Experiment - 6

Aim

Implement authentication and user roles with JWT

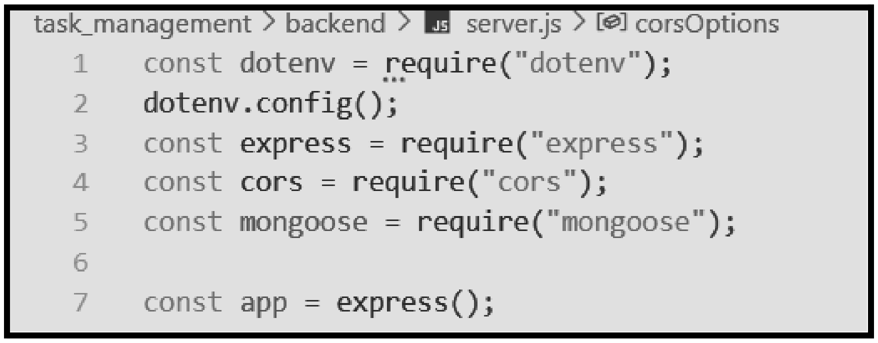
INPUT AND OUTPUT :

Backend :

A] Code :

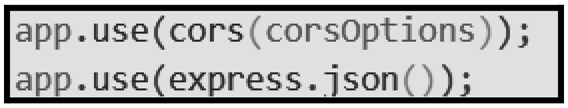
a) Server Setup (server.js)

- Show how **Express is configured**, **CORS setup**, **middleware**, and **database connection**.
- Include .env usage with process.env.MONGO_URI to highlight security.

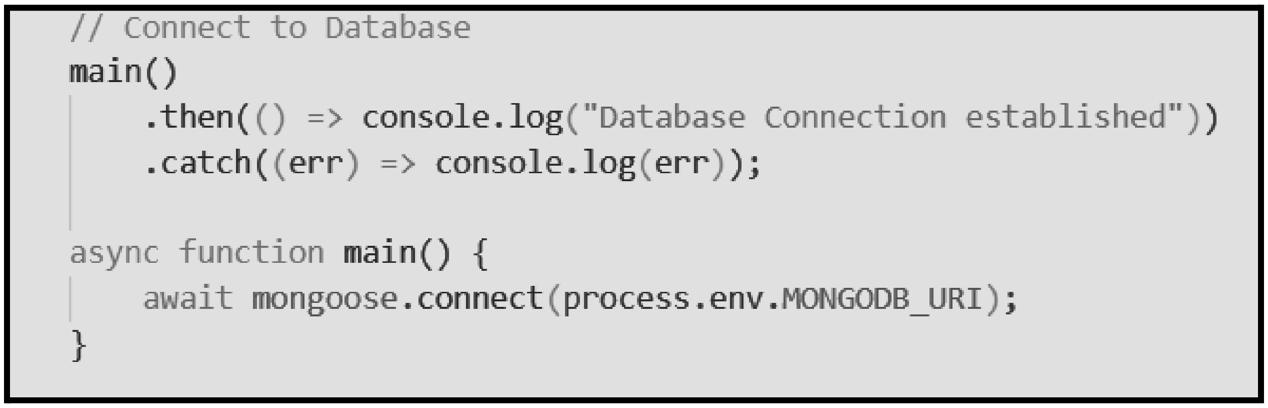


task_management > backend > server.js > corsOptions

```
const dotenv = require("dotenv");
dotenv.config();
const express = require("express");
const cors = require("cors");
const mongoose = require("mongoose");

const app = express();
```



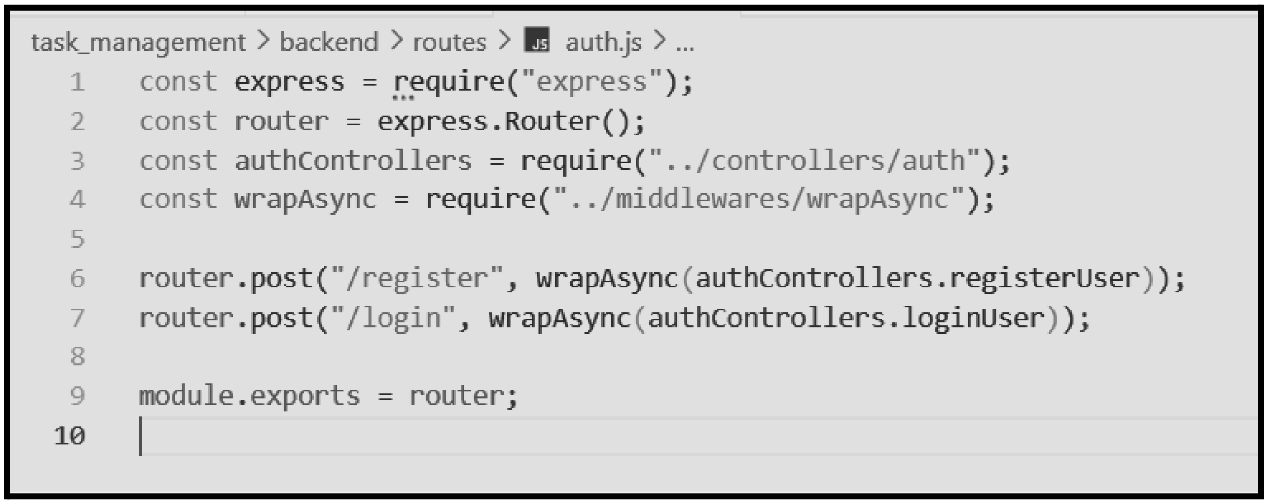
```
app.use(cors(corsOptions));
app.use(express.json());
```



```
// Connect to Database
main()
    .then(() => console.log("Database Connection established"))
    .catch((err) => console.log(err));

async function main() {
    await mongoose.connect(process.env.MONGODB_URI);
}
```

b) Authentication Route (routes/auth.js)

- Include **login and registration endpoints**, showing JWT issuance.

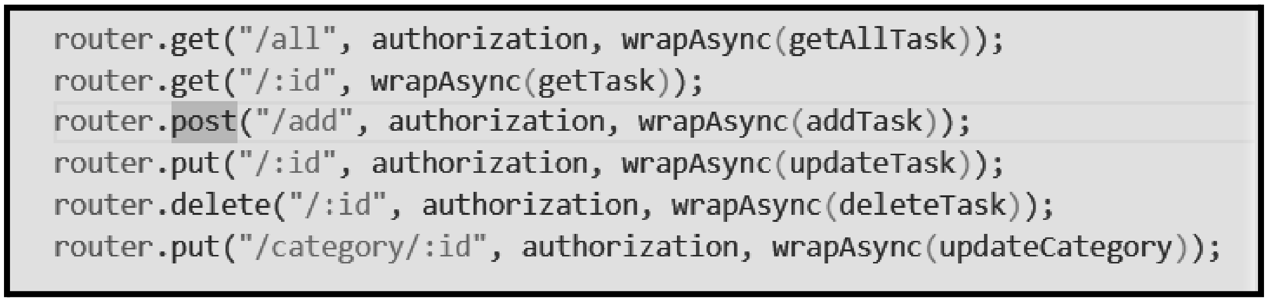


task_management > backend > routes > auth.js > ...

```
const express = require("express");
const router = express.Router();
const authControllers = require("../controllers/auth");
const wrapAsync = require("../middlewares/wrapAsync");

router.post("/register", wrapAsync(authControllers.registerUser));
router.post("/login", wrapAsync(authControllers.loginUser));

module.exports = router;

```

c) Protected Task Routes (routes/task.js)

