2020

QUIZ APPLICATION

Software Architecture Development

Submitted to: Prof.Christopher Hann,

Submitted By
Keerthana Kandaswamy,
Fatima Ahmad Dhool
Renida Dsouza
Sakshi Chaturvedi
Yazdan Zubin Banaji

SRH HEIDELBERG
9/2/2020

Acknowledgement:

We would like to express our deepest gratitude to our 'Professor Christopher Hahn' for his support and persistence throughout the module, and also for sharing his knowledge with us as well as helping us in every stage of the development of this project.

QUIZ

Teams Role & Responsibilities:

Matriculation Number	Name	Roles & Responsibilities
11013023	Keerthana Kandaswamy	Front-End Design &Development and API development
11013367	Fatima Ahmad Dhool	Front-End Design Development and Report documentation
11013285	Renida Dsouza	Database design & Implementation
11013316	Sakshi Chaturvedi	Database design & Implementation
11013621	Yazdan Zubin Banaji	Research & Presentation

Table of Contents

Project Title: Quiz Application

1. Introduction:

The quiz application is developed for educational purpose and dealing with multi-choice question learning system. The quiz applications make learning effective and add more fun. It also put a user in a situation where he/she can test or challenge their knowledge in a comfortable and responsive environment. This Quiz application consists of five different categories for instance General Knowledge, Sports, Animal, Geography and IT. User will be able to play any category quiz , after selection of desired quiz, questions will be appear on screen with multiple choices, user can see his/her progress feedback during the quiz and at the end of the quiz the application will display the result or total score.

1.1 Objective of the project:

- One of the main objectives of this project is the effective use of technologies and to be more familiar with its recent functionalities for instance, node.js, express node, JS, Bootstrap, CSS, Html, and API etc.
- To make learning fun and effective
- Provide user friendly learning environment which will reduce the manual effort

2. User & system requirement:

2.1 Functional requirement:

2.1.1 User Score management:

Application allows users to play and save their scores.

2.2 Non-Functional Requirement:

2.2.1 Performance:

The application performance should be optimized, and response time should be reduced.

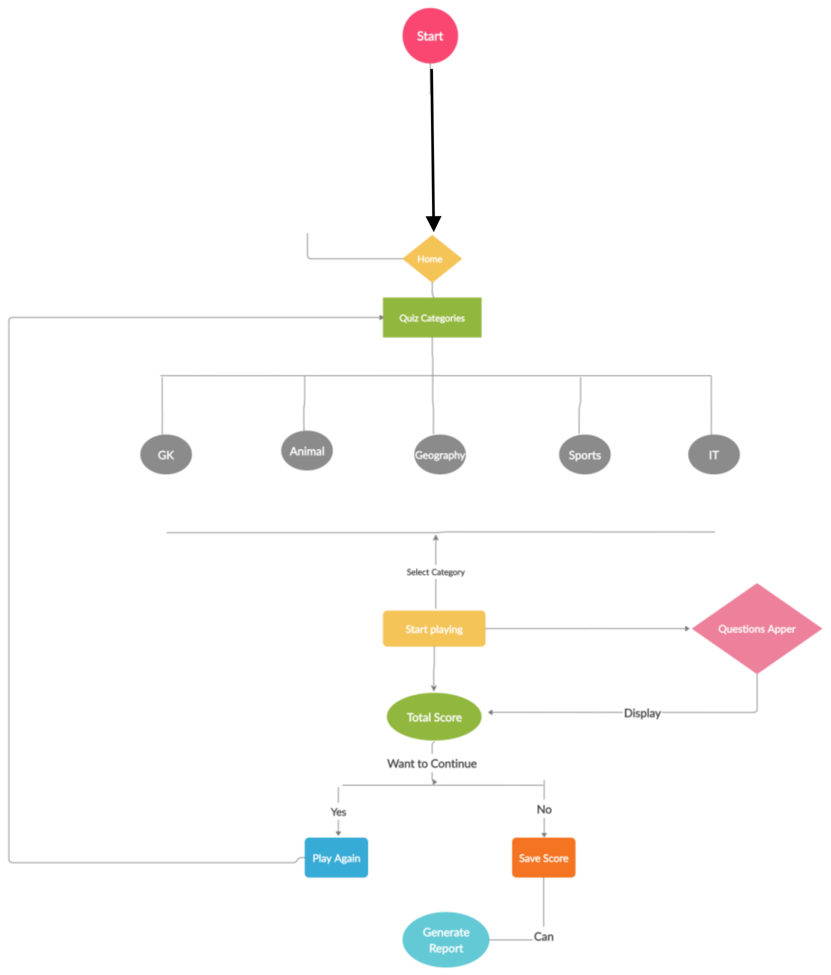
2.2.2 Availability:

Application will be available all the time, user only require internet to access account.

2.2.3 Maintainability:

The system will allow additional upgrade that can be implemented in the future.

