

Final Project

Web Engineering

Backend

Ali Hamza 20i1881

Syed Muhammad Mian Abdullah 20i0457

Mahad Rahat 20i1808

15 May, 2023

Work Division

1. Mahad Rahat

Module 1 and 4: Users must sign in to use the application. They will also be able to change their passwords and will be able to recover their accounts. Basically, there are two users such as Student (may act as team lead), Teacher (may act as supervisor) and an Admin as a separate entity.

1.1 Sign Up 1.2 Sign In 1.3 Send Notifications

1.4 Change Theme 1.5 Change Email 1.6 Change Password

1.7 FYP Status 1.8 Keep Signed In 1.9 SignOut

4 : Add, delete and edit requirements

2. Ali Hamza

Teacher will create a project and then add/edit project details. Students can search for a supervisor.

2.1 View Projects 2.2 Create a Project 2.3 Assign Project Lead

2.4 View Deliverables 2.5 Add , delete, update, get Supervisor Evaluation

2.6 Find Supervisor 2.7 Delete Projects , Update project, get project

3. Mian Abdullah

Module 3 and 4: Team Selector Team Lead will add members to the project, view evaluations and send emails and requests to teachers.

3.1 Add team member 3.2 Assign Role 3.3 Send Email

3.4 Remove member 3.5 View Team Status 3.6 Send Request

3.7 View Supervisor Evaluation 3.8 View All Projects

4: view history, set deadline, create decisions , attach files, set priorities

Ali Hamza:

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main · branch · tags

Go to file Add file · Code

hamza442-ali completed e6acd76 1 hour ago 9 commits

File	Status	Last Commit
controller	completed	1 hour ago
model	completed	1 hour ago
node_modules	first part	2 days ago
routes	completed	1 hour ago
.env	first part	2 days ago
.gitignore	ignore	16 hours ago
index.js	jwt	3 hours ago
package-lock.json	first part	2 days ago
package.json	DELIVERABLES	19 hours ago

Add a README with an overview of your project. Add a README

About No description, website, or topics provided.

0 stars 1 watching 0 forks

Releases No releases published Create a new release

Packages No packages published Publish your first package

Languages JavaScript 100.0%

Post man documentation:

<https://documenter.getpostman.com/view/23997045/2s93eeRpbq>

- **Sample request and response of the APIs (which you have developed)**

1. <http://localhost:3001/projects/createProject>

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON ↻

```
1 {  
2   "title": "OOP",  
3   "id": "111qqq",  
4   "description": "storing data into database",  
5   "file": "http/this",  
6   "startDate": "2023-05-09T19:00:00.000Z",  
7   "dueDate": "2023-05-19T19:00:00.000Z",  
8   "_id": "646249af35c8bcf81f5c7dab",  
9   "__v": 0  
10 }
```

2. <http://localhost:3001/deliverables/createDeliverable>

Pretty Raw Preview Visualize JSON ↻

```
1  {
2      "id": "55pp",
3      "project": "645f8e1b5ad943eec012d23a",
4      "title": "Web 1st",
5      "description": "This is a sample deliverable",
6      "files": [
7          "file1.pdf",
8          "file2.docx"
9      ],
10     "authors": [
11         "645fccf3b9edd862353e595e",
12         "645fccf3b9edd862353e595e"
13     ],
14     "creationDate": "2023-05-15T08:00:00.000Z",
15     "lastModifiedDate": "2023-05-15T10:30:00.000Z",
16     "status": "Pending",
17     "_id": "646252f77a9339920f35cb3d",
18     "__v": 0
19 }
```

3. <http://localhost:3001/deliverables/getAllDeliverables>

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON

```
1
2 {
3     "_id": "646150aa2030e1c4e9c99816",
4     "project": "645f8e1b5ad943eec012d23a",
5     "title": "DevOps",
6     "description": "This is a sample deliverable",
7     "files": [
8         "file1.pdf",
9         "file2.docx"
10    ],
11    "authors": [
12        "645fccf3b9edd862353e595e",
13        "645fccf3b9edd862353e595e"
14    ],
15    "creationDate": "2023-05-15T08:00:00.000Z",
16    "lastModifiedDate": "2023-05-15T10:30:00.000Z",
17    "status": "Pending",
18    "__v": 0
19},
20{
21     "_id": "646251d8c0c853391fe5e55d",
22     "project": "645f8e1b5ad943eec012d23a",
23     "title": "Web 1st ".
```

4. <http://localhost:3001/evaluations/createEvaluation>

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON

```
1
2 {
3     "id": "1122tt",
4     "project": "611f65ae9e3d874b6c69eac1",
5     "evaluators": [
6         "611f65ae9e3d874b6c69ead1",
7         "611f65ae9e3d874b6c69ead2"
8     ],
9     "criteria": "Quality",
10    "totalScore": 100,
11    "obtainedScore": 85,
12    "comments": "The work is well done with minor areas for improvement.",
13    "evaluationDate": "2023-05-15T09:00:00.000Z",
14    "_id": "64624bed5c855a47309a2bea",
15    "__v": 0
16}
```

5. <http://localhost:3001/users/updateTeamlead/:id?userId=123>

6. <http://localhost:3001/projects/getAllProjects>

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON ↲

```
1
2   {
3     "_id": "645f8e1b5ad943eec012d23a",
4     "title": "DevOps",
5     "id": "12ab",
6     "description": "storing data into database",
7     "file": "http/this",
8     "startDate": "2023-05-09T19:00:00.000Z",
9     "dueDate": "2023-05-19T19:00:00.000Z",
10    "__v": 0
11  },
12  {
13    "_id": "6462458f2a9f6c5393baf465",
14    "title": "DevOps",
15    "id": "12ab",
16    "description": "storing data into database",
17    "file": "http/this",
18    "startDate": "2023-05-09T19:00:00.000Z".
```

7. <http://localhost:3001/projects/deleteProject/:id?id=12qwert>

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON ↲

```
1  {
2    "message": "Project deleted successfully"
3 }
```

8. <http://localhost:3001/evaluations/getAllEvaluations>

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON ↻

```
1 [
2   {
3     "_id": "64617115de99c7914f5ba1c1",
4     "project": "611f65ae9e3d874b6c69eac1",
5     "evaluators": [
6       "611f65ae9e3d874b6c69ead1",
7       "611f65ae9e3d874b6c69ead2"
8     ],
9     "criteria": "Quality",
10    "totalScore": 100,
11    "obtainedScore": 85,
12    "comments": "The work is well done with minor areas for improvement.",
13    "evaluationDate": "2023-05-15T09:00:00.000Z",
14    "__v": 0
15  }
16 ]
```

9. <http://localhost:3001/evaluations/deleteEvaluation/:id?id=1122tt>

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON ↻

```
1 {
2   "message": "Evaluation deleted successfully"
3 }
```

