Drug Doc Documentation

Problem statement:

In the healthcare industry, ensuring the proper usage of medications is critical. However, patients and healthcare professionals often face challenges such as identifying appropriate drugs, determining safe dosage levels, and understanding the risks of medications during pregnancy. Manually finding accurate and reliable drug-related information is time-consuming and prone to errors, which can lead to severe health complications.

1. Introduction

This project is a web-based application that integrates a Flask backend with a Node.js frontend. The system is designed to assist users with the following functionalities:

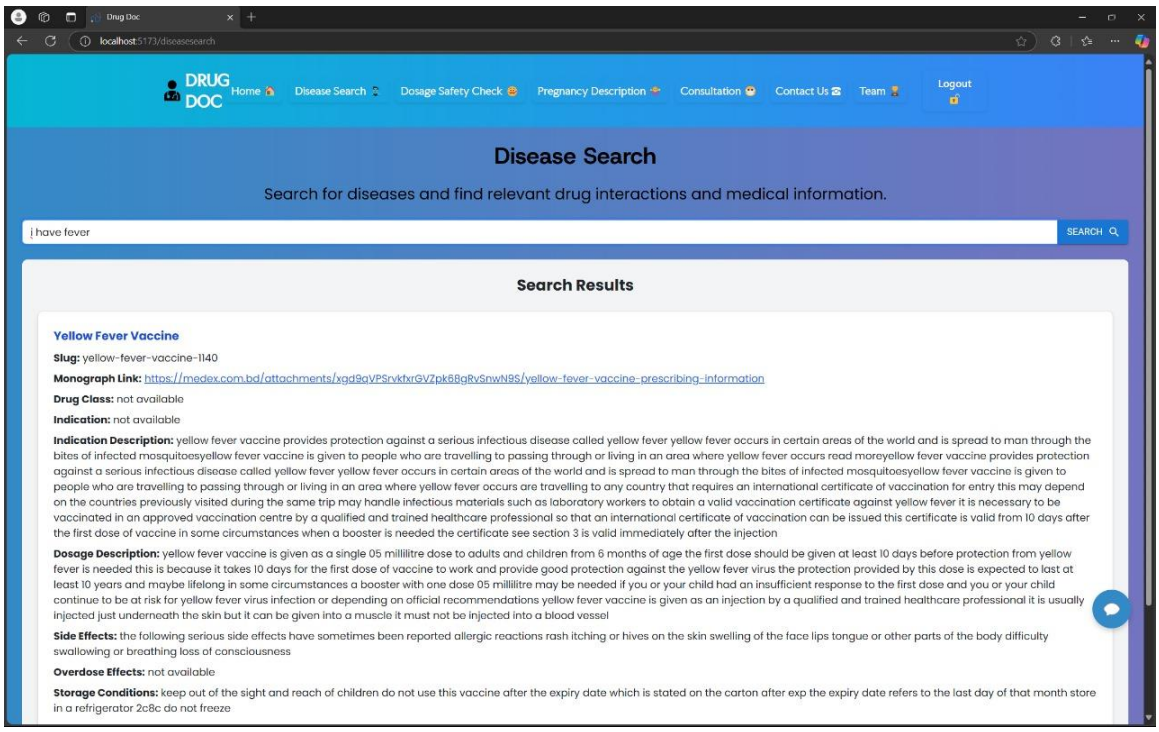
1) Disease Search,

2) Dosage Safety Check

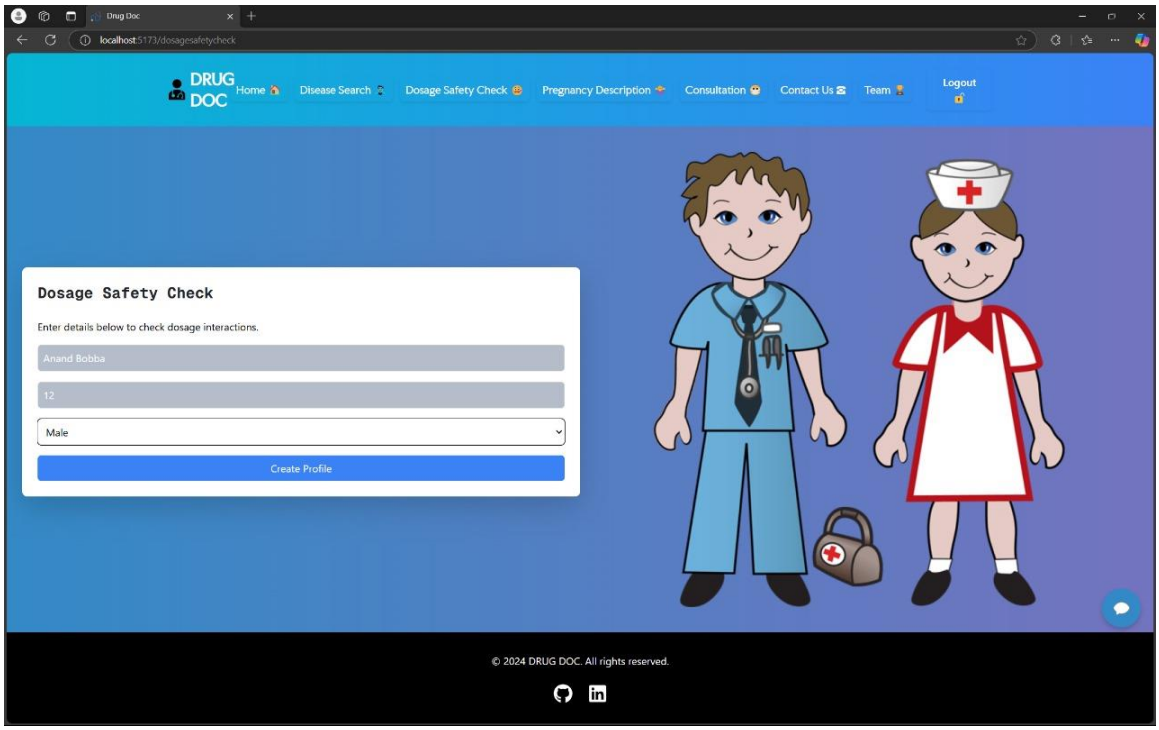
3) Pregnancy Safety Recommendations. Machine learning models trained on specific datasets are used to provide predictions based on user inputs.

