Program 1:

This program involves creating a basic RESTful API to manage a collection of books, including basic CRUD operations. This will demonstrate knowledge of Node.js, Express, and basic REST principles.

**Project: Simple Book Management API**

**Requirements**

1. **Endpoints:**
   * POST /books: Create a new book.
   * GET /books: Get all books.
   * GET /books/:id: Get a specific book by ID.
   * PUT /books/:id: Update a book by ID.
   * DELETE /books/:id: Delete a book by ID.
2. **Book Model:**
   * id: Unique identifier for the book.
   * title: Title of the book.
   * author: Author of the book.
   * publishedYear: Year the book was published.
3. **Validation:**
   * Ensure that title and author are provided when creating or updating a book.
4. **Persistence:**
   * Use mongoDB data store for simplicity (for a real-world application, a database like MongoDB would be used).

**Program 2**

**To-Do List Management API**

You are required to develop a RESTful API for a simple to-do list application. The API should allow users to create, read, update, and delete to-do items. Additionally, users should be able to mark to-do items as completed or not completed. The application will use MongoDB for data storage.

**Requirements**

1. **Endpoints:**
   * POST /todos: Create a new to-do item.
   * GET /todos: Get all to-do items.
   * GET /todos/:id: Get a specific to-do item by ID.
   * PUT /todos/:id: Update a to-do item by ID.
   * PATCH /todos/:id/complete: Mark a to-do item as completed.
   * DELETE /todos/:id: Delete a to-do item by ID.
2. **To-Do Model:**
   * id: Unique identifier for the to-do item.
   * title: Title of the to-do item (required).
   * description: Description of the to-do item (optional).
   * completed: Boolean indicating if the to-do item is completed (default: false).
   * createdAt: Timestamp of when the to-do item was created.
   * updatedAt: Timestamp of when the to-do item was last updated.
3. **Validation:**
   * Ensure that title is provided when creating or updating a to-do item.
4. **Persistence:**
   * Use MongoDB for data storage.
5. **Best Practices:**
   * Use environment variables for configuration.
   * Implement error handling and validation.
   * Structure the project in a modular way.