10. <http://localhost:3001/deliverables/deleteDeliverable/:id>

Body Cookies Headers (8) Test Results Status: 200

Pretty Raw Preview Visualize JSON ↻

```
1 {  
2   "message": "Deliverable deleted successfully"  
3 }
```

Deliverable Model:

```
const mongoose = require("mongoose");
const Schema = mongoose.Schema;
const User = require("./userModel"); // Import the User schema file
const Project = require("./projectModel");
const DeliverableSchema = new Schema({
  id: {
    type: String,
    required: true,
  },
  project: {
    type: Schema.Types.ObjectId,
    ref: "Project",
    required: true,
  },
  title: {
    type: String,
    required: true,
  },
  description: {
    type: String,
    required: true,
  },
  files: [
    {
      type: String,
      required: true,
    }],
  authors: [
    {
      type: Schema.Types.ObjectId,
      ref: "User",
      required: true,
    }],
  creationDate: {
    type: Date,
    default: Date.now,
  },
  lastModifiedDate: {
    type: Date,
    default: Date.now,
  },
  status: {
    type: String,
    required: true,
  },
});
```

Evaluation Schema:

```
const mongoose = require("mongoose");
const Schema = mongoose.Schema;
const EvaluationSchema = new Schema({
  id: {
    type: String,
    required: true,
  },
  project: {
    type: Schema.Types.ObjectId,
    ref: "Project",
    required: true,
  },
  evaluators: [
    {
      type: Schema.Types.ObjectId,
      ref: "User",
      required: true,
    }],
  criteria: {
    type: String,
    required: true,
  },
  totalScore: {
    type: Number,
    require: true
  },
  obtainedScore: {
    type: Number,
    required: true,
  },
  comments: {
    type: String,
    required: true,
  },
  evaluationDate: {
    type: Date,
    default: Date.now,
  },
});
```

Project Model:

```
const mongoose = require("mongoose");
const Schema = mongoose.Schema;
const ProjectSchema = new Schema({
  title: {
    type: String,
    required: true,
  },
  id: {
    type: String,
    required: true,
  },
  description: {
    type: String,
    required: true,
  },
  file: {
    type: String,
    required: true,
  },
  startDate: {
    type: Date,
    default: Date.now,
  },
  dueDate: {
    type: Date,
    default: () => {
      const currentDate = new Date();
      currentDate.setDate(currentDate.getDate() + 5);
      return currentDate;
    },
  }
});
```

User Schema:

```
const mongoose = require("mongoose");
const Schema = mongoose.Schema;
const UserSchema = new Schema ({
    userId: {
        type: String,
        required: true
    },

    name: {
        type: String,
        required: true
    },
    email: {
        type: String,
        required: true
    },
    contact: {

        type: String,
        required: true
    },
    password: {
        type: String,
        required: true
    },
    gender: {
        type: String,
        required: true
    },
    profile_picture: {
        type: String,
        required: true
    },
    registerAs: {
        type: String,
        enum: ['student', 'teacher', 'admin'],
        required: true
    }
});
```

Mahad Rahat:

- Postman documentation link of each member

<https://documenter.getpostman.com/view/27400726/2s93eeRpkf>

- Sample request and response of the APIs (which you have developed)

1. Changing email of a user

<http://localhost:3001/users/change-email>

The screenshot shows a Postman interface with a PUT request to `http://localhost:3001/users/change-email`. The request body contains two parameters: `email` (set to `mahad@hotmail.com`) and `userId` (set to `123`). The response status is 200 OK, with the message "Email changed successfully".

Key	Value	Description	... Bulk Edit
email	mahad@hotmail.com		
userId	123		

Status: 200 OK Time: 204 ms Size: 307 B Save as Example

2. Deletion of a user

<http://localhost:3001/users/del?userId=123>

DELETE http://localhost:3001/users/del?userId=123

Key	Value	Description	Bulk Edit
<input checked="" type="checkbox"/> userId	123		
Key	Value	Description	

```

1   "message": "User deleted successfully"
2
3

```

Status: 200 OK Time: 118 ms Size: 306 B Save as Example

3. User Creation:

http://localhost:3001/users/create

POST http://localhost:3001/users/create

Body

```

1 {
2   "userId": "user123",
3   "name": "John Doe",
4   "email": "john.doe@example.com",
5   "contact": "+1 123-456-7890",
6   "password": "mypassword123",
7   "gender": "Male",
8   "profile_picture": "https://example.com/profile.jpg",
9   "theme": {
10     "color": "white"
11   }
12 }

```

Status: 200 OK Time: 245 ms Size: 310 B Save as Example

4. Getting all the users

http://localhost:3001/users/all

http://localhost:3001/users/all

```

5 "contact": "+1 123-456-7890",
6 "password": "mypassword123",
7 "gender": "Male",
8 "profile_picture": "https://example.com/profile.jpg",
9 "theme": {
10   "color": "white",
11   "name": "light"
12 },
13 "registerAs": "student"
14
  
```

Status: 200 OK Time: 139 ms Size: 862 B

5. Getting all requirements

http://localhost:3001/requirements/getRequirements

http://localhost:3001/requirements/getRequirements

```

12   "author": "Mazy",
13   "text": "This is a great idea!"
14 },
15 [
16   {
17     "author": "Bob",
18     "text": "I think we need to clarify the requirements first"
19   }
20 ]
  
```

Status: 200 OK Time: 81 ms Size: 816 B

6: Creating Requirements:

The screenshot shows the Postman application interface. A POST request is being made to the URL `http://localhost:3001/requirements/createRequirement`. The 'Body' tab is selected, showing a JSON object with the following fields and values:

```
1 "title": "fetch Users",
2 "priority": "High",
3 "assignedTo": [
4   []
5 ],
6 "status": "In Progress",
7 "description": "all the users shall be displayed in a table",
8 "deadline": "2024-01-01T00:00:00.000Z",
9 "attachments": [
10   []
11 ],
12 "writtenby": "20I-0457",
13 "projectid": "20IP-0457",
14 "_id": "646227631e34ab33384d6a5",
15 "date": "2023-06-16T12:36:36.996Z",
16 "comments": [],
17 "__v": 0
```

7. Update requirements

The screenshot shows a Postman interface with the following details:

- Method:** PUT
- URL:** http://localhost:3001/requirements/updateRequirement/646227531034abd33384d5a5
- Body (JSON):**

```

1
2   "_id": "646227531034abd33384d5a5",
3   "title": "fetch Users",
4   "priority": "Medium",
5   "assignedTo": [
6     []
7   ],
8   "status": "Completed",
9   "description": "all the users shall be displayed in a table",
10  "deadline": "2024-01-01T00:00:00.000Z",
11  "attachments": [
12    []
13  ],
14  "writtenby": "20I-0457",
15  "projectid": "20IP-0457",
16  "date": "2023-06-16T12:36:35.996Z",
17  "comments": [],
18  "__v": 0
19

```

8: SignIn with jwt

The screenshot shows a Postman interface with the following details:

- Method:** POST
- URL:** http://localhost:3001/users/signin
- Body (form-data):**

KEY	VALUE	DESCRIPTION	...	Bulk Edit
userId	123			
password	qwerty			
Key	Value	Description		

9. Change Password:

The screenshot shows a POST request to `http://localhost:3001/users/change-password`. The request body contains the following data:

KEY	VALUE	DESCRIPTION
userId	123	
password	qwertyst	
Key	Value	Description

The response status is 500 Internal Server Error, with the message: "Cannot set properties of null (setting 'password')".

