Software Requirements Specification

**for**

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

**Version 1.1**

**Prepared by**

**FYP-SE-31**

|  |  |  |
| --- | --- | --- |
| **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*** |
| **Teaching Assistant:** | ***Miss Fatima Rani*** |
| **Date:** | ***19-12-2024*** |

**Contents**

Contents

[1 Introduction 1](#_Toc187829596)

[1.1 Document Purpose 1](#_Toc187829597)

[1.2 Product Scope 2](#_Toc187829598)

[1.3 Intended Audience and Document Overview 2](#_Toc187829599)

[1.4 Definitions, Acronyms and Abbreviations 3](#_Toc187829600)

[1.5 Document Conventions 4](#_Toc187829601)

[1.6 References and Acknowledgements 5](#_Toc187829602)

[2 Overall Description 6](#_Toc187829603)

[2.1 Project Overview 6](#_Toc187829604)

[2.3 Design and Implementation Constraints 8](#_Toc187829605)

[2.3.1 Hardware Limitations 8](#_Toc187829606)

[2.3.2 Technologies and Tools 8](#_Toc187829607)

[2.3.3 Interfaces 8](#_Toc187829608)

[2.3.4 Programming Standards 8](#_Toc187829609)

[2.3.5 Language and Protocols 8](#_Toc187829610)

[2.3.6 Security Considerations 9](#_Toc187829611)

[2.3.7 Design Conventions 9](#_Toc187829612)

[2.3.8 Parallel Operations 9](#_Toc187829613)

[2.4 Assumptions and Dependencies 9](#_Toc187829614)

[3 Specific Requirements 10](#_Toc187829615)

[3.1 External Interface Requirements 10](#_Toc187829616)

[3.1.1 User Interfaces 10](#_Toc187829617)

[3.1.2 Hardware Interfaces 14](#_Toc187829618)

[3.1.3 Software Interfaces 14](#_Toc187829619)

[3.2 Functional Requirements 15](#_Toc187829620)

[3.2.1 F1: User Registration and Login 15](#_Toc187829621)

[3.2.2 F2: Data Input and Validation 15](#_Toc187829622)

[3.2.3 F3: Field Recommendation 15](#_Toc187829623)

[3.2.4 F4: Interest Finding Chatbot 15](#_Toc187829624)

[3.2.5 F5: Aggregate Calculation 16](#_Toc187829625)

[3.2.6 F6: University Recommendation 16](#_Toc187829626)

[3.2.7 F7: Trend Visualization 16](#_Toc187829627)

[3.2.8 F8: Dashboard for Insights 16](#_Toc187829628)

[3.2.9 F9: Search and Filter Options 16](#_Toc187829629)

[3.2.10 F10: User-Friendly Interface 17](#_Toc187829630)

[3.3 Use Case Model 17](#_Toc187829631)

[3.3.1 Use Case # 1.1 17](#_Toc187829632)

[3.3.2 Use Case # 1.2 18](#_Toc187829633)

[3.3.3 Use Case # 1.3 19](#_Toc187829634)

[3.3.4 Use Case # 2.1 20](#_Toc187829635)

[3.3.5 Use Case # 2.2 21](#_Toc187829636)

[3.3.6 Use Case # 2.3 22](#_Toc187829637)

[3.3.7 Use Case # 2.4 23](#_Toc187829638)

[3.3.8 Use Case # 3.1 24](#_Toc187829639)

[3.3.9 Use Case # 3.2 25](#_Toc187829640)

[3.3.10 Use Case # 4 26](#_Toc187829641)

[3.3.10 Use Case # 5.1 27](#_Toc187829642)

[3.3.11 Use Case # 5.2 28](#_Toc187829643)

[3.3.12 Use Case # 6.1 29](#_Toc187829644)

[3.3.13 Use Case # 6.2 30](#_Toc187829645)

[3.3.14 Use Case # 7.1 31](#_Toc187829646)

[3.3.15 Use Case # 7.2 32](#_Toc187829647)

[3.3.16 Use Case # 8.1 33](#_Toc187829648)

[3.3.17 Use Case # 8.2 34](#_Toc187829649)

[3.3.18 Use Case # 9.1 35](#_Toc187829650)

[4 Other Non-functional Requirements 36](#_Toc187829651)

[4.1 Performance Requirements 36](#_Toc187829652)

[4.1.1 Response Time 36](#_Toc187829653)

[4.1.2 Concurrent Users 36](#_Toc187829654)

[4.1.3 Data Processing 36](#_Toc187829655)

[4.1.4 Dashboard Loading 36](#_Toc187829656)

[4.1.5 Database Query Performance 36](#_Toc187829657)

[4.1.6 Login and Signup 36](#_Toc187829658)

[4.1.7 Aggregate Calculation 37](#_Toc187829659)

[4.1.8 Trend Visualization 37](#_Toc187829660)

[4.1.9 University Search and Filter 37](#_Toc187829661)

[4.1.10 Peak Load 37](#_Toc187829662)

[4.1.11 Scalability 37](#_Toc187829663)

[4.1.12 Data Transfer 37](#_Toc187829664)

[4.2 Safety and Security Requirements 37](#_Toc187829665)

[4.2.1 Data Integrity 38](#_Toc187829666)

[4.2.2 Safeguards Against Unauthorized Actions 38](#_Toc187829667)

[4.2.3 System Downtime Protection 38](#_Toc187829668)

[4.2.4 Backup and Recovery 38](#_Toc187829669)

[4.2.5 User Authentication 38](#_Toc187829670)

[4.2.6 Access Control 38](#_Toc187829671)

[4.2.7 Protection Against Common Attacks 39](#_Toc187829672)

[4.2.8 User Privacy 39](#_Toc187829673)

[4.2.9 Audit Logs 39](#_Toc187829674)

[4.3 Software Quality Attributes 39](#_Toc187829675)

[4.3.1 Reliability 39](#_Toc187829676)

[4.3.2 Usability 40](#_Toc187829677)

[4.3.3 Maintainability 40](#_Toc187829678)

[4.3.4 Adaptability 40](#_Toc187829679)

[5 Design Requirements 41](#_Toc187829680)

[5.1 High Level Design 41](#_Toc187829681)

[5.1.1 Abstract Design 41](#_Toc187829682)

[5.1.2 Design oblige Architecture Patterns 42](#_Toc187829683)

[5.2 Structural Design 43](#_Toc187829684)

[5.2.1 Class Diagram 43](#_Toc187829685)

[5.2.2 Component Diagram 44](#_Toc187829686)

[5.2.3 Data Flow Diagram 44](#_Toc187829687)

[5.2.4 Graphical User Interface 46](#_Toc187829688)

[5.3 Behavioral Diagram 49](#_Toc187829689)

[5.3.1 Use Case Diagram 49](#_Toc187829690)

[5.3.2 Sequence Diagram 51](#_Toc187829691)

[5.3.3 State Machine Diagram 53](#_Toc187829692)

[6 Data Design and Relationships 55](#_Toc187829693)

[6.1 Database Schema 55](#_Toc187829694)

[6.1.1 User Table 55](#_Toc187829695)

[6.1.2 Academic Details Table 56](#_Toc187829696)

[6.1.3 Recommendations Table 56](#_Toc187829697)

[6.1.4 Trends Table 56](#_Toc187829698)

[6.2 Entity Relationship Diagram 57](#_Toc187829699)

[6.3 Data Dictionary Diagram 58](#_Toc187829700)

[7 Software Planning and Timeline 59](#_Toc187829701)

[7.1 Work Breakdown Structure 59](#_Toc187829702)

[7.2 Milestones and Deliverables 63](#_Toc187829703)

[7.2.1 Milestone 1: Planning and Research 63](#_Toc187829704)

[7.2.2 Milestone 2: Backend Development 64](#_Toc187829705)

[7.2.3 Milestone 3: Frontend Development 65](#_Toc187829706)

[7.2.4 Milestone 4: Data Visualization and Insights 66](#_Toc187829707)

[7.2.5 Milestone 5: Testing and Validation 66](#_Toc187829708)

[7.2.6 Milestone 6: Deployment and Launch 67](#_Toc187829709)

[7.2.7 Milestone 7: Post-Launch Activities 67](#_Toc187829710)

[7.3 Detailed Baseline Plan 68](#_Toc187829711)

[8 Quality Assurance Plan 70](#_Toc187829712)

[8.1 Testing Requirements 70](#_Toc187829713)

[8.1.1 Functional Testing 70](#_Toc187829714)

[8.1.2 Performance Testing 70](#_Toc187829715)

[8.1.3 Security Testing 71](#_Toc187829716)

[8.1.4 Usability Testing 71](#_Toc187829717)

[8.1.5 Integration Testing 71](#_Toc187829718)

[8.1.6 System Testing 72](#_Toc187829719)

[8.1.7 Regression Testing 72](#_Toc187829720)

[8.1.8 User Acceptance Testing (UAT) 72](#_Toc187829721)

[8.2 Acceptance Criteria 72](#_Toc187829722)

[8.2.1 User Registration and Login 72](#_Toc187829723)

[8.2.2 Data Input and Validation 73](#_Toc187829724)

[8.2.3 Field Recommendation 73](#_Toc187829725)

[8.2.4 University Recommendation 73](#_Toc187829726)

[8.2.5 Trend Analysis and Visualization 74](#_Toc187829727)

[8.2.6 Comparison Tools 74](#_Toc187829728)

[8.2.7 Chatbot Interaction 75](#_Toc187829729)

[8.2.8 Usability 75](#_Toc187829730)

[8.2.9 System Performance 75](#_Toc187829731)

[8.2.10 Security 76](#_Toc187829732)

[8.2.11 Integration 76](#_Toc187829733)

[8.2.12 User Feedback and Testing 76](#_Toc187829734)

[8.2.13 Deployment 77](#_Toc187829735)

[8.2.14 Post-Launch Monitoring 77](#_Toc187829736)

[8.3 Quality Assurance Plan 77](#_Toc187829737)

[8.3.1 Introduction 77](#_Toc187829738)

[8.3.2 Quality Objectives 77](#_Toc187829739)

[8.3.3 Scope of QA 78](#_Toc187829740)

[8.3.4 Roles and Responsibilities 78](#_Toc187829741)

[8.3.5 Quality Assurance Processes 78](#_Toc187829742)

[8.3.*6* Testing Strategy 79](#_Toc187829743)

[8.3.7 Defect Management 81](#_Toc187829744)

[8.3.8 Metrics and Reporting 82](#_Toc187829745)

[8.3.9 Tools and Resources 82](#_Toc187829746)

[8.3.10 Risk Management 82](#_Toc187829747)

[8.3.11 Continuous Improvement 82](#_Toc187829748)

[8.3.12 Approval and Sign-off 83](#_Toc187829749)

[8.4 Planned Test Suites and Test Cases 83](#_Toc187829750)

[8.4.1 Test Suite: User Registration and Login 83](#_Toc187829751)

[8.4.2 Test Suite: Data Input and Validation 84](#_Toc187829752)

[8.4.3 Test Suite: Recommendation System 84](#_Toc187829753)

[8.4.4 Test Suite: Dashboard and Trend Visualization 85](#_Toc187829754)

[8.4.5 Test Suite: Integration Testing 85](#_Toc187829755)

[8.4.6 Test Suite: Performance Testing 85](#_Toc187829756)

[8.4.7 Test Suite: Security Testing 86](#_Toc187829757)

[8.4.8 Test Suite: System Workflow Testing 86](#_Toc187829758)

[8.4.9 Test Suite: Post-Deployment Testing 86](#_Toc187829759)

**List of Tables**

[Table 1: Document versions x](#_Toc187850831)

[Table 2: Register user use case 17](#_Toc187850832)

[Table 3: Login case 18](#_Toc187850833)

[Table 4: Google auth use case 19](#_Toc187850834)

[Table 5: input academic data use case 20](#_Toc187850835)

[Table 6: Expected marks use case 21](#_Toc187850836)

[Table 7: Select stream use case 22](#_Toc187850837)

[Table 8: Validate input use case 23](#_Toc187850838)

[Table 9: Recommend field use case 24](#_Toc187850839)

[Table 10: Recommend interest based use case 25](#_Toc187850840)

[Table 11: Chatbot use case 26](#_Toc187850841)

[Table 12: Calculation use case 27](#_Toc187850842)

[Table 13: Aggregate use case 28](#_Toc187850843)

[Table 14: Recommendation aggregate based use case 29](#_Toc187850844)

[Table 15: University details use case 30](#_Toc187850845)

[Table 16: Display trends use case 31](#_Toc187850846)

[Table 17: Comparison use case 32](#_Toc187850847)

[Table 18: Personalized recommendation use case 33](#_Toc187850848)

[Table 19: Visualization use case 34](#_Toc187850849)

[Table 20: Filtration use case 35](#_Toc187850850)

[Table 21: User table 55](#_Toc187850851)

[Table 22: Academic details table 56](#_Toc187850852)

[Table 23: Recommendation table 56](#_Toc187850853)

[Table 24: Trends table 56](#_Toc187850854)

[Table 25: University table 57](#_Toc187850855)

[Table 26: Roles and Responsibilities 78](#_Toc187850856)

[Table 27: Risk and Mitigation 82](#_Toc187850857)

[Table 28: Test Suit 1 User Registration and Login 83](#_Toc187850858)

[Table 29: Test Suite Data Input and Validation 84](#_Toc187850859)

[Table 30: Test Suite Recommendation System 84](#_Toc187850860)

[Table 31: Test Suite Dashboard and Trend Visualization 85](#_Toc187850861)

[Table 32: Test Suite for Integration Testing 85](#_Toc187850862)

[Table 33: Test Suit Performance Testing 85](#_Toc187850863)

[Table 34: Test Suit Security Testing 86](#_Toc187850864)

[Table 35:Test Suit System Workflow 86](#_Toc187850865)

[Table 36: Test suit Post Deployment Testing 86](#_Toc187850866)

[Table 37: Appendix A - Data Dictionary 87](#_Toc187850867)

[Table 38: Appendix B - Group Log 89](#_Toc187850868)

**List of Figures**

[Figure 1: System workflow 7](#_Toc187829242)

[Figure 2: Landing page 10](#_Toc187829243)

[Figure 3: Home page 11](#_Toc187829244)

[Figure 4: Dashboard 11](#_Toc187829245)

[Figure 5: Analytics page 12](#_Toc187829246)

[Figure 6: About us page 13](#_Toc187829247)

