Quizzie – The Quiz Management System

REVIEW 2

Internet and Web Programming (CSE3002)

Slot: F1

By

REG. NO.	NAME
18BCE0283	ANMOL PANT
19BCB0054	TANYA GUPTA
19BCE0148	JUGAL BHATT
19BCE0312	ADITYA BERI
19BCE0541	PRIYA GOVINDASAMY

Under the guidance of

Prof. Vijayarani A.

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Abstract

This project is a complete online quiz management platform with a plethora of features for teachers and students. It provides all the basic functionalities of a quiz portal like timer, scoring, ranking etc. alongside analysis of results from the students on the teacher dashboard, summary of tests for the student, Login and signup using a Google account for both teacher and student.
It provides a smooth user interface for both the teachers and students to create and attempt different kinds of quizzes seamlessly.

Motivation

The motivation of this project came from the idea to help teachers and institute administrators create quizzes and tests in an efficient manner. With the popularity of the Internet, it is inevitable to have online quizzes as classroom assessments. In learning, online quizzes may serve two objectives, that is, for self-study or as a formal assessment. The online quiz has its advantages, such as saving the cost of paper printing and reducing the time spent for having assessments in class. However, there are weaknesses. Its primary drawback is the issue of academic dishonesty, especially when students are answering the online quiz.

Aim of the proposed work

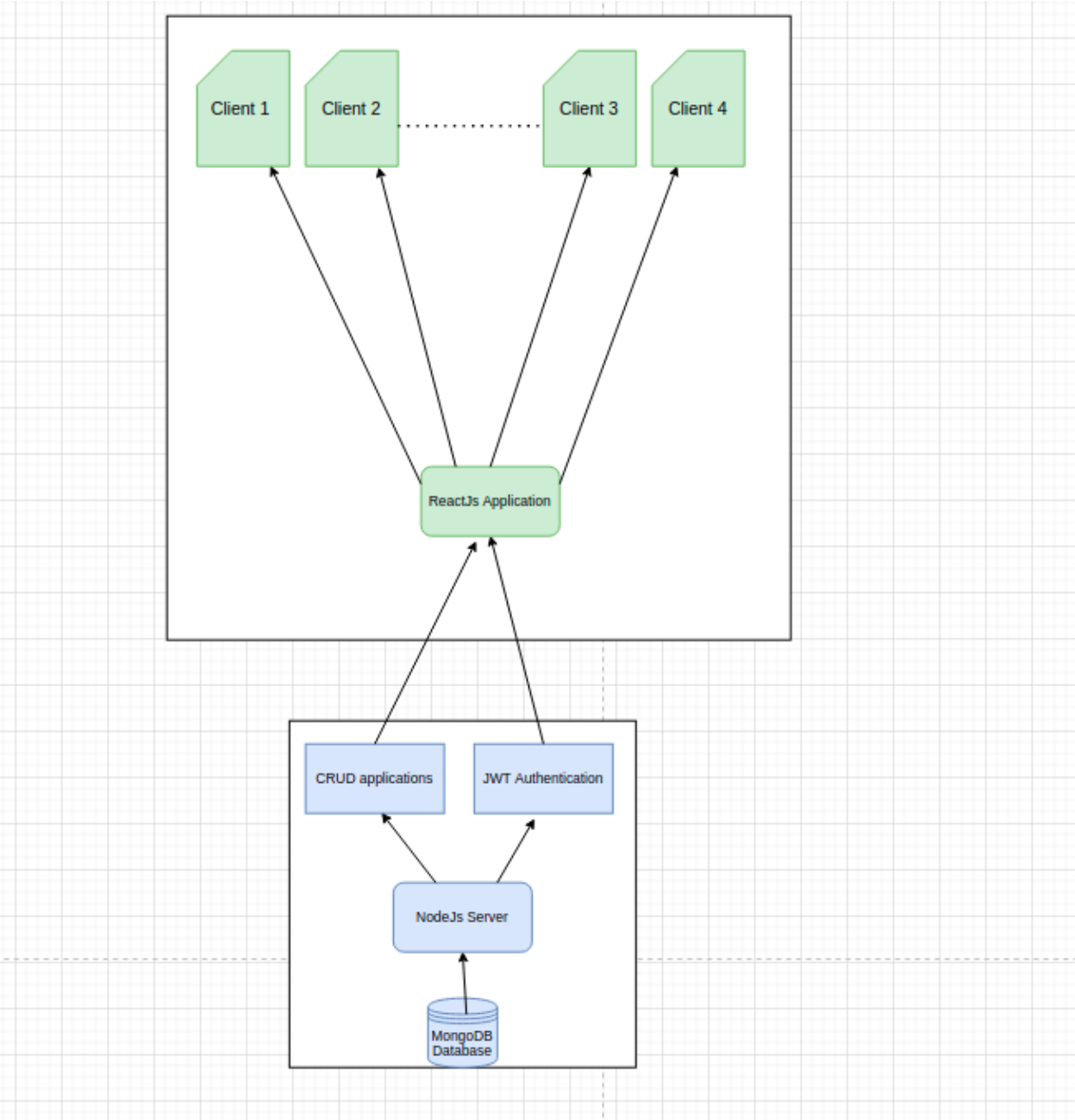
The purpose of this project is to create a web-based solution that improves the overall efficiency and reduces the hassle of conducting online quizzes. Also to provide necessary features for a teacher such as timer, scoring, ranking etc. alongside analysis of results from the students on the teacher dashboard, summary of tests. And to provide students with a seamless way of attempting quizzes, so that they can easily keep track of their progress.

