Contents

[Prompt 1: Generate Angular Project (PMS.Ui) and Provide as ZIP 1](#_Toc207034558)

[Prompt 2: Generate .NET Core API Project (PMS.WebAPI) and Provide as ZIP 3](#_Toc207034559)

[Prompt 3: Set up GitHub Repository with PR Workflow for Angular 4](#_Toc207034560)

[Prompt 4: GitHub Actions CI for Angular 5](#_Toc207034561)

[Prompt 5: Set up GitHub Repository with PR Workflow for WEBAPI Project 5](#_Toc207034562)

[Prompt 6: Authentication & Authorization 6](#_Toc207034563)

[Prompt 7: Login & Registration flow 7](#_Toc207034564)

[Prompt 8: SMTP Integration 9](#_Toc207034565)

[Prompt 9: Angular hosting 9](#_Toc207034566)

[Prompt 10: WebAPI Hosting 9](#_Toc207034567)

# Prompt 1: Generate Angular Project (PMS.Ui) and Provide as ZIP

"I want you to generate a complete Angular 20+ project named PMS\_UI and give me the source code as a downloadable ZIP file. The project should include:

1. Angular CLI setup with:
   * Routing enabled
   * SCSS styles
   * Angular Material installed with a custom theme
   * Rxjs library
2. App structure:
   * App Module as main application module with default app component with startup index.html as main html page
   * Add one shared module to share dependencies
   * A simple AuthService for login with default code
   * Also make sure all component will have jest Unit testing file(spec file)
   * Make sure all module and component has their own default code
3. A ready-to-run ng serve setup with package.json configured
4. .gitignore, README.md, and default favicon/logo replaced with 'PMS\_UI' placeholder
5. Project initialized as a Git repo, with instructions to push to GitHub
6. (Optional) Include a GitHub Actions workflow file to run lint, test, and build steps on PRs
7. Also help me with the command to setup above environment such as angular and npm version

Once done, please package the entire PMS\_UI project as a ZIP file that I can download and push to GitHub."

**Prompt provided by angular site:**

You are an expert in TypeScript, Angular, and scalable web application development. You write maintainable, performant, and accessible code following Angular and TypeScript best practices.

## Project name

- I want you to generate a complete Angular 20+ project named PMS\_UI and give me the source code as a downloadable ZIP file.

- Define all dependencies/pre-requisite for this project in readme file

- Follow one best theme for style and all required things

## TypeScript Best Practices

- Use strict type checking

- Prefer type inference when the type is obvious

- Avoid the `any` type; use `unknown` when type is uncertain

## Angular Best Practices

- Always use standalone components over NgModules

- Must NOT set `standalone: true` inside Angular decorators. It's the default.

- Use signals for state management

- Implement lazy loading for feature routes

- Do NOT use the `@HostBinding` and `@HostListener` decorators. Put host bindings inside the `host` object of the `@Component` or `@Directive` decorator instead

- Use `NgOptimizedImage` for all static images.

- `NgOptimizedImage` does not work for inline base64 images.

## Components

- Keep components small and focused on a single responsibility

- Use `input()` and `output()` functions instead of decorators

- Use `computed()` for derived state

- Set `changeDetection: ChangeDetectionStrategy.OnPush` in `@Component` decorator

- Prefer inline templates for small components

- Prefer Reactive forms instead of Template-driven ones

- Do NOT use `ngClass`, use `class` bindings instead

- DO NOT use `ngStyle`, use `style` bindings instead

## State Management

- Use signals for local component state

- Use `computed()` for derived state

- Keep state transformations pure and predictable

- Do NOT use `mutate` on signals, use `update` or `set` instead

## Templates

- Keep templates simple and avoid complex logic

- Use native control flow (`@if`, `@for`, `@switch`) instead of `\*ngIf`, `\*ngFor`, `\*ngSwitch`

- Use the async pipe to handle observables

## Services

- Design services around a single responsibility

- Use the `providedIn: 'root'` option for singleton services

- Use the `inject()` function instead of constructor injection

## Common pitfalls

- Control flow (`@if`):

- You cannot use `as` expressions in `@else if (...)`. E.g. invalid code: `@else if (bla(); as x)`.

## Unit testing

- Use jest for unit test

- Add spec file against each component/services

# Prompt 2: Generate .NET Core API Project (PMS.WebAPI) and Provide as ZIP

"I want you to generate a .NET Core Web API project named PMS\_WEBAPI and provide the entire source code as a downloadable ZIP file. The project should include:

1. .NET 8 Web API setup with:
   * A PatientsController with GET, POST, and sample data
   * Models and Services folder with a simple in-memory PatientService
   * Swagger enabled
   * CORS enabled for localhost:4200 (for Angular frontend).gitignore for .NET
2. README.md with how to run the API locally
3. Include xunit testing project as well
4. Initialized as a Git repo with instructions to push to GitHub
5. Include GitHub Actions YAML to run dotnet build and test on push

Please give me the full **PMS.WebAPI** project packaged as a ZIP file that I can download and use immediately

# Prompt 3: Set up GitHub Repository with PR Workflow for Angular

"I'm creating a GitHub repository for my Angular project and want to follow a professional workflow. Please guide me through:

1. Initializing a GitHub repo from a local Angular CLI project
2. Creating .gitignore, README.md, and basic repo structure
3. Best practices for branches (e.g. main, develop, feature/\*)
4. Creating pull requests for code changes and enabling code reviews
5. Protecting the main branch (require PR review and passing CI checks)
6. Automating testing and linting with GitHub Actions
7. Optionally setting up CODEOWNERS to auto-assign reviewers
8. Enabling squash merges with PR titles as commit messages

