Author: Pranav, Shaheryar, Judah

Date: April 14, 2023

Topic: Architecture and Code Design

For the individual features, each team member assigned to the issue will be responsible for the documentation of their code/work. But we need to come up with the high-level documentation of the tools and design to be used in our project. This will evolve throughout the project, as we include more core features, and possibly requested features, but for now we need something.

The project architecture and high-level code design should be described, forcing the team to discuss how each code task will be integrated into the project design and allowing each team member to know what to expect from other team member's code. This architecture should be addressed when code task assignments are deliberated.

Elements of this documentation should include:

- the module composition of the project
- the interface and semantics of each module
- how each code task relates to the architecture
- user interface/web mockups or drawings (possibly using a tool such as figma)

Frontend Architecture & Code Design

The following is the module composition of the frontend:

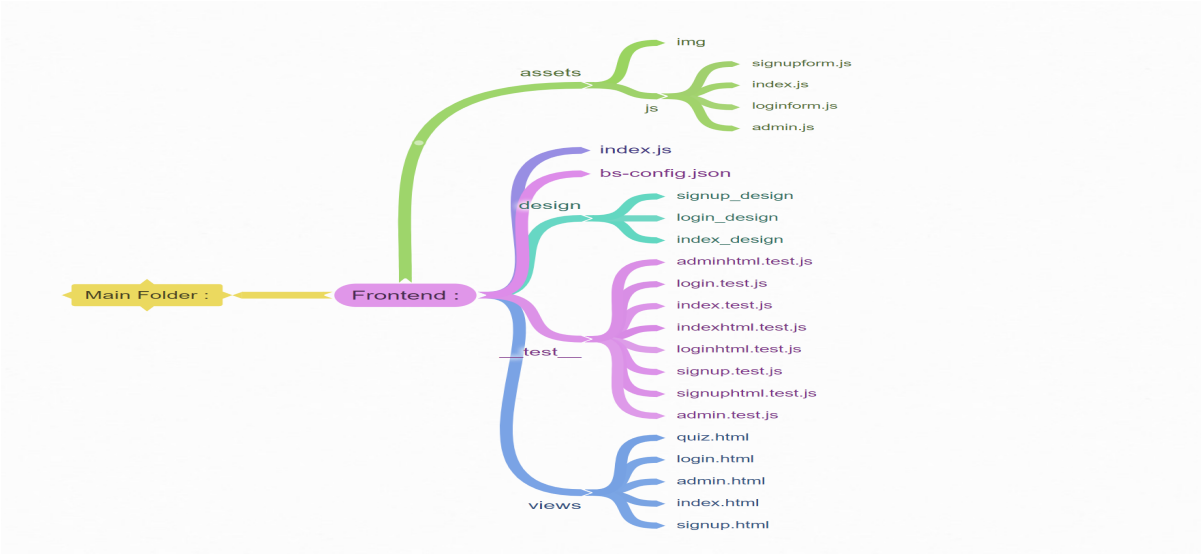


Figure 1: Frontend Folder Structure

Assignee for Frontend code tasks (module wise):

Pranav Arora (pranavarora1895)

- Previous Sprint Tasks
    - admin.html
- New Code Tasks
    - adminhtml.test.js

Balsher Singh (balshersingh10)

- New Code Tasks
    - admin.js
    - admin.test.js

Frontend Architecture and Execution Flow

The execution workflow tells how each module fits into the architecture.

