Software Requirements Specification

**for**

**AI Based Career Counselling and Career Transition Recommender System**

**Version 1.1**

**Prepared by**

**Group name :**

|  |  |  |
| --- | --- | --- |
| **Fizza Mazhar** | **21-SE-017** | **21-SE-017@Student.hitecuni.edu.pk** |
| **Asad Shah** | **21-SE-021** | **21-SE-021@Student.hitecuni.edu.pk** |

|  |  |
| --- | --- |
| **Instructor:** | ***Miss Amber Urooj*** |
| **Course:** | ***FYP*** |
| **Lab Section:** |  |
| **Teaching Assistant:** | ***Miss Fatima Rani*** |
| **Date:** | ***19-12-2024*** |

**Contents**

Contents

[1 Introduction 3](#_Toc183940280)

[1.1 Document Purpose 3](#_Toc183940281)

**Revisions**

| **Version** | **Primary Author(s)** | **Description of Version** | **Date Completed** |
| --- | --- | --- | --- |
| Version 0.1 | Fizza Mazhar | The first version of the SRS to have all the conventions included. | 30/11/24 |
| Version 0.2 | Fizza Mazhar | The functionality of the system. | 15/12/24 |
| Version 0.3 | Asad Shah | The interface design of the system and the functional requirements. | 1/1/25 |
| Version 0.4 | Fizza Mazhar | Formatting of the document. | 2/1/25 |
| Version 0.5 | Asad Shah | Adding use cases F1. | 3/1/25 |
| Version 0.6 | Fizza Mazhar | Addding use case diagram and remaining use cases. | 4/1/25 |
| Version 0.7 | Asad Shah | Added some design part, diagrams | 5/1/25 |
| Version 0.8 | Asad Shah | Created database schema | 6/1/25 |

# Introduction

The project focuses on developing a counseling platform for students in Pakistan who have completed their 12th grade. The platform addresses the challenges students face in deciding their career paths and understanding university admission processes. By providing personalized recommendations, university information and insights into field trends, the platform empowers students to make informed decisions about their higher education.

In this section you will find a comprehensive outline of the project's purpose and the problems it seeks to solve. This section highlights the challenges faced by students after 12th grade in Pakistan, including their lack of awareness about career options and university admission processes. It also introduces the platform's primary objectives, emphasizing how it provides tailored guidance, university recommendations and insights into field trends.

## Document Purpose

This document specifies the software requirements for the **Counseling Platform for Students in** **Pakistan** after 12th grade. The platform is designed to assist students in choosing suitable fields of study and identifying universities that match their academic credentials, interests and career goals. It aims to address the lack of career guidance and awareness among students regarding the field options available and the university admission processes in Pakistan. The platform, in its current scope, provides personalized field and university recommendations, trend analysis and field comparisons.

This SRS describes the full scope of the system, encompassing its core functionalities such as data collection, aggregate calculation, field and university suggestions and dashboard visualizations. It also defines the boundaries of this release which includes trend analysis tools and visualizations based on datasets from 2018 to 2024. The document ensures a detailed understanding of the system’s objectives, features and intended users forming the foundation for design, development and implementation.

## Product Scope

The **Counseling Platform for Students in Pakistan** is a web-based application designed to guide students after 12th grade in making informed decisions about their future education and career paths. The platform collects academic data (Matric, FSC marks and test scores like NTS or NET) and preferences (study stream and interests) to recommend suitable fields of study and universities. It also provides detailed information about university admission criteria, deadlines, and admission links. The platform features a dashboard for visualizing field trends and performing comparisons allowing students to explore growing and declining fields both locally and globally.

The platform benefits students by addressing the knowledge gap about available academic fields and admission processes, thus reducing the uncertainty and stress involved in making critical career decisions. It empowers users with actionable insights and data-driven recommendations, enabling them to align their academic choices with market trends and global opportunities. The website aims to enhance career awareness, streamline the university selection process and help students achieve their educational and professional goals.

## Intended Audience and Document Overview

This document is intended for a diverse group of readers involved in understanding, developing or evaluating the Counseling Platform for Students in Pakistan. Key audience types include:

1. **Client** to understand the scope, objectives and features of the platform ensuring alignment with their vision and requirements.
2. **Professor** to evaluate the project’s objectives, methodology, and alignment with academic standards and best practices.
3. **Developers** to use the technical requirements and system design as a reference for implementing the platform.
4. **Testers** to identify test cases and validate the system’s functionality against the specified requirements.
5. **Project managers** to monitor project progress and ensure all requirements are addressed within the planned timeline.
6. **Documentation writers** to create user guides, help manuals, and other instructional content based on the system’s features and functionality.

The document overview is explained below:

1. **Introduction** describes the document's purpose, scope, audience, platform objectives and challenges.
2. **Overall description** provides a high-level overview of the platform, its context and constraints.
3. **Specific requirements** details functional requirements and use cases for user interactions.
4. **Non-functional requirements** defines performance, security and quality attributes.
5. **Design requirements** includes architectural patterns, design diagrams and interface mockups.
6. **Data design and relationships** explains database schema, ER diagrams and data dictionary.
7. **Software planning and timeline** outlines work breakdown structure, milestones and project timeline.
8. **Quality assurance plan** lists testing requirements, acceptance criteria and test cases.

Suggested reding sequence is given below:

1. Start with Introduction to understand the project.
2. Move to Overall Description for high-level context.
3. Review Specific Requirements for detailed functionalities.
4. Refer to Design Requirements for system architecture.
5. Check Data Design for database understanding.
6. Look at Software Planning for project timeline insights.
7. End with Quality Assurance for validation strategies.

## Definitions, Acronyms and Abbreviations

Some acronyms and abbreviations used in document as given below

1. **AI** (Artificial Intelligence) is the simulation of human intelligence in machines.
2. **ER** Diagram (Entity-Relationship Diagram) is the visual representation of database relationships.
3. **FCS** (Faculty of Computer Science) is the study stream for computer-related fields.
4. **FSC** (Faculty of Science) is the study stream for science-related fields.
5. **GUI** (Graphical User Interface) is the visual interface for user interaction.
6. **IEEE** (Institute of Electrical and Electronics Engineers) is the standardization organization.
7. **NET** (National Engineering Test) is the entry test for engineering universities in Pakistan.
8. **NTS** (National Testing Service) is the testing body conducting standardized tests in Pakistan.
9. **SRS** (Software Requirements Specification)is the document detailing software system requirements.
10. **UML** (Unified Modeling Language) is the standardized modeling language in software engineering.

## 1.5 Document Conventions

The following standards and conventions were followed when writing this SRS:

**Font:**

1. **Text**: Times New Roman, 12pt.

**Headings**:

1. Heading 1: 16pt, Bold.
2. Heading 2: 14pt, Bold.
3. Heading 3: 12pt, Bold.
4. Figure captions: 11pt, placed below figures.
5. Table captions: 11pt, placed above tables.

**Spacing:**

1. Line Spacing: 1.5.
2. Paragraph Spacing: 12pt after Heading 1.

**Alignment:**

All text is justified.

**Margins:**

Standard 1-inch margins (Top, Bottom, Left, and Right).

**Header and Footer:**

1. Header: Project title on the right.
2. Footer: Page numbers on the right.
3. Header and footer begin from the abstract section onward.

**Section and Subsection Titles:**

1. Section titles follow a hierarchical numbering system limited to three levels (e.g., 3, 3.1, 3.1.1).
2. Titles are formatted in bold according to their respective heading levels.

**Special Text Formatting:**

1. Abstract: Italicized.
2. Comments: Italics.

**Referencing Style:**

1. APA referencing style is used for journals, conference papers, websites, and books.
2. Date and time of access are included for online references.
3. Wikipedia references are not accepted.

**Additional Conventions:**

1. Table of Contents: Updated to reflect accurate page numbers.
2. List of Figures and Tables: Included only if the document contains more than three tables or figures.
3. Figures and Tables: Properly captioned and referenced within the text.

## References and Acknowledgements

**References**

1. IEEE Software Requirements Specification Standards.
2. FYDP Documentation Formatting Style from HITEC University Taxila guidelines for documentation formatting (Version 2024).
3. APA Referencing Guidelines for scholarly citations and references.

**Web Resources**

University Admission Portals of Pakistan for links provided in the respective recommendations.

**Acknowledgments**

1. Department of Computer Science, HITEC University Taxila for providing academic guidance and formatting standards.
2. Instructor and TA for continuous support and feedback during the documentation process.
3. Data providers and contributors for sharing valuable datasets used in trend analysis and recommendation systems.

# Overall Description

## 2.1 Project Overview

The Counseling Platform for Students in Pakistan is a self-contained, web-based application designed to address the challenges students face after completing their 12th grade. It is a new initiative aimed at providing personalized field and university recommendations based on academic performance, interests and current field trends. Unlike traditional career counseling systems which often rely on manual processes or generic advice, this platform leverages data-driven insights and visualizations to make informed, personalized recommendations.

The platform integrates multiple subsystems including user data collection, field recommendation algorithms, university matching and trend analysis dashboards. It interacts with external data sources such as historical university merit aggregates (2018–2024) and global field demand statistics ensuring relevance and accuracy. The system also interfaces with university admission portals to provide students with application links and deadlines making the admission process more accessible.

The platform operates in the following context:

1. **Input**: Student data (academic scores, study streams, interests) and external datasets (field trends, admission criteria).
2. **Processing**: Aggregate calculation, field matching, trend visualization and recommendation generation.
3. **Output**: Personalized field and university recommendations, trend insights and comparison visualizations.

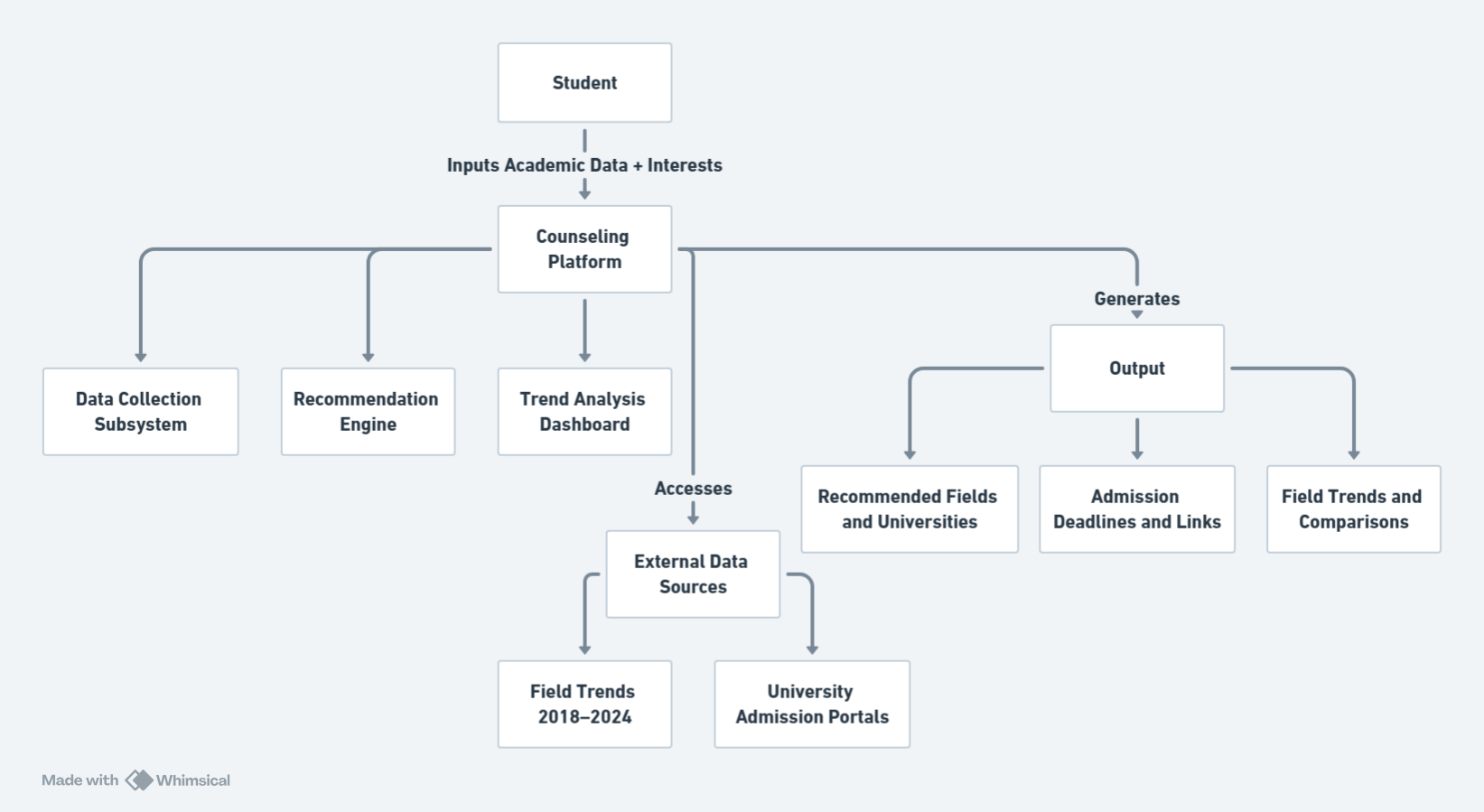


Figure 1

**2.2 Product Functionality**

1. Collects user academic details, study streams, and interests for personalized recommendations.
2. Suggests suitable fields based on academic performance and preferences.
3. Recommends universities aligned with calculated aggregates and selected fields.
4. Displays field trends through visualizations based on historical data (2018–2024).
5. Enables comparison of multiple fields regarding growth, demand and opportunities.
6. Shows global demand for fields to explore international career prospects.
7. Provides an interactive dashboard for trend exploration and detailed insights.
8. Directs users to university admission portals with deadlines and guidance.

## 2.3 Design and Implementation Constraints

The following constraints will guide the design and implementation of the platform:

### 2.3.1 Hardware Limitations

1. The platform must function efficiently on mid-range hardware configurations with limited memory and processing power.
2. The server hosting the platform should support parallel processing to handle multiple users simultaneously.

### 2.3.2 Technologies and Tools

1. The platform will be developed using React.js for the front end and Node.js for the back end.
2. MongoDB will be used as the database to manage user data and historical datasets.
3. Data visualization will be implemented using D3.js or Chart.js for dynamic graphs and dashboards.

### 2.3.3 Interfaces

1. The system will interact with external university portals to provide admission links.
2. It will integrate with APIs or external data sources for global field demand statistics.

### 2.3.4 Programming Standards

1. The software will adhere to COMET (Concurrent Object Modeling and Architectural Design Technique) for design.
2. UML (Unified Modeling Language) will be used for system modeling including class diagrams, use case diagrams and sequence diagrams.

### 2.3.5 Language and Protocols

1. The system will be built using JavaScript.
2. Communication between client and server will use HTTPS for secure data transmission.

