

# BrightBoard – A Course and Quiz System

CSCI 441 VA: Software Engineering

Jun 17, 2025

Team Members: Shadow Love-Erckert, Conner Erckert

Project URL:

GitHub URL: [https://github.com/Element713/CSCI441\\_BrightBoard.git](https://github.com/Element713/CSCI441_BrightBoard.git)

## Work Assignment

---

### a. Individual Contributions Breakdown

While both team members actively collaborated on all aspects of the project, specific responsibilities were divided to optimize workflow based on individual strengths. The following table summarizes each member's contributions across major report and development components.

Component / Task	Shadow Love-Erckert	Conner Erckert
<b>1. Requirements Specification</b>		
Functional Requirements (Use Cases)	A/R	A/R
Non-Functional Requirements (FURPS+)	A/R	A/R
Glossary of Terms	A/R	A/R
Requirements Consolidation	A/R	A/R
<b>2. Domain Modeling</b>		
Entity Identification	R	R
Relationship Mapping	C	R
Mongoose Schema Implementation	C	A/R
Data Validation Rules	A/R	A/R
<b>3. Software Design</b>		
Backend Architecture Design (Express.js)	C	A/R
File Structure Planning	C	A/R
Middleware (JWT Auth, Validation)	C	A/R
Routing Logic	A/R	A/R
API Endpoint Specification	A/R	A/R
<b>4. Report Preparation</b>		

Cover Page, Purpose, Work Plan, TOC	A/R	C
Requirements Section	A/R	A/R
Functional & UI Specification	A/R	C
Formatting and Final Edits	A/R	A/R
Diagrams and Illustrations	A/R	A/R
<b>5. Other Contributions</b>		
UI Design & Styling	R	C
Frontend Development (React, CSS)	R	A
Backend Integration (MongoDB, Express)	C	R/A
Quiz System Development	A/R	A/R
Testing & Debugging	A/R	A/R
Deployment (Local & Final)	A/R	A/R
Integration Management	A/R	C
Team Communication & Weekly Check-ins	A	C

## Table of Contents

---

<b>Work Assignment.....</b>	<b>2</b>
a. Individual Contributions Breakdown.....	2
<b>Table of Contents.....</b>	<b>4</b>
1. Customer Problem Statement.....	5
a. Problem Statement.....	5
b. Decomposition into Sub-Problems.....	6
2. Glossary of Terms.....	8
<b>3. System Requirements.....</b>	<b>9</b>
a. Business Goals.....	9
b. Functional Requirements.....	10
c. Non-Functional Requirements.....	11
d. User Interface Requirements.....	11
<b>4. Use Cases.....</b>	<b>16</b>
a. Stakeholders.....	16
b. Actors and Goals.....	16
c. Use Cases.....	17
i. Casual Description.....	17
ii. Use Case Diagram.....	18
iii. Traceability Matrix.....	19
iv. Fully-Dressed Description.....	21
<b>d. System Sequence Diagrams.....</b>	<b>23</b>
<b>5. User Interface Specification.....</b>	<b>25</b>
a. Preliminary Design.....	25
b. User Effort Estimation.....	26
<b>6. System Architecture and System Design.....</b>	<b>31</b>
a. Identifying Subsystems.....	31
b. Architectural Styles.....	32
c. Mapping Subsystems to Hardware.....	33
d. Connectors and Network Protocols.....	33
e. Global Control Flow.....	34
f. Hardware Requirements.....	35
<b>7. Project Management.....</b>	<b>37</b>
a. Plan of Work.....	37
<b>8. References.....</b>	<b>39</b>

# 1. Customer Problem Statement

---

## a. Problem Statement

The online learning landscape has experienced explosive growth in recent years, fueled by remote education, self-paced learning trends, and the rise of independent educators offering specialized content. However, the digital tools designed to support this new wave of learning have not kept pace with the needs of all instructors. Most existing Learning Management Systems (LMSs) were originally developed for large academic institutions or enterprises. As a result, they are often packed with advanced features that require technical knowledge, training, or IT support to operate effectively.

For independent tutors, freelance educators, corporate trainers, and hobby instructors, these systems can be more of a barrier than a benefit. The learning curve is steep, the interfaces are often unintuitive, and the setup process is overly complex for those who simply want to deliver straightforward, engaging content. These educators aren't looking for bloated platforms designed to manage thousands of users across departments; they need lightweight, practical tools that allow them to focus on teaching, not system configuration.

To cope with this gap, many small-scale instructors have resorted to using a mix of unrelated tools. A typical course might involve; Lesson content shared via email attachments or cloud drives (e.g., PDFs, Google Docs), quizzes built with free tools like Google Forms or Typeform, and progress and performance tracked manually in Excel or Google Sheets.

This approach creates a disconnected and inconsistent learning experience. There's no centralized hub for accessing course materials, submitting assignments, or checking quiz results. Feedback is often delayed or missing entirely. Instructors, meanwhile, have no clear visibility into how students are progressing or where they may be struggling.

The consequences of this fragmented approach are significant. Decreased student engagement, due to confusion and lack of structure, increased instructor workload, caused by repetitive tasks like grading and progress tracking limited feedback loops, preventing real-time support or course adjustments, lower course completion rates, as learners lose momentum or motivation

What's clearly missing from the market is a simple, unified, and purpose-built LMS one that strips away the unnecessary complexity and focuses solely on what matters most: delivering content, assessing understanding, and supporting learner progress.

This modern LMS should support microlearning formats, ideal for short courses, private sessions, or skill-based workshops, and allow instructors to upload content easily in either PDF

or text form, enable the creation of quizzes with automatic scoring and instant feedback, provide a clean, centralized interface for students to view lessons, take assessments, and monitor their own progress, and offer intuitive analytics for instructors, helping them identify trends, assess outcomes, and support students proactively

In short, the solution must be streamlined, scalable, and accessible removing the friction that currently exists between small scale educators and the tools they use. By reducing technical overhead and centralizing essential functions, such a platform could dramatically improve the experience for both instructors and learners, fostering higher engagement, better outcomes, and broader access to high-quality educational content.

## **Independent Tutor**

I've been teaching web design in short, one-month sprints to small groups of students. The hardest part isn't the content, it's delivering it in a way that's cohesive and engaging. Using a mix of Google Drive, emails, and Forms frustrates me and my students. I have no way to know who's keeping up, and grading quizzes manually is time-consuming. I've tried using some LMS platforms, but they're built for big institutions and are overkill for what I need.

What I really need is a lightweight, easy-to-use platform that allows me to organize my lessons, quizzes, and student progress in one place. It should give instant scoring, help me know how students are doing, and be fast to set up. Ideally, I should be able to upload a PDF or type a quick lesson, build a quiz, and be done in minutes. I don't have time to configure complex modules or install anything. If such a tool existed, it would save me hours each week and give my students a more engaging and clear learning experience.

## **Student**

I'm taking short courses on topics like photography and personal finance from different instructors. Every time, it's a different process — sometimes I get an email, other times I'm sent to Google Classroom or a random website with links. It's confusing and annoying. I often forget where things are or miss a quiz just because I didn't check my email in time.

What I want is one platform where everything lives: the lesson, the quiz, and my scores all in one place, accessible on my phone or laptop. I'd like to see what I've completed and what I have left to do. I don't want to juggle five tabs just to learn something simple. It would be great if quizzes gave me my score right away, so I know if I understood the lesson or not. A system like that would make short courses much more enjoyable and help me stay committed to finishing them.

## **b. Decomposition into Sub-Problems**

We identified the following sub-problems from the customer's narrative:

1. **Authentication & User Roles**

- Users must register and log in as students or instructors.

2. **Course Management (Instructor Side)**

- Instructors must be able to create courses, upload lessons (PDF or text), and add quizzes.

3. **Enrollment & Learning Flow (Student Side)**

- Students need to browse courses, enroll, read lessons, take quizzes, and track progress.

4. **Quiz System**

- Support multiple-choice quizzes with automated scoring and feedback.

5. **Progress Tracking**

- Record student activity and display progress to both instructors and learners.

6. **Analytics Dashboard**

- Instructors need a way to see which students are completing content and how well they are performing.

