



Ben-Gurion University
of the Negev

Exam Management System

Application Design Document (ADD)

Ofek Nov
Mor Abo
Roi Tiefenbrunn
Idan Aharoni



Contents

| | |
|--|-----------|
| Use Cases | 3 |
| System Architecture | 6 |
| Data Model | 8 |
| Description of Data Objects | 8 |
| Data Objects Relationships | 9 |
| Databases | 10 |
| Behavioral Analysis | 11 |
| Sequence Diagrams | 11 |
| Object-Oriented Analysis | 16 |
| Class Diagram | 16 |
| Packages | 17 |
| User Interface Draft | 18 |
| Testing | 20 |
| Implementation Constraints | 20 |
| Platform Constraints | 22 |
| Special Restrictions & Limitations | 22 |

Use Cases

1. Manage Courses

- **Description:** This use case allows administrators to create courses within the system.
- **Actor:** Administrator
- **Preconditions:** The user must be logged in as an administrator.
- **Basic Flow:**
 1. The administrator accesses the course management section of the system.
 2. The administrator views a list of existing courses.
 3. Administrator can add a new course, and ask a specific user to be the course admin.
 4. The course admin that was chosen gets a request to become the course admin, he must accept or deny the request.
- **Alternate Flows:**

If the course ID already exists in the system or the user that the admin chose to be the course admin doesn't exist, the course is not created, and an appropriate error message appears to the admin.

2. Create and Assign Tasks

- **Description:** This use case enables users to create tasks and assign them to specific individuals or roles.
- **Actor:** Lecturer, TA
- **Preconditions:** The user must be logged in with appropriate permissions.
- **Basic Flow:**
 1. The user accesses the task creation section of the system.
 2. The user creates a new task by providing a description, and priority, and optionally assigning it to a user or role.
 3. The user saves the task, and it becomes available for assignment or completion.
- **Alternate Flows:**

If the task is assigned to a role, the system distributes it among users with that role based on predefined algorithms.

3. Manage Questions

- **Description:** This use case allows users to manage questions within the system.
- **Actor:** Lecturer, TA
- **Preconditions:** The user must be logged in as a Lecturer or TA.
- **Basic Flow:**
 1. The user accesses the questions management section of a course.
 2. User creates new content, such as questions, stems, meta-questions, and appendices, organized by subjects and keywords.
 3. User edits, deletes, or validates existing content as needed.
- **Alternate Flows:**

Other users on the course need to review and provide feedback on proposed content changes before finalizing them.

4. Generate Exams

- **Description:** This use case allows users to generate exams based on the available content.
- **Actor:** Lecturer
- **Preconditions:** The user must be logged in as a Lecturer and the course must have defined exam properties such as subjects, number of questions, and grading system.
- **Basic Flow:**
 1. The user accesses the exam creation section of the system.
 2. The user selects to create an exam and provides cause.
 3. The system generates the exam using LaTeX-based templates and exports it to a PDF file and a Word file.
- **Alternate Flows:**

If the user wants to preview the exam before finalizing it, he can review it and make some adjustments as needed.



5. Finishing a Task

- **Description:** This use case allows users to finish a task that is assigned to them.
- **Actor:** Lecturer, TA
- **Preconditions:** The user must be logged in and a task is assigned to him.
- **Basic Flow:**
 1. The user accesses the tasks section of the system.
 2. The user selects a task to view.
 3. The user answers the task and submits his answer.
- **Alternate Flows:**

If the user doesn't have any assigned tasks, he can view all tasks that are suggested to his type of role and assign himself one.

System Architecture

1. Client-Side Components:

- **Web Browser:**
 - **Location:** Installed on user devices (client machines).
 - **Functionality:** Renders the user interface provided by the front-end framework and interacts with the backend server via rest-API requests.

2. Frontend Framework:

- **React Framework:**
 - **Location:** Bundled as JavaScript files deployed to a web server.
 - **Functionality:** Provides the structure and components for building the user interface of the WMS, including role-based dashboards and content creation forms.

3. Backend Server:

- **Node.js with Restify:**
 - **Location:** Deployed on a dedicated server or cloud platform.
 - **Functionality:** Handles HTTP requests, routes them to the appropriate handlers, and builds RESTful APIs using Restify. Implements business logic, authentication, and authorization, and interacts with the database.

4. Database:

- **PostgreSQL Database:**
 - **Location:** Hosted on a separate database server.
 - **Functionality:** Stores and manages system data, including user information, tasks, content, and version control.

5. Authentication and Authorization:

- **Authentication Service:**
 - **Location:** Part of the backend server.
 - **Functionality:** Verifies user identities, issues authentication tokens (e.g., JWT), and manages user sessions.
- **Authorization Middleware:**
 - **Location:** Integrated into the backend server.
 - **Functionality:** Enforces access control based on user roles and permissions defined in the system, ensuring that users can only access authorized resources.

6. Task Distribution and Management:

- **Task Distribution Module:**
 - **Location:** Implemented within the backend server.
 - **Functionality:** Implements algorithms for task assignment and distribution among users based on workload, priority, and expertise.
- **Task Management Service:**
 - **Location:** part of the backend server.
 - **Functionality:** Handles CRUD operations for tasks, including creation, assignment, update, and deletion.



7. Content Management:

- **Content Storage:**
 - **Location:** Integrated with the backend server.
 - **Functionality:** Stores various types of content, including questions, stems, meta-questions, and solutions.
- **Content Management Service:**
 - **Location:** Deployed within the backend server.
 - **Functionality:** Implements functionalities for creating, editing, and deleting content items, as well as version control.

