GEORGIA SOUTHERN UNIVERSITY

PharmaCheck

Drug Interaction Database System

Anish Goyal

Department of Computer Science
Georgia Southern University
Statesboro, GA

December 4, 2025

Outline (1/2)

1. Introduction

1.1. Personal Motivation

1.2. The Need

1.3. Project Overview

2. Stage 2: Project Proposal

2.1. Beta I Conception

2.2. Database Design Goals

3. Database Design

3.1. Relational Schema

3.2. Normalization

3.3. Key Tables

4. Implementation: Backend

4.1. Technology Stack

4.2. Web Scraping System

4.3. Web Scraping Code

4.4. Database Integration

4.5. API Endpoints

4.6. AI Translation Feature

Outline (2/2)

5. Implementation: Frontend

5.1. User Interface Design

5.2. Core Features

5.3. Doctor-Patient Features

5.4. Patient Management View

6. Live Demonstration

6.1. Demo: Registration

6.2. Demo: Drug Interactions

6.3. Demo: AI Translation

6.4. Demo: Food & Disease

6.5. Demo: Doctor Dashboard

7. Challenges & Solutions

7.1. Technical Challenges

7.2. Design Challenges

7.3. UI/UX Challenges

8. Beta I: First Working System

8.1. Achievements

8.2. Data Import

8.3. Metrics

9. Future Work

9.1. Planned Enhancements

Introduction

Personal motivation, the need for drug interaction systems, and project overview.

Personal Motivation

Why this project matters to me

- This semester, I received a cancer diagnosis
- Treatment required managing multiple medications simultaneously
- Witnessed firsthand how doctors track complex drug interactions internally
- Realized patients often lack visibility into potential medication conflicts
- Inspired to build an accessible system for drug interaction awareness

Personal Motivation

Why this project matters to me

- This semester, I received a cancer diagnosis
- Treatment required managing multiple medications simultaneously
- Witnessed firsthand how doctors track complex drug interactions internally
- Realized patients often lack visibility into potential medication conflicts
- Inspired to build an accessible system for drug interaction awareness

Personal Motivation

Why this project matters to me

- This semester, I received a cancer diagnosis
- Treatment required managing multiple medications simultaneously
- Witnessed firsthand how doctors track complex drug interactions internally
- Realized patients often lack visibility into potential medication conflicts
- Inspired to build an accessible system for drug interaction awareness

Personal Motivation

Why this project matters to me

- This semester, I received a cancer diagnosis
- Treatment required managing multiple medications simultaneously
- Witnessed firsthand how doctors track complex drug interactions internally
- Realized patients often lack visibility into potential medication conflicts
- Inspired to build an accessible system for drug interaction awareness

Personal Motivation

Why this project matters to me

- This semester, I received a cancer diagnosis
- Treatment required managing multiple medications simultaneously
- Witnessed firsthand how doctors track complex drug interactions internally
- Realized patients often lack visibility into potential medication conflicts
- Inspired to build an accessible system for drug interaction awareness

The Need

Why drug interaction systems matter

- Drug interactions cause 100,000+ hospitalizations annually in the US
- Patients often take 5+ medications simultaneously
- Healthcare providers use internal systems not accessible to patients
- Existing public tools lack transparency and real-time accuracy

Key Statistics

The Need

Why drug interaction systems matter

- Drug interactions cause 100,000+ hospitalizations annually in the US
- Patients often take 5+ medications simultaneously
- Healthcare providers use internal systems not accessible to patients
- Existing public tools lack transparency and real-time accuracy

Key Statistics

The Need

Why drug interaction systems matter

- Drug interactions cause 100,000+ hospitalizations annually in the US
- Patients often take 5+ medications simultaneously
- Healthcare providers use internal systems not accessible to patients
- Existing public tools lack transparency and real-time accuracy

Key Statistics

The Need

Why drug interaction systems matter

- Drug interactions cause 100,000+ hospitalizations annually in the US
- Patients often take 5+ medications simultaneously
- Healthcare providers use internal systems not accessible to patients
- Existing public tools lack transparency and real-time accuracy

Key Statistics

The Need

Why drug interaction systems matter

- Drug interactions cause 100,000+ hospitalizations annually in the US
- Patients often take 5+ medications simultaneously
- Healthcare providers use internal systems not accessible to patients
- Existing public tools lack transparency and real-time accuracy

Key Statistics

The Need

Why drug interaction systems matter

- Drug interactions cause 100,000+ hospitalizations annually in the US
- Patients often take 5+ medications simultaneously
- Healthcare providers use internal systems not accessible to patients
- Existing public tools lack transparency and real-time accuracy

Key Statistics

The Need

Why drug interaction systems matter

- Drug interactions cause 100,000+ hospitalizations annually in the US
- Patients often take 5+ medications simultaneously
- Healthcare providers use internal systems not accessible to patients
- Existing public tools lack transparency and real-time accuracy

Key Statistics

- 82% of Americans take at least 1 medication
- 29% take 5+ medications
- Drug interactions are the 4th leading cause of death

The Need

Why drug interaction systems matter

- Drug interactions cause 100,000+ hospitalizations annually in the US
- Patients often take 5+ medications simultaneously
- Healthcare providers use internal systems not accessible to patients
- Existing public tools lack transparency and real-time accuracy

Key Statistics

- 82% of Americans take at least 1 medication
- 29% take 5+ medications
- Drug interactions are the 4th leading cause of death

The Need

Why drug interaction systems matter

- Drug interactions cause 100,000+ hospitalizations annually in the US
- Patients often take 5+ medications simultaneously
- Healthcare providers use internal systems not accessible to patients
- Existing public tools lack transparency and real-time accuracy

Key Statistics

- 82% of Americans take at least 1 medication
- 29% take 5+ medications
- Drug interactions are the 4th leading cause of death

The Need

Why drug interaction systems matter

- Drug interactions cause 100,000+ hospitalizations annually in the US
- Patients often take 5+ medications simultaneously
- Healthcare providers use internal systems not accessible to patients
- Existing public tools lack transparency and real-time accuracy

Key Statistics

- 82% of Americans take at least 1 medication
- 29% take 5+ medications
- Drug interactions are the 4th leading cause of death

The Need

Why drug interaction systems matter

- Drug interactions cause 100,000+ hospitalizations annually in the US
- Patients often take 5+ medications simultaneously
- Healthcare providers use internal systems not accessible to patients
- Existing public tools lack transparency and real-time accuracy

Key Statistics

- 82% of Americans take at least 1 medication
- 29% take 5+ medications
- Drug interactions are the 4th leading cause of death

Project Overview

What is PharmaCheck?

PharmaCheck

Stay Informed, Stay Safe.

- **PharmaCheck**: A web-based drug interaction checking platform
- Built with MySQL database, Flask backend, and modern HTML/CSS/JS frontend
- Real-time web scraping from Drugs.com for up-to-date information
- AI-powered translation of professional descriptions to patient-friendly language
- Doctor-patient relationship management for

Project Overview

What is PharmaCheck?

PharmaCheck
Stay Informed, Stay Safe.

- **PharmaCheck**: A web-based drug interaction checking platform
- Built with MySQL database, Flask backend, and modern HTML/CSS/JS frontend
- Real-time web scraping from Drugs.com for up-to-date information
- AI-powered translation of professional descriptions to patient-friendly language
- Doctor-patient relationship management for

Project Overview

What is PharmaCheck?

PharmaCheck

Stay Informed, Stay Safe.

- **PharmaCheck**: A web-based drug interaction checking platform
- Built with MySQL database, Flask backend, and modern HTML/CSS/JS frontend
- Real-time web scraping from Drugs.com for up-to-date information
- AI-powered translation of professional descriptions to patient-friendly language
- Doctor-patient relationship management for

Project Overview

What is PharmaCheck?

PharmaCheck

Stay Informed, Stay Safe.

- **PharmaCheck**: A web-based drug interaction checking platform
- Built with MySQL database, Flask backend, and modern HTML/CSS/JS frontend
- Real-time web scraping from Drugs.com for up-to-date information
- AI-powered translation of professional descriptions to patient-friendly language
- Doctor-patient relationship management for

Project Overview

What is PharmaCheck?

PharmaCheck
Stay Informed, Stay Safe.

- **PharmaCheck**: A web-based drug interaction checking platform
- Built with MySQL database, Flask backend, and modern HTML/CSS/JS frontend
- Real-time web scraping from Drugs.com for up-to-date information
- AI-powered translation of professional descriptions to patient-friendly language
- Doctor-patient relationship management for

Project Overview

What is PharmaCheck?

PharmaCheck
Stay Informed, Stay Safe.

- **PharmaCheck**: A web-based drug interaction checking platform
- Built with MySQL database, Flask backend, and modern HTML/CSS/JS frontend
- Real-time web scraping from Drugs.com for up-to-date information
- AI-powered translation of professional descriptions to patient-friendly language
- Doctor-patient relationship management for

Project Overview

What is PharmaCheck?

PharmaCheck
Stay Informed, Stay Safe.

- **PharmaCheck**: A web-based drug interaction checking platform
- Built with MySQL database, Flask backend, and modern HTML/CSS/JS frontend
- Real-time web scraping from Drugs.com for up-to-date information
- AI-powered translation of professional descriptions to patient-friendly language
- Doctor-patient relationship management for

Project Overview

What is PharmaCheck?

PharmaCheck
Stay Informed, Stay Safe.

- **PharmaCheck**: A web-based drug interaction checking platform
- Built with MySQL database, Flask backend, and modern HTML/CSS/JS frontend
- Real-time web scraping from Drugs.com for up-to-date information
- AI-powered translation of professional descriptions to patient-friendly language
- Doctor-patient relationship management for

Project Overview

What is PharmaCheck?

PharmaCheck

Stay Informed, Stay Safe.

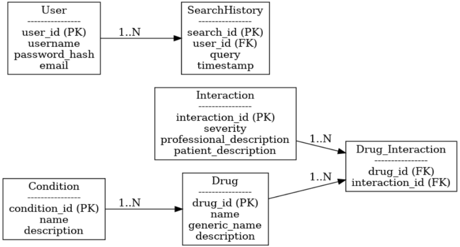
- **PharmaCheck**: A web-based drug interaction checking platform
- Built with MySQL database, Flask backend, and modern HTML/CSS/JS frontend
- Real-time web scraping from Drugs.com for up-to-date information
- AI-powered translation of professional descriptions to patient-friendly language
- Doctor-patient relationship management for

Stage 2: Project Proposal

Initial planning, ER diagram design, and Beta I conception.

Beta I Conception

Initial planning and entity identification



User
----
user_id (PK)
username
password_hash
email

1..N

SearchHistory
----
search_id (PK)
user_id (FK)
query
timestamp

Interaction
----
interaction_id (PK)
severity
professional_description
patient_description

1..N

Drug_Interaction
----
drug_id (FK)
interaction_id (FK)

Condition
----
condition_id (PK)
name
description

1..N

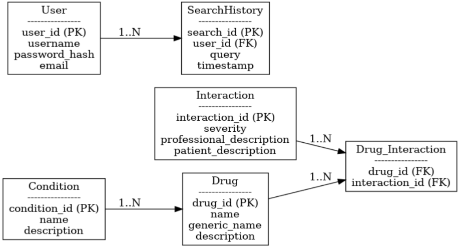
Drug
----
drug_id (PK)
name
generic_name
description

1..N

- Designed Entity-Relationship diagram
- Identified core entities:
    - User (Patient/Doctor)
    - Drug
    - Condition

Beta I Conception

Initial planning and entity identification



User
----
user_id (PK)
username
password_hash
email

1..N

SearchHistory
----
search_id (PK)
user_id (FK)
query
timestamp

Interaction
----
interaction_id (PK)
severity
professional_description
patient_description

1..N

Drug_Interaction
----
drug_id (FK)
interaction_id (FK)

1..N

Condition
----
condition_id (PK)
name
description

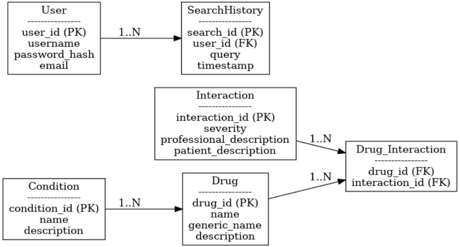
1..N

Drug
----
drug_id (PK)
name
generic_name
description

- Designed Entity-Relationship diagram
- Identified core entities:
  - User (Patient/Doctor)
  - Drug
  - Condition

Beta I Conception

Initial planning and entity identification

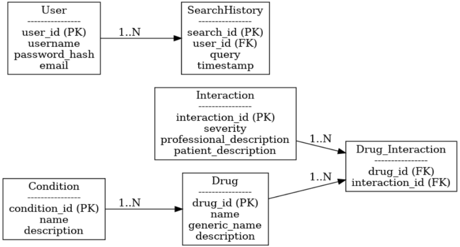


```
User
----------------
user_id (PK)
username
password_hash
email

SearchHistory
----------------
search_id (PK)
user_id (FK)
query
timestamp

Interaction
----------------
interaction_id (PK)
severity
professional_description
patient_description

Drug_Interaction
----------------
drug_id (FK)
interaction_id (FK)

Condition
----------------
condition_id (PK)
name
description

Drug
----------------
drug_id (PK)
name
generic_name
description
```

- Designed Entity-Relationship diagram
- Identified core entities:
    - User (Patient/Doctor)
    - Drug
    - Condition

Beta I Conception

Initial planning and entity identification



User
user_id (PK)
username
password_hash
email

SearchHistory
search_id (PK)
user_id (FK)
query
timestamp

Interaction
interaction_id (PK)
severity
professional_description
patient_description

Drug_Interaction
drug_id (FK)
interaction_id (FK)

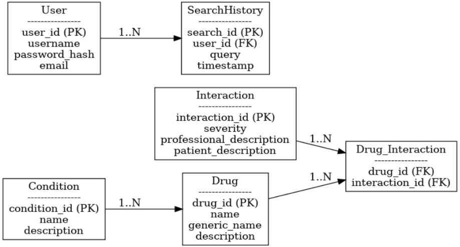
Condition
condition_id (PK)
name
description

Drug
drug_id (PK)
name
generic_name
description

- Designed Entity-Relationship diagram
- Identified core entities:
    - User (Patient/Doctor)
    - Drug
    - Condition

Beta I Conception

Initial planning and entity identification



User
----
user_id (PK)
username
password_hash
email

1..N

SearchHistory
----
search_id (PK)
user_id (FK)
query
timestamp

Interaction
----
interaction_id (PK)
severity
professional_description
patient_description

1..N

Drug_Interaction
----
drug_id (FK)
interaction_id (FK)

1..N

Condition
----
condition_id (PK)
name
description

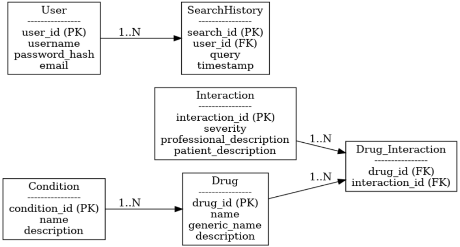
1..N

Drug
----
drug_id (PK)
name
generic_name
description

- Designed Entity-Relationship diagram
- Identified core entities:
  - User (Patient/Doctor)
  - Drug
  - Condition

