# SOFTWARE REQUIREMENT SPECIFICATION FOR ASSESSMENT PORTAL

| | |
|---|---|
| **NAME** | ABINAYA P S |
| **ROLL NO** | 7376222AL105 |
| **DEPARTMENT** | AIML |
| **PROJECT ID** | 01 |
| **PROBLEM STATEMENT** | Assessment Platform |

## 1. INTRODUCTION

### OVERVIEW:

The Assessment Platform is a comprehensive system designed to facilitate the creation, administration, and evaluation of assessments in educational settings. It aims to provide educators with a versatile tool for conducting various types of assessments, including multiple-choice, short answer, and matching type questions.

### SCOPE:

This project aims to develop an assessment interface incorporating multiple-choice, short answer, and matching question formats, along with a compiler. The interface, built using React.js for the front end, Java with Spring Boot for the back end, and MySQL for the database, will facilitate automated grading for multiple-choice questions, manual grading for short answers by faculty, and interactive matching questions. API integration (OpenAPI, SOAP, or RESTful) will enhance interoperability. The scope includes seamless user experience, robust database management, and efficient communication between front and back end components, ensuring a comprehensive assessment solution tailored for educational environments.

## OBJECTIVES:

- Provide educators with a user-friendly platform for creating and administering assessments.
- Enable automatic evaluation of multiple-choice questions to save time and effort.
- Facilitate manual correction of short answer questions by faculty to ensure accurate assessment.
- Enhance the assessment experience for both educators and students through intuitive user interfaces and interactive features.

## 2. FUNCTIONAL REQUIREMENTS

### USE CASES:

1. Create Assessment: Educators can create new assessments by specifying the title, description, and types of questions.
2. Take Assessment: Students can access and take assessments assigned to them by educators.
3. Evaluate Assessment: Faculty members can review and evaluate assessments submitted by students, including automatic evaluation of multiple-choice questions and manual correction of short answer questions.
4. View Assessment Results: Students and educators can view assessment results, including scores and feedback.
5. Slots:Admin will create slots for the assessments.

### FUNCTIONAL FEATURES:

The proposed assessment interface will include the following functional features:

1. **User Authentication**: Secure user authentication and authorization mechanisms for students and faculty members to access the platform.

2. **Question Creation:** Faculty members can create and customize multiple-choice questions (MCQs), short answer questions, and match type questions.

3. **Question Bank:** A repository to store and manage a large collection of questions for easy retrieval and reuse in assessments.

4. **Automatic Evaluation:** MCQs will be automatically graded, with correct answers determined by the system.

5. **Manual Evaluation:** Faculty members can manually evaluate short answer questions, providing feedback and grades.

6. **Real-time Feedback**: Immediate feedback to students upon completing assessments, indicating correct and incorrect answers.

7. **Compiler Integration:** Integration of a compiler for students to write, compile, and execute code within the platform for coding assessments.

8. **Results Tracking:** Tracking and recording of assessment results for individual students, allowing faculty members to monitor progress and identify areas for improvement.

9. **Security:** Implementing robust security measures to protect sensitive student data and prevent unauthorized access or tampering.

10. **Intuitive Interface:** User-friendly interface design for easy navigation and interaction, catering to both students and faculty members.

11. **Notification System:** Sending notifications to students and faculty members for upcoming assessments, deadlines, and assessment results.

## 3. NON-FUNCTIONAL REQUIREMENTS

**Performance:**

- Response Time: < 2 seconds for page loads.
- Scalability: Support for up to 1000 concurrent users.

**Security:**

- Authentication: Use of secure authentication mechanisms such as JWT.
- Authorization: Role-based access control for different user roles.

**Reliability**:

- Availability: 99.9% uptime.
- Fault Tolerance: Automatic failover and recovery mechanisms.

