Project Documentation

EHS Software Engineer 2026 - Project Task

Medicine Checker

Azra Žunić *EHS Group*

Sarajevo, 2026

Introduction

Medicine Checker is a full-stack web application developed with the goal of providing users with an accessible and informative platform for searching medicines, reviewing safety information, screening for potential allergy risks and checking medicine availability in a local pharmacy system. The application combines publicly available drug label data with a custom backend and database in order to demonstrate a realistic healthcare-oriented software solution.

The system is designed strictly for informational purposes and does not replace professional medical advice. All displayed data originates from publicly available sources and is presented in a user-friendly, structured manner.

System Overview

The application follows a client–server architecture consisting of an Angular-based frontend and an ASP.NET Core Web API backend. The frontend is responsible for user interaction, data visualization and navigation, while the backend handles business logic, communication with external APIs, data processing and persistence. A relational SQL database is used for storing application-specific data such as inventory items, notification requests, popular searches, and allergy screening logs.

Drug Search Functionality

The core feature of the application is the drug search functionality. Users may search for a medicine by entering its name into the search interface. Upon submission, the frontend sends a request to the backend, which then queries the *openFDA* public drug label API. The retrieved data includes information such as the active ingredient, safety warnings, contraindications, and dosage instructions.

The backend processes and normalizes this data before returning a structured response to the frontend, ensuring consistency and readability regardless of variations in the external API response.

Allergy Screening

Checker includes an allergy screening mechanism that allows users to input known allergens. These allergens are not pre-stored user profiles but are provided dynamically per search, allowing flexibility and privacy.

The backend performs a text-based screening by matching the provided allergen terms against the drug's warnings, contraindications, and active ingredient descriptions. The result indicates whether a potential allergen match was found and, if applicable, lists the matched terms. Each allergy check is logged in the database for traceability and potential future analysis.

It is important to emphasize that this screening is purely informational and should not be interpreted as a medical diagnosis.

Availability and Inventory Handling

The application supports a local pharmacy availability check through an internal inventory system. Each searched medicine is associated with an availability status derived from the backend inventory data. Availability states are managed using an enum-based approach, ensuring extensibility and data consistency.

If a medicine is unavailable, the user is immediately informed through the interface. The inventory system is designed to be extendable for future integration with real-time pharmacy stock management systems.

Email Notification (“Notify Me”) Feature

When a medicine is not available, users are offered the option to leave their email address in order to be notified once the medicine becomes available. Upon submission, the email request is stored in the database and an email confirmation is sent to the user.

Email notifications are implemented using *Mailgun*, a third-party email delivery service. The backend integrates directly with the Mailgun API using a secure API key, and sends a simple notification message confirming that the request has been received. This implementation demonstrates how the system can be extended to support real-world notification workflows.

Similar Medicines Recommendation

To improve usability, the application suggests similar medicines when the searched drug is unavailable. Similarity is determined based on shared active ingredients retrieved from the drug label data. This feature allows users to explore alternative options without performing additional searches and adds practical value to the system.

Popular Searches Tracking

HealthDrugCheck tracks frequently searched medicines and displays them on the Home page as popular searches. This feature is backed by the database and provides quick access to commonly requested medicines. It also demonstrates backend-driven analytics capabilities that could be expanded in future versions of the application.

Frontend Design and User Interface

The frontend is built using Angular and follows a feature-based modular structure. Styling is implemented using SCSS, with an emphasis on a clean, modern, and professional appearance suitable for a healthcare-related application. The user interface is responsive and designed to clearly separate search input, results, safety information, availability status, and notification options.

Navigation is handled through a persistent top navigation bar, allowing users to easily switch between the Home page and the Drug Search page.

Technologies Used

The frontend of the application is developed using Angular with SCSS for styling. The backend is implemented as an ASP.NET Core Web API using Entity Framework Core for database interaction. A SQL Server database is used for persistent storage. External integrations include the openFDA API for drug data retrieval and Mailgun for email notifications.

Disclaimer

The system uses publicly available drug label data and provides informational previews only. The application does not guarantee complete accuracy and is not intended to replace professional medical advice. Users are strongly encouraged to consult qualified healthcare professionals before making any medical decisions.

Conclusion

Medicine Checker demonstrates a complete full-stack solution combining frontend development, backend API design, database management, and third-party service integration. The project showcases practical software engineering concepts such as modular architecture, external API integration, user-focused UI design and extensibility for future enhancements.