Learning Path Platform - Project Report

Table of Contents

1. Executive Summary:

The Learning Path Platform is a comprehensive web-based education management system designed to facilitate structured learning through customizable learning paths, assessments, and certifications. This application enables educators to create learning paths while allowing students to track their progress and earn certificates upon completion. Key highlights:

- Full-stack web application with React frontend and Spring Boot backend
- Secure user authentication with JWT
- Learning path management and progress tracking
- Assessment system with automatic grading
-Certificate generation and verification - Data visualization for progress monitoring.

2. Project Overview:

Objective: To create a user-friendly, secure, and feature-rich learning management platform that helps users follow structured learning paths and track their educational progress.

Target Audience:

- Students seeking structured learning paths
- Educators creating learning content
- Organizations providing training programs

Core Functionalities:

- User account management

- Learning path creation and management
- Progress tracking
- Assessment system
- Certificate generation
- Search and discovery of learning paths

Technologies Used:

- Frontend: React, Material-UI
- Backend: Java 17, Spring Boot
- Database: MySQL
- Authentication: JWT
- Build Tool: Maven
- Version Control: Git

3. Technical Architecture:

The application follows a modern, scalable architecture:

Frontend:

- Framework: React.js
- UI Library: Material-UI
- HTTP Client: Axios
- Router: React Router DOM
- JWT Handling: jwt-decode

Backend:

- Framework: Spring Boot (Java 17)
- Security: Spring Security with JWT
- Database: MySQL - ORM: JPA/Hibernate

4. Features and Functionality:

1. User Authentication

- Secure signup and login process
- Password encryption
- JWT-based session management

2. Dashboard
    - Overview of enrolled learning paths
    - Progress tracking
    - Recent activities
3. Learning Path Management
    - Create and edit learning paths
    - Add topics and resources
    - Track completion status
4. Assessment System
    - Create and take assessments
    - Automatic grading
    - Progress tracking
5. Certificate Management
    - Generate certificates
    - Download certificates
    - Verify certificates
6. Progress Tracking
    - Visual progress indicators
    - Detailed progress reports
    - Achievement tracking

5. Development Process:

    The project follows a structured development process:

    1. Planning and Requirement Analysis
    2. Design (System & Database)
    3. Implementation

4. Testing
5. Deployment
6. Maintenance

6. Backend Implementation:

```
backend/
├── src/
│   ├── main/
│   │   ├── java/
│   │   │   └── com/
│   │   │       └── learning/
│   │   │           └── path/
│   │   │               ├── config/
│   │   │               ├── controller/
│   │   │               ├── model/
│   │   │               ├── repository/
│   │   │               ├── security/
│   │   │               ├── service/
│   │   │               └── BackendApplication.java
│   │   └── resources/
│   └── test/
└── pom.xml
```

Key Components:

1. Controllers
    - AuthController
    - LearningPathController
    - TopicController
    - AssessmentController
    - CertificateController
    - UserController
2. Services
    - Authentication Service
    - Learning Path Service
    - Assessment Service
    - Certificate Service
    - Progress Service
3. Models
    - User

- LearningPath
- Topic
- Assessment
- Certificate
- Progress

7. Backend Implementation:

```
frontend/
├── src/
│   ├── components/
│   │   ├── Layout.js
│   │   ├── ErrorBoundary.js
│   │   └── PrivateRoute.js
│   ├── pages/
│   │   ├── Login.js
│   │   ├── Register.js
│   │   ├── Dashboard.js
│   │   ├── LearningPaths.js
│   │   ├── Assessment.js
│   │   ├── Profile.js
│   │   └── Certificates.js
│   ├── services/
│   │   └── api.js
│   ├── context/
│   │   └── AuthContext.js
│   └── App.js
├── package.json
└── README.md
```

Key Components:

1. Pages:
   - Dashboard: Main user interface showing learning progress
   - LearningPaths: Display and manage learning paths
   - Assessment: Interface for taking assessments
   - Profile: User profile management
   - Certificates: Certificate management and display
2. Components:

- Layout: Common layout structure
- ErrorBoundary: Error handling component
- PrivateRoute: Route protection component

