CS390 Software Engineering – Fall 2024
Assignment 2: System Design

Team leader: **Ahmed Abdelbary 2221191298**
Member 1: **Lamiaa Abushara 2211150120**
Member 2: **Seba Alzaher 2201114862**

Table of Contents

Individual Contributions

Task	Ahmed	Lamiaa	Seba
Introduction	40%	30%	30%
Glossary	30%	30%	40%
System Architecture and System Design	25%	25%	50%
Class diagram and interface specification	40%	30%	30%
Interaction diagrams	50%	25%	25%
User interface design and Implementation	30%	35%	35%
Risk management	30%	40%	30%
Project plan	25%	50%	25%
References	30%	35%	35%

1. Introduction

This document outlines the complete system design and system architecture of the Computer Science Assistance Request and Credit System (CS ARCS) project. It further develops the primary analysis and requirements that were presented in Assignment 1 and elaborates on the design phase. The focus is on the most recommended methods of software engineering for designing a large, secure and user-friendly system.

The design employs architectural styles and patterns in order to assure the system's modularity, its maintainability and conformity to both functional and non-functional requirements of the system. Provides other implementation aspects like detailed modeling of interaction of various levels of system hierarchy and their subsystems and user interface enhancements.

This document presents a consistent focus on architectural decisions, much necessary subsystem design, rigorous testing and aims at the successful evolution and operationalization of the CS ARCS system.

2. Glossary

Authentication:
The process of verifying the identity of a user to grant them access to the system based on their role.

Authorization:
The process of ensuring that a user has the appropriate permissions to perform certain actions within the system.

Dashboard:
A user interface element that provides an overview of the most relevant tasks, notifications, or statuses tailored to the user's role.

Faculty Member:
A user role responsible for monitoring tasks, reviewing feedback, and awarding extra credit.

Feedback:
A structured review submitted by Junior Students to evaluate the quality of tutoring assistance provided by Senior Students.

Junior Student:
A user role responsible for creating assistance requests and providing feedback on completed tasks.

Notification:
A system-generated message that updates users about key events such as task assignment, completion, or feedback submission.

Role-Based Access Control (RBAC):
A security mechanism used to restrict system access to users based on their roles (e.g., Junior, Senior, Faculty).

Senior Student:
A user role responsible for accepting and completing assistance requests posted by Junior Students.

Task:
A request for assistance created by Junior Students and fulfilled by Senior Students.

Token:
A secure digital key used for authenticating a user session, typically generated during login.

Use Case:
A specific scenario describing how a user interacts with the system to achieve a particular goal.

User Interface (UI):
The part of the system that users interact with, including dashboards, forms, and notifications.

System Scalability:
The ability of the system to handle increasing workloads, such as additional users or tasks, without a decline in performance.

System Modularity:
The design principle of dividing the system into independent modules to improve maintainability and scalability.

API (Application Programming Interface):
A set of rules that allows different parts of the system (or external systems) to interact with each other programmatically.

SQLite:
A lightweight database used during the development phase for managing structured data.

PostgreSQL:
A robust and scalable database recommended for the production environment of the system.

Client-Server Model:
An architectural pattern where the client (e.g., a web browser) interacts with the server to perform actions and retrieve data.

Testing:
The process of evaluating the system to ensure that it functions as expected and meets its requirements.

Traceability Matrix:
A document that maps test cases to their corresponding use cases to ensure all requirements are covered.

.

3. System Architecture and System Design

Read the following example links (find more resources on your own) before starting this section: •

Wikipedia , Article 1, Article 2 , Article3

3.1. Architectural Styles

In our CS ARCS system, we utilize layered architectural pattern and client-server architecture. These styles fulfill the system's non-functional properties such as scalability, modularity, and maintainability.

The client-server model divides the system into its basic two parts:

- **Client:** Users (eg. Faculty, juniors and seniors) interacting with the system through a web interface
- **Server:** backend system responsible for processing tasks, managing data and assuring data integrity

The client-server model satisfies the non-functional system requirements such as:

- **Scalability (NF-01 , NF-03)** : The client-server design guarantees that the server can support multiple clients concurrently, ensuring that it can be accessed at any time and for long periods with very little downtime. additional servers can be deployed to handle the increased load.
- **Security (NF-04, NF-05)**: the core operations like authentication and data storage are executed at the server side , protecting much of the sensitive information from exposure. **Responsiveness (NF-03)**: tailored responses are given to a variety of client devices including mobile ones by utilizing server-side processing.

The three-layered system architecture eliminates complication and divides the system into three distinctive layers:

- **Presentation Layer:** This is user specific element of the system design which focuses on making the system functional and accessible to the users promptly. Moreover, it presents controls and interfaces, according to different users' (juniors, seniors, faculties) to the system as per the specification of NF-02.
- **Business Logic Layer:** processes the main features of the application, like managing tasks, notifications and submitting feedback.
- **Data Layer:** Manages storage, ensuring data integrity and availability.

This layered approach helps to meet the system non-functional requirements such as the following:

Modularity and Maintainability (NF-08): Each layer is independent, making it easier to implement updates or changes to specific parts of the system without impacting others.

Accessibility Standards (NF-06): The presentation layer has inbuilt features that allow it to meet the benchmarks set by the standards for the ease of use of the systems for all types of users.

Future Proofing (NF-10): It is possible to introduce changes in layers one at a time making it less disruptive.

Ability to Keep Track of Changes and Review Them (NF-09): Specific logs are kept in the said layer to make sure that all users of the system can be traced and their actions logged.

3.2. Subsystems

CS ARCS consists of many subsystems:

Authentication Subsystem:

Manages user logins, signups and assigns users to roles of junior, senior and faculty.

Concerned with checking users' identities, managing users' permissions, and preserving user sessions safely.

Task Management Subsystem:

Allows junior students to create tasks, senior students to accept and complete them, and faculty to review.

In charge of task creation/assignment, monitoring the task completion and ensuring internal equity in tasks distribution among seniors..

Admin/Faculty Console Subsystem:

Provides faculty members with tools to monitor tasks, review feedback, and award credits.

Responsible for displaying task statuses for review and enabling manual overrides for task assignments and credit allocation.

Notification Subsystem:

Notifies users of task updates, feedback reminders, and system changes.

Responsible for delivering in-app and email notifications and tracking notifications sent to users for auditing purposes.

Feedback Subsystem:

Collects feedback from junior students and allow faculty to review it for quality assurance.

Responsible for enabling juniors to rate their experience with tutors and providing tools for faculty to review feedback and award extra credits.

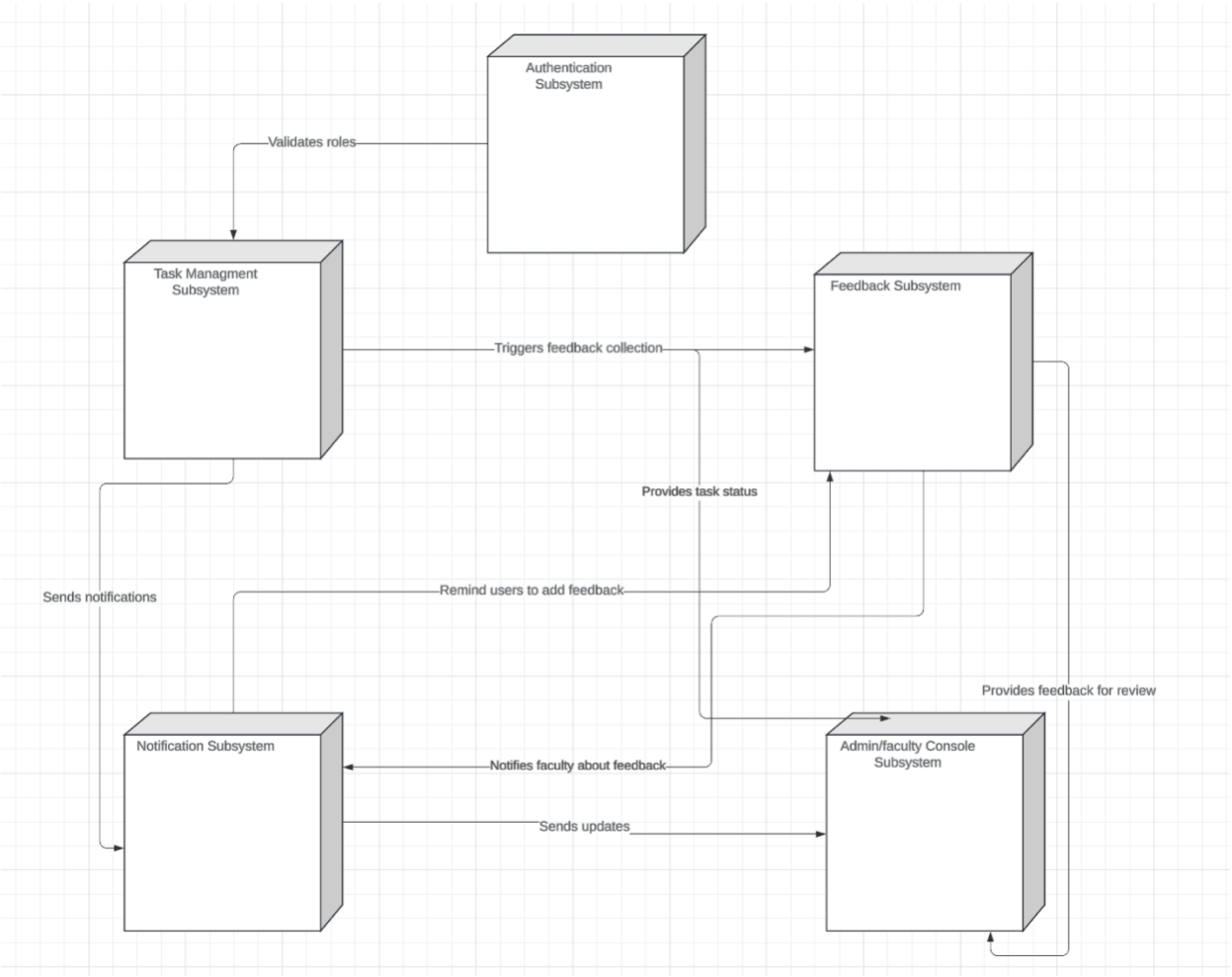


Fig.(1): UML diagram explaining the subsystems and their interactions with each other

