Dharmsinh Desai University, Nadiad

Faculty of Technology, Department of Computer Engineering

B.Tech. CE Semester – V

Subject: (CE-515) Advanced Technologies

Project Title:

Project Management System

By:

Parmar Nevil M. (Roll No: CE092 ID: 18CEUBG023)

Guided By:

Prof. Prashant M Jadav

Dharmsinh Desai University, Nadiad

Faculty of Technology, Department of Computer Engineering

CERTIFICATE

This is to certify that Advanced Technologies project entitled “Books And Mobile Phone Reselling System” is the bonafied report of work carried out by

1. **Nevil Parmar M. (Roll No. CE092 , ID: 18CEUBG023)**

Of the Department of Computer Engineering, Semester V, academic year 2019-20, under our supervision and guidance.

Guide	HOD
Prof. Prashant M Jadav	**Dr. C. K. Bhensdadia**
Assistant Professor of Department of Computer Engineering, Dharmsinh Desai University, Nadiad	Head of the Department of Department of Computer Engineering, Dharmsinh Desai University, Nadiad

Table of Contents

Abstract / Overview

This project is a tool to help in managing projects. It is more useful in current market situations where an organization is not close to a door or a city or a nation. In this case sharing documents & data related to a project from one corner of the world to another by using the internet makes our work easy. But still it is unmanaged to manage this work. We are making this tool Online Project Management System (OPM) to manage various ongoing projects in the firm and users can work/manage their tasks.

Introduction

Purpose

To develop a fully functional and user interactive online tool which can enhance and help various project management users to manage and compile their work efficiently and productively.

Any firm can manage their projects by breaking them into various small tasks which can be assigned to users and can be monitored at the same time. Users can comment and share the necessary documents with others related to the project.

Scope

- Create different users with varied roles and scopes.
- Confirm each member by providing activation codes.
- Manage all project details like tasks, deadlines, team members and resources.
- Assign different tasks to different members.
- Provide documentation to the members about the tasks being added
- Update all members about new proceedings in the project.
- Bind all the information provided by the team members at one place and show it to all others.
- Maintain start date and end date of each task
- Maintain the overall timeline of the project.

Technology used

- Cascading Style Sheet (CSS 3 + SCSS) for styling the HTML pages.
- JavaScript for providing dynamic content for the HTML pages.

- Jquery to provide dynamic data to the request pages.
- Bootstrap4 for styling the HTML pages using predefined classes of bootstrap.
- Angular for frontend.
- Express.js for routing.
- Node.js & mongoose for backend.
- MongoDB as a database manager to add, update and fetch data from the database using Node.js mongoose module.
- Akita state management
- UI modules :
    - TailwindCSS
    - Angular CDK drag and drop
    - Ng-zorro UI component : tooltip, modal, select, icon and more.
    - ngx-quill

Platform used

- localhost:4200 for Angular
- localhost:3000 for Node js
- mongodb://localhost:27017

Tool Used

- Github : As a version control method and to manage project
- Visual Studio Code : Development IDE
- MongoDB Compass : GUI tool for MongoDB
- Postman : tool to test calls to APIs

Definition, Acronyms and Abbreviations

OPM : Online Project Management System
Admin : Administrator

PM : Project Manager

HTML: Hyper text markup language

XHTML: Extensible Hypertext markup language

HTTP: Hypertext transfer protocol

Software Requirement Specification

Functional Requirements

Users Module:

- Administrator
- Project Manager
- Team Members

General Functional Requirements:

R.1 Manage Authentication

R.1.1 Login

Input: Username and Password

Output: Success or failure

Description: User can use this page to login to the system by entering personal credentials (username and password) and will be redirected to a home page to use the features of the website.

Constraints: Username must be text of at least 6 characters in length. Password must be at least 8 characters in length.

Process: Check the username and matching password in the database. If found in the database then redirect the user to his/her home page.

Exceptional Scenario: Show error message and ask the user for credentials again. If the user attempts three consecutive failed logins, then deactivate the user account and show notification to the user. User can contact the admin for activating his/her account.

Precondition: user must be registered on the website.
Postcondition: user is logged in to the website and a new session is associated with the client connection.

R.1.2 Signup

Input: User details

Output: Success or failure

Description: User can use this page to sign up to the system by entering personal credentials (username, first name, last name, phone no., email and password) and will be redirected to a home page to use the features of the website.

Constraints: Username must be text of at least 6 characters in length. Password must be at least 8 characters in length.

Process: Check the username is not already used in the database. If found in the database then redirect the user to sign up again with a proper error message.

Exceptional Scenario: User is already registered to the system. In which case an appropriate message should be displayed and the user should be redirected to the Login page.

Precondition: user must not be registered on the website.

Postcondition: user is logged in to the website and a new session is associated with the client connection.

R.1.3 Change the password

Input: User details [User email]

Output: Success or failure message

Description: User can use this page to change the password by entering personal credentials (email) and will be redirected to a home page to use the features of the website.

Constraints: email must be text of at least 6 characters in length. Password must be at least 8 characters in length.

Process: Check the email is already registered in the database. If found in the database then redirect the user to update the password page again with a proper success message.

Exceptional Scenario: User is not already registered to the system. In which case an appropriate message should be displayed and user should be redirected to the Register page.

Precondition: user must be registered on the website.

R.1.4 Search issues

Input: keyword to be searched

Output: Issues satisfying search criteria

Description: Users can search any issue by keyword from the dashboard.

Precondition: user must be logged in into the website.

R.1.5 Filter issues

Input: user input

Output: issues satisfying criteria

Description: User can filter out issues from the dashboard. Like filter issues with status “DONE/COMPLETE”.

Precondition: user must be registered on the website.

Functional Requirements of Administrator:

R.2 Manage Projects

R.2.1 Create new project

Input: A form will open, necessary information will be entered like project name, key etc.

Output: Success or failure message

Description: when admin fills necessary details about the project, a new project will be created for the company.

Constraints: projects with the same name should not be there in the company and the length of the key should be min of 3 characters.

Process: A query will be fired to the database and an invitation email will be sent to the desired project manager.

Exceptional Scenario: User has entered the name of a project which is already registered into the system. In which case an appropriate error message should be displayed.

Precondition: admin must be logged in the system.

Postcondition: On success, a new dashboard with the project name will be created and the user will be redirected to the homepage of the website.

R.2.2 Invite new project manager

Input: email / username of the project manager

Output: Success message displaying “An email invitation has been sent to the PM”
Description: an invitation email will be sent to the project manager for confirmation.

Constraints: entered email should be valid and active
Process: An invitation email will be sent to the Project Manager. PM will confirm the request. Project information will be completed and status will be updated to ready.

Exceptional Scenario: email is invalid, in which case an appropriate error message will be displayed to the user and user will be redirected to edit the page to enter an email again.

Precondition: admin must be logged in the system.

Postcondition: Success message will be displayed, and admin will be redirected to the team members' page.

R.2.3 Manage Project manager

R.2.3.1 Remove project manager

Input: User selection

Output: Success message
Description: Project manager will be removed from the particular assigned project

Precondition: admin must be logged in the system.

Postcondition: Success message will be displayed, and admin will be redirected to the team members' page.

R.2.3.2 update project manager details

Input: project manager details

Output: Success message
Description: Project manager details will be updated from the particular assigned project .

Precondition: admin must be logged in the system.

Postcondition: Success message will be displayed, and admin will be redirected to the team members' page.

Functional Requirements of Project Manager:

R.3 Project Manager

R.3.1 Manage team members

R.3.1.1 Invite new team member

Input: email / username of the team member

Output: Success message displaying "An email invitation has been sent"
Description: an invitation email will be sent to the team member for confirmation.

Constraints: entered email / username should be valid and active

Process: An invitation email will be sent to the team member. Member will confirm the request. Project information will be updated accordingly.

Exceptional Scenario: email is invalid, in which case an appropriate error message will be displayed to the user and user will be redirected to edit the page to enter an email again.

Precondition: Project Manager must be logged in the system.

Postcondition: Success message will be displayed, and PM will be redirected to the team members' page.

R.3.1.2 remove team member

Input: User selection

Output: Success message displaying "Team members has been removed"

Precondition: Project Manager must be logged in the system.

Postcondition: Success message will be displayed, and PM will be redirected to the team members' page.

R.3.2 Manage tasks

R.3.2.1 Create new task

Input: task details

Output: Success message displaying "a new task has been added" and the task board will be updated

Description: PM starts a new task and assigns team members to it.

Constraints: Team members should be registered to the system as well as well project

Process: A task will be added to the project and timelines will be displayed. Team members will update the task status and the project completion status will also be updated.

Exceptional Scenario: team member is not registered to the system, in which case appropriate error message will be displayed to the user.

Precondition: Project Manager must be logged in the system.

Postcondition: Success message will be displayed, and PM will be redirected to the team members' page.

R.3.2.2 Delete task

Input: User selection

Output: Success message displaying "a task has been deleted" and the task board will be updated

Description: PM deletes an existing task from the task board.

Constraints: Task should be pre added to the task board to perform deletion.

Exceptional Scenario: team member is not registered to the system, in which case an appropriate error message will be displayed to the user.

Precondition: Project Manager must be logged in the system.

Postcondition: Success deletion message will be displayed, and PM will be redirected to the team members' page.

R.3.2.3 Update the task details

Input: updated task details
Output: Success message displaying "a task has been updated" and the task board will be updated
Description: PM updates any task by providing updated details of the task
Process: A query to the database will be made to fetch the current task details and the updated details will be saved to the database.

Exceptional Scenario: fetching or updating of task details fails. In which case an appropriate error message will be displayed and the user will be redirected to the tasks page.

Precondition: Project Manager must be logged in the system.

Postcondition: Success message will be displayed, and PM will be redirected to the team members' page.

R.3.3 Manage messages/comments

R.3.3.1 Add new messages/comments

Input: messages/comments
Output: A new message / comment will be added to the task/issue.
Description: PM adds a new comment to the existing issue/task.

Precondition: Project Manager must be logged in the system.

Postcondition: Success message will be displayed, and PM will be redirected to the issue details page.

R.3.4 Provide documentation

R.3.4.1 Add new attachments

Input: attachment(s)
Output: A new attachment(s) will be added to the task/issue.

Description: PM adds a new attachment to the existing issue/task.

Precondition: Project Manager must be logged in the system.

Postcondition: Success message will be displayed, and PM will be redirected to the issue details page.

R.3.4.2 Remove attachments

Input: User selection
Output: attachment(s) will be deleted from the task/issue.
Description: PM removes attachment from the existing issue/task.

Precondition: Project Manager must be logged in the system.

Postcondition: Success message will be displayed, and PM will be redirected to the issue details page.

R3.5 Generate Reports

Input: User selection

Output: Report will be generated and will be available to download in the downloadable format.

Process: Report will be generated and the database entry will be made for the same.

Description: PM can generate reports for the existing project to see the various analytics.

Precondition: Project Manager must be logged in the system.

R3.6 Update project status/deadlines

Input: project details

Output: Success message

Description: PM can update the project status and its deadline

Precondition: Project Manager must be logged in the system.

Functional Requirements of Team member:

R.4 Team member

R4.1 Update issue/task status

Input: User selection

Output: Success message

Description: Team member can update the issue/task status.

Precondition: team member must be logged in the system.

R4.2 Create new comment/message

Input: comment/message

Output: Success message and comment will be added to the issue details page

Description: Team member can comment or add a message to the particular task.

Precondition: team member must be logged in the system.

Non-Functional Requirements

1. The system is resistant to faults within the system.
2. The system can be accessed from anywhere all the time.
3. The system is secured with attack protection , attack prevention by code side(Cross site scripting), encrypted communication channels(https)
4. The system can be accessed from mobile and computer devices.
5. Supports the number of users at the same time more than 100 people.
6. The system can be used easily.
7. The system can support the growth of the user base.

Design Documents

XML, DTD, XSD, XSLT.

- **XML**

```
<?xml version="1.0" ?>
<users>
<user id="u1">
<name>Nevil Parmar</name>
<email>Nevil Parmar</email>
<createdAt>Nevil Parmar</createdAt>
<updatedAt>Nevil Parmar</updatedAt>
<password>@1234</password>
<avatarUrl></avatarUrl>
</user>
</users>
```

- **DTD**

```
<?xml encoding="UTF-8"?>

<!ELEMENT users (user *)>

<!ELEMENT user
(name,email,createdAt,updatedAt,password,avatarUrl)>

<!ELEMENT name (#PCDATA)>

<!ELEMENT email (#PCDATA)>

<!ELEMENT createdAt (#PCDATA)>
```

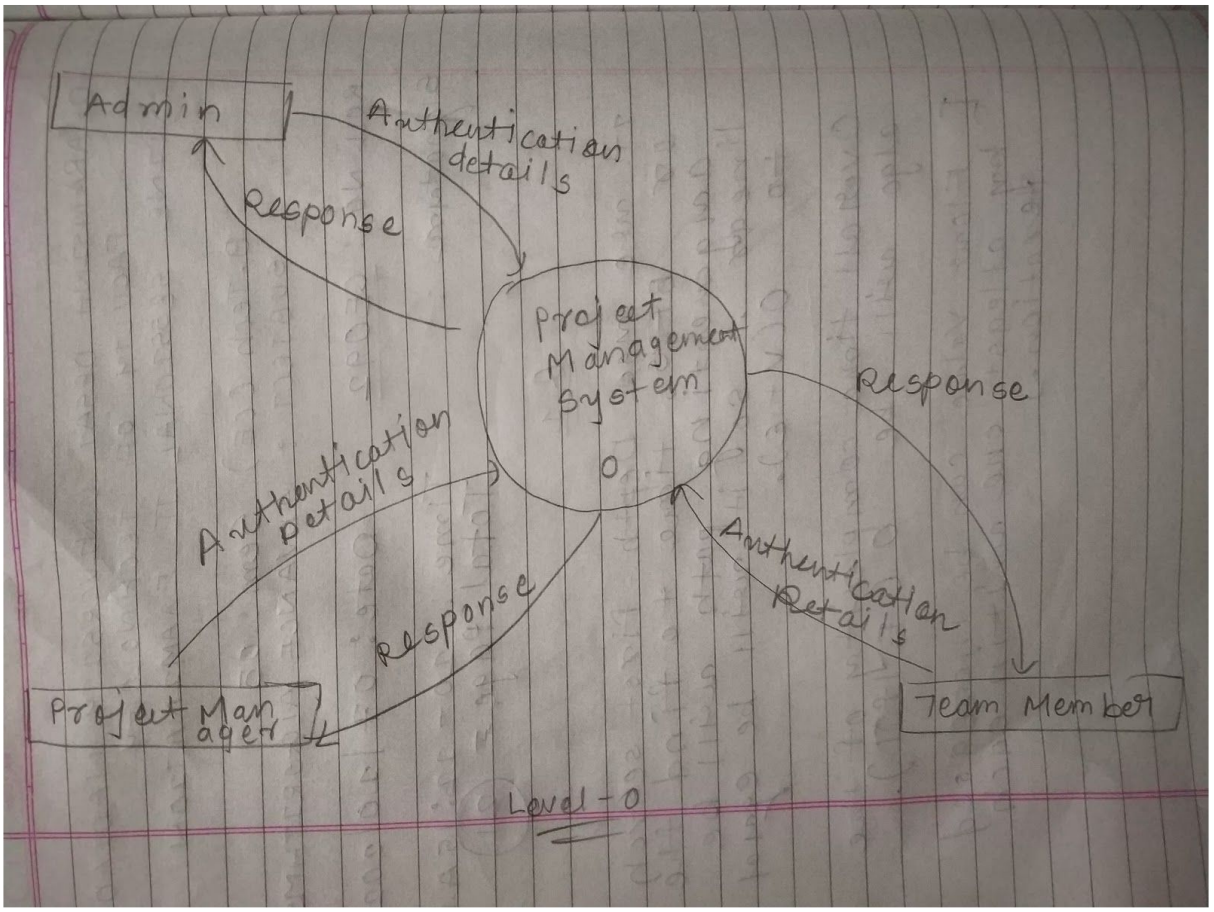
```
<!ELEMENT updatedAt (#PCDATA)>

<!ELEMENT password (#PCDATA)>

<!ELEMENT avatarUrl (#PCDATA)>
<!ATTLIST user id ID #REQUIRED>
```