Beta I Conception

Initial planning and entity identification



User
----
user_id (PK)
username
password_hash
email

1..N

SearchHistory
----
search_id (PK)
user_id (FK)
query
timestamp

Interaction
----
interaction_id (PK)
severity
professional_description
patient_description

1..N

Drug_Interaction
----
drug_id (FK)
interaction_id (FK)

1..N

Condition
----
condition_id (PK)
name
description

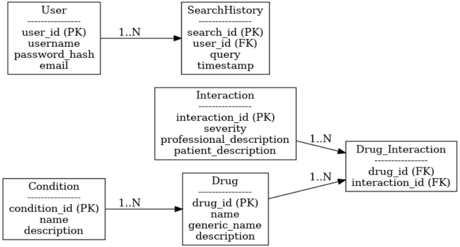
1..N

Drug
----
drug_id (PK)
name
generic_name
description

- Designed Entity-Relationship diagram
- Identified core entities:
    - User (Patient/Doctor)
    - Drug
    - Condition

Beta I Conception

Initial planning and entity identification



User
----
user_id (PK)
username
password_hash
email

1..N

SearchHistory
----
search_id (PK)
user_id (FK)
query
timestamp

Interaction
----
interaction_id (PK)
severity
professional_description
patient_description

1..N

Drug_Interaction
----
drug_id (FK)
interaction_id (FK)

1..N

Condition
----
condition_id (PK)
name
description

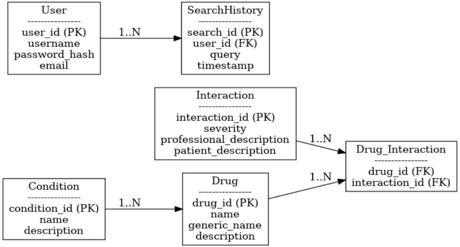
1..N

Drug
----
drug_id (PK)
name
generic_name
description

- Designed Entity-Relationship diagram
- Identified core entities:
  - User (Patient/Doctor)
  - Drug
  - Condition

Beta I Conception

Initial planning and entity identification



User
----
user_id (PK)
username
password_hash
email

1..N

SearchHistory
----
search_id (PK)
user_id (FK)
query
timestamp

Interaction
----
interaction_id (PK)
severity
professional_description
patient_description

1..N

Drug_Interaction
----
drug_id (FK)
interaction_id (FK)

1..N

Condition
----
condition_id (PK)
name
description

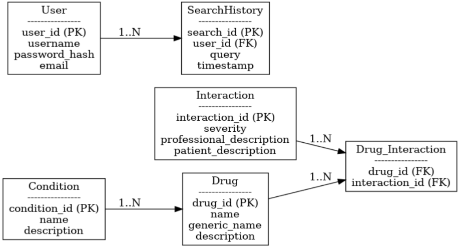
1..N

Drug
----
drug_id (PK)
name
generic_name
description

- Designed Entity-Relationship diagram
- Identified core entities:
    - User (Patient/Doctor)
    - Drug
    - Condition

Beta I Conception

Initial planning and entity identification



User
----
user_id (PK)
username
password_hash
email

1..N

SearchHistory
----
search_id (PK)
user_id (FK)
query
timestamp

Interaction
----
interaction_id (PK)
severity
professional_description
patient_description

1..N

Drug_Interaction
----
drug_id (FK)
interaction_id (FK)

1..N

Condition
----
condition_id (PK)
name
description

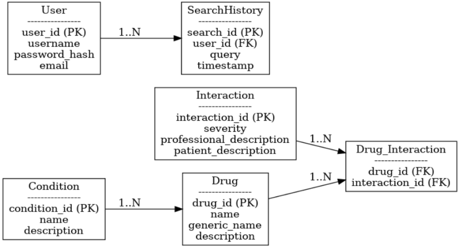
1..N

Drug
----
drug_id (PK)
name
generic_name
description

- Designed Entity-Relationship diagram
- Identified core entities:
  - User (Patient/Doctor)
  - Drug
  - Condition

Beta I Conception

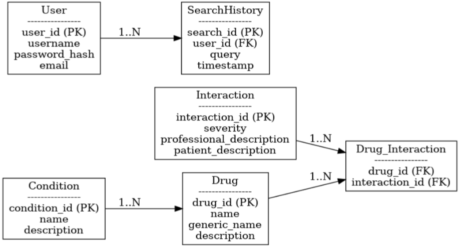
Initial planning and entity identification



- Designed Entity-Relationship diagram
- Identified core entities:
  - User (Patient/Doctor)
  - Drug
  - Condition

Beta I Conception

Initial planning and entity identification



User
----
user_id (PK)
username
password_hash
email

1..N

SearchHistory
----
search_id (PK)
user_id (FK)
query
timestamp

Interaction
----
interaction_id (PK)
severity
professional_description
patient_description

1..N

Drug_Interaction
----
drug_id (FK)
interaction_id (FK)

1..N

Condition
----
condition_id (PK)
name
description

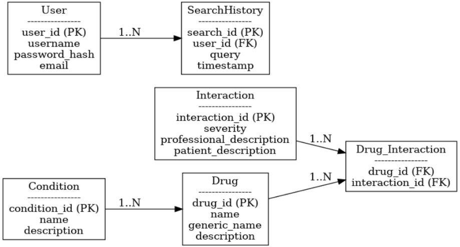
1..N

Drug
----
drug_id (PK)
name
generic_name
description

- Designed Entity-Relationship diagram
- Identified core entities:
    - User (Patient/Doctor)
    - Drug
    - Condition

Beta I Conception

Initial planning and entity identification



User
----
user_id (PK)
username
password_hash
email

1..N

SearchHistory
----
search_id (PK)
user_id (FK)
query
timestamp

Interaction
----
interaction_id (PK)
severity
professional_description
patient_description

1..N

Drug_Interaction
----
drug_id (FK)
interaction_id (FK)

1..N

Condition
----
condition_id (PK)
name
description

1..N

Drug
----
drug_id (PK)
name
generic_name
description

- Designed Entity-Relationship diagram
- Identified core entities:
  - User (Patient/Doctor)
  - Drug
  - Condition

Database Design Goals

Requirements for the system

- Support user accounts with distinct roles (Patient, Doctor)
- Track drug interactions with severity levels (Major, Moderate, Minor)
- Enable comprehensive search history for audit and review
- Support doctor-patient assignments for medical oversight
- Store both professional and patient-friendly descriptions
- Cache AI-generated translations for performance

Database Design Goals

Requirements for the system

- Support user accounts with distinct roles (Patient, Doctor)
- Track drug interactions with severity levels (Major, Moderate, Minor)
- Enable comprehensive search history for audit and review
- Support doctor-patient assignments for medical oversight
- Store both professional and patient-friendly descriptions
- Cache AI-generated translations for performance

Database Design Goals

Requirements for the system

- Support user accounts with distinct roles (Patient, Doctor)
- Track drug interactions with severity levels (Major, Moderate, Minor)
- Enable comprehensive search history for audit and review
- Support doctor-patient assignments for medical oversight
- Store both professional and patient-friendly descriptions
- Cache AI-generated translations for performance

Database Design Goals

Requirements for the system

- Support user accounts with distinct roles (Patient, Doctor)
- Track drug interactions with severity levels (Major, Moderate, Minor)
- Enable comprehensive search history for audit and review
- Support doctor-patient assignments for medical oversight
- Store both professional and patient-friendly descriptions
- Cache AI-generated translations for performance

Database Design Goals

Requirements for the system

- Support user accounts with distinct roles (Patient, Doctor)
- Track drug interactions with severity levels (Major, Moderate, Minor)
- Enable comprehensive search history for audit and review
- Support doctor-patient assignments for medical oversight
- Store both professional and patient-friendly descriptions
- Cache AI-generated translations for performance

Database Design Goals

Requirements for the system

- Support user accounts with distinct roles (Patient, Doctor)
- Track drug interactions with severity levels (Major, Moderate, Minor)
- Enable comprehensive search history for audit and review
- Support doctor-patient assignments for medical oversight
- Store both professional and patient-friendly descriptions
- Cache AI-generated translations for performance

Database Design

Relational schema, normalization, and key table structures.

Relational Schema

Eight core tables

- **User** – Authentication & roles
- **Drug** – Medication information
- **Condition** – Medical conditions
- **Interaction** – Drug-drug interactions

- **Drug_Interaction** – Junction table
- **FoodInteraction** – Food/lifestyle
- **DiseaseInteraction** – Disease interactions
- **SearchHistory** – User searches
- **Doctor_Patient** – Assignments

Relational Schema

Eight core tables

- **User** – Authentication & roles
- **Drug** – Medication information
- **Condition** – Medical conditions
- **Interaction** – Drug-drug interactions
- **Drug_Interaction** – Junction table
- **FoodInteraction** – Food/lifestyle
- **DiseaseInteraction** – Disease interactions
- **SearchHistory** – User searches
- **Doctor_Patient** – Assignments

Relational Schema

Eight core tables

- **User** – Authentication & roles
- **Drug** – Medication information
- **Condition** – Medical conditions
- **Interaction** – Drug-drug interactions
- **Drug_Interaction** – Junction table
- **FoodInteraction** – Food/lifestyle
- **DiseaseInteraction** – Disease interactions
- **SearchHistory** – User searches
- **Doctor_Patient** – Assignments

Relational Schema

Eight core tables

- **User** – Authentication & roles
- **Drug** – Medication information
- **Condition** – Medical conditions
- **Interaction** – Drug-drug interactions
- **Drug_Interaction** – Junction table
- **FoodInteraction** – Food/lifestyle
- **DiseaseInteraction** – Disease interactions
- **SearchHistory** – User searches
- **Doctor_Patient** – Assignments

Relational Schema

Eight core tables

- **User** – Authentication & roles
- **Drug** – Medication information
- **Condition** – Medical conditions
- **Interaction** – Drug-drug interactions
- **Drug_Interaction** – Junction table
- **FoodInteraction** – Food/lifestyle
- **DiseaseInteraction** – Disease interactions
- **SearchHistory** – User searches
- **Doctor_Patient** – Assignments

Relational Schema

Eight core tables

- **User** – Authentication & roles
- **Drug** – Medication information
- **Condition** – Medical conditions
- **Interaction** – Drug-drug interactions
- **Drug_Interaction** – Junction table
- **FoodInteraction** – Food/lifestyle
- **DiseaseInteraction** – Disease interactions
- **SearchHistory** – User searches
- **Doctor_Patient** – Assignments

Relational Schema

Eight core tables

- **User** – Authentication & roles
- **Drug** – Medication information
- **Condition** – Medical conditions
- **Interaction** – Drug-drug interactions

- **Drug_Interaction** – Junction table
- **FoodInteraction** – Food/lifestyle
- **DiseaseInteraction** – Disease interactions
- **SearchHistory** – User searches
- **Doctor_Patient** – Assignments

Relational Schema

Eight core tables

- **User** – Authentication & roles
- **Drug** – Medication information
- **Condition** – Medical conditions
- **Interaction** – Drug-drug interactions
- **Drug_Interaction** – Junction table
- **FoodInteraction** – Food/lifestyle
- **DiseaseInteraction** – Disease interactions
- **SearchHistory** – User searches
- **Doctor_Patient** – Assignments

Relational Schema

Eight core tables

- **User** – Authentication & roles
- **Drug** – Medication information
- **Condition** – Medical conditions
- **Interaction** – Drug-drug interactions
- **Drug_Interaction** – Junction table
- **FoodInteraction** – Food/lifestyle
- **DiseaseInteraction** – Disease interactions
- **SearchHistory** – User searches
- **Doctor_Patient** – Assignments

Relational Schema

Eight core tables

- **User** – Authentication & roles
- **Drug** – Medication information
- **Condition** – Medical conditions
- **Interaction** – Drug-drug interactions
- **Drug_Interaction** – Junction table
- **FoodInteraction** – Food/lifestyle
- **DiseaseInteraction** – Disease interactions
- **SearchHistory** – User searches
- **Doctor_Patient** – Assignments

Relational Schema

Eight core tables

- **User** – Authentication & roles
- **Drug** – Medication information
- **Condition** – Medical conditions
- **Interaction** – Drug-drug interactions
- **Drug_Interaction** – Junction table
- **FoodInteraction** – Food/lifestyle
- **DiseaseInteraction** – Disease interactions
- **SearchHistory** – User searches
- **Doctor_Patient** – Assignments

Relational Schema

Eight core tables

- **User** – Authentication & roles
- **Drug** – Medication information
- **Condition** – Medical conditions
- **Interaction** – Drug-drug interactions
- **Drug_Interaction** – Junction table
- **FoodInteraction** – Food/lifestyle
- **DiseaseInteraction** – Disease interactions
- **SearchHistory** – User searches
- **Doctor_Patient** – Assignments

Normalization

Third Normal Form (3NF) compliance

- All tables satisfy Third Normal Form (3NF)
- **User**: user_id → username, password_hash, email, role
- **Drug**: drug_id → name, generic_name, description, condition_id
- **Interaction**: interaction_id → severity, professional_description, patient_description
- No transitive dependencies exist
- Foreign keys ensure referential integrity with CASCADE operations

Normalization

Third Normal Form (3NF) compliance

- All tables satisfy Third Normal Form (3NF)
- **User**: user_id → username, password_hash, email, role
- **Drug**: drug_id → name, generic_name, description, condition_id
- **Interaction**: interaction_id → severity, professional_description, patient_description
- No transitive dependencies exist
- Foreign keys ensure referential integrity with CASCADE operations

Normalization

Third Normal Form (3NF) compliance

- All tables satisfy Third Normal Form (3NF)
- **User**: user_id → username, password_hash, email, role
- **Drug**: drug_id → name, generic_name, description, condition_id
- **Interaction**: interaction_id → severity, professional_description, patient_description
- No transitive dependencies exist
- Foreign keys ensure referential integrity with CASCADE operations

Normalization

Third Normal Form (3NF) compliance

- All tables satisfy Third Normal Form (3NF)
- **User**: user_id → username, password_hash, email, role
- **Drug**: drug_id → name, generic_name, description, condition_id
- **Interaction**: interaction_id → severity, professional_description, patient_description
- No transitive dependencies exist
- Foreign keys ensure referential integrity with CASCADE operations

Normalization

Third Normal Form (3NF) compliance

- All tables satisfy Third Normal Form (3NF)
- **User**: user_id → username, password_hash, email, role
- **Drug**: drug_id → name, generic_name, description, condition_id
- **Interaction**: interaction_id → severity, professional_description, patient_description
- No transitive dependencies exist
- Foreign keys ensure referential integrity with CASCADE operations

Normalization

Third Normal Form (3NF) compliance

- All tables satisfy Third Normal Form (3NF)
- **User**: user_id → username, password_hash, email, role
- **Drug**: drug_id → name, generic_name, description, condition_id
- **Interaction**: interaction_id → severity, professional_description, patient_description
- No transitive dependencies exist
- Foreign keys ensure referential integrity with CASCADE operations

