Hamdard University

Department of Computing

Final Year Project


# Dakhla: AI-Assisted University Admissions Platform (FYP-020/FA25)

## Software Requirements Specifications

Submitted by

Muhammad Nauman Arif (2514-2022)

Ismail Dad Khan (2927-2022)

Hamza Ali Khan (2508-2022)

Supervisor

Dr. Umer Farooq

**Fall 2025**

# Document Sign off Sheet

## Document Information

| | |
|---|---|
| **Project Title** | Dakhla: AI-Assisted University Admissions Platform |
| **Project Code** | FYP-020/FA25 |
| **Document Name** | Software Requirements Specifications |
| **Document Version** | 1.0 |
| **Document Identifier** | FYP-020/FA25-SRS |
| **Document Status** | Initial Draft |
| **Author(s)** | Muhammad Nauman Arif (2514-2022)<br>Ismail Dad Khan (2927-2022)<br>Hamza Ali Khan (2508-2022) |
| **Approver(s)** | Dr. Umer Farooq |
| **Issue Date** | 14/DEC/2025 |

| Name | Role | Signature | Date |
|---|---|---|---|
| Muhammad Nauman Arif (2514-2022) | Team Lead | | 14/DEC/2025 |
| Ismail Dad Khan (2927-2022) | Team Member 2 | | 14/DEC/2025 |
| Hamza Ali Khan (2508-2022) | Team Member 3 | | 14/DEC/2025 |
| Dr. Umer Farooq | Supervisor | | 14/DEC/2025 |
| | Co-Supervisor | | |
| | Project Coordinator | | |

# Revision History

| Date | Version | Description | Author |
|---|---|---|---|
| 14/DEC/2025 | 1.0 | Initial Draft | Ismail Dad Khan |
| | | | |
| | | | |
| | | | |

# Definition of Terms, Acronyms, and Abbreviations

| Term | Description |
|---|---|
| API | Interface for communication between software components |
| Authentication | Verification of user identity |
| Authorization | Permission control for user actions |
| RBAC | Access control based on user roles |
| CRUD | Basic data operations |
| Session | Active user interaction period |
| Input Validation | Checking correctness of input data |
| Encryption | Securing data through encoding |
| Access Control | Restricting access to system resources |
| Data Integrity | Accuracy and consistency of data |
| System Availability | System uptime and accessibility |
| Scalability | Ability to handle increased load |
| Response Time | Time taken to respond to a request |
| Fault Tolerance | Ability to operate during failures |
| Audit Log | Record of system activities |
| UI | Interface for user interaction |
| Usability | Ease of system use |

# Table of Contents

## Table of Contents

# 1. Introduction

## 1.1 Purpose of Document

The purpose of this Software Requirements Specification (SRS) document is to define and describe all requirements for Dakhla – AI-Assisted University Admissions Platform. This document outlines the system's functional and non-functional aspects, ensuring that all stakeholders share a unified understanding of the software's objectives, scope, and expected behavior [2]. This document follows the university-approved SRS structure [5].

## 1.2 Intended Audience

This document is intended for:

- **Project Supervisor:** Dr. Umer Farooq, for reviewing and evaluating system design and requirements.
- **Development Team:** Muhammad Nauman Arif, Ismail Dad Khan, and Hamza Ali Khan, for guiding implementation and testing.
- **Evaluation Committee:** To assess completeness, feasibility, and alignment with FYP objectives.
- **End Users:** Students and university representatives who will interact with the system once deployed.

The purpose of this project is to develop a centralized, AI-assisted admissions platform that streamlines the entire university admission lifecycle for both students and institutions. Dakhla allows students to search and compare universities, receive personalized recommendations, interact with an intelligent chatbot (DakhlaBot), and apply directly to multiple universities using a single profile. On the institutional side, it enables universities to manage program listings, view filtered applicant profiles, and communicate with potential candidates, making the admissions process transparent, accessible, and efficient for all stakeholder [1].

**Problem Statement:**

University admissions in Pakistan are fragmented and inefficient. Students must visit multiple university websites to gather admission information, while universities lack a centralized system to manage and attract applicants. Dakhla bridges this gap through an AI-powered, unified platform that serves both students and educational institutions.

# 2. Overall System Description

## 2.1 Project Background

The university admissions process in Pakistan is currently decentralized and inconsistent. Students must visit multiple university websites to gather information about programs, eligibility criteria, deadlines, and fee structures. This lack of centralization results in confusion, wasted effort, and missed opportunities, particularly for students from underserved or rural areas.

Dakhla is designed to address these challenges by providing a unified, AI-assisted platform that serves both students and universities. It centralizes verified university data, automates comparisons, and enables direct applications through a single profile. For institutions, it offers dashboards to manage programs, filter applicants, and communicate with prospective students — making the admission process efficient and transparent for all stakeholders.

## 2.2 Problem Statement

The current university admission process in Pakistan is fragmented, time-consuming, and lacks automation. Students are required to manually collect and compare data from numerous sources, whereas universities often have limited tools for applicant filtering and engagement.
Dakhla resolves this issue by building a centralized and intelligent admission management system that simplifies exploration, comparison, and application submission, while empowering universities to manage applicants effectively.

## 2.3 Project Scope

The project covers the design and development of a web-based platform that centralizes university admission processes for both students and institutions.

