Experiment - 5

Aim

Create secure, production-ready **RESTful APIs**

What We Have Done in This Project

1. **Backend Setup**
    - Node.js with **Express** as the backend framework.
    - **MongoDB (Atlas)** used as the database, connected securely via a .env file.
    - **Mongoose** ODM for database modeling.
2. **Security Features**
    - **JWT Authentication**: Every request is verified using JSON Web Tokens.
    - **Role-Based Authorization**: Certain routes (e.g., task creation, update, deletion) are accessible only to authorized users.
    - **Environment Variables**: Sensitive information like MONGO_URI and JWT_SECRET are stored in .env.
3. **RESTful API Design**
    - Standard CRUD routes implemented for tasks:
        - **GET** /api/task/all – Fetch all tasks (protected route)
        - **GET** /api/task/:id – Fetch a single task
        - **POST** /api/task/add – Create a new task (protected route)
        - **PUT** /api/task/:id – Update task details (protected route)
        - **DELETE** /api/task/:id – Delete a task (protected route)

        - **PUT** /api/task/category/:id – Update task category (protected route)
    - **Controllers** handle the logic while **routes** define the endpoints.
    - **Middlewares**:
        - authorization.js – Checks user role and permission
        - wrapAsync.js – Handles async errors globally
4. **Validation & Data Integrity**
    - Task schema (models/Task.js) enforces required fields: title, priority, category, userName.
    - Ensures correct data types and unique constraints.
5. **Frontend Integration**
    - React frontend consumes these RESTful APIs.
    - Tasks can be added, updated, deleted, or categorized.
    - JWT token is stored in frontend to authenticate API requests.
    - Real-time updates and notifications via React components.
6. **Production-Ready Features**
    - Server ready to run on **PORT 3000** or any environment variable.
    - Environment-based configuration using .env.
    - Secure database connection to MongoDB Atlas.
    - Proper error handling for invalid routes and server errors.

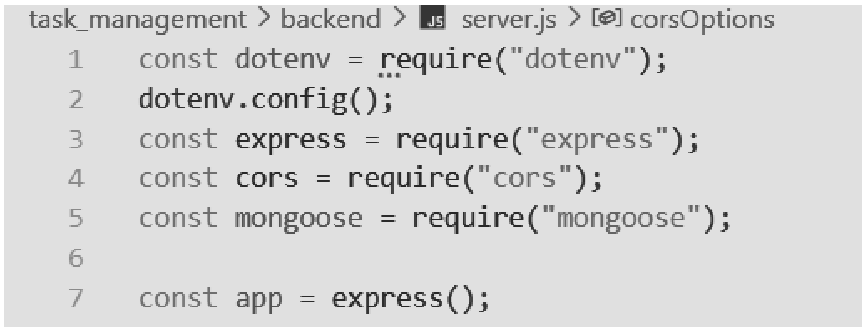
INPUT AND OUTPUT :

Backend :

A] Code :

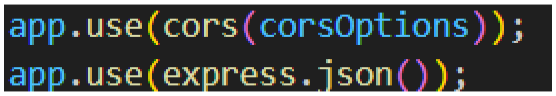
a) Server Setup (server.js)

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

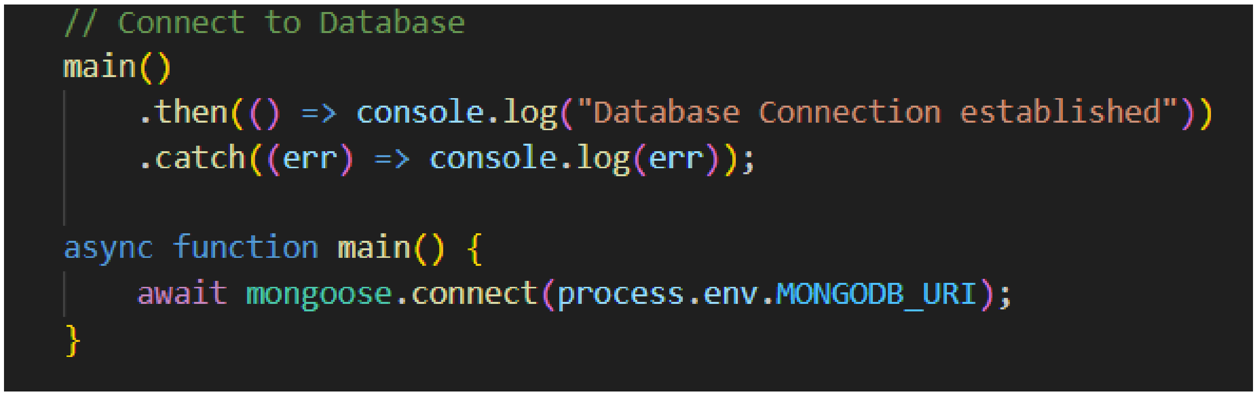


task_management > backend > server.js > corsOptions

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

const app = express();
```



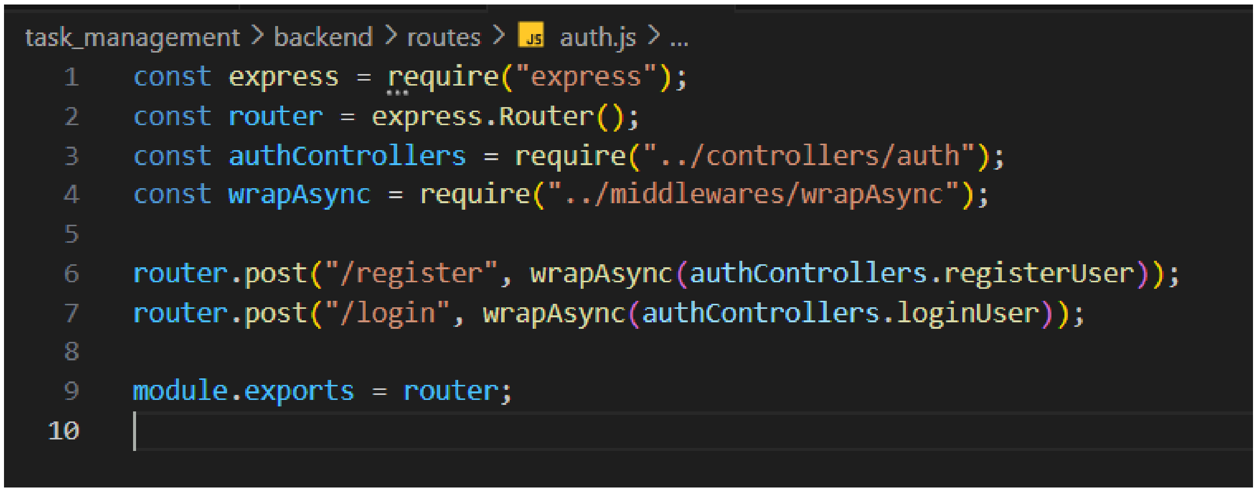
```javascript
app.use(cors(corsOptions));
app.use(express.json());
```



```javascript
// 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)

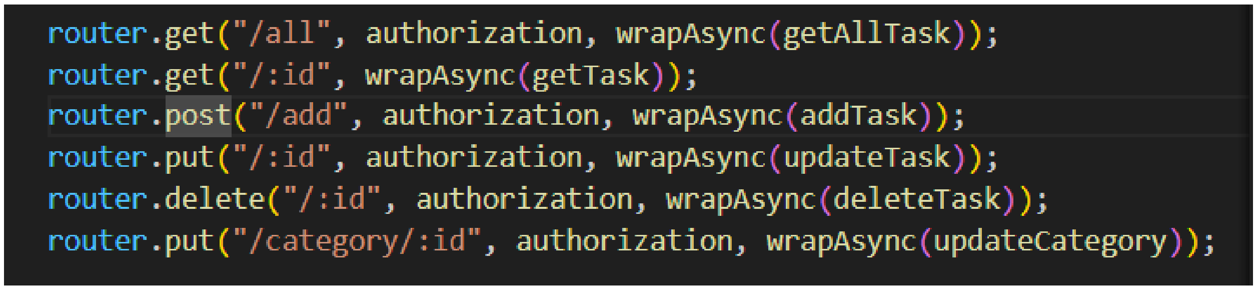
- Include **login and registration endpoints**, showing JWT issuance.