## 2. Glossary of Terms

---

Term	Definition
BrightBoard	The name of the proposed lightweight Learning Management System (LMS).
Microlearning	Short, focused learning activities designed to achieve specific learning outcomes.
Course	A collection of lessons and quizzes grouped under a single subject or topic.
Lesson	A single unit of instructional content, presented as text or in PDF format.
Quiz	A set of multiple-choice questions associated with a specific lesson or course.
Instructor	A user who creates, manages, and publishes courses and their associated content.
Student	A user who enrolls in courses, studies lessons, and attempts quizzes.
Progress Tracker	A feature that displays a student's completion status for lessons and quizzes.
Educational Institutions	Small schools or informal learning organizations using BrightBoard as a supplementary platform.
Businesses	Organizations using BrightBoard for internal microlearning and training sessions.
System Developers	Individuals responsible for implementing the backend, frontend, and integration logic.
UI Designers	Team members who craft the user interface for usability and accessibility.
Web Application System	The underlying software system that delivers content and handles all user interactions.
Device	The physical hardware (desktop, tablet, or mobile) used to access BrightBoard.
Register/Login	The process of creating an account or signing in securely to access personalized dashboards.

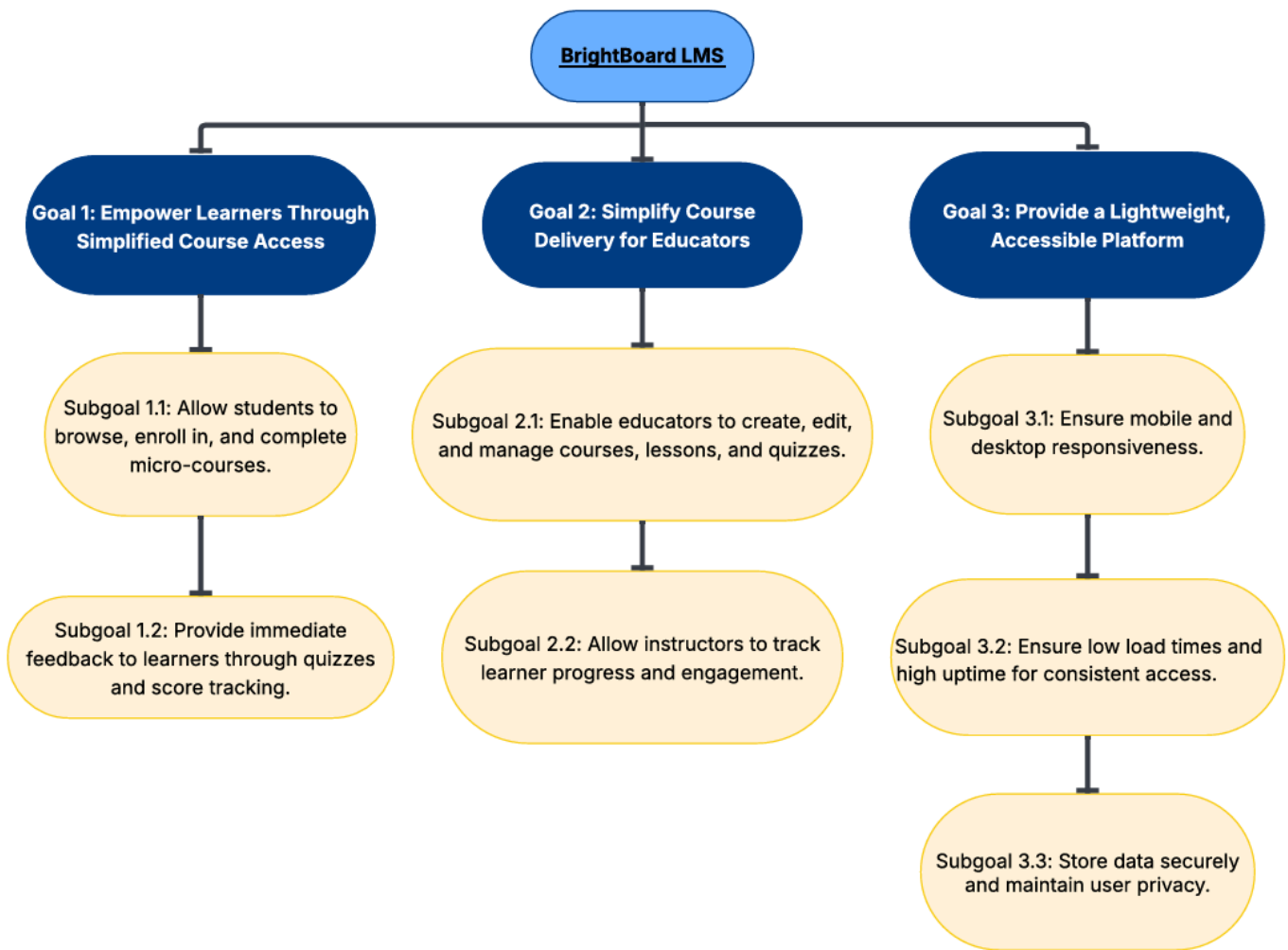


Course Management	Instructor functionality for creating, editing, and deleting courses.
Lesson Uploading	Instructor capability to upload PDF/text content to courses.
Quiz Creation	Instructor capability to design quizzes for assessments.
Enroll in Course	Student action of joining an available course.
View Lessons	Student functionality to read and interact with uploaded lessons.
Take Quiz	Student interaction with course quizzes, including auto-scoring.
JWT Authentication	JSON Web Token–based system for securely verifying user identity.
UI (User Interface)	The visual and interactive part of BrightBoard that users engage with.
Dashboard	A role-specific home interface showing available actions and user-specific data.

### 3. System Requirements

---

#### a. Business Goals



## b. Functional Requirements

Req ID	Priority	Description
REQ-1	5	The system shall allow users to register and log in securely.

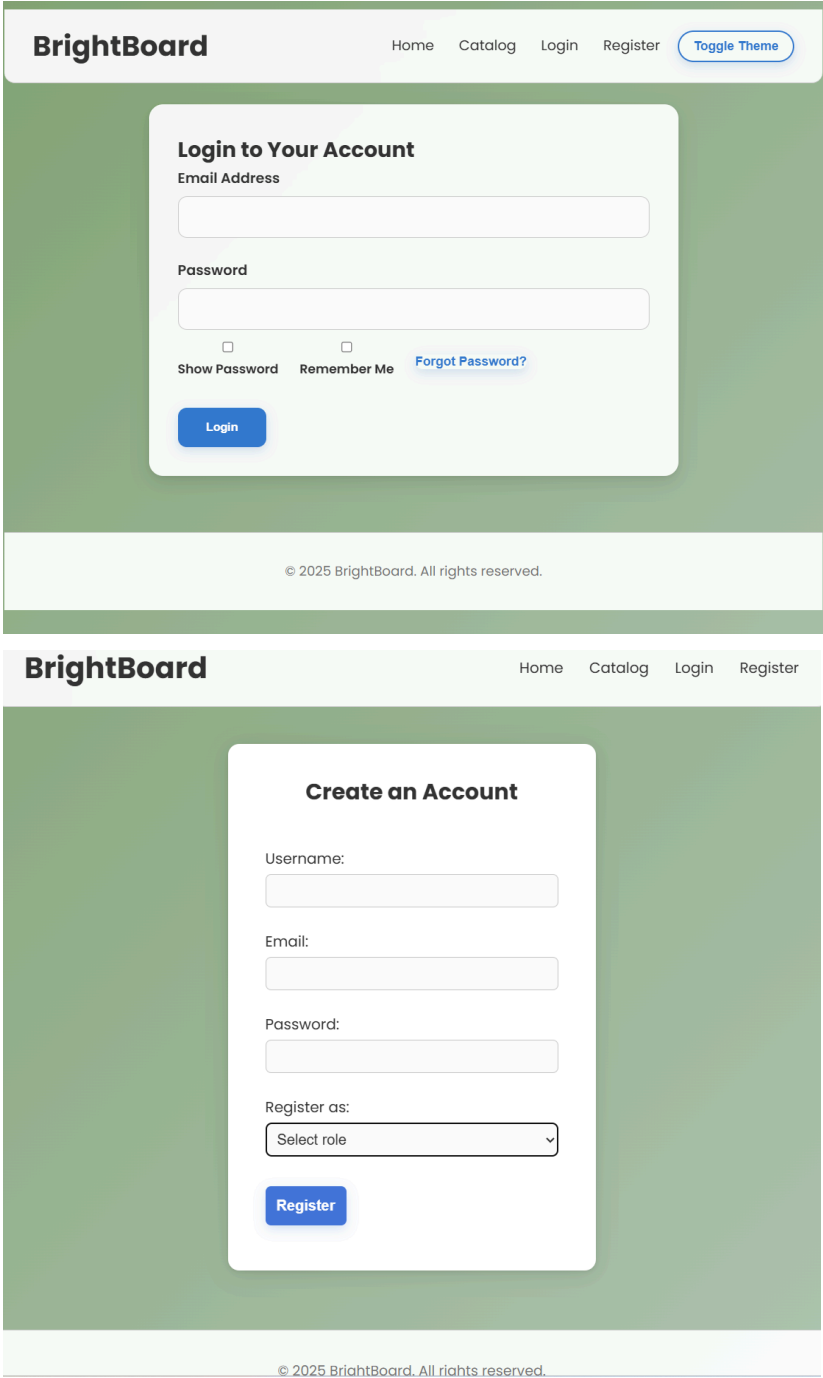
REQ-2	5	The system shall support two user roles: Instructor and Student.
REQ-3	5	Instructors shall be able to create, edit, and delete courses.
REQ-4	5	Instructors shall upload lessons in PDF or text format.
REQ-5	5	Instructors shall create multiple-choice quizzes.
REQ-6	5	Students shall browse and enroll in courses.
REQ-7	5	Students shall view lesson content.
REQ-8	5	Students shall take quizzes and receive automatic scoring.
REQ-9	3	The system shall track quiz scores and course progress.
REQ-10	3	Instructors shall view performance analytics.

