**NoSQL Databases – CSE3086**

**J Component - Project Report**

**Review III**

***DocuMedix : Medicine Information Portal***

*By*

20MIA1090 Sarvika Goel

20MIA1163 P Srinivas Reddy

MTech CSE with Specialization in Business Analytics

*Submitted to*

**Dr. A. Bhuvaneswari**

Assistant Professor Senior,

SCOPE, VIT, Chennai

**School of Computer Science and Engineering**

****

*November 2023*

*­­­*

**School of Computing Science and Engineering**

VIT Chennai

Vandalur - Kelambakkam Road, Chennai - 600 127

WINTER SEM 22-23

**Worklet details**

|  |  |  |
| --- | --- | --- |
| Program | MTech with Specialization in Business Analytics | |
| Course Name / Code | CSE3086 – NoSQL Databases | |
| Slot | C2 | |
| Faculty Name | Dr. A Bhuvaneswari | |
| Team Members Name | Reg. No | 20MIA1090 | Sarvika Goel |
| 20MIA1163 | P Srinivas Reddy |

**Team Members(s) Contributions – Tentatively planned for implementation:**

|  |  |
| --- | --- |
| *Worklet Tasks* | *Contributor’s Names* |
| Dataset Collection | Srinivas Reddy |
| Preprocessing | Sarvika Goel |
| Architecture/ Model/ Flow diagram | Srinivas Reddy |
| Model building (suitable algorithm) | Sarvika Goel & Srinivas Reddy |
| Results – Tables, Graphs | Sarvika Goel |
| Technical Report writing | Srinivas Reddy |
| Presentation preparation | Sarvika Goel |

**ABSTRACT**

The Drug Information Portal, the result of painstaking data integration from two disparate datasets, emerges as a critical asset in the realm of healthcare information dissemination. This innovative platform, driven by a desire to promote informed healthcare choices, provides a comprehensive and user-centric resource for individuals and professionals looking for detailed information on medications. A meticulously curated database that brings together a plethora of pharmaceutical insights from various sources is central to its functionality. This database contains a wide range of medication-related information, from the basics like drug names, active ingredients, and dosage forms to more intricate details like manufacturer information and current pricing trends. These details are presented in an organized and easy-to-access format, providing a comprehensive view of each medication.

The portal provides users with a plethora of features, including comprehensive medication profiles that explain therapeutic uses, modes of action, and indications, allowing them to make more informed decisions about treatment options. It promotes fiscal responsibility and cost-effective healthcare decisions by ensuring pricing transparency and allowing consumers to compare costs across different markets. Through detailed insights into potential side effects, drug interactions, and precautions, safety is prioritized, facilitating risk assessment and mitigation. Furthermore, real-time availability tracking aids in timely access to medications, which is an important aspect of healthcare management. Drug interaction checkers and dosage calculators are interactive tools that improve medication safety, while user profiles provide personalized experiences with medication histories and reminders.

As part of its commitment to health literacy, the portal provides a library of educational resources, such as articles, videos, and FAQs, allowing individuals to take control of their health. Recognizing the importance of accessibility, the portal is optimized for mobile devices, allowing users to access critical information at any time and from any location.

Finally, the Drug Information Portal represents a watershed moment in the intersection of pharmaceutical complexities and healthcare decision-making. By making comprehensive medication intelligence, safety guidance, and cost transparency easily accessible, it empowers individuals and healthcare professionals to make informed decisions, improving patient outcomes and fostering a more knowledgeable and health-conscious society.

1. **Introduction**

The Drug Information Portal is a cornerstone of healthcare knowledge dissemination, with the goal of meeting the diverse informational needs of both individuals and healthcare professionals. This cutting-edge platform is the result of meticulous data integration, which drew insights from a wide range of pharmaceutical sources. Its primary mission is to promote informed healthcare decision-making by providing a comprehensive, user-centric resource for gaining detailed insights into medications.

The bedrock of this portal's functionality is a meticulously curated database at its heart. This vast database contains a wealth of medication-related information, ranging from basic details like drug names, active ingredients, and dosage forms to more intricate details like manufacturer information and the ever-changing landscape of pricing trends. This wealth of information is presented in a logical and user-friendly manner, providing users with a comprehensive view of each medication.

The portal includes a plethora of features, all of which are intended to provide users with actionable information:

* Comprehensive Medication Profiles: Comprehensive medication profiles delve into therapeutic applications, modes of action, and indications, providing critical information for making informed treatment decisions.
* Pricing Transparency: Users can compare medication costs across markets using real-time pricing data, encouraging fiscal responsibility and facilitating cost-effective healthcare decisions.
* Insights into Safety and Efficacy: Comprehensive information on potential side effects, drug interactions, and precautionary measures prioritizes safety, allowing for risk assessment and mitigation.

The Drug Information Portal, in essence, acts as a transformative conduit, seamlessly connecting the complexities of pharmaceutical knowledge with the needs of healthcare decision-makers. It empowers individuals and healthcare professionals alike to make informed choices by facilitating easy access to comprehensive medication intelligence, safety guidance, and cost transparency, thereby elevating patient outcomes and nurturing a more knowledgeable and health-conscious society.

1. Literature Survey (sample)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S. no | Title | Author / Journal name / Year | Technique | Result |
| 1 | Using bar-code technology and medication observation methodology for safer medication administration | Richard D. Paoletti, Tina M. Suess, Michael G. Lesko, Alfred A. Feroli,  James A. Kennel And Timothy Sauders | Barcode scanning and visual confirmation of medication administration. The BCMA system reduces medication errors . | With BCMA, 50% reduction in administration errors and a 75% reduction in transcription errors. |
| 2 | Drug Information Services and Sources of Information | Matthew P. Van Cuyk,, Jason C. Cooper and James P. New | Systematic review of drug information sources and reliability through ASHP Drug Information Resource Centre. | Drug information services provide access to reliable and up-to-date drug information and are readily accessible. |
| 3 | Effects of an Integrated clinical information system on medication safety in a multi-hospital setting | Charles D. Mahoney, Christine M. Berard-Collins, Reid Coleman, and Carole M. Cotter | Implementation of vendor-based integrated clinical information technology. The study used a quasi-experimental design to compare the medication error rates of ICIS and traditional approach. | ICIS implementation decreased medication errors throughout medication-use process. |
| 4 | National Cancer Institute Drug Information System 3D Database | G. W. A. Milne, Marc C. Nicklaus, J. S. Driscoll, and Shaomeng Wang | 3D search of chemical structures for anticancer activity based on the concept of pharmacophore, which is molecular features. | It has been found that the method was able to identify compounds with potential anticancer activity that had not been previously identified. |
| 5 | User Information Seeking Behaviour in a Medical Web Portal Environment : A Preliminary Study | Dongming Zhang, Caroline Zambrowicz, Hong Zhou, and Nancy K. Roderer | Qualitative analysis of user feedback on medical web portal and with more prominence to focus groups. It is valuable tool for understanding user information seeking behaviour in a medical web portal environment. | Users seek information on medical topics to manage their health and information. |
| 6 | Mutations and Drugs Portal (MDP): A Database Linking Drug Response Data and Genomic Information | Jimmy Caroli, Cristian Taccioli, Giovanni Sorrentino, Silvio Bicciato | They used the integration of genomic and pharmacological data. The NCI60 DTP screening to obtain drug response info. | Database of drug sensitivity and genomic data for cancer research. Used to identify effective in inhibiting YAP/TAZ, a protein that is involved in cancer cell growth. |
| 7 | An information system for drug interactions: pharmaceutical care and prescription | Luciano Roberto Hirano, Silvio Bortoleto, Cláudia Seiko Yokoyama; Hugo Bulegon | They Computerized drug interaction checking to identify potential drug interactions between the medications a patient is taking. Program compares the patient's medication list to a database of known drug interactions. | An efficient computer-based system to detect and prevent drug interactions in clinical practice. It is used to improve the safety of patients. |
| 8 | Patient Drugs and Data Management by Mobile Application | Mohammed Abdulrahman Abdullah, Mustafa Alghali | Mobile app for medication tracking and management, integrated frameworks of data and semantic Web. | Mobile apps can improve medication adherence, patient outcomes and the standards of patient safety |
| 9 | Hospital: a web-based application with digital signature for drugs dispensing management | Lorenzo Rossi, Lorenzo Margola, Vania Manzelli Alessandra Bandera | Web-based application with digital signature to ensure authenticity and integrity of drug orders. It follows track and efficiency of drug dispensing. | Efficient drug dispensing in hospitals and improved efficiency, reduced errors, and increased compliance with regulations. |
| 10 | Telematics integrated system to perform drugs prescription and administration reducing adverse drug events | E. Iadanza, M.C. Pettenati, L. Bianch, L. Ciofi, F. Pirri, G. Biffi Gentili, D. Giuli | A telematics-integrated system that uses Computerized Prescription Order Entry (CPOE),, Radio-frequency identification (RFID), and barcoding to reduce ADEs. | A telematics integrated system was proposed to reduce adverse drug events (ADEs) by improving the accuracy and safety of drug prescription and administration |

