**CMPS 350 Project Phase 1 – Report**

**Education Platform**

**(10% of the course grade)**

**The report must be submitted in Word format only**

|  |  |
| --- | --- |
| **Group Members** | Hateim Elagha (202104575)  Ahmad Almashhadani (202203014)  Obada Alrefai (202110207)  **Emails:** [ha2104575@student.qu.edu.qa](mailto:ha2104575@student.qu.edu.qa); [aa2203014@student.qu.edu.qa](mailto:aa2203014@student.qu.edu.qa) ; <oa2110207@student.qu.edu.qa>; |
| **GitHub link** | Give a public link to you code (It is not acceptable to send codes by email) |

**Grades :**

**The student fills only the “Implementation Percentage”: Please specify either: *Working (completed x%)*, *Not Working (completed x%)* or *Not done*.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Criteria** | **Points** | **Implementation Percentage** | **Implementation Quality** | **Your Grade** |
| Design and implement the app Web UI and navigation using HTML, CSS and JavaScript. Including designing the App Web UI and navigation. | 50 | Working (completed 100%) |  |  |
| Design and implement the Web API and data access repositories to read/write the app data JSON files. | 30 | Working (completed 100%) |  |  |
| Application modeling (e.g. UML diagrams) to explain the data entities and the functionalities | 5 | Working (completed 100%) |  |  |
| Testing documentation using screen shots illustrating the testing results. | 5 | Working (completed 100%) |  |  |
| Team work quality. Contributions of team members - All members should collaborate and contribute equally to the project. | 5 | Working (completed 100%) |  |  |
| Project report – description of the implemented app, what is implemented, what is missed .. | 5 | Working (completed 100%) |  |  |
| **Total** | 100 |  |  |  |
| **Plagiarism, outsourcing, free riders** | -100 |  |  |  |
| **Delivery behind the deadline** | -5 |  |  |  |

**Important remark: In case of copying and/or plagiarism or not being able to explain or answer questions about the implementation, you lose the whole grade.**

**\* Criteria for grading the functionality:**

- The functionality is working: you get 70% of the assigned grade.

- The functionality is not working: you lose 40% of assigned grade.

- The functionality is not implemented: you get 0.

- The remaining grade in all cases from above **is assigned to the quality of the implementation**,

- The grades are distributed on the various use cases, when the design/implementation is partial, you get only the grades of designed/implemented use cases.

Code quality criteria, include:

- Use of meaningful identifiers for variables and functions (e.g. using JavaScript naming conventions)

- Pages are responsive

- Clean code: simple and concise code, no redundancy

- Clean implementation without unnecessary files/code

- Use of comments where necessary

- Proper code formatting and indentation.

**You lose marks** for code duplication, poor/inefficient coding practices, poor naming of identifiers, unclean/untidy submission, and unnecessary complex/poor user interface design.

**Important Remark**:

**[Grades: 100-85]:** Will be given only to **fully functional application** with **all the quality criteria cited above met** and the project has excellent **design for the various functionalities**. **The report is professional**.

**[Grades: 85-80]:** Will be given only **to functional application** **with most of all the quality criteria cited above met** and the project has good design for the various functionalities. **The report is professional**.

**[Grades: 80-75]:** 80% of the application functionalities are functional. The project respects partially the quality criteria. **The report is professional** but misses some information.

