MINOR PROJECT II

A QUIZ APPLICATION INCLUDING SPEECH RECOGNITION

BLINKQUIZ

Group No. : 04

Submitted by:

KHUSHI KALRA (9920103025)

VAISHALI RANJAN (9920103013)

VREETI AGGARWAL (9920103007)

Under the supervision of:

DR. HIMANSHU AGRAWAL

DEPARTMENT OF CSE/IT

JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY, NOIDA (U.P.)

APRIL 2023

ACKNOWLEDGEMENT

We would like to place on record my deep sense of gratitude to **DR. HIMANSHU AGRAWAL**, Assistant Professor, Jaypee Institute of Information Technology, India for his/her generous guidance, help, inspiration and constructive suggestions. New ideas and direction from him made it possible for us to sail through various areas of our project which further helped us to make it better.

We would also thank our institution and faculty members without whom this project would have been a distant reality.

We also wish to extend our thanks to seniors and other classmates for their insightful comments and constructive suggestions to improve the quality of this project work.

Signature(s) of Students

Name: VAISHALI RANJAN

Enrollment No.: 9920103013

Name: KHUSHI KALRA

Enrollment No: 9920103025

Name: VREETI AGGARWAL

Enrollment No.: 9920103007

DECLARATION

We hereby declare that this submission is our own work and that, to the best of our knowledge and beliefs, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma from a university or other institute of higher learning, except where due acknowledgment has been made in the text.

Place: NOIDA

Date:

Name: VREETI AGGARWAL

Enrollment No.: 9920103007

Name: VAISHALI RANJAN

Enrollment No.: 9920103013

Name: KHUSHI KALRA

Enrollment No: 9920103025

CERTIFICATE

This is to certify that the minor project report entitled, “A Quiz Application with Speech Recognition - BlinkQuiz” submitted by Vaishali Ranjan, Khushi Kalra and Vreeti Aggarwal of B.Tech of Jaypee Institute of Information Technology, Noida has been carried out by them under my supervision and guidance. This work has not been submitted partially or wholly to any other University or Institute for the award of any other degree or diploma.

Signature of Supervisor

DR. HIMANSHU AGRAWAL

Assistant Professor (Senior Grade)

Department of Computer Science and Engineering

Jaypee Institute of Information Technology, Sector-128, Noida- 201304

Date:

ABSTRACT

With humans moving towards higher standards of living and to a more digitized and interconnected world, computers prove to play an eminent role by providing the Most efficient and optimum ways in achieving the specified goals. Human resources and also the computer system provide the ideal paradigm of a trouble shooter. Such systems need to be user friendly, accurate, and multitasking as they are needed by every section of people. But when it comes to visually impaired people they (the software's/systems) pose an excellent deal of struggle and difficulty and therefore the complete utilization of the facilities is hampered while using the visual interface. This can be solved by using the hearing capability.

Keeping this in mind we propose BlinkQuiz **-** A web-based application using the MERN technology stack that aims to provide an accessible platform for visually impaired individuals to take quizzes independently. The system incorporates speech recognition functionality enabled by the React library, allowing users to interact with the application using voice commands while taking the quiz. The system also includes features such as user registration and authentication, quiz creation and management, and real-time quiz results display.

The project aims to address the accessibility challenges faced by visually impaired individuals in taking quizzes and promote inclusivity in the education system. The system's effectiveness and usability were evaluated through user testing and feedback, which showed positive results. Overall, the Quiz Management System for Blind People demonstrates the potential of technology to facilitate inclusivity and accessibility for individuals with visual impairments in education.

TABLE OF CONTENTS

Page No.

LIST OF FIGURES

Figure title **Page no.**

LIST OF ABBREVIATIONS

MERN	MongoDB, ExpressJs, ReactJs & NodeJs
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
UI	User Interface
PHP	Hypertext Preprocessor
HTTP	Hypertext Transfer Protocol
API	Application Programming Interface
CRUD	Create, Read, Update & Delete
DB	Database
VS Code	Visual Studio Code

CHAPTER-1

INTRODUCTION

1.1 Web Programming

Web programming refers to the process of creating websites and web applications using programming languages, frameworks, and tools that are specifically designed for the web. It involves designing, developing, testing, and maintaining web applications that are accessible over the internet or intranet.

Web programming includes a wide range of technologies and skills, including HTML, CSS, JavaScript, server-side programming languages such as PHP, Python, Ruby, databases such as MySQL, MongoDB, and frameworks such as React, Angular, Vue.js, and Django. These technologies and tools are used to create dynamic and interactive web pages, web applications, and e-commerce sites.

Web programming has become an essential skill in today's digital world, as almost every business, organization, and individual relies on the internet and web applications to communicate, transact, and perform various tasks.

1.1.1 Overview

There are two broad divisions of web development – front-end development (also known as client-side development) and back-end development (also known as server-side development).

Frontend development, also known as client-side development, refers to the process of creating the user-facing portion of a website or web application. It involves designing, developing, and implementing the interface, which includes the layout, design, and functionality of a website or web application that users interact with directly.

Frontend development primarily involves working with three main technologies: HTML, CSS, and JavaScript. HTML is used to structure the content and layout of a website, while CSS is used to style the content, and JavaScript is used to add interactivity and dynamic functionality to the user interface. Popular frontend frameworks and libraries include React, Angular, Vue.js, and jQuery, among others.

Backend development, also known as server-side development, refers to the process of creating the non-visible portion of a website or web application. It involves developing and maintaining the server-side logic, database, and APIs that support the frontend interface and enable data processing, storage, and retrieval. Backend developers work with server-side programming languages such as PHP, Python, Ruby, and Java, as well as databases such as MySQL, MongoDB, and PostgreSQL.

1.1.2 Description

React.js is an open-source JavaScript library that is used for building user interfaces specifically for single-page applications. It’s used for handling the view layer for web and mobile apps. React also allows us to create reusable UI components. React allows developers to create large web applications that can change data, without reloading the page. The main purpose of React is to be fast, scalable, and simple. It works only on user interfaces in the application.

1.2 MERN Stack

MERN Stack is a collection of powerful and robust technologies, used to develop scalable master web applications comprising **backend, front-end,** and **database components**. It is JavaScript that is used for the faster and easier development of full-stack web applications. MERN Stack is a technology that is a user-friendly full-stack JavaScript framework for building applications and dynamic websites. MERN stack comprises 4 components namely: MongoDB, ExpressJS, ReactJS and NodeJS. Using the MERN stack, developers can create fast, scalable, and efficient web applications with ease. The MERN stack is known for its high performance, scalability, flexibility, and reusability. It also benefits from being built on open-source technologies, making it free to use and customizable. Overall, the MERN stack provides developers with a powerful and flexible platform for building modern web applications that meet the needs of users in today's digital landscape. For project reports, the MERN stack can be a suitable choice for building a wide range of web applications with complex features and high levels of interactivity.

1.2.1 MERN Description

MERN is a popular web development stack that consists of four main components:

1. MongoDB: A NoSQL database that stores data in JSON-like documents with dynamic schemas. MongoDB is used to store the data for MERN stack applications.
2. Express.js: A lightweight framework for building web applications in Node.js. Express.js is used to handle server-side routing, middleware, and other HTTP-related tasks in MERN stack applications.
3. React.js: A JavaScript library for building user interfaces. React.js is used to create the frontend of MERN stack applications, enabling developers to create dynamic and interactive user interfaces.
4. Node.js: A server-side JavaScript runtime environment that allows developers to run JavaScript code on the server. Node.js is used to run the server-side code for MERN stack applications, providing a scalable and efficient runtime environment.

Together, these four components provide a complete and powerful stack for building full-stack web applications with JavaScript.

1.2.2 Global advantage

There are several global advantages of using the MERN stack for web development:

1. End-to-end JavaScript: MERN stack allows developers to use JavaScript throughout the entire development process, from frontend to backend, which can result in a more streamlined and efficient development process.
2. Open-source: All components of the MERN stack are open-source, meaning developers can leverage a vast array of third-party libraries and tools to accelerate development and solve complex problems.
3. Scalability: MERN stack provides a scalable architecture that can be easily adapted to handle increasing amounts of traffic and data. This is particularly important for web applications that are expected to grow rapidly.
4. Flexibility: MERN stack provides developers with a high degree of flexibility, allowing them to choose the tools and libraries that best suit their needs and preferences.
5. Rapid development: MERN stack is designed to simplify and speed up the development process, allowing developers to quickly prototype, test, and deploy new features and applications.

Overall, MERN stack is an excellent choice for web developers who are looking for a powerful, flexible, and efficient development environment that can handle the most complex and demanding web applications.

CHAPTER – 2

LITERATURE SURVEY

2.1 Literature Survey

Research Paper	Year	Methodology	Pros	Cons
Computerized examination for visually impaired students	2018	Here an information which presents in the screen have been delivered through voice and the blind people will select their answer through the keys on the keyboard.	The machine can read out questions for blind.	Cannot mark answers on the basis of voice commands. Third person is required to monitor and to help the blind candidates.
Voice based online examination for physically challenged	2015	An examination portal for physically challenged people using .NET.	The voice commands will be get from the user and it gets stored the database.	The disadvantage of this system is that it supports yes/no questions.
E-blind exam portal	2019	An examination portal for visually impaired with the use of embedded system and machine learning.	Here an information which presents in the screen has been delivered through voice.	The drawback of this system is that it reads the textual content present in the screen through voice commands.