- **Centralized University Directory:** Searchable and filterable listings of universities with programs, eligibility criteria, deadlines, and fees.
- **Student Account and Dashboard:** Each student will have a personalized dashboard to manage academic profiles, saved universities, application status, and recommendations.
- **AI-Based Recommendation System:** Suggests suitable universities based on academic background, preferred discipline, and location.

- **Dakhla Assessment Module:** An AI-generated entrance exam module with automated grading and anti-cheating features.
- **Intelligent Chatbot (DakhlaBot):** Provides instant responses and navigation support [3].
- **DirectApply Module:** Enables submission of one reusable application form to multiple universities.
- **University Dashboard:** Allows institutions to manage program listings, review applicant profiles, and communicate with students.
- **Automated Data Scraping:** Periodically updates university data through scheduled crawlers.

### 2.4 Not in Scope

The following functionalities are excluded from the current development cycle:

- Development of native mobile applications for Android or iOS.
- Integration of third-party payment gateways for fee processing.
- Real-time data integration with external government or HEC databases for credential verification.
- Offline or standalone desktop operation; the system will require an active internet connection.

### 2.5 Project Objectives

The objectives of the Dakhla platform are to:

- Design and implement a centralized, web-based admissions platform for Pakistani universities.
- Provide a personalized, AI-driven recommendation system to guide students toward suitable academic programs.
- To introduce an AI-driven assessment module enabling universities to conduct online entrance exams efficiently.
- To automate test creation, grading, and result generation using AI/ML techniques.
- To ensure fairness and integrity through built-in anti-cheating features.

- Develop an intuitive student dashboard for managing personal profiles, saved universities, and application status.
- Implement the DirectApply module to allow multi-institution applications through a unified form.
- Build a university-side dashboard for efficient listing management, applicant review, and communication.
- Integrate an AI-powered chatbot to offer real-time support and guidance to users. [3]
- Ensure scalability, usability, security, and data accuracy through modular architecture, automated data collection, and cloud deployment.

## 2.6 Stakeholders & Affected Groups

- **Students:** Primary users who search, compare, and apply to universities.
- **Universities / Admissions Offices:** Manage listings, review applications, and engage with applicants.
- **Project Supervisor:** Oversees project development and validates requirements.
- **Development Team:** Responsible for system design, coding, and testing.
- **Evaluation Committee:** Reviews project outcomes and documentation.

## 2.7 Operating Environment

- Frontend: React.js with Tailwind CSS (browser-based).
- Backend: Python (Flask / Django) hosted on cloud (Render / Vercel).
- Database: PostgreSQL for structured records and user data.
- AI Modules: OpenAI API or Dialogflow for chatbot and recommendations.
- Hosting & Deployment: Render, Vercel, or Firebase.
- Supported Browsers: Chrome, Edge, Firefox, Safari (latest versions).

## 2.8 System Constraints

- Continuous internet connectivity is required.
- Limited to web browsers; no native mobile app.
- Dependent on publicly available university data accuracy.
- Free-tier cloud limits may restrict concurrent API calls.
- University participation in DirectApply is voluntary during the pilot phase.

## 2.9 Assumptions & Dependencies

- Universities will provide or verify admission data.
- Students will create accurate and complete profiles.
- Scraping scripts will adapt to minor website structure changes.
- Platform will adhere to data-protection and privacy standards.
- Pilot testing will involve selected universities and student users.

# 3. External Interface Requirements

## 3.1 Hardware Interfaces

The Dakhla platform is entirely web-based and does not require any specialized hardware. The system interacts only with standard computing devices capable of internet access.

**Client Devices:**

- Any device capable of running a modern web browser (laptops, desktops, tablets, mobile phones).
- Minimum recommended specifications:
    - Dual-core processor
    - 4 GB RAM
    - Stable internet connection

**Server/Hosting Environment:**

- Cloud-hosted virtual machines or containers running the backend services.
- PostgreSQL database server for persistent storage.
- No dedicated physical hardware is required on the user side.

Since the system does not interact with specialized external hardware (e.g., printers, sensors), no physical hardware interfaces are required beyond standard device-browser interaction.

## 3.2 Software Interfaces

**Web Browsers**

The system supports modern web browsers, including:

- Google Chrome
- Mozilla Firefox
- Microsoft Edge
- Safari

Browsers must support JavaScript, cookies, and HTTPS.

**Backend Services**

- RESTful API layer for communication between frontend (React.js) and backend (Python/FastAPI/Django).
- JSON format used for all data exchange.
- JWT-based authentication for secure session handling.

**Database Interface**

- PostgreSQL relational database
- ORM layer (SQLAlchemy / Django ORM) for secure interaction
- Database stores:
    - User accounts
    - University details
    - Programs
    - Applications
    - Admin data
    - Activity logs

**AI and ML Interfaces**

- OpenAI API or Dialogflow API for chatbot responses [3]
- ML-based recommendation engine integrated into backend
- Secure API keys required with HTTPS encryption

**Web Scraping Tools**

- BeautifulSoup / Selenium / Scrapy for updating university data (if enabled)
- Periodic scripts triggered using cron/scheduled tasks

## 3.3 Communications Interfaces

**Network Communication Protocols**

- **HTTPS** for all requests between frontend and backend
- **TLS encryption** required for protecting sensitive data
- CORS configured for secure domain interactions

**API Communication**

- RESTful endpoints following standard HTTP verbs
- API supports:
    - User authentication
    - Search/filter operations
    - Application submissions
    - University portal actions
    - Admin actions

**Email/SMS (Optional Future Integration)**

- Used for:
    - Account activation
    - Password reset
    - Application notifications
- Not part of the current implementation, but included as an extendable interface.