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)

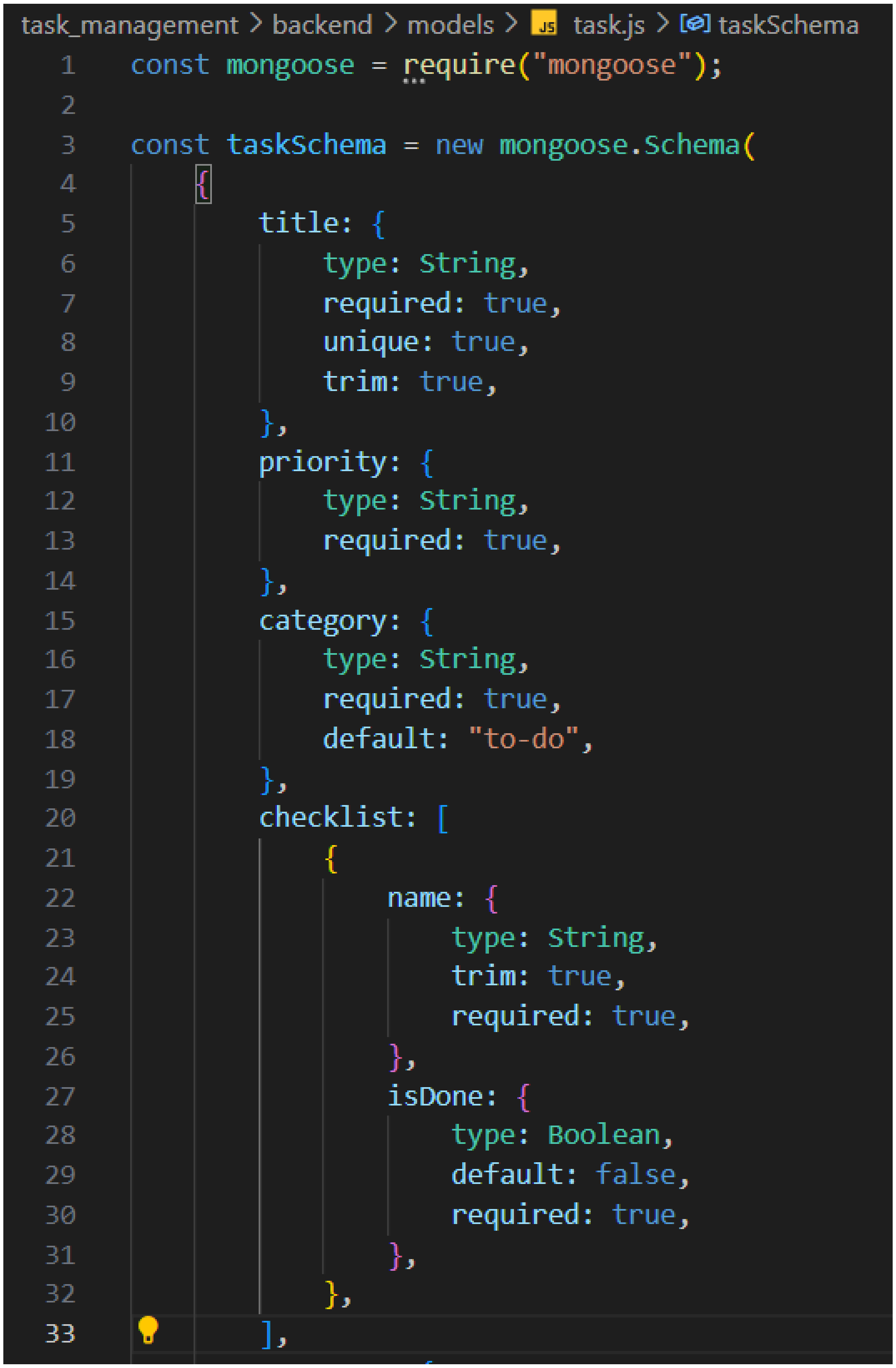
- Show **how authorization middleware is used** for secure endpoints.



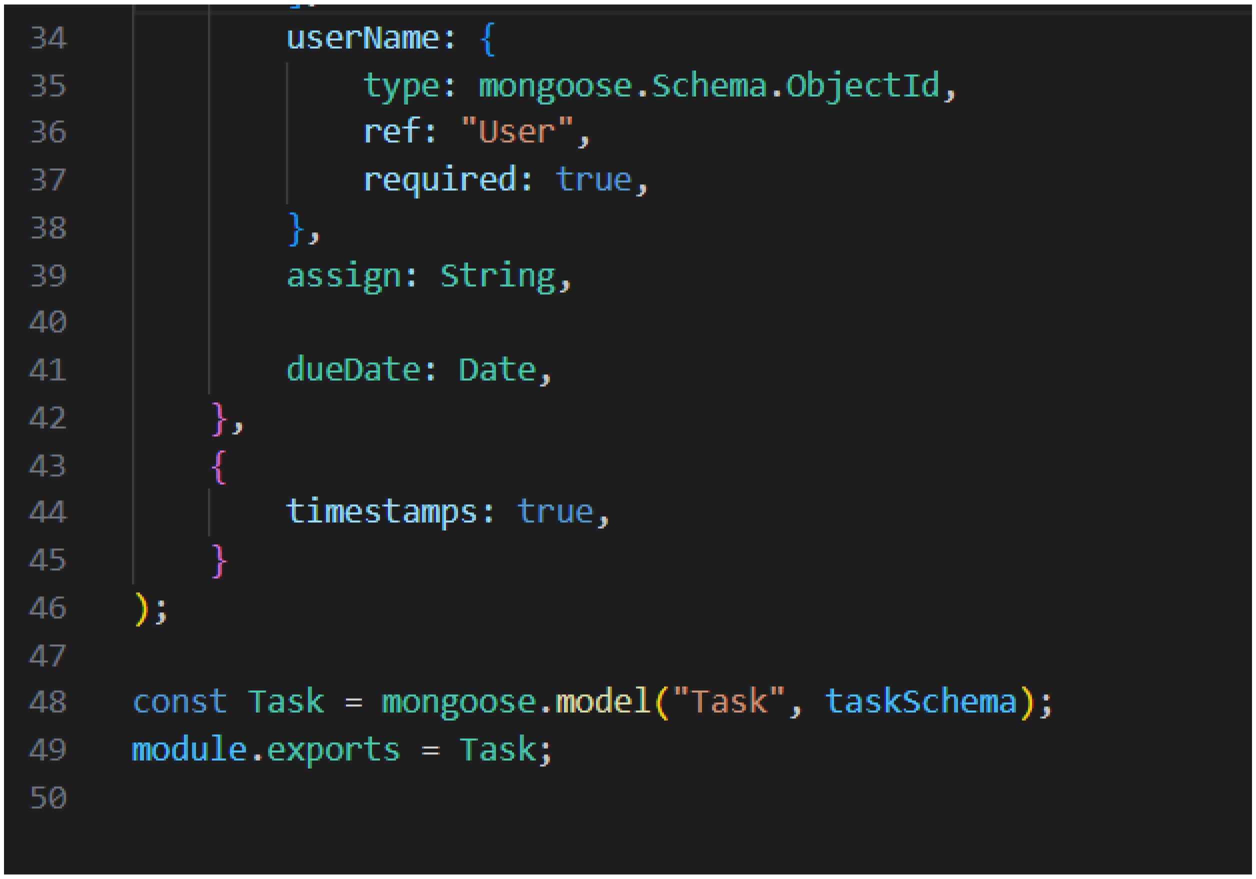
```
router.get("/all", authorization, wrapAsync(getAllTask));
router.get("/:id", wrapAsync(getTask));
router.post("/add", authorization, wrapAsync(addTask));
router.put("/:id", authorization, wrapAsync(updateTask));
router.delete("/:id", authorization, wrapAsync(deleteTask));
router.put("/category/:id",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

d) Task Model (models/task.js)

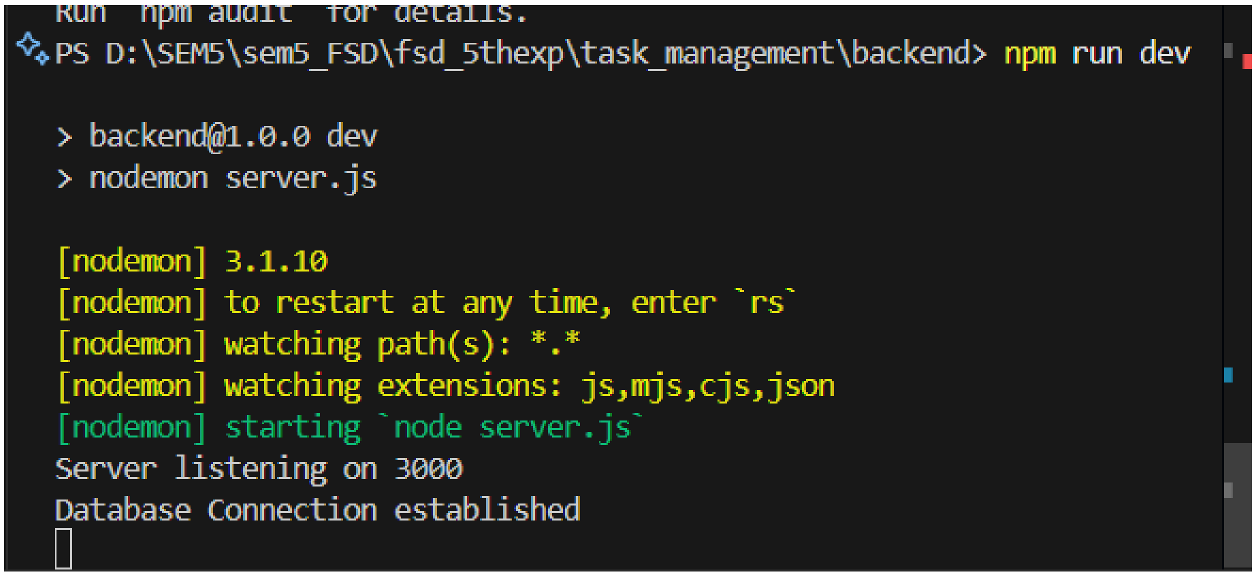
- Show **Mongoose schema**, especially required fields like title, priority, category, userName.



