on the user’s preferred countries or program types. |  |  |  | | --- | --- | |  | 4. Verify that suggested programs are similar in nature (e.g., same field of study, level of education) to the original preference. |  |  |  | | --- | --- | |  | 5. Verify that the system shows relevant details for suggested programs, such as program name, university, and tuition fee. |  |  |  | | --- | --- | |  | 6. Verify that when no alternative programs meet the criteria, an appropriate message or notification is displayed. | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Test Case ID** | | |  | | --- | | **Test Steps** | | |  |  |  | | --- | --- | --- | | |  | | --- | | **Expected Results** |  |  | | --- | |  | | | |  | | --- | | **Actual Results** | | |  | | --- | | **Status** | | |  | | --- | | **Comments** | |
| 6.1 | 1. Enter academic and program preferences.  2. View alternative program suggestions. | Relevant alternative programs should be suggested. | Alternative programs suggested. | Pass | Verify suggestions match user preferences and budget. |
| 6.2 | 1. Apply budget constraints.  2. View alternative programs. | Programs within budget should be displayed. | Budget-based programs displayed. | Pass | Check that programs fall within the specified budget range. |

**Favorites and Shortlist Functionality:**

|  |  |
| --- | --- |
| **Module Name** | **Favorites and Shortlist Functionality** |
| **Test Case ID** | 07 |
| **Test Case Description** | |  | | --- | | Verify the functionality of marking and saving universities or programs as favorites for easy comparison later. | |
| **Prerequisites** | |  | | --- | | The application must allow users to access universities and programs for marking as favorites. The user must be logged in to save favorites. | |
| **Tester Name** | [Tester Name] |
| **Environmental Information** | [Environment Details: e.g., OS, browser, version] |
| **Test Scenario** | |  | | --- | | 1. Verify that users can mark a university or program as a favorite. |  |  |  | | --- | --- | |  | 2. Verify that users can view their saved favorites in a separate section or list. |  |  |  | | --- | --- | |  | 3. Verify that users can add multiple universities or programs to their favorites. |  |  |  | | --- | --- | |  | 4. Verify that users can remove universities or programs from their favorites. |  |  |  | | --- | --- | |  | 5. Verify that users can compare their favorite universities or programs side by side for easy decision making. |  |  |  | | --- | --- | |  | 6. Verify that favorites are saved and persist across sessions. |  |  |  | | --- | --- | |  | 7. Verify that a notification or message is shown when the user successfully adds or removes an item from their favorites. | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Test Steps** | **Expected Results** | **Actual Results** | **Status** | **Comments** |
| 7.1 | 1. Select a university or program.  2. Mark it as a favorite. | The selected item should be added to the favorites list. | Item added to favorites list. | Pass | |  | | --- | | Ensure favorites are saved and can be retrieved later. |  |  | | --- | |  | |
| 7.2 | 1. View favorites list.  2. Compare two universities. | The list should display saved favorites for comparison. | Favorites list displayed. | Pass | Test comparison functionality and ensure it is user-friendly. |
| 7.3 | 1. Remove a university or program from favorites. | |  | | --- | | The item should be removed from the favorites list. |  |  | | --- | |  | | Item removed from favorites list. | Pass | Ensure deletion works correctly and updates the list immediately. |

**Real-Time Notifications:**

|  |  |
| --- | --- |
| **Module Name** | **Real-Time Notifications** |
| **Test Case ID** | 08 |
| **Test Case Description** | |  | | --- | | Verify that users receive real-time notifications about application deadlines, scholarship updates, and new programs matching their preferences. | |
| **Prerequisites** | |  |  | | --- | --- | |  | The application must have active data regarding application deadlines, scholarships, and program updates. The user must have set preferences. | |
| **Tester Name** | [Tester Name] |
| **Environmental Information** | [Environment Details: e.g., OS, browser, version] |
| **Test Scenario** | |  | | --- | | 1. Verify that users receive notifications about upcoming application deadlines based on their preferences. |  |  |  | | --- | --- | |  | 2. Verify that users receive notifications about new scholarships available based on their profile or achievements. |  |  |  | | --- | --- | |  | 3. Verify that users receive notifications about new programs that match their academic preferences or location. |  |  |  | | --- | --- | |  | 4. Verify that notifications are sent in real time without significant delay. |  |  |  | | --- | --- | |  | 5. Verify that users can view the details of the notification (e.g., the program, scholarship, or deadline information) when they click on the notification. |  |  |  | | --- | --- | |  | 6. Verify that notifications can be turned on/off through user settings. |  |  |  | | --- | --- | |  | 7. Verify that users do not receive duplicate or irrelevant notifications. | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Test Case ID** | | |  | | --- | | **Test Steps** | | |  |  |  | | --- | --- | --- | | |  | | --- | | **Expected Results** |  |  | | --- | |  | | | |  | | --- | | **Actual Results** | | |  | | --- | | **Status** | | |  | | --- | | **Comments** | |
| 8.1 | 1. Set user preferences for deadlines and scholarships.  2. Wait for notifications. | Notifications should appear for deadlines, scholarships, and new programs. | Notifications appeared. | Pass | Check if the notifications are timely and relevant. |
| 8.2 | 1. Click on a notification. | The corresponding details for the notification should be displayed. | Notification details displayed. | Pass | Ensure clicking a notification opens the correct content. |

