Project Report On

Quiz Management System

Submitted in partial fulfillment for the award of
Post Graduate Diploma in Advanced Computing
from **C-DAC ACTS (Pune)**

Guided by

Mr. Jitesh Bafna

Presented By

Mr. Abhishek Kumbhar	**PRN:230940120009**
Mr. Avishkar Hongekar	**PRN: 230940120041**
Mr. Piyush Bartakke	**PRN: 230940120044**
Mr. Kunal Ahirrao	**PRN: 230940120095**
Mr. Pranav Kalantri	**PRN: 230940120133**
Mr. Rahul Kumar Singh	**PRN: 230940120146**

Centre of Development of Advanced Computing (C-DAC), Pune

CERTIFICATE

TO WHOMSOEVER IT MAY CONCERN

This is to certify that

Mr. Abhishek Kumbhar	**PRN:230940120009**
Mr. Avishkar Hongekar	**PRN: 230940120041**
Mr. Piyush Bartakke	**PRN: 230940120044**
Mr. Kunal Ahirrao	**PRN: 230940120095**
Mr. Pranav Kalantri	**PRN: 230940120133**
Mr. Rahul Kumar Singh	**PRN: 230940120146**

have successfully completed their project on

Quiz Management System

Under the Guidance of Mr. Jitesh Bafna

Project Guide **Project Supervisor**

HOD ACTS

PG-DAC

ACKNOWLEDGEMENT

This project **"Quiz Management System"** was a great learning experience for us and we are submitting this work to Advanced Computing Training School (CDAC ACTS).

We all are very glad to mention the name of **Mr. Jitesh Bafna** for his valuable guidance to work on this project. His guidance and support helped us to overcome various obstacles and intricacies during the course of project work.

We are highly grateful to Manager (ACTS training Centre), C- DAC, for his guidance and support whenever necessary while doing this course Post Graduate Diploma in Advanced Computing (PG- DAC) through C-DAC ACTS, Pune.

Our most heartfelt thank goes to **Ms. Priyanka Ranade** (Course Coordinator, PG-DAC) who gave all the required support and kind coordination to provide all the necessities like required hardware, internet facility and extra Lab hours to complete the project and throughout the course up to the last day here in C-DAC ACTS, Pune.

Mr. Abhishek Kumbhar	PRN:230940120009
Mr. Avishkar Hongekar	PRN: 230940120041
Mr. Piyush Bartakke	PRN: 230940120044
Mr. Kunal Ahirrao	PRN: 230940120095
Mr. Pranav Kalantri	PRN: 230940120133
Mr. Rahul Kumar Singh	PRN: 230940120146

INTRODUDCTION

The Quiz Management System project is web-based application designed to streamline and enhance the process of conducting quizzes and assessments. This project employs a robust technology stack, utilizing Spring Boot for the backend services and ReactJS for the frontend user interface.

The primary objective of this system is to provide an efficient and user-friendly platform for both quiz creators and participants. The Spring Boot framework is leveraged to develop a scalable and secure backend, offering features such as user authentication, quiz creation, and result tracking. ReactJS, a powerful JavaScript library, is employed to build a dynamic and responsive frontend, ensuring an engaging experience for users.

The system incorporates a comprehensive set of functionalities including user registration and authentication, quiz creation and management, detailed results. Utilizing RESTful APIs, the frontend and backend seamlessly communicate, allowing for a smooth and interactive user experience.

Throughout the development process, industry-standard software engineering practices were employed, including requirement analysis, system design, implementation, testing, and deployment. The use of Spring Boot and ReactJS contributes to the scalability, maintainability, and performance of the system.

This project report provides a comprehensive overview of the Quiz Management System, detailing the system architecture, design choices, implementation strategies, and testing methodologies. The results and feedback obtained from user testing and evaluation are presented, along with potential areas for future enhancements.

In conclusion, the Quiz Management System project demonstrates the effective integration of Spring Boot and ReactJS technologies to create a feature-rich and scalable platform for quiz management, catering to the evolving needs of educational and professional environments.