1. **Dataset and Database Tools**

**Dataset – 1**: <https://www.kaggle.com/datasets/mohneesh7/indian-medicine-data>

The dataset contains detailed information related to various medicines and pharmaceutical products available in the Indian market. It is structured with the following attributes:

1. **sub\_category**: This column categorizes medicines into specific medical categories, providing a classification system that defines the domain in which each medicine finds its application. These categories help users identify the area of medicine to which a particular product belongs, such as pain relief, antibiotics, cardiovascular, etc.
2. **product\_name**: This attribute represents the name of the medicine or pharmaceutical product as it is known in the Indian market. It serves as a unique identifier for each product and is the primary reference point for users searching for specific medications.
3. **salt\_composition**: This column provides information about the chemical composition of each drug. It specifies the active ingredients or chemical compounds that make up the medication. Understanding the salt composition is crucial for both healthcare professionals and patients to ensure the correct medication is being used.
4. **product\_price**: This attribute represents the previous price of the product in the Indian market. However, it is important to note that this price is considered as a reference and may not accurately reflect the current market price. Medication prices can be highly volatile due to various factors, including market dynamics and regulatory changes.
5. **product\_manufactured**: This column indicates the pharmaceutical company responsible for manufacturing or producing the medicine or drug. Knowing the manufacturer is important for quality assurance and can influence trust in the product.
6. **medicine\_desc**: This attribute provides a comprehensive overview and detailed description of each specific medicine or pharmaceutical product. It may include information about the product's therapeutic uses, mode of action, indications, contraindications, dosage instructions, and other relevant details. This description offers valuable insights to users seeking in-depth information about the medication.
7. **side\_effects**: The "side\_effects" column lists potential adverse effects associated with the drug or medicine. It informs users about the possible risks and undesirable outcomes that may occur when using the medication. Understanding side effects is crucial for making informed decisions and managing the safety of a treatment.
8. **drug\_interactions**: This column provides information about interactions and effects that may occur when combining the specific medicine with other drugs. It alerts users to potential interactions that could impact the effectiveness or safety of a medication when taken concurrently with other prescribed or over-the-counter drugs.

Overall, this dataset serves as a valuable resource for individuals, healthcare professionals, and researchers interested in gaining insights into the world of pharmaceutical products available in the Indian market. It offers a comprehensive view of medicines, including their classification, composition, pricing (as a reference), manufacturer details, descriptions, potential side effects, and drug interactions, all of which are essential for informed healthcare decision-making.

**Dataset – 2:** <https://www.kaggle.com/datasets/shudhanshusingh/az-medicine-dataset-of-india>

The dataset contains comprehensive information about various medicines and pharmaceutical products available in the Indian market. Each row in the dataset represents a unique medicine or product, and the dataset is structured with the following attributes:

1. **id**: This attribute serves as a unique identifier used to represent the count of records in the dataset. It provides a sequential numbering system to differentiate and reference individual entries.
2. **name**: The "name" column contains the name of the medicine as per the market record. This is the official or commonly recognized name used for the medicine in the Indian market, which allows users to easily identify and search for specific medications.
3. **price(₹)**: This attribute represents the price of the medicine in the Indian market as of November 2022. It provides cost-related information for each medicine, helping users understand the pricing of different medications. It's important to note that this price data is specific to the given time frame and may not reflect current market prices.
4. **Is\_discontinued**: The "Is\_discontinued" column indicates whether the medicine is currently available for purchase or not. This attribute informs users about the availability status of each medication, which is crucial information for both healthcare professionals and patients.
5. **manufacturer\_name**: This column specifies the name of the pharmaceutical company responsible for producing the medicine mentioned in each row. Knowing the manufacturer's name is essential for quality assurance and can influence trust in the product.
6. **type**: The "type" attribute categorizes the medicine into different types, such as Allopathy, Homeopathy, or other relevant classifications. It provides information about the nature or category of the medicine.
7. **pack\_size\_label**: This column includes labels mentioned on the packaging of the medicine, indicating the quantity and form (e.g., Tablet, Syrup) of the medicine. It helps users understand the packaging and dosage forms available for each medication.
8. **short\_composition1**: This attribute provides information about the first composition used to make the medicine. It specifies the primary active ingredient or chemical compound that constitutes the core of the medication.
9. **short\_composition2**: If applicable, the "short\_composition2" column lists any other composition used along with the first composition to make the medicine. Some medications may have multiple active ingredients or components, and this attribute captures the secondary composition when present.

Overall, this dataset serves as a valuable resource for individuals, healthcare professionals, researchers, and those interested in gaining insights into pharmaceutical products available in the Indian market. It offers a comprehensive view of medicines, including their names, prices (as of November 2022), availability status, manufacturer details, types, packaging information, and chemical compositions, providing essential information for informed healthcare decision-making and analysis within the Indian pharmaceutical landscape.

**Database Tools: MongoDB**

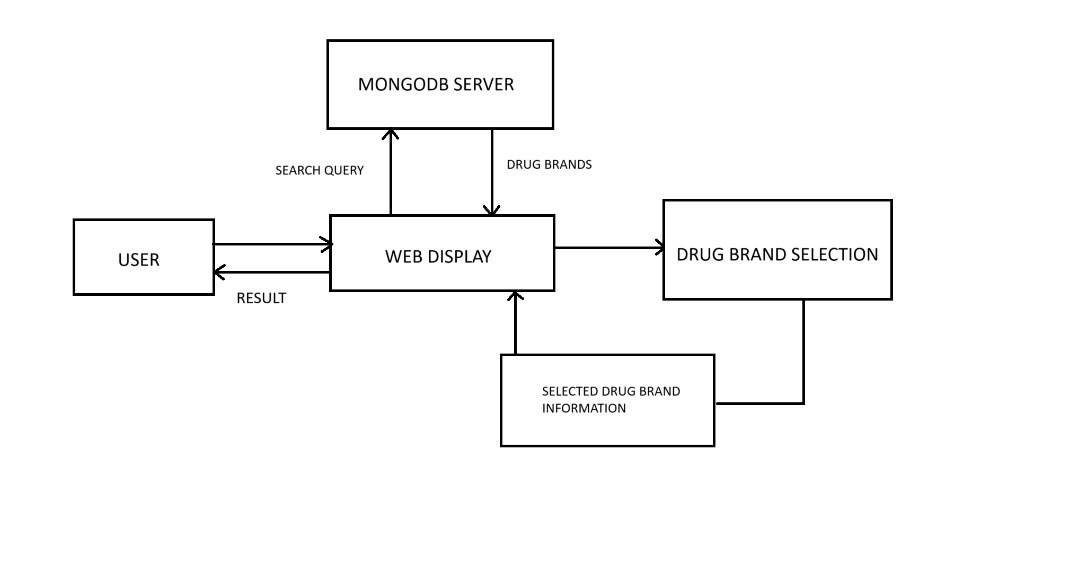
1. **MongoDB Server**: The core MongoDB database server is required to store and manage your data. You will need to install and configure MongoDB server on your hosting environment or server.
2. **MongoDB Compass**: MongoDB Compass is a graphical user interface (GUI) tool for interacting with MongoDB databases. It provides a visual way to explore your data, perform queries, and manage your database schema.
3. **MongoDB Drivers**: Depending on the programming language and framework you're using to build your portal (e.g., Node.js, Python, Java), you'll need the corresponding MongoDB driver or library to connect to and interact with the MongoDB database. For example:
4. Node.js: You can use the official MongoDB Node.js driver.
5. Python: PyMongo is the official driver for Python.
6. Java: The MongoDB Java driver is used for Java-based applications.
7. **MongoDB Atlas**: MongoDB Atlas is a cloud-based database service provided by MongoDB. It offers automated backups, scaling options, and high availability. You can consider using MongoDB Atlas to host your database in the cloud.
8. **MongoDB BI Connector**: If you need to perform business intelligence and reporting tasks on your data, the MongoDB BI Connector can help connect MongoDB to popular BI tools like Tableau, Power BI, or QlikView.
9. **MongoDB Atlas Data Lake**: If you need to analyse large volumes of data stored in MongoDB using big data analytics tools, MongoDB Atlas Data Lake allows you to query data stored in your Atlas cluster using tools like Apache Spark and AWS Athena.
10. **MongoDB Charts**: MongoDB Charts is a tool that allows you to create visualizations and dashboards directly from your MongoDB data. It can be valuable for creating data visualizations in your Drug Information Portal.
11. **Algorithms / Techniques description**

