# WEB BASED QUIZ PLATFORM

## Design Document

## C-DAC, Bangalore

**Version 1.0**


**Prepared By:**

Manoj Pentapalli – 240850120075
Mudit Kumar – 240850120079
Sudhanshu Patel – 240850120143
Utkarsh Tyagi – 240850120153
Vashu Chowdary – 240850120157

# INDEX

## Contents Pg.no

# Web-Based Quiz Platform - Design Document

## 1. Introduction

### 1.1 Purpose

The purpose of this document is to outline the design specifications for the **Web-Based Quiz Platform**, ensuring a structured and efficient development process. This platform will enable educators to create and manage quizzes while students can participate, receive scores, and view leaderboards. The system aims to enhance digital learning by automating quiz creation, submission, and evaluation.

### 1.2 Scope

This document defines the architecture, data design, and component interactions necessary to implement the Web-Based Quiz Platform. The platform will include:

- **Admin functionalities** for quiz creation, student management, and performance analysis.
- **Student functionalities** to attempt quizzes, view scores, and track progress.
- **Secure authentication mechanisms** to protect user data.
- **Database design** for efficient storage and retrieval of quizzes and scores.
- **Performance and security considerations** to ensure reliability.

### 1.3 Intended Audience

This document is intended for:

- **Project stakeholders** to understand the system design and implementation strategy.
- **Developers and engineers** working on the backend and frontend.
- **QA testers** to validate system functionality.
- **System administrators** responsible for deployment and maintenance.

### 1.4 References

- Software Requirement Specification (SRS) for Web-Based Quiz Platform
- Functional Requirement Specification (FRS)
- Database Design Best Practices
- REST API Standards for Web Applications

## 2. Acronyms, Terms, and Definitions

| Acronym | Definition |
|---|---|
| API | Application Programming Interface |
| CSV | Comma-Separated Values |
| DBMS | Database Management System |
| JWT | JSON Web Token |
| RBAC | Role-Based Access Control |
| REST | Representational State Transfer |
| UI | User Interface |

## 3. Assumptions and Constraints

### 3.1 Assumptions

- Users will have internet access to use the platform.
- Data entered by admins (quiz questions, answers) will be valid.
- Students will complete the quiz within the allotted time.

### 3.2 Constraints

- The system must handle multiple concurrent users efficiently.
- Quiz questions and scores must be securely stored and protected.
- The platform should support mobile and desktop browsers.

## 4. Basic Design Approach

- **Modular Design:** Separate modules for quiz management, authentication, and reporting.
- **Scalability:** Designed to support a growing number of users and quizzes.
- **Security:** Implementation of encryption, secure login, and role-based access control.
- **Cloud-Based Infrastructure:** Deployment using scalable cloud services.
- **Database Optimization:** Efficient indexing and query optimization for fast retrieval.

