Implementation Document

for

Academics Management Software

Version 1.0

Prepared by

Group #17: **Group Name: *same_storm***

Achint Agrawal	**180028**	achintag@iitk.ac.in
Somya Lohani	**190848**	somyaloh@iitk.ac.in
Sachin	**180639**	thakan@iitk.ac.in
Udhav Gupta	**180829**	udhavgup@iitk.ac.in
Yash Gupta	**190998**	gyash@iitk.ac.in
Himanshu Sood	**190381**	himsood@iitk.ac.in
Piyush Agarwal	**190600**	piyuagr@iitk.ac.in
Aditya Ranjan	**190068**	aranjan@iitk.ac.in
Yash Gupta	**190997**	yashbg@iitk.ac.in
Utkarsh Jain	**190928**	utkjain@iitk.ac.in

Course: CS253

Mentor TA: Ms. Shatroopa Saxena

Date: 20 March 2022

CONTENTS

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1.0	Entire Team of Developers	Starting from a template document, details of all the sections have been added.	20/03/22

Implementation Details

Provide the details of programming languages, frameworks, libraries, database systems, build systems, etc. that you have used for implementing your software.
Provide a brief justification of choosing any tool by stating its benefits over the alternatives.

We are using the **MERN stack (MongoDB, Express.js, React.js, Node.js)** for implementation. The languages we are using are HTML, CSS and JavaScript. For the frontend, we are using the Javascript library - React.js along with HTML and CSS. For the backend, we are using the Javascript runtime - Node.js and the Javascript framework - Express.js. We are using MongoDB as our database system.

We are using the MERN stack as it is one of the most popular tech stacks used for web development and uses a single programming language - JavaScript along with HTML and CSS. JavaScript is the most popular programming language for web development.

For testing the APIs, we have used **Postman** which is an HTTP client that is used for testing HTTP requests, utilizing a GUI. Through this software we can send requests corresponding to any HTTP method and get subsequent responses. Thus, with this tool we can test our backend even before integrating it with the frontend.We have selected Postman above any API testing client, because of easy usability and self-explanatory UI. One can make use of this tool without actually going through any tutorial. Also, it is easy to collaborate on Postman, and thus a team can test various endpoints and suggest any bugs in the codebase.

For Unit testing, we are planning to use the **Jest** library which provides aid in writing multiple test cases and then testing them on the app. Along with returning whether the tests were a success or failure, it also provides details on the test coverage and hence is very useful in knowing how consistent our code is.

~~We are also using **selenium**, a JavaScript library, for scraping Pingala and HelloIITK to get the course progress of the user and the course deadlines.~~

Along with all the above specified frameworks/libraries and programming languages, we have used Bash as our scripting language. Till now we have used a bash shell for running commands for starting the backend server and testing the frontend. We have also used bash commands for dealing with the file system and for navigating through the directories.

Summarizing the above, we'll be using the following -

1. Programming languages - JavaScript
2. Backend frameworks - Express.js
3. Backend runtime - Node.js
4. Frontend - React.js
5. Database system - MongoDB
6. Web scraping framework - selenium
7. API testing software - Postman

Codebase

Provide the link to your github repository.
Mention briefly how to navigate the codebase.

LINK TO REPOSITORY: https://github.com/yashbg/academics-management-software

Our repository is private. As and when required, we will add the TAs as collaborators.

STRUCTURE OF REPOSITORY:

- **TOP**
  - **my-app/**
  (directory to maintain frontend files)
    - **public/**
      (*It consists of various artifacts like app icon, main index.html for our app etc.)*
    - **src/**
      - **components/**
        (*It consists of .js files to implement interactive components like display notes, review forum*)
      - **pages/**
        (*It consists of .js files to implement static components like homepage, login, signup page etc*)
      - **App.css**
        (*This file contains all styling for App.js and its child components along with the inline styling*)
      - **App.js**
        (*This is essentially the main parent component in the application*)
      - **App.test.js**
        (*This contains test case for our application*)
      - **index.css**
        (*This file contains all styling for index.js or for "body" tag)*
      - **index.js**
        (*It is used to fetch App component and serve it to index.html*)
      - **reportWebVitals.js**
        (*This file is used to measure code performance but is currently not used in our implementation*)
      - **setupTests.js**
        (*This file is used to setup and the run test cases)*
    - **package-lock.json**
      (*It is an automatically generated file to track the dependency tree and changes made into node_modules folder whenever we run npm install)*
    - **package.json**
    (*This files keeps record of all dependencies related to the app and makes it easy for tools like npm to manage these dependencies across different contributors*)
    - **.gitignore**

(*It is a standard file to ignore confidential credentials from pushing into codebase on github*)

- **models/**
  (*This folder consists of database schemas for all the different collections*)
- **server/**
  - **index.js**
    (*This file serves as the entry point to our backend. All the middlewares have been specified here itself*)
  - **api/**
    (*This folder consists of all the API's which map to every endpoint present)*
  - **dao/**
    (*This folder consists of .js files that handle the database management related concerns and database queries)*
  - **routers/**
    (*This folder consists of .js files that handle the routing of our backend server*)
- .**env**
  (*This is the file used for declaring environment variables that can be used globally*)
- **package-lock.json**
(*It is an automatically generated file to track the dependency tree and changes made into node_modules folder whenever we run npm install)*
- **package.json**
  (*This files keeps record of all dependencies related to the app and makes it easy for tools like npm to manage these dependencies across different contributors*)
- .**gitignore**
(*It is a standard file to ignore confidential credentials from pushing into codebase on github*)

In the above description, the utility of each part is mentioned below it in the form of text enclosed in brackets in italics.

