Assignment.

Assignment Instructions: Building CRUD Operations with Express and MongoDB

Objective: Implement CRUD operations using Express.js and MongoDB with Mongoose. Focus on organizing project files into a structured folder hierarchy.

Project Folder Structure:

1. Create a new project directory named **express-mongoose-crud**.
2. Inside the project directory, create the following folders and files:
   - **models/** (for Mongoose models)
   - **controllers/** (to handle HTTP requests)
   - **routes/** (for defining routes)
   - **app.js** (main file to define Express application)
   - **.env** (for storing environment variables)

Step-by-Step Instructions:

1. Setting up the Project:

1. Navigate to the project directory in your terminal.
2. Initialize a new Node.js project using **npm init -y**.
3. Install necessary packages: **npm install express mongoose dotenv nodemon body-parser**.

2. Define the Mongoose Models:

1. Inside the **models/** folder, create Mongoose models for your application entities (e.g., User, Post, Comment.).

   - UserSchema
     I. name
     II. email
   - PostSchema
     I. title
     II. content
     III. ref to user (which user has posted)
   - CommentSchema
     I. Text
     II. User ref to user (which user has commented)
     III. Post ref to post (on which post user commented)

2. Define the schema for each model using Mongoose schema syntax.

3. Create Controllers:

1. In the **controllers/** folder, create controller files corresponding to each entity or resource (e.g., **userController.js**, **productController.js**, etc.).

2. Implement functions in each controller to handle CRUD operations (Create, Read, Update, Delete).

4. Define Routes:

1. Inside the **routes/** folder, create route files (e.g., **userRoutes.js**, **productRoutes.js**, etc.).

2. Define Express router objects in each route file and specify the endpoints for CRUD operations.
3. Map each route to the appropriate controller function.

5. Configure Express Application:

1. In **app.js**, require necessary packages (**express**, **mongoose**, **dotenv**, etc.).
2. Configure environment variables using **dotenv** by loading **.env** file.
3. Connect to MongoDB database using Mongoose and the provided database URL from **.env**.
4. Use **express.json()** middleware to parse incoming JSON requests.
5. Mount the route files to the Express application using **app.use()**.

6. Implement CRUD Operations:

1. Inside the controller files, implement functions to handle CRUD operations using Mongoose methods (**create()**, **find()**, **findOne()**, **updateOne()**, **deleteOne()**, etc.).
2. Ensure error handling for database operations and return appropriate responses.

7. Testing:

1. Start the server using **nodemon app.js** or **npm start**.
2. Use tools like Postman to send HTTP requests to test CRUD operations (POST, GET, PUT, DELETE).
3. Verify that the operations are correctly executed in the MongoDB database.

Submission Guidelines:

1. Organize your project files as per the specified folder structure.
2. Include comments in your code to explain the functionality of each file and function.
3. Ensure that your code follows best practices and conventions.
4. Provide a README.md file with instructions on how to run the project and test the CRUD operations.
5. Submit your project folder as a compressed file (e.g., .zip) or provide a link to your code repository (e.g., GitHub).

Additional Notes:

- Research and refer to Mongoose and Express documentation for guidance on schema definition, routing, and database operations.
- Feel free to reach out for assistance or clarification on any aspect of the assignment.
- Aim for a clean and modular code structure to enhance readability and maintainability.