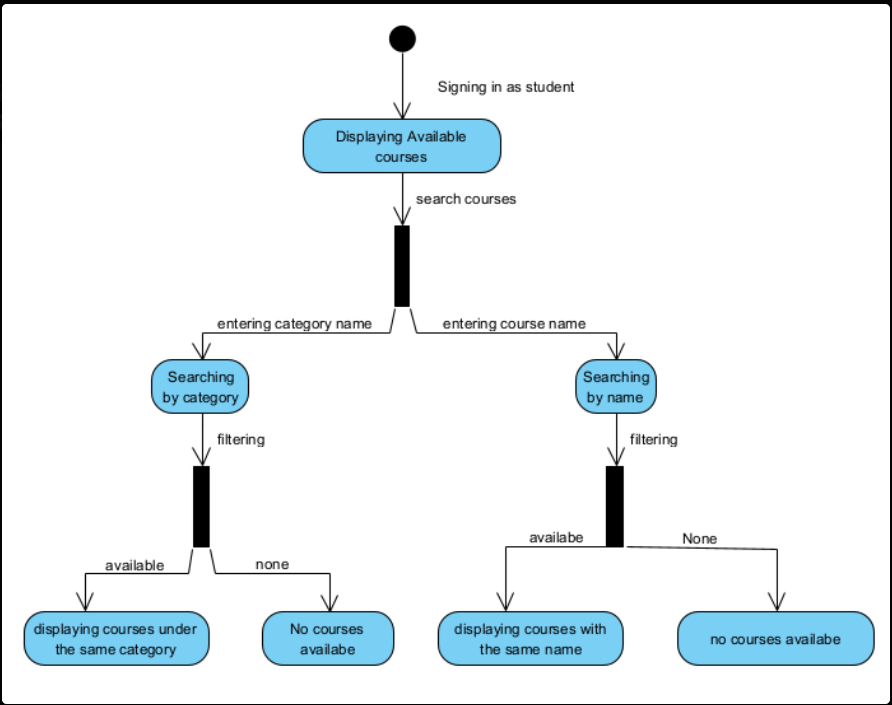
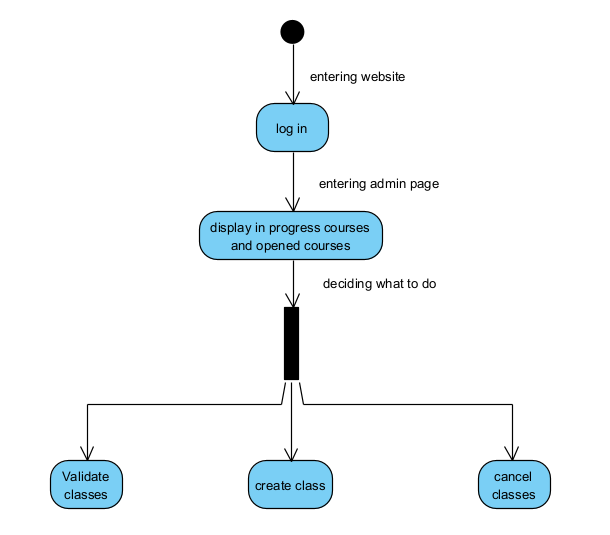
The grades are not negotiable. We expect that only a small portion (around 15%) of the class will be able to meet the criteria for the grades **[100-85]. You should work hard to and demonstrate the merits of your application to earn those grades.+**

# Description of your proposed platform

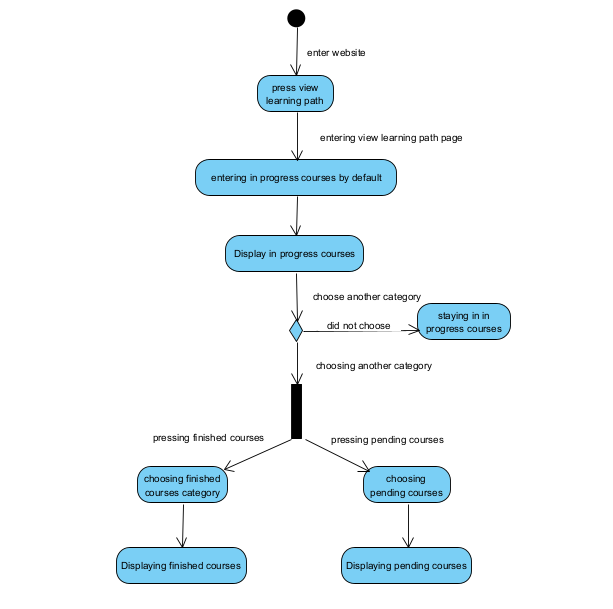
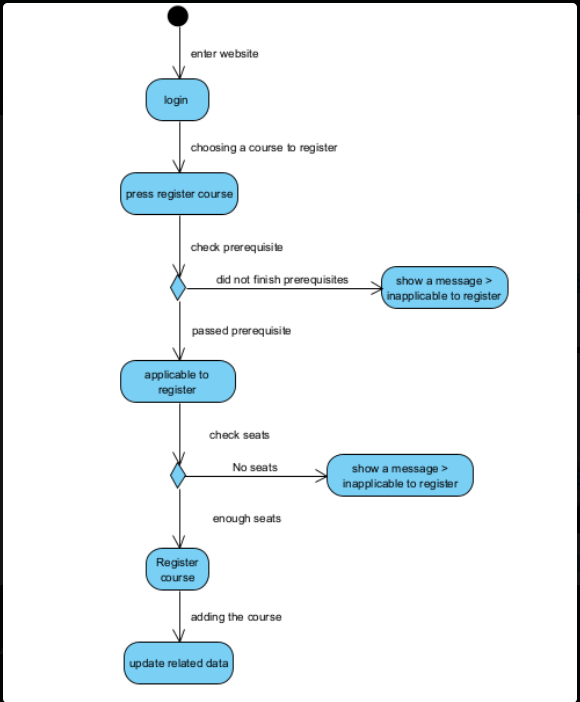
Give an overview of how your application works