# 4. System Functions / Functional Requirements

## *4.1* System Functions

This section defines all major system functions categorized by user type and system behavior. Each function is identified with attributes and constraints to ensure clarity and testability.

**Function Categories**

| Function Category | Meaning |
| --- | --- |
| **Evident** | User-facing functions such as searching universities, submitting applications, and reviewing applicants. |
| **Hidden** | Backend processes such as saving records, validating data, AI scoring, and logging activity. |
| **Frill** | Optional enhancements that do not affect core system functionality. |

### *4.1.1 Student-Side Functions*

| *Ref* | *Function* | *Category* | *Attribute* | *Details / Constraints* |
| --- | --- | --- | --- | --- |
| ***SF-01*** | *Student Registration* | *Evident* | *Authentication* | *Register using email, phone, or Google login* |
| ***SF-02*** | *Student Login* | *Evident* | *Access Control* | *JWT-based secure authentication* |
| ***SF-03*** | *Manage Student Profile* | *Evident* | *CRUD* | *Update academic, personal, and interest data* |
| ***SF-04*** | *Search Universities* | *Evident* | *Response Time* | *Results load within 3–5 seconds* |
| ***SF-05*** | *Filter Programs* | *Evident* | *Filtering Accuracy* | *Filters include city, fees, field, degree level* |
| ***SF-06*** | *Compare Universities* | *Evident* | *UI Behavior* | *Two–four universities in comparison view* |

| | | | | |
|---|---|---|---|---|
| ***SF-07*** | *AI Recommendations* | *Evident* | *Processing Time* | *Recommendations generated under 8 seconds* |
| ***SF-08*** | *Interact with Chatbot* | *Evident* | *AI Response* | *AI guidance backed by ML model* |
| ***SF-09*** | *Submit DirectApply Application* | *Evident* | *Data Validation* | *Unified application for multiple universities* |
| ***SF-10*** | *Track Application Status* | *Evident* | *Real-time Updates* | *Based on university actions* |
| ***SF-11*** | *Take Online Assessment* | *Evident* | *Assessmentt* | *Students can attempt AI-generated entrance exams* |

### *4.1.2 University-Side Functions*

| ***Ref*** | ***Function*** | ***Category*** | ***Attribute*** | ***Description / Constraints*** |
|---|---|---|---|---|
| *UF-01* | *University Registration Request* | *Evident* | *Verification* | *Must be approved by Admin* |
| *UF-02* | *University Login* | *Evident* | *Access Control* | *JWT-secured authentication* |
| *UF-03* | *Manage University Profile* | *Evident* | *CRUD* | *Update campus, departments, logo, contact details* |
| *UF-04* | *Add/Edit/Delete Programs* | *Evident* | *CRUD* | *Must follow backend schema validations* |
| *UF-05* | *View Applicants* | *Evident* | *Filtering* | *Filter by academic profile & exam score* |
| *UF-06* | *Review Applications* | *Evident* | *Decision* | *Accept / Reject / Shortlist* |
| *UF-07* | *Create Entrance Exams* | *Evident* | *Assessment* | *AI-generated exam creation system* |

| | | | | |
|---|---|---|---|---|
| *UF-08* | *View Exam Results* | *Evident* | *Assessment* | *Instant scorecards generated by AI grading* |

### *4.1.3 Assessment Module Functions*

| ***Ref*** | ***Function*** | ***Category*** | ***Attribute*** | ***Description / Constraints*** |
|---|---|---|---|---|
| *ASM-01* | *AI-Generated Exams* | *Evident* | *AI Generation* | *University defines exam subject, type, difficulty; system generates MCQs/short answers* |
| *ASM-02* | *Randomized Question Pools* | *Hidden* | *Anti-Cheating* | *Question order randomized per student* |
| *ASM-03* | *Exam Delivery* | *Evident* | *Timer* | *Full-screen, distraction-free exam environment* |
| *ASM-04* | *Automated Grading* | *Hidden* | *AI/ML* | *MCQs graded instantly; subjective answers evaluated via ML models* |
| *ASM-05* | *Suspicious Activity Detection* | *Hidden* | *Anti-Cheating* | *Detects tab-switching, idle time, repeated patterns* |
| *ASM-06* | *Publish Results* | *Evident* | *Output* | *Scorecards generated and sent to students* |

### *4.1.3 Admin-Side Functions*

#### *Super Admin*

| *Ref* | *Function* | *Category* | *Attribute* | *Details / Constraints* |
| --- | --- | --- | --- | --- |
| ***AD-01*** | *Create Admin Accounts* | *Evident* | *Security* | *Only Super Admin can access creation link* |
| ***AD-02*** | *Edit Admin Roles* | *Evident* | *Access Control* | *Set Super Admin / Regular Admin* |
| ***AD-03*** | *View List of Admins* | *Evident* | *Dashboard* | *Sort/filter admin entries* |
| ***AD-04*** | *Manage Universities* | *Evident* | *CRUD* | *Add/Delete/Edit university details* |
| ***AD-05*** | *Activate/Deactivate University Accounts* | *Evident* | *User Control* | *Immediate effect* |
| ***AD-06*** | *Activate/Deactivate Student Accounts* | *Evident* | *User Control* | *Account frozen after deactivation* |
| ***AD-07*** | *View Student & University Profiles* | *Evident* | *Access Control* | *Read-only mode* |
| ***AD-08*** | *Monitor System Logs* | *Hidden* | *Logging* | *Automated backend logs* |