```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
```

<u>Postman API request + response snippet</u>

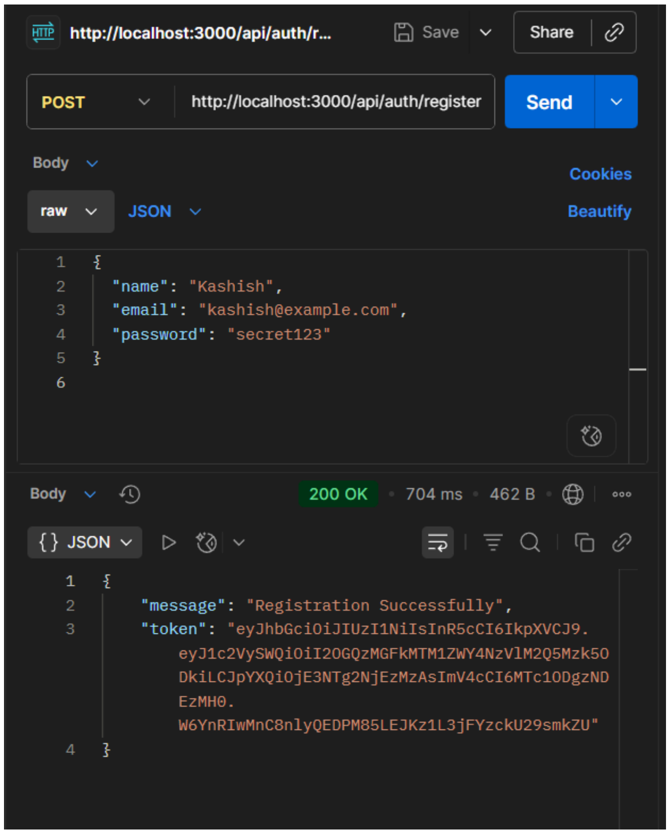
Why it's necessary

1. **Shows that your backend works:**
   Your teacher will want 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.

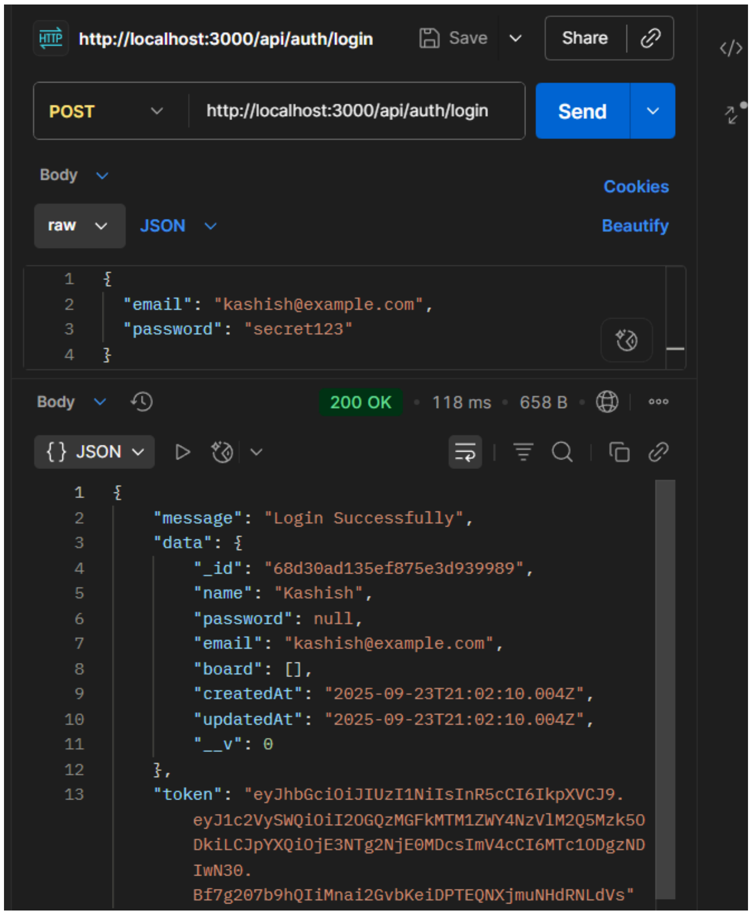


POST http://localhost:3000/api/auth/register

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

200 OK • 704 ms • 462 B

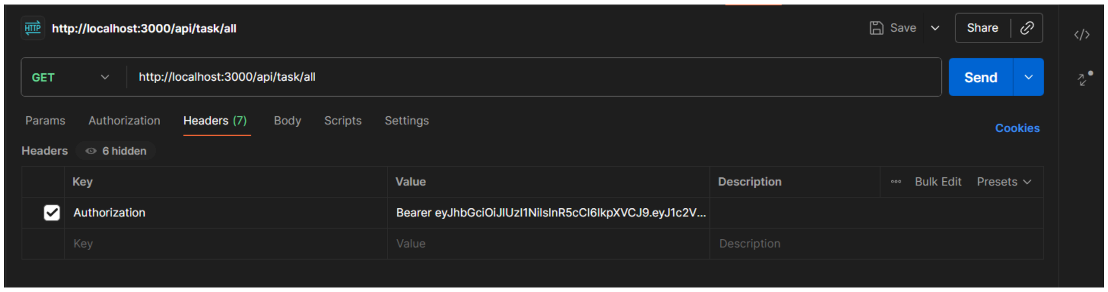
```
{
    "message": "Registration Successfully",
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJ1c2VySWQiOiI2OGQzMGFkMTM1ZWY4NzVlM2Q5Mzk5O
        DkiLCJpYXQiOjE3NTg2NjEzMzAsImV4cCI6MTc1ODgzND
        EzMH0.
        W6YnRIwMnC8nlyQEDPM85LEJKz1L3jFYzckU29smkZU"
}
```


http://localhost:3000/api/auth/login
Save
Share
POST
http://localhost:3000/api/auth/login
Send
Body
Cookies
raw
JSON
Beautify

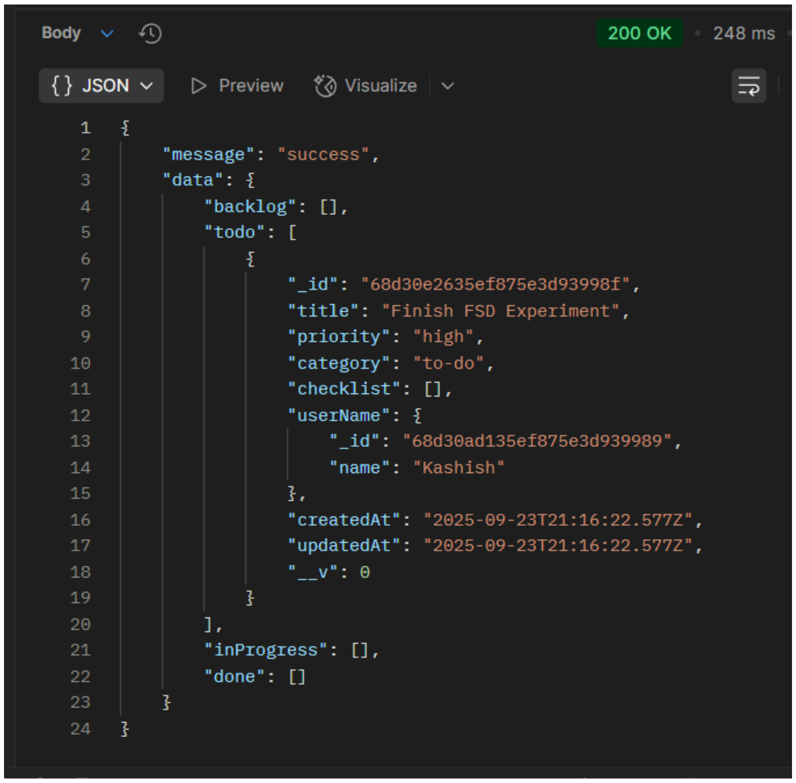
```
{
  "email": "kashish@example.com",
  "password": "secret123"
}
```

Body
200 OK
118 ms
658 B
{ } JSON

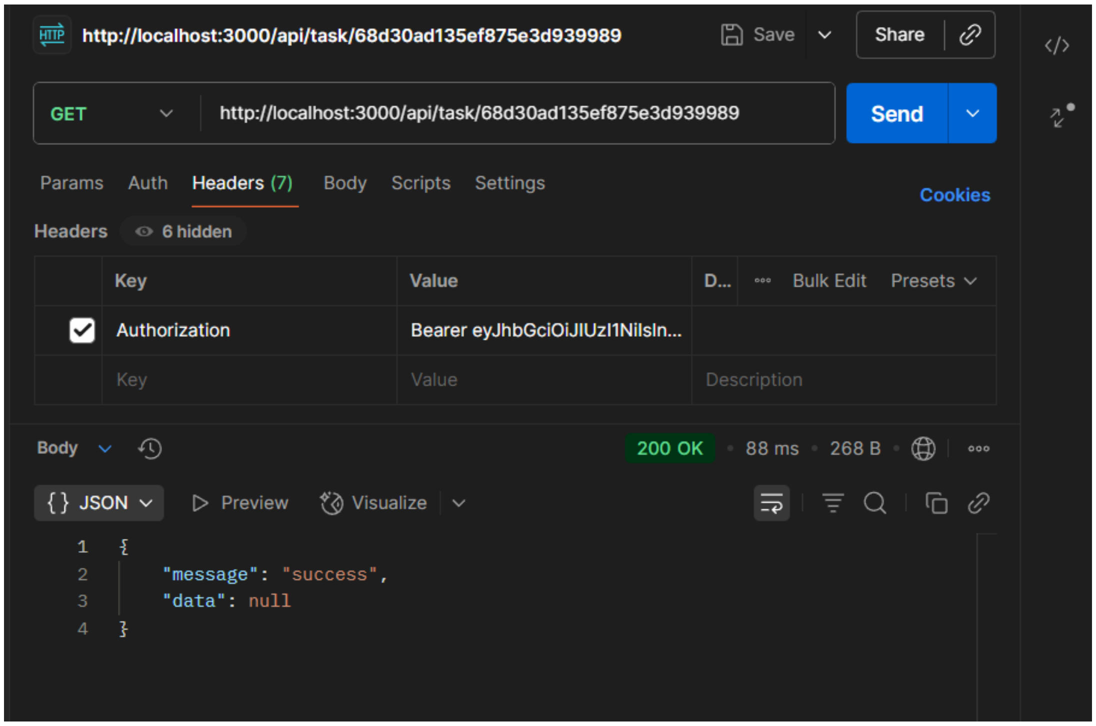
```
{
    "message": "Login Successfully",
    "data": {
        "_id": "68d30ad135ef875e3d939989",
        "name": "Kashish",
        "password": null,
        "email": "kashish@example.com",
        "board": [],
        "createdAt": "2025-09-23T21:02:10.004Z",
        "updatedAt": "2025-09-23T21:02:10.004Z",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