**Frontend Pseudocode:**

1. Initialize the Medication Search Page:
   * 1. Create a component with state variables for searchQuery, medications, isLoading, and error.
     2. Render an input field for medication search, a search button, and a results list.
2. Handle User Input:
   * 1. Listen for changes in the search input field.
     2. When the user types, update the searchQuery state variable accordingly.
3. Handle Medication Search:
   * 1. When the user clicks the search button:
        + 1. Display a loading indicator (set isLoading to true).
          2. Clear any previous error messages.
          3. Send an HTTP request to the backend API to search for medications based on the searchQuery.
     2. Upon successful response:
        + 1. Update the medications state variable with the search results.
          2. Turn off the loading indicator (set isLoading to false).
     3. If an error occurs during the request:
        + 1. Display an error message to the user.
          2. Turn off the loading indicator (set isLoading to false).
4. Display Medication Results:
   * 1. Map through the medications state variable to render a list of medication names and descriptions.
5. Render the Medication Search Page component to the DOM.

**Backend Pseudo Code:**

1. Initialize the Backend Server:
   1. Set up a backend server using a framework like Express.js.
   2. Define API endpoints for medication search and other functionalities.
2. Handle Medication Search Request:
   1. Listen for incoming GET requests to the "/api/medications/search" endpoint.
   2. Extract the search query from the request parameters.
3. Query the Database:
   1. Use the search query to query the MongoDB database for medications matching the query.
   2. Retrieve medication information including name, description, and other relevant details.
   3. If medications are found, send a JSON response with the medication data.
   4. If no medications are found, send an error response.
4. Error Handling:
   1. Implement error handling to catch and respond to any errors that may occur during the search process.
5. Start the Backend Server:
   1. Listen for incoming HTTP requests on the specified port (e.g., port 3000).
6. **Block Diagram of the proposed work / system design**

****

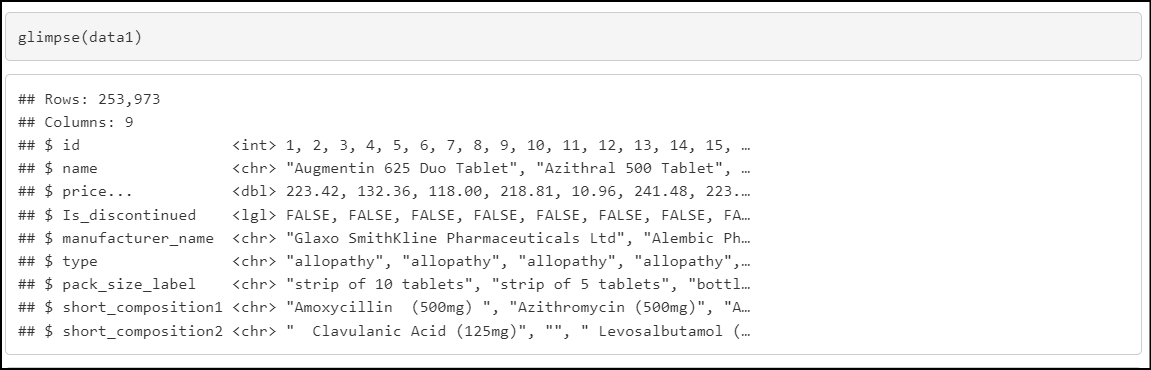
1. **GitHub Repository Link :** [**https://github.com/Sarvika17/DocuMedix.git**](https://github.com/Sarvika17/DocuMedix.git)
2. **Implementation:**
3. **Data Loading and Pre-processing**

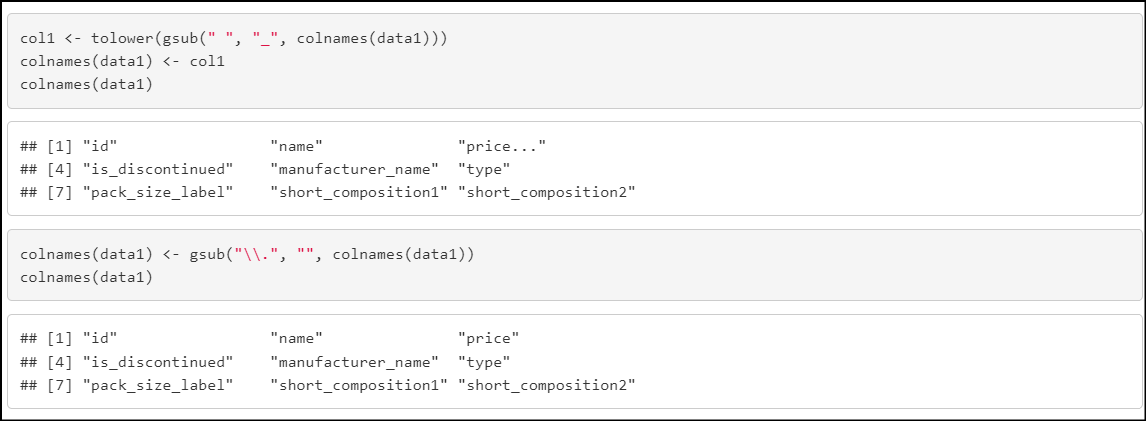
Data Loading and Pre-processing was done using RStudio Cloud as it being an easy and convenient method.

First the data was loaded in R studio cloud. There were two datasets and both needed some standard cleaning.

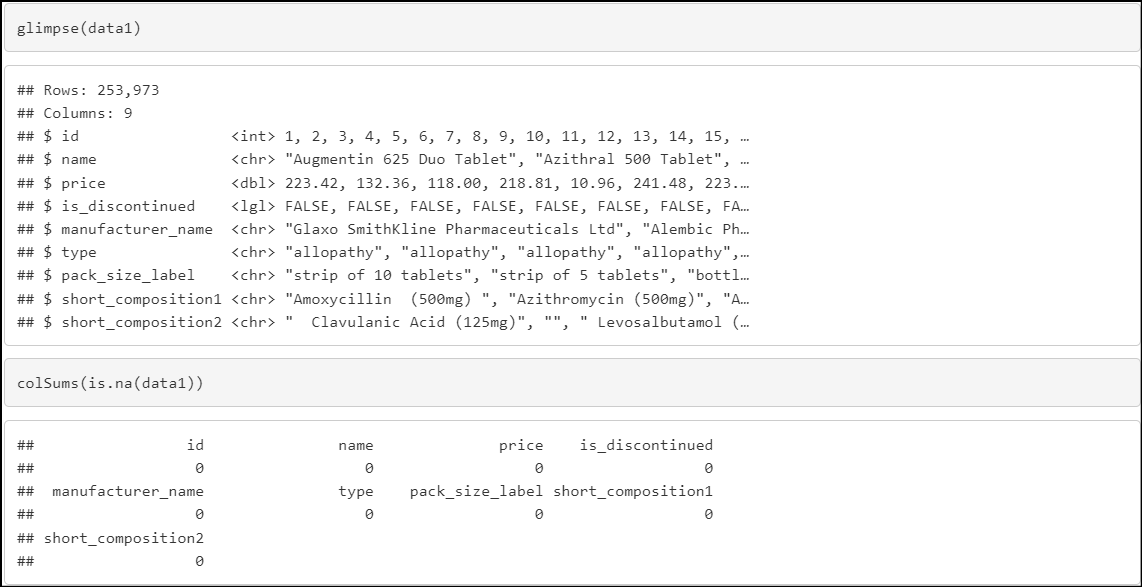
Data 1: <https://www.kaggle.com/datasets/shudhanshusingh/az-medicine-dataset-of-india>

As we can see from the glimpse of the dataset that the column names are not in standard format and require some formatting so as to not cause errors down the lane and be easier to work with. All the column names were first converted to lowercase and then any special characters like (.) dot was replaced by underscore(\_) and also and unnecessary space was also removed.





After this, we can once again see the glimpse of the dataset and observe that all the column names are now in standard format. Also, there are no missing values in the dataset thus it does not require any kind of other pre-processing.