3. Services:
   - API service for backend communication
   - Authentication service
   - File handling service

8. Database Design:

1. Users:

```
CREATE TABLE users (
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    username VARCHAR(255) UNIQUE,
    password VARCHAR(255),
    email VARCHAR(255) UNIQUE,
    role VARCHAR(50),
    created_at TIMESTAMP,
    updated_at TIMESTAMP
);
```

2. learning_paths:

```
CREATE TABLE learning_paths (
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    title VARCHAR(255),
    description TEXT,
    creator_id BIGINT,
    difficulty VARCHAR(50),
    estimated_hours INT,
    FOREIGN KEY (creator_id) REFERENCES users(id)
);
```

3. topics:

```
CREATE TABLE topics (
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(255),
    description TEXT,
    learning_path_id BIGINT,
    order_index INT,
    FOREIGN KEY (learning_path_id) REFERENCES learning_paths(id)
);
```

4. assessments:

```
CREATE TABLE assessments (
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    topic_id BIGINT,
    title VARCHAR(255),
    description TEXT,
    FOREIGN KEY (topic_id) REFERENCES topics(id)
);
```

5. certificates:

```
CREATE TABLE certificates (
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    user_id BIGINT,
    learning_path_id BIGINT,
    issue_date TIMESTAMP,
    certificate_number VARCHAR(255) UNIQUE,
    FOREIGN KEY (user_id) REFERENCES users(id),
    FOREIGN KEY (learning_path_id) REFERENCES learning_paths(id)
);
```

9. API Design:

RESTful API Endpoints:

POST /api/auth/register - Register new user

POST /api/auth/login - User login

GET /api/learning-paths - Get all learning paths

GET /api/learning-paths/{id} - Get specific learning path

POST /api/learning-paths - Create learning path

PUT /api/learning-paths/{id} - Update learning path

DELETE /api/learning-paths/{id} - Delete learning path

GET /api/topics/learning-path/{learningPathId} - Get topics by learning path

POST /api/topics - Create topic

PUT /api/topics/{id} - Update topic

DELETE /api/topics/{id} - Delete topic

GET /api/assessments/topic/{topicId} - Get assessments by topic

POST /api/assessment-attempts - Submit assessment attempt

POST /api/certificates/generate/{userId}/{learningPathId} - Generate certificate

GET /api/certificates/user/{userId} - Get user certificates

GET /api/certificates/download/{certificateId} - Download certificate

10. Authentication & Authorization:

Library: jjwt for JWT handling
Token Validity: 24 hours

Security Features

1. Password Security:
    - BCrypt encryption
    - Minimum password requirements
    - Password validation rules
2. Role-Based Access:
    - USER: Regular user access
    - ADMIN: Full system access
3. Secured Endpoints:
    - All endpoints except /api/auth/* require authentication.
    - Role-specific endpoint restrictions

11. Security Measures:

Backend Security:

1. Spring Security Configuration:
    - CORS configuration
    - CSRF protection
    - JWT authentication filter
    - Security headers
2. Data Protection:
    - Password encryption
    - Sensitive data encryption
    - Input validation

API Security:

1. Request Validation:
    - Input sanitization
    - Request size limits
    - Rate limiting

2. Error Handling:
    - Custom error responses
    - Logging of security events
    - No sensitive data in error messages

12. Setup and Installation:

1. Prerequisites:

Required Software Versions

- Java 17

- Node.js 16+ and npm

- MySQL 8.0+

- Maven 3.8+

1. Database Setup

Login to MySQL

mysql -u root -p

Create Database

CREATE DATABASE learning_path_db;

CREATE USER 'learningpath'@'localhost' IDENTIFIED BY 'your_secure_password';

GRANT ALL PRIVILEGES ON learning_path_db.* TO 'learningpath'@'localhost';

FLUSH PRIVILEGES;

2. Backend Setup:

1. Clone and Navigate:

git clone https://github.com/yourusername/learning-path-platform.git

cd learning-path-platform/backend

2. Configure src/main/resources/application.properties:

```
# Database Configuration
spring.datasource.url=jdbc:mysql://localhost:3306/learning_path_db
spring.datasource.username=learningpath
spring.datasource.password=your_secure_password
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

# Hibernate Configuration
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect

# Server Configuration
server.port=8080

# JWT Configuration
jwt.secret=your_jwt_secret_key_here
jwt.expiration=86400000

# Logging Configuration
logging.level.org.springframework.security=DEBUG
logging.level.com.learning.path=DEBUG
```

