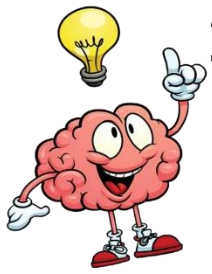
GROUP PERFORMANCE TEST #1: DEVELOPMENT OF A RESTFUL API

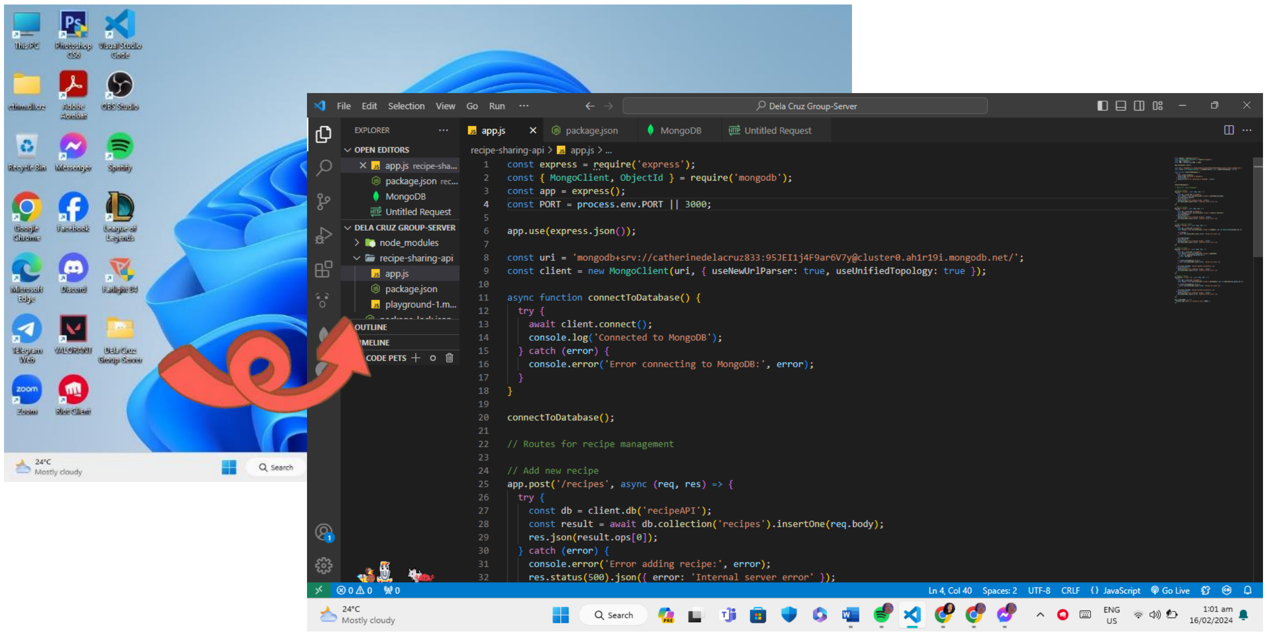
Topics Included:

- ➢ Setup and HTTP Request
- ➢ Handling Input Validation, Post, Put, Delete Requests
- ➢ Mongoose, MongoDB, CRUD

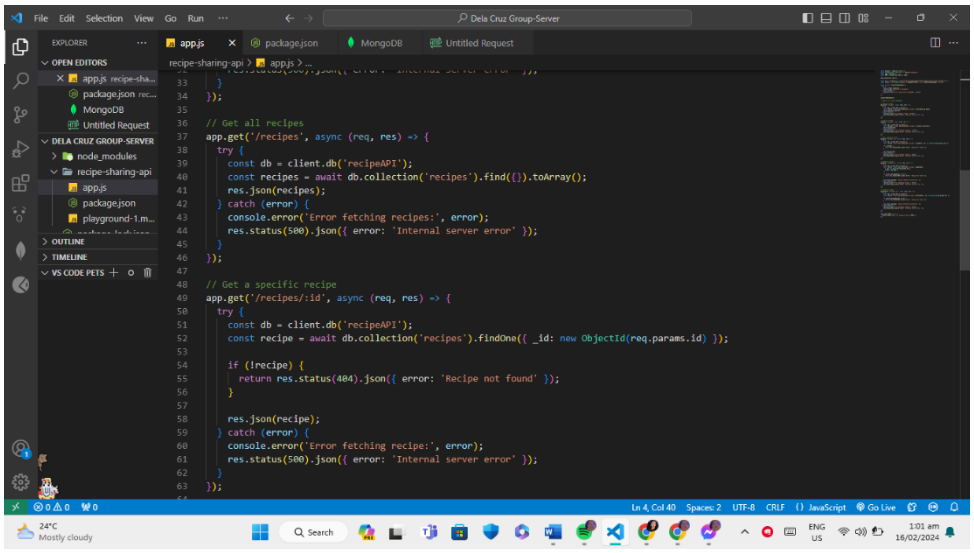
Submitted By: Famille Decena
Aldean Matthew Almonte
Cemesa Jane Magalang
Catherine Dela Cruz

- Ensure that your Recipe Sharing API project is completed and thoroughly tested. Make sure all required functionality is implemented, and your code is well-documented and organized.