3.3. Mapping subsystems to hardware

First, we will need to identify the hardware components that will be needed in this mapping for our CS ARCS system:

- Client devices such as computers, mobile phones and tablets
- Application server
- Database server
- Email server (for sending email notification to users)

Subsystem	Hardware Component	Reason
Authentication Subsystem	Application Server	Handles user authentication (e.g., login, role validation) and manages secure user sessions.
Task Management Subsystem	Application Server and Database Server	Processes task creation/updates on the application server and stores tasks persistently in the database.
Feedback Subsystem	Application Server and Database Server	Collects feedback on the application server and saves feedback data in the database for admin review.
Notification Subsystem	Application Server, Email Server, and Client Devices	Sends notifications to users via the application server; email server handles external notifications (optional).
Admin/Faculty Console Subsystem	Client Devices (faculty desktops/laptops) and Application Server	Displays feedback and task statuses using a browser on client devices; connects to the server for logic processing.

Table (1): subsystems mapped to hardware components and the reason

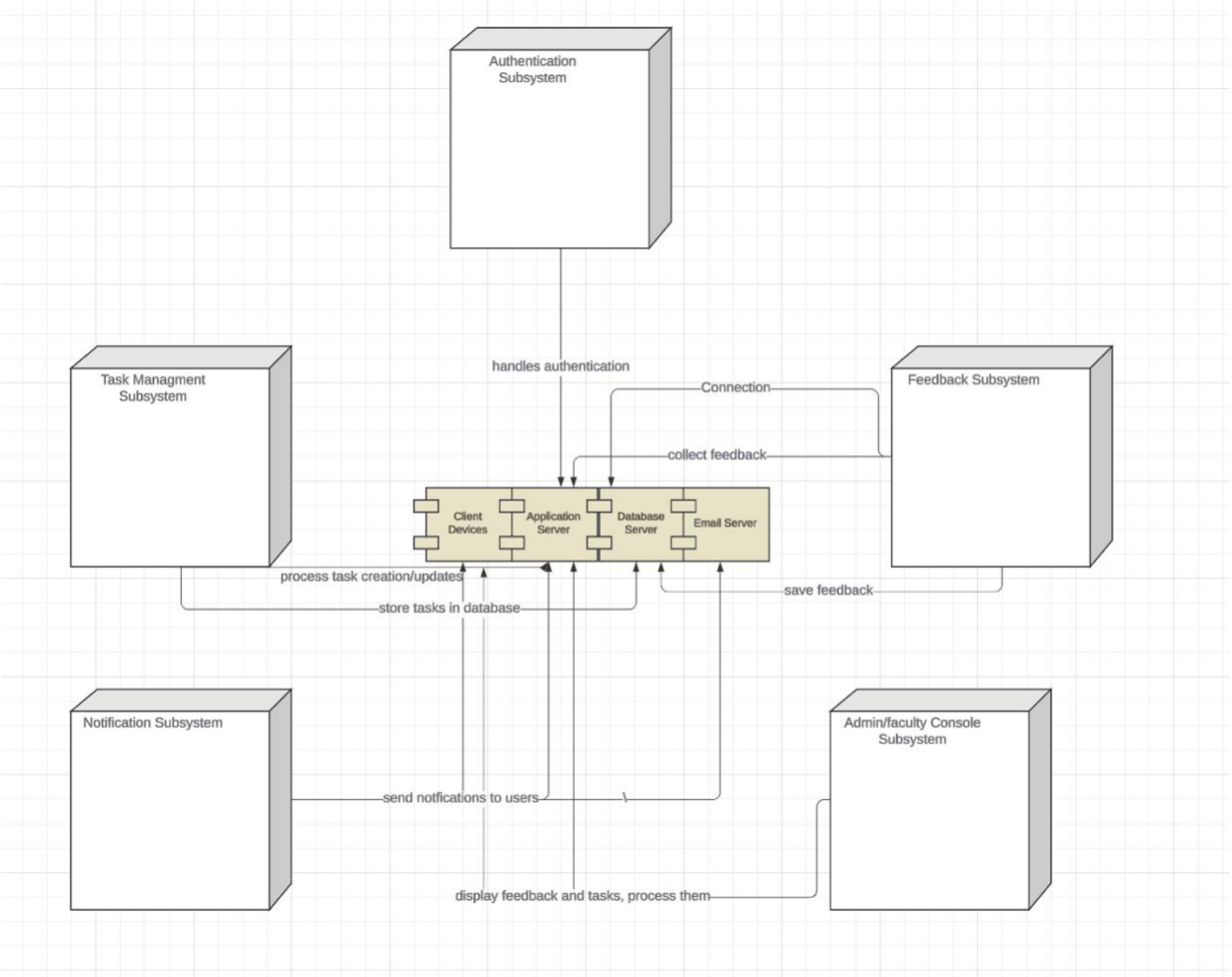


Fig. (2): UML diagram showing the mapping of subsystems to hardware components

3.4. Persistent data storage

Our CS ARCS system uses a relational database to store structured data. During development, we will use SQLite, and for production, we recommend PostgreSQL for scalability and robustness. We chose SQLite for development database because it is light-weight and file-based (does not require a server setup). Also, we choose PostgreSQL for production database because it is secure and supports advanced queries.

The system maintains the following main entities in the database:

1- User table: stores user details and role. It consists of the following fields:
   Id: identifier for each user
   Username: name of the user
   Email: email of the user
   Password: encrypted user password
   Role: junior/senior/faculty

2- Tasks table tracks tasks created by juniors and accepted bt seniors. It consists of the following fields:
    Id: identifier for each task
    Title: title of the task
    Description: brief description of the task
    Request_id: id of the junior who requested this task
    Accept_id: id of the senior who accepted this task
    Status: open/assigned/complete
    Time_created: Timestamp when the task was created
    Time_updated: Timestamp for the last update to the task

3- Feedback table: Stores feedback data submitted by juniors about seniors and tasks
    Id: identifier for each feedback
    task_id: id of the task the feedback is about
    user_id: id of the user the feedback is about
    rating: numeric rating (like the five-star system)
    comments: Additional comments about the task or the senior

4- Notifications Table: Logs notifications sent to users. It consists of the following fields:
    Id: identifier for each notification
    User_id: the id of the user the notification will be sent to
    Message: content of the notification
    Time_created: timestamp when the notification was sent

The database supports the following operations:

1- User management: create/retrieve/update/delete/assign role
2- Task management: add/delete/update/track status
3- Feedback management: collect/store/retrieve
4- Notification: log/retrieve

We also must ensure data integrity and reliability. We will achieve this through regular backups (for SQLite local file-based backups & cloud backups for PostgreSQL)