#### **c. Non-Functional Requirements**

<b>Req ID</b>	<b>Priority</b>	<b>Description</b>
NFREQ-1	5	The system should have a responsive user interface (desktop & mobile).
NFREQ-2	4	The system should load content in under 2 seconds.
NFREQ-3	3	The system should support up to 10,000 concurrent users.
NFREQ-4	5	The system should use secure password encryption and comply with privacy laws (e.g., GDPR).
NFREQ-5	4	The system should maintain 99.9% uptime.

#### **d. User Interface Requirements**

<b>ID</b>	<b>Priority</b>	<b>Requirement Description</b>
-----------	-----------------	--------------------------------

UI-1	5	<p>The interface shall include a registration/login screen.</p> <div><p>The image displays two screenshots of the BrightBoard web application interface. The top screenshot shows the 'Login to Your Account' screen. It features a header with the 'BrightBoard' logo and navigation links for 'Home', 'Catalog', 'Login', and 'Register'. A 'Toggle Theme' button is located in the top right corner. The main content area contains a login form with fields for 'Email Address' and 'Password'. Below the password field are checkboxes for 'Show Password' and 'Remember Me', along with a 'Forgot Password?' link. A blue 'Login' button is at the bottom of the form. The bottom screenshot shows the 'Create an Account' screen. It has the same header and navigation links. The main content area contains a registration form with fields for 'Username', 'Email', and 'Password'. Below these is a 'Register as:' section with a dropdown menu labeled 'Select role'. A blue 'Register' button is at the bottom of the form. Both screens have a footer with the copyright notice '© 2025 BrightBoard. All rights reserved.'</p></div>
UI-2	5	<p>The student dashboard shall show enrolled courses and progress.</p>



		<div><div><h2>Welcome, Professor!</h2><div><a href="#">Create a New Course</a></div><div><div>Course Title:</div><div></div></div><div><div>Description:</div><div></div></div><div><div>Add Course</div></div><div><a href="#">Active Courses</a></div><div><div><div><b>Intro to HTML &amp; CSS</b></div><div>Learn the basics of web development.</div></div><div><div><b>Data Science 101</b></div><div>Introduction to data analysis and visualization.</div></div></div></div></div>
UI-4	4	<div><div><h3>Lesson view should support embedded PDFs and formatted text.</h3><div><div><div><div><div>Example Conversation</div><div>Maria: Hola, ¿cómo estás?</div><div>Juan: ¡Hola Maria! Muy bien, gracias. ¿Y tú?</div></div></div><div><div>Lesson PDF</div><div><div></div></div><div><div>If you can't view the PDF, <a href="#">click here to download it.</a></div><div><div>Take Quiz</div></div></div></div></div></div></div></div>

UI-5	3	<p>Quizzes should present questions one at a time with immediate feedback.</p>
UI-6	3	<p>Navigation should be consistent and minimal across all views.</p>
UI-7	1	<p>Course list should support search and filter features.</p>

## 4. Use Cases

---

### a. Stakeholders

BrightBoard will serve a diverse group of stakeholders involved in microlearning and lightweight educational platforms. Below is a categorized list of stakeholders for BrightBoard:

- **Educators:** These are individuals teaching private or small group sessions who require simple tools for content delivery and assessment. They are primary users who directly benefit from streamlined course creation and progress tracking.
- **Students:** Learners who enroll in short courses or workshops and rely on BrightBoard for lesson access, assessments, and progress tracking.
- **Businesses:** Use BrightBoard for quick, targeted training sessions. Stakeholders here include HR departments or team leads looking for efficient internal learning tools.
- **Educational Institutions (Small Schools or Informal Learning Organizations):** May adopt BrightBoard for auxiliary teaching or as a resource for workshops and tutoring.
- **System Developers and UI Designers:** Responsible for building and maintaining the system. Their stake lies in meeting project deadlines and ensuring system quality.

### b. Actors and Goals

#### Primary Initiating Actors:

Actor	Role	Goal
Instructor	Manages courses, uploads content, and creates quizzes. Uses the system to monitor student performance.	The goal of the instructor is to: Login and register account, Create, access and interact with lessons and quizzes, Monitor class progress, Navigate cleanly through a minimal UI.
Student	Enrolls in courses, accesses lessons, takes quizzes, and reviews progress.	The goal of the student is to login and register an account, Access and interact with lessons and quizzes, Monitor personal or class progress, Navigate cleanly through a minimal UI.

#### Secondary Participating Actors:

Actor	Role
-------	------



Web Application System	The role of the Web Application System is to deliver content and facilitate all interactions between users and data.
Devices (Desktop, Mobile, Tablet)	The role of the device is to become a medium through which actors access BrightBoard.

### **c. Use Cases**

#### **i. Casual Description**

##### UC1: Register/Login

- Description: The system enables users to register and authenticate securely.
- Responds to Requirements: REQ-1, REQ-2, NFREQ-1, NFREQ-4, UI-1

##### UC2: Course Management

- Description: Instructors can create, edit, and delete courses.
- Responds to Requirements: REQ-2, REQ-3, UI-3

##### UC3: Lesson Uploading

- Description: Instructors upload lesson content as PDF or text.
- Responds to Requirements: REQ-4, UI-4

##### UC4: Quiz Creation

- Description: Instructors design multiple-choice quizzes.
- Responds to Requirements: REQ-5, UI-3, UI-5

##### UC5: Enroll in Course

- Description: Students enroll in available courses.
- Responds to Requirements: REQ-6, UI-2, UI-7

##### UC6: View Lessons

- Description: Students access course content.
- Responds to Requirements: REQ-7, UI-2, UI-4

