JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY, NOIDA

MINOR PROJECT II

# BlinkQUIZ

A QUIZ APPLICATION INCLUDING SPEECH RECOGNITION

## OUR MENTOR:

Dr. Himanshu Agrawal

## TEAM MEMBERS (GROUP NO: 4)

| | | |
|---|---|---|
| VAISHALI RANJAN | KHUSHI KALRA | VREETI AGGARWAL |
| 9920102013 | 9920103025 | 9920103007 |
| F1 | F1 | F1 |

## 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.

During examinations it becomes difficult for poor visual health students or visually impaired students to give their best.

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.

Therefore, their overall performance gets affected.

# OUR APPROACH

*Speech recognition is a library in react which when implemented in the application allows the user to take quizzes using their voice.*

*This library uses the Web Speech API supported by google chrome browser*

*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.*

## STATE OF ART

### Computerized examination for visually impaired students

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.

Limitation: Cannot mark answers on the basis of voice commands. Third person is required to monitor and to help the blind candidates.

## STATE OF ART

### Voice based online examination for physically challenged

An examination portal for physically challenged people using .NET.

The voice commands will be get from the user and it gets stored the database.

Limitation: The disadvantage of this system is that it supports yes/no questions.

# STATE OF ART

## E-Blind exam portal

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.

Limitation: The drawback of this system is that it only reads the textual content present in the screen through voice commands.

## OBJECTIVES

1 *To provide self-registration services for students.*

2 *Response by the candidates will be checked automatically and instantly.*

3 *Online examination will reduce the hectic job of assessing the answers given by the candidates.*

4 *To design a database for handling various modules like Question Bank Candidate details, result analysis.*

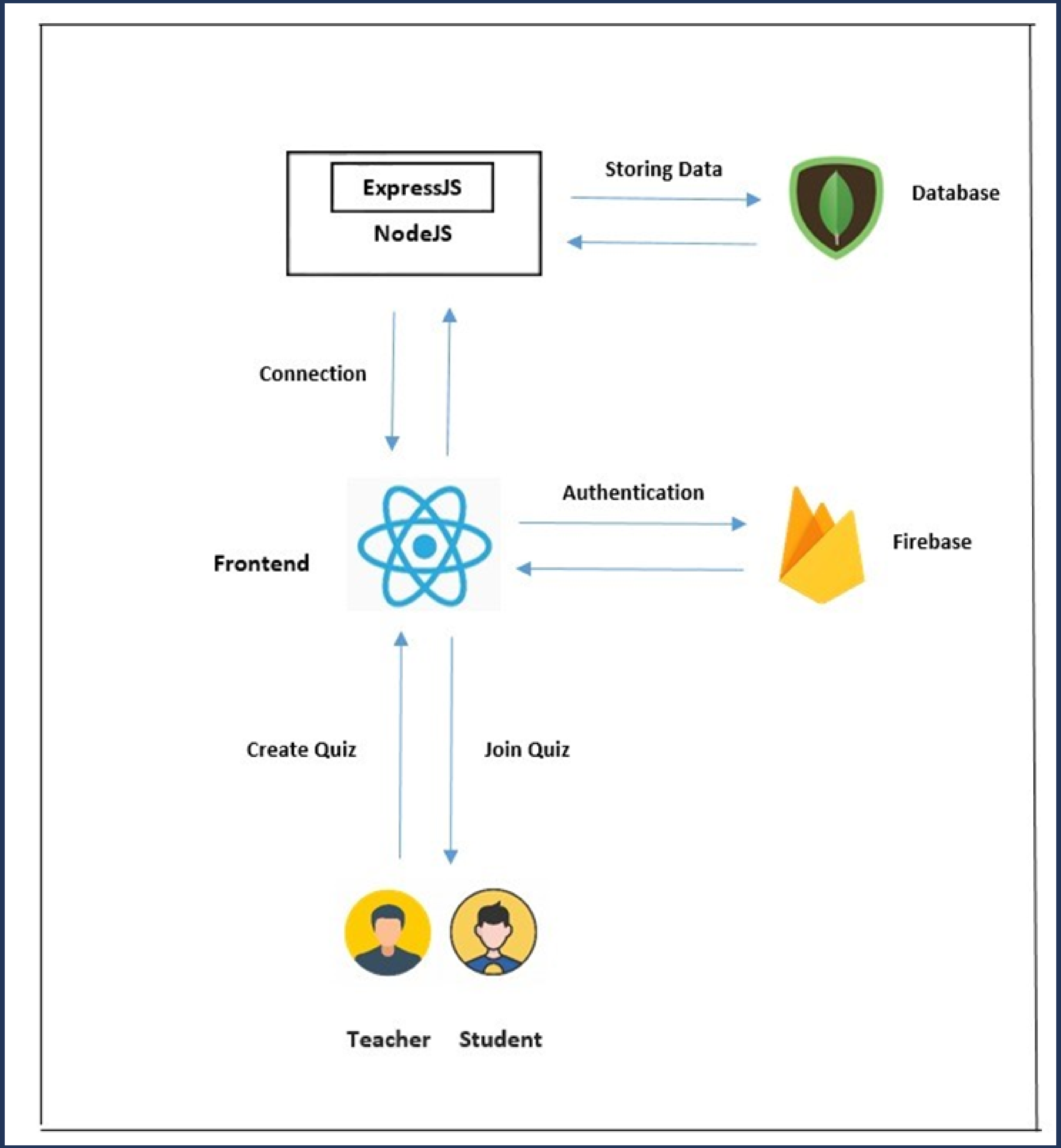
5 *To design a framework for the upliftment of blind users.*

## PROPOSED DESIGN

- 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.

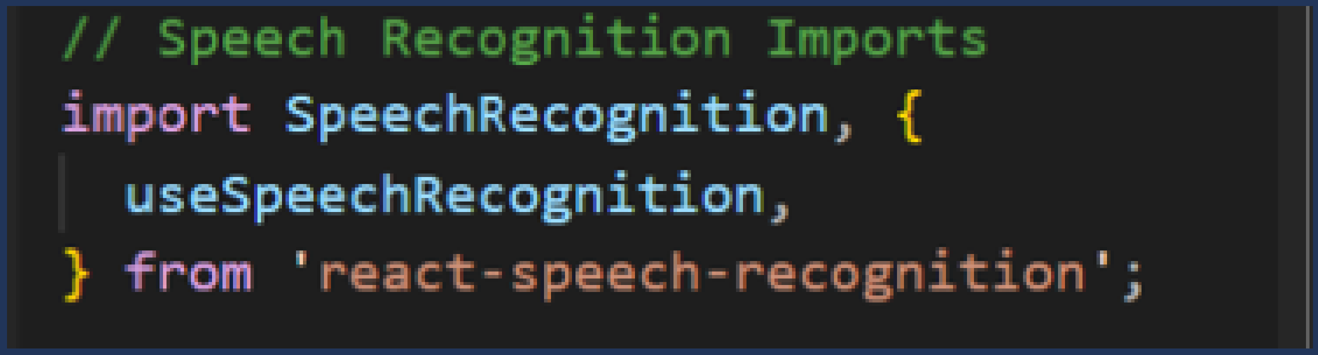
- EXPRESS.JS IS A WEB APPLICATION FRAMEWORK FOR NODE.JS, IT IS USED TO CREATE THE BACKEND OF A WEB APPLICATION THAT CONNECTS TO A DATABASE. IT USES A DATABASE TO RETRIEVE OR STORE DATA AND RETURN THE RESULTS TO THE FRONTEND.



