**CMPS 350 Project Phase 2 – Report**

**Education Platform**

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

|  |  |
| --- | --- |
| **Group Members** | 1. Alreem Alhaddad (202204202) 2. Alreem Zainal (202103014) 3. Bashayer Alnasser (202001583) 4. Leina Elsheiri (202005197)   **Emails:** [aa2204202@student.qu.edu.qa](mailto:aa2204202@student.qu.edu.qa) ; [az2103014@student.qu.edu.qa](mailto:az2103014@student.qu.edu.qa) ; [ba2001583@student.qu.edu.qa](mailto:ba2001583@student.qu.edu.qa); [le2005197@student.qu.edu.qa](mailto:le2005197@student.qu.edu.qa) |
| **GitHub link** | <https://github.com/Bashayer-qu/Project-Phase2-Education-Platform.git> |

**Grades :**

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Criteria** | **%** | **Functionality**\* | **Quality of the implementation** | **Grade** |
| Design and implement the Data Model. | 10 |  |  |  |
| Init DB: populate the database with the data from the json files in seed.js | 5 |  |  |  |
| Server actions, APIs and Repository Implementation to read/write data from the database | 25 |  |  |  |
| Statistics use-case with NextJS | 40 |  |  |  |
| **Documentation**  - Data Model diagram.  - UI Design with screenshots and description.  - Database queries.  - Conducted tests and evidence.  - **Contribution** of each team member [-10pts if not done] | 20 |  |  |  |
| **Total** | 100 |  |  |  |
| Copying and/or plagiarism or not being able to explain or answer questions about the implementation. | -100 |  |  |  |

**Table of contents**

[1. Description of your proposed platform 3](#_Toc197374206)

[2. Data Model 3](#_Toc197374207)

[3. Web API, Server Actions and repository 4](#_Toc197374208)

[4. Implemented statistics use case 5](#_Toc197374209)

[4.1. User Interface 5](#_Toc197374210)

[4.2. Implemented queries 5](#_Toc197374211)

[4.3. Data used in the statics 6](#_Toc197374213)

[4.4. Conducted tests 6](#_Toc197374214)

[4.5. Implemented queries 7](#_Toc197374216)

[5. Discussion of the project contribution of each team member 7](#_Toc197374217)

# Description of your proposed platform

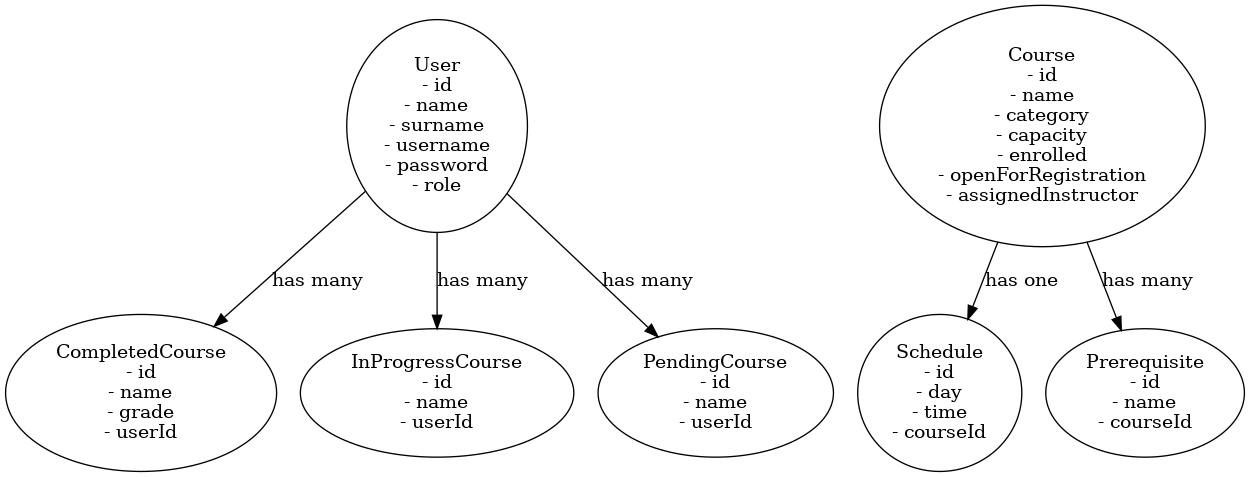
The QU Student Education Platform is a web-based system designed to manage university courses and users efficiently. It supports three roles: **students**, **instructors**, and **admin**. Users can log in securely and get redirected to their respective dashboards. Admins can assign instructors to courses, approve pending courses, and view system-wide statistics. Students and instructors have tailored dashboards that show current, pending, and completed courses. All data is loaded dynamically using JSON and queried using a Prisma-based repository architecture.