Key Tables Deep Dive

User and SearchHistory tables

User Table

```
CREATE TABLE User (
  user_id INT PRIMARY KEY
    AUTO_INCREMENT,
  username VARCHAR(64)
    NOT NULL UNIQUE,
  password_hash CHAR(60)
    NOT NULL,
  email VARCHAR(255)
    NOT NULL UNIQUE,
  role ENUM(’PATIENT’,
    ’DOCTOR’) NOT NULL
);
```

SearchHistory Table

```
CREATE TABLE SearchHistory (
  search_id BIGINT PRIMARY KEY
    AUTO_INCREMENT,
  user_id INT NOT NULL,
  query TEXT NOT NULL,
  search_type ENUM(’DRUG’,
    ’CONDITION’,
    ’INTERACTION’),
  search_data TEXT,
  created_at DATETIME
    DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (user_id)
    REFERENCES User(user_id)
);
```

Key Tables Deep Dive

User and SearchHistory tables

User Table

```sql
CREATE TABLE User (
  user_id INT PRIMARY KEY
    AUTO_INCREMENT,
  username VARCHAR(64)
    NOT NULL UNIQUE,
  password_hash CHAR(60)
    NOT NULL,
  email VARCHAR(255)
    NOT NULL UNIQUE,
  role ENUM('PATIENT',
    'DOCTOR') NOT NULL
);
```

SearchHistory Table

```sql
CREATE TABLE SearchHistory (
  search_id BIGINT PRIMARY KEY
    AUTO_INCREMENT,
  user_id INT NOT NULL,
  query TEXT NOT NULL,
  search_type ENUM('DRUG',
    'CONDITION',
    'INTERACTION'),
  search_data TEXT,
  created_at DATETIME
    DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (user_id)
    REFERENCES User(user_id)
);
```

Key Tables Deep Dive

User and SearchHistory tables

User Table

```sql
CREATE TABLE User (
  user_id INT PRIMARY KEY
    AUTO_INCREMENT,
  username VARCHAR(64)
    NOT NULL UNIQUE,
  password_hash CHAR(60)
    NOT NULL,
  email VARCHAR(255)
    NOT NULL UNIQUE,
  role ENUM('PATIENT',
    'DOCTOR') NOT NULL
);
```

SearchHistory Table

```sql
CREATE TABLE SearchHistory (
  search_id BIGINT PRIMARY KEY
    AUTO_INCREMENT,
  user_id INT NOT NULL,
  query TEXT NOT NULL,
  search_type ENUM('DRUG',
    'CONDITION',
    'INTERACTION'),
  search_data TEXT,
  created_at DATETIME
    DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (user_id)
    REFERENCES User(user_id)
);
```

Implementation: Backend

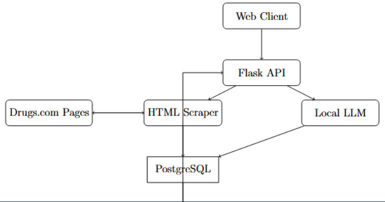
Technology stack, web scraping, database integration, and AI features.

Technology Stack

Backend components

- **Database**: MySQL
  - Originally planned PostgreSQL
  - Switched to MySQL for compatibility
- **Backend**: Flask (Python)
- **ORM**: SQLAlchemy

- **Web Scraping**: BeautifulSoup
- **Authentication**: JWT + bcrypt
- **AI Translation**: Ollama LLM
- **Data Format**: JSON APIs

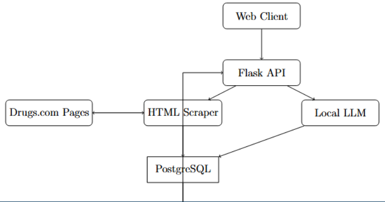


Technology Stack

Backend components

- **Database**: MySQL
  - Originally planned PostgreSQL
  - Switched to MySQL for compatibility
- **Backend**: Flask (Python)
- **ORM**: SQLAlchemy

- **Web Scraping**: BeautifulSoup
- **Authentication**: JWT + bcrypt
- **AI Translation**: Ollama LLM
- **Data Format**: JSON APIs

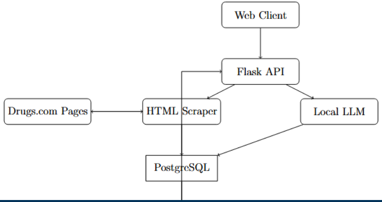


Technology Stack

Backend components

- **Database**: MySQL
  - Originally planned PostgreSQL
  - Switched to MySQL for compatibility
- **Backend**: Flask (Python)
- **ORM**: SQLAlchemy

- **Web Scraping**: BeautifulSoup
- **Authentication**: JWT + bcrypt
- **AI Translation**: Ollama LLM
- **Data Format**: JSON APIs

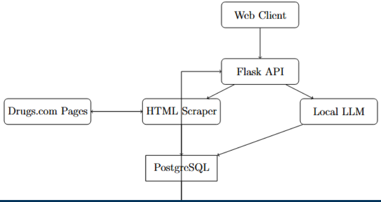


Technology Stack

Backend components

- **Database**: MySQL
  - Originally planned PostgreSQL
  - Switched to MySQL for compatibility
- **Backend**: Flask (Python)
- **ORM**: SQLAlchemy

- **Web Scraping**: BeautifulSoup
- **Authentication**: JWT + bcrypt
- **AI Translation**: Ollama LLM
- **Data Format**: JSON APIs

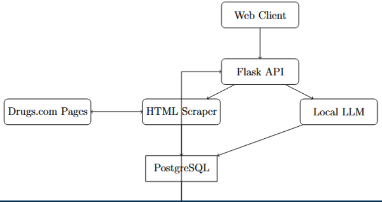


Technology Stack

Backend components

- **Database**: MySQL
  - Originally planned PostgreSQL
  - Switched to MySQL for compatibility
- **Backend**: Flask (Python)
- **ORM**: SQLAlchemy

- **Web Scraping**: BeautifulSoup
- **Authentication**: JWT + bcrypt
- **AI Translation**: Ollama LLM
- **Data Format**: JSON APIs

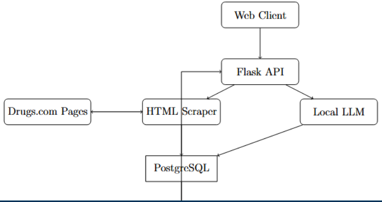


Technology Stack

Backend components

- **Database**: MySQL
  - Originally planned PostgreSQL
  - Switched to MySQL for compatibility
- **Backend**: Flask (Python)
- **ORM**: SQLAlchemy

- **Web Scraping**: BeautifulSoup
- **Authentication**: JWT + bcrypt
- **AI Translation**: Ollama LLM
- **Data Format**: JSON APIs

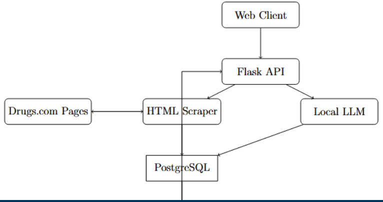


Technology Stack

Backend components

- **Database**: MySQL
  - Originally planned PostgreSQL
  - Switched to MySQL for compatibility
- **Backend**: Flask (Python)
- **ORM**: SQLAlchemy

- **Web Scraping**: BeautifulSoup
- **Authentication**: JWT + bcrypt
- **AI Translation**: Ollama LLM
- **Data Format**: JSON APIs

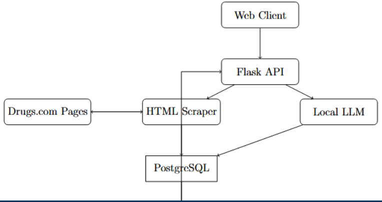


Technology Stack

Backend components

- **Database**: MySQL
  - Originally planned PostgreSQL
  - Switched to MySQL for compatibility
- **Backend**: Flask (Python)
- **ORM**: SQLAlchemy

- **Web Scraping**: BeautifulSoup
- **Authentication**: JWT + bcrypt
- **AI Translation**: Ollama LLM
- **Data Format**: JSON APIs

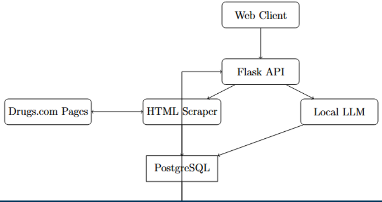


Technology Stack

Backend components

- **Database**: MySQL
  - Originally planned PostgreSQL
  - Switched to MySQL for compatibility
- **Backend**: Flask (Python)
- **ORM**: SQLAlchemy

- **Web Scraping**: BeautifulSoup
- **Authentication**: JWT + bcrypt
- **AI Translation**: Ollama LLM
- **Data Format**: JSON APIs

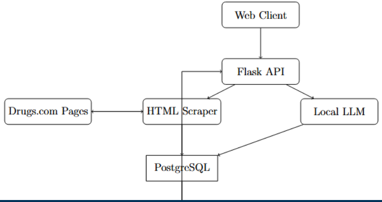


Technology Stack

Backend components

- **Database**: MySQL
  - Originally planned PostgreSQL
  - Switched to MySQL for compatibility
- **Backend**: Flask (Python)
- **ORM**: SQLAlchemy

- **Web Scraping**: BeautifulSoup
- **Authentication**: JWT + bcrypt
- **AI Translation**: Ollama LLM
- **Data Format**: JSON APIs

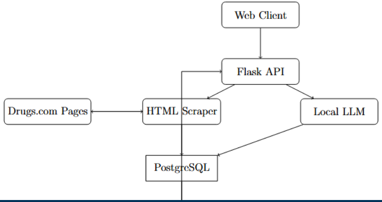


Technology Stack

Backend components

- **Database**: MySQL
    - Originally planned PostgreSQL
    - Switched to MySQL for compatibility
- **Backend**: Flask (Python)
- **ORM**: SQLAlchemy

- **Web Scraping**: BeautifulSoup
- **Authentication**: JWT + bcrypt
- **AI Translation**: Ollama LLM
- **Data Format**: JSON APIs

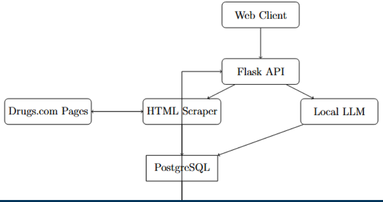


Technology Stack

Backend components

- **Database**: MySQL
  - Originally planned PostgreSQL
  - Switched to MySQL for compatibility
- **Backend**: Flask (Python)
- **ORM**: SQLAlchemy

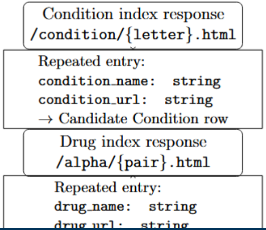
- **Web Scraping**: BeautifulSoup
- **Authentication**: JWT + bcrypt
- **AI Translation**: Ollama LLM
- **Data Format**: JSON APIs



Web Scraping System

Real-time data acquisition from Drugs.com

- **DrugInteractionChecker**: Primary scraper class
- **FoodInteractionScraper**: Food/lifestyle interactions
- **DiseaseInteractionScraper**: Disease-related interactions
- HTML parsing with BeautifulSoup
- Intelligent caching to reduce redundant requests
- Brand name → Generic name resolution (e.g., Valium → Diazepam)



Condition index response
`/condition/{letter}.html`

Repeated entry:
`condition_name:  string`
`condition_url:  string`
→ Candidate Condition row

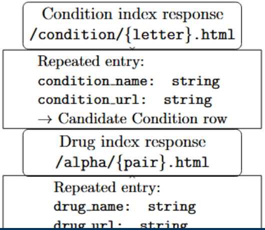
Drug index response
`/alpha/{pair}.html`

Repeated entry:
`drug_name:  string`
`drug_url:  string`

Web Scraping System

Real-time data acquisition from Drugs.com

- **DrugInteractionChecker**: Primary scraper class
- **FoodInteractionScraper**: Food/lifestyle interactions
- **DiseaseInteractionScraper**: Disease-related interactions
- HTML parsing with BeautifulSoup
- Intelligent caching to reduce redundant requests
- Brand name → Generic name resolution (e.g., Valium → Diazepam)



Condition index response
/condition/{letter}.html

Repeated entry:
condition_name: string
condition_url: string
→ Candidate Condition row

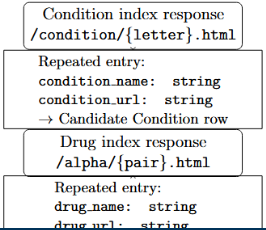
Drug index response
/alpha/{pair}.html

Repeated entry:
drug_name: string
drug_url: string

Web Scraping System

Real-time data acquisition from Drugs.com

- **DrugInteractionChecker**: Primary scraper class
- **FoodInteractionScraper**: Food/lifestyle interactions
- **DiseaseInteractionScraper**: Disease-related interactions
- HTML parsing with BeautifulSoup
- Intelligent caching to reduce redundant requests
- Brand name → Generic name resolution (e.g., Valium → Diazepam)



Condition index response
/condition/{letter}.html

Repeated entry:
condition_name: string
condition_url: string
→ Candidate Condition row

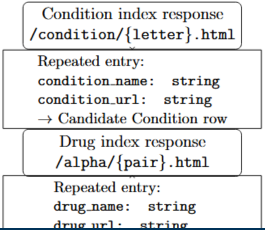
Drug index response
/alpha/{pair}.html

Repeated entry:
drug_name: string
drug_url: string

Web Scraping System

Real-time data acquisition from Drugs.com

- **DrugInteractionChecker**: Primary scraper class
- **FoodInteractionScraper**: Food/lifestyle interactions
- **DiseaseInteractionScraper**: Disease-related interactions
- HTML parsing with BeautifulSoup
- Intelligent caching to reduce redundant requests
- Brand name → Generic name resolution (e.g., Valium → Diazepam)



Condition index response
`/condition/{letter}.html`

Repeated entry:
`condition_name: string`
`condition_url: string`
→ Candidate Condition row

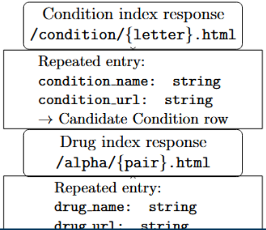
Drug index response
`/alpha/{pair}.html`

Repeated entry:
`drug_name: string`
`drug_url: string`

Web Scraping System

Real-time data acquisition from Drugs.com

- **DrugInteractionChecker**: Primary scraper class
- **FoodInteractionScraper**: Food/lifestyle interactions
- **DiseaseInteractionScraper**: Disease-related interactions
- HTML parsing with BeautifulSoup
- Intelligent caching to reduce redundant requests
- Brand name → Generic name resolution (e.g., Valium → Diazepam)



Condition index response
`/condition/{letter}.html`

Repeated entry:
`condition_name: string`
`condition_url: string`
→ Candidate Condition row

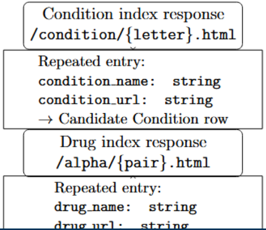
Drug index response
`/alpha/{pair}.html`

Repeated entry:
`drug_name: string`
`drug_url: string`

Web Scraping System

Real-time data acquisition from Drugs.com

- **DrugInteractionChecker**: Primary scraper class
- **FoodInteractionScraper**: Food/lifestyle interactions
- **DiseaseInteractionScraper**: Disease-related interactions
- HTML parsing with BeautifulSoup
- Intelligent caching to reduce redundant requests
- Brand name → Generic name resolution (e.g., Valium → Diazepam)