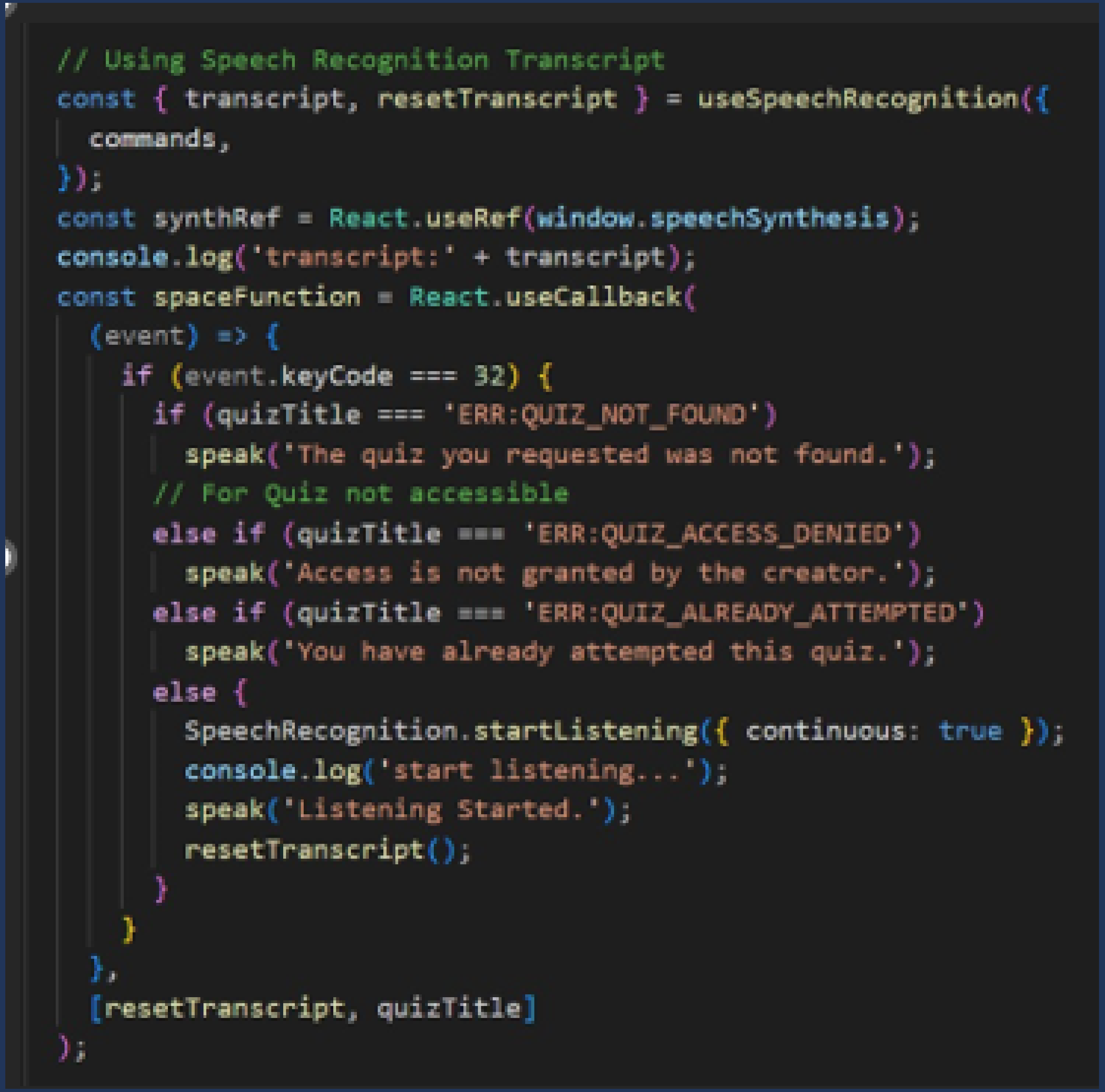
# IMPLEMENTATION

## REACT SPEECH RECOGNITION LIBARRAY

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.



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

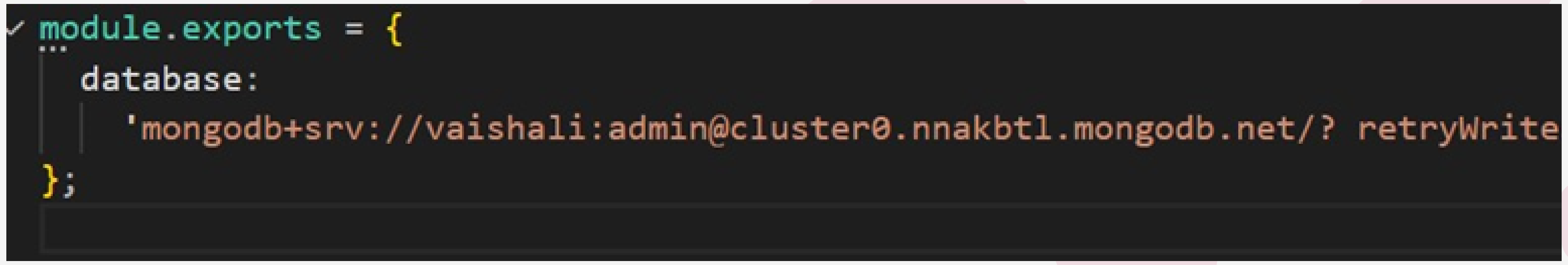


```
// Using Speech Recognition Transcript
const { transcript, resetTranscript } = useSpeechRecognition({
  commands,
});
const synthRef = React.useRef(window.speechSynthesis);
console.log('transcript:' + transcript);
const spaceFunction = React.useCallback(
  (event) => {
    if (event.keyCode === 32) {
      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]
);
```