3. Application Activity Flow Diagram:



3.1 Figure 02: Activity Diagram

3.2 Details about activity diagram:

Activity	Description
Home	User can play different quizzes.
Select Categories	There are different quiz categories like GK, sports, animal, Geography and IT. User can play any of them and enhance their knowledge.
Play quiz/Save results	User can play any selected category and save their score.

4. Technologies:

4.1 Client side:

- HTML
- CSS
- Bootstarp
- Javascript

4.2 Server side:

- MySQL: Use to create database to keep users information
- Node.js
- Express
- RestAPI

4.3 Other tools:

- **Postman:** Available as Chrome extension and used to handle API requests.

5. System Architecture Pattern and Design:

The Quiz Application system architecture is using MVC (Model-View-Controller) model which explains the concept of three ties architecture, server tier, client tier and data tier.

5.1 MVC- Model-Controller-View

5.1.1 Model:

- Model is a server tier which defines data structure and updates application to reflect selected or inserted data
- Update request result to the view so that user can see its request result
- Usually model contains business logic which helps to make use of a web-service
- Data tier stores in MYSQL database where all the user information and quiz score will be stored

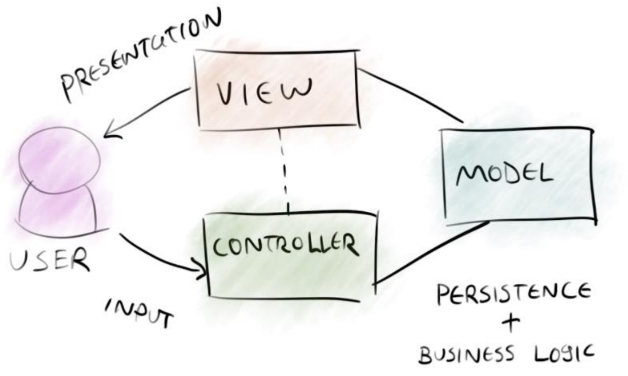
5.1.2 View:

- Defines display (UI) like user can select any quiz category and play
- It is also responsible for taking inputs from users and sends it to controller
- Provide content to the user and pass on users requests to the controller
- For this quiz app UI is designed by using HTML code &forms, CSS, Bootstrap and JavaScript

5.1.3 Controller:

- It contains control logics, receives update from view then notifies model to select something or add data etc.
- Then make some changes and send the request to the model
- Sometimes controller updates directly to the view
- Simply controller interacts with the model to generate data for the view (display)

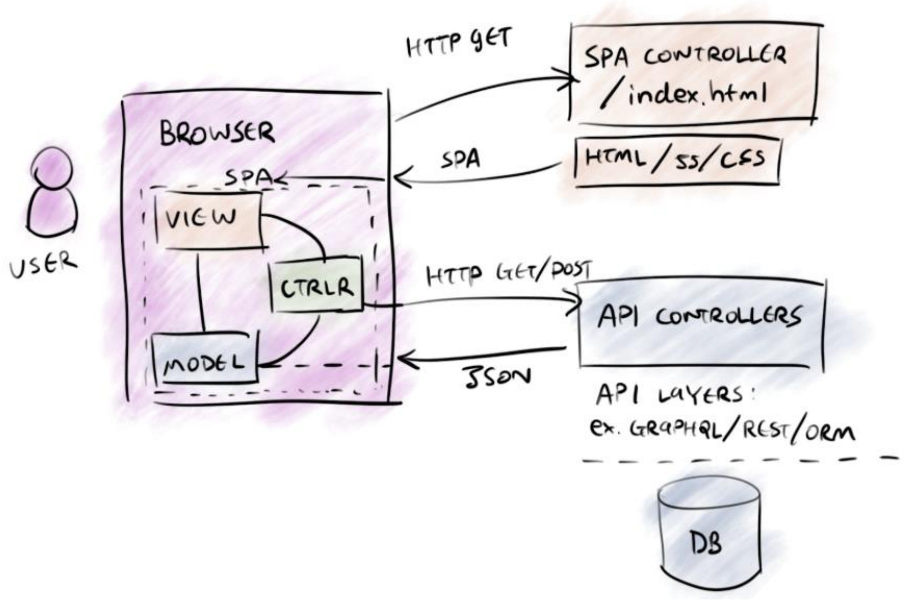
5.1.4 Traditional MVC Model:



5.1.4.1 Figure 03: MVC

5.1.5 Single page application (SPA) MVC:

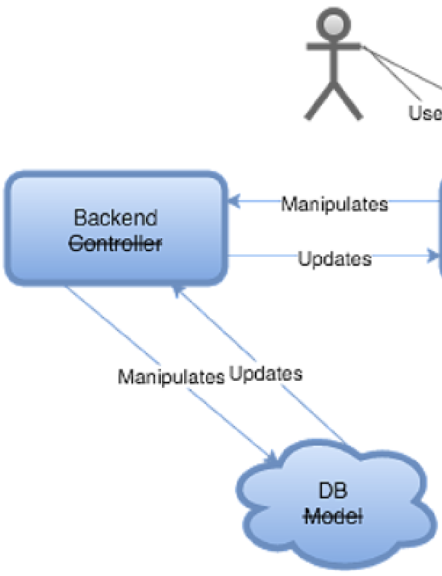
- **API:** Rest and fetch Api
- **Protocol:** HTTP
- **Web-services:** Chrome browser



