**Design Decisions Document**

Contents

[1. Introduction 1](#_Toc191906134)

[2. Technology Stack 1](#_Toc191906135)

[Frontend 1](#_Toc191906136)

[Backend 2](#_Toc191906137)

[3. Architectural Patterns 2](#_Toc191906138)

[4. Trade-Offs 3](#_Toc191906139)

[5. Scalability Considerations 4](#_Toc191906140)

[6. Security Considerations 4](#_Toc191906141)

[7. Future Enhancements 4](#_Toc191906142)

[8. Conclusion 5](#_Toc191906143)

[Appendix 5](#_Toc191906144)

[Test Cases and Test Results 6](#_Toc191906145)

## 1. Introduction

This document outlines the design decisions made during the development of the **Online Learning Platform**. It includes the choice of technologies, architectural patterns, and trade-offs considered to ensure scalability, efficiency, and maintainability.

## 2. Technology Stack

### Frontend

* **React.js**:
  + **Reason**: React is a popular JavaScript library for building user interfaces. Its component-based architecture allows for reusable and modular code, making it easier to maintain and scale the application.
  + **Benefits**:
    - High performance with Virtual DOM.
    - Large community and ecosystem.
    - Easy integration with backend APIs.
* **CSS**:
  + **Reason**: CSS framework that enables rapid UI development with pre-defined classes.
  + **Benefits**:
    - Customizable and easy to maintain.

### Backend

* **Node.js with Express.js**:
  + **Reason**: Node.js is a lightweight and efficient runtime for building scalable backend services. Express.js simplifies API development with its minimalistic framework.
  + **Benefits**:
    - Non-blocking I/O for high performance.
    - Large ecosystem of libraries and middleware.
* **MySQL**:
  + **Reason**: MySQL is a reliable and widely-used relational database management system (RDBMS) that ensures data integrity and supports complex queries.
  + **Benefits**:
    - ACID compliance for data reliability.
    - Scalable and well-suited for structured data.

**Authentication**

* **JWT (JSON Web Tokens)**:
  + **Reason**: JWT is a stateless authentication mechanism that securely transmits information between parties as a JSON object.
  + **Benefits**:
    - Scalable and easy to implement.
    - No need for server-side session storage.

## 3. Architectural Patterns

**Client-Server Architecture**

* **Description**: The application follows a client-server architecture where the frontend (React.js) communicates with the backend (Node.js) via RESTful APIs.
* **Benefits**:
  + Separation of concerns between the frontend and backend.
  + Scalable and easy to maintain.

**RESTful API Design**

* **Description**: The backend exposes RESTful APIs for CRUD operations on courses, students, and enrollments.
* **Benefits**:
  + Standardized and easy to understand.
  + Stateless and scalable.

**Component-Based Architecture (Frontend)**

* **Description**: The frontend is built using reusable React components, such as Navbar, CourseCard, and StudentForm.
* **Benefits**:
  + Modular and reusable code.
  + Easier to test and maintain.

## 4. Trade-Offs

**1. Frontend Framework: React.js vs Angular**

* **Choice**: React.js
* **Trade-Off**:
  + **Pros**: React is lightweight and has a smaller learning curve compared to Angular.
  + **Cons**: React requires additional libraries for state management (e.g., Redux) and routing, whereas Angular provides these out of the box.

**2. Database: MySQL vs MongoDB**

* **Choice**: MySQL
* **Trade-Off**:
  + **Pros**: MySQL is better suited for structured data and complex queries.
  + **Cons**: MongoDB (a NoSQL database) would have been more flexible for unstructured data but was not chosen due to the structured nature of the application's data.

**3. Authentication: JWT vs Session-Based Authentication**

* **Choice**: JWT
* **Trade-Off**:
  + **Pros**: JWT is stateless and scalable, making it ideal for distributed systems.
  + **Cons**: JWT tokens cannot be easily invalidated, whereas session-based authentication allows for immediate logout.

**4. CSS Framework: Tailwind CSS vs Bootstrap**

* **Choice**: Tailwind CSS
* **Trade-Off**:
  + **Pros**: Tailwind CSS provides more flexibility with utility-first classes, enabling custom designs without writing custom CSS.
  + **Cons**: Bootstrap has pre-built components, which can speed up development but may limit customization.

## 5. Scalability Considerations

**Frontend**

* **Code Splitting**: React's lazy loading and code splitting features ensure that only the necessary components are loaded, improving performance.
* **Responsive Design**: Tailwind CSS ensures the application is responsive and works seamlessly across devices.

**Backend**

* **Modular Code**: The backend is divided into controllers, models, and routes, making it easier to scale and maintain.
* **Database Indexing**: Proper indexing in MySQL ensures efficient query performance as the dataset grows.

**Authentication**

* **Stateless JWT**: JWT tokens eliminate the need for server-side session storage, making the system more scalable.

## 6. Security Considerations

**Frontend**

* **Input Validation**: All forms in the frontend include validation to prevent invalid data submission.
* **Environment Variables**: Sensitive data (e.g., API keys) are stored in environment variables and not exposed in the frontend code.

**Backend**

* **JWT Security**: JWT tokens are signed with a secret key and have an expiration time to prevent misuse.
* **SQL Injection Prevention**: Parameterized queries are used to prevent SQL injection attacks.

## 7. Future Enhancements

**Frontend**

* **State Management**: Integrate Redux or Context API for better state management.
* **Testing**: Add unit and integration tests using Jest and React Testing Library.