Objective

1. The objective is to help in the administration of online quizzes. With the popularity on the Internet, it is inevitable to have online quizzes as classroom assessments.
2. The teacher's would have access to a dashboard with features like scoring,

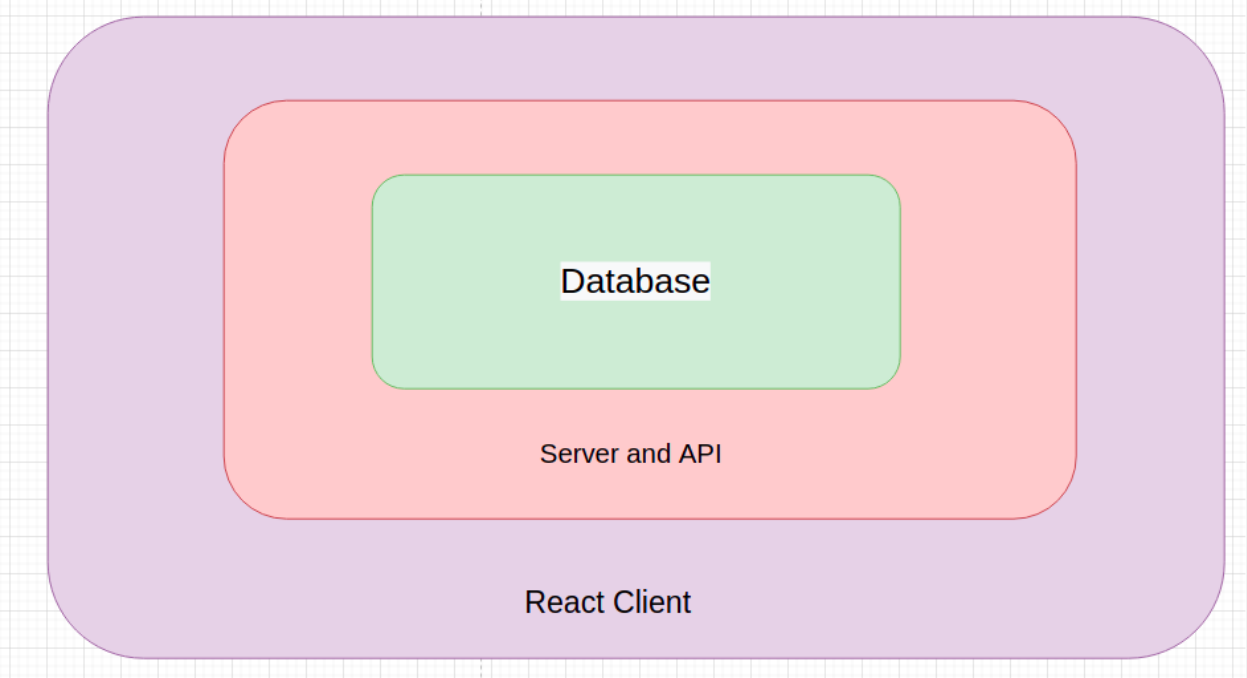
timing, and detailed statistics for each quiz.
3. The student's would have access to a dashboard with all quizzes attempted, upcoming quizzes, and summary of all attempted tests.

1. Architecture diagram



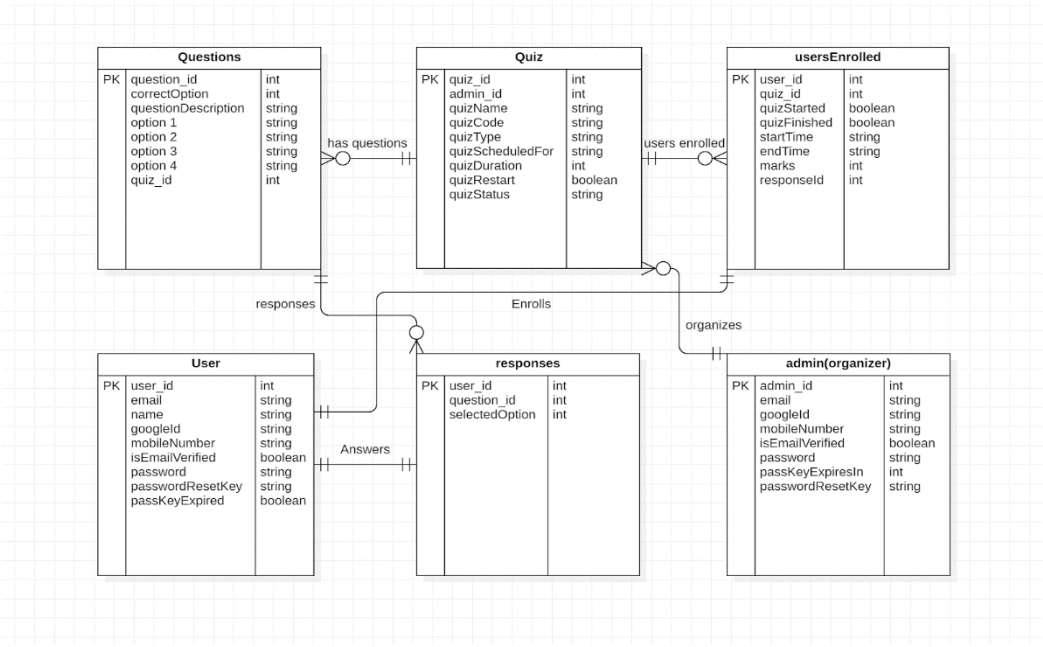
The architecture diagram shows the interaction of the different components used to create the system. Users interact with the frontend which is a ReactJS application. The