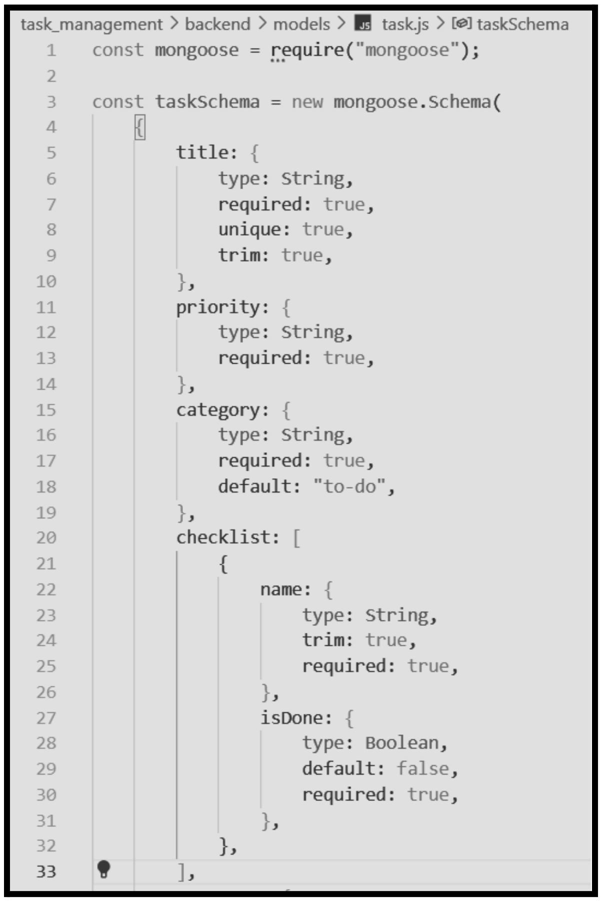
- Show **how authorization middleware is used** for secure endpoints.



```
router.get("/all", authorization, wrapAsync(getAllTask));
router.get("/:id", wrapAsync(getTask));
router.post("/add", authorization, wrapAsync(addTask));
router.put("/:id", authorization, wrapAsync(updateTask));
router.delete("/:id", authorization, wrapAsync(deleteTask));
router.put("/category/:id", authorization, wrapAsync(updateCategory));
```

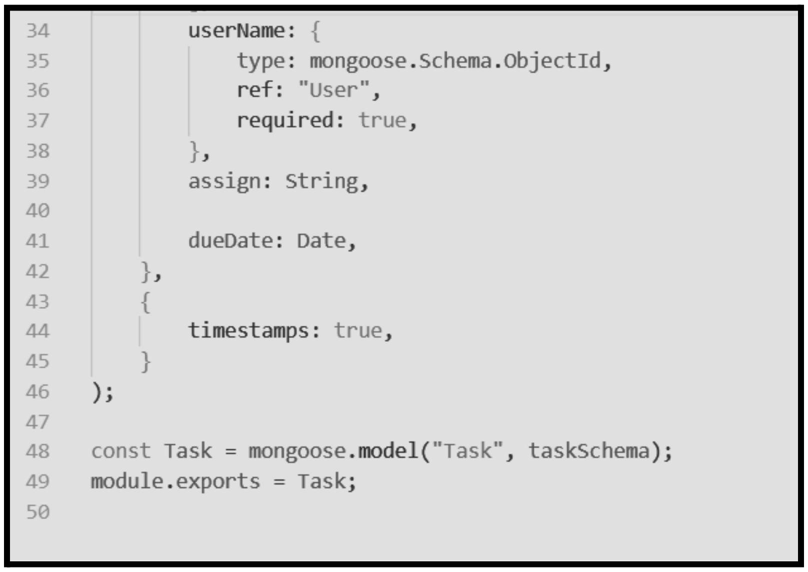
d) Task Model (models/task.js)

- Show **Mongoose schema**, especially required fields like title, priority, category, userName.

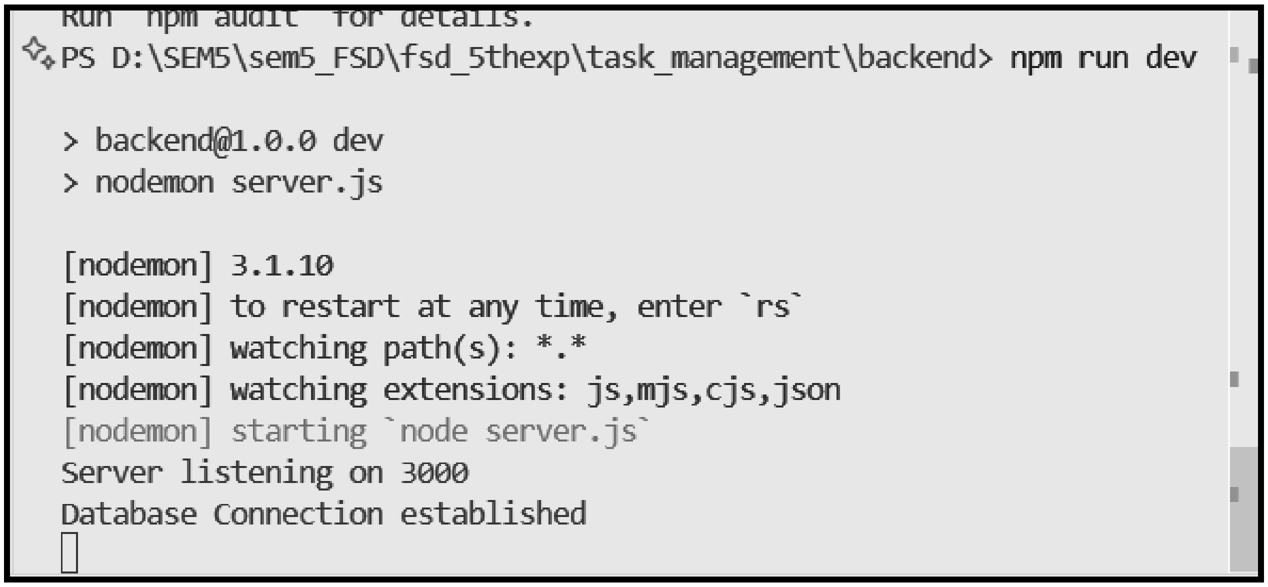


task_management > backend > models > task.js > taskSchema

```javascript
const mongoose = require("mongoose");

const taskSchema = new mongoose.Schema(
    {
        title: {
            type: String,
            required: true,
            unique: true,
            trim: true,
        },
        priority: {
            type: String,
            required: true,
        },
        category: {
            type: String,
            required: true,
            default: "to-do",
        },
        checklist: [
            {
                name: {
                    type: String,
                    trim: true,
                    required: true,
                },
                isDone: {
                    type: Boolean,
                    default: false,
                    required: true,
                },
            },
        ],
```

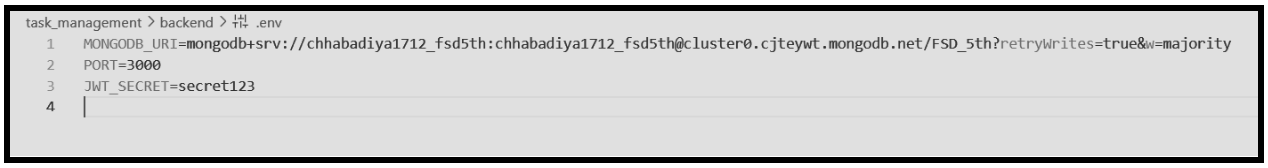


```javascript
        userName: {
            type: mongoose.Schema.ObjectId,
            ref: "User",
            required: true,
        },
        assign: String,

        dueDate: Date,
    },
    {
        timestamps: true,
    }
);

const Task = mongoose.model("Task", taskSchema);
module.exports = Task;
```



```
  Run `npm audit` for details.
PS D:\SEM5\sem5_FSD\fsd_5thexp\task_management\backend> npm run dev

> backend@1.0.0 dev
> nodemon server.js

[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
Server listening on 3000
Database Connection established
```

D] .env



task_management > backend > .env

```
MONGODB_URI=mongodb+srv://chhabadiya1712_fsd5th:chhabadiya1712_fsd5th@cluster0.cjteywt.mongodb.net/FSD_5th?retryWrites=true&w=majority
PORT=3000
JWT_SECRET=secret123

```

Postman API request + response snippet

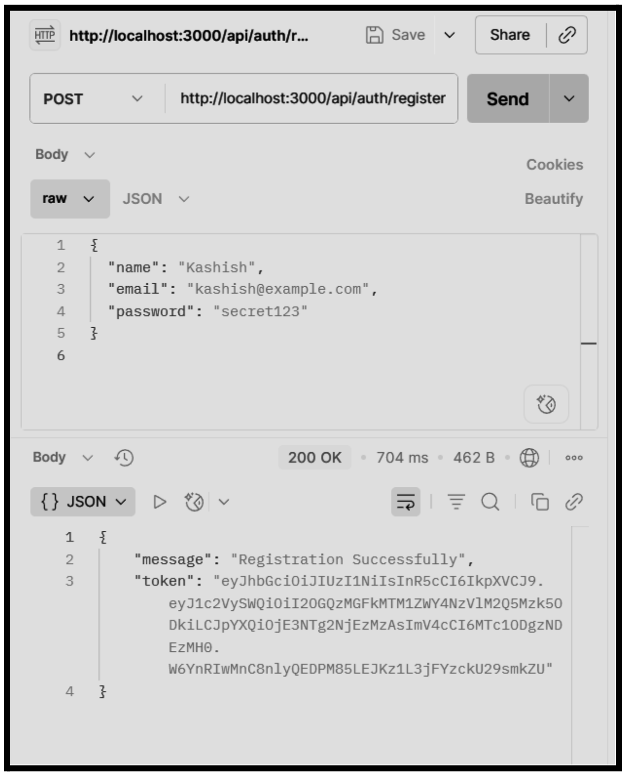
Why it's necessary

1. **Shows that your backend works:**
   A proof that your RESTful APIs are functional. Screenshots from Postman act as evidence.