**Usability:**

- Intuitive User Interface Design
- Accessibility Compliance (WCAG)

**Compatibility:**

- Browser Compatibility: Chrome, Firefox, Safari
- Device Compatibility: Desktop, Tablet, Mobile

**Legal and Compliance:**

- Compliance with GDPR regulations for data protection.

# 4. SYSTEM ARCHITECTURE

The system architecture for the assessment interface can be designed using a combination of frontend and backend technologies along with a database for data storage. Here's a high-level overview of the system architecture:

**1. Frontend (Client-side):**

- React.js: Utilized for building the user interface (UI) of the assessment platform. React.js offers component-based architecture, enabling modular development and efficient rendering of UI elements.
- HTML/CSS/JavaScript: Used for structuring the UI, styling, and implementing interactive features and client-side logic.

**2. Backend (Server-side):**

- Java with Spring Boot: Employed for building the backend server application. Spring Boot facilitates rapid development of production-ready Spring applications, providing features such as dependency injection, MVC framework, and RESTful web services.
- RESTful API: Designed to enable communication between the frontend and backend components. RESTful APIs allow for stateless communication over HTTP, providing scalability and interoperability.

**3. Database:**

- MySQL: Chosen as the relational database management system (RDBMS) for storing assessment-related data such as user accounts, questions, answers, and assessment results. MySQL offers robust transaction support, data integrity, and scalability.

**4. Integration:**

- Compiler Integration: Integration of a compiler component to enable students to write, compile, and execute code within the platform for coding assessments. This could involve utilizing third-party compiler APIs or libraries.

**5. Security:**

- Authentication & Authorization: Implementation of secure authentication mechanisms such as JSON Web Tokens (JWT) for user authentication and role-based access control (RBAC) for authorization. This ensures that only authorized users can access specific functionalities and data.

**6. Deployment:**

- Containerization: Containerization using Docker for packaging the application along with its dependencies into containers, providing consistency across different environments and simplifying deployment.
- Orchestration: Orchestration using Kubernetes for automating deployment, scaling, and management of containerized applications, ensuring high availability and scalability.

**7. Monitoring & Logging:**

- Logging: Implementation of logging mechanisms to capture system events, errors, and user activities for troubleshooting and auditing purposes.
- Monitoring: Integration of monitoring tools such as Prometheus and Grafana to monitor system performance, resource utilization, and availability in real-time.

## 5.ADVANCED FEATURES

The project will include advanced features to enhance the assessment interface.

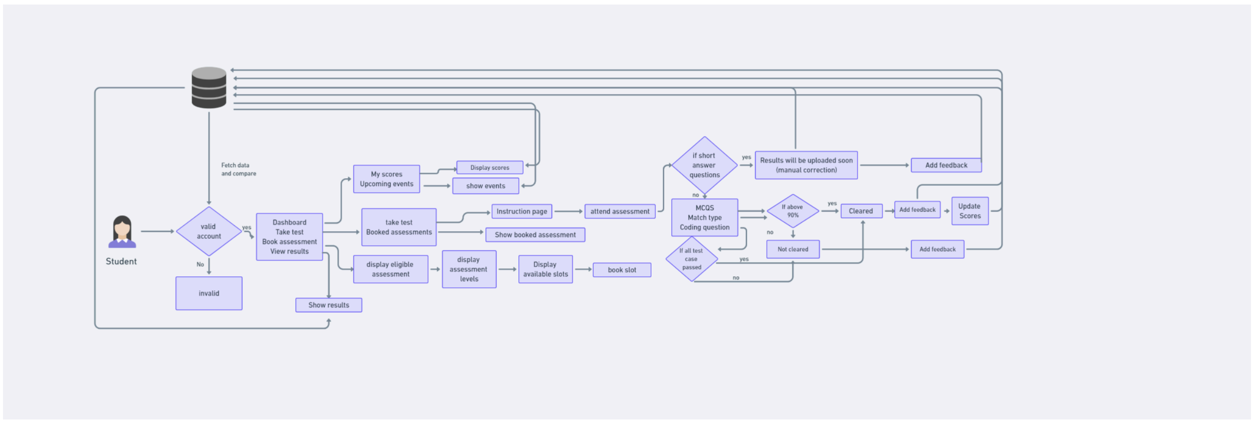
- A drag-and-drop functionality for matching questions will be implemented to improve user interaction.
- Each question will display the time taken to answer, providing valuable insights for both students and instructors.
- AI-generated questions will ensure a diverse and dynamic question bank.
- A feedback mechanism will be incorporated to allow students to provide feedback on questions and the overall assessment, aiding in continuous improvement.

