IBRA CISSE

jason Jensen

MEdicine recommandation fINAL REPORT

Table of Contents

[**Abstract** 2](#_Toc450484109)

[**Introduction** 3](#_Toc450484110)

[**Product Requirements** 3](#_Toc450484111)

[**Table Relationships** 4](#_Toc450484112)

[**Use Cases and Scenarios for the database** 5](#_Toc450484113)

[**E/R Diagram** 6](#_Toc450484114)

[**Database table design** 7](#_Toc450484115)

[**Test Plan** 9](#_Toc450484116)

[**Testing Strategy** 9](#_Toc450484117)

[**Conclusion** 11](#_Toc450484118)

# **Abstract**

When sick, a trip to the drug store is an unpleasant endeavor. Even knowing your illness, choosing the best medication to alleviate your symptoms can be difficult due to the large number of options available. This project would make this choice, and therefore the trip, much simpler. By inputting a list of symptoms, the database will cross reference and recommend the best medication for your symptoms. Medication is ranked according to effectiveness based on an ever growing list of surveys. These symptoms would include: headaches, toothaches, stomachaches, allergies, etc.

# **Introduction**

This project is meant to help regular people choose the best over the counter medication for minor illnesses. The concept is to develop a system where users can input the symptoms that they are experiencing and based on research and feedback from customers and medical documentation, we provide a recommendation of the best over the counter pills to take to alleviate the symptoms. The application will use several tables and databases which will have several indexed tables. The databases will be tied to each other with an index id to quicken queries. Once queried the application will search the databases that contain a medicines description and rating, based on reviews from other users and scientific documentations, queries will then return the top sorted medicines for the specified symptoms along with their description. The results will be presented through a custom made user interface. Users will be able to select from drop down menus to view medicine descriptions and rating. Users will also have the choice to log in and write reviews which will influence the rating of medication.

# **Product Requirements**

The database will focus on 3 different views, User, Client, and Administrator. Each one of the following has specific roles and regulation on what they can do and access based on their rights set up by the database manager:

1. **Users**

Users are unregistered and can only search for symptoms and medicine.

1. **Client**

Clients are registered users who can log in to write reviews and rate medicine. They have to be authenticated from the database and given the respect view.

1. **Administrator role and functionalities**

The Administrator is the database manager and controls everything within the database. The administrator has the right and roles to create new tables, update tables, and change table outcomes. The administrator can also delete tables or delete the existing database as well as initiate a backup of the database. Administrator can also change the rating of medicines and delete reviews.

1. **Database Functionalities**

* The database will provide the rating, brand name, and generic name for any symptom.
* The user will only be able to search for information and view already written reviews.
* The client will also be able to write reviews and rate medicines.
* The administrator will also be able to remove and edit reviews.

# **Table Relationships**

* The client and medicine tables will be linked based on IDs which will be the primary key. The rest will be foreign keys.
* The medicine table will be linked with the symptom table, each medicine will become a member of a symptom, which will hold a description of that symptom and the medicine most recommended to alleviate it.
* Ratings and reviews will be linked to the symptom and the medicine tables, it will be based on how many reviews have been received. Each review will be rated from 1 to 10.
* Updating will go to the targeted field based on customer ID, then the symptom table and the medicine will be given a review and rating. The average rating, along with number of reviews, will be display towards top bar of the pages
  + Two different type of client: regular user and administrator
  + Administrator can make changes towards core database
  + User can only update information and rating systems through reviews.
* The symptoms Table will contains all the symptoms for that specific illness. The symptoms will be linked to the client and Administrator table based on primary key with the same IDs field, names, and last names.
* Review will also be linked by primary keys to the client and Administrator table, from there, they will be linked to medicine, and symptoms.
* Medicine Sales values table will have the foreign keys to medicine table, and review.

# **Use Cases and Scenarios for the database**

The following has several possible use cases for the databases and its entities.

**Case 1: User ask for average rating on medicine.**

Return the average rating from Rating view.

**Case 2: User asked for highest rated medicine for a specific symptom.**

Return generic name of medicine with the highest rating.

**Case 3: Client writes a review for a medicine**

A new row is added to Review table with clients username, review, medicine reviewed, and rating.

**Case 4: User try to create a table**

Users cannot create new attributes or change attributes due to their restrictive rights.

**Case 5: User signs up to be a client**

Insert client into Client table with username, first name, last name, password.

**Case 6: Client signs in**

Input is verified and the client can see their view. If the information is not matching, client will be redirected to entering username and password.

**Case 7: Client signs out**

Closes the view and bring back the regular view which can only search and view basic information.

**Case 8: Administrator log on**

Input is verified and the administrator can see their view. If the information is not matching, client will be redirected to entering username and password.

**Case 9: Administrator logs out**

Closes the view and bring back the regular view which can only search and view basic information.

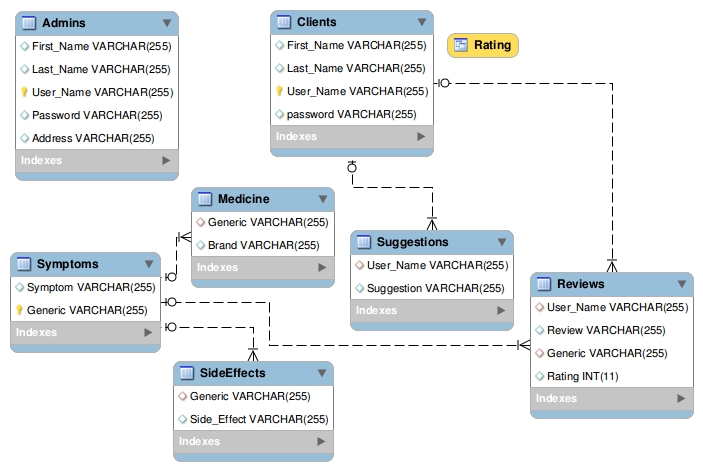
**Case 10: Administrator adds to the a table**

Inserts new record into the specified table.