Fig 2.1 Literature Survey

CHAPTER – 3

REQUIREMENT ANALYSIS

3.1 Problem statement

With the increasing use of the internet and internet devices, many people suffer from eye problems. It has become a common problem for people with weak visual health to suffer from frequent headaches and other related problems.

1. During examinations it becomes difficult for poor visual health students or visually impaired students to give their best.
2. For the visually impaired candidates who want to take the examination requires a writer. The writer writes the answers which the candidates dictate. These candidates face difficulties in using the scribes for examinations. It is hard for them to dictate the answers to the scribes. Because, the scribes might be of lower qualification hence they cannot interpret their words and write them down as they are.
3. Therefore, their overall performance gets affected.

3.2 Solution to problem statement

To approach this problem the application is integrated with the feature of speech recognition provided by ReactJS.

1. Speech recognition is a library in react which when implemented in the application allows the user to take quizzes using their voice.
2. This library uses the Web Speech API supported by google chrome browser.
3. The candidate can now give their examination using this feature. They just have to follow the instructions provided by their mentor and can use the commands given to them to smoothly attempt the exam.

3.3 Software and Hardware Requirements

I. **Software requirements**:

- ❖ Visual Code Studio (VS Code)
- ❖ Web browser
- ❖ MongoDB Atlas
- ❖ Firebase

II. **Hardware requirements**

- ❖ PC/System

3.4 VS Code

VS Code, short for Visual Studio Code, is a free and open-source code editor developed by Microsoft. It is used by developers for writing and editing code in various programming languages such as JavaScript, Python, and C++. VS Code provides a range of features that make it popular among developers.

3.4.1 VS Code UI

Visual Studio Code, also commonly referred to as VS Code, is a source-code editor made by Microsoft with the Electron Framework, for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality. In the Stack Overflow 2021 Developer Survey, Visual Studio Code was ranked the most popular developer environment tool among 82,000 respondents, with 70% reporting that they use it.

VS Code adopts a common user interface and layout of an explorer on the left, showing all of the files and folders one has access to, and an editor on the right, showing the content of the files one has opened. The UI is divided into five areas i.e. Editors, Side bar, Status bar, activity bar and Panels. Below figure 3.2 shows the features that are included in the VS code editor. Below figure 3.2 shows the features that are included in the VS code editor.

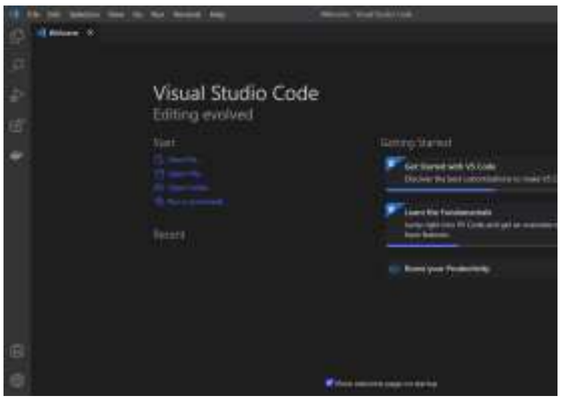


Fig 3.1 VS Code UI

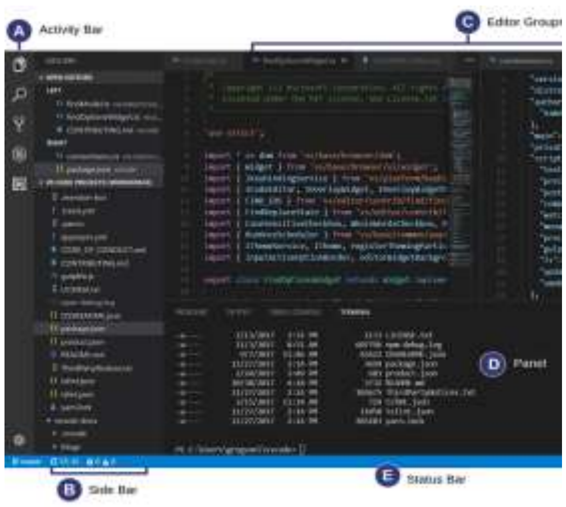


Fig 3.2 VS Code Feature

CHAPTER-4

DETAILED DESIGN

4.1 Architecture of the project

We designed a tentative architecture of our application. Here, the user (teacher/student) gets authenticated through Firebase whenever he/she tries to log in to the system through the frontend. Each user has his/her own dashboard. When the teacher creates a quiz, a secret code is generated and the quiz gets stored in the database. The quiz can later be accessed by the students. Express.js is a web application framework for Node.js, it is used to create the backend of a web application that connects to a database. It uses a database to retrieve or store data and return the results to the frontend.

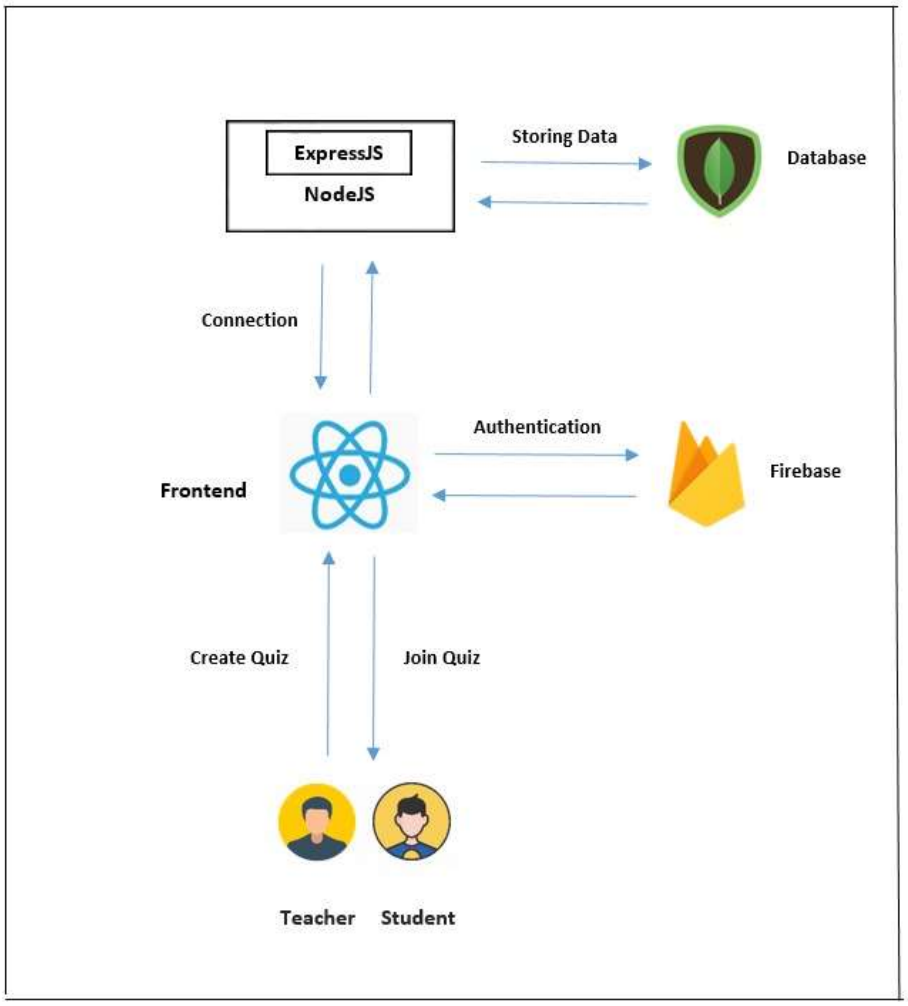


Fig 4.1 Architecture of the Project

CHAPTER 5

IMPLEMENTATION

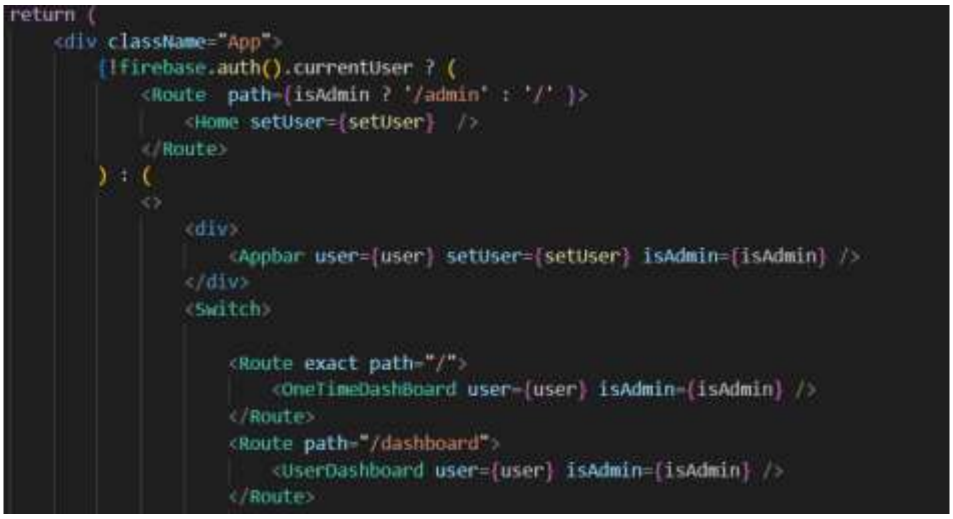
5.1 Frontend development

We designed a basic frontend to interact with our Web application and test the functionality. We worked on integrating MongoDB and ExpressJS with our React frontend using NodeJS. The login page (as shown below) allows a new user to register as a Teacher or a Student. The teachers can create a quiz and share the link to students so that the students can join and attempt the Quiz with the additional provided feature of speech recognition for the visually impaired. The attempted Quiz with the result of each student is stored on the portal.