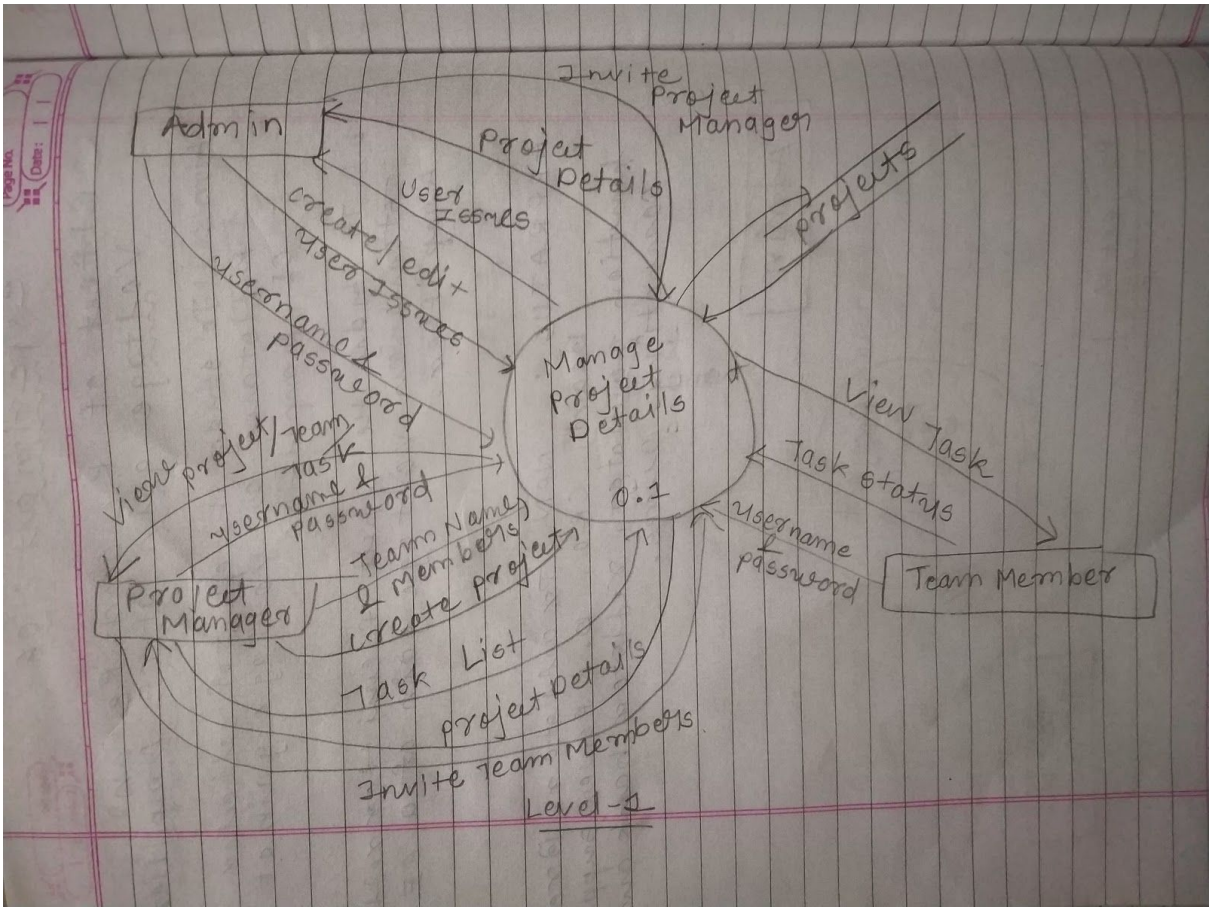
- **XSD**

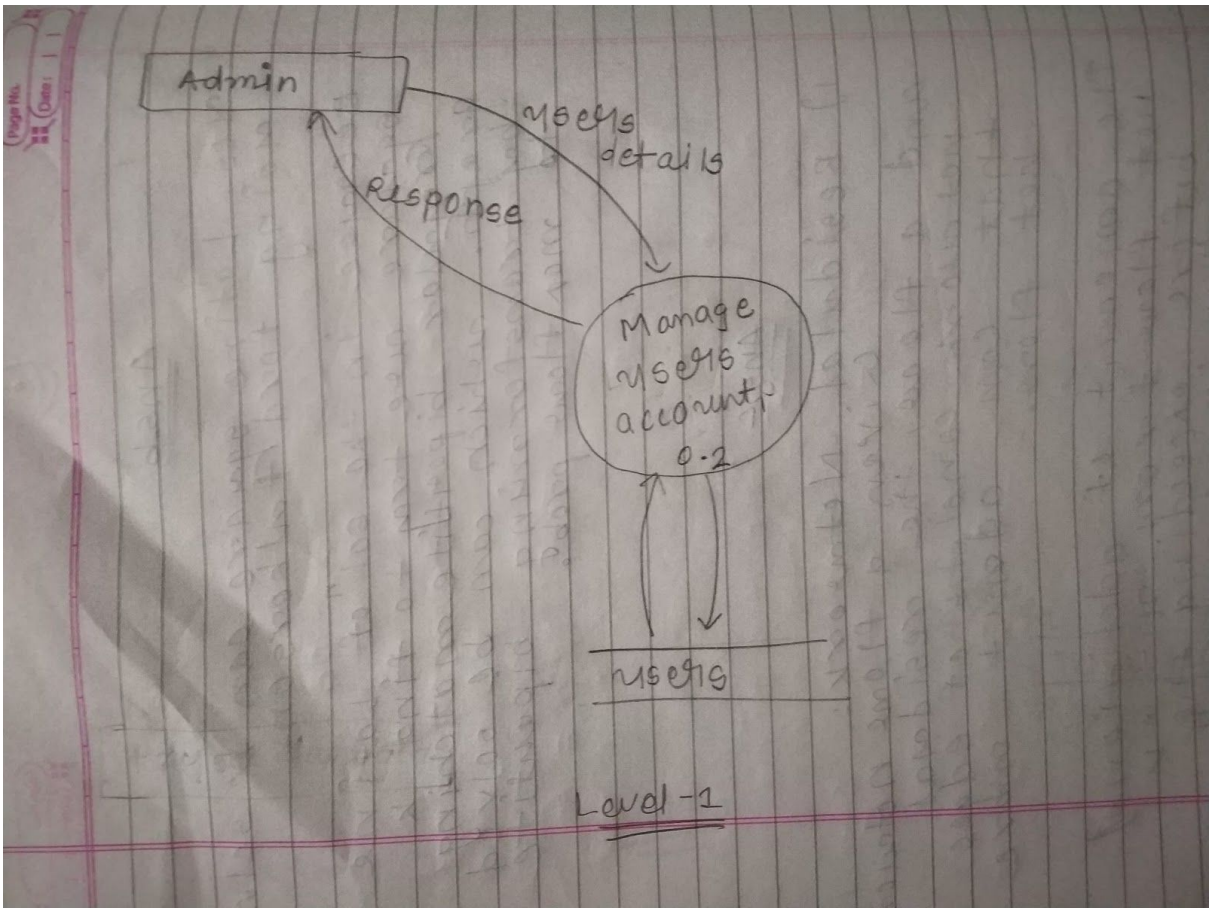
```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xs:element name="users">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="user"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="user">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="name"/>
        <xs:element ref="email"/>
        <xs:element ref="createdAt"/>
        <xs:element ref="updatedAt"/>
        <xs:element ref="password"/>
        <xs:element ref="avatarUrl"/>
      </xs:sequence>
      <xs:attribute name="id" use="required" type="xs:NCName"/>
    </xs:complexType>
```

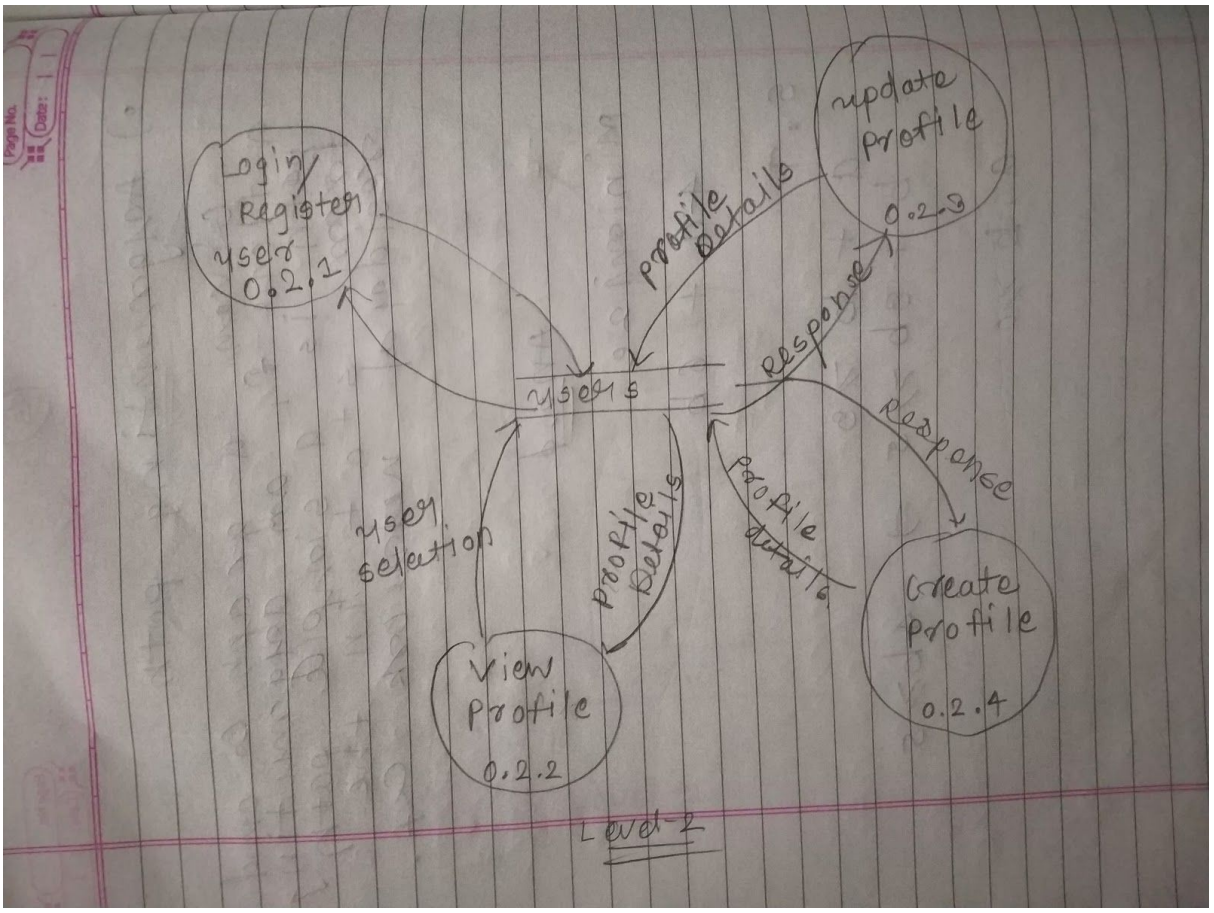
```
  </xs:element>
  <xs:element name="name" type="xs:string"/>
  <xs:element name="email" type="xs:string"/>
  <xs:element name="createdAt" type="xs:string"/>
  <xs:element name="updatedAt" type="xs:string"/>
  <xs:element name="password" type="xs:string"/>
  <xs:element name="avatarUrl">
    <xs:complexType/>
  </xs:element>
</xs:schema>
```

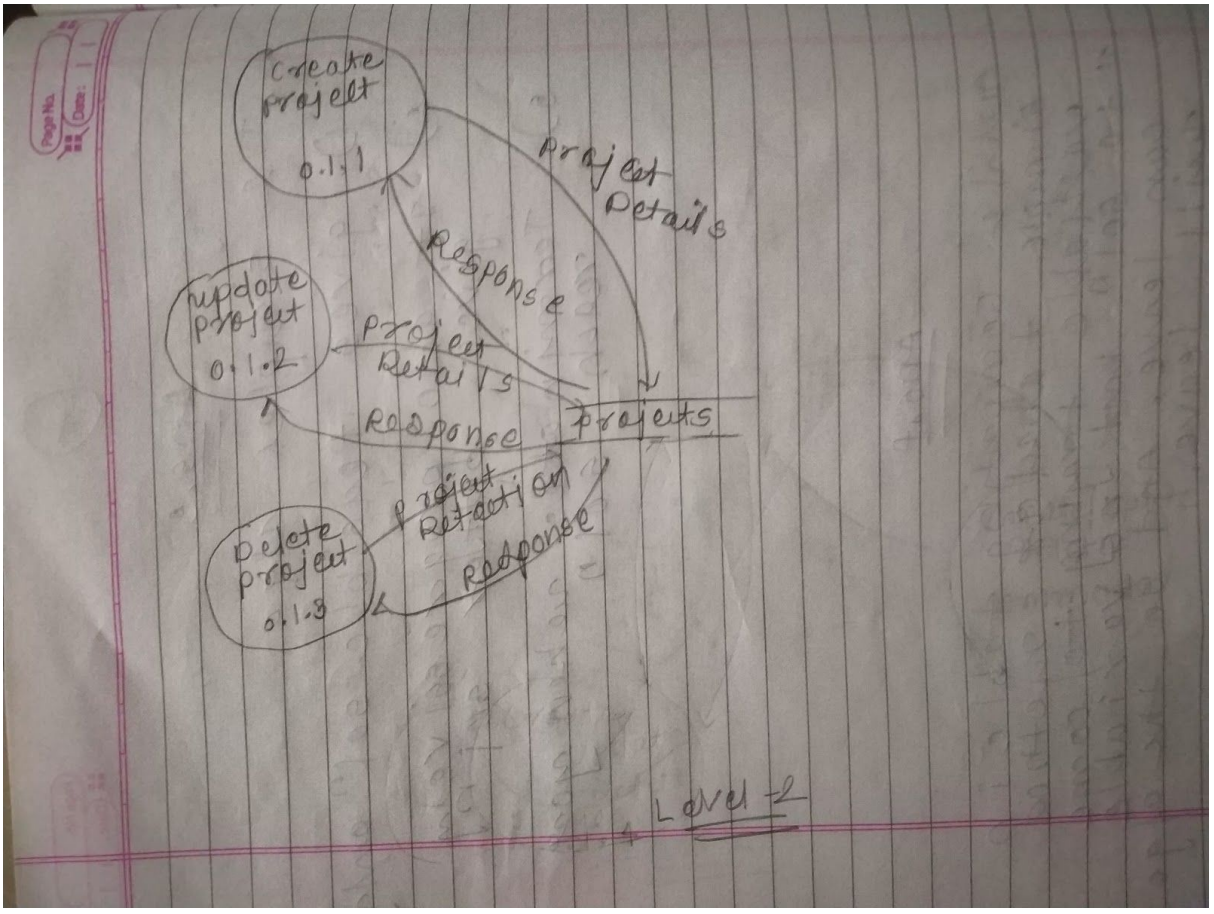
DFD Diagram

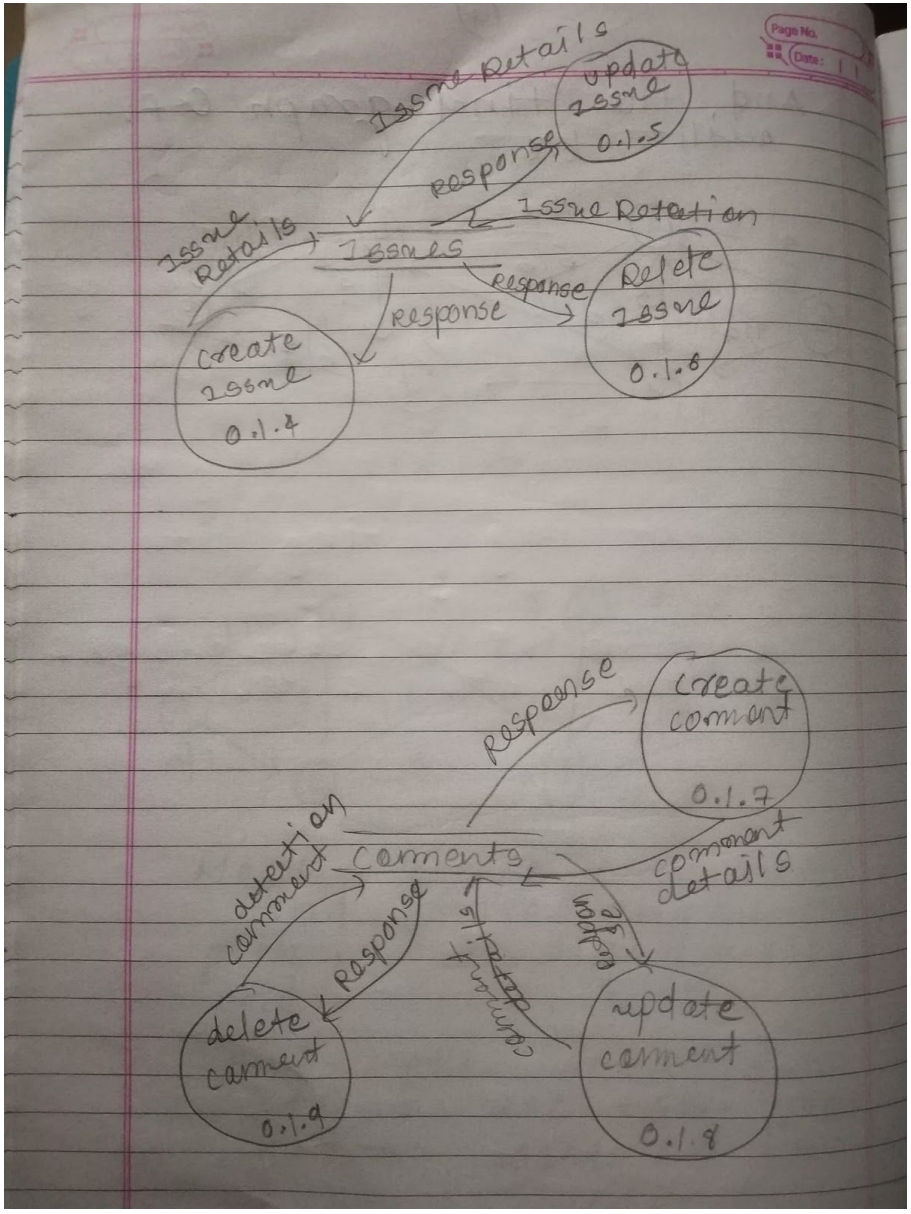



Admin
Invite Project Manager
Project Details
User Issues
create/edit User Issues
projects
username & password
Manage Project Details
View Task
Task status
view project/team Task
username & password
0.1
username password
Team Member
Project Manager
Team Name & Members
create projects
Task List
project Details
Invite Team Members
Level-0

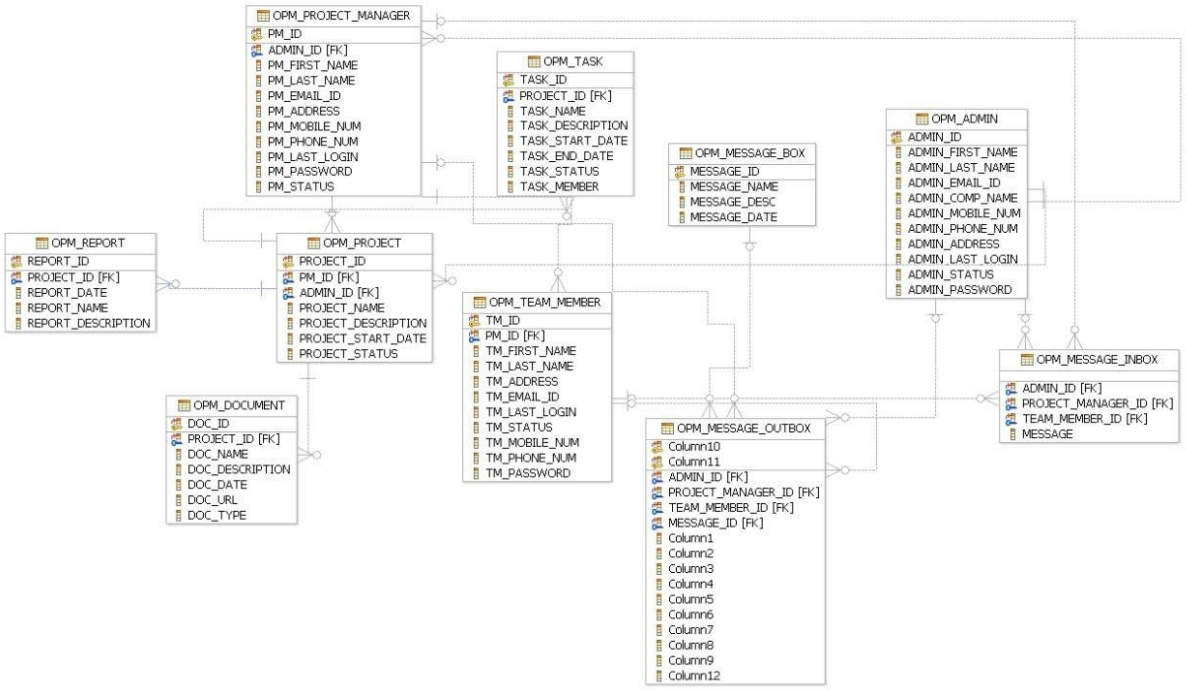

Admin
users details
response
Manage users accounts
0.2
users
Level -1


Login/
Register
user
0.2.1
update
Profile
0.2.3
Profile Details
response
users
Response
user
selection
Profile
Details
Profile
details
View
Profile
0.2.2
Create
Profile
0.2.4
Level-2