8. LaTeX Processing:

- **LaTeX Engine:**
 - **Location:** Part of the backend server.
 - **Functionality:** Converts LaTeX input into printable documents such as exams, keys, and solutions.
- **LaTeX Processing Service:**
 - **Location:** Integrated with the backend server.
 - **Functionality:** Generates LaTeX output based on user inputs, ensuring flexibility and customization for exam creation.

Data Model

This chapter describes the main information data domain of the application. Objects are real-world entities that have counterparts within the system. Associated with each object is a set of attributes and functions. Associating the functions will be handled in Chapter 5. The remainder of this chapter is devoted to analyzing the attributes.

Description of Data Objects

1. User

- **Attributes:**
 - **Username** (string): User's username for authentication.
 - **Password** (string): User's password for authentication.
 - **Type** (enum): User's role in the system (e.g., instructor, TA, grader).
 - **Course** (Course): User's assigned course in the system.

2. Task:

- **Attributes:**
 - **TaskId** (int): Unique identifier for the task.
 - **ForWhom** (User): Reference to the user that is in charge of the task.
 - **Priority** (int): Priority level of the task.
 - **Type** (enum): The type of the task (e.g., CourseAdminRequest, NewTaRequest).
 - **Properties** (tuple<string, string>): Properties of the task course (CourseId, CourseName).
 - **Description** (string): Description of the task.
 - **AssignedUser** (User): Reference to the user to whom the task is assigned.
 - **Action** (Lambda): The action that will run when a task is completed.
 - **Finished** (boolean): Task finished or not.

3. Question:

- **Attributes:**
 - **Question** (string): The question itself.
 - **Answers** (List<string>): List of correct answers.
 - **Diversions** (List<string>): List of diversions for the question.

4. Meta-Question:

- **Attributes:**
 - **Subject** (string): The subject of the meta-question.
 - **Questions** (List<Question>): List of all the questions that are part of the meta-question.
 - **appendix** (File): An appendix to the meta-question.

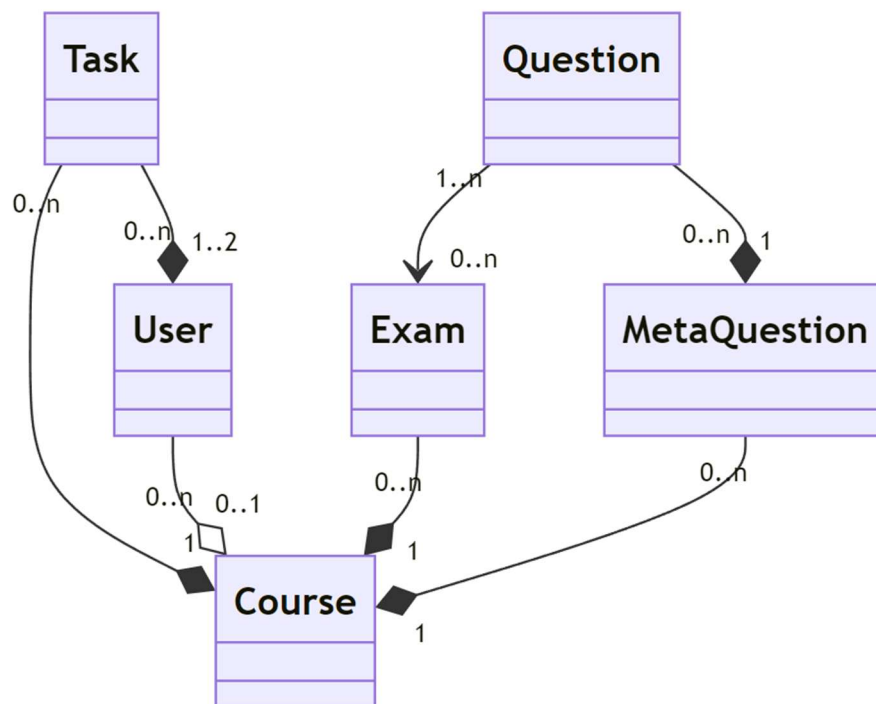
5. **Course:**

- **Attributes:**
 - **Properties** (tuple<string, string>): Properties of the task course (CourseId, CourseName).
 - **Lecturers** (List<User>): List of all the lecturers in the course.
 - **TAs** (List<User>): List of all the TAs in the course.
 - **MetaQuestions** (List<MetaQuestion>): List of all the Meta-Questions that have been created in the course.
 - **PastExams** (List<Exam>): List of all the past exams in the course.