backend involves a CRUD (Create, Read, Update, Delete) application and JWT (JSON Web Token) Authentication that interact with the NodeJS server. All the data used in the application is stored in the MongoDB database.



The application uses a three-tiered architecture. The React Client acts as the first tier or the presentation layer. The middle tier which contains all the server logic is implemented using a NodeJS server and other APIs. The Database containing all the data relevant to the application forms the third and final tier. It is implemented in MongoDB.

2. DB DESIGN



The entity-relationship diagram above provides a clear description of the implementation of the database. Admins can organize one or many quizzes. Each quiz can have multiple questions and multiple enrolled users. The questions will have different responses depending on the user. The records for all the tables in the database are uniquely identified by their corresponding ID.

Schema:

Users	
PK	user_id String not null
	name String not null
	GoogleId int
	email String not null
	Password String (encrypted)
	Mobile Number Int
	IsEmailVerified Boolean
	passwordResetKey String
	PassKeyExpiresIn Date

Admins(organizers)	
PK	admin_id String not null
	name String not null
	GoogleId int
	email String not null
	Password String (encrypted)
	Mobile Number Int
	IsEmailVerified Boolean
	passwordResetKey String
	PassKeyExpiresIn Date

Quizzes	
PK	quiz_id String Not Null
FK1	admin_id not null
	quizName String not null
	quizCode String not null
	quizType String not null (default : Public)
	quizScheduledFor Date (timestamp)
	quizDuration Int (minutes)
	quizStatus Int (started/yet to be started/over)
	quizRestart Boolean (if it's to be restarted for a user

Questions	
PK	question_id not null
FK	quiz_id String not null
	questionDescription String
	option1 String
	option2 String
	option3 String
	option4 String
	correctOption String

responses	
FK1	question_id string not null
	selected_option_text string not null
FK1	user_id string not null

usersEnrolled	
FK	quiz_id String not null
FK	user_id String not null
	quizStarted Boolean
	quizFinished Boolean
	marks int
	startTime date (timestamp)
	endTime date (timeStamp)

The database schema is a skeleton structure that represents the logical view of the entire database. It shows the associations between the tables through foreign keys. It also shows all the attributes possessed by each table.

Database model codes:

1. Organizer:

```
const mongoose = require("mongoose");

const adminSchema = mongoose.Schema({
 _id: mongoose.Schema.Types.ObjectId,
 userType: { type: String, default: "Admin" },
 name: { type: String, required: true },
 googleId: { type:Number },
 email: {
   type: String,
   required: true,
   match: /[a-z0-9!#$%&'*+/=?^_`{|}~-]+(?:\.[a-z0-9!#$%&'*+/=?^_`{|}~-]+)*@(?:[a-z0-
9](?:[a-z0-9-]*[a-z0-9])?\.)+[a-z0-9](?:[a-z0-9-]*[a-z0-9])?/,
 },
 password: { type: String},


 mobileNumber: {
   type: Number,
   match: /^([7-9][0-9]{9})$/g,
 },
 quizzes: [
   {
     quizId: { type: mongoose.Schema.Types.ObjectId, ref: "Quiz" },
   },
 ],
 token: {
```

```
    type: String,
  },

  passResetKey: { type: String },
  passKeyExpires: { type: Number },
  verificationKey: { type: String },
  verificationKeyExpires: { type: Number },
  isEmailVerified: { type: Boolean ,default:false},
});

module.exports = mongoose.model("Admin", adminSchema);
```