Create
project
0.1.1
Project
Details
Response
update
project
0.1.2
Project
Details
Response
Projects
Delete
project
0.1.3
Project
Retation
Response
Level-2


Issue Details
update issue
0.1.5
response
Issue Details
Issues
Issue Retation
response
response
Delete issue
0.1.6
create issue
0.1.4
response
create comment
0.1.7
detection comment
comments
comment details
Response
update comment
delete comment
0.1.9
0.1.8

E-R Diagram



Data Dictionary

Admin

Key	Name	Data type	Length	Nullable
	ADMIN_ID	INTEGER	4	No
	ADMIN_FIRST_NAME	VARCHAR	40	No
	ADMIN_LAST_NAME	VARCHAR	40	No
	ADMIN_EMAIL_ID	VARCHAR	40	No
	ADMIN_COMP_NAME	VARCHAR	40	No
	ADMIN_MOBILE_NUM	DOUBLE	8	No
	ADMIN_PHONE_NUM	DOUBLE	8	No
	ADMIN_ADDRESS	VARCHAR	225	No
	ADMIN_LAST_LOGIN	VARCHAR	225	No
	ADMIN_STATUS	INTEGER	4	No
	ADMIN_PASSWORD	VARCHAR	225	No

Constraint name	Key Type	Columns
CC1223450537843	Primary	ADMIN_ID

Project_Manager

Constraint name	Key Type	Columns
CC1223450896421	Primary	PM_ID
CC1229249555765	Foreign	ADMIN_ID ON HB.OPM_ADMIN

Key	Name	Data type	Length	Nullable
	ADMIN_ID	INTEGER	4	No
	PM_ID	INTEGER	4	No
	PM_FIRST_NAME	VARCHAR	40	No
	PM_LAST_NAME	VARCHAR	40	No
	PM_EMAIL_ID	VARCHAR	40	No
	PM_ADDRESS	VARCHAR	225	No
	PM_MOBILE_NUM	DOUBLE	8	No
	PM_PHONE_NUM	DOUBLE	8	No
	PM_LAST_LOGIN	VARCHAR	225	No
	PM_PASSWORD	VARCHAR	40	No
	PM_STATUS	INTEGER	4	No

TEAM_MEMBER

Key	Name	Data type	Length	Nullable
	PM_ID	INTEGER	4	No
	TM_ID	INTEGER	4	No
	TM_FIRST_NAME	VARCHAR	40	No
	TM_LAST_NAME	VARCHAR	40	No
	TM_ADDRESS	VARCHAR	225	No
	TM_EMAIL_ID	VARCHAR	50	No
	TM_LAST_LOGIN	VARCHAR	225	No
	TM_STATUS	INTEGER	4	No
	TM_MOBILE_NUM	BIGINT	8	No
	TM_PHONE_NUM	BIGINT	8	No
	TM_PASSWORD	VARCHAR	40	No

Constraint name	Key Type	Columns
CC1223451529687	Primary	TM_ID
CC1229249673046	Foreign	PM_ID ON HB.OPM_PROJECT_MANAGER

Project

Key	Name	Data type	Length	Nullable
	PM_ID	INTEGER	4	No
	ADMIN_ID	INTEGER	4	No
	PROJECT_ID	INTEGER	4	No
	PROJECT_NAME	VARCHAR	40	No
	PROJECT_DESCRIPTION	"LONG VARCHA...	32700	No
	PROJECT_START_DATE	DATE	4	No
	PROJECT_STATUS	VARCHAR	30	No

Constraint name	Key Type	Columns
CC1223451299968	Primary	PROJECT_ID
CC1229249476437	Foreign	ADMIN_ID ON HB.OPM_ADMIN
CC1229249523218	Foreign	PM_ID ON HB.OPM_PROJECT_MANAGER

Task/Issue

Key	Name	Data type	Length	Nullable
	PROJECT_ID	INTEGER	4	No
🔑	TASK_ID	INTEGER	4	No
	TASK_NAME	VARCHAR	40	No
	TASK_DESCRIPTION	VARCHAR	225	No
	TASK_START_DATE	DATE	4	No
	TASK_END_DATE	DATE	4	No
	TASK_STATUS	VARCHAR	30	No
	TASK_MEMBER	VARCHAR	30	No

Constraint name	Key Type	Columns
CC1223452393531	Primary	TASK_ID
CC1228370470343	Foreign	PROJECT_ID ON HB.OPM_PROJECT

Document

Key	Name	Data type	Length	Nullable
	PROJECT_ID	INTEGER	4	No
🔑	DOC_ID	BIGINT	8	No
	DOC_NAME	VARCHAR	40	No
	DOC_DESCRIPTION	VARCHAR	225	No
	DOC_DATE	VARCHAR	225	No
	DOC_URL	VARCHAR	225	No
	DOC_TYPE	VARCHAR	10	No

Constraint name	Key Type	Columns
CC1223454284468	Primary	DOC_ID
CC1229249128421	Foreign	PROJECT_ID ON HB.OPM_PROJECT

MessageBox

Key	Name	Data type	Length	Nullable
🔑	MESSAGE_ID	INTEGER	4	No
	MESSAGE_NAME	VARCHAR	100	No
	MESSAGE_DESC	VARCHAR	2000	No
	MESSAGE_DATE	DATE	4	No

Constraint name	Key Type	Columns
CC1224729429890	Primary	MESSAGE_ID

MessageInbox

Key	Name	Data type	Length	Nullable
	ADMIN_ID	INTEGER	4	Yes
	PROJECT_MANAGER_ID	INTEGER	4	Yes
	TEAM_MEMBER_ID	INTEGER	4	Yes
	MESSAGE	VARCHAR	1000	Yes

Constraint name	Key T...	Columns
CC1229249200781	Foreign	ADMIN_ID ON HB.OPM_ADMIN
CC1229249208765	Foreign	PROJECT_MANAGER_ID ON HB.OPM_PROJECT_MANAGER
CC1229249240484	Foreign	TEAM_MEMBER_ID ON HB.OPM_TEAM_MEMBER

MessageOutbox

Key	Name	Data type	Length	Nullable
	ADMIN_ID	INTEGER	4	Yes
	PROJECT_MANAGER_ID	INTEGER	4	Yes
	TEAM_MEMBER_ID	INTEGER	4	Yes
	MESSAGE_ID	INTEGER	4	Yes

Constraint name	Key Type	Columns	Use during optimization
CC1229249362296	Foreign	ADMIN_ID ON...	Yes
CC1229249379500	Foreign	PROJECT_MA...	Yes
CC1229249401750	Foreign	MESSAGE_ID ...	Yes
CC1229249415125	Foreign	TEAM_MEMB...	Yes

Reports

Key	Name	Data type	Length	Nullable
	PROJECT_ID	INTEGER	4	No
	REPORT_ID	INTEGER	4	No
	REPORT_DATE	VARCHAR	225	No
	REPORT_NAME	VARCHAR	40	No
	REPORT_DESCRIPTION	VARCHAR	225	No

Constraint name	Key Type	Columns
CC1223454392296	Primary	REPORT_ID
CC1229251419937	Foreign	PROJECT_ID ON HB.OPM_PROJECT

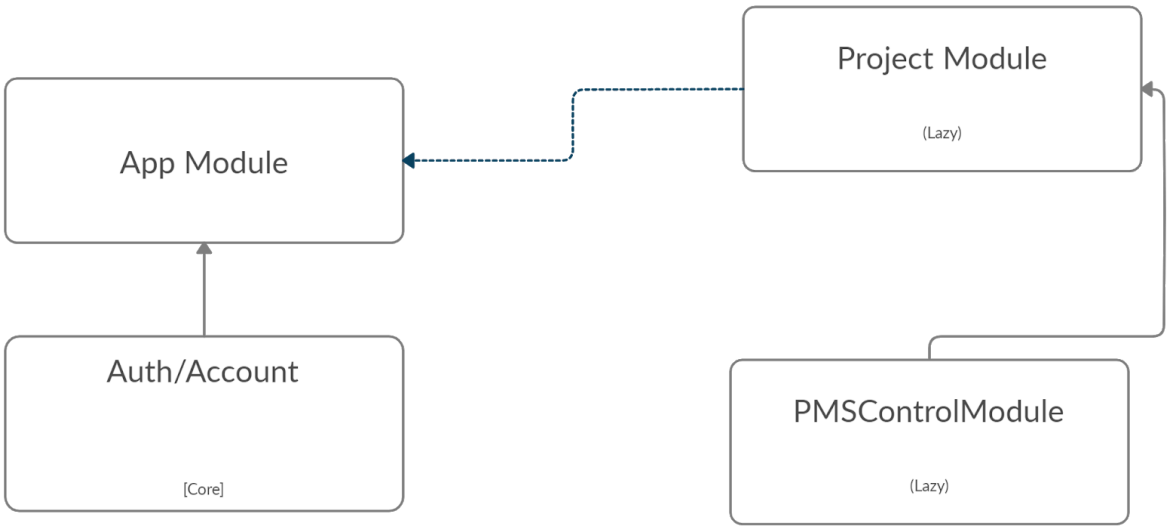
Implementation Details

The system consists of 2 modules mainly.

1. Account Module
    - Manages User login/registration
2. Project Module

    2.1 PMSControl Module [sub module]

- Manages projects, related issues and comments and various dashboard components.



Each module consists of several methods to implement the required functionality. Implementation is done using ExpressJs on top of NodeJs. Database used in these modules is MongoDB. Connection to database is using mongoose. The frontend is rendered using Angular and Bootstrap.

I have an app module that will import :

- Angular needed modules such as browsermodule, formsmodule etc.
- And I also configured the router to lazy load any modules only when I needed. Otherwise, everything will be loaded when I start the application. For instance, LoginModule when I open the URL at /login and ProjectModule when the URL is /project. Inside each module, I could import whatever modules that are required. Such as I need the PMSControlModule for some custom UI components for the ProjectModule

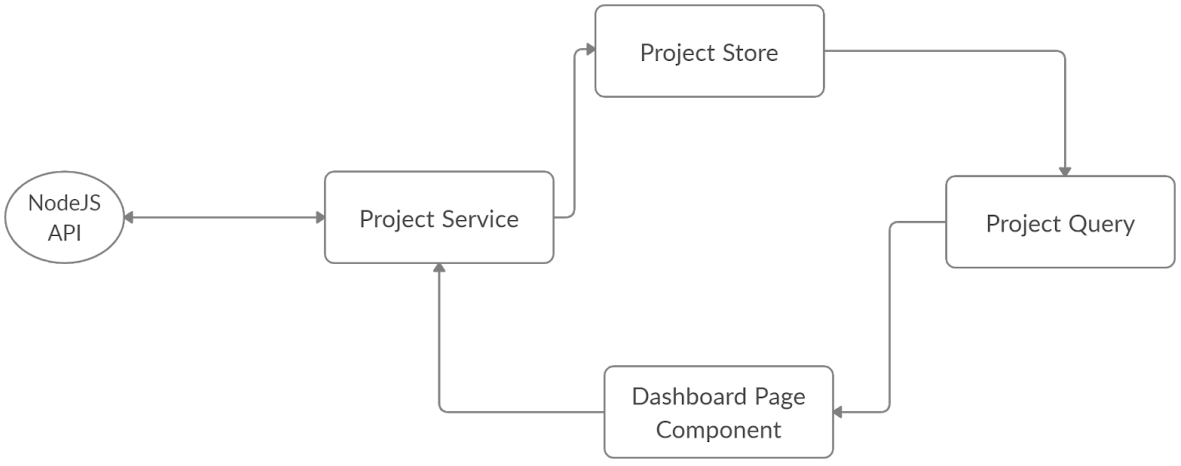
Account Module:

This module is the base for authentication and authorization to ensure the security aspect of the system.

It is the core module of the application that needs to be available on the whole platform.

It internally uses different roll type user models, services, authguard and interceptor to ensure the security of the system.

Project Module :



This module handles the flow of project records to and from the server and the akita store on the user machine. It internally uses PMSControlmodule as a submodule, which has the necessary UI components for the dashboard and their services to handle callbacks.

As I am using akita state management, I follow the Akita documentation for the data flow.
I set up a project state with initial data. The main heavy lifting part is the project service, it contains all the interacting with the project store. Such as after fetching the project successfully, I update the store immediately inside the service itself. The last lego block was to expose the data through project query. Any components can start to inject project query and consume data from there.
Which eventually reduces the API calls and provides better UI/State management to the user on the site.

Function Prototypes

Total 4 different types of backend APIs.

```
app.use('/user-api', userApi);
app.use('/project-api', projectApi);
app.use('/issue-api', issueApi);
app.use('/comment-api', commentApi);
```

```
router.post('/login', verifyToken, (req, res) => {
    let userData = req.body;
    console.log(userData.email);
    console.log(userData.password);
    users.findOne({ email: userData.email }, (err, user) => {
        if (err) {
            console.log(err)
        } else {
            if (!user) {
                res.status(401).send('Invalid Email')
            } else
            if (user.password !== userData.password) {
                res.status(401).send('Invalid Password')
            } else {
                let payload = { subject: user._id }
                let token = jwt.sign(payload, 'secretKey')
                res.status(200).send({ token, user })
            }
        }
    })
})
```

```
function verifyToken(req, res, next) {
    if (!req.headers.authorization) {
        return res.status(401).send('Unauthorized request')
    }
    let token = req.headers.authorization.split(' ')[1]
    if (token === 'null') {
        return res.status(401).send('Unauthorized request')
    }
    let payload = jwt.verify(token, 'secretKey')
    if (!payload) {
        return res.status(401).send('Unauthorized request')
    }
    req.userId = payload.subject
    next()
}

router.post('/register', (req, res) => {
    let userData = req.body
    let user = new users(userData)
    user.save((err, registeredUser) => {
        if (err) {
            console.log(err)
        } else {
            let payload = { subject: registeredUser._id }
            let token = jwt.sign(payload, 'secretKey')
            res.status(200).send({ token })
        }
    })
})
```

```
router.post('/comment/create/:id', (req, res) => {
    var newcomment = new comments(req.body);
    newcomment.save(async(err, result) => {
        if (err) {
            console.log("error occured in project");
            res.send("error occured");
            res.status(400);
        } else {
            console.log(result);
            await issues.findByIdAndUpdate(req.params.id, {
                $push: {
                    "comments": newcomment._id
                }
            });

            res.status(200);
            res.json({ "message": "New comment Created Successfully" });
        }
    });
})
```

```
createIssue(issue: Issue) {
  issue.updatedAt = DateUtil.getNow();
  this._http
  .post<Issue>(`${this.baseUrl}/issue-api/issue/create/${_id}`, issue)
  .pipe(
    setLoading(this._store),
    tap((issue) => {
      this._store.update((state) => {
        const issues = arrayUpsert(state.issues, issue._id, issue);
        return {
          ...state,
          issues
        };
      });
      this.getProject();
    }),
    catchError((error) => {
      this._store.setError(error);
      return of(error);
    })
  )
  .subscribe();

  console.log(this._store);
}
```

```
router.post('/issue/create/:id', (req, res) => {
    var newIssue = new issues(req.body);
    newIssue.save(async(err, result) => {
        if (err) {
            console.log("error occured in issue create");
            res.send("error occured");
            res.status(400);
        } else {
            console.log(result);
            res.status(200);
            await projects.findByIdAndUpdate(req.params.id, {
                $push: {
                    "issues": newIssue._id
                }
            });
            res.json({ "message": "New Issue Created Successfully" });
        }
    });
})
```

```
router.post('/issue/delete/:pid/:iid', (req, res) => {
    issues.findOneAndRemove({ "_id": req.params.iid },
        async(err, result) => {
            if (err) {
                res.status(400);
                res.send("Unable to find an Issue");
            } else {
                console.log(result);
                res.status(200);
                await projects.findByIdAndUpdate(req.params.pid, {
                    $pull: {
                        "issues": result._id
                    }
                });

                await comments.remove({
                    "issueId": req.params.iid
                });

                res.json({ "message": "New Issue Removed Successfully" });
            }
        });
})
```

```
getProject() {
  this._http
    .get<Project>(`${this.baseUrl}/project-api/project/5f9b109e856c60c2c2d349a6`)
    .pipe(
      setLoading(this._store),
      tap((project) => {
        this._store.update((state) => {
          return {
            ...state,
            ...project
          };
        });
      }),
      catchError((error) => {
        this._store.setError(error);
        return of(error);
      })
    )
    .subscribe();

console.log("fetching project for the first time from the mongodb");
console.log(this._store);
}
```

```
router.get('/project/:id', async(req, res) => {
    console.log(req.params.id);
    var result = await projects.findById(req.params.id).populate([{
        path: 'admin pm issues users',
        populate: {
            path: 'comments',
            model: 'comments',
        }
    }]);
    res.send(result);
})

router.post('/project/update/:id', (req, res) => {
    console.log(req.params.id);
    var newUpdatedDoc = JSON.parse(JSON.stringify(req.body))
    projects.findOneAndUpdate({ "_id": req.params.id.toString() }, newUpdatedDoc, { new: true },
        function(err, result) {
            if (err) {
                console.log("error occured in project");
                res.send("error occured")
            } else if (!result) {
                res.send("No Project Found")
            } else {
                console.log(result);
                res.json(result);
            }
        });
})
```