5.1.5.1 Figure: 04 SPA MVC

5.1.6 Updated MVC model:

It's a known fact that MVC (Model-View-Controller) is made up for full stack development, where the application or software consists of UI (user interface) 'View', Business logics 'Controller' and database 'Model'. Most of the full stack applications are using MVC but now it's not the same as it was once because of new terms and strategies for example view use as client side or mobile display, controller as backend while the model has seen as Database, as we can see in the following diagram.



5.1.6.1 Figure: 05 Updated view of MVC

Basically backend made up to fulfill certain objectives like;

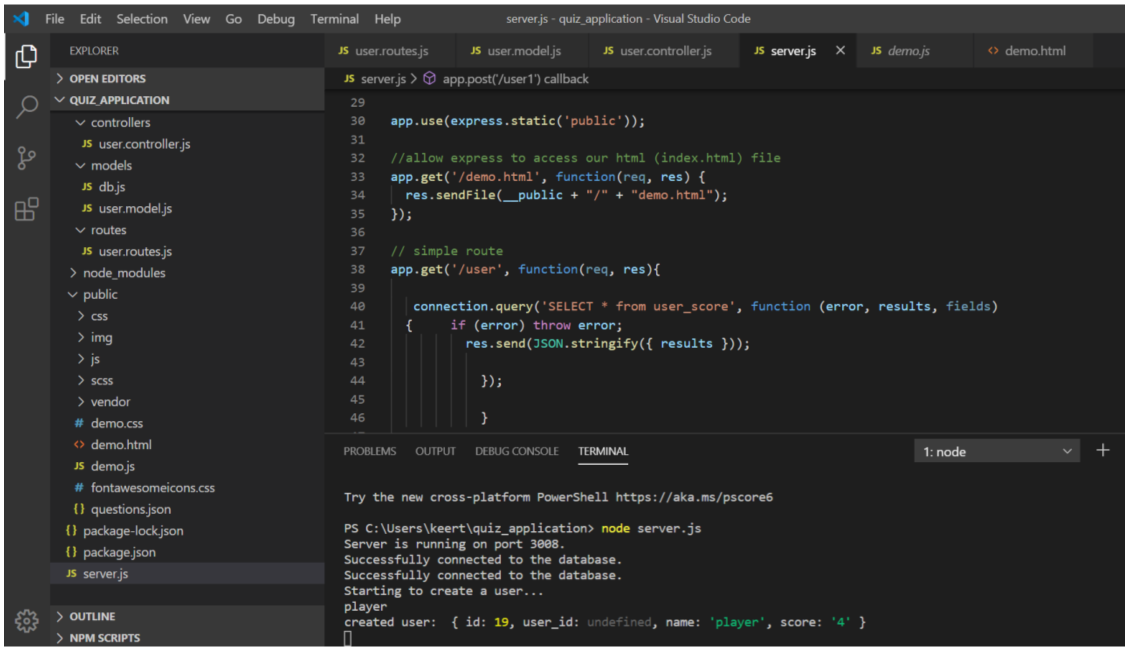
1. GET request to search some data
2. POST data in Database
3. PUT data in database
4. Delete data from database
5. Convert data into required or right format
6. Execute or return demanded data

5.2 API Development:

The API programs are available under the 'app' folder in the quiz application.

5.2.1 GET Function:

In our application, the GET function is used to connect the UI and database to read the details of all the players and their scores.