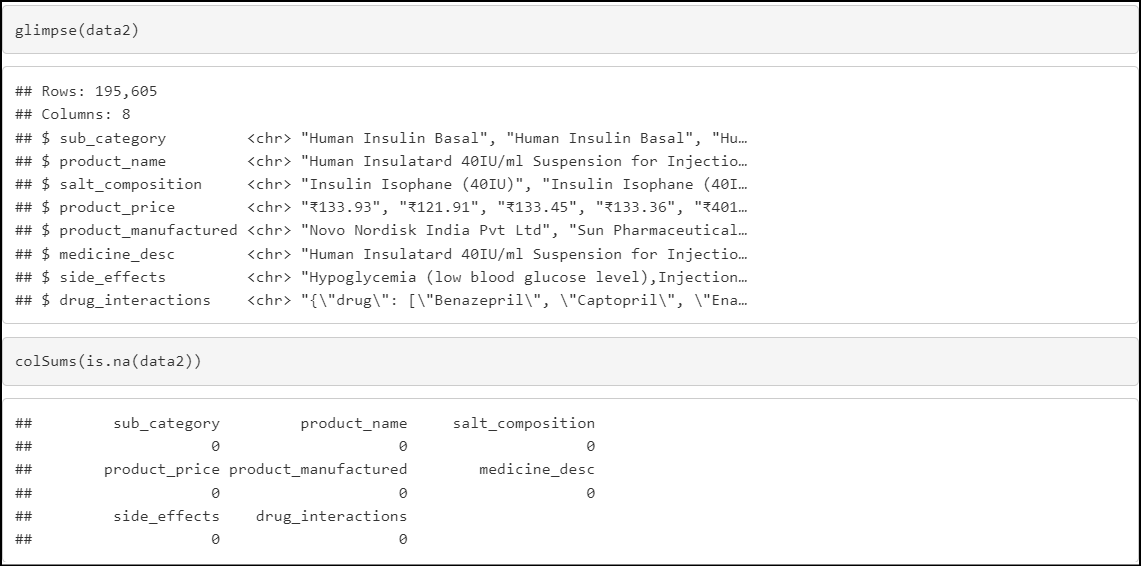
Here is the glimpse of dataset 2: <https://www.kaggle.com/datasets/mohneesh7/indian-medicine-data>

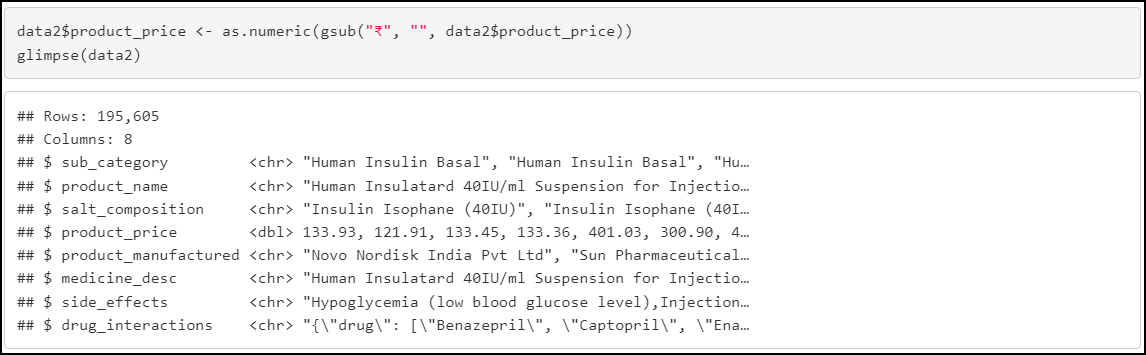
Dataset 2 also has issues with the column names and previously loading the dataset couldn’t detect the “product\_name” column. So, after standardizing the column names in this dataset as well, the issues were resolved.



Next, we see that “product\_price” column is taken as a character type instead of numeric type. This is due to the presence of Rupee symbol in the dataset. The Rupee Symbol was then removed and only the numbers were left.

Also, no missing values were found in this dataset as well. Thus, no other data cleaning is required.