5.2 MERN Stack

5.2.1 React JS

We used React.js for the development of frontend of our project. It has the ability to handle complex user interfaces and its strong community support. We started the implementation of React.js by first creating a basic project structure using a tool like Create React App. We then began creating components for the various parts of our user interface, using React's component-based architecture. We also utilized React's state and props system to manage the data flow between components and ensure that the UI remained responsive and up-to-date.



```
return (
    <div className="App">
        {!firebase.auth().currentUser ? (
            <Route  path={isAdmin ? '/admin' : '/' }>
                <Home setUser={setUser}  />
            </Route>
        ) : (
            <>
                <div>
                    <Appbar user={user} setUser={setUser} isAdmin={isAdmin} />
                </div>
                <Switch>

                    <Route exact path="/">
                        <OneTimeDashBoard user={user} isAdmin={isAdmin} />
                    </Route>
                    <Route path="/dashboard">
                        <UserDashboard user={user} isAdmin={isAdmin} />
                    </Route>
```

Fig 5.1 ReactJS

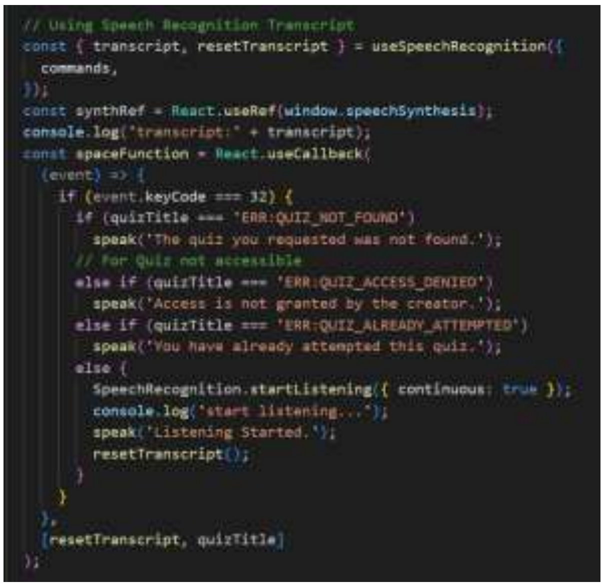
5.2.1.1 React-speech-recognition

In order to implement the Speech Recognition feature for the visually impaired users, here we use the library known as React-Speech-Recognition.

useSpeechRecognition is a React hook that gives a component access to a transcript of speech picked up from the user's microphone. SpeechRecognition manages the global state of the Web Speech API, exposing functions to turn the microphone on and off.

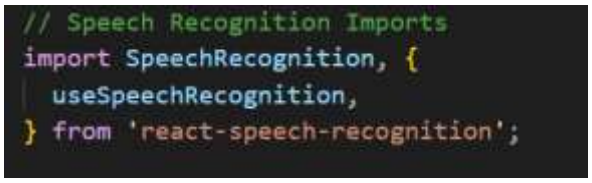
In order to use this library we first installed it using npm and imported it in our project files as shown in fig 5.3.

Next, we use this import in our function using the Speech Recognition Transcript function provided by the library as shown in fig 5.2. We check for whether the Quiz is found or has been already attempted or needs to be attempted. The function speak() is used to pass an alert in the form of voice output for every test case respectively.



```
// Using Speech Recognition Transcript
const { transcript, resetTranscript } = useSpeechRecognition({
  commands,
});
const synthRef = React.useRef(window.speechSynthesis);
console.log('transcript:' + transcript);
const spaceFunction = React.useCallback(
  (event) => {
    if (event.keyCode === 32) {
      if (quizTitle === 'ERR:QUIZ_NOT_FOUND')
        speak('The quiz you requested was not found.');
      // For Quiz not accessible
      else if (quizTitle === 'ERR:QUIZ_ACCESS_DENIED')
        speak('Access is not granted by the creator.');
      else if (quizTitle === 'ERR:QUIZ_ALREADY_ATTEMPTED')
        speak('You have already attempted this quiz.');
      else {
        SpeechRecognition.startListening({ continuous: true });
        console.log('start listening...');
        speak('Listening Started.');
        resetTranscript();
      }
    }
  },
  [resetTranscript, quizTitle]
);
```

Fig 5.2 Speech Recognition

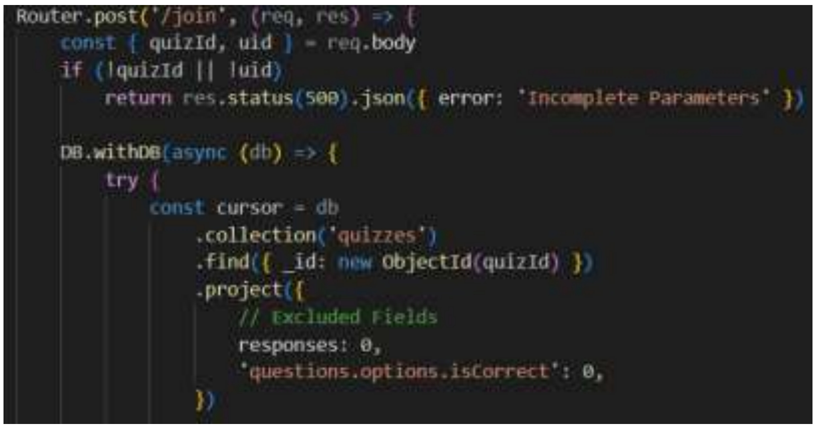


```
// Speech Recognition Imports
import SpeechRecognition, {
  useSpeechRecognition,
} from 'react-speech-recognition';
```

Fig 5.3: Import Speech Recognition

5.2.2 Node JS

We used Node.js for the backend of our project due to its event-driven, non-blocking I/O model and scalability. We started by setting up a basic project structure using a tool like Express.js, which allowed us to handle HTTP requests and responses and implement routing as shown in fig 5.4. We also used middleware like body-parser to parse request bodies. Using Node.js, we were able to easily connect to our database using a package like Mongoose, which provided an object modeling system and made it easy to interact with MongoDB.

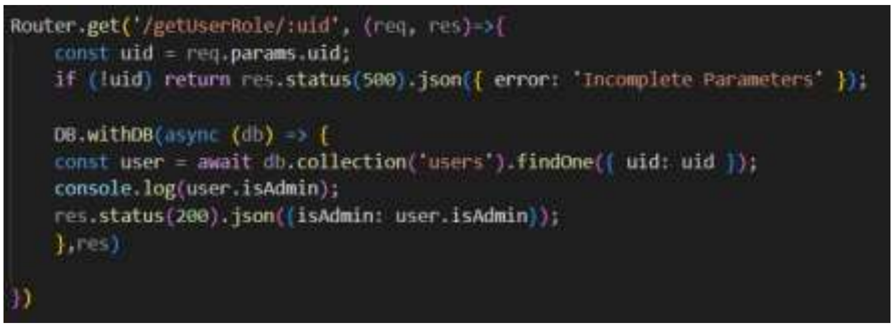


```
Router.post('/join', (req, res) => {
    const { quizId, uid } = req.body
    if (!quizId || !uid)
        return res.status(500).json({ error: 'Incomplete Parameters' })

    DB.withDB(async (db) => {
        try {
            const cursor = db
                .collection('quizzes')
                .find({ _id: new ObjectId(quizId) })
                .project({
                    // Excluded Fields
                    responses: 0,
                    'questions.options.isCorrect': 0,
                })
```

Fig 5.4 NodeJS

5.2.3 Express JS

Express.js provided a flexible and powerful framework for handling HTTP requests and responses and implementing routing. With Express.js, we were able to easily create RESTful API endpoints for our frontend to interact with, using middleware to parse request bodies and handle errors as shown in fig 5.5.

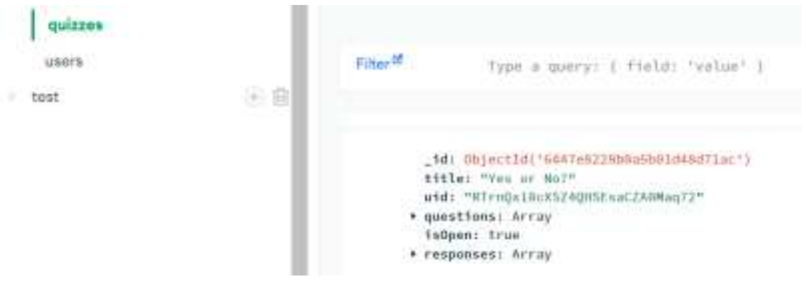


```
Router.get('/getUserRole/:uid', (req, res)=>{
    const uid = req.params.uid;
    if (!uid) return res.status(500).json({ error: 'Incomplete Parameters' });

    DB.withDB(async (db) => {
    const user = await db.collection('users').findOne({ uid: uid });
    console.log(user.isAdmin);
    res.status(200).json({isAdmin: user.isAdmin});
    },res)

})
```

Fig 5.5 ExpressJS

5.2.4 MongoDB

We used the MongoDB Atlas to connect the frontend of our project with the use of username and password in the key generated by the official MongoDB platform and stored the key in a db_config.js file. The collection of database is shown in fig 5.6.



