Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

A Project Report
on
"Project Hub"

[Code No: COMP 206]

(For partial fulfilment of II/II Year/Semester in Computer Science/Engineering)

Submitted by

Abiral Adhikari (02)

Aakriti Banjara (05)

Nischal Baral (06)

Prashant Manandhar (30)

Samir Wagle (60)

Submitted to

Dr. Rajani Chulyadyo

Department of Computer Science and Engineering

Submission Date: 15/02/2023

Bona fide Certificate

This project work on

“ProjectHub”

is the bona fide work of

“Abiral Adhikari (02)

Aakriti Banjara (05)

Nischal Baral (06)

Prashant Manandhar(30)

Samir Wagle (60)”

who carried out the project work under my supervision.

Project Supervisor

_____

Dr. Rabindra Bista

Associate Professor

Department of Computer Science and Engineering

Date:

Abstract

Our Semester Project, Project Hub is a web-based service used to manage the task and view work progress for certain projects. For the development of the project, we have used the MERN (MongoDB Express React Node) stack. We have used React for the frontend and MongoDB for backend and Express for RESTful API. The working project is also deployed on a web at https://projecthub-78g5.onrender.com/ .

This project aims to facilitate the user to handle the project among the team members by dividing the workload. This project is used to assign the task among the team members for certain projects. User can create their own project or join the project only via the project code. The team members can collaborate with each other through chat, see each other's progress rate, resources and assigned tasks. The progress rate is automatically tracked after the user completed the task. This has simple design and minimalistic UI so the user can easily work around the website.

Keywords: Project Hub, MERN stack, project management, Mongo DB.

Table of Contents

List of Figures

List of Tables

Acronyms/Abbreviations

The list of all abbreviations used in the documentation is included in this section. See the example below:

VR	Virtual Reality
RAM	Random Access Memory
DMA	Dynamic Memory Allocation

Chapter 1. Introduction

1.1 Background

Today, a lot of engineering projects have become very large. A project employs a lot of people and managing all of them has become a difficult task. Managers are facing several problems in managing the project, deadline, resources, and workload, especially with information sharing between the team member. This has made it difficult for teams to manage projects effectively and led to delays, missed deadlines, and decreased productivity.

Project Management Services like Trello were developed to address such issues and provide teams with the tools and features to help teams plan, organize, and execute projects effectively. But the value per cost of these services is not suitable for entry-level developers like students and interns. Our project, “Project Hub” aims to provide a web-based platform for developers, engineers, and designers to collaborate remotely on a project and track each other’s progress on the go using collaboration tools for real-time reporting and analytics.

Project Hub helps developers to increase visibility and transparency in a given project by enabling task management, progress tracking, and group discussions. So, teams can work together and stay on the same page which in turn helps to reduce the risk of miscommunication with the feature of discussion, resource sharing, and task management. Project Hub is an essential tool for any team looking to complete their projects quickly and efficiently.

1.2 Objectives

The main objectives of our application are as follows.

- Formation of a project group along with work division.
- To enable team members to update and track their progress.

- To facilitate the discussion for project planning.
- Making remote collaboration easy and efficient.
- To improve our skills as MERN stack developers.

1.3 Motivation and Significance

Any big project is a collaborative process. For the successful completion of a project, all members need to do their fair share of work. Keeping track of what everyone is doing and making sure everyone is doing their part becomes difficult. In our last project, tracking the individual's contribution became difficult and we couldn't hit the scheduled milestone. That was the breakthrough moment when we realized the importance of a platform for managing a project and tracking everyone's performance.

This project aims to develop a platform for managing projects by tracking through activities of everyone. We aim to provide visibility and transparency in each project by managing tasks, deadlines, and progress. By providing a centralized platform for project management, this website can help organizations deliver high-quality results.

Chapter 2. Related Works

2.1 ClickUp:

ClickUp is one of the popular platforms for project management with customization tailored to each project and team. ClickUp provides users with an all-in-one free project management software. It includes every tool you need to plan, organize, and collaborate on projects, with an easy-to-use interface. And with over 1,000 native and third-party integrations, you can sync virtually any app with ClickUp to truly "bring all your work into one place". ClickUp offers free and paid subscriptions. It has its own API for custom integration and provides facilities like integration with Figma, GitHub, Slag, DropBox, and Google Calendar.

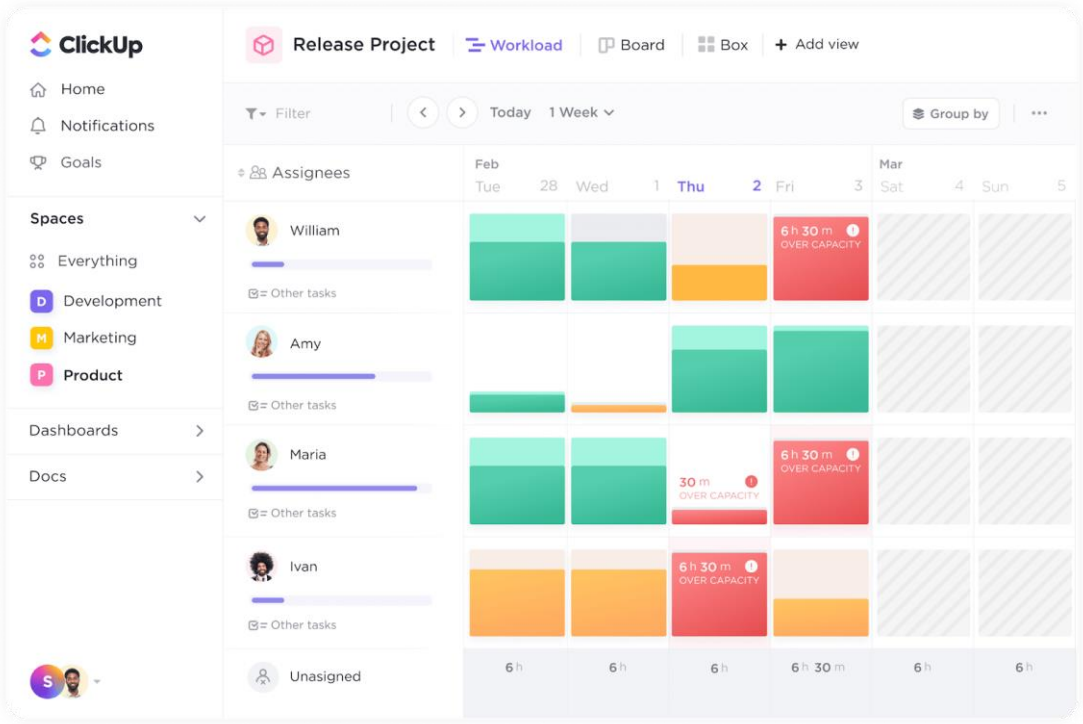

Figure 1 ClickUp 1

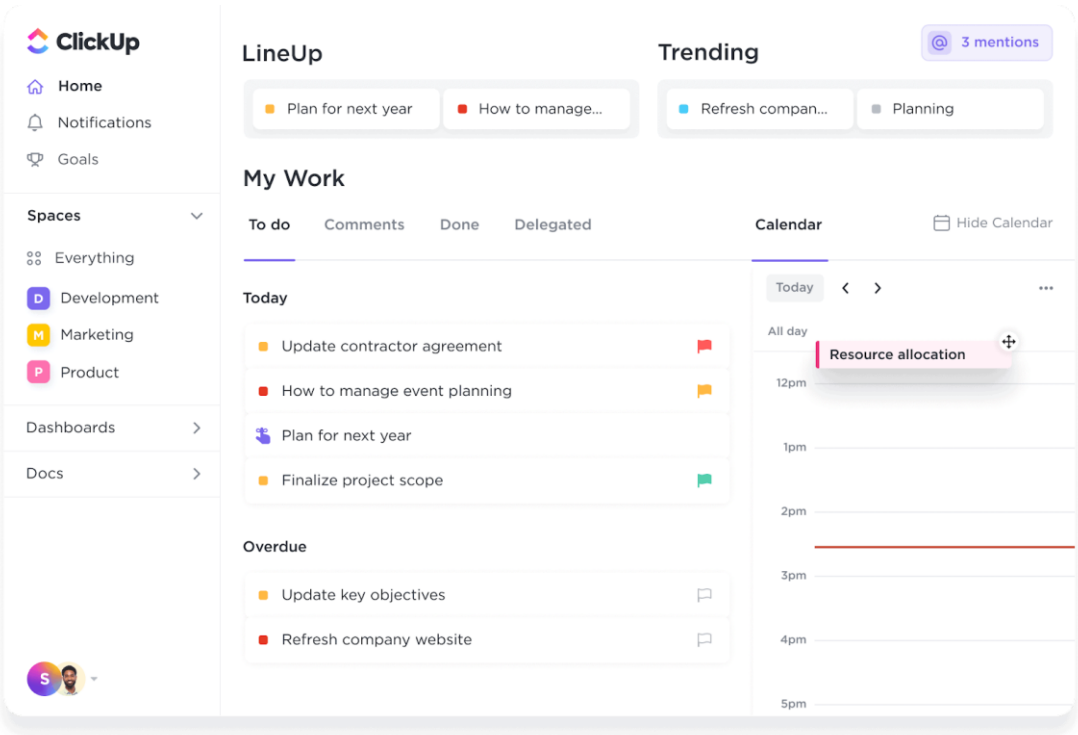

Figure 2 ClickUp 2

○ 2.2 Trello

Trello is also a web-based, list-making application and was founded in 2011. In Trello, users can create task birds in different columns and move the task between the project members. To Do, InProgress, and Done are generally shown in columns. This website is best used by estate managers, software project managers, school bulletin boards, lesson planning, accounting, web designing, gaming, and law office case management.