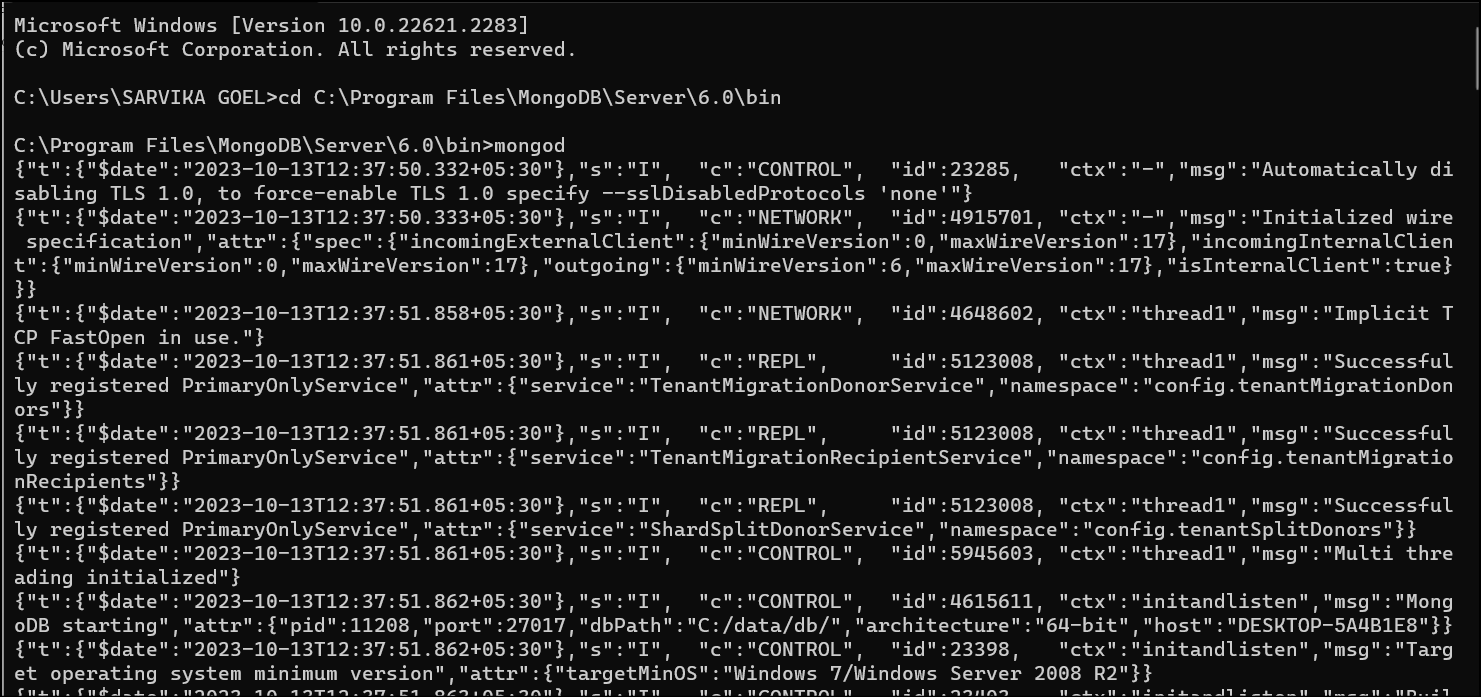


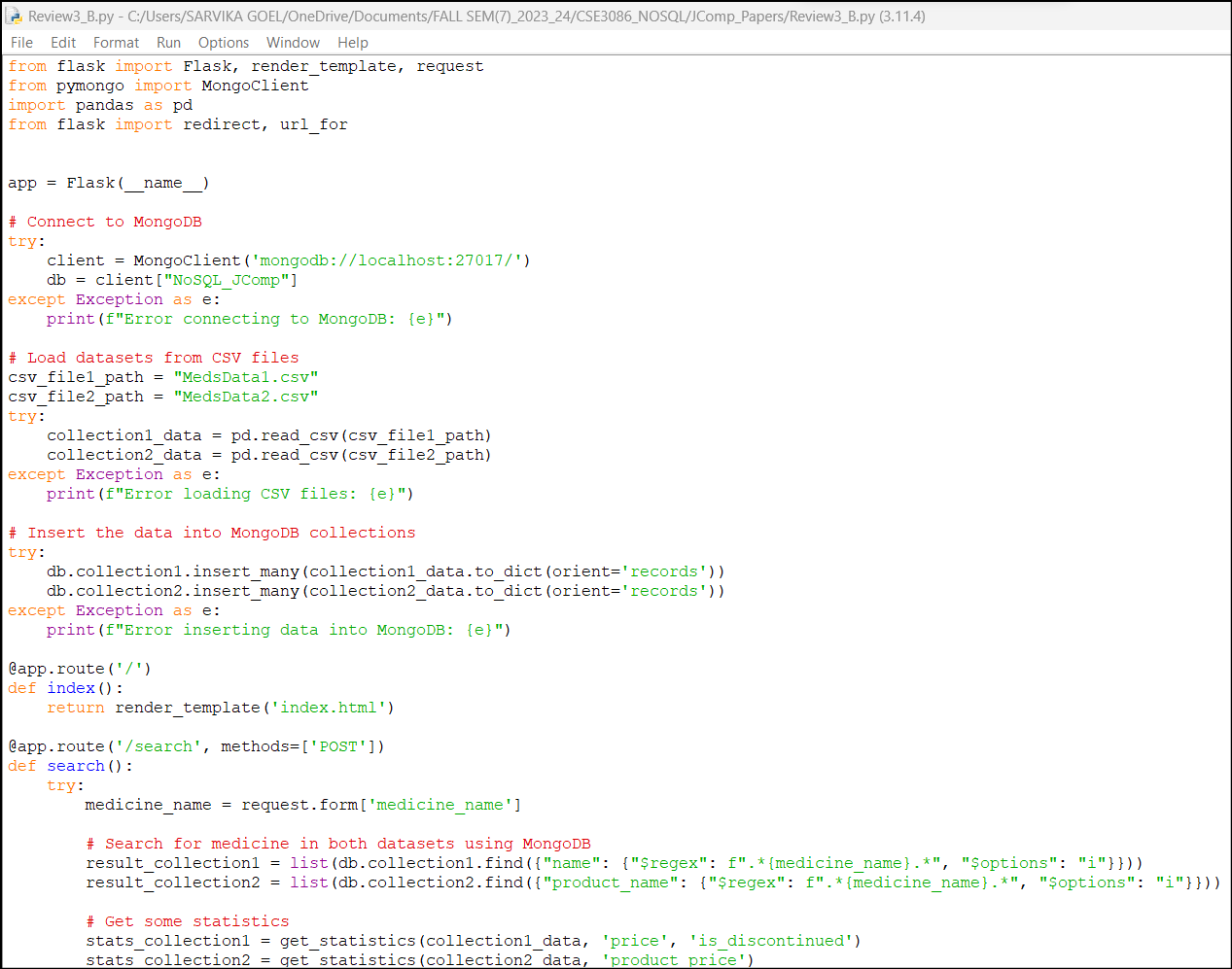




1. **Connecting Python and MongoDB**

Now, the task is to connect our Python interface that is the Spyder IDE to MongoDB. To achieve this task, pymongo was installed on the Spyder IDE and connected through MongoClient(). Also, the mongo server was turned on. And the python code was run.





**CODE:**

from flask import Flask, render\_template, request

from pymongo import MongoClient

import pandas as pd

from flask import redirect, url\_for

app = Flask(\_\_name\_\_)

# Connect to MongoDB

try:

client = MongoClient('mongodb://localhost:27017/')

db = client["NoSQL\_JComp"]

except Exception as e:

print(f"Error connecting to MongoDB: {e}")

# Load datasets from CSV files

csv\_file1\_path = "MedsData1.csv"

csv\_file2\_path = "MedsData2.csv"

try:

collection1\_data = pd.read\_csv(csv\_file1\_path)

collection2\_data = pd.read\_csv(csv\_file2\_path)

except Exception as e:

print(f"Error loading CSV files: {e}")

# Insert the data into MongoDB collections

try:

db.collection1.insert\_many(collection1\_data.to\_dict(orient='records'))

db.collection2.insert\_many(collection2\_data.to\_dict(orient='records'))

except Exception as e:

print(f"Error inserting data into MongoDB: {e}")

@app.route('/')

def index():

return render\_template('index.html')

@app.route('/search', methods=['POST'])

def search():

try:

medicine\_name = request.form['medicine\_name']

# Search for medicine in both datasets using MongoDB

result\_collection1 = list(db.collection1.find({"name": {"$regex": f".\*{medicine\_name}.\*", "$options": "i"}}))

result\_collection2 = list(db.collection2.find({"product\_name": {"$regex": f".\*{medicine\_name}.\*", "$options": "i"}}))

# Get some statistics

stats\_collection1 = get\_statistics(collection1\_data, 'price', 'is\_discontinued')

stats\_collection2 = get\_statistics(collection2\_data, 'product\_price')

return render\_template('result.html', result\_collection1=result\_collection1, result\_collection2=result\_collection2,

stats\_collection1=stats\_collection1, stats\_collection2=stats\_collection2)

except Exception as e:

print(f"Error in search: {e}")

return render\_template('error.html', error\_message="An error occurred during the search.")

def get\_statistics(data, price\_column, discontinued\_column=None):

statistics = {

'average\_price': data[price\_column].mean(),

'median\_price': data[price\_column].median(),

'max\_price': data[price\_column].max(),

'min\_price': data[price\_column].min(),

}

if discontinued\_column:

statistics['num\_discontinued'] = data[data[discontinued\_column]].shape[0] if discontinued\_column in data.columns else 0

return statistics

@app.errorhandler(500)

def internal\_server\_error(e):

error\_message = "Internal Server Error. Please try again later."

return render\_template('error.html', error\_message=error\_message), 500

@app.route('/statistics')

def statistics():

try:

# Get overall statistics directly from MongoDB

total\_medicines\_collection1 = db.collection1.count\_documents({})

total\_medicines\_collection2 = db.collection2.count\_documents({})

unique\_medicines\_collection1 = db.collection1.distinct("name")

unique\_medicines\_collection2 = db.collection2.distinct("product\_name")

common\_medicines = len(set(unique\_medicines\_collection1).intersection(unique\_medicines\_collection2))

overall\_stats = {

'total\_medicines\_collection1': total\_medicines\_collection1,

'total\_medicines\_collection2': total\_medicines\_collection2,

'unique\_medicines\_collection1': len(unique\_medicines\_collection1),

'unique\_medicines\_collection2': len(unique\_medicines\_collection2),

'common\_medicines': common\_medicines,

}

type\_distribution\_collection1 = db.collection1.aggregate([

{"$group": {"\_id": "$type", "count": {"$sum": 1}}}

])

type\_distribution\_collection1 = {doc['\_id']: doc['count'] for doc in type\_distribution\_collection1}

type\_distribution\_collection2 = db.collection2.aggregate([

{"$group": {"\_id": "$sub\_category", "count": {"$sum": 1}}}

])

type\_distribution\_collection2 = {doc['\_id']: doc['count'] for doc in type\_distribution\_collection2}

manufacturer\_distribution\_collection1 = db.collection1.aggregate([

{"$group": {"\_id": "$manufacturer\_name", "count": {"$sum": 1}}}

])

manufacturer\_distribution\_collection1 = {doc['\_id']: doc['count'] for doc in manufacturer\_distribution\_collection1}

manufacturer\_distribution\_collection2 = db.collection2.aggregate([

{"$group": {"\_id": "$product\_manufactured", "count": {"$sum": 1}}}

])

manufacturer\_distribution\_collection2 = {doc['\_id']: doc['count'] for doc in manufacturer\_distribution\_collection2}

# Prepare data for charts

chart\_data = {

'labels\_collection1': list(type\_distribution\_collection1.keys()),

'data\_collection1': list(type\_distribution\_collection1.values()),

'labels\_collection2': list(type\_distribution\_collection2.keys()),

'data\_collection2': list(type\_distribution\_collection2.values()),

}

return render\_template('statistics.html', overall\_stats=overall\_stats,

manufacturer\_distribution\_collection1=manufacturer\_distribution\_collection1,

manufacturer\_distribution\_collection2=manufacturer\_distribution\_collection2,

chart\_data=chart\_data)

except Exception as e:

print(f"Error in statistics: {e}")

@app.route('/redirect\_index')

def redirect\_index():

return redirect(url\_for('index'))