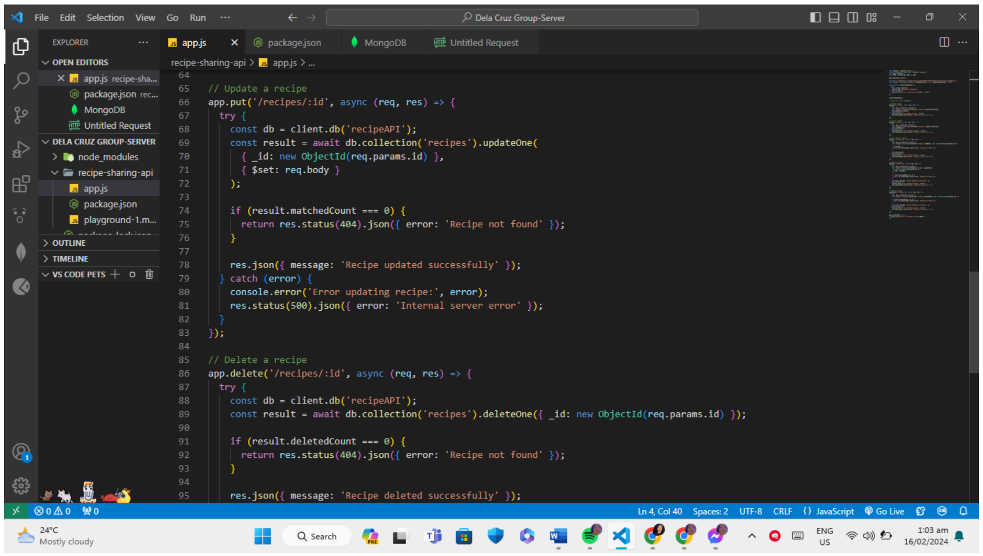




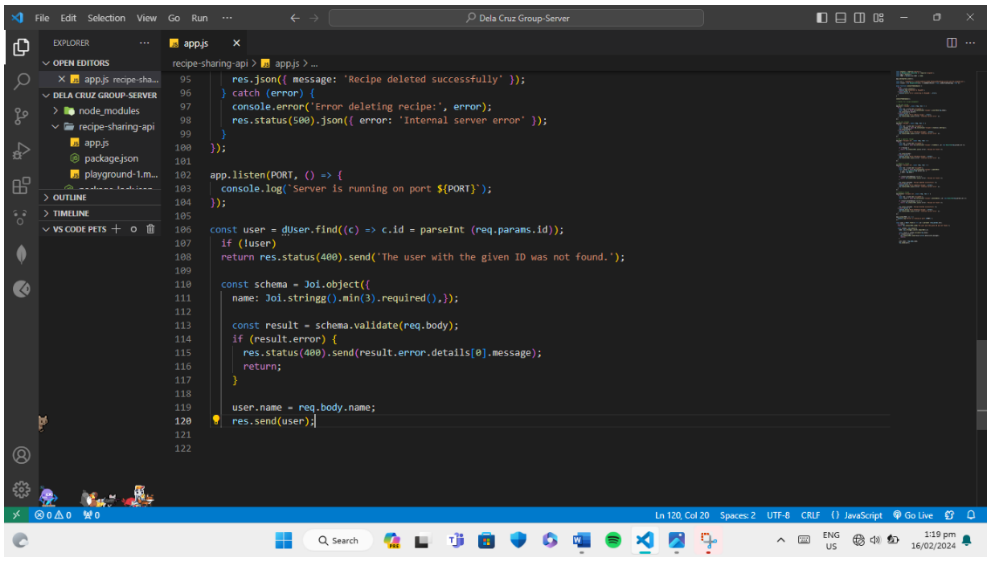
```javascript
const express = require('express');
const { MongoClient, ObjectId } = require('mongodb');
const app = express();
const PORT = process.env.PORT || 3000;

app.use(express.json());

const uri = 'mongodb+srv://catherinedelacruz833:95JEI1j4F9ar6V7y@cluster0.ah1r19i.mongodb.net/';
const client = new MongoClient(uri, { useNewUrlParser: true, useUnifiedTopology: true });

async function connectToDatabase() {
  try {
    await client.connect();
    console.log('Connected to MongoDB');
  } catch (error) {
    console.error('Error connecting to MongoDB:', error);
  }
}

connectToDatabase();

// Routes for recipe management

// Add new recipe
app.post('/recipes', async (req, res) => {
  try {
    const db = client.db('recipeAPI');
    const result = await db.collection('recipes').insertOne(req.body);
    res.json(result.ops[0]);
  } catch (error) {
    console.error('Error adding recipe:', error);
    res.status(500).json({ error: 'Internal server error' });
```



```javascript
  }
});

// Get all recipes
app.get('/recipes', async (req, res) => {
  try {
    const db = client.db('recipeAPI');
    const recipes = await db.collection('recipes').find({}).toArray();
    res.json(recipes);
  } catch (error) {
    console.error('Error fetching recipes:', error);
    res.status(500).json({ error: 'Internal server error' });
  }
});

// Get a specific recipe
app.get('/recipes/:id', async (req, res) => {
  try {
    const db = client.db('recipeAPI');
    const recipe = await db.collection('recipes').findOne({ _id: new ObjectId(req.params.id) });

    if (!recipe) {
      return res.status(404).json({ error: 'Recipe not found' });
    }

    res.json(recipe);
  } catch (error) {
    console.error('Error fetching recipe:', error);
    res.status(500).json({ error: 'Internal server error' });
  }
});
```

