**Assignment 8: Express.js CRUD Operations**

**Integrate CRUD operations with Express.js routes.**

**Test your Express.js routes for CRUD functionality**

Below is an updated version of your app.js file, integrating CRUD operations with Express.js routes. This will include routes for creating, reading, updating, and deleting todos. We'll use the express and mongoose packages for this integration.

Step 1: Install Required Packages

Make sure you have the necessary packages installed:

npm install express mongoose body-parser

Step 2: Update app.js

| 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);  // CRUD routes  // Create 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 });  }  });  // Read 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 });  }  });  // Read 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 });  }  });  // 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: Test CRUD Functionality

Use Postman or any API testing tool to test your CRUD functionality.

Create a new todo (POST):

Send a POST request to http://localhost:3000/todos with JSON data in the request body.

Read all todos (GET):

Send a GET request to http://localhost:3000/todos.

Read a specific todo by ID (GET):

Send a GET request to http://localhost:3000/todos/:id where :id is the actual ID of a todo.

Update a specific todo by ID (PUT):

Send a PUT request to http://localhost:3000/todos/:id with updated JSON data in the request body.

Delete a specific todo by ID (DELETE):

Send a DELETE request to http://localhost:3000/todos/:id where :id is the actual ID of a todo.

Verify that the responses and changes in the database match the expected behavior.

Congratulations! You have integrated CRUD operations with Express.js routes and tested the functionality using Postman or a similar tool.