2. **Captures API input/output:**
   You can show the **request payload** (JSON sent to API) and the **response** (JSON returned). This is critical for documentation.

3. **Validates security features:**
   Testing with JWT authentication shows that **only authorized users** can perform actions like creating or deleting tasks.

4. **Required for report completeness:**
   Most experiments require **code + output images**. Without testing APIs, your report will be incomplete.

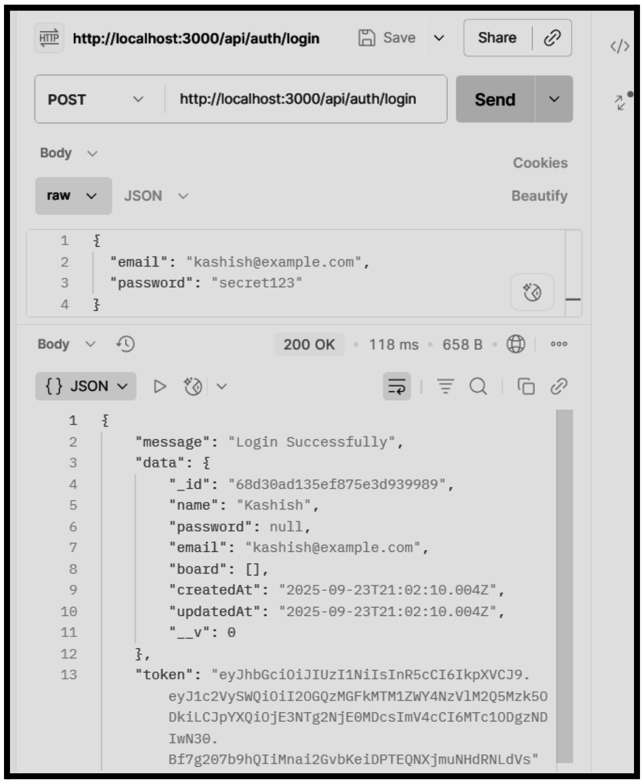


POST http://localhost:3000/api/auth/register

```
{
  "name": "Kashish",
  "email": "kashish@example.com",
  "password": "secret123"
}
```

200 OK • 704 ms • 462 B

```
{
    "message": "Registration Successfully",
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJ1c2VySWQiOiI2OGQzMGFkMTM1ZWY4NzVlM2Q5Mzk5O
        DkiLCJpYXQiOjE3NTg2NjEzMzAsImV4cCI6MTc1ODgzND
        EzMH0.
        W6YnRIwMnC8nlyQEDPM85LEJKz1L3jFYzckU29smkZU"
}
```

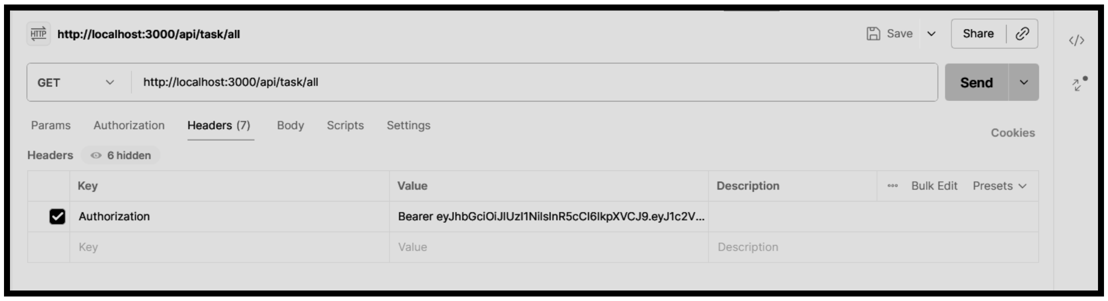


POST http://localhost:3000/api/auth/login

```
{
  "email": "kashish@example.com",
  "password": "secret123"
}
```

200 OK • 118 ms • 658 B

```
{
    "message": "Login Successfully",
    "data": {
        "_id": "68d30ad135ef875e3d939989",
        "name": "Kashish",
        "password": null,
        "email": "kashish@example.com",
        "board": [],
        "createdAt": "2025-09-23T21:02:10.004Z",
        "updatedAt": "2025-09-23T21:02:10.004Z",
        "__v": 0
    },
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJ1c2VySWQiOiI2OGQzMGFkMTM1ZWY4NzVlM2Q5Mzk5O
        DkiLCJpYXQiOjE3NTg2NjE0MDcsImV4cCI6MTc1ODgzND
        IwN30.
        Bf7g207b9hQIiMnai2GvbKeiDPTEQNXjmuNHdRNLdVs"
```

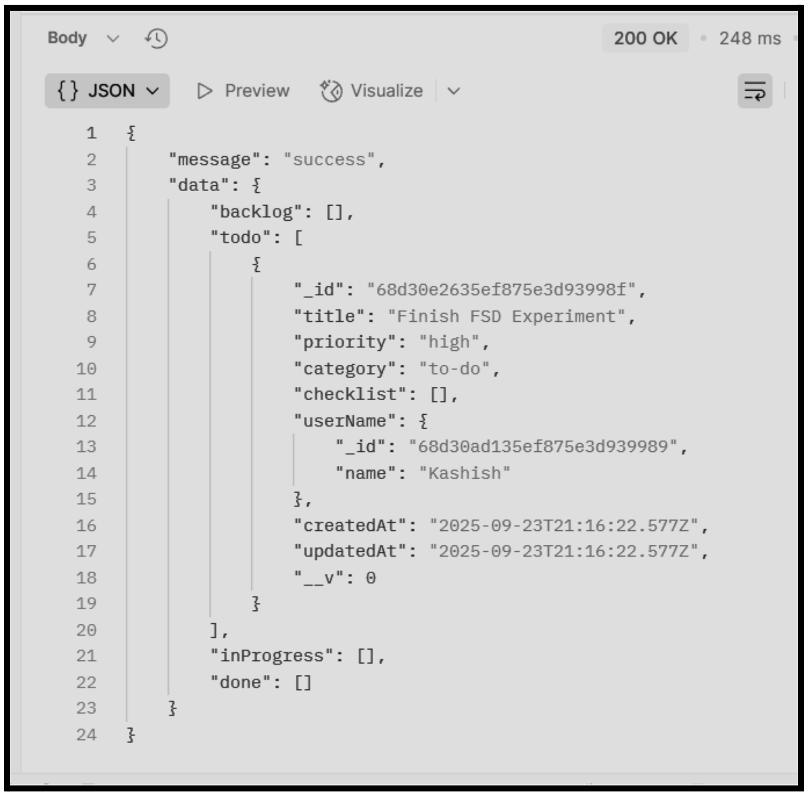


http://localhost:3000/api/task/all

GET http://localhost:3000/api/task/all

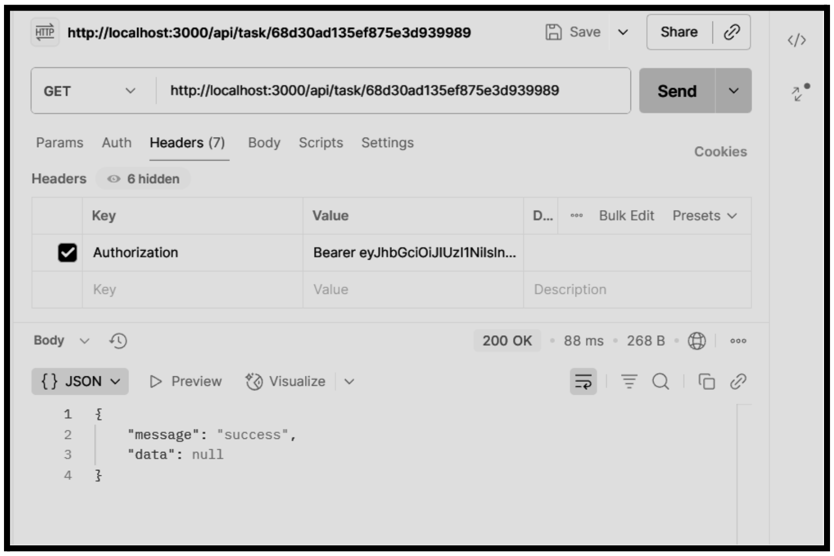
Headers 6 hidden

	Key	Value	Description
✓	Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2V...	
	Key	Value	Description