FUNCTIONAL REQUIREMENT

1. User Registration: The system should allow users to register by providing necessary information such as username, email, and password.
2. Quiz Creation and Management:
   a. Multiple Quiz Creation: Allow quiz creators to create multiple quizzes, each with a unique set of questions and settings.
   b. Quiz Details: Enable quiz creators to specify details such as title, description, duration, and passing criteria for each quiz.
3. Participant Registration and Management:
   a. Participant Registration: Participants should be able to register by providing essential information.
   b. Profile Management: Participants should have the ability to manage their profiles, including updating personal information and changing password.
4. Quiz Participation: Participants should be able to browse and select quizzes they want to participate in.

NON-FUNCTIONAL REQUIREMENT

1. The system should support concurrent quiz creation and participation by multiple users without significant performance degradation.
2. Response time for user interactions, such as creating quizzes, submitting answers, and viewing results, should be minimal to provide a seamless user experience.
3. Scalability: The system should be designed to scale horizontally to handle an increasing number of users, quizzes, and concurrent quiz sessions.
4. Reliability: The system should have high availability to ensure users can access and use the platform reliably.
5. Security: User authentication and authorization mechanisms should be robust to ensure secure access to the system.
6. Usability: The user interface should be intuitive and user-friendly, promoting easy navigation and efficient quiz creation.

SYSTEM REQUIREMENT SPECIFICATION

HARDWARE & SOFTWARE REQUIREMENT

USER

- Hardware Requirements:
    - Personal Computer or Laptop with minimum system requirements (e.g., processor, RAM, and storage) to support web browsing and online transactions.
    - Reliable internet connection to ensure smooth website access and uninterrupted online shopping experience.
- Software Requirements:
    - Web Browser (e.g., Google Chrome, Mozilla Firefox, Safari) with the latest version installed to access and interact with website.
    - Operating System (e.g., Windows, macOS, Linux) compatible with the chosen web browser for seamless website navigation and functionality.

DEVELOPER

- Integrated Development Environment (IDE) such as Spring Tool Suite
- Backend (Spring Boot):
    - Java Development Kit (JDK) 11
    - Build Tool: Apache Maven
- Frontend (ReactJS):
    - Node.js and npm (Node Package Manager).
- Code Editor: Visual Studio Code, Sublime Text, or Atom.
- Database Management System (DBMS): MySQL
- Version Control: Git for version control.
- API Documentation: Swagger
- Deployment: Choose a hosting service (VERCEL) to deploy website and configure the necessary server configurations.

METHODOLOGY

1. Spring Boot:
    a. Backend Development: Spring Boot is utilized to create a robust and scalable backend for the Quiz Management System. It facilitates the development of RESTful APIs, handles business logic, and connects with the database.
    b. Spring MVC: Manages the web layer, handling HTTP requests and responses.
    c. Spring Data JPA: Simplifies database operations by providing an abstraction layer over the data access layer.
    d. RESTful API: Creates APIs to handle communication between the frontend and backend.

2. ReactJS:
    a. Frontend Development: ReactJS is employed to build a dynamic and interactive user interface for quiz creation, participation, and result display.
    b. Component-Based Architecture: Facilitates modular development and maintenance.
    c. React Router: Manages navigation within the application.
    d. State Management: Utilizes local component state and Redux for managing global state.