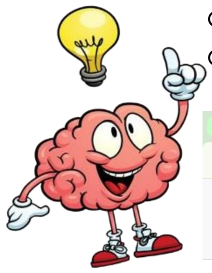


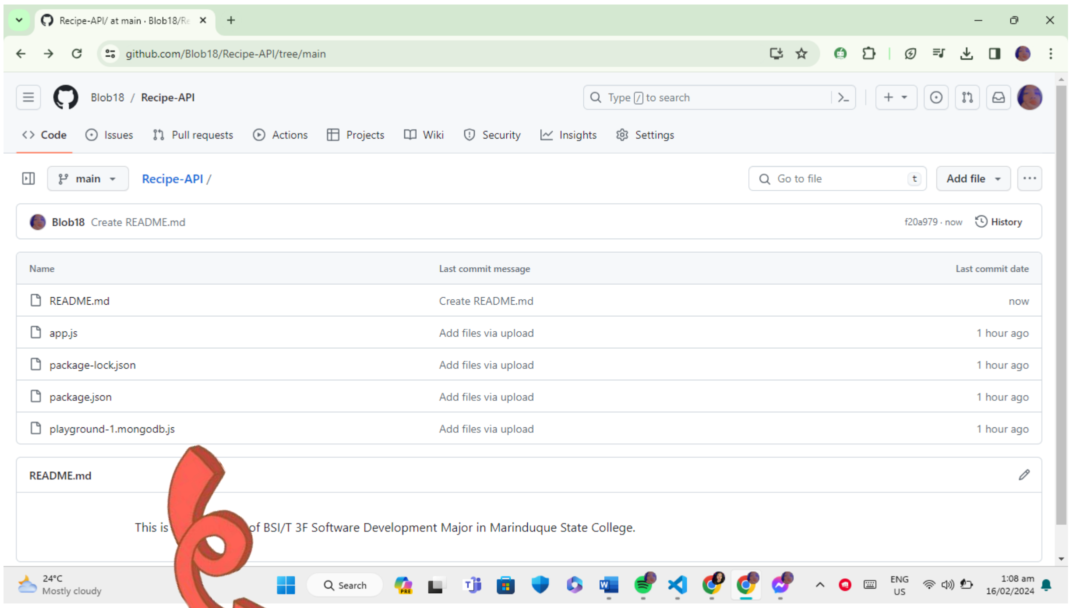
```javascript

// Update a recipe
app.put('/recipes/:id', async (req, res) => {
  try {
    const db = client.db('recipeAPI');
    const result = await db.collection('recipes').updateOne(
      { _id: new ObjectId(req.params.id) },
      { $set: req.body }
    );

    if (result.matchedCount === 0) {
      return res.status(404).json({ error: 'Recipe not found' });
    }

    res.json({ message: 'Recipe updated successfully' });
  } catch (error) {
    console.error('Error updating recipe:', error);
    res.status(500).json({ error: 'Internal server error' });
  }
});

// Delete a recipe
app.delete('/recipes/:id', async (req, res) => {
  try {
    const db = client.db('recipeAPI');
    const result = await db.collection('recipes').deleteOne({ _id: new ObjectId(req.params.id) });

    if (result.deletedCount === 0) {
      return res.status(404).json({ error: 'Recipe not found' });
    }

    res.json({ message: 'Recipe deleted successfully' });
```



```
    res.json({ message: 'Recipe deleted successfully' });
  } catch (error) {
    console.error('Error deleting recipe:', error);
    res.status(500).json({ error: 'Internal server error' });
  }
});

app.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}`);
});

const user = dUser.find((c) => c.id = parseInt (req.params.id));
  if (!user)
  return res.status(400).send('The user with the given ID was not found.');

  const schema = Joi.object({
    name: Joi.stringg().min(3).required(),});

    const result = schema.validate(req.body);
    if (result.error) {
      res.status(400).send(result.error.details[0].message);
      return;
    }

    user.name = req.body.name;
    res.send(user);
```