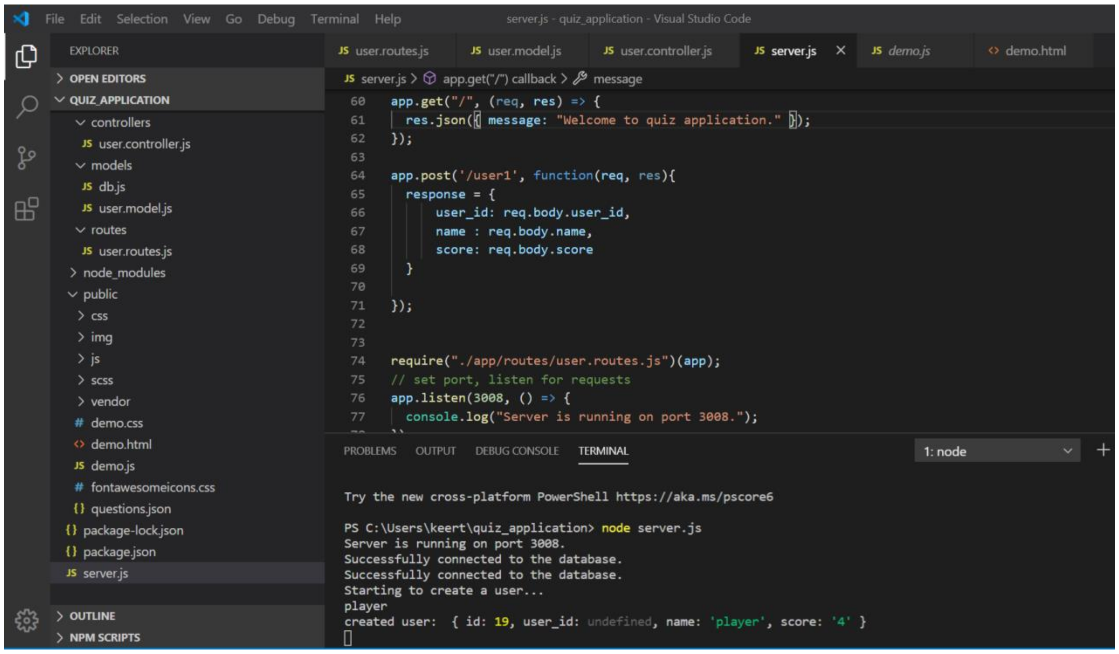

File Edit Selection View Go Debug Terminal Help
server.js - quiz_application - Visual Studio Code
user.routes.js
user.model.js
user.controller.js
server.js
demo.js
demo.html
server.js > app.post('/user1') callback
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: node

```
app.use(express.static('public'));

//allow express to access our html (index.html) file
app.get('/demo.html', function(req, res) {
  res.sendFile(__public + "/" + "demo.html");
});

// simple route
app.get('/user', function(req, res){

   connection.query('SELECT * from user_score', function (error, results, fields)
  {     if (error) throw error;
          res.send(JSON.stringify({ results }));

            });

            }
```

```
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\keert\quiz_application> node server.js
Server is running on port 3008.
Successfully connected to the database.
Successfully connected to the database.
Starting to create a user...
player
created user:  { id: 19, user_id: undefined, name: 'player', score: '4' }
```

5.2.2 Post Function:

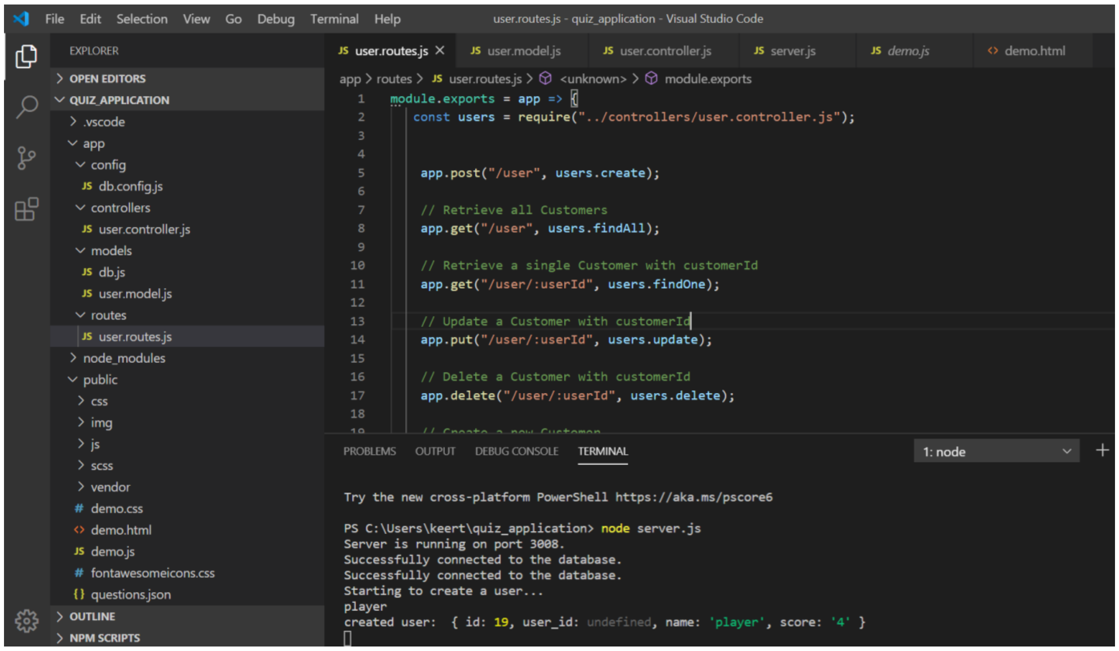
The main role of the post function is to send the user details and scores to the database in order to save the player details.