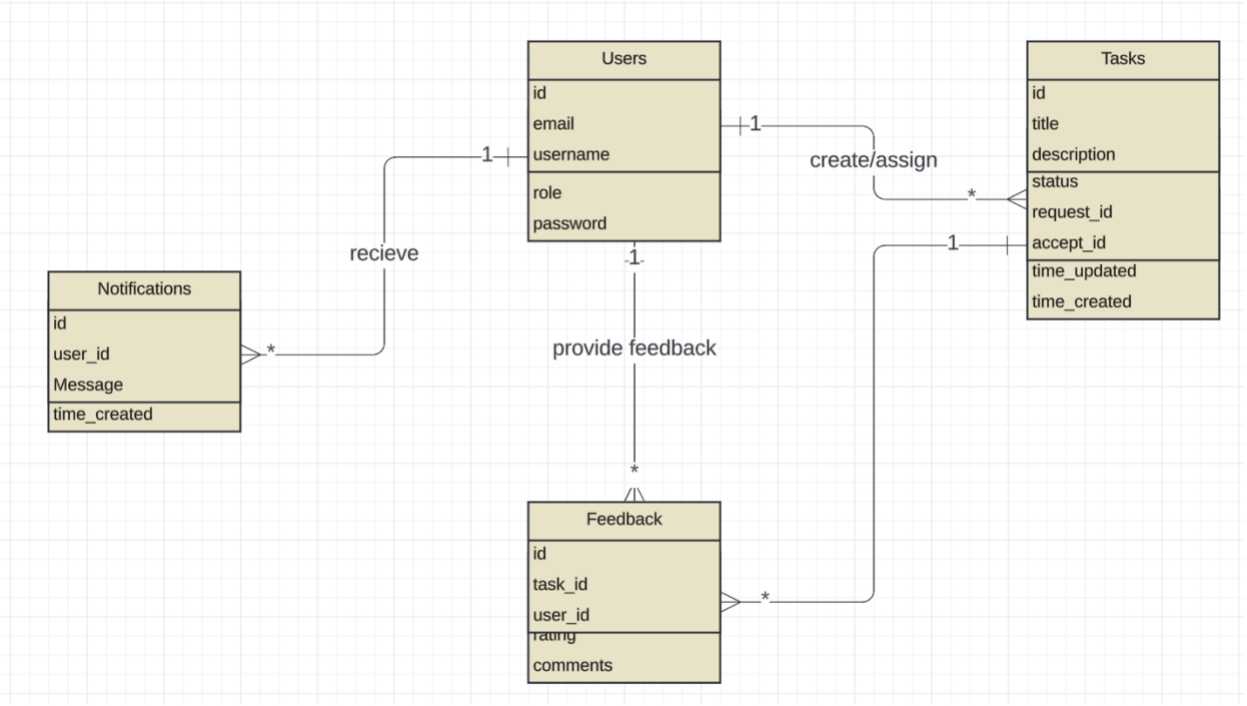


Fig.(3): ERD showing the relationships between the tables in the database system, where an arrow marked with 1→* means a one-to-many relationship

3.5. Network protocol

CS ARCS works over networks since it is a web-based application. It uses the following network protocols:

1. HTTPS: Enables communication between client devices (browsers) and the application server. Also, we chose the secure version of HTTP for security purposes.
2. TCP/IP: Handles reliable communication between the application server and the database server. It also guarantees data integrity and delivery.
3. SMTP: Used to send email notifications to users about task updates, reminders, and feedback.

3.6. Hardware requirements

CS ARCS requires low hardware requirements for clients since it is a web-based application. This achieves our goal of our program being reachable for everyone. The client devices' minimum hardware requirements are:

- Any device that runs a modern web browser (desktop, laptop, smartphone etc.)
- Modern web browser (e.g. Google Chrome, Mozilla Firefox)
- Any OS that can run a modern web browser (MacOS, iOS, Windows, Linux, etc.)
- A moderate internet speed (preferably above 2 Mbps)

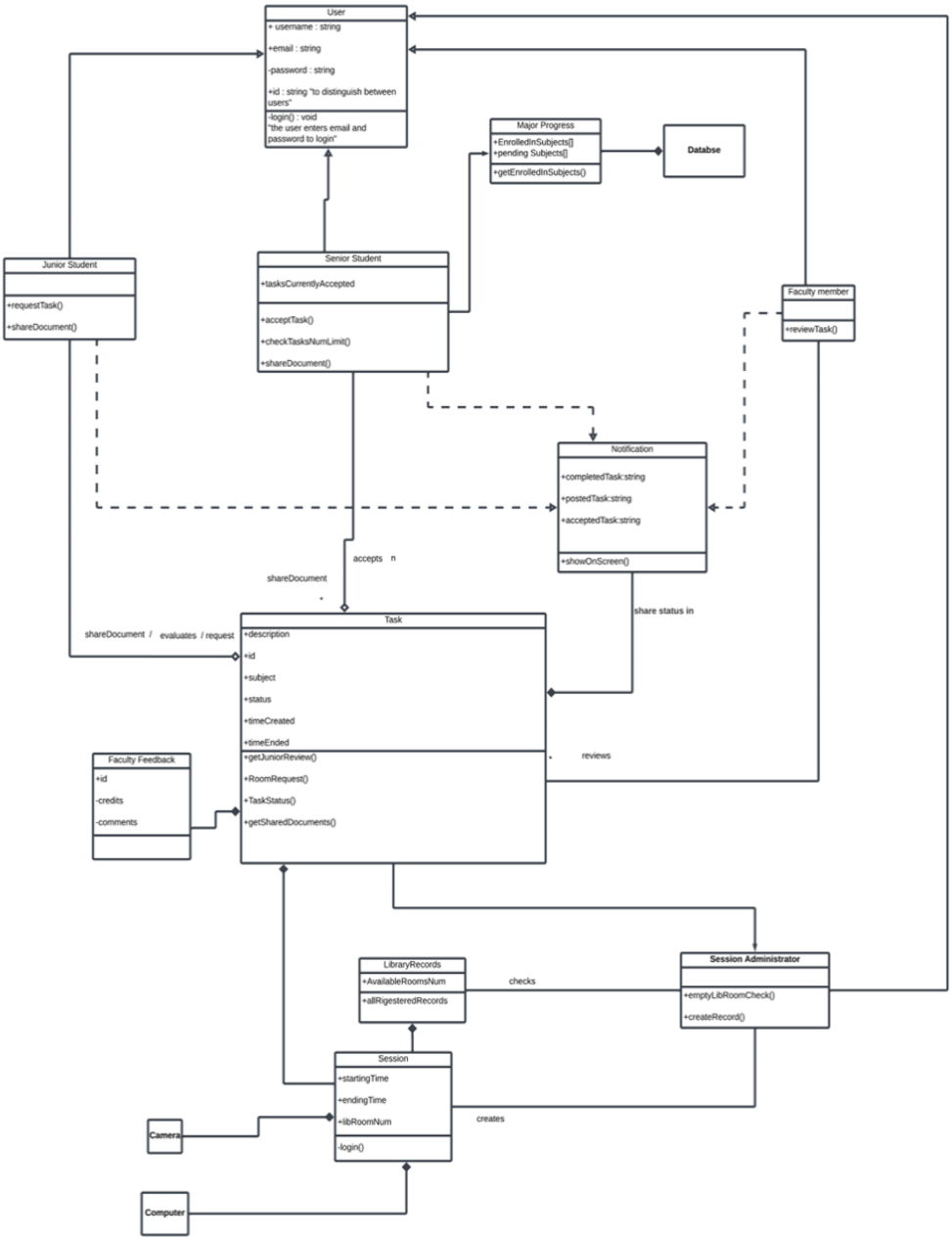
As for the application server's recommended hardware requirements:

- Quad-core processor
- Minimum of 8 GB RAM
- Minimum of 100 GB SSD storage
- Modern operating system

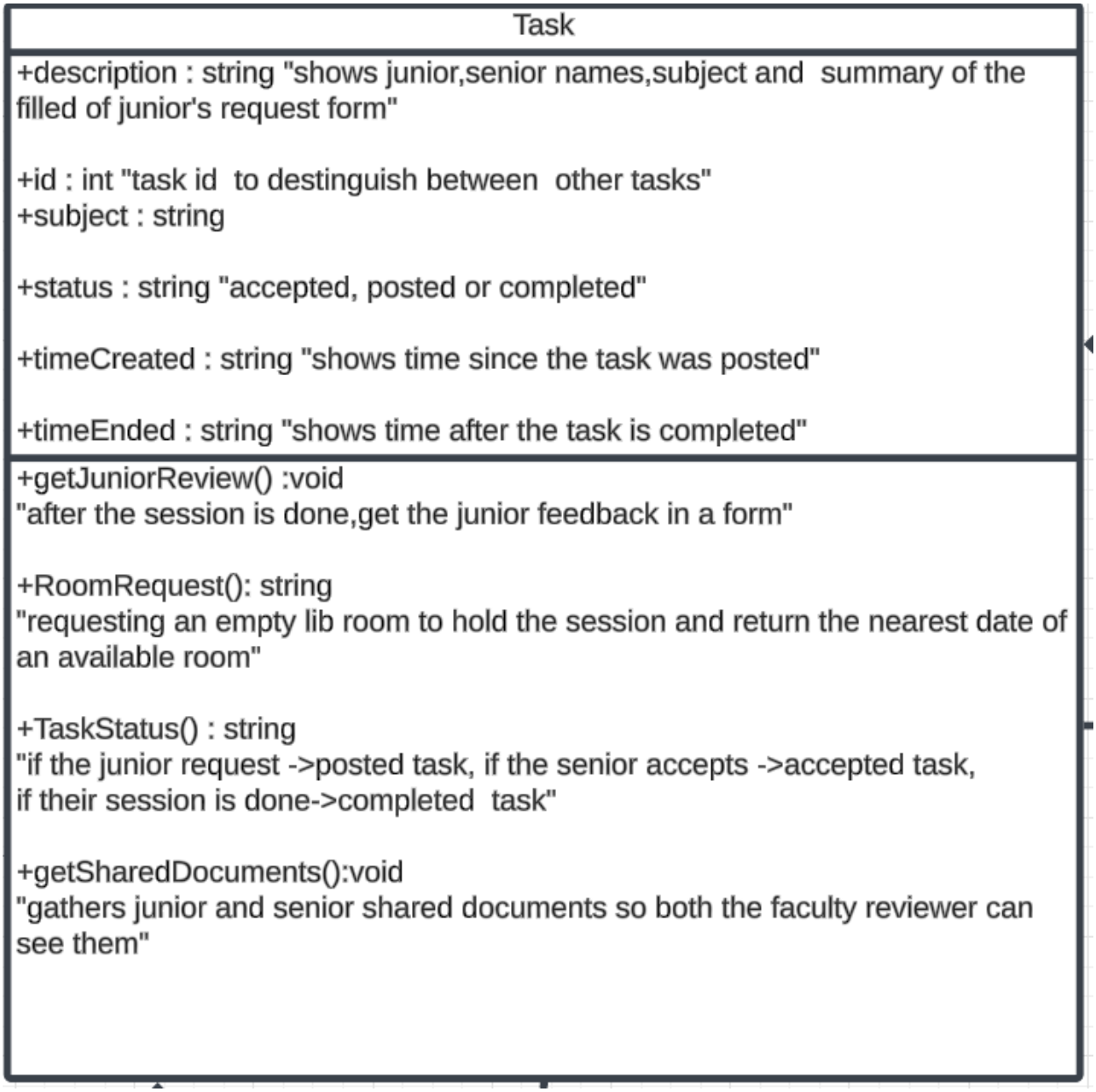
These choices guarantee a fast and reliable server that can serve multiple clients concurrently.

