Implementation Document

for

SmartTutor

Version 0.1

Prepared by

Group :16 **Group Name:** Software avengers

Sonu Kumar	211052	sonuk21@iitk.ac.in
Ch Hemanth Kumar	210277	chandaka21@iitk.ac.in
Sarthak Paswan	220976	sarthakp22@iitk.ac.in
Yash Gothwal	211189	yashg21@iitk.ac.in
Kantule Ritesh Ramdas	210488	kantulerr21@iitk.ac.in
Saurav Kumar	210950	sauravk21@iitk.ac.in
Surendra kumar ahirwar	211083	surendrak21@iitk.ac.in
Rishit Bhutra	210857	rishitb21@iitk.ac.in
Krishna Chandu	220832	pkrishna22@iitk.ac.in

Course:CS253

Project Mentor - Sumit Chaduhry

Date-18th March 2024

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
0.1	Sarthak Paswan Yash Gothwal Saurav Kumar Kantule Ritesh Ramdas Ch Hemanth	First Draft	18/03/2024

1 Implementation Details

SmartTutor's System implementation includes three components. The components of the architecture are:

● Front End: The front end of the SmartTutor system is responsible for delivering the user interface and user experience to students, mentors and admins.

● Back End: The back end of the SmartTutor system handles the logic and processing of data, user authentication, and communication with the database.

● Database: The database component of the SmartTutor system stores and manages data related to users.

1.1 Front End

The presentation layer is accessible to users via a browser and consists of user interface components that enable interaction with the system.

It is developed using three core technologies: HTML, CSS, and JavaScript. While HTML is the code that determines what the website will contain, CSS controls how it will look. JavaScript and its frameworks make the website interactive and responsive to a user's actions.

We have used React, a JavaScript library for building user interfaces.
It is used particularly for single-page applications.

- It allows us to create reusable UI components, manage their state and handle efficiently.
- React is known for its virtual DOM feature, which optimizes the way updates are made to the page, leading to a better performance.
- With JSX (JavaScript XML), developers can write HTML-like code directly within JavaScript, making it easier to visualize and maintain the UI structure.

1.2 Back End

The Back end accepts user requests from the browser, processes them and determines the routes through which the data will be accessed. The workflows by which the data and requests travel are encoded in this layer. The back end is implemented in node.js and express.js.

Some advantages of using **node.js** are -

- **JavaScript Proficiency:**
- Makes use of the general understanding of JavaScript, which lowers the learning curve for developers.

- **Event-Driven, Non-Blocking I/O:**
- Because of its non-blocking I/O paradigm and event-driven design, Node.js is a great tool for managing large numbers of concurrent connections.

- **Scalability:**
- Node.js applications are easily scalable to meet increasing traffic demands, both vertically (by adding more worker processes) and horizontally (by upgrading server resources).

- **A Wealthy Open-Source Package Ecosystem:**
- The Node Package Manager (**npm**) has access to over two million third-party modules that offer pre-built functionality for a variety of applications.

And the advantages of using **express.js** are -

- **Fast prototyping and development are made possible by Express.js:**

- It provides a simple and adaptable framework that lets us take care of middleware, routing, and templating in addition to rapidly establishing a fundamental server structure.

- **Routing:**
- Express.js offers an organized method for creating routes that associate URLs with particular handlers or functions inside our application which made it possible for us to efficiently arrange code and design clear endpoints for processing HTTP requests.

- **Middleware:**
- Tasks like permission, logging, error management, and authentication are made easier with this modular approach with handling incoming requests and outgoing responses before they get to their intended location.

- **Templating Engines:**
- A number of templating engines, such as Jade, EJS, or Pug, are supported by Express.js, which makes it easier to create dynamic HTML content.

- **Designing RESTful APIs:**
- Express.js helps create Application Programming Interfaces (APIs) that follow best practices for **CRUD** (creating, reading, updating, and deleting) activities on resources. Consistency and compatibility across many applications and services are guaranteed by this specification.