#### *Regular Admin*

| *Ref* | *Function* | *Category* | *Attribute* | *Details / Constraints* |
| --- | --- | --- | --- | --- |
| ***AD-09*** | *Approve University Registrations* | *Evident* | *Validation* | *Basic admin role* |
| ***AD-10*** | *Manage University Listings* | *Evident* | *CRUD* | *But cannot modify admin roles* |
| ***AD-11*** | *View User Profiles* | *Evident* | *Access Control* | *Limited permissions* |

#### *4.1.4 System Functions (Backend + AI + DB)*

| *Ref* | *Function* | *Category* | *Attribute* | *Details / Constraints* |
| --- | --- | --- | --- | --- |
| ***SYS-01*** | *Recommendation Engine* | *Hidden* | *ML Model* | *Uses student profile & search history* |
| ***SYS-02*** | *AI Chatbot Integration* | *Hidden* | *NLP Processing* | *Uses OpenAI/Dialogflow* |
| ***SYS-03*** | *University Data Scraper* | *Hidden* | *Automation* | *Scheduled scraping for data updates* |
| ***SYS-04*** | *Secure Database Storage* | *Hidden* | *DB Integrity* | *ACID compliance* |
| ***SYS-05*** | *REST API Services* | *Hidden* | *API Security* | *Token-based authentication* |
| ***SYS-06*** | *Data Encryption* | *Hidden* | *Security* | *Sensitive data encrypted at rest & transit* |

## 4.2 Use Cases

### 4.2.1 List of Actors

| *Actor* | *Description* |
| --- | --- |
| ***Student*** | *Explores universities, applies, takes exams* |
| ***University Representative*** | *Manages programs, exams, reviews applicants* |
| ***Super Admin*** | *Full platform control* |
| ***Regular Admin*** | *Limited management features* |
| ***AI Chatbot*** | *Answers user queries* |
| ***Recommendation Engine*** | *Suggests suitable universities* |
| ***AI Assessment Engine*** | *Generates and grades exams* |

### 4.2.2 List of Use Cases

***Level 1 – Student Use Cases***

- *UC-01: Register Account*
- *UC-02: Login*
- *UC-03: Search Universities*
- *UC-04: Apply Filters*
- *UC-05: Compare Universities*
- *UC-06: View Recommendations*
- *UC-07: Interact with Chatbot*
- *UC-08: Submit DirectApply Form*
- *UC-09: Track Application Status*
- *UC-10: Take Entrance Exam*

***Level 2 – University Use Cases***

- *UC-11: University Login*
- *UC-12: Manage Programs*
- *UC-13: View Applicants*
- *UC-14: Review Applications*
- *UC-15: Create Exams*
- *UC-16: View Exam Results*