### 2.3.6 Security Considerations

1. User data will be stored securely, adhering to GDPR-like privacy standards.
2. All sensitive information such as login credentials will be encrypted.
3. The platform will use authentication mechanisms (e.g., OAuth 2.0) for secure user access.

### 2.3.7 Design Conventions

1. Follows the HITEC University Taxila documentation standards for consistency.
2. The UI/UX design will adhere to accessibility standards ensuring usability for all users.

### 2.3.8 Parallel Operations

The platform must support concurrent users accessing recommendations and visualizations without performance degradation.

## 2.4 Assumptions and Dependencies

Assumptions of the project are as following

1. It is assumed that users will enter accurate and complete academic and interest data for personalized recommendations.
2. Historical datasets (2018–2024) and global field demand statistics are assumed to be accurate, complete and free from significant discrepancies.
3. Users will have a stable internet connection to access the platform and interact with dynamic visualizations.
4. The platform will be accessed on modern browsers (e.g., Chrome, Firefox) with standard features enabled (e.g., JavaScript).

Dependencies of the project are given below

1. The platform relies on APIs or data files for university merit lists and global field trends. Any changes or unavailability in these sources could affect functionality.
2. Data visualization depends on libraries like D3.js or Chart.js, which must remain supported and updated.
3. The system depends on reliable hosting services for server and database deployment.
4. The use of OAuth 2.0 or similar third-party authentication protocols is assumed for secure user login.
5. Components from similar projects or open-source libraries may be reused to expedite development, assuming compatibility with the platform.

# Specific Requirements

## 3.1 External Interface Requirements

### 3.1.1 User Interfaces

The interface design in Figure 2 defines a user login and signup functionality when the user visits our website.

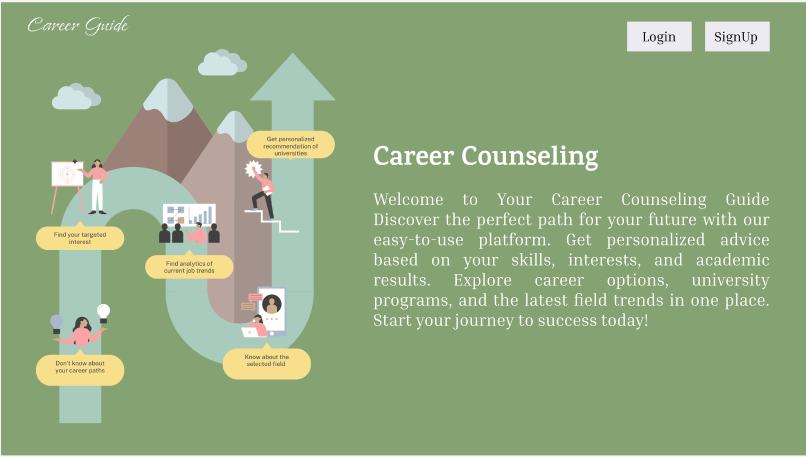


Figure 2

After signup and login, the user is directed to homepage shown in Figure 3 to have options to navigate to other pages. Homepage has basic introduction of our website and a get started button which navigates towards the finding of their career.

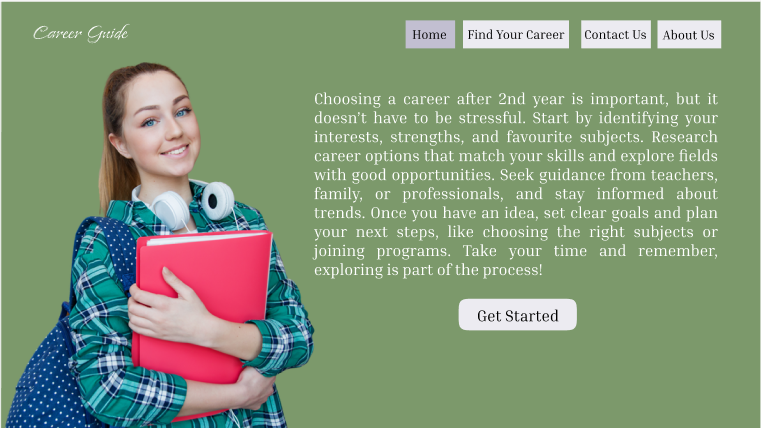


Figure 3

After clicking the get started button, we are navigated to find our career section shown in Figure 4 where user can select any field according to his interest, enter matric marks, enter FSC marks and then the universities related to the given data is shown in the below fields which navigate to the web pages of the respected universities. If anyone is not clear about his interests, he can chat with the chatbot to find his interest by telling the likes and dislikes which help the chatbot to find the interest.

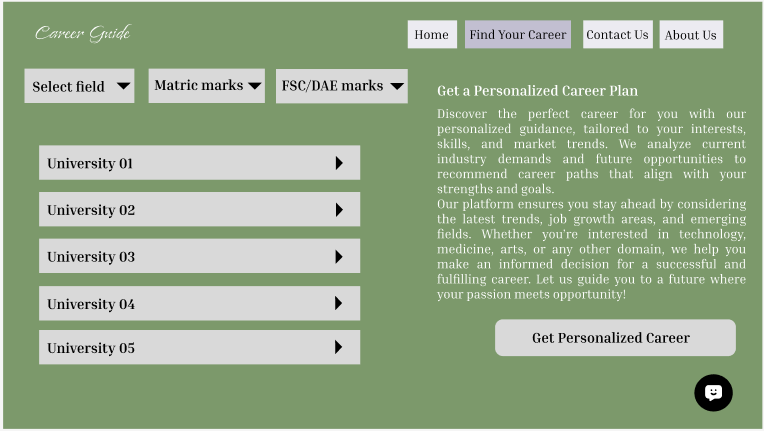


Figure 4

The Figure 4 has help to suggest the universities according to the interest, matric and FSC marks. The get personalized career button navigates to the page shown in Figure 5 to help the student if he wants to select the career according to market trends and the jobs availability. User can select particular filed to find the trends, find the trending fields and how they grow in last 5 years and can have the comparison of selected fields.

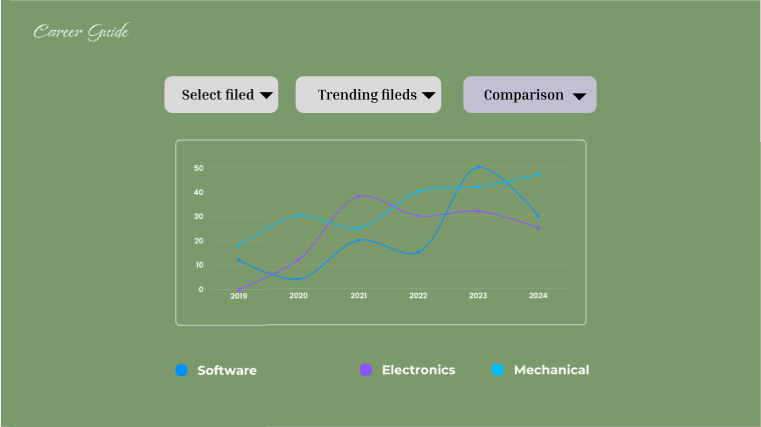


Figure 5

We also have our about page which shows the purpose of our website shown in Figure 6.

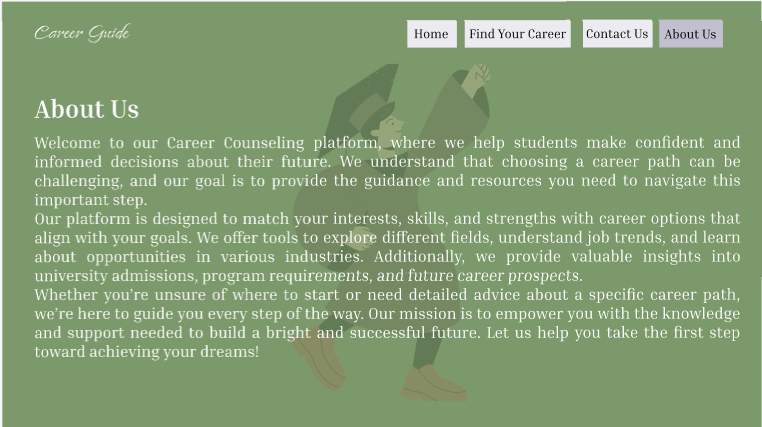
****

Figure 6

Contact us page help to connect the users for any means by sending their name, email and a message shown in Figure 7.

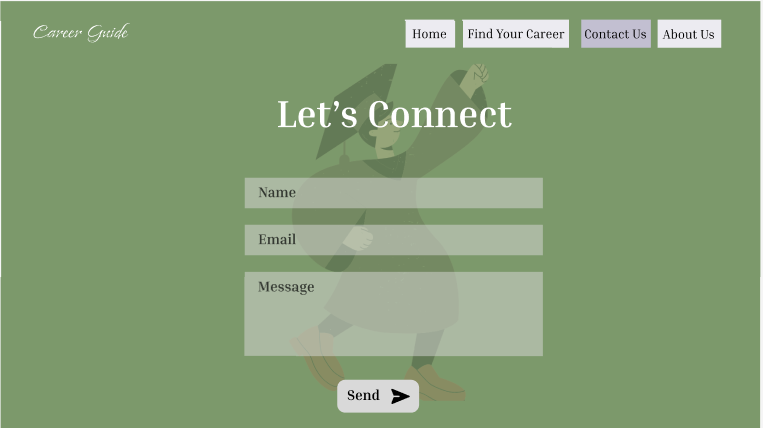


Figure 7

### 3.1.2 Hardware Interfaces

**Desktop/Laptop Computers**

1. The website is accessible through modern web browsers (e.g. Chrome, Firefox, Safari) installed on desktops or laptops.
2. Interaction includes input via a keyboard and mouse.
3. Users require a functional desktop or laptop computer with internet connectivity.

**Mobile Phones and Tablets**

1. The website is designed to be responsive and compatible with mobile devices.
2. Interaction includes input via a touchscreen for navigation, form filling and menu selection.
3. Devices include smartphones and tablets with functional touchscreens and internet access.

**Servers and Hosting Infrastructure**

1. The backend of the website is hosted on a web server that manages client requests, processes user data and serves web pages.
2. Interactions include handling database queries and sending responses to user actions.
3. Cloud-based or physical servers maintained by the hosting provider.

**Input and Output Devices**

1. Keyboard and mouse (for desktops/laptops) or touchscreen (for mobile/tablets) to navigate and interact with the website.
2. Monitors/screens to display the website content and visualizations (e.g. trend graphs).

### 3.1.3 Software Interfaces

1. The website interacts with a database to store and retrieve user data such as academic details, interests and career trends.
2. Google Oauth API for user login and signup.
3. Data fetching from LinkedIn website/app.
4. Data Visualization Libraries (e.g., Chart.js, D3.js) for rendering graphical representations of career trends.

## 3.2 Functional Requirements

### 3.2.1 F1: User Registration and Login

1. The system shall allow users to register with their email address and password or log in using existing credentials.
2. The system shall provide an option for Google OAuth for simplified login.

### 3.2.2 F2: Data Input and Validation

1. The system shall allow users to input their Matric, FSC marks and NTS or NET test scores.
2. The system shall allow users to enter expected marks if their results are pending.
3. The system shall ask the user to select their study stream (e.g. FSC, FCS, Pre-Engineering, Pre-Medical).
4. The system shall validate all entered data to ensure it meets expected formats (e.g. numeric values for marks).

### 3.2.3 F3: Field Recommendation

1. The system shall suggest potential fields of study based on the user's entered marks and study stream.
2. The system shall provide field options based on the user’s stated interests, if any. The user interest is stated by user himself, he knows himself or can find after chatting with career counseling chatbot.

### 3.2.4 F4: Interest Finding Chatbot

1. User interest can be found by user himself after chatting with chatbot by telling him interest in studies and games.

### 3.2.5 F5: Aggregate Calculation

1. The system shall calculate the user's aggregate based on their entered marks using predefined formulas.
2. The system shall display the calculated aggregate to the user for transparency.

### 3.2.6 F6: University Recommendation

### The system shall recommend universities based on the user's calculated aggregate.

1. The system shall provide details about each university, including:
2. Admission criteria.
3. Application deadlines.
4. Online application links.

### 3.2.7 F7: Trend Visualization

1. The system shall display trends for various fields (e.g., growing vs. declining fields) using graphical visualizations (e.g. bar charts, line graphs).
2. The system shall allow users to compare multiple fields and view demand trends in Pakistan and foreign countries.

### 3.2.8 F8: Dashboard for Insights

1. The system shall provide a dashboard where users can:
2. View personalized recommendations for fields and universities.
3. Visualize trends and comparisons between different fields.

### 3.2.9 F9: Search and Filter Options

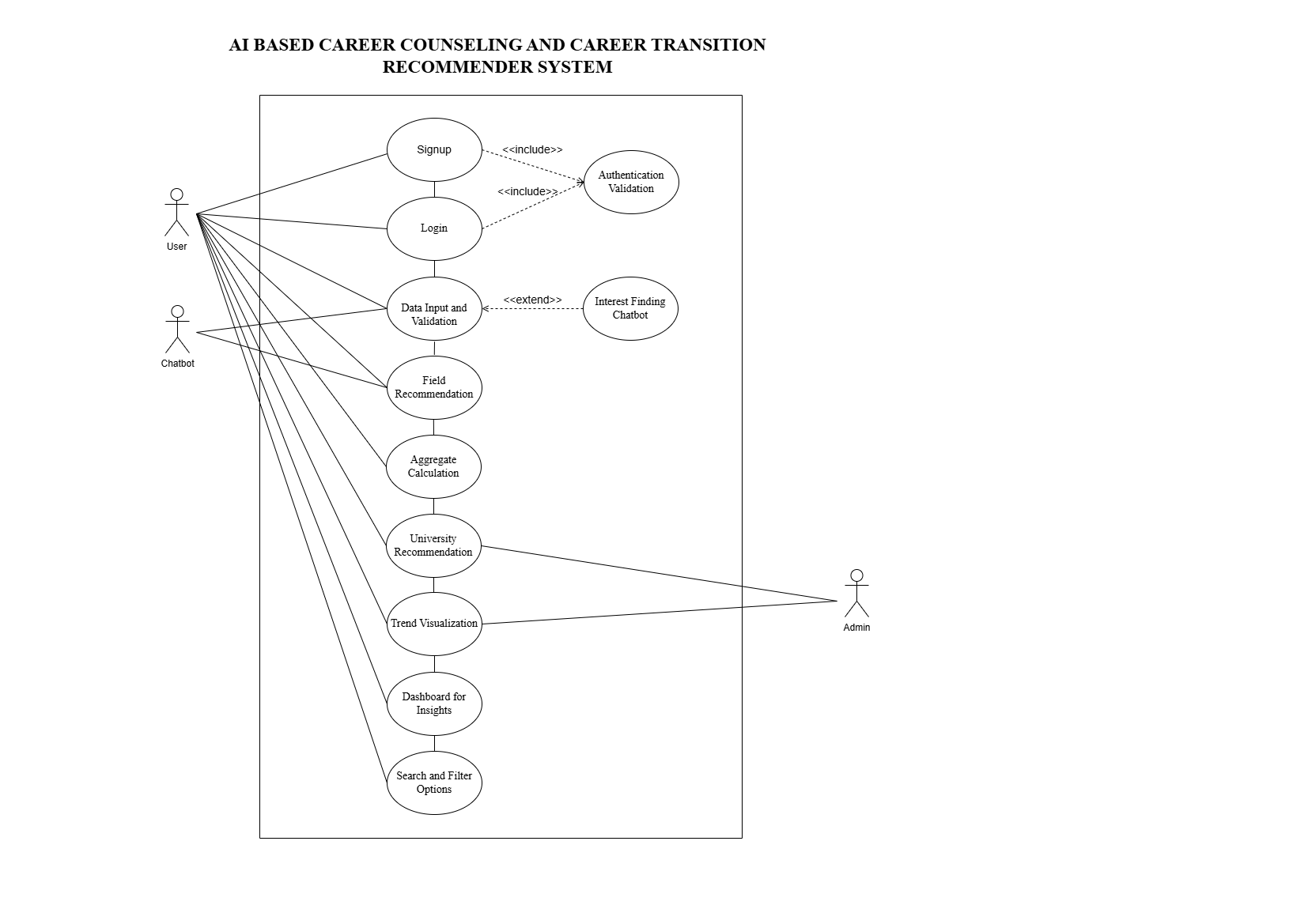
The system shall allow users to search and filter university options based on:

1. Location.
2. Aggregate eligibility.

### 3.2.10 F10: User-Friendly Interface

1. The system shall provide an intuitive and user-friendly interface to ensure ease of use for all users.
2. The system shall ensure accessibility by incorporating responsive design for both desktop and mobile devices.