3. Update pom.xml dependencies:

```
<dependencies>
  <!-- Spring Boot Starter Dependencies -->
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
  </dependency>

  <!-- Database -->
  <dependency>
    <groupId>com.mysql</groupId>
    <artifactId>mysql-connector-j</artifactId>
    <scope>runtime</scope>
  </dependency>

  <!-- JWT -->
```

```
    <dependency>
        <groupId>io.jsonwebtoken</groupId>
        <artifactId>jjwt</artifactId>
        <version>0.9.1</version>
    </dependency>

    <!-- Validation -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-validation</artifactId>
    </dependency>

    <!-- Lombok -->
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <optional>true</optional>
    </dependency>
</dependencies>
```

Build and Run Backend:

```
# Clean and install dependencies
mvn clean install
```

Run the application

mvn spring-boot:run

3. Frontend Setup:

Navigate to frontend directory:

cd ../frontend

Create .env file in frontend root:

REACT_APP_API_URL=http://localhost:8080/api

REACT_APP_JWT_SECRET=your_jwt_secret_key_here

Install dependencies:

Using npm

npm install --legacy-peer-deps

Or using yarn

yarn install

Update package.json:

```
{
  "name": "learning-path-frontend",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@emotion/react": "^11.11.0",
    "@emotion/styled": "^11.11.0",
    "@mui/icons-material": "^5.11.16",
```

```
  "@mui/lab": "^5.0.0-alpha.130",
  "@mui/material": "^5.13.1",
  "@mui/x-data-grid": "^6.4.0",
  "axios": "^1.4.0",
  "jwt-decode": "^3.1.2",
  "react": "^18.2.0",
  "react-dom": "^18.2.0",
  "react-router-dom": "^6.11.2",
  "react-scripts": "5.0.1"
},
"scripts": {
  "start": "react-scripts start",
  "build": "react-scripts build",
  "test": "react-scripts test",
  "eject": "react-scripts eject"
},
"eslintConfig": {
  "extends": [
    "react-app",
    "react-app/jest"
  ]
},
"browserslist": {
```

```
  "production": [
    ">0.2%",
    "not dead",
    "not op_mini all"
  ],
  "development": [
    "last 1 chrome version",
    "last 1 firefox version",
    "last 1 safari version"
  ]
 }
}
```

Run the frontend:

```
npm start
```

4. Verify Installation:

1. Backend Verification:

- Access: http://localhost:8080/api/health
- Expected response: {"status": "UP"}

2. Frontend Verification:

- Access: http://localhost:3000
- Should see the login page

3. Test Registration:

```
curl -X POST http://localhost:8080/api/auth/register \
```

```
-H "Content-Type: application/json" \
-d '{
  "username": "testuser",
  "password": "Test@123",
  "email": "test@example.com",
  "firstName": "Test",
  "lastName": "User"
}'
```

13. Future Enhancements:

1. **Technical Improvements:**
   - Implement caching system
   - Add real-time notifications
   - Implement file upload for learning materials
   - Add video streaming capabilities
2. **Feature Enhancements:**
   - Discussion forums for each learning path
   - Peer review system
   - Advanced analytics dashboard
   - Mobile application development
3. **Integration Possibilities:**
   - Third-party authentication providers
   - Payment gateway integration
   - Video conferencing integration
   - Content management system

14. Conclusion:

The Learning Path Platform successfully implements a comprehensive solution for online learning management. Key achievements include:

1. Robust Architecture:
    - Scalable backend design
    - Responsive frontend implementation
    - Secure authentication system
2. Feature Completeness:
    - Full learning path management
    - Assessment system
    - Certificate generation
    - Progress tracking
3. Security and Performance:
    - JWT-based security
    - Optimized database queries
    - Protected API endpoints

The platform provides a solid foundation for future enhancements and can be extended to meet growing educational needs. The modular architecture ensures easy maintenance and scalability for future requirements.

Appendix

A. API Response Formats

B. Database Schema Details

C. Security Configuration Details

D. Testing Documentation