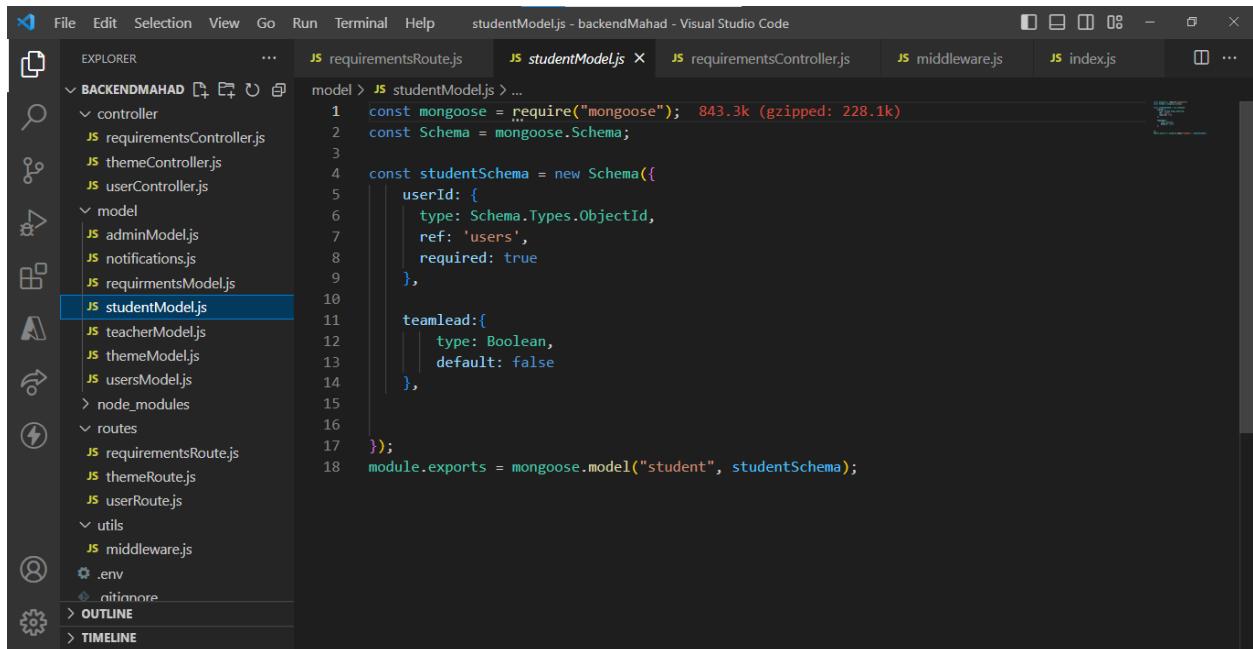
• Database Schemas.

1. User Schema:

The screenshot shows the `userSchema.js` file in Visual Studio Code. The code defines a schema for a user:

```
const UserSchema = new Schema({
  userId: {
    type: String,
    required: true
  },
  name: {
    type: String,
    required: true
  },
  email: {
    type: String,
    required: true
  },
  contact: {
    type: String,
    required: true
  },
  password: {
    type: String,
    required: true
  },
  gender: {
    type: String,
    required: true
  },
  profile_picture: {
    type: String,
    required: true
  },
  theme: {
    color: {
      type: String,
      default: "white"
    },
    name: {
      type: String,
      default: "light"
    }
  },
  registerAs: [
    type: String,
    enum: ['student', 'teacher', 'admin'],
    required: true
  ]
});
```

2. Student Schema:



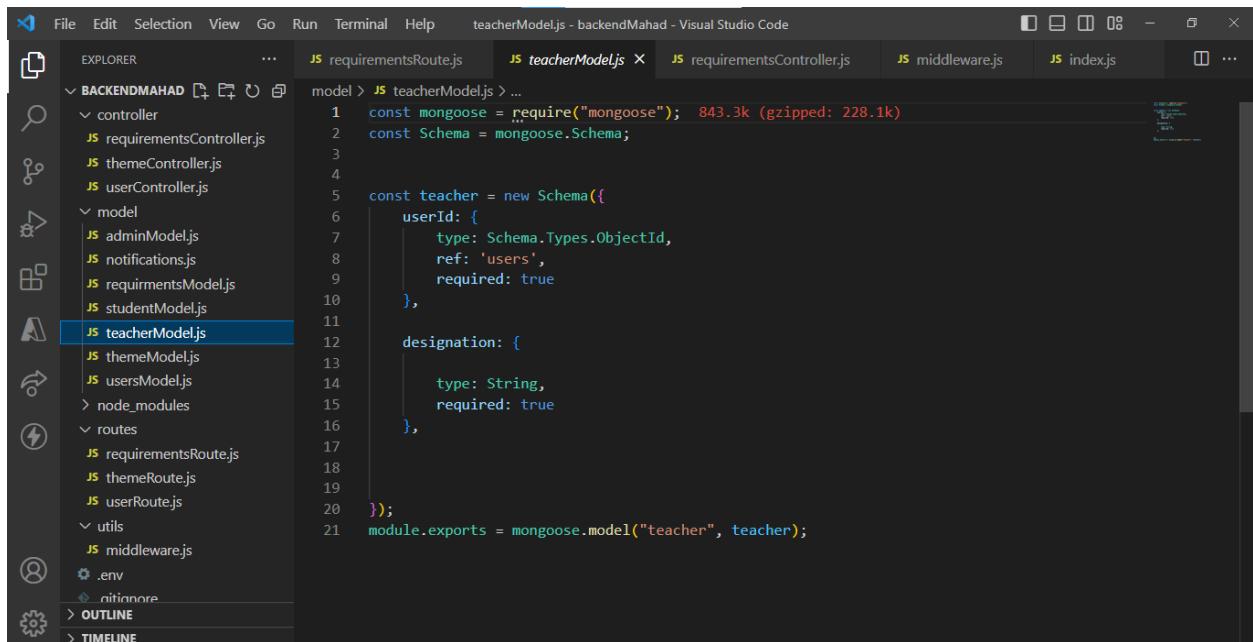
The screenshot shows the Visual Studio Code interface with the title bar "studentModel.js - backendMahad - Visual Studio Code". The left sidebar is titled "EXPLORER" and shows a tree view of the project structure under "BACKENDMAHAD". The "model" folder contains "studentModel.js", which is currently selected and highlighted with a blue background. The code editor displays the following JavaScript code:

```
const mongoose = require("mongoose");
const Schema = mongoose.Schema;

const studentSchema = new Schema({
  userId: {
    type: Schema.Types.ObjectId,
    ref: 'users',
    required: true
  },
  teamLead: {
    type: Boolean,
    default: false
  }
});

module.exports = mongoose.model("student", studentSchema);
```