x

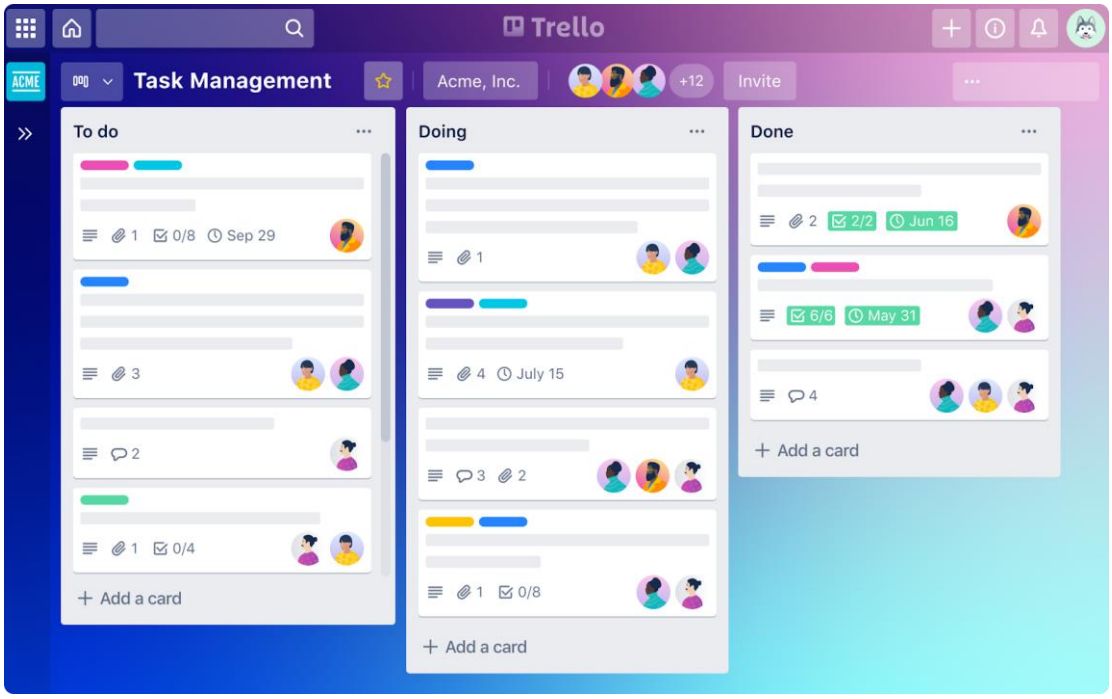

Figure 3 Trello 1

2.3 Asana

Asana was invented by two Facebook alumni in 2008. It provides user-friendly interference and its board applicability for most teams and uses cases. It provides a ton of flexibility over how it looks and can modify/move the task around easily. It provides services such as integration with other applications to become a hub for all your business activities. It provides unlimited features on the Free version. It also provides paid subscription with unlimited Dashboard, advanced integration, portfolios, and goals.

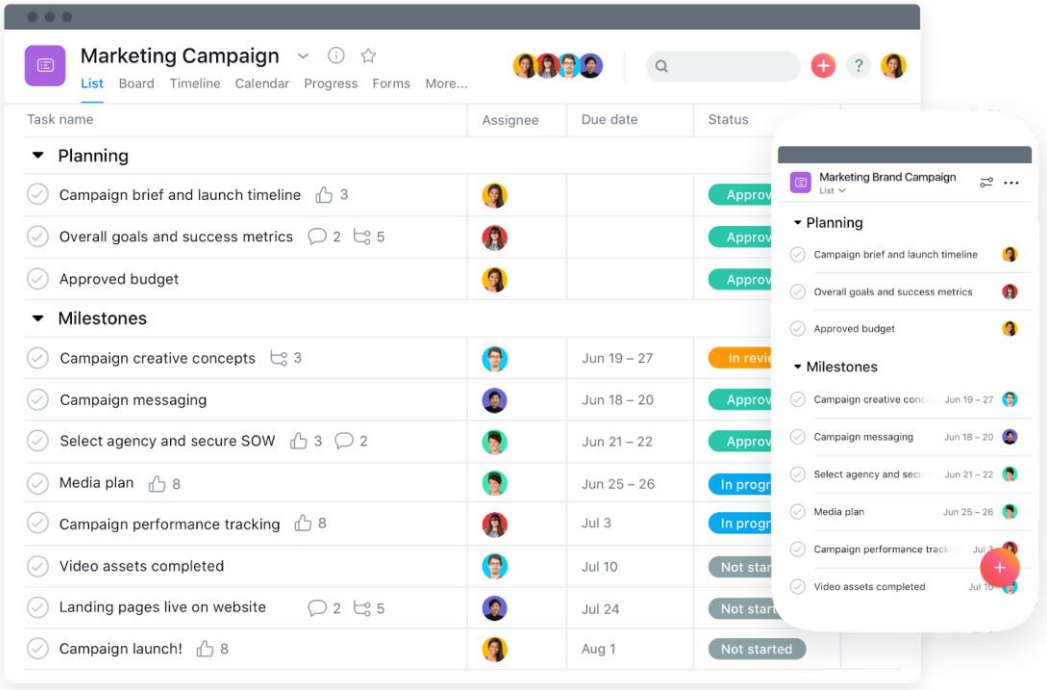

Figure 4 Asana 1

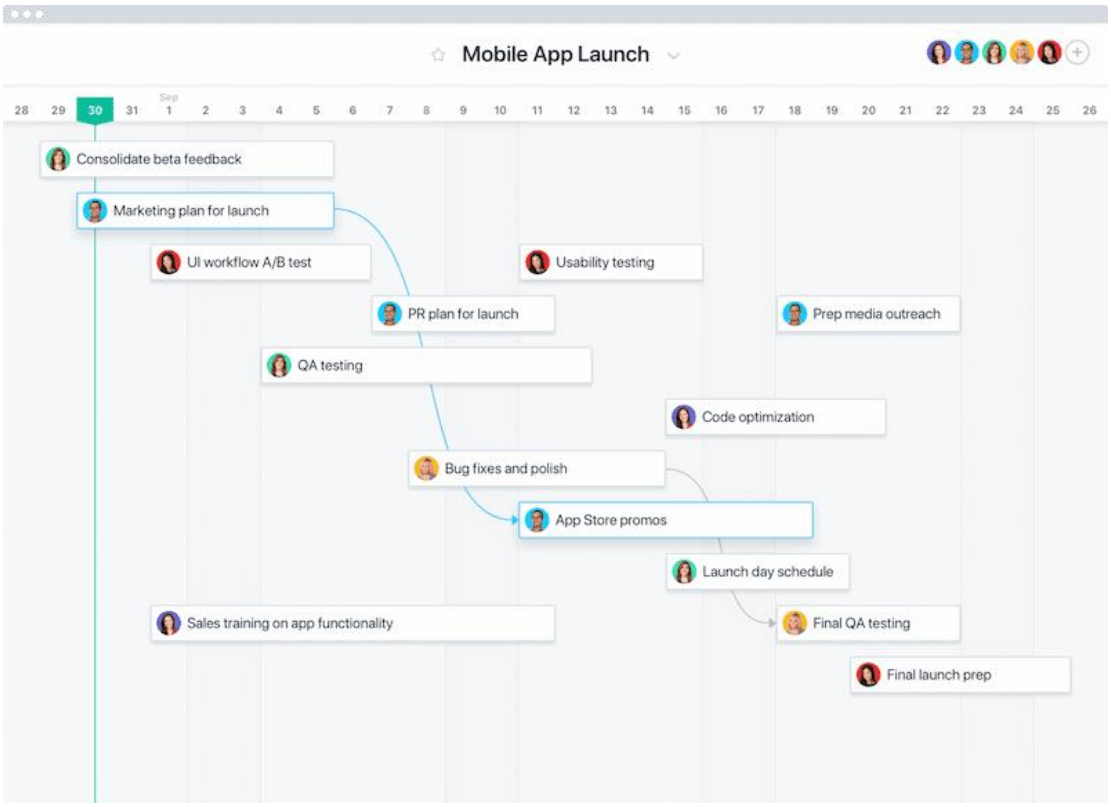

Figure 5 Asana 2

2.4 Jira

Jira is a popular software development platform that is primarily developed to assist teams with task planning, monitoring, and management. It was created by Atlassian, and thousands of organizations and teams use it globally. Jira is a web-based program that gives teams a central location to establish and manage their tasks, projects, and issues. Teams may manage workflows, give tasks to team members, and work together on projects while tracking their progress. Jira manages projects using an agile style and has tools including user stories, sprints, backlogs, and kanban boards. Jira may be customized by teams to meet their unique requirements thanks to its support for custom processes and issue categories. Jira connects with a broad variety of third-party tools and services in addition to its core functionality, including Github, Bitbucket, Slack, and many others. Jira is popular among

software development teams, but it may also be used to manage tasks and projects by other departments, including marketing, HR, and operations.