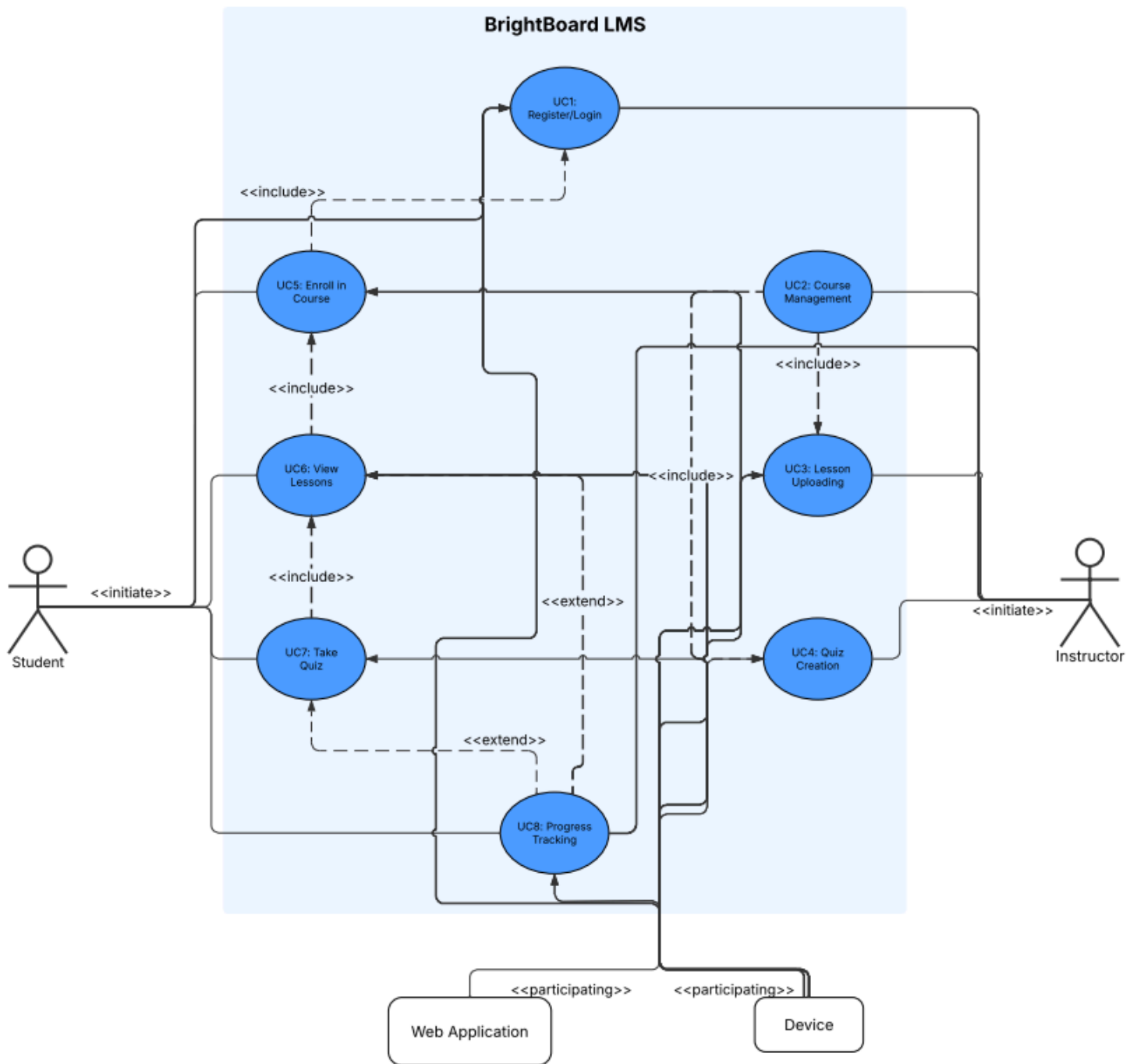
##### UC7: Take Quiz and Score Automatically

- Description: Students take quizzes and receive immediate feedback.
- Responds to Requirements: REQ-8, UI-5

##### UC8: Progress Tracking

- Description: Tracks quiz completion and learning progress.
- Responds to Requirements: REQ-9, REQ-10, UI-2, UI-3

## ii. Use Case Diagram



### iii. Traceability Matrix

Req't	PW	U C1	U C2	U C3	U C4	U C5	U C6	U C7	U C8	U C9	U C1 0	U C1 1	U C1 2	U C1 3	U C1 4	UC 15
REQ-1	5	X														
REQ-2	5	X	X													
REQ-3	5		X													
REQ-4	5			X												
REQ-5	5				X											
REQ-6	5					X										
REQ-7	5						X									
REQ-8	5							X								
REQ-9	3								X							
REQ-10	3								X							
NFREQ -1	5	X														
NFREQ -2	4															
NFREQ -3	3															
NFREQ -4	5	X														
NFREQ -5	4															
UI-1	5	X														
UI-2	5					X	X		X							
UI-3	5				X				X							

UI-4	4			X			X									
UI-5	3		X		X			X								
UI-6	3															
UI-7	1					X										

Use Case	Related Requirements	Priority Weight
Register/Login	REQ-1, REQ-2, NFREQ-1, NFREQ-4, UI-1	$5 + 5 + 5 + 5 + 5 = 25$
Course Management	REQ-2, REQ-3, UI-3	$5 + 5 + 5 = 15$
Lesson Uploading	REQ-4, UI-4	$5 + 4 = 9$
Quiz Creation	REQ-5, UI-3, UI-5	$5 + 5 + 3 = 13$
Enroll in Course	REQ-6, UI-2, UI-7	$5 + 5 + 1 = 11$
View Lessons	REQ-7, UI-2, UI-4	$5 + 5 + 4 = 14$
Take Quiz	REQ-8, UI-5	$5 + 3 = 8$
Progress Tracking	REQ-9, REQ-10, UI-2, UI-3	$3 + 3 + 5 + 5$

The weight calculated in the above table includes the Nonfunctional and User Interface Requirements. If we were to exclude these then the use case “Register/Login”, and “Course Management” have the highest priority weight.

#### **Elaboration:**

##### **1. Register/Login:**

The Register/Login functionality is the most essential entry point into the BrightBoard platform and is therefore a top priority for the first demo. It establishes the foundational user authentication and access control system upon which all other features depend. Without it, no user whether student or instructor can securely access their personalized dashboard or interact with otherwise protected content. This component not only demonstrates secure user authentication using industry standards like JWT but also introduces role-based navigation,

immediately showing how the platform tailors itself to different user types. By including Register/Login in the first demo, the system will be displayed with basic integrity and usability while laying the groundwork for deeper functionality in subsequent stages.

## 2. Course Management:

Course Management is the core of the instructor experience and highlights BrightBoard's value as a lightweight, focused learning platform. It allows instructors to create courses, upload lessons, and build quizzes all critical to delivering educational content. Including this feature in the Demonstration of course management early also gives a clear sense of how instructor role users will interact with the system on a daily basis. It reflects the workflow of an educator and reinforces BrightBoard's mission to simplify teaching and enhance learning.

### iv. Fully-Dressed Description

UC-1: Register/Login
<p><b>Related Requirements:</b> REQ-1, REQ-2, NFREQ-1, NFREQ-4, UI-1</p> <p><b>Initiating Actor:</b> User</p> <p><b>Participating Actors:</b> None</p> <p><b>Preconditions:</b> The user is not logged into the system.</p> <p><b>Postconditions:</b> The user is authenticated and gains access to their dashboard.</p> <p><b>Flow of Events for Main Success Scenario:</b></p> <ol style="list-style-type: none"><li>1. User enters registration/login credentials.</li><li>2. System validates user credentials.</li><li>3. Upon success, the user is logged in and redirected to their dashboard.</li></ol> <p><b>Flow of Events for Extensions (Alternate Scenarios):</b></p> <p>2a. Credentials are invalid</p> <ul style="list-style-type: none"><li>• System displays an error message.</li><li>• User re-enters valid credentials.</li></ul> <p>2b. New user registration</p>

- User enters an email and password.
- System registers new user and sends confirmation.

