CS 411 Project Proposal

1) **Project Topic:**

   OptimHealth: A Medicine and Pharmacy Finder

2) **Project Summary: It should be a 1-2 paragraph description of what your project is.**

   Our project idea is to create an app that helps people find the most optimal medicines from their closest pharmacies. We will algorithmically find the 'optimal' medicines through 4 factors: how well it treats the patient's symptoms, its side effects, its price, and the drug availability/distance required to get the drug. After the optimal medicines are found, we will also map the user to the closest pharmacy/supermarket that carries each medicine.

   We will leverage three primary databases on our backend to achieve this end. One contains pharmaceutical information about a plethora of medicines, including ratings and side effects. The second contains a list of pharmacies. To check for the closest pharmacies to a user's location, we will use the Google Maps API. The third (which is initially empty) will connect the first two databases by containing a list of pharmacies and the medicines they carry. We will gradually fill this database by making API calls to the pharmacies in the second database.

3) **Description of an application of your choice. State as clearly as possible what you want to do. What problem do you want to solve, etc.?**

   One sample application would be a Detroit resident who is suffering from a minor illness (e.g. the user has diarrhea, flu-like symptoms, etc).

   Currently, the patient would either need to go search on the internet for medicine or ask a doctor. There are drawbacks to both of these approaches:

   1. The user decides to see a doctor.
      a. Doctor's visits are often very expensive, and if the user believes that their symptoms are minor, they may be unwilling to go. Even if they do want to see a doctor, it may be hard for a user to schedule an appointment in a timely manner. Additionally, certain groups of

people (such as the uninsured or undocumented immigrants) may not have access to a general practitioner.

2. Ask the internet for help
    a. Internet results are often inaccurate and might even lead the user to hallucinate symptoms that they are not actually suffering from/give the wrong medicine.

Our system would solve this problem by allowing users to receive accurate medicine recommendations without having to see a doctor's office. Additionally, our program would base its medicine recommendation on specific details about the user, such as their location.

Going back to the Detroit patient with the minor illness, using our system, they would enter their address and what symptoms they are having (e.g. headache, cough with mucus, diarrhea, etc), which we would then use to find the most optimal medicine and also whether it requires a prescription, and also provide them the closest pharmacy from where they can get it from.

4) **What would be a good creative component that can improve the functionality of your application? To get a better sense of what a creative component is, these are technically challenging features that improve the user experience of your application. Some examples include interactive visualization (using several packages with some level or engineering), using several APIs to support some information presentation or using smart transformations to process data. Some examples that are NOT creative components include: software features that are completed with a few lines of code (e.g. adding a google maps iframe). Again, if you are unsure, discuss with your project TA.**

A good creative component we could have is to include visualizations of the price, distance, and efficacy (some combination of user ratings, side effects, and how well it treats the user's symptoms) of different medicines. For instance, if the user has flu-like symptoms, our website could list common medicines to treat those symptoms, such as ibuprofen, Tylenol, etc., and then create visualizations showing how each of these medications compares by price, distance to the user, and relative effectiveness, in addition to showing its optimal recommended medication. Another example of a visualization that we could achieve would be a heatmap of the US and its territories based on pharmacy locations. Additionally, we may also be able to extract some other interesting information from our Pharmacies dataset, such as state-wise coverage, which may help the user make financial decisions based on medical availability. For example, they can decide whether a certain town is a good place to live based on access to insurance coverage or whether they'll have constant access to an exact high-rated medicine in case they have a certain chronic medical condition.

5) **Usefulness. Explain as clearly as possible why your chosen application is useful. What are the basic functions of your web application? (What can users of this website do? Which simple and complex features are there?). Make sure to answer the following questions: Are there any similar websites/applications out there? If so, what are they, and how is yours different?**

   Finding the right medicine can be stressful and time-consuming. Patients need to find a balance between effectiveness, side effects, affordability, and convenience, but this information is usually scattered between many different sources. Our application simplifies this process by providing a single platform. Patients only need to enter their symptoms into a search bar. The app parses the input, recommends medicines, and maps nearby pharmacies that carry those medicines.
   The core features of this application include symptom-based search, medicine recommendations, location setup, and a pharmacy locator. Additional features include filtering by cost/side effects/etc and data visualization, allowing users to compare different medicines using graphs and charts.
   Existing apps include GoodRx (focused on prices), WebMD (focused on symptoms), and various pharmacy websites (focused on inventory at a single pharmacy). Our application is different because it integrates all of these different aspects into a single streamlined workflow. By combining medicine comparison, symptom checking, price comparison, availability, and mapping, our system makes the process of finding the right medicine as easy as possible.

