Problem Statement for Task Management App with MongoDB and Mongoose

Objective: Create an Express server to manage and serve task data using MongoDB as the database and Mongoose as the ODM (Object Data Modeling) library. The server should provide APIs to list all tasks, retrieve a task based on its ID, add new tasks, update the status or content of a task, and delete tasks based on their ID.

Requirements:

1. **Express Server Setup:**
    - Set up an Express server to handle incoming HTTP requests.
2. **MongoDB Database Setup:**
    - Set up a MongoDB database to store the task data.
    - Use Mongoose to define the schema and model for the tasks.
3. **Task Data Model:**
    - Define a Mongoose schema for tasks with the following fields:
        - `title` (string, required): The title of the task.
        - `description` (string, required): A brief description of the task.
        - `status` (string, default: "pending"): The status of the task, which could be "pending", "in-progress", or "completed".
        - `createdAt` (Date): Timestamp of when the task was created.
        - `updatedAt` (Date): Timestamp of when the task was last updated.
    - Create a Mongoose model based on the schema.
4. **API Endpoints:**
    - **GET /tasks**: Retrieve a list of all tasks.
        - **Implementation:** Use the Mongoose `find` method to retrieve all tasks from the database.
    - **GET /tasks/:id**: Retrieve a task based on its unique ID.
        - **Implementation:** Use the Mongoose `findById` method to retrieve a specific task by its ID.
    - **POST /tasks**: Add a new task to the database. The request body should include the `title` and `description`.
        - **Implementation:** Use the Mongoose `create` method to add a new task to the database.
    - **PUT /tasks/:id**: Update the content or status of a specific task based on its ID. The new data will be provided in the request body.
        - **Implementation:** Use the Mongoose `findByIdAndUpdate` method to update the task's content and status
    - **DELETE /tasks/:id**: Delete a task based on its ID.
        - **Implementation:** Use the Mongoose `findByIdAndDelete` method to delete a task by its ID.

5. **Error Handling:**
    - Implement appropriate error handling for cases such as requesting a non-existent task ID or attempting to update/delete a non-existent task.
    - Return appropriate HTTP status codes (e.g., 404 for not found, 400 for bad requests).
6. **Testing:**
    - Ensure the server and API endpoints are tested to confirm they return the expected data.
    - Test CRUD operations on the tasks stored in MongoDB.
7. **Database Connection:**
    - Ensure the Express server connects to the MongoDB database using Mongoose.
    - Handle scenarios where the database connection fails, and ensure the server does not start if the connection is not established.

Assumptions:

- The MongoDB server is running locally or a remote MongoDB URI is provided.
- Mongoose will be used for interacting with the MongoDB database.
- The server will run locally.