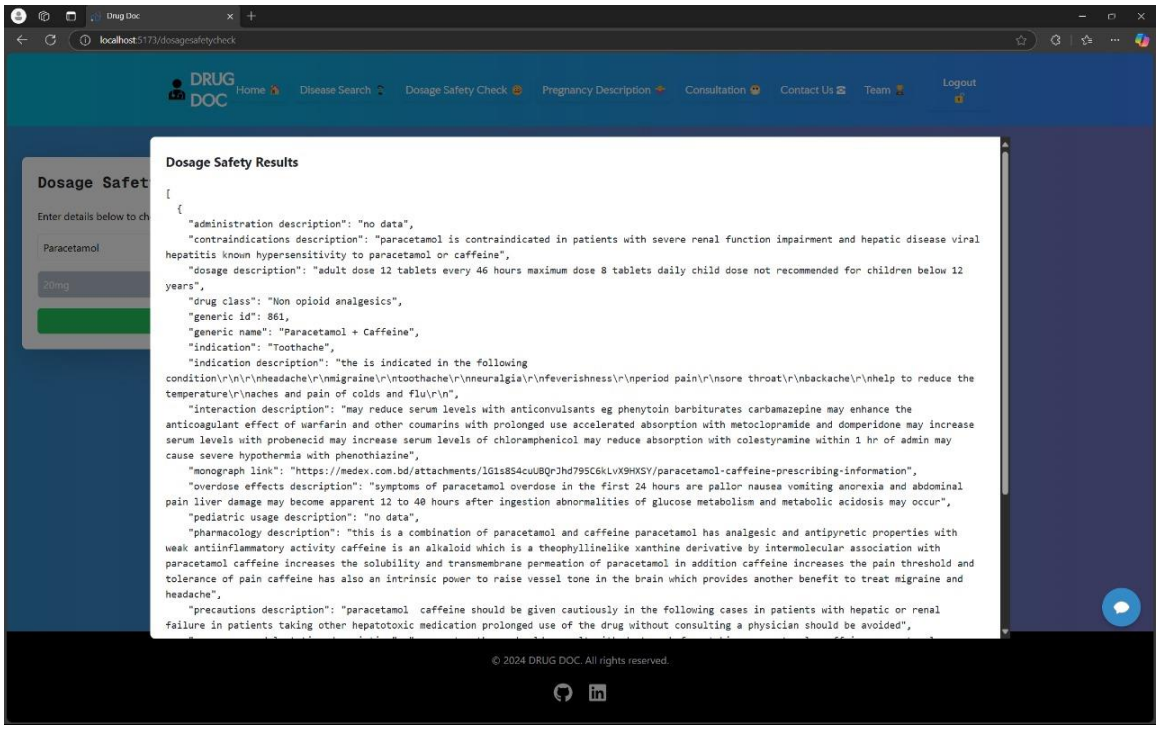
4)Mini Doctor

1.Disease Search

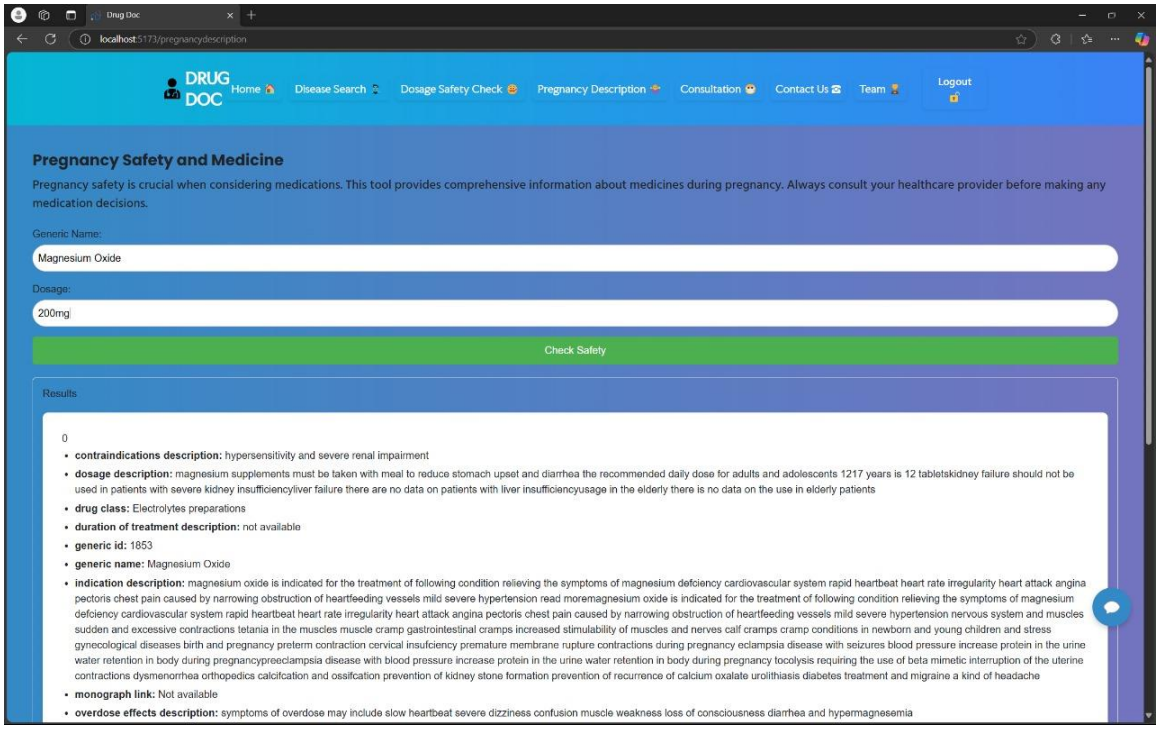


2.Dosage Safety Check

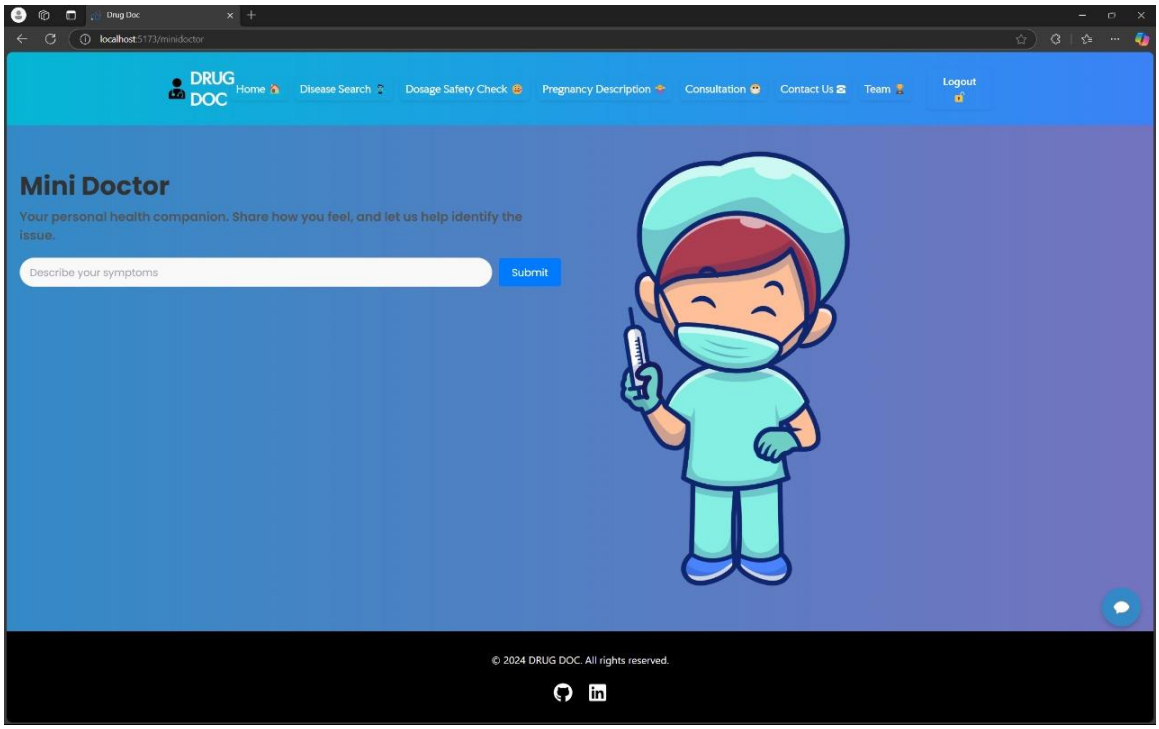




3.Pregnancy Description



4.Mini Doctor



2. Folder Structure

my_project/

├── data/ # Contains all the datasets used in the project

│ ├── dosageByme.csv # Dataset for Dosage Safety Check (used by dosage.pkl)

│ ├── generic.csv # Dataset with generic drug information

│ ├── madeby_me.csv # Dataset for Disease Search (used by Model.pkl)

│ ├── medical_combinations.csv# Additional medical data

│ └── pregnantByme.csv # Dataset for Pregnancy Safety (used by pregnant.pkl)

│

├── flaskbackend/ # Flask backend folder

│ ├── app.py # Main Flask application (backend API)

│ ├── dosagemodel.pkl # Model for Dosage Safety Check

│ ├── Model.pkl # Model for Disease Search

```
│   ├── pregnant.pkl          # Model for Pregnancy Safety