- Set up a Git repository (e.g., on GitHub, GitLab, Bitbucket) to host your project code. Initialize the repository with your project files, including the source code, documentation, and any additional resources.





github.com/Blob18/Recipe-API/tree/main

Blob18 / Recipe-API

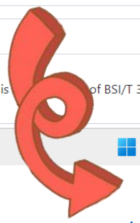
main · Recipe-API /

Blob18 Create README.md — f20a979 · now — History

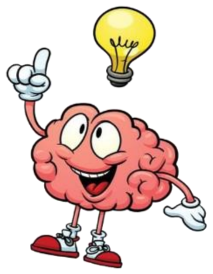
Name	Last commit message	Last commit date
README.md	Create README.md	now
app.js	Add files via upload	1 hour ago
package-lock.json	Add files via upload	1 hour ago
package.json	Add files via upload	1 hour ago
playground-1.mongodb.js	Add files via upload	1 hour ago

README.md

This is of BSI/T 3F Software Development Major in Marinduque State College.

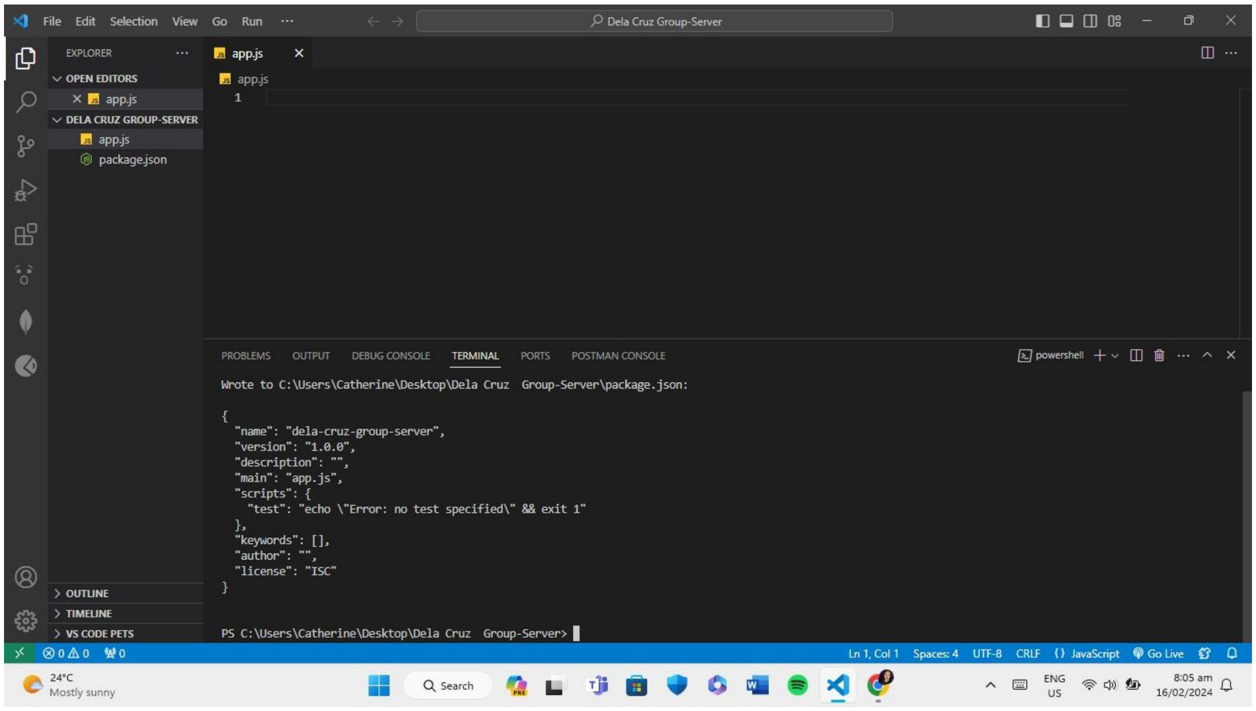


https://github.com/Blob18/Recipe-API

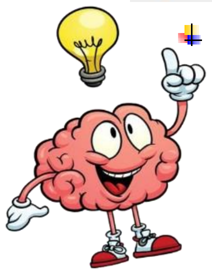


HOW DO WE SET UP THE HTTP REQUEST?

