CS 520: Project Documentation Template

Spring 2025, University of Massachusetts Amherst

Sameen Shaik/sshaik03 Aareb Chowdhury /akchdhry

Ahmed Abdul Rahman / ahmedar04 Yusuf Raza / Yraz60

Team Name: FBI

GoogleDrive Project Document Link: https://docs.google.com/document/d/1zTc_u19G92xD_WrukgfTl76ux1WHNNAakzQeqdsNWNA/edit?usp=sharing

Github Repo Link: https://github.com/sshaik03/Online_LMS

Video Demo: https://drive.google.com/file/d/1ZhxdqVWRnpMY6jFSHEvMaJnCy6SOYVkE/view?usp=drivesdk

1. Requirements

1.1. Overview

Our Online Learning Management System (LMS) initiative, called the LearningHub, aims to change education delivery and management by offering a centralized, digital platform for educators and students. The system intends to simplify course creation, content delivery, assignment administration, and instructor-student contact in an interactive and user-friendly environment.

Objectives:

Streamline Educational Processes: To reduce administrative costs in typical educational environments, automate simple operations including course enrollment, assignment submission, and grading.

Enhance Engagement: Encourage interactive learning by using discussion boards and live video integration to create a collaborative environment between students and instructors.

Improve Accessibility: Make course materials and learning tools available at all times and from any device, allowing for flexible study schedules and remote education.

Intended Audience: The LMS is designed for academic institutions, business training programs, and any organization wishing to provide online learning. It is aimed at instructors who require a powerful tool to manage courses and their classes, as well as students who are seeking an accessible and engaging platform to participate in their education.

Motivations: This project, driven by the rapid change to digital education in today's age, addresses the rising demand for scalable and efficient online learning environments. We are motivated to improve education outcomes with features and tools that will facilitate students' learning and instructors' teaching.

.

1.2. Features

Authentication and Role Management: A secure login for users with particular roles. Students and instructors will have unique pages specific to their respective roles, ensuring that each user only sees functionality and content that is relevant to them.

Course Content Management: A complete content management system enables teachers to create, organize, and update their courses. With unique course information like IDs and modular course frameworks, the application will allow for dynamic learning experiences.

Assignment Submission and Grading System: An integrated system that allows students to submit assignments online while teachers analyze, grade, and provide extensive feedback. It also tracks submission deadlines and supports a variety of assignment file formats.

Discussion Forums with Real-Time Collaboration: Interactive discussion forums with posts, likes, and replies encourage collaboration between students, as well as instructor feedback. This feature facilitates inter-student conversations, Q&A sessions, and live messaging, hence increasing engagement and peer-to-peer learning, an aspect essential for student success.

Analytics and Reporting Dashboard: A built-in dashboard that provides real-time insights into course engagement and student performance. Instructors can view detailed reports on attendance, assignment scores, and participation metrics, while students can view notifications for their soon-due assignments, quizzes, and discussion replies.

1.3. Functional Requirements (Use cases)

1. User Authentication and Role Management
    a. Use Case: In order to ensure that only authorized users have access to the proper areas of the system, as an instructor, I wish to have the ability to create, modify, and edit my courses and assignments without students having access to those pages
    b. Description: An instructor must create courses and assignments for students. Based on their designated role, each user is authenticated at login and is subsequently given access to courses and administrative capabilities, or a lack thereof.
2. Account Registration and Login
    a. Use Case: As a student, I'd like to be able to create a secure account in the LearningHub system and use it for login to complete my courses and assignments. As an instructor, I'd like a secure account for use to login and manage courses and grades.
    b. Description: Creating an account will save the credentials into the database, using hashing for the passwords so that everything is stored securely. Users will then be able to use their accounts to login at any time on any device.
3. Course Creation and Management
    a. Use Case: In order to give well-structured course material, I want to design and oversee courses as an instructor, including adding course titles, descriptions, and multimedia materials.
    b. Description: Instructors have the ability to arrange modules or lessons, edit course information, and create new courses. To keep the course catalog up to date, they can also eliminate courses if needed.

4. Assignment Submission and Grading
    a. Use Case: In order to expedite the evaluation process, I would like to submit projects online as a student and grade them and offer feedback as an instructor.
    b. Description: Instructors have the ability to monitor submissions, assign grades, and post feedback, while students can upload files or text responses to assignments. The system keeps track of submission statuses and deadlines.
5. Course Enrollment
    a. Use Case: In order to access the instructional materials and take part in class activities, I, as a student, want to be able to sign up for classes.
    b. Description: The system will allow students to enroll in classes via use of a course ID. If a student satisfies the requirements, they can enroll, which also modifies their profile and course roster.
6. Discussion Forums
    a. Use Case: In order to interact with peers and teachers, exchange ideas, and ask questions, I would like to join discussion boards pertaining to my courses and assignments as a student.
    b. Every course has a forum where students can create discussion posts, reply to other postings, and get alerts when new ones are published. Features for moderation guarantee that conversations stay productive.
7. Deleting Discussion Posts
    a. Use Case: As a student, if I create a discussion post and ask a question that I find the answer to, or accidentally make a post public, I want to be able to delete my post or reply to hide it from the rest of the class.
    b. Each discussion post or reply has a delete option, which will remove it from view and remove it from the database as well.
8. Content Scheduling and Notifications
    a. Use Case: In order to keep students informed and involved, as an instructor, I would like to plan when the course materials would be released and automatically alert them of impending due dates or live sessions.
    b. Description: The system notifies enrolled students via in-app notifications or automated emails when instructors schedule assignments, live events, or material postings.
9. Feedback and Evaluation
    a. Use Case: In order to continuously enhance the course material, I, as a student, would like to offer feedback on my classes, and as a teacher, I would like to see and act upon that feedback.
    b. Students can review and rate the material after finishing a course or module. Teachers can modify their teaching strategies and course materials as necessary thanks to the comprehensive feedback and summary reports they receive.