200 OK • 248 ms

```
{
    "message": "success",
    "data": {
        "backlog": [],
        "todo": [
            {
                "_id": "68d30e2635ef875e3d93998f",
                "title": "Finish FSD Experiment",
                "priority": "high",
                "category": "to-do",
                "checklist": [],
                "userName": {
                    "_id": "68d30ad135ef875e3d939989",
                    "name": "Kashish"
                },
                "createdAt": "2025-09-23T21:16:22.577Z",
                "updatedAt": "2025-09-23T21:16:22.577Z",
                "__v": 0
            }
        ],
        "inProgress": [],
        "done": []
    }
}
```



http://localhost:3000/api/task/68d30ad135ef875e3d939989

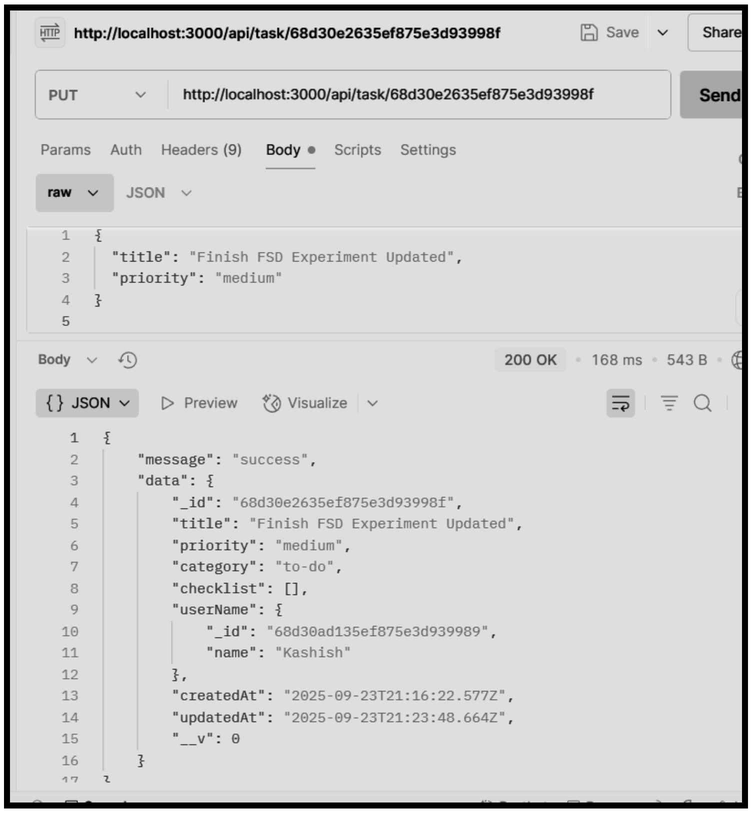
GET http://localhost:3000/api/task/68d30ad135ef875e3d939989

Headers 6 hidden

	Key	Value	D...
✓	Authorization	Bearer eyJhbGciOiJIUzI1NiIsIn...	
	Key	Value	Description

200 OK • 88 ms • 268 B

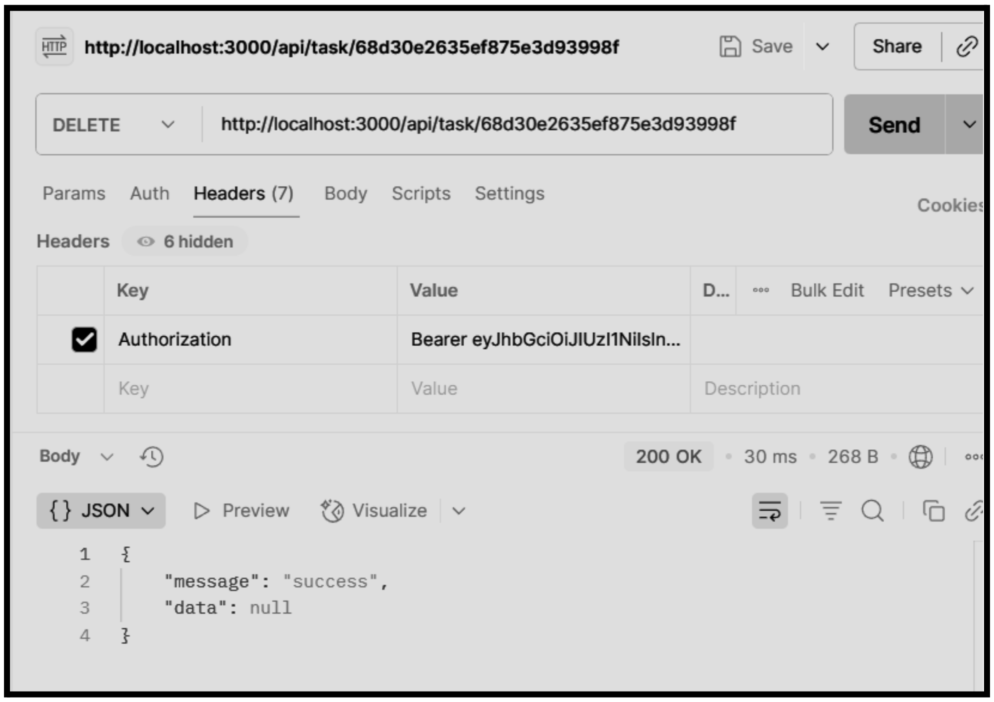
```
{
    "message": "success",
    "data": null
}
```



PUT http://localhost:3000/api/task/68d30e2635ef875e3d93998f

```
{
  "title": "Finish FSD Experiment Updated",
  "priority": "medium"
}

```

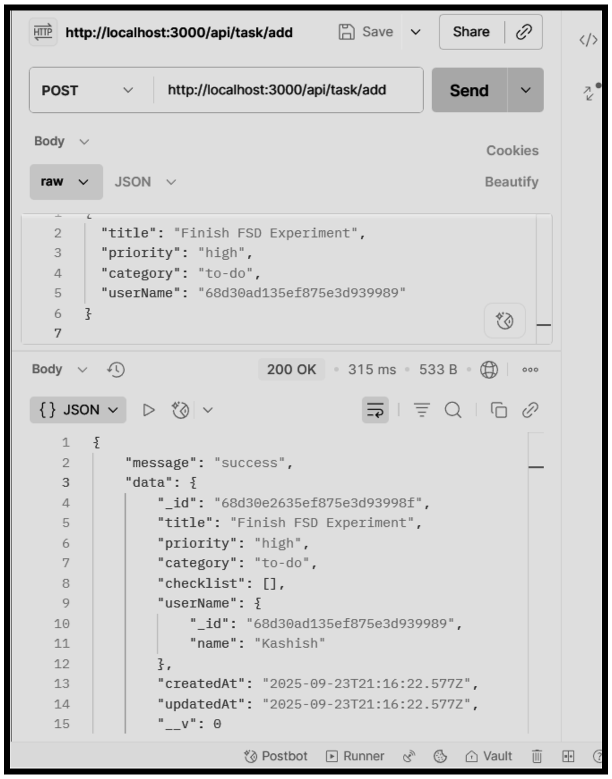
```
{
    "message": "success",
    "data": {
        "_id": "68d30e2635ef875e3d93998f",
        "title": "Finish FSD Experiment Updated",
        "priority": "medium",
        "category": "to-do",
        "checklist": [],
        "userName": {
            "_id": "68d30ad135ef875e3d939989",
            "name": "Kashish"
        },
        "createdAt": "2025-09-23T21:16:22.577Z",
        "updatedAt": "2025-09-23T21:23:48.664Z",
        "__v": 0
    }
}
```



DELETE http://localhost:3000/api/task/68d30e2635ef875e3d93998f

	Key	Value	D...
✓	Authorization	Bearer eyJhbGciOiJIUzI1NiIsIn...	
	Key	Value	Description