## 3.3 Use Case Model



### 3.3.1 Use Case # 1.1

|  |  |  |  |
| --- | --- | --- | --- |
| **1.1** | **Register New User** | | |
| **Author** | Asad Shah | | |
| **Purpose** | Allow new users to create an account to access the system's features. | | |
| **Requirements Traceability** | F1.1 | | |
| **Priority** | High | | |
| **Preconditions** | User must have a valid email address. | | |
| **Postconditions** | A new account is created and the user can log in to the system. | | |
| **Actors** | User (Student or Father) | | |
| **Includes** | Authentication Validation | | |
| **Flow of Events** | Basic | 1 | User provides email and password. |
| 2 | User clicks "Register. |
| 3 | System validates input and creates an account. |
| 4 | System confirms successful registration. |
| Alternative | User inputs invalid email/password: system prompts for correction. | |
| Exceptions | Email already exists: system alerts user to use a different email. | |

### 3.3.2 Use Case # 1.2

|  |  |  |  |
| --- | --- | --- | --- |
| **1.2** | **Login with Email and Password** | | |
| **Author** | Asad Shah | | |
| **Purpose** | Allow existing users to log in securely using their email and password. | | |
| **Requirements Traceability** | F1.2 | | |
| **Priority** | High | | |
| **Preconditions** | User must have previously registered an account. | | |
| **Postconditions** | The user is authenticated and granted access to the system. | | |
| **Actors** | User (Student or Father) | | |
| **Includes** | Authentication Validation | | |
| **Flow of Events** | Basic | 1 | User provides email and password. |
| 2 | User clicks "Login." |
| 3 | System verifies credentials. |
| 4 | System grants access if valid. |
| Alternative | User inputs invalid credentials: system prompts for retry or password reset. | |
| Exceptions | Too many failed login attempts: system temporarily locks the account. | |

### 3.3.3 Use Case # 1.3

|  |  |  |  |
| --- | --- | --- | --- |
| **1.3** | **Login with Google OAuth** | | |
| **Author** | Asad Shah | | |
| **Purpose** | Simplify the login process for users with an existing Google account. | | |
| **Requirements Traceability** | F1.3 | | |
| **Priority** | Medium | | |
| **Preconditions** | User must have a valid Google account. | | |
| **Postconditions** | The user is authenticated via Google and granted access to the system. | | |
| **Actors** | User (Student or Father), Google Authentication Service | | |
| **Includes** | Authentication Validation | | |
| **Flow of Events** | Basic | 1 | User clicks "Login with Google." |
| 2 | System redirects to Google OAuth. |
| 3 | User authorizes. |
| 4 | System grants access if valid. |
| Alternative | Google OAuth authorization fails: system prompts for retry or alternative login method. | |
| Exceptions | Google service unavailable: system prompts user to try later or use alternative login. | |

### 3.3.4 Use Case # 2.1

|  |  |  |  |
| --- | --- | --- | --- |
| **2.1** | **Input Academic Marks** | | |
| **Author** | Asad Shah | | |
| **Purpose** | Allow users to input Matric, FSC, and test scores for processing. | | |
| **Requirements Traceability** | F2.1 | | |
| **Priority** | High | | |
| **Preconditions** | User must be logged in to the system. | | |
| **Postconditions** | Marks are successfully stored for aggregate calculation. | | |
| **Actors** | User (Student or Father) | | |
| **Flow of Events** | Basic | 1 | User enters Matric, FSC, and test scores. |
| 2 | System saves the data. |
| 3 | System confirms successful input. |
| Alternative | User skips some marks: system prompts to fill in all required fields. | |
| Exceptions | System encounters data storage error: prompts user to retry. | |

### 3.3.5 Use Case # 2.2

|  |  |  |  |
| --- | --- | --- | --- |
| **2.2** | **Input Expected Marks** | | |
| **Author** | Asad Shah | | |
| **Purpose** | Allow users to input expected marks if actual results are pending. | | |
| **Requirements Traceability** | F2.2 | | |
| **Priority** | Medium | | |
| **Preconditions** | User must indicate that results are pending. | | |
| **Postconditions** | Expected marks are saved for temporary use in calculations. | | |
| **Actors** | User (Student or Father) | | |
| **Flow of Events** | Basic | 1 | User selects "Expected Marks Pending." |
| 2 | User inputs expected marks. |
| 3 | System saves the data. |
| Alternative | User skips entering expected marks: system skips aggregate calculation until marks are entered. | |
| Exceptions | None | |

### 3.3.6 Use Case # 2.3

|  |  |  |  |
| --- | --- | --- | --- |
| **2.3** | **Select Study Stream** | | |
| **Author** | Asad Shah | | |
| **Purpose** | Allow users to select their study stream for relevant field recommendations. | | |
| **Requirements Traceability** | F2.3 | | |
| **Priority** | High | | |
| **Preconditions** | User must have entered valid marks or expected marks. | | |
| **Postconditions** | Study stream is stored for field recommendation processing. | | |
| **Actors** | User (Student or Father) | | |
| **Flow of Events** | Basic | 1 | User selects their study stream from predefined options. |
| 2 | System saves the selection. |
| Alternative | User selects "Other": system prompts for additional details or confirmation. | |
| Exceptions | System error: prompts user to reselect the stream. | |

### 3.3.7 Use Case # 2.4

|  |  |  |  |
| --- | --- | --- | --- |
| **2.4** | **Validate Input Data** | | |
| **Author** | Asad Shah | | |
| **Purpose** | Ensure all input data meets required format and validity criteria. | | |
| **Requirements Traceability** | F2.4 | | |
| **Priority** | High | | |
| **Preconditions** | User must have entered data in the respective input fields. | | |
| **Postconditions** | All validated data is successfully stored. | | |
| **Actors** | System | | |
| **Flow of Events** | Basic | 1 | System validates numeric fields for marks. |
| 2 | System flags invalid entries for correction. |
| 3 | Valid data is stored. |
| Alternative | None | |
| Exceptions | Invalid data detected: prompts user for correction. | |

### 3.3.8 Use Case # 3.1

|  |  |  |  |
| --- | --- | --- | --- |
| **3.1** | **Recommend Fields Based on Marks and Stream** | | |
| **Author** | Asad Shah | | |
| **Purpose** | Suggest potential fields of study based on the user's entered marks and selected study stream. | | |
| **Requirements Traceability** | F3.1 | | |
| **Priority** | High | | |
| **Preconditions** | User must have entered valid marks and selected a study stream. | | |
| **Postconditions** | The system displays a list of potential fields tailored to the user’s marks and study stream. | | |
| **Actors** | User (Student or Father) | | |
| **Flow of Events** | Basic | 1 | System processes user-entered marks and stream. |
| 2 | System applies predefined rules or thresholds. |
| 3 | System displays suitable fields to the user. |
| Alternative | User’s marks are insufficient for most fields: system suggests alternate pathways or preparatory options. | |
| Exceptions | System cannot calculate eligibility due to missing data: prompts user to provide required inputs. | |

### 3.3.9 Use Case # 3.2

|  |  |  |  |
| --- | --- | --- | --- |
| **3.2** | **Recommend Fields Based on Stated Interests** | | |
| **Author** | Asad Shah | | |
| **Purpose** | Provide personalized field recommendations based on the user’s stated interests. | | |
| **Requirements Traceability** | F3.2 | | |
| **Priority** | Medium | | |
| **Preconditions** | User must have interacted with the chatbot or directly entered their interests. | | |
| **Postconditions** | The system displays a list of potential fields aligned with the user’s stated interests. | | |
| **Actors** | User (Student or Father) | | |
| **Flow of Events** | Basic | 1 | User states interests via chatbot or input. |
| 2 | System maps interests to field options. |
| 3 | System displays matching fields to the user. |
| Alternative | User’s stated interests do not map directly to predefined fields: system prompts for refinement. | |
| Exceptions | User provides vague or incomplete interests: system suggests common fields based on trends. | |

### 3.3.10 Use Case # 4

|  |  |  |  |
| --- | --- | --- | --- |
| **4** | **Chat with Interest Finding Chatbot** | | |
| **Author** | Asad Shah | | |
| **Purpose** | Help users discover their interests by interacting with a chatbot that analyzes their input. | | |
| **Requirements Traceability** | F4 | | |
| **Priority** | Medium | | |
| **Preconditions** | User must be logged in and must have basic familiarity with using a chatbot interface. | | |
| **Postconditions** | The chatbot captures and saves the user’s interests for use in recommending relevant fields. | | |
| **Actors** | User (Student or Father), chatbot | | |
| **Flow of Events** | Basic | 1 | User initiates a chat with the chatbot. |
| 2 | Chatbot prompts user with questions about their study and game preferences. |
| 3 | User provides responses. |
| 4 | Chatbot analyzes responses and identifies potential interests. |
| 5 | Chatbot saves the interests for further processing. |
| Alternative | User skips chatbot interaction: system defaults to general interest-based recommendations or prompts to try later. | |
| Exceptions | Chatbot fails to analyze input: prompts user to re-enter responses or suggests manual entry. | |



### 3.3.10 Use Case # 5.1

|  |  |  |  |
| --- | --- | --- | --- |
| **5.1** | **Calculate Aggregate** | | |
| **Author** | Fizza Mazhar | | |
| **Purpose** | Calculate the user’s aggregate based on predefined formulas and the entered data. | | |
| **Requirements Traceability** | F5.1 | | |
| **Priority** | High | | |
| **Preconditions** | User must have entered valid academic marks and selected their study stream. | | |
| **Postconditions** | The system calculates the aggregate and stores it for use in university recommendations. | | |
| **Actors** | System | | |
| **Flow of Events** | Basic | 1 | System retrieves user-entered marks and stream. |
| 2 | System applies the aggregate calculation formula. |
| 3 | System stores the calculated aggregate for further use. |
| Alternative | Insufficient data for calculation: system prompts user to enter missing marks or stream. | |
| Exceptions | Calculation error due to invalid data: system flags the issue and prompts user for correction. | |
| **Notes / Issues** | Ensure formulas are accurate and in line with local educational board standards. | | |

### 3.3.11 Use Case # 5.2

|  |  |  |  |
| --- | --- | --- | --- |
| **5.2** | **Display Aggregate** | | |
| **Author** | Fizza Mazhar | | |
| **Purpose** | Display the calculated aggregate to the user for transparency and verification. | | |
| **Requirements Traceability** | F5.2 | | |
| **Priority** | High | | |
| **Preconditions** | Aggregate must have been successfully calculated. | | |
| **Postconditions** | The aggregate is displayed to the user in an easy-to-understand format. | | |
| **Actors** | User (Student or Father), System | | |
| **Flow of Events** | Basic | 1 | System retrieves the calculated aggregate. |
| 2 | System displays the aggregate on the user dashboard. |
| 3 | User reviews the displayed aggregate. |
| Alternative | Aggregate display includes a breakdown of components (e.g. weightage of marks, tests). | |
| Exceptions | System fails to fetch aggregate: prompts user to try again or contact support. | |

### 3.3.12 Use Case # 6.1

|  |  |  |  |
| --- | --- | --- | --- |
| **6.1** | **Recommend Universities Based on Aggregate** | | |
| **Author** | Fizza Mazhar | | |
| **Purpose** | Suggest universities where the user is eligible to apply based on their calculated aggregate. | | |
| **Requirements Traceability** | F6.1 | | |
| **Priority** | High | | |
| **Preconditions** | User must have a calculated aggregate and preferences for university search (e.g., location). | | |
| **Postconditions** | A list of universities matching the user’s eligibility is displayed. | | |
| **Actors** | User (Student or Father), System | | |
| **Flow of Events** | Basic | 1 | System retrieves the user’s aggregate and preferences. |
| 2 | System compares the data with predefined university criteria. |
| 3 | System generates a list of eligible universities. |
| 4 | System displays the list to the user. |
| Alternative | User changes preferences (e.g., location): system updates the university recommendation list. | |
| Exceptions | No universities meet the user’s eligibility: system suggests preparatory programs or alternatives. | |
| **Notes / Issues** | Ensure university data is up-to-date, especially admission deadlines and criteria. | | |

### 3.3.13 Use Case # 6.2

|  |  |  |  |
| --- | --- | --- | --- |
| **6.2** | **Provide University Details** | | |
| **Author** | Fizza Mazhar | | |
| **Purpose** | Provide detailed information about the recommended universities to assist the user. | | |
| **Requirements Traceability** | F6.2 | | |
| **Priority** | Medium | | |
| **Preconditions** | University recommendations must be generated and available for further details. | | |
| **Postconditions** | Detailed university information is presented to the user for informed decision-making. | | |
| **Actors** | User (Student or Father), System | | |
| **Flow of Events** | Basic | 1 | User selects a university from the recommendation list. |
| 2 | System retrieves detailed information about the selected university. |
| 3 | System displays admission criteria, deadlines, and application links. |
| Alternative | User requests more details for a university: system displays extended details, if available. | |
| Exceptions | Information for selected university is incomplete: system prompts the user to check back later. | |
| **Notes / Issues** | Confirm application links are accurate and functional. | | |

