Hello everyone! Today, our team will introduce and demonstrate a Dockerized Task Management System project.

This project consists of three main components: Node.js backend, MongoDB for data storage, and Docker Compose for synchronous application deployment.

We are using Express to set up a RESTful API on Node.js, Mongoose to connect to MongoDB, and Docker to containerize the entire system. Docker Compose is used to easily manage services and configurations.

Let's take a look at the code. First is the project structure with the index.js file, which defines the API endpoints for CRUD requests.

The index.ejs file contains HTML code rendered with Express, which is the user interface for them to add, edit, and delete tasks.

The Dockerfile in the backend directory sets up the Node.js environment and loads the dependencies. We have a CMD command to run the application when the container starts.

Finally, the docker-compose.yml file, which acts as a container orchestrator. The app-network is created so that the services can connect properly to each other.

Our task data will be stored in MongoDB, a NoSQL database that can store dynamic data. Docker Compose also helps us set up MongoDB easily.

Now our team would like to deploy and run the application. But before running, we have to open Docker Desktop which our team has downloaded and is also mentioned in the ReadMe.txt file.

Now we will run the docker-compose up --build command to start the services.

Docker Compose will start both the backend and MongoDB, checking that both containers are running and connected to each other successfully.

This is the user interface of the application. We will try to add a new task so you can see that this task will be displayed below the add task section.

Now I'll update, mark complete, and delete a task. You can see that all of this works smoothly.

All information is returned and displayed correctly thanks to the APIs that have been set up.