**Progress Tracker:**

|  |  |
| --- | --- |
| **Module Name** | **Progress Tracker** |
| **Test Case ID** | 09 |
| **Test Case Description** | |  | | --- | | Verify the functionality of the progress tracker, including monitoring submission status and deadlines for shortlisted universities. | |
| **Prerequisites** | |  |  | | --- | --- | |  | The user must have shortlisted universities. The application must track submission statuses and deadlines for those universities. | |
| **Tester Name** | [Tester Name] |
| **Environmental Information** | [Environment Details: e.g., OS, browser, version] |
| **Test Scenario** | |  | | --- | | 1. Verify that users can view the submission status of their applications for shortlisted universities. |  |  |  | | --- | --- | |  | 2. Verify that users can view the upcoming deadlines for applications to shortlisted universities. |  |  |  | | --- | --- | |  | 3. Verify that users can mark the progress (e.g., submitted, in review) of their applications. |  |  |  | | --- | --- | |  | 4. Verify that the progress tracker updates in real time as users move through different stages of the application process. |  |  |  | | --- | --- | |  | 5. Verify that the progress tracker displays an appropriate notification or message when a deadline is approaching. |  |  |  | | --- | --- | |  | 6. Verify that the user can filter or sort their shortlisted universities based on submission status or deadlines. |  |  |  | | --- | --- | |  | 7. Verify that users can set reminders or alerts for upcoming deadlines and submission status updates. | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Test Case ID** | | |  | | --- | | **Test Steps** | | |  |  |  | | --- | --- | --- | | |  | | --- | | **Expected Results** |  |  | | --- | |  | | | |  | | --- | | **Actual Results** | | |  | | --- | | **Status** | | |  | | --- | | **Comments** | |
| 9.1 | 1. Open progress tracker.  2. View status of university applications. | |  | | --- | | The application status should be displayed correctly for each university. |  |  | | --- | |  | | Application status displayed. | Pass | Ensure progress is updated in real-time and accurately reflects submission status. |
| 9.2 | 1. Check upcoming deadlines for shortlisted universities. | Upcoming deadlines should be displayed correctly. | Deadlines displayed correctly. | Pass | Verify deadlines match with university application portals. |

**Feedback and Support System:**

|  |  |
| --- | --- |
| **Module Name** | **Feedback and Support System** |
| **Test Case ID** | 10 |
| **Test Case Description** | |  | | --- | | Verify the functionality of the feedback and support system, including submitting feedback via a form or email-based query system. | |
| **Prerequisites** | |  | | --- | | The application must have an accessible feedback form or email query system. The user must be logged in to submit feedback or queries. | |
| **Tester Name** | [Tester Name] |
| **Environmental Information** | [Environment Details: e.g., OS, browser, version] |
| **Test Scenario** | |  | | --- | | 1. Verify that users can access the feedback form or email query system from the app. |  |  |  | | --- | --- | |  | 2. Verify that users can submit feedback via the form or email query system. |  |  |  | | --- | --- | |  | 3. Verify that users receive a confirmation or acknowledgment message after submitting feedback or a query. |  |  |  | | --- | --- | |  | 4. Verify that users can select the type of feedback (e.g., general feedback, technical issue) when submitting a query. |  |  |  | | --- | --- | |  | 5. Verify that the feedback system correctly logs the feedback or query in the backend for future reference. |  |  |  | | --- | --- | |  | 6. Verify that users can follow up on their submitted queries or feedback if needed. |  |  |  | | --- | --- | |  | 7. Verify that feedback submitted via the form is routed to the appropriate support team for response. | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Test Case ID** | | |  | | --- | | **Test Steps** | | |  |  |  | | --- | --- | --- | | |  | | --- | | **Expected Results** |  |  | | --- | |  | | | |  | | --- | | **Actual Results** | | |  | | --- | | **Status** | | |  | | --- | | **Comments** | |
| 10.1 | 1. Open feedback form.  2. Submit feedback. | The feedback form should be submitted successfully. | |  | | --- | | Feedback submitted successfully. |  |  | | --- | |  | | Pass | Test if form submission works and data is properly saved. |
| 10.2 | 1. Open support query system.  2. Submit a support query. | Support query should be submitted and routed to the correct team. | Support query routed correctly. | Pass | Ensure support team receives the query and provides a timely response. |