6) **Realness. We want you to build a real application. So, make sure to locate real datasets. Describe your data sources (Where is the data from? In what format [csv, xls, txt,...], data size [cardinality and degree], what information does the data source capture?). It would be hard to satisfy stage 2 requirements with one dataset. Thus, we strongly recommend identifying at least two different data sources for your project, part of the dataset (i.e., one of the table) should have at least 1K data.**

   Our first dataset is the "11000 Medicine details" dataset from Kaggle. This dataset is a 4.37 MB CSV file, with 9 columns (degree) and 11498 rows (cardinality). This dataset contains important information about over 11,000 medicines, including, but not limited to, symptoms it treats, salt composition, side effects, the number of excellent/average/bad ratings, manufacturer and even image URLs. It is a dataset used by real medical professionals and researchers.

Our second dataset is the "Pharmacies" dataset, also from Kaggle. This dataset is a 39.2 MB CSV file, with 93 columns (degree) and 62974 rows (cardinality). It contains a large list of pharmacies within the United States and its territories: (American Samoa, Guam, Puerto Rico, the Commonwealth of the Northern Mariana Islands, and the Virgin Islands). It provides the full addresses and identification numbers for each one. We plan to leverage all this information to help with the user's procurement of their needed medicines.

Finally, as mentioned before, the third dataset (which is initially empty) will connect the first two databases by containing a list of pharmacies and the medicines they carry so that we can provide information about a particular medicine's local availability. We will gradually fill this database by making API calls to the pharmacies in the second database.

7) **A detailed description of the functionality that your website offers. This is where you talk about what the website delivers. Talk about how a user would interact with the application (i.e., things that one could create, delete, update, or search for). Read the requirements for stage 4 to see what other functionalities you want to provide to the users. You should include:**

    a) **A clear list of the functionality: Who [add/insert/update/delete] what types of data in your application when? (e.g., the user would submit a form with [what data] when they [want to do something].)**

    We want to make it simple for the user. The target user is a patient, and thus, we want to reduce the mental workload on their end. So, we will ask them to just enter their symptoms in a search bar. We would then clean and parse the input, and then search our medical database for the top N optimal medicines. When the user first uses the search, they would also need to input their home address, or allow for location sharing, so that we can find pharmacies near their location. There will be additional options to filter by price if they need to, as well as data visualizations to aid any additional research they may want to do. We also aim to provide the user with real-time data about the inventory that the closest stores to them have. To do this, we plan to use API calls to the pharmacies. While this depends on permissions based on the particular pharmacy chain, Walgreens, a major pharmacy chain, does allow developers to check medicine stock in their stores. At the end of this process, the optimal medicines and the pharmacies that carry them will be inserted into database 3 (the initially empty database). If the pharmacy is already in database 3, we will read the database to see if it is up to date (AKA did the pharmacy add any new medicines or run out of stock of any) and then update/delete tuples in database 3 as necessary, allowing for CRUD functionality. The user address will be stored locally (not in a database).

b) **A low-fidelity UI mockup: What do you imagine your final application's interface might look like? A PowerPoint slide or a pencil sketch on a piece of paper works!**
(below)