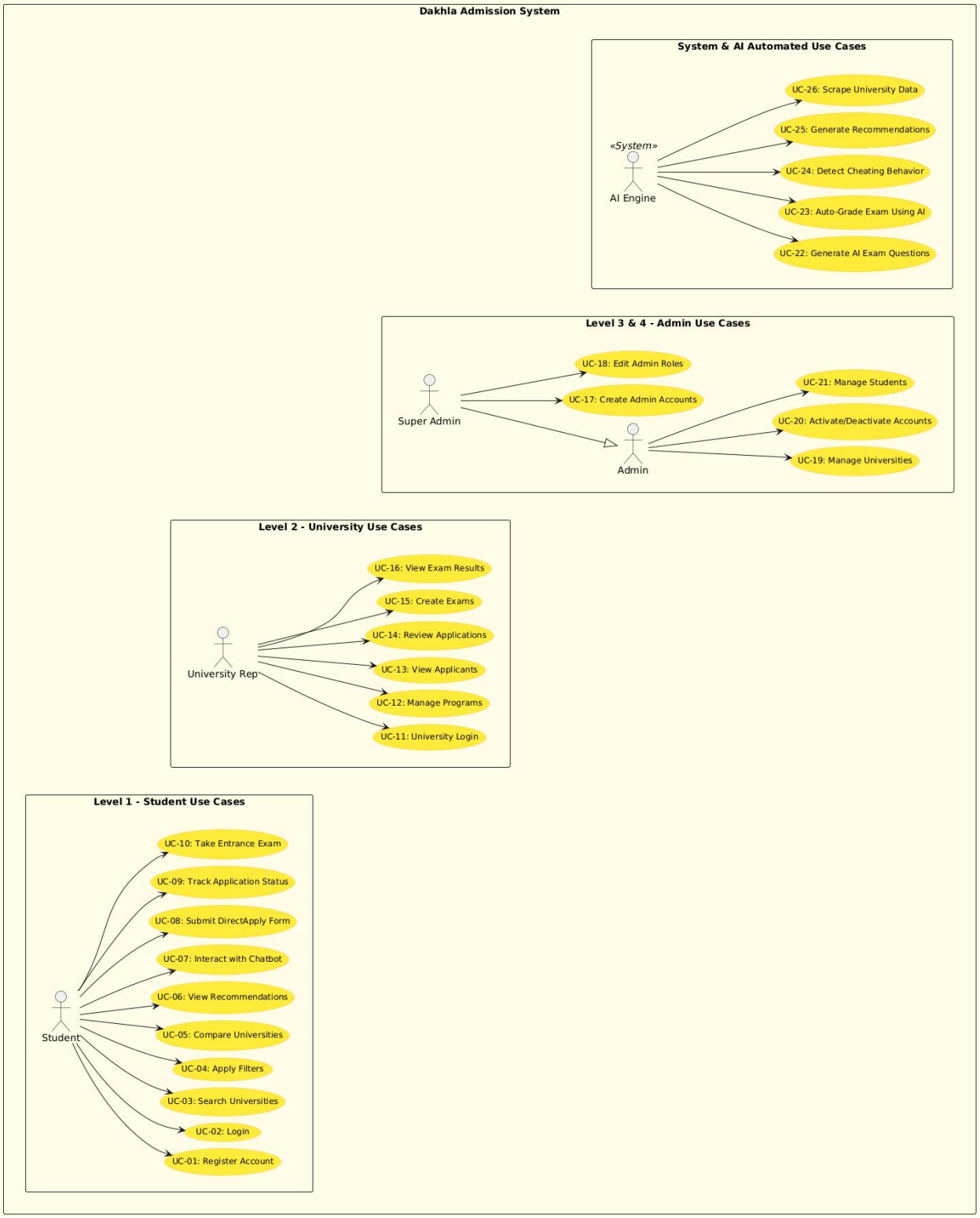
***Level 3 & 4 – Admin Use Cases***

- *UC-17: Create Admin Accounts*
- *UC-18: Edit Admin Roles*
- *UC-19: Manage Universities*
- *UC-20: Activate / Deactivate Accounts*
- *UC-21: Manage Students*

***System & AI Automated Use Cases***

- *UC-22: Generate AI Exam Questions*
- *UC-23: Auto-Grade Exam Using AI*
- *UC-24: Detect Cheating Behavior*
- *UC-25: Generate Recommendations*
- *UC-26: Scrape University Data*

### 4.2.3 Use Case Diagram [6]



### 4.2.4 Description of Use Cases

**UC-01: Student Registration**

**Section:** Main
**Name:** Student Registration
**Actors:** Student

**Purpose:**
To allow a new student to create an account on the Dakhla platform.

**Description:**
A student creates an account by providing basic personal and login information to access the system.

**Cross References:**
Functions: R1.1, R1.2

**Pre-Conditions:**
The student is not already registered in the system.

**Successful Post-Conditions:**
A new student account is created and stored in the system.

**Failure Post-Conditions:**
Account is not created and the student remains unregistered.

**Typical Course of Events**

| Actor Action | System Response |
| --- | --- |
| 1. Student selects “Register Account” | |
| 2. Student enters required details | |
| | 3. System validates entered information |
| | 4. System creates student account |
| | 5. System confirms successful registration |

Alternative Courses

- Step 2: Email already exists → Error message
- Step 2: Weak password → Prompt stronger password

**UC-02: Login**

**Section:** Main
**Name:** Login
**Actors:** Student

**Purpose:**
To authenticate a registered student.

**Description:**
A student logs into the system using valid credentials.

**Cross References:**
Functions: R1.3

**Pre-Conditions:**
Student account exists.

**Successful Post-Conditions:**
Student dashboard is displayed.

**Failure Post-Conditions:**
Access denied.

**Typical Course of Events**

| Actor Action | System Response |
| --- | --- |
| 1. Student enters credentials | |
| | 2. System verifies credentials |
| | 3. Dashboard displayed |

**Alternative Courses**

- Invalid credentials → Error message

**UC-03: Search Universities**

**Section:** Main
**Name:** Search Universities
**Actors:** Student

**Purpose:**
To allow students to find universities.

**Description:**
Students search universities based on keywords or criteria.

**Cross References:**
Functions: R2.1

**Pre-Conditions:**
Student is logged in.

**Successful Post-Conditions:**
Matching universities are displayed.

**Failure Post-Conditions:**
No results returned.

Typical Course of Events

| Actor Action | System Response |
| --- | --- |
| 1. Student enters search criteria | |
| | 2. System retrieves results |
| | 3. Results displayed |

**UC-04: Apply Filters**

**Section:** Main
**Name:** Apply Filters
**Actors:** Student

**Purpose:**
To refine university search results.

**Description:**
Student applies filters to narrow down results.

**Cross References:**
Functions: R2.2

**Pre-Conditions:**
Search results available.

**Successful Post-Conditions:**
Filtered results shown.

**Failure Post-Conditions:**
No matching universities.

**Typical Course of Events**

| Actor Action | System Response |
| --- | --- |
| 1. Student applies filters | |
| | 2. System updates results |

**UC-05: Compare Universities**

**Section:** Main
**Name:** Compare Universities
**Actors:** Student

**Purpose:**
To compare universities side by side.

**Description:**
Student selects universities to compare features.

**Cross References:**
Functions: R2.3

**Pre-Conditions:**
Universities selected.

**Successful Post-Conditions:**
Comparison displayed.

**Failure Post-Conditions:**
Comparison not generated.

**Typical Course of Events**

| Actor Action | System Response |
| --- | --- |
| 1. Student selects universities | |
| | 2. Comparison displayed |