3. Teacher Schema:



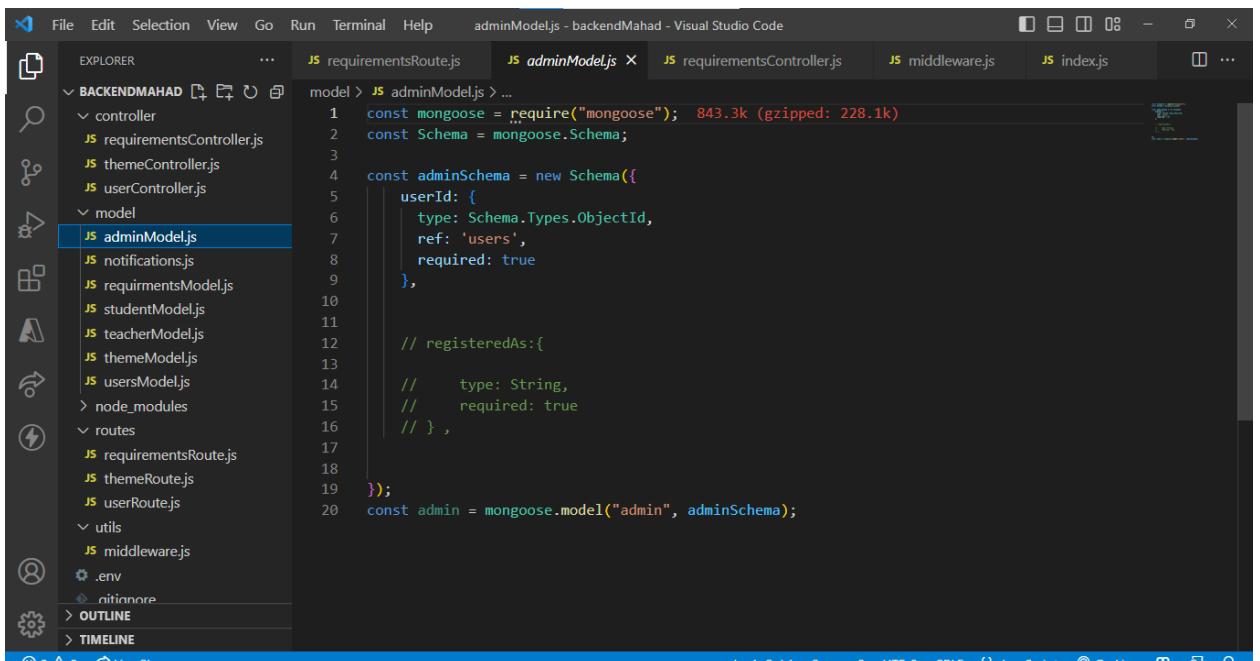
The screenshot shows the Visual Studio Code interface with the title bar "teacherModel.js - backendMahad - Visual Studio Code". The left sidebar is titled "EXPLORER" and shows a tree view of the project structure under "BACKENDMAHAD". The "model" folder contains "teacherModel.js", which is currently selected and highlighted with a blue background. The code editor displays the following JavaScript code:

```
const mongoose = require("mongoose");
const Schema = mongoose.Schema;

const teacher = new Schema({
  userId: {
    type: Schema.Types.ObjectId,
    ref: 'users',
    required: true
  },
  designation: {
    type: String,
    required: true
  }
});

module.exports = mongoose.model("teacher", teacher);
```

4. Admin Schema:



```
const mongoose = require("mongoose");
const Schema = mongoose.Schema;

const adminSchema = new Schema({
  userId: {
    type: Schema.Types.ObjectId,
    ref: 'users',
    required: true
  },
  // registeredAs: {
  //   type: String,
  //   required: true
  // } ,
});

const admin = mongoose.model("admin", adminSchema);
```

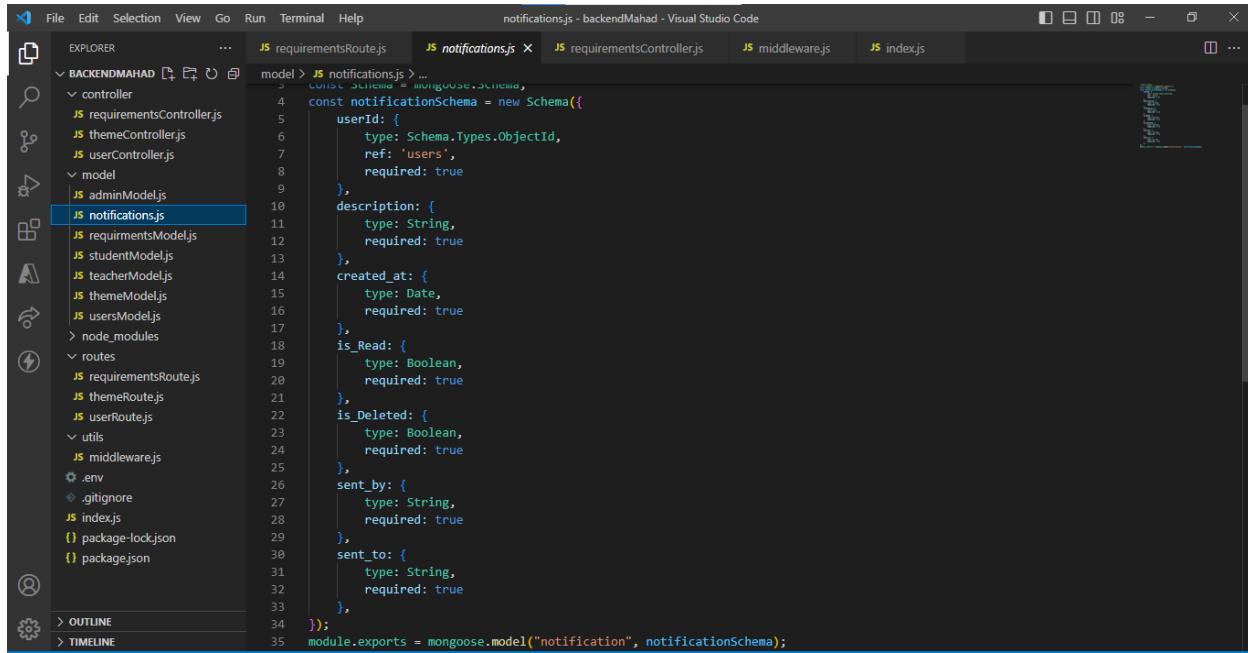
5. Requirement Schema:

```
const commentSchema = new mongoose.Schema({
  content: {
    type: String,
    required: true,
  },
  createdBy: {
    type: String,
  },
  createdAt: {
    type: Date,
    default: Date.now,
  },
});

const requirementSchema = new mongoose.Schema({
  title: {
    type: String,
    required: true,
  },
  priority: {
    type: String,
    enum: ['Low', 'Medium', 'High'],
    default: 'Low',
  },
  assignedTo: {
    type: Array,
    default: []
  },
  status: {
    type: String,
    enum: ['Pending', 'In Progress', 'Completed'],
    default: 'Pending',
  },
  date: {
    type: Date,
    default: Date.now,
  },
  description: {
    type: String,
    required: true,
  },
  deadline: {
    type: Date,
    required: true,
  },
  attachments: {
    type: Array,
    default: []
  },
  comments: [
    {
      type: [commentSchema],
    }
  ],
});

const Requirement = mongoose.model('Requirement', requirementSchema);
```