[Figure 7: Contact us page 13](#_Toc187829248)

[Figure 8: Abstract design 41](#_Toc187829249)

[Figure 9: Layered Architecture Pattern 42](#_Toc187829250)

[Figure 10: Class Diagram 43](#_Toc187829251)

[Figure 11: Component Diagram 44](#_Toc187829252)

[Figure 12: Context level DFD 44](#_Toc187829253)

[Figure 13: Level 1 DFD 45](#_Toc187829254)

[Figure 14: Level 2 DFD 45](#_Toc187829255)

[Figure 15: GUI Landing page 46](#_Toc187829256)

[Figure 16: GUI Home page 47](#_Toc187829257)

[Figure 17: GUI Dashboard 47](#_Toc187829258)

[Figure 18: GUI Trends Analytics 48](#_Toc187829259)

[Figure 19: GUI About us page 48](#_Toc187829260)

[Figure 20: GUI Connect us page 49](#_Toc187829261)

[Figure 21: Use Case Diagram 50](#_Toc187829262)

[Figure 22: User registration 51](#_Toc187829263)

[Figure 23: User login and forgot password 52](#_Toc187829264)

[Figure 24: Input Data and University Recommendation 52](#_Toc187829265)

[Figure 25: Trend analytics 53](#_Toc187829266)

[Figure 26: Comparison analytics 53](#_Toc187829267)

[Figure 27: State Machine Diagram 54](#_Toc187829268)

[Figure 28: Entity Relationship Diagram 57](#_Toc187829269)

[Figure 29: Data Dictionary Diagram 58](#_Toc187829270)

[Figure 30: Work Breakdown Structure 59](#_Toc187829271)

[Figure 31: Planning and Research WBS 60](#_Toc187829272)

[Figure 32: Frontend Development WBS 61](#_Toc187829273)

[Figure 33: Backend Development WBS 61](#_Toc187829274)

[Figure 34: Data Visualization and Insights WBS 62](#_Toc187829275)

[Figure 35: Testing and Validation WBS 62](#_Toc187829276)

[Figure 36: Deployment and Launch, Post Launch Activities WBS 63](#_Toc187829277)

[Figure 37: Milestone 1Planning and Research 64](#_Toc187829278)

[Figure 38: Milestone 2 Backend Development 65](#_Toc187829279)

[Figure 39: Milestone 3 Frontend Development 65](#_Toc187829280)

[Figure 40: Data Visualization and Insights 66](#_Toc187829281)

[Figure 41: Testing and Validation 67](#_Toc187829282)

[Figure 42: Deployment and Launch 67](#_Toc187829283)

[Figure 43: Milestone 7 Post Launch Activities 68](#_Toc187829284)

[Figure 44: Baseline Plan part 1 68](#_Toc187829285)

[Figure 45: Baseline Plan part 2 69](#_Toc187829286)

[Figure 46: Baseline Plan part 3 69](#_Toc187829287)

**Revisions**

Table 1 shows the revision of document for version controlling.

Table 1: Document versions

|  |  |  |  |
| --- | --- | --- | --- |
| **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 |
| Version 1 | Asad Shah | Completed 1 version of SRS | 10/1/25 |
| Version 1.1 | Asad Shah | Add some formatting | 13/1/25 |

# 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** detail functional requirements and use cases for user interactions.
4. **Non-functional requirements** define 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.

## 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 shows the workflow of our system.

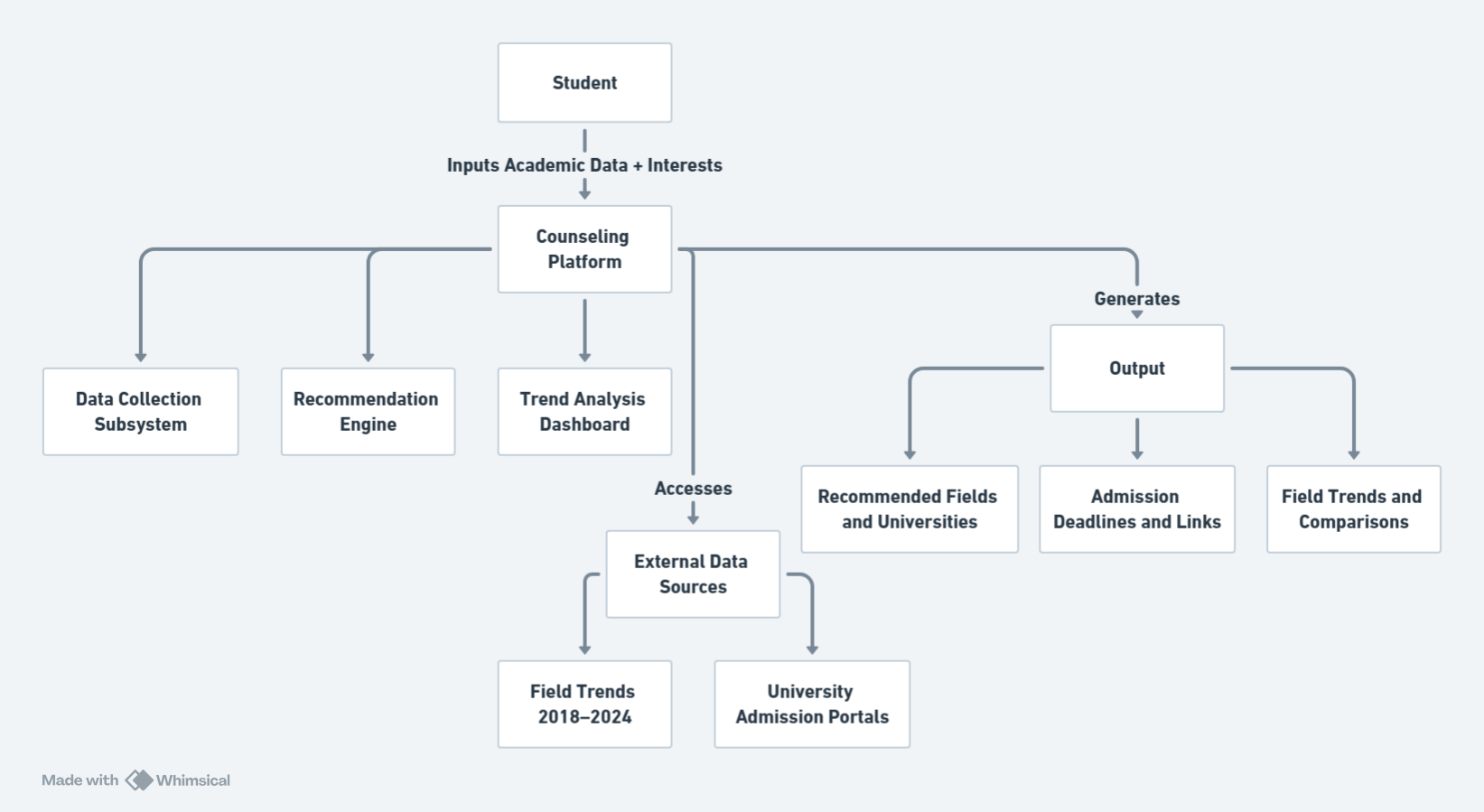


Figure 1: System workflow

**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.

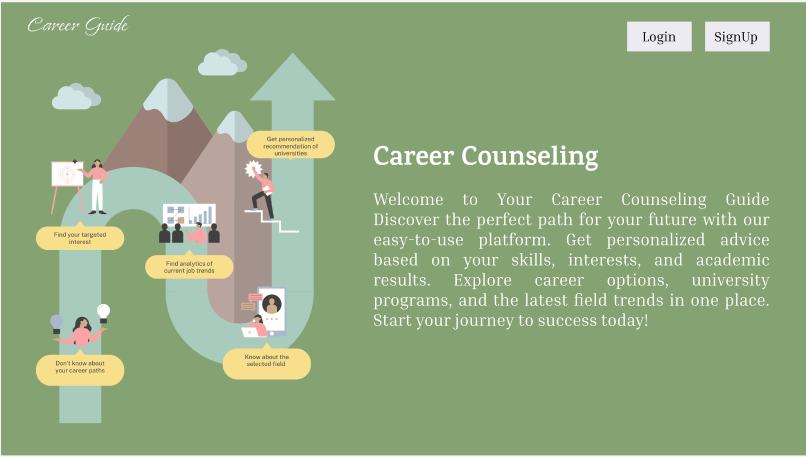


Figure 2: Landing page

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.

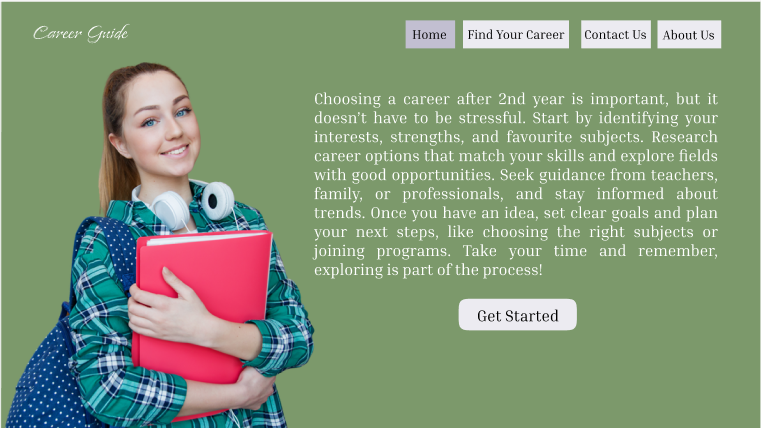


Figure 3: Home page

After clicking the get started button, we are navigated to find our career section shown in Figure 4.

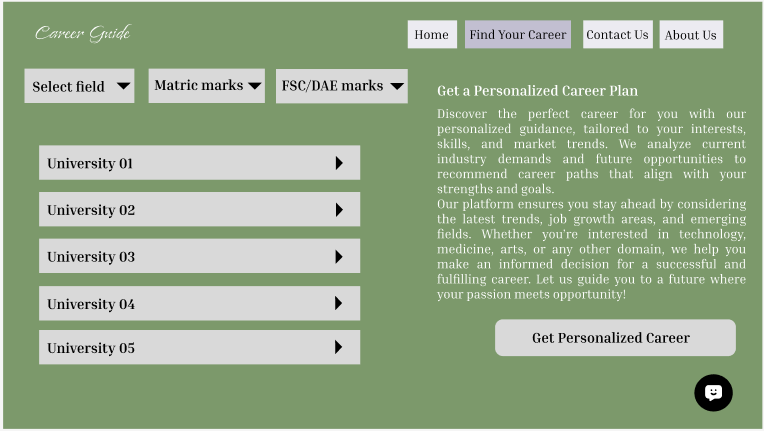


Figure 4: Dashboard

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.

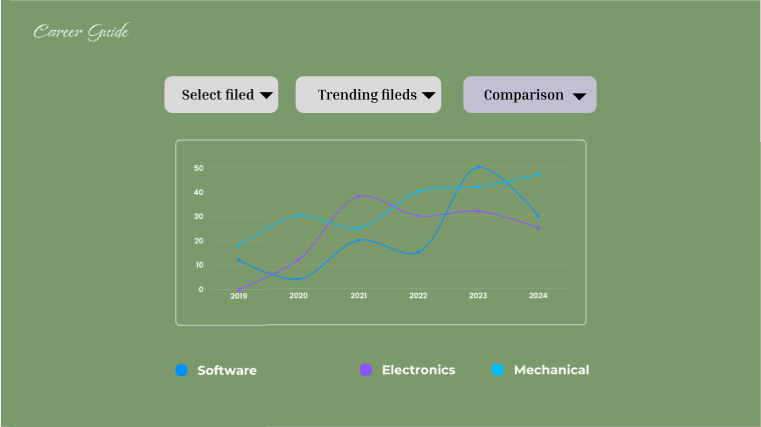


Figure 5: Analytics page

We also have our about page, which provides a detailed explanation of the purpose and objectives of our website. This page aims to inform users about the vision and mission behind the platform, highlighting how it addresses the challenges faced by students after completing their 12th grade. It emphasizes the role of the website in helping students make informed decisions about their academic and career paths. The about page is designed to create a sense of trust and transparency, ensuring users understand the value and intent of the services offered. The layout and content of this page are depicted in Figure 6.

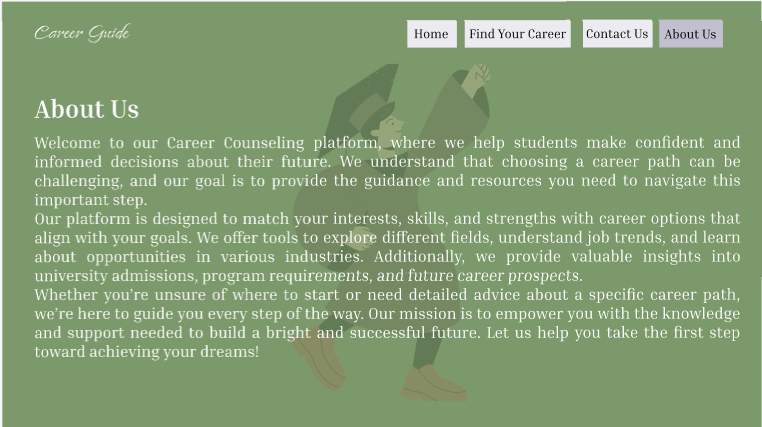
****

Figure 6: About us page

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

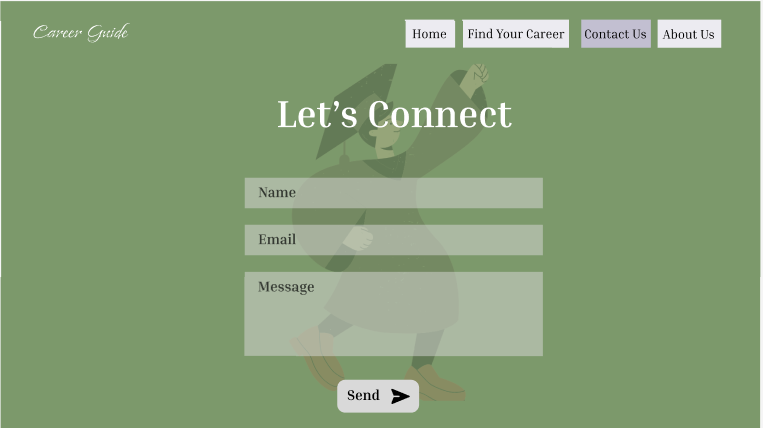


Figure 7: Contact us page

### 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.
2. The chatbot further refines recommendations by analyzing user responses, identifying preferences and suggesting fields of study or career paths aligned with their expressed interests.