**Case 11: Administrator deletes a record from the database**

Record is deleted from specified table.

# **E/R Diagram**



# **Database table design**

CREATE TABLE Admins (

First\_Name VARCHAR(255),

Last\_Name VARCHAR(255),

User\_Name VARCHAR(255) NOT NULL Primary Key,

Password VARCHAR(255),

Address VARCHAR(255)

);

CREATE TABLE Clients (

First\_Name VARCHAR(255),

Last\_Name VARCHAR(255),

User\_Name VARCHAR(255) NOT NULL Primary Key,

password VARCHAR(255)

);

CREATE TABLE Symptoms (

Symptom VARCHAR(255),

Generic VARCHAR(255) NOT NULL Primary Key

);

CREATE TABLE SideEffects (

Generic VARCHAR(255),

Foreign Key(Generic) REFERENCES Symptoms(Generic),

Side\_Effect VARCHAR(255)

);

CREATE TABLE Medicine (

Generic VARCHAR(255),

Foreign Key(Generic) REFERENCES Symptoms(Generic),

Brand VARCHAR(255)

);

CREATE TABLE Suggestions (

User\_Name VARCHAR(255),

Foreign Key(User\_Name) REFERENCES Clients(User\_Name),

Suggestion VARCHAR(255)

);

CREATE TABLE Reviews (

User\_Name VARCHAR(255),

Foreign Key(User\_Name) REFERENCES Clients(User\_Name),

Review VARCHAR(255),

Generic VARCHAR(255),

Foreign Key(Generic) REFERENCES Symptoms(Generic),

Rating INT

);

CREATE VIEW Rating AS

SELECT Medicine.Generic,

IFNULL(ROUND(AVG(Reviews.Rating), 2), 0) AS Rating

FROM Medicine

LEFT OUTER JOIN Reviews

ON Medicine.Generic = Reviews.Generic

GROUP BY Generic;

# **Test Plan**

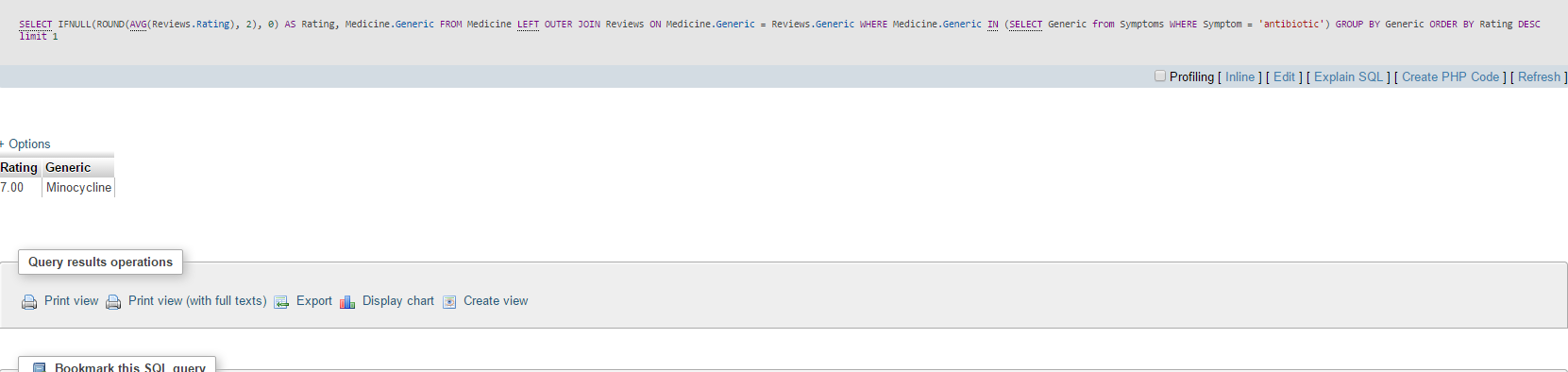
The test plan was to see if any of the functionalities work. This includes doing join tables, aggressions such as sum, average, and count.

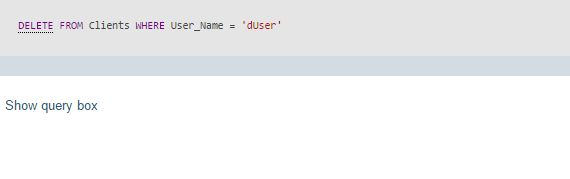
# **Testing Strategy**

The overall approach to testing this database is to enter as many queries to find the correct answer. Each function of the database will be tested. Searching for a will be tested by entering random medicine information. This approach will be repeated for all search functions. The update function will be tested by entering random details some will be false and verifying that the database interacts correctly.Each function will be run at least once and any errors will be written down. The user requirements will be tested. Functional testing will be performed to check the functions of database. The functional testing is carried out by feeding the input and validates the output.









# **Conclusion**

The database shows all the entities that correspond with the medicines. It displays the invoice that only correspond with certain symptoms. It lets the user update individual battery quantity and displays the symptoms as well as the rating for a specific medication.