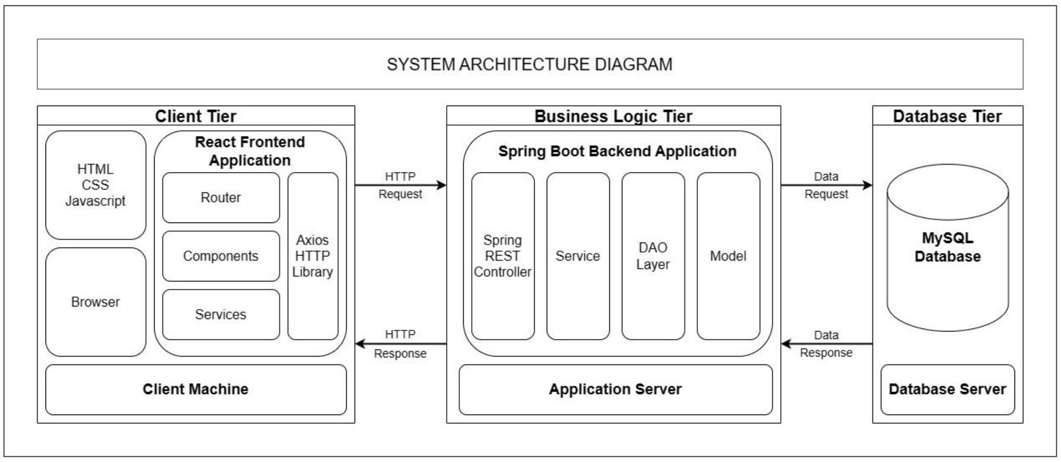
3. MySQL:
    a. Database Management: MySQL is chosen as the relational database system to store and manage quiz data, user profiles, and results.
    b. Database Schema: Defines tables for quizzes, questions, users, and results.

4. Redux:
    a. State Management: Redux is implemented for managing the global state of the application, especially for complex state interactions.
    b. Store: Holds the global state of the application.
    c. Actions: Defines events that trigger state changes.
    d. Reducers: Handles the state changes based on dispatched actions.

5. Bootstrap:
    a. Frontend Styling: Bootstrap is employed for frontend styling, providing a responsive and consistent design across different devices and browsers.
    b. Responsive Design: Ensures the application is accessible on various devices.
    c. Bootstrap Components: Utilizes pre-built UI components for buttons, forms, navigation bars, etc., saving development time.
    d. Grid System: Supports responsive layouts, aiding in the creation of a visually appealing and user-friendly interface.

6. Git:
    a. Version Control: Git is employed for source code versioning, enabling collaboration, tracking changes, and maintaining a history of the project.
    b. Branching and Merging: Supports parallel development and integration of features.
    c. Pull Requests: Facilitates code review before merging changes.
    d. Commit Messages: Provides a clear history of changes and their purposes.

SYSTEM ARCHITECTURE



System Architecture

Frontend Design (ReactJS):

- Component-Based Architecture: Developed reusable React components for user authentication, quiz creation, participation, and result display.
- React Router: Implement different routes for various sections like creating quizzes, participating in quizzes, and viewing results.
- State Management: Used Redux for global state management, especially for user authentication, quiz state, and result data.
- Bootstrap Components: Leverage Bootstrap components for UI elements such as buttons, forms, navigation bars, and modals, ensuring a consistent and visually appealing user interface.