File Edit Selection View Go Debug Terminal Help
server.js - quiz_application - Visual Studio Code
user.routes.js
user.model.js
user.controller.js
server.js
demo.js
demo.html
server.js > app.get("/") callback > message
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: node

```
app.get("/", (req, res) => {
  res.json({ message: "Welcome to quiz application." });
});

app.post('/user1', function(req, res){
  response = {
      user_id: req.body.user_id,
      name : req.body.name,
      score: req.body.score
  }

});


require("./app/routes/user.routes.js")(app);
// set port, listen for requests
app.listen(3008, () => {
  console.log("Server is running on port 3008.");
```

```
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\keert\quiz_application> node server.js
Server is running on port 3008.
Successfully connected to the database.
Successfully connected to the database.
Starting to create a user...
player
created user:  { id: 19, user_id: undefined, name: 'player', score: '4' }
```

5.2.3 Routes:

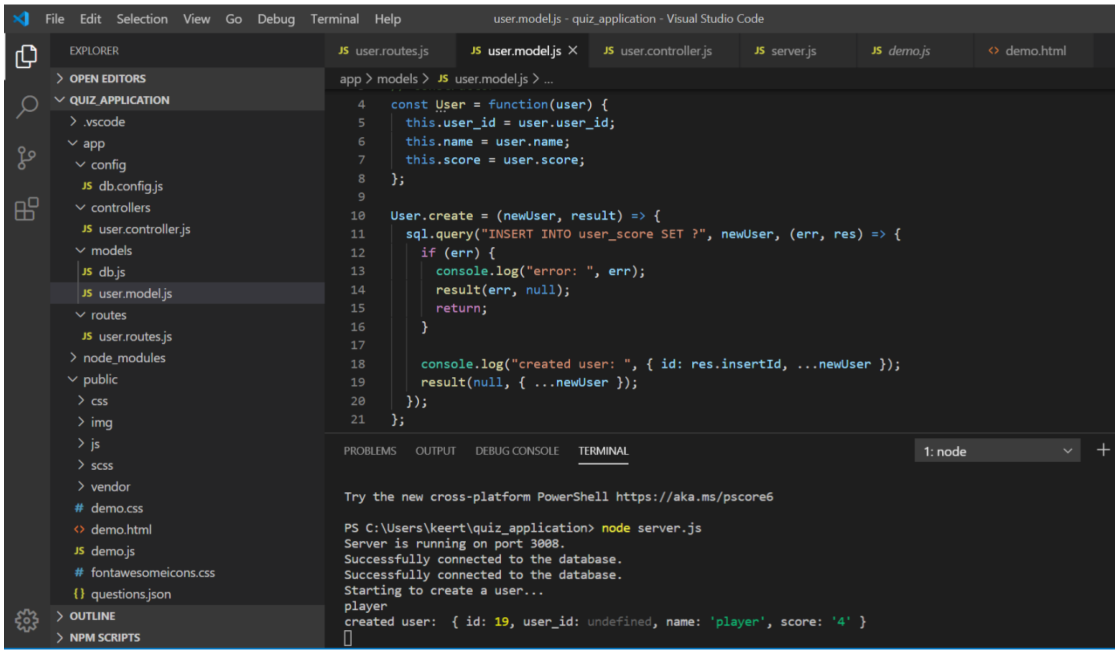


```
module.exports = app => {
  const users = require("../controllers/user.controller.js");


   app.post("/user", users.create);

   // Retrieve all Customers
   app.get("/user", users.findAll);

   // Retrieve a single Customer with customerId
   app.get("/user/:userId", users.findOne);

   // Update a Customer with customerId
   app.put("/user/:userId", users.update);

   // Delete a Customer with customerId
   app.delete("/user/:userId", users.delete);

   // Create a new Customer
```

```
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\keert\quiz_application> node server.js
Server is running on port 3008.
Successfully connected to the database.
Successfully connected to the database.
Starting to create a user...
player
created user:  { id: 19, user_id: undefined, name: 'player', score: '4' }
```

5.2.4 Model:

The model holds all the logic for functions such as create, read, update and delete.