1.4. Non-Functional Requirements

1. Performance: Even during busy times, the system should be able to accommodate up to 200 users at once with page loads taking less than two seconds on average. This guarantees that there are no discernible delays in the real-time interaction and access to course materials between teachers and students.

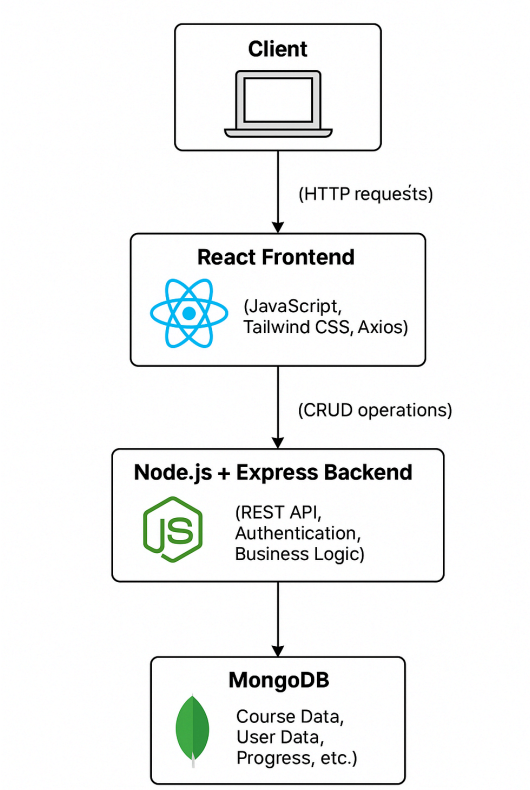
2. Security: The platform needs to have robust authorization and authentication procedures in place, and all data transfers must be SSL/TLS secured. To prevent unwanted access, sensitive user data, including passwords and private information, must be safely stored (for example, by hashing passwords).
3. Reliability: Regular automated backups, thorough error reporting, and failover procedures should all help the LMS reach at least 99.5% uptime. This guarantees that users may depend on the system during crucial learning times and reduces service interruptions.
4. Usability: The interface needs to follow accessibility guidelines and be simple to use on a range of devices, including desktops, tablets, and smartphones. This guarantees that everyone can utilize the system efficiently, on any device of choice.

1.5. Challenges & Risks

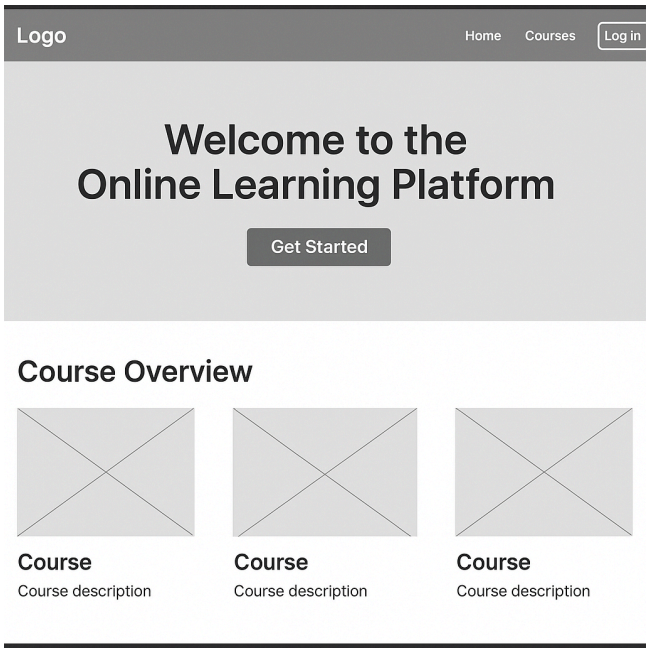
1. Data security and privacy: It's imperative to safeguard private student and teacher data. Maintaining secure authentication, appropriate encryption, and adherence to data protection laws (like the GDPR) is difficult and necessitates constant observation to stop illegal access or data breaches.
2. Scalability and Performance: The system must continue to respond quickly and steadily as the number of concurrent users increases, particularly during busy periods like live sessions or assignment deadlines. One major risk is creating a scalable design that can support growing loads without experiencing performance issues.
3. Integration with External Tools: The LMS may incorporate analytics software, cloud storage, or video conferencing. It can be difficult and potentially create more points of failure to manage these interconnections while maintaining compatibility, security, and dependable performance.
4. Usability and Adoption: It can be difficult to serve a wide range of users with different technical proficiency levels. Any complexity in the user interface could prevent instructors and students from adopting the system, which needs to be simple and easy to use. Another level of complexity is added by making sure accessibility standards are followed.
5. System Uptime and Reliability: Availability is important, especially during live tests or classes. To reduce downtime, the system needs to have strong backup, recovery, and failover procedures in place. Any disruptions have the potential to seriously impair learning.