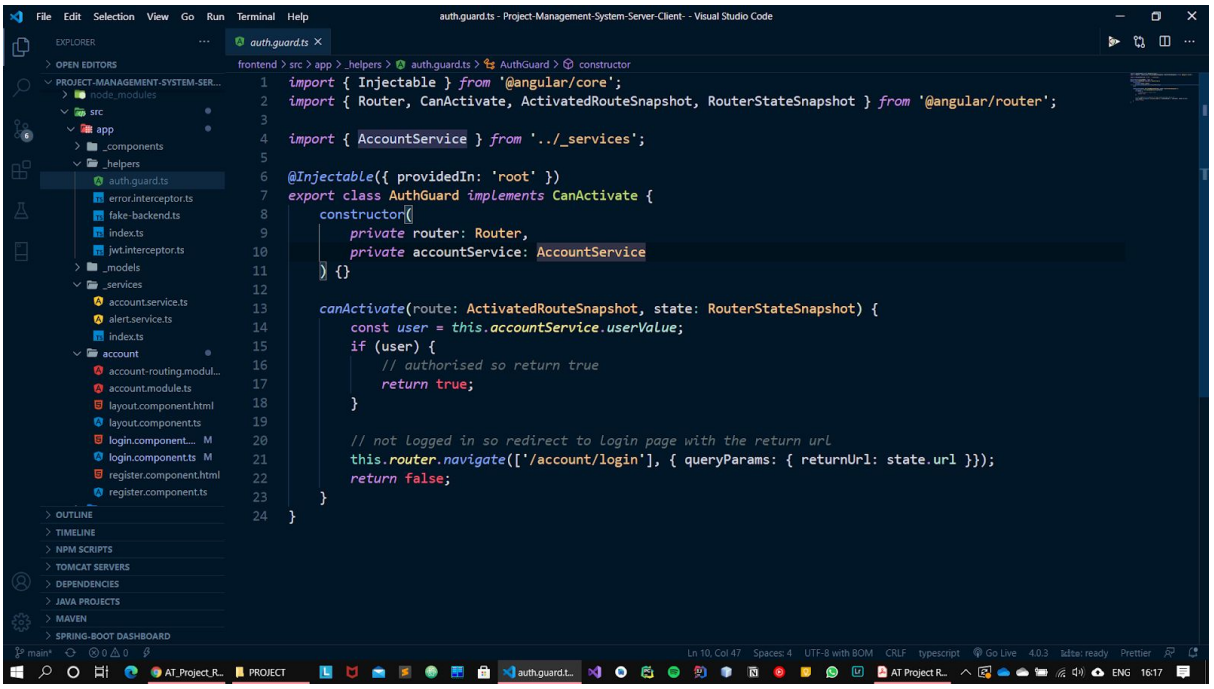
```
updateProject(project: Partial<Project>) {

  this._http
  .post<Project>(`${this.baseUrl}/project-api/project/update/${project._id}`, project)
  .pipe(
    setLoading(this._store),
    tap((project) => {
      this._store.update((state) => {
        return {
          ...state,
          ...project
        };
      });

      this.getProject();
    }),
    catchError((error) => {
      this._store.setError(error);
      return of(error);
    })
  )
  .subscribe();

}
```

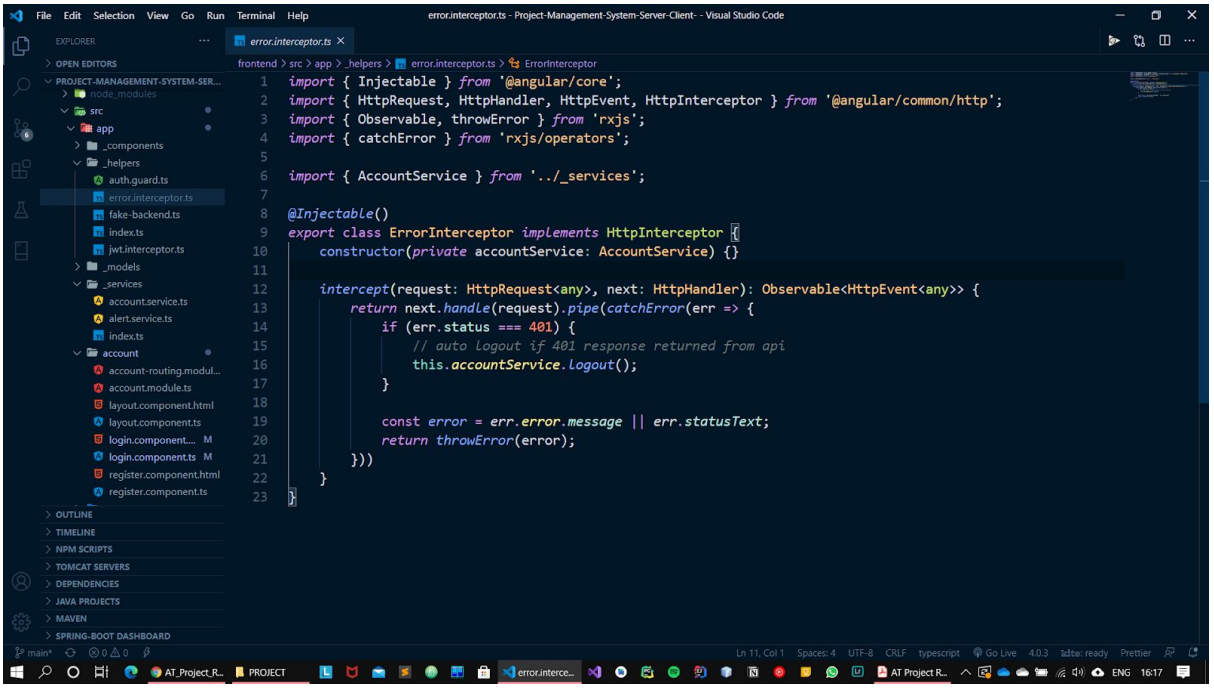
```
updateIssue(issue: Issue) {
  issue.updatedAt = DateUtil.getNow();
  this._http
  .post<Issue>(`${this.baseUrl}/issue-api/issue/update/${issue._id}`, issue)
  .pipe(
    setLoading(this._store),
    tap((issue) => {
      this._store.update((state) => {
        const issues = arrayUpsert(state.issues, issue._id, issue);
        return {
          ...state,
          issues
        };
      });
      this.getProject();
    }),
    catchError((error) => {
      this._store.setError(error);
      return of(error);
    })
  )
  .subscribe();

  console.log(this._store);
}
```

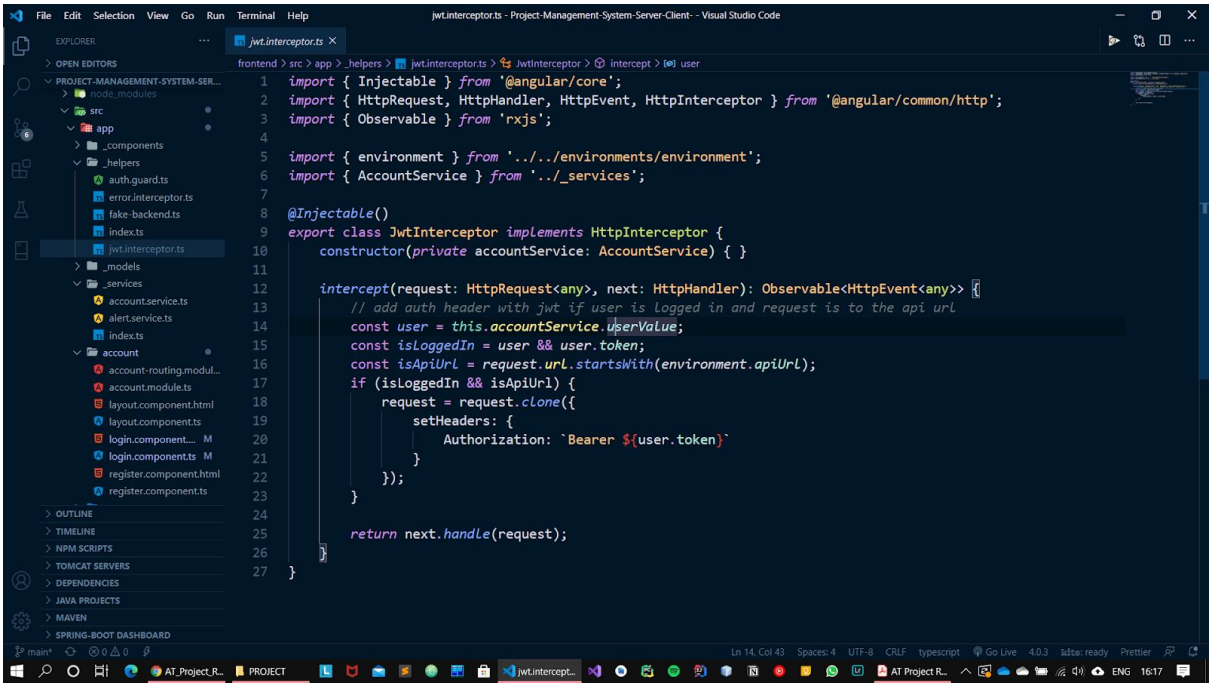
```
updateIssueComment(issueId: string, comment: Comment) {
  const allIssues = this._store.getValue().issues;
  const issue = allIssues.find((x) => x._id === issueId);
  if (!issue) {
    return;
  }



  this._http
  .post<Comment>(`${this.baseUrl}/comment-api/comment/create/${issueId}`,{})
  .pipe(
    setLoading(this._store),
    tap((comment) => {
      this._store.update((state) => {
        const comments = arrayUpsert(issue.comments ?? [], comment._id, comment);
        this.updateIssue({
          ...issue,
          comments
        });
      });
      this.getProject();
    }),
    catchError((error) => {
      this._store.setError(error);
      return of(error);
    })
  )
  .subscribe();
}
```

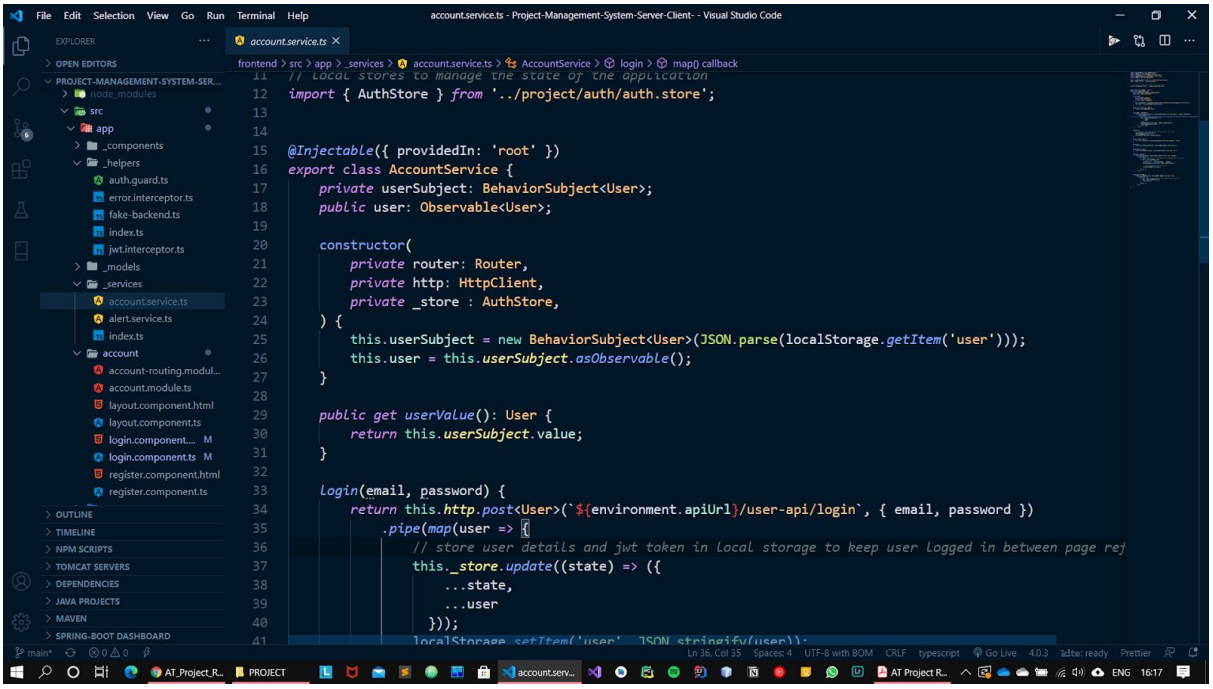
```
updateProject(project: Partial<Project>) {

  this._http
  .post<Project>(`${this.baseUrl}/project-api/project/update/5f9b109e856c60c2c2d349a6`, project)
  .pipe(
    setLoading(this._store),
    tap((project) => {
      this._store.update((state) => {
        return {
          ...state,
          ...project
        };
      });

      this.getProject();
    }),
    catchError((error) => {
      this._store.setError(error);
      return of(error);
    })
  )
  .subscribe();

}
```

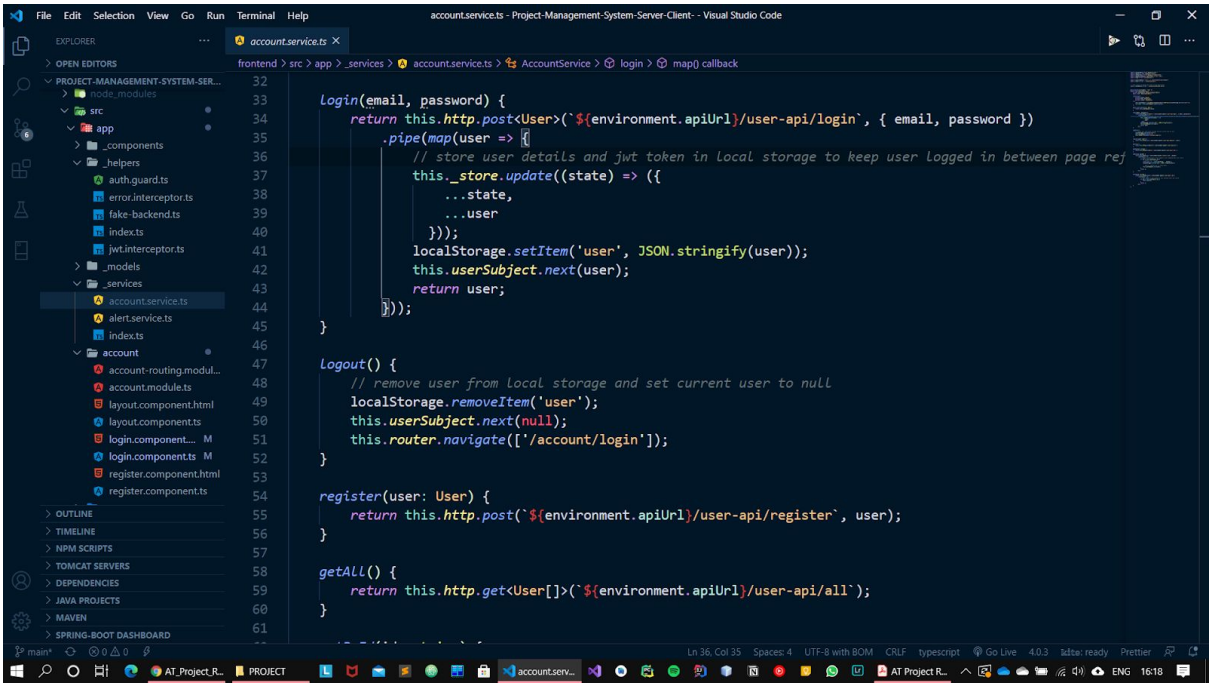
Few more services and guards to protect different routes with unauthorized access.

