Course Swap Platform Specification Document

Ayesha Eiman, Amnah Qureshi, Roozain Zehra, Ifrah Chisti

March 9, 2025

1 Introduction

1.1 Purpose

This document outlines the requirements, architecture, and functionality of the Course Swap Platform for Habib University. The platform aims to streamline the course exchange process by connecting students who want to swap or offer courses.

1.2 Scope

The platform will allow students to:

- Browse available course swap offers.
- Post course requests or offers.
- Receive algorithmically matched suggestions for potential swaps.
- Communicate securely with other students via provided contact details.

1.3 Definitions, Acronyms, and Abbreviations

- **HU**: Habib University
- **Firebase Auth**: Firebase Authentication
- **FCM**: Firebase Cloud Messaging

1.4 References

- University Data Protection Policies
- Firebase Documentation
- Node.js + Express.js Documentation

1.5 Overview

This document describes the system's functional and non-functional requirements, architecture, constraints, and acceptance criteria.

2 System Overview

2.1 Product Perspective

The Course Swap Platform is a web-based system hosted on Firebase Hosting. It interacts with MongoDB Atlas for data storage and Firebase Auth for authentication. Firebase Cloud Messaging is used to send push notifications.

2.2 Product Functions

Key platform features include:

- Login/Sign-up for authenticated access.
- A Catalog to browse course offers and requests.
- A Profile page for student details and contact information.
- Course Offering functionality to post requests or offers.
- An Algorithmic Matching System to suggest potential swaps.

2.3 User Characteristics

- **Primary Users:** Habib University students.
- **Skill Level:** Basic technical knowledge; minimal training required.

3 Functional Requirements

3.1 Login/Sign-up

- **FR-1:** The system shall authenticate students using their HU email via Firebase Auth.
- **FR-2:** The system shall automatically create a profile upon successful login.

3.2 Catalog

- **FR-3:** The system shall display all open course swap offers in a list format.
- **FR-4:** Clicking the "Interested" button shall notify the original poster.

3.3 Profile Management

- **FR-5:** The system shall allow users to view each other's university email addresses.
- **FR-6:** Users may add an optional phone number for contact purposes.

3.4 Course Offering and Requests

- **FR-7:** Students shall be able to create course swap offers or requests.
- **FR-8:** The system shall allow students to request specific time slots or courses fulfilling designated requirements.

3.5 Algorithmic Matching

- **FR-9:** The system shall suggest potential matches based on course codes, time slots, and elective types.

4 Non-Functional Requirements

4.1 Performance

- The platform shall handle **100 concurrent users** with minimal lag.
- Response time for queries shall not exceed **2 seconds**.
- The system must efficiently process up to **1000 swap requests per day**.

4.2 Usability

- The UI shall be **intuitive and accessible** for non-technical users.
- User should complete a swap-request in **under 5 clicks**.

4.3 Security

- Only authenticated students shall access the platform.
- All personal data shall be **encrypted**.
- The system shall implement **role-based access control**.

4.4 Availability

- The platform must maintain **99.5% uptime**.
- Maintenance downtime shall be communicated **24 hours in advance**.

4.5 Maintainability & Scalability

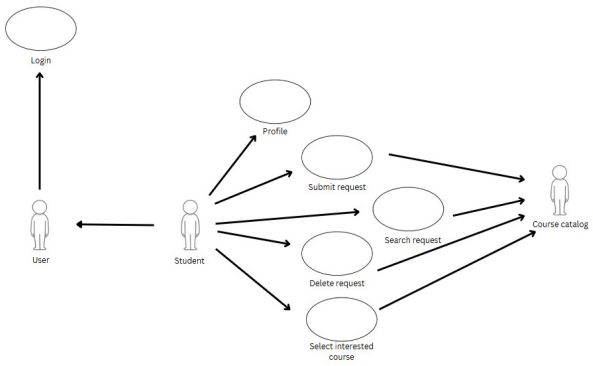
- The codebase shall follow a **modular design** for easy updates.
- The system should be scalable for **future integrations**.

5 System Architecture

5.1 Technology Stack

- **Front End:** CSS + Bootstrap for UI design.
- **Back End:** Node.js + Express.js handles API endpoints and business logic.
- **Database:** MongoDB Atlas stores course swap requests and student data.
- **Authentication:** Firebase Auth secures logins.
- **Notifications:** Firebase Cloud Messaging alerts users of matches.
- **Hosting:** Firebase Hosting delivers the web platform.