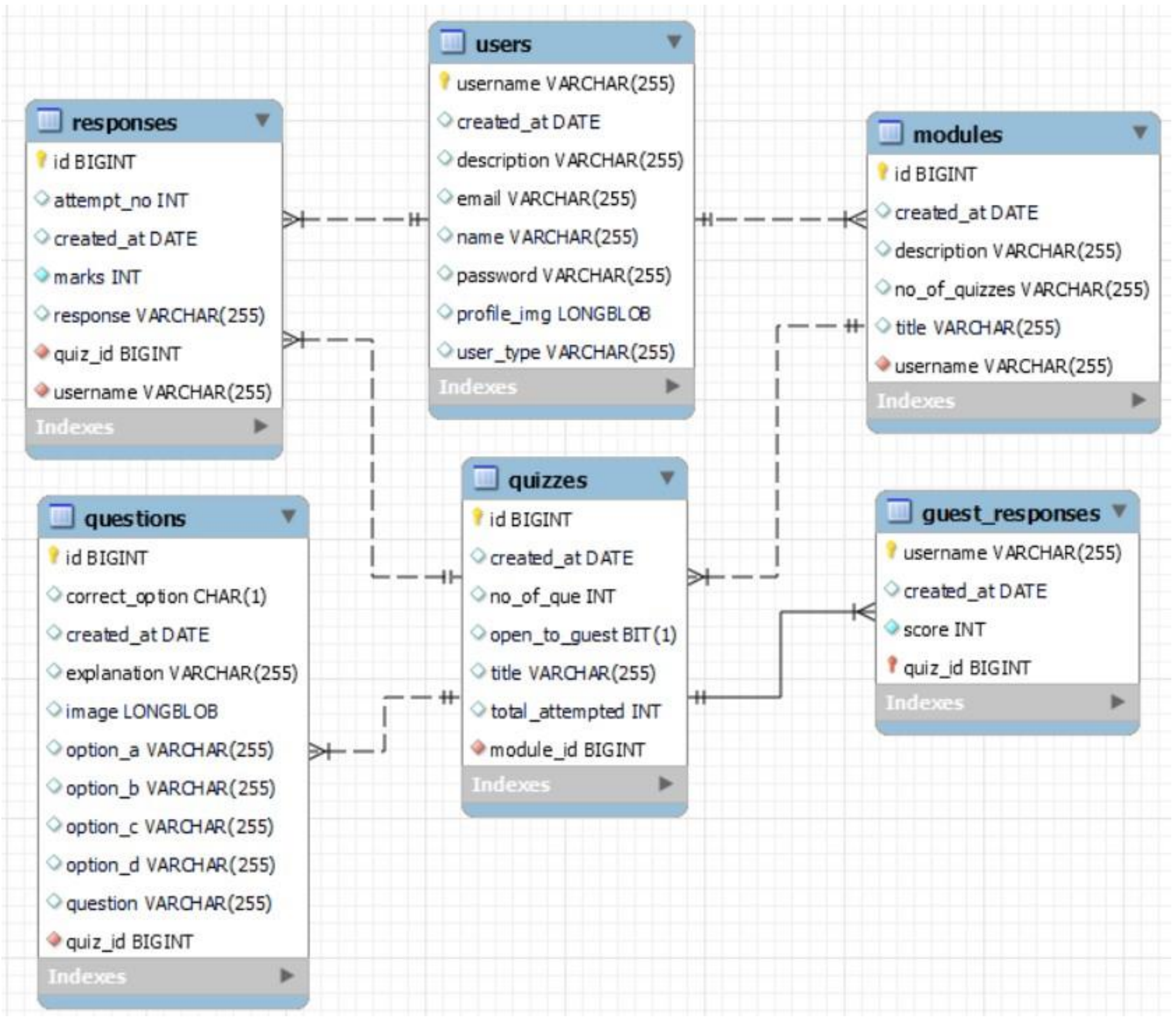
Backend Design (Spring Boot):

- RESTful API Design: Defined APIs for user authentication, quiz creation, quiz participation, and result retrieval. Follow RESTful principles for a clean and standardized API.
- Data Access Layer: Used Spring Data JPA to interact with the MySQL database. Design entities representing quizzes, questions, users, and results.