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



```
{
    "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": []
    }
}
```

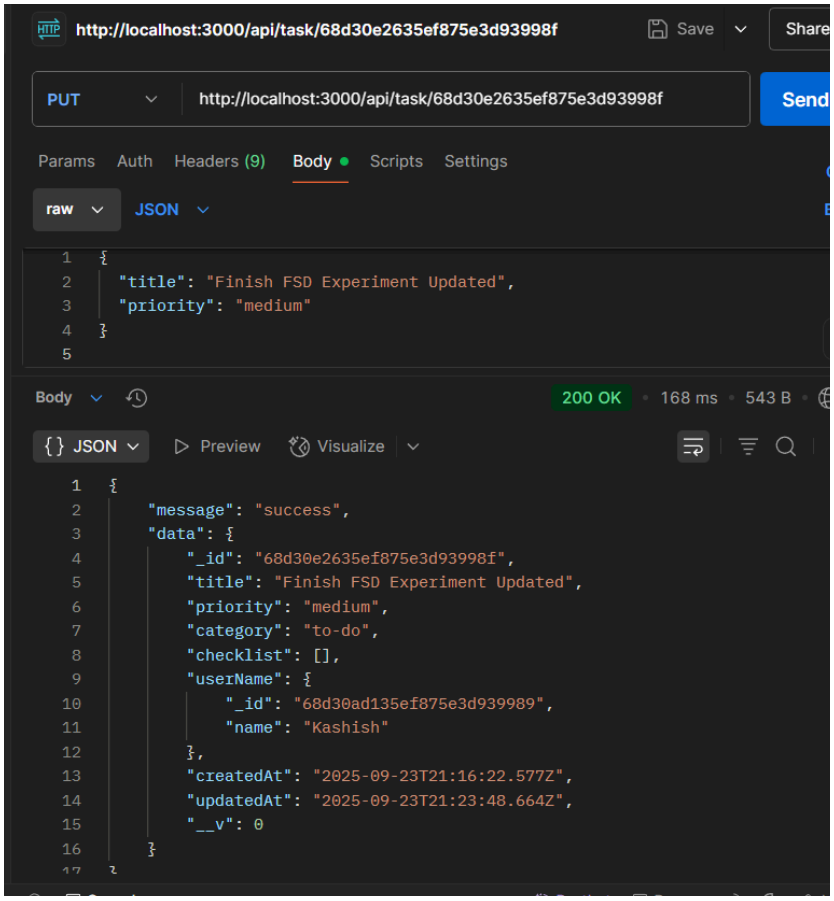


http://localhost:3000/api/task/68d30ad135ef875e3d939989

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

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

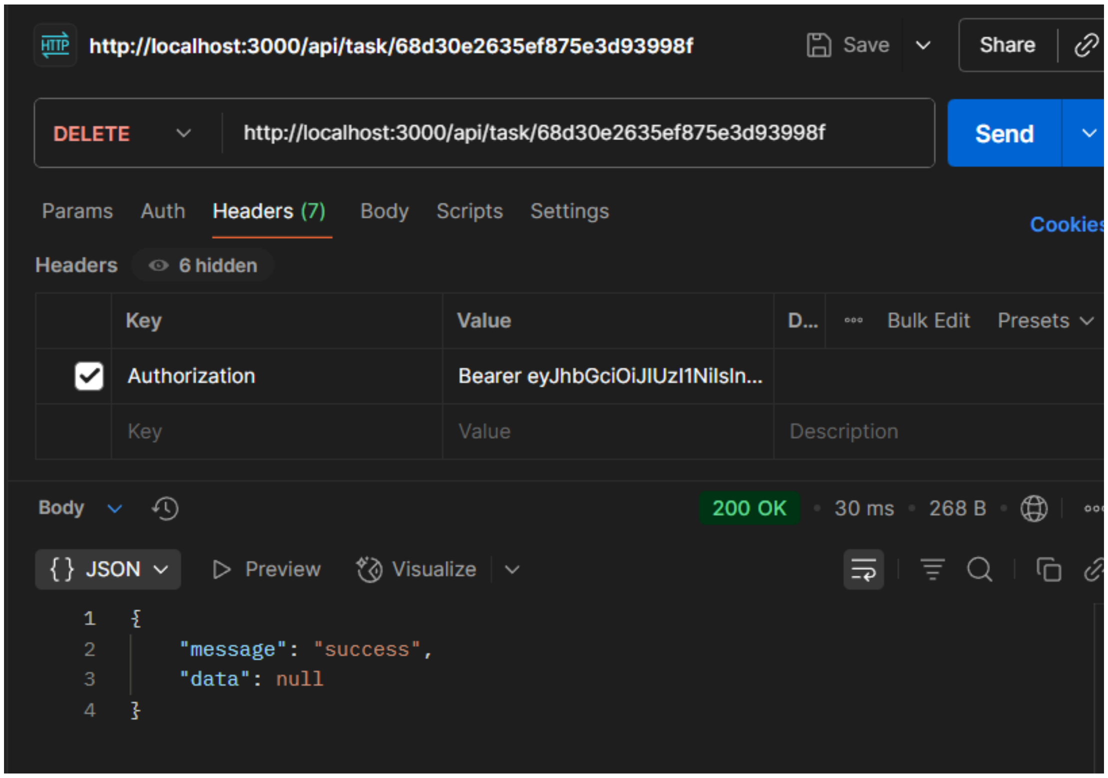
```
{
    "message": "success",
    "data": null
}
```


http://localhost:3000/api/task/68d30e2635ef875e3d93998f
Save
Share
PUT
Send
Params Auth Headers (9) Body Scripts Settings
raw JSON
Body
200 OK · 168 ms · 543 B
{} JSON Preview Visualize

