🚀 AI-Powered MFA Implementation

Technology Stack

- **Frontend:** Angular (TypeScript)
- **Backend:** ASP.NET Core Web API (C#)
- **Database:** SQL Server (or PostgreSQL)
- **MFA Methods:** Twilio (SMS) + Google Authenticator (TOTP)
- **Authentication:** ASP.NET Core Identity + JWT

Step 1: Setup Angular & ASP.NET Core API

📌 Prompt 1: Set up Angular & ASP.NET Core API

> "Generate a step-by-step guide to set up an Angular frontend with an ASP.NET Core Web API backend. Include the installation of necessary packages, configuration of CORS, and a basic API controller."

✅ Expected AI Output:

- Install .NET SDK & Angular CLI.
- Create Angular app (`ng new radflow-mfa`).
- Create ASP.NET Core API (`dotnet new webapi -n RadFlowMFA`).
- Set up **CORS** in ASP.NET Core (`AllowAnyOrigin()`).

Step 2: Implement Authentication with JWT

📌 Prompt 2: Implement JWT Authentication

> "Generate an ASP.NET Core API authentication system using JWT with ASP.NET Core Identity. Include user registration, login, and token generation."

✅ **Expected AI Output:**

- Configure `IdentityUser`.
- Create authentication endpoints (`Register`, `Login`).
- Generate JWT tokens.
- Store users in SQL Server (EF Core).

📌 Prompt 3: Implement Angular Authentication Service

> "Generate an Angular authentication service that handles user login, token storage, and authentication state management using HttpClient and localStorage."

✅ **Expected AI Output:**

- Create `AuthService` (`login()`, `logout()`).
- Store JWT tokens in `localStorage`.
- Protect routes using `AuthGuard`.

Step 3: MFA User Settings Page (Frontend)

📌 Prompt 4: Angular UI for MFA Settings

> "Generate an Angular component for a user MFA settings page. It should allow users to enable/disable MFA, choose between SMS or Authenticator App, and view/generate backup codes."

✅ **Expected AI Output:**

- **MFA Toggle:** Enable/disable MFA.
- **Dropdown:** Select SMS or TOTP.
- **Backup Codes:** Display & regenerate.

Step 4: MFA Backend Logic

📌 Prompt 5: ASP.NET Core MFA Backend

> "Generate an ASP.NET Core API controller for handling MFA. It should support enabling MFA, sending an OTP via Twilio SMS, verifying the OTP, and handling TOTP (Google Authenticator)."

✅ Expected AI Output:

- **Enable MFA:** Store user preference.
- **Send OTP via Twilio (SMS)**
- **Generate QR Code for Google Authenticator**
- **Verify OTP** using TOTP.

Step 5: MFA Login Integration

📌 Prompt 6: Modify Login to Support MFA

> "Modify the ASP.NET Core login API to check if MFA is enabled for the user. If enabled, require the user to enter an OTP before issuing a JWT token."

✅ Expected AI Output:

- Check MFA status on login.
- Prompt for OTP if MFA is enabled.
- Verify OTP before issuing JWT.

📌 Prompt 7: Angular MFA Login Flow

> "Modify the Angular login component to support MFA. If MFA is enabled for a user, display an OTP input field after successful password authentication."

✅ Expected AI Output:

- Show OTP input **only if MFA is enabled**.
- Send OTP to the backend for verification.

- Issue JWT token **only after OTP verification**.

Step 6: Admin MFA Management

📌 Prompt 8: Admin Panel for MFA

> "Generate an Angular admin component to manage MFA for users. It should allow enabling/disabling MFA per user, resetting MFA settings, and viewing MFA logs."

✅ **Expected AI Output:**

- **Enable/Disable MFA per user.**
- **Reset MFA settings.**
- **View MFA logs.**

📌 Prompt 9: ASP.NET Core Admin API

> "Generate an ASP.NET Core API controller for admin MFA management, including enabling/disabling MFA for users, resetting MFA settings, and viewing MFA logs."

✅ **Expected AI Output:**

- **Admin API Controller** (`EnableMFA`, `ResetMFA`).
- **Log failed MFA attempts.**

Step 7: Security Enhancements

📌 Prompt 10: Secure MFA Data

> "Generate ASP.NET Core Identity code to securely store MFA preferences and backup codes in an encrypted format."

✅ **Expected AI Output:**

- Encrypt MFA settings.
- Securely store backup codes.

📌 Prompt 11: Rate Limiting & Logging

> "Generate ASP.NET Core middleware for rate limiting MFA attempts and logging failed MFA verifications."

✅ **Expected AI Output:**

- Implement rate limiting.
- Log failed OTP attempts.

Step 8: Testing & AI Reflection

📌 Prompt 12: Generate Unit Tests

> "Generate unit tests for the ASP.NET Core MFA API using xUnit and Moq."

✅ **Expected AI Output:**

- **Test authentication & MFA flows.**
- **Mock Twilio & Google Authenticator.**

✅ **Expected AI Output:**

- **AI's role in coding & debugging.**
- **Challenges & solutions.**
- **Final thoughts on AI pair programming.**