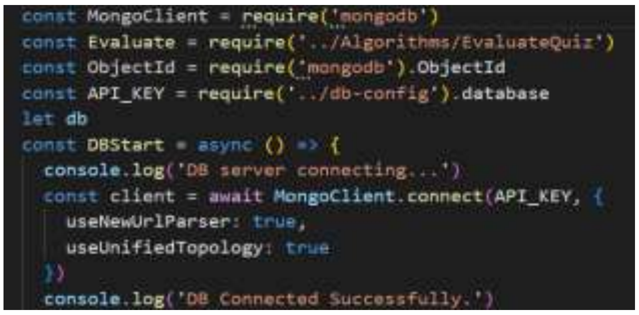
```
_id: ObjectId('6447e8228b0a5b01d48d71ac')
title: "Yes or No?"
uid: "RTrnQx18cX5Z4QHSEsaCZA0Maq72"
▸ questions: Array
isOpen: true
▸ responses: Array
```

Fig 5.6 MongoDB Atlas Connection

5.2.4.1 MongoClient

MongoClient is the official MongoDB driver for Node.js. It allows Node.js applications to connect to a MongoDB database and perform database operations. With MongoClient we can perform CRUD operations.

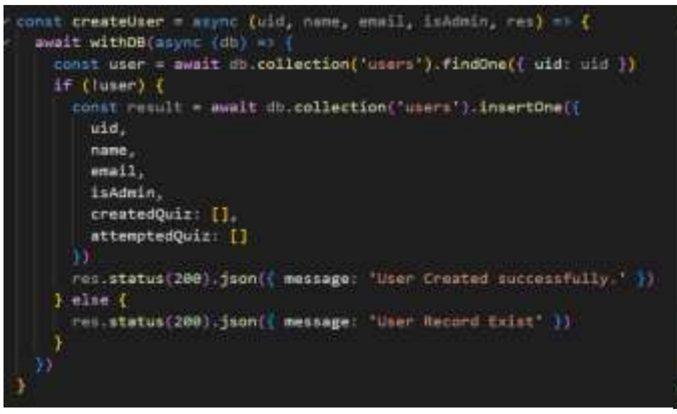
To use MongoClient in our Node.js application, we first installed it using npm. Then created a new MongoClient instance and connected to our MongoDB database by importing MongoClient in the files as shown in fig 5.7. After that we wrote a function DBStart() to establish connection.



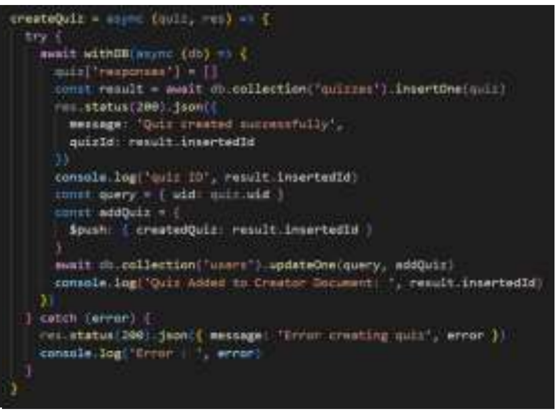
```
const MongoClient = require('mongodb')
const Evaluate = require('../Algorithms/EvaluateQuiz')
const ObjectId = require('mongodb').ObjectId
const API_KEY = require('../db-config').database
let db
const DBStart = async () => {
  console.log('DB server connecting...')
  const client = await MongoClient.connect(API_KEY, {
    useNewUrlParser: true,
    useUnifiedTopology: true
  })
  console.log('DB Connected Successfully.')
```

Fig 5.7 MongoClient

Next, after establishing the connection we made several functions in order to create a cluster and collection to create the user (Teacher or Student) and store the data inputted by the Teacher in order to create a Quiz as shown in fig 5.8 and 5.9 to record the responses received by each of the students who have attempted the Quiz for the further evaluation of the result.



```
const createUser = async (uid, name, email, isAdmin, res) => {
  await withDB(async (db) => {
    const user = await db.collection('users').findOne({ uid: uid })
    if (!user) {
      const result = await db.collection('users').insertOne({
        uid,
        name,
        email,
        isAdmin,
        createdQuiz: [],
        attemptedQuiz: []
      })
      res.status(200).json({ message: 'User Created successfully.' })
    } else {
      res.status(200).json({ message: 'User Record Exist' })
    }
  })
}
```



```
createQuiz = async (quiz, res) => {
  try {
    await withDB(async (db) => {
      quiz['responses'] = []
      const result = await db.collection('quizzes').insertOne(quiz)
      res.status(200).json({
        message: 'Quiz created successfully',
        quizId: result.insertedId
      })
      console.log('Quiz ID', result.insertedId)
      const query = { uid: quiz.uid }
      const addQuiz = {
        $push: { createdQuiz: result.insertedId }
      }
      await db.collection('users').updateOne(query, addQuiz)
      console.log('Quiz Added to Creator Document: ', result.insertedId)
    })
  } catch (error) {
    res.status(200).json({ message: 'Error creating quiz', error })
    console.log('Error : ', error)
  }
}
```

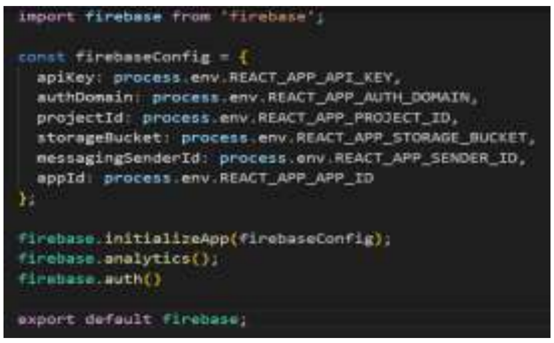
Fig 5.8 Function to Create User

Fig 5.9 Function to Create Quiz

5.2.5 Firebase

Chrome Firebase is a combination of the Google Chrome web browser and Firebase, a mobile and web application development platform. It offers real-time database services, authentication, cloud messaging, and analytics to help developers create robust and scalable applications.

We used Firebase in our project for the authentication of the user login. We made an account on the chrome Firebase and added the code provided by the documentation in the firebase.js file as shown in fig 5.10. Next, we imported this file in our home page and used it to provide the options of google and email login as shown in fig 5.11.



```
import firebase from 'firebase';

const firebaseConfig = {
  apiKey: process.env.REACT_APP_API_KEY,
  authDomain: process.env.REACT_APP_AUTH_DOMAIN,
  projectId: process.env.REACT_APP_PROJECT_ID,
  storageBucket: process.env.REACT_APP_STORAGE_BUCKET,
  messagingSenderId: process.env.REACT_APP_SENDER_ID,
  appId: process.env.REACT_APP_APP_ID
};

firebase.initializeApp(firebaseConfig);
firebase.analytics();
firebase.auth()

export default firebase;
```

