Creating a RESTful API using express.js and create a database and index in MongoDB

Name: Satyala Supriya

Roll Number: 20KH1A0594

College: Sri Venkateswara College Of Engineering, Kadapa (CSE)

Set up your project:

- Initialize a new Node.js project: **npm init -y**
- Install Express.js and MongoDB: **npm install express mongoose**

Create your Express.js server:

- Create a **server.js** file.
- Set up your Express server and connect to MongoDB.

```
const express = require('express');
const mongoose = require('mongoose');
const app = express();
const PORT = 3000;
mongoose.connect('mongodb://localhost:27017/mydatabase', {
  useNewUrlParser: true,
  useUnifiedTopology: true
});
app.listen(PORT, () => {
console.log(`Server is running on port ${PORT}`);
})
```

Define your MongoDB schema and model:

- Define a schema for your data (e.g., a simple todo list item).
- Create a model using the schema.

```
const todoSchema = new mongoose.Schema({
  title: String,
  completed: Boolean
});
const Todo = mongoose.model('Todo', todoSchema);
```

Create routes for CRUD operations:

- Define routes for creating, reading, updating, and deleting todos.

```
// Create a new todo
app.post('/todos', async (req, res) => {
  const newTodo = new Todo({
    title: req.body.title,
    completed: req.body.completed || false
  });
  await newTodo.save();
  res.status(201).json(newTodo);
});

// Get all todos
app.get('/todos', async (req, res) => {
  const todos = await Todo.find();
  res.json(todos);
});
```

```
// Update a todo
app.put('/todos/:id', async (req, res) => {
  const updatedTodo = await Todo.findByIdAndUpdate(req.params.id, req.body, { new: true });
  res.json(updatedTodo);
});

// Delete a todo
app.delete('/todos/:id', async (req, res) => {
  const deletedTodo = await Todo.findByIdAndDelete(req.params.id);
  res.json(deletedTodo);
});
```

Testing API:

- Use tools like Postman or curl to test your API endpoints.

Output:

When you start your Express.js server, you should see the following message indicating that the server is running:

Server is running on port 3000

when we create a new todo, we might receive a response like this:

when we create a new todo, we will receive a response like this:


```
{
  "_id": "611f4987b3c6dcb1f8101c95",
  "title": "Buy groceries",
  "completed": false,
  "__v": 0
}
```


And when we retrieve all todos, we will receive a response like this:


```
[
  {
    "_id": "611f4987b3c6dcb1f8101c95",
    "title": "Buy groceries",
    "completed": false,
    "__v": 0
  },
  {
    "_id": "611f49a4b3c6dcb1f8101c96",
    "title": "Walk the dog",
    "completed": true,
    "__v": 0
  }
]
```