

User Management system Design Document

Introduction- Welcome to our Note taking Portal (User Management system). Enjoy a seamless sign up and login experience. Once logged in, explore all the users effortlessly. You can add user by clicking on "Add user" button. This application offers a user-friendly interface for efficient navigation. After adding and viewing user securely log out using the designated button.

UI design-

1. **Create account Page-** I provided a form with some fields like name, email, mobile, gender, password, source, city & state. All fields are necessary to fill. For it I used validation in the form. After successfully account creation user can go to login page and can log in.

The screenshot shows the 'Create New Account' form within the 'User Management System' interface. The form is titled 'Create New Account' and contains the following fields and options:

- User's Name:** A text input field with a placeholder 'Enter your name' and a validation message 'Please fill name field !'.
- User's Email:** A text input field with a placeholder 'Enter your Email' and a validation message 'Please fill email field !'.
- User's Phone:** A text input field with a placeholder 'Enter your Phone no.' and a validation message 'Please fill mobile field !'.
- User's Password:** A password input field with a placeholder '*****'.
- Gender:** Radio buttons for 'Male', 'Female', and 'others', with a validation message 'Please fill gender field !'.
- How did you hear about us?:** Radio buttons for 'LinkedIn', 'Friends', 'Job Portal', and 'others', with a validation message 'Please fill source field !'.
- City:** A dropdown menu with 'select one' and a validation message 'Please fill city field !'.
- State:** A text input field with a placeholder 'Type state name ...' and a validation message 'Please fill state field !'.

A 'Save' button is located at the bottom right of the form.

The screenshot shows the 'User Login' form within the 'User Management System' interface. The form is titled 'User Login' and contains the following fields:

- Email id:** A text input field with the value 'paritosh3tmishra@gmail.com'.
- Password:** A password input field with a placeholder '*****'.

A success message overlay is displayed in the center of the screen, featuring a green checkmark icon and the text 'Account Created Successfully !!'. An 'OK' button is located at the bottom right of the overlay.

- Login Page** - I provided a form with two fields like email & password. All fields are necessary to fill. For it I used validation in the form. User can login by filling both fields. Once logged in, you will be directed to home page where you can explore all the users details and perform crud (create, retrieve, update & delete) operations.

User Management System

Create AccountLogin

User Login

Email id

Paritosh31mishra@gmail.com

Password

Login

- Users Page (home page)** - Once you successfully logged in you will directed to user page. On user page You can explore all the users effortlessly. All the available users are fetched and shown in form of table with fields title, content, edit, delete. Each row represent a user. By clicking on edit button you can edit the user and by clicking on delete button you can delete a user. And by clicking on the link view details you can see the user in details.

Features:

- Search box** - you can search users by categories like name, email & mobile number.

User Management System

Add Userparitosh Logout

Name

rahul







S.No.	Name	Email	Mobile	Delete	Edit	View	Last modified
1	rahul	rahul@gmail.com	6786786578			view details	1/7/2024, 8:59:57 AM

2. **Delete** - By clicking on delete button in each row you can delete that user from database.

User Management System

localhost:3000 says
Record deleted successfully !
OK

[Add User](#) [paritosh Logout](#)

S.No.	Name	Email	Mobile	Delete	Edit	View	Last modified
1	Rahul	rahul@gmail.com	6786786578			view details	1/7/2024, 9:01:03 AM
2	paritosh	paritosh@gmail.com	7665766767			view details	1/7/2024, 8:59:42 AM
3	sharuuuuuu	admin@gmail.com	567675676			view details	1/6/2024, 9:52:47 PM

3. **Edit** - For Editing existing user.

User Management System

[Add User](#) [paritosh Logout](#)