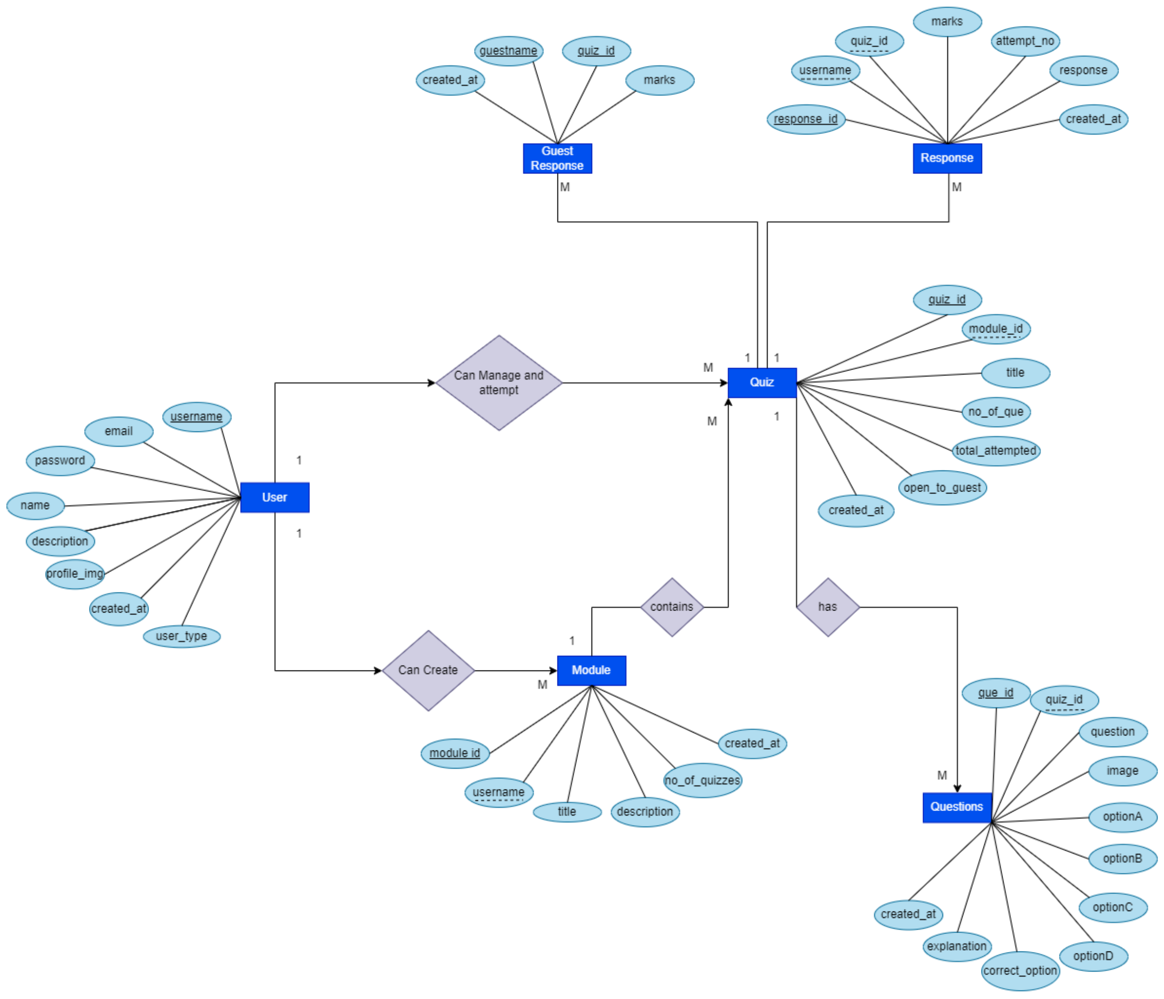
Database Design (MySQL):

- Database Schema: Design a relational database schema with tables for users, quizzes, questions, options, and results.
- Entity Relationships: Establish relationships between entities, such as a one-to-many relationship between quizzes and questions.

DATABASE DESIGN



1.1 IMPLEMENTATION DETAILS



- Setting Up the Development Environment:
  - Backend (Spring Boot):
    - Installed Java Development Kit (JDK) and an Integrated Development Environment (IDE) such as STS or Eclipse.
    - Created a new Spring Boot project using Spring Initializer.
    - Added necessary dependencies such as Spring Web, Spring Data JPA.
    - Set up a MySQL database and configured the application.properties file with database connection details.

  - Frontend (ReactJS):
    - Installed Node.js and npm for managing frontend dependencies.

        - Created a new React application using create-react-app.
        - Installed additional libraries such as Redux for state management and Bootstrap for styling.
        - Set up a folder structure for organizing React components, actions, reducers, and other files.