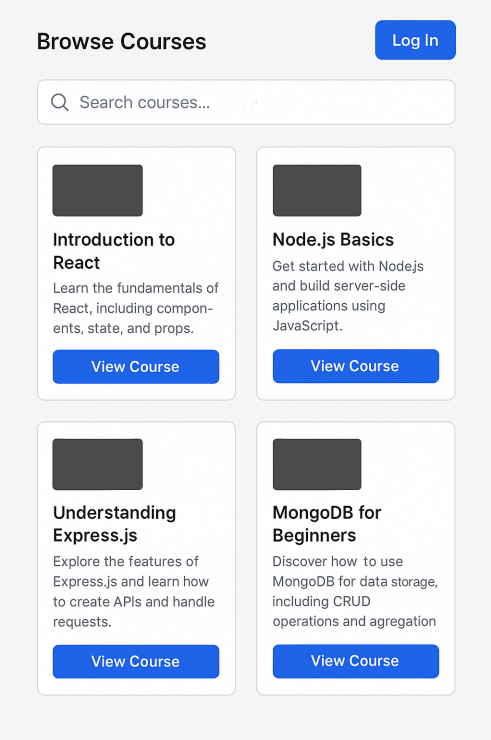
2. Design

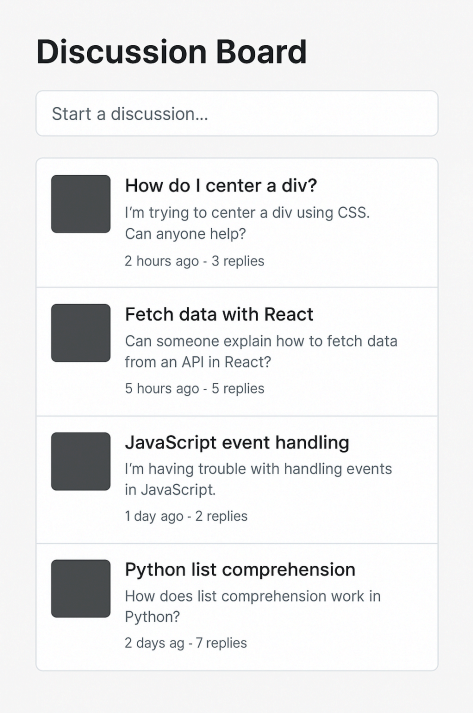
2.1. Architecture Diagram


Client
(HTTP requests)
React Frontend
(JavaScript,
Tailwind CSS, Axios)
(CRUD operations)
Node.js + Express Backend
(REST API,
Authentication,
Business Logic)
MongoDB
Course Data,
User Data,
Progress, etc.)

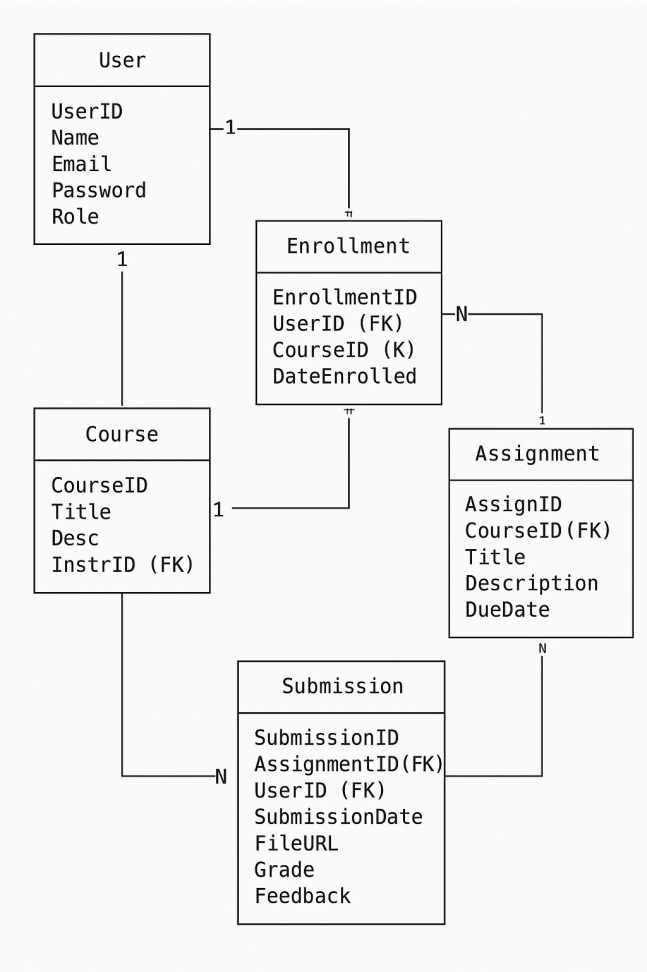
2.2. UI Design


Logo
Home
Courses
Log in
Welcome to the
Online Learning Platform
Get Started
Course Overview
Course
Course description
Course
Course description
Course
Course description