### 3.3.14 Use Case # 7.1

|  |  |  |  |
| --- | --- | --- | --- |
| **7.1** | **Display Field Trends** | | |
| **Author** | Fizza Mazhar | | |
| **Purpose** | Provide graphical visualizations of trends for growing and declining fields. | | |
| **Requirements Traceability** | F7.1 | | |
| **Priority** | Medium | | |
| **Preconditions** | Historical trend data for various fields must be available in the system. | | |
| **Postconditions** | Users can view graphical representations of field trends. | | |
| **Actors** | User (Student or Father), System | | |
| **Flow of Events** | Basic | 1 | User selects a field to view trends. |
| 2 | System retrieves historical data for the selected field. |
| 3 | System generates a graphical visualization (e.g., bar chart, line graph). |
| 4 | System displays the visualization to the user. |
| Alternative | User selects a time range for trends: system updates the graph based on the chosen time frame. | |
| Exceptions | Trend data unavailable for the selected field: system informs the user and suggests alternatives. | |
| **Notes / Issues** | Ensure visualizations are clear, intuitive, and easy to interpret. | | |

### 3.3.15 Use Case # 7.2

|  |  |  |  |
| --- | --- | --- | --- |
| **7.3** | **Compare Fields and View Demand Trends** | | |
| **Author** | Fizza Mazhar | | |
| **Purpose** | Allow users to compare multiple fields and view demand trends. | | |
| **Requirements Traceability** | F7.2 | | |
| **Priority** | High | | |
| **Preconditions** | Field trend data and comparison mechanisms must be ready for processing. | | |
| **Postconditions** | Users can compare fields and make informed decisions based on demand trends. | | |
| **Actors** | User (Student or Father), System | | |
| **Flow of Events** | Basic | 1 | User selects multiple fields for comparison. |
| 2 | System retrieves trend data for selected fields. |
| 3 | System generates a comparative graphical visualization. |
| 4 | System displays the comparison to the user. |
| Alternative | User specifies a country for demand trends: system filters the data accordingly. | |
| Exceptions | Comparison data unavailable for one or more selected fields: system adjusts the display or notifies the user. | |

### 3.3.16 Use Case # 8.1

|  |  |  |  |
| --- | --- | --- | --- |
| **8.1** | **View Personalized Recommendations** | | |
| **Author** | Fizza Mazhar | | |
| **Purpose** | Provide users with a personalized dashboard displaying recommended fields and universities. | | |
| **Requirements Traceability** | F8.1 | | |
| **Priority** | High | | |
| **Preconditions** | User must have completed data input, and recommendations must be generated. | | |
| **Postconditions** | Users can view their personalized recommendations. | | |
| **Actors** | User (Student or Father), System | | |
| **Flow of Events** | Basic | 1 | User logs into the system. |
| 2 | System retrieves personalized field and university recommendations. |
| 3 | System displays recommendations on the dashboard. |
| 4 | User interacts with recommendations for details. |
| Alternative | User customizes recommendation criteria: system updates recommendations dynamically. | |
| Exceptions | Recommendations cannot be generated due to incomplete data: system prompts user to complete inputs. | |

### 3.3.17 Use Case # 8.2

|  |  |  |  |
| --- | --- | --- | --- |
| **8.2** | **Visualize Trends and Comparisons** | | |
| **Author** | Fizza Mazhar | | |
| **Purpose** | Enable users to visualize trends and compare different fields on the dashboard. | | |
| **Requirements Traceability** | F8.2 | | |
| **Priority** | Medium | | |
| **Preconditions** | Trend data and comparison functionalities must be available in the system. | | |
| **Postconditions** | Users can see trends and comparisons of fields in an intuitive graphical format. | | |
| **Actors** | User (Student or Father), System | | |
| **Flow of Events** | Basic | 1 | User navigates to the trends section of the dashboard. |
| 2 | System retrieves relevant trend and comparison data. |
| 3 | System generates graphical visualizations. |
| 4 | System displays visualizations to the user. |
| Alternative | User filters trend data by specific parameters (e.g. time, region): system updates graphs. | |
| Exceptions | Trend data unavailable for selected filters: system notifies the user and suggests adjustments. | |

### 3.3.18 Use Case # 9.1

|  |  |  |  |
| --- | --- | --- | --- |
| **9.1** | **Search Universities** | | |
| **Author** | Fizza Mazhar | | |
| **Purpose** | Allow users to search for universities based on specific keywords or criteria. | | |
| **Requirements Traceability** | F9.1 | | |
| **Priority** | High | | |
| **Preconditions** | University data must be indexed and searchable in the system. | | |
| **Postconditions** | The system displays a list of universities matching the search query. | | |
| **Actors** | User (Student or Father), System | | |
| **Flow of Events** | Basic | 1 | User enters a keyword or phrase in the search bar. |
| 2 | System processes the query. |
| 3 | System retrieves matching universities. |
| 4 | System displays the search results. |
| Alternative | User enters incomplete or vague keywords: system suggests related terms or provides broader results. | |
| Exceptions | No universities match the search query: system informs the user and suggests alternative keywords. | |

### 3.3.19 Use Case # 9.2

|  |  |  |  |
| --- | --- | --- | --- |
| **9.2** | **Filter Universities** | | |
| **Author** | Fizza Mazhar | | |
| **Purpose** | Enable users to filter university options based on location, aggregate eligibility. | | |
| **Requirements Traceability** | F9.2 | | |
| **Priority** | High | | |
| **Preconditions** | User must have selected at least one filtering criterion. | | |
| **Postconditions** | The system displays a filtered list of universities based on the selected criteria. | | |
| **Actors** | User (Student or Father), System | | |
| **Flow of Events** | Basic | 1 | User selects filtering options (e.g. location, aggregate eligibility). |
| 2 | System applies the filters to the university database. |
| 3 | System displays the filtered results to the user. |
| Alternative | User selects multiple conflicting filters: system resolves conflicts or notifies the user. | |
| Exceptions | Filter criteria do not return any results: system prompts user to adjust filters. | |

# Other Non-functional Requirements

## 4.1 Performance Requirements

The performance requirements for the Career Counseling Platform are designed to ensure the system operates efficiently under various circumstances and provides a seamless experience for users. These requirements are specified to guide developers in making suitable design choices.

### ****4.1.1 Response Time****

The system shall provide a response time of **4 seconds or less** for user interactions, such as:

1. Displaying field recommendations.
2. Generating aggregate calculations.
3. Rendering graphs and visualizations on the dashboard.

### ****4.1.2 Concurrent Users****

The platform shall support up to **1000 concurrent users** without degradation in performance.

### ****4.1.3 Data Processing****

The system shall process user data (e.g., marks, preferences, interests) and provide results (e.g., field and university recommendations) within **3 seconds** after submission.

### ****4.1.4 Dashboard Loading****

The dashboard (with trends and visualizations) shall load completely within **4 seconds** on a stable internet connection (5 Mbps or higher).

### ****4.1.5 Database Query Performance****

All database queries (e.g., fetching university details, admission deadlines, trend data) shall execute within **1 second**.

### ****4.1.6 Login and Signup****

1. The system shall authenticate user credentials or create a new account within **2 seconds**.
2. If using Google OAuth, the process shall complete within **3 seconds**, including the external API response.

### ****4.1.7 Aggregate Calculation****

The system shall calculate the aggregate and display the results within **1 second** of receiving the required input.

### ****4.1.8 Trend Visualization****

Graphical visualizations (e.g., bar charts, line graphs) shall render within **2 seconds** of user interaction or query submission.

### ****4.1.9 University Search and Filter****

The system shall display filtered university results based on the user's criteria (e.g., location, eligibility) within **3 seconds**.

### ****4.1.10 Peak Load****

The system shall maintain performance under peak load conditions of **5000 users/hour** during university admission periods.

### ****4.1.11 Scalability****

The system shall scale to support up to **10000 concurrent users** in the future with minimal reconfiguration.

### ****4.1.12 Data Transfer****

The system shall handle real-time data transfer (e.g., between front-end and back-end) with a latency of **1 seconds or less**.

## 4.2 Safety and Security Requirements

The Career Counseling Platform must ensure the safety of its users and secure sensitive user data to prevent unauthorized access, data breaches or harm caused by improper system usage. The following safety and security requirements are specified:

### ****4.2.1 Data Integrity****

1. The system shall ensure that all user data, including academic marks and preferences is stored and processed accurately.
2. Any system errors or failures shall not lead to the corruption or loss of user data.

### ****4.2.2 Safeguards Against Unauthorized Actions****

1. The system shall prevent unauthorized users from accessing or modifying user data.
2. The platform shall implement session timeouts after **15 minutes** of inactivity to prevent misuse in shared or public devices.

### ****4.2.3 System Downtime Protection****

1. The system shall have a mechanism to inform users of scheduled maintenance or downtime.
2. Any user-submitted data during an outage shall be saved and processed when the system is restored.

### ****4.2.4 Backup and Recovery****

1. The system shall automatically back up critical data daily to ensure recovery in case of a disaster.
2. The recovery time objective (RTO) shall be no more than **2 hours**.

### ****4.2.5 User Authentication****

1. The system shall implement secure user authentication mechanisms, including:
   * Password-based login with strong password policies (minimum 8 characters, mix of uppercase, lowercase, numbers, and symbols).
   * Google OAuth for secure third-party authentication.
2. All passwords shall be stored as **hashed and salted values**.

### ****4.2.6 Access Control****

### The system shall restrict access to administrative functionalities and user data to authorized personnel only.

### Role-based access control (RBAC) shall be implemented for admins and users.

### ****4.2.7 Protection Against Common Attacks****

The system shall safeguard against the **SQL Injection attack** by using parameterized queries to prevent malicious database queries.

### ****4.2.8 User Privacy****

1. The system shall provide users with detailed privacy policies outlining how their data is used and stored.
2. Users shall have the right to request the deletion of their data from the platform.

### ****4.2.9 Audit Logs****

The system shall maintain logs of all critical actions, including login attempts, data modifications, and admin actions for auditing purposes.

## 4.3 Software Quality Attributes

### ****4.3 Software Quality Attributes****

The Career Counseling Platform must adhere to high software quality standards to meet user and developer expectations. Below are the specified quality attributes and how they will be achieved:

### ****4.3.1 Reliability****

The platform shall operate consistently and deliver accurate results with minimal downtime or errors.

1. Use exception handling mechanisms to gracefully manage unexpected errors.
2. Provide user-friendly error messages in case of failures.
3. Conduct unit tests, integration tests, and regression tests to ensure the system functions as expected under various scenarios.
4. Perform load and stress testing to verify the system can handle peak traffic conditions (e.g., during university admissions).
5. Maintain data redundancy and automated daily backups to ensure data integrity during failures.
6. Use tools like New Relic or Pingdom to monitor system uptime and address issues immediately.

### ****4.3.2 Usability****

The platform shall be user-friendly, intuitive and accessible to ensure ease of use for students, parents and counselors.

1. Use frameworks like Bootstrap or Material-UI to ensure the platform adapts to various devices (e.g., desktops, tablets, mobile phones).
2. Provide clear and logical navigation with labeled buttons, breadcrumbs, and menus.
3. Follow **WCAG 2.1 guidelines** for accessibility to ensure the platform is usable by people with disabilities.
4. Conduct user testing with a sample audience to identify and address usability challenges.
5. Incorporate feedback loops for continuous improvement.
6. Use data visualization libraries (e.g., Chart.js) to create interactive graphs for trends and comparisons.

### ****4.3.3 Maintainability****

The platform shall be designed to accommodate future updates and modifications with minimal effort.

1. Use a microservices architecture to decouple components (e.g., authentication, recommendation engine, visualization module).
2. Use Git for version control to track changes and facilitate collaborative development.
3. Employ CI/CD pipelines to ensure quick and reliable deployment of updates.
4. Maintain comprehensive documentation for developers to understand system components, workflows and APIs.

### ****4.3.4 Adaptability****

1. The platform shall be adaptable to handle changes, such as adding new datasets or integrating with additional systems.
2. Design flexible database schemas to accommodate new fields or datasets without major restructuring.
3. Parameterize key rules (e.g. aggregate formulas, eligibility criteria) to allow changes without modifying code.

# Design Requirements

## 5.1 High Level Design

### 5.1.1 Abstract Design

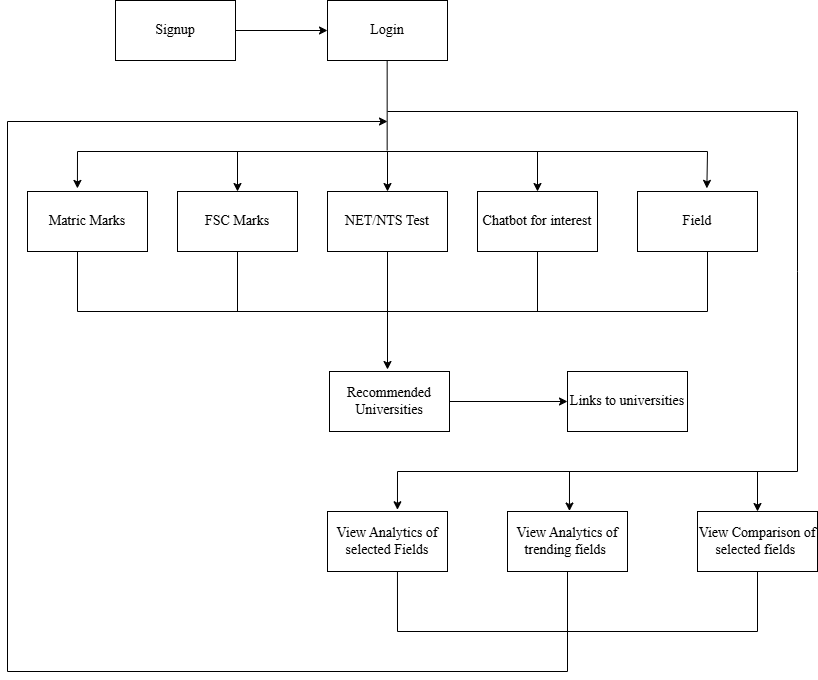


Figure 8

### 5.1.2 Design oblige Architecture Patterns

**Layered Architecture Pattern**

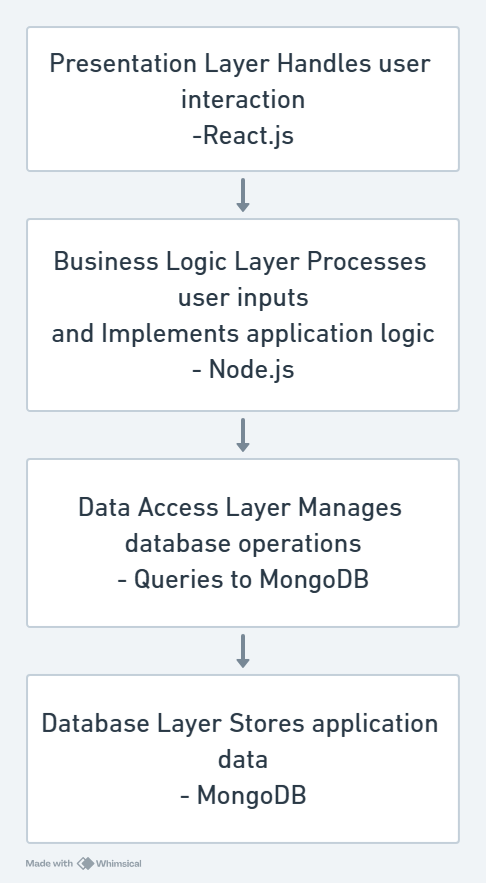
The system is divided into layers as shown in Figure 9, each responsible for specific concerns. Common layers include:

1. **Presentation Layer**: Handles user interaction and displays data (e.g., React.js or Angular for the frontend).
2. **Business Logic Layer**: Processes user inputs and implements application logic (e.g., Node.js or Django for the backend).
3. **Data Access Layer**: Manages database operations (e.g., queries to MySQL/PostgreSQL).
4. **Database Layer**: Stores user data, university details, trends, etc.