6. **Exam:**

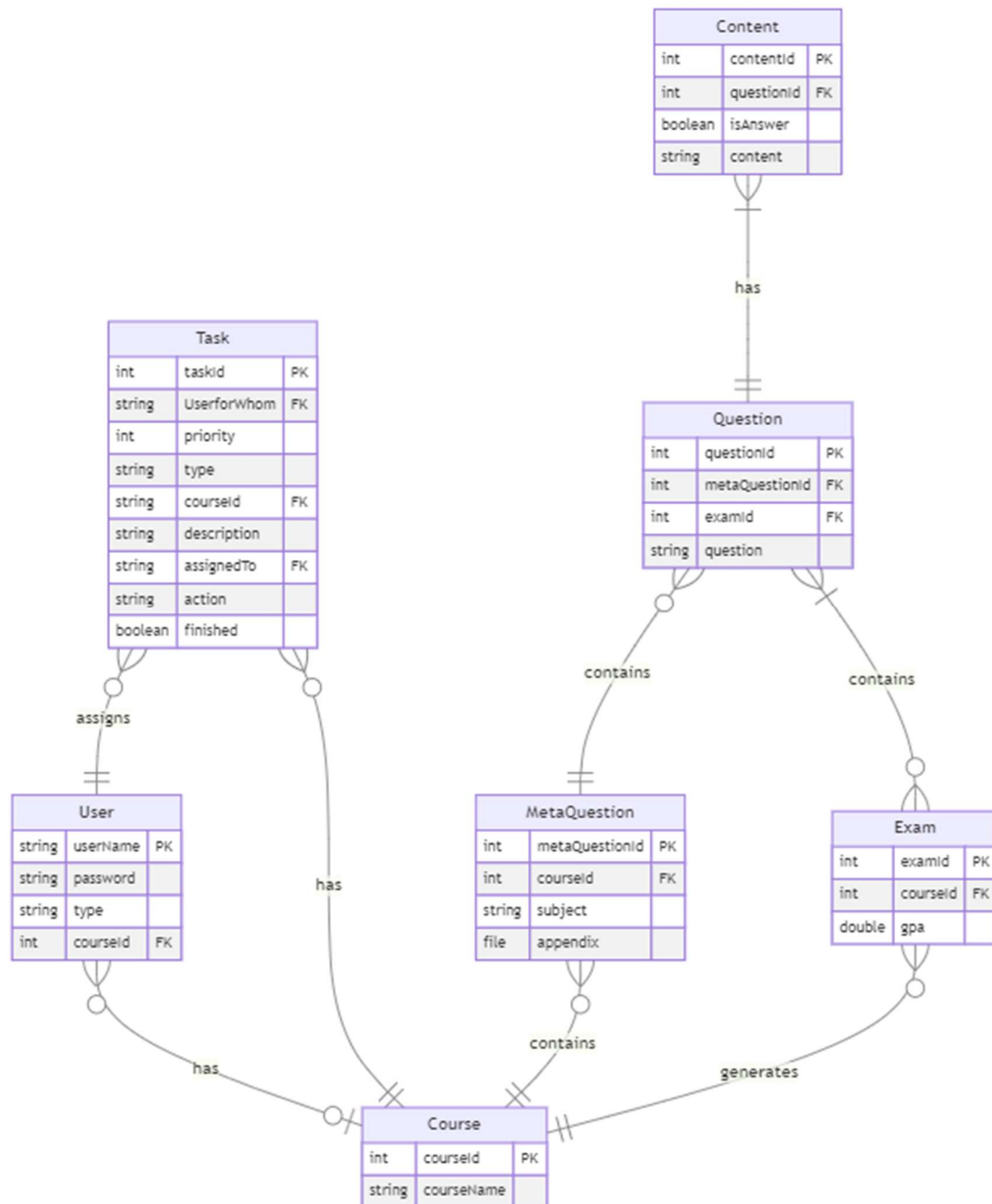
- **Attributes:**
 - **GPA** (double): Average score in the exam.
 - **Questions** (List<Question>): List of all the Questions in the exam.

Data Objects Relationships





Databases

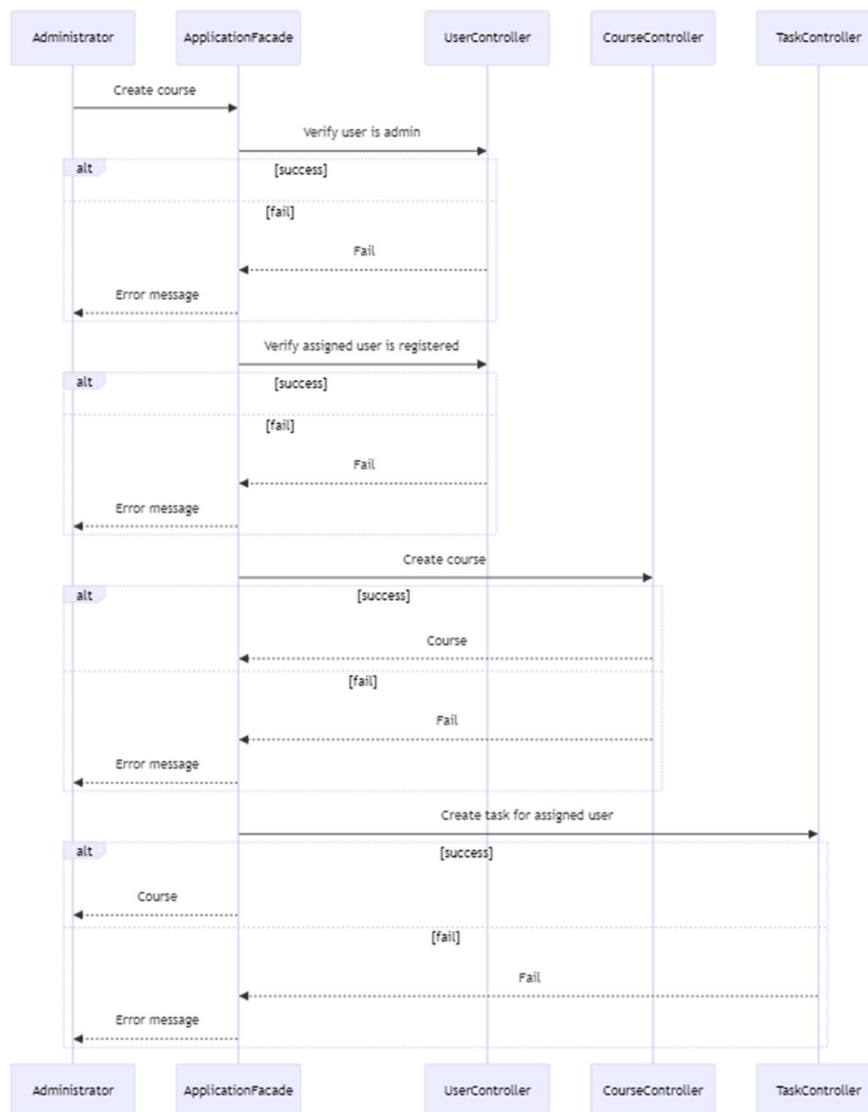


Behavioral Analysis

This chapter describes the control flow of our system.