1.3 Database

The Data Tier, sometimes called Database Tier, is where the information processed by the application is stored and managed. We have used MongoDB, which is more than a database. It's a complete developer data platform. With MongoDB Atlas, the cloud offering by MongoDB, you have access to a collection of services that all integrate nicely with your database. The main advantages of using MongoDB as our database are -

- **Adaptability:**
- Without a plan Design: Does not rely on inflexible predefined schemas to conform to evolving data structures.

- Semi-structured Data: Accommodates complicated interactions by storing data with different degrees of structure within a document.

- **Scalability:**
- Horizontal scaling: Increases the number of shards (data partitions) to effectively manage growing user traffic and data volumes.

- Cloud-Friendly: Easily deploys, manages, and scales automatically through seamless integration with leading cloud providers.

- **Achievement:**

- Fast Reads and Writes: Provides a responsive user experience by enabling speedy data retrieval and modification.

- MQL, or Rich Query Language: allows for effective data manipulation with sophisticated filtering, aggregation, and geographic queries.

- **JSON-like Structure:**

- Developers will find it intuitive since it aligns nicely with object-oriented programming languages.

- Rich Ecosystem: Development is made easier by copious documentation, tutorials, libraries, and community support resources.

1.4 Development and version control environment

We have used Git as our version control system. It is the most commonly used version control system that is used for software development and other version control tasks.

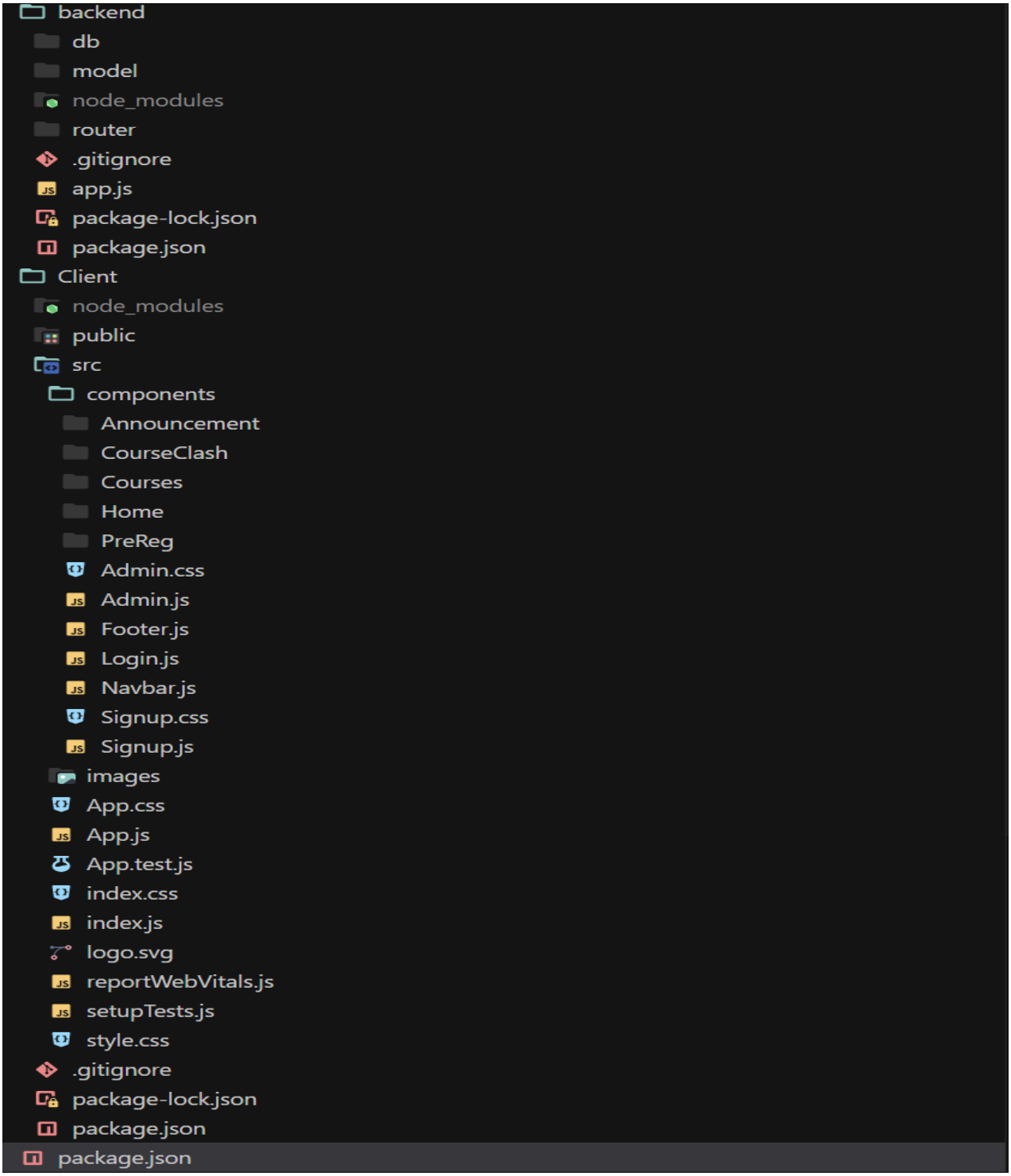
It tracks the changes you make to files, so you have a record of what has been done, and you can revert to specific versions should you ever need to. It also makes collaboration easier, allowing changes by multiple people to all to be merged into one source.