7. Notification Schema:



The screenshot shows a Visual Studio Code interface with the title bar "notificationsjs - backendMahad - Visual Studio Code". The left sidebar (EXPLORER) lists files in the "BACKENDMAHAD" folder, including "requirementsRoute.js", "controller", "model", "routes", "utils", ".env", ".gitignore", "index.js", "package-lock.json", and "package.json". The "model" folder contains "adminModel.js", "notifications.js", "requirementsModel.js", "studentModel.js", "teacherModel.js", "themeModel.js", and "usersModel.js". The "routes" folder contains "requirementsRoute.js", "themeRoute.js", "userRoute.js". The "utils" folder contains "middleware.js". The "notificationSchema.js" file is selected in the Explorer and is displayed in the main editor area. The code defines a Mongoose schema for notifications.

```
const Schema = mongoose.Schema;
const notificationSchema = new Schema({
  userId: {
    type: Schema.Types.ObjectId,
    ref: 'users',
    required: true
  },
  description: {
    type: String,
    required: true
  },
  created_at: {
    type: Date,
    required: true
  },
  is_Read: {
    type: Boolean,
    required: true
  },
  is_Deleted: {
    type: Boolean,
    required: true
  },
  sent_by: {
    type: String,
    required: true
  },
  sent_to: {
    type: String,
    required: true
  }
});
module.exports = mongoose.model("notification", notificationSchema);
```

Mian Abdullah:

Github Repository

The screenshot shows a GitHub repository page. At the top, there's a yellow banner indicating a recent push to the 'version_2' branch 31 minutes ago. On the right, there's a green 'Compare & pull request' button. Below the banner, there are navigation links for 'main' (selected), '2 branches', '0 tags', and buttons for 'Go to file', 'Add file', and 'Code'. A message box states 'Your main branch isn't protected' with a 'Protect this branch' button and a close 'X' icon. The main content area displays a single commit from 'abdullah117765' titled 'first commit' made 4 days ago. This commit includes changes to 'backend' and '.gitignore'. At the bottom, there's a light blue box prompting to 'Add a README with an overview of your project.' and a green 'Add a README' button.

<https://github.com/abdullah117765/WebProject.git>

Models

```
teamMember.js > teamMemberSchema.js > student_role.js > default.js
const mongoose = require('mongoose'); 841.1k (gzipped: 227.5k)

const teamMemberSchema = new mongoose.Schema({
  student_id: {
    type: String,
    required: true,
    unique: true,
  },
  team_lead_id: {
    type: String,
    required: true,
  },
  student_name: {
    type: String,
    required: true,
  },
  student_email: {
    type: String,
    required: true,
    unique: true,
  },
  student_status: {
    type: String,
    required: true,
  },
  requirements: {
    type: Array,
  },
  projectid: {
    type: String,
    required: true,
  },
  student_role: {
    type: String,
    default: " "
  }
});

const TeamMember = mongoose.model('TeamMember', teamMemberSchema);
```

```
const mongoose = require('mongoose');  841.1k (gzipped: 227.5k)

const commentSchema = new mongoose.Schema({
  content: {
    type: String,
    required: true,
  },
  createdBy: {
    type: String,
  },
  createdAt: {
    type: Date,
    default: Date.now,
  },
});

const requirementSchema = new mongoose.Schema({
  title: {
    type: String,
    required: true,
  },
  priority: {
    type: String,
    enum: ['Low', 'Medium', 'High'],
    default: 'Low',
  },
  assignedTo: {
    type: Array,
    default: []
  },
  status: {
    type: String,
    enum: ['Pending', 'In Progress', 'Completed'],
    default: 'Pending',
  },
  date: {
    type: Date,
    default: Date.now,
  },
  description: {
    type: String,
    required: true,
  },
  deadline: {
    type: Date,
    required: true,
  },
  attachments: {
    type: Array,
    default: []
  },
  comments: {
    type: [commentSchema],
  },
  writtenby: {//roll number
    type: String,
    required: true,
  },
  projectid: {// unique project id
    type: String,
    required: true,
  }
});
```

```
const mongoose = require('mongoose'); 841.1k (gzipped: 227.5k)

const emailSchema = new mongoose.Schema({
  sender: {
    type: String,
    required: true,
  },
  recipient: {
    type: String,
    required: true,
  },
  subject: {
    type: String,
    required: true,
  },
  text: {
    type: String,
    required: true,
  },
  sentAt: {
    type: Date,
    default: Date.now,
  },
});

const Email = mongoose.model('Email', emailSchema);

module.exports = Email;
```

Email Module

Send an email to a particular person

The screenshot shows a Postman interface for sending a POST request to `http://localhost:3001/email/send`. The request body is defined with the following parameters:

Key	Value	Description
sender	axiomshah@gmail.com	
recipient	i200457@nu.edu.pk	
subject	testing emial 3	
text	lets see if it works 3rd time	
sentAt		

The response tab shows a successful 200 OK status with the message "Email sent successfully".