Browse Courses
Log In
Search courses...
Introduction to React
Learn the fundamentals of React, including components, state, and props.
View Course
Node.js Basics
Get started with Node.js and build server-side applications using JavaScript.
View Course
Understanding Express.js
Explore the features of Express.js and learn how to create APIs and handle requests.
View Course
MongoDB for Beginners
Discover how to use MongoDB for data storage, including CRUD operations and agregation
View Course


Discussion Board
Start a discussion...
How do I center a div?
I'm trying to center a div using CSS. Can anyone help?
2 hours ago - 3 replies
Fetch data with React
Can someone explain how to fetch data from an API in React?
5 hours ago - 5 replies
JavaScript event handling
I'm having trouble with handling events in JavaScript.
1 day ago - 2 replies
Python list comprehension
How does list comprehension work in Python?
2 days ag - 7 replies

2.3. Data Model



2.4. Tech Stack With Justification

Backend Technologies

- **Node.js with Express.js**: Selected for rapid API development with excellent middleware support, non-blocking I/O operations, and scalability for handling concurrent requests from multiple users, which is critical during high-traffic periods like assignment submissions.

Frontend Technologies

- **React.js with TypeScript**: Selected for component-based UI architecture with strong typing, enabling faster development, better code quality, and fewer runtime errors. The component reusability facilitates consistent UX across different parts of the application.

- **React Router**: Implemented for client-side routing that enables smooth navigation between application views without full page reloads, creating a responsive single-page application experience.

- **React Context API**: Used for state management across components, providing an efficient way to share authentication state and user data throughout the application without prop drilling.

- **Tailwind CSS**: Chosen for utility-first CSS framework that accelerates UI development with responsive design patterns. The low-level utility classes enable custom designs while maintaining consistency across the platform.

Data Storage & Infrastructure

- **MongoDB**: For scalable cloud storage of course materials and assignment submissions, providing high availability and durability for educational content with support for large files.

Integration & Development Tools

- **RESTful API Architecture**: Designed for standardized communication between client and server, enabling future integrations with external services and tools.

- **Environment Configuration**: Using dotenv for managing environment-specific configurations, allowing for seamless transitions between development, testing, and production environments.

- **CORS**: Implemented for secure cross-origin requests between frontend and backend services, maintaining proper security boundaries.

2.5. Challenges & Risks

Scalability & Performance:

Challenge: The system must support a large number of concurrent users, particularly during peak times such as live sessions without performance degradation.

Mitigation: Design a scalable architecture using load balancing, caching mechanisms, and optimized database queries to maintain fast response times.

Data Security & Privacy:

Challenge: Protecting sensitive user data (such as login credentials, personal information, and academic records) is vital, because the platform will store information about students and instructors.

Mitigation: Implement robust security measures including SSL/TLS encryption, secure authentication protocols, data anonymization, and regular security audits to safeguard data.

Integration with Third-Party Services:

Challenge: Integrating external APIs (e.g., video conferencing tools, payment systems, or cloud storage services) may introduce compatibility issues and add complexity to the system.

Mitigation: Choose well-documented and widely supported third-party services, conduct thorough integration testing, and implement fallback mechanisms in case of service disruptions.

User Experience & Accessibility:

Challenge: Designing an intuitive and accessible user interface for a diverse audience, including users with varying technical skills and accessibility needs will be difficult.

Mitigation: Follow established UI/UX best practices, adhere to accessibility standards, and conduct regular user testing to ensure the interface meets the needs of all users. Have potential users test.

Maintenance and Future Scalability:

Challenge: As the platform evolves and the user base grows, maintaining the system and scaling it can be challenging.

Mitigation: Use good design principles, enforce coding standards and regular code reviews, and maintain comprehensive documentation to ease future updates and scaling efforts.

3. Implementation

3.1. Work Plan

Overall Project Timeline

Sprint 1 (until Tuesday 04/02/25):

Focus: Develop a mock front-end with basic pages (landing page, navbar, login page, course overview page) and set up initial project documentation.

Goals:

Develop wireframes and UI mockups.

Build a simple static front-end prototype demonstrating basic navigation and page layouts.

Establish project repository, documentation, and initial coding standards.

Sprint 2 (2-3 Weeks):

Focus: Build core use-case functionality; implement backend REST APIs and database schema; start to integrate front-end with back-end.

Sprint 3 (Final Sprint – 4 Weeks):

Focus: Refine user interface, complete remaining features, add unit/integration tests, and conduct comprehensive code reviews and documentation.

Road Map/Roles/Timeline:

Ahmed:

Sprint 1: Design wireframes and create UI mockups for the landing page, navbar, login page, and course overview page. Implement the static front-end prototype using HTML, CSS, and a basic JavaScript framework (or React).

Sprint 2: Assist in integrating the front-end with backend APIs. Refine the UI components based on user feedback and initial testing.

Sprint 3: Lead final UI refinements and responsive design adjustments. Connecting frontend to backend for all features. Collaborate in user acceptance testing and address front-end bugs.

Aareb:

Sprint 1: Set up the initial project repository, configure coding standards, and establish the development environment. Create a basic project architecture document.

Sprint 2: Develop core backend REST APIs (e.g., authentication, course management) using the chosen framework (e.g., Django, Spring Boot, or Node.js). Ensure secure API endpoints and implement error handling.