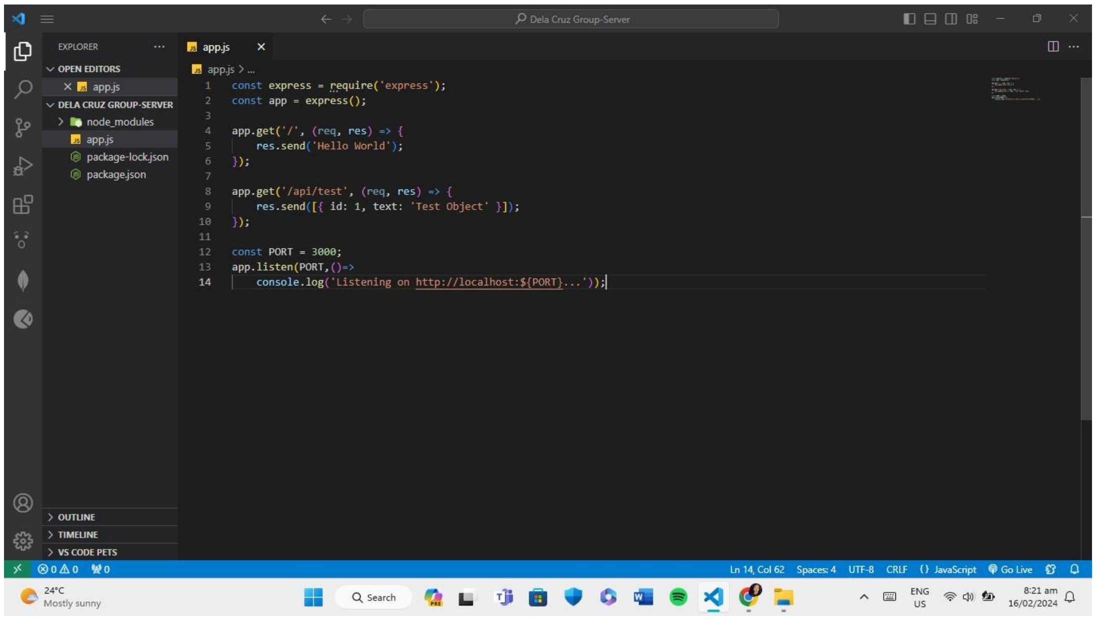
- We download first and install the node.js. After downloading we create our own folder in desktop and we named it as " Dela Cruz Group-Server" and inside the folder that we create a new file call app.js, we use visual studio code by the way



```
Wrote to C:\Users\Catherine\Desktop\Dela Cruz  Group-Server\package.json:

{
  "name": "dela-cruz-group-server",
  "version": "1.0.0",
  "description": "",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}

PS C:\Users\Catherine\Desktop\Dela Cruz  Group-Server>
```

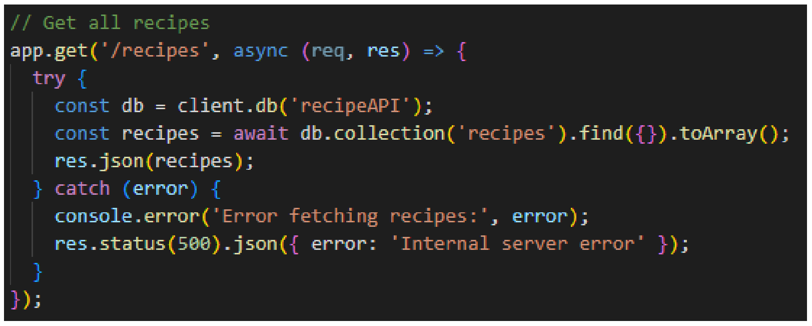


We also install the nodemon package using the npm command, the nodemon environment should be capture any changes on our code base. So after we install the nodemon we run the command nodemon app.js and then we update the PORT variable.

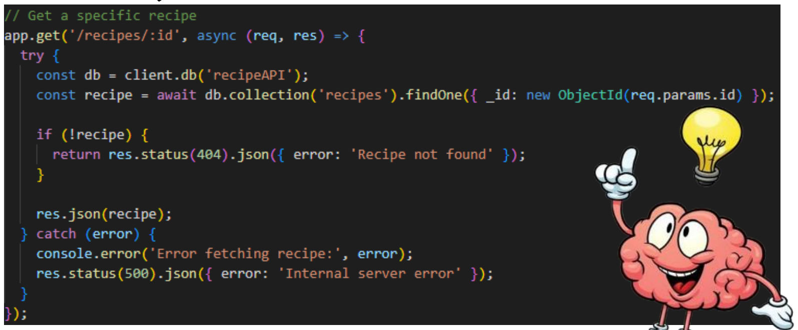


```js
const express = require('express');
const app = express();

app.get('/', (req, res) => {
    res.send('Hello World');
});

app.get('/api/test', (req, res) => {
    res.send([{ id: 1, text: 'Test Object' }]);
});

const PORT = 3000;
app.listen(PORT,()=>
    console.log('Listening on http://localhost:${PORT}...'));
```