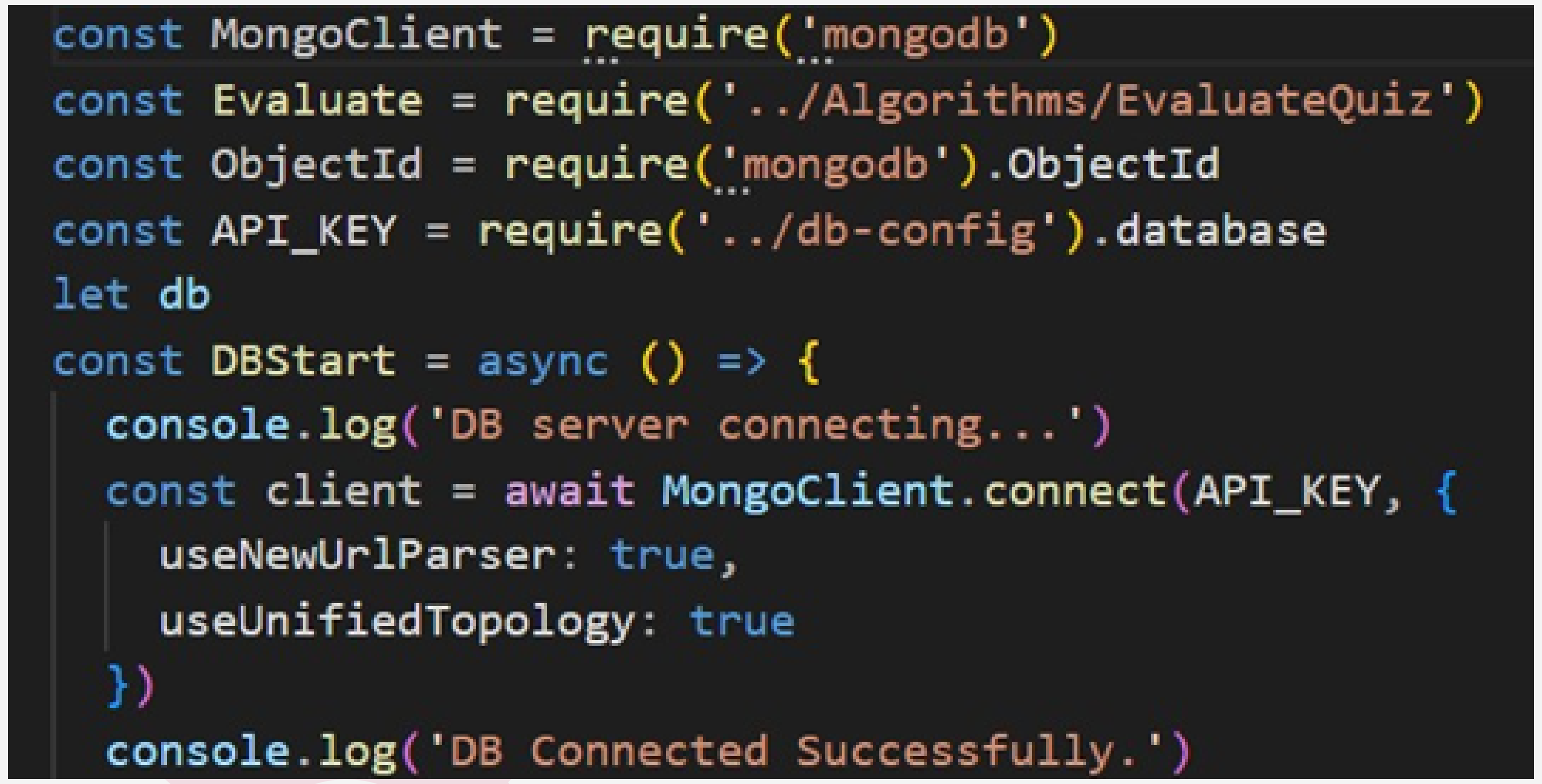
# IMPLEMENTATION

## MONGODB

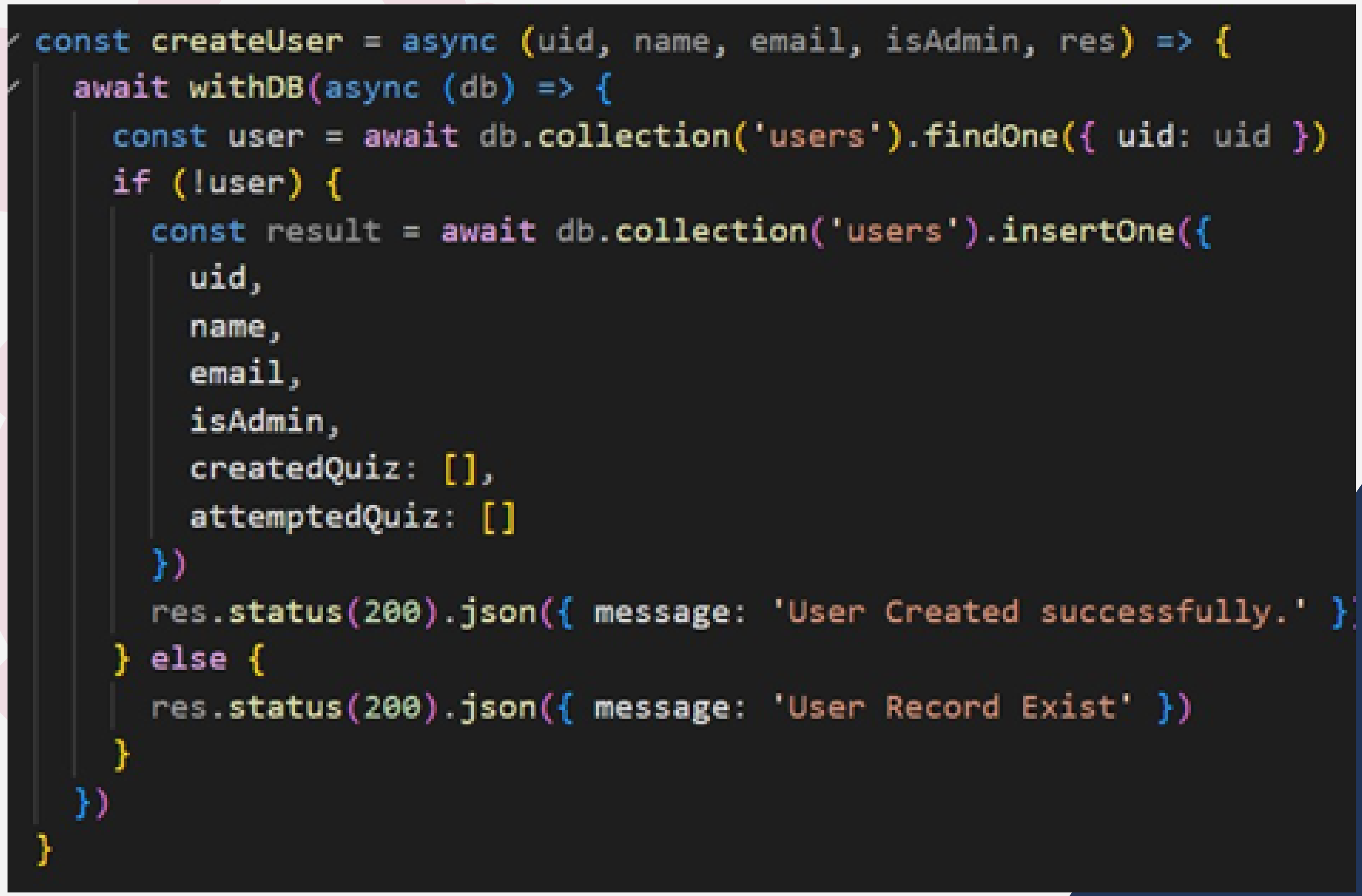
- We used the MongoDB Atlas to connect the frontend of our project with the use of username and password in the key generated by the official MongoDb platform and stored the key in a db_config.js file.
- MongoClient is the official MongoDB driver for Node.js. It allows Node.js applications to connect to a MongoDB database and perform database operations. With MongoClient we can perform CRUD operations.



```
module.exports = {
  database:
    'mongodb+srv://vaishali:admin@cluster0.nnakbtl.mongodb.net/? retryWrite
};
```



```
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.')
```