Sprint 3: Optimize and refactor backend code for scalability. Work on performance testing and documentation of API endpoints.

Yusuf:

Sprint 1: Plan out database design and set up the initial database schema (e.g., for courses, users, assignments). Assist with documenting technical architecture and data flow diagrams.

Sprint 2: Implement the database interactions for the backend. Develop unit tests for data model operations and API integration.

Sprint 3: Enhance database performance and create backup/restore procedures. Oversee integration testing between the database, backend, and front-end.

Sameen:

Sprint 1: Write and maintain project documentation (README, work plan, project overview). Coordinate team meetings and manage the repository (ensuring proper branching, pull requests, and code reviews).

Sprint 2: Monitor progress through sprint reviews; perform code reviews to ensure quality and adherence to coding standards. Update documentation with new features and architectural changes.

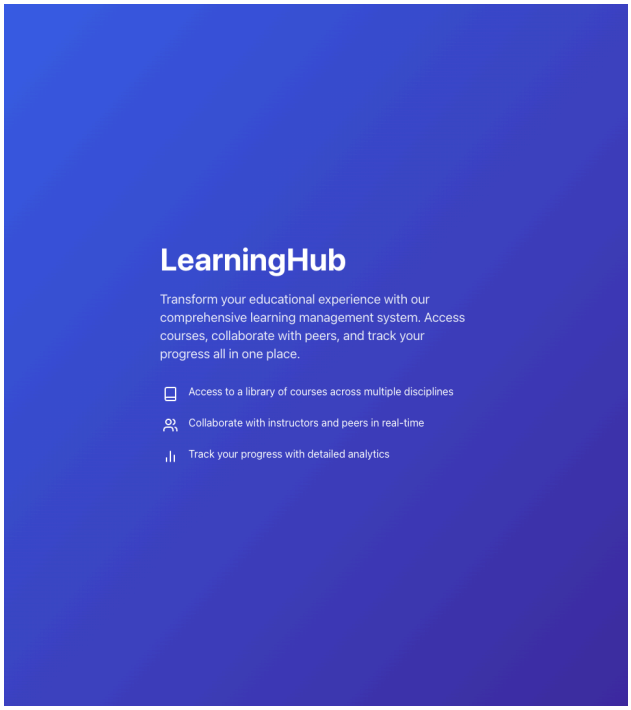
Sprint 3: Finalize comprehensive documentation (including user guides, technical documentation, and testing results). Lead the quality assurance process, coordinating integration and system tests, and preparing the final deliverable package.

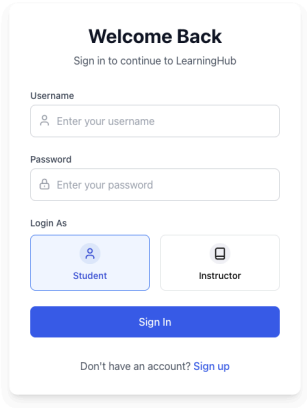
3.2. Project Organization

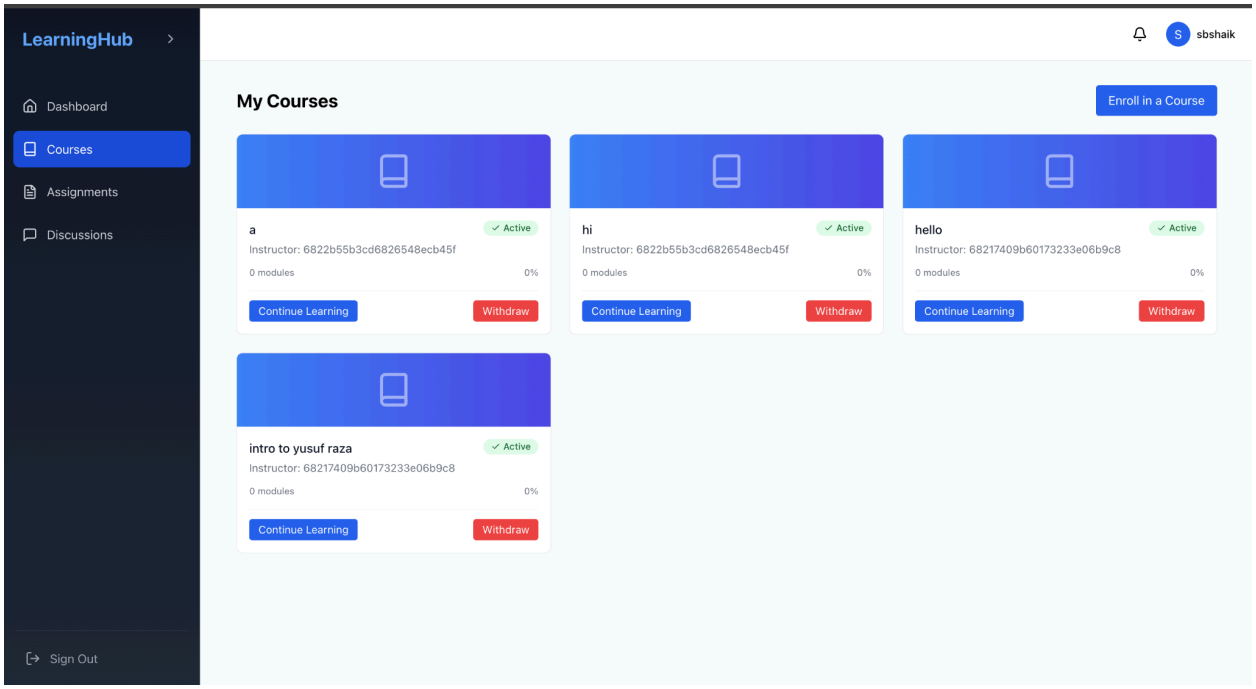
Provide one or more screenshots of your project organization in the git repository.