# Data Model

The application uses Prisma ORM to define and manage its data model. The schema includes the following main entities:

* User: stores user info including name, role, and login credentials.
* Course: represents courses with capacity, schedule, and enrollment data.
* CompletedCourse, InProgressCourse, PendingCourse: link students to their courses.
* Schedule: stores course timings and days.
* Prerequisite: defines prerequisite courses for any course.

The schema is defined in schema.prisma and synced to the SQLite database using npx prisma db push.

* + Entity diagram:-

# Web API, Server Actions and repository

The backend logic is organized using repository classes in the repo/ folder:

* course-repo.js: handles all course-related queries (getAllCourses, createCourse, updateCourse, etc.)
* user-repo.js: manages user authentication and course associations.
* schedule-repo.js: handles course schedule records.
* prerequisite-repo.js: manages prerequisite relationships.
* statistics-repo.js: provides server actions to compute user role counts, top enrolled courses, and failing students.

These methods are used in api/statistics/route.js to simulate server actions that return data in JSON format. Although this project doesn't use a full server framework like Next.js, these files act as server-side logic handlers and respond to client-side requests using Live Server and mock APIs.

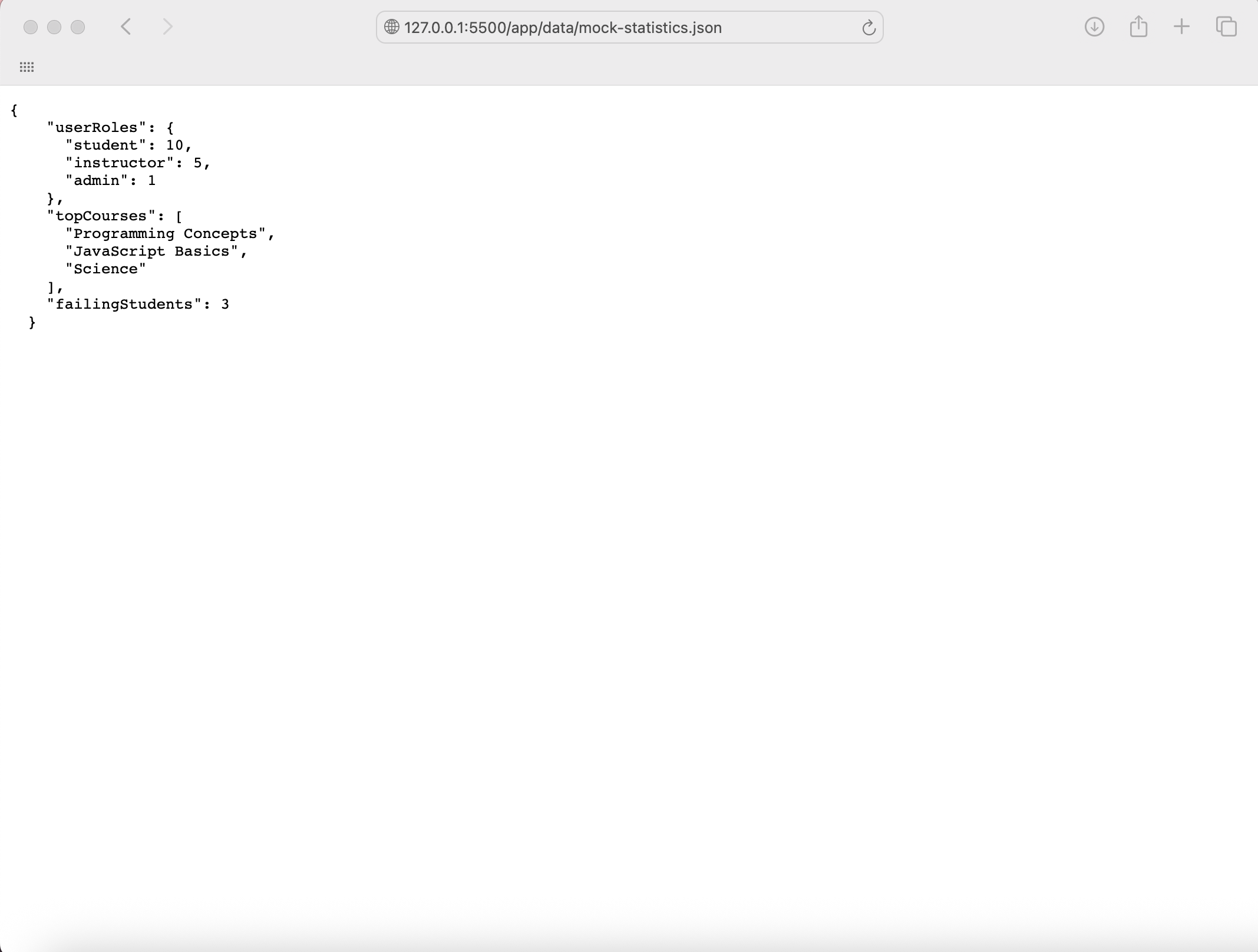
# Implemented statistics use case

# A screenshot of a computer AI-generated content may be incorrect.User Interface (Admin statistics dashboard showing live-loaded mock data via Live Server)