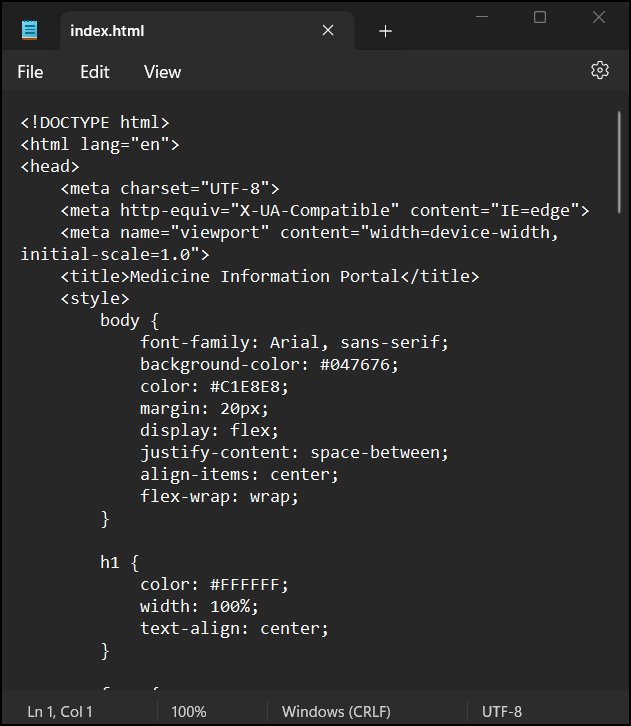
if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

1. **Creating the Web Page**

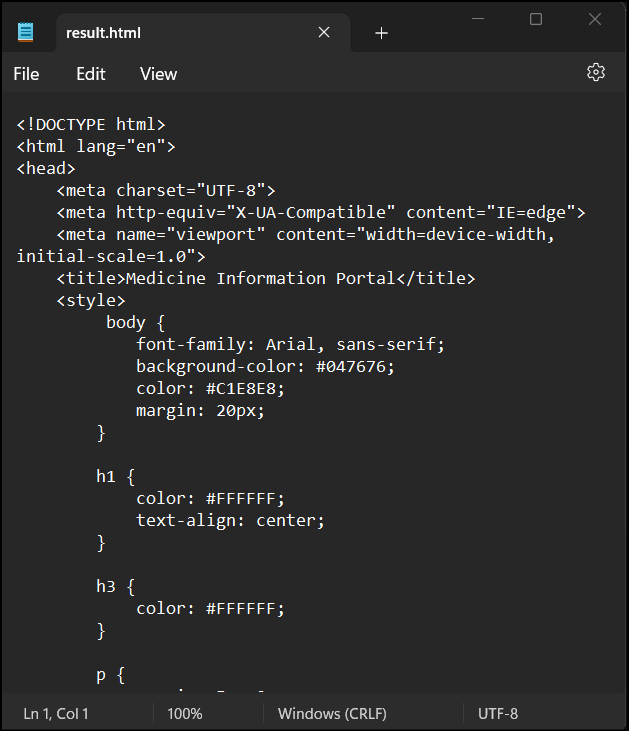
Now, frontend part was to make a app using Flask, a web page using html and CSS to use as user interface. This was done by creating two templates:

* Index.html

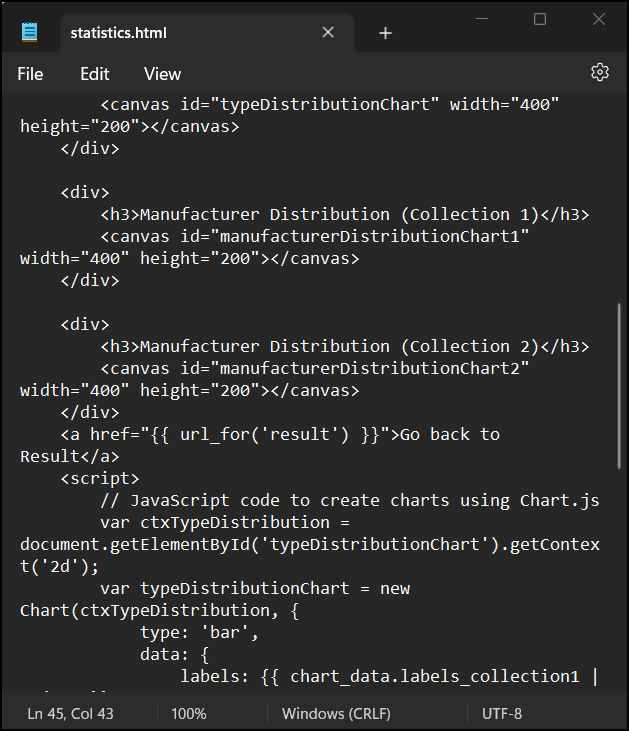


This is just a glance of how it looks. The full code is uploaded on the GitHub repository.