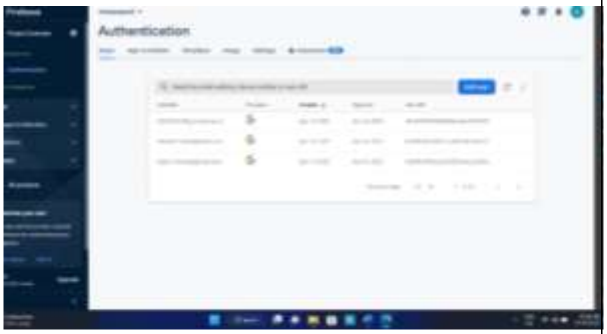


Fig 5.10 Firebase Configuration

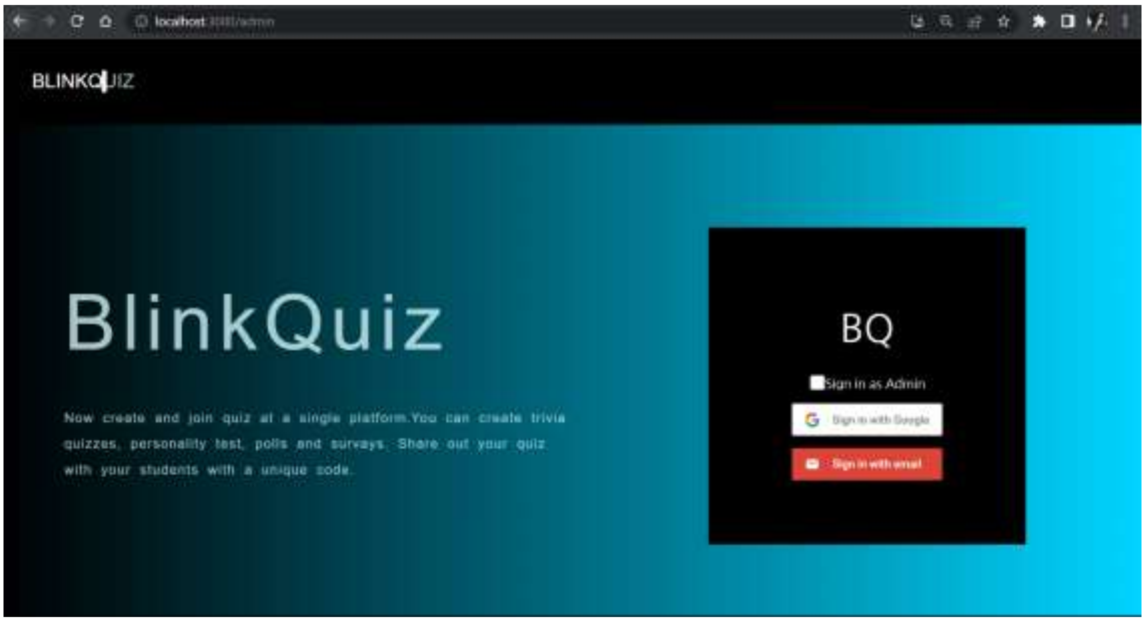
Fig 5.11 Authorization Provider

CHAPTER 6

EXPERIMENTAL RESULTS AND ANALYSIS

6.1 Homepage

Homepage of our application consists of a small introduction of our project with login model where the user can choose his/her role as a teacher or as a student as shown in fig 6.1.



Now create and join quiz at a single platform.You can create trivia quizzes, personality test, polls and surveys. Share out your quiz with your students with a unique code.

Fig 6.1 Homepage

6.1 Login as student/teacher

Two buttons are provided by the application for the login to the respective portals of the Student and Teacher. Teachers have to tick on the checkbox whether or not he/she is logging as a teacher as shown in fig 6.3 whereas the students can login directly as shown in fig 6.2. After doing so the user can enter their credentials and can login/ sign-up using Google account or using their email-id as shown in fig 6.4. After entering the details, the authorization is done with the help of Google firebase and hence the respective portals open up.

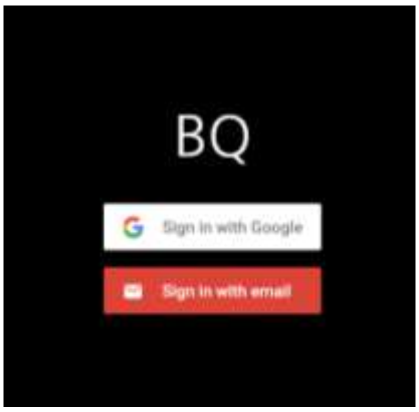


Fig 6.2 Login as a Student

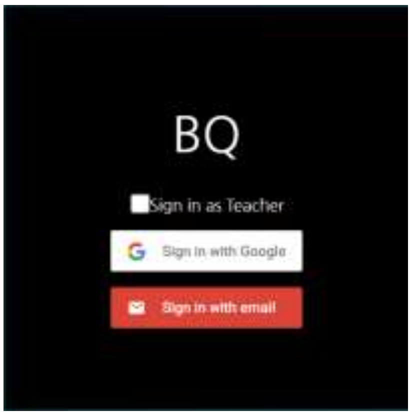


Fig 6.3 Login as a Teacher

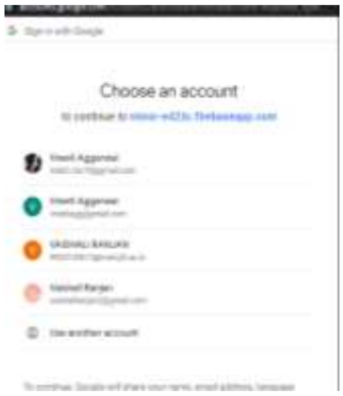


Fig 6.4 Choose Account

6.2 Teachers' Portal

Upon logging in as a teacher, a portal appears which gives us various options as shown in fig 6.5:

- ❖ Dashboard
- ❖ Create quiz

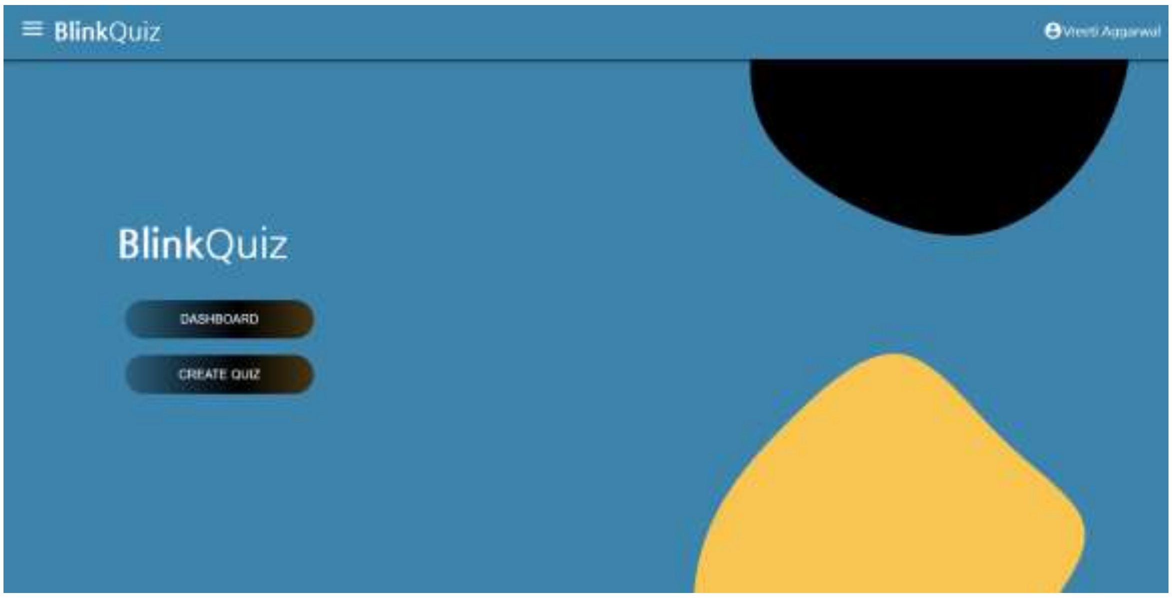


Fig 6.5 Teacher's Portal

6.2.1 Teacher’s Dashboard

On clicking on the dashboard a page opens which displays all the quizzes created by the teacher. On opening a quiz, the teacher can see the score of the students who have attempted the quiz and all the other relevant information related to the quiz as shown in fig 6.6. The Teacher can also close the quiz to disable the code. When the teacher disables the quiz, the students can no longer access the quiz.

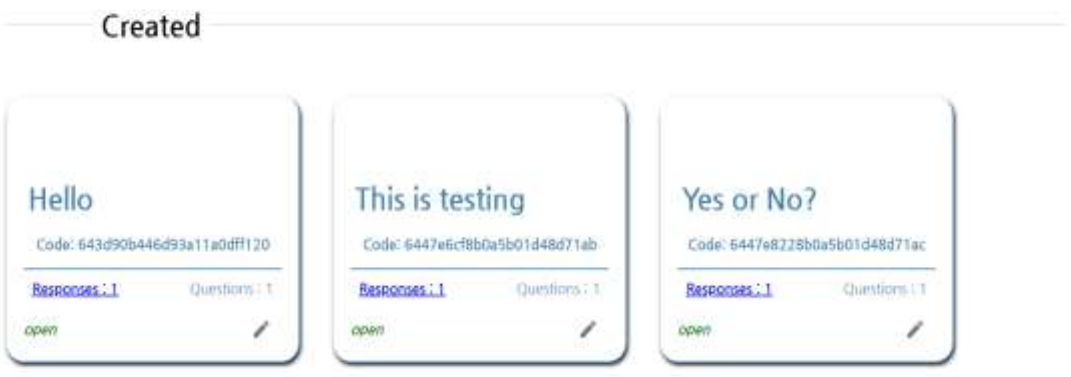


Fig 6.6 Teacher’s Dashboard

Teachers/students can also log out from the system by clicking on the “Sign Out” button present on the sidebar of the portal as shown in fig 6.7. When the user signs out from the system, the user is redirected to the homepage of our system.

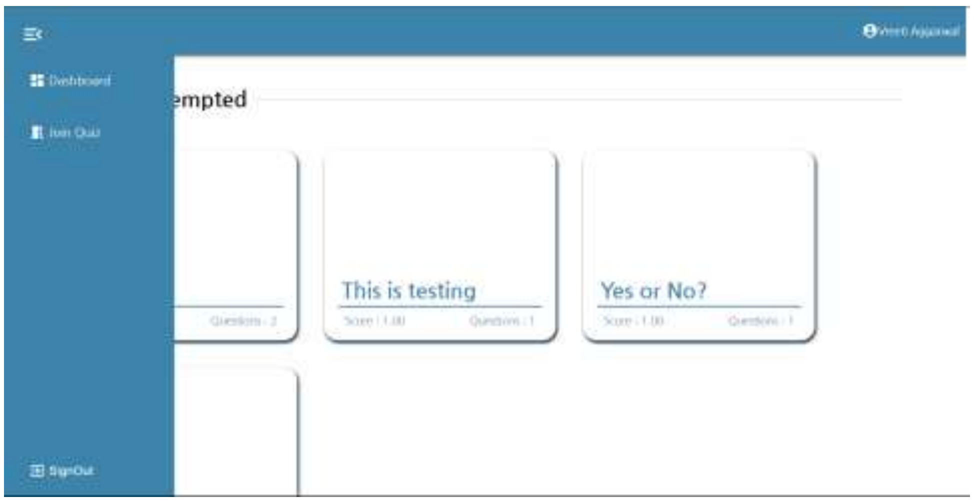


Fig 6.7 Sidebar

6.2.2 Create Quiz

On clicking the create quiz, the teacher can create a quiz. To create a quiz the teacher first has to give a Title of the quiz. They can add questions one by one. He/she can add any number of questions or options. The exact view of creation of quiz can be seen in fig 6.8.