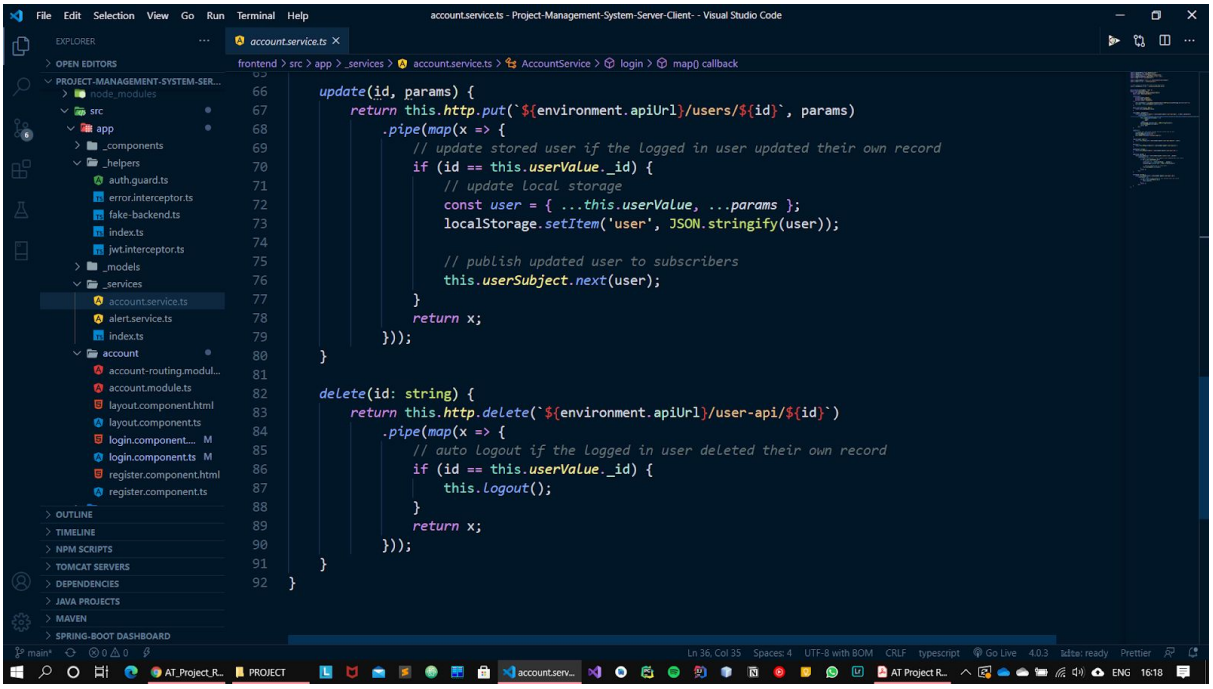


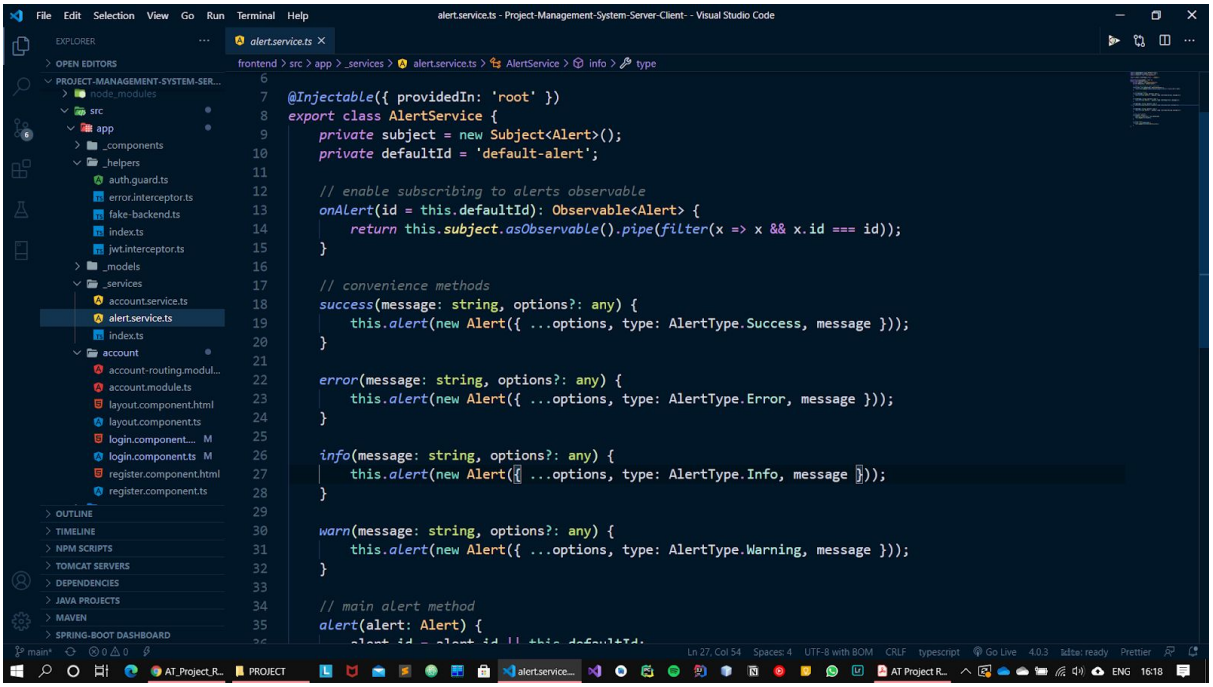


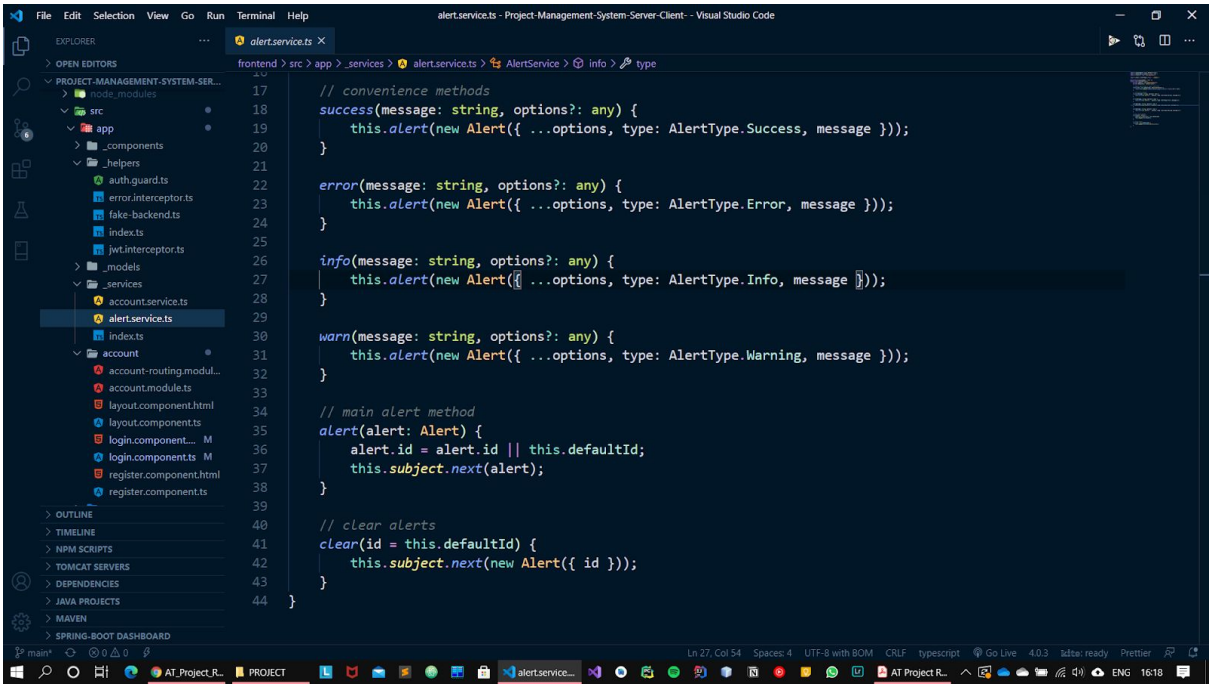

jwt.interceptor.ts - Project-Management-System-Server-Client- - Visual Studio Code
frontend > src > app > _helpers > jwt.interceptor.ts > JwtInterceptor > intercept > user
import { Injectable } from '@angular/core';
import { HttpRequest, HttpHandler, HttpEvent, HttpInterceptor } from '@angular/common/http';
import { Observable } from 'rxjs';
import { environment } from '../../environments/environment';
import { AccountService } from '../_services';
@Injectable()
export class JwtInterceptor implements HttpInterceptor {
constructor(private accountService: AccountService) { }
intercept(request: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
// add auth header with jwt if user is logged in and request is to the api url
const user = this.accountService.userValue;
const isLoggedIn = user && user.token;
const isApiUrl = request.url.startsWith(environment.apiUrl);
if (isLoggedIn && isApiUrl) {
request = request.clone({
setHeaders: {
Authorization: `Bearer ${user.token}`
}
});
}
return next.handle(request);
}
}

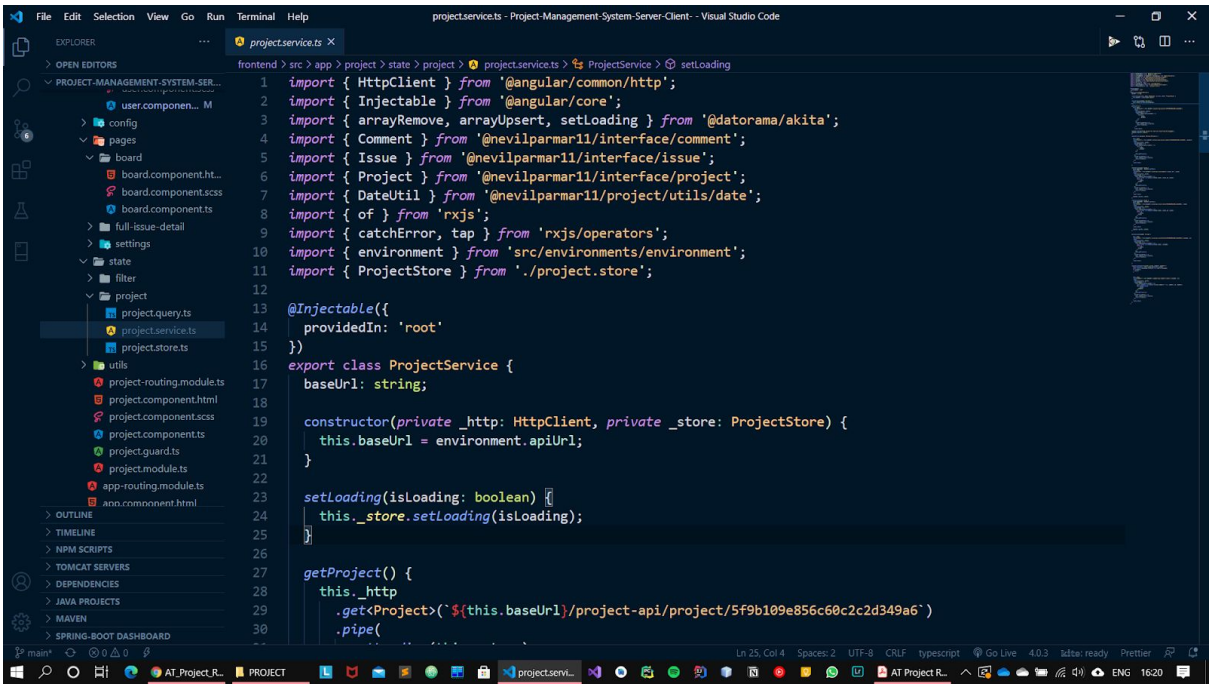

account.service.ts - Project-Management-System-Server-Client- - Visual Studio Code
frontend > src > app > _services > account.service.ts > AccountService > login > map() callback
// local stores to manage the state of the application
import { AuthStore } from '../project/auth/auth.store';
@Injectable({ providedIn: 'root' })
export class AccountService {
private userSubject: BehaviorSubject<User>;
public user: Observable<User>;
constructor(
private router: Router,
private http: HttpClient,
private _store : AuthStore,
) {
this.userSubject = new BehaviorSubject<User>(JSON.parse(localStorage.getItem('user')));
this.user = this.userSubject.asObservable();
}
public get userValue(): User {
return this.userSubject.value;
}
login(email, password) {
return this.http.post<User>(`${environment.apiUrl}/user-api/login`, { email, password })
.pipe(map(user => {
// store user details and jwt token in local storage to keep user logged in between page ref
this._store.update((state) => ({
...state,
...user
}));


File Edit Selection View Go Run Terminal Help
account.service.ts - Project-Management-System-Server-Client- - Visual Studio Code
EXPLORER
account.service.ts
frontend > src > app > _services > account.service.ts > AccountService > login > map() callback
OPEN EDITORS
PROJECT-MANAGEMENT-SYSTEM-SER...
node_modules
src
app
_components
_helpers
auth.guard.ts
error.interceptor.ts
fake-backend.ts
index.ts
jwt.interceptor.ts
_models
_services
account.service.ts
alert.service.ts
index.ts
account
account-routing.modul...
account.module.ts
layout.component.html
layout.component.ts
login.component.... M
login.component.ts M
register.component.html
register.component.ts
OUTLINE
TIMELINE
NPM SCRIPTS
TOMCAT SERVERS
DEPENDENCIES
JAVA PROJECTS
MAVEN
SPRING-BOOT DASHBOARD
login(email, password) {
return this.http.post<User>(`${environment.apiUrl}/user-api/login`, { email, password })
.pipe(map(user => {
// store user details and jwt token in local storage to keep user logged in between page rel
this._store.update((state) => ({
...state,
...user
}));
localStorage.setItem('user', JSON.stringify(user));
this.userSubject.next(user);
return user;
}));
}
logout() {
// remove user from local storage and set current user to null
localStorage.removeItem('user');
this.userSubject.next(null);
this.router.navigate(['/account/login']);
}
register(user: User) {
return this.http.post(`${environment.apiUrl}/user-api/register`, user);
}
getAll() {
return this.http.get<User[]>(`${environment.apiUrl}/user-api/all`);
}
Ln 36, Col 35 Spaces: 4 UTF-8 with BOM CRLF typescript Go Live 4.0.3 Idle: ready Prettier
AT_Project_R... PROJECT account.serv... AT Project R... ENG 16:18


File Edit Selection View Go Run Terminal Help
account.service.ts - Project-Management-System-Server-Client- - Visual Studio Code
EXPLORER
account.service.ts
frontend > src > app > _services > account.service.ts > AccountService > login > map() callback
update(id, params) {
return this.http.put(`${environment.apiUrl}/users/${id}`, params)
.pipe(map(x => {
// update stored user if the logged in user updated their own record
if (id == this.userValue._id) {
// update local storage
const user = { ...this.userValue, ...params };
localStorage.setItem('user', JSON.stringify(user));
// publish updated user to subscribers
this.userSubject.next(user);
}
return x;
}));
}
delete(id: string) {
return this.http.delete(`${environment.apiUrl}/user-api/${id}`)
.pipe(map(x => {
// auto logout if the logged in user deleted their own record
if (id == this.userValue._id) {
this.logout();
}
return x;
}));
}
}
Ln 36, Col 35 Spaces: 4 UTF-8 with BOM CRLF typescript Go Live 4.0.3 Idle: ready Prettier
AT_Project_R... PROJECT account.serv... AT Project R... ENG 16:18


File Edit Selection View Go Run Terminal Help
alert.service.ts - Project-Management-System-Server-Client- - Visual Studio Code
EXPLORER
alert.service.ts
frontend > src > app > _services > alert.service.ts > AlertService > info > type
OPEN EDITORS
PROJECT-MANAGEMENT-SYSTEM-SER...
node_modules
src
app
_components
_helpers
auth.guard.ts
error.interceptor.ts
fake-backend.ts
index.ts
jwt.interceptor.ts
_models
_services
account.service.ts
alert.service.ts
index.ts
account
account-routing.modul...
account.module.ts
layout.component.html
layout.component.ts
login.component.... M
login.component.ts M
register.component.html
register.component.ts
OUTLINE
TIMELINE
NPM SCRIPTS
TOMCAT SERVERS
DEPENDENCIES
JAVA PROJECTS
MAVEN
SPRING-BOOT DASHBOARD
@Injectable({ providedIn: 'root' })
export class AlertService {
private subject = new Subject<Alert>();
private defaultId = 'default-alert';
// enable subscribing to alerts observable
onAlert(id = this.defaultId): Observable<Alert> {
return this.subject.asObservable().pipe(filter(x => x && x.id === id));
}
// convenience methods
success(message: string, options?: any) {
this.alert(new Alert({ ...options, type: AlertType.Success, message }));
}
error(message: string, options?: any) {
this.alert(new Alert({ ...options, type: AlertType.Error, message }));
}
info(message: string, options?: any) {
this.alert(new Alert({ ...options, type: AlertType.Info, message }));
}
warn(message: string, options?: any) {
this.alert(new Alert({ ...options, type: AlertType.Warning, message }));
}
// main alert method
alert(alert: Alert) {
Ln 27, Col 54 Spaces: 4 UTF-8 with BOM CRLF typescript Go Live 4.0.3 tidte: ready Prettier
AT_Project_R... PROJECT alert.service... AT Project R... ENG 16:18


File Edit Selection View Go Run Terminal Help
alert.service.ts - Project-Management-System-Server-Client- - Visual Studio Code
EXPLORER
alert.service.ts
frontend > src > app > _services > alert.service.ts > AlertService > info > type
OPEN EDITORS
PROJECT-MANAGEMENT-SYSTEM-SER...
node_modules
src
app
_components
_helpers
auth.guard.ts
error.interceptor.ts
fake-backend.ts
index.ts
jwt.interceptor.ts
_models
_services
account.service.ts
alert.service.ts
index.ts
account
account-routing.modul...
account.module.ts
layout.component.html
layout.component.ts
login.component.... M
login.component.ts M
register.component.html
register.component.ts
OUTLINE
TIMELINE
NPM SCRIPTS
TOMCAT SERVERS
DEPENDENCIES
JAVA PROJECTS
MAVEN
SPRING-BOOT DASHBOARD
// convenience methods
success(message: string, options?: any) {
this.alert(new Alert({ ...options, type: AlertType.Success, message }));
}
error(message: string, options?: any) {
this.alert(new Alert({ ...options, type: AlertType.Error, message }));
}
info(message: string, options?: any) {
this.alert(new Alert({ ...options, type: AlertType.Info, message }));
}
warn(message: string, options?: any) {
this.alert(new Alert({ ...options, type: AlertType.Warning, message }));
}
// main alert method
alert(alert: Alert) {
alert.id = alert.id || this.defaultId;
this.subject.next(alert);
}
// clear alerts
clear(id = this.defaultId) {
this.subject.next(new Alert({ id }));
}
}
Ln 27, Col 54 Spaces: 4 UTF-8 with BOM CRLF typescript Go Live 4.0.3 tidte: ready Prettier
AT_Project_R... PROJECT alert.service... AT Project R... ENG 16:18


File Edit Selection View Go Run Terminal Help
project.service.ts - Project-Management-System-Server-Client- - Visual Studio Code
EXPLORER
project.service.ts
frontend > src > app > project > state > project > project.service.ts > ProjectService > setLoading
OPEN EDITORS
PROJECT-MANAGEMENT-SYSTEM-SER...
config
pages
board
board.component.ht...
board.component.scss
board.component.ts
full-issue-detail
settings
state
filter
project
project.query.ts
project.service.ts
project.store.ts
utils
project-routing.module.ts
project.component.html
project.component.scss
project.component.ts
project.guard.ts
project.module.ts
app-routing.module.ts
OUTLINE
TIMELINE
NPM SCRIPTS
TOMCAT SERVERS
DEPENDENCIES
JAVA PROJECTS
MAVEN
SPRING-BOOT DASHBOARD
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { arrayRemove, arrayUpsert, setLoading } from '@datorama/akita';
import { Comment } from '@nevilparmar11/interface/comment';
import { Issue } from '@nevilparmar11/interface/issue';
import { Project } from '@nevilparmar11/interface/project';
import { DateUtil } from '@nevilparmar11/project/utils/date';
import { of } from 'rxjs';
import { catchError, tap } from 'rxjs/operators';
import { environment } from 'src/environments/environment';
import { ProjectStore } from './project.store';
@Injectable({
providedIn: 'root'
})
export class ProjectService {
baseUrl: string;
constructor(private _http: HttpClient, private _store: ProjectStore) {
this.baseUrl = environment.apiUrl;
}
setLoading(isLoading: boolean) {
this._store.setLoading(isLoading);
}
getProject() {
this._http
.get<Project>(`${this.baseUrl}/project-api/project/5f9b109e856c60c2c2d349a6`)
.pipe(
Ln 25, Col 4 Spaces: 2 UTF-8 CRLF typescript Go Live 4.0.3 tidte: ready Prettier
ENG 16:20

```
getById(id: string) {
    return this.http.get<User>(`${environment.apiUrl}/user-api/${id}`);
}

update(id, params) {
    return this.http.put(`${environment.apiUrl}/users/${id}`, params)
        .pipe(map(x => {
            // update stored user if the logged in user updated their own record
            if (id == this.userValue._id) {
                // update local storage
                const user = { ...this.userValue, ...params };
                localStorage.setItem('user', JSON.stringify(user));

                // publish updated user to subscribers
                this.userSubject.next(user);
            }
            return x;
        }));
}
```

project.guard.ts

frontend > src > app > project > project.guard.ts > ProjectGuard > canActivate

```
import { ProjectService } from './state/project/project.service';
import { ProjectState } from './state/project/project.store';

@Injectable({
  providedIn: 'root'
})
export class ProjectGuard implements CanActivate {
  constructor(private _projectQuery: ProjectQuery, private _projectService: ProjectService) {}
  canActivate(next: ActivatedRouteSnapshot, state: RouterStateSnapshot): Observable<boolean> {
    return this.getFromStoreOrApi().pipe(
      switchMap(() => of(true)),
      catchError(() => of(false))
    );
  }

  getFromStoreOrApi(): Observable<ProjectState> {
    return combineLatest([this._projectQuery.all$, this._projectQuery.isLoading$]).pipe(
      map(([state, loading]) => {
        if (!loading) {
          this._projectService.getProject();
        }
        return state;
      }),
      filter((state) => !!state._id),
      take(1),
      catchError((error) => of(error))
    );
  }
}
```