2. User :

```
const mongoose = require("mongoose");

const userSchema = mongoose.Schema({
  _id: mongoose.Schema.Types.ObjectId,
  googleId: {
    type: String,
  },
  name: { type: String },

  userType: { type: String, default: "User" },
  email: {
    type: String,
    lowercase: true,
    match: /[a-z0-9!#$%&'*+/=?^_`{|}~-]+(?:\.[a-z0-9!#$%&'*+/=?^_`{|}~-]+)*@(?:[a-z0-
9](?:[a-z0-9-]*[a-z0-9])?\.)+[a-z0-9](?:[a-z0-9-]*[a-z0-9])?/,
  },
  mobileNumber: {
    type: Number,
    match: /^([7-9][0-9]{9})$/g,
  },
  password: { type: String },
  quizzesGiven: [
    {
      quizId: { type: mongoose.Schema.Types.ObjectId, ref: "Quiz" },
      marks: { type: Number },
      responses: [],
      timeEnded: { type: Number },
      timeStarted:{type:Number}
    },
  ],
  quizzesStarted: [
    {
      quizId: { type: mongoose.Schema.Types.ObjectId, ref: "Quiz" },
    },
  ],
  quizzesEnrolled: [
    {
      quizId: { type: mongoose.Schema.Types.ObjectId, ref: "Quiz" },
    },
  ],
  token: {
    type: String,
  },
  passResetKey: { type: String },
```

```
 passKeyExpires: { type: Number },
 verificationKey: { type: String },
 verificationKeyExpires: { type: Number },
 isEmailVerified: { type: Boolean ,default:false},
});

module.exports = mongoose.model("User", userSchema);
```

3. Quiz:

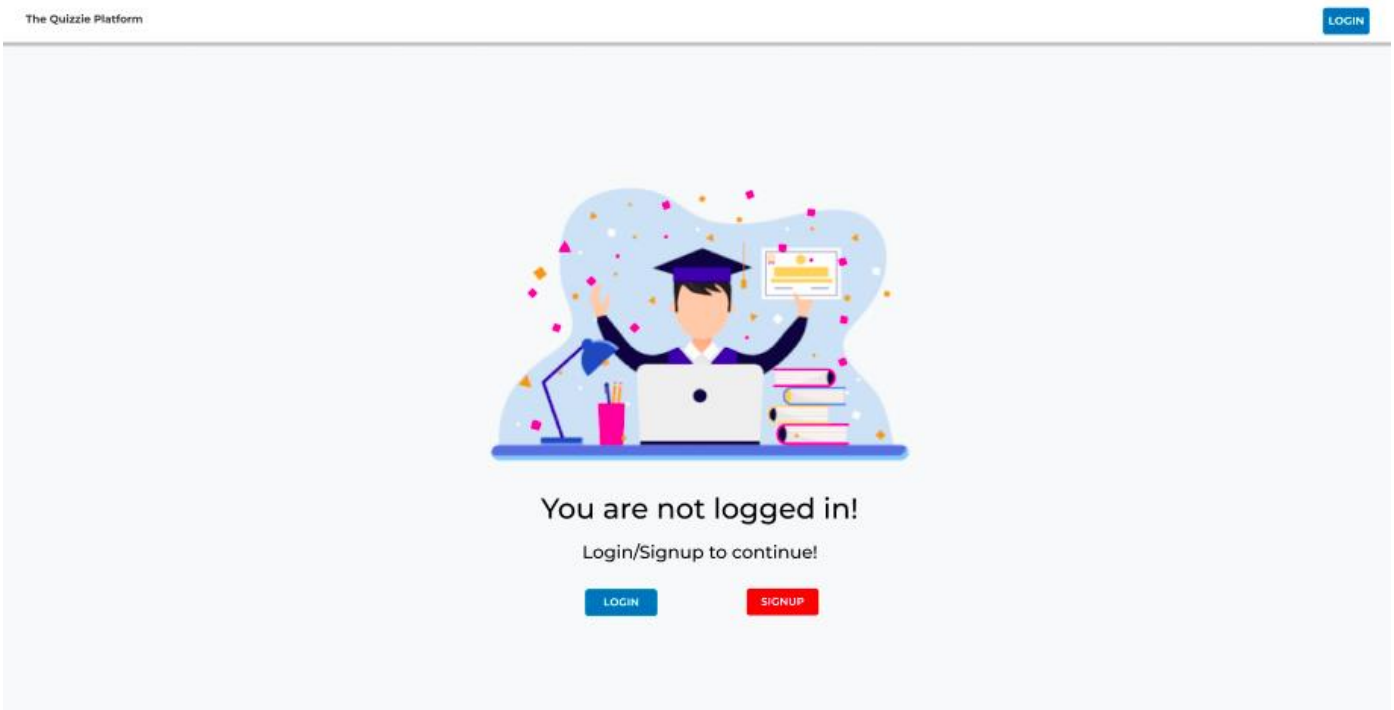
```
const mongoose = require("mongoose");

const QuizSchema = new mongoose.Schema({
 _id: mongoose.Schema.Types.ObjectId,
 quizName: { type: String, required: true },
 quizCode: { type: String },
 adminId: { type: mongoose.Schema.Types.ObjectID, ref: "Admin" },
 quizType: { type: String },
 usersParticipated: [
   {
     userId: { type: mongoose.Schema.Types.ObjectID, ref: "User" },
     marks:{type:Number},
     responses:[],
     timeEnded:{type:Number},
     timeStarted:{type:Number}
   },
 ],
 usersEnrolled: [
 scheduledFor: { type: String },
 scheduledForString: { type: String },
 quizDuration: {
   type: String,
 },
   {
     userId: { type: mongoose.Schema.Types.ObjectID, ref: "User" },
   },
 ],
 quizStatus: {
   type: Number,
   default: 0,
 },
 quizRestart:{
   type: Number,
   default:0
 },
 reminderSent: {
   type: Boolean,
   default: false,
 }
});

module.exports = mongoose.model("Quiz", QuizSchema);
```