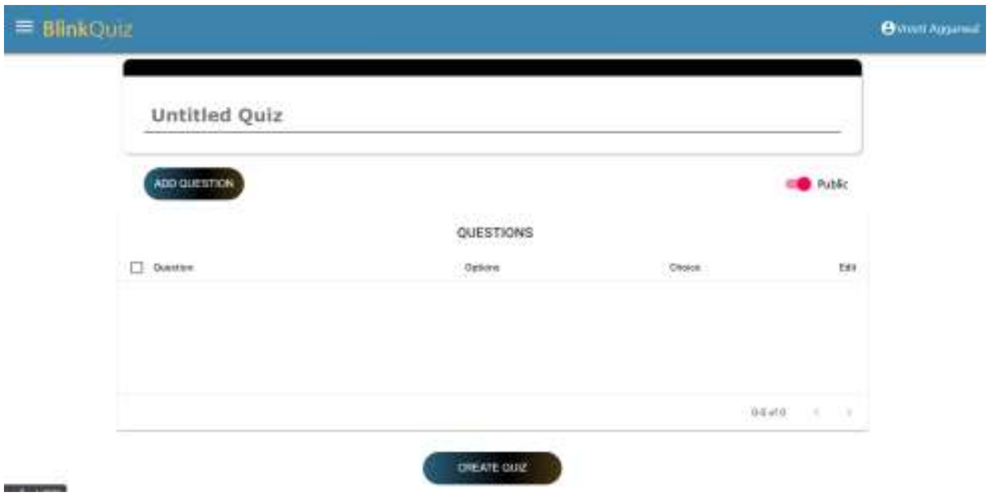


Fig 6.8 Create Quiz

They can select if a particular question has multiple correct answers or a single answer, it can be seen that the current question consist of single correct answer in fig 6.9. If the question has a single correct answer, then the system will use radio buttons. If the question consists of multiple correct answers, then the system will use checkboxes.

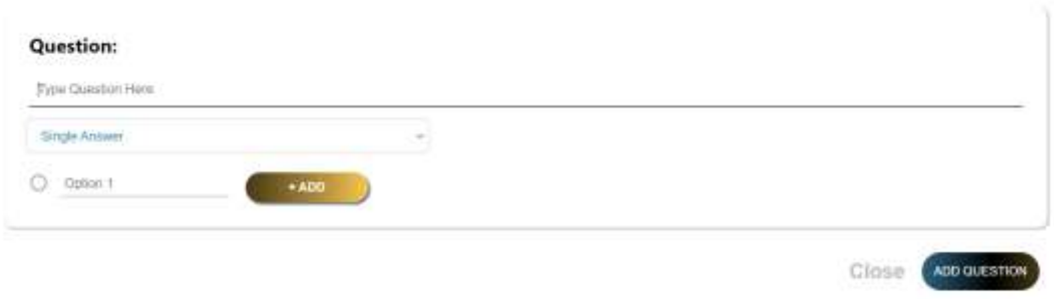
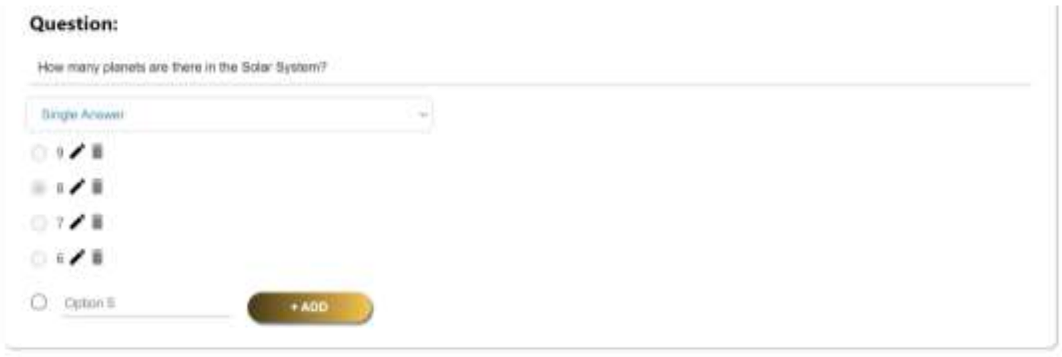


Fig 6.9 Add Questions

After entering the quiz title and questions, the teacher can add options available for the question. The teacher can also edit the options as shown in fig 6.10. An option cannot be repeated more than once.


Question:

How many planets are there in the Solar System?

Single Answer

9

8

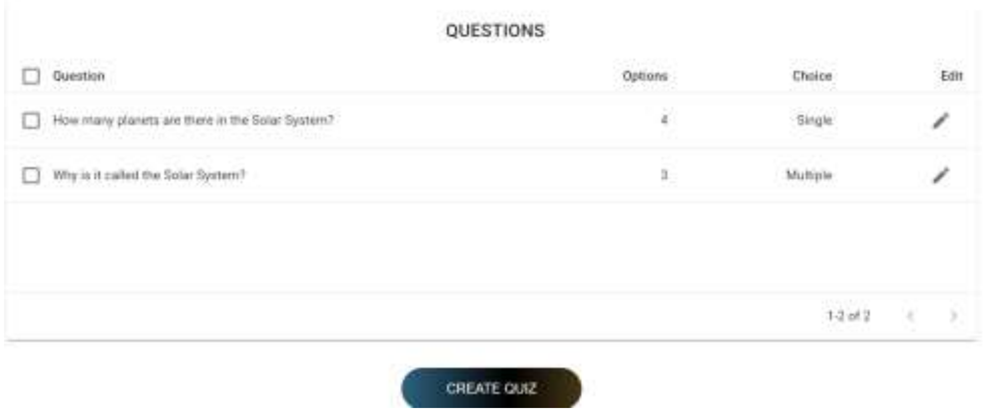
7

6

Option 5

Fig 6.10 Add Options

After entering the quiz title, required questions and their options, a screen with questions, single/multiple choice and number of options is displayed to the teacher as shown in fig 6.11. Teacher can edit the question as well as options available. The screen also displays the number of questions in the quiz. Once the teacher finalize the questions and options he/she can press onto create quiz button to generate the code of the quiz created.


QUESTIONS

Question	Options	Choice	Edit
How many planets are there in the Solar System?	4	Single	
Why is it called the Solar System?	3	Multiple	

1-2 of 2

Fig 6.11 Questions in the Quiz

After creating the quiz, a special code is generated which can be shared amongst the students to join. The questions along with the options are stored in the database. Every quiz generates a unique code which can be used by the students to join the quiz as shown in fig 6.12. After generating the code teacher can copy the code and share it to the students and move back to dashboard.

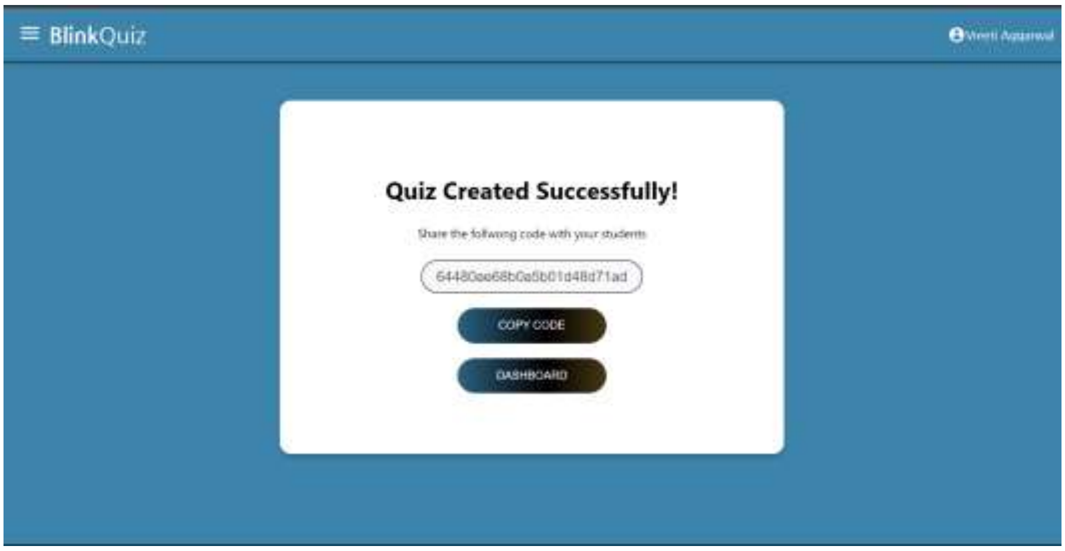

BlinkQuiz
Vineti Aggarwal
Quiz Created Successfully!
Share the following code with your students
64480ee68b0a5b01d48d71ad
COPY CODE
DASHBOARD

Fig 6.12 Generated Code

6.2 Student's Portal

Upon logging in as a student, a portal appears which gives us various options as shown in fig 6.13:

- Dashboard
- Join quiz

Students can also log out from the system by clicking on the "Sign Out" button present on the sidebar of the portal. When the user sign out from the system, the user is redirected to the homepage of our system.