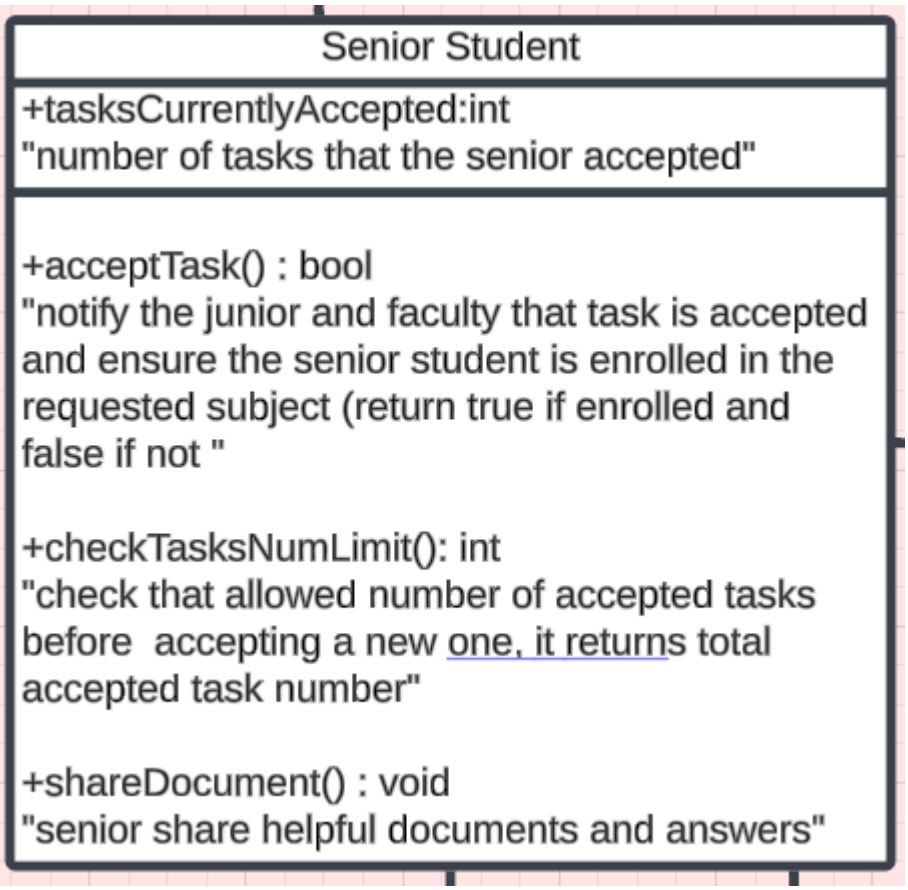
4. Class Diagram and Interface specification

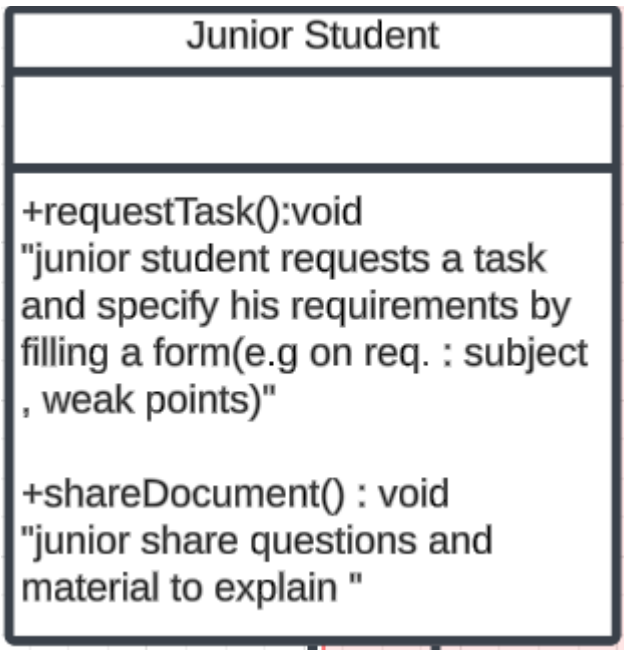
4.1. Class Diagram

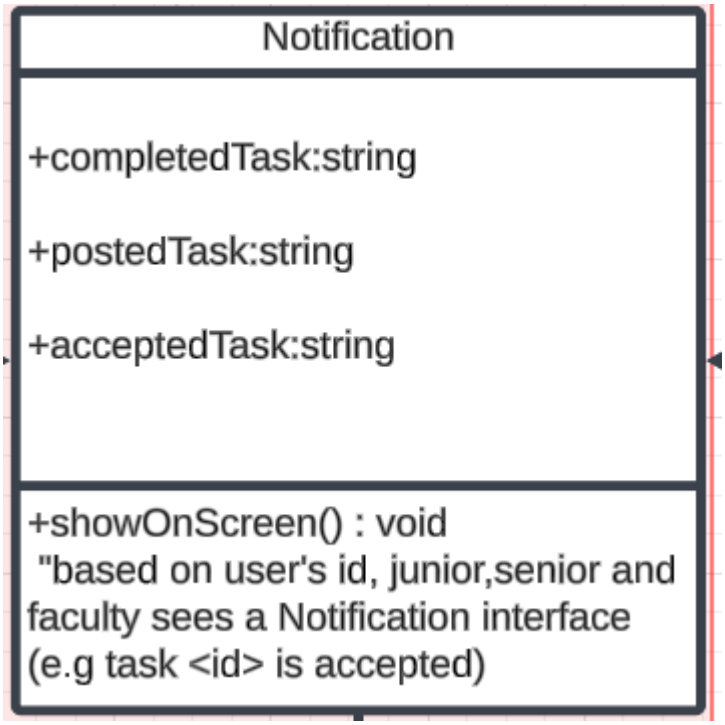


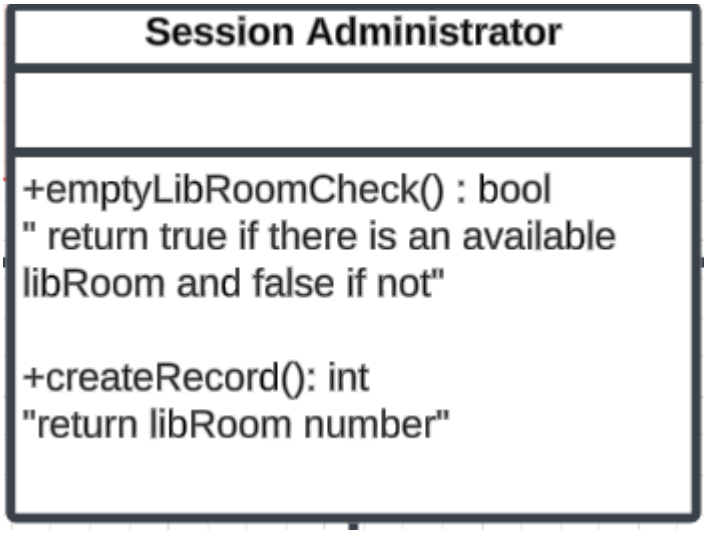
4.2. Data types and signatures

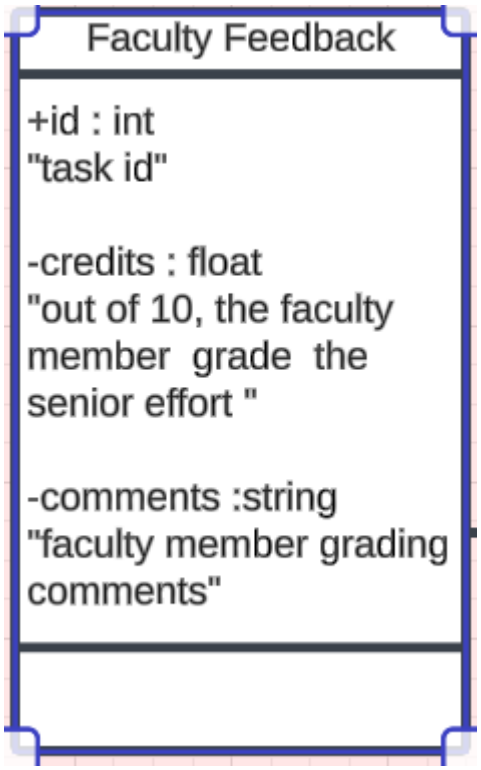



Senior Student
+tasksCurrentlyAccepted:int
"number of tasks that the senior accepted"
+acceptTask() : bool
"notify the junior and faculty that task is accepted and ensure the senior student is enrolled in the requested subject (return true if enrolled and false if not "
+checkTasksNumLimit(): int
"check that allowed number of accepted tasks before accepting a new one, it returns total accepted task number"
+shareDocument() : void
"senior share helpful documents and answers"


