Mizān – A Workload Management Web App

Project Phase 1 – Web App Design and Implementation

Worth 15%. The project submission is due by 11:59pm Thursday 10th April 2025.

1. Requirements

Students have lot on their plate! What assignment is due? Is there any in class assessment today? When is my midterm? Sometimes it is hard to keep up with everything and remember everything you must do. Moreover, some instructors tend to forget that students have other classes to study for and have a life to enjoy! On the other hand, some students tend to procrastinate, leading to unnecessary stress!

As part of the university's Road to Student Success initiative, you are tasked with designing and developing Mizān, a workload management web app that helps students and instructors manage assessments effectively. Mizān ensures students stay informed about their coursework, enabling better time management and preventing deadline congestion. Mizān enables instructors to enter course assessments—including quizzes, assignments, projects, exams, presentations, and lab tests—at the start of the semester. Students can stay informed about their workload.

The key use cases to deliver are shown in Figure 1.

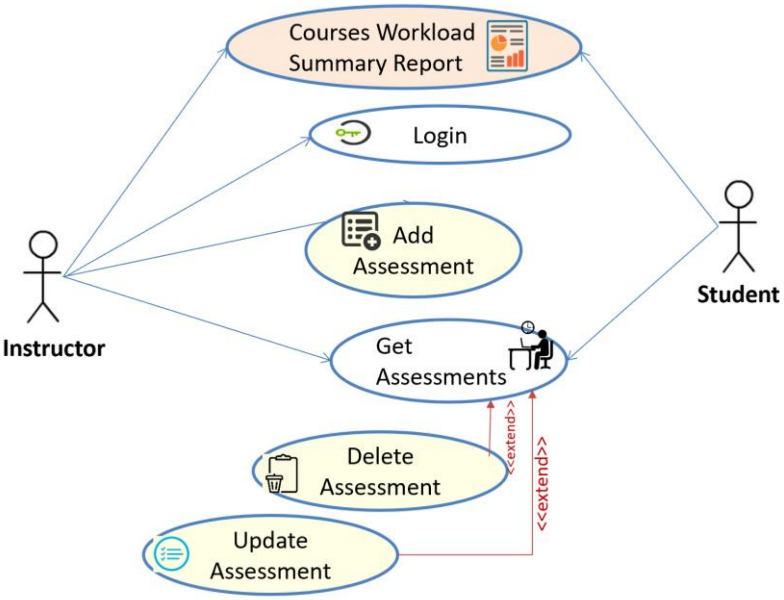
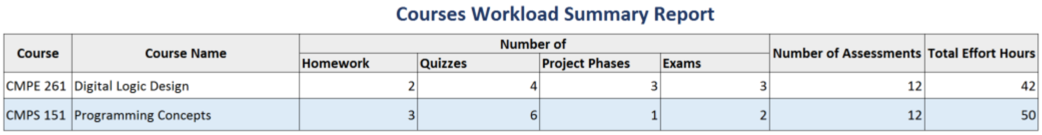
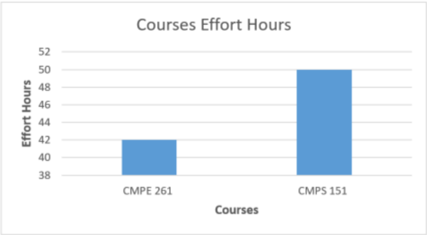


Figure 1. Mizān Use cases

Table 1. Use cases description

<table>
<tr><td><b>Login</b></td><td>Allows the user (i.e., <i>Instructor</i> and <i>Student</i>) to login to use the app.</td></tr>
<tr><td><b>Add Assessment</b></td><td>Instructor can add assessments (e.g., quizzes, assignments, projects, exams) to the courses they teach during a semester.

For each assessment, the instructor should enter the <b>Due date</b>, the average number of <b>Effort Hours</b> required to complete the assessment and the <b>Weight</b> (i.e., percentage towards Final Grade), <b>Assessment Type</b> (i.e., Homework, Quiz, Midterm exam, Final exam, Project) and <b>Title</b>. By default, the assessment type should be assigned as the title but the user can change it. If a previous assessment of the same type was entered before then the system should append a numeric value to the title to indicate its sequential number. For example, when entering the second midterm then its default title should be auto set to ‘Midterm 2’ but the user can edit it.