Benefits include

1. Encourages separation of concerns, making the system maintainable and scalable.
2. Easy to test and modify individual layers without impacting others.

**Example**: When a user inputs their marks, the Presentation Layer sends the data to the Business Logic Layer, which calculates the aggregate and fetches relevant recommendations from the Data Access Layer.

****

**Microservices Architecture Pattern**

The platform is divided into small, independent services, each handling a specific functionality, such as:

1. **Authentication Service**: Handles user login and signup.
2. **Recommendation Engine**: Suggests fields and universities.
3. **Visualization Service**: Renders trends and comparisons.
4. **Notification Service**: Sends email alerts for deadlines.

Benefits include

1. Individual services can be scaled independently.
2. If one service fails, others remain unaffected.
3. Different services can be implemented in different programming languages or technologies.

**Model View Controller Pattern**

Organizes the platform into three components:

1. **Model**: Manages data and business rules (e.g., aggregate calculations, database interactions).
2. **View**: Handles the presentation layer (e.g., rendering user interfaces).
3. **Controller**: Processes user requests, interacts with the model, and selects the appropriate view.

Benefits include

1. Simplifies development by clearly separating data, logic and presentation.
2. Enhances testability and maintainability.

### ****Event-Driven Architecture Pattern****

The platform reacts to events generated by users or the system itself.

1. User inputs data for aggregate calculation.
2. Deadline for university admission is approaching (triggering an email notification).
3. User compares fields, triggering a request to the visualization service.

Benefits include

1. Asynchronous: Improves performance by processing events independently.
2. Scalable: Easily handles high traffic or multiple concurrent user interactions.

### ****Dependency Injection Pattern****

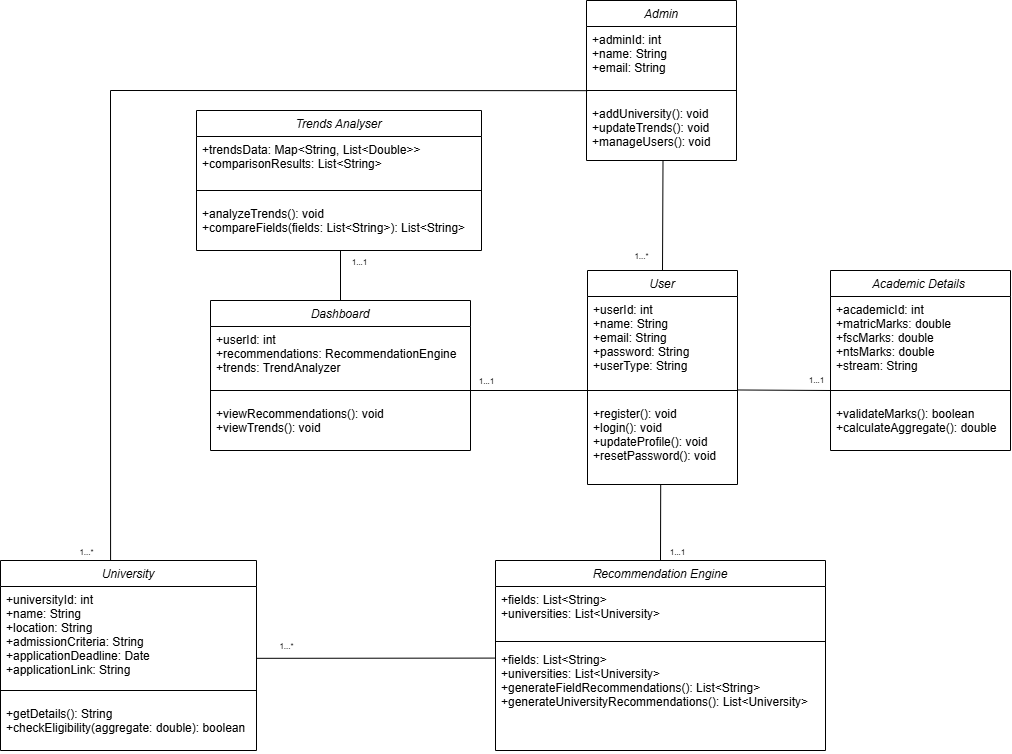
A design pattern where dependencies (e.g., services, repositories) are injected into components rather than being hardcoded.

Benefits Include

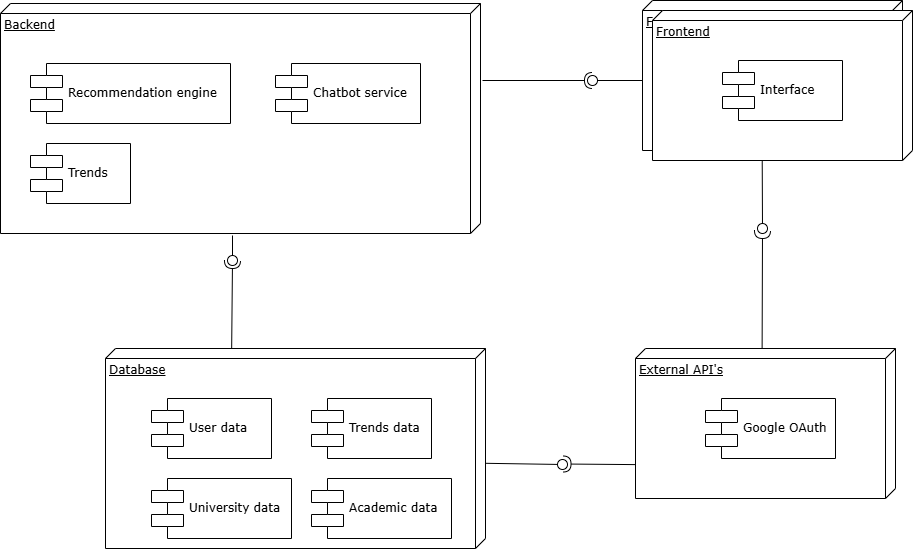
1. Increases testability by allowing mock dependencies.
2. Reduces coupling, making the system easier to modify and extend.