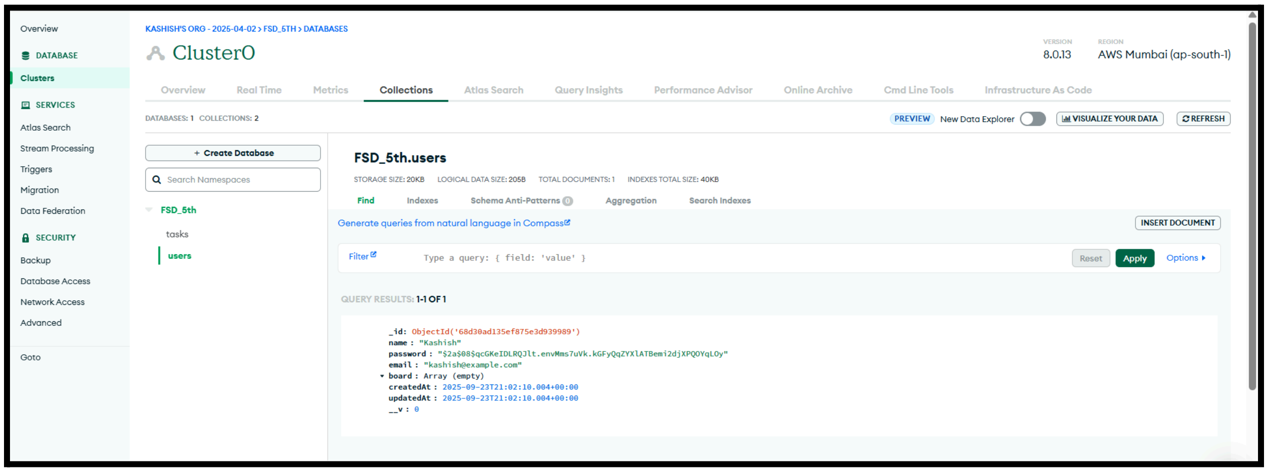
```
{
    "message": "success",
    "data": null
}
```



POST http://localhost:3000/api/task/add

```
{
    "title": "Finish FSD Experiment",
    "priority": "high",
    "category": "to-do",
    "userName": "68d30ad135ef875e3d939989"
}
```

```
{
    "message": "success",
    "data": {
        "_id": "68d30e2635ef875e3d93998f",
        "title": "Finish FSD Experiment",
        "priority": "high",
        "category": "to-do",
        "checklist": [],
        "userName": {
            "_id": "68d30ad135ef875e3d939989",
            "name": "Kashish"
        },
        "createdAt": "2025-09-23T21:16:22.577Z",
        "updatedAt": "2025-09-23T21:16:22.577Z",
        "__v": 0
```



```
_id: ObjectId('68d30ad135ef875e3d939989')
name : "Kashish"
password : "$2a$08$qcGKeIDLRQJlt.envMms7uVk.kGFyQqZYXlATBemi2djXPQOYqLOy"
email : "kashish@example.com"
▾ board : Array (empty)
createdAt : 2025-09-23T21:02:10.004+00:00
updatedAt : 2025-09-23T21:02:10.004+00:00
__v : 0
```

<u>30% extra :</u>

Task enhancements: categories, priority levels, checklists, and assignments

Analytics endpoints for task statistics

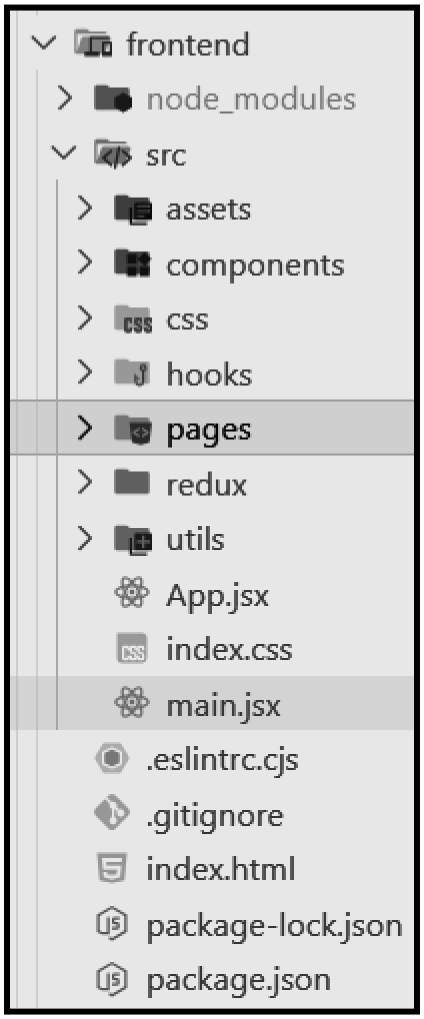
Frontend integration: React + Vite with JWT token handling

Real-time updates reflected on the UI

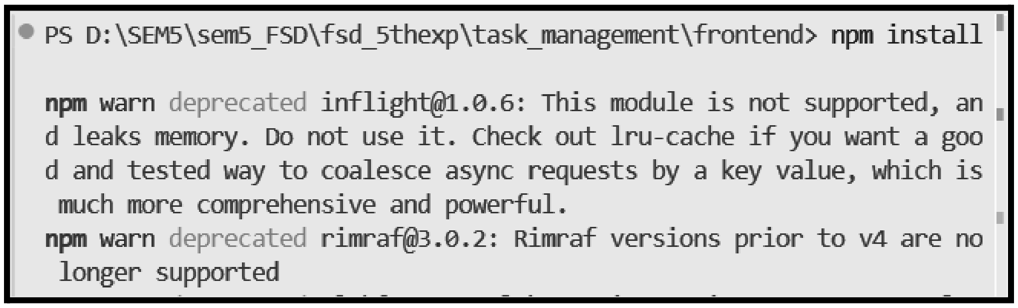
React-Redux for state management

Environment-based config using .env

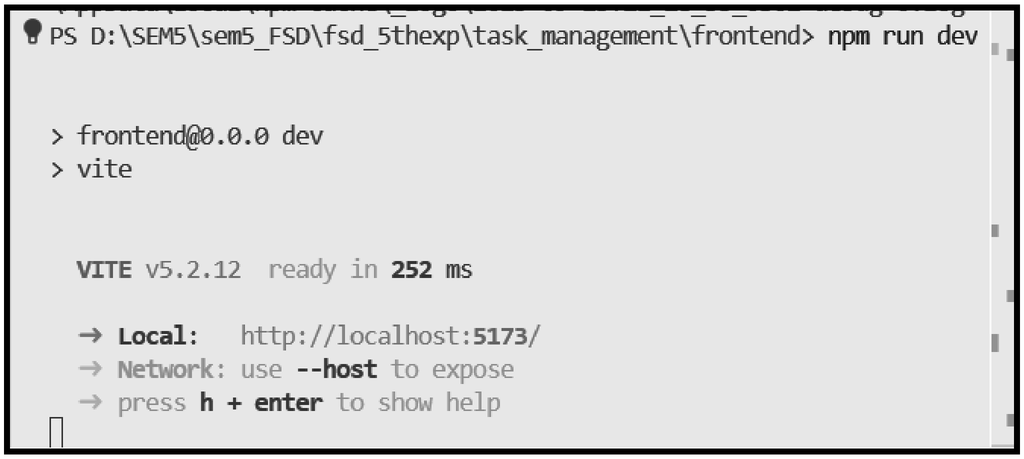
<u>Frontend</u>



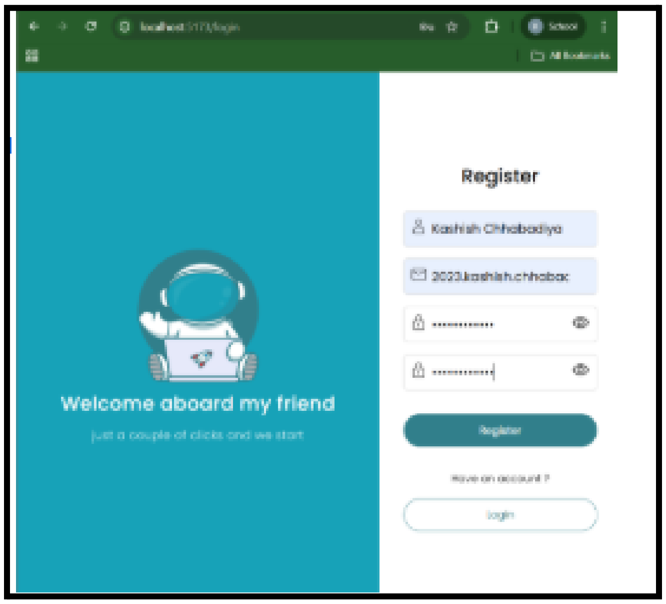
```
frontend
  node_modules
  src
    assets
    components
    css
    hooks
    pages
    redux
    utils
    App.jsx
    index.css
    main.jsx
  .eslintrc.cjs
  .gitignore
  index.html
  package-lock.json
  package.json
```