Edit User Details

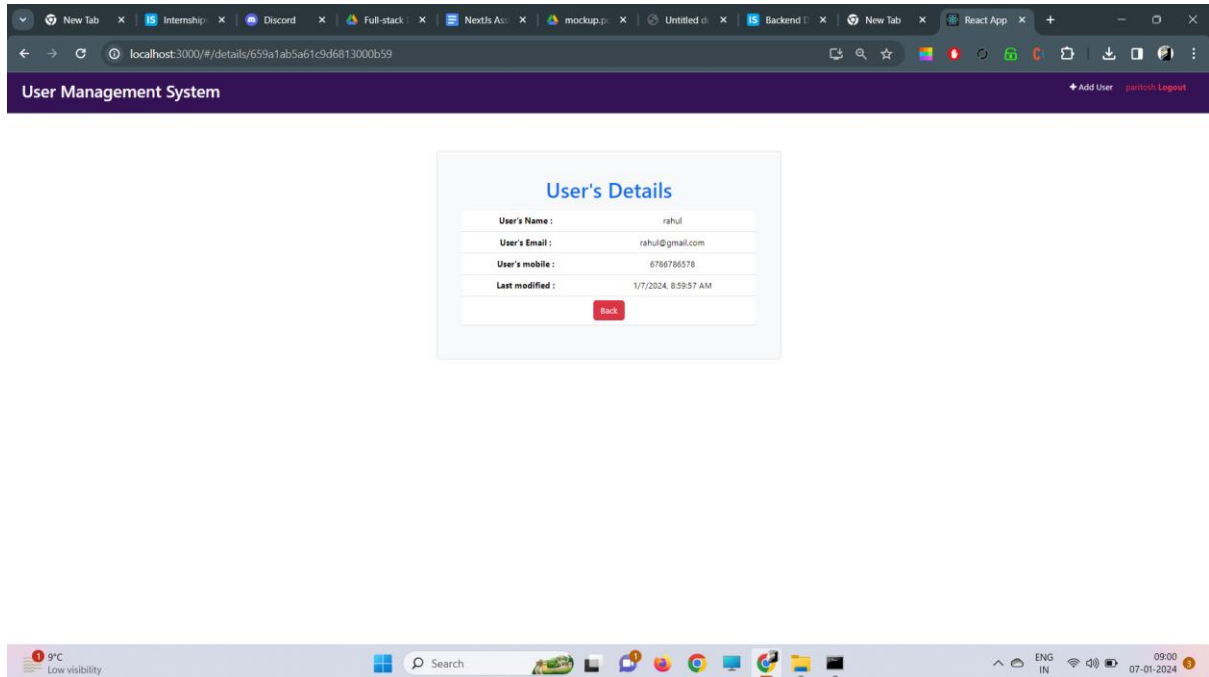
User's Name :

User's Email :

User's mobile :

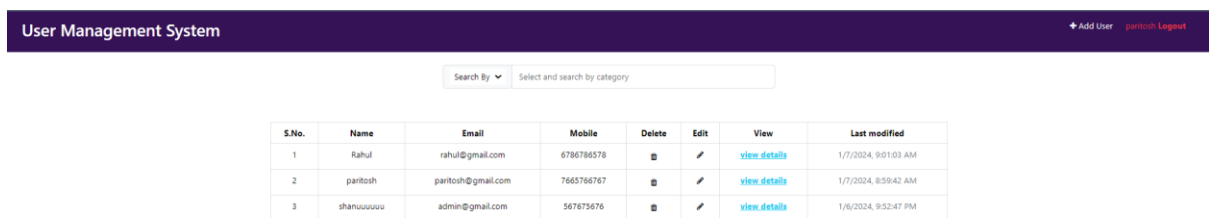
update user

4. View details - view user in details.



In right side above I provided two buttons **Adduser** and **logout**.

- Adduser for adding new user.
- Logout for logged out.



Api endpoints used –

- Get all user - "<http://localhost:3333/retrieveuser>";
- For edit user - "<http://localhost:3333/edituser>";
- For deleting user– "<http://localhost:3333/deleteuser/>"+id;
- View details- "<http://localhost:3333/user/>"+userid

4. **Adduser** - When you click on Adduser button you will be directed to add user page that have a form with fields Name, Email & mobile.

I have been used validation in this form, all three fields are required to submit the form.

User Management System

◆ Add User paritosh Logout

Add User

User's Name :

User's Email :

User's mobile :

Add user

User Management System

◆ Add User paritosh Logout

Search By ▼ Select and search by category

S.No.	Name	Email	Mobile	Delete	Edit	View	Last modified
1	anu	anu@gmail.com	65766978			view details	1/7/2024, 9:27:05 AM
2	paritosh	paritosh@gmail.com	7665766767			view details	1/7/2024, 8:59:42 AM
3	shanuuuuuu	admin@gmail.com	567673676			view details	1/6/2024, 9:52:47 PM

User Added Successfully !!