## 5.2 Structural Design

### 5.2.1 Class Diagram

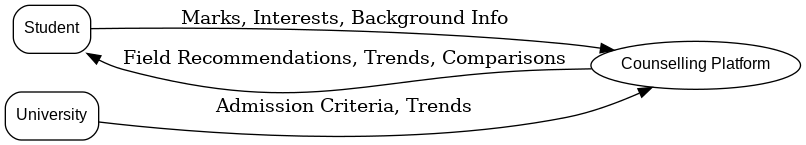


### 5.2.2 Component Diagram

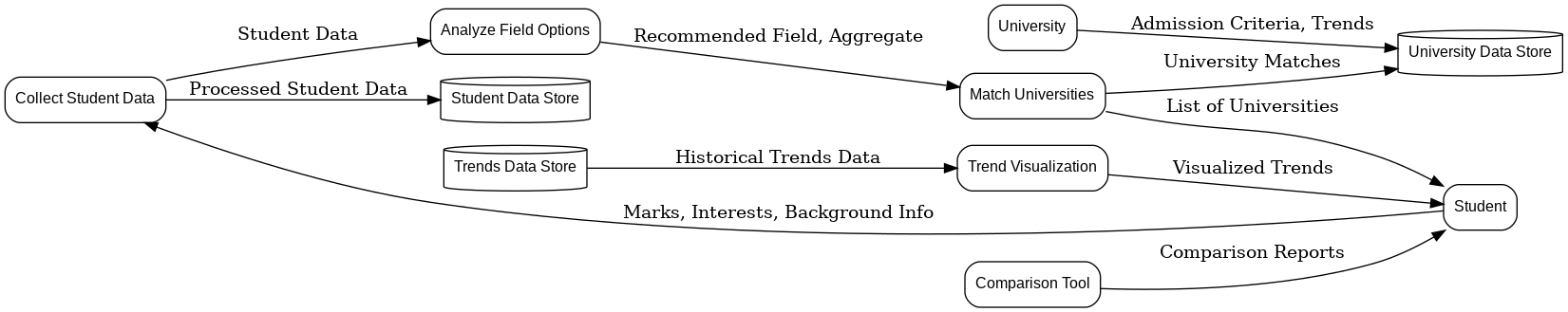


### 5.2.3 Data Flow Diagram

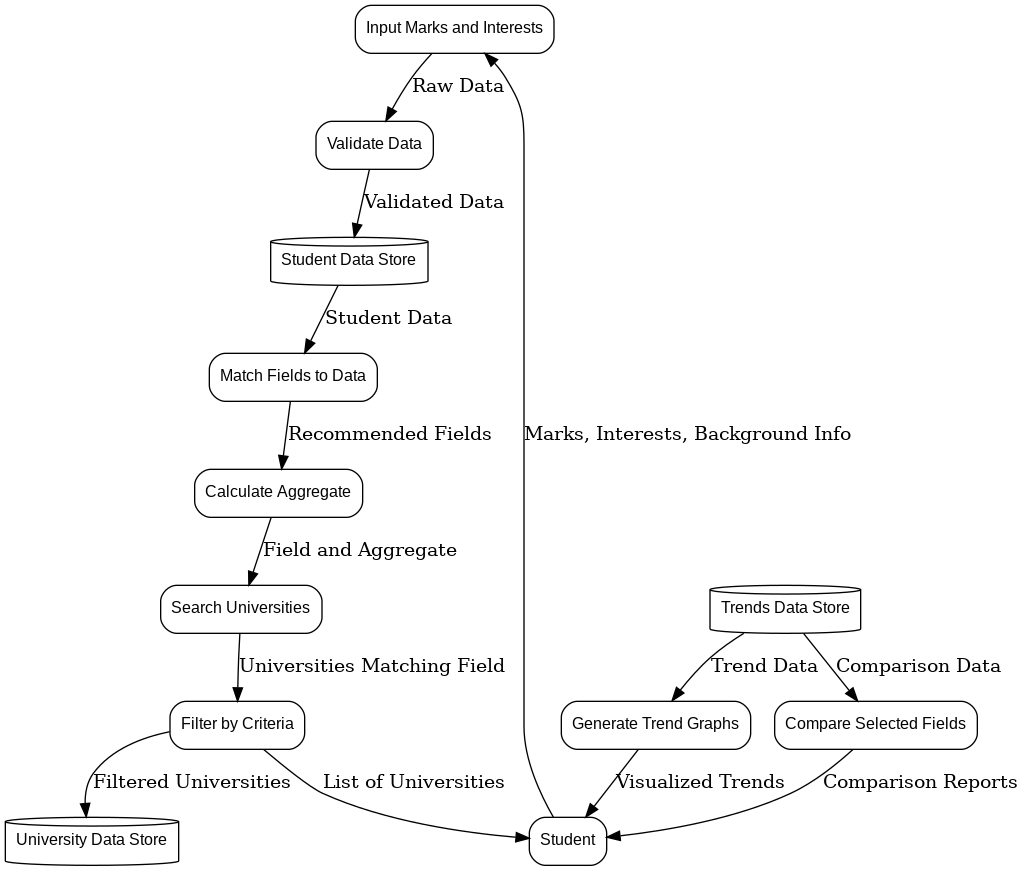
**Context Level DFD**



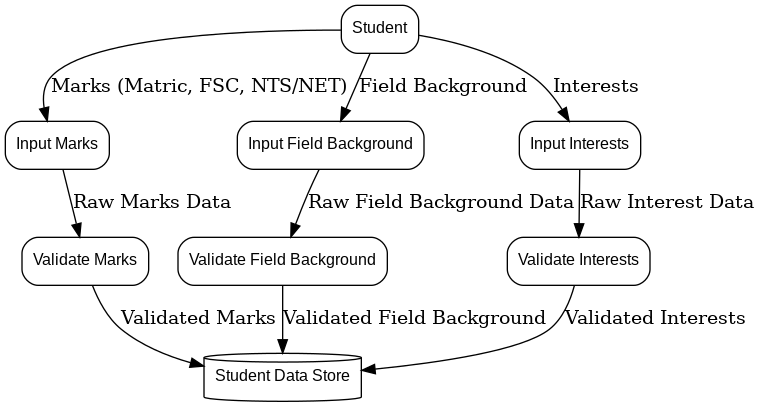
**Level 1 DFD**



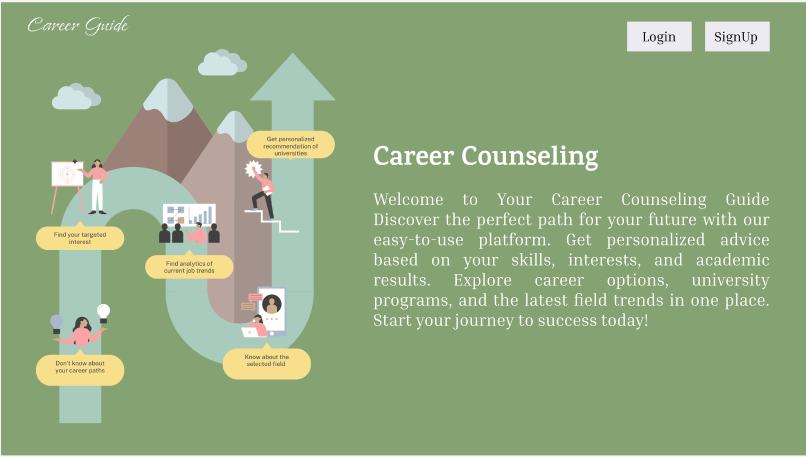
**Level 2 DFD**

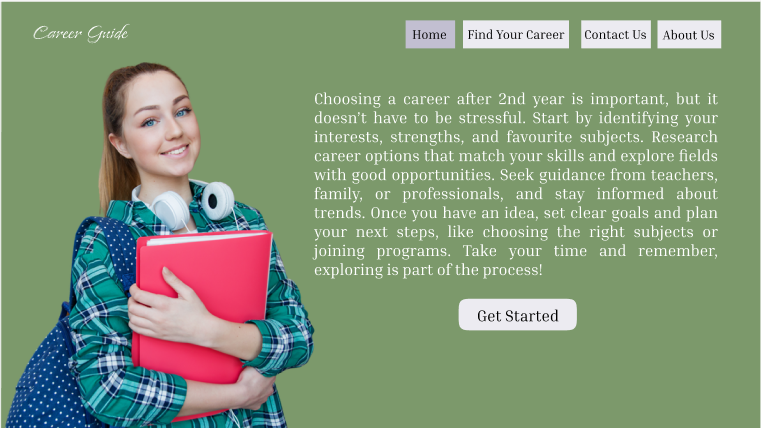


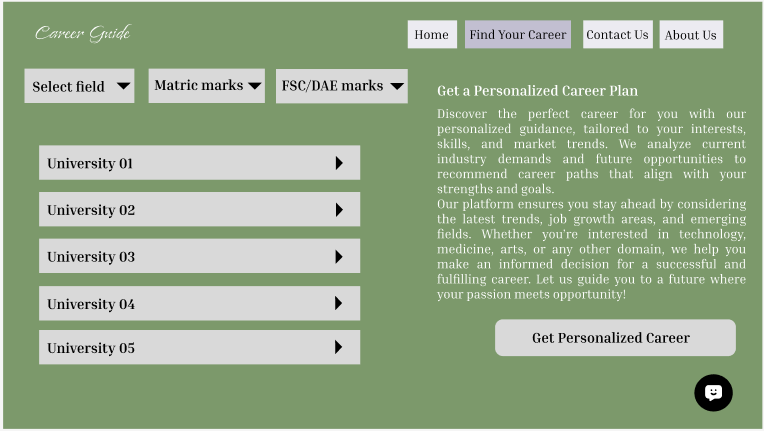
**Level 3 DFD**

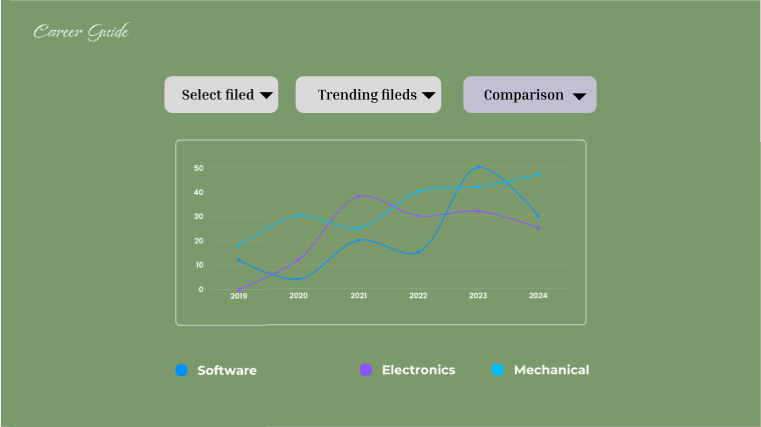


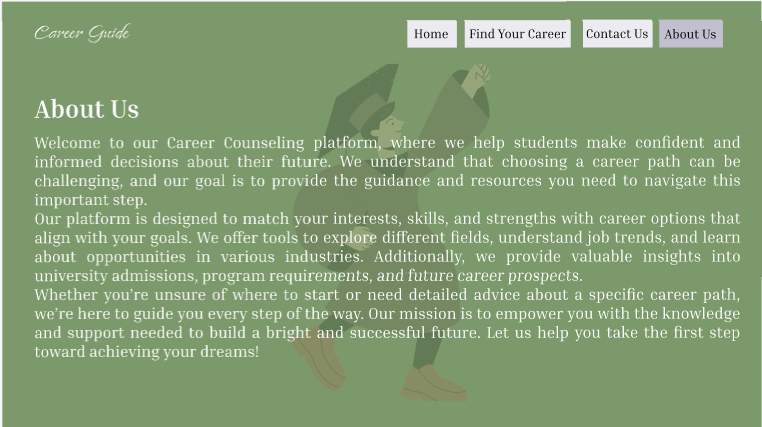
### 5.2.4 Graphical User Interface

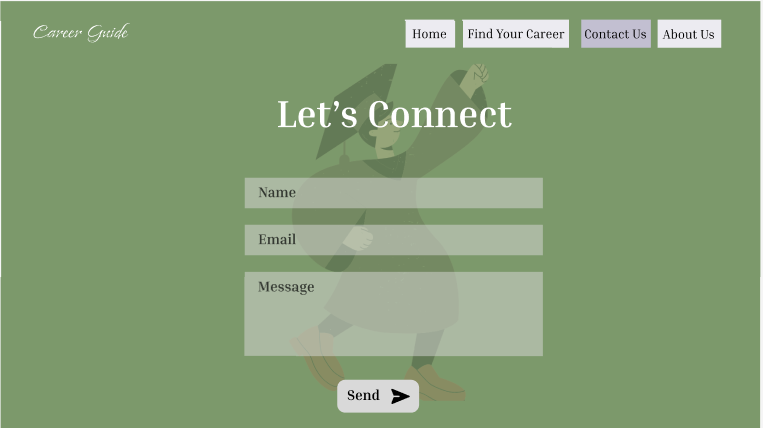






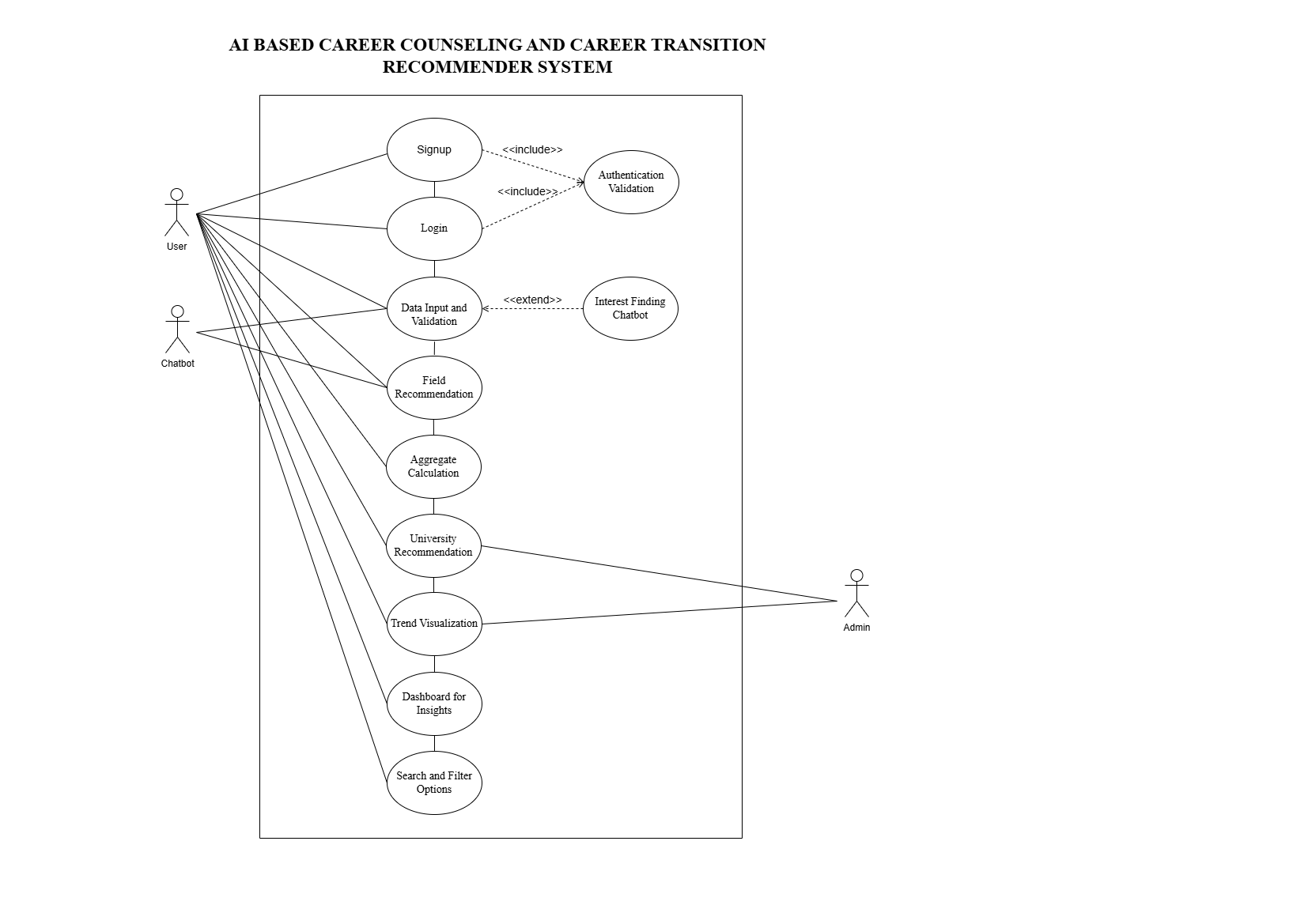


****

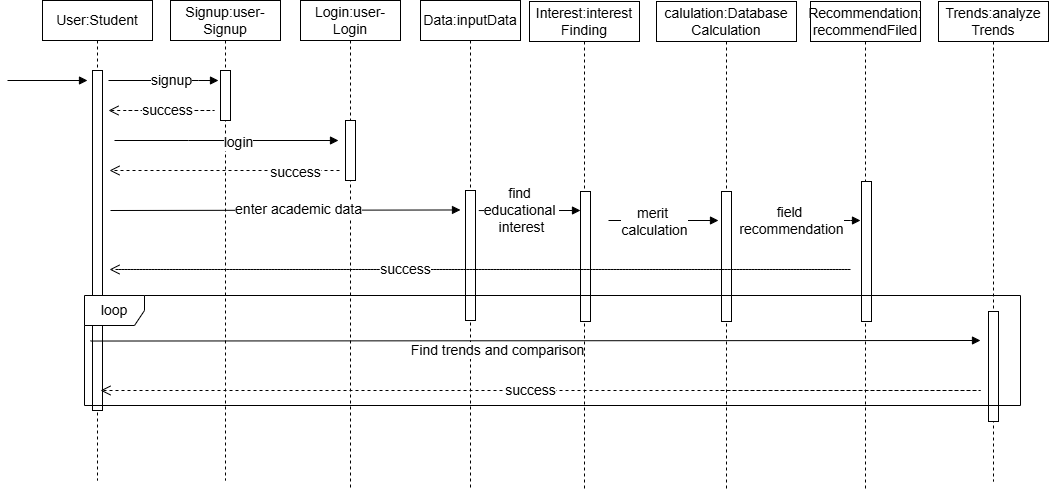


## 5.3 Behavioral Diagram

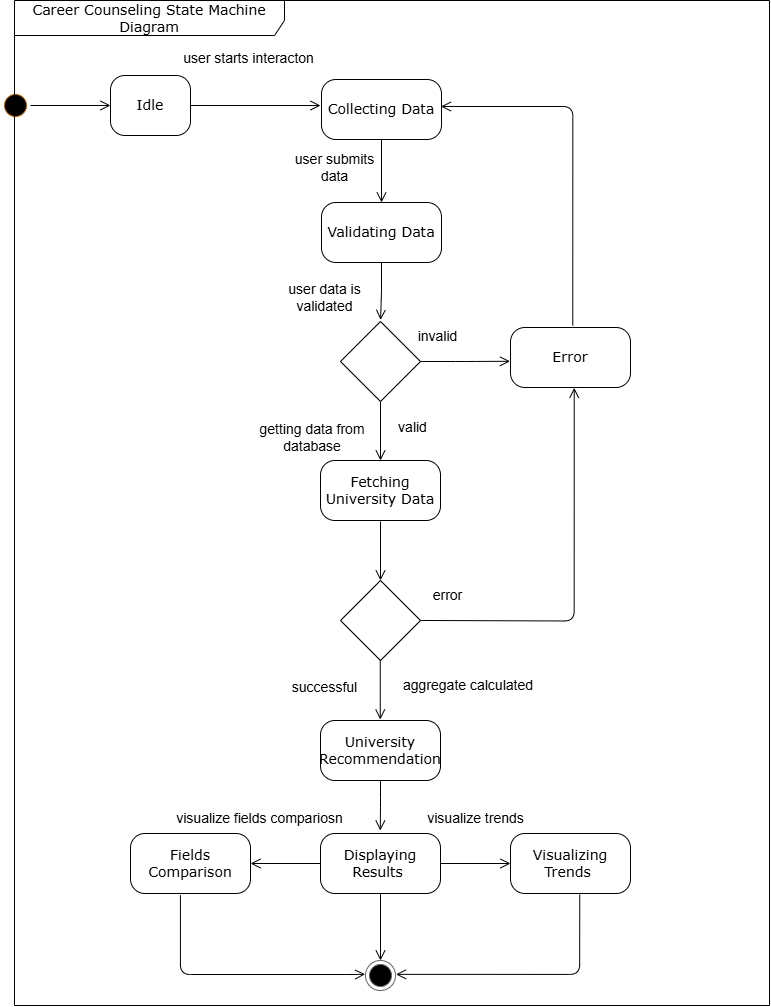
### 5.3.1 Use Case Diagram



### 5.3.2 Sequence Diagram



### 5.3.3 State Machine Diagram



# Data Design and Relationships

## 6.1 Database Schema

**User Table**

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Constraints** |
| user\_id | INT | PRIMARY KEY, AUTO\_INCREMENT |
| Name | VARCHAR(255) | NOT NULL |
| email | VARCHAR(255) | UNIQUE, NOT NULL |
| password | VARCHAR(255) | NOT NULL |
| user\_type | ENUM('student', 'admin') | DEFAULT 'student' |
| created\_at | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP |

**Academic Details Table**

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Constraints** |
| academic\_id | INT | PRIMARY KEY, AUTO\_INCREMENT |
| user\_id | INT | FOREIGN KEY (Users.user\_id) |
| matric\_marks | FLOAT | NOT NULL |
| fsc\_marks | FLOAT | NOT NULL |
| nts\_marks | FLOAT | NULLABLE |
| stream | ENUM('Pre-Medical', 'Pre-Engineering', 'FCS') | NOT NULL |
| aggregate | FLOAT | NULLABLE |

**Recommendations Table**

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Constraints** |
| recommendation\_id | INT | PRIMARY KEY, AUTO\_INCREMENT |
| user\_id | INT | FOREIGN KEY (Users.user\_id) |
| field | VARCHAR(255) | NOT NULL |
| university\_id | INT | FOREIGN KEY (Universities.university\_id) |