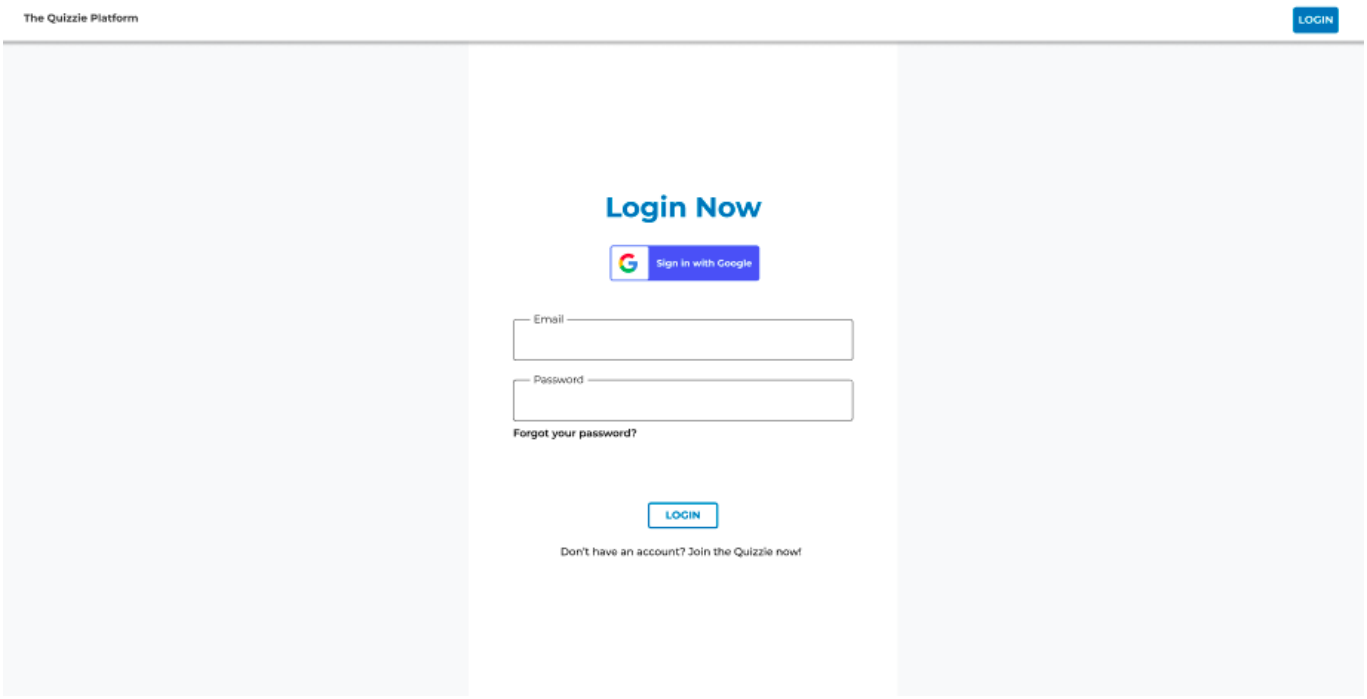
4. Questions:

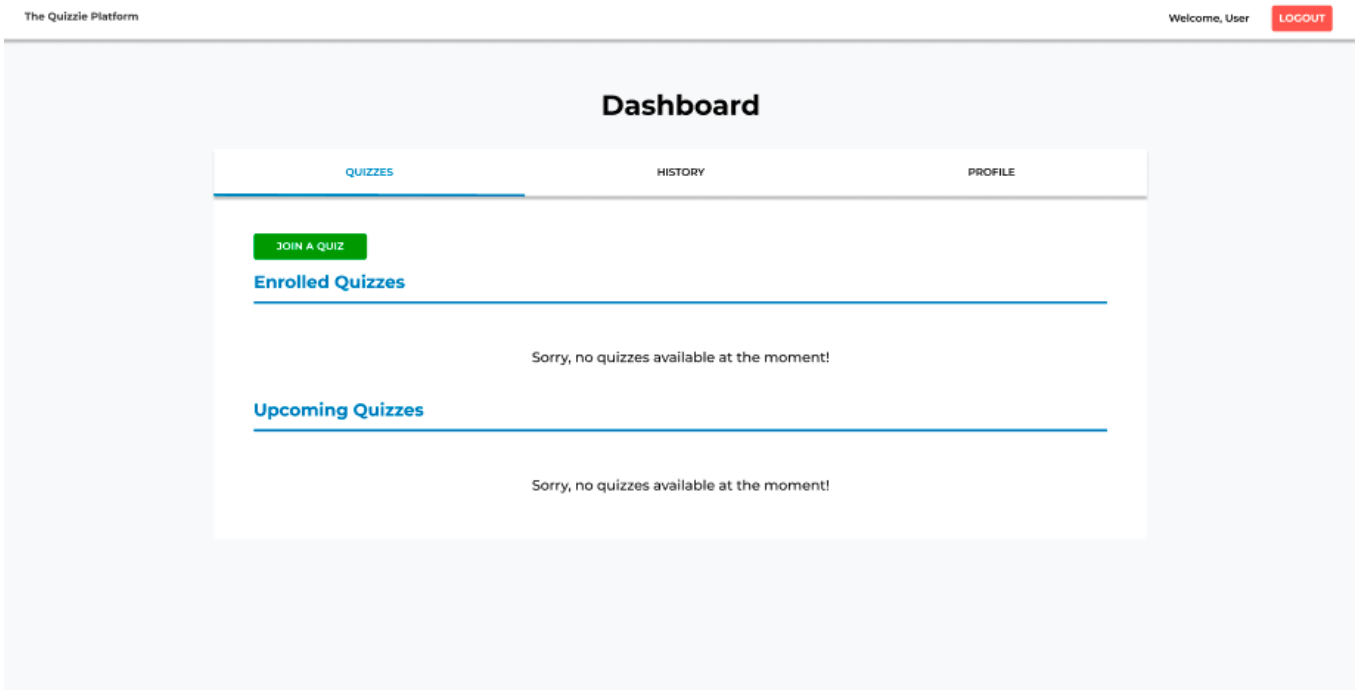
```
const mongoose = require("mongoose");

const questionSchema = new mongoose.Schema({
```