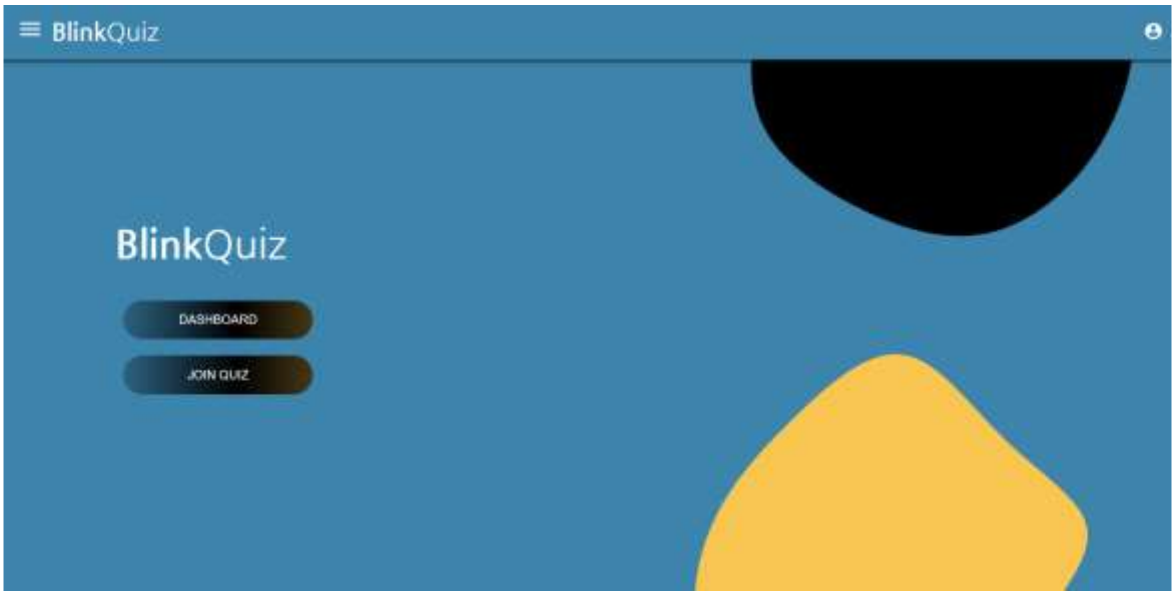
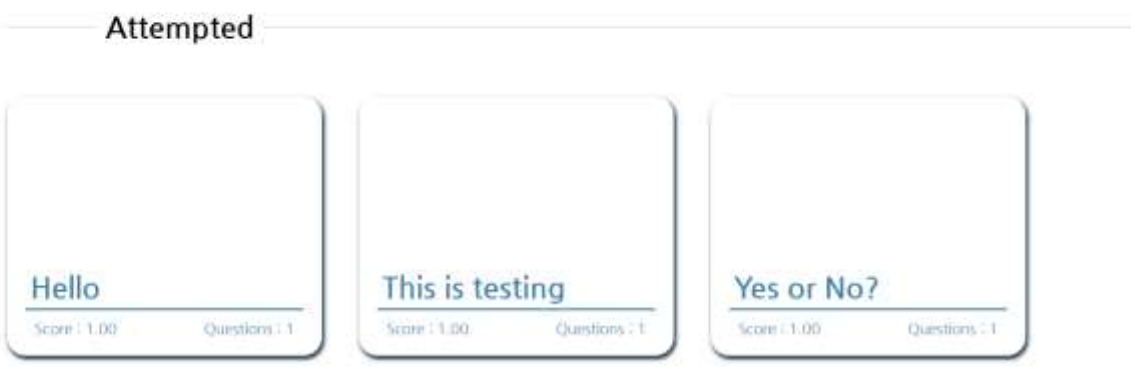


Fig 6.13 Student's Portal

6.2.1 Student's Dashboard

On clicking on the dashboard a page opens which displays all the quizzes attempted by the student. It also highlights the secured score in each quiz as shown in fig 6.14.



6.14 Student's Dashboard

6.2.2 Join Quiz

Students can join the quiz by clicking on the "Join Quiz" button. This opens up a modal where the student has to enter the code provided by the teacher which is generated when the teacher creates the quiz as shown in fig 6.15.

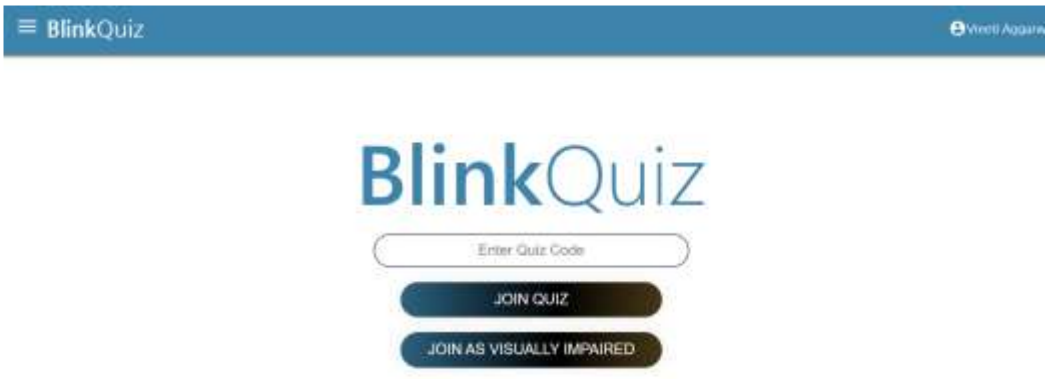
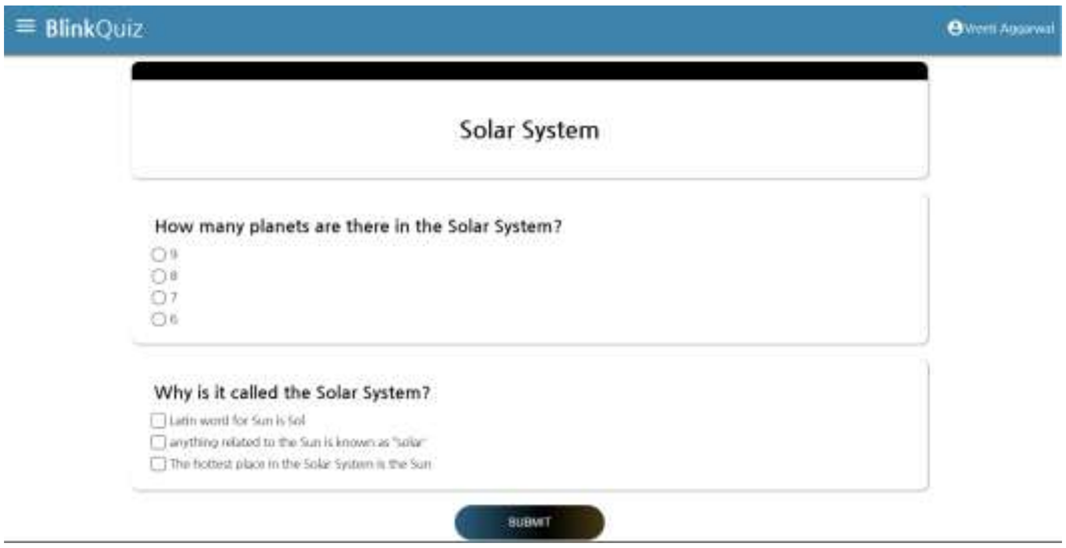


Fig 6.15 Join Quiz

When the quiz gets started, a screen appears with questions and options. The options can be of multiple choice or single choice. Students can take the quiz and submit the quiz at the end. The view of attempting the quiz can be seen in fig 6.16.



Solar System

How many planets are there in the Solar System?