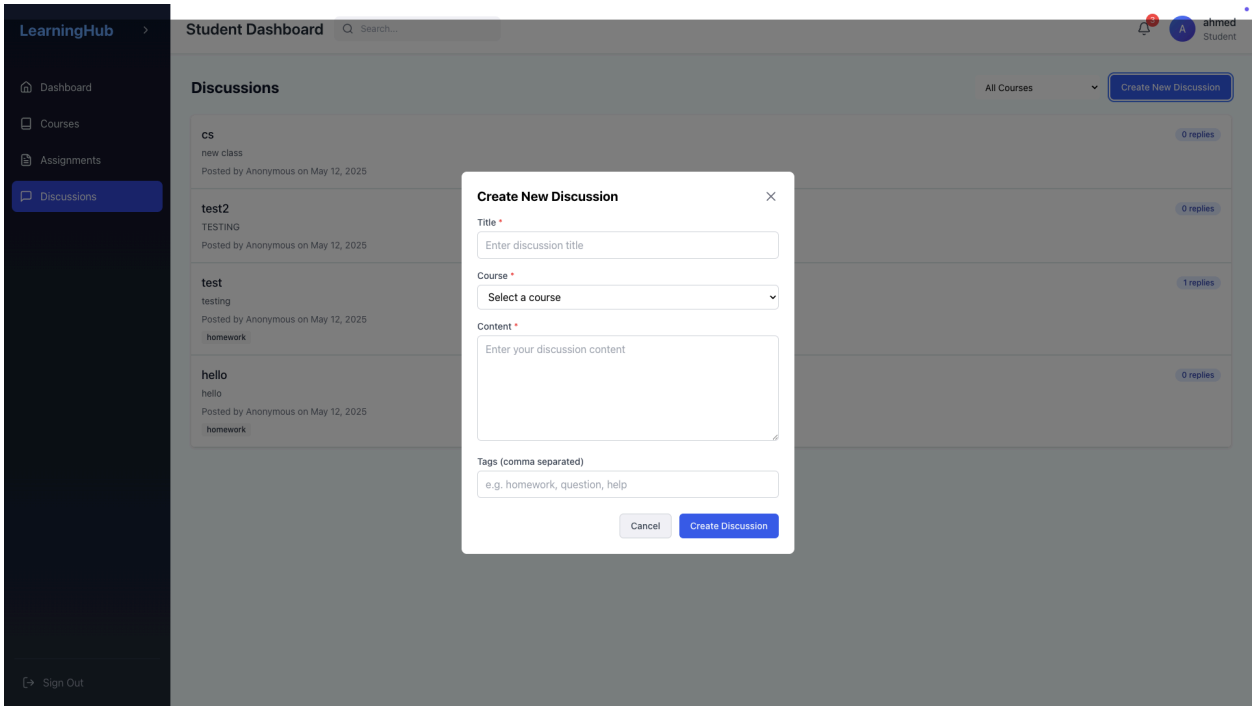
3.3. UI Prototype

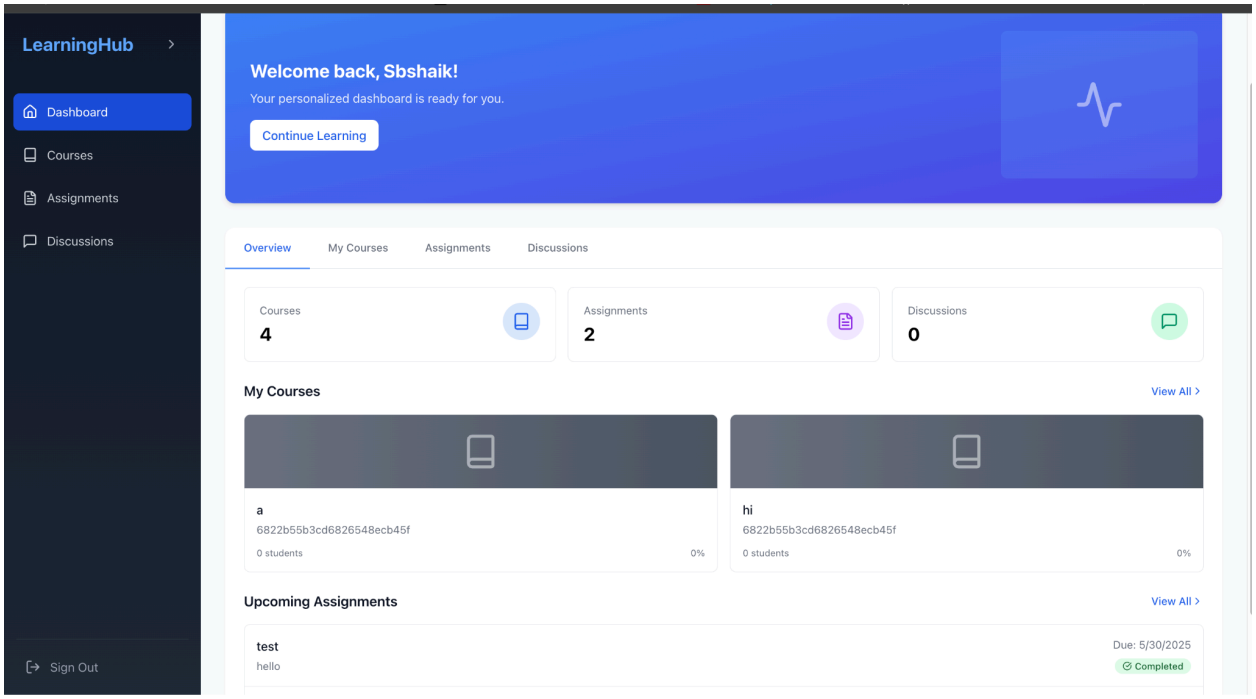
Provide screenshots of the main pages in your UI prototype.