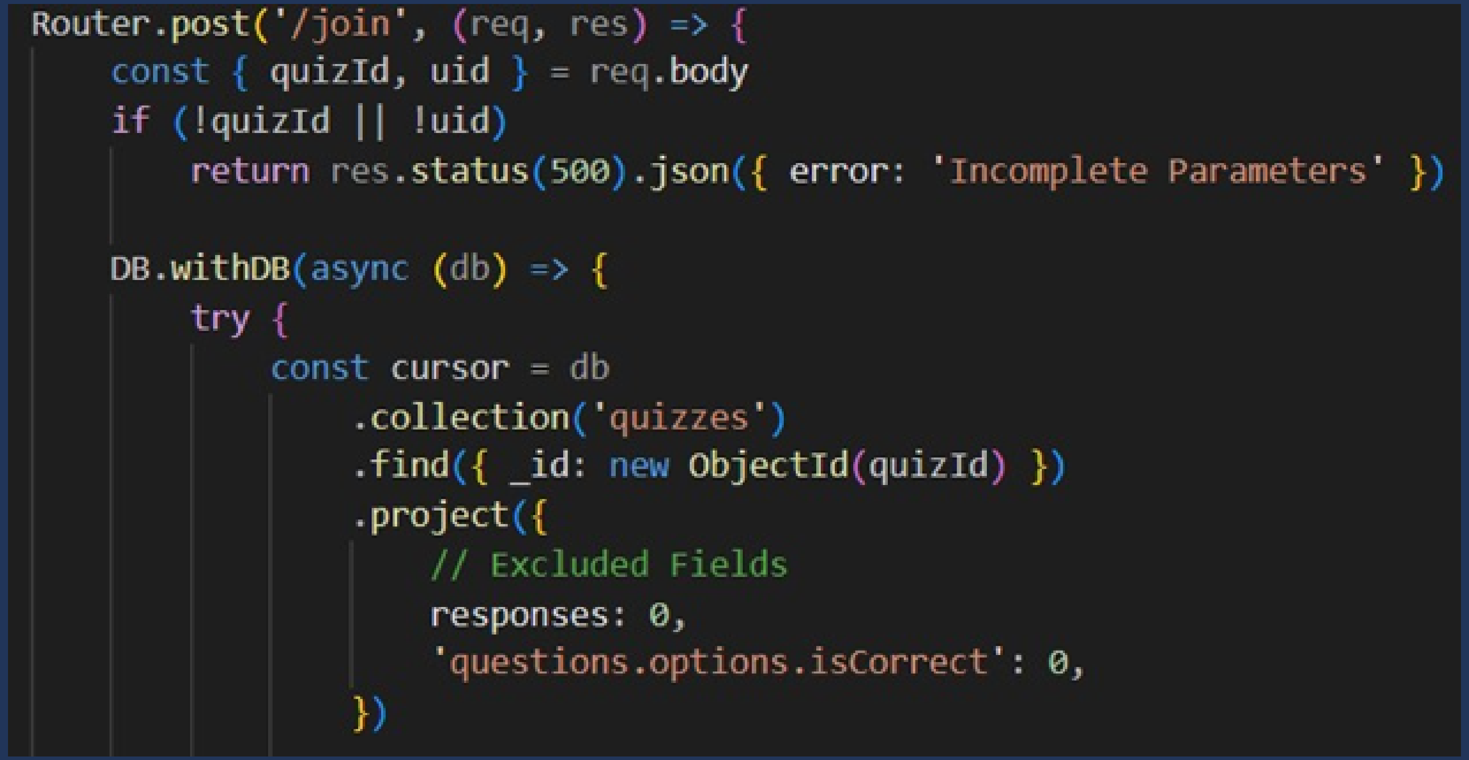


```
const createUser = async (uid, name, email, isAdmin, res) => {
  await withDB(async (db) => {
    const user = await db.collection('users').findOne({ uid: uid })
    if (!user) {
      const result = await db.collection('users').insertOne({
        uid,
        name,
        email,
        isAdmin,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

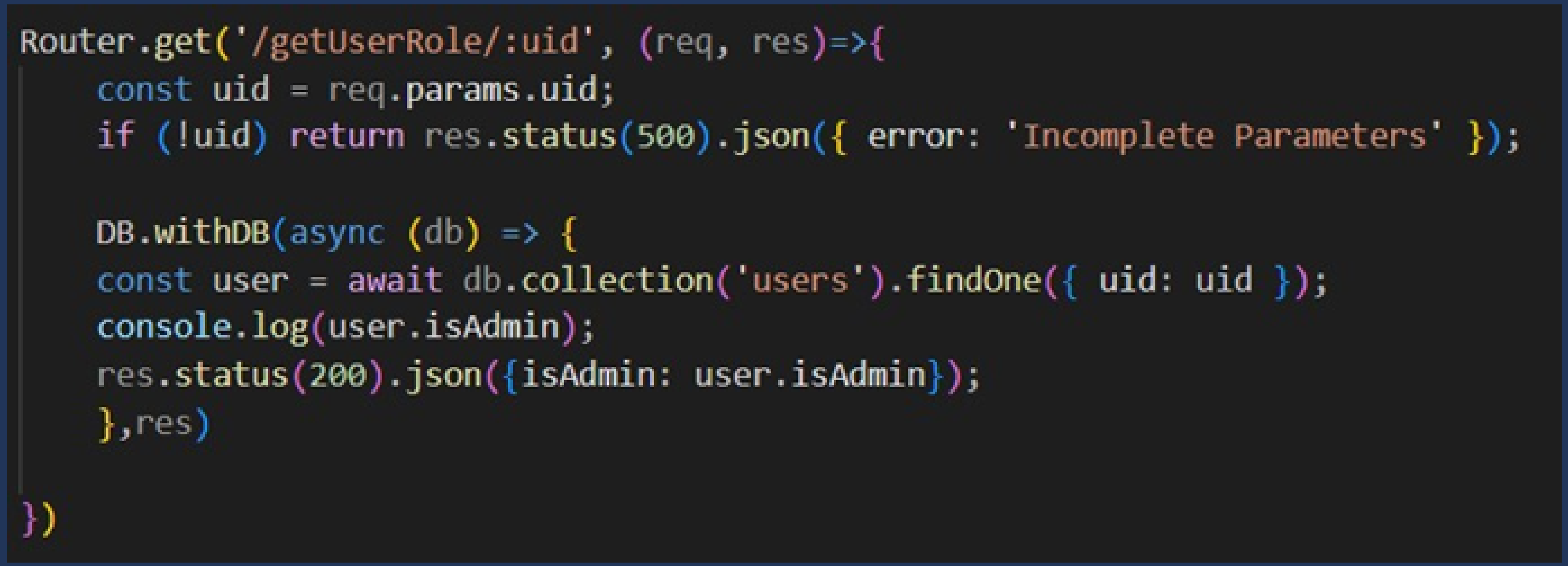
# IMPLEMENTATION

## NODEJS AND EXPRESSJS

- 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
- 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