Junior Student
+requestTask():void
"junior student requests a task and specify his requirements by filling a form(e.g on req. : subject , weak points)"
+shareDocument() : void
"junior share questions and material to explain "


Notification
+completedTask:string
+postedTask:string
+acceptedTask:string
+showOnScreen() : void
"based on user's id, junior,senior and faculty sees a Notification interface (e.g task <id> is accepted)


Session Administrator
+emptyLibRoomCheck() : bool
" return true if there is an available libRoom and false if not"
+createRecord(): int
"return libRoom number"



5. Interaction Diagrams

Following are our top 6 use cases and their detailed sequence diagrams. We took the sequence diagrams that we designed in assignment 1 and developed them such that they satisfy the requirements of this assignment.

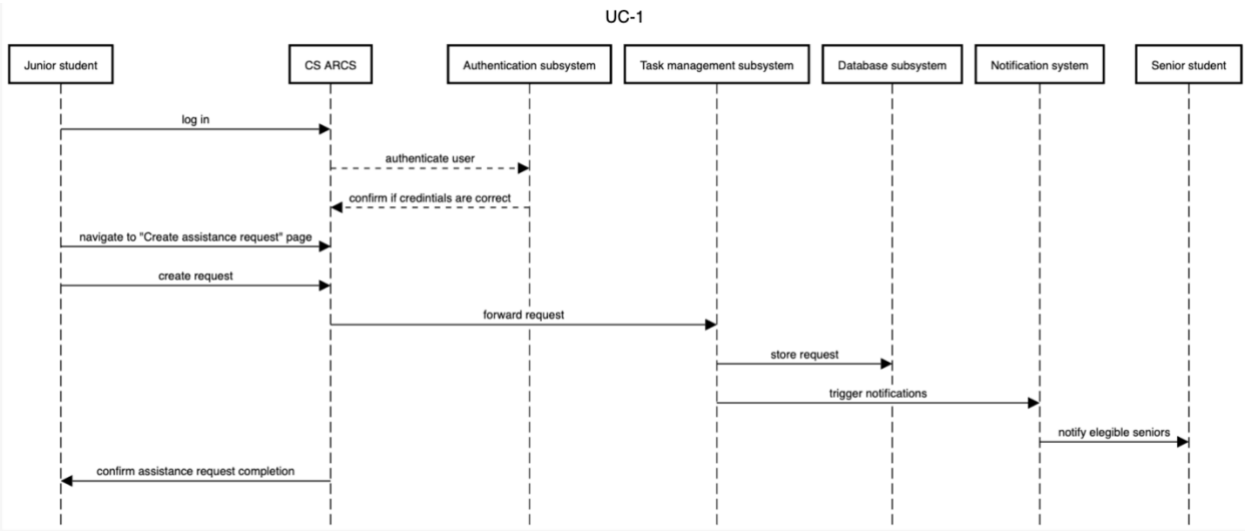


Fig. (4): detailed sequence diagram for UC-1 regarding creating assistance request

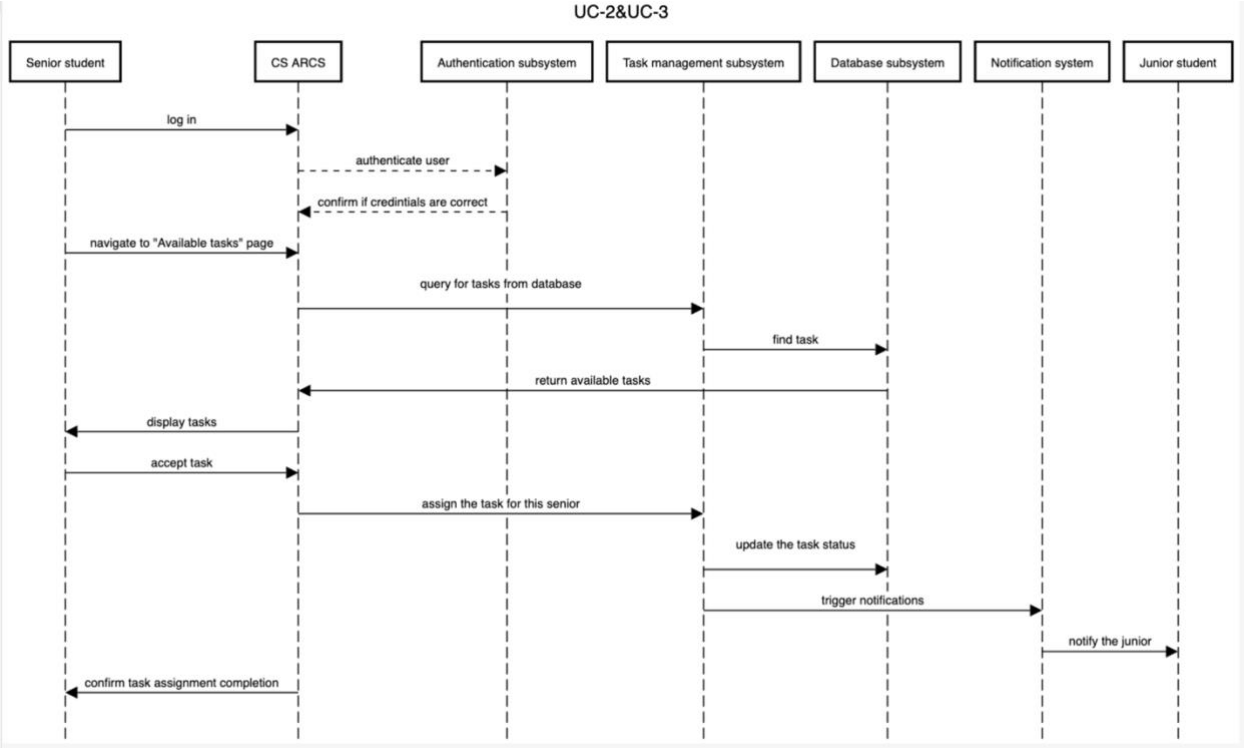


Fig. (5): detailed sequence diagram for UC-2 & UC-3 regarding navigating available tasks and accepting tasks

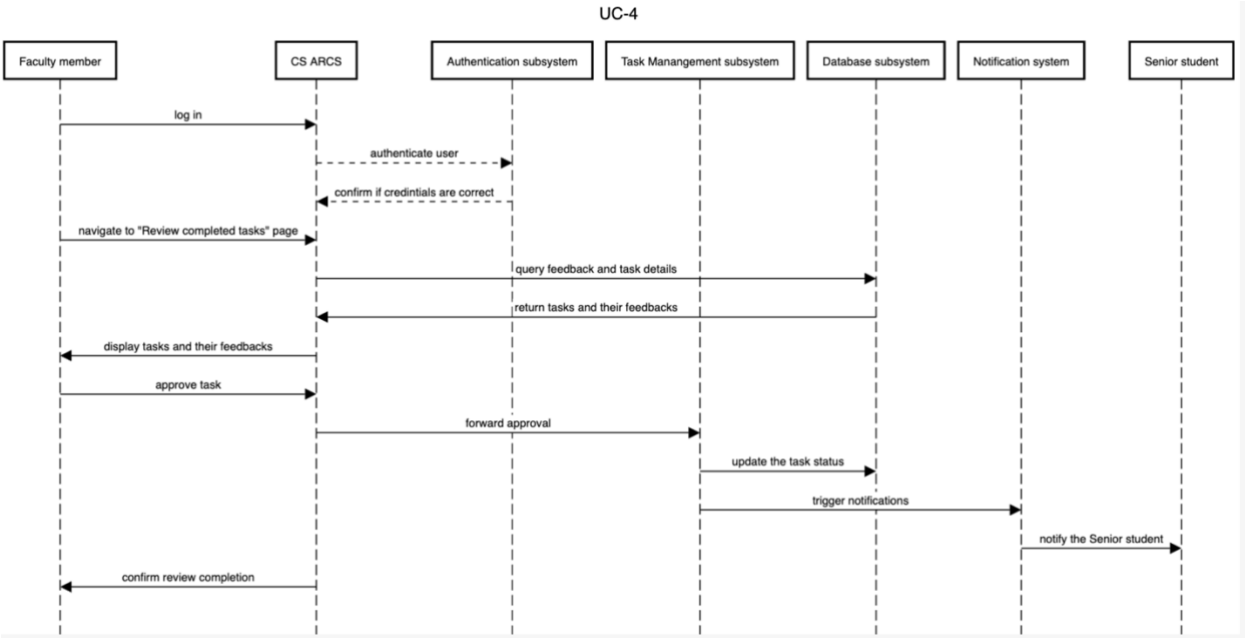


Fig. (6): detailed sequence diagram for UC-4 regarding reviewing feedback and awarding extra credit