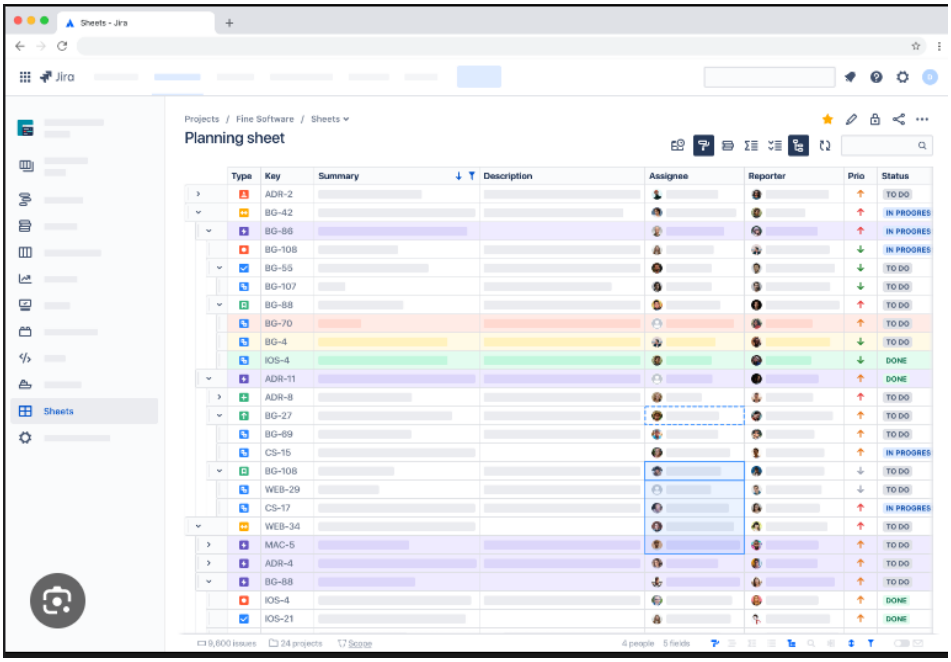

Figure 6 Jira 1

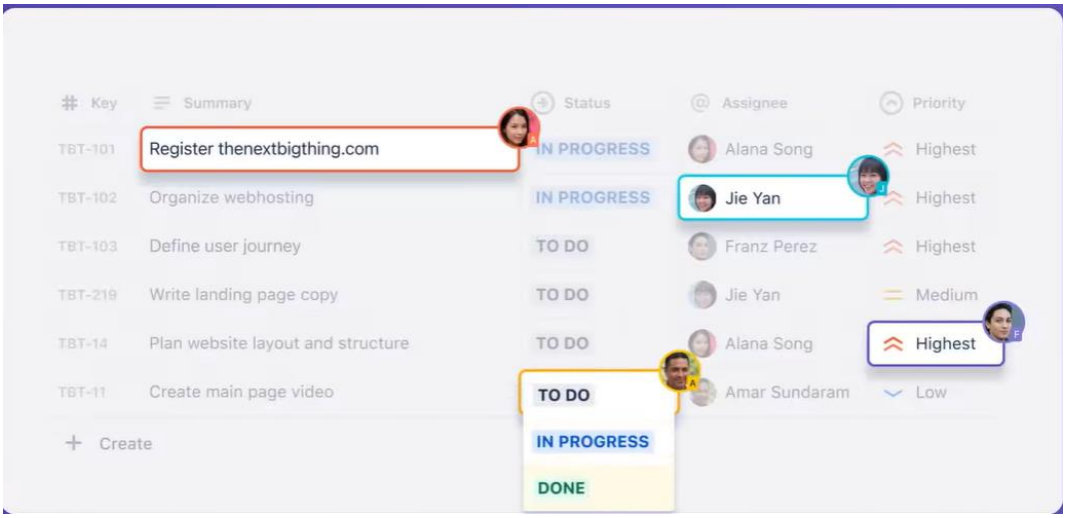

Figure 7 Jira 2

Chapter 3. Design and Implementation

3.1 FrontEnd

3.1.1 Design and Interface

We used HTML, CSS, and React library of JavaScript to create the designs and working of the frontend. At first, we got comfortable with React and started to work on the project. Then we started by working with the login and signup page. The login and signup page has a plain background and a portal to enter the user credentials for logging in or signing up on the platform. There, after collecting the credentials from the user, it is sent to the backend to create a new user or to verify the credentials of the existing users.

Then the user is taken to the dashboard page where they can access the projects they are in or they can create or join new projects. Creating a new project generates a unique code for each project which is used for other users to join the project. The user there can view their projects, the deadlines, and the creators of the projects.

After selecting a project, the project landing page is generated which shows the details of the projects. The details include features like individual completion rates, and project codes along with the options of a to-do page, resources pages, and discussion pages.

The to-do page shows the cards of all the assigned tasks, their descriptions, and the assigned deadlines. The creator of the project has the option to create new tasks and assign them to the members of the projects. The user can change the status of the assigned tasks between backlog, todo, in progress, and review. After the task is reviewed by the creator, the task can be moved to the completed stage.

The resources page contains all the resources shared by the users in the project. It shows the list of all the resources, their links, creator, and creation date. The user

is also provided with the option to add resources to the project on the same page by filling up the form while creating.

The discussion page is the place for the members to discuss the project. There they can exchange messages which are displayed there for everyone in the project to see. When a new message is added, the message is sent to the backend to store in the database and the same message is posted in the page too.

The profile page contains information about the user where they can see the information they entered when they created the account which includes their name, GitHub link, email address, and their contact number. It also includes things about me, my hobbies, and my skills.

3.1.2 Routing and Dynamic Range Rendering

Routing in the MERN (MongoDB, Express, React, Node.js) stack involves defining server-side routes using Express.js and client-side routes using React Router.

Server-side routing:

The server-side routing in a MERN stack involves defining the HTTP methods for handling requests and responses. This is done in the server-side code using the Express.js framework. The routes define the URL paths that the server can handle, and they are associated with specific controller functions that handle the requests.

For example, In our project, a route was defined for handling GET requests to the /API/user/login URL path. This route can be associated with a controller function that retrieves data from the MongoDB database and returns it as a response to the client. Similarly, a route can be defined for handling POST requests to the API/user/signup URL path, which can be associated with a controller function that adds data to the MongoDB database.

Client-side routing:

Client-side routing in a MERN stack involves defining the routes that the React application can handle. In client-side routing, routing is handled internally instead

of redirecting to the server even though the URL of the website is changed. For this, we have to use the React Router library, which provides a set of components that can be used to define the routes and their associated components.

Navigation:

To enable navigation between different routes in a MERN stack application, the React Router provides the Route component. This component can be used to create routes between different routes in the application. For creating a route we have to use BrowserRouter and Route to define our in app.js.When a user clicks on a component, the associated route is rendered with the help of the useNavigate function from react-router, and the URL in the browser's address bar is updated accordingly.

3.1.3 User Access Control and Session Tracking

In our website, the token is used for the authentication of the user which is saved in the cookies of the browser. The cookies automatically expired after 3 days. If the user login after that the user again has to log in and generate a new token. Without the token, the user cannot access any of the services of the website. To retrieve data for any page, a token is passed in the header of the fetch API as authentication. If the token is valid, the data is sent from the backend, otherwise, the error is shown as a popup in the frontend, and no access is granted.

3.2 API:

API is the set of programming codes to transfer data from one software to another. In this project, we have a fetch API to fetch data and update databases. For the authentication, we have used the help of a token which was sent in the header for the authentication of the user. The project id is sent in the form of a param in URL which helps in determining the extraction of the data from the backend. The project id is unique to every project created.

For the login and signup page, the user's credential is passed in the form of JSON in the body of the fetch API, and in return, the token is sent from the backend which was saved in cookies if the connection was successful otherwise an error is shown. In the dashboard, the first token is sent in the header as authentication. If the authentication was valid, it sends back the data needed for the frontend otherwise blank data is returned.

3.3 Database and UML Diagram

The Models gathered in our project are modeled using a Document-oriented database, MongoDB. Our database design is structured into five collections: discussion, resource, project, todo, and user.