We used GitHub, a web-based Git repository hosting service to manage our repositories and collaborate amongst ourselves.

2 Codebase

The GitHub repository for storing and collaborating on the source code of this project is https://github.com/sauravkr21/group16.

The repository contains the following files:



The front end deployment is done in the ‘Client’ folder. The ‘backend’ folder contains the backend deployment of the project.

To locally run the project, first clone the GitHub repository with the git clone command. Then open two terminals, one in the Client directory and one in the backend directory. Run the command ‘npm i’ in both the directories’ terminals. Then, in the Client directory’s terminal run the command ‘npm start’. Switch to the backend directory’s terminal and run the command ‘node app.js’. The project will then be live at port 3000 and can be accessed by going to ‘localhost:3000’.

3 Completeness

In the Software Requirements Document, the *"Section 3: Specific Requirements"* subsection *"Section 3.1: External Interface Requirements"* lists all the desired product functionality which is implemented in our project SmartTutor's frontend.

Different user interfaces displayed in section 3.1 of the SRS document are implemented using React. The different views include - Homepage, login, register, preregistration, clashing courses and courses. It also includes the student and admin interfaces.

In the Software Requirements Document, the *"Section 3: Specific Requirements"* subsection *"Section 3.2: Functional Requirements"* lists all the desired product functionality out of which most of them are implemented in our project SmartTutor's backend.

Our application **SmartTutor** aims at improvement and facilitation of the preregistration process with ease. Currently, the implementation lacks some of the features mentioned in the SRS document, so in addition to complete implementation of the listed features we have a future development plan to improve the functionality of our product by adding more features which include:

- Check whether students meet the required prerequisites of the course before adding/enrolling in it.
- Help students track their progress in enrolled courses, including assignments completed, grades received, and upcoming deadlines.
- Appropriate database implementation for discussion forum conversations.
- A feature that allows verified users who have taken the course in the past semesters to evaluate and rate our application.
- Create a feedback loop where students may offer suggestions for ways to enhance and improve the courses.

Appendix A - Group Log

After the midterm examinations, much of the implementation was completed through asynchronous sessions conducted over WhatsApp in our group and in-person meetings at each other's resident halls of residence.

Date	Minutes
26/2	Planned the implementation and assigned the learning objectives to each group member
4/3	Discussed the functionalities in detail
8/3	Begin the work on the backend
9/3	Created user schema
10/3	Made sure of one id - one registration
13/3	Created admin access at the backend
15/3	Created a database on Mongodb
17/3	Worked on the software implementation document