## UC-2: Course Management

**Related Requirements:** REQ-2, REQ-3, UI-3

**Initiating Actor:** Instructor

**Participating Actors:** None

**Preconditions:** Instructor is logged in.

**Postconditions:** Course is created, updated, or deleted in the system.

### Flow of Events for Main Success Scenario:

1. Instructor creates a new course by entering required details.
2. System saves and displays the new course.

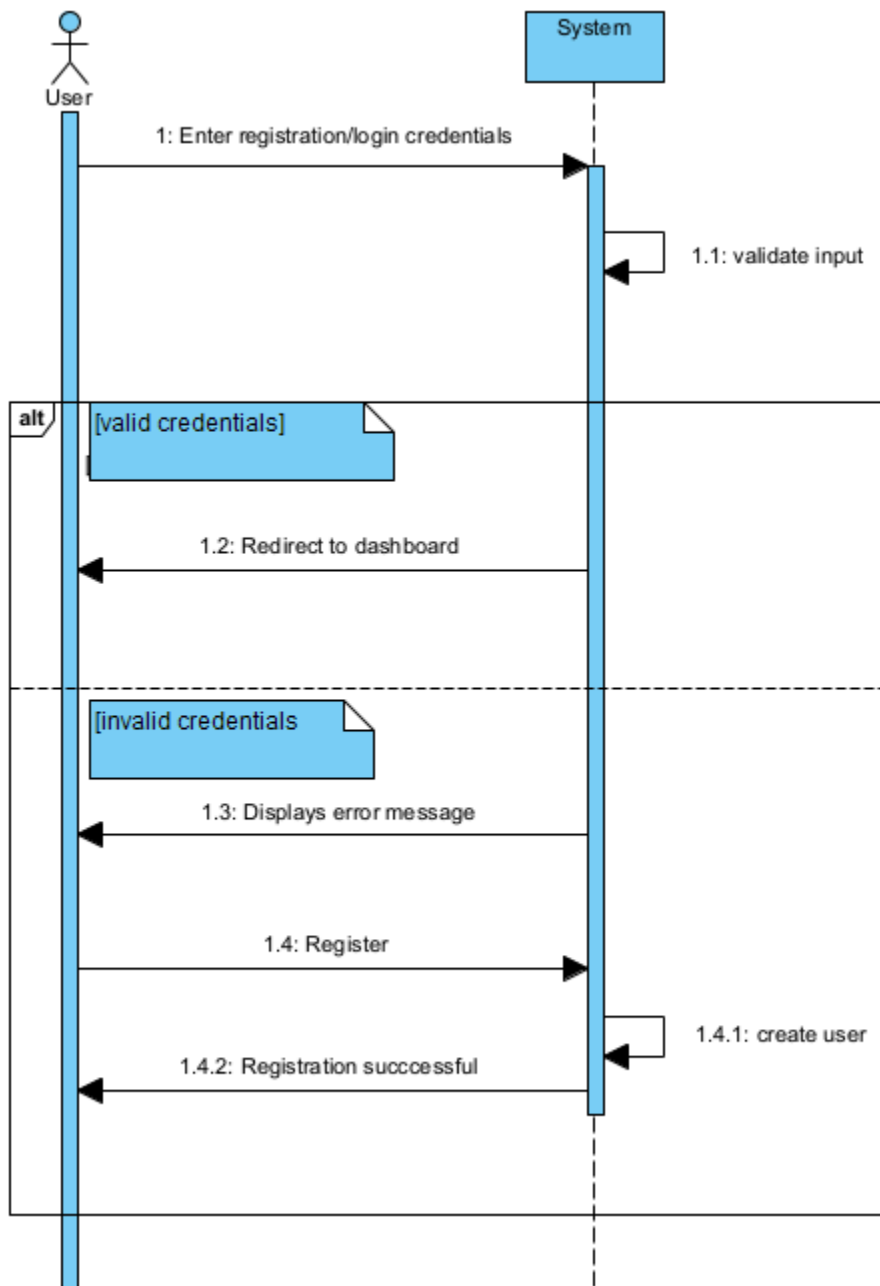
### Flow of Events for Extensions:

#### 1a. Instructor edits existing course

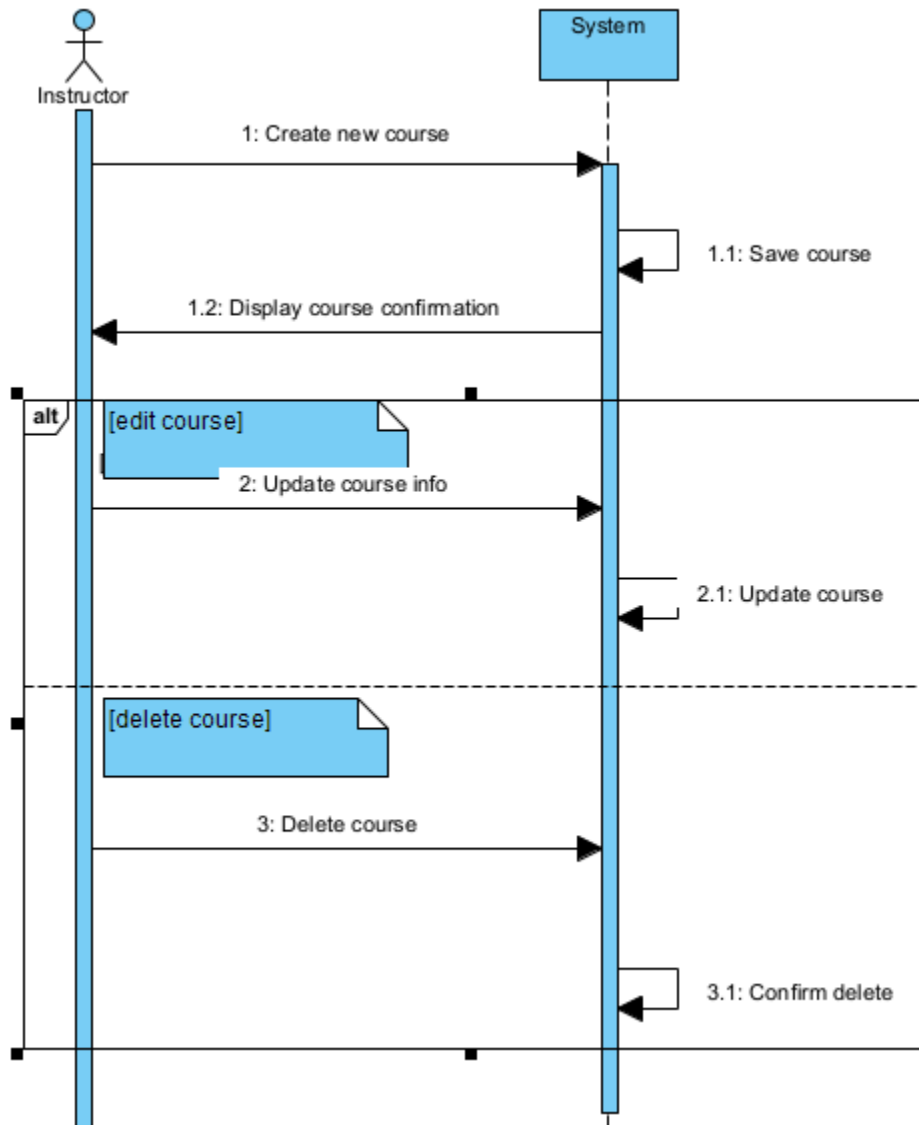
- System loads existing course for update.
- Changes are saved and reflected in the course list.
  - 1b. Instructor deletes a course
- System confirms deletion and removes course from the list.

## d. System Sequence Diagrams

### 1. System Sequence Diagram– UC-1: Register/Login



## 2. System Sequence Diagram – UC-2: Course Management





## 5. User Interface Specification

---

### a. Preliminary Design

#### 1. Course Management – User Interface Specification

**a.** After successfully signing in to their role-specific interface, Instructors will be able to navigate to the “Manage Courses” section, where they can create, view, edit, and delete courses. This section is accessible via a clearly labeled button on their dashboard.

**b.** Instructors can interact with the course list using intuitive action buttons like “Edit”, “Delete”, or “Create New Course”, streamlining course organization and updates within the BrightBoard platform.

#### 2. Register/Login- User Interface Specification

**a.** On accessing the BrightBoard platform, users are greeted with a Login/Register screen. This interface allows users to either sign in using existing credentials or register for a new account.

New users click the “Register” button, which opens a form where they enter: full name, email address, password, select role (Instructor or Student), confirm password. Returning users use the “Login” form by providing email address and password. After successful login, users are directed to their role-specific dashboard: Instructors are taken to the “Instructor dashboard” interface. Students are taken to the “Student dashboard” view.