The Qatar University Student Management Application is a web-based platform built with HTML/CSS/JavaScript using JSON files for data storage. It supports three user roles: **students** (search/register for courses, view learning paths with completed/in-progress/pending courses), **instructors** (submit grades), and **administrators** (create/validate courses, manage registrations). Key features include role-based login authentication, dynamic course search/filtering, prerequisite validation for registrations, and responsive UI designs for desktop/mobile. Students can register only for "pending" courses meeting prerequisites, while admins approve registrations and manage course status. Instructors update grades via forms, which reflect in student profiles. The application enforces clean code practices, modular APIs for data handling, and rigorous testing for bug finding and fixing.

# Use case diagram

A diagram of a course

AI-generated content may be incorrect.A diagram of a username

AI-generated content may be incorrect.

# Entities class diagram

Describe your data as a class diagram or Entity Association diagram

# Web API class

Local storage:

A screenshot of a computer

AI-generated content may be incorrect.

List all the methods (functions) to query your data entities

# Implementation

# Implemented use-cases

# Use case 1:

**Description:**

The user will insert his username and password, then the system has to check its JSON files to know whether the inserted data exists or not. If it exists and belongs to a student, it will navigate the user to the Available Course page. If it exists and belongs to an instructor, it will navigate the user to the Instructor Current Courses. If it exists and belongs to an administrator, it will navigate the user to the Validate Course page. If it doesn’t exist, it will alert a message that indicates invalid data.

**Non-functional requirement:**

1. Usability: easy to use and provide error messages
2. Security: Not everyone can log in, only admins, instructors and students can
3. Portability: a different layout is provided for each PC screen and Mobile screen

# Use case 2:

**Description:**

By default, all offered courses and classes will be displayed to the student. The student can search for a particular course or a class using either course name, course number, or course category

**Non-functional requirement:**

1. Usability: simple interface with labeled button and readable text
2. Portability: a different layout is provided for each PC screen and Mobile screen

# Use case 3:

**Description:**

A student can register for a course with the desired instructor if he/she has passed all the prerequisites. If a student tried to register for a class while he/she didn’t pass all its perquisites, or register a class that belongs to another campus, or register a class that has 0 available seats, it will alert a message describing the issue.

**Non-functional requirement:**

1. Usability: simple interface with labeled button and readable text
2. Reliability: the system prevents invalid registration
3. Security: require student login to be able to register for a course
4. Portability: a different layout is provided for each PC screen and Mobile screen

# Use case 4:

**Description:**

**Non-functional requirement:**

# Use case 5:

**Description:**

Administrators log in to view courses/classes categorized by status (e.g., "open," "in-progress") and manage their lifecycle. They validate courses with sufficient registrations, cancel undersubscribed ones, and create new courses/classes with details like prerequisites, instructors, and sections.

**Non-functional requirement:**

1. Security: ensure only admins can validate/create courses.
2. Performance: changes are implemented in local arrays (used to store displayed data) before local storage, to ensure immediate visual changes.
3. Responsiveness: Admin dashboard supports mobile and desktop layouts (e.g., collapsible tables).

# Use case 6:

**Description:**

Instructors access their assigned classes and submit final grades for students. Grades are saved to student profiles in students.json, updating their learning path and GPA.

**Non-functional requirement:**

* + 1. Security: Restrict grade submission to instructors assigned to the specific class section.
    2. Data Accuracy: Input validation for grades(min,max).
    3. Responsiveness: Grade submission forms adapt to mobile screens.

# Unimplemented use-cases and not functioning parts

All use cases are implemented and work as it’s required

# Testing

# Use case 1

Login page:

A screenshot of a computer

AI-generated content may be incorrect.

Student username login:

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Instructor username login:

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

admin username login:

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Mobile screen:

A screenshot of a login screen

AI-generated content may be incorrect.

# Use case 2

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Searching using Course name:

A screenshot of a computer

AI-generated content may be incorrect.