```
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,
                })
```

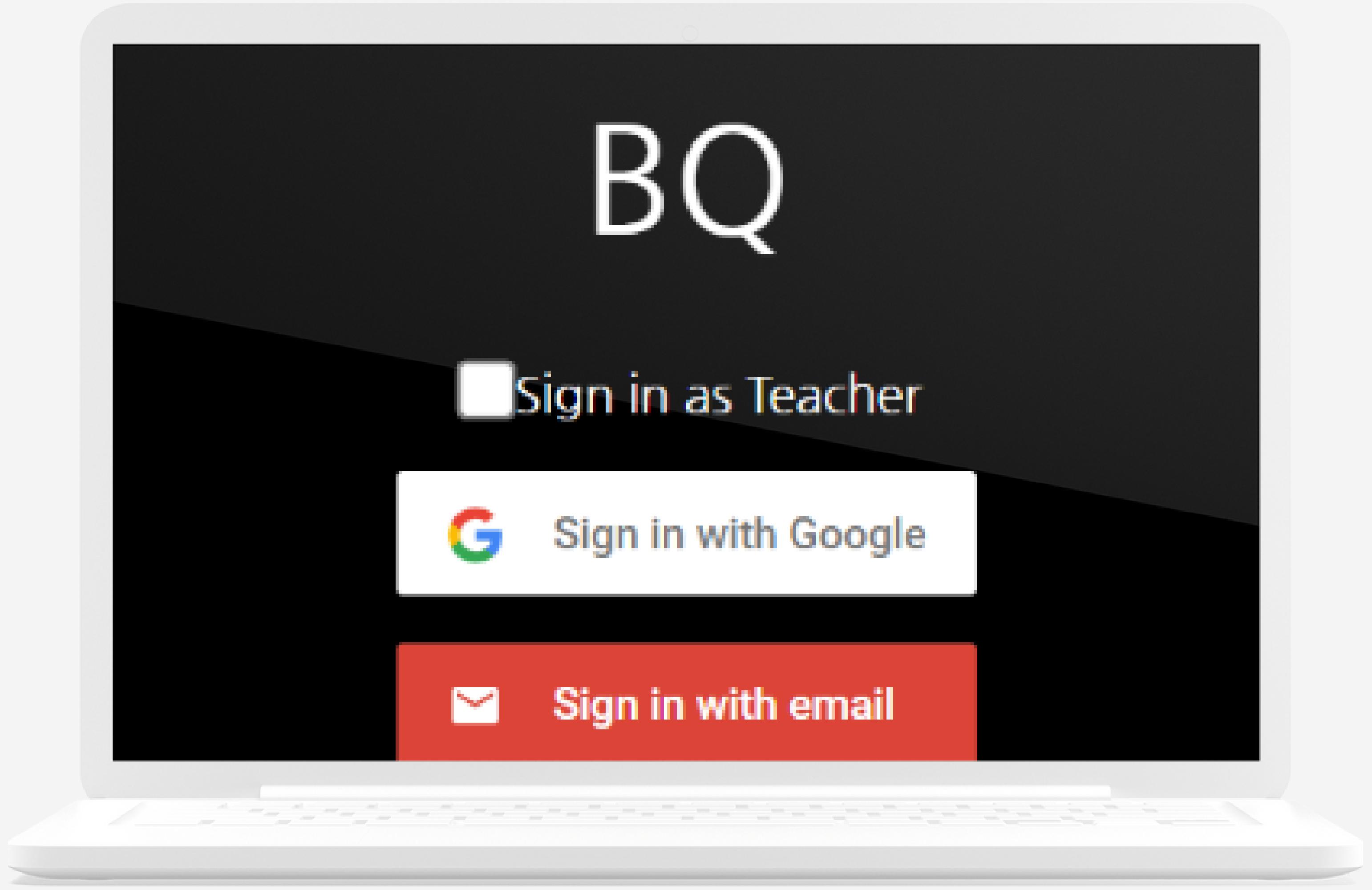


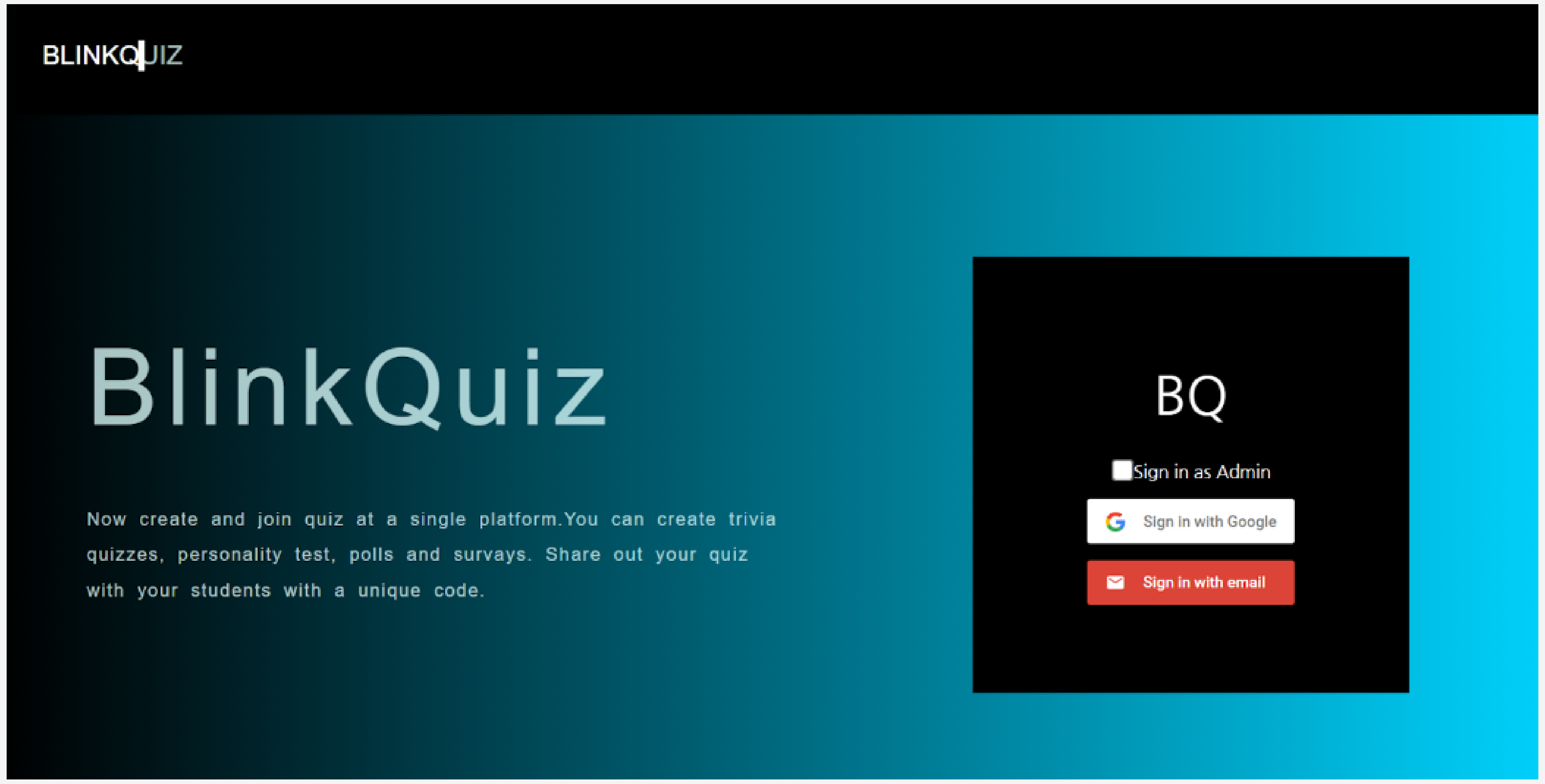
```
Router.get('/getUserRole/:uid', (req, res)=>{
    const uid = req.params.uid;
    if (!uid) return res.status(500).json({ error: 'Incomplete Parameters' });

    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)