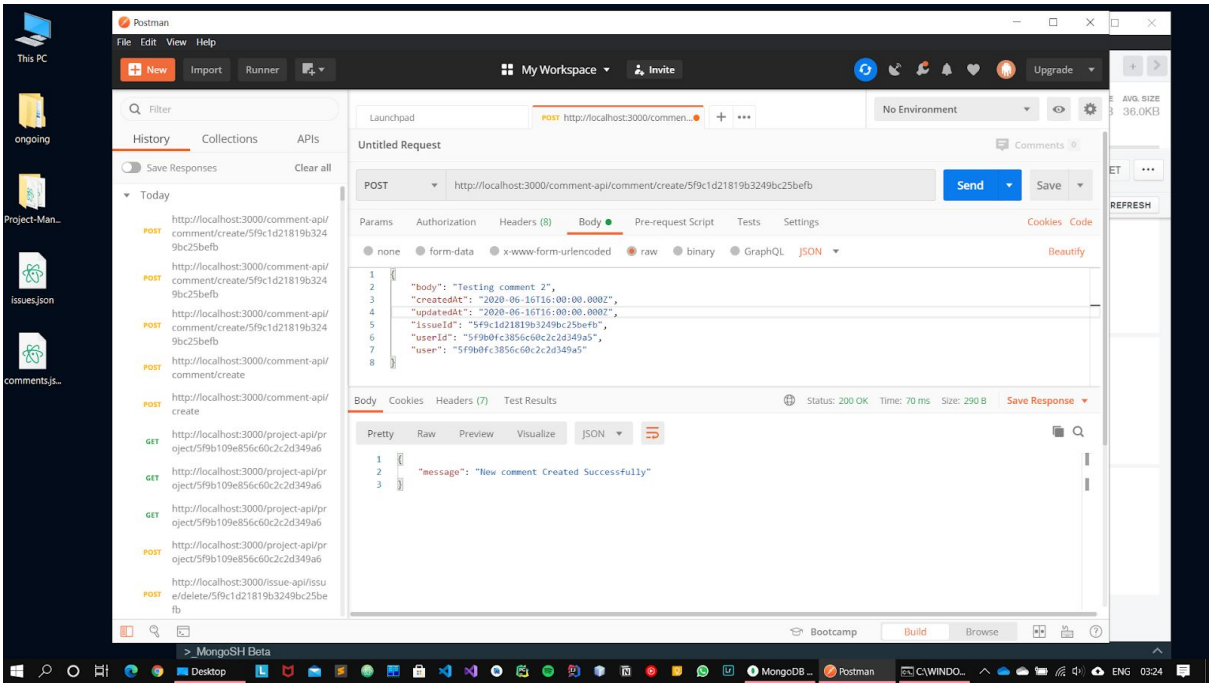
Testing

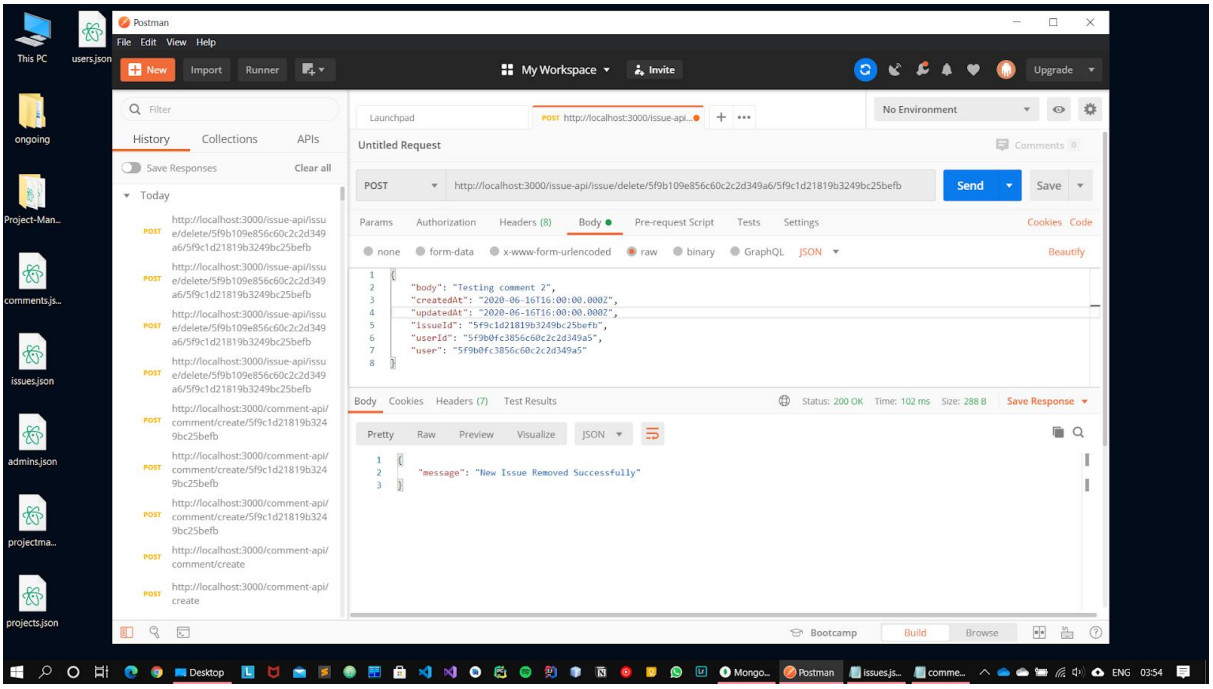
Testing Frontend Services :

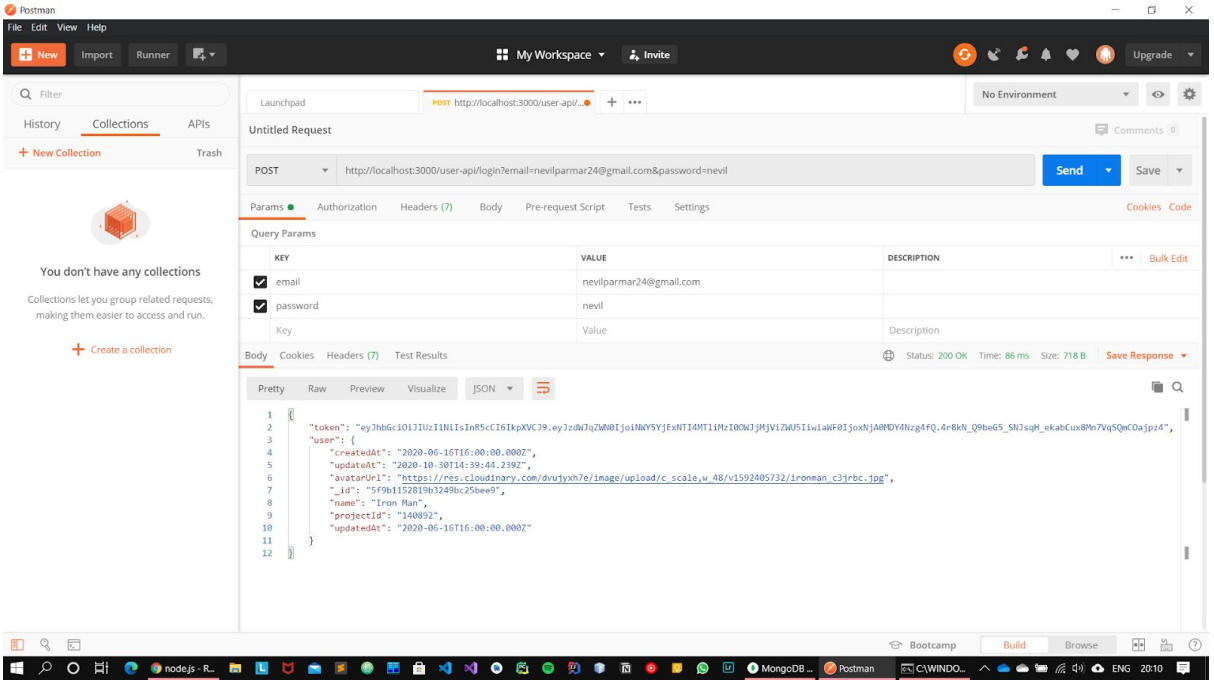
Test cases were made to test some of the components of the angular. Also manual testing was performed to find and fix the bug in the development.

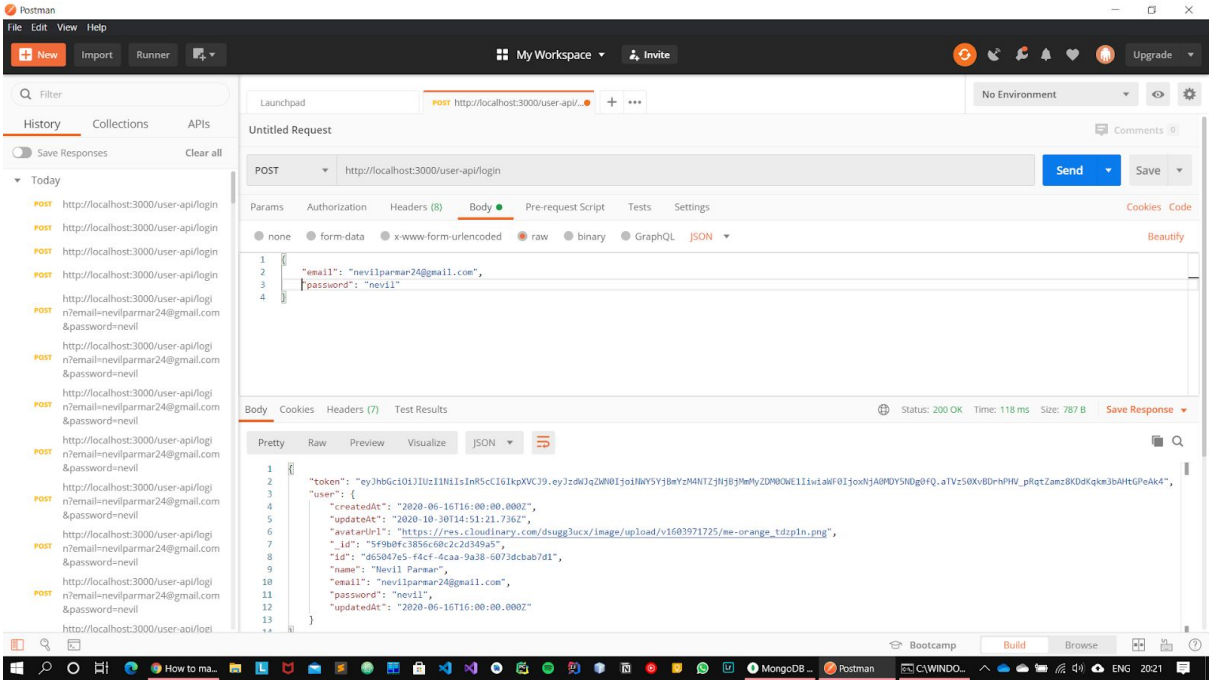
Testing Backend Services:

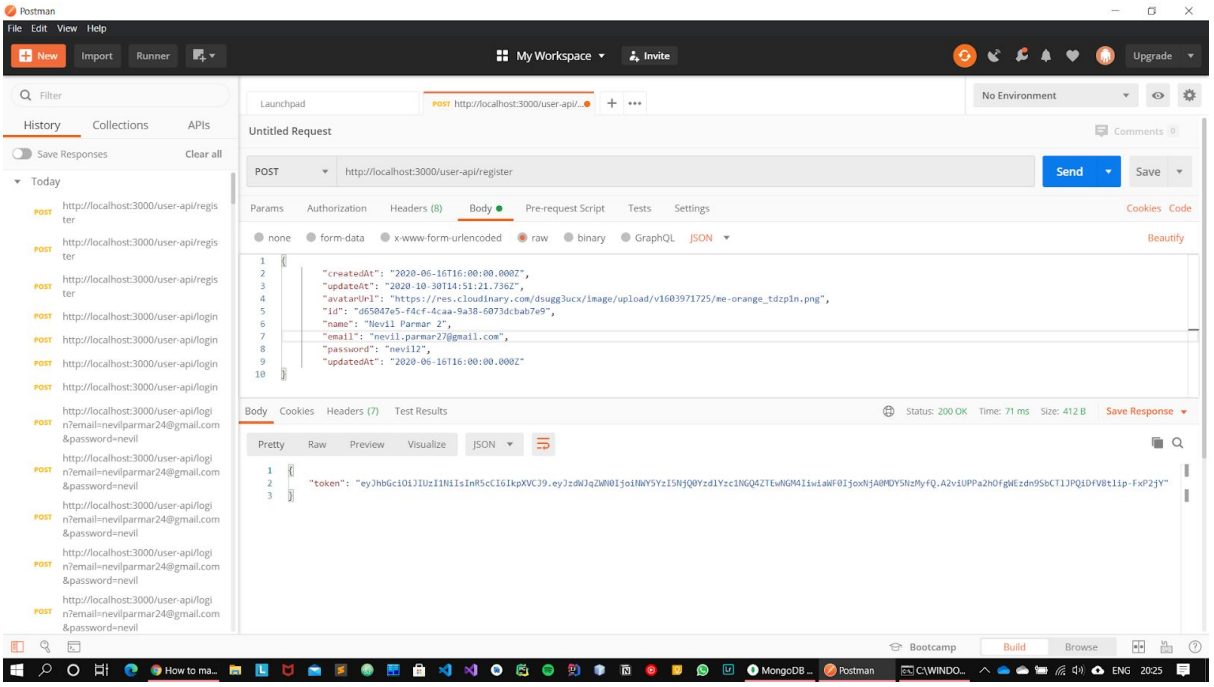
The backend services were tested using “POSTMAN” in order to fix bugs during development.


Postman
File Edit View Help
New Import Runner
My Workspace
Invite
Upgrade
Filter
History Collections APIs
Save Responses Clear all
Today
Launchpad
Untitled Request
No Environment
POST
http://localhost:3000/comment-api/comment/create/5f9c1d21819b3249bc25befb
Send
Save
Params Authorization Headers (8) Body Pre-request Script Tests Settings
Cookies Code
none form-data x-www-form-urlencoded raw binary GraphQL JSON
Beautify
{
"body": "Testing comment 2",
"createdAt": "2020-06-16T16:00:00.000Z",
"updatedAt": "2020-06-16T16:00:00.000Z",
"issueId": "5f9c1d21819b3249bc25befb",
"userId": "5f9b0fc3856c60c2c2d349a5",
"user": "5f9b0fc3856c60c2c2d349a5"
}
Body Cookies Headers (7) Test Results
Status: 200 OK Time: 70 ms Size: 290 B
Save Response
Pretty Raw Preview Visualize JSON
{
"message": "New comment Created Successfully"
}
Bootcamp Build Browse
>_MongoSH Beta

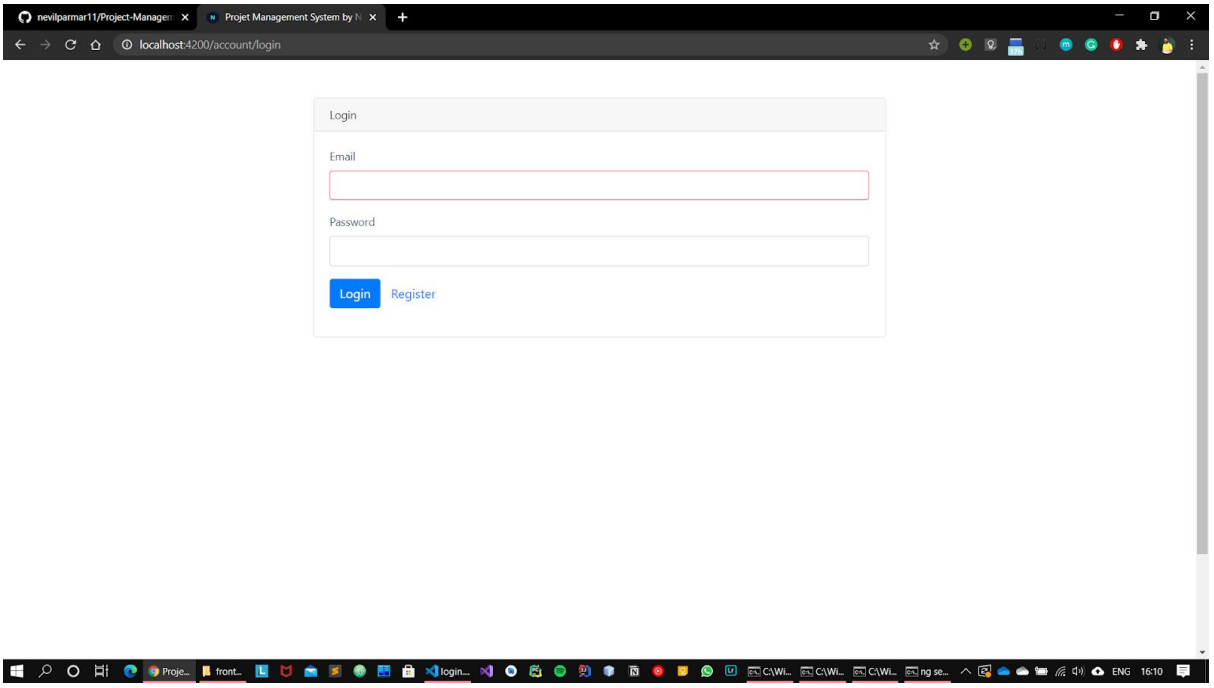

Postman
File Edit View Help
New Import Runner
My Workspace
Invite
Upgrade
Filter
History Collections APIs
Save Responses Clear all
Today
Launchpad
Untitled Request
No Environment
POST
http://localhost:3000/issue-api/issue/delete/5f9b109e856c60c2c2d349a6/5f9c1d21819b3249bc25befb
Send
Save
Params Authorization Headers (8) Body Pre-request Script Tests Settings
Cookies Code
none form-data x-www-form-urlencoded raw binary GraphQL JSON
Beautify
{
"body": "Testing comment 2",
"createdAt": "2020-06-16T16:00:00.000Z",
"updatedAt": "2020-06-16T16:00:00.000Z",
"issueId": "5f9c1d21819b3249bc25befb",
"userId": "5f9b0fc3856c60c2c2d349a5",
"user": "5f9b0fc3856c60c2c2d349a5"
}
Body Cookies Headers (7) Test Results
Status: 200 OK Time: 102 ms Size: 288 B
Save Response
Pretty Raw Preview Visualize JSON
{
"message": "New Issue Removed Successfully"
}
Bootcamp Build Browse