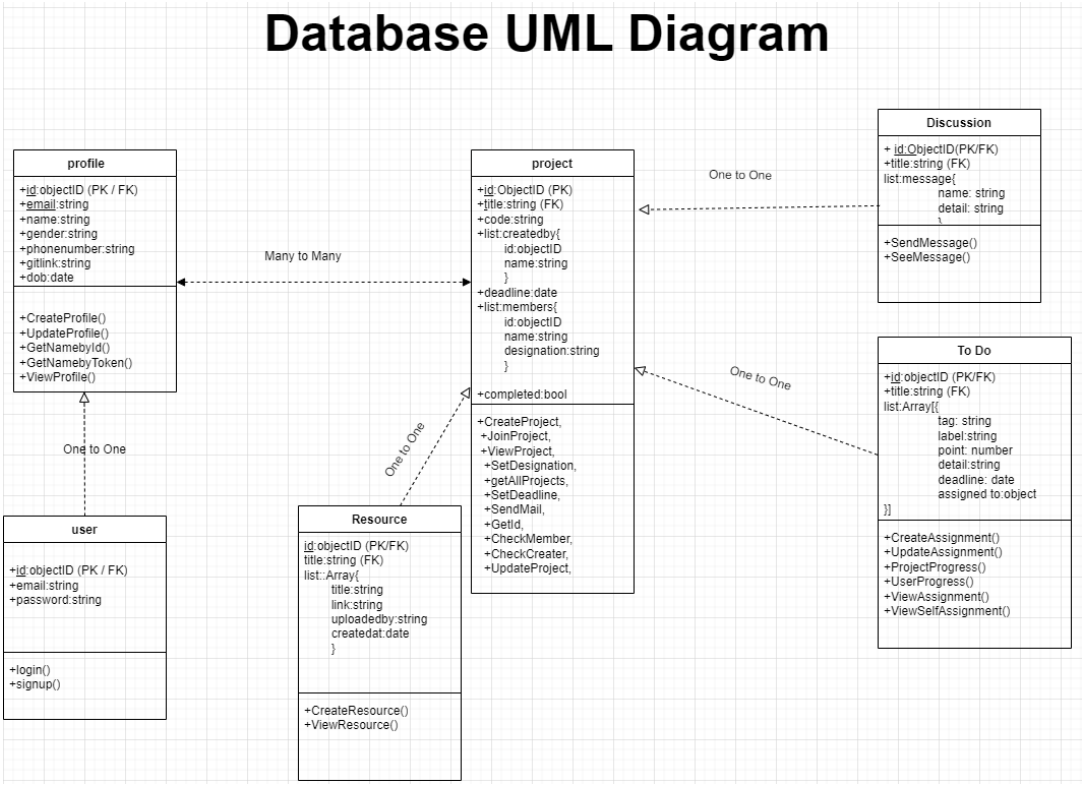

Figure 8 UML Diagram

3.3.1 Discussion:

"Discussion" collection stores the messages sent by the project members. The "_id" field is used to ensure that only members of a project can access and

participate in the discussion. “message” field uses an array of message objects. Each message object contains information about the message, including the sender, timestamp, and content.

3.3.2 Resource:

Resource collection allows the project members to add the resources such as GitHub link which are useful for the project. The “_id” field references the Project collection that identifies the project to which resources are to be uploaded to. “list” field is an array of objects that contains a link to the GitHub link, including the sender and timestamp.

3.3.3 project:

Every project in the “project” collection uses a “code” field which is unique and indexed for speeding up queries and preventing duplication. The “Created By” field is an object that contains a unique _id and the name of the project member. The “Deadline” field stores the date when the project needs to be completed. Other than that, this collection has the completed flag with type boolean which shows if the project has been completed or not.

3.3.4 User:

The user collection stores all the information of the account owner. Each user has the _id field for each user. The password is hashed before storing it in the database. The user collection also stores the name, email, and GitHub link.

3.3.5 Todo:

Each project has assignments defined by the todo collection. The collection has the _id field referencing the main project. The todo has a “tag” property defining the stage of completion of the assignment. It also has the assigned to property referencing the profile of the member to which the task has been assigned. The collection also stores the date of the deadline and has a boolean flag for completion of the assignment.

3.4 Backend: ExpressJS REST API

3.4.1 Project directory setup and structure,

In the beginning, we need to set up the project. We are using the MERN stack in our project. There are different files and folders we need to configure at the beginning. First of all, we have a “controller” folder that controls all the views. Second, we have “middleware”. Inside middleware, there is an “auth.js” file that controls the authorization using a bearer token. In the “database” folder there is “database.js” that establishes the connection with MongoDB. The “Routes” folder contains all the paths that are available in our project. It directs the path. “Server.js” is there to initialize the project. “.env” is there to set up the environment and to give a port number for the database. “Package.json” and “package-lock.json” contain all the dependencies of the project.

3.4.2 Backend Flow Logic:

The server starts with the “npm run dev” command at “server.js” using the environment variable values in the “.env” file. When the server starts the express app in “app.js” also runs calling functions :dbconnect, czors, listen, and middleware present in it. The dbconnect function imported from database.js in the database folder is used to connect to the backend server to the Mongo DataBase hosted on the web.

The server then forwards the request to either of the following route files: user.js,project.js,todo.js,discussion.js and resource.js in the “routes” folder. From the routes file the appropriate function present in one of the controller files among UserController.js, ProfileController.js, DiscussionController.js, and TodoController.js ProjectController.js and ResourceController.js inside of the “controllers” folder is invoked. But before invoking the controller functions ReqAuth function from “auth.js” in middleware is invoked to check for authorization.

The invoked controller function takes the request body from the frontend as an argument and returns a JSON object with status code 200 for success and usually 404 or other for failure. When a controller function is called it may access the database collections using the schemas previously defined inside respective model files: user.js, project.js, discussion.js, todo.js, and resource.js present in the models folder.

3.4.3 Server Configuration

Our REST API is built on an express web framework. This section briefly explains the configuration and extensions used in our application.

a) **MongoDB configuration:** We store our data in MongoDB Atlas. We communicate our application to the database using dbconnect() in database.js. It allows us to access and query our remote database.

b) **Token Authentication:** To use services on our server, the web application must be authenticated with a unique BearerToken. The Bearer is a time-serialized token that is generated using JWT (JSON web tokens) package which as the name suggests "allows access to the bearer".It is made up of three parts: Header, Payload, and Secret. The header contains header info, the payload contains info like user id and the secret is a private key for encoding. This ensures that clients must be authorized to send requests on our endpoints.

3.4.4 Endpoints:

Our express application listens a total. These endpoints can be categorized into:

3.4.4.1 User Route(/api/user)

3.4.4.1.1 User Login and Signup:

a.) **SignUp:** These endpoints provide user signup

Table Route:"/signup"		
Function: SignUp		
Method	POST	

Request	Body:
	{
	"email": String,
	"password": String
	}
		email: Required, Unique: str@str.str
		password: Required,minimum8 character with 1 UpperCase,1 LowerCase,1 Number and 1 Special Character.

Response	Success:200	Fail:404

	{
	"createuser": Object,
	"token": String,
	}
		“All fields must be filled” or
		"Please enter a valid email address" or
		"Password and email shouldn't be same" or
		"Please enter a strong password with….Character" or
		‘'Error!! Email already exists'’

Table 1 User Signup Route

Login: This endpoint validates user login

Route:" /login"	
Function: Login	
Method	POST

Request	Body:
	{
	"email": String,
	"password": String,
	}
		Required

Response	Success:200	Fail:404

	{
	"email": String,
	"token": String,
	}
		"All fields must be filled"
		"Incorrect Email"
		"Invalid Login Credentials"

Table 2 User Login Route

3.4.4.1.2 User Name:

These endpoints give username by user id or Bearer Token.

Route:"/getname"		
Function:GetNamebyToken		
Method	GET	
Response	Success:200	Fail:404
	{"name": String}	“Not Found”
Route:"/getname/:userid"		
Function: GetNamebyId		
Method	GET	

Route:"/getname"		
Function:GetNamebyToken		
Request	Params:userid	
Response	Success:200	Fail:404
	{"name": String}	“Not Found”

Table 3 Username Routes

3.4.4.1.3 User Profile Routes:

This route creates and shows the user profile

Route:"/createprofile"		
Function:CreateProfile		
Method	POST	

Request	Params:None
	
	Body:
	{
	"name": String,
	"phonenumber": Number,
	"gender": String,
	“dob”: Date,
	“gitlink”: String,
	}
		Required:
		name,email,phonenumber,gitlink
		Optional:gender,dob

Response	Success:200	Fail:404

	CreateProfile: Object (which contains details of user profile created)	"You must provide a firstname,phonenumber and gitlink compulsarily" or "Profile already exists"

Table 4 Create Profile Route

Route:"/viewprofile"		
Function:ViewProfile		
Method	GET	
Response	Success:200	Fail:404
	profile:Object (which contains details of user profile)	"Error Message"

Table 5 View Profile Route

3.4.4.2 ToDo Route(/api/todo):

ToDo routes provide endpoints for different todo services..