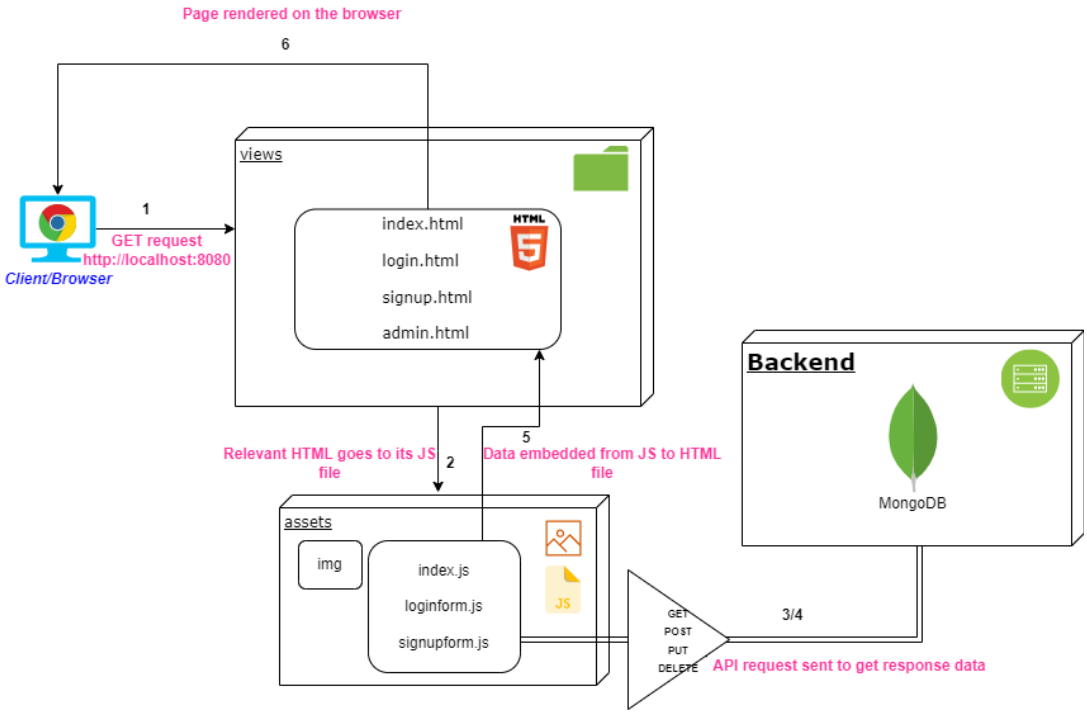


Figure 2: Execution Workflow of Frontend Architecture

Architecture Description:

1. Client opens up the browser and enters GET Request to http://localhost:8080
2. index.html comes up by default and calls its JS file index.js.
3. JS file will send the following API requests to the backend:
    a. index.js - GET to quiz API

    b. signupform.js - POST to user API
    c. loginform.js - GET to user API
4. After authentication (JWT) from the backend, the data will be received as a response.
5. The response data will be embedded to the HTML file.
6. The final HTML file will be rendered on the browser/client.

Frontend Architecture & Code Design for Assignment 2

The following is the module composition of the frontend:

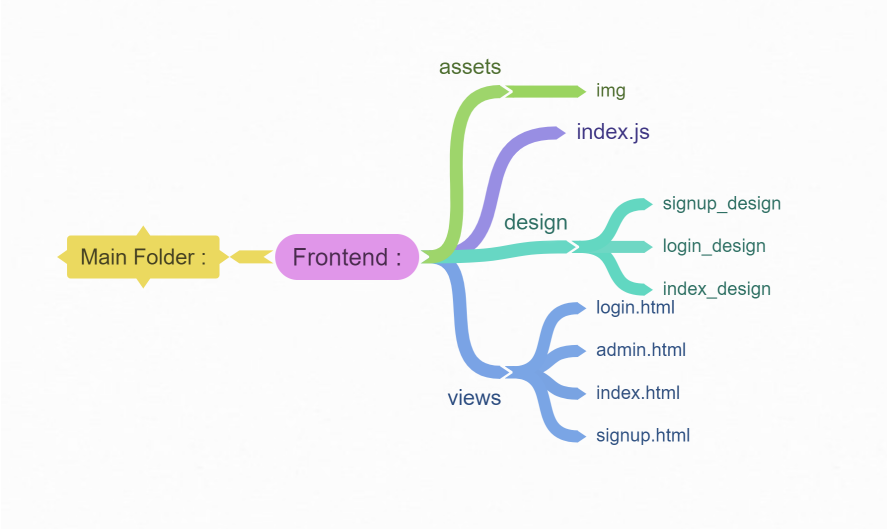


Figure 3: Frontend Folder Structure

Assignee for Frontend code tasks (module wise):

Pranav Arora (pranavarora1895)
- index.js
- index.html

Balsher Singh (balshersingh10)
- login.html
- signup.html

User Interface Drawings and Designs

You can view all the UI designs/drawings at the given repo link:
https://github.com/MUN-COMP6905/project-cteam/tree/master/frontend/design

Frontend Architecture and Execution Flow

The execution workflow tells how each module fits into the architecture.