│   ├── tfidf_vectorizer.pkl  # TF-IDF vectorizer for text preprocessing (Disease Search)
│   └── ...
│
├── my_env/                   # Virtual environment for Flask (Python dependencies)
│   ├── bin/                  # Binary executables for Python and pip
│   ├── lib/                  # Installed Python packages
│   └── ...
│
├── public/                   # Public assets for the frontend (e.g., static files, images)
│
├── src/                      # React frontend source code
│   ├── components/           # React components for different pages
│   │   ├── ChatWidget.jsx    # Chat widget component (if applicable)
│   │   ├── DiseaseSearchPage.jsx # Disease Search page
│   │   ├── DosageSafetyCheckPage.jsx # Dosage Safety Check page
│   │   ├── PregnancyPage.jsx  # Pregnancy Safety page
│   │   ├── Footer.jsx        # Footer component
│   │   ├── Navbar.jsx        # Navigation bar
│   │   ├── Layout.jsx        # Layout wrapper for pages
│   │   ├── SignIn.jsx        # Sign-in form (Node.js integration)
│   │   └── SignUp.jsx        # Sign-up form (Node.js integration)
│   │
│   ├── pages/                # React pages (optional structure for routing)
│   ├── styles/               # CSS or SCSS stylesheets
│   ├── App.jsx               # Main React App component
│   └── index.js              # React entry point
│
├── node_modules/             # Node.js dependencies (auto-generated by npm)
│
```

```
├── package.json            # Node.js configuration for React (dependencies, scripts)
├── package-lock.json       # Lockfile for npm packages
│
├── README.md               # Documentation for the project
└── ...
```

3. Backend (Flask)

The Flask backend, located in the 'flaskbackend' folder, provides APIs for interacting with the three machine learning models. Each API endpoint is responsible for handling a specific frontend page.