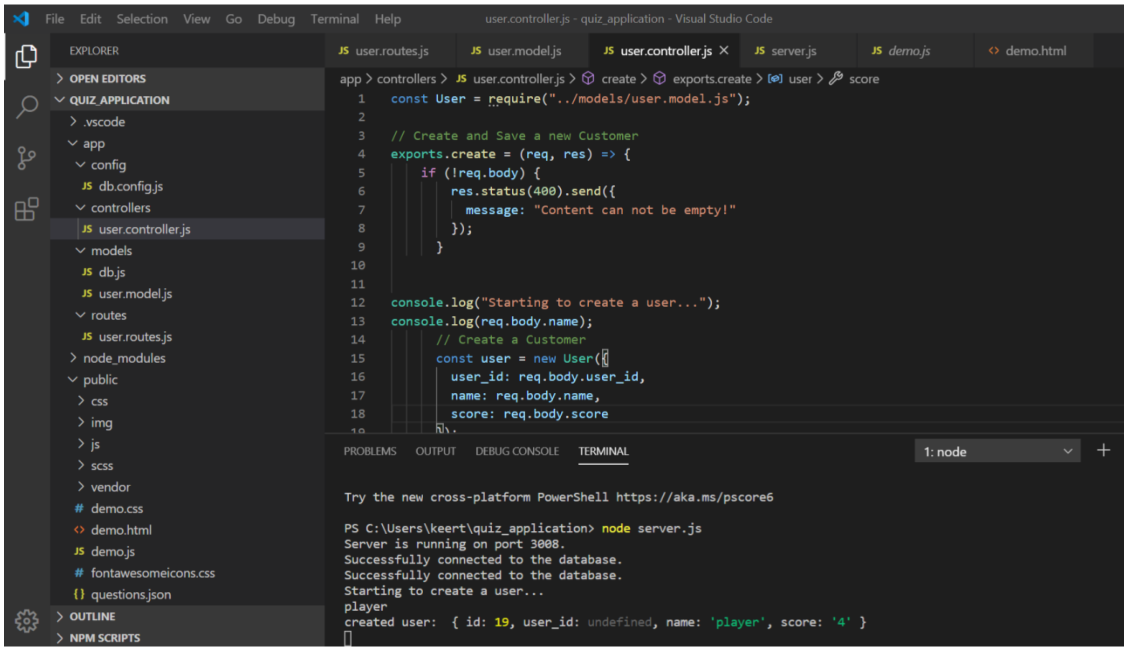


```
const User = function(user) {
  this.user_id = user.user_id;
  this.name = user.name;
  this.score = user.score;
};

User.create = (newUser, result) => {
  sql.query("INSERT INTO user_score SET ?", newUser, (err, res) => {
    if (err) {
      console.log("error: ", err);
      result(err, null);
      return;
    }

    console.log("created user: ", { id: res.insertId, ...newUser });
    result(null, { ...newUser });
  });
};
```

```
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\keert\quiz_application> node server.js
Server is running on port 3008.
Successfully connected to the database.
Successfully connected to the database.
Starting to create a user...
player
created user:  { id: 19, user_id: undefined, name: 'player', score: '4' }
```

5.2.5 Controller:

The controller holds the mapping for the UI and the database and necessary error handling.



```
const User = require("../models/user.model.js");

// Create and Save a new Customer
exports.create = (req, res) => {
    if (!req.body) {
        res.status(400).send({
          message: "Content can not be empty!"
        });
      }


console.log("Starting to create a user...");
console.log(req.body.name);
      // Create a Customer
      const user = new User({
        user_id: req.body.user_id,
        name: req.body.name,
        score: req.body.score
```

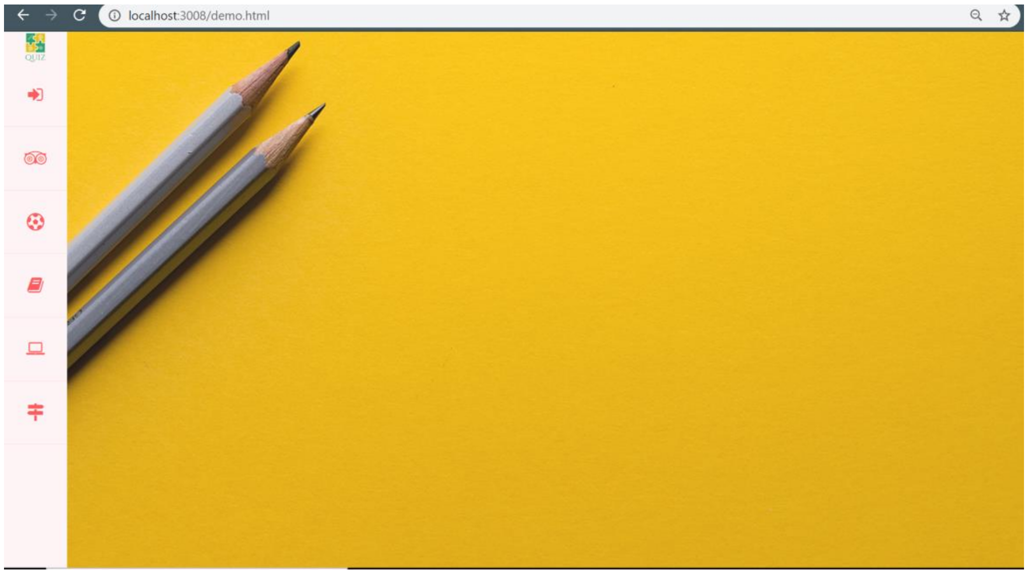
```
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\keert\quiz_application> node server.js
Server is running on port 3008.
Successfully connected to the database.
Successfully connected to the database.
Starting to create a user...
player
created user:  { id: 19, user_id: undefined, name: 'player', score: '4' }
```