### 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

1. The system shall recommend universities based on the user's calculated aggregate.
2. The system shall provide details about each university, including:
3. Admission criteria.
4. Application deadlines.
5. 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

Table 2 illustrates the use case detailing the process of how a new user registers on the platform. This use case outlines the step-by-step interaction between the user and the system, ensuring a seamless and secure account creation process. The registration process begins with the user navigating to the registration page and providing essential details such as their name, email, and password. The system validates these inputs to ensure they meet the required format.

Table 2: Register user use case

|  |  |  |  |
| --- | --- | --- | --- |
| **UC1.1** | **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

Table 3 outlines the use case for logging into the platform using an email and password. This use case explains the process users follow to securely access their accounts. It starts with the user navigating to the login page and entering their registered email address and password. The system then verifies these credentials against the stored data in the database to ensure they match an existing account.

If the credentials are valid, the system grants the user access and redirects them to their personalized dashboard. However, if the credentials are incorrect, the system displays an appropriate error message, prompting the user to re-enter their details or recover their password. This use case ensures a secure and user-friendly authentication process, protecting user accounts while maintaining accessibility.

The system incorporates security measures, such as encryption of passwords, to safeguard user data during the authentication process. It also provides an option for users to reset their password if they forget their credentials, ensuring uninterrupted access to the platform.

Table 3: Login case

|  |  |  |  |
| --- | --- | --- | --- |
| **UC1.2** | **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

Table 4 explains the use case for logging into the platform using Google OAuth. This method allows users to authenticate quickly and securely by leveraging their existing Google accounts. The process begins with the user clicking the "Login with Google" button on the login page, which redirects them to Google's authentication service. The user then selects their Google account and grants permission for the platform to access basic profile information, such as their name and email address.

Once the Google authentication is successful, the platform retrieves the user's information and checks whether they already have an account. If the account exists, the user is seamlessly logged in and redirected to their dashboard. If no account exists, the platform automatically registers the user and logs them in. This method provides a convenient, fast, and secure alternative to traditional login, reducing the need for users to remember additional passwords.

Table 4: Google auth use case

|  |  |  |  |
| --- | --- | --- | --- |
| **UC1.3** | **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

Table 5 outlines the use case for inputting academic marks into the system. This process allows users to provide their academic details, which are essential for generating personalized recommendations. The use case begins with the user navigating to the academic marks input form. Users are required to enter their Matric marks, FSC marks, and, if applicable, their NTS or NET test scores.

The system validates the entered data to ensure that it adheres to the expected format (e.g., numeric values and within valid ranges). If the user’s results are pending, the system also allows them to input expected marks. Once the data is submitted, it is stored in the database and linked to the user's account. This data is then used by the system to calculate the aggregate and suggest suitable fields of study and universities. The academic marks input feature ensures data accuracy and enables the system to provide relevant recommendations.

The system provides error messages for invalid or incomplete data entries, ensuring the user corrects any mistakes before submission. This functionality ensures that the recommendations generated are accurate and tailored to the user's academic performance, enhancing the overall user experience.

Table 5: input academic data use case

|  |  |  |  |
| --- | --- | --- | --- |
| **UC2.1** | **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

Table 6 describes the use case for inputting expected marks when a user's actual results are pending. This feature allows users to provide their anticipated scores for Matric, FSC, or NTS/NET tests, enabling the system to generate preliminary recommendations. The process begins with the user selecting the "Enter Expected Marks" option and filling out the input fields with their projected scores.

The system validates the entered data to ensure it is in the correct format (e.g., numeric values). After successful validation, the expected marks are stored in the database and linked to the user's account. These scores are used temporarily to calculate aggregates and suggest potential fields of study and eligible universities. Once actual results are available, users can update their marks for more accurate recommendations. This feature ensures that users can engage with the system even before receiving their final scores, providing a proactive approach to career counseling.

The system also provides a clear interface to differentiate between actual and expected marks, ensuring users can easily update their data when results are finalized. This feature empowers students to start planning their academic and career paths early, minimizing delays and maximizing their opportunities for informed decision-making.

Table 6: Expected marks use case

|  |  |  |  |
| --- | --- | --- | --- |
| **UC2.2** | **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

Table 7 outlines the use case for selecting a study stream, which is a crucial step in tailoring the system's recommendations. Users are presented with options such as Pre-Medical, Pre-Engineering, Computer Science, or any other relevant streams. The process begins with the user navigating to the study stream selection page and choosing their current or intended academic stream.

Once a stream is selected, the system validates the choice and stores it in the database, linking it to the user's profile. This input plays a key role in determining suitable fields of study and career paths, as the system uses the selected stream to narrow down recommendations and relevant trends. If users are unsure about their stream, the system can guide them through suggestions based on their academic marks or interests. This feature ensures personalized guidance, enabling users to explore fields aligned with their educational background.

The system allows users to modify their selected stream later if they change their academic direction, ensuring flexibility. This step enhances the accuracy of recommendations by aligning them with the user's academic preferences and qualifications, making the guidance more relevant and effective.

Table 7: Select stream use case

|  |  |  |  |
| --- | --- | --- | --- |
| **UC2.3** | **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

Table 8 describes the use case for validating input data to ensure the accuracy and consistency of user-provided information. The system begins by checking the entered data, such as academic marks, expected scores, and selected study streams, against predefined validation rules. These rules include ensuring numeric values for marks, proper formatting for email addresses, and the presence of all required fields.

If the data meets the validation criteria, it is saved in the database and linked to the user's profile. In cases where invalid or incomplete data is detected, the system displays appropriate error messages, prompting the user to correct the information before proceeding. This validation process ensures that only accurate and meaningful data is used for calculations, recommendations, and trend visualizations, enhancing the reliability of the system's outputs. Additionally, validation safeguards against errors that could compromise the user experience or system integrity.

The system also incorporates real-time validation to provide immediate feedback, reducing the likelihood of errors during the data entry process. It ensures all mandatory fields are completed before submission and highlights any discrepancies for user correction.

Table 8: Validate input use case

|  |  |  |  |
| --- | --- | --- | --- |
| **UC2.4** | **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

Table 9 outlines the use case for recommending fields of study based on the user's marks and selected stream. The process starts with the system analyzing the user's academic data, such as Matric, FSC, or expected marks, in conjunction with their chosen study stream. The system applies predefined algorithms to calculate eligibility and suitability for various fields and generates a list of recommended fields tailored to the user's academic performance.

The recommendations are displayed in an easily understandable format, along with brief descriptions of each field to help users make informed decisions. Users can explore these suggestions and select fields of interest for further analysis, such as university options or career prospects. The system ensures that the recommendations are updated dynamically if the user modifies their marks or study stream.

Additionally, the system prioritizes growing or high-demand fields based on trends data, ensuring users are guided toward relevant and promising options.

Table 9: Recommend field use case

|  |  |  |  |
| --- | --- | --- | --- |
| **UC3.1** | **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

Table 10 describes the use case for recommending fields of study based on the user's stated interests. The process begins with the user interacting with a chatbot or directly selecting interests such as technology, medicine, arts, or any other areas of focus. The system collects these inputs and maps them to relevant fields of study using a predefined database of interest-to-field mappings.

The system then generates a list of recommended fields that align with the user's interests, accompanied by brief descriptions to help the user understand each option. If a user selects multiple interests, the system provides a ranked list of fields based on demand, relevance, and potential growth. This ensures that the recommendations are both meaningful and aligned with the user's preferences.

Additionally, the system dynamically adjusts recommendations if the user updates their stated interests. It also incorporates global and local trends to refine the suggestions, ensuring relevance in both current and future job markets. This functionality encourages users to explore academic paths they are passionate about while staying informed about potential opportunities in those fields.

Table 10: Recommend interest based use case

|  |  |  |  |
| --- | --- | --- | --- |
| **UC3.2** | **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

Table 11 outlines the use case for interacting with the interest-finding chatbot to identify the user's preferences and align them with suitable fields of study. The process begins when the user initiates a chat session with the chatbot. The chatbot engages the user with questions about their hobbies, favorite subjects, extracurricular activities, and future aspirations.

Based on the user’s responses, the chatbot analyzes their interests using Natural Language Processing (NLP) algorithms and maps them to relevant academic fields. The system then presents a list of potential fields tailored to the user's expressed interests, providing brief descriptions and possible career paths for each. The chatbot adapts its questions dynamically based on user feedback to refine the interest analysis.

Table 11: Chatbot use case

|  |  |  |  |
| --- | --- | --- | --- |
| **UC3.4** | **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

Table 12 describes the use case for calculating the aggregate based on the user's academic marks and predefined formulas. The process begins with the user submitting their Matric, FSC, and, if applicable, NTS or NET marks. The system retrieves the marks from the database and applies the aggregate calculation formula specific to the selected study stream or university admission criteria.

The calculated aggregate is then displayed to the user, providing transparency and ensuring they understand their eligibility for different fields and universities. This aggregate is stored in the database and used to generate tailored recommendations for universities and fields of study.

The system allows for recalculation if the user updates their marks or enters expected scores. It ensures accuracy by applying validation checks on the input data before performing the calculations. This feature gives users a clear understanding of where they stand academically and guides them in making informed decisions about their future academic paths.

Table 12: Calculation use case

|  |  |  |  |
| --- | --- | --- | --- |
| **UC5.1** | **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

Table 13 outlines the use case for displaying the calculated aggregate to the user. Once the system computes the aggregate based on the user's input marks and predefined formulas, it presents the result in a user-friendly format on the dashboard or a dedicated results page. The aggregate is shown as a percentage or score, making it easy for users to understand their academic standing.

The display includes a brief explanation of how the aggregate was calculated, such as the weightage given to Matric, FSC, and test scores (if applicable), to ensure transparency. Additionally, users are provided with a comparison of their aggregate against the eligibility criteria of recommended fields and universities.

The system also provides a graphical representation or a ranking system to help users visualize their position among potential peers. This feature enhances the decision-making process by clearly linking the aggregate to personalized recommendations, empowering users to explore suitable academic and career options confidently.

Table 13: Aggregate use case

|  |  |  |  |
| --- | --- | --- | --- |
| **UC5.2** | **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

Table 14 outlines the use case for recommending universities based on the user's calculated aggregate. The process begins with the system retrieving the user’s aggregate score from the database. It compares the score against predefined admission criteria for universities stored in the database, including eligibility thresholds for specific fields of study.

The system generates a list of universities where the user meets or exceeds the eligibility criteria. Each recommendation includes details such as the university name, location, admission criteria, application deadlines, and a link to the online application portal.

The system highlights universities offering scholarships or financial aid options for users who qualify. Users can also filter the recommendations by location, field, or admission deadlines, ensuring a customized and user-centric experience.

Table 14: Recommendation aggregate based use case

|  |  |  |  |
| --- | --- | --- | --- |
| **UC6.1** | **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

Table 15 outlines the use case for providing detailed information about universities to the user. Once the user selects a university from the recommended list, the system retrieves and displays all relevant details about that university from the database. This includes the university name, location, admission criteria, fields of study offered and a direct link to the application portal.

The system also provides additional details, such as available scholarships, tuition fees, campus facilities, and global rankings (if available), giving users a comprehensive understanding of their options. The details are presented in a structured and user-friendly format, ensuring that users can easily access the information they need to make informed decisions. The system allows users to bookmark or save specific universities for future reference.

Table 15: University details use case

|  |  |  |  |
| --- | --- | --- | --- |
| **UC6.2** | **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

Table 16 outlines the use case for displaying field trends to the user. The system retrieves data from the database that contains growth trends, demand indices, and other relevant statistics for various fields of study. This information is visualized using charts, graphs, or tables to make it easy for users to interpret.

The trends include historical growth rates, current demand in the job market, and future projections for specific fields. Users can filter these trends by location (e.g., Pakistan or global), time periods and field categories (e.g., Engineering, Medical, Arts). This allows users to explore which fields are growing, stable or declining.

The system highlights key insights, such as the top five growing fields or fields with the highest global demand, providing actionable information for decision-making.

Table 16: Display trends use case

|  |  |  |  |
| --- | --- | --- | --- |
| **UC7.1** | **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

Table 17 outlines the use case for comparing fields and viewing demand trends. The system allows users to select two or more fields of interest from a predefined list. Once the fields are chosen, the system retrieves data from the database and generates a comparison based on key metrics such as growth rates, demand indices, and career prospects.

The comparison is displayed using graphical tools like bar charts, line graphs, or tables, highlighting differences and similarities between the selected fields. Users can view trends over specific time periods and across different locations, such as Pakistan or global markets. This feature provides users with a clear understanding of which fields are growing, stable, or declining, helping them make more informed choices.

The system provides actionable insights, such as highlighting the most in-demand field or identifying fields with strong future projections.

Table 17: Comparison use case

|  |  |  |  |
| --- | --- | --- | --- |
| **UC7.2** | **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

Table 18 outlines the use case for viewing personalized recommendations tailored to the user's academic profile, interests, and preferences. The system retrieves user-specific data, including marks, selected study streams, interests, and calculated aggregates, to generate a personalized list of recommended fields and universities.

The recommendations are displayed in a dashboard format, providing an easy-to-navigate interface where users can explore detailed information about suggested fields of study and universities. Each field recommendation includes its relevance to the user's academic profile and market trends. Similarly, university recommendations highlight eligibility criteria, deadlines, and application links.

This feature enhances decision-making by offering a data-driven, user-centric approach to academic and career planning.

Table 18: Personalized recommendation use case

|  |  |  |  |
| --- | --- | --- | --- |
| **UC8.1** | **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

Table 19 outlines the use case for visualizing trends and comparing multiple fields of study. The system provides users with an interactive dashboard that retrieves data on growth trends, demand indices, and historical and future projections for various academic fields. Users can select specific fields and view their trends over time using graphical representations such as bar charts, line graphs, or pie charts.

The system also enables users to compare multiple fields side-by-side based on factors like growth rate, demand in local and global markets, and career opportunities. This comparison is visually highlighted to show key differences and similarities, helping users understand which fields are more promising or aligned with their goals.

Users can customize the parameters for the visualizations, such as filtering by location or time range, and save their comparisons for future reference. This feature enhances the platform's utility by making complex trend data accessible and actionable for users.