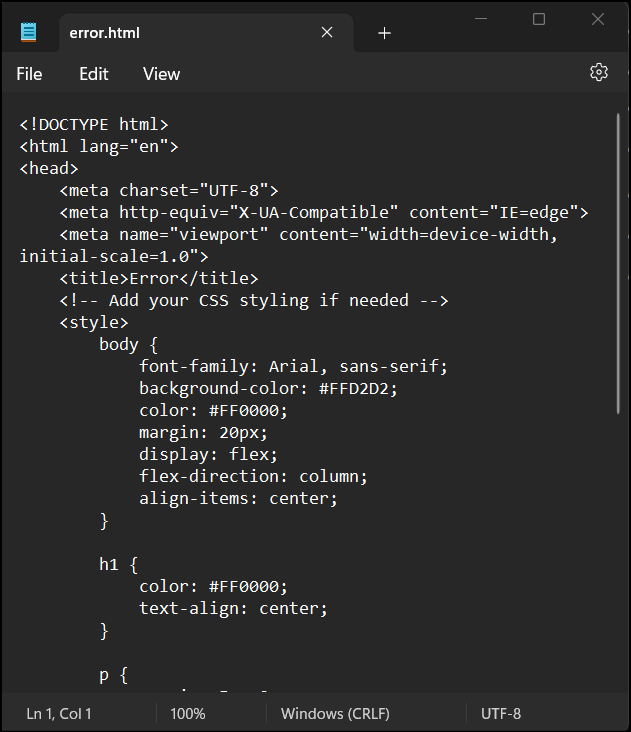
* Result.html



* Statistic.html



* Error.html

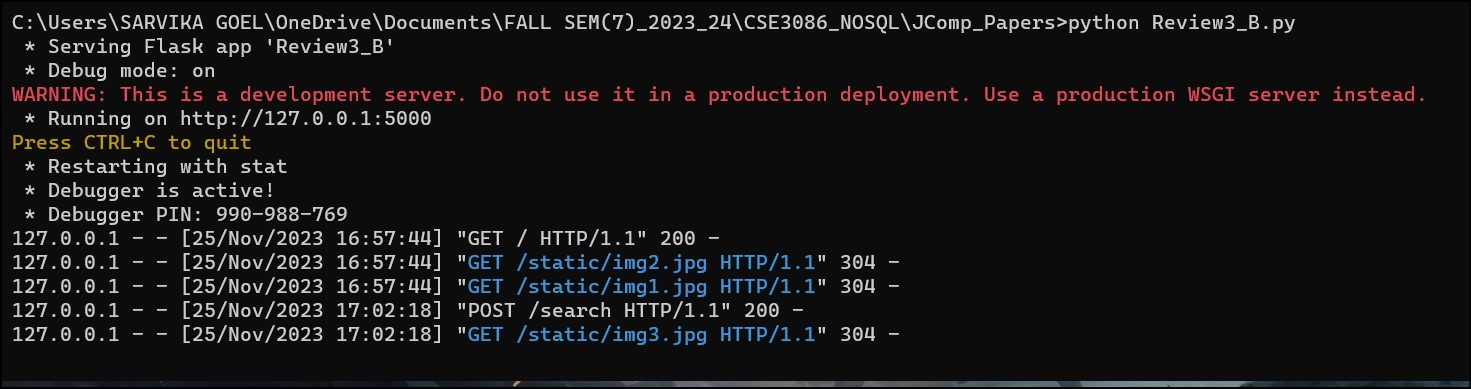


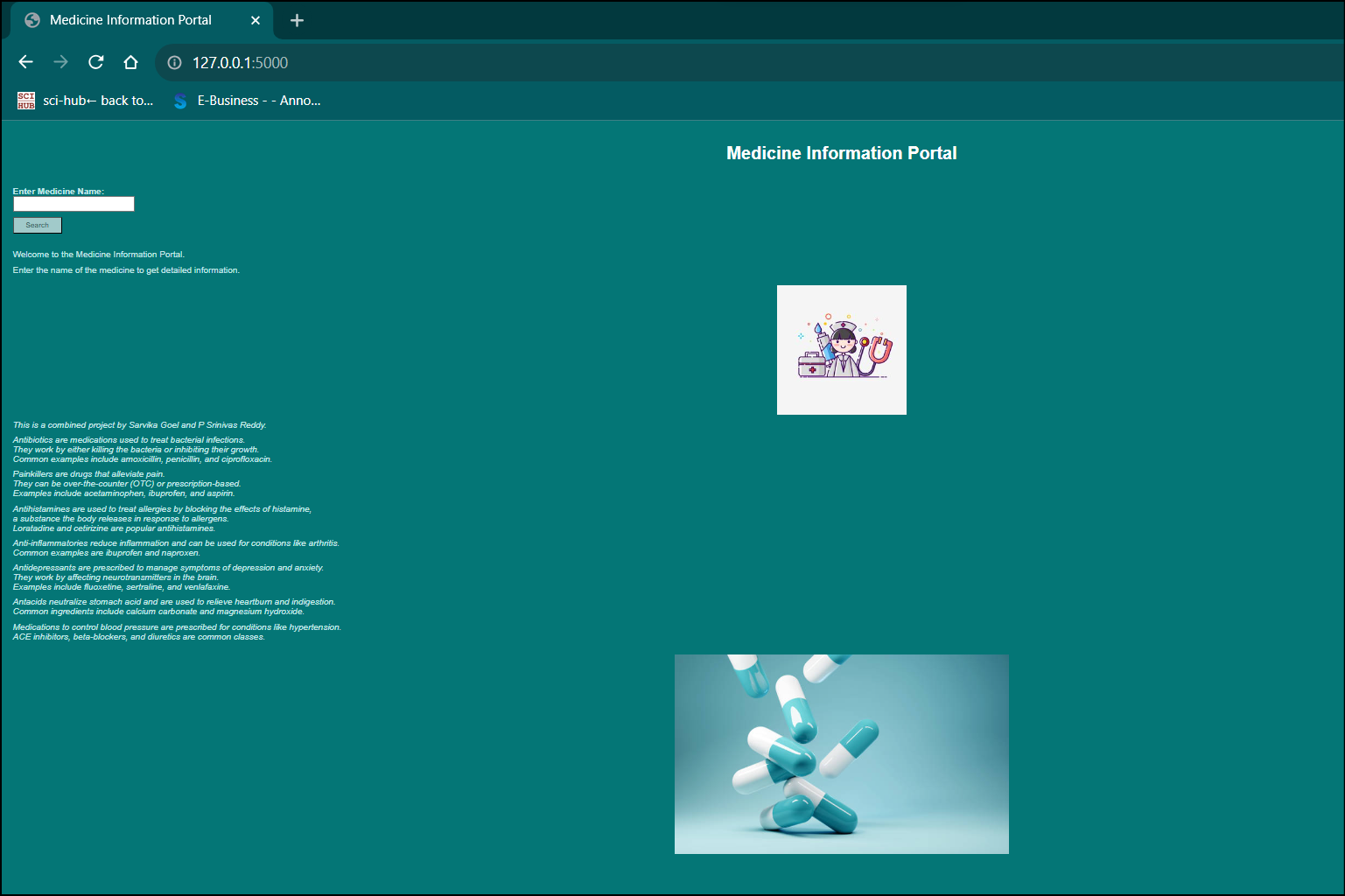
These two templates connect the user to the application.

1. **Output**

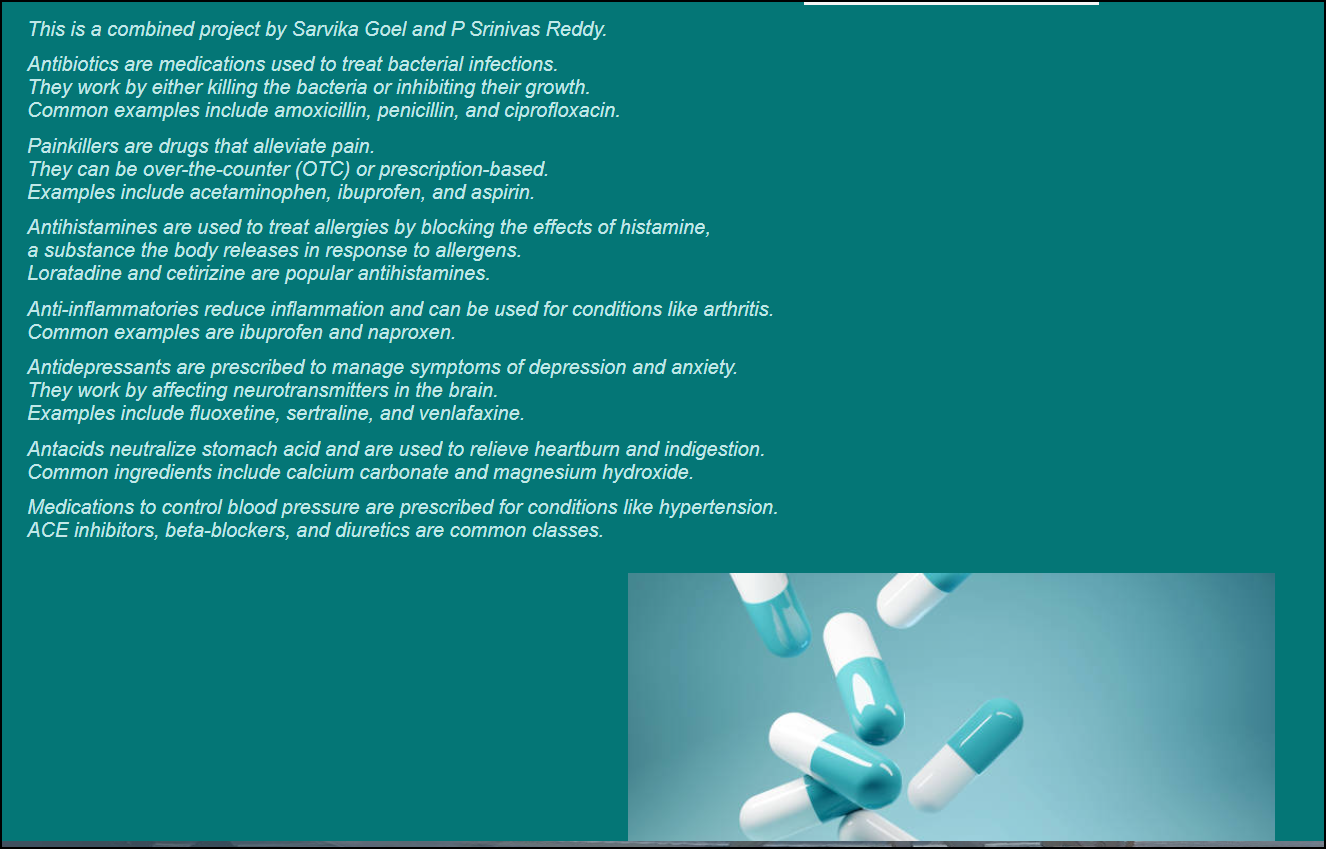
Now, let’s see how it looks.

As we type the IP Address of our **Medicine Information Portal : 127.0.0.1:5000,** the index.html page opens. The following is the first look we get of the page. It contains the title of the portal, the search box from which we can search the name of a particular medicine and then we get other small basic information about the types of medicines.

