Author: Pranav, Shaheryar, Judah

Date: March 27, 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 for Assignment 3

The following is the module composition of the frontend:

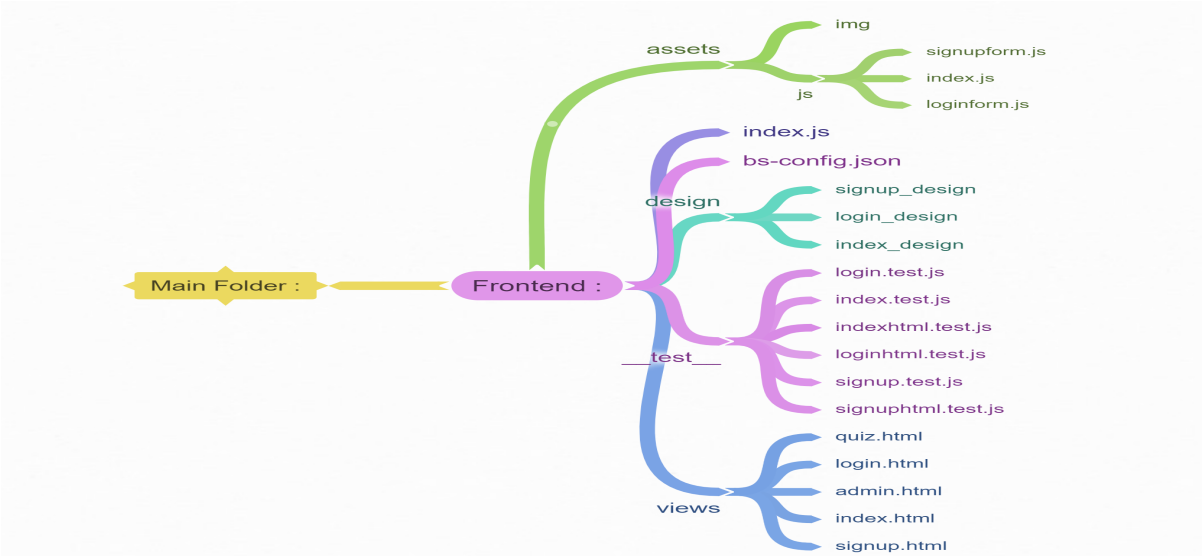

Figure 1: Frontend Folder Structure

Assignee for Frontend code tasks (module wise):

Pranav Arora (pranavarora1895)

- Previous Sprint Tasks
  - index.js
  - index.html
- New Code Tasks
  - index.test.js
  - indexhtml.test.js
  - loginhtml.test.js

Balsher Singh (balshersingh10)

- Previous Sprint Tasks
  - loginform.js
  - signupform.js
- New Code Tasks
  - login.test.js
  - signup.test.js
  - signuphtml.test.js

Frontend Architecture and Execution Flow

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

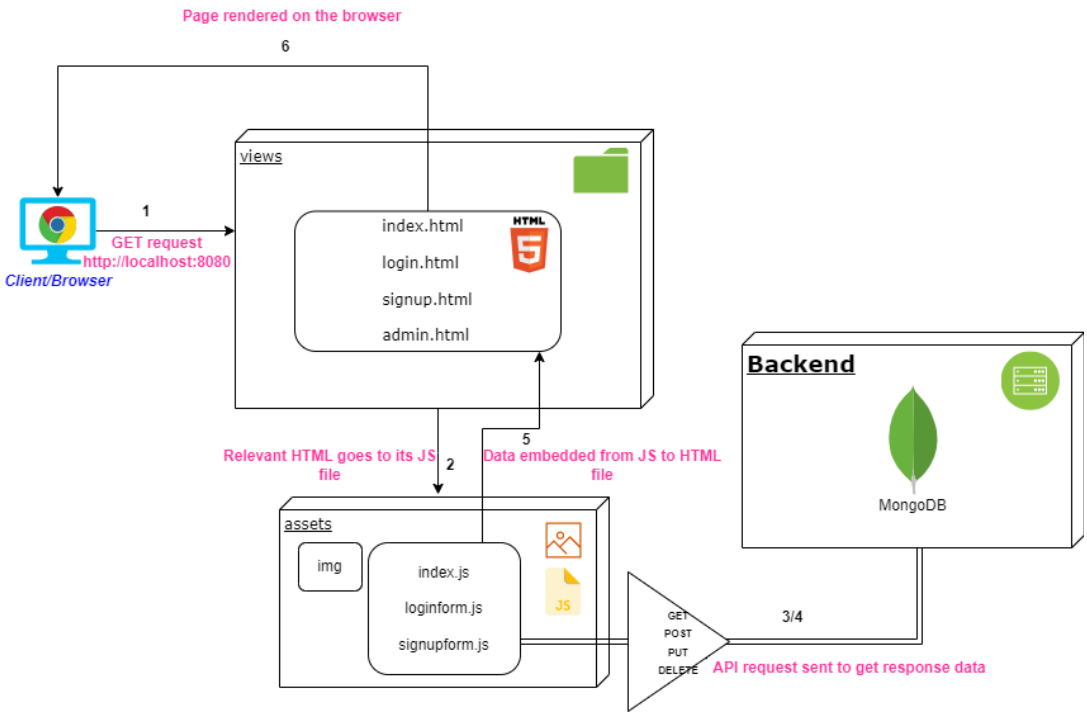

Figure 2: Execution Workflow of Frontend Architecture

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:

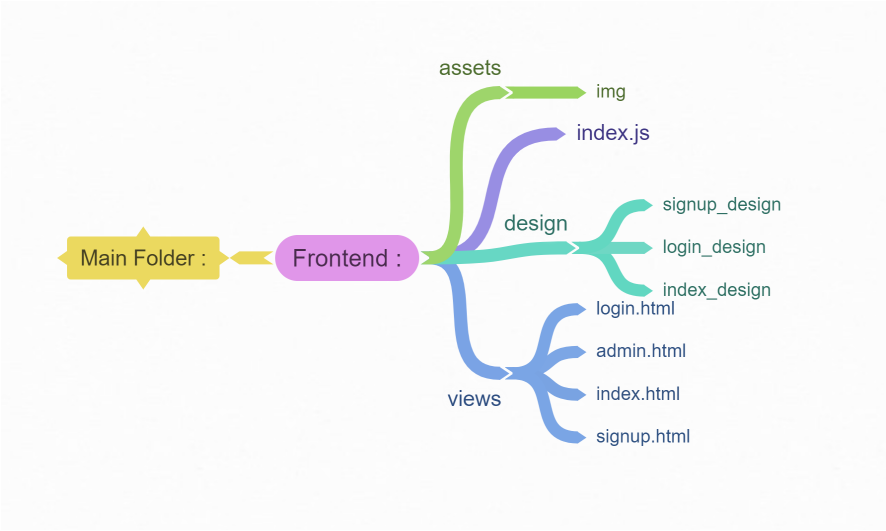

Figure 3: Frontend Folder Structure

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.

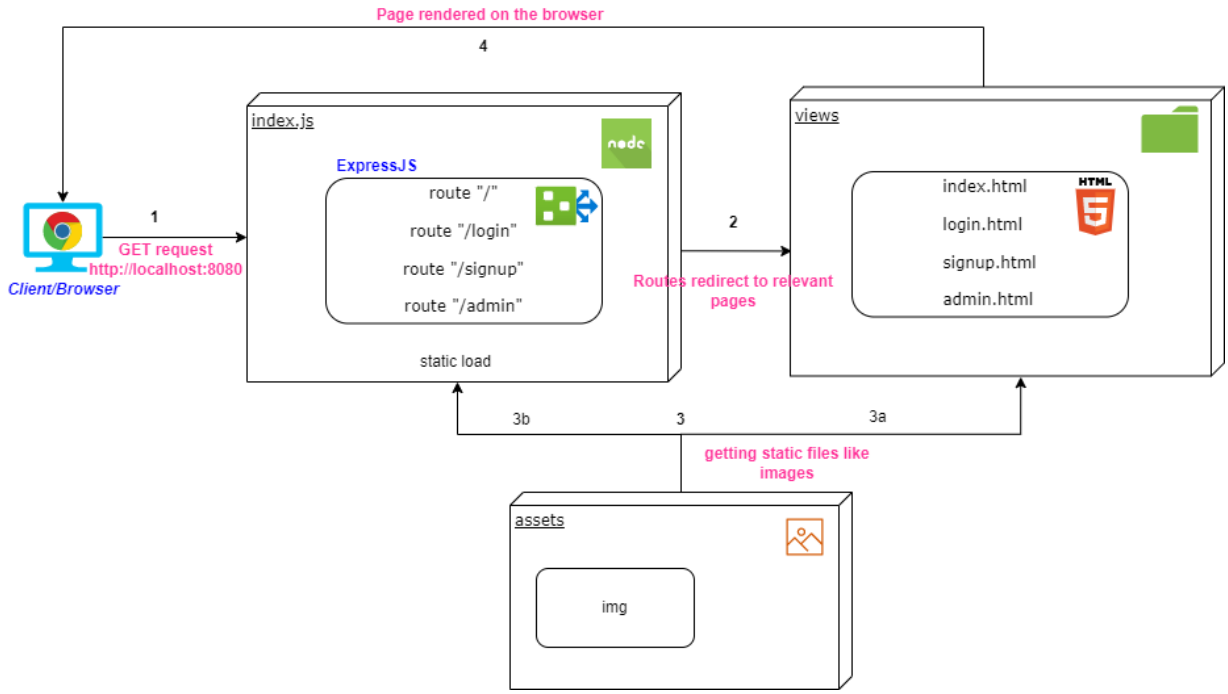

Figure 4: Execution Workflow of Frontend Architecture

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:

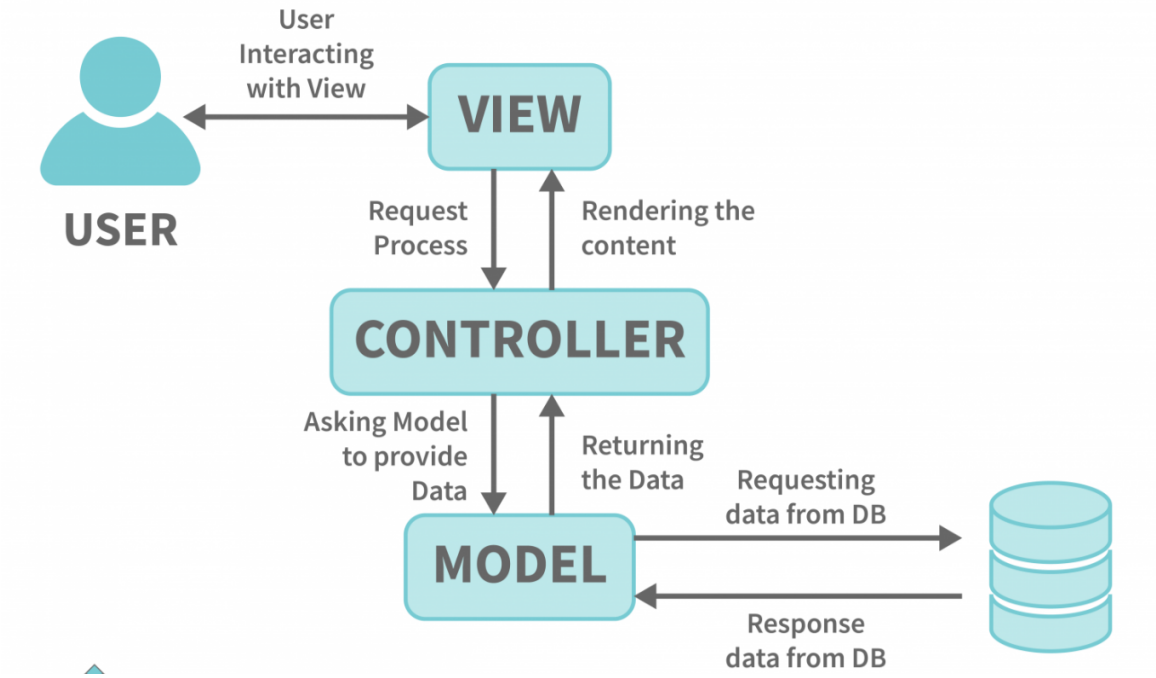

Figure 3: Backend MVC Structure

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.

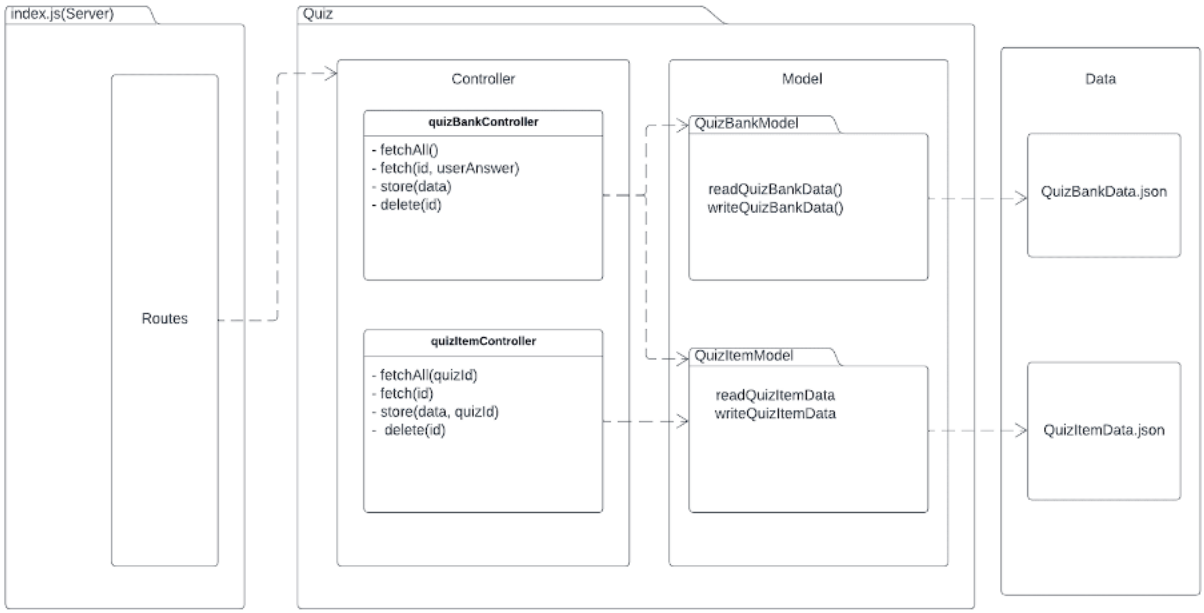

Figure 4: Backend Architecture