})
```

# RESULT AND ANALYSIS

## LOGIN

- A small introduction of our project proceed with login page where the user can choose his/her role as a teacher or as a student.



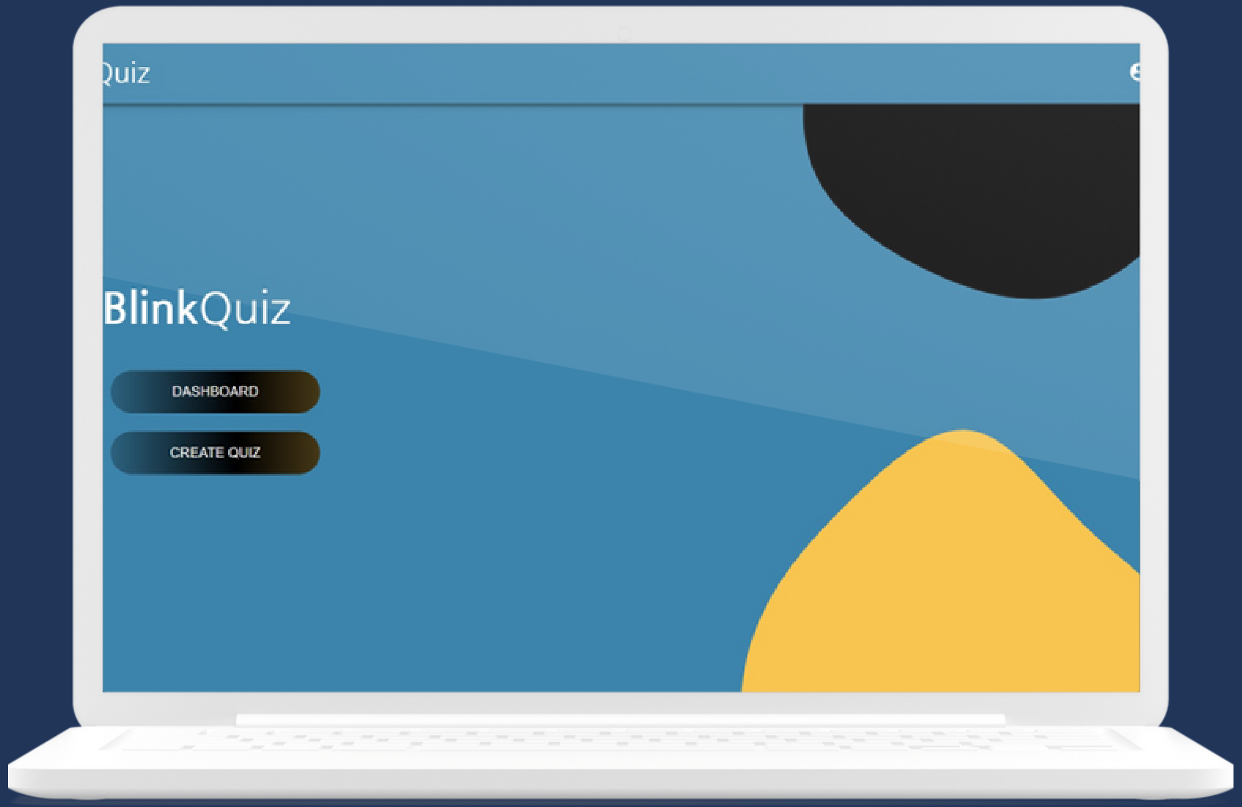


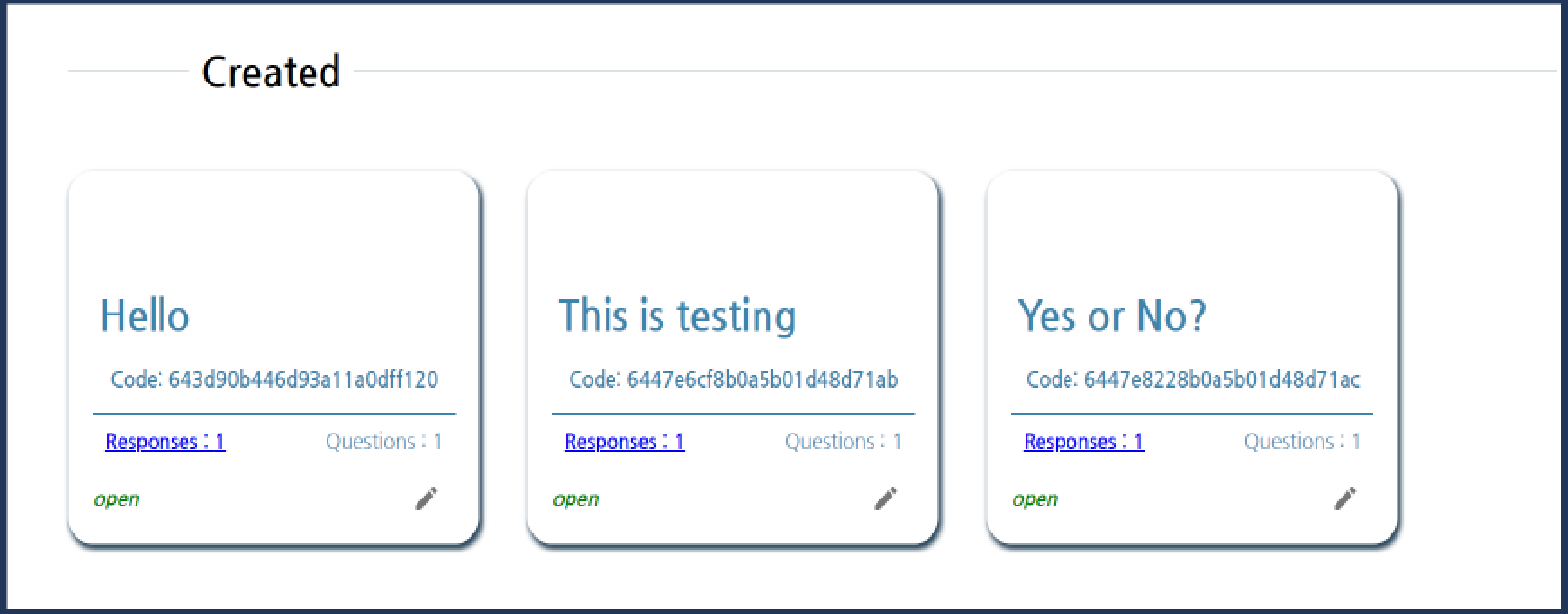
Now create and join quiz at a single platform.You can create trivia quizzes, personality test, polls and survays. Share out your quiz with your students with a unique code.

# RESULT AND ANALYSIS

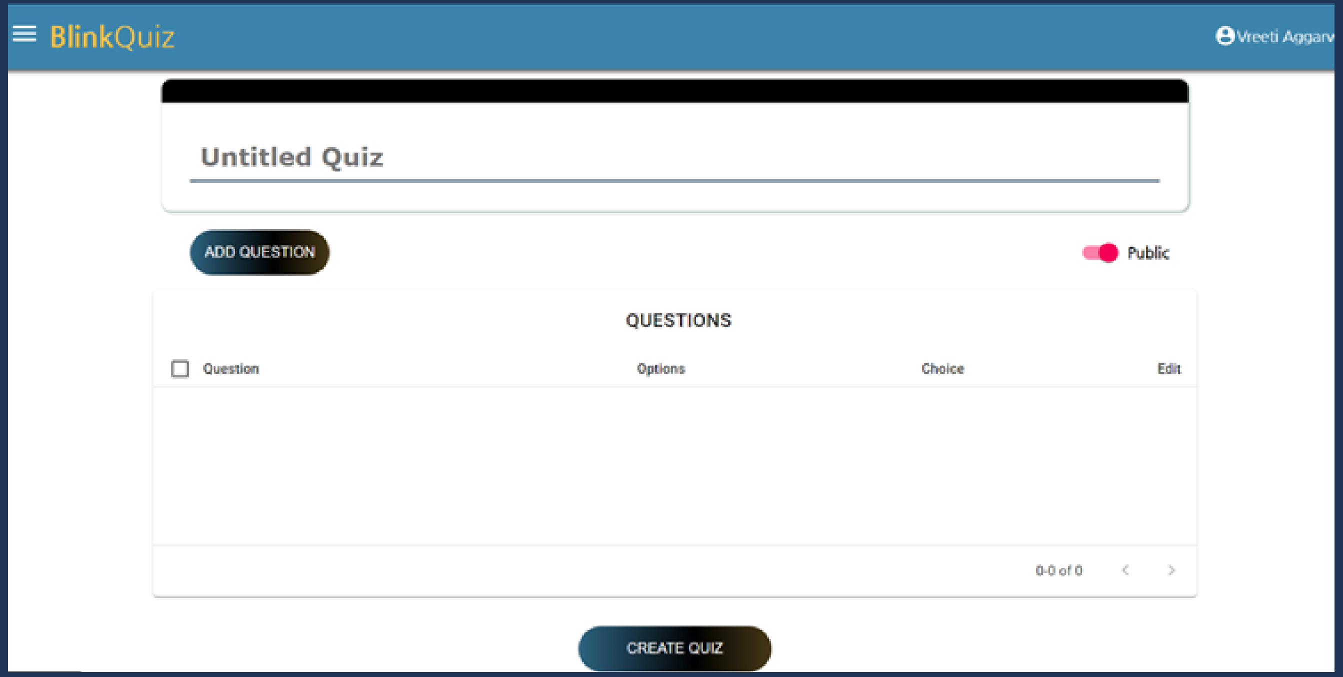
## TEACHER'S PORTAL

- A dashboard to see created quizzes.
- Teacher can create the quiz with as many questions and options he/she wishes to upload.
- An option cannot be repeated.
- Teacher can see the responses from the students.
- Teacher can close the quiz anytime.




Created

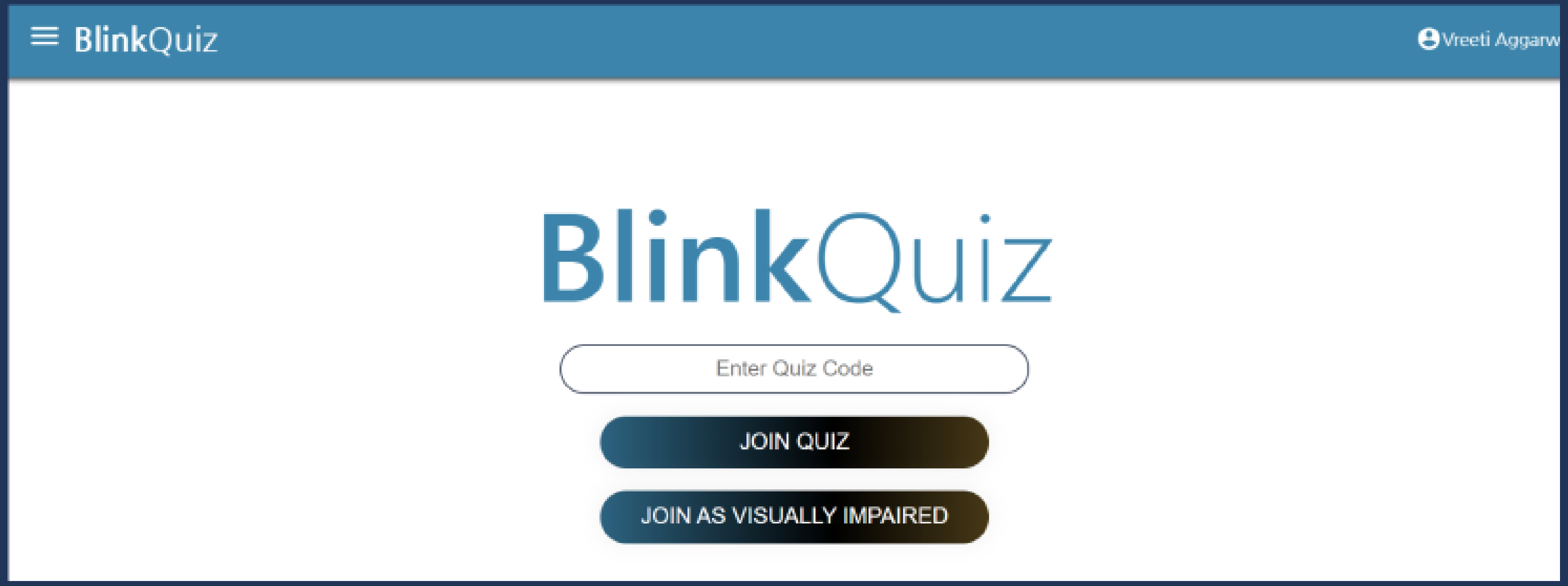