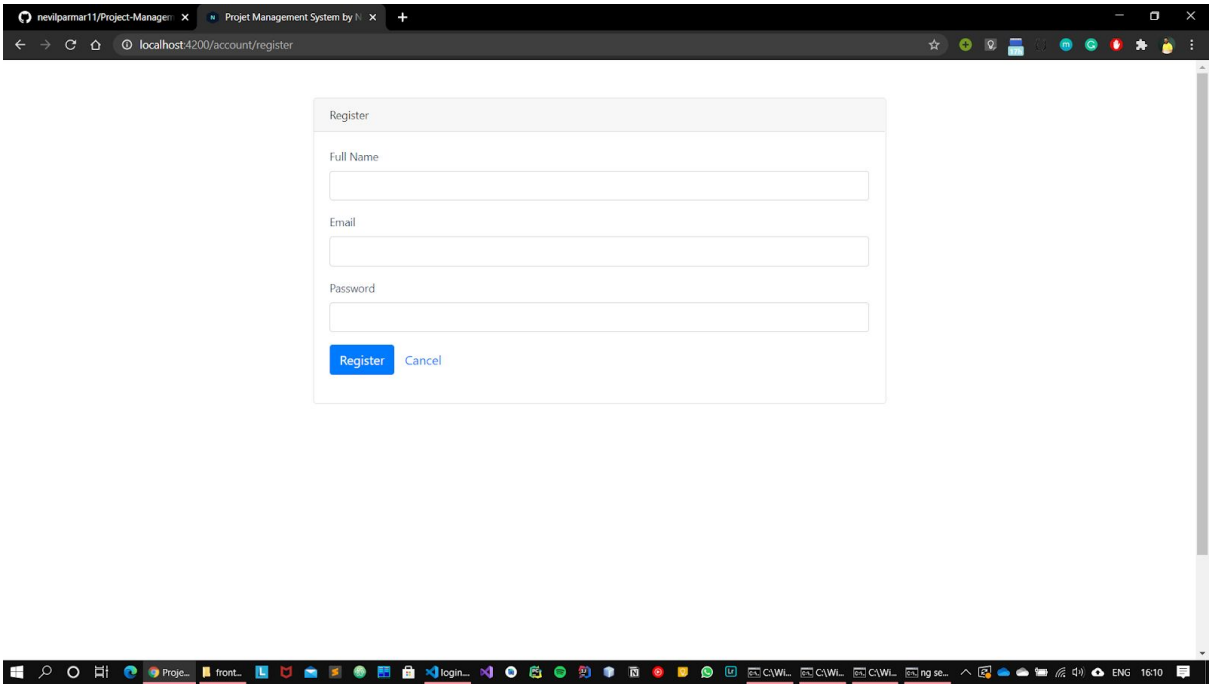

Postman
File Edit View Help
New
Import
Runner
My Workspace
Invite
Upgrade
Filter
History
Collections
APIs
New Collection
Trash
You don't have any collections
Collections let you group related requests, making them easier to access and run.
Create a collection
Launchpad
POST http://localhost:3000/user-api/...
No Environment
Untitled Request
Comments 0
POST
http://localhost:3000/user-api/login?email=nevilparmar24@gmail.com&password=nevil
Send
Save
Params
Authorization
Headers (7)
Body
Pre-request Script
Tests
Settings
Cookies
Code
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
email
nevilparmar24@gmail.com
password
nevil
Key
Value
Description
Body
Cookies
Headers (7)
Test Results
Status: 200 OK
Time: 86 ms
Size: 718 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWJqZWN0IjoiNWY5YjExNTI4MTliMzI0OWJjMjViZWU5IiwiaWF0IjoxNjA0MDY4Nzg4fQ.4r8kN_Q9beG5_SNJsqH_ekabCux8Mn7VqSQmCOajpz4",
"user": {
"createdAt": "2020-06-16T16:00:00.000Z",
"updateAt": "2020-10-30T14:39:44.239Z",
"avatarUrl": "https://res.cloudinary.com/dvujyxh7e/image/upload/c_scale,w_48/v1592405732/ironman_c3jrbc.jpg",
"_id": "5f9b1152819b3249bc25bee9",
"name": "Iron Man",
"projectId": "140892",
"updatedAt": "2020-06-16T16:00:00.000Z"
}
}
Bootcamp
Build
Browse
node.js - R...
MongoDB ...
Postman
C:\WINDO...
ENG
20:10


Postman
File Edit View Help
New Import Runner
My Workspace
Invite
Upgrade
Filter
History Collections APIs
Save Responses Clear all
Today
Launchpad
No Environment
Untitled Request
POST
http://localhost:3000/user-api/login
Send
Save
Params Authorization Headers (8) Body Pre-request Script Tests Settings
Cookies Code
none form-data x-www-form-urlencoded raw binary GraphQL JSON
Beautify
{
"email": "nevilparmar24@gmail.com",
"password": "nevil"
}
Body Cookies Headers (7) Test Results
Status: 200 OK Time: 118 ms Size: 787 B
Save Response
Pretty Raw Preview Visualize JSON
{
"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWJqZWN0IjoiNWY5YjBmYzN4NTZjNjBjMmMyZDM0OWE1IiwiaWF0IjoxNjA0MDY5NDg0fQ.aTVz50XvBDrhPHV_pRqtZamz8KDdKqkm3bAHtGPeAk4",
"user": {
"createdAt": "2020-06-16T16:00:00.000Z",
"updatedAt": "2020-10-30T14:51:21.736Z",
"avatarUrl": "https://res.cloudinary.com/dsugg3ucx/image/upload/v1603971725/me-orange_tdzp1n.png",
"_id": "5f9b0fc3856c60c2c2d349a5",
"id": "d65047e5-f4cf-4caa-9a38-6073dcbab7d1",
"name": "Nevil Parmar",
"email": "nevilparmar24@gmail.com",
"password": "nevil",
"updatedAt": "2020-06-16T16:00:00.000Z"
}
Bootcamp Build Browse
How to ma... MongoDB ... Postman C:\WINDO... ENG 20:21

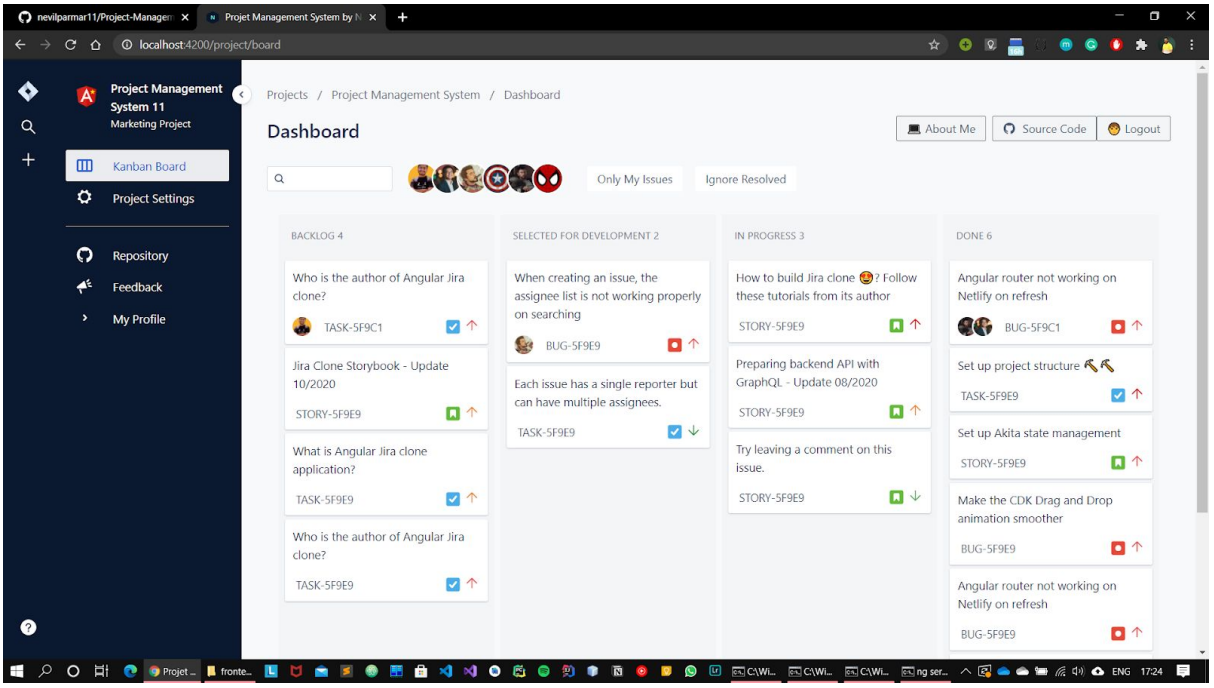

Postman
File Edit View Help
New Import Runner
My Workspace
Invite
Upgrade
Filter
History Collections APIs
Save Responses Clear all
Today
Launchpad
No Environment
Untitled Request
POST
http://localhost:3000/user-api/register
Send
Save
Params Authorization Headers (8) Body Pre-request Script Tests Settings
Cookies Code
none form-data x-www-form-urlencoded raw binary GraphQL JSON
Beautify
{
"createdAt": "2020-06-16T16:00:00.000Z",
"updatedAt": "2020-10-30T14:51:21.736Z",
"avatarUrl": "https://res.cloudinary.com/dsugg3ucx/image/upload/v1603971725/me-orange_tdzp1n.png",
"id": "d65047e5-f4cf-4caa-9a38-6073dcbab7e9",
"name": "Nevil Parmar 2",
"email": "nevil.parmar27@gmail.com",
"password": "nevil2",
"updatedAt": "2020-06-16T16:00:00.000Z"
}
Body Cookies Headers (7) Test Results
Status: 200 OK Time: 71 ms Size: 412 B
Save Response
Pretty Raw Preview Visualize JSON
{
"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWJqZWN0IjoiNWY5YzI5NjQ0Yzd1Yzc1NGQ4ZTEwNGM4IiwiaWF0IjoxNjA0MDY5NzMyfQ.A2viUPPa2hOfgWEzdn9SbCTlJPQiDfV8t1ip-FxP2jY"
}
Bootcamp Build Browse
How to ma... MongoDB ... Postman C:\WINDO... ENG 20:25

Workflow

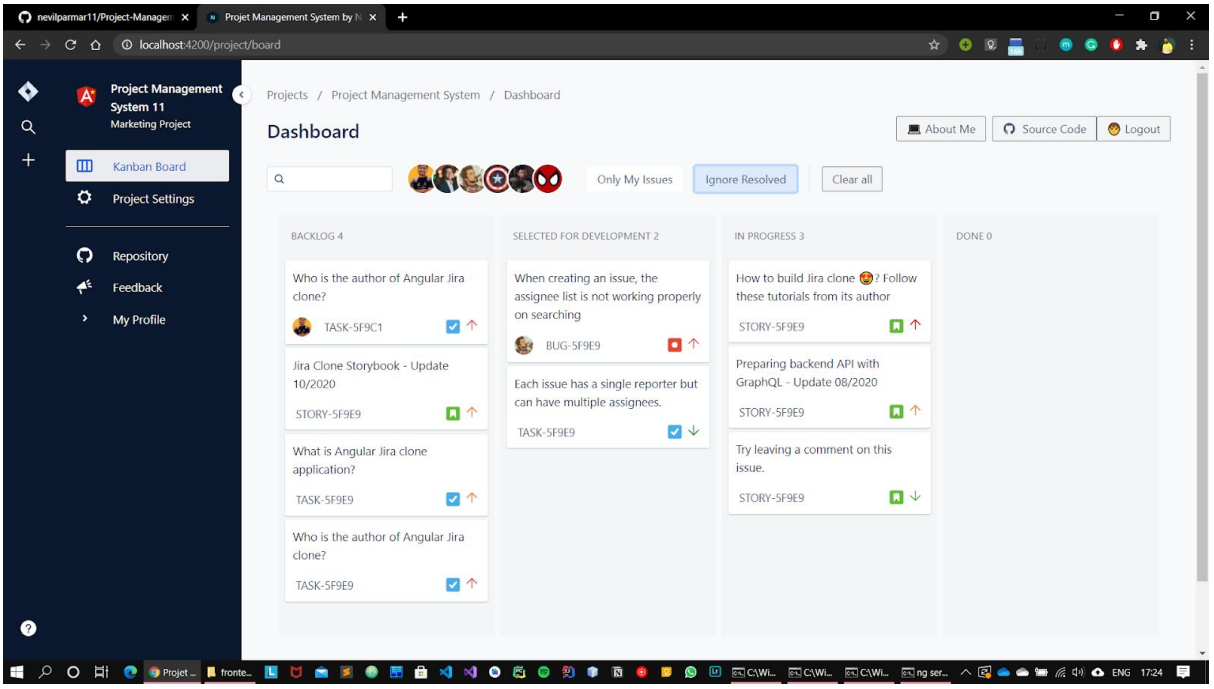

nevilparmar11/Project-Manager
Projet Management System by N
localhost:4200/account/login
Login
Email
Password
Login
Register


nevilparmar11/Project-Manager
Projet Management System by N
localhost:4200/account/register
Register
Full Name
Email
Password
Register
Cancel