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

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

```

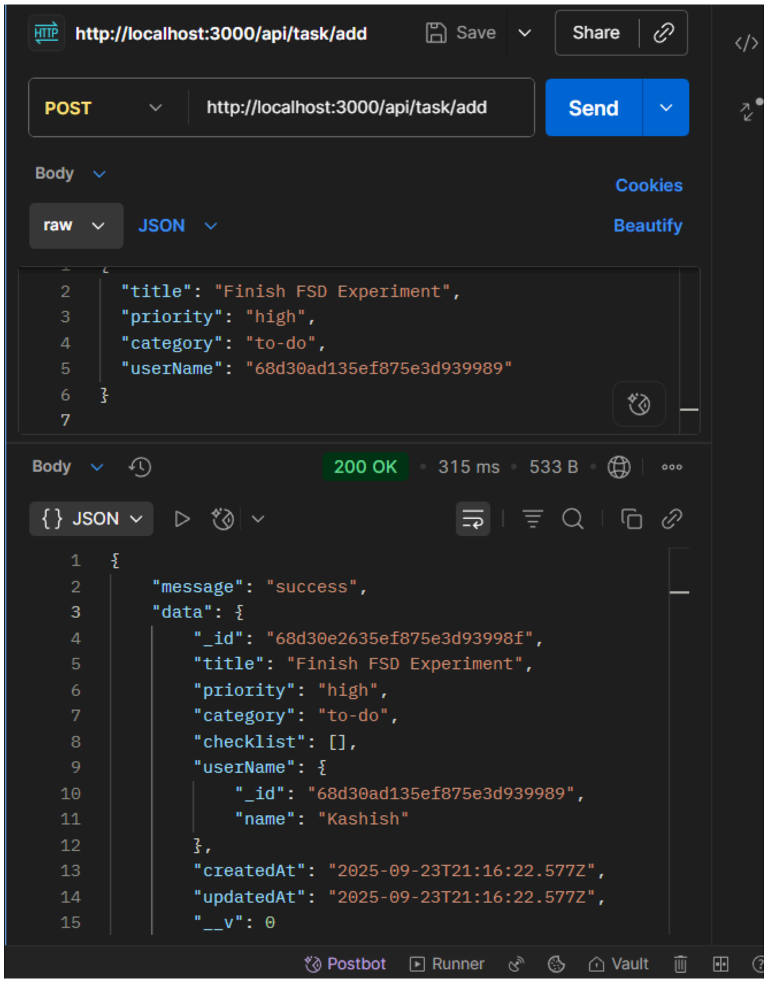
```
{
    "message": "success",
    "data": {
        "_id": "68d30e2635ef875e3d93998f",
        "title": "Finish FSD Experiment Updated",
        "priority": "medium",
        "category": "to-do",
        "checklist": [],
        "userName": {
            "_id": "68d30ad135ef875e3d939989",
        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


http://localhost:3000/api/task/68d30e2635ef875e3d93998f
Save
Share
DELETE
Send
Params Auth Headers (7) Body Scripts Settings
Cookies
Headers 6 hidden
Bulk Edit
Presets
Body
200 OK · 30 ms · 268 B
{} JSON Preview Visualize

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

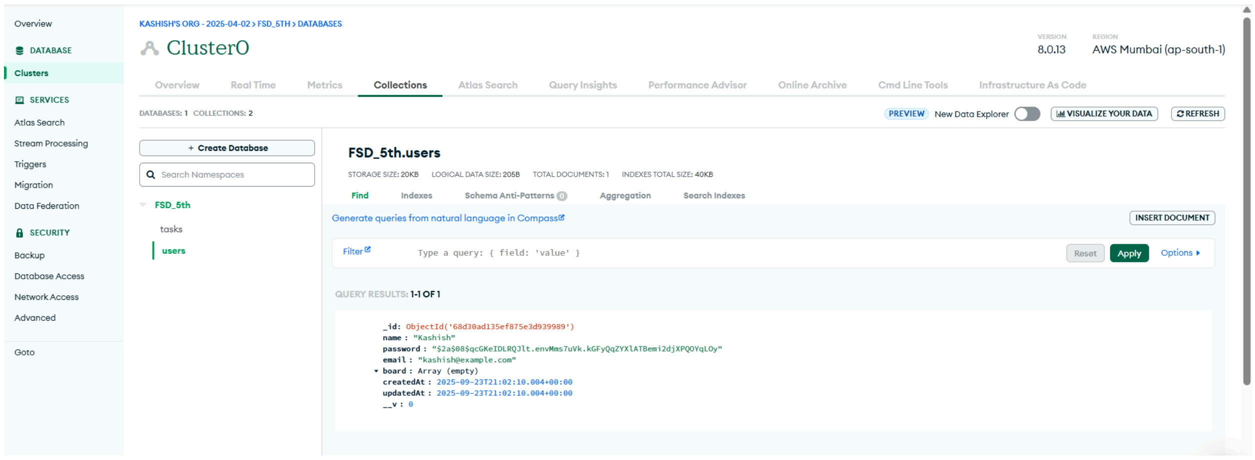
	Key	Value	D...
✓	Authorization	Bearer eyJhbGciOiJIUzI1NiIsIn...	
	Key	Value	Description

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



```
{
  "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
```

30% extra :

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

Analytics endpoints for task statistics

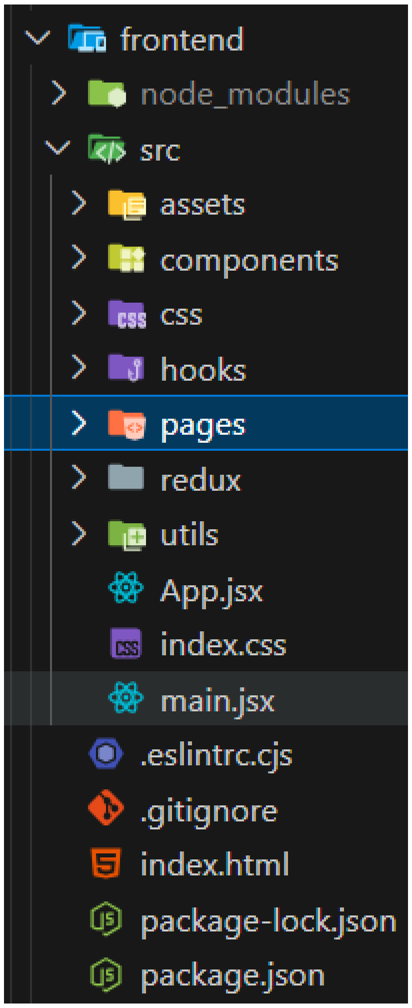
Frontend integration: React + Vite with JWT token handling