| | | |
|---|---|---|
| Hello | This is testing | Yes or No? |
| Code: 643d90b446d93a11a0dff120 | Code: 6447e6cf8b0a5b01d48d71ab | Code: 6447e8228b0a5b01d48d71ac |
| Responses : 1 Questions : 1 | Responses : 1 Questions : 1 | Responses : 1 Questions : 1 |
| open | open | open |

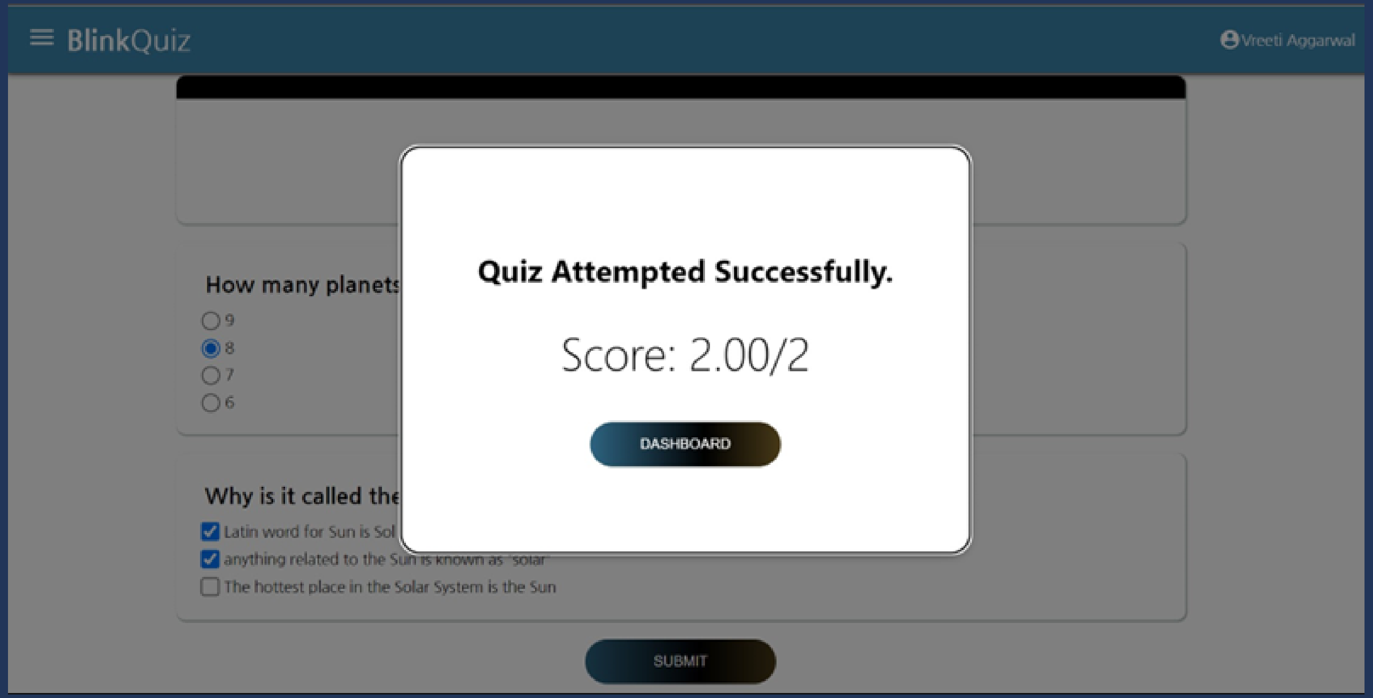


# RESULT AND ANALYSIS

## STUDENT'S PORTAL

- Students can join the quiz by clicking on the “Join As Visually Impaired”OR "JOIN QUIZ" button.
- A dashboard to see created quizzes.
- A quiz form with multiple choice questions.
- A modal which displays the score secured by the student.





Quiz Attempted Successfully.

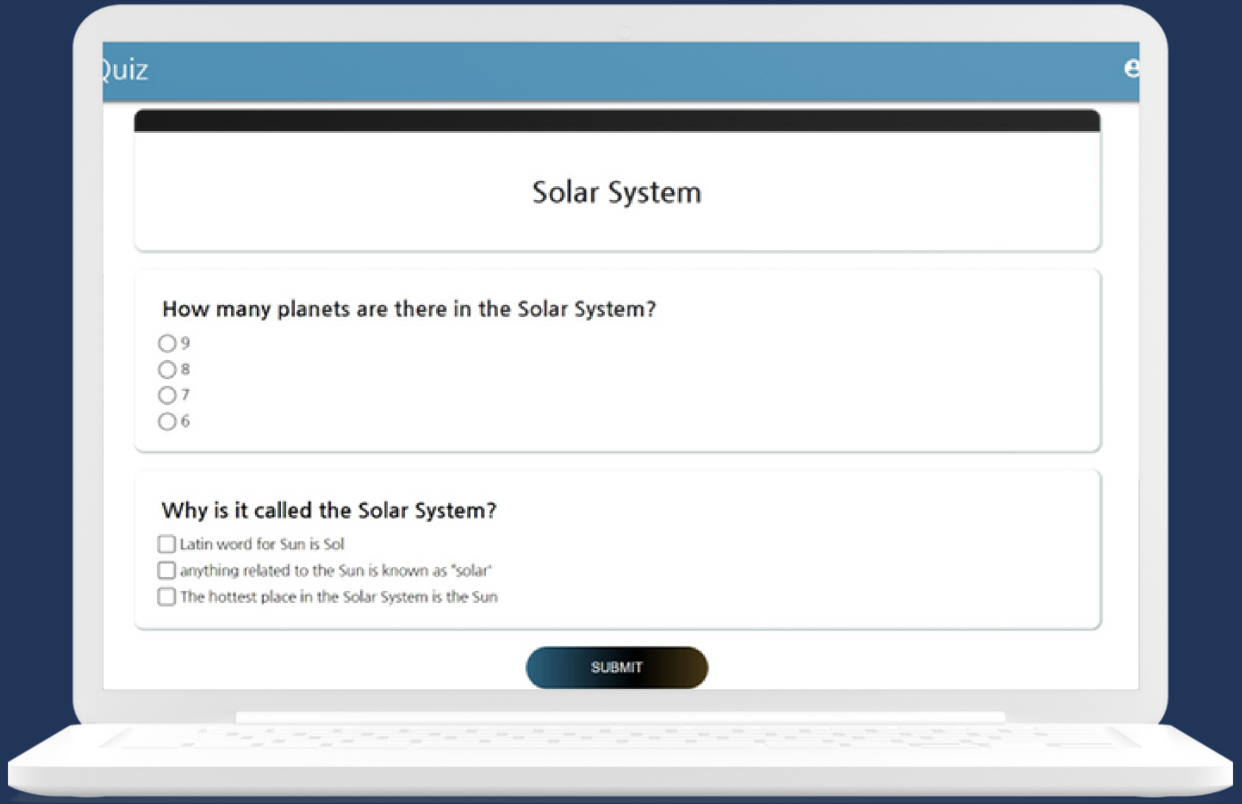
Score: 2.00/2

How many planets

- 9
- [✓ 8]
- 7
- 6

Why is it called the

- [✓ 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