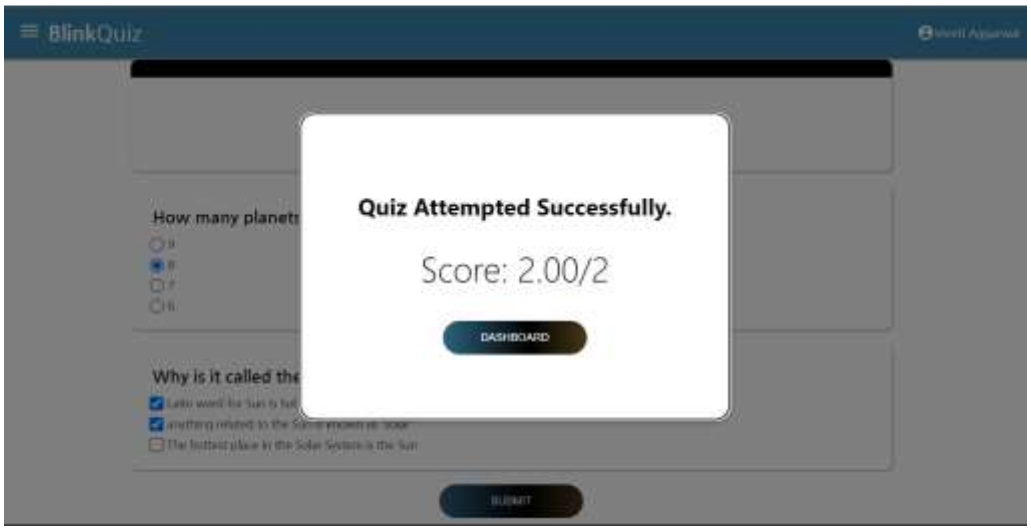
- ○ 9
- ○ 8
- ○ 7
- ○ 6

Why is it called the Solar System?

- ☐ Latin word for Sun is Sol
- ☐ anything related to the Sun is known as "solar"
- ☐ The hottest place in the Solar System is the Sun

Fig 6.16 Attempting Quiz

When the student submits the quiz, a modal appears on the screen which displays the score secured by the student as shown in fig 6.17. The score also gets updated in the teacher's portal.



Quiz Attempted Successfully.

Score: 2.00/2

Fig 6.17 Score Generated

6.2.3 Join Quiz as a visually impaired person

Students can join the quiz by clicking on the **"Join as Visually Impaired"** button as shown in fig 6.18. This opens up a modal where the student has to enter the code provided by the teacher which is generated when the teacher creates the quiz. When the quiz gets started, a screen appears with questions and options. The options can be of multiple choice or single choice. To start the feature, Students need to press the spacebar to turn on the microphone to use any command. Once the microphone is ON, the student can speak some specific voice command mentioned ahead.

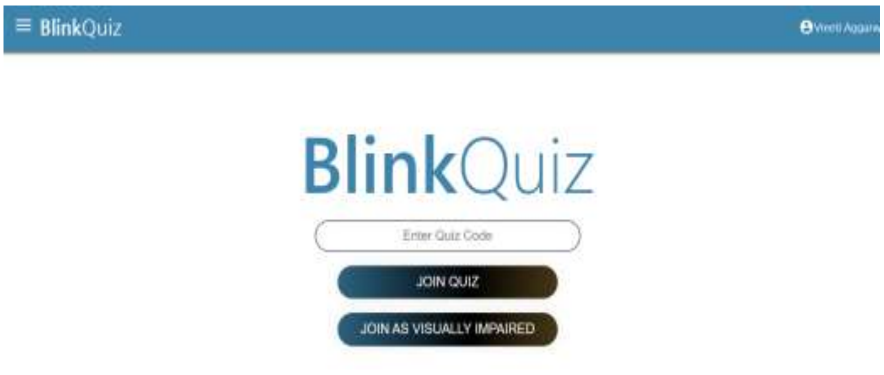


Fig 6.18 Join Quiz as Visually Impaired

To listen to all the possible commands using the **"Instructions"** voice command. The student can use the **"Start Quiz"** voice command to start the quiz, the AI would start dictating the title of the quiz and first question with options and type of question (Single choice/Multiple Choice). In order to select an option, **"Select option [number]"** voice command can be used.

To increment the question index and move to the next question and listen to it, the student can use the **"Next Question"** voice command. To decrement the question index and move to the previous question and listen to it, the student can use the **"Previous Question"** voice command.

The student can also use the **"Repeat Question [number]"** voice command to hear the specific question. **"Repeat Current Question"** can also be used to hear the question again. Lastly when submitting the quiz, the **"Submit Quiz"** voice command should be used. When submitting the quiz, the student gets to know his/her score.

Commands:

- Press space to turn the microphone on.
- Instructions: To listen to all the possible commands.
- Start Quiz or title: To hear the Quiz title and first Question.
- Select Option [Number]: To mark the option of the current Question.
- Next question: to increment the question index and move to the next question and listen to it.
- Previous question: to decrement the question index and move to the previous question and listen to it.
- Repeat Question [Number]: To hear a specific Question.
- Repeat Current Question: To repeat the current Question.
- Submit quiz: to submit the quiz.

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 Conclusion

In conclusion, the development of a quiz application with speech recognition technology using the MERN stack provides a highly accessible and user-friendly experience for all users. The use of MongoDB, Express.js, React.js, and Node.js in the MERN stack provides a flexible and scalable development environment, enabling developers to create a highly functional and interactive application.

The integration of speech recognition technology into the quiz application makes it more inclusive and accessible for individuals who have disabilities or struggle with typing. The application's additional features, including immediate evaluation, make it highly engaging and promote continued use among users. Furthermore, the application's separate portals for teachers and students with login/sign up features ensure a secure and organized experience for all users.

Overall, the MERN stack quiz application with speech recognition technology provides an innovative and user-friendly solution for individuals looking to improve their knowledge and skills in a highly interactive and accessible way.

7.2 Future Scope

This project proposes a method and a good way to create a revolution in the world of education via providing a simpler way for visually challenged people to take tests simply as typical scholars do.

Scope of this project can be extended, by adding a feature that facilitates fingerprint sensor for user login purpose. Login to the examination can be done at ease and moreover, malpractices can be reduced to utmost thereby security can be enhanced. The project can be customized to cater to different exam types such as multiple-choice, essay, or practical exams. This would provide a more tailored experience for students and ensure that the system is suitable for a wide range of exams.

By functional addition of above mentioned features, this project can be more likely used by the visually challenged people to attempt the online examination.

REFERENCES

Book

[1] Eddy Wilson Iriarte Koroliova, *"MERN Quick Start Guide: Build web applications with MongoDB, Express.js, React and Node"*. Packt Publishing.

[2] Vasan Subramanian, "*Pro MERN Stack Second Edition*". Apress.

[3] Greg Lim, "*Beginning MERN Stack (MongoDB, Express, React, Node.js)*".

Journal Article

[4] J. Deepika, D. Jayashree and D. Yamuna Thangam, "*Computerized examination for visually impaired students*", International Journal of Innovative Research in Computer and Communication Engineering, 2017.

[5] Sania Khan, Sanskriti Verma, Shweta Agarwal, Prateek Krishnatrey and Shivam Sharma, "*Voice Based Online Examination for Physically Challenged*", MIT International Journal of Computer Science and Information Technology, 2015.

[6] Pawan Bharadwaj, Tirumala Balaji G, Vallesh Prabhu, Shreehari N and Rahul Kumar, "*E-Blind Exam Portal*", IJISE, 2019.

[7] Mukul Chowdary, Reshma Priyanka, G. Srinivas, M. Rajesh and Dr. N. Leelavathy, "*ONLINE EXAMINATION SYSTEM FOR VISUALLY CHALLENGED*", JETIR, 2019.

Online

[8] "React – A JavaScript library for building user interfaces", https://reactjs.org/

[9] "Introduction to MongoDB", https://learn.mongodb.com/

[10] "Express - Node.js web application framework", http://expressjs.com/

[11] "The MERN Stack Tutorial", https://blog.logrocket.com/mern-stack-tutorial/

[12] "Full Stack MERN Tutorial", https://codedec.com/tutorials/full-stack-mern-tutorial/

Online Courses

[13] Angela Yu, "The Complete 2023 Web Development Bootcamp", Udemy.

[14] Thapa Technical, "MERN Stack in Hindi 2023", Youtube.

[15] Coder Dost, "12-Hour NodeJS Express MongoDB MasterClass 2023", Youtube.

[16] PedroTech, "Modal in ReactJS - Code a React Modal Tutorial using Hooks", Youtube.

[17] Codevolution, "ReactJS Tutorial", Youtube.

[18] Coder Dost, "10-Hour React MasterClass 2023 - Zero to Advanced", Youtube.

Work Distribution

1. All of us explored the basics of the technologies mentioned.
2. All of us explored research papers. Khushi explored the Computerized examination for visually impared students, Vreeti explored the Voice based online examination for physically challenged and Vaishali explored the E-Blind exam portal.
3. All of us brainstormed together to come up with potential features. Vreeti proposed we should use a teacher’s portal to create the quiz. Khushi proposed we should add student’s portal to attempt the quiz. Vaishali proposed that we should use speech-recognition library of ReactJS to attempt the quiz as visually impaired individual.
4. Vaishali worked on the architecture diagram of our project.
5. All of us discussed with some of our classmates and decided to use firebase for our login purpose.
6. All of us discussed the possible structure of the create quiz. Vreeti came up with a basic create quiz form which was further tested and improved by Khushi & Vaishali.
7. Khushi worked on designing the user interface and linking the app with the backend, while Vreeti & Vaishali focussed on creating the server-side of the application.
8. Vaishali and Vreeti worked on integrating the MongoDB, while Khushi continued improving the user interface.
9. All of us brainstormed the required attributes and functions in react to support the speech-recognition feature and developed the Frontend.
10. Khushi focused on adding questions feature while Vreeti looked at editing questions and Vaishali focused on adding options feature.
11. All of us tested our application, refactored the code and prepared the final report and presentation for final evaluation and submission.