# Implemented queries (JavaScript logs confirming successful fetch and processing of statistics data)

# 

# Data used in the statics (Mock JSON data used to simulate backend API in Live Server)

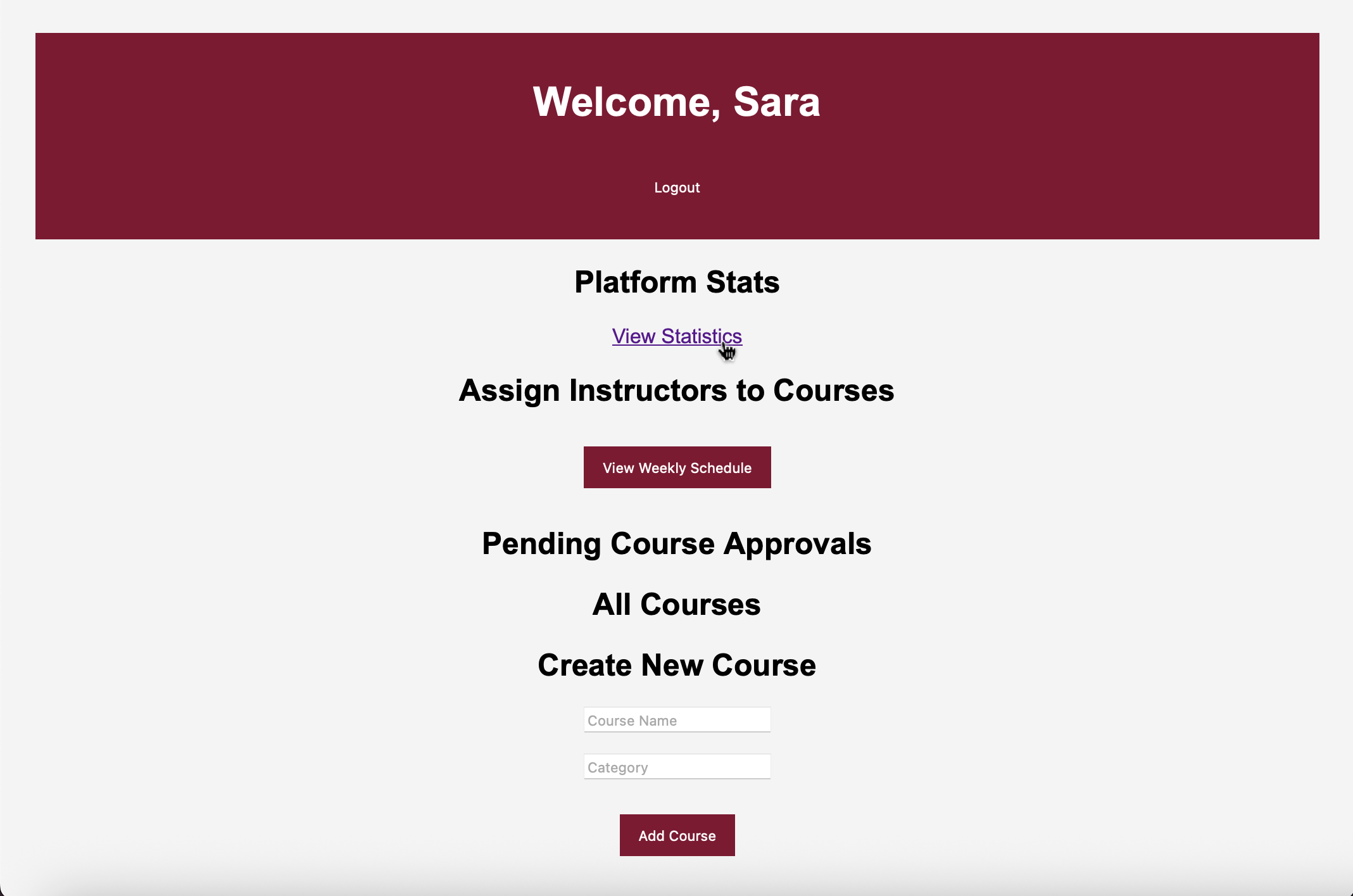


# Conducted tests (Testing role-based access and page navigation using Live Server)

# 

# 

# Implemented queries (Prisma-based repository functions used to calculate statistical values in backend)



# Discussion of the project contribution of each team member

|  |  |
| --- | --- |
| **Student name** | **Student contributions** |
| Alreem Alhaddad (202204202) | A (1, 2) |
| Alreem Zainal (202103014) | A (3, 4) |
| Leina Elsheiri (202005197) | A (5) |
| Bashayer Alnasser (202001583) | B |