Help me with exact steps and file examples where needed

# Prompt 4: GitHub Actions CI for Angular

"I want to set up a GitHub Actions pipeline for my Angular project. Please help me:

* Write a .github/workflows/ci.yml file
* Steps:
  + Checkout code
  + Set up Node.js (v18)
  + Run npm ci
  + Lint the project
  + Run unit tests (with ChromeHeadless)
  + Build the Angular app
* Trigger the pipeline on pull requests to main
* Use the best practices for Angular + GitHub Actions
* Show how to connect this pipeline to PR checks so it blocks merging if tests fail

# Prompt 5: Set up GitHub Repository with PR Workflow for WEBAPI Project

"I'm creating a GitHub repository for my WEBAPI project and want to follow a professional workflow. Please guide me through:

1. Initializing a GitHub repo from a local WEBAPI project
2. Creating .gitignore, README.md, and basic repo structure
3. Best practices for branches (e.g. main, develop, feature/\*)
4. Creating pull requests for code changes and enabling code reviews
5. Protecting the main branch (require PR review and passing CI checks)
6. Automating testing and linting with GitHub Actions
7. Optionally setting up CODEOWNERS to auto-assign reviewers
8. Enabling squash merges with PR titles as commit messages

Help me with exact steps and file examples where needed

# Prompt 6: Authentication & Authorization

I am building a full-stack web application with the following stack:

\* \*\*Frontend\*\*: Angular

\* \*\*Backend\*\*: .NET Core (latest stable version)

\* \*\*Database\*\*: Supabase (PostgreSQL)

I want to implement \*\*token-based authentication (JWT)\*\* from scratch, with these requirements:

1. \*\*Frontend (Angular)\*\*:

\* User login and registration forms.

\* Use HttpInterceptor to send the JWT in headers.

\* Do \*\*not store the JWT in localStorage or sessionStorage or cookies\*\*. Use in-memory storage only.

\* Handle token expiration and logout logic.

2. \*\*Backend (.NET Core)\*\*:

\* Endpoints for login, register, and a protected sample endpoint.

\* Issue JWT tokens upon successful login.

\* Secure endpoints with JWT authentication.

\* Validate and decode tokens.

\* Passwords should be securely hashed.

3. \*\*Supabase\*\*:

\* Use it only as a \*\*PostgreSQL DB\*\*, not using Supabase's built-in auth.

\* Store user credentials (email, hashed password, etc.) in a `users` table.

I need a \*\*step-by-step guide\*\*, including:

\* Project setup instructions for Angular and .NET Core

\* How to connect .NET Core to Supabase

\* Code examples for backend endpoints (register, login, protected route)

\* Angular code for auth service, interceptors, and handling in-memory tokens

\* JWT generation and validation on the server

\* Any best practices for security in this setup

Please give me clear code samples and folder structure suggestions.

# Prompt 7: Login & Registration flow

I am building a **Patient Management System** with the following tech stack:

* **Frontend**: Angular 20
* **Backend API**: .NET Core 8 Web API
* **Database**: Supabase (free subscription, PostgreSQL based)

I need help implementing a **Login & Registration flow** with the following requirements:

**Roles:**

* Admin (first user in system, created manually)
* Doctor (can belong to multiple clinics, each clinic can have multiple sites)
* Staff

**Registration Flow:**

* Admin creates **Doctor** or **Staff** accounts.
* While registering a **Doctor**:
  + Store login details (email, encrypted password with salt & hashing – e.g., BCrypt/ASP.NET Identity).
  + Store basic doctor info in UserDetail.
  + Associate with **ClinicId & SiteId** (doctor can belong to multiple clinics/sites).
  + Assign **Feature-level access** (CanAdd, CanEdit, CanDelete, CanView) for:
    1. Patient
    2. Appointment
    3. Treatment Plan
    4. Feature 4
    5. Feature 5
* Same process applies for **Staff**.

**Login Flow:**

* When Admin creates a Doctor/Staff user:
  + A **default password** is generated.
  + An **email is sent with login link** and default password.
  + On **first login**, user must **reset their password**.
  + Reset password link should redirect to frontend app URL.

**Database Tables:**

1. **Users** – Stores login credentials, encrypted password, reset tokens.
2. **UserDetail** – Stores profile info (name, email, phone, etc.).
3. **UserAccessDetail** – Stores rights per feature (CanAdd, CanEdit, CanDelete, CanView).
4. **Roles** – Master role data (Admin, Doctor, Staff).
5. **Feature** – Master feature data.

Please generate:

1. Database schema (tables with keys & relationships in Supabase/Postgres).
2. Backend implementation in **.NET Core 8 API** for:
   * User registration (Admin creates users).
   * Login & JWT authentication.
   * Reset password (first login flow).
   * Role & feature-based access control.
3. Angular 20 frontend implementation for:
   * Login page.
   * Registration form (for Admin).
   * Password reset flow.
   * Guard service for role & feature-based access control.
4. Email sending integration (via Supabase functions or backend SMTP).
5. Security best practices (hashed passwords, JWT refresh tokens, role claims).

Make sure the solution is **production-ready, scalable, and secure**.

# Prompt 8: SMTP Integration

Help me to implement SMTP integration with registration workflow when on register send email with default password, also help me how to setup app password from gmail

# Prompt 9: Angular hosting

Please provide steps to host angular application on versal which is exist in my github repo

# Prompt 10: WebAPI Hosting

Please provide steps to host .net core web api application on render which is exist in my github repo