Condition index response
`/condition/{letter}.html`

Repeated entry:
`condition_name:` `string`
`condition_url:` `string`
→ Candidate Condition row

Drug index response
`/alpha/{pair}.html`

Repeated entry:
`drug_name:` `string`
`drug_url:` `string`

Web Scraping System

[CODE DEMO] scraper.py - DrugInteractionChecker

[DEMO: Show scraper.py in IDE]

Key methods to highlight:

- `get_drug_interactions()` – Fetches drug-drug interactions
- `get_food_interactions()` – Fetches food/lifestyle interactions
- `_get_generic_name()` – Resolves brand names

Web Scraping System

[CODE DEMO] scraper.py - DrugInteractionChecker

[DEMO: Show scraper.py in IDE]

Key methods to highlight:

- `get_drug_interactions()` – Fetches drug-drug interactions
- `get_food_interactions()` – Fetches food/lifestyle interactions
- `_get_generic_name()` – Resolves brand names

Web Scraping System

[CODE DEMO] scraper.py - DrugInteractionChecker

[DEMO: Show scraper.py in IDE]

Key methods to highlight:

- `get_drug_interactions()` – Fetches drug-drug interactions
- `get_food_interactions()` – Fetches food/lifestyle interactions
- `_get_generic_name()` – Resolves brand names

Database Integration

SQLAlchemy ORM models

```python
class User(Base):
    __tablename__ = 'User'

    user_id = Column(Integer, primary_key=True, autoincrement=True)
    username = Column(String(64), nullable=False, unique=True)
    password_hash = Column(String(60), nullable=False)
    email = Column(String(255), nullable=False, unique=True)
    role = Column(Enum('PATIENT', 'DOCTOR'), nullable=False)

    # Relationships
    search_history = relationship('SearchHistory', back_populates='user')
    patients = relationship('User', secondary=doctor_patient_table,
                           primaryjoin=(user_id == doctor_patient_table.c.doctor_id),
                           secondaryjoin=(user_id == doctor_patient_table.c.patient_id))

```

[DEMO: Show database.py in IDE]

API Endpoints

RESTful API design

Authentication

- POST /auth/register
- POST /auth/login
- GET /auth/me

Drug Search

- GET /search_drugs
- GET /search_conditions

Interactions

- POST /check_drug_interactions
- GET /food_interactions
- GET /disease_interactions

Doctor-Patient

- GET /doctors/patients
- POST /patients/request_doctor

[DEMO: Show api.py endpoints in IDE]

API Endpoints

RESTful API design

Authentication

- POST /auth/register
- POST /auth/login
- GET /auth/me

Drug Search

- GET /search_drugs
- GET /search_conditions

Interactions

- POST /check_drug_interactions
- GET /food_interactions
- GET /disease_interactions

Doctor-Patient

- GET /doctors/patients
- POST /patients/request_doctor

[DEMO: Show api.py endpoints in IDE]

API Endpoints

RESTful API design

Authentication

- POST /auth/register
- POST /auth/login
- GET /auth/me

Drug Search

- GET /search_drugs
- GET /search_conditions

Interactions

- POST /check_drug_interactions
- GET /food_interactions
- GET /disease_interactions

Doctor-Patient

- GET /doctors/patients
- POST /patients/request_doctor

[DEMO: Show api.py endpoints in IDE]

API Endpoints

RESTful API design

Authentication

- POST /auth/register
- POST /auth/login
- GET /auth/me

Drug Search

- GET /search_drugs
- GET /search_conditions

Interactions

- POST /check_drug_interactions
- GET /food_interactions
- GET /disease_interactions

Doctor-Patient

- GET /doctors/patients
- POST /patients/request_doctor

[DEMO: Show api.py endpoints in IDE]

API Endpoints

RESTful API design

Authentication

- POST /auth/register
- POST /auth/login
- GET /auth/me

Drug Search

- GET /search_drugs
- GET /search_conditions

Interactions

- POST /check_drug_interactions
- GET /food_interactions
- GET /disease_interactions

Doctor-Patient

- GET /doctors/patients
- POST /patients/request_doctor

[DEMO: Show api.py endpoints in IDE]

API Endpoints

RESTful API design

Authentication

- POST /auth/register
- POST /auth/login
- GET /auth/me

Drug Search

- GET /search_drugs
- GET /search_conditions

Interactions

- POST /check_drug_interactions
- GET /food_interactions
- GET /disease_interactions

Doctor-Patient

- GET /doctors/patients
- POST /patients/request_doctor

[DEMO: Show api.py endpoints in IDE]

API Endpoints

RESTful API design

Authentication

- POST /auth/register
- POST /auth/login
- GET /auth/me

Drug Search

- GET /search_drugs
- GET /search_conditions

Interactions

- POST /check_drug_interactions
- GET /food_interactions
- GET /disease_interactions

Doctor-Patient

- GET /doctors/patients
- POST /patients/request_doctor

[DEMO: Show api.py endpoints in IDE]

API Endpoints

RESTful API design

Authentication

- POST /auth/register
- POST /auth/login
- GET /auth/me

Drug Search

- GET /search_drugs
- GET /search_conditions

Interactions

- POST /check_drug_interactions
- GET /food_interactions
- GET /disease_interactions

Doctor-Patient

- GET /doctors/patients
- POST /patients/request_doctor

[DEMO: Show api.py endpoints in IDE]

API Endpoints

RESTful API design

Authentication

- POST /auth/register
- POST /auth/login
- GET /auth/me

Drug Search

- GET /search_drugs
- GET /search_conditions

Interactions

- POST /check_drug_interactions
- GET /food_interactions
- GET /disease_interactions

Doctor-Patient

- GET /doctors/patients
- POST /patients/request_doctor

[DEMO: Show api.py endpoints in IDE]

API Endpoints

RESTful API design

Authentication

- POST /auth/register
- POST /auth/login
- GET /auth/me

Drug Search

- GET /search_drugs
- GET /search_conditions

Interactions

- POST /check_drug_interactions
- GET /food_interactions
- GET /disease_interactions

Doctor-Patient

- GET /doctors/patients
- POST /patients/request_doctor

[DEMO: Show api.py endpoints in IDE]

API Endpoints

RESTful API design

Authentication

- POST /auth/register
- POST /auth/login
- GET /auth/me

Drug Search

- GET /search_drugs
- GET /search_conditions

Interactions

- POST /check_drug_interactions
- GET /food_interactions
- GET /disease_interactions

Doctor-Patient

- GET /doctors/patients
- POST /patients/request_doctor

[DEMO: Show api.py endpoints in IDE]

API Endpoints

RESTful API design

Authentication

- POST /auth/register
- POST /auth/login
- GET /auth/me

Drug Search

- GET /search_drugs
- GET /search_conditions

Interactions

- POST /check_drug_interactions
- GET /food_interactions
- GET /disease_interactions

Doctor-Patient

- GET /doctors/patients
- POST /patients/request_doctor

[DEMO: Show api.py endpoints in IDE]

API Endpoints

RESTful API design

Authentication

- POST /auth/register
- POST /auth/login
- GET /auth/me

Drug Search

- GET /search_drugs
- GET /search_conditions

Interactions

- POST /check_drug_interactions
- GET /food_interactions
- GET /disease_interactions

Doctor-Patient

- GET /doctors/patients
- POST /patients/request_doctor

[DEMO: Show api.py endpoints in IDE]

AI Translation Feature

Ollama LLM integration

- Translates professional medical descriptions to patient-friendly language
- Uses locally-deployed Ollama LLM (privacy-preserving)
- Caches translations in database for performance
- Users can toggle between professional and AI-translated views

Before (Professional)

"Using fluoxetine together with diazePAM may increase side effects such as dizziness, drowsiness, confusion, and difficulty concentrating..."

After (AI Translated)

"Taking these two medications together may make you feel more dizzy or sleepy than usual. Be careful when driving..."

AI Translation Feature

Ollama LLM integration

- Translates professional medical descriptions to patient-friendly language
- Uses locally-deployed Ollama LLM (privacy-preserving)
- Caches translations in database for performance
- Users can toggle between professional and AI-translated views

Before (Professional)

"Using fluoxetine together with diazePAM may increase side effects such as dizziness, drowsiness, confusion, and difficulty concentrating..."

After (AI Translated)

"Taking these two medications together may make you feel more dizzy or sleepy than usual. Be careful when driving..."

AI Translation Feature

Ollama LLM integration

- Translates professional medical descriptions to patient-friendly language
- Uses locally-deployed Ollama LLM (privacy-preserving)
- Caches translations in database for performance
- Users can toggle between professional and AI-translated views

Before (Professional)

"Using fluoxetine together with diazePAM may increase side effects such as dizziness, drowsiness, confusion, and difficulty concentrating..."

After (AI Translated)

"Taking these two medications together may make you feel more dizzy or sleepy than usual. Be careful when driving..."

AI Translation Feature

Ollama LLM integration

- Translates professional medical descriptions to patient-friendly language
- Uses locally-deployed Ollama LLM (privacy-preserving)
- Caches translations in database for performance
- Users can toggle between professional and AI-translated views

Before (Professional)

"Using fluoxetine together with diazePAM may increase side effects such as dizziness, drowsiness, confusion, and difficulty concentrating..."

After (AI Translated)

"Taking these two medications together may make you feel more dizzy or sleepy than usual. Be careful when driving..."

AI Translation Feature

Ollama LLM integration

- Translates professional medical descriptions to patient-friendly language
- Uses locally-deployed Ollama LLM (privacy-preserving)
- Caches translations in database for performance
- Users can toggle between professional and AI-translated views

Before (Professional)

"Using fluoxetine together with diazePAM may increase side effects such as dizziness, drowsiness, confusion, and difficulty concentrating..."

After (AI Translated)

"Taking these two medications together may make you feel more dizzy or sleepy than usual. Be careful when driving..."

AI Translation Feature

Ollama LLM integration

- Translates professional medical descriptions to patient-friendly language
- Uses locally-deployed Ollama LLM (privacy-preserving)
- Caches translations in database for performance
- Users can toggle between professional and AI-translated views

Before (Professional)

"Using fluoxetine together with diazePAM may increase side effects such as dizziness, drowsiness, confusion, and difficulty concentrating..."

After (AI Translated)

"Taking these two medications together may make you feel more dizzy or sleepy than usual. Be careful when driving..."

AI Translation Feature

Ollama LLM integration

- Translates professional medical descriptions to patient-friendly language
- Uses locally-deployed Ollama LLM (privacy-preserving)
- Caches translations in database for performance
- Users can toggle between professional and AI-translated views

Before (Professional)

"Using fluoxetine together with diazePAM may increase side effects such as dizziness, drowsiness, confusion, and difficulty concentrating..."

After (AI Translated)

"Taking these two medications together may make you feel more dizzy or sleepy than usual. Be careful when driving..."

AI Translation Feature

Ollama LLM integration

- Translates professional medical descriptions to patient-friendly language
- Uses locally-deployed Ollama LLM (privacy-preserving)
- Caches translations in database for performance
- Users can toggle between professional and AI-translated views

Before (Professional)

"Using fluoxetine together with diazePAM may increase side effects such as dizziness, drowsiness, confusion, and difficulty concentrating..."

After (AI Translated)

"Taking these two medications together may make you feel more dizzy or sleepy than usual. Be careful when driving..."

AI Translation Feature

Ollama LLM integration

- Translates professional medical descriptions to patient-friendly language
- Uses locally-deployed Ollama LLM (privacy-preserving)
- Caches translations in database for performance
- Users can toggle between professional and AI-translated views

Before (Professional)

"Using fluoxetine together with diazePAM may increase side effects such as dizziness, drowsiness, confusion, and difficulty concentrating..."

After (AI Translated)

"Taking these two medications together may make you feel more dizzy or sleepy than usual. Be careful when driving..."

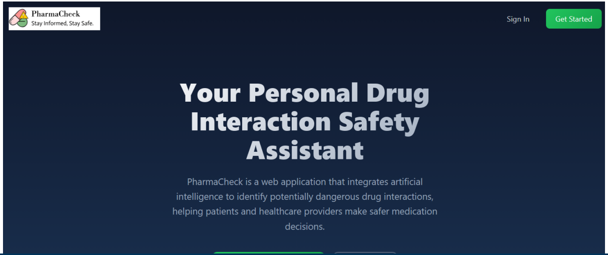
Implementation: Frontend

User interface design, core features, and doctor-patient views.

User Interface Design

Modern, responsive web application

- Built with vanilla HTML, CSS, and JavaScript
- No heavy framework overhead
- Flask serves static files directly
- Responsive design for desktop and mobile
- Clean, medical-professional aesthetic



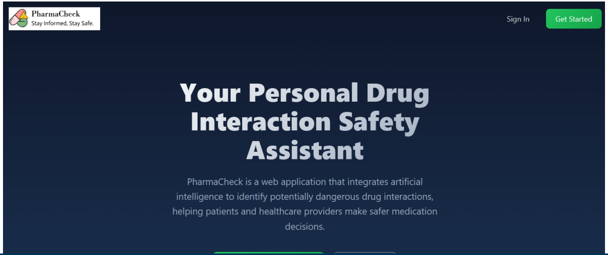
Your Personal Drug Interaction Safety Assistant

PharmaCheck is a web application that integrates artificial intelligence to identify potentially dangerous drug interactions, helping patients and healthcare providers make safer medication decisions.

User Interface Design

Modern, responsive web application

- Built with vanilla HTML, CSS, and JavaScript
- No heavy framework overhead
- Flask serves static files directly
- Responsive design for desktop and mobile
- Clean, medical-professional aesthetic



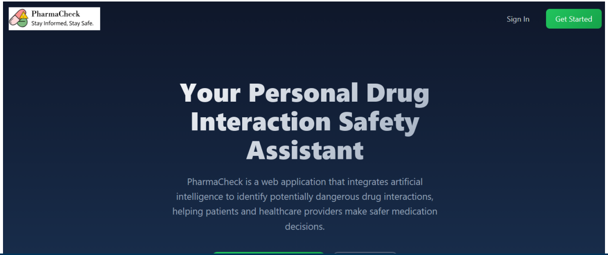
Your Personal Drug Interaction Safety Assistant

PharmaCheck is a web application that integrates artificial intelligence to identify potentially dangerous drug interactions, helping patients and healthcare providers make safer medication decisions.

User Interface Design

Modern, responsive web application

- Built with vanilla HTML, CSS, and JavaScript
- No heavy framework overhead
- Flask serves static files directly
- Responsive design for desktop and mobile
- Clean, medical-professional aesthetic



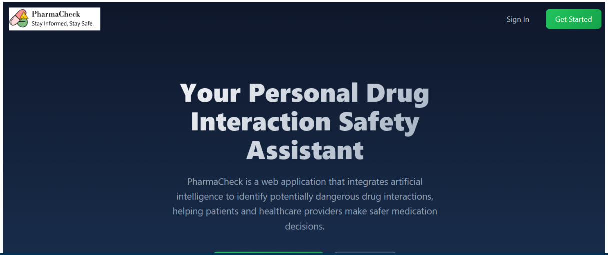
Your Personal Drug Interaction Safety Assistant

PharmaCheck is a web application that integrates artificial intelligence to identify potentially dangerous drug interactions, helping patients and healthcare providers make safer medication decisions.