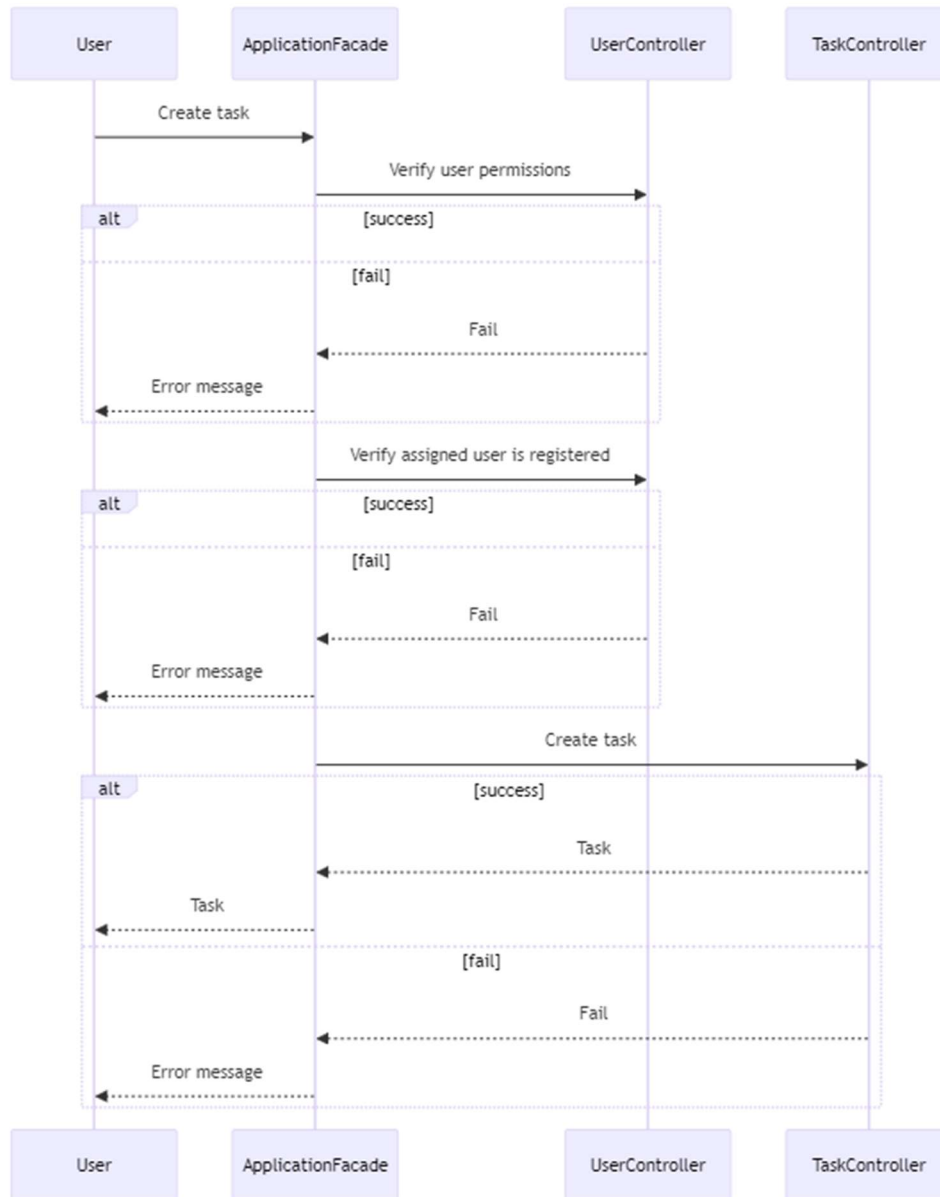
Sequence Diagrams

1) Manage Courses



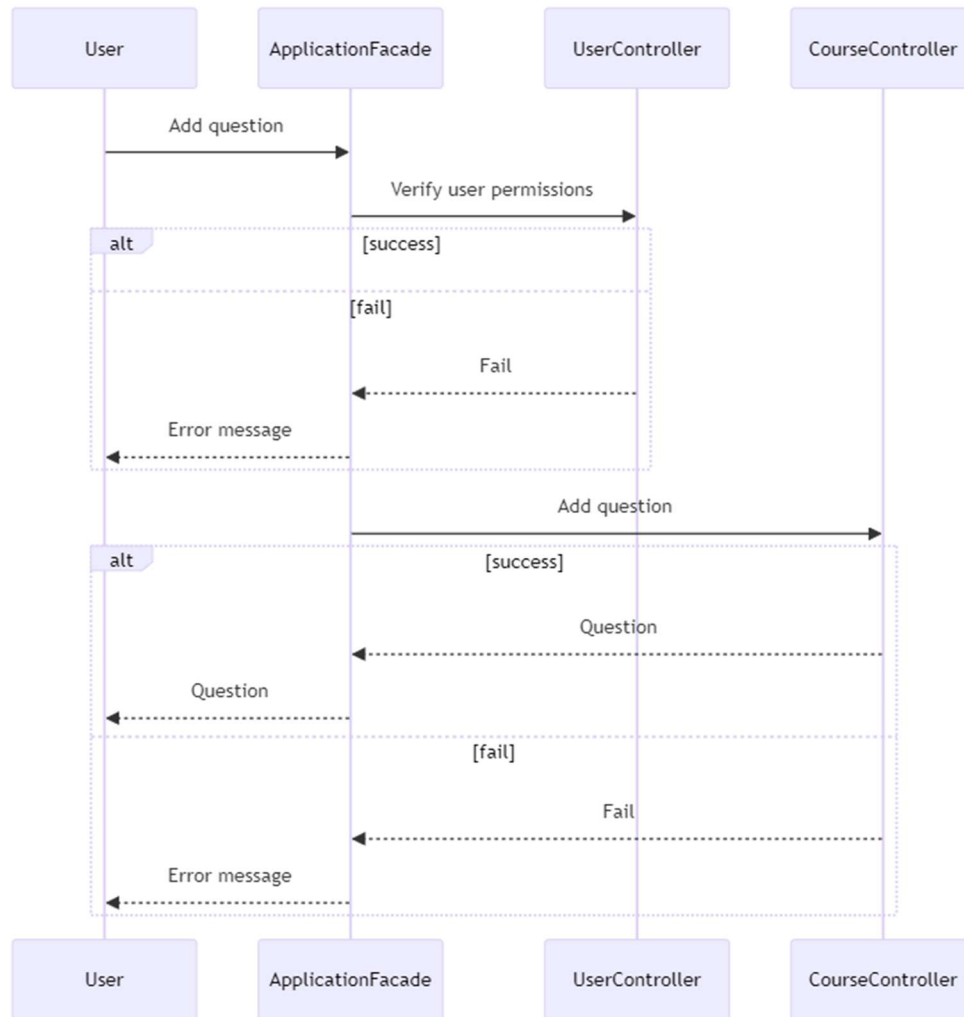


2) Create and Assign Tasks



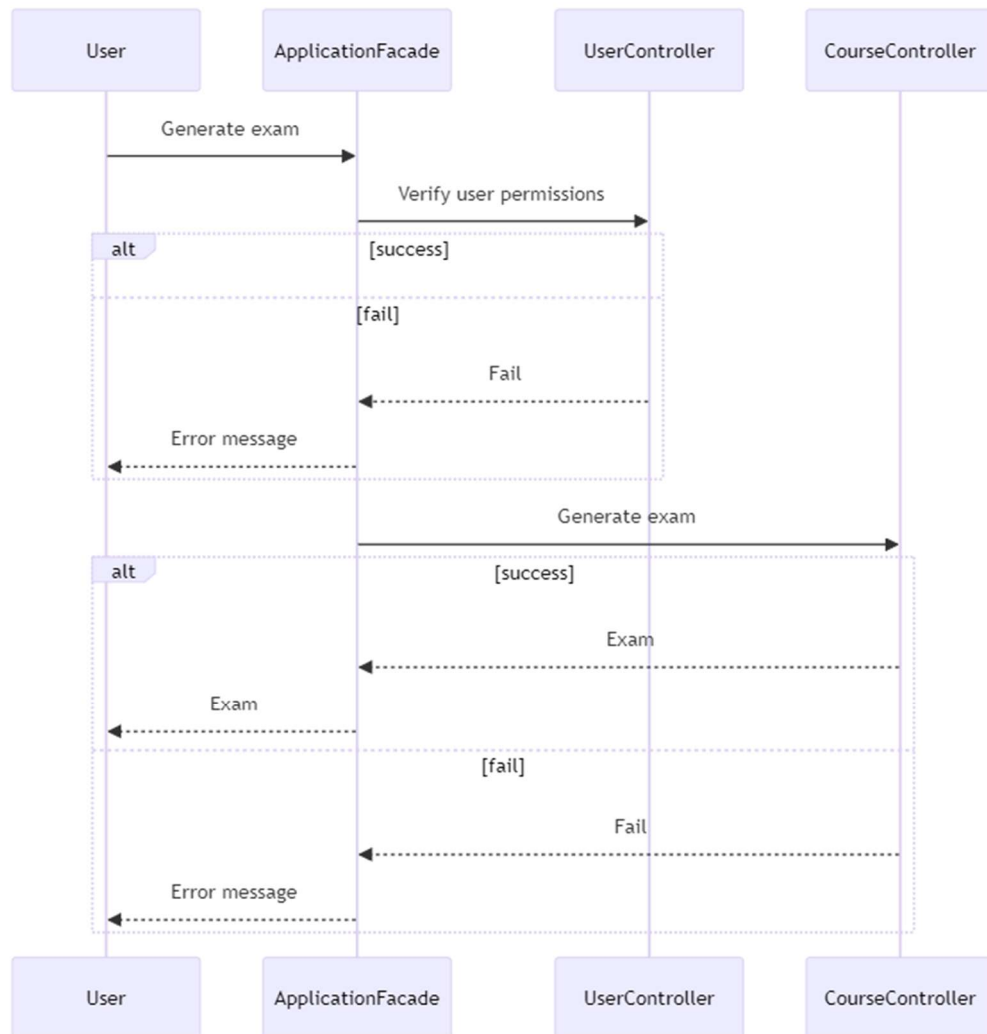


3) Manage Questions



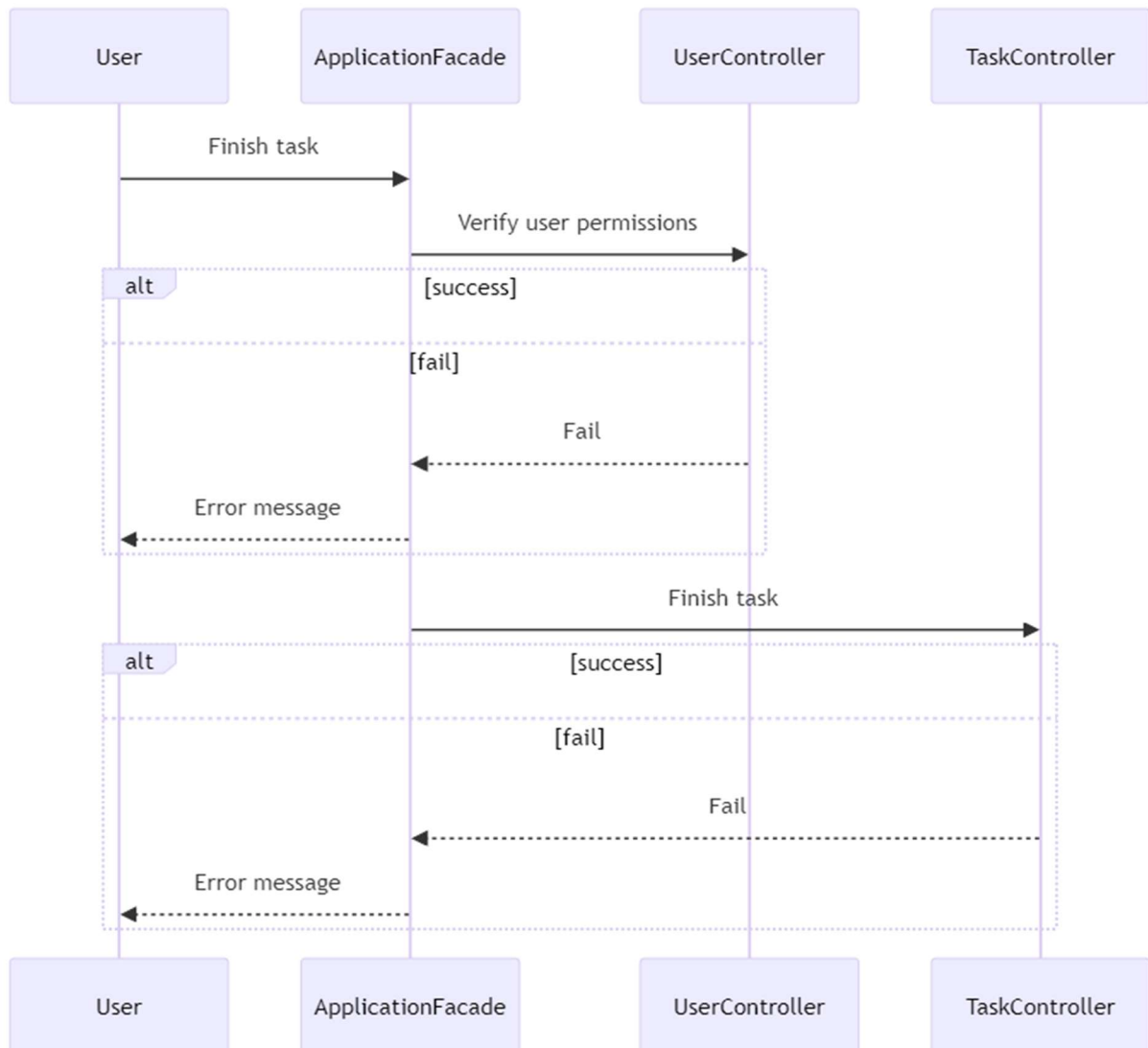


4) Generate Exams



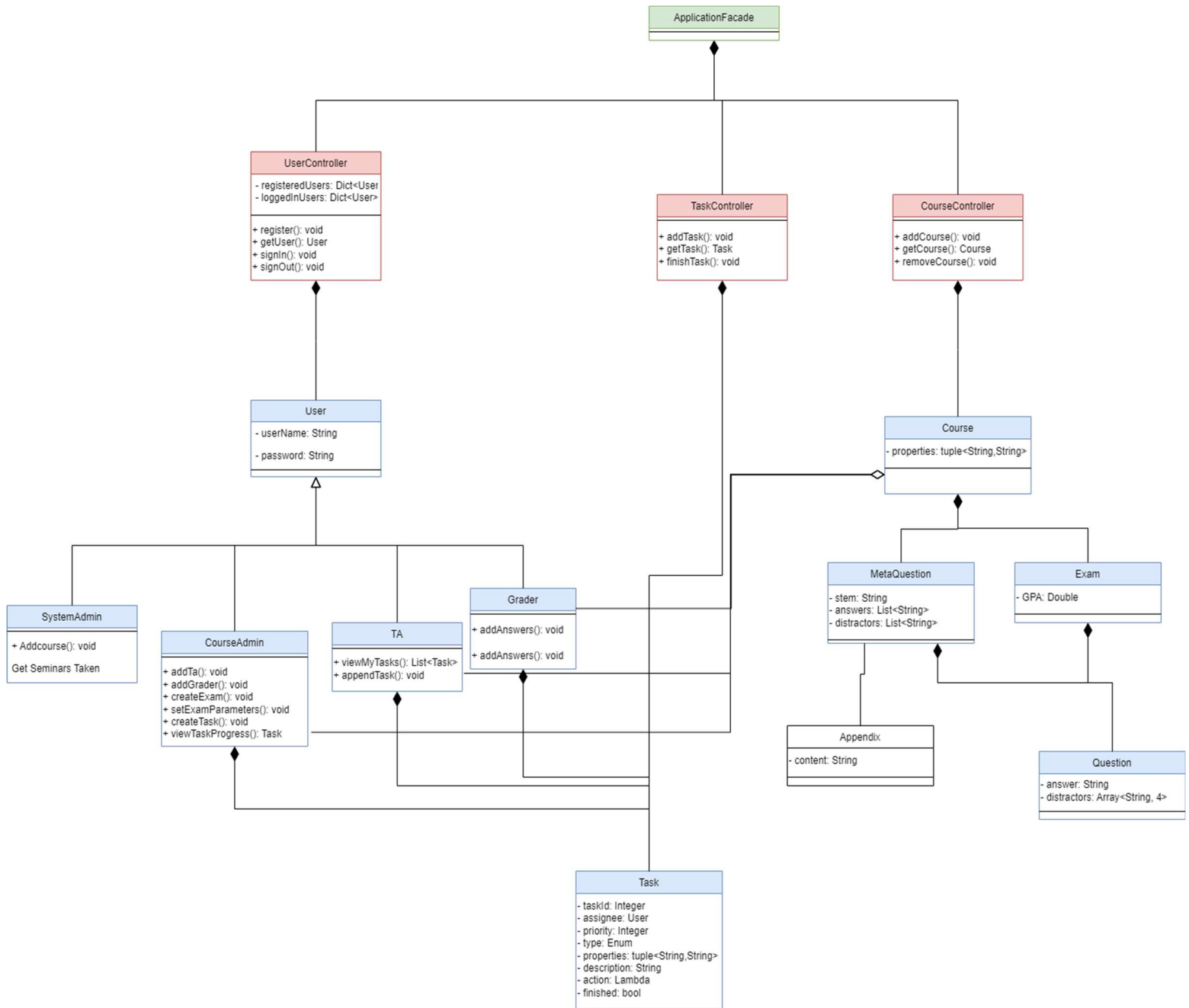


5) Finishing a Task



Object-Oriented Analysis

Class Diagram



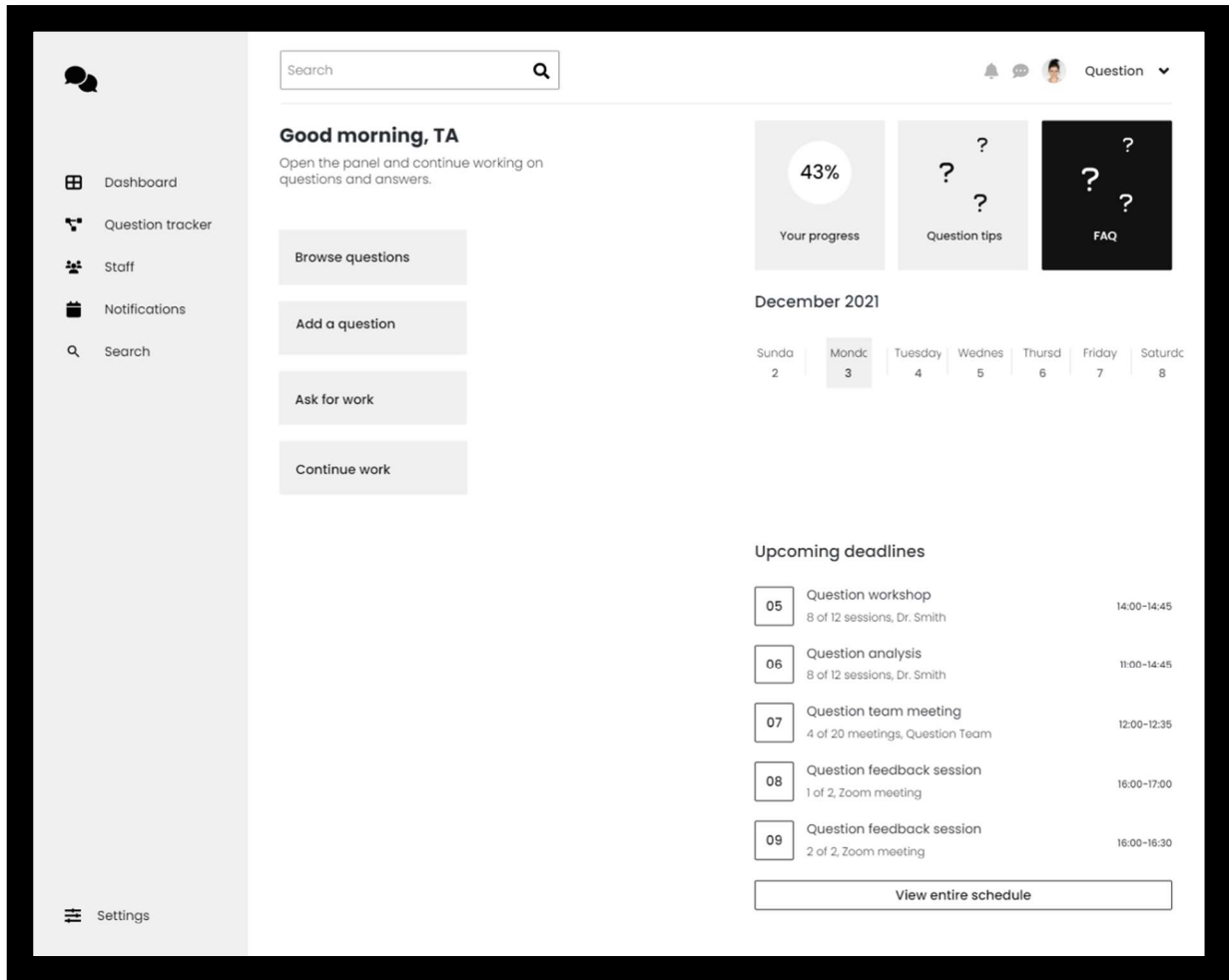


Packages

- **Src**
 - **Main**
 - **Business** (Contains all the business logic of the application)
 - **CourseManager** (Contains all the course-related entities)
 - **TaskManager** (Contains all the task-related entities)
 - **UserManager** (Contains all the user-related entities)
 - **Test**
 - **CourseTests**
 - **TaskTests**
 - **UserTests**


User Interface Draft


Course Dashboard





The course dashboard will have all the course functionalities based on user permissions. For example, the TA can go to “Browse questions” to see all the course questions that are in the course database, add a new question to the course database, and ask for a new task from the course manager.


Question Page


 Dashboard

 Question tracker




 Staff

 Notifications

 Search

 Settings

Work Page

 Example 

Question:

סמנו את נקודת השבת הקטנה ביותר של M

Possible Answer:

I (Y (I M))

Check the correct value of this answer:

☒ Correct answer

☐ Distractor

Save Changes

The question page presents to the user the question itself, and the answer, and will let the user choose the option to change the value of the question to be an answer or a distractor (only if the user has the correct permissions).

Testing

Implementation Constraints

Performance:

1. Support for Multiple Users:

- **Test:** Simulate concurrent user activity on the system and measure response times and throughput.
- **Expected Result:** The system should handle a specified number of concurrent users without significant degradation in performance.
- **Actual Result:** Use load testing tools to simulate concurrent user activity and verify that the system meets performance requirements.

Reliability & Stability:

1. Data Recovery from Database:

- **Test:** Introduce data corruption or loss in the database and attempt to recover the data.
- **Expected Result:** The system should successfully recover data from the backup and restore it to its original state.
- **Actual Result:** Test database backup and recovery processes and verify successful data recovery.

2. Rollback Mechanism:

- **Test:** Introduce errors or failures (e.g., internet connection loss, hardware failure) and observe system behavior.
- **Expected Result:** The system should roll back all related updates to the last stable version without data loss.
- **Actual Result:** Introduce errors and verify that the system rolls back updates and maintains data integrity.

Safety & Security:

1. Encryption of Sensitive Data:

- **Test:** Store sensitive data (e.g., passwords) in the database and verify that it is encrypted.
- **Expected Result:** Sensitive data should be stored securely using encryption techniques.
- **Actual Result:** Store passwords in the database and verify that they are stored encrypted.

2. Access Control:

- **Test:** Attempt to access sensitive data without appropriate permissions.
- **Expected Result:** Access should be denied to users without proper authorization.
- **Actual Result:** Test access control mechanisms and verify that unauthorized users cannot access sensitive data.

Portability:

1. **Network Compatibility:**

- **Test:** Access the system from different networks and verify connectivity.
- **Expected Result:** The system should be accessible only through the university network.
- **Actual Result:** Verify that only through the university network we can connect to the system.

2. **Browser Compatibility:**

- **Test:** Access the system from different web browsers (e.g., Chrome, Firefox, Safari) and verify functionality.