3.4.4.2.1 View Todo: This endpoint enables user to view their todo list..

Route:"/view/:projectid"		
Function:ViewAssignment		
Method	GET	
Request:	params:projectid	
Response	Success:200	Fail:404

	todolist:Object (contains all the todo assigned to user with the token)	"You are not member of the project" Or "Error Message"

Table 6 View Todo Route

3.4.4.2.2 Create Todo: This endpoint enables the head to assign todo to a project member.

Route:"/create/:projectid"		
Function:CreateAssignment		
Method	PATCH	

Request	params:projectid
	
	body:
	{
	"tag": String,
	"title": String,
	"label": String,
	"detail": String,
	"point": Number,
	"deadline": Date,
	"assignedto": ObjectId
	}
		All required

Response	Success:200	Fail:404
	"ToDo Successfully Assigned"	"Fill all the fields" or "There is no such project of which you are manager"

Table 7 Create Todo Route

3.4.4.2.3. Update Todo: This endpoint enables the head and assigned person to update todo.

Route:"/update/:projectid/:todoid"		
Function:UpdateAssignment		
Method	PATCH	

Request	params:projectid,todoid
	
	body:
	{
	"tag": String,
	"title": String,
	"label": String,
	"detail": String,
	"point": Number,
	"deadline": Date,
	"assignedto": ObjectId,
	}
		All optional except assignedto

Response	Success:200	Fail:404
	"Todo Successfully Updated"	"You are not member of the project" or, "No such todo exist belonging to you" or, "No such todo exists assigned to you"

Table 8 Update Todo Route

3.4.4.2.4. ProjectProgress:

This endpoint gives details of project completion based on point allocated and complete tag belonging to a user.

Route:"/projectprogress/:projectid"		
Function:ProjectProgress		
Method	GET	
Request	params:projectid	
Response	Success:200	Fail:404

	{
	"name": String,
	"TotalPoints": Number,
	"Completed": Number
	}
		"Error Message"

Table 9 Project Progress Route

3.4.4.4 Resources Route(/api/resource):

Resources Route provides endpoints for services regarding resources.

3.4.4.4.1 Create Resource: This endpoint enables project members to create new resource links related to the project.

Route:"/create/:projectid"	
Function:CreateResource	
Method	PATCH

Request	params:projectid
	
	body:
	{
	"title": String,
	"link": String
	}
		All required

Response	Success:200	Fail:404
	"Successfull uploaded resource",	"Fill all the fields" or, "Project with given id doesn't exist" or, "You are not a member of this project"

Table 10 Create Resource Route

3.4.4.4.2 View Resource: This endpoint enables project members to view all resources related to the project.

Route:"/view/:projectid"		
Function:CreateResource		
Method	PATCH	
Request	params:projectid	
Response	Success:200	Fail:404

	list:Object (list of all the resources for given project id),	"Project with given id doesn't exist" or "You are not a member of this project"

Table 11 View Resource Route

3.4.4.5 Project Route(/api/project):

3.4.4.5.1 Create Project: This endpoint allows user to create new projects.

Route: /create		
Functions: CreateProject		
Method	POST	

Request	{
	"title": String,
	"details": String,
	"deadline": String,
	
	}
		
		Title
		Details

Response	Success:200	Fail:404

	createProject: Object (Details of the created project with project code)	“Project with the same name already exists”.

Table 12 Create Project Route

3.4.4.5.2 Join Project:

This endpoint is used to join the project created by the manager.

Join Project		
Route: /join		
Functions: JoinProject		
Method	PATCH	

Request	{
	
	"code": “String”
	
	}
	
		Project Code Required

Response	Success:200	Fail:404
	“You have successfully joined project”	“Project code is not valid”

Table 13 Join Project Route

3.4.4.5.3 Get all projects:

This function lists all the projects joined by the user

Get all projects		
Route: /getall		
Function: getAllProject		
Method	GET	
Response	Success:200	Fail:404
	projectmatch: Object (returns list of all the projects in which user is a member)	"You have not joined any project"

Table 14 Getall Projects Route

3.4.4.5.4 View project:

It sends the detail of the selected project.

View Project	
Route: /view/:projectid	
Function: ViewProject	
Method	GET

Response	Success:200	Fail:404
	Projectcheck: object or Projectdetails: object (Sends detail of project depending upon the head or just member)	"You are not the member of this project"

Table 15 View Project Route

3.4.4.5.5 Update Project:

It is used to update the details, deadline and whether the project is complete or not.

Update Project		
Route: /update/:projectid		
Function: UpdateProject		
Method	PATCH	

Request	{
	"iscomplete": "Boolean",
	"details": "String",
	"deadline": "String"
	}
		This is the update field. All are optional.

Response	Success:200	Fail:404

	"ProjectSuccessfully Updated" "Congratulation Project successfully completed	"You are not the manager". "No such project exists".

Table 16 Update Project Route

3.4.4.5.6 Get ID

It is used to obtain user id

Get ID Route: /project/getid Function: GetId		
Method	GET	
Response	Success:200	Fail:404

	{
	Userid: objectid
	}
		"Failed to get ID"

Table 17 Get ID Route

3.4.4.5.7 Check Member

It is used to check whether the member is found or not

Check Member Route: /ismember/:projectid Function: CheckMember		
Method	GET	
Response	Success:200	Fail:404

	{
	flag: boolean
	}
		"No member found"

Table 18 Check Member Route

3.4.4.5.8 Check Creator

This function is used to check the creator of the project.

Check Creator Route: /iscreater/:projectid Functions: CheckCreator		
Method	GET	
Response	Success:200	Fail:404

	{
	flag: boolean
	}
		"Creator not found"

Table 19 Check Creator Route

3.4.4.6 Discussion Route(/api/chat):

3.4.4.6.1See Message:

This function is used to view the sent message.

See Message Route: /see/:projectid Functions: SeeMessage		
Method	GET	
Response	Success:200	Fail:404
	List: Object (sends list containing all the messages of the project along with sender's name and time)	"You are not the member of the project" "Project with given ID does not exist"

Table 20 See Messages Route

3.4.4.6.2 Send Message:

This function is used to send the message in the discussion tab.

Send Message Route: /send/:projectid Function:		
Method	PATCH	

Request	{
	"name": “String”,
	"details": “String”,
	}
	
	Message details is asked	

Response	Success:200	Fail:404
	“Successfully Send The Message”	“You are not the member of this project” “Project with given ID does not exist”

Table 21 Send Message Route

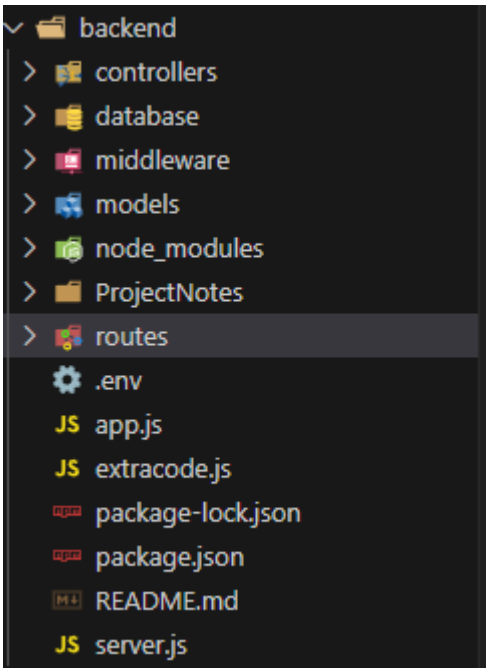

Figure 9 Backend Directory Structure

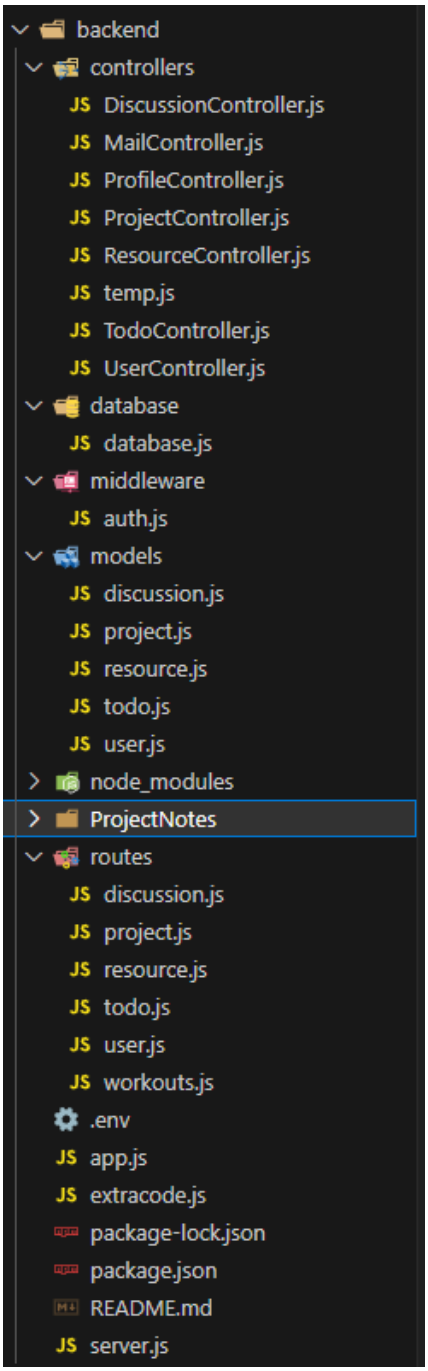

Figure 10 Backend Extended Directory Structure

3.5 Web Hosting:

We have hosted our web application using the free hosting service provided by "render.com".