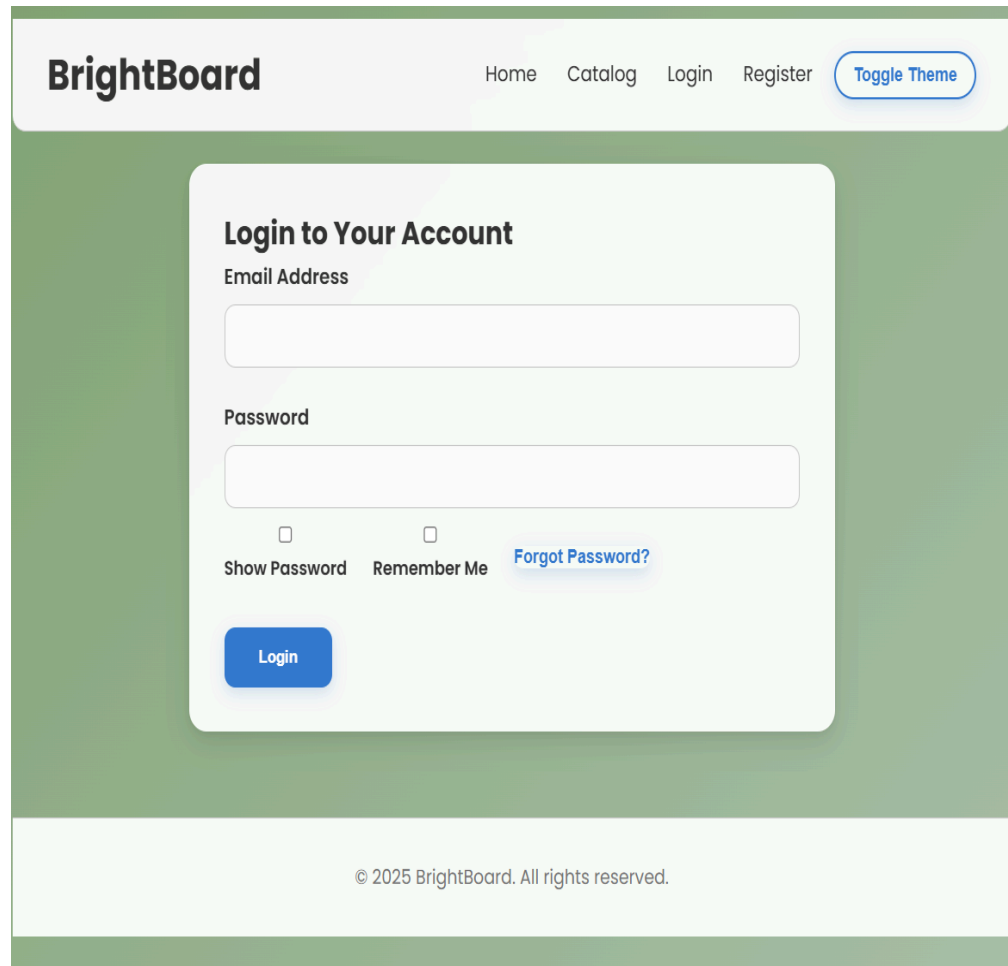
**b.** If login credentials are incorrect or required fields are missing during registration, error messages are displayed clearly under each relevant field (e.g., "Invalid email or password").

The navigation is optimized for ease of use, allowing users to quickly transition from registration or login to their core platform tasks.

## b. User Effort Estimation

### Scenario: Completing a 3-question quiz

- *Figure 1: BrightBoard System Login page*



The image shows the BrightBoard System Login page. At the top, there is a navigation bar with the BrightBoard logo on the left and links for Home, Catalog, Login, Register, and a Toggle Theme button on the right. The main content area features a login form titled "Login to Your Account". The form includes an "Email Address" input field, a "Password" input field, and two checkboxes: "Show Password" and "Remember Me". A "Forgot Password?" link is also present. A blue "Login" button is at the bottom of the form. The footer contains the copyright notice: "© 2025 BrightBoard. All rights reserved."

**BrightBoard** Home Catalog Login Register [Toggle Theme](#)

### Login to Your Account

Email Address

Password

☐ Show Password ☐ Remember Me [Forgot Password?](#)

Login

© 2025 BrightBoard. All rights reserved.

Figure 2: Student Dashboard

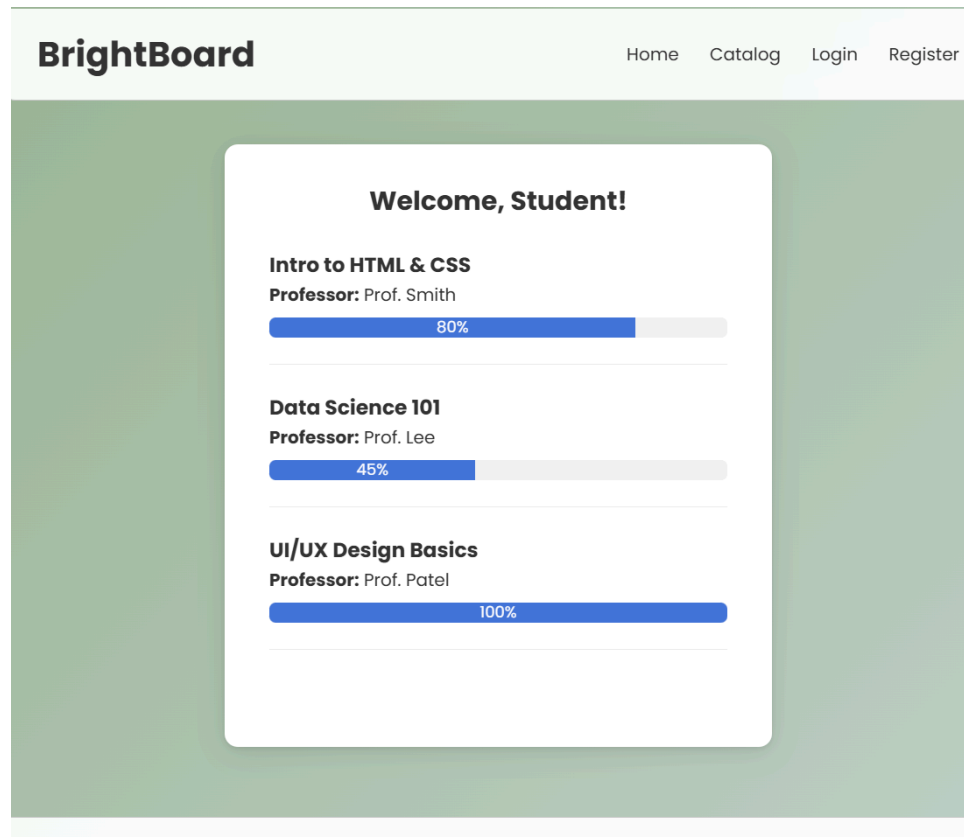


Figure 3: Course/lesson selection

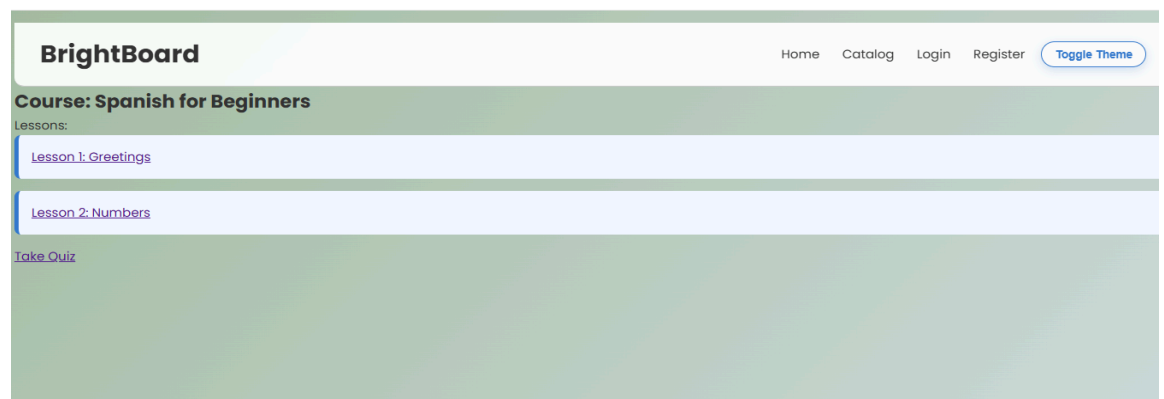


Figure 4: Sample Quiz

**Quiz: Spanish Greetings**

**1. How do you say "Good morning" in Spanish?**

Hola ☐

Buenos días ☐

Buenas noches ☐

**2. What is the Spanish word for "Goodbye"?**

Adiós ☐

Gracias ☐

Por favor ☐

**3. "Buenas tardes" means:**

Good afternoon ☐

Good night ☐

Good morning ☐

**Submit Quiz**

## 1. Quiz:

The following sequence of actions would allow a student enrolled in a course to complete a 3 question quiz

**NAVIGATION:** Total 9 mouse clicks, as follows:

- a. From the login screen, click on the username and password fields.
- b. Click on the "Login" button.
- c. From the student dashboard click on the desired course.
- d. Click the desired lesson within the course.
- e. Click the begin quiz button.
- f. Click on answers for quiz questions (in this example, 3).
- g. Click the submit quiz button.

**DATA ENTRY:** Total of 2 sets of keystrokes, as follows. **a.** Type Username **b.** Type password

**Estimated Time:** 1–2 minutes per quiz.

## **2. Enroll in Course**

The following sequence of actions would allow a student to enroll in an available course using 7 mouse clicks and 2 sets of keystrokes.