OK

While submitting the form /adduser Api have been called from the backend nodejs.

Api endpoints used –

- Url = "<http://localhost:3333/adduser>"
- Method: Post
- Body: { name, email, mobile,date }
- author field get from localStorage.getItem("fullname");

Backend Services –

1. /account –

```
const accountapi = require("./accountapi");
app.use("/account", accountapi);
```

- **Description –**
 - RestApis are the secure apis. So path of the api call /account has been changed to /accountapi. It protect database from direct access.

2. /retrieveuser –

```
const manageuser = require("./userapi");
app.use("/retrieveuser", manageuser);
```

- **Description –**
 - RestApis are the secure apis. So path of the api call /retrieveuser has been changed to /manageuser. It protect database from direct access.

3. /edituser –

```
const manageuser = require("./userapi");
app.use("/edituser", manageuser);
```

- **Description –**
 - RestApis are the secure apis. So path of the api call /edituser has been changed to /manageuser. It protect database from direct access.

4. /deleteuser –

```
const manageuser = require("./userapi");
app.use("/deleteuser", manageuser);
```

- **Description –**
 - RestApis are the secure apis. So path of the api call /deleteuser has been changed to /manageuser. It protect database from direct access.

5. /adduser –

```
const manageuser = require("./userapi");
app.use("/adduser", manageuser);
```

- **Description –**
 - RestApis are the secure apis. So path of the api call /adduser has been changed to /manageuser. It protect database from direct access.

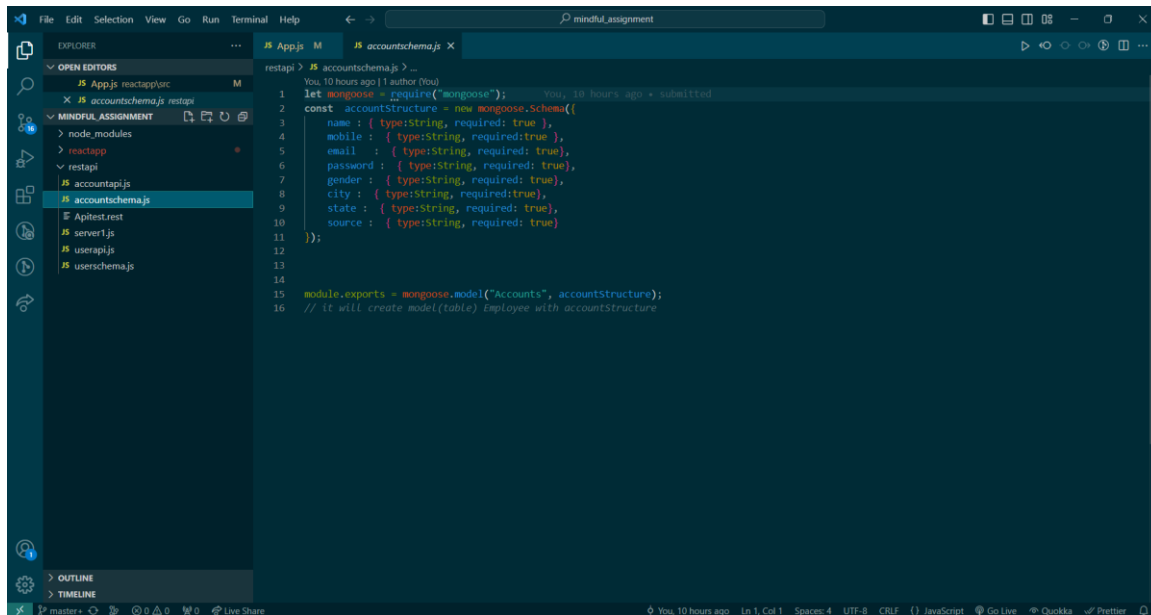
6. /user –

```
const manageuser = require("./userapi");  
app.use("/user", manageuser);
```

- **Description –**
 - RestApis are the secure apis. So path of the api call /user has been changed to /manageuser. It protect database from direct access.