Real-time updates reflected on the UI

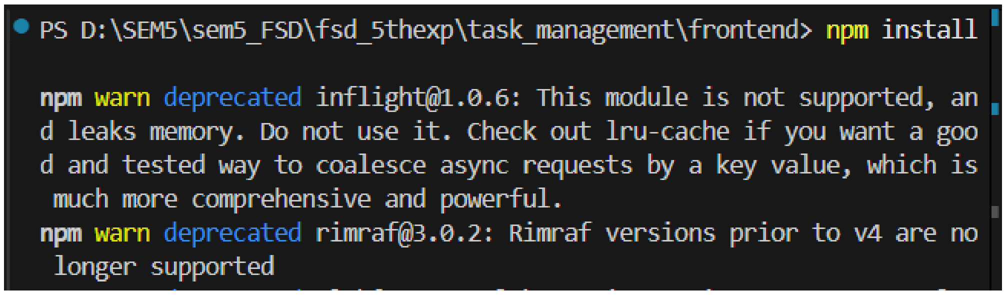
React-Redux for state management

Environment-based config using .env

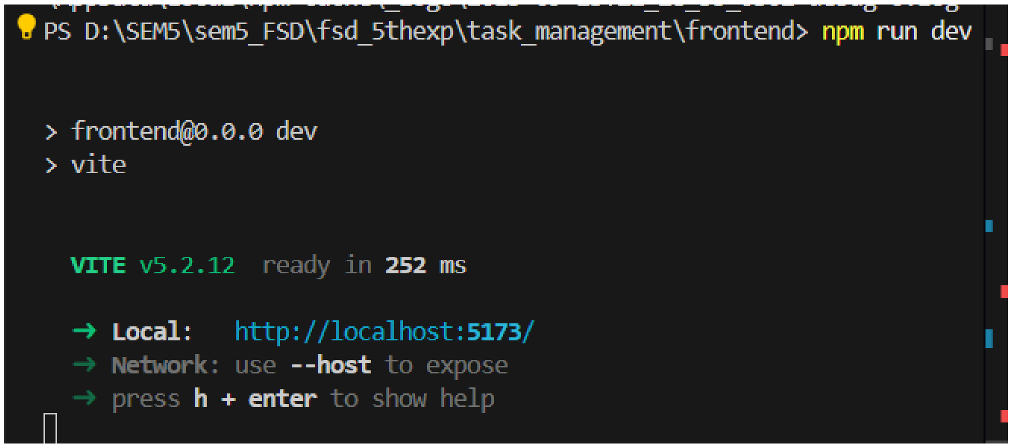
Frontend



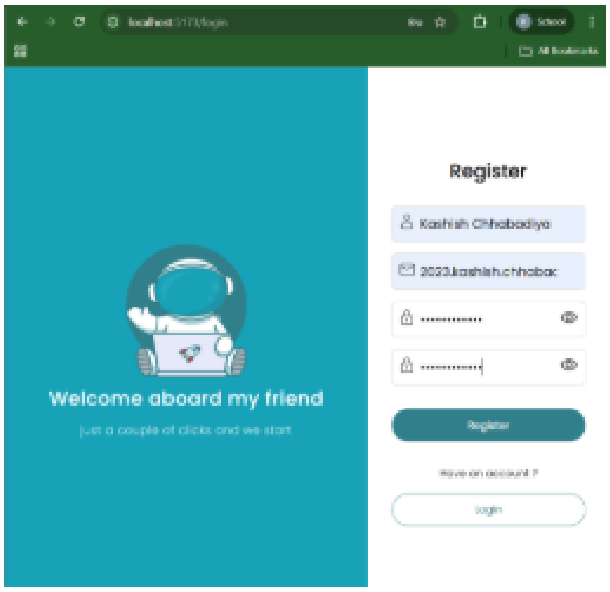
```
frontend
  node_modules
  src
    assets
    components
    css
    hooks
    pages
    redux
    utils
    App.jsx
    index.css
    main.jsx
  .eslintrc.cjs
  .gitignore
  index.html
  package-lock.json
  package.json
```



```
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
```



localhost:5173/login

Register

Kashish Chhabadiya

2023.kashish.chhabac

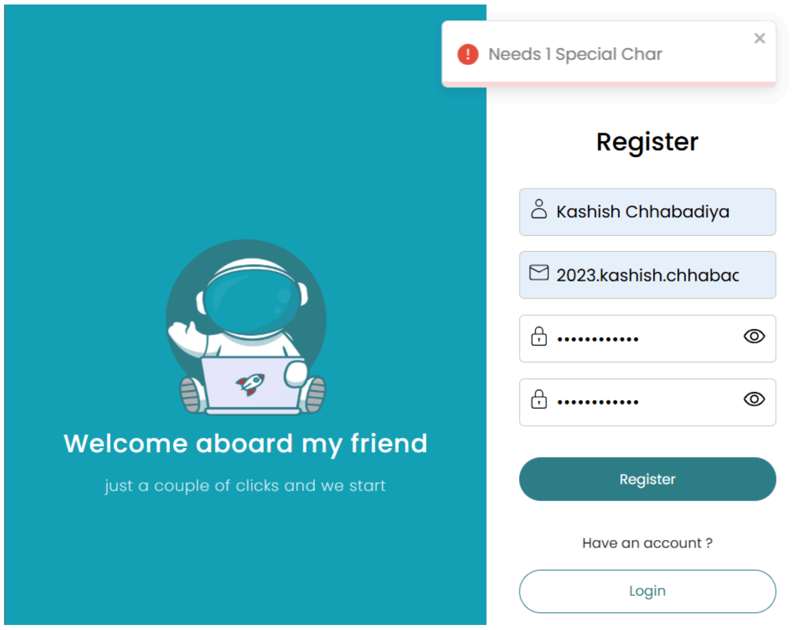
Register

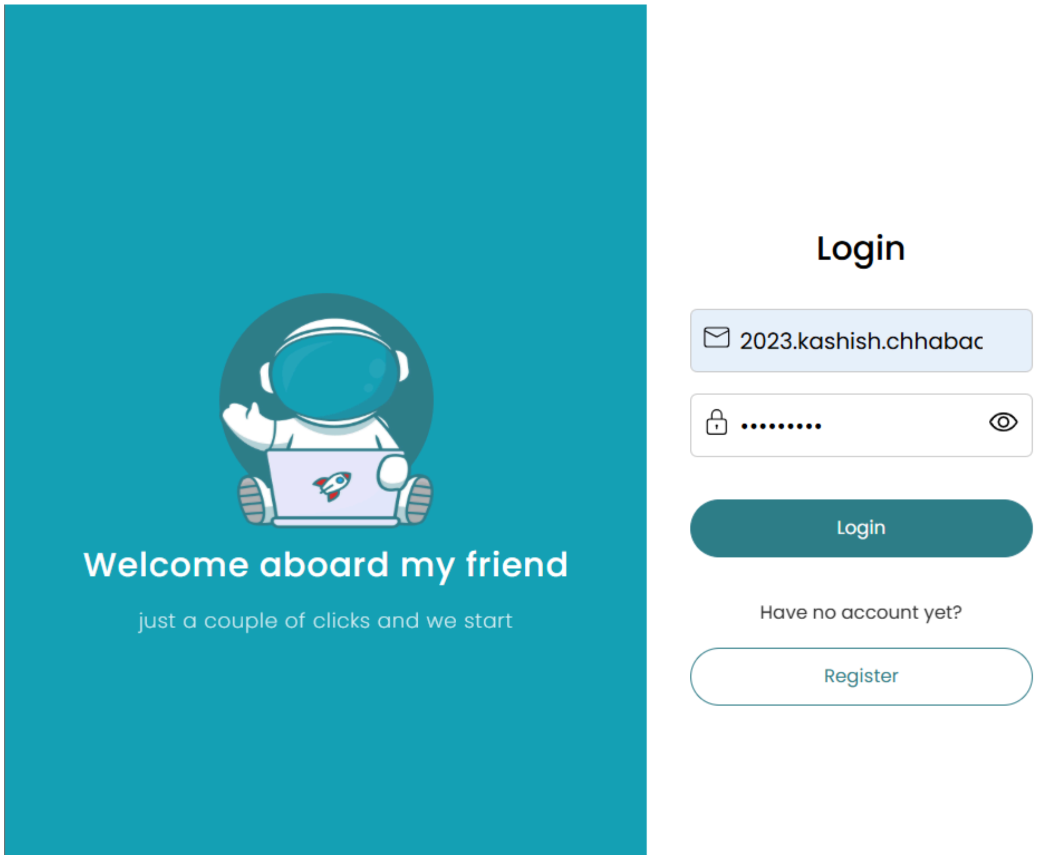
Have an account ?