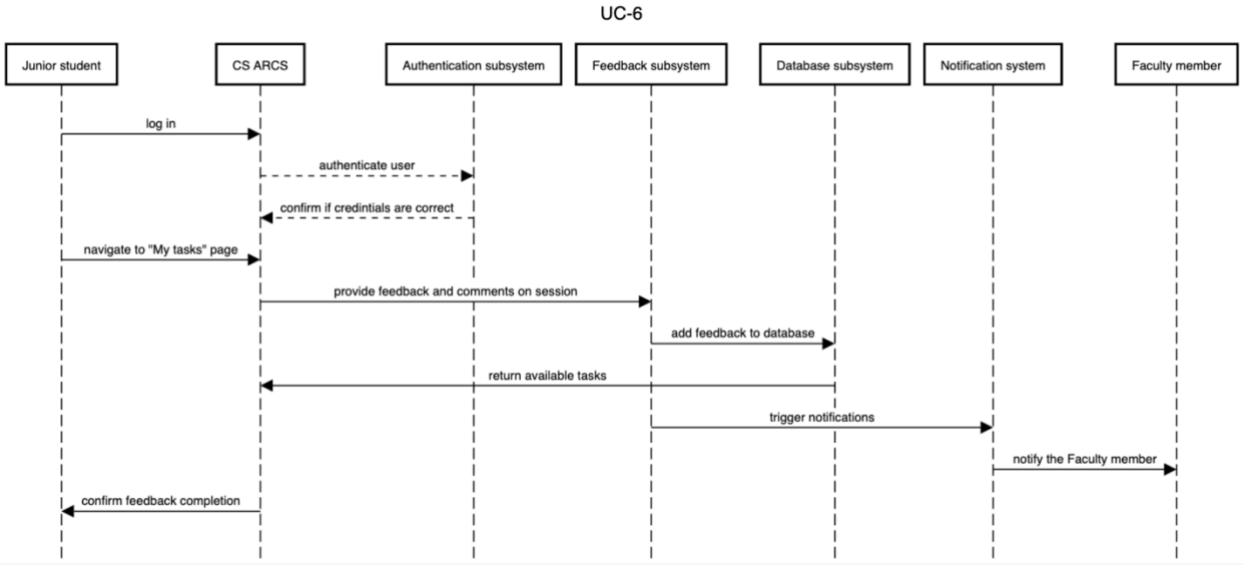


Fig. (7): detailed sequence diagram for UC-6 regarding providing feedback about completed tasks