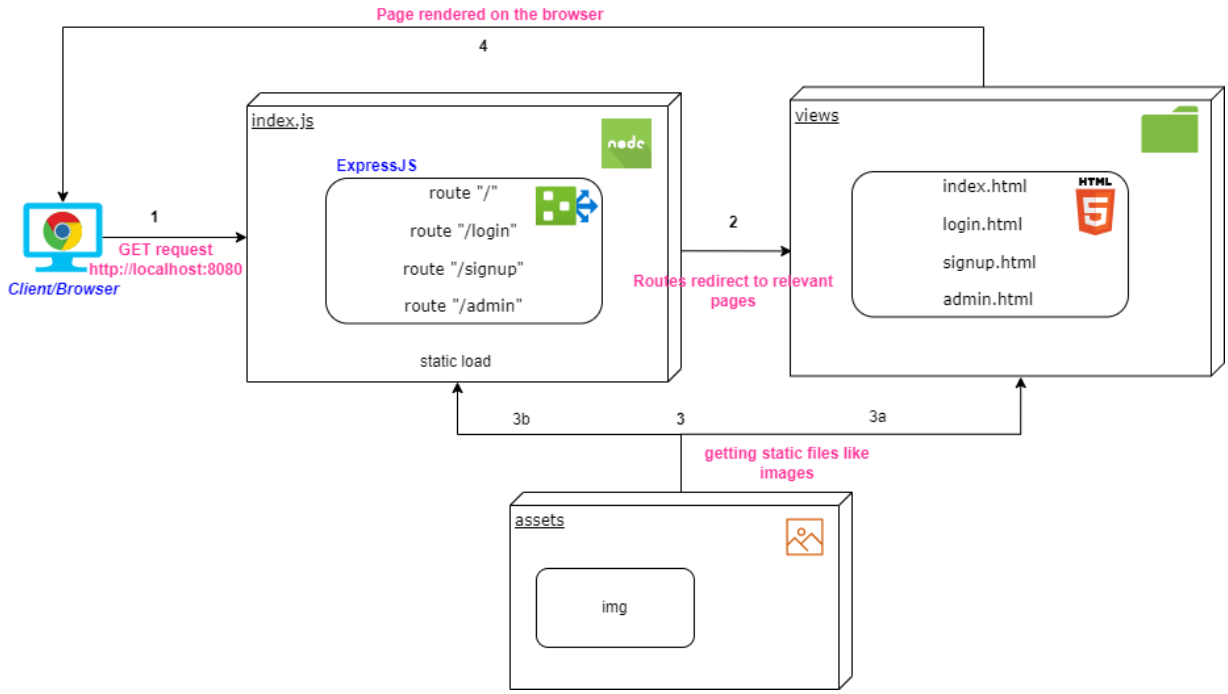


Figure 4: Execution Workflow of Frontend Architecture

Description

The following diagram shows the execution workflow of the frontend code. The steps will guide the execution path:

1. The client sends the GET request to http://localhost:8080.
2. index.js receives the request and checks the requested route. If the route matches with the route list mentioned in the file, it will return the connected HTML template in the views folder. If the request does not match, it will throw an error.

```
/**
* route: /
* method: GET
*/
app.get("/", (req,res) => {
                                // ...
}

/**
* route: /login
* method: GET
*/
app.get("/login", (req,res) => {
                                // ...
}

/**
* route: /signup
* method: GET
*/
app.get("/signup", (req,res) => {
                                // ...
}

/**
* route: /admin
* method: GET
*/
app.get("/admin", (req,res) => {
                                // ...
}
```

Route list in index.js

3. a. All the HTML templates will receive the images from the img folder under assets folder.
   b. To load the static files, expressJS needs to know the static file location. So it will be mentioned in index.js
4. The HTML template will be rendered on the client/browser.