**UC-06: View Recommendations**

**Section:** Main
**Name:** View Recommendations
**Actors:** Student

**Purpose:**
To receive personalized university recommendations.

**Description:**
System generates recommendations using student data.

**Cross References:**
Functions: R2.4

**Pre-Conditions:**
Student profile exists.

**Successful Post-Conditions:**
Recommendations displayed.

**Failure Post-Conditions:**
Recommendations unavailable.

**Typical Course of Events**

| Actor Action | System Response |
| --- | --- |

| Actor Action | System Response |
| --- | --- |
| 1. Student requests recommendations | |
| | 2. System generates recommendations |
| | 3. Results displayed |

**UC-07: Interact with Chatbot**

**Section:** Main
**Name:** Interact with Chatbot
**Actors:** Student

**Purpose:**
To assist students through AI chat.

**Description:**
Student interacts with chatbot for guidance.

**Cross References:**
Functions: R2.5

**Pre-Conditions:**
System online.

**Successful Post-Conditions:**
Response provided.

**Failure Post-Conditions:**
No response generated.

**Typical Course of Events**

| Actor Action | System Response |
| --- | --- |
| 1. Student asks a question | |
| | 2. Chatbot replies |

### UC-08: Submit DirectApply Form

**Section:** Main
**Name:** Submit DirectApply Form
**Actors:** Student

**Purpose:**
To submit applications to multiple universities.

**Description:**
Student submits a unified application form.

**Cross References:**
Functions: R3.1

**Pre-Conditions:**
Student logged in.

**Successful Post-Conditions:**
Application submitted.

**Failure Post-Conditions:**
Submission failed.

**Typical Course of Events**

| Actor Action | System Response |
|---|---|
| 1. Student opens DirectApply | |
| 2. Fills form | |
| 3. Selects universities | |
| | 4. System validates data |
| | 5. Application submitted |

### UC-09: Track Application Status

**Section:** Main
**Name:** Track Application Status
**Actors:** Student

**Purpose:**
To view application progress.

**Description:**
Student views current application status.

**Cross References:**
Functions: R3.2

**Pre-Conditions:**
Application exists.

**Successful Post-Conditions:**
Status displayed.

**Failure Post-Conditions:**
Status unavailable.

**UC-10: Take Entrance Exam**

**Section:** Main
**Name:** Take Entrance Exam
**Actors:** Student

**Purpose:**
To attempt university entrance exams.

**Description:**
Student attempts online exam.

**Cross References:**
Functions: R4.1

**Pre-Conditions:**
Exam assigned.

**Successful Post-Conditions:**
Exam submitted.

**Failure Post-Conditions:**
Exam incomplete.

**UC-11: University Login**

**Section:** Main
**Name:** University Login
**Actors:** University Representative

**Purpose:**
To allow university representatives to access the system.

**Description:**
A university representative logs into the system using valid credentials.

**Cross References:**
Functions: R5.1

**Pre-Conditions:**
University account exists and is active.

**Successful Post-Conditions:**
University dashboard is displayed.

**Failure Post-Conditions:**
Access is denied.

**Typical Course of Events**

| Actor Action | System Response |
|---|---|
| 1. Representative enters login credentials | |
| | 2. System verifies credentials |
| | 3. Dashboard displayed |

**Alternative Courses**

- Invalid credentials → Error message
- Account inactive → Access denied

**UC-12: Manage Programs**

**Section:** Main
**Name:** Manage Programs
**Actors:** University Representative

**Purpose:**
To manage academic programs offered by the university.

**Description:**
The university representative adds, updates, or removes academic programs.

**Cross References:**
Functions: R5.2

**Pre-Conditions:**
University representative is logged in.

**Successful Post-Conditions:**
Program information is updated.

**Failure Post-Conditions:**
Changes are not saved.

**Typical Course of Events**

| Actor Action | System Response |
|---|---|
| 1. Selects manage programs | |
| 2. Adds/Edits/Deletes program | |
| | 3. System validates data |
| | 4. Changes saved |

**UC-13: View Applicants**

**Section:** Main
**Name:** View Applicants
**Actors:** University Representative

**Purpose:**
To view students who applied to programs.

**Description:**
University representative views the list of applicants.

**Cross References:**
Functions: R5.3

**Pre-Conditions:**
Applications exist.

**Successful Post-Conditions:**
Applicant list displayed.

**Failure Post-Conditions:**
No applicants found.

**Typical Course of Events**

| Actor Action | System Response |
| --- | --- |
| 1. Opens applicant list | |
| | 2. System displays applicants |

**UC-14: Review Applications**

**Section:** Main
**Name:** Review Applications
**Actors:** University Representative

**Purpose:**
To evaluate student applications.

**Description:**
University representative reviews application details and records decisions.

**Cross References:**
Functions: R5.4

**Pre-Conditions:**
Applicant selected.

**Successful Post-Conditions:**
Application decision recorded.

**Failure Post-Conditions:**
Decision not saved.

**Typical Course of Events**

| Actor Action | System Response |
| --- | --- |
| 1. Selects applicant | |
| | 2. System displays application |
| 3. Accepts/Rejects/Shortlists | |
| | 4. Decision saved |
| | 5. Student notified |

**UC-15: Create Exams**

**Section:** Main
**Name:** Create Exams
**Actors:** University Representative

**Purpose:**
To create entrance exams for applicants.

**Description:**
University representative creates exams for admission screening.

**Cross References:**
Functions: R6.1

**Pre-Conditions:**
University logged in.

**Successful Post-Conditions:**
Exam created successfully.

**Failure Post-Conditions:**
Exam not created.

**Typical Course of Events**

| Actor Action | System Response |
| --- | --- |
| 1. Defines exam details | |
| | 2. System creates exam |