Searching using Course Number:

A screenshot of a computer

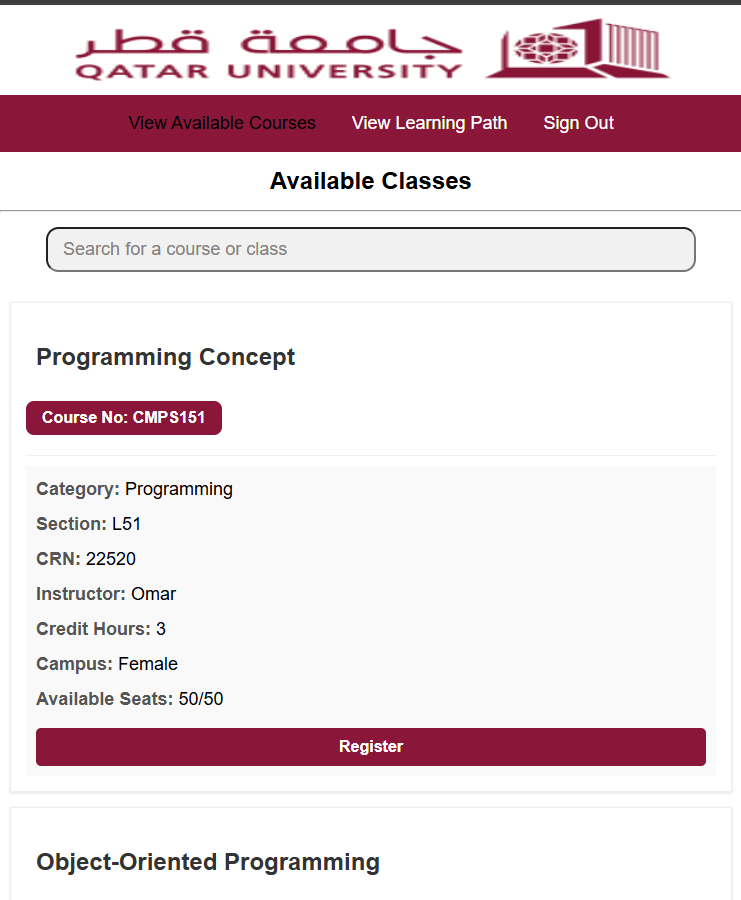
AI-generated content may be incorrect.

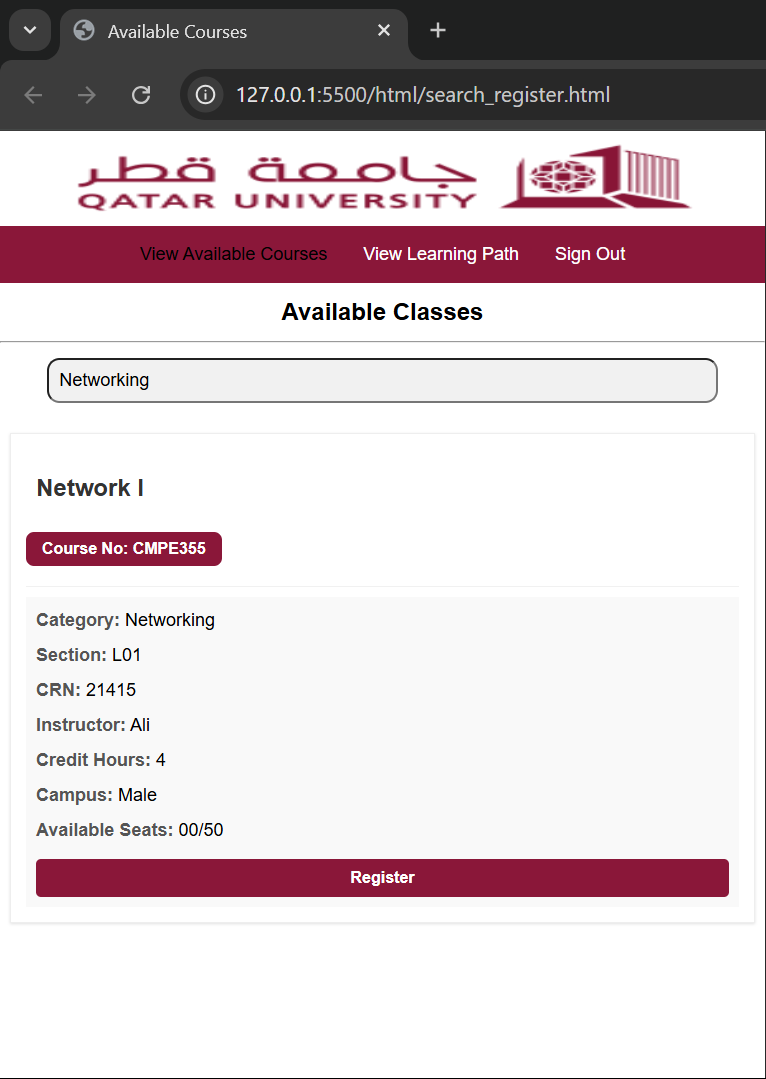
Searching using Course category:

A screenshot of a computer

AI-generated content may be incorrect.