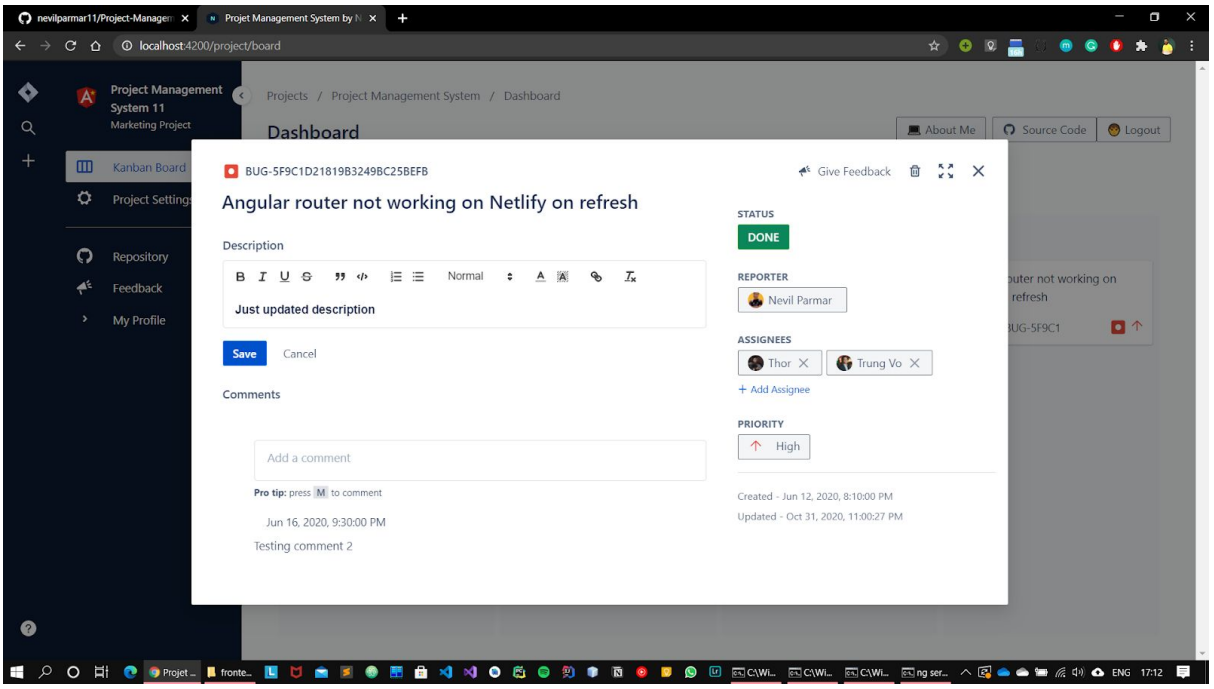
Full dashboard

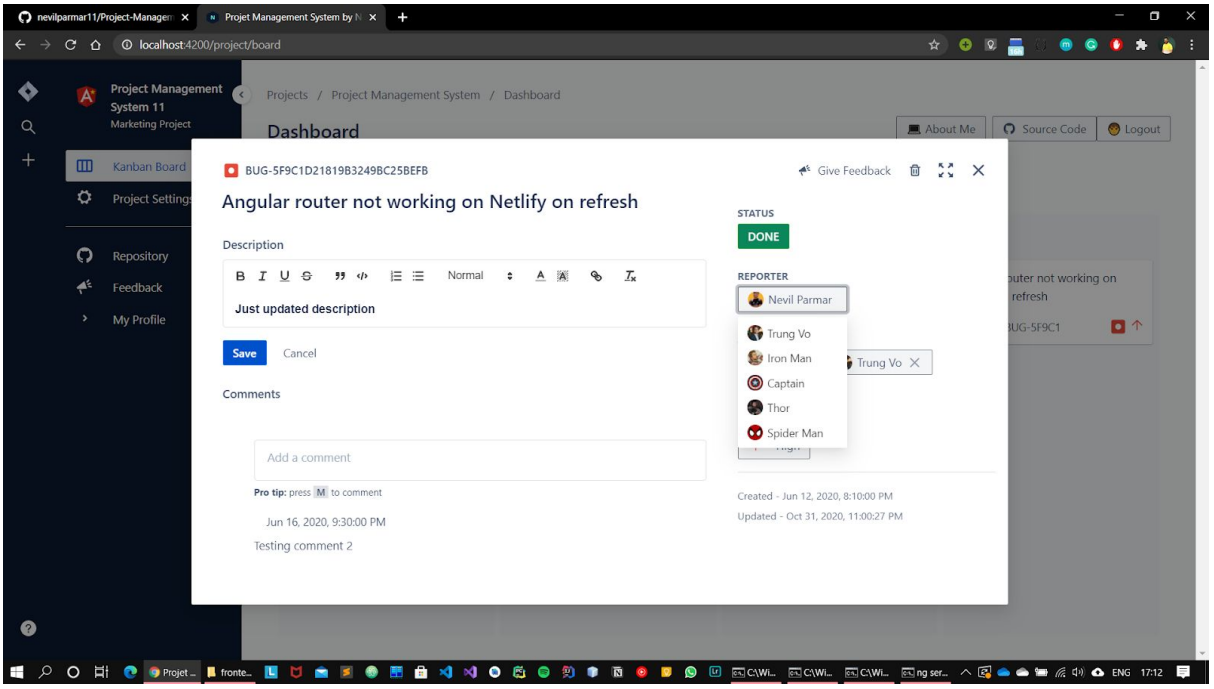


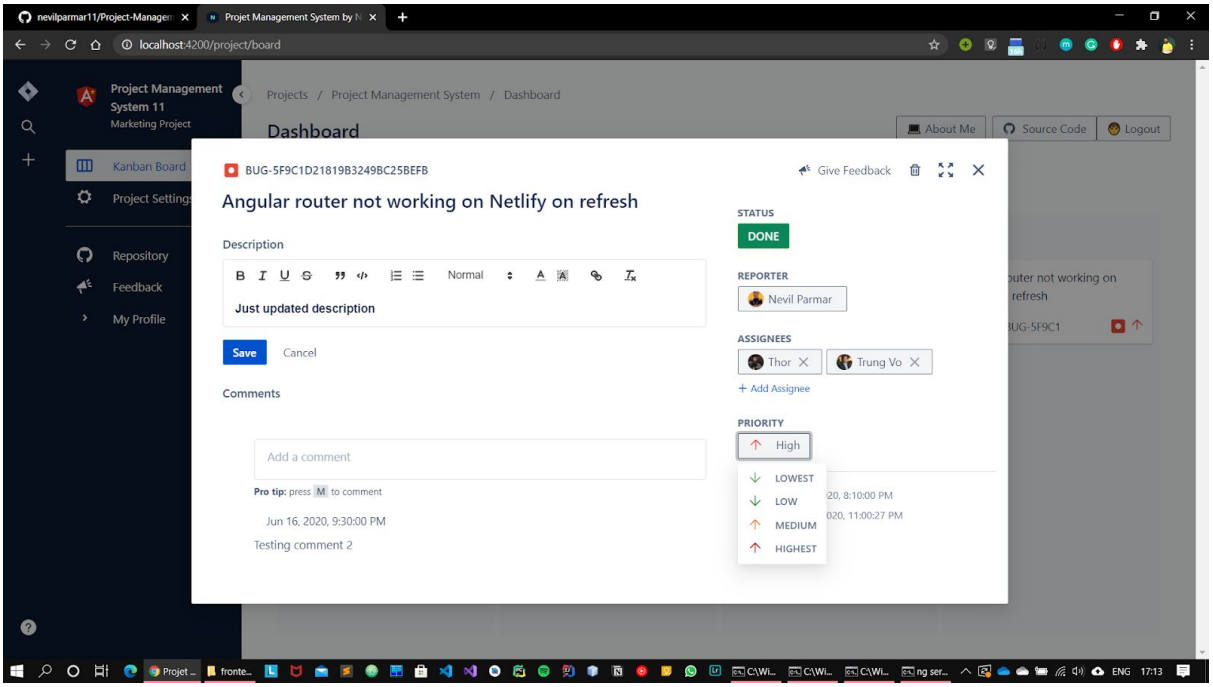
Ignore the resolved filter.

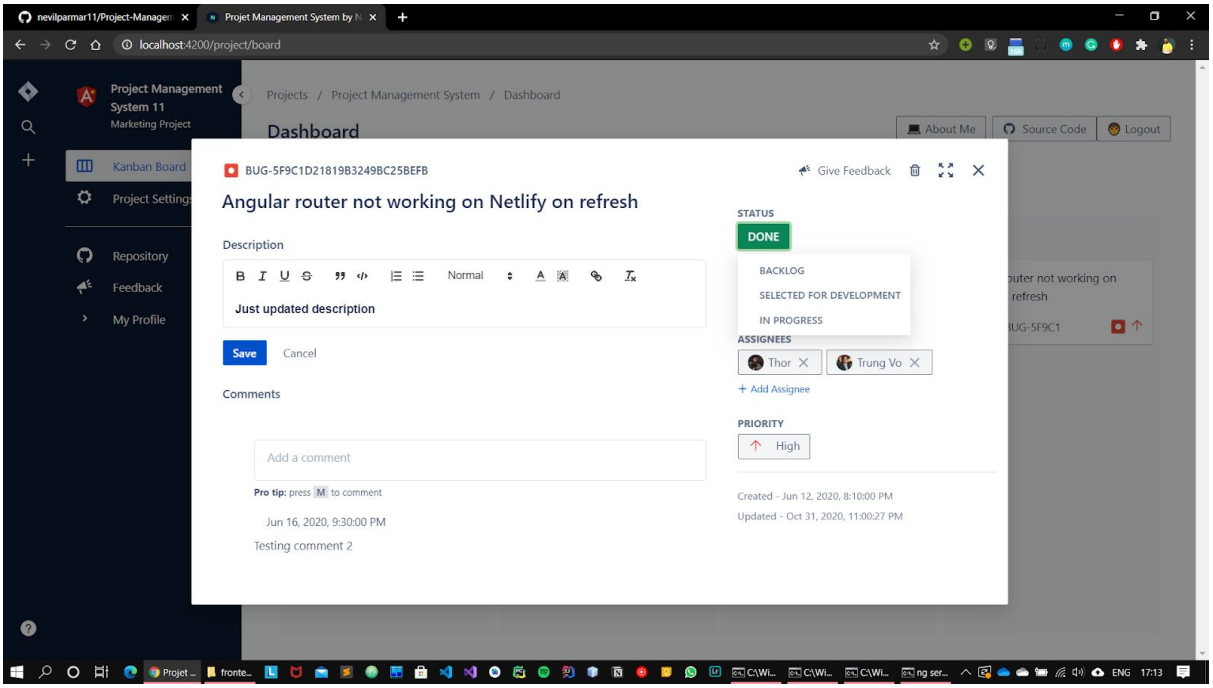


Issue details page.

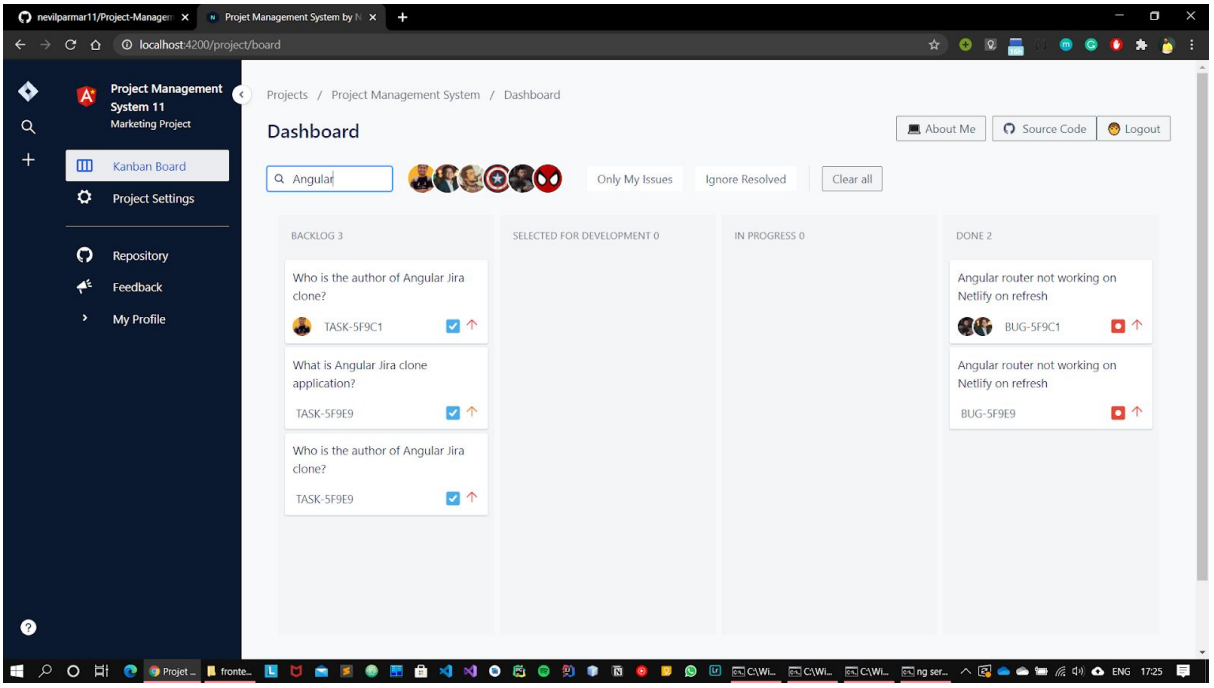


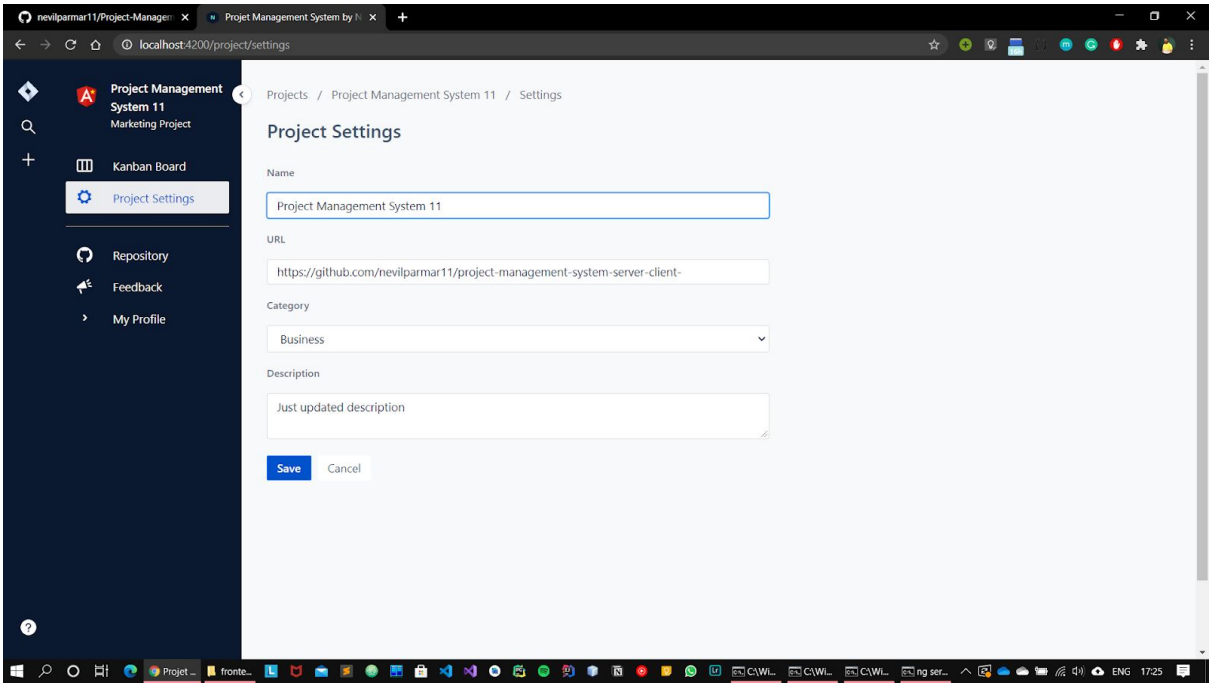


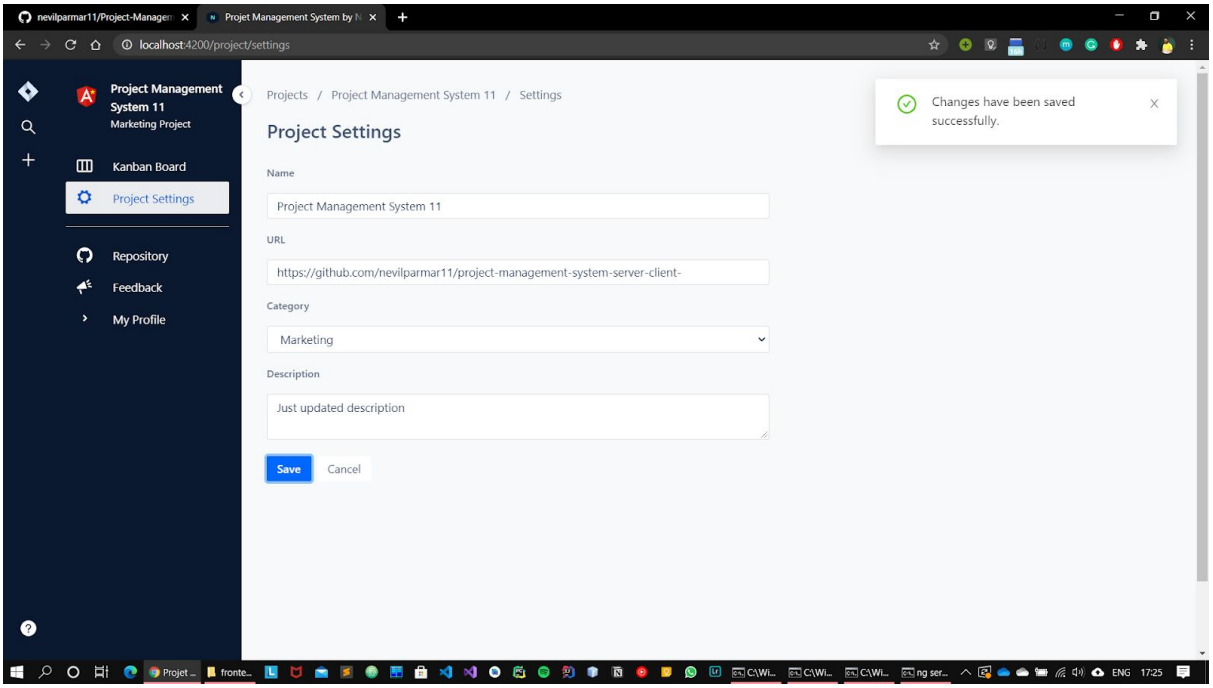

localhost:4200/project/board
Project Management System 11
Marketing Project
Kanban Board
Project Settings
Repository
Feedback
My Profile
Projects / Project Management System / Dashboard
Dashboard
About Me
Source Code
Logout
BUG-5F9C1D21819B3249BC25BEFB
Give Feedback
Angular router not working on Netlify on refresh
Description
Normal
Just updated description
Save
Cancel
Comments
Add a comment
Pro tip: press M to comment
Jun 16, 2020, 9:30:00 PM
Testing comment 2
STATUS
DONE
REPORTER
Nevil Parmar
ASSIGNEES
Thor
Trung Vo
+ Add Assignee
PRIORITY
High
LOWEST
LOW
MEDIUM
HIGHEST


localhost:4200/project/board
Project Management System 11
Marketing Project
Kanban Board
Project Settings
Repository
Feedback
My Profile
Projects / Project Management System / Dashboard
Dashboard
About Me
Source Code
Logout
BUG-5F9C1D21819B3249BC25BEFB
Give Feedback
Angular router not working on Netlify on refresh
Description
Normal
Just updated description
Save
Cancel
Comments
Add a comment
Pro tip: press M to comment
Jun 16, 2020, 9:30:00 PM
Testing comment 2
STATUS
DONE
BACKLOG
SELECTED FOR DEVELOPMENT
IN PROGRESS
ASSIGNEES
Thor
Trung Vo
+ Add Assignee
PRIORITY
High
Created - Jun 12, 2020, 8:10:00 PM
Updated - Oct 31, 2020, 11:00:27 PM

Search filter.




nevilparmar11/Project-Manager
Project Management System by N
localhost:4200/project/settings
Project Management System 11
Marketing Project
Kanban Board
Project Settings
Repository
Feedback
My Profile
Projects / Project Management System 11 / Settings
Project Settings
Name
Project Management System 11
URL
https://github.com/nevilparmar11/project-management-system-server-client-
Category
Business
Description
Just updated description
Save
Cancel


nevilparmar11/Project-Manager
Project Management System by N
localhost:4200/project/settings
Project Management System 11
Marketing Project
Kanban Board
Project Settings
Repository
Feedback
My Profile
Projects / Project Management System 11 / Settings
Changes have been saved successfully.
Project Settings
Name
Project Management System 11
URL
https://github.com/nevilparmar11/project-management-system-server-client-
Category
Marketing
Description
Just updated description
Save
Cancel

Conclusion

The functionalities were implemented in the system after understanding all the system modules according to the requirements. Functionalities That are successfully implemented in the system are:

- User registration containing all the necessary validations on field
- Login with validations
- User authentication
- Logout
- CRUD with projects, Issues/Tasks and comments on the issues
- I.e add/update/delete project details, issues and comments
- Search issues
- Filter issues

Limitation and Future Extension

We are solely following one traditional approach for any software project management by breaking up tasks into several tasks/issues, which might not be the best convenient method for any firm. In that case the system needs to be updated according to their needs and priorities.

In future, the system can be extended to provide more analytical views on the ongoing and past projects and reports. We can train a few models to predict the future values and using them the system can alert/suggest at an apt time.

All the Distinct Functionalities according to the development have been implemented except “inviting project manager/team member” and the report generation feature from the SRS.

Bibliography

Following links and websites were referred during the development of this project.

Express.js:
https://expressjs.com/en/api.html

Node.js:
https://nodejs.org/en/

MongoDB:
https://www.mongodb.com/

Akita:
https://datorama.github.io/akita/

Ng-zorro :
https://ng.ant.design/docs/introduce/en

TaiwindCSS:
https://tailwindcss.com/

Ngx-quill:
https://www.npmjs.com/package/ngx-quill

Angular CDK drag and drop:
https://material.angular.io/cdk/drag-drop/overview

HTML5: https://www.w3schools.com/html/default.asp

CSS: https://www.w3schools.com/css/default.adp

JavaScript: https://www.w3schools.com/js/default.asp

AJAX: https://www.w3schools.com/js/js_ajax_intro.asp

Jquery: https://www.w3schools.com/jquery/default.asp

XML: https://www.w3schools.com/xml/default.asp

XSLT: https://www.w3schools.com/xml/xsl_intro.asp

XSD: https://www.w3schools.com/xml/schema_intro.asp

JSON: https://www.w3schools.com/js/js_json_intro.asp

Bootstrap4:
https://getbootstrap.com/docs/4.3/getting-started/introduction/

Icons:
https://fontawesome.com/icons?d=gallery

Fonts:
https://fonts.google.com/