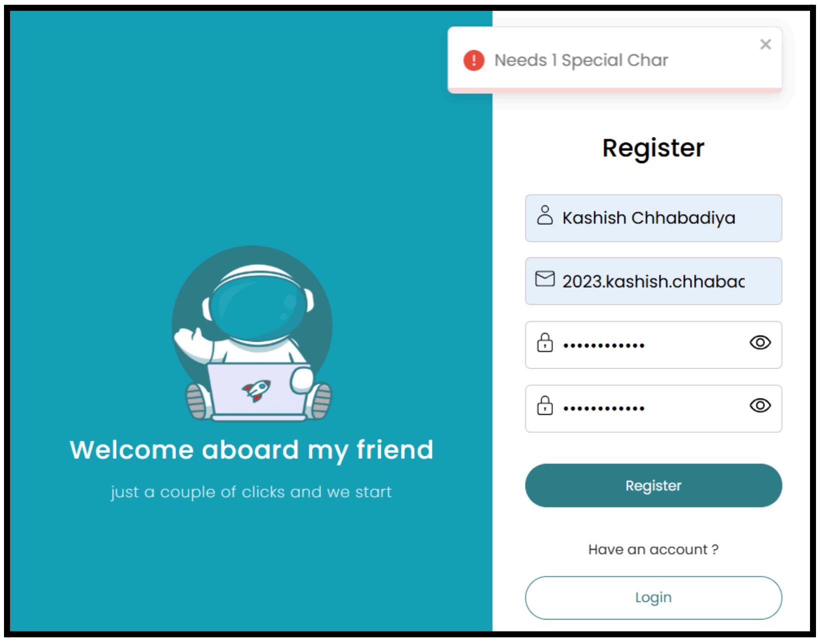
```
PS D:\SEM5\sem5_FSD\fsd_5thexp\task_management\frontend> npm install

npm warn deprecated inflight@1.0.6: This module is not supported, an
d leaks memory. Do not use it. Check out lru-cache if you want a goo
d and tested way to coalesce async requests by a key value, which is
 much more comprehensive and powerful.
npm warn deprecated rimraf@3.0.2: Rimraf versions prior to v4 are no
 longer supported
```



```
PS D:\SEM5\sem5_FSD\fsd_5thexp\task_management\frontend> npm run dev

> frontend@0.0.0 dev
> vite


  VITE v5.2.12  ready in 252 ms

  ➜  Local:   http://localhost:5173/
  ➜  Network: use --host to expose
  ➜  press h + enter to show help
```

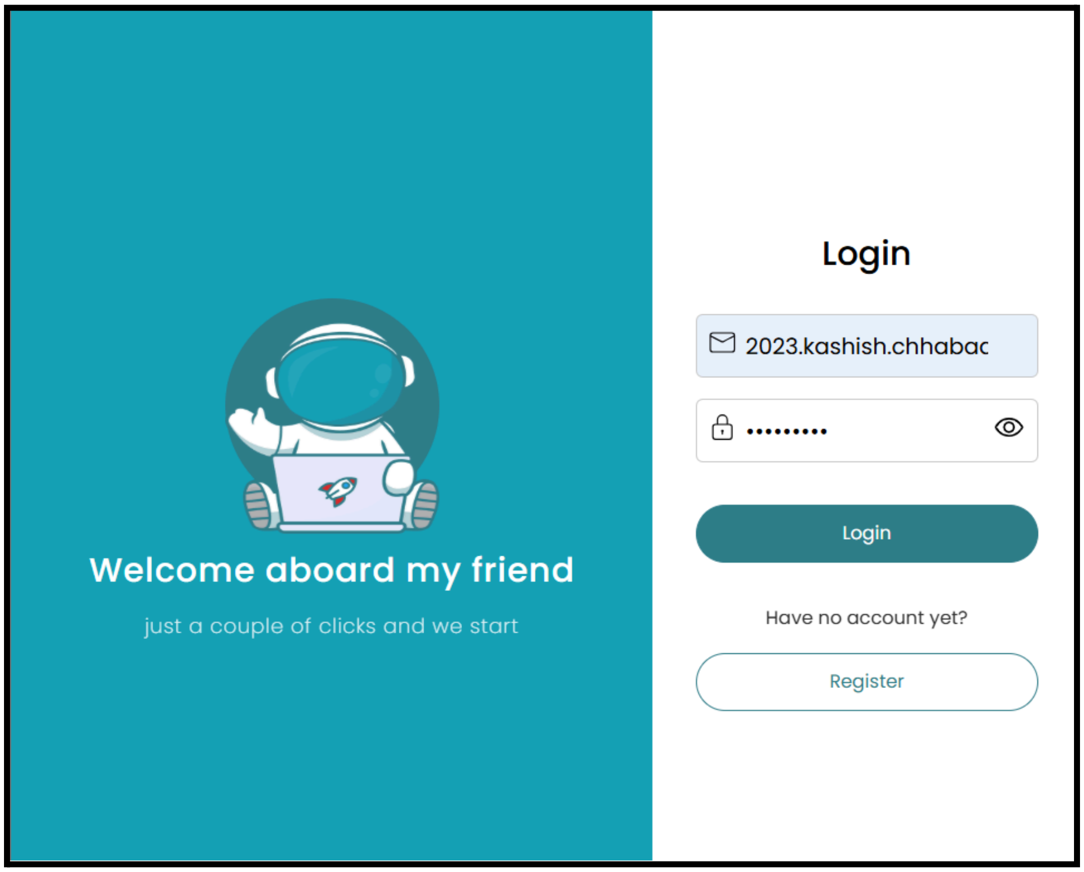


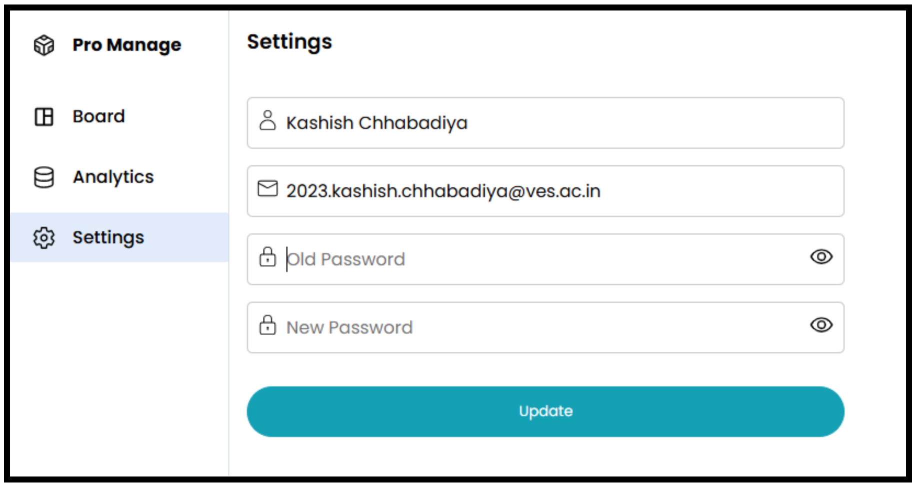
Kashish Chhabadiya
2023.kashish.chhabac
••••••••••••
••••••••••••