Table 19: Visualization use case

|  |  |  |  |
| --- | --- | --- | --- |
| **UC8.2** | **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

Table 20 outlines the use case for filtering universities based on user preferences and eligibility. The system provides an interface where users can apply various filters, such as location, field of study, admission criteria, and application deadlines. The filters are designed to help users narrow down their options and focus on universities that align with their academic profile and preferences.

Once the filters are applied, the system retrieves data from the database and displays a customized list of universities that meet the selected criteria. Each university listing includes essential details such as location, admission requirements, available fields, application links, and deadlines.

System allows users to sort the filtered results by ranking, proximity, or relevance to their selected field of study. Users can also save their filtered lists for later reference or apply additional filters for further refinement. This feature ensures a streamlined and efficient process for exploring university options tailored to the user's unique needs and goals.

Table 20: Filtration use case

|  |  |  |  |
| --- | --- | --- | --- |
| **UC9.1** | **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.

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

1. The system shall restrict access to administrative functionalities and user data to authorized personnel only.
2. 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

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.

### ****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 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. Conduct user testing with a sample audience to identify and address usability challenges.
4. Incorporate feedback loops for continuous improvement.
5. 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 Git for version control to track changes and facilitate collaborative development.
2. Maintain comprehensive documentation for developers to understand system components and workflows.

### ****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.
4. Incorporate a layered architecture to enable seamless integration of new features, APIs, or third-party tools, ensuring each layer remains independent and adaptable as the system evolves.

# Design Requirements

## 5.1 High Level Design

### 5.1.1 Abstract Design

Figure 8 represents the abstract design of the system, showcasing the user's journey from signup and login to inputting academic details, selecting fields, and interacting with the chatbot for interest identification. Based on the provided data, the system recommends universities with links to their applications and allows users to view analytics and comparisons of selected or trending fields.

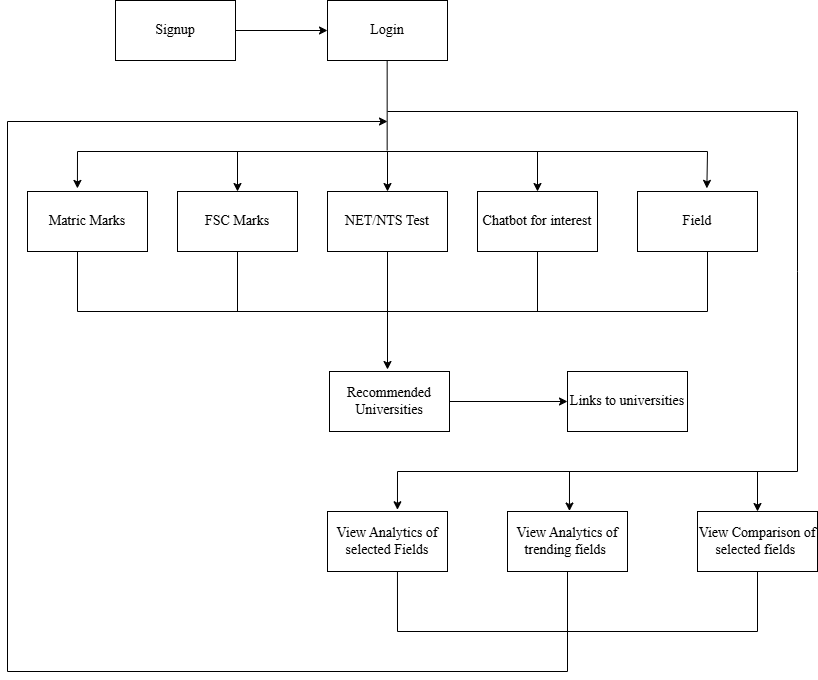


Figure 8: Abstract design

### 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.

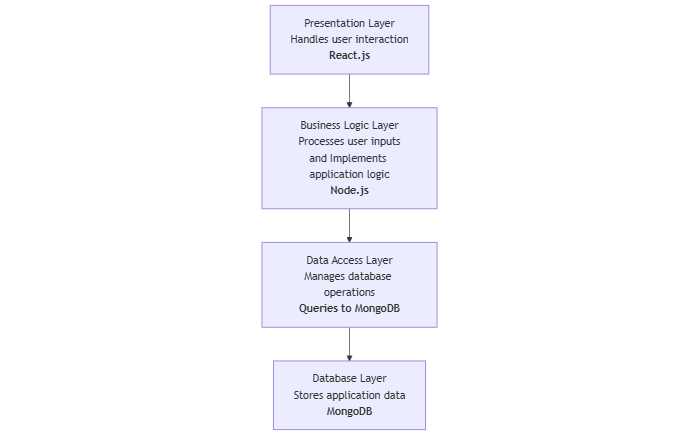
****

Figure 9: Layered Architecture Pattern

## 5.2 Structural Design

### 5.2.1 Class Diagram

Figure 10 illustrates the class diagram of the system, detailing the primary classes, their attributes, and methods, along with the relationships between them. The diagram includes key classes such as User, Admin, Academic Details, University, Recommendation Engine, Trends Analyzer, and Dashboard, each with relevant attributes and operations that define their functionality.

The User class connects to Academic Details for storing user marks and to the Dashboard for accessing recommendations and trends. The Recommendation Engine processes user data to generate field and university recommendations while the Trends Analyzer provides insights into field trends and comparisons. The Admin class manages data updates and user administration, ensuring the system's smooth operation. This structure demonstrates a cohesive and modular design that supports core system functionalities.

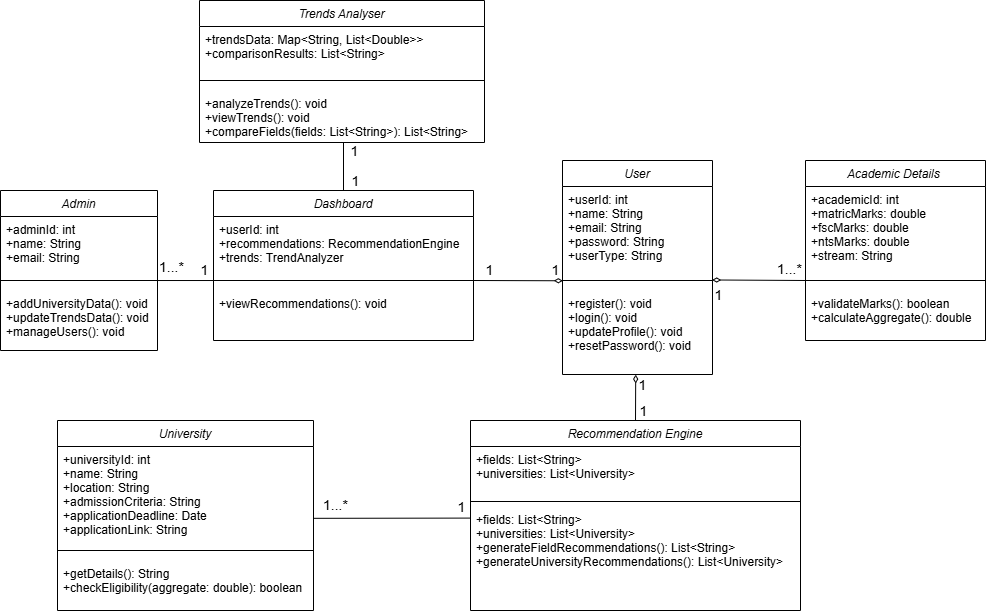


Figure 10: Class Diagram

### 5.2.2 Component Diagram

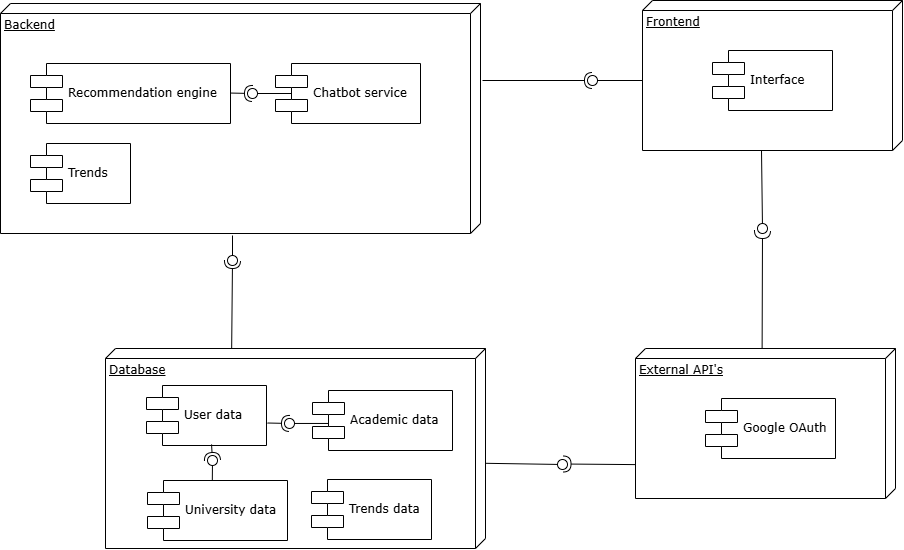
**Figure 11** represents the component diagram of the system, illustrating the modular architecture and interactions between different components.

Figure 11: Component Diagram

### 5.2.3 Data Flow Diagram

**Figure 12, Figure 13 and Figure 14** depict the progression of Data Flow Diagrams for the system showing how data flows through different levels of abstraction.

**Context Level DFD**

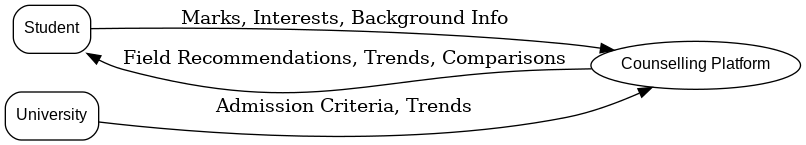


Figure 12: Context level DFD

**Level 1 DFD**

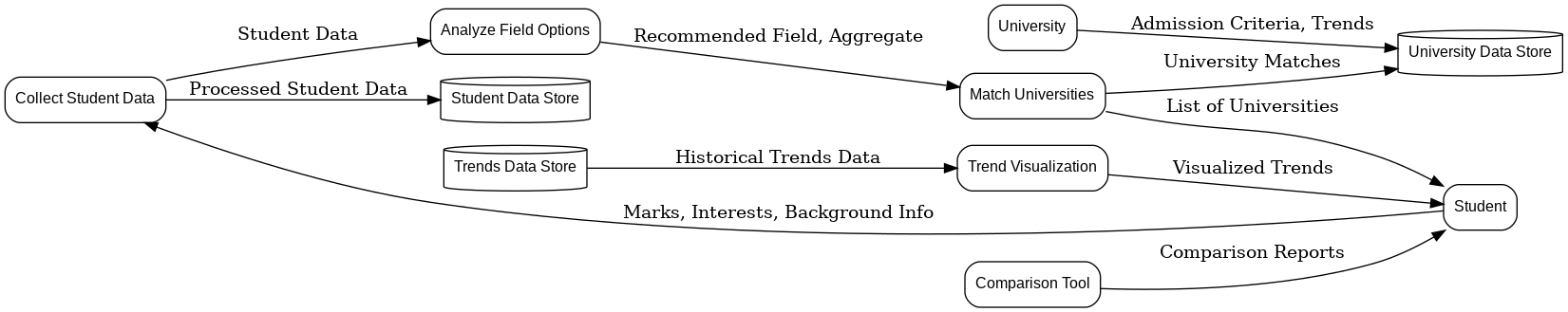


Figure 13: Level 1 DFD

**Level 2 DFD**

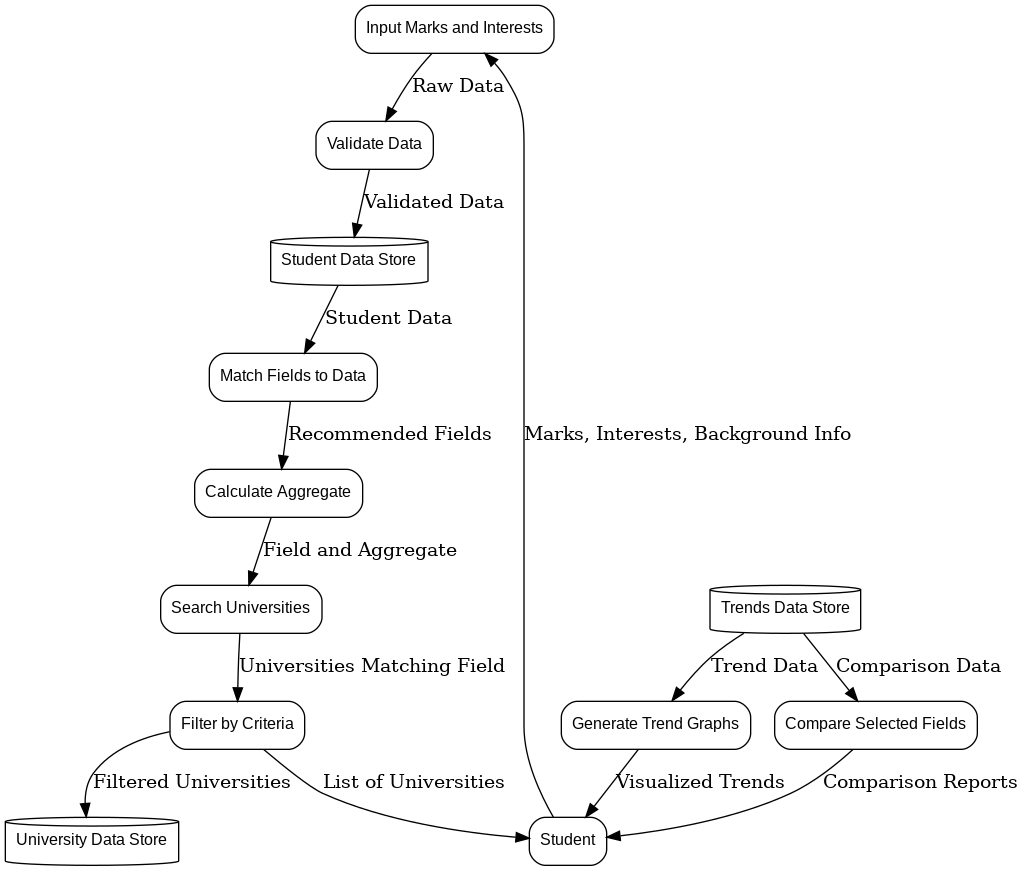


Figure 14: Level 2 DFD

### 5.2.4 Graphical User Interface

Figures 15 to 20 showcase the graphical user interface design of the system, presenting an intuitive and user-friendly layout that guides users through the platform's functionalities. The Landing Page Figure 15 serves as the system's introduction, welcoming users with a clean design that provides options for signing up, logging in or exploring the platform's key features. Once logged in users are directed to the Home Page Figure 16, which serves as a central hub for navigating the system's functionalities. The home page is designed to provide users with quick access to input academic data, view recommendations and explore analytics, ensuring a streamlined user journey.

