**Assignment 4: RESTful API Basics**

**Create basic CRUD routes (GET, POST, PUT, DELETE) in Express.**

**Implement simple data models for your MongoDB collections.**

Below is an example of creating basic CRUD (Create, Read, Update, Delete) routes in Express for a simple RESTful API using MongoDB. We'll also implement simple data models for MongoDB collections.

Step 1: Install Required Packages

Make sure you have the required packages installed:

npm install express mongoose body-parser

Step 2: Create the Express.js Application

Create a file named app.js (or any other preferred name) and add the following code:

| const express = require('express');  const mongoose = require('mongoose');  const bodyParser = require('body-parser');  const app = express();  const port = 3000;  // Connect to MongoDB  mongoose.connect('mongodb://localhost:27017/todo-app', { useNewUrlParser: true, useUnifiedTopology: true });  // Check MongoDB connection  const db = mongoose.connection;  db.on('error', console.error.bind(console, 'MongoDB connection error:'));  db.once('open', () => {  console.log('Connected to MongoDB');  });  // Use body-parser middleware for parsing JSON  app.use(bodyParser.json());  // Define a simple data model for a ToDo item  const todoSchema = new mongoose.Schema({  title: { type: String, required: true },  description: { type: String },  completed: { type: Boolean, default: false },  });  const Todo = mongoose.model('Todo', todoSchema);  // Define CRUD routes  // GET all todos  app.get('/todos', async (req, res) => {  try {  const todos = await Todo.find();  res.json(todos);  } catch (error) {  res.status(500).json({ message: error.message });  }  });  // GET a specific todo by ID  app.get('/todos/:id', async (req, res) => {  try {  const todo = await Todo.findById(req.params.id);  if (todo) {  res.json(todo);  } else {  res.status(404).json({ message: 'Todo not found' });  }  } catch (error) {  res.status(500).json({ message: error.message });  }  });  // POST a new todo  app.post('/todos', async (req, res) => {  const todo = new Todo({  title: req.body.title,  description: req.body.description,  completed: req.body.completed || false,  });  try {  const newTodo = await todo.save();  res.status(201).json(newTodo);  } catch (error) {  res.status(400).json({ message: error.message });  }  });  // PUT/update a specific todo by ID  app.put('/todos/:id', async (req, res) => {  try {  const updatedTodo = await Todo.findByIdAndUpdate(  req.params.id,  {  title: req.body.title,  description: req.body.description,  completed: req.body.completed || false,  },  { new: true }  );  if (updatedTodo) {  res.json(updatedTodo);  } else {  res.status(404).json({ message: 'Todo not found' });  }  } catch (error) {  res.status(400).json({ message: error.message });  }  });  // DELETE a specific todo by ID  app.delete('/todos/:id', async (req, res) => {  try {  const deletedTodo = await Todo.findByIdAndDelete(req.params.id);  if (deletedTodo) {  res.json({ message: 'Todo deleted successfully' });  } else {  res.status(404).json({ message: 'Todo not found' });  }  } catch (error) {  res.status(500).json({ message: error.message });  }  });  // Start the server  app.listen(port, () => {  console.log(`Server listening at http://localhost:${port}`);  }); |
| --- |

Step 3: Run the Application

Start your Express.js server:

node app.js

Step 4: Test the CRUD Operations

Use tools like Postman or any REST API client to test the CRUD operations:

GET all todos: GET http://localhost:3000/todos

GET a specific todo: GET http://localhost:3000/todos/:id

POST a new todo: POST http://localhost:3000/todos

Body: { "title": "Task 1", "description": "Description for Task 1", "completed": false }

PUT/update a specific todo: PUT http://localhost:3000/todos/:id

Body: { "title": "Updated Task 1", "description": "Updated Description", "completed": true }

DELETE a specific todo: DELETE http://localhost:3000/todos/:id

Ensure you replace :id with the actual ID of a todo when testing the "GET by ID," "PUT," and "DELETE" operations.

Congratulations! You have created a basic Express.js application with CRUD routes and connected it to a MongoDB database.