Todo API Requirements Document

Overview

This document outlines the requirements for developing a Todo API using Express.js and MongoDB with Mongoose. The API will provide end-to-end CRUD operations, tag functionality, filtering by tags, and more.

Objectives

1. Develop a RESTful API using Express.js.
2. Integrate MongoDB as the database with Mongoose as the ODM.
3. Implement CRUD operations for todos.
4. Add functionality to tag todos.
5. Allow filtering todos by tags.
6. Ensure proper error handling and validation.
7. Include appropriate unit and integration tests.

API Endpoints

1. Todo CRUD Operations

Create Todo

- **Endpoint**: `POST /api/v1/todos`
- **Description**: Create a new todo item.
- **Request Body**:

```
{
  "title": "string",
  "description": "string",
  "dueDate": "date",
  "tags": ["string"]
}
```

- **Responses**:
  - `201 Created` : Returns the created todo item.

- `400 Bad Request` : If the request body is invalid.

Get All Todos

- **Endpoint**: `GET /api/v1/todos`
- **Description**: Retrieve all todo items.
- **Responses**:
  - `200 OK` : Returns an array of todos.
  - `500 Internal Server Error` : If there is a server error.

Get Todo by ID

- **Endpoint**: `GET /api/v1/todos/:id`
- **Description**: Retrieve a single todo item by its ID.
- **Responses**:
  - `200 OK` : Returns the todo item.
  - `404 Not Found` : If the todo item is not found.
  - `500 Internal Server Error` : If there is a server error.

Update Todo

- **Endpoint**: `PUT /api/v1/todos/:id`
- **Description**: Update a todo item by its ID.
- **Request Body**:

```
{
  "title": "string",
  "description": "string",
  "dueDate": "date",
  "tags": ["string"]
}
```

- **Responses**:
  - `200 OK` : Returns the updated todo item.
  - `400 Bad Request` : If the request body is invalid.
  - `404 Not Found` : If the todo item is not found.
  - `500 Internal Server Error` : If there is a server error.

Delete Todo

- **Endpoint**: `DELETE /api/v1/todos/:id`
- **Description**: Delete a todo item by its ID.
- **Responses**:

- `204 No Content` : If the deletion is successful.
- `404 Not Found` : If the todo item is not found.
- `500 Internal Server Error` : If there is a server error.

2. Tag Functionality

Add Tag to Todo

- **Endpoint**: `POST /api/v1/todos/:id/tags`
- **Description**: Add a tag to a todo item.
- **Request Body**:

```
{
  "tag": "string"
}
```

- **Responses**:
  - `200 OK` : Returns the updated todo item with the new tag.
  - `400 Bad Request` : If the request body is invalid.
  - `404 Not Found` : If the todo item is not found.
  - `500 Internal Server Error` : If there is a server error.

Remove Tag from Todo

- **Endpoint**: `DELETE /api/v1/todos/:id/tags`
- **Description**: Remove a tag from a todo item.
- **Request Body**:

```
{
  "tag": "string"
}
```

- **Responses**:
  - `200 OK` : Returns the updated todo item without the removed tag.
  - `400 Bad Request` : If the request body is invalid.
  - `404 Not Found` : If the todo item is not found.
  - `500 Internal Server Error` : If there is a server error.

3. Filtering by Tags

Get Todos by Tag

- **Endpoint**: `GET /api/v1/todos?tag=tagname`
- **Description**: Retrieve todos filtered by a specific tag.
- **Responses**:
  - `200 OK` : Returns an array of todos with the specified tag.
  - `500 Internal Server Error` : If there is a server error.

Data Model

Todo

```
{
  "id": "ObjectId",
  "title": "string",
  "description": "string",
  "dueDate": "date",
  "tags": ["string"],
  "createdAt": "date",
  "updatedAt": "date"
}
```

Validation

- Title: Required, string
- Description: Optional, string
- Due Date: Optional, date
- Tags: Optional, array of strings

Error Handling

- Ensure proper validation of request bodies.
- Return meaningful error messages.
- Handle common HTTP errors such as 400, 404, and 500.

Testing

- Unit tests for individual functions.

- Integration tests for API endpoints.
- Use testing frameworks like Mocha, Chai, and Supertest.

Tools and Technologies

- **Backend**: Node.js, Express.js
- **Database**: MongoDB, Mongoose
- **Testing**: Mocha, Chai, Supertest
- **Validation**: Joi or express-validator

Project Structure

```
├── src
|   ├── controllers
|   |   └── todoController.js
|   ├── repositories
|   |   └── todoRepository.js
|   ├── models
|   |   └── todoModel.js
|   ├── routes
|   |   └── todoRoutes.js
|   ├── services
|   |   └── todoService.js
|   └── app.js
└── package.json
```

Conclusion

This requirements document serves as a comprehensive guide to developing a Todo API with Express.js and MongoDB using Mongoose. It covers the necessary CRUD operations, tagging functionality, filtering by tags, data modeling, error handling, and testing strategies.