How to run our application locally:

1. Clone the GitHub repo from the repository link given above -
`git clone https://github.com/yashbg/academics-management-software.git`

2. Run the following commands in the exact same sequence -
   a) `cd academics-management-software`
   b) `npm install`
   c) `npm start // starts a server running on PORT - 3001`
   d) `cd my-app`
   e) `npm install`
   f) `npm start  // starts a client interface on PORT - 3000`

As soon as the react server starts, you will be directed to the `localhost:3000/`. You can manually visit the URL if not redirected automatically.

Completeness

Provide the details of the part of the SRS that have been completed in the implementation.

Provide the future development plan by listing down the features that will be added in the (may be hypothetical) future versions.

1. Login / Signup and Authorization
Login, signup and logout functionalities have been completed. For authorizing a user to access various endpoints of the web-app, we have used JSON Web Tokens with an expiration time of 15 minutes. Currently, tokens do not get refreshed i.e., a user will need to login again after 15 minutes. Along with this, a middleware ***authorization*** has been created that validates the JSON Web Token embedded in the cookie of the web browser (it is not stored in the browser storage as it makes the application vulnerable to XSS attack), and authorize a user only if the token is present in the cookie and it has not expired already.

2. Course Feedback
Course feedback functionality of our web application is completely done. With this, a user (student) can give a review for a course and ~~also can mark a course as favourite, which will be associated with raising the number of **stars** of a course~~. Also, other users (students) can now like or dislike and comment over the review given by a user (student).
NOTE: Currently the rating feature allows a student to give an overall rating of a course. Our team is supposed to make some changes in the database design. After doing these changes, we will then include various parameters like grading scheme, course content, etc. for rating.

3. Note-taking feature
Implemented backend logic for note taking feature. It broadly includes following:

1. Add note: Implemented functions corresponding to adding notes to the database. This function is different from the function used for modifying notes.
2. Delete note: To remove a particular note from the database, provide its noteID.
3. ~~Modify tag: In order to change text content, adding tags to notes or modifying other attributes like title of the notes~~
4. Fetch Notes: This function is getting used to fetch notes by notesID or by subject or by tag. The same feature will be used in display note feature yet to get implemented

4. Front-end for application
Front-end for login/signup and add-note feature is implemented. Also a basic skeleton for navigating through the app is ready. Since we are using React for our front-end, we are breaking our whole app into components, simple components mentioned above(home, navbar, signup, etc) are already complete.

Future Development Plans

In the upcoming versions of the app, we are supposed to complete the two remaining functional requirements -

1. Calendar Management - User will be able to manage his/her schedule just like Google Calendar with the added benefit of getting the academic deadlines already imported as events in the calendar. The calendar system will be integrated with the course planner.

2. Course Planner - User will be able to plan his/her upcoming semesters wisely using the data available from the Course Feedback feature of the application, and will also be able to use the calendar system for avoiding the potential of clashes of multiple courses.

After completing the functional requirements of the application we will work a little towards improving the GUI, particularly of the calendar section.

3. **Notes** - Allow rich text in nodes, modification of notes and searching of notes by tag.

4. **Course Feedback** - Allow users to modify their feedback and reply to comments

5. Add admin functionality to the system.

3. **Non Functional Requirements** -
With a working frontend and backend in hand, we will work towards making our application responsive. We will try to complete the following non-functional requirements -

- Usability

- Quick tour on the first time the user logs into the application
- User manual will be provided to the users which details the various steps needed to use the application
- Proper feedback and error messages/pop-ups will be shown on the completion of various tasks

- Availability

- The application will be tested regularly and extensively to ensure the removal of major bugs early on.
- Various issues and possible points of failures to be noted with their severity as well as likelihood of happening (risk matrix)
- Proper automatic error handling to be implemented in the code so that failures are notified as soon as possible.

Appendix A - Group Log

<*Please include here all the minutes from your group meetings, your group activities, and any other relevant information that will assist in determining the effort put forth to implement your software*>

DATE	MEMBERS PRESENT	TOPIC OF DISCUSSION
02- 03- 2022	All members were present	A selection of the language, library frameworks based on advantageous features and team members' knowledge.
06- 03- 2022	All members were present	A more precise division of tasks was done and the broad structure of the codebase was decided.
10- 03- 2022	All members were present	Progress on the tasks assigned was gauged and allocation of new tasks was done.
13- 03- 2022	All members were present	Implementation document was brought up and how to approach it was discussed, besides progress check.
17- 03- 2022	All members were present	Updates were done in favour of better features and the code completed was discussed.
19- 03- 2022	All members were present	The code completed was discussed and changes to be made were discussed.