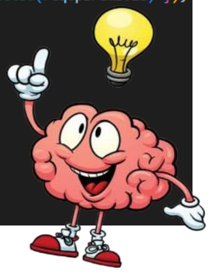
THE HTTP GET REQUEST

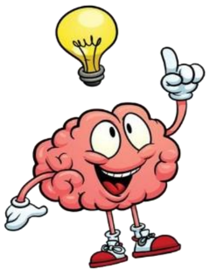


```js
// Get all recipes
app.get('/recipes', async (req, res) => {
  try {
    const db = client.db('recipeAPI');
    const recipes = await db.collection('recipes').find({}).toArray();
    res.json(recipes);
  } catch (error) {
    console.error('Error fetching recipes:', error);
    res.status(500).json({ error: 'Internal server error' });
  }
});
```

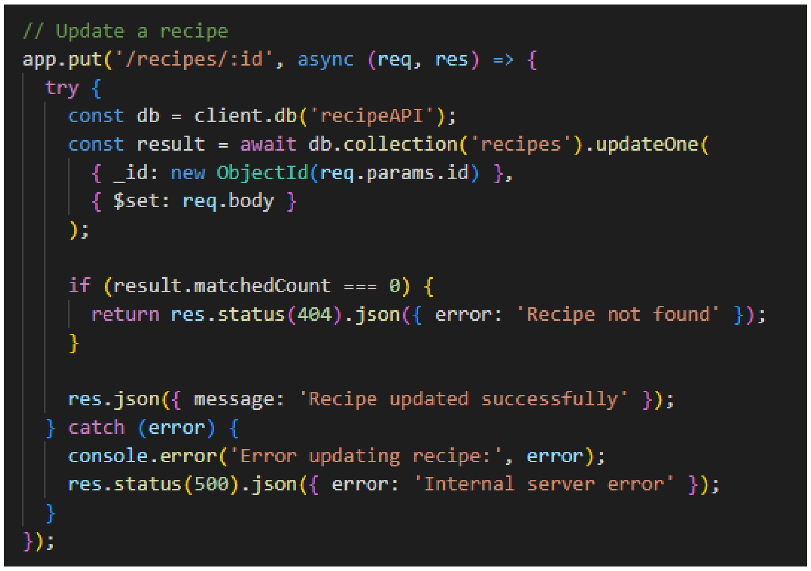


```js
// Get a specific recipe
app.get('/recipes/:id', async (req, res) => {
  try {
    const db = client.db('recipeAPI');
    const recipe = await db.collection('recipes').findOne({ _id: new ObjectId(req.params.id) });

    if (!recipe) {
      return res.status(404).json({ error: 'Recipe not found' });
    }

    res.json(recipe);
  } catch (error) {
    console.error('Error fetching recipe:', error);
    res.status(500).json({ error: 'Internal server error' });
  }
});
```



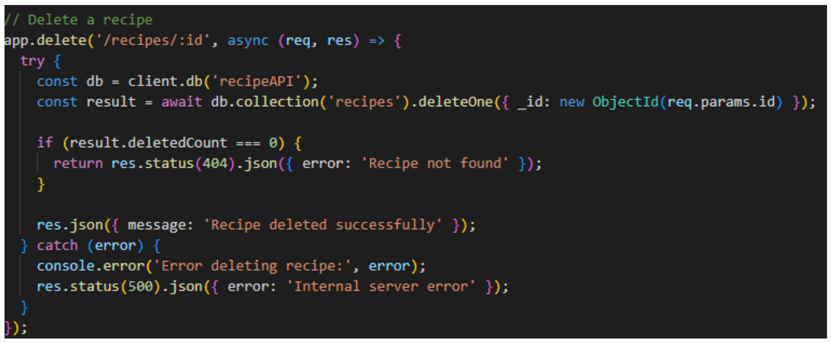


THE HTTP UPDATE REQUEST

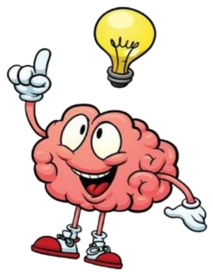


```javascript
// Update a recipe
app.put('/recipes/:id', async (req, res) => {
  try {
    const db = client.db('recipeAPI');
    const result = await db.collection('recipes').updateOne(
      { _id: new ObjectId(req.params.id) },
      { $set: req.body }
    );

    if (result.matchedCount === 0) {
      return res.status(404).json({ error: 'Recipe not found' });
    }

    res.json({ message: 'Recipe updated successfully' });
  } catch (error) {
    console.error('Error updating recipe:', error);
    res.status(500).json({ error: 'Internal server error' });
  }
});
```