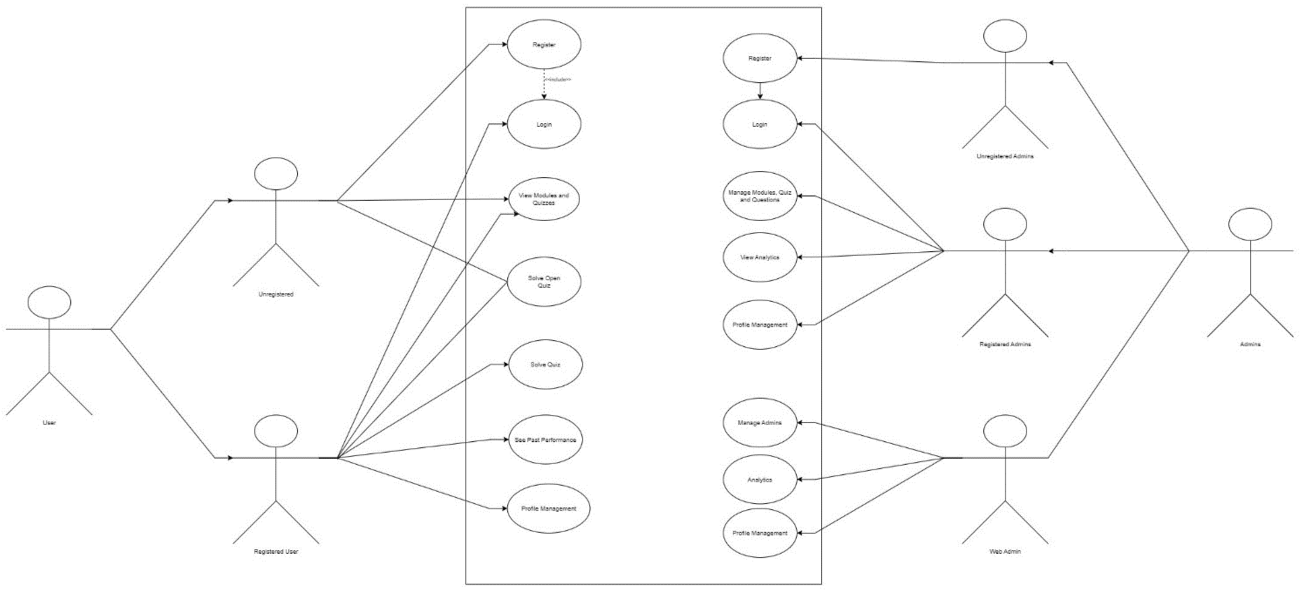
- Backend Development:
    - User Authentication:
        - Implemented user registration and login endpoints.
    - Quiz Creation:
        - Created RESTful API endpoints for quiz creation, question addition, and option inclusion.
        - Implemented data validation to ensure the integrity of quiz-related information.
    - Quiz Participation:
        - Developed API endpoints for fetching quiz details and submitting user responses.
        - Evaluated user responses, calculated scores, and stored results in the database.
    - Result Retrieval:
        - Implemented API endpoints to retrieve user-specific or quiz-specific result data
        - Ensure proper authorization to access result information.

- Frontend Development:
    - User Authentication:
        - Designed login and registration forms.
        - Implemented React components for user authentication, integrating with backend authentication endpoints.
    - Quiz Creation:
        - Created React components for creating quizzes, adding questions, and managing options.
        - Implemented Redux actions and reducers to manage quiz-related state.
    - Quiz Participation:
        - Designed React components for viewing available quizzes and participating in them.

        - Integrated Redux to manage the state of quiz participation, including user responses.
    - Result Display:
        - Developed React components to display quiz results.

- API Communication:
    - Utilized Axios or another HTTP client in the ReactJS frontend to make requests to the Spring Boot backend.
    - Implemented Redux actions to dispatch API requests and manage asynchronous behavior.
    - Ensured proper handling of API responses and errors in the frontend components.

- Integration:
    - Integrated the ReactJS frontend with the Spring Boot backend by configuring CORS (Cross-Origin Resource Sharing) settings.
    - Tested the end-to-end communication between frontend and backend components.

