PixelLMS Documentation

Table of Contents

Overview

PixelLMS is a dynamic web-based platform designed to streamline the process of managing courses, assignments, events, and facilitating discussions for educators and students. The system is built with ease of use and flexibility in mind to cater to various academic needs. The platform allows teachers to create and manage courses and assignments, while students can track assignments and participate in academic discussions. The system provides a comprehensive set of features aimed at improving the overall learning experience.

Features

1. Course Management

- **Add, Update, and Delete Courses**: Teachers and admins can add new courses to the system. They can also update or delete existing courses. This ensures that the course offerings remain up-to-date as per the academic requirements.
- **Role-based Permissions**: Only admins and teachers have the permissions to manage courses, ensuring that they have full control over course content.

2. Assignment Management

- **Assigning Assignments**: Teachers can assign assignments to students, specifying deadlines and other relevant details.
- **Tracking Assignments**: Teachers can monitor the progre==ss of assignments, check submission statuses, and grade them efficiently.
- **Student Participation**: Students can view and submit assignments, track their progress, and receive feedback.

3. Login and User Management

- **Secure Login**: Users can securely log in to the system using unique credentials, which are hashed using SHA256 or MD5 (passwords mainly), with the implementation of the Factory Design Pattern for generating hash functions.
- **User Roles**: The system supports multiple user roles, including admins, teachers, and students. Admins can create accounts for teachers and students, while teachers can only create student accounts.

4. Event Management

- **Manage Events**: Admins and teachers can create, update, and delete events. Events can be related to courses, school-wide activities, and other academic events.
- **Calendar Integration**: The platform integrates an event calendar, allowing users to view upcoming events in a structured format.

5. Discussion Management

- **Create Specific Discussions**: Teachers can create and manage discussions related to course content, assignments, and other academic topics. Students can participate in these discussions, asking questions and sharing ideas, but they cannot create new discussions.

6. Student Registration

- **Teacher Account Creation**: Teachers can create student accounts, ensuring that students have access to the LMS to track their assignments and courses.
- **Admin Privileges**: Admins have the ability to create accounts for teachers, students, and other admins. This ensures efficient user management.

7. Authorization and Permissions

The system checks for proper permissions and authorization using various security mechanisms to ensure only authorized users can perform certain actions. This is managed through a combination of command patterns and hashed password authentication.

Technical Details

Architecture

The system follows the Model-View-Controller (MVC) architecture to ensure modularity and easy maintenance. The architecture allows the separation of concerns, making it easier to manage both the frontend and backend components independently.

- **Frontend**: Built using Vue.js, a progressive JavaScript framework that facilitates the creation of dynamic and responsive user interfaces.
- **Backend**: Developed with Spring Boot, which provides a solid foundation for building scalable and secure Java applications.

Frameworks and Libraries

- **Spring Boot**: A Java framework for building backend services. It simplifies Java development and follows the convention-over-configuration principle.
- **Vue.js**: A JavaScript framework used to create a reactive and easy-to-maintain frontend.
- **JPA**: Although JPA is not used for model interaction, raw SQL queries are implemented for database interaction.

Security and Authentication

- **Hashing**: Passwords are hashed using SHA256 or MD5 hashing algorithms to ensure that sensitive user information is securely stored. The Factory Design Pattern is used to select the appropriate hashing function.
- **Bearer Token Authentication**: The system uses Bearer token authentication for securing API endpoints. Users must provide a valid token to access specific resources or perform certain actions, ensuring secure communication between the frontend and backend.
- **Role-based Access Control (RBAC)**: Different user roles (admin, teacher, student) have different levels of access to various features, ensuring that only authorized users can perform certain actions.

- **GlobalExceptionHandler**: A global exception handler is implemented using Spring's @ControllerAdvice to handle all exceptions and provide detailed error messages to the users.

Design Patterns

1. Factory Design Pattern

The Factory Design Pattern is used for creating hash functions for password security. Depending on the system's configuration, the appropriate hashing algorithm (SHA256 or MD5) is selected dynamically.

2. Strategy Design Pattern

The Strategy Design Pattern is used to implement row mapping from database results to Java objects. This pattern allows different strategies for mapping based on the query structure or result set.

3. Command Pattern

The Command Pattern is used for handling various user actions such as course creation, assignment submission, and event management. Each action is encapsulated as an object and executed through a command registry, providing flexibility and scalability in handling requests.