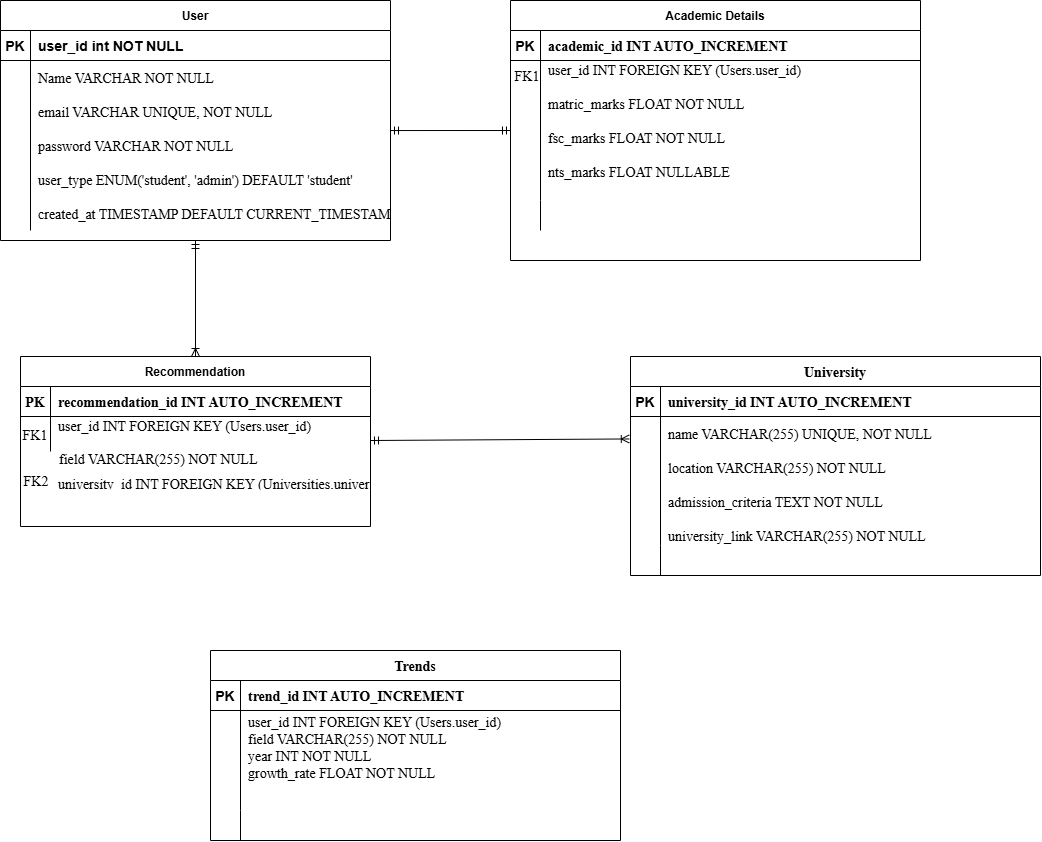
**Trends Table**

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Constraints** |
| trend\_id | INT | PRIMARY KEY, AUTO\_INCREMENT |
| field | VARCHAR(255) | NOT NULL |
| year | INT | NOT NULL |
| growth\_rate | FLOAT | NOT NULL |

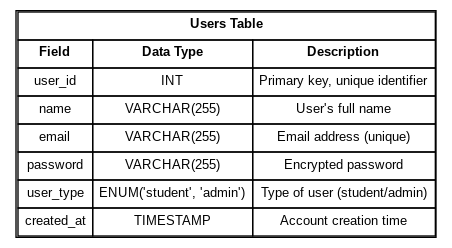
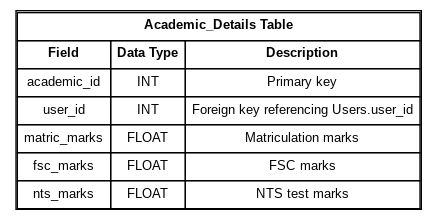
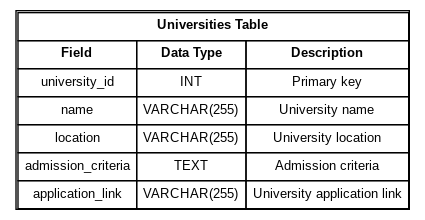
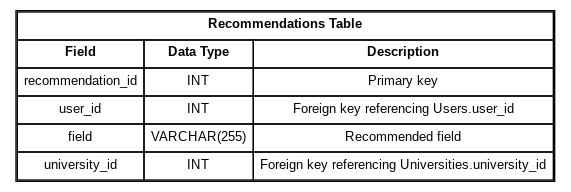
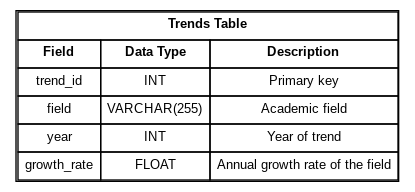
**Universities Table**

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Constraints** |
| university\_id | INT | PRIMARY KEY, AUTO\_INCREMENT |
| name | VARCHAR(255) | UNIQUE, NOT NULL |
| location | VARCHAR(255) | NOT NULL |
| admission\_criteria | TEXT | NOT NULL |
| university\_link | VARCHAR(255) | NOT NULL |

## 6.2 Entity Relationship Diagram



## 6.3 Data Dictionary Diagram

****

# Software Planning and Timeline

## 7.1 Work Breakdown Structure

|  |
| --- |
| **AI Based Career Counselling and Career Transition Recommender System** |
| 1. **Planning and Research** |
|  |
| * + 1. Identify user needs and project goals. |
| * + 1. Research university admission criteria and fields of study. |
| * + 1. Collect data for trends and comparisons (2000–2004 dataset). |
| * 1. **Project Scope Definition** |
| * + 1. Define features and functionalities. |
| * + 1. Outline system architecture. |
| * 1. **Timeline and Resources** |
| * + 1. Develop project timeline and resource allocation |
| 1. **Backend Development** |
| 1. **Database Design** |
| * 1. Create schema for user data, university details, trends, and field comparisons. |
| 1. **Backend Logic** |
| * 1. Develop Algorithm for field recommendations and university suggestions. |
| * 1. Implement aggregate calculation algorithms. |
| 1. **Integration of Trends and Comparisons** |
| * 1. Process and store trend data for dashboard visualizations. |
| 1. **Testing Backend Functions** |
| * 1. Ensure accuracy of recommendations and API functionality. |
| 1. **Frontend Development** |
| 1. **User Interface Design** |
| * 1. Design intuitive layouts for form submissions and dashboard. |
| 1. **Frontend Implementation** |
| * 1. Develop forms for data input (marks, expected scores, interests). |
| * 1. Build dynamic recommendation pages for fields and universities. |
| * 1. Implement dashboard for trend visualizations and comparisons. |
| 1. **Frontend-Backend Integration** |
| * 1. Connect APIs to frontend components. |
| 1. **Data Visualization and Insights** |
| 1. **Trend Analysis** |
| * 1. Generate graphs for growing and declining fields. |
| * 1. Display global demand for respective fields. |
| 1. **Comparison Tools** |
| * 1. Implement feature for comparing fields based on user-selected criteria. |
| 1. **Testing and Validation** |
| 1. **System Testing** |
| * 1. Perform end-to-end testing of user flows and functionalities. |
| 1. **User Feedback Collection** |
| * 1. Conduct beta testing with students and educators. |
| 1. **Bug Fixing and Optimization** |
| * 1. Resolve issues identified during testing. |
| 1. **Deployment and Launch** |
| 1. **Hosting and Deployment** |
| * 1. Deploy website to a hosting platform. |
| 1. **User Documentation** |
| * 1. Create guides for students on using the platform. |
| 1. **Post-Launch Activities** |
| 1. **Monitoring and Maintenance** |
| * 1. Monitor platform performance and user feedback. |
| * 1. Make updates and improvements. |
| 1. **Promotion and Outreach** |
| * 1. Promote the platform to schools and through social media. |

## 7.2 Milestones and Deliverables

### ****Milestone 1: Planning and Research****

#### Deliverables:

1. Identified user needs and project goals.
2. Research report on university admission criteria and fields of study.
3. Dataset for trends and comparisons (2000–2004 dataset).

### ****Milestone 2: Project Scope Definition****

#### Deliverables:

1. Documented features and functionalities.
2. System architecture blueprint.

### ****Milestone 3: Timeline and Resource Allocation****

#### Deliverables:

1. Project timeline with key phases and deadlines.
2. Resource allocation plan (team roles, tools, and technologies).

### ****Milestone 4: Backend Development****

#### **Sub-Milestone 4.1: Database Design**

Deliverables:

* 1. Schema for user data, university details, trends, and field comparisons.

#### **Sub-Milestone 4.2: Backend Logic**

Deliverables:

* 1. Algorithm for field recommendations and university suggestions.
  2. Aggregate calculation algorithm.

#### **Sub-Milestone 4.3: Integration of Trends and Comparisons**

Deliverables:

* 1. Processed and stored trend data for dashboard visualizations.

#### **Sub-Milestone 4.4: Backend Testing**

Deliverables:

1. Tested backend functionality ensuring accuracy of recommendations and API functionality.

### ****Milestone 5: Frontend Development****

#### **Sub-Milestone 5.1: User Interface Design**

Deliverables:

* 1. Wireframes and intuitive layouts for form submissions and dashboard.

#### **Sub-Milestone 5.2: Frontend Implementation**

Deliverables:

1. Data input forms (marks, expected scores, interests).
2. Dynamic pages for field and university recommendations.
3. Dashboard for trend visualizations and comparisons.

#### **Sub-Milestone 5.3: Frontend-Backend Integration**

Deliverables:

1. Functional APIs connected to frontend components.

### ****Milestone 6: Data Visualization and Insights****

#### **Sub-Milestone 6.1: Trend Analysis**

Deliverables:

1. Graphs for growing and declining fields.
2. Visualization of global demand for respective fields.

#### **Sub-Milestone 6.2: Comparison Tools**

Deliverables:

* 1. Feature for comparing fields based on user-selected criteria.

### ****Milestone 7: Testing and Validation****

#### **Sub-Milestone 7.1: System Testing**

Deliverables:

* 1. End-to-end tested workflows and functionalities.

#### **Sub-Milestone 7.2: User Feedback Collection**

Deliverables:

* 1. Feedback report from beta testing with students and educators.

#### **Sub-Milestone 7.3: Bug Fixing and Optimization**

Deliverables:

* 1. Resolved issues and optimized system performance.

### ****Milestone 8: Deployment and Launch****

#### **Sub-Milestone 8.1: Hosting and Deployment**

Deliverables:

* 1. Live website deployed to a hosting platform.

#### **Sub-Milestone 8.2: User Documentation**

Deliverables:

* 1. User guides and tutorials for using the platform.

### ****Milestone 9: Post-Launch Activities****

#### **Sub-Milestone 9.1: Monitoring and Maintenance**

Deliverables:

* 1. Performance monitoring reports.
  2. Updates and improvements based on user feedback.

#### **Sub-Milestone 9.2: Promotion and Outreach**

Deliverables:

* 1. Marketing materials (e.g., brochures, posters, social media campaigns).
  2. Outreach program for schools.

## 7.3 Detailed Baseline Plan

# Quality Assurance Plan

## 8.1 Testing Requirements

The following are the detailed **testing requirements** for the system, categorized based on various testing types to ensure its functionality, performance, security, usability and overall reliability.

### ****8.1.1 Functional Testing****

1. **Verify user registration and login functionality:** 
   1. Ensure successful account creation and secure authentication.
   2. Validate Google OAuth integration.
2. **Test data input forms:** 
   1. Allow users to input academic marks, expected scores and preferences.
   2. Display error messages for invalid data entries.
3. **Validate field recommendation system:** 
   1. Ensure recommendations align with user input (marks, streams, interests).
4. **Validate university recommendation system:** 
   1. Ensure universities match calculated aggregates and eligibility criteria.
5. **Test trend analysis and visualizations:** 
   1. Ensure trends in growing and declining fields are displayed accurately.
6. **Validate comparison tools:** 
   1. Allow users to compare fields based on selected criteria.

### ****8.1.2 Performance Testing****

1. **Load Testing**:
   1. Ensure the system handles **1,000 concurrent users** without performance degradation.
2. **Stress Testing**:
   1. Determine system performance during peak usage (e.g., 5,000 users/hour).
3. **Scalability Testing**:
   1. Test scalability for future growth up to **10,000 concurrent users**.
4. **Response Time Testing**:
   1. Ensure recommendations and visualizations are displayed within **2 seconds**.

### ****8.1.3 Security Testing****

1. **Test authentication mechanisms:** 
   1. Validate password encryption and secure storage.
   2. Verify Google OAuth token handling.
2. **Test data security**:
   1. Ensure data in transit is encrypted with **SSL/TLS protocols**.
   2. Verify sensitive data (e.g., user marks) is encrypted at rest using **AES-256**.
3. **Identify vulnerabilities to common attacks:** 
   1. Protect against **SQL injection**, **XSS**, **CSRF**, and **DDoS**.
4. **Test role-based access control:** 
   1. Ensure only authorized users (e.g., admins) can access restricted areas.

### ****8.1.4 Usability Testing****

1. **Test navigation:** 
   1. Verify all pages are easily accessible with a clear navigation flow.
2. **Test responsive design:** 
   1. Ensure the platform works seamlessly across devices (mobile, tablet, desktop).
3. **Conduct user testing:** 
   1. Collect feedback from a focus group of students and educators.
4. **Ensure accessibility:** 
   1. Follow **WCAG 2.1 guidelines** for accessibility.

### ****8.1.5 Integration Testing****

1. **Test frontend-backend integration:** 
   1. Ensure APIs correctly communicate with frontend components.
2. **Validate API functionality:** 
   1. Test API responses for recommendations, trends, and comparisons.
3. **Test external API integration:** 
   1. Verify Google OAuth and external university data API connections.

### ****8.1.6 System Testing****

1. **Test complete user workflows:** 
   1. Registration → Data Input → Recommendations → Visualizations.
2. **Verify system requirements:** 
   1. Ensure recommendations and visualizations align with project goals.
3. **Perform boundary testing:** 
   1. Test the system with minimum and maximum data inputs.

### ****8.1.7 Regression Testing****

1. Retest all workflows after updates.
2. Maintain automated regression test scripts for critical features.

### ****8.1.8 User Acceptance Testing (UAT)****

1. **Conduct beta testing:** 
   1. Invite students and educators to use the platform and provide feedback.
2. **Validate user experience:** 
   1. Test ease of use, accuracy of recommendations, and clarity of insights.

## 8.2 Acceptance Criteria

Acceptance criteria define the specific conditions that must be met for the system to be considered complete, functional and acceptable to the stakeholders (students and educators). Below are the key acceptance criteria categorized by system features and functionalities

### ****8.2.1 User Registration and Login****

1. Users must be able to create an account using their email and password.
2. Google OAuth integration must allow users to log in seamlessly.
3. Validation
   * Invalid email or password should prompt an appropriate error message.
   * Secure password storage using encryption must be implemented.
4. Success Criteria
   * User successfully logs in and is redirected to their personalized dashboard.

### ****8.2.2 Data Input and Validation****

1. The system must allow users to input their **Matric**, **FSC**, and **NTS** or **NET** scores.
2. Users must have the option to enter **expected marks** if their results are pending.
3. Validation to
   * Ensure only numeric values are accepted for marks.
   * Notify users if any required fields are missing or invalid.
4. Success Criteria
   * User data is saved successfully, and validated input is processed without errors.