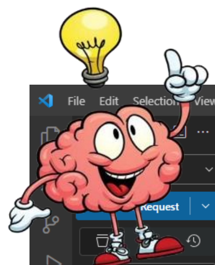
THE HTTP DELETE REQUEST

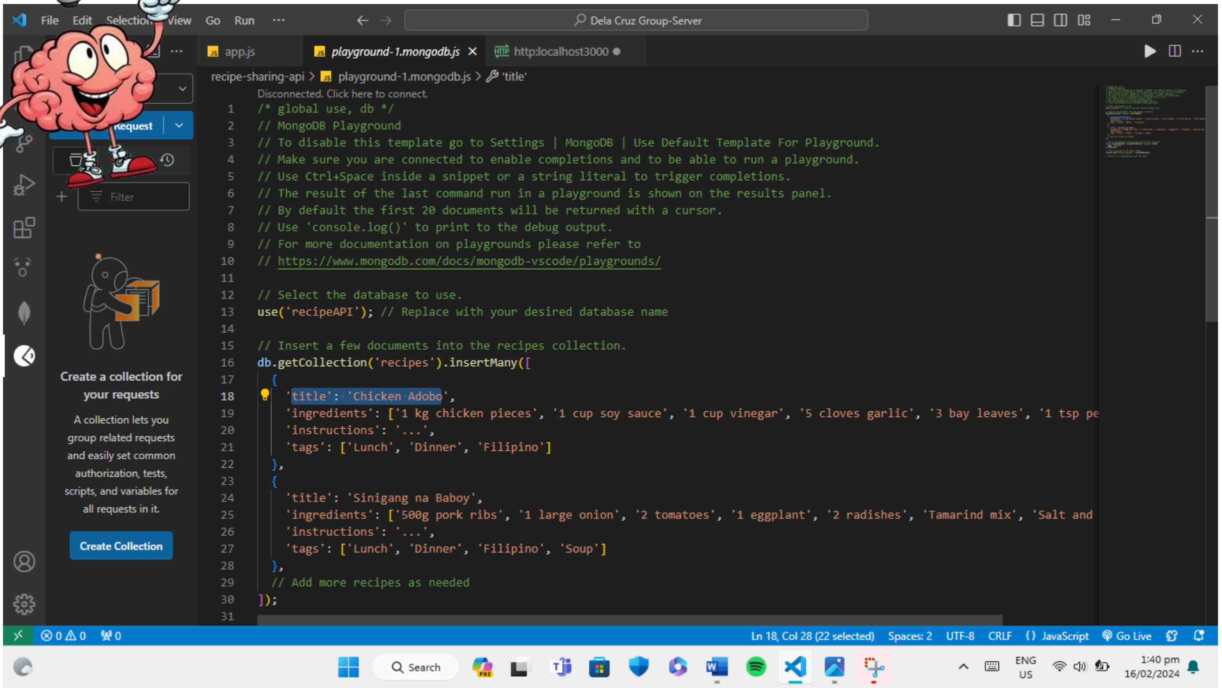


```javascript
// Delete a recipe
app.delete('/recipes/:id', async (req, res) => {
  try {
    const db = client.db('recipeAPI');
    const result = await db.collection('recipes').deleteOne({ _id: new ObjectId(req.params.id) });

    if (result.deletedCount === 0) {
      return res.status(404).json({ error: 'Recipe not found' });
    }

    res.json({ message: 'Recipe deleted successfully' });
  } catch (error) {
    console.error('Error deleting recipe:', error);
    res.status(500).json({ error: 'Internal server error' });
  }
});
```