These features aim to create a more engaging, insightful, and adaptive assessment experience for all users.

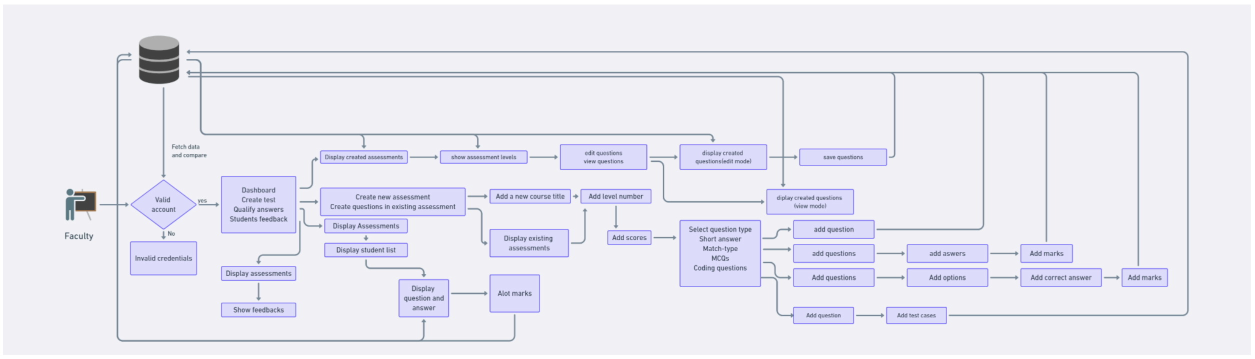
## STACK:

| | |
|---|---|
| FRONT END | React JS |
| BACK END | Java with Spring Boot |
| DATABASE | My SQL and PostgreSQL |

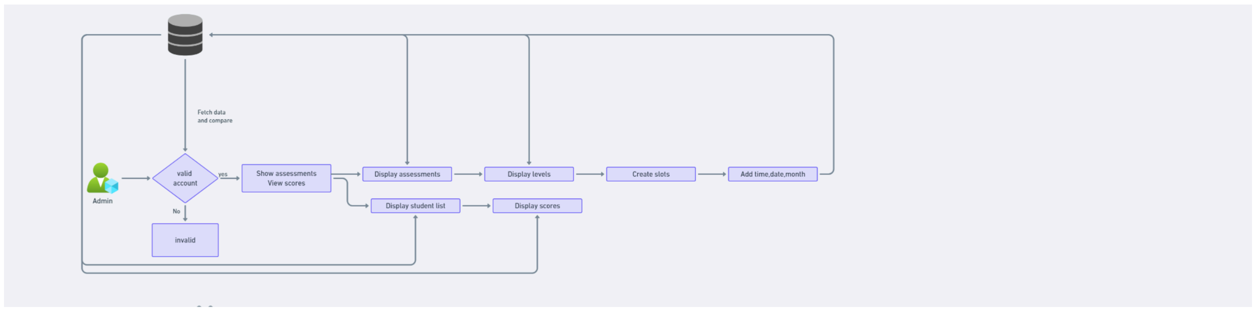
## USER INTERFACE:



## FACULTY INTERFACE:



## ADMIN INTERFACE:


Fetch data
and compare
Admin
valid
account
yes
No
invalid
Show assessments
View scores
Display assessments
Display levels
Create slots
Add time,date,month
Display student list
Display scores