Attributions

- Figure 1,3: coggle.it
- Figure 2,4: diagrams.net

Backend Architecture & Code Design

The following is the module composition of the backend:

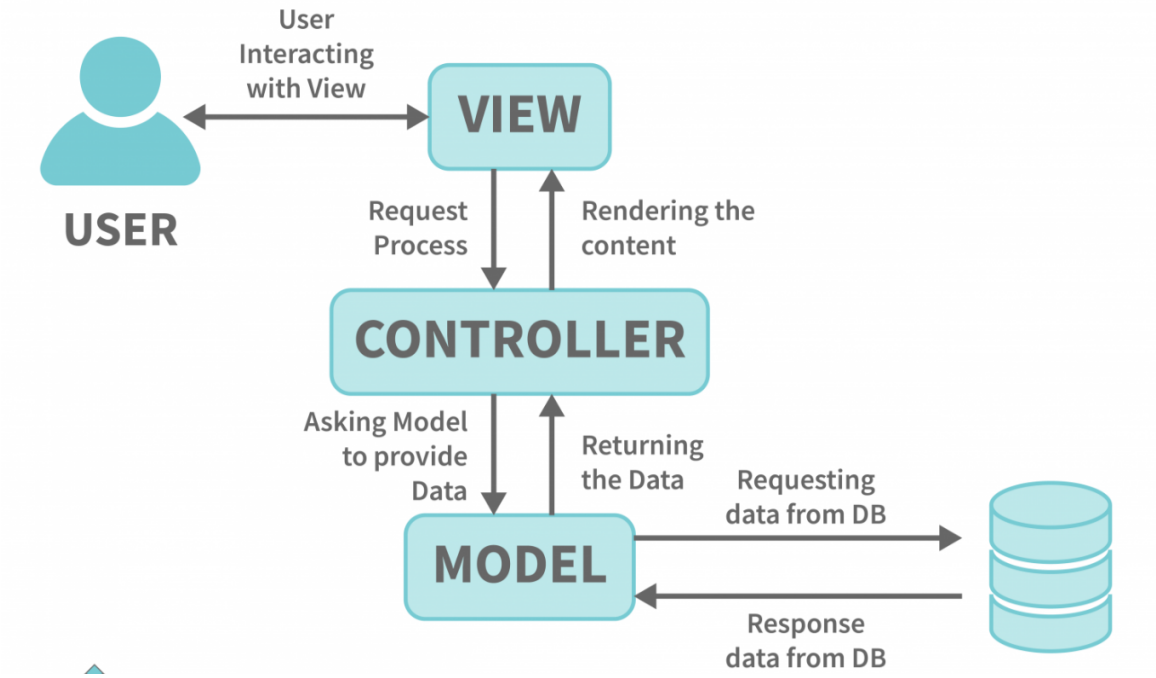


Figure 3: Backend MVC Structure

Why did we choose this framework and architecture as an alternative?

The Model-View-Controller (MVC) architecture is a popular design pattern used in many web application frameworks, including ExpressJS. The main reason for choosing MVC architecture in ExpressJS is to separate concerns and provide a clear organization for the code.

