

Project Manual

Backend Developer Intern Assignment - PrimeTrade.ai

Prepared by: RV Nayan

September 28, 2025

Contents

1	Introduction	2
2	Setup & Running the Project	2
2.1	Running Locally	2
2.1.1	Backend Setup	2
2.1.2	Frontend Setup	2
2.1.3	Setting up MongoDB via Docker	2
2.2	Running via Docker Compose - Recommended	3
3	Docker Compose Overview	3
4	Usage Tutorial	3
4.1	Login / Registration	3
4.2	User Features	3
4.3	Admin Features	4
4.4	Task Creation Features	4
5	Environment Notes	4
6	Troubleshooting	4
7	Scalability & MongoDB in Docker	5
7.1	Why MongoDB in Docker	5
7.2	Scalability Considerations	5
8	Appendix — Dependency Files	5
8.1	Backend - package.json	5
8.2	Frontend - package.json	5
9	Links	6

1 Introduction

This manual provides step-by-step instructions to set up, run, and use the Intern Assignment project both locally and with Docker Compose. It also explains key features, environment requirements, troubleshooting tips, and scalability considerations.

2 Setup & Running the Project

2.1 Running Locally

2.1.1 Backend Setup

```
cd backend
npm init -y
npm install express mongoose bcryptjs jsonwebtoken dotenv cors helmet
  morgan express-rate-limit yup swagger-ui-express yamls
npm install -D nodemon
```

Create the .env file:

```
PORT=4000
MONGO_URI=mongodb://localhost:27017/intern_assignment
JWT_SECRET=supersecretkey
JWT_EXPIRES_IN=15m
CORS_ORIGIN=http://localhost:5173
RATE_LIMIT_WINDOW_MS=60000
RATE_LIMIT_MAX=100
```

Run MongoDB locally:

```
mongod --dbpath /path/to/data
```

Run the backend:

```
npm run dev
```

2.1.2 Frontend Setup

```
cd frontend
npm create vite@latest frontend
npm install axios react-router-dom recharts
npm run dev
```

2.1.3 Setting up MongoDB via Docker

Ensure the Docker engine is running when using these commands:

```
docker run -d --name intern-mongo -p 27017:27017 mongo:6
```

Start the container and access MongoDB:

```
docker start intern-mongo
docker exec -it intern-mongo mongosh
```

Open the browser:

```
http://localhost:5173
```

2.2 Running via Docker Compose - Recommended

Your `docker-compose.prod.yml` file is configured to run the backend, frontend, and MongoDB with a single command:

```
docker pull rnyn666140/primetrade-backend:latest
docker pull rnyn666140/primetrade-frontend:latest
docker compose -f docker-compose.prod.yml up --build
```

The frontend will be available at:

```
http://localhost:5173
```

3 Docker Compose Overview

The Compose file runs three services: MongoDB for the database, the backend API connected to MongoDB, and the frontend React UI. Environment variables in the Compose file ensure proper service connections. The backend connects to MongoDB using `mongodb://mongo:27017/intern_assignment`, while the frontend communicates with the backend on port 4000.

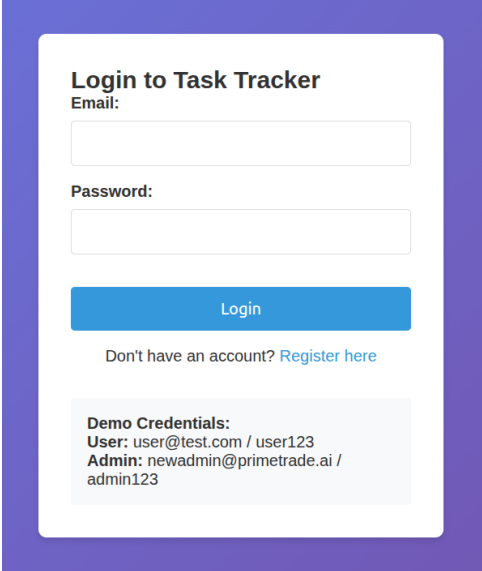
4 Usage Tutorial

4.1 Login / Registration

To begin using the application, open the frontend URL in your browser. You can register a new account by filling in your details. If you are an administrator, you must use the special code 'LMAO' during login. Once logged in, the system will return a JWT token, which will be used for authentication across all subsequent requests.