6. Results:

6.1 Front-end Development:

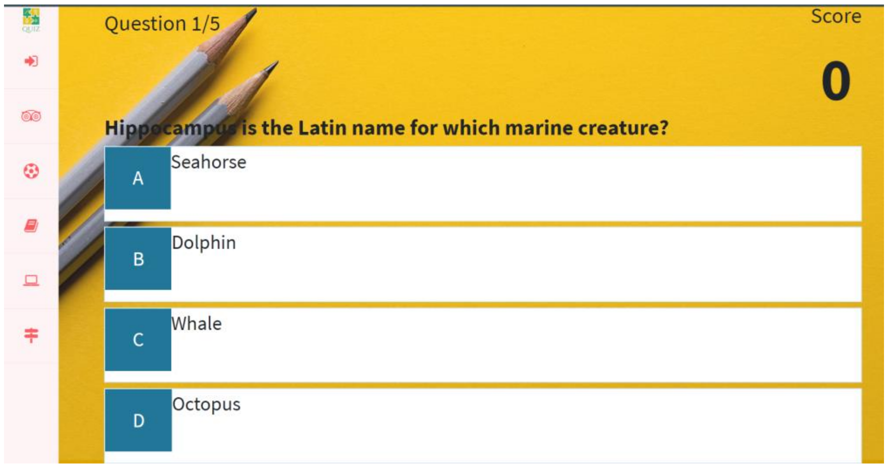
6.1.1 Home page:

Here user can select any category he/she wants to play, for instance Animal, Sports, General Knowlwdge, IT and Geograpgy etc.



6.1.2 Quiz Category 01:

Once the user selects a category and starts playing the game, the user can select one answer out of four different options. The score gets incremented by one point for every correct answer.



Question 1/5

Score

0

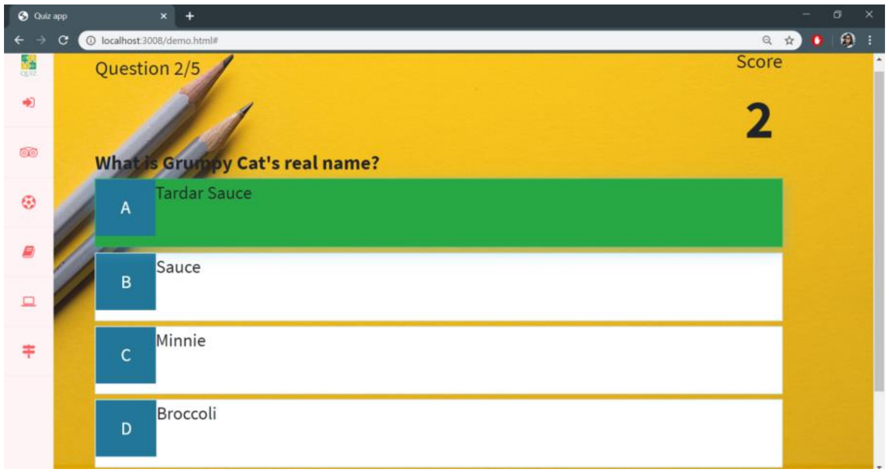
Hippocampus is the Latin name for which marine creature?

A Seahorse

B Dolphin

C Whale

D Octopus



Question 2/5

Score

2

What is Grumpy Cat's real name?