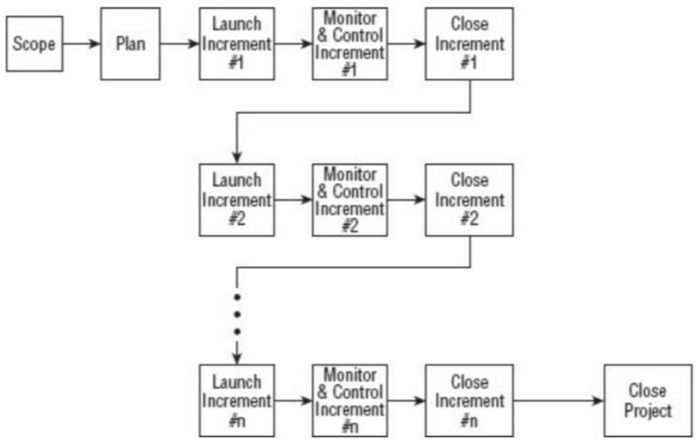


RESULT

- Login:
  - Authenticate users securely with username and password for personalized access to the Quiz Management System.
- Signup:
  - Register new quiz creators, capturing essential information and ensuring a smooth onboarding process.
- Homepage:
  - Present a personalized dashboard highlighting quizzes created, ongoing modules, and recent results for multiple quiz creators.
- Module List:
  - Display a comprehensive list of quiz modules, facilitating easy navigation for quiz creators in managing and editing modules.
- Quiz Page:
  - Enable quiz creators to design and customize quizzes, specifying details like time limits and question types effortlessly.
- Quiz Questions:
  - Allow quiz creators to add, edit, and organize questions with options, ensuring flexibility and creativity in quiz content creation.
- Result:
  - Present individual results, empowering quiz creators to assess participant performance and make data-driven improvements.
- Profile Management:
  - Empower quiz creators to personalize their profiles, manage account settings, and track their contribution history within the Quiz Management System.
- Manage Content:
  - Modules: Create, edit, and organize learning modules seamlessly, allowing quiz creators to structure content logically.
- Quizzes:
  - Efficiently manage quizzes, enabling quiz creators to set parameters, schedule, and monitor participation details.
- Questions:
  - Streamline question management, offering quiz creators the ability to add, edit, and categorize questions for diverse quiz content.

1.1 **PROJECT MANAGEMENT**

This model can be used when the requirements of the complete system are clearly defined and understood, like the case of this project where;



Incremental Project Management Life Cycle

The Incremental model is much better equipped to handle change. Each incremental functionality is verified by the customer and hence the relative risk in managing large and complex projects is substantially reduced.

Incremental SDLC provide plethora of advantages inducing;

- Generates working software quickly and early during the software life cycle.
- This model is more flexible and less costly to change scope and requirements.
- It is easier to test and debug during a smaller iteration.
- In this model customer can respond to each built.

Code storage GITHUB

CONCLUSION AND FUTURE WORK

Conclusion:

The Quiz Management System, developed using a technology stack comprising Spring Boot, ReactJS, MySQL, Redux, HTML, CSS, and Bootstrap, presents a robust and feature-rich platform for efficient quiz creation, participation, and result management. The system embraces the advantages of each technology, ensuring a scalable, secure, and user-friendly experience.

The use of Spring Boot on the backend facilitates the development of RESTful APIs, ensuring seamless communication with the ReactJS-based frontend. MySQL serves as a reliable database management system, storing quiz data, user profiles, and results in a well-designed relational schema. ReactJS, coupled with Redux, provides a dynamic and responsive frontend, offering an intuitive user interface for quiz-related activities. The incorporation of Bootstrap ensures a consistent and visually appealing design across different devices.

Authentication mechanisms guarantee secure access to the system, while the modular architecture allows for easy maintenance and future enhancements. The system's design promotes collaboration between frontend and backend teams, facilitating a smooth workflow through version control with Git.

Future Work:

- Enhanced User Analytics: Implement advanced analytics features to provide detailed insights into user performance, popular quiz topics, and participation trends.
- Gamification Elements: Incorporate gamification elements, such as badges, leaderboards, and achievements, to motivate users and enhance user engagement.
- Real-Time Collaboration: Implement real-time collaboration features, allowing multiple users to collaboratively create quizzes or participate in quizzes simultaneously.
- Integration with Learning Management Systems (LMS): Integrate the Quiz Management System with popular Learning Management Systems for seamless incorporation into educational environments.
- Mobile Application Development: Develop a dedicated mobile application to expand accessibility and provide users with a more convenient way to engage with quizzes on the go.