Todo Laravel API Project – Documentation

1. Project Overview

This project is a minimal **Todo app backend** built using **Laravel** and **Sanctum** for token-based authentication. The project is containerized with **Docker**, uses **MySQL** as the database, and is served via **Nginx**.

Key features:

- User registration and login
- Token-based (Bearer) authentication
- CRUD operations on Todo items
- Dockerized setup and deployment
- Ready-to-use API for frontend or testing via `curl`

2. Folder & File Structure

```
app/                  # Laravel application
├─ app/Models/Item.php
├─ app/Http/Controllers/AuthController.php
├─ app/Http/Controllers/UserController.php
├─ app/Http/Controllers/TodoController.php
├─ ...Laravel standard structure
├─ docker-compose.yml    # Docker services
├─ Dockerfile            # PHP-FPM Laravel container
├─ nginx/
├─ └─ default.conf       # Nginx config
├─ run.sh                # Install Laravel + dependencies
└─ setup.sh              # Build and run Docker + migrations
```

3. Database Schema

3.1 Users Table

Laravel default users table with additional `username` column:

Column	Type	Notes
id	BIGINT	Primary key
username	VARCHAR	Unique
name	VARCHAR	Fallback for Laravel
email	VARCHAR	Required for Laravel
password	VARCHAR	Hashed
created_at	TIMESTAMP	Auto
updated_at	TIMESTAMP	Auto

4. API Documentation

All API endpoints are prefixed with `/api/` and use **JSON** for request/response. All endpoints except `register` and `login` require **Bearer token authentication**.

4.1 Register User

- **Endpoint:** `POST /api/register`
- **Description:** Create a new user account.
- **Request Body (JSON):**

```
{
  "username": "alice",
  "password": "pass123"
}
```

- **Response (JSON):**

4.2 Login User

- **Endpoint:** `POST /api/login`
- **Description:** Authenticate a user and issue a Bearer token.
- **Request Body (JSON):**

```
{
  "username": "alice",
  "password": "pass123"
}
```

- **Response (JSON):**

```
{
  "ok": true,
  "token": "eyJpdiI6..."
}
```

- **Notes:** Save this token in the client and include it in the `Authorization` header for

4.3 Logout User

- **Endpoint:** `POST /api/logout`
- **Authentication:** Bearer token required
- **Description:** Revoke the current API token.
- **Request Headers:**

```
Authorization: Bearer <token>
```

- **Response (JSON):**

```
{
  "ok": true
}
```

- **Notes:** After logout, the token is invalid.

4.4 Get Current User

- **Endpoint:** `GET /api/user`
- **Authentication:** Bearer token required
- **Response (JSON):**

```
{
  "user": {
    "id": 1,
    "username": "alice",
    "created_at": "2025-12-03T17:00:00.000000Z"
  }
}
```

4.5 List Todos

- **Endpoint:** `GET /api/todos`
- **Authentication:** Bearer token required
- **Response (JSON):**

```
{
  "items": [
    {
      "id": 1,
      "title": "Buy milk",
      "completed": false,
      "created_at": "2025-12-03T17:15:00.000000Z"
    }
  ]
}
```

4.6 Create Todo

- **Endpoint:** `POST /api/todos`
- **Authentication:** Bearer token required
- **Request Body (JSON):**

```
{
  "title": "Buy milk"
}
```

- **Response (JSON):**

```
{
  "ok": true,
  "id": 1
}
```

4.7 Update Todo

- **Endpoint:** `PUT /api/todo?id={todo_id}`
- **Authentication:** Bearer token required
- **Request Body (JSON):** Optional fields

```
{
  "title": "Buy almond milk",
  "completed": true
}
```

- **Response (JSON):**

```
{
  "ok": true,
  "updated": 1
}
```

- **Notes:** Only fields provided in request are updated.

4.8 Delete Todo

- **Endpoint:** `DELETE /api/todo?id={todo_id}`
- **Authentication:** Bearer token required
- **Response (JSON):**

```
{
  "ok": true,
  "deleted": 1
}
```

- **Notes:** Deletes the specified todo if it belongs to the authenticated user.

5. Authentication Flow

1. **Register user** → create account
2. **Login user** → receive Bearer token
3. **Use token** in `Authorization` header for all protected endpoints
4. **Logout** → token revoked

Header example:

```
Authorization: Bearer <token>
```

6. Example `curl` Usage

```
# Register
curl -X POST http://localhost:8080/api/register -H "Content-Type: application/json" -d '{"username":"alice","password":"pass123"}'

# Login
curl -X POST http://localhost:8080/api/login -H "Content-Type: application/json" -d '{"username":"alice","password":"pass123"}'

# Set token variable
TOK=<token_from_login>

# Get current user
curl -X GET http://localhost:8080/api/user -H "Authorization: Bearer $TOK"

# List todos
curl -X GET http://localhost:8080/api/todos -H "Authorization: Bearer $TOK"

# Create todo
curl -X POST http://localhost:8080/api/todos -H "Authorization: Bearer $TOK" -H "Content-Type: application/json" -d '{"title":"Buy milk"}'

# Update todo
curl -X PUT "http://localhost:8080/api/todo?id=1" -H "Authorization: Bearer $TOK" -H "Content-Type: application/json" -d '{"completed":true}'

# Delete todo
curl -X DELETE "http://localhost:8080/api/todo?id=1" -H "Authorization: Bearer $TOK"

# Logout
curl -X POST http://localhost:8080/api/logout -H "Authorization: Bearer $TOK"
```

7. Docker & Deployment

7.1 Build and run containers

```
./setup.sh
```

This will:

1. Build PHP-FPM Laravel container
2. Build Nginx container
3. Build MySQL container
4. Run Laravel migrations

7.2 Access

- **API & frontend:** `http://localhost:8080`
- **MySQL inside container:** `db:3306`

8. Notes

- Tokens expire only if you implement expiry in Sanctum (default: none). Logout revokes token immediately.
- Laravel Eloquent ORM handles all database queries.
- Minimal Docker + Nginx setup, easy to extend.
- Compatible with curl, Postman, or browser JS frontend.