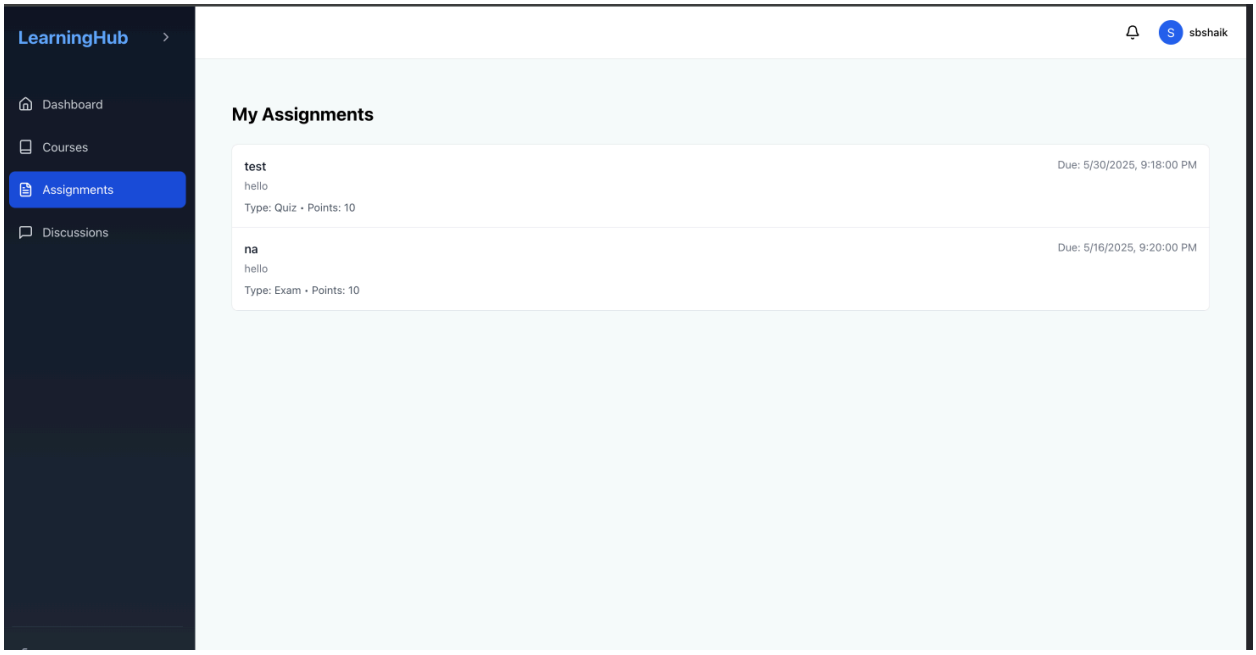

LearningHub
Transform your educational experience with our comprehensive learning management system. Access courses, collaborate with peers, and track your progress all in one place.
Access to a library of courses across multiple disciplines
Collaborate with instructors and peers in real-time
Track your progress with detailed analytics


Welcome Back
Sign in to continue to LearningHub
Username
Enter your username
Password
Enter your password
Login As
Student
Instructor
Sign In
Don't have an account? Sign up


LearningHub
Dashboard
Courses
Assignments
Discussions
Sign Out
sbshaik
My Courses
Enroll in a Course
a
Active
Instructor: 6822b55b3cd6826548ecb45f
0 modules
0%
Continue Learning
Withdraw
hi
Active
Instructor: 6822b55b3cd6826548ecb45f
0 modules
0%
Continue Learning
Withdraw
hello
Active
Instructor: 68217409b60173233e06b9c8
0 modules
0%
Continue Learning
Withdraw
intro to yusuf raza
Active
Instructor: 68217409b60173233e06b9c8
0 modules
0%
Continue Learning
Withdraw


LearningHub
Student Dashboard
Search...
ahmed
Student
Dashboard
Courses
Assignments
Discussions
Sign Out
Discussions
All Courses
Create New Discussion
cs
new class
Posted by Anonymous on May 12, 2025
0 replies
test2
TESTING
Posted by Anonymous on May 12, 2025
0 replies
test
testing
Posted by Anonymous on May 12, 2025
homework
1 replies
hello
hello
Posted by Anonymous on May 12, 2025
homework
0 replies
Create New Discussion
Title *
Enter discussion title
Course *
Select a course
Content *
Enter your discussion content
Tags (comma separated)
e.g. homework, question, help
Cancel
Create Discussion