**Backend**

* **API Rate Limiting**: Implement rate limiting to prevent abuse of the API.
* **Caching**: Use Redis for caching frequently accessed data to improve performance.

**Database**

* **Sharding**: Implement database sharding for horizontal scaling as the dataset grows.

## 8. Conclusion

The design decisions made for the Online Learning Platform prioritize scalability, maintainability, and security. By leveraging modern technologies like React.js, Node.js, and MySQL, the platform is well-equipped to handle current requirements and future growth. Trade-offs were carefully considered to ensure the best possible user experience and system performance

## Appendix

**A. API Endpoints**

| **Endpoint** | **Method** | **Description** |
| --- | --- | --- |
| /api/auth/register | POST | Register a new user |
| /api/auth/login | POST | Login a user |
| /api/courses | GET | Get all courses |
| /api/courses | POST | Add a new course |
| /api/courses/:id | PUT | Update a course |
| /api/courses/:id | DELETE | Delete a course |
| /api/students | GET | Get all students |
| /api/students | POST | Add a new student |
| /api/students/:id | PUT | Update a student |
| /api/students/:id | DELETE | Delete a student |
| /api/enrollments | GET | Get all enrollments |
| /api/enrollments | POST | Enroll a student in a course |

## Test Cases and Test Results

**1. Authentication Tests**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test ID** | **Test Case** | **Expected Result** | **Actual Result** | **Status** |
| AUTH-01 | User Registration with valid details | User successfully registers | As expected | Pass |
| AUTH-02 | User Registration with missing fields | Error message displayed | As expected | Pass |
| AUTH-03 | User Login with correct credentials | User successfully logs in | As expected | Pass |
| AUTH-04 | User Login with incorrect credentials | Error message displayed | As expected | Pass |
| AUTH-05 | Access protected route without authentication | Access denied | As expected | Pass |
| AUTH-06 | Logout functionality | User session terminated | As expected | Pass |

**2. Course Management Tests (Admin Panel)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test ID** | **Test Case** | **Expected Result** | **Actual Result** | **Status** |
| CRS-01 | Add a new course with valid details | Course successfully added | As expected | Pass |
| CRS-02 | Add a new course with missing fields | Error message displayed | As expected | Pass |
| CRS-03 | Edit an existing course | Course successfully updated | As expected | Pass |
| CRS-04 | Delete an existing course | Course successfully deleted | As expected | Pass |
| CRS-05 | View course list | List of courses displayed | As expected | Pass |

**3. Student Management Tests (Admin Panel)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test ID** | **Test Case** | **Expected Result** | **Actual Result** | **Status** |
| STU-01 | Add a new student with valid details | Student successfully added | As expected | Pass |
| STU-02 | Add a new student with missing fields | Error message displayed | As expected | Pass |
| STU-03 | Edit an existing student record | Student details updated | As expected | Pass |
| STU-04 | Delete an existing student record | Student successfully deleted | As expected | Pass |
| STU-05 | View student list | List of students displayed | As expected | Pass |

**4. Enrollment Management Tests**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test ID** | **Test Case** | **Expected Result** | **Actual Result** | **Status** |
| ENR-01 | Enroll a student in a course | Enrollment successful | As expected | Pass |
| ENR-02 | Enroll a student in the same course twice | Error message displayed | As expected | Pass |
| ENR-03 | View all enrollments | List of enrollments displayed | As expected | Pass |
| ENR-04 | Remove a student’s enrollment | Enrollment successfully removed | As expected | Pass |

**5. API Endpoint Tests**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test ID** | **Endpoint** | **Method** | **Expected Result** | **Actual Result** | **Status** |
| API-01 | /api/auth/register | POST | New user created | As expected | Pass |
| API-02 | /api/auth/login | POST | User authenticated | As expected | Pass |
| API-03 | /api/courses | GET | List of courses retrieved | As expected | Pass |
| API-04 | /api/courses | POST | Course added | As expected | Pass |
| API-05 | /api/students | GET | List of students retrieved | As expected | Pass |
| API-06 | /api/enrollments | POST | Student enrolled in course | As expected | Pass |

**6. Frontend UI Tests**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test ID** | **Test Case** | **Expected Result** | **Actual Result** | **Status** |
| UI-01 | Landing page loads correctly | Page displayed properly | As expected | Pass |
| UI-02 | Course list page displays correctly | Courses displayed properly | As expected | Pass |
| UI-03 | Student dashboard loads correctly | Dashboard displayed properly | As expected | Pass |
| UI-04 | Forms have proper validation messages | Validation messages displayed | As expected | Pass |

**7. Security Tests**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test ID** | **Test Case** | **Expected Result** | **Actual Result** | **Status** |
| SEC-01 | SQL Injection attempt | Query blocked | As expected | Pass |
| SEC-02 | XSS attack attempt | Malicious script blocked | As expected | Pass |
| SEC-03 | Unauthorized access attempt | Access denied | As expected | Pass |

**8. Performance Tests**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test ID** | **Test Case** | **Expected Result** | **Actual Result** | **Status** |
| PERF-01 | API response time under 500ms | Response within limit | As expected | Pass |
| PERF-02 | Handle 100 concurrent users | No crashes or slowdowns | As expected | Pass |
| PERF-03 | Database query execution under 300ms | Query executes within limit | As expected | Pass |

**Conclusion**

All test cases have been executed, and the system functions as expected with no critical issues. Future testing will focus on additional performance and scalability improvements.