Page No.
Date

Task Management APP

④

We have to create 5 API's

(APIs)

- Post /task → on success return id / on failure return error message
- Get /tasks → fetch all task from db.
  - ↳ Support query param by status
  - ↳ retrive single task via Id.
- Put /task/:id
  - ↳ update task via task id.
- delete /task/:id
  - ↳ delet a task by it's id.

① Post → Create. (Post /tasks)

② get → ③ → GET /tasks

Get /tasks → ↳ ① fetch All task add Pagination.

/tasks?status= → ↳ ② find by id take id from query param.

/task/:id → ↳ ③ find by status → take status from query param.

↓

① Completed ② ~~Pending~~ In Progress ③ Aborted.

③ Put → find by id and update
↳ /tasks/:id

④ delete → find by id and delete.
↳ /tasks/:id

Page No.
Date

Database.

→ mongodb ∈ with mongoose.
→ design or create model. (with validation)

- title (string, required)
- description (string, optional)
- status (string, default value: Pending
  enum: ["Pending", "in progress", "completed"])

→ model created at, updated at.

---

will use

port = 2000.
mongo URI

Folder Structure:

→ Project → Task Manager.

- model
  - ↳ Task.js (model or mongoose schema)
- routes
  - ↳ task.js (API routes).
    - ↳ Right now we have only 4 api to mong
- controllers
  - ↳ task controller.js (logics)
- config
  - ↳ db.js (database connection)
- app.js (express app.
- ↳ server.js

Page No.
Date

what we have done (first step).

(1) Created folder structure as per requirement.
↳ done npm init in this process.

(2) installed all the dependencies.
(1) npm i express
(2) npm i mongoose
(3) npm i dotenv
(4) npm i mongodb

(3) Setted git : git init
Initialised

(4) Connecting to git hub.

first git push → A Basic folder structure done and pushed to github.

(5) getting mongodb string.
↳ got mongodb connection string.

Page No.
Date

Connecting to database. ✓✓

We will use Mongoose.connect() ⇐ method.

Imported mongoose
Imported "dotenv/config";

Const ~~ConnectionString~~ = process.env.connection_String.

Const dbConnect = async () ⇒
  await mongoose.connect(string)
  }.

↑ use try catch block.

and exported the dbconnect.

Creating Server. ✓✓

import express from "express"
import dbConnect from "Path"
import "dotenv/config";

const port = process.env.port;

Const app = express()

dbconnect().then(() ⇒ {
  try {
    app.listen(port, () ⇒ {
      console.log("server is running")
    });
  catch (error) { log error)
  }});

Page No.
Date

Creating Model.

Imported mongoose.

created Schema.

↳ const taskSchema = new mongoose.Schema({

```
title: {
    type: String,
    require: true,
},
description: {
    type: String
},
status: {
    type: String,
    default: "Pending",
    enum: {
        values: ["Pending", "inProgress", "completed"],
        message: '${VALUE} is not valid type',
    },
    require: true,
},
{ timeStamps: true }
);
```

(Id will get created automatic)

only this values will get selected else ~~use~~ error.

it will set timeStamp automatic

const Task = mongoose.model("task", taskSchema)

Export default Task;

A = Attackers View (followed Secure Practices)

Page No.
Date

API creation.

Post API /tasks.

A if it is Post api → First thing to do is validate and sanitize req.body
It will prevent us from Db Polution, and also with attacks.

Let's creat function where we will validate and Sanitize the data.

① - destructure → title, description, status
② if !title then throw error title required.
↳ only because title is only required.

A ② validated the length of fields just because attacker can fill our database with lots of data and that can be Problem

done validation

then created Post route added validations

get API /tasks.

A for get API → we always try to send data to only Authorize user, and we try not to expose to much information as it can be vulnerable and it can create. allocks.

we will try not to expose to much data.

A we dont have any requirment to create auth middleware thats why we are setting this route open, but in production we will use cors so that it cannot be accessible to other website.

we created a route and taken Status from query,param.

```
const Status = req.query.Status;
let taskData;
if (Status) {
    taskData = await Task.find({ Status: Status });
}
else {
    taskData = await Task.find();
}
res.send(Taskdata);
```

Page No.
Date

Task via id

So i have taken id via = req.params.id

```
const dataTask.findbyID (id)
    if (!data) {
        throw new error ("invalid object id")
    }
res.send (data)
```

Task.Put ("/task/:id")

Validation run (req)

// destructure and Sanitize req.body.

```
const { title, status, description } = req.body.
const id = req.params.id ;
```

// Created new object.

```
const updatedata = { }
    if (title) updatedata.title = title ;
    if (status) updatedata.status = status
    if (description) updatedata.description = description
```

```
const datatoSend = await Task.findbyIdAndUpdate (
    id, updatedata, { new: true, runValidator: true }
)
```

Id to find → id
updatedata → updatedata
runValidator: true → if will run validator to ensure we are Sanitizing data.

res.send (datatoSend)

Page No.
Date

Created new function or module just to verify the Id

```
export const validateId = (id) => {
    if (!id) {
      throw new Error("Id is required");
    }
    if (!mongoose.Types.ObjectId.isValid(id))
     {
      throw new error("invalid id")
    }
};
```

Learn new thing while creating this Project.

Search feature.

Task.find({title: {$regex: search, $options: "i"}})

It will give all the data matching to the keyword ~~if will search all the~~

It makes Keyword case-insensitive

done Registration,