Validation rules:
- Only one final exam can be added
- At most 2 midterm exams can be added
- No two assessments for a course can have the same due date
- Project phases should be between 1 to 4
- Maximum 8 homework assignments per course</td></tr>
<tr><td><b>Update Assessment</b></td><td>Update an assessment associated with a course. Update should use the same UI as Add Assessment.</td></tr>
<tr><td><b>Delete Assessment</b></td><td>Delete an assessment associated with a course.</td></tr>
<tr><td><b>Get Assessments</b></td><td>Get the assessments associated with a course of set of courses:
- The instructor can only access the assessments of the courses they teach for a particular semester. They can get all the assessments or filter them by course.
- The student can only access the assessments associated with their courses. They can get all the assessments or filter them by course.</td></tr>
<tr><td><b>Courses Workload Summary Report</b></td><td>This report allows the student or the instructor to get a summary of the workload associated with their courses. An example report content and format is shown below:

Courses Workload Summary Report
<table>
<tr><th>Course</th><th>Course Name</th><th>Number of</th><th></th><th></th><th></th><th>Number of Assessments</th><th>Total Effort Hours</th></tr>
<tr><th></th><th></th><th>Homework</th><th>Quizzes</th><th>Project Phases</th><th>Exams</th><th></th><th></th></tr>
<tr><td>CMPE 261</td><td>Digital Logic Design</td><td>2</td><td>4</td><td>3</td><td>3</td><td>12</td><td>42</td></tr>
<tr><td>CMPS 151</td><td>Programming Concepts</td><td>3</td><td>6</td><td>1</td><td>2</td><td>12</td><td>50</td></tr>
</table>

</td></tr>
</table>

Deliverables:

1) Document the app design to deliver Mizān use cases including:

The design documentation should include at least the following:

- Class Diagram showing Entities, Repositories and Services.
- UI Design and navigation.

During the weekly project meetings, you are required to present and discuss your design with the instructor and get feedback.

2) Implement the client-side and the server-side Web components to deliver Mizān use cases based on your previously developed and validated design.

   The HalaqaMetrash should be fully implemented using Next.js. The application data can be managed either using json files or https://mockapi.io/. The web pages could use, HTML, CCS and JavaScript. The pages should comply with Web user interface design best practices. Also remember that 'there is elegance in simplicity'.

3) Deploy the app to a cloud hosting service such as https://vercel.com/

Push your implementation and documentation to your group GitHub repository as you make progress.

2. Grading rubric

Criteria	%	Functionality*	Quality of the implementation
Application Design • Class Diagram showing Entities, Repositories and Services. • UI Design and navigation.	15		
Complete and correct implementation of the requirements:			
Add Assessment	20		
Update Assessment	15		
Delete Assessment	5		
Get Assessments	15		
Courses Workload Summary Report	15		
Testing documentation with evidence of correct implementation using unit tests and snapshots illustrating the results of testing.	10		
Deploy the app to a cloud hosting service such as https://vercel.com/	5		
Total	100		

Copying and/or plagiarism or not being able to explain or answer questions about the implementation	- 100%		

* **Possible grading for functionality**: ***Complete and Working*** (get 70% of the assigned grade), ***Complete and Not working*** (lose 40% of assigned grade) and ***Not done*** get 0. The remaining grade is assigned to the quality of the implementation. In case your implementation is not working then 40% of the grade will be lost and the remaining 60% will be determined based on of the code quality and how close your solution to the working implementation. Quality includes **correct application of MVC**, meaningful naming of identifiers, no redundant code, simple and efficient design, clean code without unnecessary files/code, use of comments where necessary, proper white space and indentation. **Marks will be reduced** for code duplication, poor/inefficient coding practices, poor naming of identifiers and unnecessary complex/poor user interface design.

3. Ground Rules

- All assignments **must be your own original work**, not based on the work of other students, online examples/tutorials, or any other material from any other source. Any assignments found to be based on work other than your own will automatically be given a **grade of zero**, and may lead to further disciplinary action as per QU policy.
- All assignments must be submitted electronically to Github. You should push your work to Github as you make progress. Late submission policy: 10 points deduction for each late day and 0 after 3 days.