User Interface Design

Modern, responsive web application

- Built with vanilla HTML, CSS, and JavaScript
- No heavy framework overhead
- Flask serves static files directly
- Responsive design for desktop and mobile
- Clean, medical-professional aesthetic



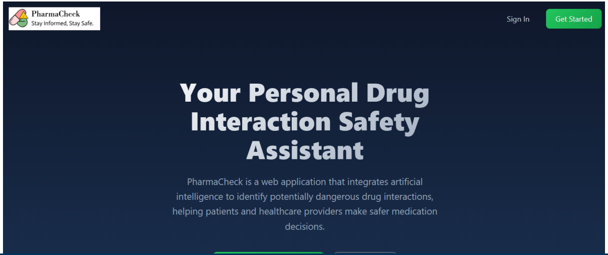
Your Personal Drug Interaction Safety Assistant

PharmaCheck is a web application that integrates artificial intelligence to identify potentially dangerous drug interactions, helping patients and healthcare providers make safer medication decisions.

User Interface Design

Modern, responsive web application

- Built with vanilla HTML, CSS, and JavaScript
- No heavy framework overhead
- Flask serves static files directly
- Responsive design for desktop and mobile
- Clean, medical-professional aesthetic



Your Personal Drug Interaction Safety Assistant

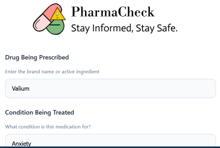
PharmaCheck is a web application that integrates artificial intelligence to identify potentially dangerous drug interactions, helping patients and healthcare providers make safer medication decisions.

Core Features

Drug interaction checking capabilities

- Multi-drug interaction checker (up to 5 drugs)
- Autocomplete drug search
- Severity-coded results (Major, Moderate, Minor)
- Expandable interaction details

- Food/lifestyle interaction checker
- Disease interaction checker
- Search history with clickable restoration
- AI translation on-demand


PharmaCheck
Stay Informed, Stay Safe.

Drug Being Prescribed
Enter the brand name or active ingredient
Valium

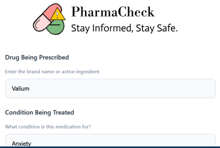
Condition Being Treated
What condition is this medication for?
Anxiety

Core Features

Drug interaction checking capabilities

- Multi-drug interaction checker (up to 5 drugs)
- Autocomplete drug search
- Severity-coded results (Major, Moderate, Minor)
- Expandable interaction details

- Food/lifestyle interaction checker
- Disease interaction checker
- Search history with clickable restoration
- AI translation on-demand



PharmaCheck
Stay Informed, Stay Safe.

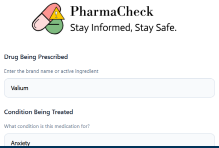
Drug Being Prescribed
Enter the brand name or active ingredient
Valium

Condition Being Treated
What condition is this medication for?
Anxiety

Core Features

Drug interaction checking capabilities

- Multi-drug interaction checker (up to 5 drugs)
- Autocomplete drug search
- Severity-coded results (Major, Moderate, Minor)
- Expandable interaction details
- Food/lifestyle interaction checker
- Disease interaction checker
- Search history with clickable restoration
- AI translation on-demand



PharmaCheck
Stay Informed, Stay Safe.

Drug Being Prescribed
Enter the brand name or active ingredient
Valium

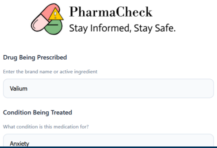
Condition Being Treated
What condition is this medication for?
Anxiety

Core Features

Drug interaction checking capabilities

- Multi-drug interaction checker (up to 5 drugs)
- Autocomplete drug search
- Severity-coded results (Major, Moderate, Minor)
- Expandable interaction details

- Food/lifestyle interaction checker
- Disease interaction checker
- Search history with clickable restoration
- AI translation on-demand



PharmaCheck
Stay Informed, Stay Safe.

Drug Being Prescribed
Enter the brand name or active ingredient
Valium

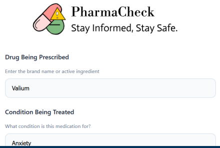
Condition Being Treated
What condition is this medication for?
Anxiety

Core Features

Drug interaction checking capabilities

- Multi-drug interaction checker (up to 5 drugs)
- Autocomplete drug search
- Severity-coded results (Major, Moderate, Minor)
- Expandable interaction details

- Food/lifestyle interaction checker
- Disease interaction checker
- Search history with clickable restoration
- AI translation on-demand



PharmaCheck
Stay Informed, Stay Safe.

Drug Being Prescribed
Enter the brand name or active ingredient
Valium

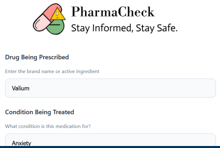
Condition Being Treated
What condition is this medication for?
Anxiety

Core Features

Drug interaction checking capabilities

- Multi-drug interaction checker (up to 5 drugs)
- Autocomplete drug search
- Severity-coded results (Major, Moderate, Minor)
- Expandable interaction details

- Food/lifestyle interaction checker
- Disease interaction checker
- Search history with clickable restoration
- AI translation on-demand



Drug Being Prescribed

Enter the brand name or active ingredient

Valium

Condition Being Treated

What condition is this medication for?

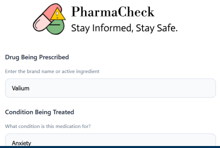
Anxiety

Core Features

Drug interaction checking capabilities

- Multi-drug interaction checker (up to 5 drugs)
- Autocomplete drug search
- Severity-coded results (Major, Moderate, Minor)
- Expandable interaction details

- Food/lifestyle interaction checker
- Disease interaction checker
- Search history with clickable restoration
- AI translation on-demand


PharmaCheck

Stay Informed, Stay Safe.

Drug Being Prescribed

Enter the brand name or active ingredient

Valium

Condition Being Treated

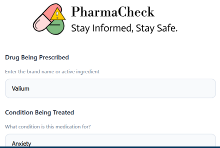
What condition is this medication for?

Anxiety

Core Features

Drug interaction checking capabilities

- Multi-drug interaction checker (up to 5 drugs)
- Autocomplete drug search
- Severity-coded results (Major, Moderate, Minor)
- Expandable interaction details
- Food/lifestyle interaction checker
- Disease interaction checker
- Search history with clickable restoration
- AI translation on-demand



PharmaCheck
Stay Informed, Stay Safe.

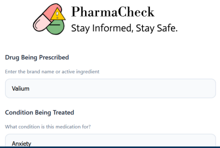
Drug Being Prescribed
Enter the brand name or active ingredient
Valium

Condition Being Treated
What condition is this medication for?
Anxiety

Core Features

Drug interaction checking capabilities

- Multi-drug interaction checker (up to 5 drugs)
- Autocomplete drug search
- Severity-coded results (Major, Moderate, Minor)
- Expandable interaction details
- Food/lifestyle interaction checker
- Disease interaction checker
- Search history with clickable restoration
- AI translation on-demand



PharmaCheck
Stay Informed, Stay Safe.

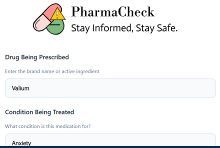
Drug Being Prescribed
Enter the brand name or active ingredient
Valium

Condition Being Treated
What condition is this medication for?
Anxiety

Core Features

Drug interaction checking capabilities

- Multi-drug interaction checker (up to 5 drugs)
- Autocomplete drug search
- Severity-coded results (Major, Moderate, Minor)
- Expandable interaction details
- Food/lifestyle interaction checker
- Disease interaction checker
- Search history with clickable restoration
- AI translation on-demand



PharmaCheck

Stay Informed, Stay Safe.

Drug Being Prescribed

Enter the brand name or active ingredient

Valium

Condition Being Treated

What condition is this medication for?

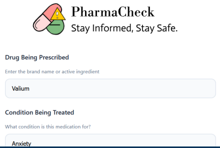
Anxiety

Core Features

Drug interaction checking capabilities

- Multi-drug interaction checker (up to 5 drugs)
- Autocomplete drug search
- Severity-coded results (Major, Moderate, Minor)
- Expandable interaction details

- Food/lifestyle interaction checker
- Disease interaction checker
- Search history with clickable restoration
- AI translation on-demand



PharmaCheck

Stay Informed, Stay Safe.

Drug Being Prescribed

Enter the brand name or active ingredient

Valium

Condition Being Treated

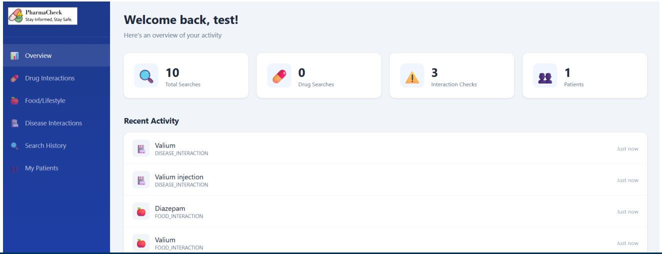
What condition is this medication for?

Anxiety

Doctor-Patient Features

Oversight and monitoring capabilities

- **Patient Registration**: Select a doctor during signup
- **Doctor Dashboard**: View all assigned patients
- **Search History Access**: Doctors can view patient search history
- **Real-time Tracking**: Recent searches displayed at a glance
- **Patient Control**: Patients can add/remove doctor assignments



Welcome back, test!

Here's an overview of your activity

10	Total Searches
0	Drug Searches
3	Interaction Checks
1	Patients

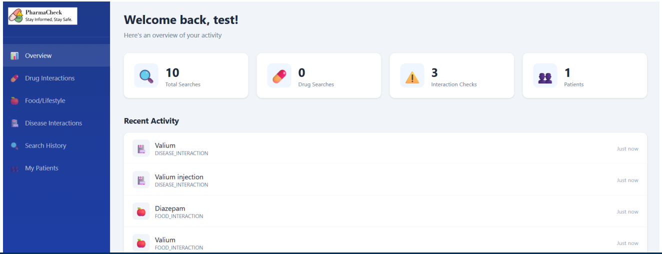
Recent Activity

Valium	DISEASE_INTERACTION	Just now
Valium injection	DISEASE_INTERACTION	Just now
Diazepam	FOOD_INTERACTION	Just now
Valium	FOOD_INTERACTION	Just now

Doctor-Patient Features

Oversight and monitoring capabilities

- **Patient Registration**: Select a doctor during signup
- **Doctor Dashboard**: View all assigned patients
- **Search History Access**: Doctors can view patient search history
- **Real-time Tracking**: Recent searches displayed at a glance
- **Patient Control**: Patients can add/remove doctor assignments



Welcome back, test!

Here's an overview of your activity

10	Total Searches
0	Drug Searches
3	Interaction Checks
1	Patients

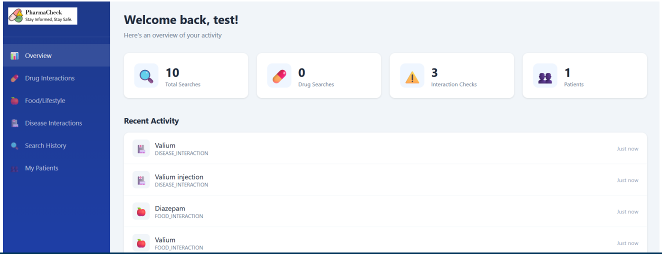
Recent Activity

Valium	DISEASE_INTERACTION	Just now
Valium injection	DISEASE_INTERACTION	Just now
Diazepam	FOOD_INTERACTION	Just now
Valium	FOOD_INTERACTION	Just now

Doctor-Patient Features

Oversight and monitoring capabilities

- **Patient Registration**: Select a doctor during signup
- **Doctor Dashboard**: View all assigned patients
- **Search History Access**: Doctors can view patient search history
- **Real-time Tracking**: Recent searches displayed at a glance
- **Patient Control**: Patients can add/remove doctor assignments



Welcome back, test!

Here's an overview of your activity

10 Total Searches

0 Drug Searches

3 Interaction Checks

1 Patients

Recent Activity

Valium
DISEASE_INTERACTION
Just now

Valium injection
DISEASE_INTERACTION
Just now

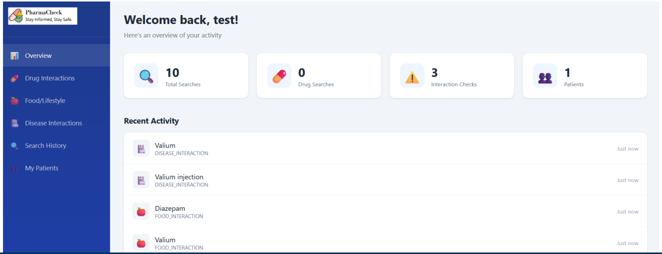
Diazepam
FOOD_INTERACTION
Just now

Valium
FOOD_INTERACTION
Just now

Doctor-Patient Features

Oversight and monitoring capabilities

- **Patient Registration**: Select a doctor during signup
- **Doctor Dashboard**: View all assigned patients
- **Search History Access**: Doctors can view patient search history
- **Real-time Tracking**: Recent searches displayed at a glance
- **Patient Control**: Patients can add/remove doctor assignments



Welcome back, test!

Here's an overview of your activity

10 Total Searches

0 Drug Searches

3 Interaction Checks

1 Patients

Recent Activity

Valium
DISEASE_INTERACTION
Just now

Valium injection
DISEASE_INTERACTION
Just now

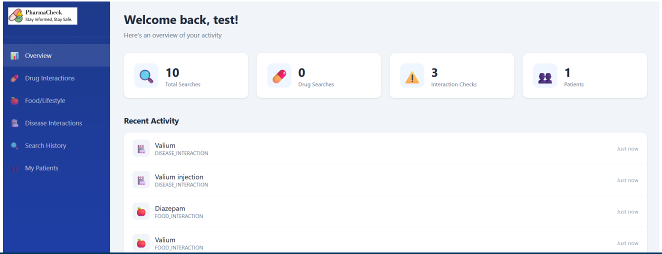
Diazepam
FOOD_INTERACTION
Just now

Valium
FOOD_INTERACTION
Just now

Doctor-Patient Features

Oversight and monitoring capabilities

- **Patient Registration**: Select a doctor during signup
- **Doctor Dashboard**: View all assigned patients
- **Search History Access**: Doctors can view patient search history
- **Real-time Tracking**: Recent searches displayed at a glance
- **Patient Control**: Patients can add/remove doctor assignments



Welcome back, test!