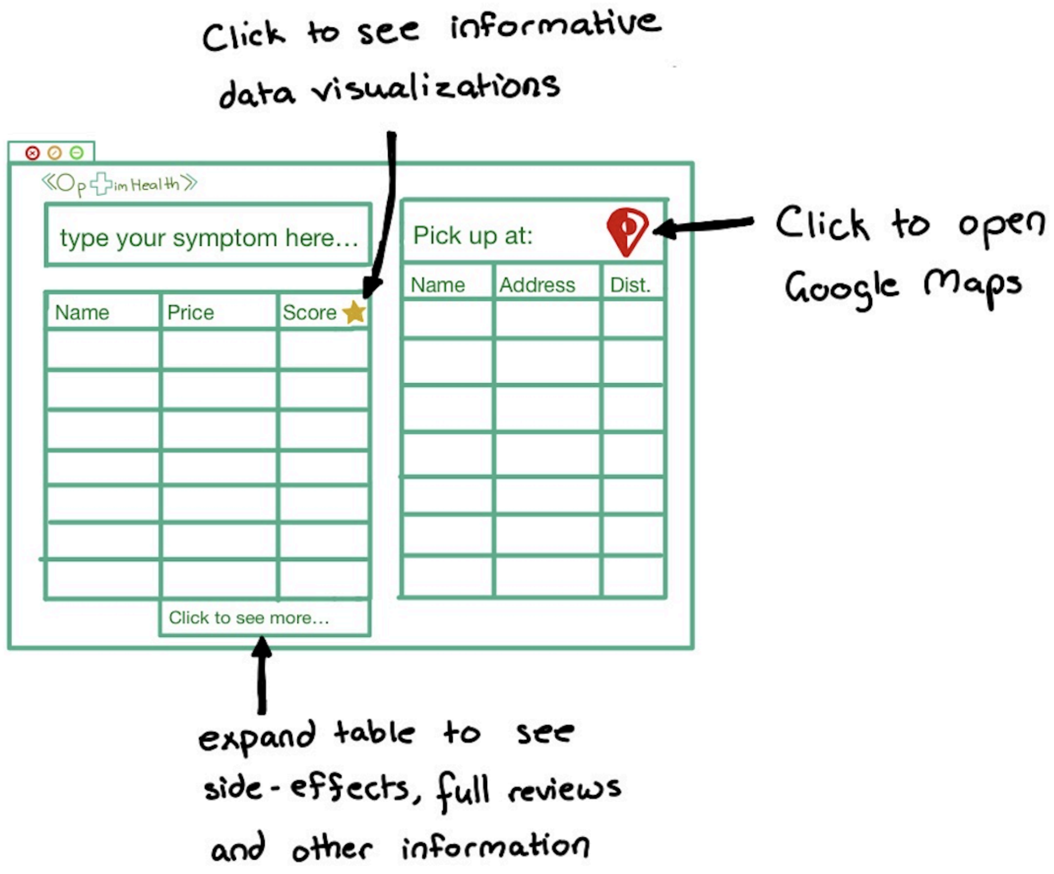
Click to see informative data visualizations
OptimHealth
type your symptom here...
Name
Price
Score
Click to see more...
Pick up at:
Name
Address
Dist.
Click to open Google Maps
expand table to see side-effects, full reviews and other information

Fig 1. The left table is the resulting list of optimal medicines while the right table provides the closest pharmacies they can pick the medicines up at. The star opens up a graph of numerical ratings, as well as some anecdotal ratings (reviews). There is also a sub-menu for more medical information and the pin opens up Google Maps.

In the final, high-fidelity prototype, there may also be toggles for light/dark mode and filter buttons to choose how to order data. By default, we would be returning medicines by Score DESC with Price ASC for tiebreaking.

c) **Project work distribution: Who will be responsible for each of the tasks or subtasks?**
Explain how backend systems will be distributed across members. Be as specific as possible as this could be part of the final peer evaluation metrics.

Ryan will be responsible for building the pipeline to integrate Google Maps or another type of Geographical Information System (GIS) API, so that we can calculate distances between pharmacies and the user. Ryan and Tuan will be working on implementing the frontend. Specifically, the will figure out the input parsing. They will also be implementing light/dark mode and the interactive buttons for filtering query results. Maahum will work on integrating our medicine database with our frontend, and Tuan's query software. She will work on ranking the medicine by price and rating, and also formulating a fair rating score formula. Vihaan will be working on integrating the pharmacy database with Ryan's GIS API. He will also be figuring out how to rank pharmacies based on their locations as well as optimized paths via transport modes such as trains. We all will be iteratively designing and passing feedback on everyone else's parts, so that we can build a robust and helpful product.

8) **In addition, if you lost points on Stage 0, you will be allowed to revise your Stage 0 to get points back for this stage. To do so, you will need to do the following**
   a) **Create new versions of the required documents for Stage 0 that contain the necessary revisions**
   b) **Create a new file named stage0_revisions.md in the doc folder that explains what parts of the original submission were changed and which comments they address**
   c) **Include the documents in your stage.1 release**
   These changes will allow you to earn up to 80% of the points you lost from Stage 0. For instance, if you got 11/13 on Stage 0, losing 2 points, if you address all comments you can earn up to 12.6/13, or 1.6 points back. Failure to follow these instructions will make you ineligible for earning points back. These points can only be earned back if you make revisions by the Stage 1 deadline.