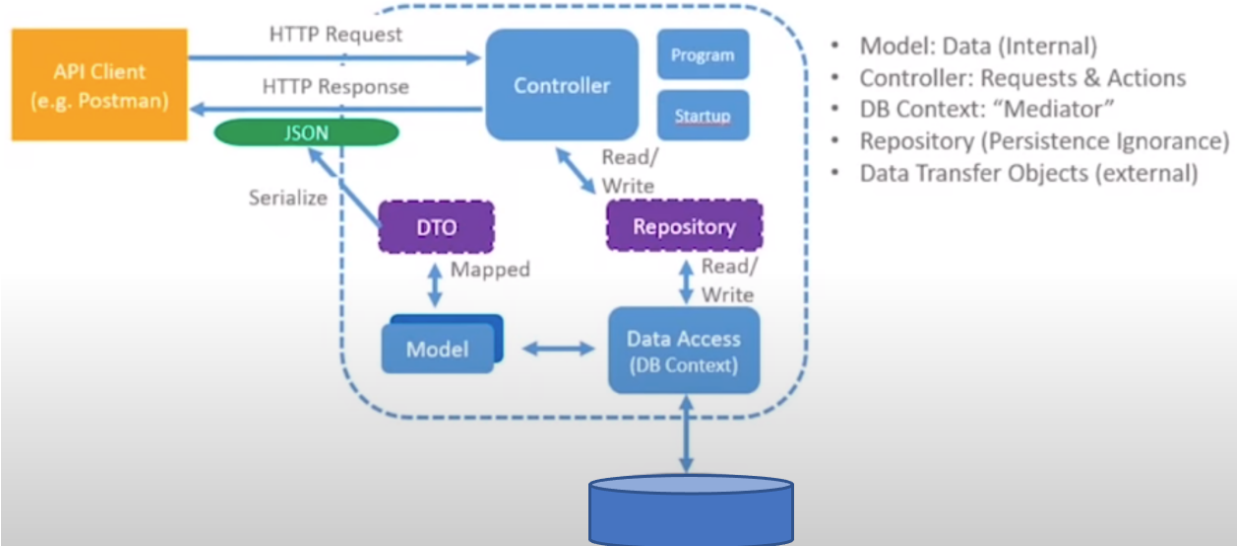
In an MVC architecture, the application is divided into three main components:

Model: The model represents the data and the business logic of the application. It is responsible for handling the data storage and retrieval, as well as performing any necessary computations on the data.

View: The view is responsible for rendering the data to the user interface. It takes the data from the model and presents it to the user in a format that is easy to understand and interact with.

Controller: The controller acts as an intermediary between the model and the view. It receives input from the user and uses it to manipulate the model data. It also updates the view with any changes to the data. By separating the application into these three components, it becomes easier to maintain and modify the codebase. Changes to the model do not affect the view or the controller, and vice versa. This makes it easier to add new features, fix bugs, and improve the overall quality of the application.

QuizApp Architecture and APIs workFlow



Assignee for Backend code tasks:

Muhammad Shaheryar (Muhammad-Shaheryar)

- Basic code structure (server setup)
- Authentication
- User Account Management
- Setup Jest (2nd Sprint)
- Base Unit Tests (2nd Sprint)
- Auth Unit Tests (2nd Sprint)
- Remove Password property from user API responses (2nd Sprint)
- User Feedback (2nd Sprint)
- User Feedback Unit Tests (2nd Sprint)

Yaser Aldammad (Yaser-Aldammad)

- User model
- User APIs
- User APIs Unit Tests (2nd Sprint)
- Email Notification (2nd Sprint)

Prabin Kshrestha (prabinKshrestha)

- Quiz model
- Quiz APIs
- Quiz APIs Unit Tests (2nd Sprint)
- Quiz History (2nd Sprint)
- Quiz HIstory Unit Tests (2nd Sprint)

Judah Sholola (JCK07115)

- Quiz Question model
- Quiz Question APIs
- Quiz Question APIs Unit Tests (2nd Sprint)
- Quiz MCQs APIs (2nd Sprint)
- Quiz MCQs APIs Unit Tests (2nd Sprint)

Backend Architecture and Class Diagram

The class diagram tells how each module fits into the architecture.