A Tardar Sauce

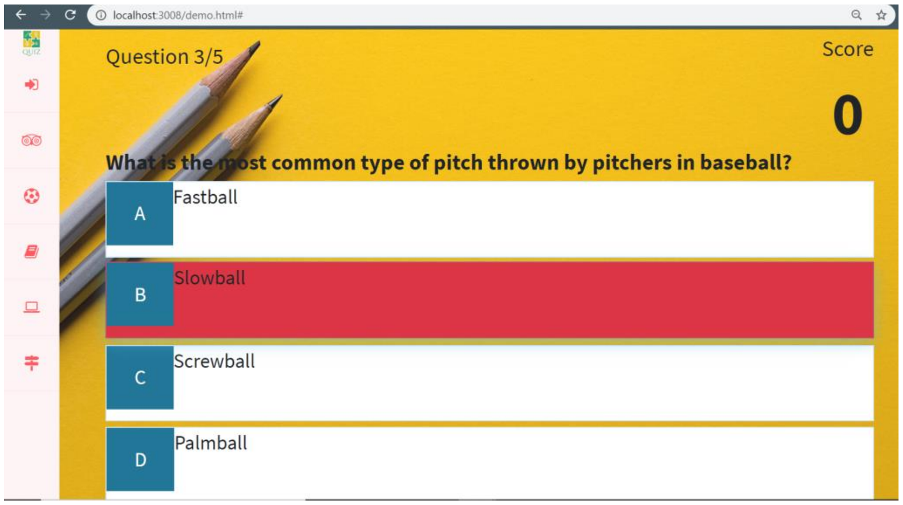
B Sauce

C Minnie

D Broccoli

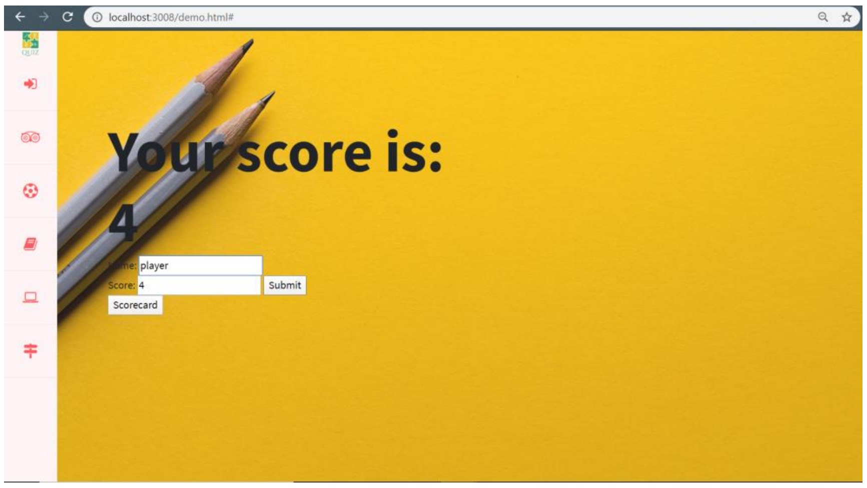
6.1.3 Quiz Category 02:

When the user selects a wrong answer, his score is zero but the application will still hold the total score of the game.


localhost:3008/demo.html#
Question 3/5
Score
0
What is the most common type of pitch thrown by pitchers in baseball?
A
Fastball
B
Slowball
C
Screwball
D
Palmball

6.1.4 Final Score:

Once the user answers his last question, the final score of the user is displayed in the UI. To store the score in the database, the user has to give his name and press the submit button. These details are then sent to the API for further processing. With the help of 'Scorecard' button the user can also get the score of all the players and their scores.

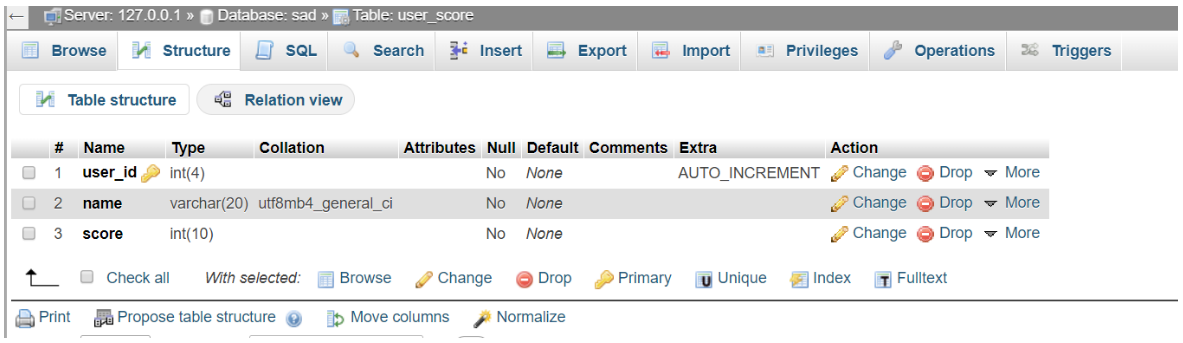


6.2 Database design and Development (Backend):

MySQL database is used in this application to store the user name and score. A table 'user_score' is created which stores the data sent from the UI.

6.2.1 User-Score:

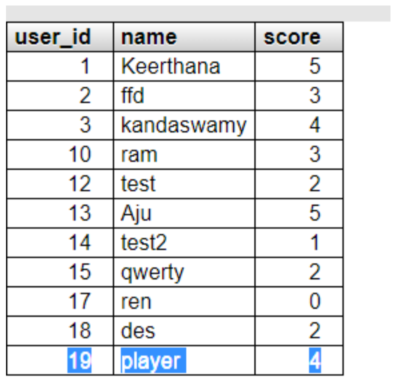
Once the user answers his last question in the game and clicks the submit button, the score and user name is posted to the database with the help of middleware. The ''user_id" field is the primary key of the table which gets automatically incremented for every new game.



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	user_id	int(4)			No	*None*		AUTO_INCREMENT	Change Drop More
2	name	varchar(20)	utf8mb4_general_ci		No	*None*			Change Drop More
3	score	int(10)			No	*None*			Change Drop More

6.2.2 Players Final Score:

The final score can be displayed in JSON and also as a table. Once the user clicks the 'scorecard' button after answering the last question in the game, the score and name details of all players are fetched and displayed in the UI. Below is a sample of the fetched data from the MYSQL database. The most recent player details which are inserted in the last line can be seen in the diagram.



user_id	name	score
1	Keerthana	5
2	ffd	3
3	kandaswamy	4
10	ram	3
12	test	2
13	Aju	5
14	test2	1
15	qwerty	2
17	ren	0
18	des	2
19	player	4

The designed quiz application can be enhanced further in future with more functionality such as calculation of the highest scores and session implementation.