**UC-16: View Exam Results**

**Section:** Main
**Name:** View Exam Results
**Actors:** University Representative

**Purpose:**
To view results of entrance exams.

**Description:**
University representative reviews student exam results.

**Cross References:**
Functions: R6.2

**Pre-Conditions:**
Exam completed.

**Successful Post-Conditions:**
Results displayed.

**Failure Post-Conditions:**
Results unavailable.

**Typical Course of Events**

| Actor Action | System Response |
| --- | --- |
| 1. Opens exam results | |
| | 2. Results displayed |

**UC-17: Create Admin Accounts**

**Section:** Main
**Name:** Create Admin Accounts
**Actors:** Super Admin

**Purpose:**
To create new admin accounts for system management.

**Description:**
The Super Admin creates new admin accounts by providing required details.

**Cross References:**
Functions: R7.1

**Pre-Conditions:**
Super Admin is logged into the system.

**Successful Post-Conditions:**
New admin account is created.

**Failure Post-Conditions:**
Admin account is not created.

**Typical Course of Events**

| Actor Action | System Response |
| --- | --- |
| 1. Super Admin opens admin management | |
| 2. Enters admin details | |
| | 3. System validates information |
| | 4. Admin account created |
| | 5. Confirmation displayed |

Alternative Courses

- Invalid details → Error message

**UC-18: Edit Admin Roles**

**Section:** Main
**Name:** Edit Admin Roles
**Actors:** Super Admin

**Purpose:**
To manage admin roles and permissions.

**Description:**
Super Admin modifies roles assigned to admin accounts.

**Cross References:**
Functions: R7.2

**Pre-Conditions:**
Admin account exists.

**Successful Post-Conditions:**
Admin role updated.

**Failure Post-Conditions:**
Role update failed.

**Typical Course of Events**

| Actor Action | System Response |
| --- | --- |
| 1. Selects admin account | |
| 2. Chooses new role | |
| | 3. System updates role |
| | 4. Confirmation displayed |

Alternative Courses

- Unauthorized action → Access denied

**UC-19: Manage Universities**

**Section:** Main
**Name:** Manage Universities
**Actors:** Admin

**Purpose:**
To manage university records in the system.

**Description:**
Admin adds, edits, or removes university information.

**Cross References:**
Functions: R8.1

**Pre-Conditions:**
Admin is logged in.

**Successful Post-Conditions:**
University data updated.

**Failure Post-Conditions:**
Changes not saved.

**Typical Course of Events**

| Actor Action | System Response |
|---|---|
| 1. Admin opens university management | |
| 2. Adds/Edits/Deletes university | |
| | 3. System validates data |
| | 4. Changes saved |

Alternative Courses

- Invalid data → Error message

**UC-20: Activate / Deactivate Accounts**

**Section:** Main
**Name:** Activate / Deactivate Accounts
**Actors:** Admin

**Purpose:**
To control user access to the system.

**Description:**
Admin activates or deactivates user accounts.

**Cross References:**
Functions: R8.2

**Pre-Conditions:**
User account exists.

**Successful Post-Conditions:**
Account status updated.

**Failure Post-Conditions:**
Status unchanged.

**Typical Course of Events**

| Actor Action | System Response |
| --- | --- |
| 1. Admin selects user account | |
| 2. Chooses activate/deactivate | |
| | 3. System updates account status |

Alternative Courses

- Unauthorized action → Access denied

**UC-21: Manage Students**

**Section:** Main
**Name:** Manage Students
**Actors:** Admin

**Purpose:**
To manage student records.

**Description:**
Admin views and manages student profiles.

**Cross References:**
Functions: R8.3

**Pre-Conditions:**
Student exists.

**Successful Post-Conditions:**
Student data updated.

**Failure Post-Conditions:**
Update failed.

**Typical Course of Events**

| Actor Action | System Response |
|---|---|
| 1. Admin views student list | |
| 2. Selects student | |
| | 3. System displays profile |
| 4. Updates status/details | |
| | 5. Changes saved |

**UC-22: Generate AI Exam Questions**

**Section:** Main
**Name:** Generate AI Exam Questions
**Actors:** System (AI Engine)

**Purpose:**
To automatically generate exam questions.

**Description:**
The system generates exam questions using AI.

**Cross References:**
Functions: R9.1

**Pre-Conditions:**
Exam request exists.

**Successful Post-Conditions:**
Questions generated.

**Failure Post-Conditions:**
Generation failed.

**Typical Course of Events**

| Actor Action | System Response |
| --- | --- |
| 1. Exam request triggered | |
| | 2. AI generates questions |
| | 3. Questions stored |

**UC-23: Auto-Grade Exam Using AI**

**Section:** Main
**Name:** Auto-Grade Exam Using AI
**Actors:** System (AI Engine)

**Purpose:**
To evaluate exams automatically.

**Description:**
The system grades exams using AI.

**Cross References:**
Functions: R9.2

**Pre-Conditions:**
Exam submitted.

**Successful Post-Conditions:**
Scores generated.

**Failure Post-Conditions:**
Grading failed.

**Typical Course of Events**

| Actor Action | System Response |
| --- | --- |
| 1. Exam submission received | |
| | 2. AI evaluates responses |
| | 3. Scores saved |