The Dashboard Page Figure 17 is the system's core interface, where users can access personalized recommendations, track their academic data and explore insights about universities and fields of study. It integrates data visualizations and highlights key trends to facilitate informed decision-making. The Trends Analytics Page Figure 18 provides in-depth analysis and comparisons of field trends using graphical tools such as charts and graphs, helping users evaluate the growth and demand of various fields. Supporting pages like About Us Figure 19 and Contact Us Figure 20 enhance user engagement by offering transparency about the platform.

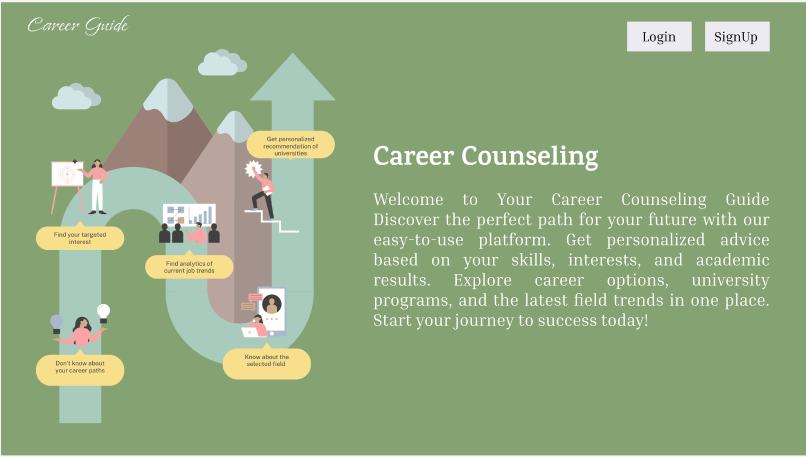


Figure 15: GUI Landing page

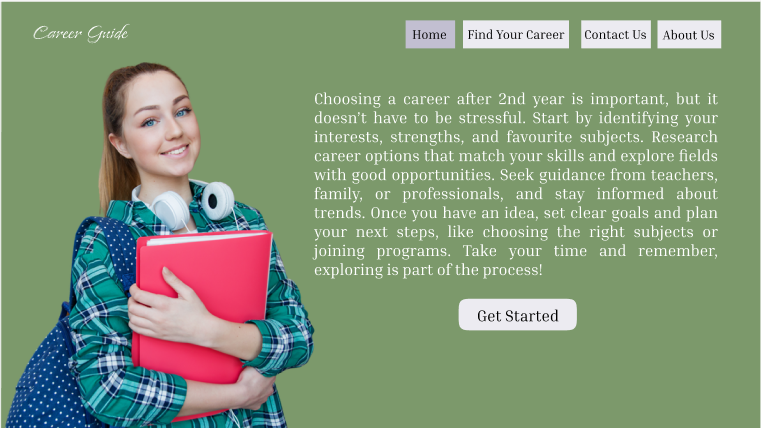


Figure 16: GUI Home page

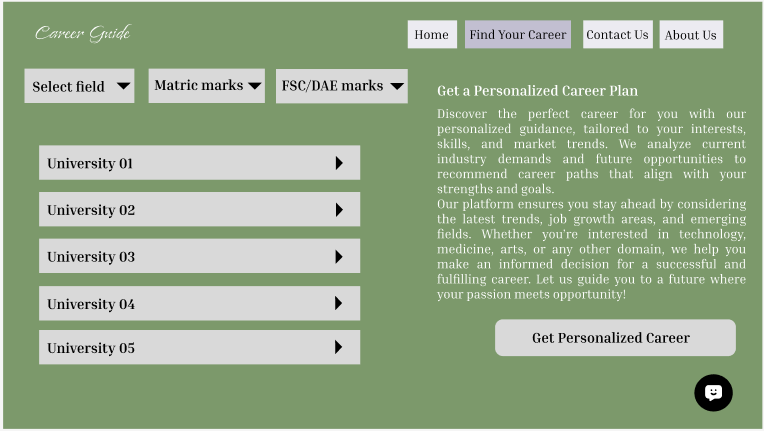


Figure 17: GUI Dashboard

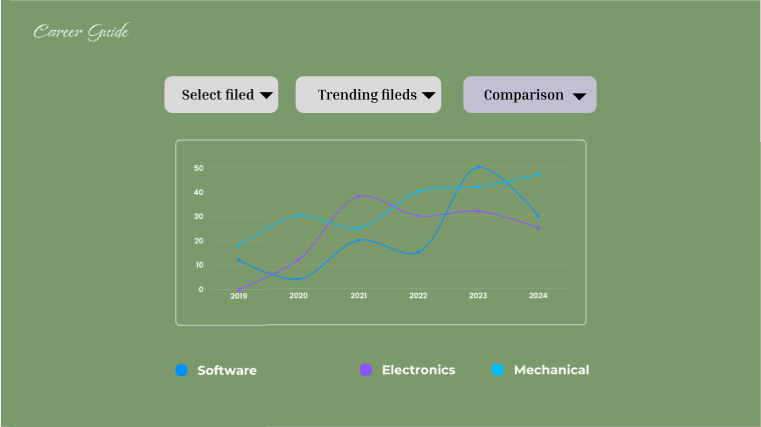


Figure 18: GUI Trends Analytics

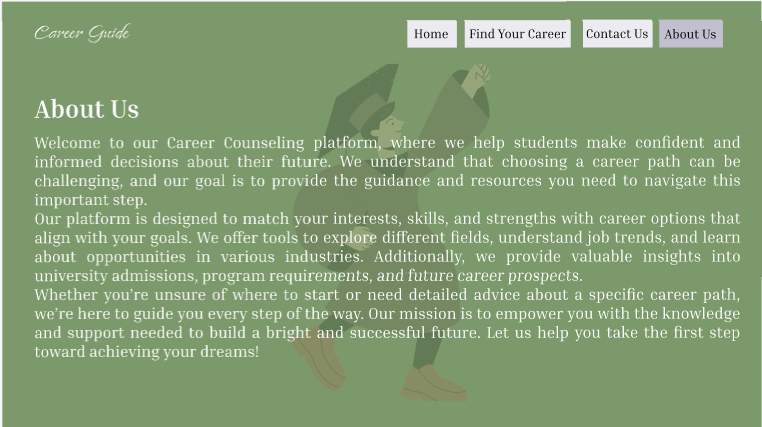
****

Figure 19: GUI About us page

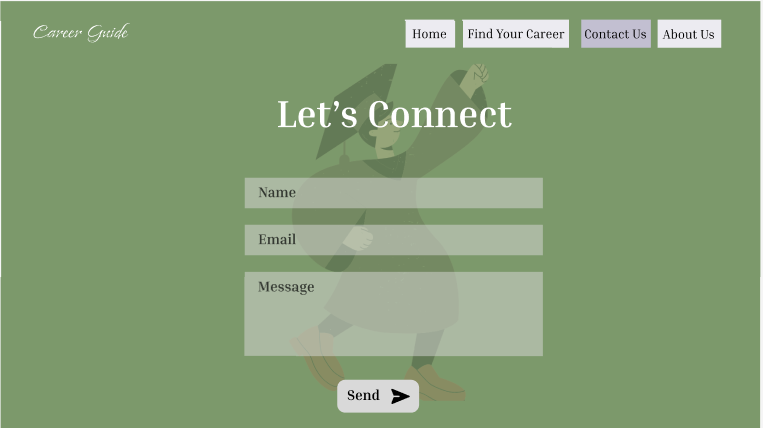


Figure 20: GUI Connect us page

## 5.3 Behavioral Diagram

### 5.3.1 Use Case Diagram

Figure 21 illustrates the use case diagram for the AI-Based Career Counseling and Career Transition Recommender System, highlighting the interactions between the User, Admin, and the system's core functionalities. The diagram provides an overview of the primary use cases, such as Signup, Login, Data Input and Validation, Field Recommendation, Aggregate Calculation, University Recommendation, Trend Visualization, Dashboard for Insights, and Search and Filter Options.

The User interacts with the system for activities like registration, providing academic data, receiving recommendations, and exploring trends. The Admin oversees key functionalities, such as managing the database and updating system data. Include and Extend relationships are used to depict dependencies and optional features. For example, Authentication Validation is included in both the Signup and Login processes to ensure secure access, while the Interest Finding Chatbot extends the Data Input and Validation use case to refine recommendations based on user interests. This diagram effectively demonstrates the modular structure and user-centered design of the system.

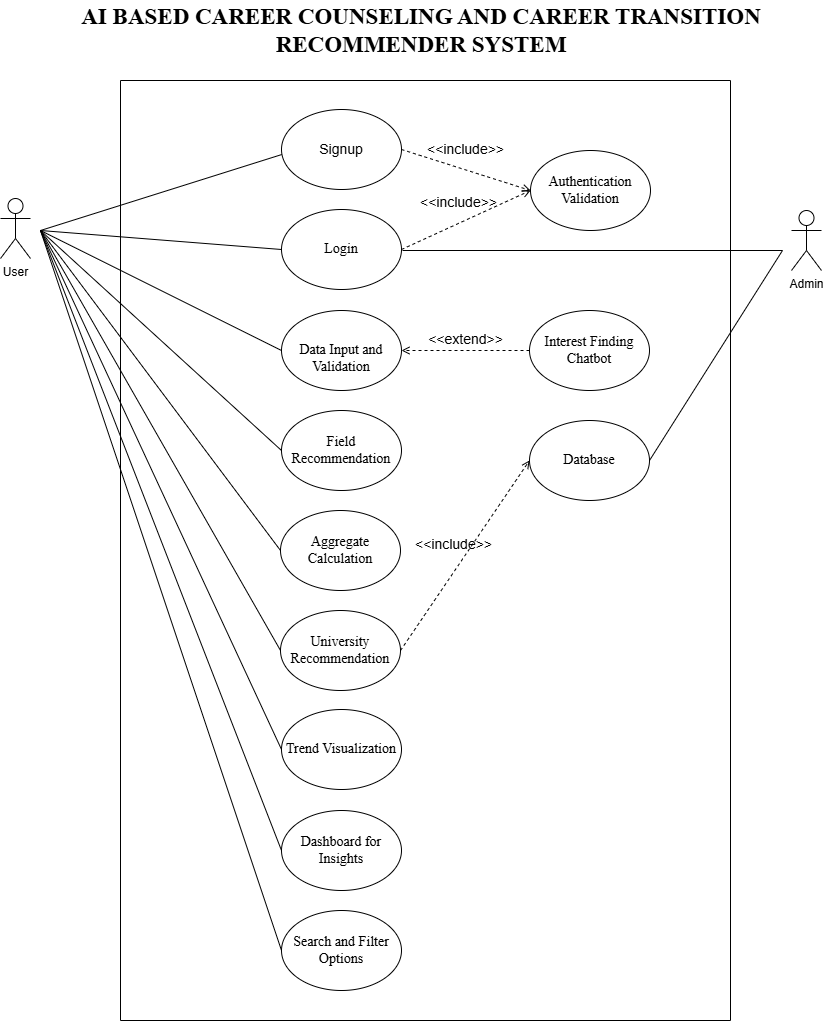


Figure 21: Use Case Diagram

### 5.3.2 Sequence Diagram

Figures 22 to 26 collectively illustrate the sequence diagrams for the primary functionalities of the AI-Based Career Counseling and Career Transition Recommender System, detailing the flow of interactions between the user and the system. Figure 22 demonstrates the user registration process, where the user submits their details, which are validated by the system before being securely stored in the database. A confirmation is sent back to the user, ensuring a smooth onboarding process. Similarly, Figure 23 outlines the login and forgot password sequences. In the login process, the system validates the user’s credentials and grants access to their personalized dashboard, while the forgot password process involves sending a reset link or code to the user’s email, enabling secure password recovery.

Figures 24 to 26 delve into the system's analytical and recommendation functionalities. Figure 24 showcases the sequence of inputting academic data and generating university recommendations, where the system calculates the user’s aggregate and queries the database for eligible universities based on the user’s data. Figure 25 represents trends analytics, highlighting how the system retrieves, processes, and visualizes data on field trends and demand indices, providing users with actionable insights. Lastly, Figure 26 focuses on field comparison analytics, enabling users to compare multiple fields side by side using growth rates, demand, and career opportunities, presented through intuitive visualizations.

