Welcome to the challenge! We hope you're excited to showcase your skills. Don't hesitate to unleash your creativity!

Task 1: To-Do List CRUD API

1. Set Up Project:
   - Initialise a new Node.js project using npm or yarn.
   - Install necessary dependencies (e.g., express, body-parser).

2. To-Do List API:
Create RESTful API endpoints for basic to-do list operations:
   - GET /api/todos: Retrieve a list of all to-do items.
   - GET /api/todos/:id: Retrieve a specific to-do item by ID.
   - POST /api/todos: Create a new to-do item.
   - PUT /api/todos/:id: Update an existing to-do item (mark as done or change the content).
   - DELETE /api/todos/:id: Delete a to-do item.

3. To-Do Item Schema:
Define schema for to-do items. Should have at least the following properties:
   - content: Content of the to-do item.
   - done: Boolean indicating whether the item is done.

4. Validation and Error Handling:
   - Implement basic validation for creating and updating to-do items.
   - Provide meaningful error messages for validation failures or other issues.

4. Submission:
   - Provide your project code, either GitHub repo or simply a zip file.

Task 2: Add Authentication and Authorization

1. Copy Code from Task 1:
    - Create a copy of the code from Task 1. This will serve as the starting point for Task 2.

2. User Registration API:
    - Create a `POST` route for user registration, e.g., `/api/register`.
    - Accept data containing `name`, `email`, `password`, and a `profile` (profile image).
    - Validate the input data.
    - [Optional] Hash the provided password using bcrypt to enhance security.
    - Utilize the multer middleware to handle file uploads and store the uploaded profile image on the server's file system.
    - Return API response with the registered user's information along with an authentication token upon successful registration with appropriate status code.

3. User Login API:
    - Create a `POST` route for user login, e.g., `/api/login`.
    - Accept data containing `email` and `password`.
    - Check if credentials are correct.
    - Return API response with the logged in user's information along with an authentication token upon successful registration with appropriate status code.

4. Create Authorization Middleware
    - Implement middleware to verify JWT tokens for protected routes.
    - Modify the existing to-do API from Task 1, to ensure that users can only view, update, and delete their own to-do items.

4. Submission:
    - Provide your project code, either GitHub repo or simply a zip file.

Good luck with the challenge! If you have any questions, feel free to reach out. Happy coding!