Solar System

How many planets are there in the Solar System?

- 9
- 8
- 7
- 6

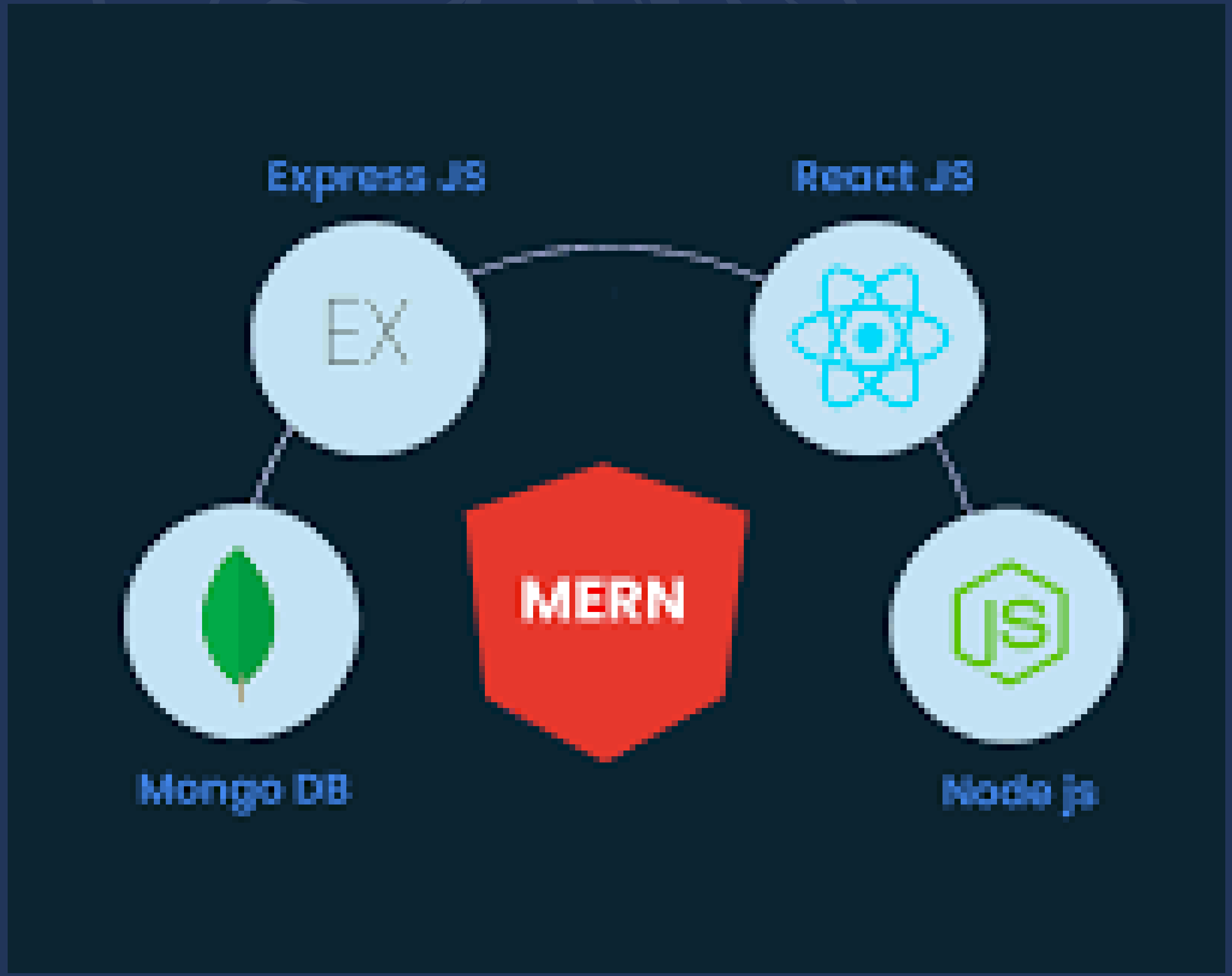
Why is it called the Solar System?

- 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

## VOICE 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.

## Conclusion

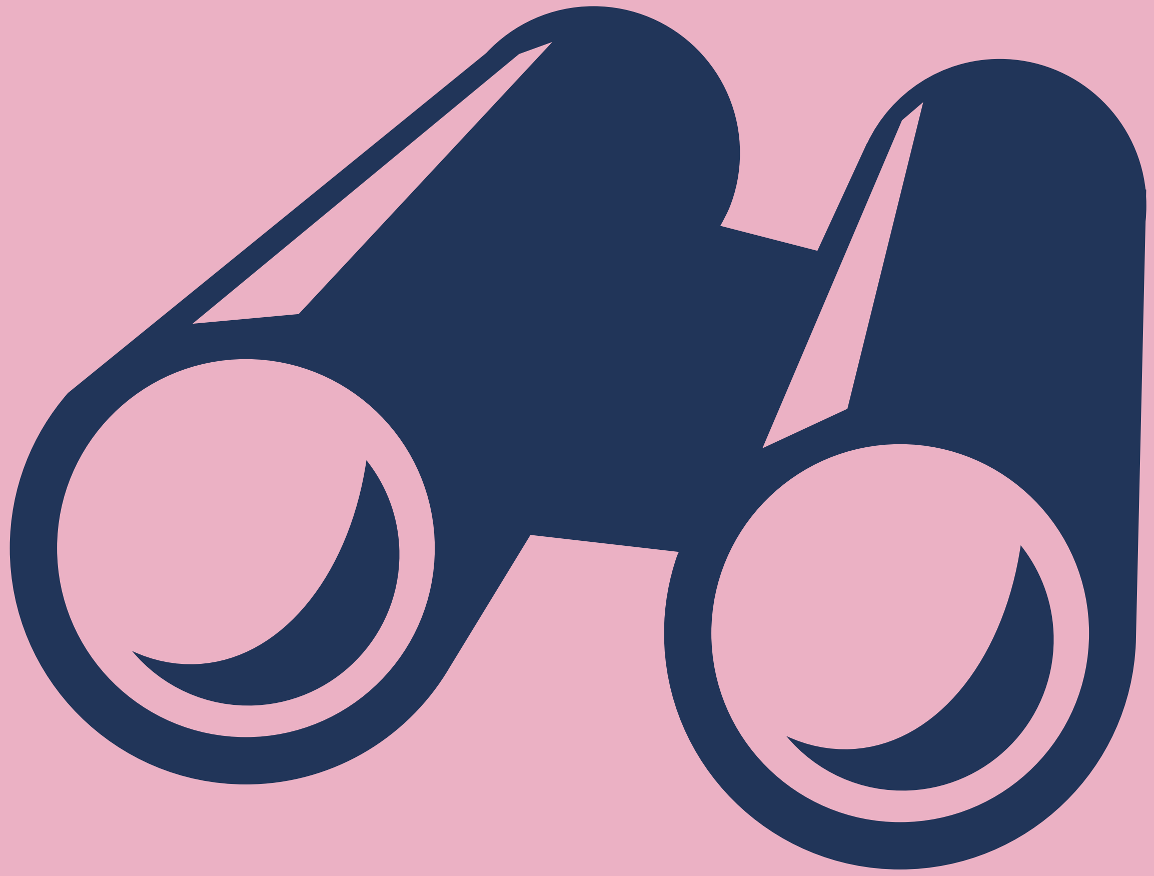


In conclusion, the development of a quiz application with speech recognition technology using the MERN stack provides a highly accessible and user-friendly experience for all users.

The integration of speech recognition technology into the quiz application makes it more inclusive and accessible for individuals who have disabilities or struggle with typing. 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.

## FUTURE SCOPE



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.