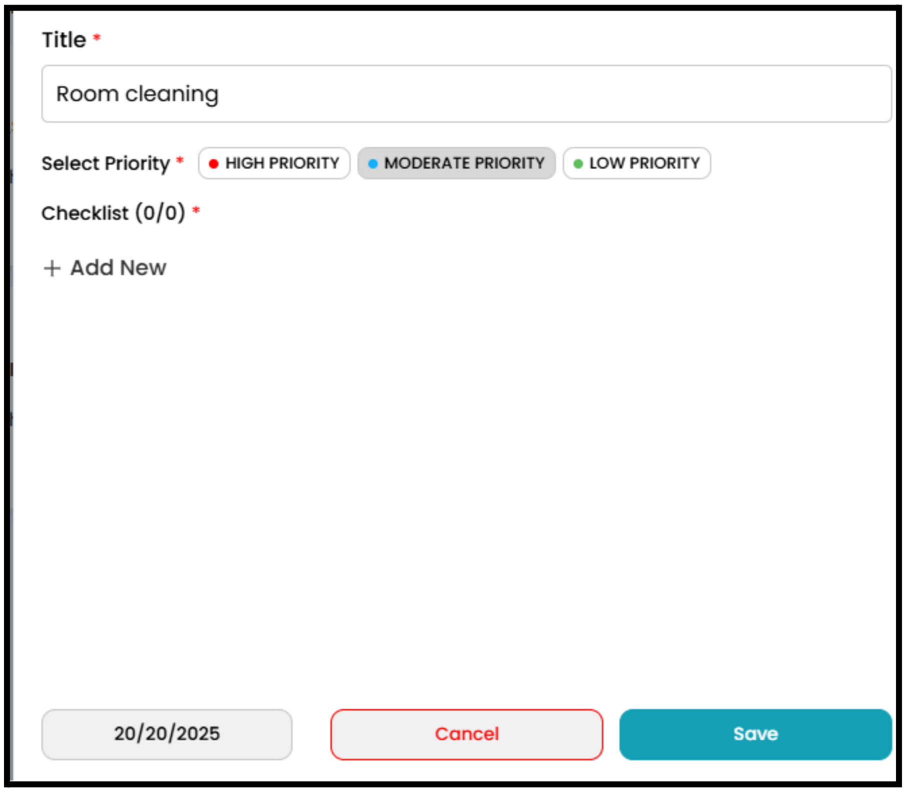
Needs 1 Special Char

Kashish Chhabadiya
2023.kashish.chhabac
••••••••••••
••••••••••••


Welcome aboard my friend
just a couple of clicks and we start
Login
2023.kashish.chhabac
•••••••••
Login
Have no account yet?
Register


Pro Manage
Board
Analytics
Settings
Settings
Kashish Chhabadiya
2023.kashish.chhabadiya@ves.ac.in
Old Password
New Password
Update

Creation of tasks :

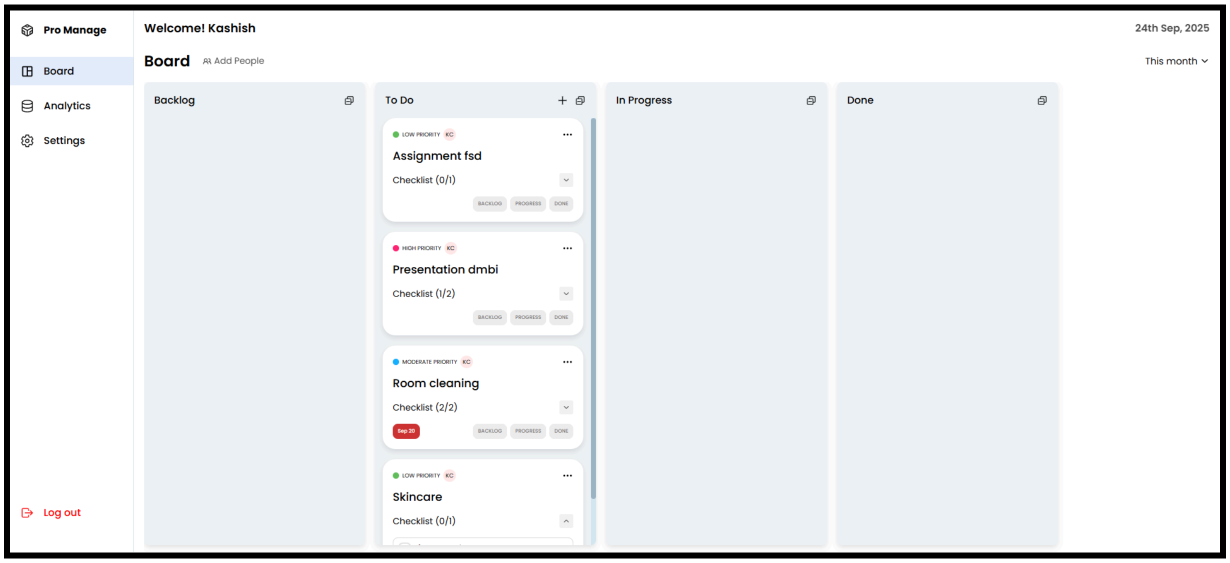


Title *

Room cleaning

Select Priority * ● HIGH PRIORITY [✓ ● MODERATE PRIORITY] ● LOW PRIORITY

Checklist (0/0) *



Welcome! Kashish

24th Sep, 2025

Board

Backlog

To Do

● LOW PRIORITY KC
Assignment fsd
Checklist (0/1)

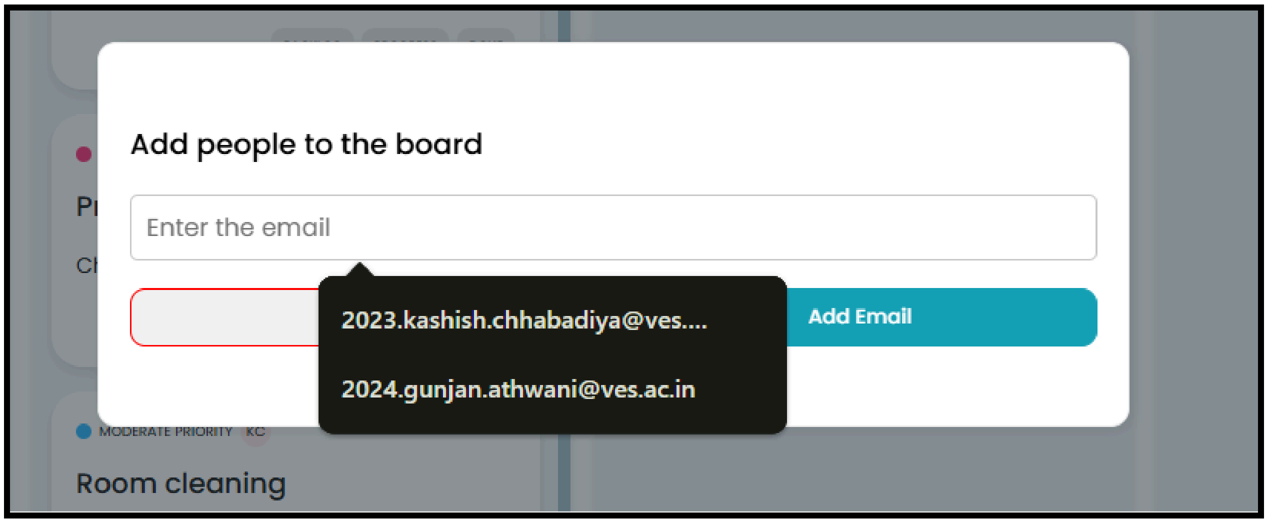
● HIGH PRIORITY KC
Presentation dmbi
Checklist (1/2)

● MODERATE PRIORITY KC
Room cleaning
Checklist (2/2)
Sep 20

● LOW PRIORITY KC
Skincare
Checklist (0/1)