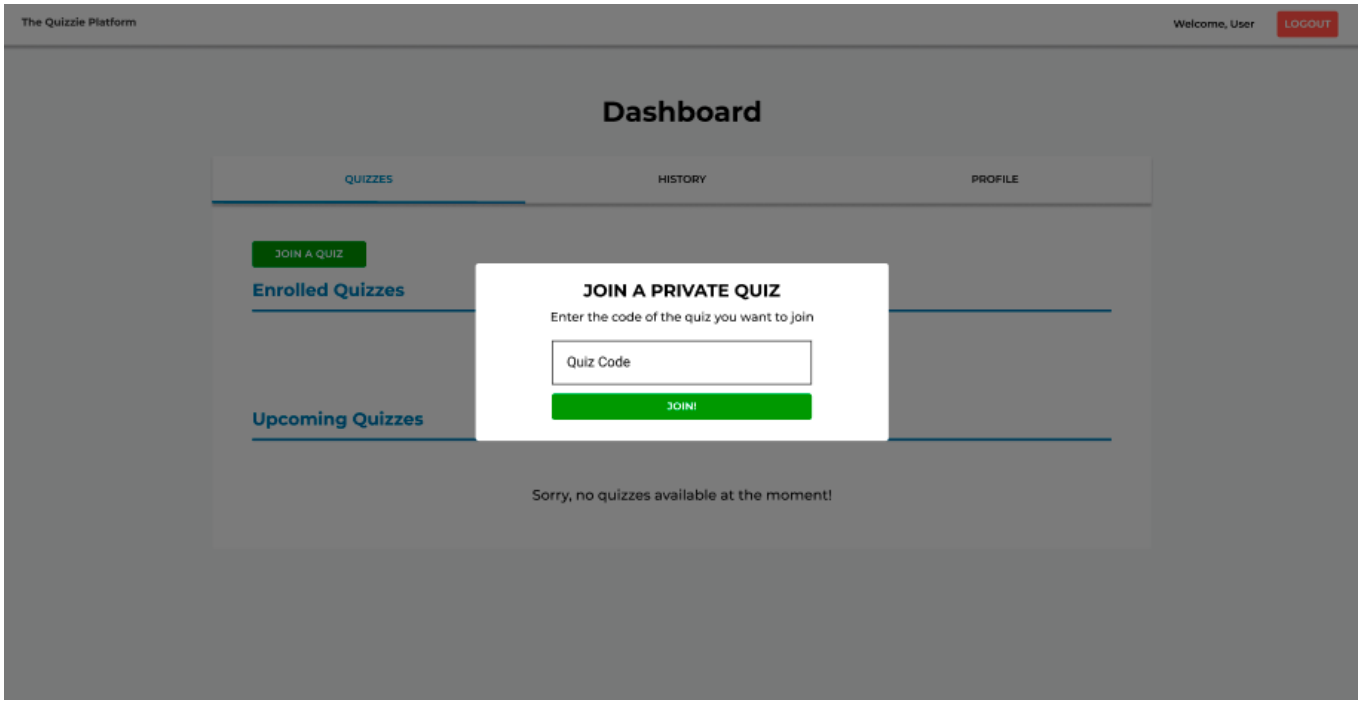
```
  _id: mongoose.Schema.Types.ObjectId,
  quizId: { type: mongoose.Schema.Types.ObjectID, ref: "Quiz" },
  description: {
    type: String,
    required: true,
  },
  options: [
    {
      text: {
        type: String,
        required: true,
      },
    },
  ],
  correctAnswer: {
    type: String,
    required: true,
  },
});

module.exports = mongoose.model("Question", questionSchema);
```