PLAYGROUND MONGODB .JS



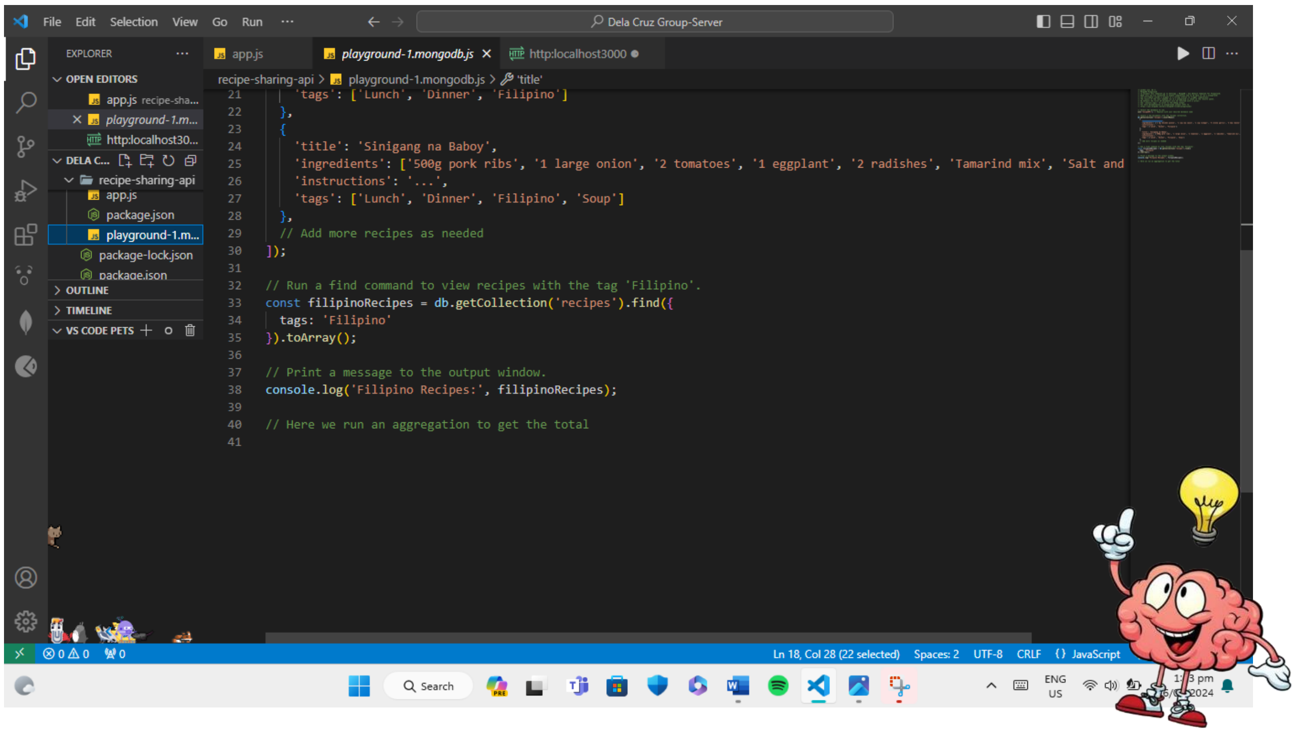

File Edit Selection View Go Run
Dela Cruz Group-Server
app.js
playground-1.mongodb.js
http:localhost3000
recipe-sharing-api > playground-1.mongodb.js > 'title'
Request
Filter
Create Collection
Ln 18, Col 28 (22 selected) Spaces: 2 UTF-8 CRLF {} JavaScript Go Live
Search
ENG US
1:40 pm 16/02/2024

Create a collection for your requests

A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.

Disconnected. Click here to connect.

```javascript
/* global use, db */
// MongoDB Playground
// To disable this template go to Settings | MongoDB | Use Default Template For Playground.
// Make sure you are connected to enable completions and to be able to run a playground.
// Use Ctrl+Space inside a snippet or a string literal to trigger completions.
// The result of the last command run in a playground is shown on the results panel.
// By default the first 20 documents will be returned with a cursor.
// Use 'console.log()' to print to the debug output.
// For more documentation on playgrounds please refer to
// https://www.mongodb.com/docs/mongodb-vscode/playgrounds/

// Select the database to use.
use('recipeAPI'); // Replace with your desired database name

// Insert a few documents into the recipes collection.
db.getCollection('recipes').insertMany([
  {
    'title': 'Chicken Adobo',
    'ingredients': ['1 kg chicken pieces', '1 cup soy sauce', '1 cup vinegar', '5 cloves garlic', '3 bay leaves', '1 tsp pe
    'instructions': '...',
    'tags': ['Lunch', 'Dinner', 'Filipino']
  },
  {
    'title': 'Sinigang na Baboy',
    'ingredients': ['500g pork ribs', '1 large onion', '2 tomatoes', '1 eggplant', '2 radishes', 'Tamarind mix', 'Salt and
    'instructions': '...',
    'tags': ['Lunch', 'Dinner', 'Filipino', 'Soup']
  },
  // Add more recipes as needed
]);
```



```javascript
    'tags': ['Lunch', 'Dinner', 'Filipino']
  },
  {
    'title': 'Sinigang na Baboy',
    'ingredients': ['500g pork ribs', '1 large onion', '2 tomatoes', '1 eggplant', '2 radishes', 'Tamarind mix', 'Salt and
    'instructions': '...',
    'tags': ['Lunch', 'Dinner', 'Filipino', 'Soup']
  },
  // Add more recipes as needed
]);

// Run a find command to view recipes with the tag 'Filipino'.
const filipinoRecipes = db.getCollection('recipes').find({
  tags: 'Filipino'
}).toArray();

// Print a message to the output window.
console.log('Filipino Recipes:', filipinoRecipes);

// Here we run an aggregation to get the total
```