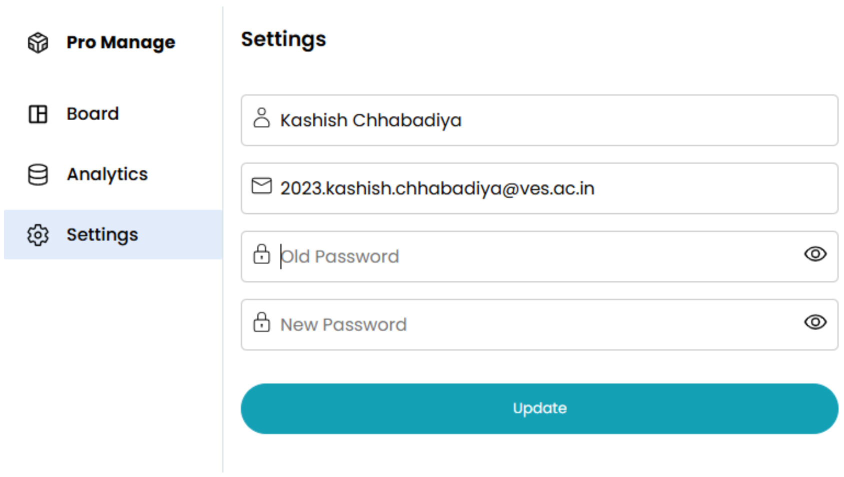
login

Welcome aboard my friend

just a couple of clicks and we start


Needs 1 Special Char
Welcome aboard my friend
just a couple of clicks and we start
Register
Kashish Chhabadiya
2023.kashish.chhabac
••••••••••••
••••••••••••
Register
Have an account ?
Login


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



Settings

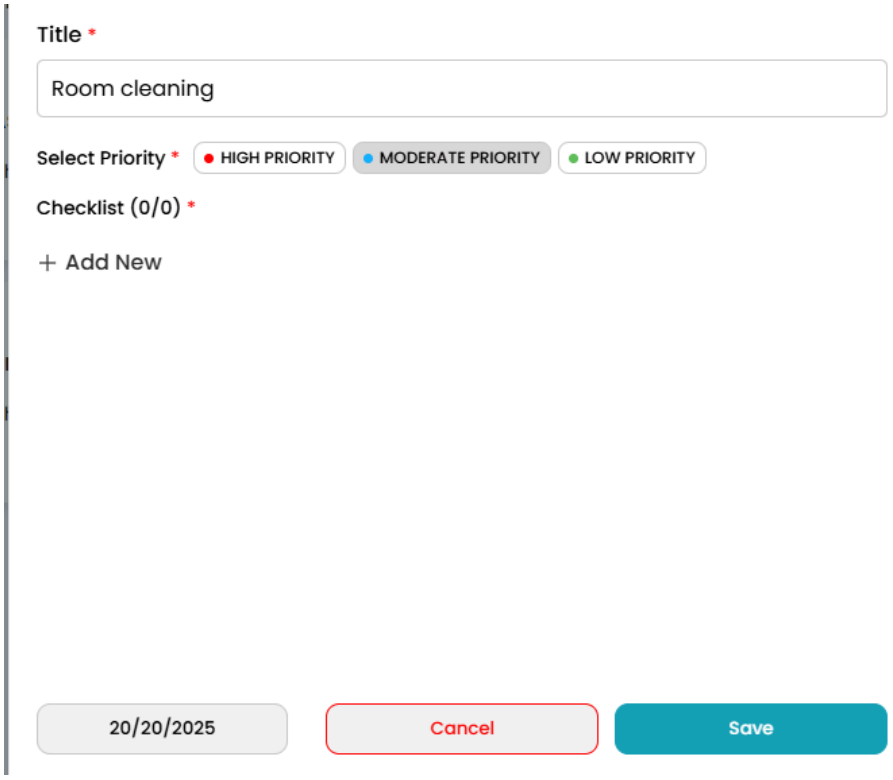
Kashish Chhabadiya

2023.kashish.chhabadiya@ves.ac.in

Old Password

New Password

Creation of tasks :