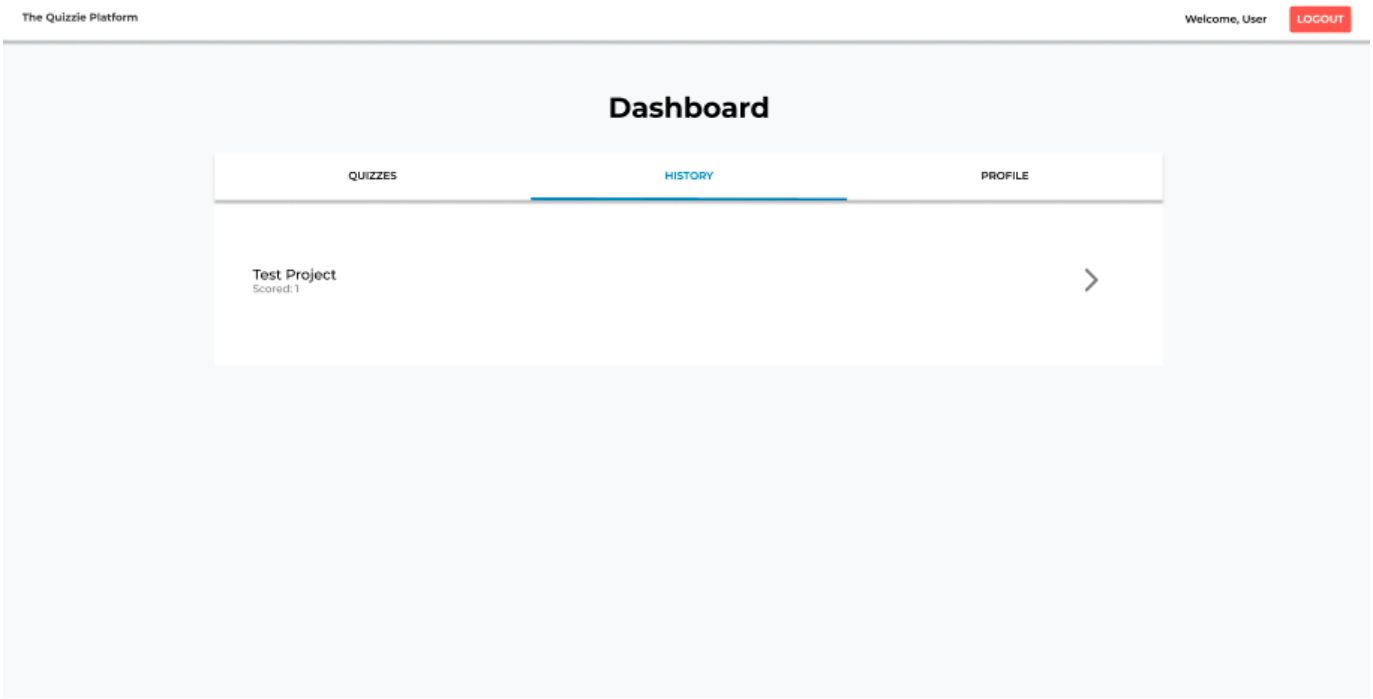
3. Interfaces

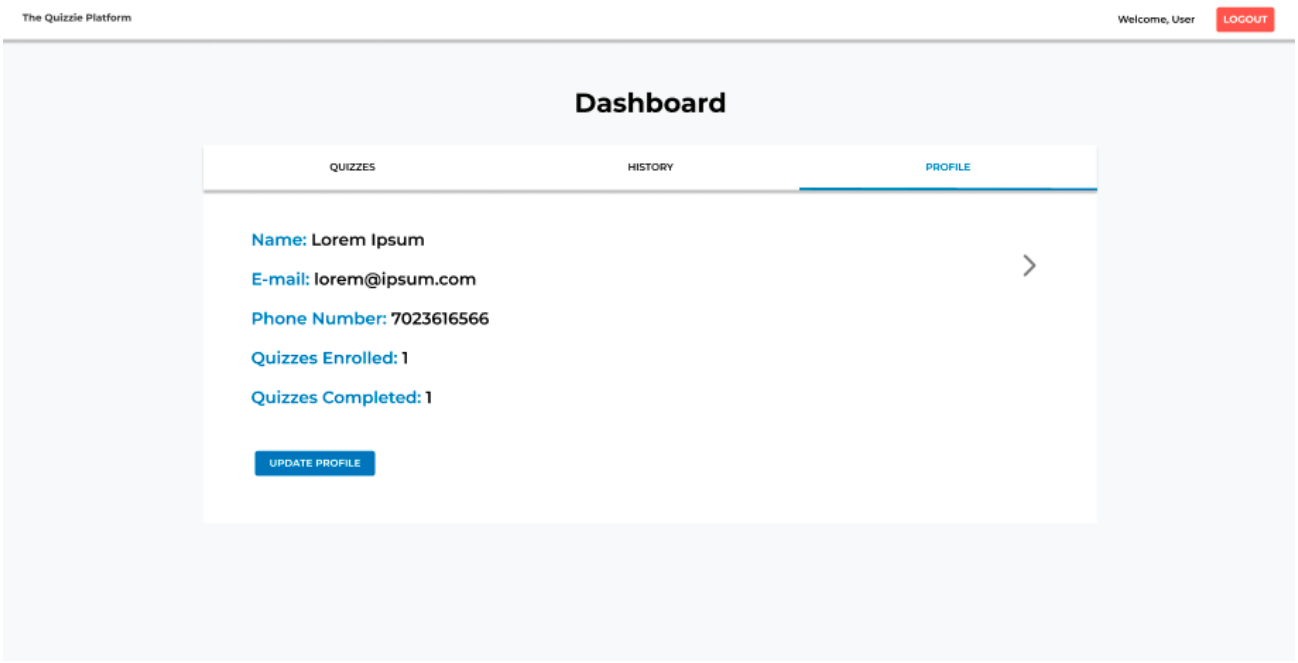


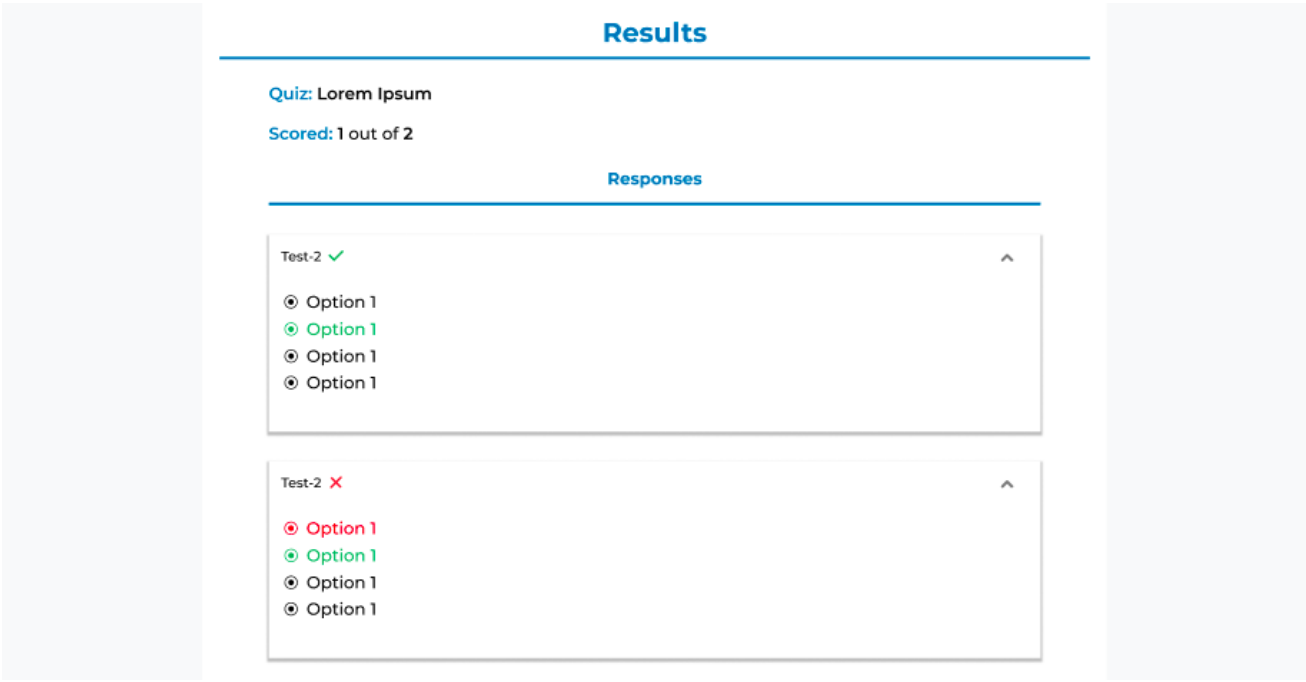

The Quizzie Platform
LOGIN
Login Now
Sign in with Google
Email
Password
Forgot your password?
LOGIN
Don't have an account? Join the Quizzie now!


The Quizzie Platform
Welcome, User
LOGOUT
Dashboard
QUIZZES
HISTORY
PROFILE
JOIN A QUIZ
Enrolled Quizzes
Sorry, no quizzes available at the moment!
Upcoming Quizzes
Sorry, no quizzes available at the moment!


The Quizzie Platform
Welcome, User
LOGOUT
Dashboard
QUIZZES
HISTORY
PROFILE
JOIN A QUIZ
Enrolled Quizzes
JOIN A PRIVATE QUIZ
Enter the code of the quiz you want to join
Quiz Code
JOIN!
Upcoming Quizzes
Sorry, no quizzes available at the moment!


The Quizzie Platform
Welcome, User
LOGOUT
Dashboard
QUIZZES
HISTORY
PROFILE
Test Project
Scored: 1


The Quizzie Platform
Welcome, User
LOGOUT
Dashboard
QUIZZES
HISTORY
PROFILE
Name: Lorem Ipsum
E-mail: lorem@ipsum.com
Phone Number: 7023616566
Quizzes Enrolled: 1
Quizzes Completed: 1
UPDATE PROFILE


Results
Quiz: Lorem Ipsum
Scored: 1 out of 2
Responses
Test-2
Option 1
Option 1
Option 1
Option 1
Test-2
Option 1
Option 1
Option 1
Option 1

4. Important links

- Code Base
  https://github.com/jugaldb/quizzie-iwp

- Website link
  https://iwp.jugaldb.com

* * *