Medicinal Drug Search & Inventory Management System

1. Project Overview

This project is a **Flask-based web application** designed to digitize the management and retrieval of pharmaceutical drug information. The system allows users to search for drugs and view detailed information including ingredients, manufacturers, and specific store availability. It leverages a **Microsoft SQL Server (MSSQL)** database for robust data persistence.

2. Technology Stack

Component	Technology	Description
Backend	Python (Flask)	Lightweight WSGI web application framework.
Database	Microsoft SQL Server	Relational database management system.
ORM	SQLAlchemy	Python SQL toolkit and Object Relational Mapper.
Frontend	HTML, CSS, Jinja2	Server-side templating for dynamic views.
Driver	PyODBC	DB API module for connecting to SQL Server.

3. Project Structure

The application follows a modular package structure:

```
medicinal_drug_search/
├── app/                  # Main Application Package
│   ├── __init__.py           # App factory, DB initialization, config loading
│   ├── models.py             # Database Schema (SQLAlchemy Models)
│   └── routes.py             # View functions and URL routing
├── static/               # Static assets
│   └── style.css             # Application styling
├── templates/            # Jinja2 HTML Templates
│   ├── index.html            # Landing page
│   ├── search.html           # Search input form
│   ├── search_results.html   # Search results listing
│   └── drug_detail.html      # Comprehensive drug view
├── test/                 # Testing Scripts
│   ├── test.py               # DB connection verification
│   └── d.py                  # Data processing (CSV cleaning)
├── config.py             # Configuration settings (Keys, DB connection)
├── run.py                # Application entry point
└── requirements.txt      # Python dependencies
```

Key File Descriptions

- **config.py**: Handles the connection string to the MSSQL database (Sample1 on DESKTOP-6ET4EQU\MSSQLSERVER01) and Flask security keys.
- **app/__init__.py**: Initializes the Flask app context and SQLAlchemy engine. It is responsible for creating database tables if they do not exist on startup.
- **app/models.py**: Contains the class definitions for the database tables.
- **app/routes.py**: Manages the HTTP requests for Home, Search, and Detail views.

4. Database Schema & Data Modeling

The system uses a relational database model designed to handle complex pharmaceutical relationships.

Core Models (models.py)

1. **User**: Stores authentication details (Username, Email, Password).
2. **Drug**: The central entity containing Name, Description, Mfg/Exp Dates, and Cost.
3. **Ingredient**: Lists individual chemical components.
4. **PharmaCompany**: Stores manufacturer details.
5. **Store**: Represents physical pharmacy locations with GPS coordinates (Lat/Long).
6. **Inventory**: Manages stock levels of specific drugs at specific stores.
7. **UserDrug**: Analytics table tracking which user searched for which drug.

Key Relationships

- **Drug ↔ Ingredient (Many-to-Many)**: Managed via the DrugIngredient junction table. One drug can have multiple ingredients; one ingredient can be in multiple drugs.
- **Drug ↔ Store (Many-to-Many)**: Managed via the Inventory table. Tracks stock quantity per location.
- **Drug ↔ PharmaCompany (Many-to-One)**: Each drug is linked to a specific manufacturer.

5. Application Features

Drug Search

- Users can perform case-insensitive searches for drugs.
- The backend filters results based on drug name prefixes using ILIKE queries.

Detailed Drug Information

- **Overview**: Displays description, price, and manufacturing/expiry dates.
- **Composition**: Lists all active ingredients and their quantities.
- **Manufacturer**: Shows the pharmaceutical company responsible for the drug.

Store Locator & Inventory

- Identifies which stores currently hold stock of the searched drug.
- Provides precise GPS coordinates (Latitude/Longitude) for store locations.

Analytics

- **Search Tracking**: The system records every search query made by a user into the UserDrug table, allowing for analysis of popular medications.

6. Installation & Configuration

Prerequisites

- Python 3.x
- Microsoft SQL Server (MSSQL)
- ODBC Driver for SQL Server

Setup Steps

1. **Database Configuration**:
   - Open config.py.
   - Update the SQLALCHEMY_DATABASE_URI to match your local MSSQL instance credentials.
   - *Note: Currently configured for Windows Authentication*.
2. **Install Dependencies**:
   ```
   pip install flask sqlalchemy pyodbc
   ```
3. **Run the Application**:
   ```
   python run.py
   ```
   The application will start in debug mode (usually at http://127.0.0.1:5000).