Body	Cookies	Headers (8)	Test Results
1 2 3 "message": "Email sent successfully"			200 OK 288 ms 304 B Save as Example

Get all emails that were send by a particular user

The screenshot shows the Postman application interface. At the top, the URL `http://localhost:3001/email/getsent/axiomshah@gmail.com` is entered. Below the URL, there are tabs for **GET**, **Authorization**, **Headers (10)**, **Body**, **Pre-request Script**, **Tests**, and **Settings**. The **Params** tab is currently selected. In the **Query Params** section, there are two columns: **Key** and **Value**. The first row has a **Key** of "Key" and a **Value** of "Value". The second row has a **Key** of "Description" and a **Value** of "Description". To the right of these tabs, there is a **Cookies** tab. On the far right, there are buttons for **Save**, **Send**, and **</>**.

Below the main interface, there is a detailed view of the response. It shows the following JSON data:

```
21   "_id": "646210786a09668b156cc235",
22   "sender": "axiomshah@gmail.com",
23   "recipient": "i200457@nu.edu.pk",
24   "subject": "testing emial 2",
25   "text": "lets see if it works 2nd time",
26   "sentAt": "2023-05-15T10:59:04.563Z",
27   "v": 0
```

The response status is **200 OK** with **193 ms** latency and **897 B** size. There are also buttons for **Save as Example** and three dots.

Get all emails that were received by a particular user

The screenshot shows the Postman interface with the following details:

- URL:** `http://localhost:3001/email/getreceived/i200457@nu.edu.pk`
- Method:** GET
- Headers:** Authorization (with a red dot indicating it's required), Headers (10), Body (with a red dot), Pre-request Script, Tests, Settings, Cookies.
- Query Params:** A table with columns Key, Value, Description, and Bulk Edit. It contains one row with Key: "Key" and Value: "Value".
- Body:** Cookies, Headers (8), Test Results.
- Response Headers:** 200 OK, 190 ms, 478 B, Save as Example, etc.
- Response Body (Pretty JSON):**

```
1 {  
2   "_id": "64608d6e908c60bad8d128ae",  
3   "sender": "axiomshah@gmail.com",  
4   "recipient": "i200457@nu.edu.pk",  
5   "subject": "testing emial 2",  
6   "text": "lets see if it works 2nd time",  
7   "sentAt": "2023-05-14T07:27:42.256Z",  
8   "__v": 0  
9 }
```

Requirement Module

Create a Requirement

The screenshot shows a Postman interface for creating a requirement. The URL is `http://localhost:3001/requirements/createRequirement`. The method is `POST`. The body is set to `form-data`. The following fields are filled:

Key	Value
priority	High
status	In Progress
text	lets see if it works 3rd time
description	all the users shall be displayed in a table
deadline	2024-01-01
writtenby	20I-0457
projectid	20IP-0457

The JSON response is:

```
1
2   "title": "fetch Users",
3   "priority": "High",
4   "assignedTo": [
5     []
6   ],
7   "status": "In Progress",
8   "description": "all the users shall be displayed in a table",
9   "deadline": "2024-01-01T00:00:00.000Z",
10  "attachments": [
11    []
12  ],
13  "writtenby": "20I-0457",
14  "projectid": "20IP-0457",
15  "_id": "646227631934ab533384d6a6",
16  "date": "2023-06-16T12:36:35.996Z",
17  "comments": [],
18  "__v": 0
19
```

Find requirement by Projectid

The screenshot shows a Postman interface for finding requirements by project ID. The URL is `http://localhost:3001/requirements/getRequirements/20IP-0457`. The method is `GET`. The body is set to `form-data`. The following query parameters are filled:

Key	Value
Key	Value

The screenshot shows a REST API tool interface with the following details:

- Body:** Contains the JSON representation of a requirement document.
- Cookies:** Headers (8) Test Results
- Pretty:** Selected (highlighted in red).
- Raw:** Raw JSON view.
- Preview:** Preview tab.
- Visualize:** Visualize tab.
- JSON:** JSON dropdown.

```
1  [
2  {
3    "_id": "646227531034abd33384d5a5",
4    "title": "fetch Users",
5    "priority": "High",
6    "assignedTo": [
7      []
8    ],
9    "status": "In Progress",
10   "description": "all the users shall be displayed in a table",
11   "deadline": "2024-01-01T00:00:00.000Z",
12   "attachments": [
13     []
14   ],
15   "writtenby": "20I-0457",
16   "projectid": "20IP-0457",
17   "date": "2023-05-15T12:36:35.996Z",
18   "comments": [],
19   "__v": 0
--
```

update requirement

_Updating the priority to Medium and status to Complete

PUT <http://localhost:3001/requirements/updateRequirement/646227531034abd33384d5a5>

Params Authorization Headers (10) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

<input checked="" type="checkbox"/>	priority	Medium
<input checked="" type="checkbox"/>	status	Completed
<input checked="" type="checkbox"/>	text	lets see if it works 3rd time
<input checked="" type="checkbox"/>	description	all the users shall be dispalyed in a table
<input checked="" type="checkbox"/>	deadline	2024-01-01
<input checked="" type="checkbox"/>	writtenby	20I-0457
<input checked="" type="checkbox"/>	projectid	20IP-0457
	Key	Description

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON

```

1
2   "_id": "646227531034abd33384d5a5",
3   "title": "fetch Users",
4   "pxriority": "Medium",
5   "assignedTo": [
6     []
7   ],
8   "status": "Completed",
9   "description": "all the users shall be dispalyed in a table",
10  "deadline": "2024-01-01T00:00:00.000Z",
11  "attachments": [
12    []
13  ],
14  "writtenby": "20I-0457",
15  "projectid": "20IP-0457",
16  "date": "2023-06-16T12:36:35.996Z",
17  "comments": [],
18  "__v": 0
19

```

Upload attachment to a requirement

The screenshot shows a Postman interface with the following details:

URL: <http://localhost:3001/requirements/uploadAttachment/646227531034abd33384d5a5>

Method: POST

Headers (10):

- Content-Type: multipart/form-data
- Accept: application/json
- Content-Length: 1000
- Host: localhost:3001
- Connection: keep-alive
- User-Agent: PostmanRuntime/7.32.3
- TE: trailers
- Cache-Control: no-cache
- Postman-Token: 12345678901234567890123456789012

Body (10):

Params:

- attachment (checkbox checked): classdiagramm.png
- priority (checkbox): Medium
- status (checkbox): Completed
- text (checkbox): lets see if it works 3rd time
- description (checkbox): all the users shall be displayed in a table
- deadline (checkbox): 2024-01-01
- writtenby (checkbox): 20I-0457

Body (Pretty JSON):

```
1  {
2    "_id": "646227531034abd33384d5a5",
3    "title": "fetch Users",
4    "priority": "Medium",
5    "assignedTo": [
6      []
7    ],
8    "status": "Completed",
9    "description": "all the users shall be displayed in a table",
10   "deadline": "2024-01-01T00:00:00.000Z",
11   "attachments": [
12     [],
13     "classdiagramm.png"
14   ],
15   "writtenby": "20I-0457",
16   "projectid": "20IP-0457",
17   "date": "2023-05-15T12:36:35.996Z",
18   "updated": "2023-05-15T12:36:35.996Z"
}
```