- **Expected Result:** The system should be accessible and functional on different browsers.
- **Actual Result:** Test the system on various browsers and ensure compatibility.

3. **Language Support:**

- **Test:** Use the system with Hebrew language settings and verify the correct display and functionality.
- **Expected Result:** The system should support Hebrew language input and display.
- **Actual Result:** Set the system language to Hebrew and verify the correct display and functionality.

Usability:

1. **User Interface Testing:**

- **Test:** Provide the system to users with varying levels of computer expertise and collect feedback.
- **Expected Result:** Users should find the interface intuitive and easy to use.
- **Actual Result:** Gather user feedback and make necessary improvements to the interface.

Availability:

1. **Continuous Availability:**

- **Test:** Monitor system availability 24/7 and respond to any downtime promptly.
- **Expected Result:** The system should be available for use at all times except during scheduled maintenance.
- **Actual Result:** Monitor system uptime and address any issues affecting availability immediately.

Platform Constraints

1. **Interactive Inputs:**
 - **Test:** Allow end users to interact with the system and observe their inputs.
 - **Expected Result:** Inputs provided by end users should be processed correctly by the system.
 - **Actual Result:** Allow end users to interact with the system and verify correct processing of inputs.
2. **Access to Student Exam Answers:**
 - **Test:** Access student exam answers and use them for analysis.
 - **Expected Result:** The system should be able to access student exam answers and perform analysis as required.
 - **Actual Result:** Access student exam answers and verify that the system can analyze them accurately.

Special Restrictions & Limitations

1. **Remote Installation:**
 - **Test:** Install the system remotely on the university's servers and verify successful installation.
 - **Expected Result:** The system should be installed and deployed on the university's servers without issues.
 - **Actual Result:** Install the system remotely and verify successful deployment.
2. **End User Cooperation:**
 - **Test:** Communicate system requirements and dependencies to end users and ensure their cooperation.
 - **Expected Result:** End users should understand their role in the system and provide necessary cooperation for data flow.
 - **Actual Result:** Communicate system requirements to end users and ensure their cooperation throughout the testing process.