5.2 User case architecture



6 Interface Requirements

6.1 User Interface Wireframes

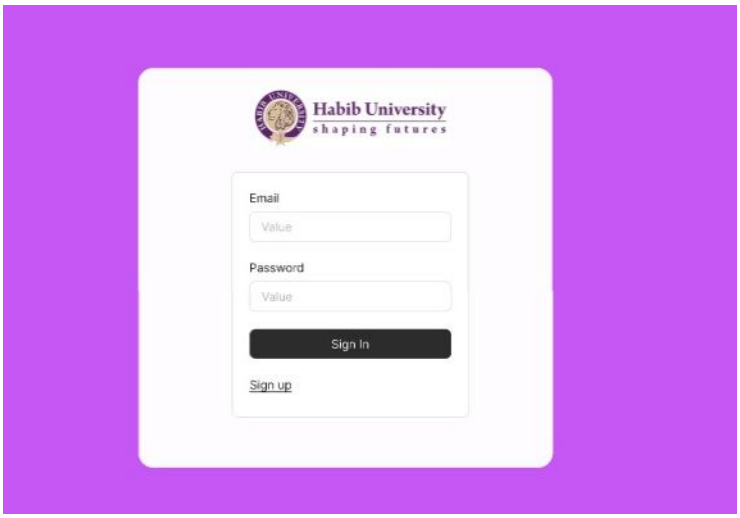


Figure 1: Login/Sign-up Page - Users can sign in using their HU credentials.

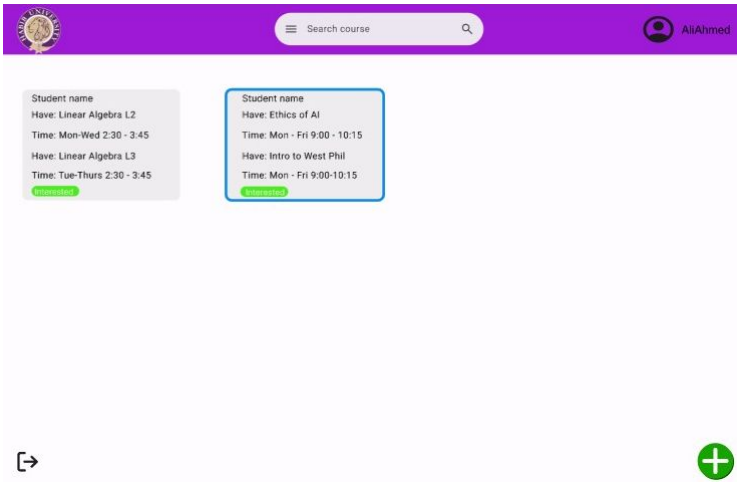


Figure 2: Course Catalog Page - Displays available course swap offers.

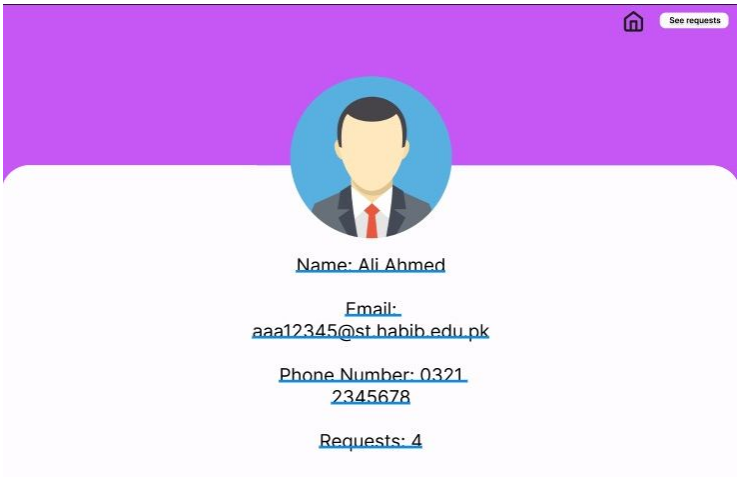


Figure 3: Profile Page - Shows user details and contact information.

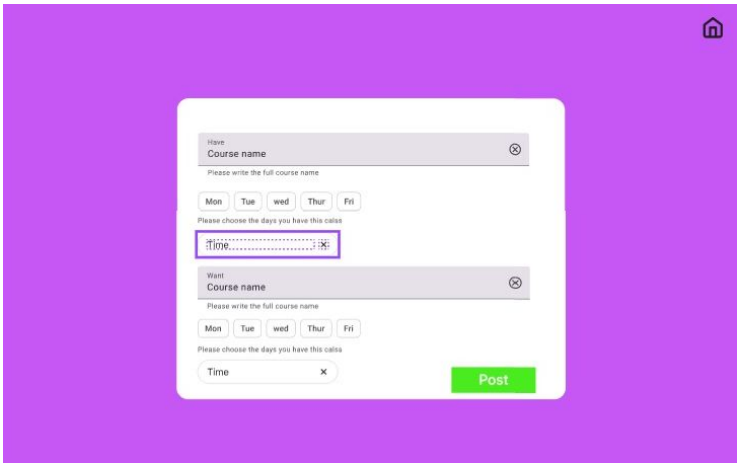


Figure 4: Add Swap Request Page - Users can submit new swap requests.