### ****8.2.3 Field Recommendation****

1. The system must provide a list of potential fields based on
   * Marks entered by the user.
   * Selected study stream (e.g., Pre-Medical, Pre-Engineering).
   * Stated interests (if provided via chatbot interaction).
2. Success Criteria
   * Recommendations align with user inputs and expected trends.

### ****8.2.4 University Recommendation****

1. The system must recommend universities based on
   * Calculated aggregate scores.
   * University admission criteria and eligibility.
2. Recommendations must include
   * Admission deadlines.
   * Links to online application portals.
   * Information on university locations.
3. Success Criteria
   * Recommendations are accurate, personalized and relevant.

### ****8.2.5 Trend Analysis and Visualization****

1. The dashboard must display visualizations showing
   * Trends for growing and declining fields based on historical data.
   * Global demand for respective fields.
2. Success Criteria
   * Graphs and charts are interactive, clear, and provide meaningful insights to users.

### ****8.2.6 Comparison Tools****

1. Users must be able to compare multiple fields of study based on
   * Growth trends.
2. Success Criteria
   * Users can easily select fields to compare, and the system provides accurate results with visual aids.

### ****8.2.7 Chatbot Interaction****

1. The chatbot must allow users to
   * Discover their interests through guided questions.
   * Receive recommendations based on their stated preferences.
2. Success Criteria
   * Chatbot interactions are seamless, with relevant and meaningful responses provided.

### ****8.2.8 Usability****

1. The platform must be user-friendly and accessible on
   * Desktop.
   * Mobile.
   * Tablet.
2. Success Criteria
   * Users navigate the platform easily, with no confusion or difficulty in finding key features.

### ****8.2.9 System Performance****

1. The system must perform efficiently under various conditions
   * **Response Time**: Recommendations, dashboards, and comparisons must load within **2 seconds**.
   * **Concurrent Users**: The system must handle at least **1,000 concurrent users** without performance degradation.
2. Success Criteria
   * Performance metrics meet defined benchmarks during load and stress testing.

### ****8.2.10 Security****

1. User data must be securely stored and transmitted
   * Passwords must be encrypted.
   * Data in transit must be encrypted using **SSL/TLS protocols**.
2. The system must prevent vulnerabilities such as **SQL injection**, **XSS**, and **CSRF attacks**.
3. Success Criteria
   * Security tests are passed with no critical vulnerabilities identified.

### ****8.2.11 Integration****

1. The system must integrate seamlessly between
   * Frontend and backend components.
   * External APIs (e.g., Google OAuth and university data APIs).
2. Success Criteria
   * All integrated systems function as expected without errors or delays.

### ****8.2.12 User Feedback and Testing****

1. Beta testing with students and educators must
   * Identify areas for improvement.
   * Ensure the platform meets user expectations in terms of functionality, performance, and usability.
2. Success Criteria
   * Feedback is overwhelmingly positive, with no major usability issues reported.

### ****8.2.13 Deployment****

1. The platform must be deployed successfully to a live hosting environment
   * Accessible to users on all major browsers (Chrome, Firefox, Safari, Edge).
   * Compatible with devices running Windows, macOS, Android, and iOS.
2. Success Criteria
   * Users access the live system without issues or downtime.

### ****8.2.14 Post-Launch Monitoring****

1. The system must
   * Continuously monitor performance and user feedback.
   * Provide actionable insights for future updates.
2. Success Criteria
   * Maintenance tasks are completed on schedule, with identified issues resolved promptly.

## 8.3 Quality Assurance Plan

### ****8.3.1 Introduction****

**Purpose**: The QA Plan ensures the platform meets functional, performance, security, and usability requirements while aligning with user needs.

**Scope**: Covers the testing, validation, and review processes for all system components, including user registration, recommendations, visualizations, and external API integrations.

**Target Audience**: Students, educators and administrators using the platform.

### ****8.3.2 Quality Objectives****

1. Deliver a secure and reliable platform for students to receive personalized career guidance.
2. Ensure all system functionalities, such as recommendations and visualizations, perform efficiently and accurately.
3. Achieve high user satisfaction with an intuitive and responsive interface.

### ****8.3.3 Scope of QA****

The QA plan includes

1. Functional testing for core features (e.g., recommendations, visualizations, and comparisons).
2. Performance testing under various loads.
3. Security testing to ensure data protection.
4. Usability testing for a user-friendly interface.
5. Integration testing to validate the interaction between components (e.g., frontend-backend).

### ****8.3.4 Roles and Responsibilities****

|  |  |
| --- | --- |
| **Role** | **Responsibilities** |
| **QA Lead** | Oversee QA activities, prepare test plans and ensure adherence to standards. |
| **Test Engineers** | Conduct functional, performance, security and usability testing. |
| **Developers** | Fix bugs and issues identified during testing. |
| **Project Manager** | Ensure QA timelines align with project deadlines. |

### ****8.3.5 Quality Assurance Processes****

1. **Planning Phase**:
   * Define functional and non-functional requirements.
   * Prepare test cases and scenarios.
2. **Design Review**:
   * Conduct peer reviews of UI/UX wireframes, database schema, and algorithms.
3. **Development QA**:
   * Perform static code analysis to ensure code quality.
4. **Testing**:
   * Conduct various types of testing (detailed below).
5. **Validation**:
   * Confirm that the platform meets user needs through UAT.
6. **Post-Launch Monitoring**:
   * Continuously monitor performance and collect user feedback.

### ****8.3.*6* Testing Strategy****

##### **1 Functional Testing**

**Objective**: Validate that all features work as intended.

**Key Areas**:

1. User registration and login.
2. Data input and validation.
3. Recommendation system (fields and universities).
4. Visualizations (trends and comparisons).

**Tool**: cypress.

##### **2 Performance Testing**

**Objective**: Ensure the platform performs efficiently under load.

**Key Metrics**:

1. Response time: < 2 seconds for recommendations and dashboards.
2. Scalability: Handle up to 10,000 concurrent users.

**Tools**: JMeter.

##### **3 Security Testing**

**Objective**: Ensure user data is protected from threats.

**Key Areas**:

1. Test against SQL Injection and DDoS attacks.
2. Role-based access control testing.

**Tools**: Burp Suite.

##### **4 Usability Testing**

**Objective**: Ensure the platform is intuitive and accessible.

**Key Areas**:

1. Navigation flow.
2. Responsiveness across devices.
3. Accessibility compliance.

**Tools**: Accessibility Insights.

##### **5 Integration Testing**

**Objective**: Validate interactions between system components.

**Key Areas**:

1. Frontend-backend communication via APIs.
2. External API integrations (e.g., Google OAuth, university data APIs).

**Tools**: Postman.

##### **6 Regression Testing**

**Objective**: Ensure new updates don’t break existing features.

**Key Areas**:

1. Retest core functionalities after updates.
2. Maintain automated regression scripts.

**Tools**: Selenium.

##### **7 User Acceptance Testing (UAT)**

**Objective**: Validate the platform meets user expectations.

**Process**:

1. Conduct beta testing with students and educators.
2. Collect feedback on usability, accuracy, and performance.

##### **8 Post-Deployment Testing**

**Objective**: Ensure the live system performs as expected.

**Key Areas**:

1. Monitor uptime (target: 99.9%).
2. Validate response times and error rates.

**Tools**: Google Analytics.

### ****8.3.8 Defect Management****

1. Track and manage defects using Jira or Trello.
2. Classify defects by severity (Critical, High, Medium, Low).
3. Prioritize and resolve issues based on impact.

### ****8.3.9 Metrics and Reporting****

1. **Defect Density**: Target < 1 defect per 1,000 lines of code.
2. **Test Case Pass Rate**: Achieve > 95% pass rate.
3. **Performance Metrics**: Response times < 2 seconds, uptime 99.9%.
4. **User Satisfaction**: Target average score of 4.5/5 in feedback.

### ****8.3.10 Tools and Resources****

1. **Testing Tools**: Selenium, Postman, JMeter, OWASP ZAP.
2. **Version Control**: Git/GitHub.
3. **Bug Tracking**: Jira, Trello.
4. **Environments**:
   * Staging: For internal testing.
   * Production: For live user testing.

### ****8.3.11 Risk Management****

|  |  |
| --- | --- |
| **Risk** | **Mitigation** |
| Insufficient testing time | Allocate buffer time in the project schedule. |
| High defect rate during development | Perform continuous testing alongside development. |
| Security vulnerabilities | Conduct periodic security audits. |
| Compatibility issues | Test across devices, browsers, and operating systems. |

### ****8.3.12 Continuous Improvement****

1. Collect feedback from users and stakeholders post-launch.
2. Regularly update test cases to align with system changes.
3. Conduct retrospectives to improve QA processes.

### ****8.3.13 Approval and Sign-off****

1. Obtain approval from stakeholders (e.g., project manager, QA lead) before deploying the system.
2. Ensure all defects marked as "Critical" or "High" are resolved.

## 8.4 Planned Test Suites and Test Cases

### ****8.4.1 Test Suite: User Registration and Login****

#### **Test Case 1.1: User Registration**

* **Description**: Validate that users can successfully create an account.
* **Steps**:
  1. Navigate to the registration page.
  2. Enter valid email, password, and confirm password.
  3. Click "Register."
* **Expected Result**: Account is created successfully, and a confirmation email is sent.
* **Negative Test Case**:
  1. Enter an invalid email or mismatched passwords. Ensure an error message is displayed.

#### **Test Case 1.2: User Login**

* **Description**: Validate that registered users can log in.
* **Steps**:
  1. Navigate to the login page.
  2. Enter valid email and password.
  3. Click "Login."
* **Expected Result**: User is redirected to the dashboard.
* **Negative Test Case**:
  1. Enter incorrect email/password. Ensure an error message is displayed.

#### **Test Case 1.3: Google OAuth Login**

* **Description**: Validate login via Google OAuth.
* **Steps**:
  1. Click "Login with Google."
  2. Choose a valid Google account.
* **Expected Result**: User is redirected to the dashboard.

### ****8.4.2 Test Suite: Data Input and Validation****

#### **Test Case 2.1: Input Valid Data**

* **Description**: Validate that users can input valid academic details.
* **Steps**:
  1. Enter Matric and FSC marks (numeric values).
  2. Select a study stream (e.g., Pre-Medical, Pre-Engineering).
  3. Click "Submit."
* **Expected Result**: Data is saved successfully.

#### **Test Case 2.2: Input Invalid Data**

* **Description**: Test how the system handles invalid data.
* **Steps**:
  1. Enter non-numeric or negative values for marks.
  2. Leave required fields empty.
* **Expected Result**: Error messages are displayed.

#### **Test Case 2.3: Enter Expected Marks**

* **Description**: Validate that users can enter expected marks if results are pending.
* **Steps**:
  1. Select "Enter Expected Marks."
  2. Input expected scores for Matric, FSC, and tests.
* **Expected Result**: Expected marks are saved successfully.

### ****8.4.3 Test Suite: Recommendation System****

#### **Test Case 3.1: Field Recommendations**

* **Description**: Validate that fields are recommended based on user data.
* **Steps**:
  1. Enter marks and study stream.
  2. Click "Get Recommendations."
* **Expected Result**: List of suitable fields is displayed.

#### **Test Case 3.2: University Recommendations**

* **Description**: Validate that universities are recommended based on calculated aggregate.
* **Steps**:
  1. Select a field of study.
  2. View recommended universities.
* **Expected Result**: Relevant universities are displayed with admission criteria and deadlines.

#### **Test Case 3.3: Interest-Based Recommendations**

* **Description**: Validate recommendations based on user interest.
* **Steps**:
  1. Interact with the chatbot.
  2. Provide details about interests.
  3. View recommended fields and universities.
* **Expected Result**: Recommendations align with user interests.

### ****8.4.4 Test Suite: Dashboard and Trend Visualization****

#### **Test Case 4.1: View Trends**

* **Description**: Validate that trends for fields of study are displayed on the dashboard.
* **Steps**:
  1. Navigate to the dashboard.
  2. Select "View Trends."
* **Expected Result**: Graphs for growing and declining fields are displayed.

#### **Test Case 4.2: Compare Fields**

* **Description**: Validate the comparison of multiple fields.
* **Steps**:
  1. Select two or more fields for comparison.
  2. Click "Compare."
* **Expected Result**: Comparison data (growth rate, demand index) is displayed.

### ****8.4.5 Test Suite: Integration Testing****

#### **Test Case 5.1: Frontend-Backend Integration**

* **Description**: Validate the integration between frontend forms and backend APIs.
* **Steps**:
  1. Fill out data input forms.
  2. Check API calls using a tool like Postman.
* **Expected Result**: Data is saved and processed correctly by the backend.

#### **Test Case 5.2: External API Integration**

* **Description**: Validate Google OAuth and university data API integrations.
* **Steps**:
  1. Log in using Google OAuth.
  2. Fetch real-time university data.
* **Expected Result**: Data is retrieved and displayed without errors.

### ****8.4.6 Test Suite: Performance Testing****

#### **Test Case 6.1: Load Testing**

* **Description**: Validate system performance under load.
* **Steps**:
  1. Simulate 1,000 concurrent users.
  2. Monitor response times.
* **Expected Result**: Response time remains under 2 seconds.

#### **Test Case 6.2: Stress Testing**

* **Description**: Test system behavior under peak traffic.
* **Steps**:
  1. Simulate 5,000 concurrent users.
  2. Monitor system stability.
* **Expected Result**: System handles high load without crashing.

### ****8.4.7 Test Suite: Security Testing****

#### **Test Case 7.1: SQL Injection**

* **Description**: Test for SQL injection vulnerabilities.
* **Steps**:
  1. Enter malicious SQL code in input fields.
* **Expected Result**: System rejects malicious inputs.

#### **Test Case 7.2: Data Encryption**

* **Description**: Validate encryption for sensitive data.
* **Steps**:
  1. Check stored passwords and user data.
  2. Verify SSL/TLS encryption for data in transit.
* **Expected Result**: Data is encrypted using AES-256 and SSL/TLS.

### ****8.4.8 Test Suite: Usability Testing****

#### **Test Case 8.1: Mobile Responsiveness**

* **Description**: Validate that the platform works on mobile devices.
* **Steps**:
  1. Access the platform on a mobile browser.
  2. Navigate through forms and dashboards.
* **Expected Result**: Platform is fully responsive.

#### **Test Case 8.2: Navigation Flow**

* **Description**: Validate ease of navigation.
* **Steps**:
  1. Navigate through different pages (e.g., registration, recommendations, dashboard).
* **Expected Result**: All pages are easily accessible.

### ****8.4.9 Test Suite: Post-Deployment Testing****

#### **Test Case 9.1: Monitor Uptime**

* **Description**: Validate uptime and system availability.
* **Steps**:
  1. Monitor the live system for 24 hours.
* **Expected Result**: Uptime is 99.9%.

#### **Test Case 9.2: Monitor Performance**

* **Description**: Test live system performance under real-world conditions.
* **Steps**:
  1. Monitor response times and error rates.
* **Expected Result**: Response times remain under 2 seconds.