**User Registration**

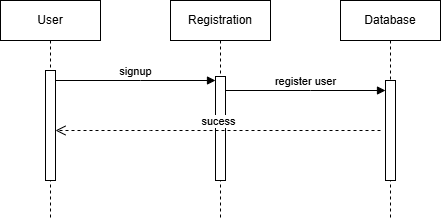


Figure 22: User registration

**Login and Forgot Password**

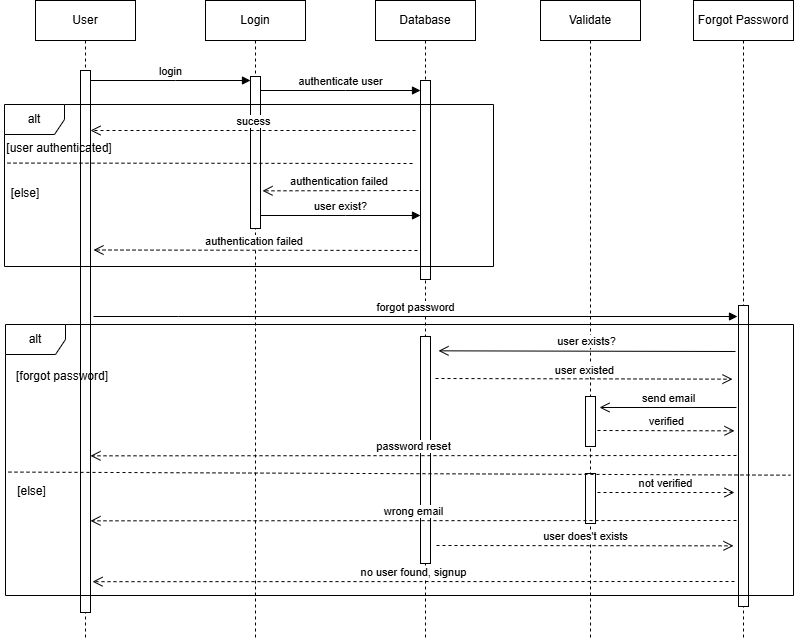


Figure 23: User login and forgot password

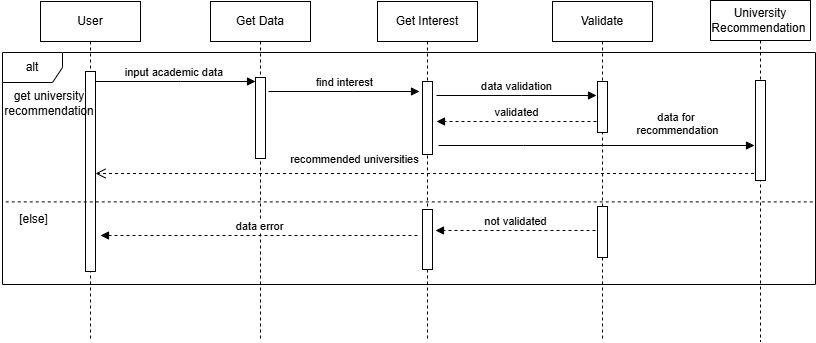


Figure 24: Input Data and University Recommendation

**Trends Analytics**

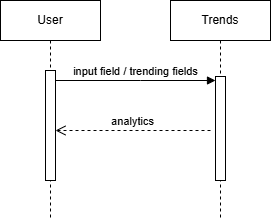


Figure 25: Trend analytics

**Fields Comparison Analytics**

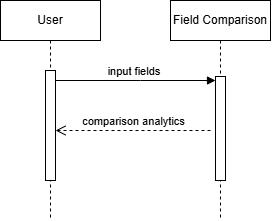


Figure 26: Comparison analytics

### 5.3.3 State Machine Diagram

Figure 27 represents the state machine diagram for the career counseling system, illustrating the flow of states and transitions based on user interactions and system responses. It provides a clear depiction of how the system processes user inputs, handles errors and delivers actionable outputs like recommendations and analytics.

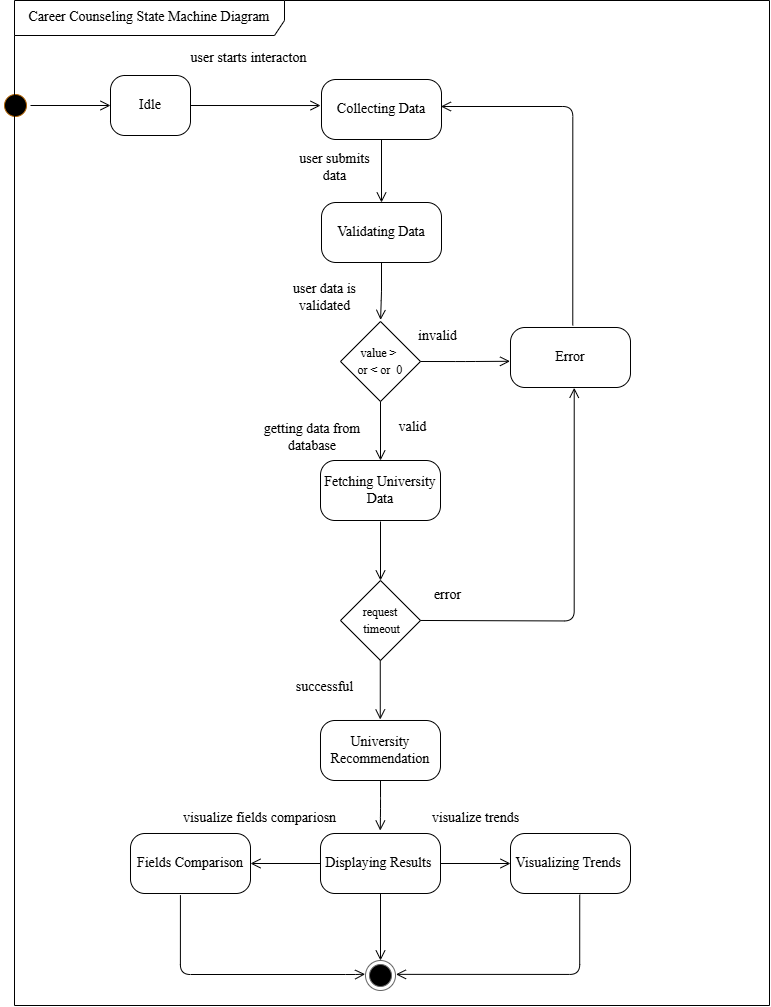


Figure 27: State Machine Diagram

# Data Design and Relationships

## 6.1 Database Schema

The database design for the career counseling system is structured to efficiently manage and store all the necessary data for personalized recommendations, analytics, and user management. Table 21 serves as the foundation, storing essential user information and roles, while Table 22 users' academic performance and selected streams, enabling aggregate calculations for field and university recommendations.

Supporting this core functionality, Table 23 links users to their recommended fields and universities, while Table 24 stores data on field growth rates and demand trends to power trend analytics. Table 25 provides comprehensive information about available universities, ensuring accurate and tailored suggestions for users. Together, these interconnected tables create a cohesive and scalable database that supports the platform's goal of providing effective career counseling and academic planning.

The Academic Details table works in conjunction with the Trends table to provide users with recommendations that are not only based on their academic performance but also aligned with current market trends and field growth rates. Similarly, the Recommendations table integrates with the Universities table to provide users with detailed and actionable university suggestions, including application links and eligibility criteria.

### 6.1.1 User Table

Table 21: 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 |

### 6.1.2 Academic Details Table

Table 22: 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 |

### 6.1.3 Recommendations Table

Table 23: Recommendation 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) |

### 6.1.4 Trends Table

Table 24: 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**

Table 25: University 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

Figure 28 shows the Entity Relationship Diagram for our system including database tables

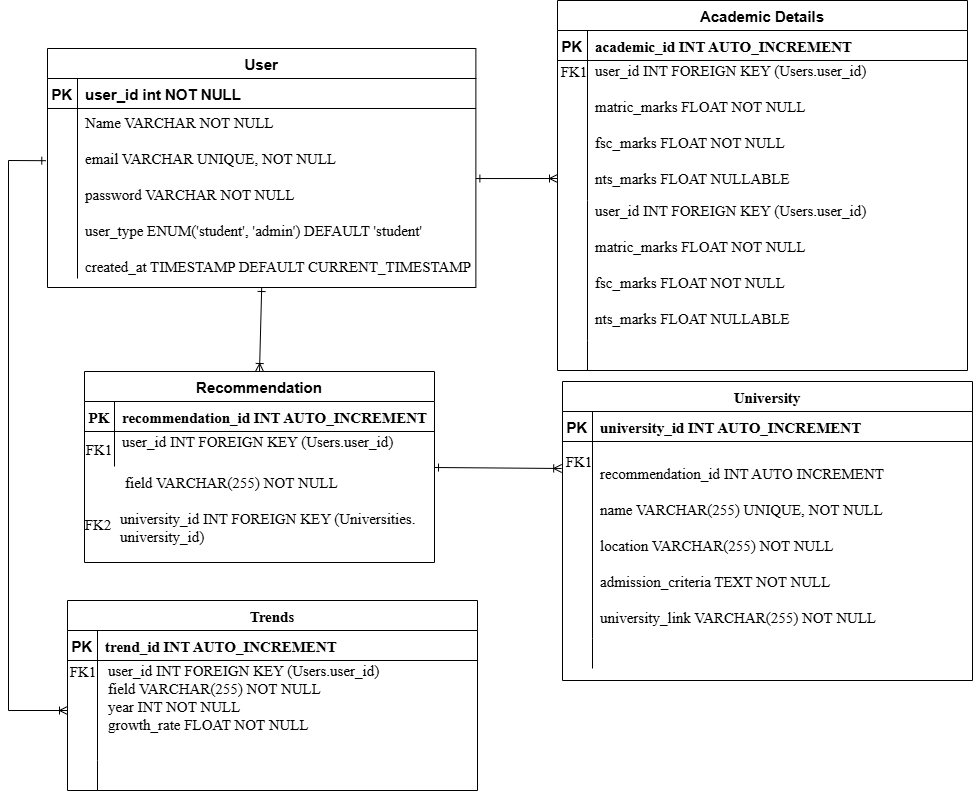


Figure 28: Entity Relationship Diagram

## 6.3 Data Dictionary Diagram

Figure 29 shows the Data Dictionary Diagram of the system

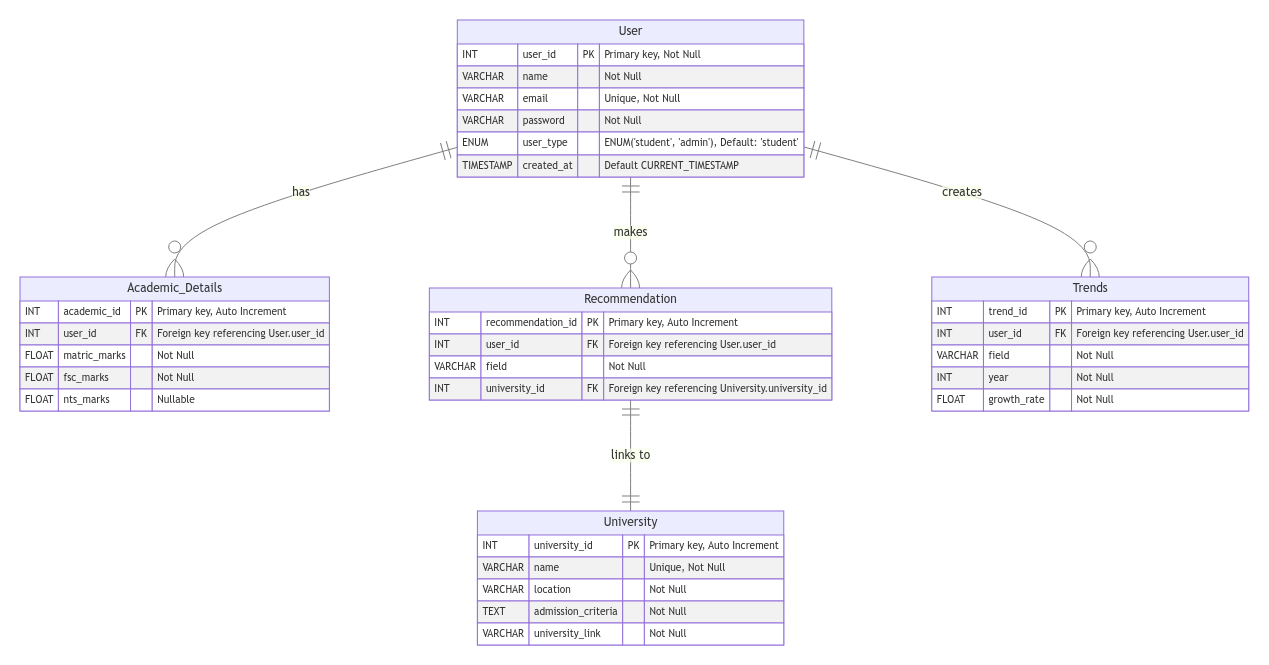


Figure 29: Data Dictionary Diagram

# Software Planning and Timeline

## 7.1 Work Breakdown Structure

**Figure 30** illustrates the Work Breakdown Structure for the career counseling system.

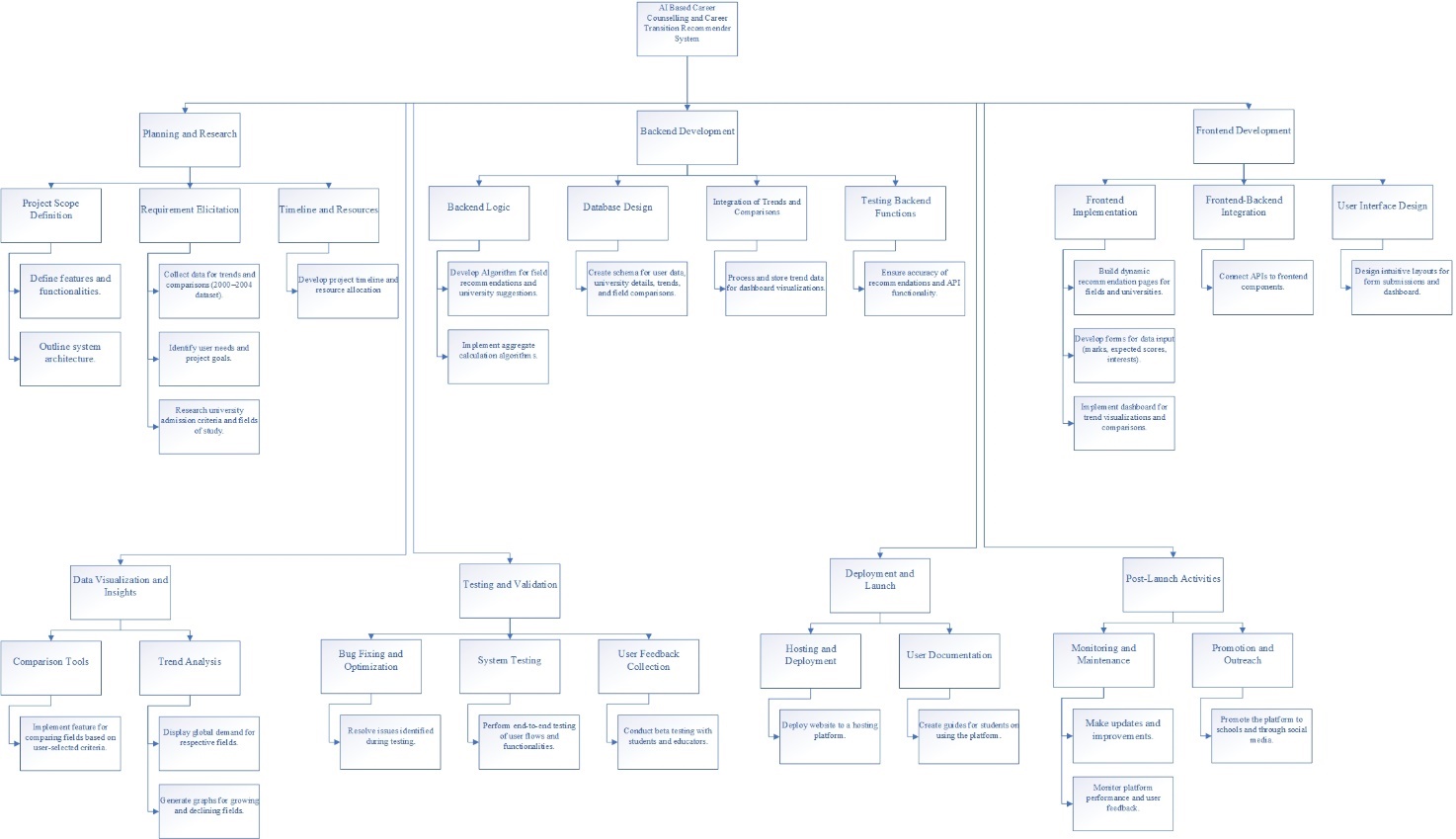
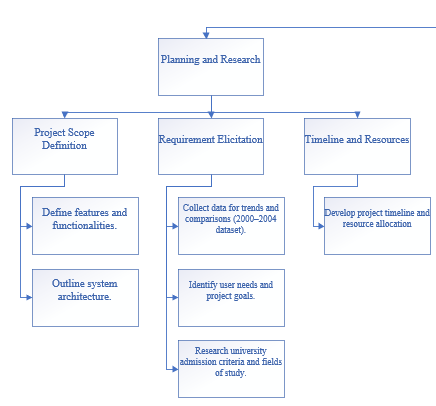
****