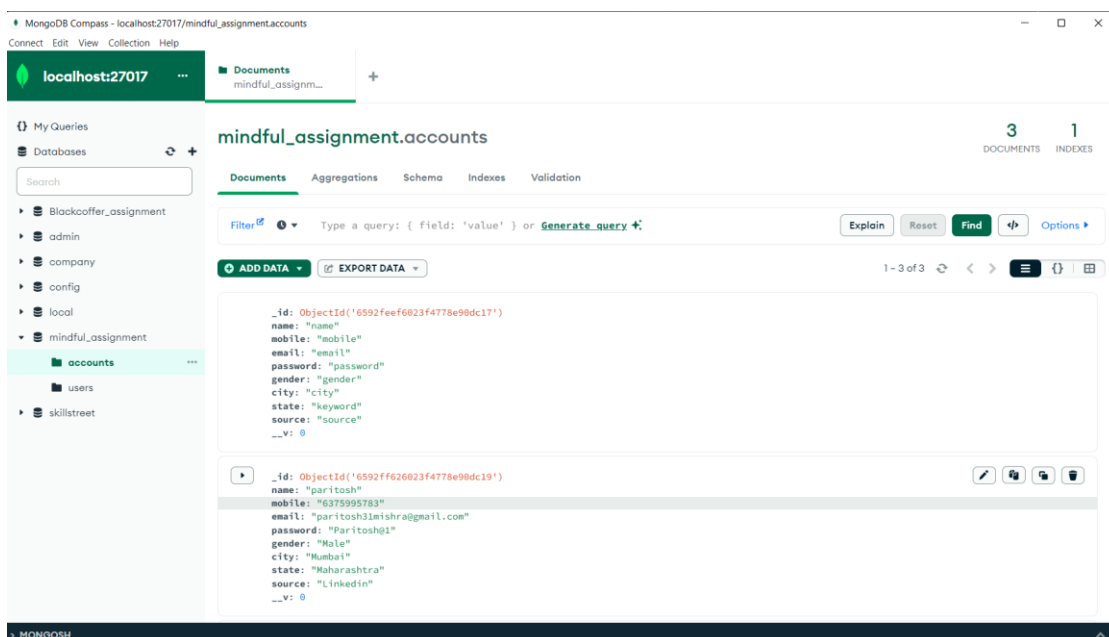
Mongo DB -

- **Account –**



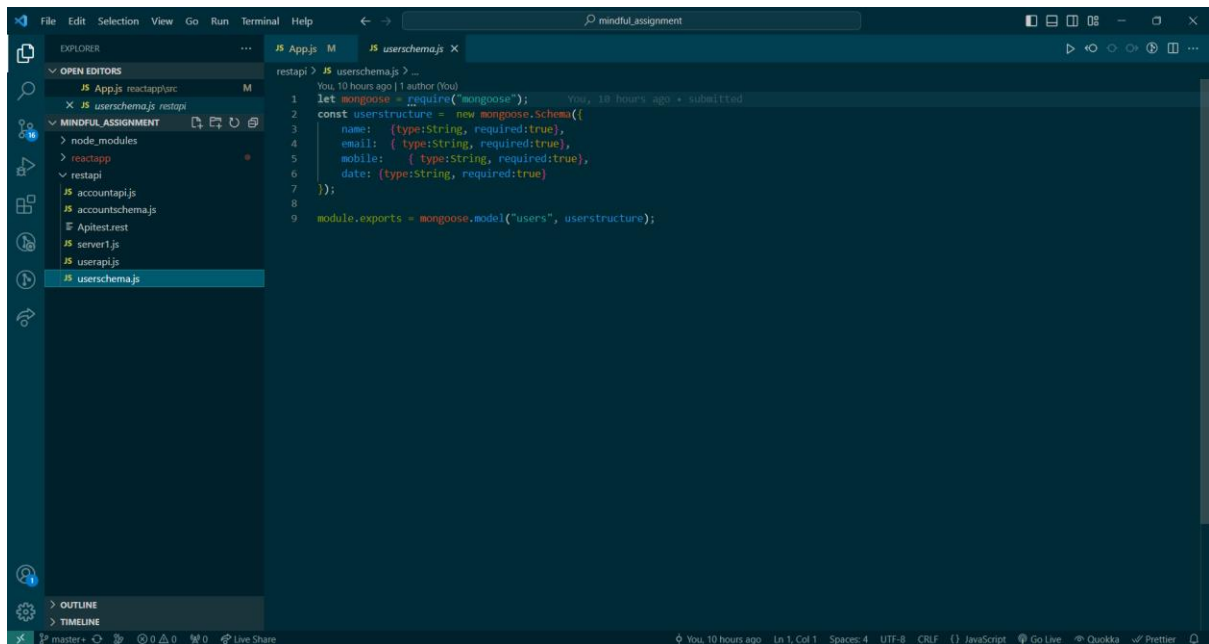
```
1 let mongoose = require("mongoose");  
2 const accountStructure = new mongoose.Schema({  
3   name : { type:String, required: true },  
4   mobile : { type:String, required:true },  
5   email : { type:String, required: true},  
6   password : { type:String, required: true},  
7   gender : { type:String, required: true},  
8   city : { type:String, required:true},  
9   state : { type:String, required: true},  
10  source : { type:String, required: true}  
11 });  
12  
13  
14 module.exports = mongoose.model("Accounts", accountStructure);  
15 // it will create model(table) employee with accountStructure  
16
```