After the both frontend and backend applications were developed and connected. We generated the 'build' folder containing the optimized and bundled files that are ready for deployment in a production environment. The build folder along with the backend folder was pushed to the GitHub repository named "project hub" which was then connected to the "Render.com" account. We then deployed the build as a static site with the URL "https://projecthub-78g5.onrender.com" while the backend server was named "project observer" and deployed as webservice at "https://projecthubserver.onrender.com".

After successful deployment, the base endpoint of both frontend and backend servers was changed from localhost to the above URLs. When the applications were redeployed we got the running web application

3.6 Flowcharts and Class Diagram

3.6.1 Program Flow

The program flow of the frontend is shown in the figure. The website starts with a login/signup page and is redirected to the dashboard page if no error occurs. From Dashboard, the user has the option to create, join a project, log out and go to the profile page and open a certain project. After clicking the project, todo, resources, and discussion page can be accessed. The individual progress can also be seen on the project landing page. In todo pages, the task assigned to other members and to the user can be seen. If the user is the head of the project, s/he can create a task and change the category of the task and set the task as completed. The category of the task also is changed by the team members to whom the task is assigned. On the resources page, the list of resources uploaded by the team member can be seen. The user can also update the resources. On the profile page, the details of the user can be seen and finally, on the Discussion page, the discussion among the team members can be seen and the user can also send a message about the project.

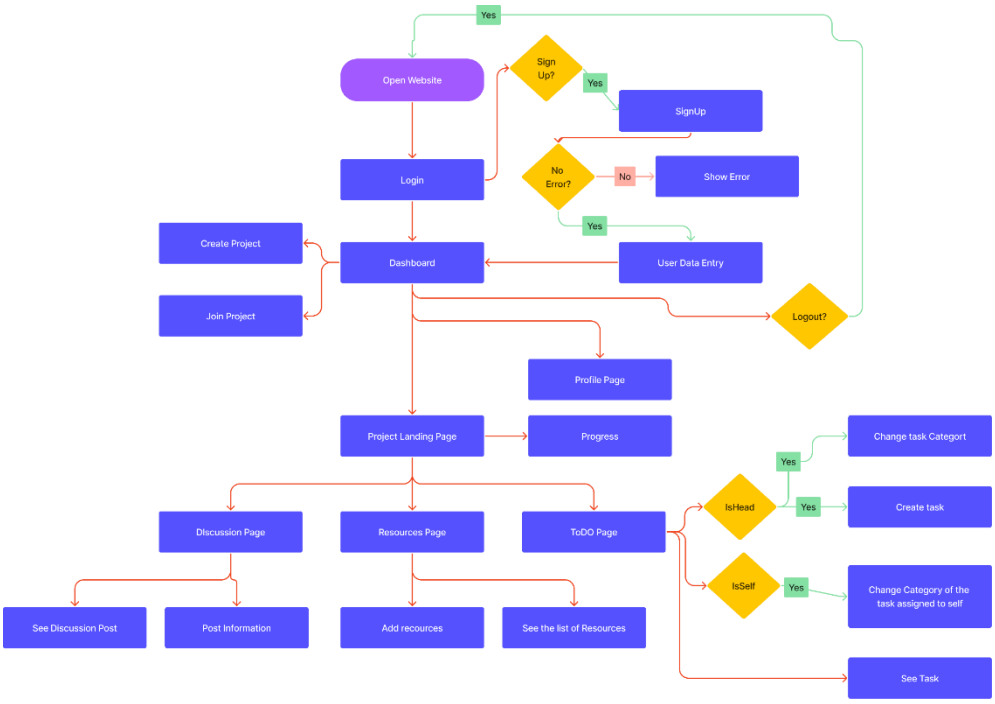

Figure 11 Program Flow

3.6.2 Database Entries

The data that are sent to the database from the following action is shown in the figure below:

Figure 12 Database Entries

3.6.3 Database Extraction

The data that are extracted from the database while exploring the website are mentioned below:

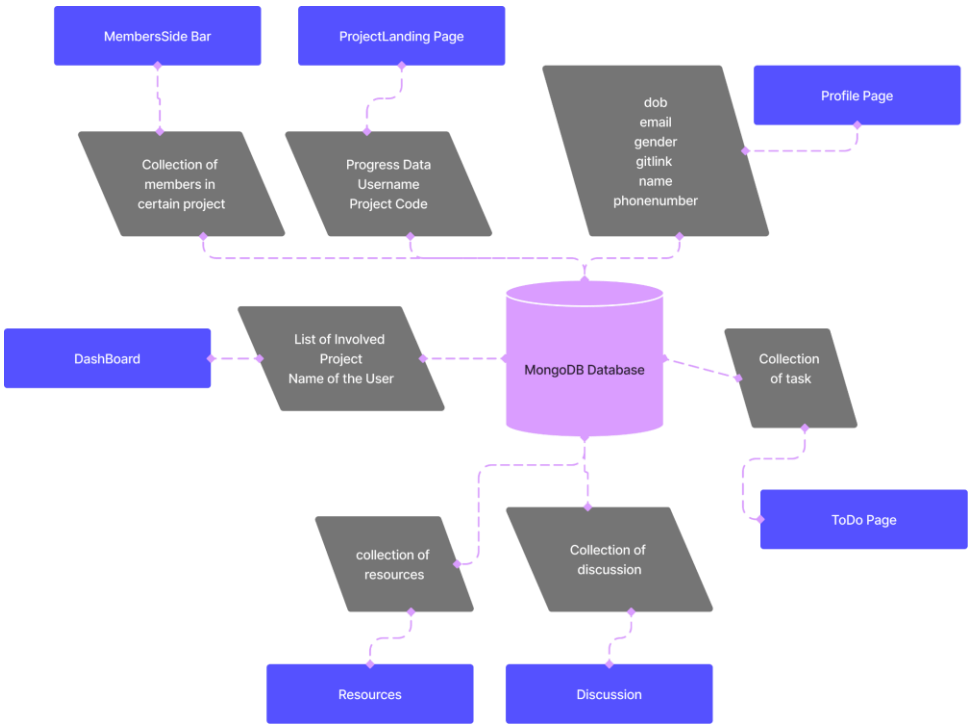

Figure 13 Database Extraction

3.6.4 UseCase Diagram

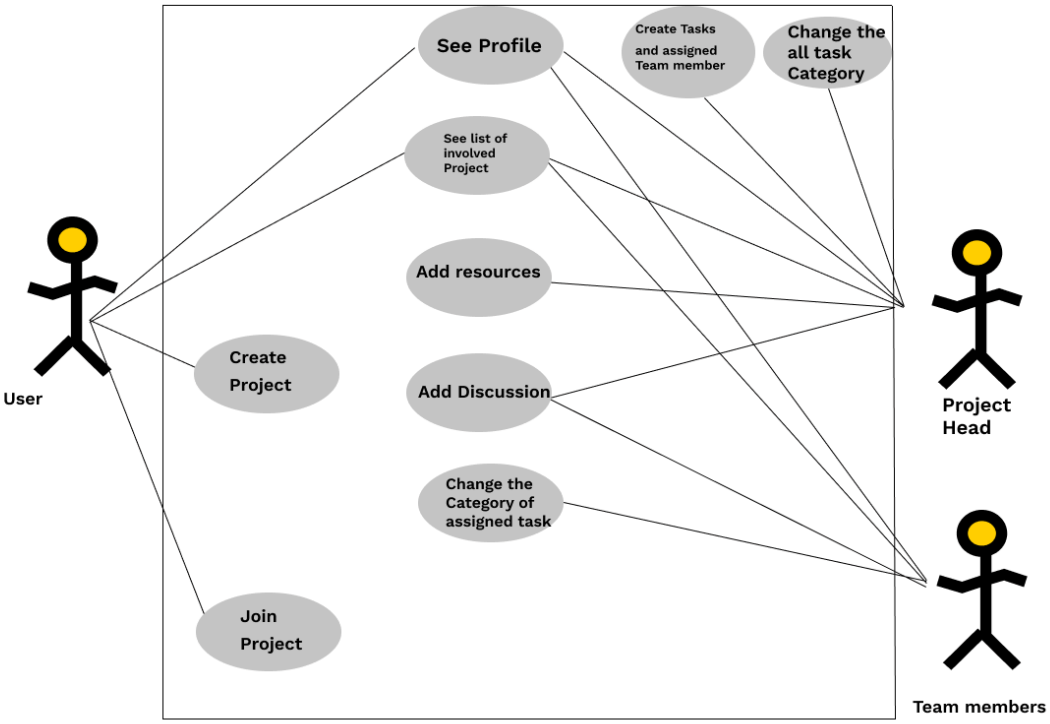

Figure 14 Use Case Diagram

3.7 Packages Used

We have used the following packages in the frontend to fulfil our needs for the design:

Packages	Uses
@emotion/react	CSS styles
@emotion/styled	Styled components
@material-ui/core	UI components

@mui/icons-material	Material icons
@mui/material	Material UI
@testing-library/jest-dom	Testing library
@testing-library/react	React testing
@testing-library/user-event	User interactions
js-cookie	Cookies handling
react	UI rendering
react-datepicker	Datepicker component
react-dom	DOM rendering
react-icons	Icon library
react-loader-spinner	Loader component
react-router-dom	Routing
react-scripts	React scripts

reactjs-popup	Popup component
web-vitals	Metrics tracking

Table 22 Frontend Packages

- **Backend:**

ExpressJS Library	Version
body-parser	^1.20.0
bcrypt	^5.1.0
cors	^2.8.5
dotenv	^16.0.3
express	^4.18.2
jsonwebtoken	^9.0.0
mongoose	^7.1.0
nodemon	^2.0.22
shortid	^2.2.16
validator	^13.9.0

Table 23 Backend Packages

Body-parser: This is a middleware module used in Node.js applications to extract the body portion of an incoming HTTP request.