Figure 30: Work Breakdown Structure

Figures 31 to 36 depict the detailed Work Breakdown Structures (WBS) for various phases of the career counseling system, outlining the specific tasks involved in each stage of the project. Figure 31 focuses on the planning and research phase, which includes identifying user needs, researching university admission criteria and collecting datasets for trends and comparisons. These foundational tasks ensure that the project goals are well-defined and grounded in accurate, comprehensive data. Figure 32 highlights frontend development.

Figures 33 to 36 delve into backend development, data visualization, testing, and deployment. Figure 33 emphasizes backend development tasks like database schema creation, implementing recommendation algorithms, and integrating trends and university data for seamless functionality. Figure 34 outlines tasks for generating visual analytics, including trend graphs and field comparisons to provide actionable insights to users. Figure 35 ensures system reliability through extensive testing and validation, while Figure 36 addresses deployment and post-launch activities, such as hosting the platform, creating user documentation, and ongoing maintenance.



|  |
| --- |
| Figure 31: Planning and Research WBS    Figure 32: Frontend Development WBS    Figure 33: Backend Development WBS    Figure 34: Data Visualization and Insights WBS    Figure 35: Testing and Validation WBS    Figure 36: Deployment and Launch, Post Launch Activities WBS |

## 7.2 Milestones and Deliverables

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

Figure 37 shows the milestone 1 of planning and esearch ad described below

#### **Sub-Milestone 1.1: Requirement Elicitation**

1. Identified user needs and project goals.
2. Research report on university admission criteria and fields of study.

Dataset for trends and comparisons (2000–2004 dataset).

#### **Sub-Milestone 1.2: Project Scope Definition**

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

**Sub-Milestone 1.3: Timeline and Resource Allocation**

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

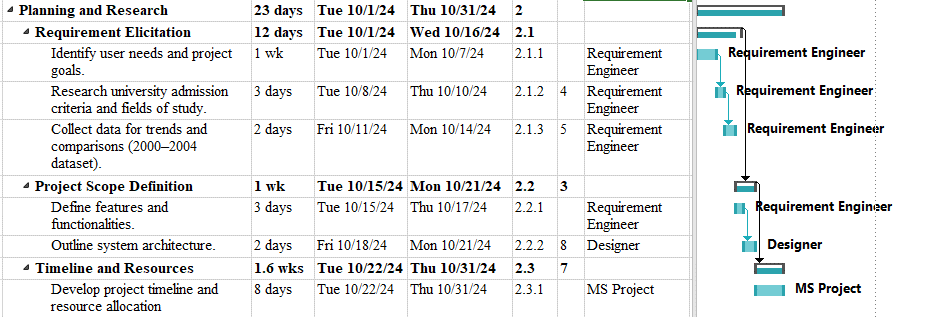


Figure 37: Milestone 1Planning and Research

### ****7.2.2 Milestone 2: Backend Development****

Figure 38 shows the milestone 2 of backend development explained below

#### **Sub-Milestone 2.1: Database Design**

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

#### **Sub-Milestone 2.2: Backend Logic**

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

#### **Sub-Milestone 2.3: Integration of Trends and Comparisons**

1. Processed and stored trend data for dashboard visualizations.

#### **Sub-Milestone 2.4: Backend Testing**

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

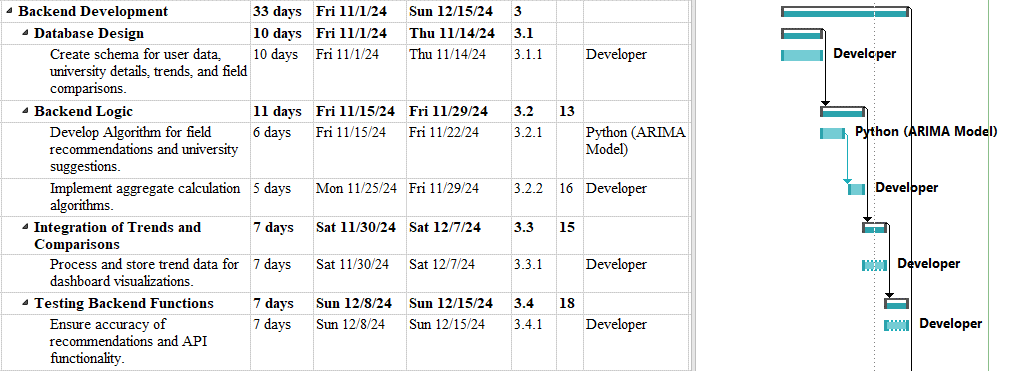


Figure 38: Milestone 2 Backend Development

### ****7.2.3 Milestone 3: Frontend Development****

Figure 39 shows the milestone 3 of frontend development described below

#### **Sub-Milestone 3.1: User Interface Design**

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

#### **Sub-Milestone 3.2: Frontend Implementation**

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 3.3: Frontend-Backend Integration**

1. Functional APIs connected to frontend components.

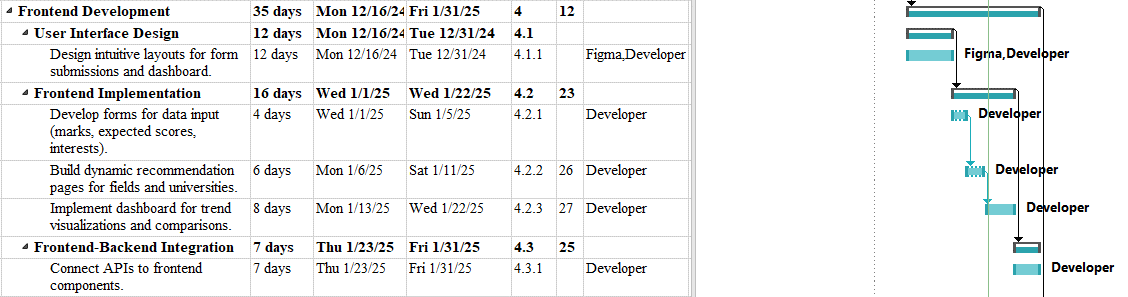


Figure 39: Milestone 3 Frontend Development

### ****7.2.4 Milestone 4: Data Visualization and Insights****

Figure 40 explains the milestone 4 of data visualization and insights describe below

#### **Sub-Milestone 4.1: Trend Analysis**

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

#### **Sub-Milestone 4.2: Comparison Tools**

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

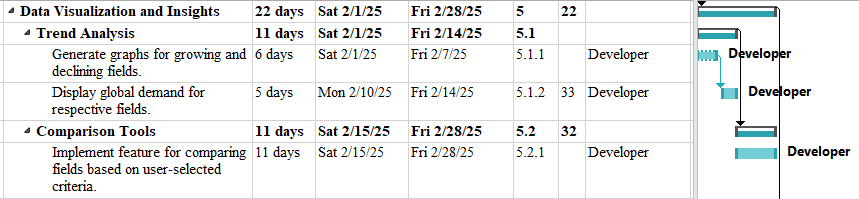


Figure 40: Data Visualization and Insights

### ****7.2.5 Milestone 5: Testing and Validation****

Figure 50 shows the milestone 5 of testing and validation described below

#### **Sub-Milestone 5.1: System Testing**

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

#### **Sub-Milestone 5.2: User Feedback Collection**

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

#### **Sub-Milestone 5.3: Bug Fixing and Optimization**

1. Resolved issues and optimized system performance.

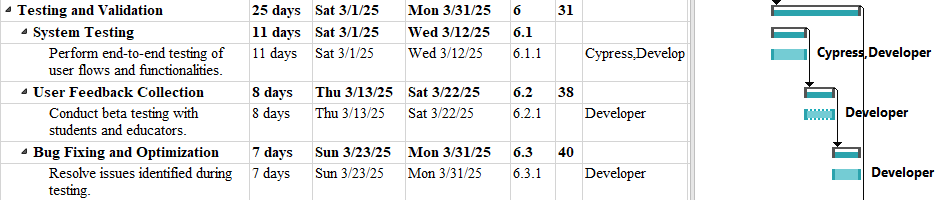


Figure 41: Testing and Validation

### ****7.2.6 Milestone 6: Deployment and Launch****

Figure 45 shows the milestone 6 of deployment and launch explained below

#### **Sub-Milestone 6.1: Hosting and Deployment**

1. Live website deployed to a hosting platform.

#### **Sub-Milestone 6.2: User Documentation**

1. User guides and tutorials for using the platform.

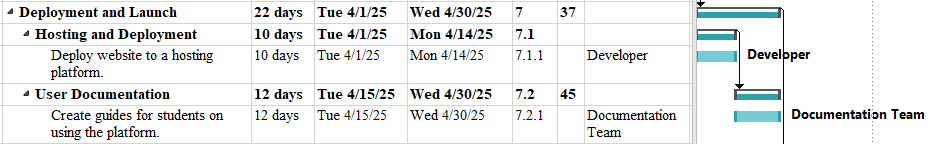


Figure 42: Deployment and Launch

### ****7.2.7 Milestone 7: Post-Launch Activities****

Figure 43 shows the milestone 7 of post launch activities described below

#### **Sub-Milestone 7.1: Monitoring and Maintenance**

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

#### **Sub-Milestone 7.2: Promotion and Outreach**

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

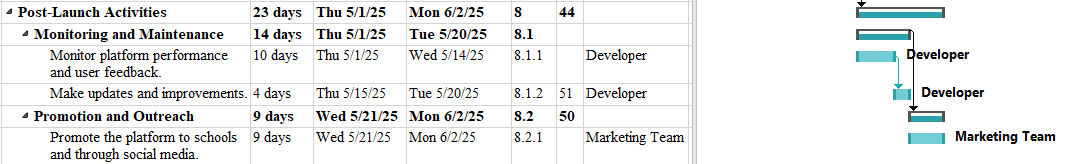


Figure 43: Milestone 7 Post Launch Activities

## 7.3 Detailed Baseline Plan

Figures 31 to 36 present the Work Breakdown Structures for the career counseling system, covering all major phases of the project. These include planning and research Figure 31, where user needs and datasets are identified, frontend development Figure 32, focusing on user-friendly interfaces and responsive dashboards, and backend development Figure 33, which involves database creation, recommendation algorithms, and data integration. Additionally, data visualization tasks Figure 34 focus on generating trend graphs and comparisons, while testing and validation Figure 35 ensure system reliability. Finally, deployment and post-launch activities Figure 36 involve hosting, documentation, and maintenance. Together, these WBS diagrams provide a structured and comprehensive roadmap for efficient project execution.

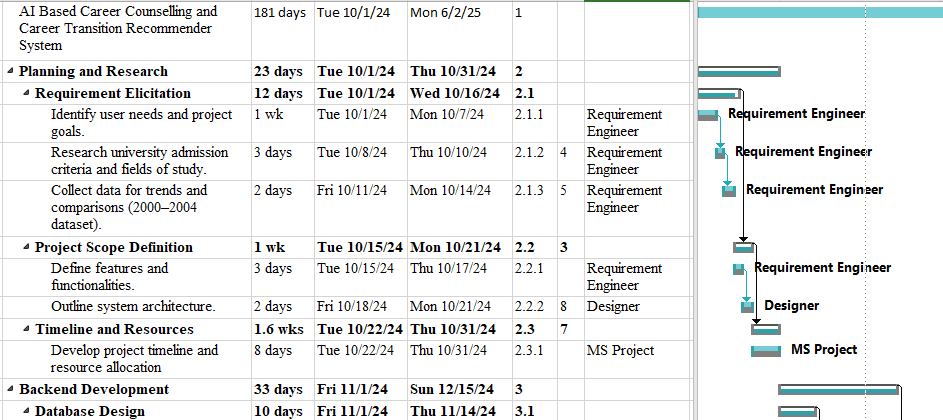


Figure 44: Baseline Plan part 1

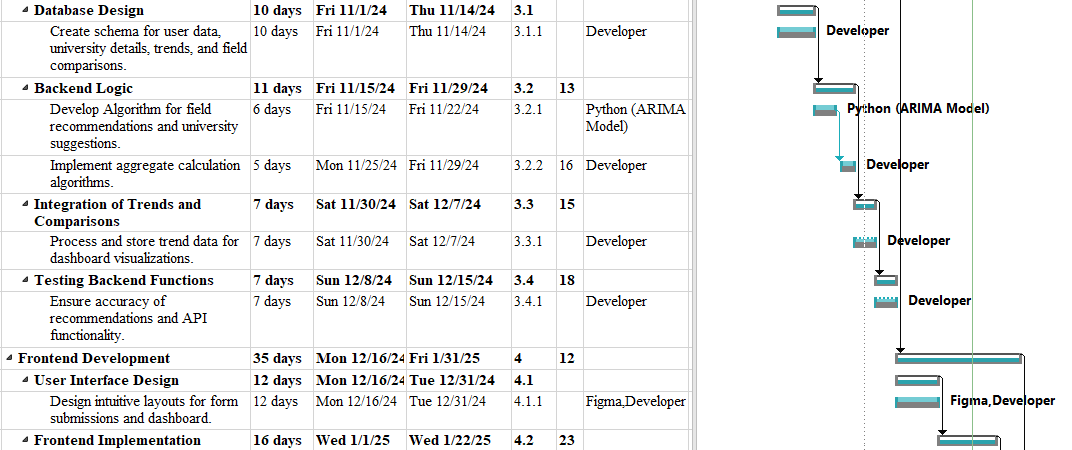


Figure 45: Baseline Plan part 2

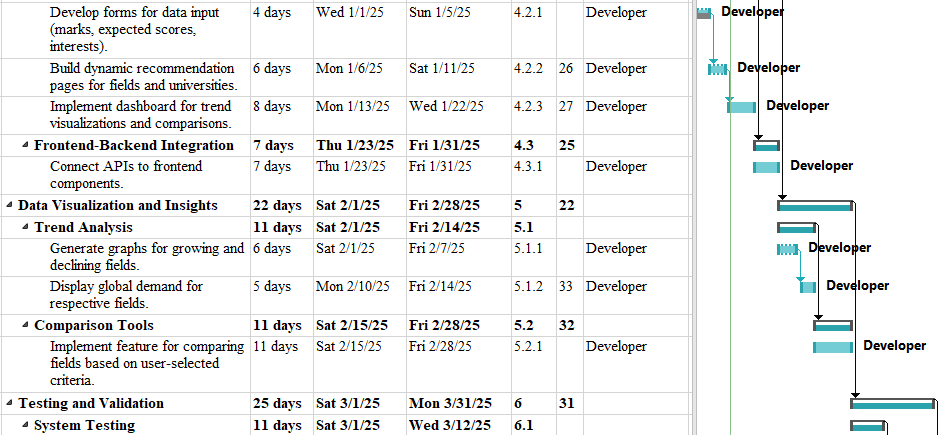


Figure 46: Baseline Plan part 3

# 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 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 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****