### **NAVIGATION**

**Total:** 7 mouse clicks, as follows:

- a.** From the login screen, click on the username and password fields.
- b.** Click on the “Login” button.
- c.** From the student dashboard, click on the “Course Catalog” tab.
- d.** Scroll or browse the list and click on the desired course.
- e.** Click on the “Enroll” button to open the confirmation dialog.
- f.** Click on the “Confirm” button to finalize enrollment.
- g.** Click on “My Courses” to view the newly enrolled course.

### **DATA ENTRY**

**Total:** 2 sets of keystrokes, as follows:

- a.** Type username
- b.** Type password

**Estimated Time:** 30 seconds to 1 minute per course enrollment.

Here’s the detailed effort estimation using the Use Case Points (UCP) method:

### **Project Duration Effort Estimation Using Use Case Points**

#### **Total Use Case Points (UCP)**

Based on the traceability matrix and priorities:

Use Case	Priority Weight
Register/Login	5
Course Management	5
Lesson Uploading	5
Quiz Creation	5
Enroll in Course	5
View Lessons	5
Take Quiz	5
Progress Tracking	3
Export Progress Reports	2
Send Notifications	2

Total UCP =  $5 + 5 + 5 + 5 + 5 + 5 + 5 + 5 + 3 + 2 + 2 = 42$  UCP

### Productivity Factor (PF)

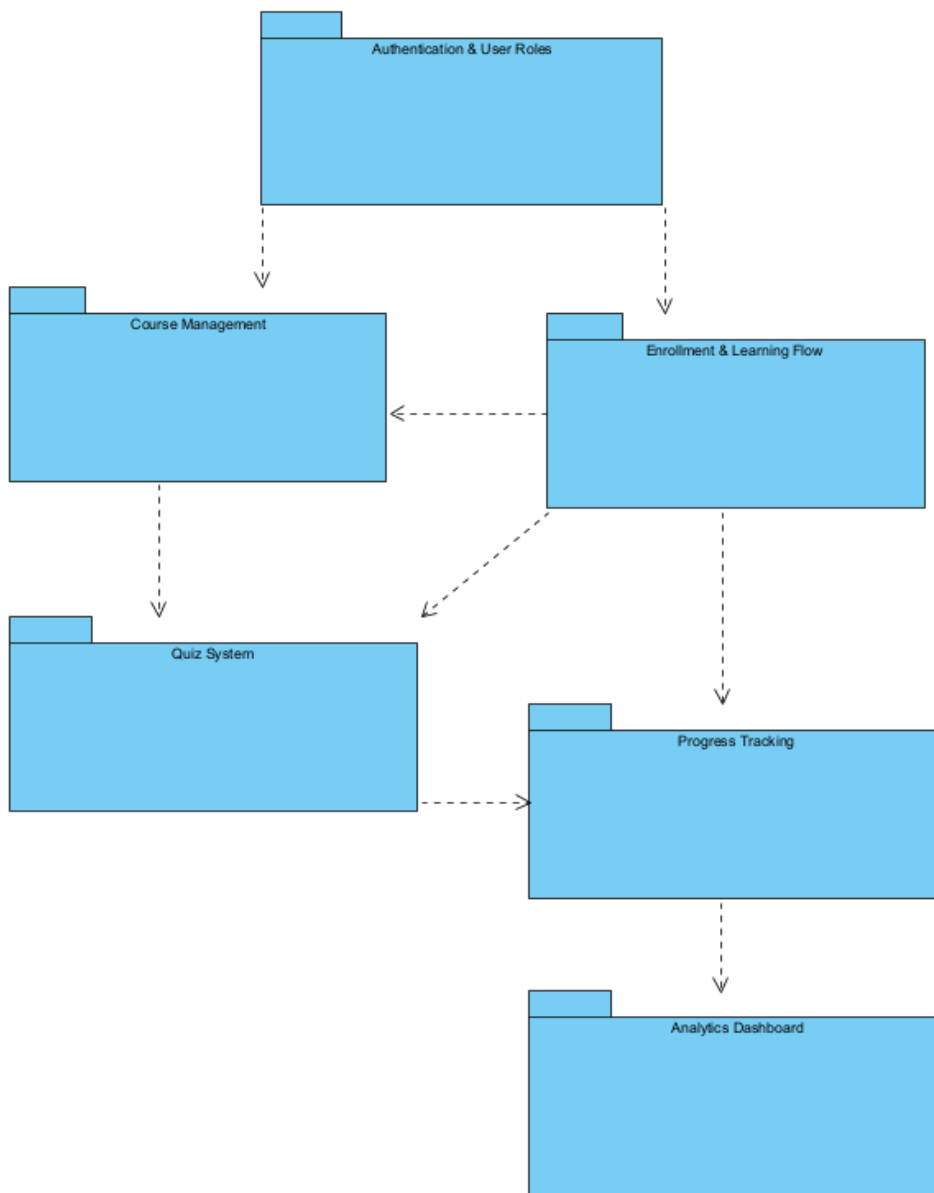
Using a standard industry average of PF = 20–28 hours/UCP: PF = 25 hours for balanced complexity.

**Effort = UCP × PF =  $42 \times 25 = 1,050$  hours**

## 6. System Architecture and System Design

### a. Identifying Subsystems

The architecture of BrightBoard was designed with simplicity, modularity, and scalability in mind, catering specifically to independent educators and small learning cohorts. The system provides a centralized platform for course delivery, quiz administration, and student progress tracking, removing the need for multiple disjointed tools.



## UML Package Diagram Description:

While not presented as a formal diagram in the original report, the subsystems of BrightBoard can be visualized in a package-oriented UML structure. The **Authentication** module connects with both **Course Management** and **Enrollment**, setting the foundation for user access and interaction. **Course Management** interfaces with the **Quiz System**, while **Progress Tracking** and **Analytics** tie together performance data from across the platform. This modular setup supports extensibility and maintenance.

### b. Architectural Styles

BrightBoard is structured around a client-server architecture, a well-established architectural style that separates the user interface (client) from the core application logic and data handling (server). This separation of concerns ensures that the system remains maintainable, scalable, and easy to deploy in diverse environments. The frontend and backend operate independently yet communicate seamlessly via a well-defined interface, allowing for modular development and the potential for future upgrades or third-party integrations.

On the client side, BrightBoard uses React, a modern JavaScript library for building user interfaces. React was chosen for its component-based architecture, which promotes code reusability, state management, and fast rendering performance. This makes it ideal for delivering an interactive and responsive user experience across various devices, including desktops, tablets, and smartphones. Components are responsible for rendering views, managing user interactions (like course enrollment or quiz taking), and sending asynchronous requests to the backend via RESTful APIs.

The backend is implemented using Node.js with the Express.js framework. This combination allows for fast, non-blocking server-side logic and robust API creation. Express handles routing, middleware, and HTTP request management, serving as the backbone of server-side operations. Business logic, such as user authentication, lesson processing, quiz scoring, and progress tracking, is processed server-side to ensure consistency, security, and centralized control.

BrightBoard also integrates the Model-View-Controller (MVC) architectural pattern within its backend logic. This layered structure enhances clarity and separation within the codebase:

- Models are created using Mongoose, an ODM (Object Data Modeling) library for MongoDB. These schemas define the structure and constraints of data entities such as users, courses, lessons, and quizzes.



- Controllers manage the business logic and serve as intermediaries between models and views. For example, when a student submits a quiz, the controller validates the data, calculates the score, and updates the relevant records.
- Views, while traditionally rendered server-side in MVC, are managed entirely on the client side via React components in BrightBoard’s architecture. This hybrid model allows for a clean API-driven interface and decouples backend logic from frontend rendering responsibilities.

This combination of client-server and MVC styles provides several advantages. It allows teams to work in parallel on frontend and backend components, improves scalability by enabling horizontal distribution (e.g., load-balancing frontend servers), and simplifies debugging by keeping concerns well-organized. Additionally, the system is well-prepared for future enhancements, such as adding real-time features via WebSockets or expanding to mobile platforms using React Native.

In summary, BrightBoard’s architectural design ensures both flexibility and robustness, using tried-and-true patterns that support the project’s goals of maintainability, user responsiveness, and straightforward deployment.