3.1 Flask API Endpoints

 /disease-search: Handles **DiseaseSearchPage.jsx**.

- **Input**: Symptom description provided as text.
- **Output**: Predicted generic name and additional details retrieved from the madeby_me.csv dataset.

 /dosage-safety: Handles **DosageSafetyCheckPage.jsx**.

- **Input**: Generic name and dosage.
- **Output**: Predicted slug and additional details retrieved from the dosageByme.csv dataset.

 /pregnancy-safety: Handles **PregnancyPage.jsx**.

- **Input**: Generic name and dosage.
- **Output**: Predicted slug and additional details retrieved from the pregnantByme.csv dataset.

 /mini-doctor: Handles **MiniDoctorPage.jsx**.

- **Input**: User's medical query in natural language (e.g., "I have a sore throat, what should I do?").
- **Output**:
    - A context-aware response generated by the trained Mini Doctor model.
    - The response provides personalized medical advice or guidance based on the doctor-patient interaction dataset.

4. Frontend (React Components)

The frontend is implemented using React and consists of the following pages:

- **DiseaseSearchPage.jsx**:
  Allows users to input symptom descriptions and view related drug recommendations.

- **DosageSafetyCheckPage.jsx**:
  Enables users to check safety and dosage information for specific drugs.

- **PregnancyPage.jsx**:
  Provides pregnancy safety recommendations based on drug inputs.

- **MiniDoctorPage.jsx**:
  Offers an interactive feature where users can ask medical questions in natural language. The Mini Doctor model responds with context-aware, personalized medical advice based on the doctor-patient interaction dataset.

5. Machine Learning Models

The project uses three pre-trained machine learning models:
 model.pkl:

- **Used for**: DiseaseSearchPage.jsx.

- **Training Details**: Trained with 'reconstitution description' as the feature and 'generic name' as the target.

- **Output**: Predicts the 'generic name' and retrieves related details from madeby_me.csv.

 dosagemodel.pkl:

- **Used for**: DosageSafetyCheckPage.jsx.

- **Training Details**: Trained with 'generic name' and 'dosage' as features and 'slug' as the target.

- **Output**: Outputs all related data from dosageByme.csv.

 pregnant.pkl:

- **Used for**: PregnancyPage.jsx.

- **Training Details**: Trained with 'generic name' and 'dosage' as features and 'slug' as the target.

- **Output**: Outputs all related data from pregnantByme.csv.

 mini_doctor_model.pkl:

- **Used for**: MiniDoctorPage.jsx.

- **Training Details**: Trained on a doctor-patient interaction dataset to simulate medical consultations.

    - Model architecture: LSTM-based with embedding layers for understanding natural language queries.

- **Output**: Generates context-aware responses to user medical queries, offering personalized advice or guidance.

6. Data Files

The following datasets are used in this project:

- madeby_me.csv`: Contains information about generic names and their details.
- dosageByme.csv`: Includes dosage and safety information for drugs.
- pregnantByme.csv`: Contains pregnancy safety details for drugs.
- generic.csv`: Provides additional drug information.

8. Tech Stack

- Frontend: React, Tailwind.css, Material-UI, Bootstrap, Framer-motion

- Backend: Node.js, Express.js, Flask, MongoDB

- AI & ML: TensorFlow, Machine Learning for predictions

8. Conclusion

The **Medicine Interaction Checker** is more than just a technological tool—it's a step toward making healthcare more accessible, informed, and personalized. By combining a reliable Flask backend with an intuitive React frontend, the system simplifies the process of finding the right medication while offering essential safety insights, such as dosage guidance and pregnancy considerations. Its foundation in machine learning ensures not just accuracy but also adaptability as medical knowledge evolves. At its core, this project empowers users to take charge of their health with confidence, fostering trust and clarity in an often-complicated space. It’s a small yet meaningful contribution to making healthcare smarter and more human-centered.