Here's an overview of your activity

10 Total Searches

0 Drug Searches

3 Interaction Checks

1 Patients

Recent Activity

Valium
DISEASE_INTERACTION
Just now

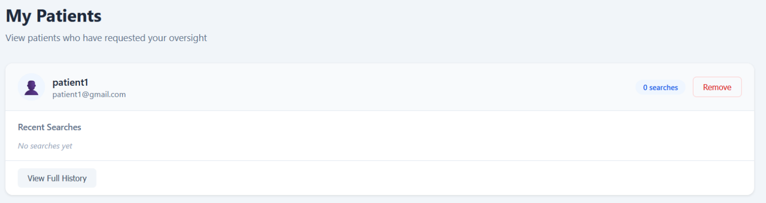
Valium injection
DISEASE_INTERACTION
Just now

Diazepam
FOOD_INTERACTION
Just now

Valium
FOOD_INTERACTION
Just now

Patient Management View

Doctor’s patient list interface



My Patients

View patients who have requested your oversight

patient1

patient1@gmail.com

Recent Searches

No searches yet

Live Demonstration

Live walkthrough of system features.

Demo Part 1: User Registration & Login

[LIVE DEMO]

Demonstrating:

- Creating a new patient account
- Selecting a doctor from dropdown
- Login authentication flow
- Dashboard navigation

Demo Part 1: User Registration & Login

[LIVE DEMO]

Demonstrating:

- Creating a new patient account
- Selecting a doctor from dropdown
- Login authentication flow
- Dashboard navigation

Demo Part 1: User Registration & Login

[LIVE DEMO]

Demonstrating:

- Creating a new patient account
- Selecting a doctor from dropdown
- Login authentication flow
- Dashboard navigation

Demo Part 1: User Registration & Login

[LIVE DEMO]

Demonstrating:

- Creating a new patient account
- Selecting a doctor from dropdown
- Login authentication flow
- Dashboard navigation

Demo Part 2: Drug Interaction Check

[LIVE DEMO]

Demonstrating:

- Checking interactions between Prozac and Valium
- Viewing severity levels (color-coded)
- Expanding interaction details
- Browsing through multiple interactions

Demo Part 2: Drug Interaction Check

[LIVE DEMO]

Demonstrating:

- Checking interactions between Prozac and Valium
- Viewing severity levels (color-coded)
- Expanding interaction details
- Browsing through multiple interactions

Demo Part 2: Drug Interaction Check

[LIVE DEMO]

Demonstrating:

- Checking interactions between Prozac and Valium
- Viewing severity levels (color-coded)
- Expanding interaction details
- Browsing through multiple interactions

Demo Part 2: Drug Interaction Check

[LIVE DEMO]

Demonstrating:

- Checking interactions between Prozac and Valium
- Viewing severity levels (color-coded)
- Expanding interaction details
- Browsing through multiple interactions

Demo Part 3: AI Translation

[LIVE DEMO]

Demonstrating:

- Clicking “Translate to Patient-Friendly” button
- Animated loading dots during translation
- Before/after description comparison
- Translation caching behavior

Demo Part 3: AI Translation

[LIVE DEMO]

Demonstrating:

- Clicking “Translate to Patient-Friendly” button
- Animated loading dots during translation
- Before/after description comparison
- Translation caching behavior

Demo Part 3: AI Translation

[LIVE DEMO]

Demonstrating:

- Clicking “Translate to Patient-Friendly” button
- Animated loading dots during translation
- Before/after description comparison
- Translation caching behavior

Demo Part 3: AI Translation

[LIVE DEMO]

Demonstrating:

- Clicking “Translate to Patient-Friendly” button
- Animated loading dots during translation
- Before/after description comparison
- Translation caching behavior

Demo Part 4: Food & Disease Interactions

[LIVE DEMO]

Demonstrating:

- Checking food/lifestyle interactions for Diazepam
- Checking disease interactions
- Brand name to generic name resolution
- Dashboard navigation between checkers

Demo Part 4: Food & Disease Interactions

[LIVE DEMO]

Demonstrating:

- Checking food/lifestyle interactions for Diazepam
- Checking disease interactions
- Brand name to generic name resolution
- Dashboard navigation between checkers

Demo Part 4: Food & Disease Interactions

[LIVE DEMO]

Demonstrating:

- Checking food/lifestyle interactions for Diazepam
- Checking disease interactions
- Brand name to generic name resolution
- Dashboard navigation between checkers

Demo Part 4: Food & Disease Interactions

[LIVE DEMO]

Demonstrating:

- Checking food/lifestyle interactions for Diazepam
- Checking disease interactions
- Brand name to generic name resolution
- Dashboard navigation between checkers

Demo Part 5: Doctor Dashboard

[LIVE DEMO]

Demonstrating:

- Logging in as a doctor account
- Viewing list of assigned patients
- Seeing patient’s recent searches at a glance
- Accessing full patient search history

Demo Part 5: Doctor Dashboard

[LIVE DEMO]

Demonstrating:

- Logging in as a doctor account
- Viewing list of assigned patients
- Seeing patient's recent searches at a glance
- Accessing full patient search history

Demo Part 5: Doctor Dashboard

[LIVE DEMO]

Demonstrating:

- Logging in as a doctor account
- Viewing list of assigned patients
- Seeing patient's recent searches at a glance
- Accessing full patient search history

Demo Part 5: Doctor Dashboard

[LIVE DEMO]

Demonstrating:

- Logging in as a doctor account
- Viewing list of assigned patients
- Seeing patient’s recent searches at a glance
- Accessing full patient search history

Challenges & Solutions

Technical, design, and UI/UX challenges encountered during development.

Technical Challenges

Backend and scraping issues

- **HTML Structure Changes**: Drugs.com format evolved during development
  - Solution: Robust parsing with multiple fallback selectors
- **Brand Name Resolution**: Prozac $\neq$ Fluoxetine in URLs
  - Solution: Implemented `_get_generic_name()` lookup
- **Performance**: Initial scraping took 2+ minutes
  - Solution: Caching mechanism with `use_cache=True`
- **Python Indentation Errors**: Inconsistent formatting
  - Solution: Careful code review and syntax checking

Technical Challenges

Backend and scraping issues

- **HTML Structure Changes**: Drugs.com format evolved during development
  - Solution: Robust parsing with multiple fallback selectors
- **Brand Name Resolution**: Prozac $\neq$ Fluoxetine in URLs
  - Solution: Implemented `_get_generic_name()` lookup
- **Performance**: Initial scraping took 2+ minutes
  - Solution: Caching mechanism with `use_cache=True`
- **Python Indentation Errors**: Inconsistent formatting
  - Solution: Careful code review and syntax checking

Technical Challenges

Backend and scraping issues

- **HTML Structure Changes**: Drugs.com format evolved during development
  - Solution: Robust parsing with multiple fallback selectors
- **Brand Name Resolution**: Prozac $\neq$ Fluoxetine in URLs
  - Solution: Implemented `_get_generic_name()` lookup
- **Performance**: Initial scraping took 2+ minutes
  - Solution: Caching mechanism with `use_cache=True`
- **Python Indentation Errors**: Inconsistent formatting
  - Solution: Careful code review and syntax checking

Technical Challenges

Backend and scraping issues

- **HTML Structure Changes**: Drugs.com format evolved during development
  - Solution: Robust parsing with multiple fallback selectors
- **Brand Name Resolution**: Prozac $\neq$ Fluoxetine in URLs
  - Solution: Implemented `_get_generic_name()` lookup
- **Performance**: Initial scraping took 2+ minutes
  - Solution: Caching mechanism with `use_cache=True`
- **Python Indentation Errors**: Inconsistent formatting
  - Solution: Careful code review and syntax checking

Technical Challenges

Backend and scraping issues

- **HTML Structure Changes**: Drugs.com format evolved during development
  - Solution: Robust parsing with multiple fallback selectors
- **Brand Name Resolution**: Prozac $\neq$ Fluoxetine in URLs
  - Solution: Implemented `_get_generic_name()` lookup
- **Performance**: Initial scraping took 2+ minutes
  - Solution: Caching mechanism with `use_cache=True`
- **Python Indentation Errors**: Inconsistent formatting
  - Solution: Careful code review and syntax checking

Technical Challenges

Backend and scraping issues

- **HTML Structure Changes**: Drugs.com format evolved during development
  - Solution: Robust parsing with multiple fallback selectors
- **Brand Name Resolution**: Prozac $\neq$ Fluoxetine in URLs
  - Solution: Implemented `_get_generic_name()` lookup
- **Performance**: Initial scraping took 2+ minutes
  - Solution: Caching mechanism with `use_cache=True`
- **Python Indentation Errors**: Inconsistent formatting
  - Solution: Careful code review and syntax checking

Technical Challenges

Backend and scraping issues

- **HTML Structure Changes**: Drugs.com format evolved during development
  - Solution: Robust parsing with multiple fallback selectors
- **Brand Name Resolution**: Prozac $\neq$ Fluoxetine in URLs
  - Solution: Implemented `_get_generic_name()` lookup
- **Performance**: Initial scraping took 2+ minutes
  - Solution: Caching mechanism with `use_cache=True`
- **Python Indentation Errors**: Inconsistent formatting
  - Solution: Careful code review and syntax checking

Technical Challenges

Backend and scraping issues

- **HTML Structure Changes**: Drugs.com format evolved during development
  - Solution: Robust parsing with multiple fallback selectors
- **Brand Name Resolution**: Prozac $\neq$ Fluoxetine in URLs
  - Solution: Implemented `_get_generic_name()` lookup
- **Performance**: Initial scraping took 2+ minutes
  - Solution: Caching mechanism with `use_cache=True`
- **Python Indentation Errors**: Inconsistent formatting
  - Solution: Careful code review and syntax checking

Design Challenges

Architecture and schema evolution

- **Schema Evolution**: Added FoodInteraction and DiseaseInteraction tables mid-development
- **Search History**: Implementing clickable restoration required storing full JSON results
- **Doctor-Patient Flow**: Original design had doctors adding patients
  - Reversed to: Patients request doctor oversight
- **File Protocol Issues**: Direct file:// serving broke navigation
  - Solution: Flask serves all static files via HTTP

Design Challenges

Architecture and schema evolution

- **Schema Evolution**: Added FoodInteraction and DiseaseInteraction tables mid-development
- **Search History**: Implementing clickable restoration required storing full JSON results
- **Doctor-Patient Flow**: Original design had doctors adding patients
  - Reversed to: Patients request doctor oversight
- **File Protocol Issues**: Direct file:// serving broke navigation
  - Solution: Flask serves all static files via HTTP

Design Challenges

Architecture and schema evolution

- **Schema Evolution**: Added FoodInteraction and DiseaseInteraction tables mid-development
- **Search History**: Implementing clickable restoration required storing full JSON results
- **Doctor-Patient Flow**: Original design had doctors adding patients
  - Reversed to: Patients request doctor oversight
- **File Protocol Issues**: Direct file:// serving broke navigation
  - Solution: Flask serves all static files via HTTP

Design Challenges

Architecture and schema evolution

- **Schema Evolution**: Added FoodInteraction and DiseaseInteraction tables mid-development
- **Search History**: Implementing clickable restoration required storing full JSON results
- **Doctor-Patient Flow**: Original design had doctors adding patients
  - Reversed to: Patients request doctor oversight
- **File Protocol Issues**: Direct file:// serving broke navigation
  - Solution: Flask serves all static files via HTTP

Design Challenges

Architecture and schema evolution

- **Schema Evolution**: Added FoodInteraction and DiseaseInteraction tables mid-development
- **Search History**: Implementing clickable restoration required storing full JSON results
- **Doctor-Patient Flow**: Original design had doctors adding patients
    - Reversed to: Patients request doctor oversight
- **File Protocol Issues**: Direct file:// serving broke navigation
    - Solution: Flask serves all static files via HTTP

Design Challenges

Architecture and schema evolution

- **Schema Evolution**: Added FoodInteraction and DiseaseInteraction tables mid-development
- **Search History**: Implementing clickable restoration required storing full JSON results
- **Doctor-Patient Flow**: Original design had doctors adding patients
    - Reversed to: Patients request doctor oversight
- **File Protocol Issues**: Direct file:// serving broke navigation
    - Solution: Flask serves all static files via HTTP

UI/UX Challenges

Frontend polish and user experience

- **Logo Display**: SVG logo appeared as white rectangle on some pages
  - Solution: Removed brightness/invert CSS filters
- **Dropdown State**: Cards collapsed after AI translation
  - Solution: Re-expand card after render using index tracking
- **Multi-Selector**: Comma-separated input was error-prone
  - Solution: Tag-based multi-selector with autocomplete
- **Loading States**: No feedback during long operations
  - Solution: Animated loading dots CSS animation

UI/UX Challenges

Frontend polish and user experience

- **Logo Display**: SVG logo appeared as white rectangle on some pages
  - Solution: Removed brightness/invert CSS filters
- **Dropdown State**: Cards collapsed after AI translation
  - Solution: Re-expand card after render using index tracking
- **Multi-Selector**: Comma-separated input was error-prone
  - Solution: Tag-based multi-selector with autocomplete
- **Loading States**: No feedback during long operations
  - Solution: Animated loading dots CSS animation

UI/UX Challenges

Frontend polish and user experience

- **Logo Display**: SVG logo appeared as white rectangle on some pages
  - Solution: Removed brightness/invert CSS filters
- **Dropdown State**: Cards collapsed after AI translation
  - Solution: Re-expand card after render using index tracking
- **Multi-Selector**: Comma-separated input was error-prone
  - Solution: Tag-based multi-selector with autocomplete
- **Loading States**: No feedback during long operations
  - Solution: Animated loading dots CSS animation

UI/UX Challenges

Frontend polish and user experience

- **Logo Display**: SVG logo appeared as white rectangle on some pages
  - Solution: Removed brightness/invert CSS filters
- **Dropdown State**: Cards collapsed after AI translation
  - Solution: Re-expand card after render using index tracking
- **Multi-Selector**: Comma-separated input was error-prone
  - Solution: Tag-based multi-selector with autocomplete
- **Loading States**: No feedback during long operations
  - Solution: Animated loading dots CSS animation

UI/UX Challenges

Frontend polish and user experience

- **Logo Display**: SVG logo appeared as white rectangle on some pages
  - Solution: Removed brightness/invert CSS filters
- **Dropdown State**: Cards collapsed after AI translation
  - Solution: Re-expand card after render using index tracking
- **Multi-Selector**: Comma-separated input was error-prone
  - Solution: Tag-based multi-selector with autocomplete
- **Loading States**: No feedback during long operations
  - Solution: Animated loading dots CSS animation

UI/UX Challenges

Frontend polish and user experience

- **Logo Display**: SVG logo appeared as white rectangle on some pages
  - Solution: Removed brightness/invert CSS filters
- **Dropdown State**: Cards collapsed after AI translation
  - Solution: Re-expand card after render using index tracking
- **Multi-Selector**: Comma-separated input was error-prone
  - Solution: Tag-based multi-selector with autocomplete
- **Loading States**: No feedback during long operations
  - Solution: Animated loading dots CSS animation

UI/UX Challenges

Frontend polish and user experience

- **Logo Display**: SVG logo appeared as white rectangle on some pages
  - Solution: Removed brightness/invert CSS filters
- **Dropdown State**: Cards collapsed after AI translation
  - Solution: Re-expand card after render using index tracking
- **Multi-Selector**: Comma-separated input was error-prone
  - Solution: Tag-based multi-selector with autocomplete
- **Loading States**: No feedback during long operations
  - Solution: Animated loading dots CSS animation

UI/UX Challenges

Frontend polish and user experience

- **Logo Display**: SVG logo appeared as white rectangle on some pages
    - Solution: Removed brightness/invert CSS filters
- **Dropdown State**: Cards collapsed after AI translation
    - Solution: Re-expand card after render using index tracking
- **Multi-Selector**: Comma-separated input was error-prone
    - Solution: Tag-based multi-selector with autocomplete
- **Loading States**: No feedback during long operations
    - Solution: Animated loading dots CSS animation

Beta I: First Working System

First working system achievements and metrics.