### **c. Mapping Subsystems to Hardware**

BrightBoard is designed to operate across multiple machines, following a distributed system model. The frontend runs in the user's web browser, serving both students and instructors. The backend operates on a remote server, handling all authentication, course logic, and data storage. The database, MongoDB, may be hosted either locally or on a cloud instance, depending on deployment needs. This setup ensures that user-facing interfaces remain lightweight, while processing and data management occur server-side.

### **d. Connectors and Network Protocols**

BrightBoard relies on a combination of well-established web communication protocols and frameworks to ensure reliable, secure, and efficient data exchange between system components. The architecture is intentionally designed around widely adopted technologies to promote compatibility, ease of deployment, and long-term scalability.

At the core of communication between the frontend (React) and the backend (Node.js/Express.js) is the HTTP/HTTPS protocol, with all client-server interactions conducted over HTTPS to ensure encrypted and secure data transmission. HTTPS (Hypertext Transfer Protocol Secure) is essential for protecting sensitive user data, such as login credentials and quiz results, against man-in-the-middle attacks and other forms of interception during transmission.

The application uses a RESTful API architecture, enabling the frontend to interact with backend resources in a stateless manner. Each API endpoint corresponds to specific actions within the system, such as creating courses, submitting quiz responses, or retrieving user progress. RESTful APIs ensure a consistent and scalable interface that allows future enhancements, such as third-party integrations or mobile clients, without major architectural changes.

For authentication and access control, BrightBoard uses JWT (JSON Web Tokens). When a user logs in, the server issues a token that is securely stored on the client side (typically in local storage or cookies). This token is included in the header of subsequent requests, allowing the server to verify the user's identity and role without the need for session persistence. JWT is well-suited for distributed systems and improves performance by avoiding frequent lookups in a session store.

The backend also uses MongoDB, a NoSQL, document-oriented database, to persist application data. Communication with MongoDB is handled through Mongoose, an Object Data Modeling (ODM) library for Node.js that defines schemas, validation rules, and query logic. This structured approach helps maintain data integrity while still benefiting from the flexibility of a NoSQL system.

In terms of connectors, the application stack is composed entirely of web-based protocols and libraries, including:

- **HTTPS** for encrypted client-server communication
- **HTTP REST** endpoints for functional API interaction
- **JWT** for stateless authentication and session management
- **MongoDB** as the database backend, connected via Mongoose ODM

There is no use of low-level socket communication (e.g., TCP sockets or WebSockets), as BrightBoard does not require real-time updates or peer-to-peer messaging. Its event-driven but non-real-time nature makes RESTful HTTP over HTTPS the ideal choice.

This architecture and protocol stack together offer a secure, scalable, and maintainable foundation that aligns well with the needs of a modern, web-based learning management system.

#### **e. Global Control Flow**

BrightBoard operates as an event-driven system, meaning the application responds to user-generated actions such as logging in, enrolling in a course, viewing lessons, or completing

quizzes. These actions do not follow a strict sequence; instead, they trigger specific server-side responses and dynamically update the user interface based on the user's role and progress. This interaction model ensures flexibility and a personalized learning experience.

- **Execution Orderliness:**

The flow of execution in BrightBoard is non-linear and entirely driven by user input. For instance, one student might choose to take a quiz immediately after enrolling in a course, while another might prefer to read through all available lessons before attempting any assessments. This flexibility supports self-paced learning and accommodates a variety of learning styles. The system is designed to allow users to navigate content in the order that best fits their needs, rather than enforcing a rigid, predetermined path.

- **Time Dependency:**

BrightBoard is not a real-time system and does not depend on timers or continuous scheduling mechanisms. However, the system does incorporate timestamps on quiz submissions to record the exact date and time each quiz was completed. This information is valuable for both students and instructors, as it provides transparency, helps track progress over time, and may be used for analytics or time-based feedback. Despite this time-aware feature, the overall system remains asynchronous and event-driven, with no hard real-time constraints or requirements for continuous uptime synchronization.

## **f. Hardware Requirements**

BrightBoard is designed with modest hardware requirements to ensure ease of deployment and broad accessibility for both instructors and students. The system relies on commonly available technology components without the need for specialized hardware or sensors. Below is a breakdown of the essential hardware and system resources:

Client-Side Requirements:

- **Screen Display:**  
Minimum resolution of  $640 \times 480$  pixels to support responsive interface design and lesson readability.
- **Web Browser:**  
A modern web browser such as Chrome, Firefox, or Edge capable of rendering dynamic, JavaScript-based user interfaces.
- **Communication Network:**  
An active internet connection with a recommended minimum bandwidth of 56 Kbps (modern broadband is preferred for optimal performance and loading times).

- Input Devices:  
Standard keyboard and mouse or touchscreen interface for navigating lessons, quizzes, and dashboard interactions.

#### Server-Side Requirements:

- Processing Environment:  
A hosting environment capable of running a Node.js backend and connecting to a MongoDB database.
- Disk Storage:  
Minimum of 2 GB storage capacity, sufficient for handling uploaded PDF files, user data, and database growth.
- Network Availability:  
Continuous uptime with at least 99.9% availability to ensure user access and minimal downtime.
- Communication Protocols:  
Support for secure HTTPS connections to protect data transmission and comply with privacy requirements.
- Memory & CPU:  
Modest RAM and CPU resources suitable for handling lightweight requests, expected user concurrency, and real-time feedback (e.g., 2 GB RAM and a dual-core processor or cloud equivalent)

## 7. Project Management

---

### a. Plan of Work

Task	Developer	Est. Begin	Est. Complete
Set up GitHub repo & task board	Both	6/7/2025	6/8/2025
Define tech stack	Both	6/7/2025	6/8/2025
Design DB schema (users, courses, etc.)	Conner	6/7/2025	6/10/2025
Set up backend project (Express.js)	Conner	6/7/2025	6/10/2025
Wireframe UI: login, dashboard, courses	Shadow	6/7/2025	6/10/2025
Setup frontend scaffolding (React, CSS)	Shadow	6/9/2025	6/11/2025
Implement auth (login, register)	Conner	6/12/2025	6/15/2025
Role-based access (Instructor/Student)	Conner	6/12/2025	6/15/2025
Build login/register UI	Shadow	6/12/2025	6/14/2025
Responsive styling + validation	Shadow	6/13/2025	6/15/2025
Create course + lesson API	Conner	6/16/2025	6/19/2025
Instructor dashboard UI	Shadow	6/16/2025	6/19/2025
Lesson upload + preview	Shadow	6/18/2025	6/20/2025
Student course listing + enroll API	Conner	6/23/2025	6/25/2025
Serve lessons securely (student view)	Conner	6/24/2025	6/26/2025
Student dashboard UI (browse/enroll/view)	Shadow	6/23/2025	6/26/2025
Add mobile responsiveness	Shadow	6/25/2025	6/27/2025
Quiz creation API (MCQs)	Conner	6/30/2025	7/2/2025
Quiz-taking + feedback API	Conner	7/2/2025	7/4/2025
Quiz builder UI (MCQs)	Shadow	6/30/2025	7/2/2025
Quiz interface + feedback design	Shadow	7/2/2025	7/4/2025
Progress tracking API	Conner	7/7/2025	7/9/2025
Visual progress UI (bars, badges)	Shadow	7/7/2025	7/9/2025
UI polish + accessibility	Shadow	7/9/2025	7/11/2025
User docs + test cases	Shadow	7/9/2025	7/11/2025

Final testing + bug fixes	Both	7/10/2025	7/11/2025
Deploy app to server	Both	7/11/2025	7/12/2025
Prepare final demo & presentation	Both	7/14/2025	7/18/2025

## 8. References

---

Almrashdeh, I. A., et al. "Distance Learning Management System Requirements from Student's Perspective." *Journal of Theoretical and Applied Information Technology*, vol. 24, no. 1, 2011, pp. 17–27.

Frاند, Jason L. "The Information Age Mindset: Changes in Students and Implications for Higher Education." *Educause Review*, vol. 35, no. 5, 2000, pp. 14–24.

Rhode, Jason, et al. "Understanding Faculty Use of the Learning Management System." *Online Learning*, vol. 21, no. 3, 2017, pp. 68–86. <https://doi.org/10.24059/olj.v21i3.1217>.

Turnbull, Deborah, Rajeev Chugh, and Jo Luck. "Learning Management Systems, An Overview." *Encyclopedia of Education and Information Technologies*, edited by Arthur Tatnall, Springer, 2020. [https://doi.org/10.1007/978-3-030-10576-1\\_248](https://doi.org/10.1007/978-3-030-10576-1_248).