Mobile View:





# Use case 3

Before Registering:

A screenshot of a computer

AI-generated content may be incorrect.

After registering:

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Trying to register for the same course:

A screenshot of a computer

AI-generated content may be incorrect.

Trying to register for another campus class:

A screenshot of a computer

AI-generated content may be incorrect.

Trying to register for a course without finishing its prerequisite:

A screenshot of a computer

AI-generated content may be incorrect.

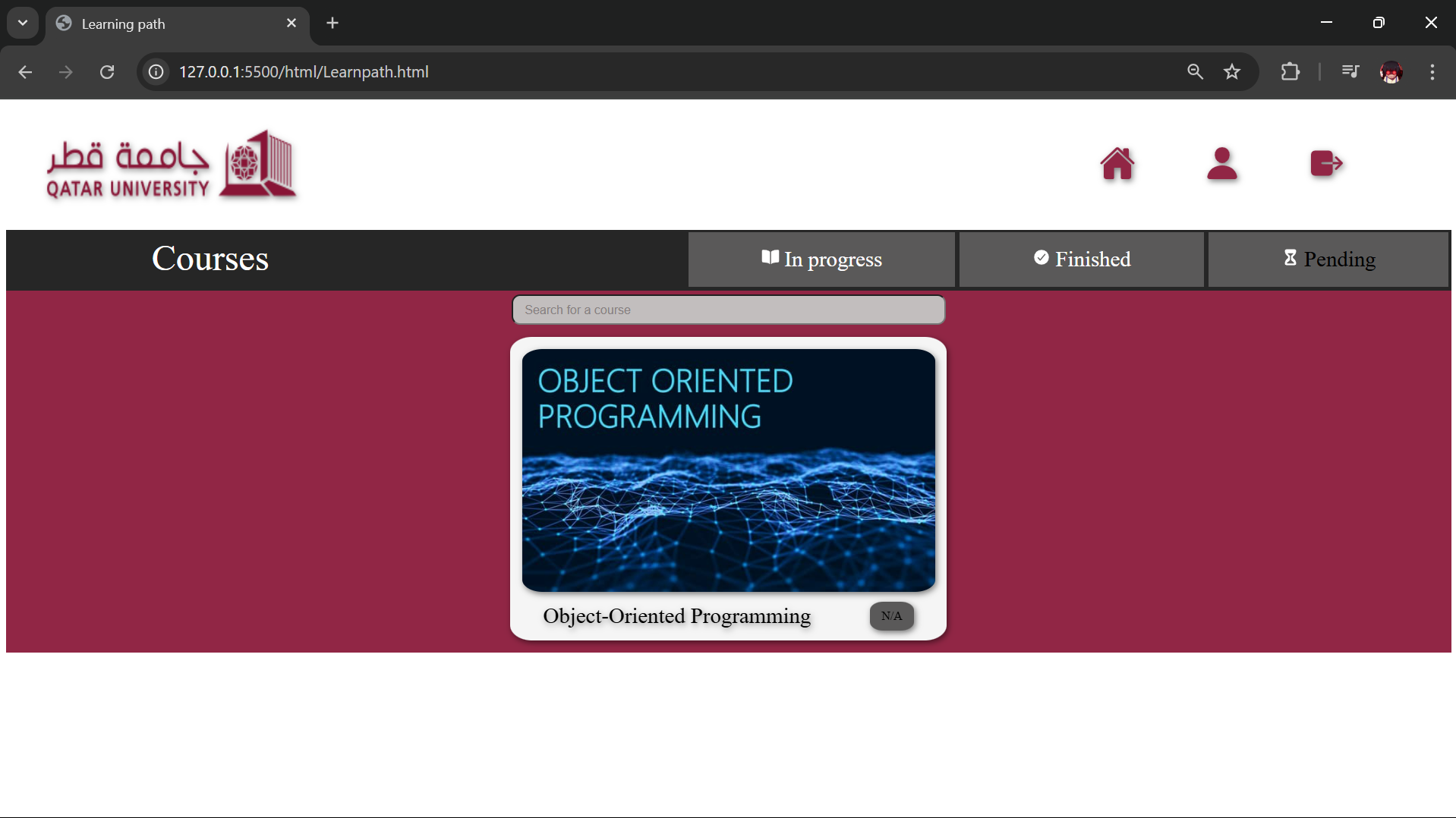
Trying to register for a full class:

A screenshot of a computer

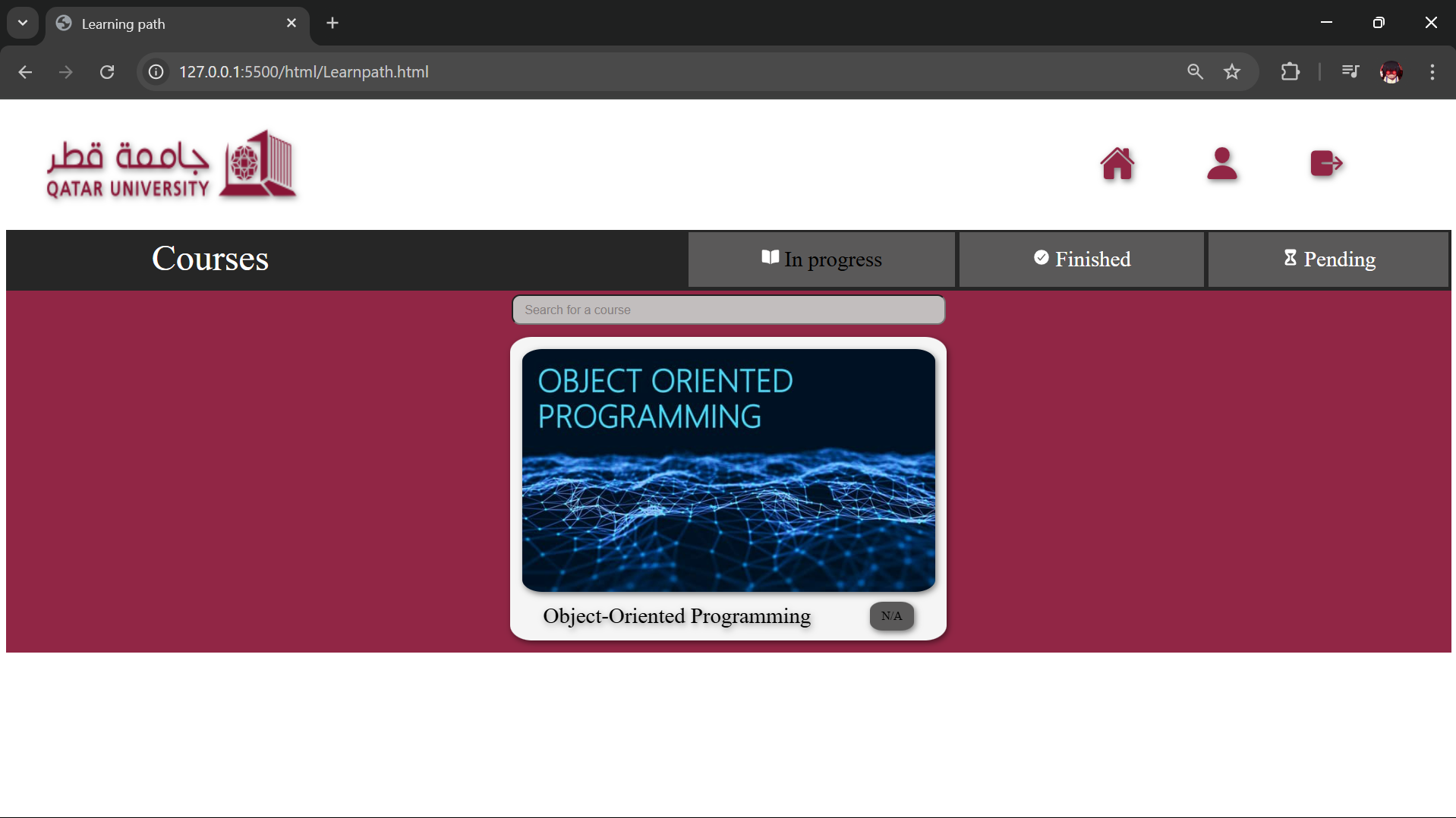
AI-generated content may be incorrect.