4. GlobalExceptionHandler (Controller Advice)

The GlobalExceptionHandler is implemented to intercept and handle exceptions globally across the application. This ensures that users receive meaningful error messages when something goes wrong.

Transactions and Procedures

Transactional Integrity

The system makes use of transactions to ensure atomicity in operations. For instance, when creating or updating a course, all related actions (e.g., updating the database, sending notifications) are part of a single transaction. This guarantees that either all changes are committed, or none are, preventing partial updates that could lead to data inconsistency.

Procedures

Various procedures are implemented to ensure data consistency and integrity in the system. These procedures include:

- **Course Creation Procedure**: This procedure involves adding a new course and its associated details to the database while ensuring that the teacher, students, and relevant events are properly linked to the course.
- **Assignment Submission Procedure**: This procedure includes the steps involved in submitting assignments, updating submission statuses, and grading assignments.
- **Event Creation Procedure**: When an event is created, the procedure ensures that all event details are correctly saved and linked to the relevant courses and users.

Testing

Test-Driven Development (TDD)

The project follows Test-Driven Development (TDD) principles. Tests are written before the actual implementation to ensure that each feature works as expected. JUnit is used for backend testing, while Vue Test Utils and Jest are used for frontend testing.

Types of Tests:

- **Unit Tests**: Test individual components and functions in isolation to ensure they perform as expected.
- **Integration Tests**: Test the interaction between different components of the system to ensure they work together seamlessly.

Robustness Testing:

For robustness checks, Python-generated dummy data (1 million rows) was used to simulate real-world usage and ensure the system could handle large amounts of data without performance degradation. Additionally, testing was carried out using Postman for verifying API endpoints.

Agile Scrum Methodology

The project followed Agile Scrum methodology to manage the development process efficiently. Jira was used to track tasks, create backlogs, and manage sprints. The team divided the work into smaller tasks and completed them within time-boxed sprints. Weekly meetings were scheduled to discuss the progress, define new tasks, and assign them to the team members.

The key aspect of our approach was the rotation of tasks. The team ensured that each member gained experience with different types of tasks across the project. For example, if one member created the controller in User Management, they would not be assigned the same task in Course Management. This rotation allowed everyone to be involved in various components of the project, enhancing their skills and providing a comprehensive view of the system.

Regarding version control, we did not need to push our work to GitHub throughout the development process because we used Live Share in Visual Studio for real-time collaboration. This enabled the team to work synchronously, making it easier to assist each other, agree on similar structures, entities, and folder names, and ensure consistency across the project. As a result, the team didn't push code to GitHub until the final stage when the project was ready for deployment. This approach helped maintain a streamlined workflow and minimized conflicts during development.

At the end of each sprint, the team would review the completed tasks, re-prioritize new tasks, and plan for the next sprint, ensuring continuous progress and effective collaboration.

Notifications and Quizzes

Due to time constraints, notifications and quizzes were not fully implemented. However, these features are planned for future development to enhance the platform’s interactivity and engagement.

Installation and Deployment

1. **Run SQL Scripts**: Execute the SQL scripts to create and configure the database.

```
psql -U postgres -d pixel_db -f "dir\sql\database-DDL.sql" > dir\logs\DDL.log
psql -U postgres -d pixel_db -f "dir\sql\database-storedproc.sql" > dir\logs\Procedures.log
psql -U postgres -d pixel_db -f "dir\sql\database-transactions.sql" > dir\logs\Transactions.log
```

2. **Install Dependencies (Front END):**

```
npm install
```

3. **Build the Project (Front END):**

```
npm run build
```

4. **Integrate with Backend:** Move the generated files to the backend directory.

5. **Clean and Install Backend:**

```
./mvnw clean install
```

6. **Deploy:** Deploy the application in a Tomcat container.

Conclusion

Pixel LMS is a robust and flexible learning management system that supports a variety of academic activities such as course management, assignment tracking, event scheduling, and discussions. The system was developed using modern technologies such as Spring Boot and Vue.js, employing industry-standard design patterns like Factory, Strategy, and Command Patterns to enhance flexibility, scalability, and maintainability. The project followed Agile Scrum methodology using Jira, ensuring efficient task management and collaboration among team members.