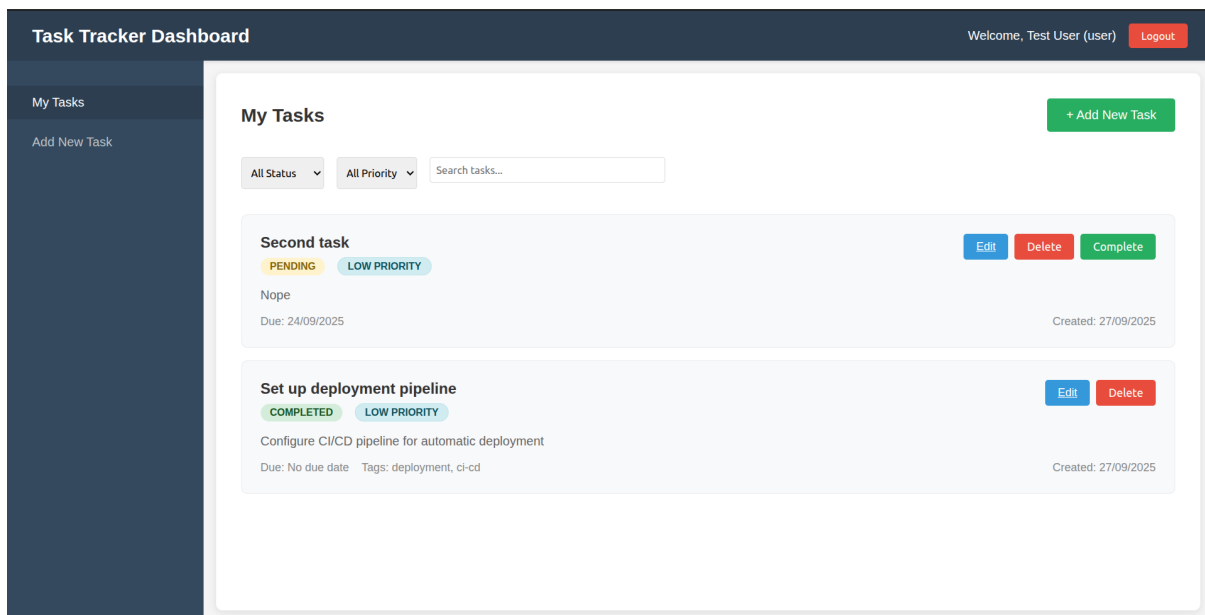
4.2 User Features

After logging in, users can create tasks by providing a title, description, due date, and status. Tasks can later be updated directly from the dashboard, where users can edit details as needed. The dashboard also provides filters, allowing tasks to be viewed based on their current status, making it easier to track progress.

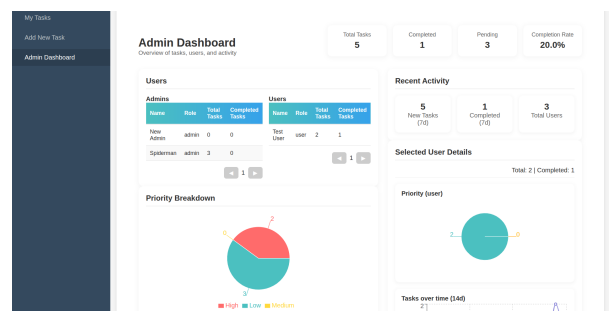


The screenshot shows a login form titled "Login to Task Tracker". It has two input fields: "Email:" and "Password:". Below the password field is a blue "Login" button. Under the button, there is a link: "Don't have an account? [Register here](#)". At the bottom, there is a section titled "Demo Credentials:" with the following text: "User: user@test.com / user123", "Admin: newadmin@primetrade.ai / admin123".

4.3 Admin Features



Administrators have access to additional features beyond those available to regular users. They can view all registered users in the system, track tasks across multiple accounts, and monitor both task statuses and deadlines. This makes it easier to oversee team progress and identify bottlenecks.



4.4 Task Creation Features

Tasks can include a title, description, due date, and priority level. Administrators can assign tasks to specific users, ensuring accountability. Each task can be marked with a status such as Pending, In Progress, or Completed, which helps both users and administrators track progress effectively.

5 Environment Notes

When running locally, connect to MongoDB using `mongodb://localhost:27017/intern_assignment`. When running with Docker Compose, MongoDB is managed internally, and the connection string should be `mongodb://mongo:27017/intern_assignment`.

6 Troubleshooting

If the backend is not connecting to MongoDB, verify that the `MONGO_URI` in the `.env` file or in the `docker-compose.yml` is set correctly. If ports are already in use, stop the conflict containers or processes. If Docker cannot find the required images, pull them man-

ually with `docker pull rnyn666140/primetrade-backend:latest` and `docker pull rnyn666140/primetrade-frontend:latest`.

7 Scalability & MongoDB in Docker

7.1 Why MongoDB in Docker

Running MongoDB inside Docker ensures isolation from the host environment, portability across different machines, and consistency between development and production setups. It also enables scalability through replication and sharding.

7.2 Scalability Considerations

The project is designed with scalability in mind. Docker Compose provides single-command deployment. The REST API follows a modular structure, making it easier to scale individual services. MongoDB supports sharding and replica sets, ensuring database scalability. JWT authentication allows stateless scaling across servers, and container orchestration tools can handle load balancing.

8 Appendix — Dependency Files

8.1 Backend - package.json

```
{
  "name": "backend",
  "version": "1.0.0",
  "main": "src/server.js",
  "scripts": {
    "dev": "nodemon src/server.js"
  },
  "dependencies": {
    "bcryptjs": "^2.4.3",
    "cors": "^2.8.5",
    "dotenv": "^16.0.3",
    "express": "^4.18.2",
    "express-rate-limit": "^6.5.1",
    "helmet": "^6.0.1",
    "jsonwebtoken": "^9.0.0",
    "mongoose": "^7.0.4",
    "morgan": "^1.10.0",
    "swagger-ui-express": "^4.6.3",
    "yamljs": "^0.3.0",
    "yup": "^1.0.2"
  },
  "devDependencies": {
    "nodemon": "^2.0.22"
  }
}
```

8.2 Frontend - package.json

```
{
  "name": "frontend",
  "version": "1.0.0",
  "scripts": {
    "dev": "vite"
  },
  "dependencies": {
    "axios": "^1.4.0",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-router-dom": "^6.14.2",
    "recharts": "^2.7.0"
  },
  "devDependencies": {
    "vite": "^5.0.0"
  }
}
```

9 Links

- GitHub: <https://github.com/RVNayan/PrimeT-FB>
- Dockerhub:
 - Frontend: <https://hub.docker.com/r/rnyn666140/primetrade-frontend>
 - Backend: <https://hub.docker.com/r/rnyn666140/primetrade-backend>