In Progress

Done

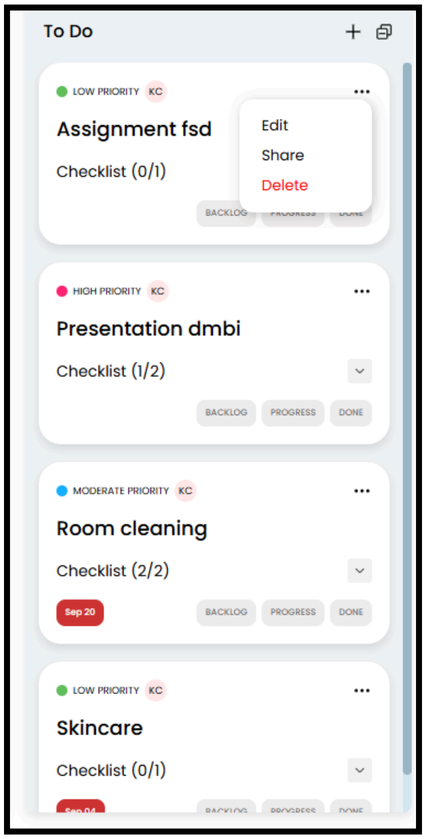

Add people to the board

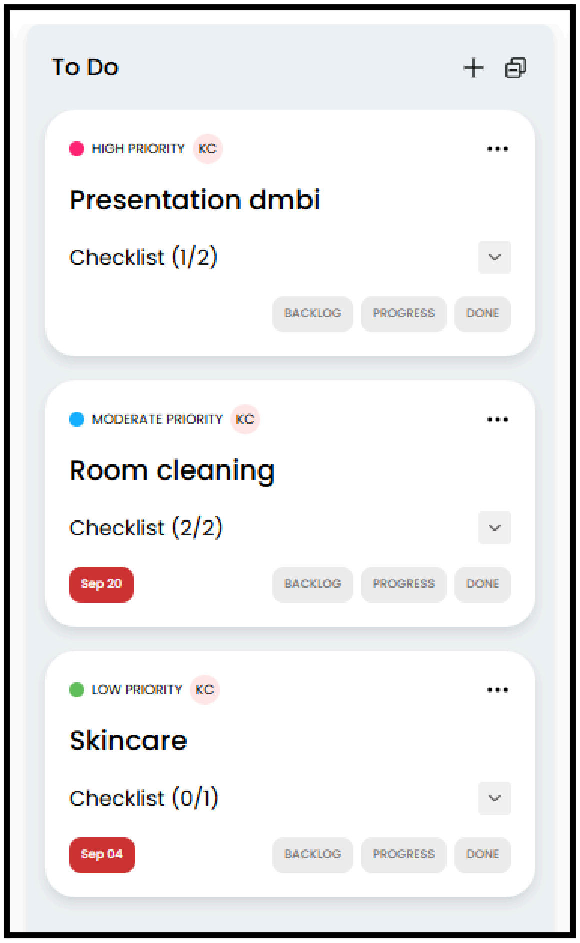
Enter the email

2023.kashish.chhabadiya@ves....

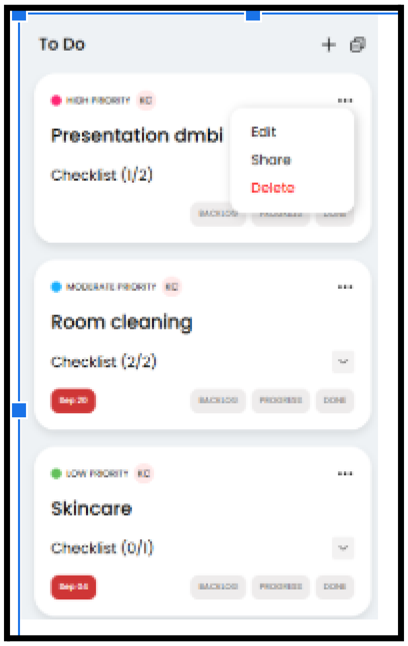
2024.gunjan.athwani@ves.ac.in

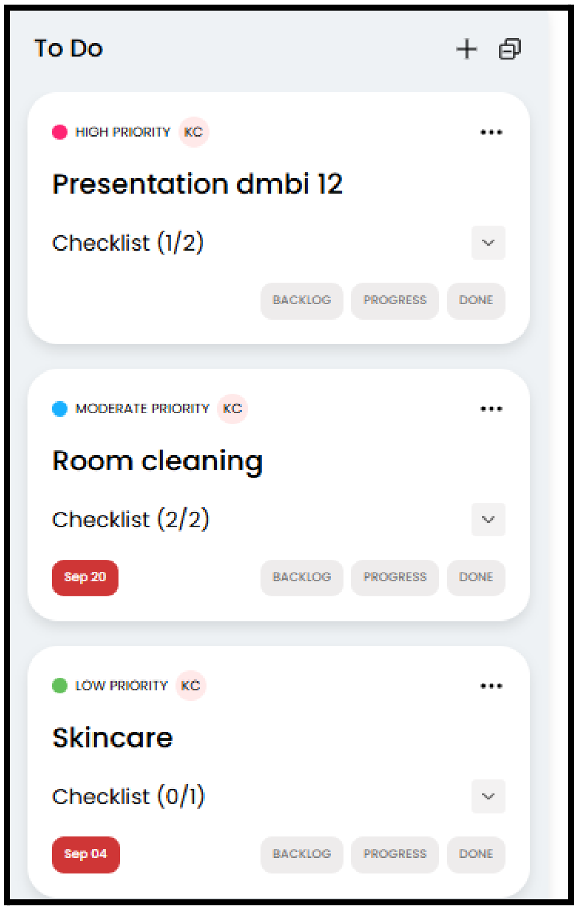
Delete :



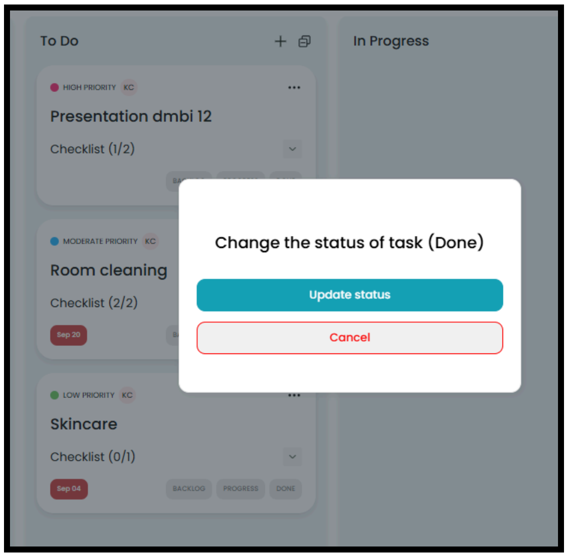


Editing tasks :





Update task status :



To Do

HIGH PRIORITY KC

Presentation dmbi 12

Checklist (1/2)

MODERATE PRIORITY KC

Room cleaning

Checklist (2/2)

Sep 20

LOW PRIORITY KC

Skincare

Checklist (0/1)

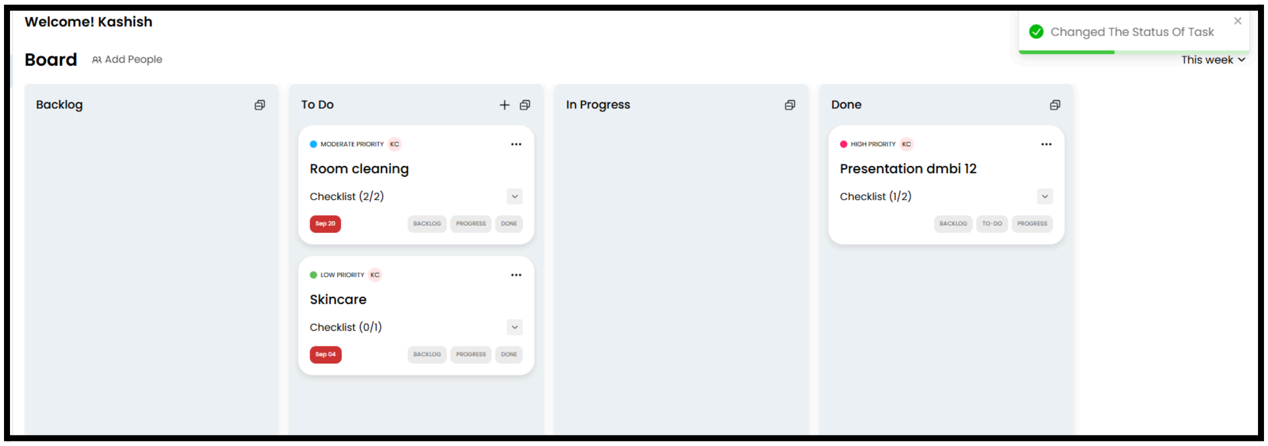
Sep 04 BACKLOG PROGRESS DONE

In Progress

Change the status of task (Done)

Update status

Cancel



Welcome! Kashish

Changed The Status Of Task

Board Add People

This week

Backlog

To Do

MODERATE PRIORITY KC

Room cleaning

Checklist (2/2)

Sep 20 BACKLOG PROGRESS DONE

LOW PRIORITY KC

Skincare

Checklist (0/1)

Sep 04 BACKLOG PROGRESS DONE

In Progress

Done

HIGH PRIORITY KC

Presentation dmbi 12

Checklist (1/2)

BACKLOG TO-DO PROGRESS