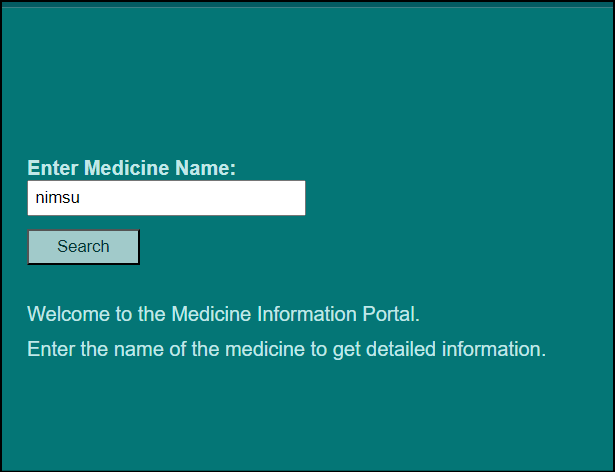


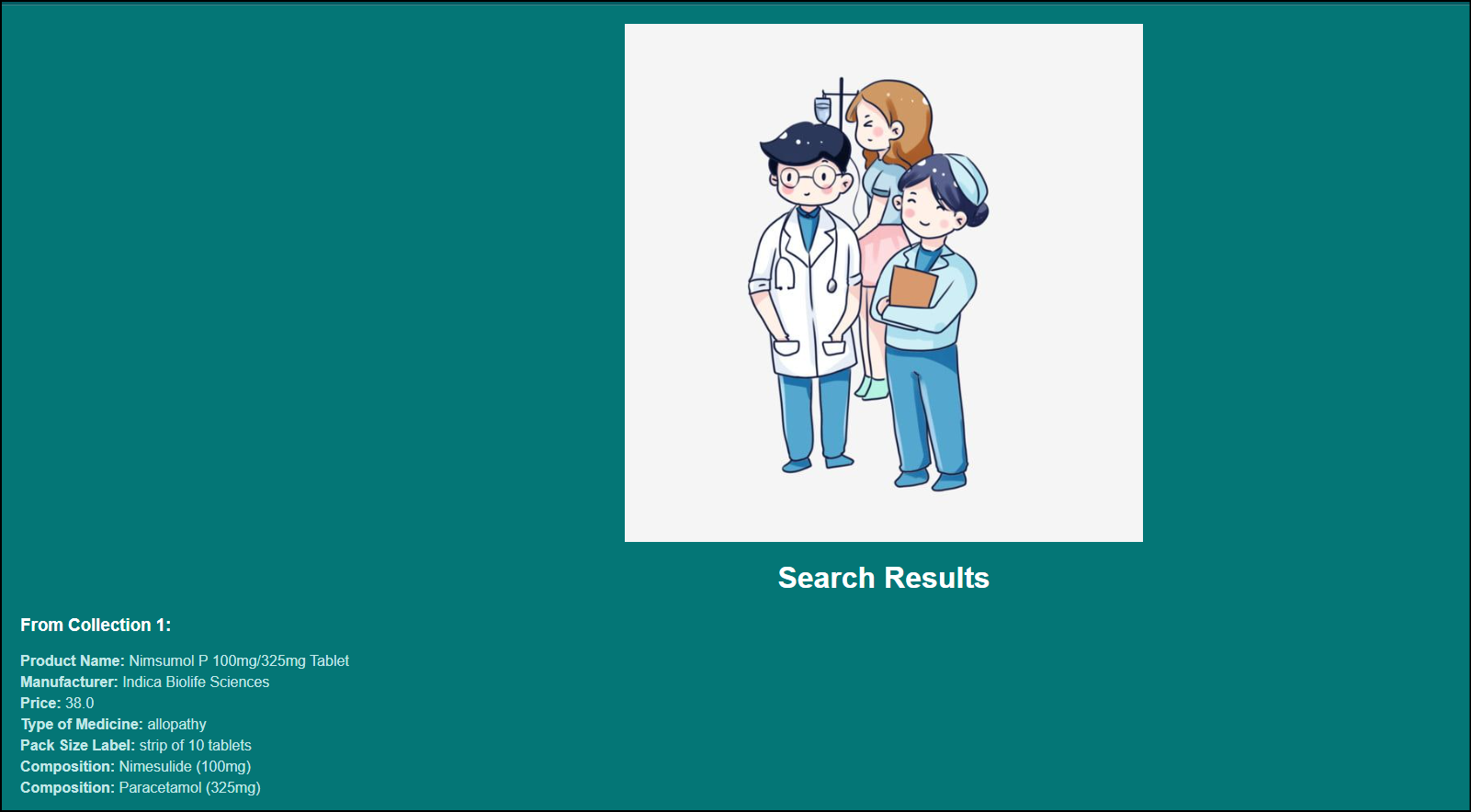


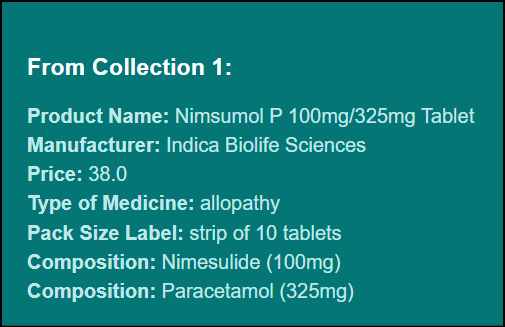


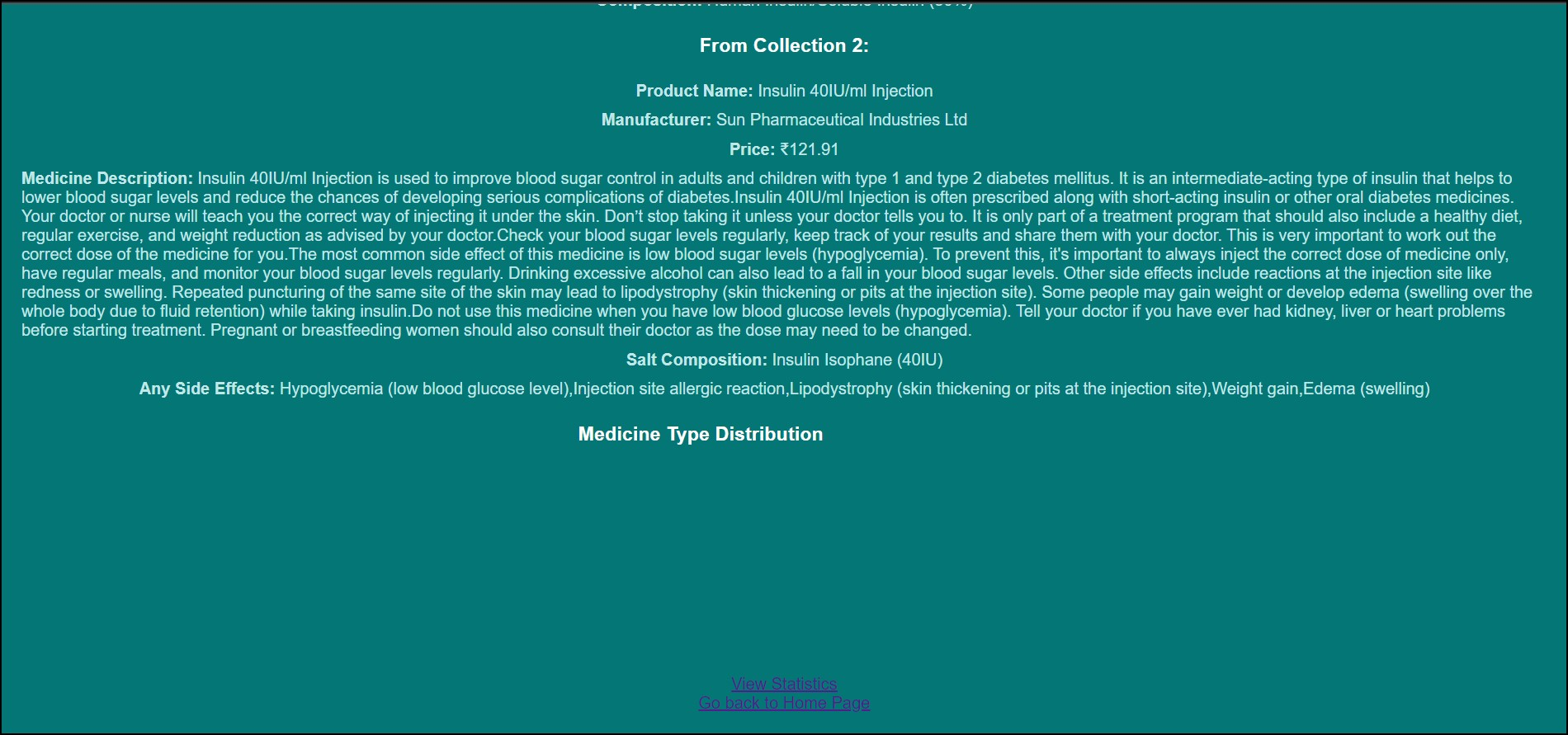


Now, let’s try searching a medicine and see what results it shows:











****

Like this it is showing the results.

**REFERENCES**

1. <https://academic.oup.com/ajhp/article-abstract/64/18/1969/5135100>
2. <https://www.healthaffairs.org/doi/abs/10.1377/hlthaff.26.3.731>
3. <https://www.sciencedirect.com/science/article/abs/pii/B978012814276900009X>
4. <https://pubs.acs.org/doi/pdf/10.1021/ci00021a032>
5. <https://academic.oup.com/ajhp/article-abstract/64/5/536/5135264>
6. <https://onlinelibrary.wiley.com/doi/abs/10.1002/asi.20001>
7. <https://ieeexplore.ieee.org/document/7545951>
8. <https://ieeexplore.ieee.org/document/5393548>
9. <https://ieeexplore.ieee.org/document/9429621>
10. <https://ieeexplore.ieee.org/document/6347381>