# Use case 4

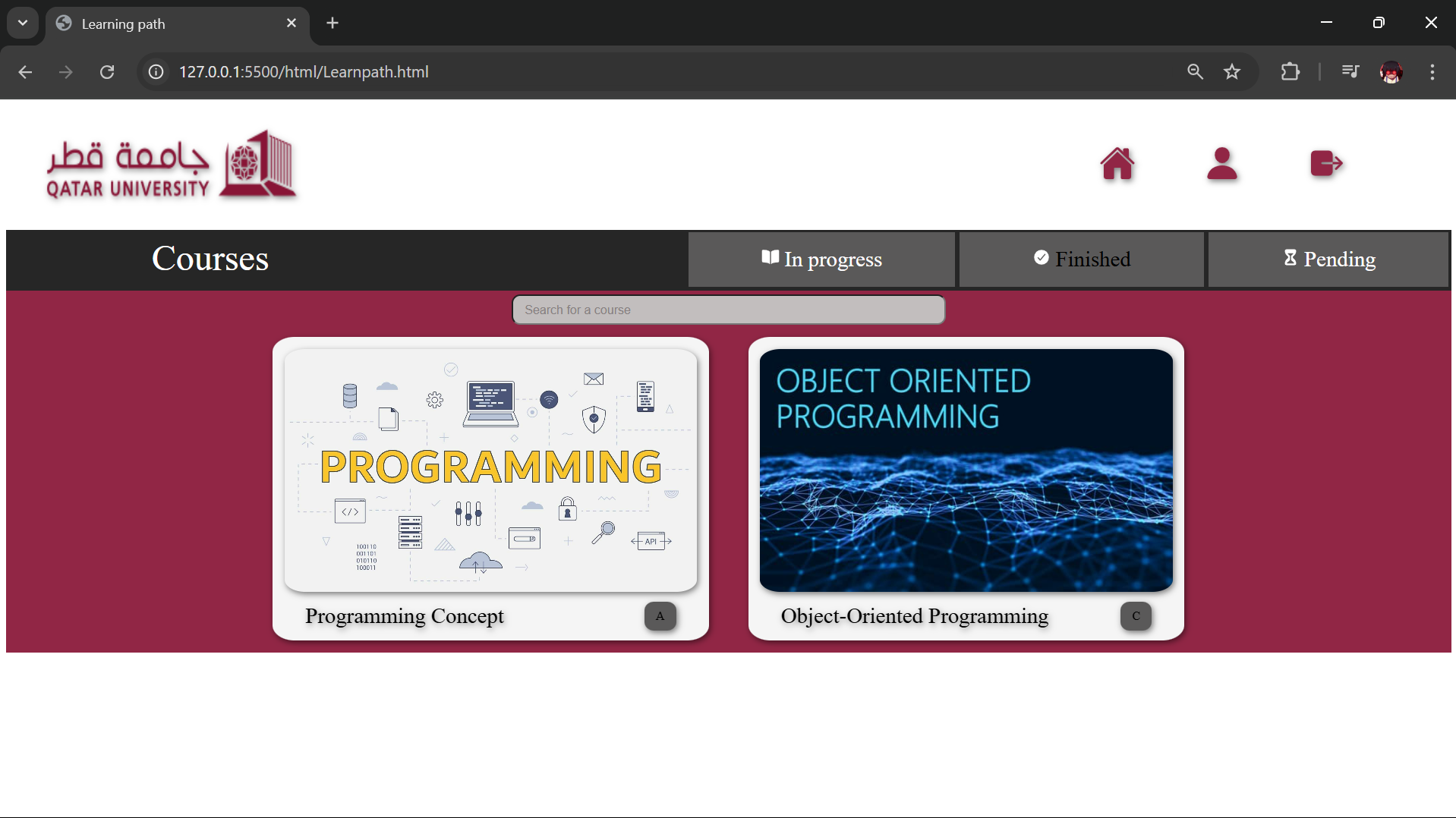
Registering open courses:



After the admin validate the course:



After the instructor submits the grade:



Mobile screen:



# Use case 5

admin view Showing courses and classes:

A screenshot of a computer

AI-generated content may be incorrect.Adding course:

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.Course added:

A screenshot of a computer

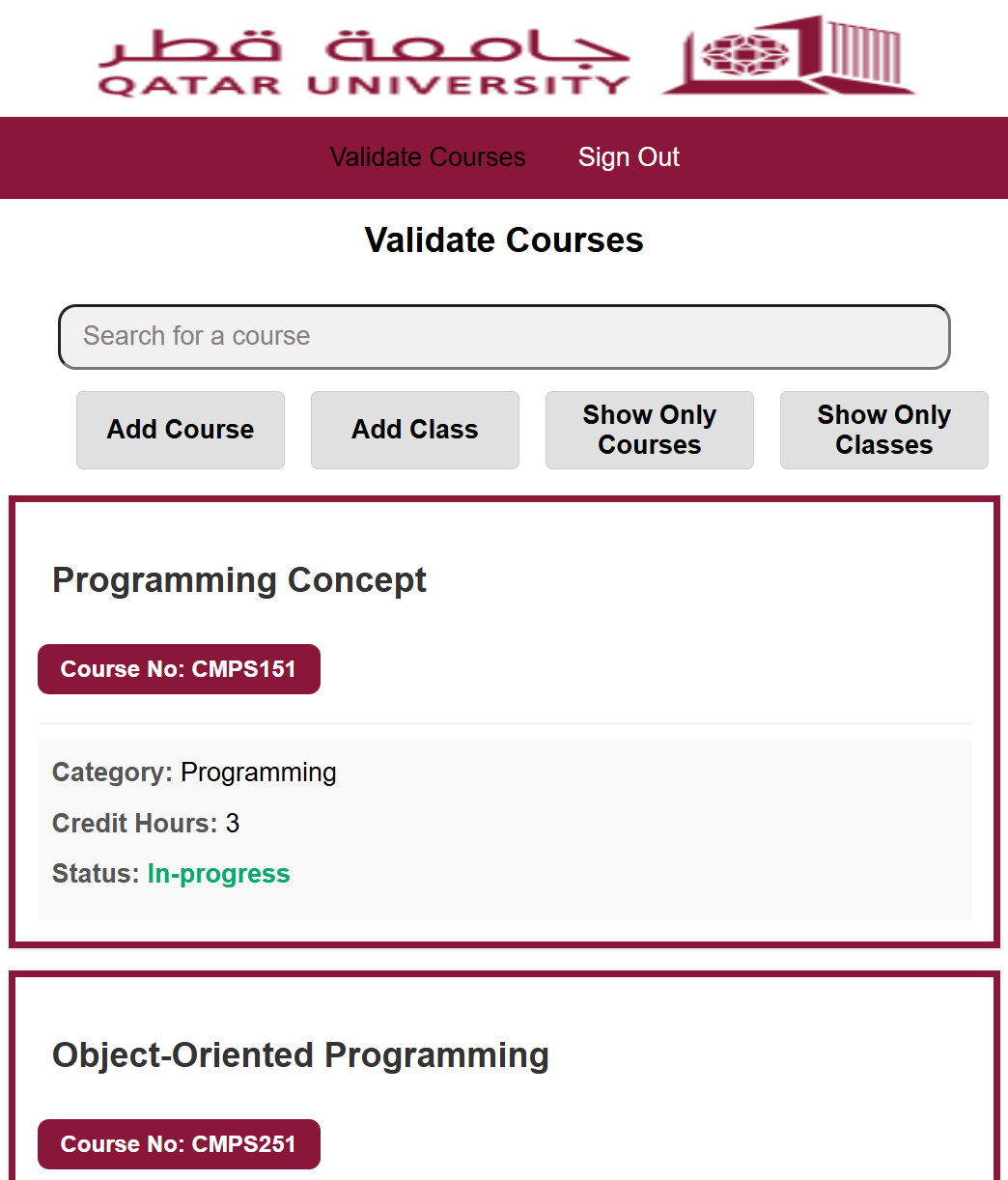
AI-generated content may be incorrect.Adding class:

A screenshot of a computer

AI-generated content may be incorrect.Class added:

A screenshot of a computer

AI-generated content may be incorrect.Validate class and deleted course:

Mobile view:

# Use case 6

Screen when instructor logs on:



A screenshot of a computer

AI-generated content may be incorrect.grading screen of the selected class, having grades entered:

A screenshot of a computer

AI-generated content may be incorrect.screen after grade submission:

Mobile grading screen:

# A screen shot of a computer AI-generated content may be incorrect.Discussion of the project contribution of each team member

|  |  |
| --- | --- |
| **Student name** | **Student contributions** |
| Hateim Elagha | 33.3% (Use case 2,3) |
| Ahmad Almashhadani | 33.3% (use case 1,4) |
| Obada Alrefai | 33.3% (use case 5,6) |
|  |  |
|  |  |