A logo with a light bulb and various tools

Description automatically generatedA logo for a company

Description automatically generated

# Programmer Manual

Version 1.0

17/04/2024

Table of Contents

[1. Introduction 3](#_Toc164429922)

[System Architecture Overview 3](#_Toc164429923)

[Technology Stack 3](#_Toc164429924)

[Environment Setup 3](#_Toc164429925)

[2. Key Functions Documentation 4](#_Toc164429926)

[Login Function 4](#_Toc164429927)

[Sign Up Function 4](#_Toc164429928)

[Fetch and Render Categories 5](#_Toc164429929)

[Create Post 6](#_Toc164429930)

[Show Post 7](#_Toc164429931)

[Post Comment 8](#_Toc164429932)

[3. Backend Architecture and Features 10](#_Toc164429933)

[Data Transfer Objects (DTOs) 10](#_Toc164429934)

[JWT for Authentication 10](#_Toc164429935)

[Controllers and Repositories 10](#_Toc164429936)

[Authentication Guard 10](#_Toc164429937)

[Security Considerations 10](#_Toc164429938)

[4. Backend Code Examples 11](#_Toc164429939)

[Post Controller 11](#_Toc164429940)

[Post Service 12](#_Toc164429941)

[Post Repository (Extends Entity Repository) 13](#_Toc164429942)

[5. Sources 14](#_Toc164429943)

## Introduction

This guide is essential for developers working on the Maker Mingle project. It provides detailed documentation of the system's architecture, key functionalities, and integrations. The purpose is to ensure consistency in development practices and to aid new developers in quickly coming up to speed with the project's technical environment.

### System Architecture Overview

The system is structured around a React frontend and presumably a Nest.js backend, with Tailwind CSS for styling. The application is designed to be responsive and modular, facilitating maintenance and scalability.

#### Technology Stack

* Frontend: React, Tailwind CSS
* Backend: Nest.js
* Database: MongoDB
* Deployment: Vercel

#### Environment Setup

* Prerequisites: Nest.js, npm, Git
* Installation:

**git clone [repository-url]**

**cd [project-directory]**

**npm install**

**npm start**

* Environment Variables: Necessary for API endpoints, which should be stored in a .env file.

## Key Functions Documentation

### Login Function

Manages user authentication.

**A screen shot of a computer program

Description automatically generated**

### Sign Up Function

Handles new user registration.

A screen shot of a computer program

Description automatically generated

### Fetch and Render Categories

Used in the Explore page to display categories from the database.

A screen shot of a computer program

Description automatically generated

### Create Post

The form for creating a post is defined with fields for project name, description, materials, process, hashtags, and image upload.  
This function is bound to the form's submit event and manages data gathering, validation, and submission to the backend via the POST API.

A screen shot of a computer program

Description automatically generated

### Show Post

The "Show Post" functionality displays details about a specific DIY project. Here’s how it's structured:

* **Fetching Post Data:** When a user selects a post, the frontend fetches detailed information about the post from the backend using the post's ID.
* **Rendering Details:** The post details, including images, materials needed, and step-by-step instructions, are dynamically rendered on the page.

Example Interaction: The function renderPost would take care of rendering these details onto the page.

A computer screen shot of a program code

Description automatically generated



### Post Comment

The comments section allows users to interact with posts by adding comments.  
Like the "Create Post" function, submitting a comment would involve collecting the comment text from the textarea, validating it, and sending it to the backend using an API call.



## Backend Architecture and Features

The backend of Maker Mingle is designed to be robust and secure, facilitating efficient data handling and user interactions through a well-structured API.

### Data Transfer Objects (DTOs)

The backend utilizes DTOs to ensure that incoming requests have the correct data structure and content. If a request fails validation, the API responds with an appropriate error message, helping front-end developers understand what went wrong.

### JWT for Authentication

We use JSON Web Tokens (JWT) to manage user sessions. When a user registers, their password is hashed using a JWT-compatible library before being stored in the database, enhancing security by ensuring that actual passwords are never stored or transmitted in a readable form.

### Controllers and Repositories

* **Controllers:** Receive requests from the UI, use DTOs for validation, and pass valid requests to the service layer.
* **Services:** Contain business logic for handling requests. After processing, they interact with the repository layer if data persistence or retrieval is necessary.
* **Repositories:** Interact directly with the database, performing data queries and updates.

### Authentication Guard

An auth guard is implemented to check whether a user is logged in before allowing them to create posts or comments. This system helps in differentiating between authenticated and unauthenticated users, securing user interactions on the platform.

### Security Considerations

Input Validation: Using DTOs for input validation prevents common vulnerabilities such as SQL injection by ensuring only properly formatted data is processed.

## Backend Code Examples

### Post Controller

A screen shot of a computer program

Description automatically generated

### Post Service

A screen shot of a computer program

Description automatically generated

### Post Repository (Extends Entity Repository)

A screen shot of a computer code

Description automatically generated

A screen shot of a computer program

Description automatically generated

## Sources

**General Development and Documentation**

**React Documentation:** For comprehensive understanding and latest updates in React.

<https://legacy.reactjs.org/docs/getting-started.html>

**Tailwind CSS Documentation:** To explore more about utility classes and custom configurations.

<https://tailwindcss.com/docs/installation>

**GitHub:** For version control and code sharing with your team.

[https://github.com](https://github.com/)

**chatGPT:**  
<https://openai.com/chatgpt>

**API Development and Security**

**Express.js:** The standard server framework for Node.js.

[https://expressjs.com](https://expressjs.com/)

**JWT.io:** Learn about JSON Web Tokens for secure transmission.

[https://jwt.io](https://jwt.io/)

**Database Management**

**MongoDB:** Official documentation for using MongoDB, especially useful if you’re using a NoSQL database.

<https://www.mongodb.com/docs/>

**Mongoose:** Documentation for Mongoose, which simplifies MongoDB interactions.

<https://mongoosejs.com/docs/>

**Learning and Community Support**

**Stack Overflow:** A valuable resource for troubleshooting and community advice.

[https://stackoverflow.com](https://stackoverflow.com/)

**Reddit:** Subreddits like r/reactjs and r/webdev can be great for getting help and discussing ideas.

<https://www.reddit.com/r/reactjs/?rdt=48266>

<https://www.reddit.com/r/webdev/>

**Link to DB**

mongodb+srv: //guypar1997: [gp123@project-web.8kaignp.mongodb.net/](http://gp123@project-web.8kaignp.mongodb.net/)