Bcrypt: This module is used for hashing the user password.

cors: This module is used to enable CORS support.

dotenv: This module loads environment variables from the .env file.

express: Express is a fast, unopinionated, minimalist web framework for Node.js.

jsonwebtoken: This library is used for creating, signing, and verifying JWTs in Node.js applications.

mongoose: This ODM library enables straightforward integration to MongoDB databases in Node.js applications using JS objects.

nodemon: This module automatically restarts the node application when file changes in the directory are detected.

shortid: This module is used to obtain short non-sequential unique project code.

validator: This library provides a collection of validation functions for common data types and input validation tasks.

Chapter 4. System Requirement Specifications

This section specifies the requirements of the developed system. It may include software specifications and hardware specifications.

4.1 Hardware specifications

- **CPU:** Entry level SOC
- **RAM:** 2GB of RAM Recommended
- **System Memory:** 25 MB of free space

4.2 Software Specifications

Our website is hosted on the web so any desktop connection to the internet and having a browser can load our website.

Chapter 5. Discussion on the achievements

5.1 Challenges

5.1.1 Dynamic reload

The main challenge on the front end was dynamic reload. The data was extracted from the API but the data is only shown after reloading the website. The data was not uploaded while adding/updating tasks and joining/creating projects. The data was updated in the backend but the manual reload was needed for data to be shown. This problem is solved by using forced reload with the help of the useEffect and useReducer function of the react which automatically calls the API after the data is updated in the background.

5.1.2. Backend Challenges:

One of the main challenges on the backend was CORS(Cross-Origin Resource Sharing) error as the backend server was not receiving data from the frontend. Another major challenge was changes in schemas time and again and differences in the structure of new and old documents. This caused a lot of hurdles and needed time and again flushing of the database.

5.2 Features

5.2.1 Login and registration:

Registration and login systems have been implemented. On registration and log in each user is provided with a unique token which is used to identify and authenticate the person in future use. The token expires after 3 days which will provide security to the user. After the token expires the user has to login again to access the website.

5.2.2 Project Creation and Joining:

There is a system to create a project and join the said project by using the unique code generated.

5.2.3 Resource Collection:

There is a service for users to upload links to resources related to the project of which they are members.

5.2.4 Progress Bar:

There is a progress bar for every project created that tracks the projects/tasks completed based on the “completed” tag and points allocated to it.

5.2.5 User Profile:

The user profile page contains information about users such as email address, phone number and gitlink.

5.2.6 Group Chat:

This service allows members of the same project to post and see messages related to the project .

5.2.7 Assignments:

This service allows project managers to assign tasks to be completed to members of the project with deadlines.This service categories the tasks to either of following tags:BackLog,ToDo ,InProgress,Review and Completed. Only assigned member can change to other tags except Completed ,as only manager is allowed to verify whether a task is completed or not.

Chapter 6. Project Planning and Scheduling

Being new to MERN stack web development, we had to adapt ourselves by learning it first and then practicing. Since we are completed this project in 9 weeks' time, we decided to divide the project period as:

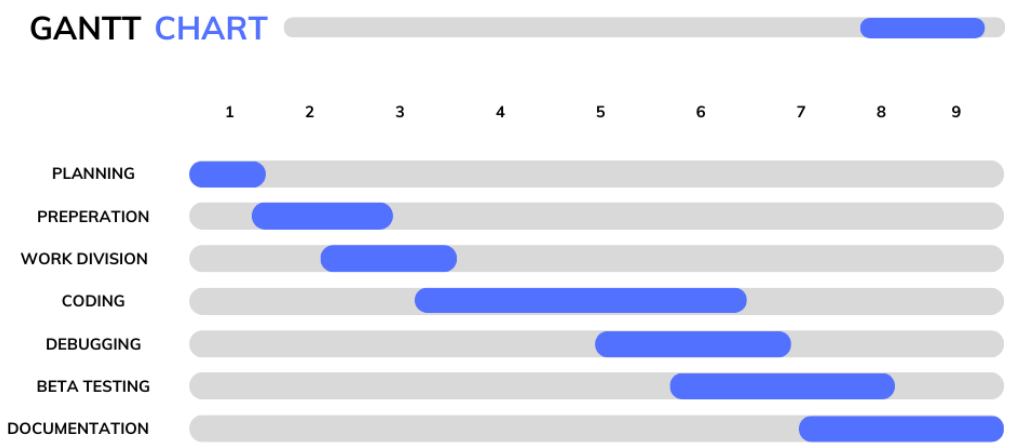

Figure 15 Gantt Chart

Chapter 7. Conclusion and Recommendation

Summarise your achieved/unachieved goals with valid reasoning. The conclusion should not contradict with the objectives of the project.

7.1 Limitations

- Users can’t see the profile page of other users.
- There is no system to delete completed projects.
- The completed tasks are not deleted but rather stay under the completed tag.
- The site isn’t as robust as desired.

7.2 Future Enhancement

- Email containing code to join projects can be sent.
- Email can be sent to reset the forgotten password.
- Improvise the chat system with socket.io library
- Sort the todos and projects based on the deadlines.

References

1. (n.d.). Trello: Manage Your Team's Projects From Anywhere. Retrieved February 14, 2023, from https://trello.com
2. (n.d.). Manage your team's work, projects, & tasks online • Asana. Retrieved February 14, 2023, from https://asana.com/
3. (n.d.). Express - Node.js web application framework. Retrieved February 14, 2023, from https://expressjs.com/
4. *Documentation*. (n.d.). Node.js. Retrieved February 14, 2023, from https://nodejs.org/en/docs/
5. Guide, S. (n.d.). *Getting Started – React*. React. Retrieved February 14, 2023, from https://reactjs.org/docs/getting-started.html
6. *Project Management Software: 10 Best Tools for Team Productivity*. (n.d.). ClickUp. Retrieved February 14, 2023, from https://clickup.com/project-management-software
7. *Welcome to the MongoDB Documentation*. (n.d.). MongoDB. Retrieved February 14, 2023, from https://www.mongodb.com/docs/
8. A. (2001, October 1). *Jira | Issue & Project Tracking Software | Atlassian*. Atlassian. https://www.atlassian.com/software/jira