**UC-24: Detect Cheating Behavior**

**Section:** Main
**Name:** Detect Cheating Behavior
**Actors:** System (AI Engine)

**Purpose:**
To identify suspicious behavior during exams.

**Description:**
The system monitors exams and flags anomalies.

**Cross References:**
Functions: R9.3

**Pre-Conditions:**
Exam in progress.

**Successful Post-Conditions:**
Suspicious activity flagged.

**Failure Post-Conditions:**
No detection performed.

**Typical Course of Events**

| Actor Action | System Response |
| --- | --- |
| 1. Exam session starts | |
| | 2. System monitors behavior |
| | 3. Suspicious activity flagged |

Alternative Courses

- No suspicious activity detected → Exam continues normally

**UC-25: Generate Recommendations**

**Section:** Main
**Name:** Generate Recommendations
**Actors:** System (AI Engine)

**Purpose:**
To provide personalized recommendations.

**Description:**
The system generates recommendations based on student data.

**Cross References:**
Functions: R9.4

**Pre-Conditions:**
Student profile exists.

**Successful Post-Conditions:**
Recommendations generated.

**Failure Post-Conditions:**
Recommendations unavailable.

**Typical Course of Events**

| Actor Action | System Response |
| --- | --- |

| Actor Action | System Response |
| --- | --- |
| 1. Recommendation request triggered | |
| | 2. System analyzes data |
| | 3. Recommendations displayed |

Alternative Courses

- Insufficient student data → System requests profile completion

**UC-26: Scrape University Data**

**Section:** Main
**Name:** Scrape University Data
**Actors:** System (AI Engine)

**Purpose:**
To collect university data automatically.

**Description:**
The system retrieves and stores verified university information.

**Cross References:**
Functions: R9.5

**Pre-Conditions:**
Source available.

**Successful Post-Conditions:**
Data stored.

**Failure Post-Conditions:**
Data retrieval failed.

**Typical Course of Events**

| Actor Action | System Response |
| --- | --- |

| Actor Action | System Response |
| --- | --- |
| 1. Data source accessed | |
| | 2. Data scraped |
| | 3. Data stored |

Alternative Courses

- Data source unavailable → Retry scheduled

# 5. Non - Functional Requirements

## 5.1 Performance Requirements

- The system shall support a minimum of **1,000 concurrent users**.
- Search operations shall return results within **2 seconds**.
- DirectApply submission shall complete within **3 seconds**.
- AI exam grading shall complete within **30 seconds**.
- Admin and University dashboards shall load within **5 seconds**.
- Chatbot responses shall be generated within **1–2 seconds** (excluding external API latency).

## 5.2 Safety Requirements

- The system shall perform daily automatic backups with a retention period of 30 days.
- The system shall implement fail-safe mechanisms (transaction rollback, auto-save).
- The exam module shall auto-submit answers during a crash or disconnection.
- The system shall require confirmation for deletion of critical data.
- The system shall prevent data loss during update, migration, or server failure. [4]

## 5.3 Security Requirements

- All communication shall occur over **HTTPS/TLS 1.2+**.
- Passwords shall be hashed with salting.
- The system shall use JWT-based authentication.
- Role-based access control (RBAC) shall separate Student, University, Admin, and Super Admin privileges.

- Admin/Super Admin shall activate or deactivate user accounts.
- The system shall protect against SQL Injection, CSRF, XSS, and unauthorized file access.
- Exam activity (tab switching, multiple attempts) shall be logged.

### 5.4 Reliability Requirements

- The system shall provide **99% uptime**.
- Critical operations shall be atomic and recoverable.
- The system shall automatically retry failed background tasks (e.g., notifications).
- The exam module shall ensure no data loss due to timeout or crash.

### 5.5 Usability Requirements

- The system shall support responsive UI for desktop and mobile.
- Form fields shall include validation and clear error prompts.
- Navigation to major features shall require no more than three steps.
- The interface shall meet accessibility guidelines for readability and clarity.

### 5.6 Supportability Requirements

- The system shall support modular architecture for easy updates.
- Logs shall be maintained for debugging and auditing purposes.
- The design shall allow integration with external AI services (OpenAI/Dialogflow).
- The database schema shall support scaling as universities and programs increase.

## 5.7 User Documentation

- The system shall provide onboarding instructions for students and universities.
- A help center or FAQ page shall be available within the platform.
- Admin-specific documentation shall include role permissions and panel usage.
- University representatives shall receive documentation for managing programs, exams, and applicants.

# 6. References

[1] Department of Computer Science, “Dakhla Project Proposal.” 2025. [Online]. Available: https://docs.google.com/document/d/1TazFQmYLS8rWed2cGglXHmbIfywwcqVFhLluT0uMNJU/edit

[2] IEEE, “IEEE Recommended Practice for Software Requirements Specifications,” no. IEEE Std 830-1998. 1998.

[3] OpenAI, “OpenAI API Documentation.” [Online]. Available: https://platform.openai.com/docs

[4] OWASP Foundation, “OWASP Web Application Security Risks.” [Online]. Available: https://owasp.org/

[5] Hamdard University, “Software Requirements Specification Template.” 2024. [Online]. Available: https://docs.google.com/document/d/1RdWuCHUJW0qwZCRPdxLk432F-j4l3eL7/edit

[6] Object Management Group, “Unified Modeling Language Specification.” [Online]. Available: https://www.omg.org/spec/UML/