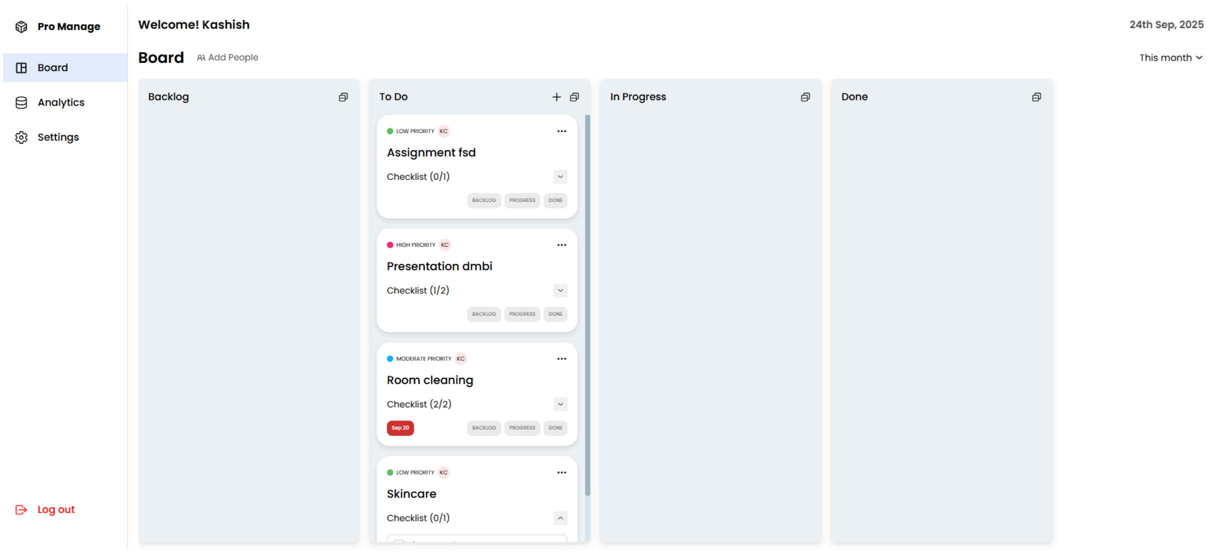
Title *

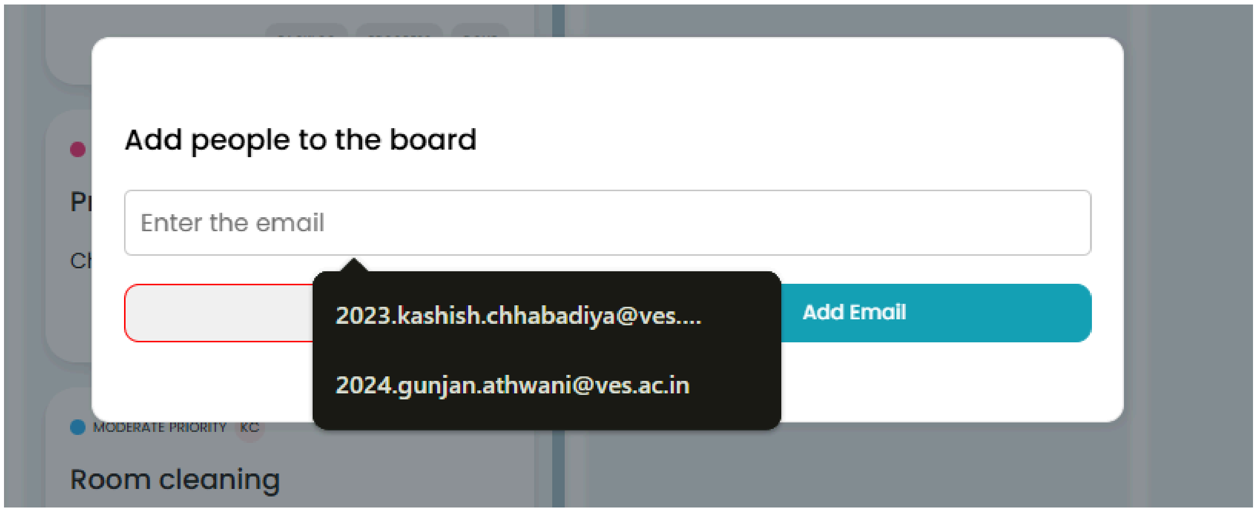
Room cleaning

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

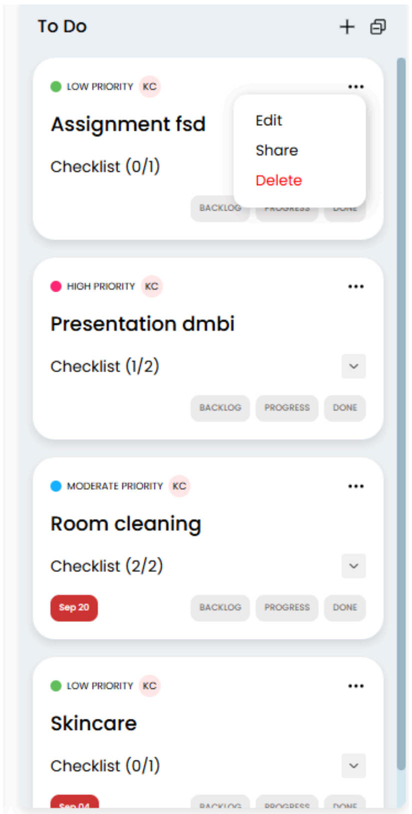
Checklist (0/0) *

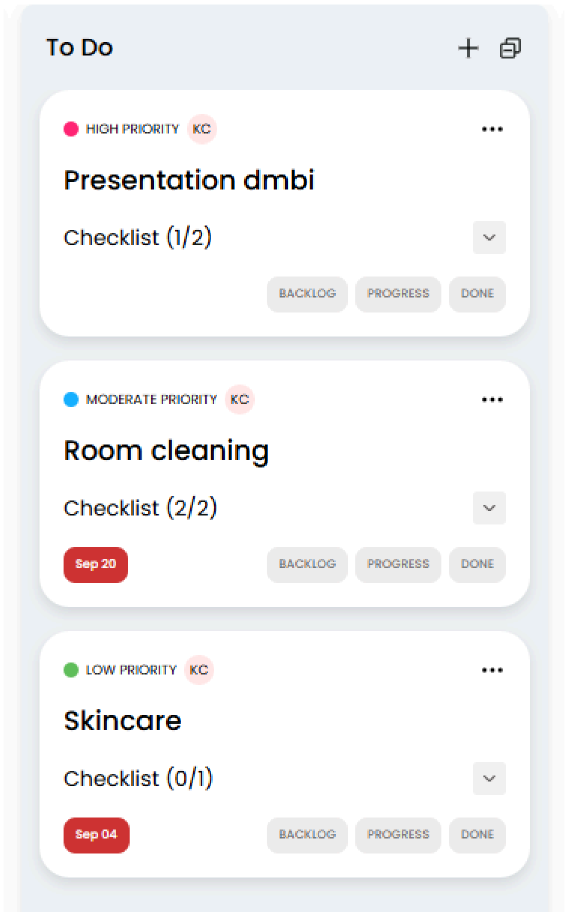
20/20/2025


Pro Manage
Board
Analytics
Settings
Log out
Welcome! Kashish
Board Add People
24th Sep, 2025
This month
Backlog
To Do
In Progress
Done
LOW PRIORITY KC
Assignment fsd
Checklist (0/1)
BACKLOG PROGRESS DONE
HIGH PRIORITY KC
Presentation dmbi
Checklist (1/2)
BACKLOG PROGRESS DONE
MODERATE PRIORITY KC
Room cleaning
Checklist (2/2)
Sep 20
BACKLOG PROGRESS DONE
LOW PRIORITY KC
Skincare
Checklist (0/1)

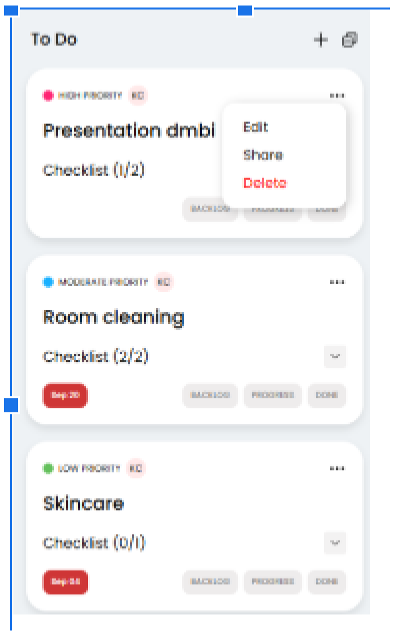

Add people to the board
Enter the email
Add Email
2023.kashish.chhabadiya@ves....
2024.gunjan.athwani@ves.ac.in

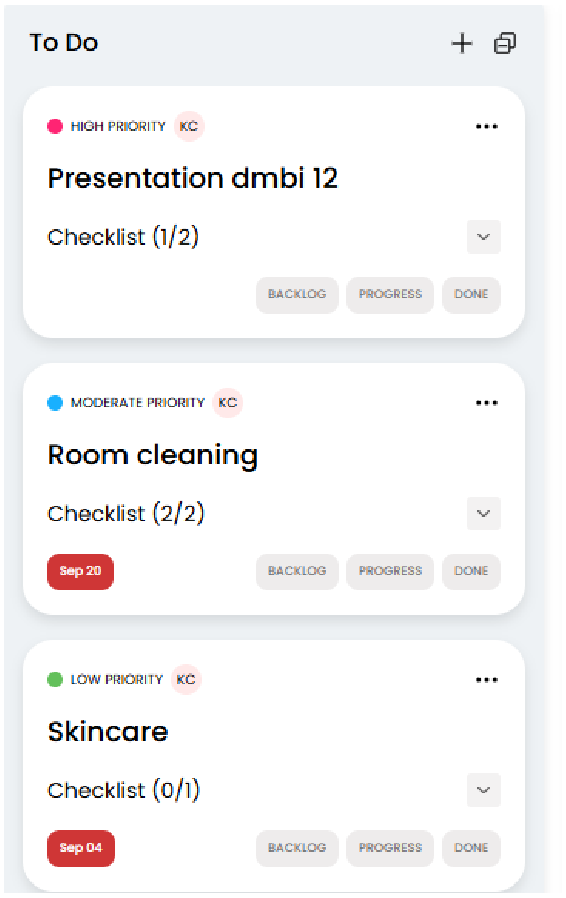
<u>Delete :</u>





<u>Editing tasks :</u>




To Do

● HIGH PRIORITY KC

Presentation dmbi 12

Checklist (1/2)

BACKLOG PROGRESS DONE

● MODERATE PRIORITY KC

Room cleaning

Checklist (2/2)

Sep 20 BACKLOG PROGRESS DONE

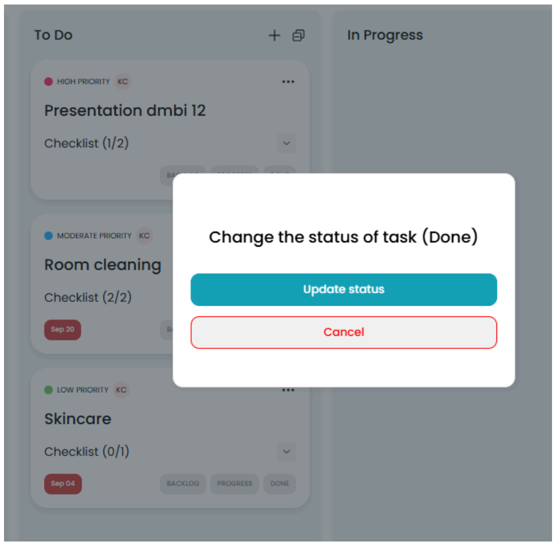
● LOW PRIORITY KC

Skincare

Checklist (0/1)

Sep 04 BACKLOG PROGRESS DONE

Update task status :


To Do

In Progress

Presentation dmbi 12

Checklist (1/2)

Room cleaning

Checklist (2/2)

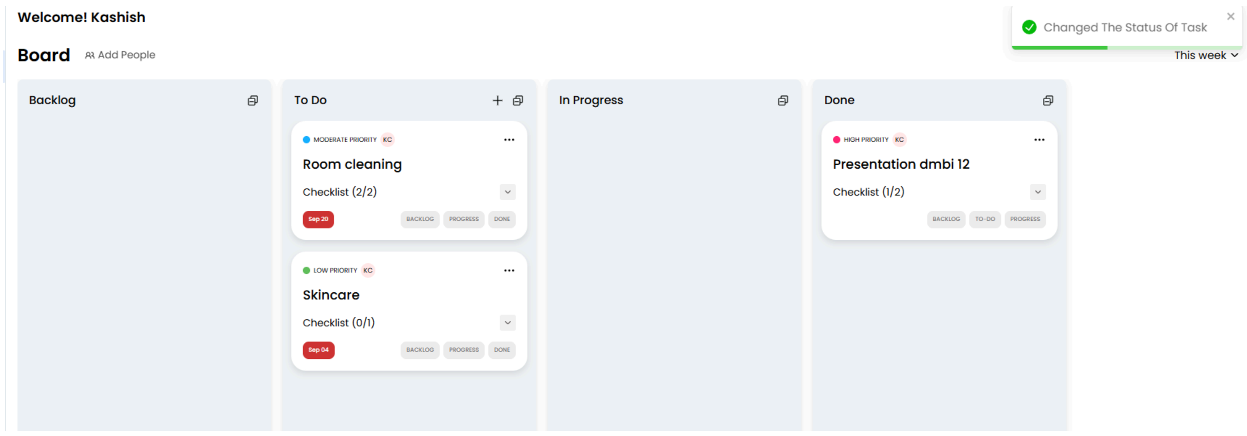
Skincare

Checklist (0/1)

Change the status of task (Done)

Update status

Cancel



Welcome! Kashish

Board

Add People

Changed The Status Of Task

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