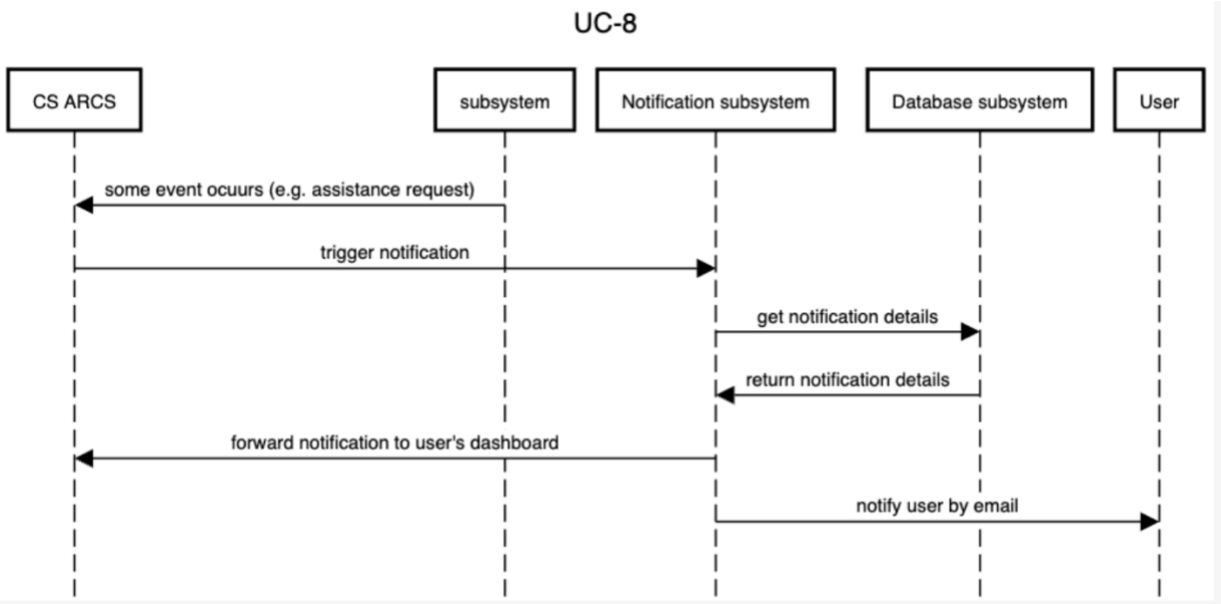


Fig. (8): detailed sequence diagram for UC-8 regarding notifying users

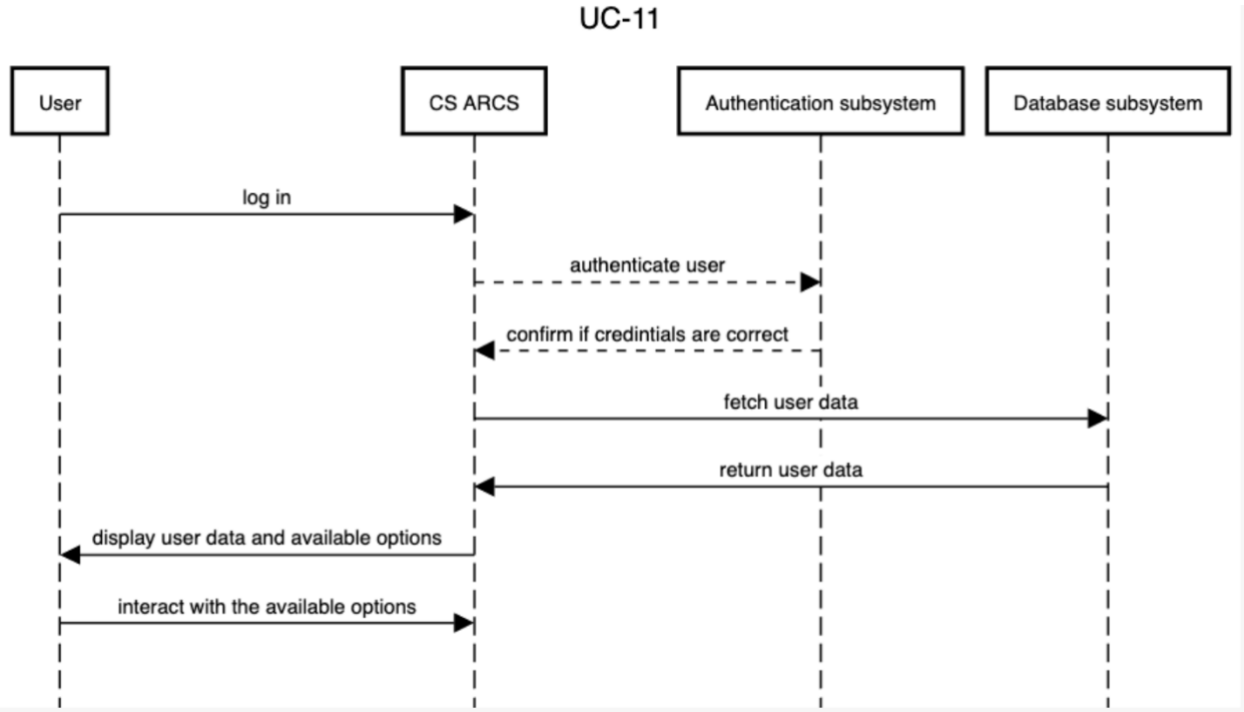
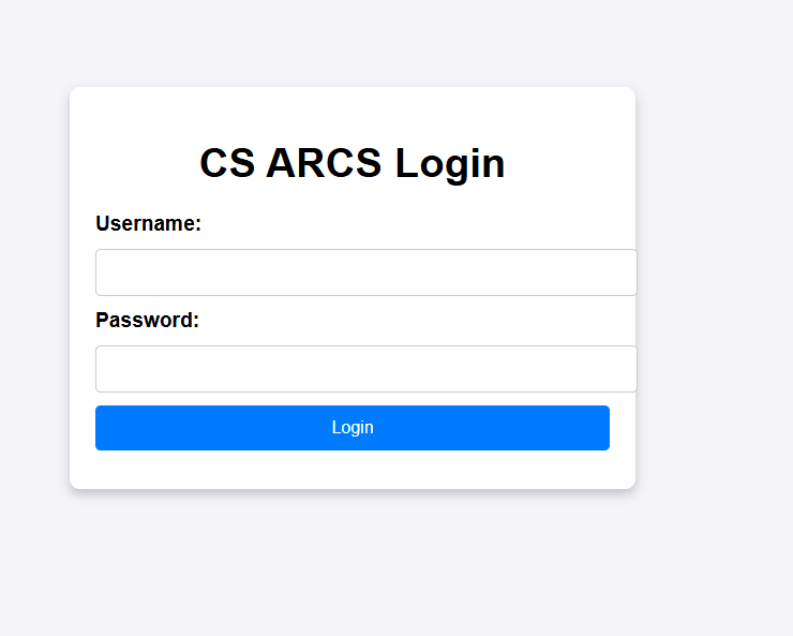
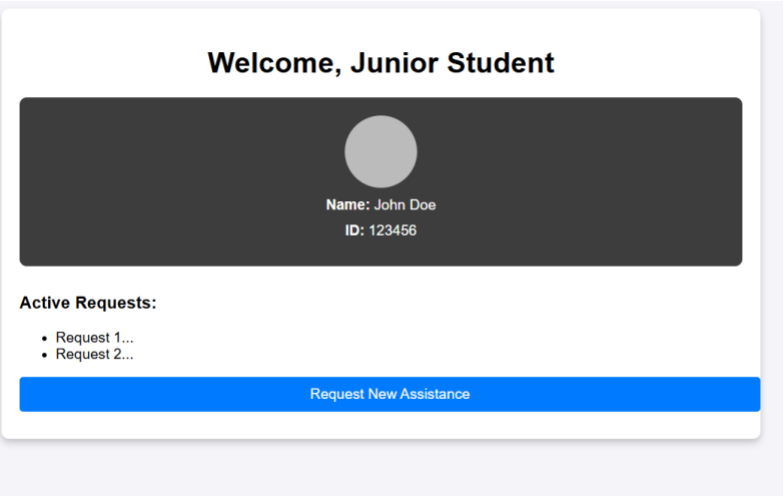
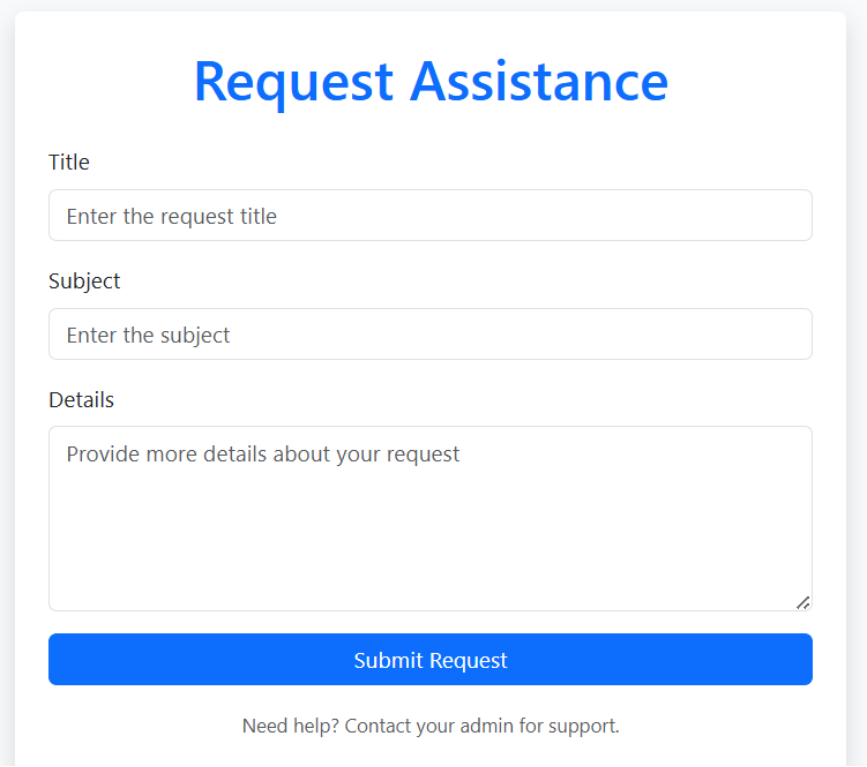


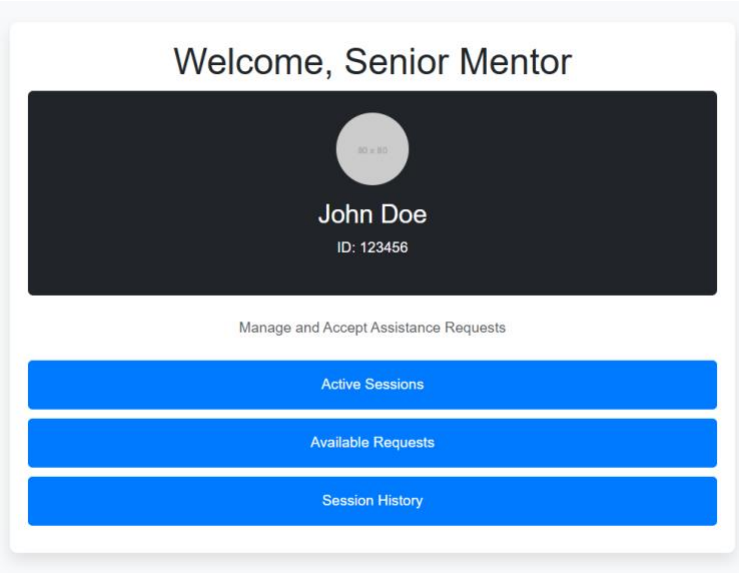
Fig. (9): detailed sequence diagram for UC-11 regarding logging in and authentication

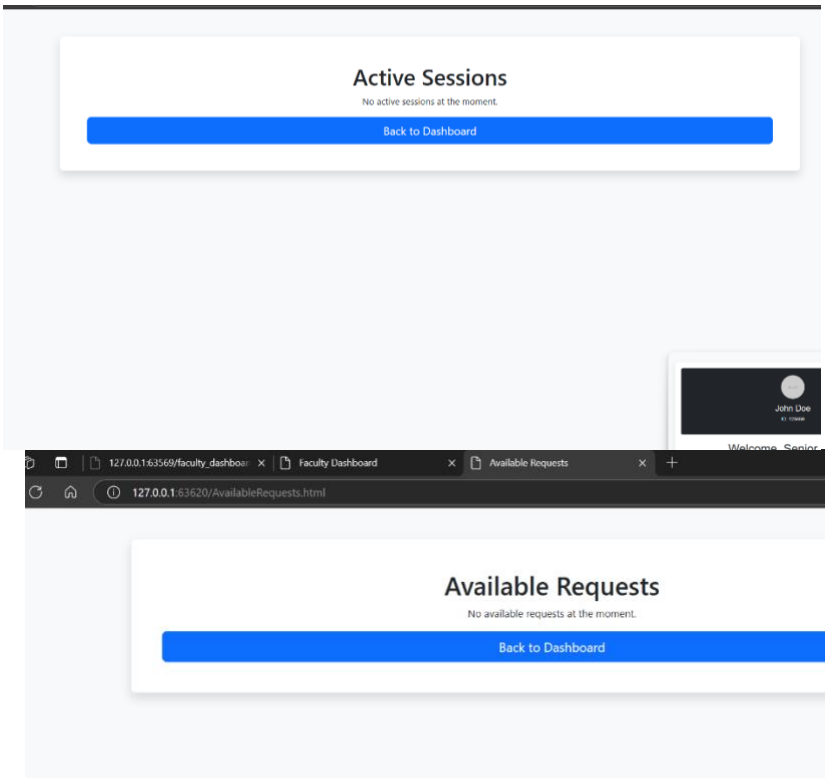
6. User Interface Design and Implementation