Account schema

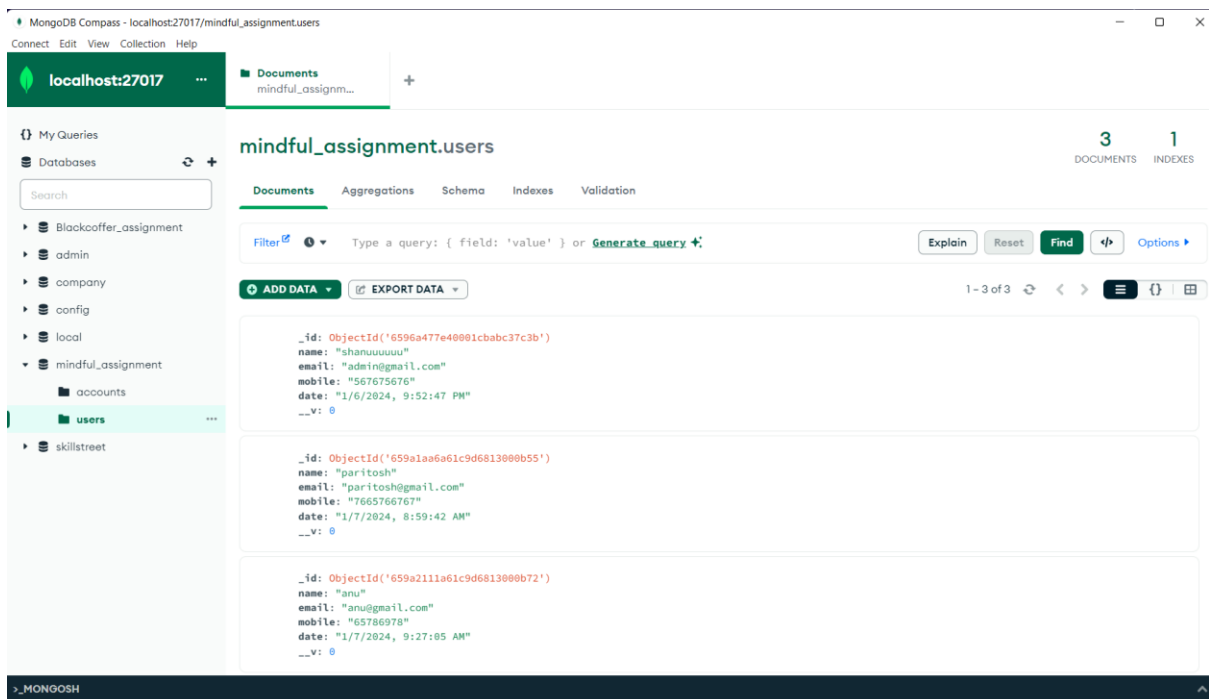


Account database

- Users –



User schema



user database

APIs testing -

- /account –

The screenshot shows the VS Code interface with the Explorer, Open Editors, and REST Client tabs. The REST Client tab is active, showing a REST API client configuration for the /account endpoint. The request is a POST request to http://localhost:3333/account with a Content-Type of application/json. The response is a 201 Created status with a JSON body containing user details.

```
restapi > Apitester > GET /retrieveuser
You, 1 second ago | 1 author (You)

1 Send Request
2 POST http://localhost:3333/account
3 Content-Type: application/json
4 {
5   "uname": "name",
6   "umobile": "mobile",
7   "uemail": "email",
8   "upassword": "password",
9   "ugender": "gender",
10  "ucity": "city",
11  "ustate": "keyword",
12  "usource": "source"
13 }
14 //spacing matter
15 Send Request
16 GET http://localhost:3333/retrieveuser
17 You, 12 seconds ago • Uncommitted changes
18 {
19   "name": "name",
20   "mobile": "mobile",
21   "email": "email",
22   "password": "password",
23   "gender": "gender",
24   "city": "city",
25   "state": "keyword",
26   "source": "source",
27   "_id": "659a27ca206a727812c7964c",
28   "_v": 0
29 }
```

- /adduser –

The screenshot shows the VS Code interface with the Explorer, Open Editors, and REST Client tabs. The REST Client tab is active, showing a REST API client configuration for the /adduser endpoint. The request is a POST request to http://localhost:3333/adduser with a Content-Type of application/json. The response is a 201 Created status with a JSON body containing user details.

```
restapi > Apitester > POST /adduser
You, 1 second ago • Uncommitted changes

1 Send Request
2 POST http://localhost:3333/adduser
3 Content-Type: application/json
4 {
5   "name": "aryan",
6   "mobile": "876786887",
7   "email": "aryan@gmail.com",
8   "date": "1/6/2024, 9:52:47 PM"
9 }
10 //spacing matter
11 Send Request
12 GET http://localhost:3333/retrieveuser
13 You, 1 second ago • Uncommitted changes
14 {
15   "name": "aryan",