## 5. Risks

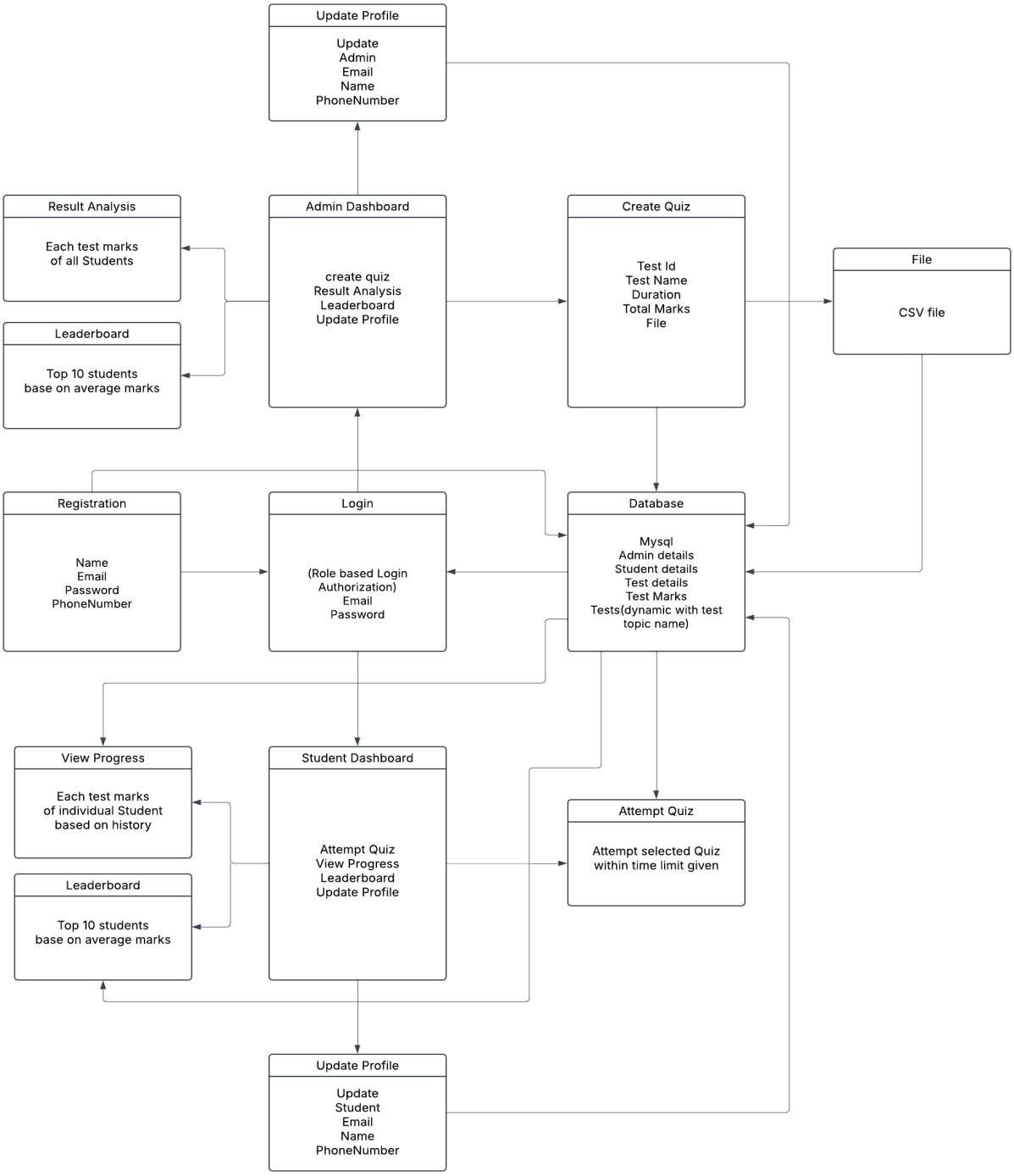
| Risk | Mitigation Strategy |
|---|---|
| High server load due to multiple users | Implement caching and load balancing |
| Unauthorized access to quiz data | Use JWT authentication and RBAC |
| Data loss due to system failure | Schedule regular database backups |
| Quiz tampering or cheating | Implement session tracking and anti-cheat mechanisms |

## 6. System Overview

The Web-Based Quiz Platform operates through a three-tier architecture:

1. **Frontend (React.js):** Manages user interactions and quiz display.
2. **Backend (Spring Boot & REST API):** Handles authentication, quiz logic, and score calculation.
3. **Database (MySQL):** Stores user data, quizzes, questions, and scores.

# 7. Architecture Design



## 8. Database Design

| Table Name | Description |
|---|---|
| **admin_details** | Stores administrator details, including login credentials and role-based access. |
| **student_details** | Contains student information, including registration details and login credentials |
| **test_details** | Stores quiz metadata such as subject, duration, and test availability. |
| **test_marks** | Tracks student quiz attempts, scores, and performance. |
| **questions** | Stores quiz questions dynamically, supporting multiple subjects and difficulty levels. |

## 9. Component Design

- **User Authentication Module**: Manages login and user roles.
- **Quiz Management Module**: Handles quiz creation, updates, and deletions.
- **Score Processing Module**: Computes and stores quiz results.
- **Leaderboard Module**: Ranks students based on scores.
- **CSV Upload Module**: Allows bulk quiz uploads.

## 10. Interface Design

- **Admin Dashboard**: Manage quizzes, students, and scores.
- **Student Dashboard**: Attempt quizzes and view performance reports.
- **Leaderboard Page**: Displays top performers per subject.

## 11. Specific Design Considerations

- **Security:** JWT authentication, encrypted data storage, HTTPS communication.
- **Scalability:** Load balancing and caching strategies.
- **Accessibility:** Mobile and desktop compatibility.

## 12. Design Testing

- **Unit Testing:** Ensuring individual modules function correctly.
- **Integration Testing:** Validating API interactions between frontend and backend.
- **Performance Testing:** Checking system response under heavy load.
- **Security Testing:** Assessing vulnerabilities in authentication and data storage.

## 13. Cross-Reference with SRS

| SRS Requirement | Design Implementation |
|---|---|
| User Registration & Authentication | Role-based login |
| Quiz Management | REST API endpoints for CRUD operations |
| Score Calculation | Real-time processing on quiz submission |
| Leaderboard | Database query optimization for ranking |

---