Table 26 shows roles and responsibilities in the system.

Table 26: 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).

**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.

### ****8.3.7 Defect Management****

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

### ****8.3.8 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.

### ****8.3.9 Tools and Resources****

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

### ****8.3.10 Risk Management****

Table 27 shows risk and their mitigation strategies.

Table 27: Risk and Mitigation

|  |  |
| --- | --- |
| **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. |

### ****8.3.11 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.
4. Version controlling of document and code.
5. Proper configuration management process is followed.

### ****8.3.12 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****

Table 28 show the test case for user registration and login.

Table 28: Test Suit 1 User Registration and Login

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Description** | **Steps** | **Expected Result** |
| Test Case 1.1: User Registration | Validate that users can successfully create an account. | 1. Navigate to the registration page.  2. Enter valid email, password, and confirm password.  3. Click "Register." | Account is created successfully, and a confirmation email is sent. |
| Negative Test Case | Enter an invalid email or mismatched passwords. Ensure an error message is displayed. | Attempt registration with invalid inputs. | Error message is displayed. |
| Test Case 1.2: User Login | Validate that registered users can log in. | 1. Navigate to the login page.  2. Enter valid email and password.  3. Click "Login." | User is redirected to the dashboard. |
| Negative Test Case | Enter incorrect email/password. Ensure an error message is displayed. | Attempt login with incorrect credentials. | Error message is displayed. |
| Test Case 1.3: Google OAuth Login | Validate login via Google OAuth. | 1. Click "Login with Google."  2. Choose a valid Google account. | User is redirected to the dashboard. |

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

Table 29 shows the test case for data input and validation.

Table 29: Test Suite Data Input and Validation

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Description** | **Steps** | **Expected Result** |
| Test Case 2.1: Input Valid Data | Validate that users can input valid academic details. | 1. Enter Matric and FSC marks (numeric values).  2. Select a study stream (e.g., Pre-Medical, Pre-Engineering).  3. Click "Submit." | Data is saved successfully. |
| Test Case 2.2: Input Invalid Data | Test how the system handles invalid data. | 1. Enter non-numeric or negative values for marks.  2. Leave required fields empty. | Error messages are displayed. |
| Test Case 2.3: Enter Expected Marks | Validate that users can enter expected marks if results are pending. | 1. Select "Enter Expected Marks."  2. Input expected scores for Matric, FSC, and tests. | Expected marks are saved successfully. |

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

Table 30 shows the test case for recommendation system.

Table 30: Test Suite Recommendation System

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Description** | **Steps** | **Expected Result** |
| Test Case 3.1: Field Recommendations | Validate that fields are recommended based on user data. | 1. Enter marks and study stream.  2. Click "Get Recommendations." | List of suitable fields is displayed. |
| Test Case 3.2: University Recommendations | Validate that universities are recommended based on calculated aggregate. | 1. Select a field of study.  2. View recommended universities. | Relevant universities are displayed with admission criteria and deadlines. |
| Test Case 3.3: Interest-Based Recommendations | Validate recommendations based on user interest. | 1. Interact with the chatbot.  2. Provide details about interests.  3. View recommended fields and universities. | Recommendations align with user interests. |

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

Table 31 shows the test cases for dashboard and trend visualization.

Table 31: Test Suite Dashboard and Trend Visualization

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Description** | **Steps** | **Expected Result** |
| Test Case 4.1: View Trends | Validate that trends for fields of study are displayed on the dashboard. | 1. Navigate to the dashboard.  2. Select "View Trends." | Graphs for growing and declining fields are displayed. |
| Test Case 4.2: Compare Fields | Validate the comparison of multiple fields. | 1. Select two or more fields for comparison.  2. Click "Compare." | Comparison data (growth rate, demand index) is displayed. |

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

Table 32 shows the test cases for integration testing.

Table 32: Test Suite for Integration Testing

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Description** | **Steps** | **Expected Result** |
| Test Case 5.1: Frontend-Backend Integration | Validate the integration between frontend forms and backend APIs. | 1. Fill out data input forms.  2. Check API calls using a tool like Postman. | Data is saved and processed correctly by the backend. |
| Test Case 5.2: External API Integration | Validate Google OAuth | Log in using Google OAuth. | Data is retrieved and displayed without errors. |

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

Table 33 shows the test cases for performance testing.

Table 33: Test Suit Performance Testing

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Description** | **Steps** | **Expected Result** |
| Test Case 6.1: Load Testing | Validate system performance under load. | 1. Simulate 1,000 concurrent users.  2. Monitor response times. | Response time remains under 2 seconds. |
| Test Case 6.2: Stress Testing | Test system behavior under peak traffic. | 1. Simulate 5,000 concurrent users.  2. Monitor system stability. | System handles high load without crashing. |

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

Table 34 shows the test case for security tetsing

Table 34: Test Suit Security Testing

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Description** | **Steps** | **Expected Result** |
| Test Case 7.1: SQL Injection | Test for SQL injection vulnerabilities. | 1. Enter malicious SQL code in input fields. | System rejects malicious inputs. |

### ****8.4.8**** Test Suite: System Workflow Testing

Table 35 shows the test cases for system workflow.

Table 35:Test Suit System Workflow

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Description** | **Steps** | **Expected Result** |
| Test Case 8.1: Mobile Responsiveness | Validate that the platform works on mobile devices. | 1. Access the platform on a mobile browser.  2. Navigate through forms and dashboards. | Platform is fully responsive. |
| Test Case 8.2: Navigation Flow | Validate ease of navigation. | 1. Navigate through different pages (e.g., registration, recommendations, dashboard). | All pages are easily accessible. |

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

Table 36 shows the test cases for post deployment.

Table 36: Test suit Post Deployment Testing

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Description** | **Steps** | **Expected Result** |
| Test Case 9.1: Monitor Uptime | Validate uptime and system availability. | 1. Monitor the live system for 24 hours. | Uptime is 99.9%. |
| Test Case 9.2: Monitor Performance | Test live system performance under real-world conditions. | 1. Monitor response times and error rates. | Response times remain under 2 seconds. |

**Appendix A – Data Dictionary**

Table 37: Appendix A - Data Dictionary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Description** | **Related Operations** | **Functional Requirements** |
| **userId** | Integer | Unique identifier for each user. | Create, Read, Update, Delete (CRUD). | User registration, login, account management. |
| **name** | String | User's full name. | Input during registration, view/edit profile. | User registration. |
| **email** | String | User's email address. | Input during registration, login, profile management. | User authentication and notifications. |
| **password** | String | User's encrypted password. | Input during registration and login. | User authentication. |
| **userType** | String | Type of user (e.g., admin, student). | Assigned at registration. | Role-based access control. |
| **marksMatric** | Float | User's Matric marks. | Input during data submission. | Aggregate calculation, recommendations. |
| **marksFSC** | Float | User's FSC marks. | Input during data submission. | Aggregate calculation, recommendations. |
| **testScores** | Float | User's test scores (e.g., NTS, NET). | Input during data submission. | Aggregate calculation, recommendations. |
| **expectedMarks** | Float | User's expected marks if actual results are not available. | Input during data submission. | Aggregate calculation, recommendations. |
| **studyStream** | String | User's selected study stream. | Input during data submission. | Field recommendations. |
| **recommendedFields** | Array (String) | List of fields recommended to the user. | Generated based on user data. | Field recommendations. |
| **Recommended-Universities** | Array (String) | List of universities recommended based on eligibility. | Generated based on aggregate and criteria. | University recommendations. |
| **aggregate** | Float | Calculated aggregate based on user-provided marks and predefined formulas. | Calculate, display to user. | Aggregate calculation. |
| **universityName** | String | Name of a university in the system database. | Stored, updated, or retrieved from the database. | University recommendations. |
| **admissionCriteria** | String | Admission requirements for a specific university. | Retrieved from the database and displayed to the user. | University recommendations. |
| **admissionDeadline** | Date | Application deadline for a university. | Stored and retrieved for notifications. | Notifications and recommendations. |
| **trendsData** | Dataset | Historical data on field growth trends. | Processed and visualized in the dashboard. | Trend analysis, field comparison. |
| **trendGraphType** | String | Type of graph used for trend visualization. | User-selected or auto-generated for visualization. | Trend visualization. |
| **comparisonFields** | Array (String) | List of fields selected by the user for comparison. | Compare data and display trends. | Field comparison. |
| **chatbotResponses** | String | User-provided responses during interaction with the chatbot. | Processed to determine user interests. | Interest-based recommendations. |
| **apiRequests** | Integer | Number of API calls made by the system to fetch external data (e.g., Google OAuth, university APIs). | Tracked and logged. | API integrations (login, university data). |
| **status** | String | Status of a user's request or action. | Updated dynamically for various user actions. | Notifications, user feedback. |
| **errorMessage** | String | Error messages displayed to the user during validation or processing. | Generated dynamically based on user actions. | Error handling and user guidance. |

**Appendix B - Group Log**

Table 38: Appendix B - Group Log

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Activity** | **Details** | **Participants** |
| **30/11/24** | Initial Project Discussion | Discussed the project's objectives, identified the main problem statement, and agreed on the scope of work. Finalized the idea of a career counseling platform for students. | Asad Shah, Fizza Mazhar |
| **02/12/24** | Research on Existing Systems | Researched existing career counseling platforms to identify their strengths, weaknesses, and features. Documented gaps to address in the proposed system. | Asad Shah, Fizza Mazhar |
| **05/12/24** | Requirement Gathering | Conducted a brainstorming session to outline user needs and project requirements, including key features like recommendations, trend visualizations, and chatbot integration. Identified data sources for trends and admission details. | Asad Shah, Fizza Mazhar |
| **07/12/24** | Problem Statement Finalization | Finalized the problem statement and documented challenges faced by students in career selection after 12th grade. | Asad Shah, Fizza Mazhar |
| **10/12/24** | Preliminary System Design | Drafted a high-level system architecture, including user flow, database, and functional components. | Asad Shah, Fizza Mazhar |
| **12/12/24** | Task and Work Allocation | Assigned roles and responsibilities: Asad Shah to handle backend development (algorithms and data processing), Fizza Mazhar to focus on frontend design, UI/UX, and documentation. Also planned the workflow for data collection. | Asad Shah, Fizza Mazhar |
| **15/12/24** | Dataset Identification and Research | Researched potential data sources, including online admission portals, educational databases, and historical trends for career fields from 2000–2004. Discussed initial dataset schema. | Asad Shah |
| **17/12/24** | University Data Collection | Gathered admission criteria, deadlines, and application links for top universities in Pakistan. Documented in a structured format. | Asad Shah |
| **20/12/24** | Dataset Collection and Cleaning | Gathered datasets related to admission criteria, field trends, and demand analysis for Pakistan and foreign countries. Cleaned the data to ensure consistency and format uniformity. | Asad Shah |
| **22/12/24** | User Data Inputs Planning | Defined the required input fields for users (e.g., Matric marks, FSC marks, study stream, test scores). Planned validation rules for data input. | Fizza Mazhar |
| **25/12/24** | Finalizing Dataset Structure | Created a structured dataset with tables for user data, university details, historical trends, and admission deadlines. Designed preliminary database schema based on collected data. | Asad Shah |
| **28/12/24** | Algorithm Design Planning | Outlined the logic for aggregate calculation and field recommendations. Discussed initial logic for matching user input with trends and admission criteria. | Asad Shah, Fizza Mazhar |
| **30/12/24** | Backend Planning | Planned the backend architecture, including database connectivity, API structure, and algorithm integration. | Asad Shah |
| **02/01/25** | Backend Development | Started developing algorithms for aggregate calculation, field recommendations, and university suggestions. Worked on connecting the backend logic to the dataset. | Asad Shah |
| **05/01/25** | Backend Progress Review | Reviewed progress on the backend algorithms and data processing. Addressed challenges with handling incomplete user input (e.g., expected marks). | Asad Shah, Fizza Mazhar |
| **07/01/25** | User Interface Wireframe Design | Created wireframes for forms, dashboards, and chatbot interaction screens. Focused on a responsive and user-friendly layout. | Fizza Mazhar |
| **10/01/25** | Frontend Development Planning | Planned the frontend architecture, including UI/UX for forms, dashboards, and chatbot integration. Discussed designs for trend visualizations and user interaction flows. | Fizza Mazhar |