Beta I Achievements

First working version milestones

- Fully functional MySQL database with 8+ tables
- Complete web scraping pipeline
- Working AI translation feature
- JWT-based authentication system
- Role-based access control
- Doctor-patient relationship management
- Three interaction checkers (Drug, Food, Disease)
- Search history with restoration
- Responsive web interface
- Real-time data from Drugs.com

Beta I Achievements

First working version milestones

- Fully functional MySQL database with 8+ tables
- Complete web scraping pipeline
- Working AI translation feature
- JWT-based authentication system
- Role-based access control

- Doctor-patient relationship management
- Three interaction checkers (Drug, Food, Disease)
- Search history with restoration
- Responsive web interface
- Real-time data from Drugs.com

Beta I Achievements

First working version milestones

- Fully functional MySQL database with 8+ tables
- Complete web scraping pipeline
- Working AI translation feature
- JWT-based authentication system
- Role-based access control
- Doctor-patient relationship management
- Three interaction checkers (Drug, Food, Disease)
- Search history with restoration
- Responsive web interface
- Real-time data from Drugs.com

Beta I Achievements

First working version milestones

- Fully functional MySQL database with 8+ tables
- Complete web scraping pipeline
- Working AI translation feature
- JWT-based authentication system
- Role-based access control
- Doctor-patient relationship management
- Three interaction checkers (Drug, Food, Disease)
- Search history with restoration
- Responsive web interface
- Real-time data from Drugs.com

Beta I Achievements

First working version milestones

- Fully functional MySQL database with 8+ tables
- Complete web scraping pipeline
- Working AI translation feature
- JWT-based authentication system
- Role-based access control
- Doctor-patient relationship management
- Three interaction checkers (Drug, Food, Disease)
- Search history with restoration
- Responsive web interface
- Real-time data from Drugs.com

Beta I Achievements

First working version milestones

- Fully functional MySQL database with 8+ tables
- Complete web scraping pipeline
- Working AI translation feature
- JWT-based authentication system
- Role-based access control

- Doctor-patient relationship management
- Three interaction checkers (Drug, Food, Disease)
- Search history with restoration
- Responsive web interface
- Real-time data from Drugs.com

Beta I Achievements

First working version milestones

- Fully functional MySQL database with 8+ tables
- Complete web scraping pipeline
- Working AI translation feature
- JWT-based authentication system
- Role-based access control

- Doctor-patient relationship management
- Three interaction checkers (Drug, Food, Disease)
- Search history with restoration
- Responsive web interface
- Real-time data from Drugs.com

Beta I Achievements

First working version milestones

- Fully functional MySQL database with 8+ tables
- Complete web scraping pipeline
- Working AI translation feature
- JWT-based authentication system
- Role-based access control

- Doctor-patient relationship management
- Three interaction checkers (Drug, Food, Disease)
- Search history with restoration
- Responsive web interface
- Real-time data from Drugs.com

Beta I Achievements

First working version milestones

- Fully functional MySQL database with 8+ tables
- Complete web scraping pipeline
- Working AI translation feature
- JWT-based authentication system
- Role-based access control
- Doctor-patient relationship management
- Three interaction checkers (Drug, Food, Disease)
- Search history with restoration
- Responsive web interface
- Real-time data from Drugs.com

Beta I Achievements

First working version milestones

- Fully functional MySQL database with 8+ tables
- Complete web scraping pipeline
- Working AI translation feature
- JWT-based authentication system
- Role-based access control

- Doctor-patient relationship management
- Three interaction checkers (Drug, Food, Disease)
- Search history with restoration
- Responsive web interface
- Real-time data from Drugs.com

Beta I Achievements

First working version milestones

- Fully functional MySQL database with 8+ tables
- Complete web scraping pipeline
- Working AI translation feature
- JWT-based authentication system
- Role-based access control
- Doctor-patient relationship management
- Three interaction checkers (Drug, Food, Disease)
- Search history with restoration
- Responsive web interface
- Real-time data from Drugs.com

Beta I Achievements

First working version milestones

- Fully functional MySQL database with 8+ tables
- Complete web scraping pipeline
- Working AI translation feature
- JWT-based authentication system
- Role-based access control
- Doctor-patient relationship management
- Three interaction checkers (Drug, Food, Disease)
- Search history with restoration
- Responsive web interface
- Real-time data from Drugs.com

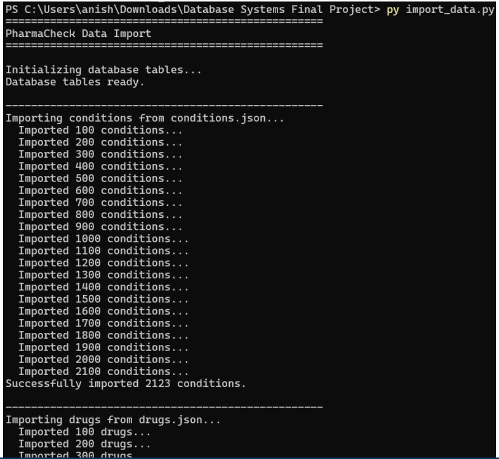
Beta I Achievements

First working version milestones

- Fully functional MySQL database with 8+ tables
- Complete web scraping pipeline
- Working AI translation feature
- JWT-based authentication system
- Role-based access control
- Doctor-patient relationship management
- Three interaction checkers (Drug, Food, Disease)
- Search history with restoration
- Responsive web interface
- Real-time data from Drugs.com

Data Import Process

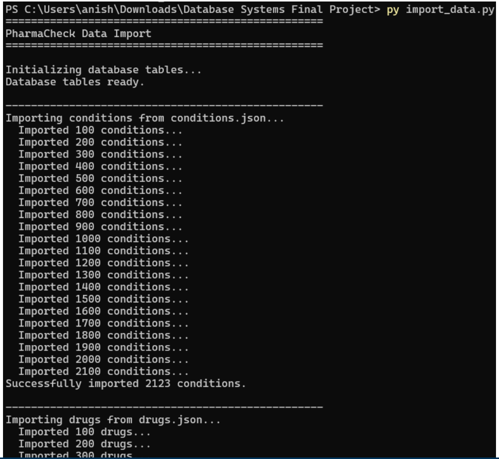
Populating the database with drugs and conditions



```
PS C:\Users\anish\Downloads\Database Systems Final Project> py import_data.py
==================================================
PharmaCheck Data Import
==================================================

Initializing database tables...
Database tables ready.

--------------------------------------------------
Importing conditions from conditions.json...
  Imported 100 conditions...
  Imported 200 conditions...
  Imported 300 conditions...
  Imported 400 conditions...
  Imported 500 conditions...
  Imported 600 conditions...
  Imported 700 conditions...
  Imported 800 conditions...
  Imported 900 conditions...
  Imported 1000 conditions...
  Imported 1100 conditions...
  Imported 1200 conditions...
  Imported 1300 conditions...
  Imported 1400 conditions...
  Imported 1500 conditions...
  Imported 1600 conditions...
  Imported 1700 conditions...
  Imported 1800 conditions...
  Imported 1900 conditions...
  Imported 2000 conditions...
  Imported 2100 conditions...
Successfully imported 2123 conditions.

--------------------------------------------------
Importing drugs from drugs.json...
  Imported 100 drugs...
  Imported 200 drugs...
  Imported 300 drugs
```

Data Import Process

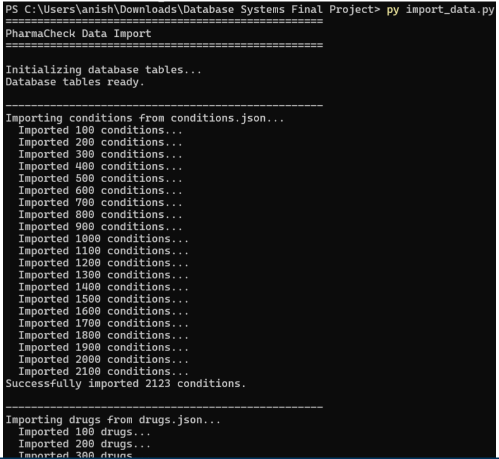
Populating the database with drugs and conditions



```
PS C:\Users\anish\Downloads\Database Systems Final Project> py import_data.py
==================================================
PharmaCheck Data Import
==================================================

Initializing database tables...
Database tables ready.

--------------------------------------------------
Importing conditions from conditions.json...
  Imported 100 conditions...
  Imported 200 conditions...
  Imported 300 conditions...
  Imported 400 conditions...
  Imported 500 conditions...
  Imported 600 conditions...
  Imported 700 conditions...
  Imported 800 conditions...
  Imported 900 conditions...
  Imported 1000 conditions...
  Imported 1100 conditions...
  Imported 1200 conditions...
  Imported 1300 conditions...
  Imported 1400 conditions...
  Imported 1500 conditions...
  Imported 1600 conditions...
  Imported 1700 conditions...
  Imported 1800 conditions...
  Imported 1900 conditions...
  Imported 2000 conditions...
  Imported 2100 conditions...
Successfully imported 2123 conditions.

--------------------------------------------------
Importing drugs from drugs.json...
  Imported 100 drugs...
  Imported 200 drugs...
  Imported 300 drugs
```

Data Import Process

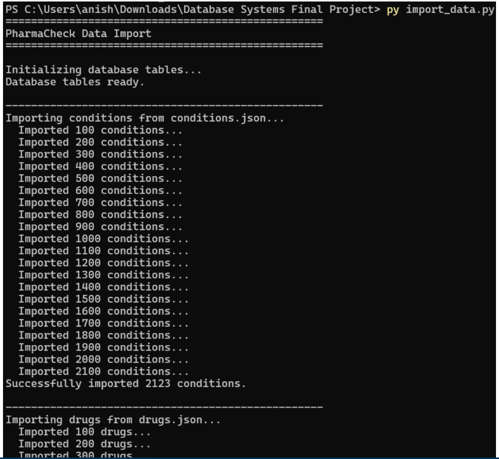
Populating the database with drugs and conditions



```
PS C:\Users\anish\Downloads\Database Systems Final Project> py import_data.py
==================================================
PharmaCheck Data Import
==================================================

Initializing database tables...
Database tables ready.

--------------------------------------------------
Importing conditions from conditions.json...
  Imported 100 conditions...
  Imported 200 conditions...
  Imported 300 conditions...
  Imported 400 conditions...
  Imported 500 conditions...
  Imported 600 conditions...
  Imported 700 conditions...
  Imported 800 conditions...
  Imported 900 conditions...
  Imported 1000 conditions...
  Imported 1100 conditions...
  Imported 1200 conditions...
  Imported 1300 conditions...
  Imported 1400 conditions...
  Imported 1500 conditions...
  Imported 1600 conditions...
  Imported 1700 conditions...
  Imported 1800 conditions...
  Imported 1900 conditions...
  Imported 2000 conditions...
  Imported 2100 conditions...
Successfully imported 2123 conditions.

--------------------------------------------------
Importing drugs from drugs.json...
  Imported 100 drugs...
  Imported 200 drugs...
  Imported 300 drugs
```

Data Import Process

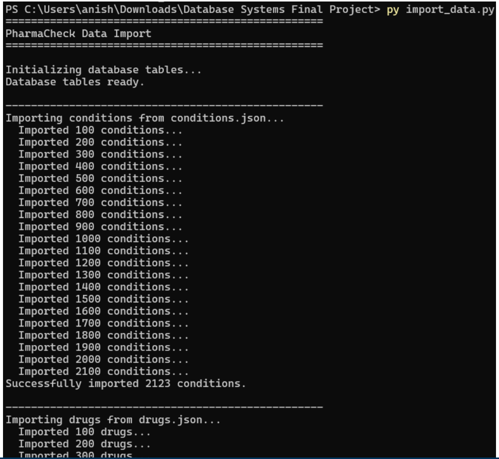
Populating the database with drugs and conditions



```
PS C:\Users\anish\Downloads\Database Systems Final Project> py import_data.py
==================================================
PharmaCheck Data Import
==================================================

Initializing database tables...
Database tables ready.

--------------------------------------------------
Importing conditions from conditions.json...
  Imported 100 conditions...
  Imported 200 conditions...
  Imported 300 conditions...
  Imported 400 conditions...
  Imported 500 conditions...
  Imported 600 conditions...
  Imported 700 conditions...
  Imported 800 conditions...
  Imported 900 conditions...
  Imported 1000 conditions...
  Imported 1100 conditions...
  Imported 1200 conditions...
  Imported 1300 conditions...
  Imported 1400 conditions...
  Imported 1500 conditions...
  Imported 1600 conditions...
  Imported 1700 conditions...
  Imported 1800 conditions...
  Imported 1900 conditions...
  Imported 2000 conditions...
  Imported 2100 conditions...
Successfully imported 2123 conditions.

--------------------------------------------------
Importing drugs from drugs.json...
  Imported 100 drugs...
  Imported 200 drugs...
  Imported 300 drugs
```

Data Import Process

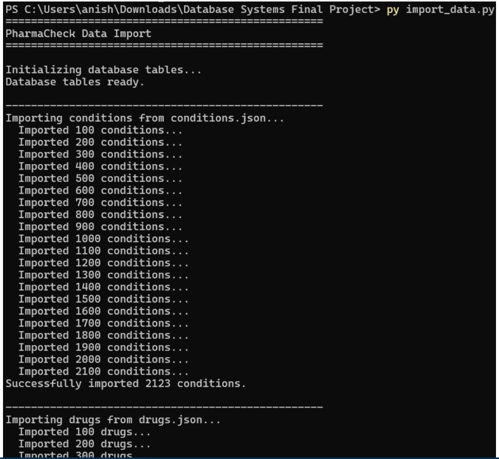
Populating the database with drugs and conditions



```
PS C:\Users\anish\Downloads\Database Systems Final Project> py import_data.py
==================================================
PharmaCheck Data Import
==================================================

Initializing database tables...
Database tables ready.

--------------------------------------------------
Importing conditions from conditions.json...
  Imported 100 conditions...
  Imported 200 conditions...
  Imported 300 conditions...
  Imported 400 conditions...
  Imported 500 conditions...
  Imported 600 conditions...
  Imported 700 conditions...
  Imported 800 conditions...
  Imported 900 conditions...
  Imported 1000 conditions...
  Imported 1100 conditions...
  Imported 1200 conditions...
  Imported 1300 conditions...
  Imported 1400 conditions...
  Imported 1500 conditions...
  Imported 1600 conditions...
  Imported 1700 conditions...
  Imported 1800 conditions...
  Imported 1900 conditions...
  Imported 2000 conditions...
  Imported 2100 conditions...
Successfully imported 2123 conditions.

--------------------------------------------------
Importing drugs from drugs.json...
  Imported 100 drugs...
  Imported 200 drugs...
  Imported 300 drugs
```

