**ToDo App API Documentation**

**Introduction**

This document provides a detailed guide to the ToDo App API, outlining its endpoints, data models, database setup using Entity Framework Core, implementation of the repository pattern, and testing.

**1. Data Models**

**Task Model**

The Task model represents a task entity within the application. It includes properties such as Id for unique identification, Title for the task name, Description for additional details, and Completed to track the task's status (true for completed, false for incomplete).

**2. Database Setup**

**Entity Framework Core**

Entity Framework Core is utilized for managing data persistence. It provides a flexible and powerful ORM (Object-Relational Mapping) framework for interacting with databases in ASP.NET applications.

**DbContext**

The ToDoDbContext class extends DbContext from Entity Framework Core. It serves as the main entry point for the application's interactions with the database. It includes a DbSet<Task> property to represent the collection of Task entities stored in the database.

**3. Repository Pattern**

**Task Repository Interface**

The ITaskRepository interface defines the contract for managing tasks in the application. It includes methods such as GetTaskByIdAsync, GetAllTasksAsync, AddTaskAsync, UpdateTaskAsync, and DeleteTaskAsync for CRUD (Create, Read, Update, Delete) operations on tasks.

**Task Repository Implementation**

The TaskRepository class implements the ITaskRepository interface. It encapsulates the logic for interacting with the ToDoDbContext to perform database operations related to tasks, such as fetching tasks, adding new tasks, updating existing tasks, and deleting tasks.

**4. Controllers**

**TasksController**

The TasksController class is an ASP.NET Web API controller responsible for handling HTTP requests related to tasks. It exposes endpoints (GET, POST, PUT, DELETE) that clients can interact with to manage tasks in the system.

**5. Testing**

**Unit Testing**

Unit testing ensures the functionality and reliability of the API endpoints. Tests are written using frameworks like xUnit and Moq to simulate requests to the API endpoints and verify expected behaviors and outcomes.

**Conclusion**

This documentation provides a comprehensive overview of the ToDo App API, detailing the technologies and components used at each stage, from data modeling to testing. It aims to facilitate understanding and maintenance of the application's architecture and functionality.