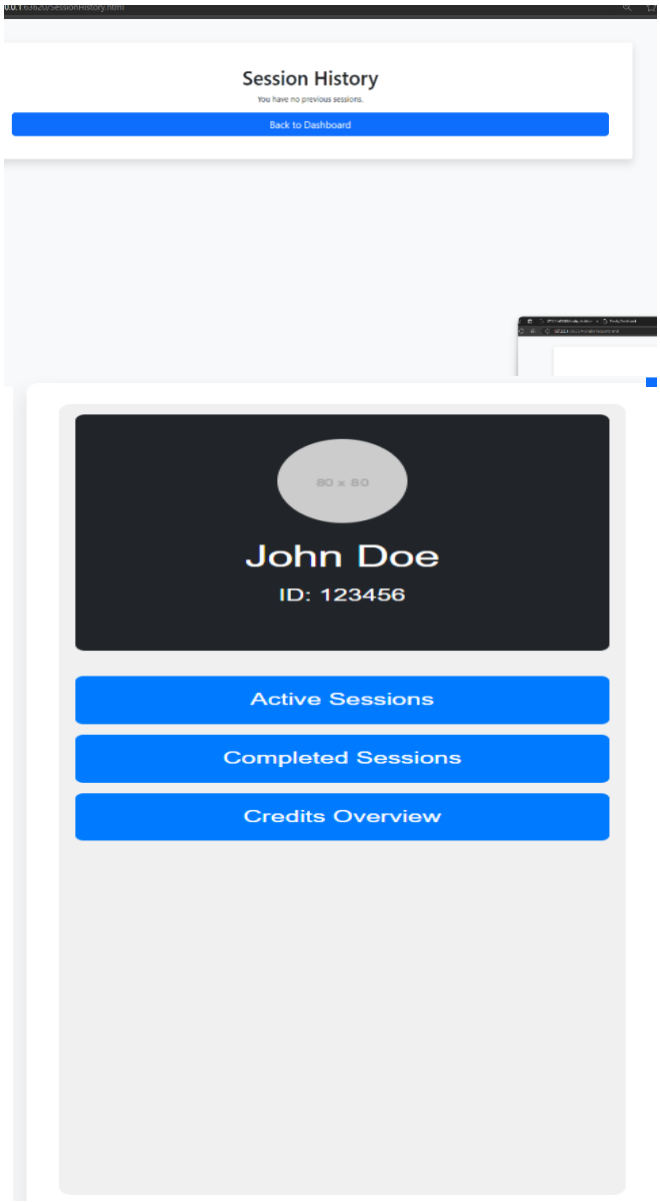



Welcome, Junior Student
Name: John Doe
ID: 123456
Active Requests:
Request 1...
Request 2...
Request New Assistance


Request Assistance
Title
Enter the request title
Subject
Enter the subject
Details
Provide more details about your request
Submit Request
Need help? Contact your admin for support.


Welcome, Senior Mentor
John Doe
ID: 123456
Manage and Accept Assistance Requests
Active Sessions
Available Requests
Session History


Active Sessions
No active sessions at the moment.
Back to Dashboard
John Doe
Faculty Dashboard
Available Requests
127.0.0.1:63620/AvailableRequests.html
Available Requests
No available requests at the moment.
Back to Dashboard


Session History
You have no previous sessions.
Back to Dashboard
80 x 80
John Doe
ID: 123456
Active Sessions
Completed Sessions
Credits Overview

7. Testing

First, we will start by addressing our test cases that we used for out program and the category they fall under:

1- **Creating Assistance Request**

Test ID	Description	Input	Expected Result	Actual results
TC-1	Create request with valid inputs	title="Math Help", subject="Algebra", details="Need guidance"	201 Created. Request created successfully.	pass
TC-2	Create request with missing required fields	title="", subject="Algebra"	400 Bad Request. Error: "All fields are required."	pass
TC-3	Create request without authentication	None	401 Unauthorized. Error: "Unauthorized."	fail

2- **Viewing Available Tasks**

Test ID	Description	Input	Expected Result	Actual results
TC-4	Fetch tasks relevant to the Senior Student's courses	Valid JWT for Senior role	200 OK. List of filtered tasks returned.	pass
TC-5	Fetch tasks with unauthorized role (e.g., Junior)	Valid JWT for Junior role	403 Forbidden. Error: "Access denied for role: junior."	pass
TC-6	Fetch tasks without providing a token	None	401 Unauthorized. Error: "Unauthorized."	pass

3- **Accept Task**

Test ID	Description	Input	Expected Result	Actual results
TC-7	Accept a valid task	Valid task_id=1, Senior JWT	200 OK. Task accepted successfully.	Pass
TC-8	Accept a task that doesn’t exist	Invalid task_id=999, Senior JWT	404 Not Found. Error: "Task not found."	pass
TC-9	Accept a task with unauthorized role	Valid task_id=1, Junior JWT	403 Forbidden. Error: "Access denied for role: junior."	pass

4- Approve Task Completion

Test ID	Description	Input	Expected Result	Actual results
TC-10	Approve task with valid input	Valid task_id=1, Faculty JWT	200 OK. Task approved successfully.	pass
TC-11	Approve a task that doesn’t exist	Invalid task_id=999, Faculty JWT	404 Not Found. Error: "Task not found."	fail
TC-12	Approve task with unauthorized role	Valid task_id=1, Junior JWT	403 Forbidden. Error: "Access denied for role: junior."	pass

5- Provide Feedback on Assistance

Test ID	Description	Input	Expected Result	Actual result
TC-16	Fetch notifications for a valid user	Valid JWT	200 OK. Notifications fetched successfully.	Pass
TC-17	Fetch notifications for a non-existent user	Invalid `user_id=999`	404 Not Found. Error: "No notifications found."	Pass
TC-18	Fetch notifications without authentication	None	401 Unauthorized. Error: "Unauthorized."	Pass

6- Receive notifications

Test ID	Description	Input	Expected Result	Actual result
TC-16	Fetch notifications for a valid user	Valid JWT	200 OK. Notifications fetched successfully.	Pass
TC-17	Fetch notifications for a non-existent user	Invalid `user_id=999`	404 Not Found. Error: "No notifications found."	Pass
TC-18	Fetch notifications without authentication	None	401 Unauthorized. Error: "Unauthorized."	Pass

7- Logging in

Test ID	Description	Input	Expected Result	Actual result
TC-19	Log in with valid credentials	email="user@example.com", password="correctpassword"	200 OK. Token generated successfully.	Pass

Test ID	Description	Input	Expected Result	Actual result
TC-20	Log in with invalid credentials	email="user@example.com", password="wrongpassword"	401 Unauthorized. Error: "Invalid credentials."	Pass
TC-21	Log in with missing required fields	email="", password=""	400 Bad Request. Error: "Email and password are required."	Pass

Now let us introduce the traceability matrix to map the use cases to the test cases (note: go back to assignment 1 to see the fully dressed use cases, their IDs and descriptions):

ID	UC-1	UC-2	UC-3	UC-4	UC-5	UC-6	UC-7	UC-8	UC-9	UC-10	UC-11	UC-12
TC-1	X											
TC-2	X											
TC-3	X											
TC-4		X										
TC-5		X										
TC-6		X										
TC-7			X									
TC-8			X									

TC-9			X									
TC-10				X								
TC-11				X								
TC-12				X								
TC-13						X						
TC-14						X						
TC-15						X						
TC-16								X				
TC-17								X				
TC-18								X				
TC-19									X			
TC-20									X			
TC-21									X			

8. Risk Management

Update the main risks of your project (submitted in part 1). Include Risk, Probability, and Effects (as discussed in class). Remember that you have completed the initial stages of your project, now, new risks might have surfaced, and old risks might have become less of a risk. Risks should always be updated. Clearly identify which risks have changed (modified/deleted), and which risks have been added.

Risk	Probability	Effects	Change in List (deleted/modified/added)

Major sheet modification	1	Some subjects in the old Major sheet could be combined or deleted in the new major sheet resulting in a gap between senior and junior batches.	added
Library rooms are temporary closed	2	The system's restriction on library rooms as the only permitted spaces has resulted in no sessions taking place.	added
Cameras and computers in all library rooms are not working	3	There is no way to verify attendance in the session.	added
Outdated database	3	It may take time for recent junior students to be added to the system, and some senior students might face rejection when attempting to accept tasks due to newly enrolled tasks not being visible in the system.	added

Challenge	Probability	Effect
dishonest feedback	4	System goal is unachieved due to manipulation/ faculty doubt its credibility/ unfair advantages for some students, while others may be undervalued.
library rooms number are small causing late session appointments	3	Late appointments may result in junior students turning to ask other students directly, bypassing the system.

9. Project Plan

Update the project plan for your project (submitted in part 1).

We extended the design phase by a week.

Week	Milestones	Lamiaa	Ahmed	Seba
1	Research	Research and understand project		
1	Gathering requirements	documenting, and understanding the needs and goals of a project		
2-3	System design	Design UI	Create system architecture and diagrams	Design data flow
4-6	Development phase 1	Develop UI components	Develop task management functionality	Develop notification system
7-9	Development phase 2	Develop feedback system	Integrate all components	
10	Testing phase 1	Write test case	Implement test cases	Testing and fixing bugs

10. References

- Hipp, D. R. (n.d.). *SQLite documentation*. SQLite. Retrieved from
- https://www.sqlite.org/docs.ibhtml
- PostgreSQL Global Development Group. (n.d.). *PostgreSQL documentation*. PostgreSQL. Retrieved from https://www.postgresql.org/docs/
- IBM. (n.d.). *What is an entity relationship diagram (ERD)?* IBM Think. Retrieved from https://www.ibm.com/think/topics/entity-relationship-diagram
- SequenceDiagram.org. (n.d.). *Online Sequence Diagram Tool*. Retrieved from https://sequencediagram.org/
- Microsoft Support. (n.d.). *Create a UML Sequence Diagram*. Retrieved from https://support.microsoft.com/en-us/office/create-a-uml-sequence-diagram-c61c371b-b150-4958-b128-902000133b26