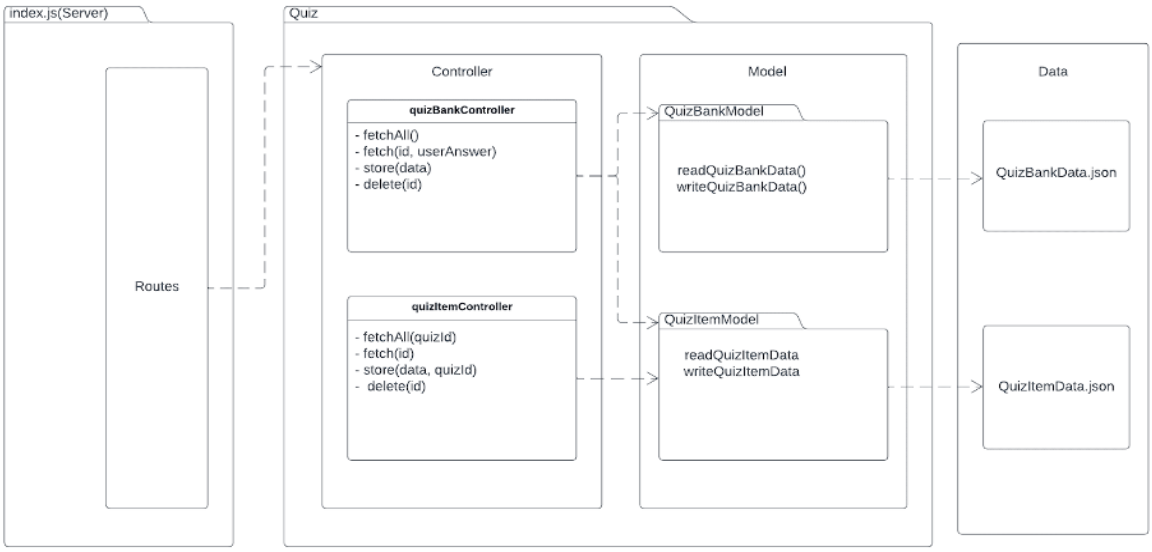


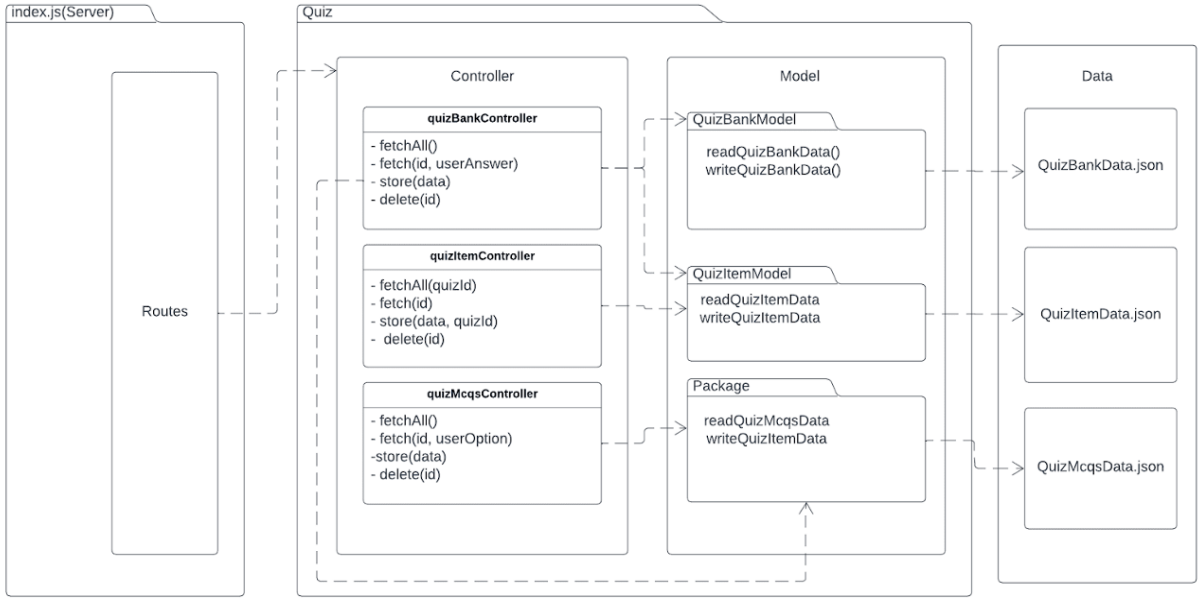
Figure 4: Backend Architecture

The figure below shows our basic project idea with model and controller. Firstly there is a server composed of routes, routes are responsible for associating HTTP verbs with a URL path/pattern and a function that is called to handle that pattern. There is a Quiz module with 2 packages model and controller. Firstly, the controller, it is comprising of two classes; QuizBank class which has record of all the quizzes and a QuizItem which has record of questions.Furthermore, model package has two models names Quizabank and QuizItem which are mainly responsible for implementing the domain logic.Finally, the last module is for data consisting of all the data regarding the quiz and the quiz questions separated in different files names QuizBankData and QuizItemData.