16   "email": "aryan@gmail.com",
17   "mobile": "876786887",
18   "date": "1/6/2024, 9:52:47 PM",
19   "_id": "659a295b206a727812c79655",
20   "_v": 0
21 }
```

- /retrieveuser-

The screenshot shows a REST client interface in VS Code. The request is a GET to `http://localhost:3333/retrieveuser`. The response is a 201 status code with a JSON body containing user details.

```

1 HTTP/1.1 201 Created
2 X-Powered-By: Express
3 Access-Control-Allow-Origin: *
4 Content-Type: application/json; charset=utf-8
5 Content-Length: 415
6 ETag: W/"19f-Kl0Wae08139jX+D1XALqF5j1V7Q"
7 Date: Sun, 07 Jan 2024 04:25:36 GMT
8 Connection: close
9
10 {
11   "id": "6596a477e40001chabc37c3b",
12   "name": "shamuuuuu",
13   "email": "admin@gmail.com",
14   "mobile": "567675676",
15   "date": "1/6/2024, 9:52:47 PM",
16   "_v": 0
17 },
18 {
19   "id": "659a1aa6a61c9d6813000b55",
20   "name": "paritosh",
21   "email": "paritosh@gmail.com",
22   "mobile": "7665766767",
23   "date": "1/7/2024, 8:59:42 AM",
24   "_v": 0
25 },
26 {
27   "id": "659a2111a61c9d6813000b72",
28   "name": "anuui",
29   "email": "anu@gmail.com",
30   "mobile": "65786978",
31   "date": "1/7/2024, 9:53:26 AM",
32   "_v": 0
33 }

```

- /deleteuser-

The screenshot shows a REST client interface in VS Code. The request is a DELETE to `http://localhost:3333/deleteuser/659a1aa6a61c9d6813000b55`. The response is a 201 status code with a JSON body containing a success message.

```

1 HTTP/1.1 201 Created
2 X-Powered-By: Express
3 Access-Control-Allow-Origin: *
4 Content-Type: application/json; charset=utf-8
5 Content-Length: 44
6 ETag: W/"2c-7myckp3kZBungTUQyva0LlK1sMs"
7 Date: Sun, 07 Jan 2024 04:27:34 GMT
8 Connection: close
9
10 {
11   "message": "Record deleted successfully !"
12 }

```

Setup Details:

- Clone the repository <link> to a directory.
- Start the react server by going to the directory of the mindful_assignment > reactapp and run 'npm start'
- Start the node js server by going to the directory of mindful_assignment > nodeapp and run 'nodemon server1.js'.

Here server1.js is the name of the node js service file.