LearningHub
Dashboard
Courses
Assignments
Discussions
Sign Out
Welcome back, Sbshaik!
Your personalized dashboard is ready for you.
Continue Learning
Overview
My Courses
Assignments
Discussions
Courses
4
Assignments
2
Discussions
0
My Courses
View All
a
6822b55b3cd6826548ecb45f
0 students
0%
hi
6822b55b3cd6826548ecb45f
0 students
0%
Upcoming Assignments
View All
test
hello
Due: 5/30/2025
Completed



3.4. Data Model Prototype

Provide screenshots of the main entities and their relationships in your data model prototype.

3.5. Challenges & Risks

Challenges:

We first faced many challenges when it came to setting up the backend, specifically the MongoDB database. It took us some time to connect our database using the Mongo API key, but once that was connected, we started to adapt and excel in using it. We then faced challenges with connecting our frontend to our backend, but after time it was done successfully.

Risks:

Some risks in our project were implementing the database with MongoDB to ensure that passwords we used were securely saved. We got around this using password hashing before they were saved into the database.

4. Evaluation

4.1 Evaluation of Functional Requirements

Model Testing: Jest unit tests for data models achieved 93.75% statement coverage.

- User.js: 100% coverage
- Course.js: 100% coverage
- Discussion.js: 100% coverage

Model Tests verified the following:

- Model creation and validation
- Required fields functionality
- Data relationships between models
- Password security features
- Course status updates

Manual system and integration testing ensured functionality between the frontend and backend:

- Frontend-backend communication
- User role restrictions
- Course enrollment process
- Discussion forum functionality
- API endpoints for authentication, courses, and discussions
- Database operations with MongoDB
- Full authentication flows (registration, login, token generation)

4.2 Evaluation of Non-functional Requirements

Usability Assessment:

- We conducted usability testing by collecting feedback from our fellow CS520 students during the final survey, and that feedback allowed us to cut down on some of the complex aspects of our features we were implementing, to make sure it was usable and intuitive for our user audience.

Reliability Assessment:

- We tested our system, both front end and back end, throughout the day on every single day. As long as MongoDB was up, our Online Learning Management Service was available reliably at any time of the day or night.

Security Assessment:

- We conducted security testing by ensuring that our database doesn't store exact passwords. If passwords were stored directly, passwords could be revealed to a hacker if the database was ever breached. Thus, we decided to utilize hashing for the passcode, so the database would never store any codes that could be of use to a hacker.

5. Discussion [Lessons Learned, Limitations & Future Plans]

Reflect on the overall performance of your project, including any challenges or limitations encountered, and outline future development plans.

5.1. Lessons Learned

Organization:

- Our team organized ourselves very well using our GitHub repository. We took advantage of using different remote branches to optimize our time so we could work on different features at the same time. Examples include frontend UI, backend models, backend routing, jest testing, and mongo connections.

Communication:

- Our team primarily used iMessage messaging as a communication platform, but we also repeatedly took advantage of Google Meet meetings in order to ensure we were all on the same page with our visions for the Learning Management Service we were building. Google Meet facilitated the process primarily because of the share-screen feature, since we were able to show each other exactly which bugs and features we wanted to work on.

SE Model, tools, and techniques:

- As for SE Model, we utilized the RESTful API to ensure we could satisfy non-functional requirements that the API was good for. Examples are Maintainability, Scalability, Flexibility, and Simplicity.
- For most of us, it was our first time using MongoDB compass to manage our database and control it remotely. This way, we were able to ensure no accidental test data stayed in the database and messed things up
- It was also our first time using Jest, since we're actually used to using Junit instead. Fortunately, Jest wasn't very hard to manage, so we learned quite quickly.

Tech Stack:

- Our Tech Stack includes, but not limited to, Node, React, Express, MongoDB/Mongoose, which classifies this project as a MERN stack project.
- We also used JSX for some components as well as Tailwind CSS, which were new members of the Tech Stack for some of our members. The project really helped in teaching us about these useful new technologies.
- We utilized Jest for testing, so we were able to ensure we had high coverage and correct outputs.

5.2. Limitations (e.g., time constraints, integration, debugging, teamwork)

A limitation we had was time constraints. We are all seniors and have different commitments and schedules which made it difficult to meet in person. Instead we met on zoom meetings up to milestone deadlines. An issue from this was we had difficulty assigning roles and maintaining responsibilities through the project.

Teamwork could use some work since many group members were not prompt in responding to messages, and along with time constraints each individual faced, made it difficult to find appropriate time to work together.

5.3. Future Development Plans

Our application has many opportunities to scale and add on to. For example, adding a calendar feature that includes assignments under the due date to help users keep track of assignment due dates.

Another feature to add is an analytics page for instructors, which could show engagement / completement rates, enrollment amounts, and other important information which an instructor may want to know Furthermore, the dashboard was designed with scalability in mind. More sections can be added, including a little list of assignments due soon, overdue tasks, etc. to make the user experience better. Additionally