The quizBankController is associated with quizBankModel and QuizItemModel.Rather than giving dependency between quizBankController and quizItemController we have interlinked quizBankController with every item in the model, loosening the coupling.

Backend Architecture and Class Diagram Modified (MCQs Feature)

The class diagram tells how each module fits into the architecture.



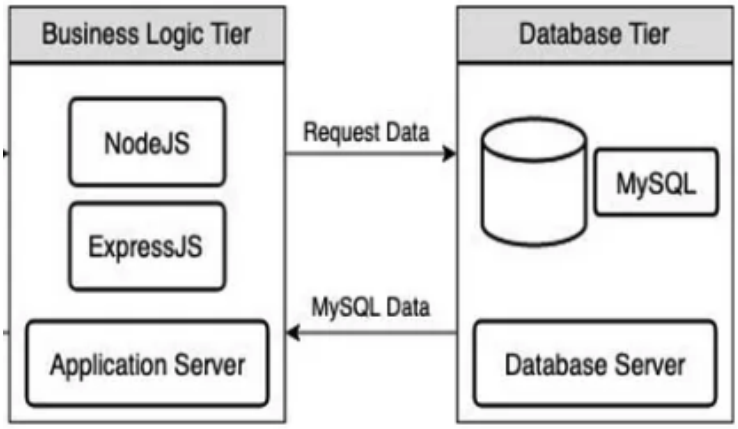
The figure below shows our basic project idea with model and controller. Firstly there is a server composed of routes, routes are responsible for associating HTTP verbs with a URL path/pattern and a function that is called to handle that pattern. There is a Quiz module with 2 packages model and controller. Firstly, the controller, it is comprising of two classes; QuizBank class which has record of all the quizzes and a QuizItem which has record of questions.Furthermore, model package has two models names Quizabank and QuizItem which are mainly responsible for implementing the domain logic.Finally, the last module is for data consisting of all the data regarding the quiz and the quiz questions separated in different files names QuizBankData and QuizItemData.

The quizBankController is associated with quizBankModel and QuizItemModel.Rather than giving dependency between quizBankController and quizItemController we have interlinked quizBankController with every item in the model, loosening the coupling.

Backend Code Structure Tree:

```
project
  ├── client
  └── server
    └── controller   – Storing APIs (GET, POST, PUT, DELETE)
     ├── index.js
     └── user.controller.js
    └── database
     ├── model       – store all the models of the project
     └── schema      – create attribute for each model
    ├── .eslintrc    – config ESLint Airbnb Coding Style
    ├── .babelrc     – migrate ES6->ES5 to run on different browsers
    ├── package.json – config ESLint Airbnb Coding Style
    └── App.js       – Everything a server needs to start
```

2-tier of Backend QuizApp Architecture



The Controller tier, also known as the Business Logic Tier, will be implemented using NodeJs and ExpressJS. This tier serves as the Application Server, facilitating communication between the Client tier and the Database tier. It will generate HTML pages for the user and handle HTTP requests and responses accordingly.

The Model tier, which is responsible for storing and retrieving data, will be powered by MongoDB and hosted on the Database tier. This tier stores all the necessary data required for the application to function smoothly.

Assignee for Backend code tasks (sprint 4):

Yaser Aldammad (Yaser-Aldammad)
- User roles middleware

Prabin Kshrestha (prabinKshrestha)
- Quiz leaderboard and ranking

Judah Sholola (JCK07115)
- Quiz Question MC and SATA validation

Muhammad Shaheryar (Muhammad-Shaheryar)
- Quiz Scheduling feature added