Update Deadline of requirement

The screenshot shows a Postman interface with a PATCH request to `http://localhost:3001/requirements/updateDeadline/646227531034abd33384d5a5`. The 'Body' tab is selected, showing a JSON payload with a single key-value pair: `deadline: "2024-04-15"`. Below the body, the response tab displays a detailed JSON object representing a requirement document.

```
1  {
2    "_id": "646227531034abd33384d5a5",
3    "title": "fetch Users",
4    "priority": "Medium",
5    "assignedTo": [
6      []
7    ],
8    "status": "Completed",
9    "description": "all the users shall be dispalyed in a table",
10   "deadline": "2024-04-15T00:00:00.000Z",
11   "attachments": [
12     []
13     "classdiagramm.png"
14   ],
15   "writtenby": "20I-0457",
16   "projectid": "20IP-0457",
17   "date": "2023-05-15T12:36:35.996Z",
18   "comments": [],
19   "__v": 1
20 }
```

Update Priority of requirement

The screenshot shows a Postman interface for a PATCH request to `http://localhost:3001/requirements/updatePriority/646227531034abd33384d5a5`. The 'Body' tab is selected, showing a JSON payload with a single key-value pair: `priority: High`. The response body is displayed in a code block, showing a requirement document with its priority updated to "High".

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
{
  "_id": "646227531034abd33384d5a5",
  "title": "fetch Users",
  "priority": "High",
  "assignedTo": [
    []
  ],
  "status": "Completed",
  "description": "all the users shall be dispalyed in a table",
  "deadline": "2024-04-15T00:00:00.000Z",
  "attachments": [
    [],
    "classdiagram.png"
  ],
  "writtenby": "20I-0457",
  "projectid": "20IP-0457",
  "date": "2023-05-15T12:36:35.996Z",
  "comments": [],
  "__v": 1
}
```

Add comments to requirement

The screenshot shows the Postman application interface. At the top, the URL is `http://localhost:3001/requirements/addComment/646227531034abd33384d5a5`. Below the URL, there are buttons for Save, Edit, and Delete. The main area is a POST request to the same URL. The Body tab is selected, showing two key-value pairs: "content" with the value "I guess it should be displayed in a html/css card with details in I" and "createdBy" with the value "20I-0457". The Headers tab shows "Content-Type: application/json". The Response tab displays the JSON response of the request.

Body

```
1: {
2:   "_id": "646227531034abd33384d5a5",
3:   "title": "fetch Users",
4:   "priority": "High",
5:   "assignedTo": [
6:     []
7:   ],
8:   "status": "Completed",
9:   "description": "all the users shall be dispalyed in a table",
10:  "deadline": "2024-04-15T08:00:00Z",
11:  "attachments": [
12:    [],
13:    "classdiagram.png"
14:  ],
15:  "writtenby": "20I-0457",
16:  "projectid": "28IP-0457",
17:  "date": "2023-05-15T12:36:35.996Z",
18:  "comments": [
19:    {
20:      "content": "I guess it should be displayed in a html/css card with details in lower section",
21:      "createdBy": "20I-0457",
22:      "_id": "64624dfab6ff94f911211ea1",
23:      "createdAt": "2023-05-15T15:21:38.437Z"
24:    }
25:  ]
26: }
```

Status: 200 OK Time: 415 ms Size: 814 B

Display comments of a requirement

The screenshot shows the Postman interface with a GET request to `http://localhost:3001/requirements/getComments/646227531034abd33384d5a5`. The 'Body' tab is selected, showing two parameters: 'content' and 'createdBy'. The 'content' parameter has a value of "I guess it should be displayed in a html/css card with details in lower section". The 'createdBy' parameter has a value of "20I-0457". Below the body, the JSON response is displayed in a pretty-printed format:

```
1 [ {  
2   "content": "I guess it should be displayed in a html/css card with details in lower section",  
3   "createdBy": "20I-0457",  
4   "_id": "64624dfab60f94f911211ea1",  
5   "createdAt": "2023-05-15T15:21:30.437Z"  
6 } ]
```

Team Member Module

Adding team member

The screenshot shows the Postman application interface for making a POST request to the endpoint `http://localhost:3001/teamMember/addMember`. The request body is defined as form-data with the following fields:

Key	Value
student_id	20I-1808
team_lead_id	20I-0987
student_name	Mahad khan
student_email	i201808@gmail.com
student_status	active
projectid	20IP-0457

The response body is displayed in JSON format:

```
1 "message": "Team member added successfully"
```

Getting all team member by Projectid

http://localhost:3001/teamMember/getMembers/20IP-0457

The screenshot shows the Postman application interface. At the top, there is a header bar with tabs for 'GET' (selected), 'Params', 'Authorization', 'Headers (10)', 'Body', 'Pre-request Script', 'Tests', and 'Settings'. Below this is a 'Query Params' section with a table:

Key	Value
Key	Value

Below the table are tabs for 'Body', 'Cookies', 'Headers (8)', and 'Test Results'. The 'Body' tab is selected and contains a JSON response with line numbers from 1 to 24. The JSON data is as follows:

```
1  {
2   "_id": "646268ac22f31bb1edb99e142",
3   "student_id": "20I-0457",
4   "team_lead_id": "20I-0987",
5   "student_name": "syed abdullah shah",
6   "student_email": "axiomshah@gmail.com",
7   "student_status": "active",
8   "requirements": [],
9   "projectid": "20IP-0457",
10  "__v": 0
11 },
12 [
13   {
14     "_id": "646258d122f31bb1edb99e144",
15     "student_id": "20I-3333",
16     "team_lead_id": "20I-0987",
17     "student_name": "jk",
18     "student_email": "jk@gmail.com",
19     "student_status": "active",
20     "requirements": [],
21     "projectid": "20IP-0457",
22     "__v": 0
23   }
24 ]
```

Getting a single team member by id

http://localhost:3001/teamMember/findMember/646258d122f31bb1db99e144

The screenshot shows the Postman application interface. At the top, there is a header bar with tabs for 'GET' (selected), 'Authorization', 'Headers (10)', 'Body', 'Pre-request Script', 'Tests', and 'Settings'. Below the header, the URL 'http://localhost:3001/teamMember/findMember/646258d122f31bb1db99e144' is entered. Under the 'Params' tab, there is a table labeled 'Query Params' with two rows. The first row has an empty 'Key' field and an empty 'Value' field. The second row has a 'Key' field containing 'Key' and a 'Value' field containing 'Value'. In the main body area, there are tabs for 'Body', 'Cookies', 'Headers (8)', and 'Test Results'. The 'Body' tab is selected and contains sub-tabs for 'Pretty' (selected), 'Raw', 'Preview', 'Visualize', and 'JSON' (with a dropdown menu). Below these tabs is a JSON-formatted response with line numbers 1 through 11. The response object is:

```
1  {
2   "id": "646258d122f31bb1db99e144",
3   "student_id": "20I-3333",
4   "team_lead_id": "20I-0987",
5   "student_name": "JK",
6   "student_email": "jk@gmail.com",
7   "student_status": "active",
8   "requirements": [],
9   "projectid": "20IP-0457",
10  "__v": 0
11 }
```

Updating the role of a team member

The screenshot shows a Postman interface with the following details:

- Request URL:** PATCH http://localhost:3001/teamMember/assignRole/64625d47d9b8d5e7de60ef99
- Method:** PATCH
- Headers:** (10)
- Body:** (selected)
- Params:** (green dot)
- Authorization:** (green dot)
- Tests:**
- Settings:**
- Form Data Fields:**

Key	Value
<input checked="" type="checkbox"/> student_role	member
<input type="checkbox"/> team_lead_id	20I-0987
<input type="checkbox"/> student_name	syed Abdullah shah
<input type="checkbox"/> student_email	axiomshah@gmail.com
- Body Options:** Pretty, Raw, Preview, Visualize, JSON (selected),
- JSON Response (Pretty Print):**

```
1  "_id": "64625d47d9b8d5e7de60ef99",
2  "student_id": "20I-0333",
3  "team_lead_id": "20I-0987",
4  "student_name": "JK",
5  "student_email": "jk@gmail.com",
6  "student_status": "active",
7  "requirements": [],
8  "projectid": "20IP-0457",
9  "student_role": "member",
10 " __v": 0
```

Assigning a requirement to a team member

The screenshot shows the Postman application interface. At the top, it displays a 'PATCH' method and the URL `http://localhost:3001/teamMember/assignReq/64625d47d9b8d5e7de60ef99`. Below the URL, there are tabs for 'Params', 'Authorization', 'Headers (10)', 'Body', 'Pre-request Script', 'Tests', and 'Settings'. The 'Body' tab is currently selected and has a red underline. Under the 'Body' tab, there are several input fields: 'requirements' (checkbox checked, value: 646227531034abd33384d5a5), 'team_lead_id' (checkbox unchecked, value: 20I-0987), 'student_name' (checkbox unchecked, value: syed Abdullah shah), and 'student_email' (checkbox unchecked, value: axiomshah@gmail.com). Below these fields, there are tabs for 'Body', 'Cookies', 'Headers (8)', and 'Test Results'. The 'Body' tab is selected and has a red underline. Under the 'Body' tab, there are four buttons: 'Pretty' (selected), 'Raw', 'Preview', and 'Visualize'. To the right of these buttons is a dropdown menu set to 'JSON'. Below the buttons, the JSON response is displayed in a code editor-like format with line numbers from 1 to 14. The JSON content is as follows:

```
1 {
2   "_id": "64625d47d9b8d5e7de60ef99",
3   "student_id": "20I-0333",
4   "team_lead_id": "20I-0987",
5   "student_name": "JK",
6   "student_email": "jk@gmail.com",
7   "student_status": "active",
8   "requirements": [
9     "646227531034abd33384d5a5"
10 ],
11   "projectid": "20IP-0457",
12   "student_role": "member",
13   "__v": 0
14 }
```

DatabaseMongodb

Fast_Deadline-Solution.teammembers

STORAGE SIZE: 20KB LOGICAL DATA SIZE: 437B TOTAL DOCUMENTS: 2 INDEXES TOTAL SIZE: 20KB

Find Indexes Schema Anti-Patterns Aggregation Search Indexes Charts

INSERT DOCUMENT

Filter Type a query: { field: 'value' }

Reset Apply More Options

QUERY RESULTS: 1-2 OF 2

```
_id: ObjectId('646258ac22f31bb1db99e142')
student_id: "20I-0457"
team_lead_id: "20I-0987"
student_name: "syed abdullah shah"
student_email: "axiomshah@gmail.com"
student_status: "active"
requirements: Array
projectid: "20IP-0457"
__v: 0

_id: ObjectId('646258d122f31bb1db99e144')
student_id: "20I-3333"
team_lead_id: "20I-0987"
student_name: "jih@gmail.com"
student_email: "jih@gmail.com"
student_status: "active"
requirements: Array
projectid: "20IP-0457"
__v: 0
```

Fast_Deadline-Solution.emails

STORAGE SIZE: 6KB LOGICAL DATA SIZE: 532B TOTAL DOCUMENTS: 3 INDEXES TOTAL SIZE: 36KB

Find Indexes Schema Anti-Patterns Aggregation Search Indexes Charts

INSERT DOCUMENT

Filter Type a query: { field: 'value' }

Reset Apply More Options

```
_id: ObjectId('645fc5b90430d5615ce22ed')
sender: "i200457@nu.edu.pk"
recipient: "axiomshah@gmail.com"
subject: "testing email"
text: "lets see if it works"
sentAt: 2023-05-13T12:02:35.499+00:00
__v: 0

_id: ObjectId('64600d6e908c60bad8d128ae')
sender: "axiomshah@gmail.com"
recipient: "i200457@nu.edu.pk"
subject: "testing email 2"
text: "lets see if it works 2nd time"
sentAt: 2023-05-14T07:27:42.256+00:00
__v: 0

_id: ObjectId('646220893cf7ca95020a24a')
sender: "axiomshah@gmail.com"
recipient: "i200457@nu.edu.pk"
subject: "testing email 3"
text: "lets see if it works 3rd time"
sentAt: 2023-05-15T12:07:37.303+00:00
__v: 0
```

Fast_Deadline-Solution.requirements

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 493B TOTAL DOCUMENTS: 1 INDEXES TOTAL SIZE: 20KB

[Find](#) [Indexes](#) [Schema Anti-Patterns](#) [Aggregation](#) [Search Indexes](#) [Charts](#)

Filter

Type a query: { field: 'value' }

QUERY RESULTS: 1-1 OF 1

```
_id: ObjectId('646227531034abd33384d5a5')
title: "fetch Users"
priority: "High"
assignedTo: []
  0: []
status: "Completed"
description: "all the users shall be dispalyed in a table"
deadline: 2024-04-15T00:00:00.000+00:00
attachments: []
  0: []
    1: "classdiagram.png"
writtenby: "20I-0457"
projectid: "20IP-0457"
date: 2023-05-15T12:36:35.996+00:00
comments: []
  0: Object
    content: "I guess it should be displayed in a html/css card with details in lowe_"
    createdBy: "20I-0457"
    _id: ObjectId('64624dfab60f94f911211ea1')
    createdAt: 2023-05-15T15:21:30.437+00:00
--v: 2
```