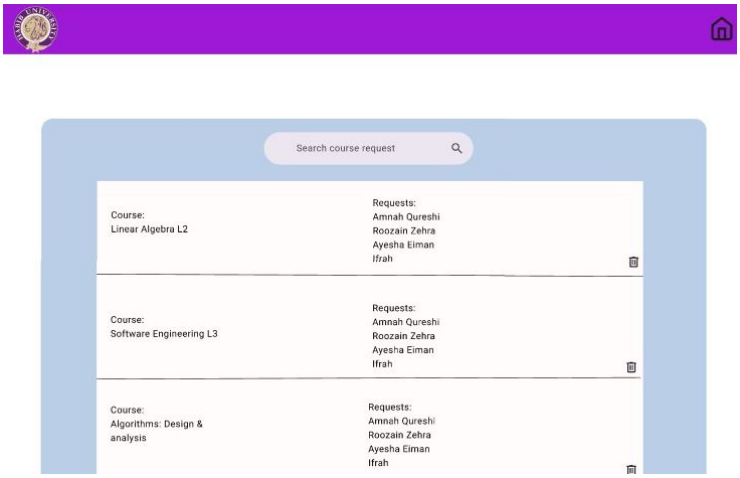


Figure 5: Your Request Catalog Page - Displays user's submitted requests.

6.2 Authentication API

6.2.1 User Registration

Endpoint: POST /api/auth/register
Description: Registers a new user using their university email.
Request:

```
{
    "email": "student@university.edu",
    "password": "securepassword",
    "fullName": "John Doe"
}
```

Response:

```
{
    "message": "User registered successfully",
    "userId": "user123"
}
```

6.2.2 User Login

Endpoint: POST /api/auth/login
Description: Logs in a user and returns an authentication token.
Request:

```
{
    "email": "student@university.edu",
    "password": "securepassword"
}
```

Response:

```
{
    "token": "jwt-token",
    "userId": "user123"
}
```

6.3 Course Swap API

6.3.1 Create a Swap Request

Endpoint: POST /api/swaps/create
Description: Allows a user to input a swap request.
Request:

```
{
    "userId": "user123",
    "offeringCourse": "CS101",
```

```
    "desiredCourse": "CS201",
    "semester": "Spring 2025"
}
```

Response:

```
{
    "message": "Swap request created",
    "swapId": "swap456"
}
```

6.3.2 Get All Swap Requests

Endpoint: GET /api/swaps
Description: Retrieves all swap requests for browsing.
Response:

```
{
    "swapId": "swap456",
    "userId": "user123",
    "offeringCourse": "CS101",
    "desiredCourse": "CS201",
    "semester": "Spring 2025"
}
```

6.3.3 Get Swap Requests by User

Endpoint: GET /api/swaps/user/:userId
Description: Fetches swap requests for a specific user.
Response:

```
[
    {
        "swapId": "swap456",
        "offeringCourse": "CS101",
        "desiredCourse": "CS201",
        "semester": "Spring 2025"
    }
]
```

6.4 Matching API

6.4.1 Finding Matching Swaps

Endpoint: GET /api/matches/:userId
Description: Find potential swap matches for the user.
Response:

```
[
    {
        "matchId": "match789",
        "swapId": "swap456",
        "matchingUserId": "user456",
        "offeringCourse": "CS201",
        "desiredCourse": "CS101"
    }
]
```

6.5 Notification API

6.5.1 Send Match Notifications

Endpoint: POST /api/notifications/send
Description: Sends a notification when a match is found.
Request:

```
{
    "userId": "user123",
    "message": "A match has been found for your course swap request!"
}
```

Response:

```
{
    "message": "Notification sent"
}
```

7 Data Requirements

- All course swap offers shall be stored in **MongoDB Atlas.**
- User information (emails, profile data) shall be stored securely with **Firebase Auth.**

8 Constraints

- **Real-Time Matching:** May require batch processing during peak traffic for scalability.
- **Complex Queries:** Matching swaps involving multiple conditions may require additional optimization.
- **Node.js Limitations:** CPU-heavy operations may slow down performance.

- **Firebase Cloud Messaging:** Limited to push notifications, requiring students to use third-party apps (e.g., Teams, WhatsApp) for real-time chat.

9 Assumptions and Dependencies

- The system assumes all users have valid **Habib University credentials.**
- Firebase and MongoDB services must remain active for platform functionality.
- Users must have an **active internet connection** to access the platform.

10 Acceptance Criteria

- **Login System:** Students should successfully authenticate via Firebase Auth.
- **Catalog Functionality:** Students should view available swap offers and submit new requests.
- **Matching Algorithm:** The algorithm should suggest matches with **85% accuracy** based on user-defined criteria.

11 Appendix

- Sample course catalog layout.
- Example Firebase Auth rules.
- Sample MongoDB document structure.