APPENDIX

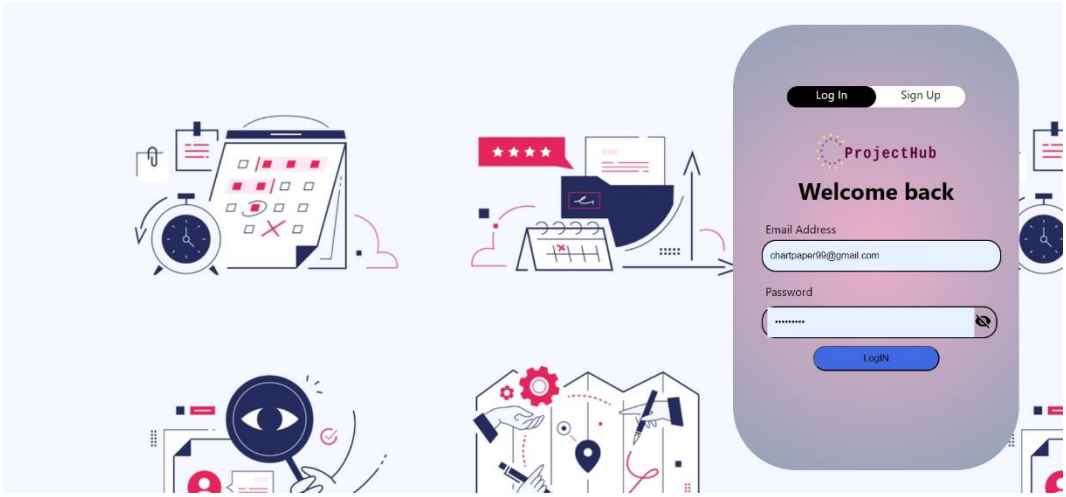

Figure 16 Login page

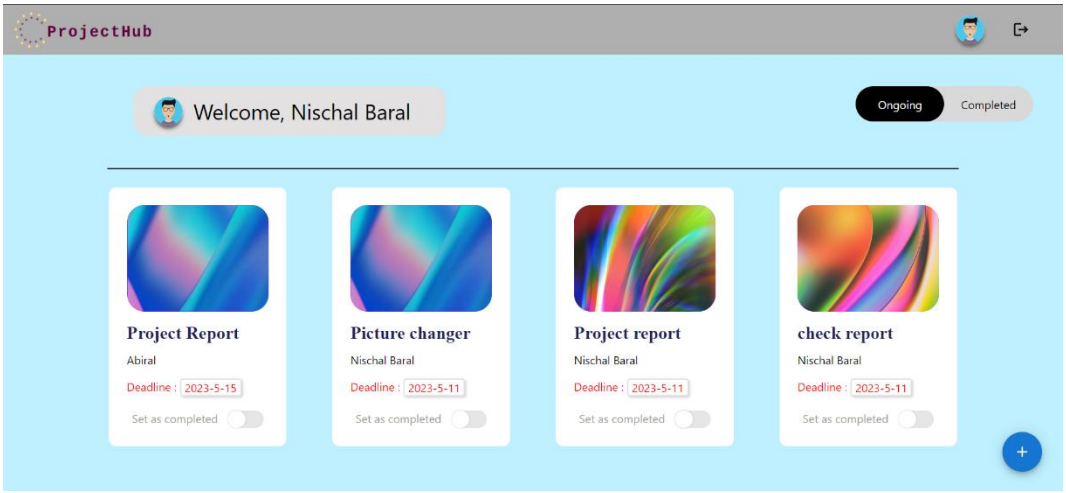

Figure 17 Dashboard Page

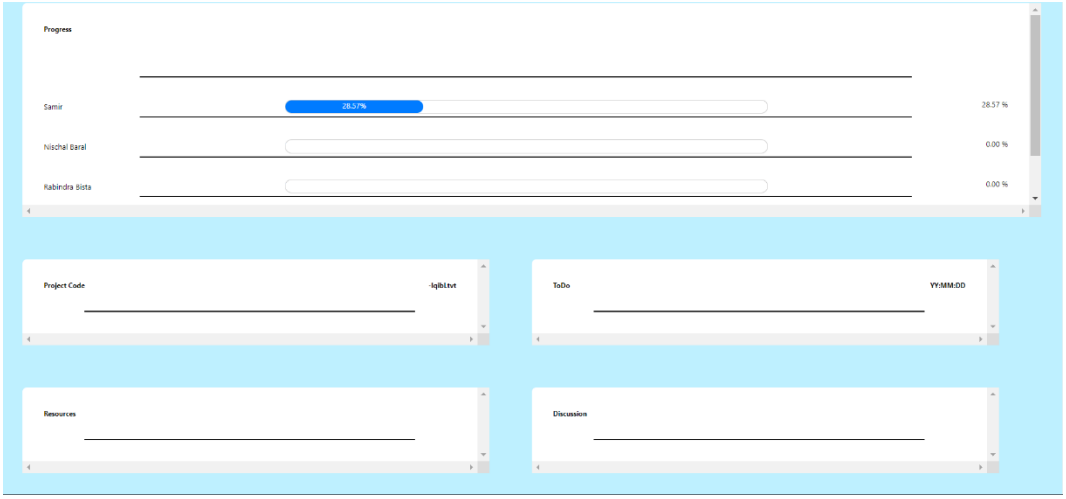

Figure 18 Project Page

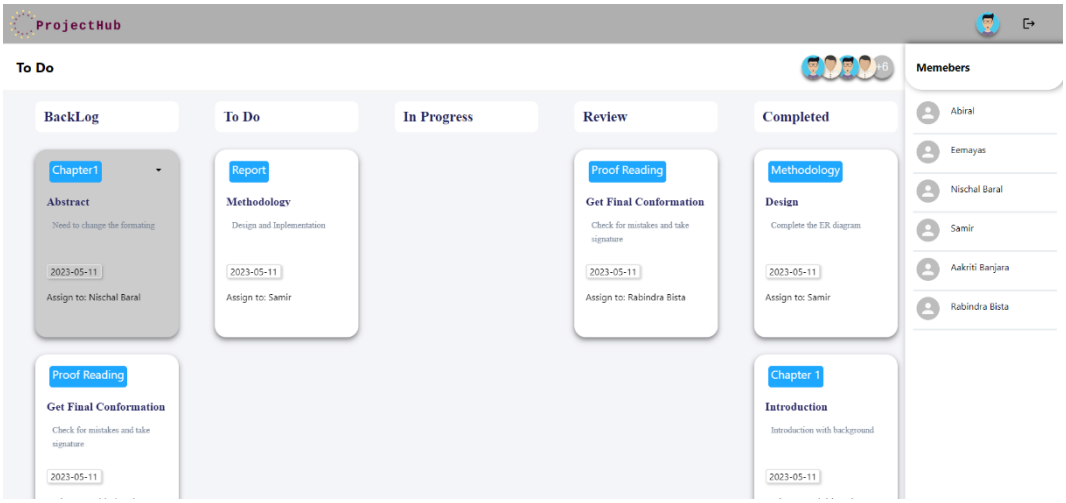

Figure 19 To Do Page

Figure 20 Resources Page

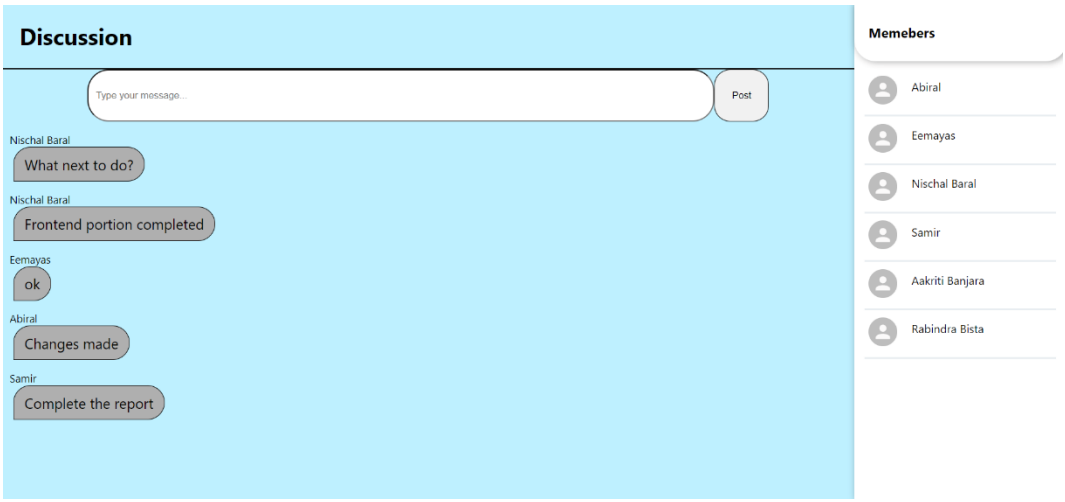

Figure 21 Discussion Page

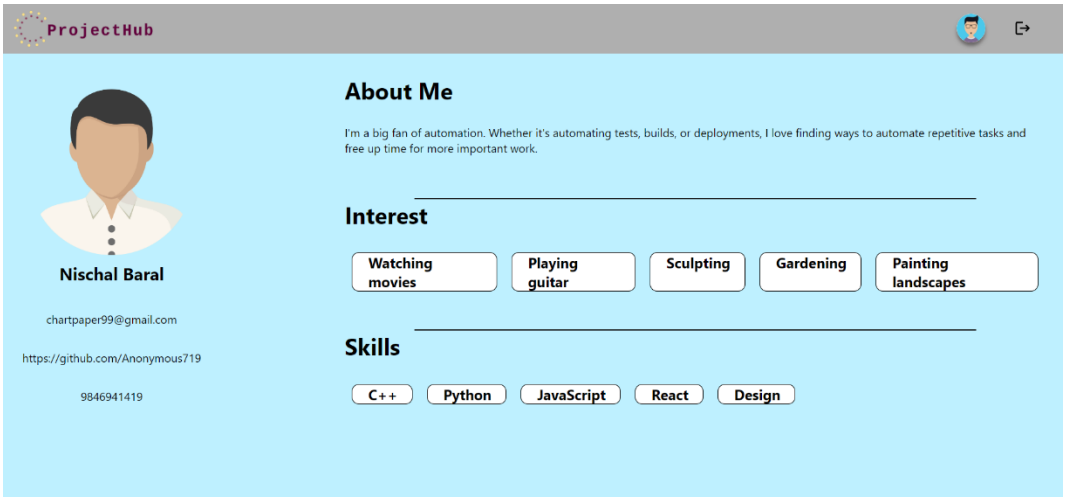

Figure 22 Profile Page