Data Import Process

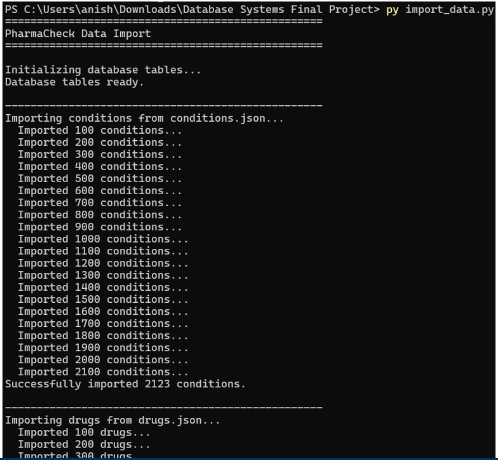
Populating the database with drugs and conditions



```
PS C:\Users\anish\Downloads\Database Systems Final Project> py import_data.py
==================================================
PharmaCheck Data Import
==================================================

Initializing database tables...
Database tables ready.

--------------------------------------------------
Importing conditions from conditions.json...
  Imported 100 conditions...
  Imported 200 conditions...
  Imported 300 conditions...
  Imported 400 conditions...
  Imported 500 conditions...
  Imported 600 conditions...
  Imported 700 conditions...
  Imported 800 conditions...
  Imported 900 conditions...
  Imported 1000 conditions...
  Imported 1100 conditions...
  Imported 1200 conditions...
  Imported 1300 conditions...
  Imported 1400 conditions...
  Imported 1500 conditions...
  Imported 1600 conditions...
  Imported 1700 conditions...
  Imported 1800 conditions...
  Imported 1900 conditions...
  Imported 2000 conditions...
  Imported 2100 conditions...
Successfully imported 2123 conditions.

--------------------------------------------------
Importing drugs from drugs.json...
  Imported 100 drugs...
  Imported 200 drugs...
  Imported 300 drugs
```

Data Import Process

Populating the database with drugs and conditions



```
PS C:\Users\anish\Downloads\Database Systems Final Project> py import_data.py
==================================================
PharmaCheck Data Import
==================================================

Initializing database tables...
Database tables ready.

--------------------------------------------------
Importing conditions from conditions.json...
  Imported 100 conditions...
  Imported 200 conditions...
  Imported 300 conditions...
  Imported 400 conditions...
  Imported 500 conditions...
  Imported 600 conditions...
  Imported 700 conditions...
  Imported 800 conditions...
  Imported 900 conditions...
  Imported 1000 conditions...
  Imported 1100 conditions...
  Imported 1200 conditions...
  Imported 1300 conditions...
  Imported 1400 conditions...
  Imported 1500 conditions...
  Imported 1600 conditions...
  Imported 1700 conditions...
  Imported 1800 conditions...
  Imported 1900 conditions...
  Imported 2000 conditions...
  Imported 2100 conditions...
Successfully imported 2123 conditions.

--------------------------------------------------
Importing drugs from drugs.json...
  Imported 100 drugs...
  Imported 200 drugs...
  Imported 300 drugs
```

Beta I Metrics

Project statistics

Codebase Size

- **api.py**: 1,115 lines
- **scraper.py**: 927 lines
- **database.py**: 351 lines
- **auth.py**: 303 lines
- Frontend: 1,500+ lines (HTML/CSS/JS)

Data Coverage

- **15,775** drugs in database
- **2,126** medical conditions
- Real-time access to all Drugs.com interactions
- Support for food and disease interactions

Beta I Metrics

Project statistics

Codebase Size

- **api.py**: 1,115 lines
- **scraper.py**: 927 lines
- **database.py**: 351 lines
- **auth.py**: 303 lines
- Frontend: 1,500+ lines (HTML/CSS/JS)

Data Coverage

- **15,775** drugs in database
- **2,126** medical conditions
- Real-time access to all Drugs.com interactions
- Support for food and disease interactions

Beta I Metrics

Project statistics

Codebase Size

- **api.py**: 1,115 lines
- **scraper.py**: 927 lines
- **database.py**: 351 lines
- **auth.py**: 303 lines
- Frontend: 1,500+ lines (HTML/CSS/JS)

Data Coverage

- **15,775** drugs in database
- **2,126** medical conditions
- Real-time access to all Drugs.com interactions
- Support for food and disease interactions

Beta I Metrics

Project statistics

Codebase Size

- **api.py**: 1,115 lines
- **scraper.py**: 927 lines
- **database.py**: 351 lines
- **auth.py**: 303 lines
- Frontend: 1,500+ lines (HTML/CSS/JS)

Data Coverage

- **15,775** drugs in database
- **2,126** medical conditions
- Real-time access to all Drugs.com interactions
- Support for food and disease interactions

Beta I Metrics

Project statistics

Codebase Size

- **api.py**: 1,115 lines
- **scraper.py**: 927 lines
- **database.py**: 351 lines
- **auth.py**: 303 lines
- Frontend: 1,500+ lines (HTML/CSS/JS)

Data Coverage

- **15,775** drugs in database
- **2,126** medical conditions
- Real-time access to all Drugs.com interactions
- Support for food and disease interactions

Beta I Metrics

Project statistics

Codebase Size

- **api.py**: 1,115 lines
- **scraper.py**: 927 lines
- **database.py**: 351 lines
- **auth.py**: 303 lines
- Frontend: 1,500+ lines (HTML/CSS/JS)

Data Coverage

- **15,775** drugs in database
- **2,126** medical conditions
- Real-time access to all Drugs.com interactions
- Support for food and disease interactions

Beta I Metrics

Project statistics

Codebase Size

- **api.py**: 1,115 lines
- **scraper.py**: 927 lines
- **database.py**: 351 lines
- **auth.py**: 303 lines
- Frontend: 1,500+ lines (HTML/CSS/JS)

Data Coverage

- **15,775** drugs in database
- **2,126** medical conditions
- Real-time access to all Drugs.com interactions
- Support for food and disease interactions

Beta I Metrics

Project statistics

Codebase Size

- **api.py**: 1,115 lines
- **scraper.py**: 927 lines
- **database.py**: 351 lines
- **auth.py**: 303 lines
- Frontend: 1,500+ lines (HTML/CSS/JS)

Data Coverage

- **15,775** drugs in database
- **2,126** medical conditions
- Real-time access to all Drugs.com interactions
- Support for food and disease interactions

Beta I Metrics

Project statistics

Codebase Size

- **api.py**: 1,115 lines
- **scraper.py**: 927 lines
- **database.py**: 351 lines
- **auth.py**: 303 lines
- Frontend: 1,500+ lines (HTML/CSS/JS)

Data Coverage

- **15,775** drugs in database
- **2,126** medical conditions
- Real-time access to all Drugs.com interactions
- Support for food and disease interactions

Beta I Metrics

Project statistics

Codebase Size

- **api.py**: 1,115 lines
- **scraper.py**: 927 lines
- **database.py**: 351 lines
- **auth.py**: 303 lines
- Frontend: 1,500+ lines (HTML/CSS/JS)

Data Coverage

- **15,775** drugs in database
- **2,126** medical conditions
- Real-time access to all Drugs.com interactions
- Support for food and disease interactions

Beta I Metrics

Project statistics

Codebase Size

- **api.py**: 1,115 lines
- **scraper.py**: 927 lines
- **database.py**: 351 lines
- **auth.py**: 303 lines
- Frontend: 1,500+ lines (HTML/CSS/JS)

Data Coverage

- **15,775** drugs in database
- **2,126** medical conditions
- Real-time access to all Drugs.com interactions
- Support for food and disease interactions

Beta I Metrics

Project statistics

Codebase Size

- **api.py**: 1,115 lines
- **scraper.py**: 927 lines
- **database.py**: 351 lines
- **auth.py**: 303 lines
- Frontend: 1,500+ lines (HTML/CSS/JS)

Data Coverage

- **15,775** drugs in database
- **2,126** medical conditions
- Real-time access to all Drugs.com interactions
- Support for food and disease interactions

Future Work

Planned enhancements and technical improvements.

Planned Enhancements

Feature roadmap

- **Doctor-Patient Communication**: Direct messaging system
- **Medication Reminders**: Scheduled notifications
- **Prescription Management**: Track current prescriptions
- **Drug Allergy Tracking**: Alert on known allergens
- **Enhanced Patient Profiles**: Medical history, demographics
- **Doctor Notes**: Add annotations to patient interactions

Planned Enhancements

Feature roadmap

- **Doctor-Patient Communication**: Direct messaging system
- **Medication Reminders**: Scheduled notifications
- **Prescription Management**: Track current prescriptions
- **Drug Allergy Tracking**: Alert on known allergens
- **Enhanced Patient Profiles**: Medical history, demographics
- **Doctor Notes**: Add annotations to patient interactions

Planned Enhancements

Feature roadmap

- **Doctor-Patient Communication**: Direct messaging system
- **Medication Reminders**: Scheduled notifications
- **Prescription Management**: Track current prescriptions
- **Drug Allergy Tracking**: Alert on known allergens
- **Enhanced Patient Profiles**: Medical history, demographics
- **Doctor Notes**: Add annotations to patient interactions

Planned Enhancements

Feature roadmap

- **Doctor-Patient Communication**: Direct messaging system
- **Medication Reminders**: Scheduled notifications
- **Prescription Management**: Track current prescriptions
- **Drug Allergy Tracking**: Alert on known allergens
- **Enhanced Patient Profiles**: Medical history, demographics
- **Doctor Notes**: Add annotations to patient interactions

Planned Enhancements

Feature roadmap

- **Doctor-Patient Communication**: Direct messaging system
- **Medication Reminders**: Scheduled notifications
- **Prescription Management**: Track current prescriptions
- **Drug Allergy Tracking**: Alert on known allergens
- **Enhanced Patient Profiles**: Medical history, demographics
- **Doctor Notes**: Add annotations to patient interactions

Planned Enhancements

Feature roadmap

- **Doctor-Patient Communication**: Direct messaging system
- **Medication Reminders**: Scheduled notifications
- **Prescription Management**: Track current prescriptions
- **Drug Allergy Tracking**: Alert on known allergens
- **Enhanced Patient Profiles**: Medical history, demographics
- **Doctor Notes**: Add annotations to patient interactions

Technical Improvements

Infrastructure and performance

- **Background Job Queue**: Async scraping with Celery
- **Full-Text Search**: Elasticsearch integration
- **Mobile Application**: React Native or Flutter app
- **PDF Reports**: Export search history as reports
- **Pharmacy Integration**: Connect with pharmacy systems
- **Analytics Dashboard**: Patient adherence tracking for doctors

Technical Improvements

Infrastructure and performance

- **Background Job Queue**: Async scraping with Celery
- **Full-Text Search**: Elasticsearch integration
- **Mobile Application**: React Native or Flutter app
- **PDF Reports**: Export search history as reports
- **Pharmacy Integration**: Connect with pharmacy systems
- **Analytics Dashboard**: Patient adherence tracking for doctors

Technical Improvements

Infrastructure and performance

- **Background Job Queue**: Async scraping with Celery
- **Full-Text Search**: Elasticsearch integration
- **Mobile Application**: React Native or Flutter app
- **PDF Reports**: Export search history as reports
- **Pharmacy Integration**: Connect with pharmacy systems
- **Analytics Dashboard**: Patient adherence tracking for doctors

Technical Improvements

Infrastructure and performance

- **Background Job Queue**: Async scraping with Celery
- **Full-Text Search**: Elasticsearch integration
- **Mobile Application**: React Native or Flutter app
- **PDF Reports**: Export search history as reports
- **Pharmacy Integration**: Connect with pharmacy systems
- **Analytics Dashboard**: Patient adherence tracking for doctors

Technical Improvements

Infrastructure and performance

- **Background Job Queue**: Async scraping with Celery
- **Full-Text Search**: Elasticsearch integration
- **Mobile Application**: React Native or Flutter app
- **PDF Reports**: Export search history as reports
- **Pharmacy Integration**: Connect with pharmacy systems
- **Analytics Dashboard**: Patient adherence tracking for doctors

Technical Improvements

Infrastructure and performance

- **Background Job Queue**: Async scraping with Celery
- **Full-Text Search**: Elasticsearch integration
- **Mobile Application**: React Native or Flutter app
- **PDF Reports**: Export search history as reports
- **Pharmacy Integration**: Connect with pharmacy systems
- **Analytics Dashboard**: Patient adherence tracking for doctors

Conclusion

Project summary

- Successfully built an end-to-end drug interaction checking system
- Implements real-time web scraping with intelligent caching
- AI-powered translation makes medical information accessible
- Doctor-patient features enable professional oversight
- Personal goal achieved: Building a tool I would have wanted during treatment
- Potential for real-world impact in patient medication safety

Thank you!

Conclusion

Project summary

- Successfully built an end-to-end drug interaction checking system
- Implements real-time web scraping with intelligent caching
- AI-powered translation makes medical information accessible
- Doctor-patient features enable professional oversight
- Personal goal achieved: Building a tool I would have wanted during treatment
- Potential for real-world impact in patient medication safety

Thank you!

Conclusion

Project summary

- Successfully built an end-to-end drug interaction checking system
- Implements real-time web scraping with intelligent caching
- AI-powered translation makes medical information accessible
- Doctor-patient features enable professional oversight
- Personal goal achieved: Building a tool I would have wanted during treatment
- Potential for real-world impact in patient medication safety

Thank you!

Conclusion

Project summary

- Successfully built an end-to-end drug interaction checking system
- Implements real-time web scraping with intelligent caching
- AI-powered translation makes medical information accessible
- Doctor-patient features enable professional oversight
- Personal goal achieved: Building a tool I would have wanted during treatment
- Potential for real-world impact in patient medication safety

Thank you!

Conclusion

Project summary

- Successfully built an end-to-end drug interaction checking system
- Implements real-time web scraping with intelligent caching
- AI-powered translation makes medical information accessible
- Doctor-patient features enable professional oversight
- Personal goal achieved: Building a tool I would have wanted during treatment
- Potential for real-world impact in patient medication safety

Thank you!

Conclusion

Project summary

- Successfully built an end-to-end drug interaction checking system
- Implements real-time web scraping with intelligent caching
- AI-powered translation makes medical information accessible
- Doctor-patient features enable professional oversight
- Personal goal achieved: Building a tool I would have wanted during treatment
- Potential for real-world impact in patient medication safety

Thank you!

Complete System Demonstration

Full walkthrough

[LIVE DEMO]

Complete walkthrough of PharmaCheck

`http://localhost:5000`

References

[1] R. Ramakrishnan and J. Gehrke, *Database management systems*, 3rd Edition. McGraw-Hill, 2003.

[2] Drugs.com, Drugs.com drug interaction checker. 2024.

[3] Ollama, Ollama: Run large language models locally. 2024.

[4] Pallets Projects, Flask: A python microframework. 2024.

[5] SQLAlchemy, SQLAlchemy: The python SQL toolkit and ORM. 2024.

GEORGIA SOUTHERN UNIVERSITY

PharmaCheck

Drug Interaction Database System

Thank you for your attention!

Anish Goyal

Department of Computer Science
Georgia Southern University
Statesboro, GA

December 4, 2025