**TODO API Documentation**

**ProjectOverView:**

1. TODO application is designed to manage TODO items with CRUD operations, supporting functionalities for creating, reading, updating, and deleting tasks.
2. **Technologies Used**: .NET 6 or .NET 8, SQL Server for the database, and JWT for authentication.

#### Features

* **CRUD Operations**:
  + Create, Read, Update, and Delete operations for TODO items.
* **Data Fields**: Each TODO item consists of the following fields:
  + id: Unique identifier for the TODO item.
  + title: Title of the TODO item.
  + description: Description of the TODO item.
  + status: Indicates whether the TODO item is completed or not.

**3. API Endpoints**

* **POST /api/TodoListAPI/CreateTodoItem**: Create a new TODO item.
* **GET /** **api/TodoListAPI/GetAllTodos**: Retrieve all TODO items.
* **GET /GetTodoById/{id}**: Retrieve a single TODO item by its ID.
* **PUT /UpdateTodoItem/{id}**: Update a TODO item by its ID.
* **DELETE /DeleteTodoItem/{id}**: Delete a TODO item by its ID.

#### 4. HTTP Methods and Status Codes

* **HTTP Methods**: Utilized appropriate HTTP methods for each operation (GET, POST, PUT, DELETE).
* **Status Codes**: Implemented relevant HTTP status codes for success and failure scenarios (e.g., 200 OK, 201 Created, 400 Bad Request, 404 Not Found, 500 Internal Server Error).

#### 5. Error Handling and Validation

* Implemented basic error handling to manage exceptions during API calls.
* Input validation ensures that required fields are to expected formats.

#### 6. Authentication and Authorization

* Implemented JWT authentication for secure access to the API endpoints.
* Authorization checks are in place for protected endpoints to ensure only authorized users can create or modify TODO items.

#### 7. Abstraction and Interface Implementation

* Used abstraction to define interfaces for data access and business logic, promoting a clean architecture.
* Created an injection class for dependency injection to manage dependencies effectively.

#### 8. Database Integration

* Integrated with SQL Server for persistent storage of TODO items.
* Provided a backup file of the database to facilitate easy recovery.

#### 9. Helper Classes

* Added helper classes for common operations (e.g., token generation, error response handling) to streamline code and enhance readability.

### **Conclusion**

This documentation provides a comprehensive overview of the TODO application API. It serves as a reference for developers and stakeholders to understand the functionality, design decisions, and implementation details.