Pavitra Aneja
Nico Luo
Caleb Yu
Luke Zhao
Team 46

Project Track 1 Project Report - SickSense

- Please list out changes in the directions of your project if the final project is different from your original proposal (based on your stage 1 proposal submission).
    - Broadly, our final project stuck to our original proposal pretty well, since as we stated there, our goal was to "create a web application that will utilize an AI chatbot to be able to help users with their health issues and provide medication by taking their symptoms and other medical information as inputs. The user will be able to set up their own account where they will be able to enter various things like age, gender, medical history etc. After inputting their symptoms, and with their saved information, the user will be shown probable diseases that they may be experiencing." We were able to accomplish this and while there were some changes in our database, which are listed in detail in the points below, nothing changed on a general level.
- Discuss what you think your application achieved or failed to achieve regarding its usefulness.
    - The application was able to achieve a lot of things, the most notable being disease prediction. Given three symptoms we were able to give a user a predicted disease along with any medication and other possible diseases. This was the main functionality of our web application. We were also able to provide user login/sign up, the ability to add disease and symptoms in case of new symptoms, and also a chatbot. But, one thing that failed to achieve was the chatbot feature. Our chatbot is not integrated into our database. It's still useful as it can answer any question the user may have. However, it lacks the connective feature and does not personalize it to the user.

- Discuss if you changed the schema or source of the data for your application
    - Our relational schema mostly remained the same besides a couple things. First, we removed the address attribute from the user table as we thought this data was too personal for our purpose and application. We also phased out the chatbot table, so this relational schema was removed from our database. The source of our data was not changed, but we did generate new data. We used the same datasets as we had initially gathered, but hand-generated new data using Python scripts in order to mimic user data. Therefore, we could use this user data for testing our CRUD operations and creating our disease prediction algorithm.

- Discuss what you change to your ER diagram and/or your table implementations. What are some differences between the original design and the final design? Why? What do you think is a more suitable design?
    - We made some necessary changes to our original UML diagram, like adding a password column to the Users table to facilitate a proper login system and moving the Rarity column to the Disease table as it fit it better. Other changes were made to streamline the database, like removing the unnecessary Chatbot table (as we handled the chatbot api through NodeJS only), adding upto 3 symptoms in our Diagnosis table, using the diseases as foreign keys in the Symptom table instead of the other way around, removing redundant columns such as DiseaseID (as DiseaseName is our primary key) and a description of the diseases etc. We also added a lot of more useful information to the Medicines Table such as composition, side effects, the most suitable uses etc.
- Discuss what functionalities you added or removed. Why?
    - We added some functionalities that were not in our original proposal like being able to change our database directly by being able to add new symptoms and diseases to improve the prediction algorithm for other users, as we felt this let us work more with CRUD operations in our database and improve the user experience. While we didn't mention it explicitly in our project proposal, we wanted to use the user information we get like gender, weight, age etc to be used in the prediction algorithm too which we were not able to do as we felt that our database was not large enough to add so many layers of filters.
- Explain how you think your advanced database programs complement your application.
    - The main functionality of our web application was based on advanced database programs. Our disease prediction utilized our database in order to make a prediction based on the three symptoms given. Our database contains a large amount of data regarding symptoms, medicine, and diseases and because of this, it allows for a more accurate prediction. We also incorporated a way for users to add diseases and symptoms to our database. This means that the prediction could change based on the information given by users.

- Each team member should describe one technical challenge that the team encountered. This should be sufficiently detailed such that another future team could use this as helpful advice if they were to start a similar project or where to maintain your project.
    - Luke - One technical challenge that I/our team faced was integrating our API's with the frontend. Because I didn't have too much experience on the UI side of things, even though UI was not a major factor in the project, I often used the wrong syntax or called the wrong backend APIs which would lead to invalid pages. Specifically, I didn't understand how div, body, form_actions worked so it

took several hours of understanding documentation to grasp the connection between frontend and backend. I recommend future teams to have an idea of frontend (specifically HTML) syntax and the remaining implementation of the UI should be fine to develop.

- Caleb - One technical challenge that our team faced was trying to integrate our chatbot to our database. This was unfamiliar to all of us so it was very difficult to understand what we needed to do. We were uncertain on where to start or even how to do it. In the end, we were unable to implement it within the designated time frame. However, it's possible that if we had more time we could have implemented it. If future teams want to implement something that they are unfamiliar/lack experience with, I would recommend that they start very early as learning new things such as this could cause a lot of unexpected delays.
- Pavitra - One technical challenge that our team faced was getting some frontend elements properly working, like dropdown menus for selecting symptoms for the prediction algorithm or when adding new symptoms as HTML syntax was quite new to me. Once I got the prediction algorithm working to show a result on our webpage, another big problem was the formatting as I had to figure out how the output of the algorithm (the table data) was being parsed on the frontend side, and I had to spend a decent amount of time realizing that the way that stored procedures return their outputs is different to just writing the SQL query. I think the most important lesson I learned was that the frontend should be designed in a streamlined and pre-planned manner so as to not make things complicated as changes are made throughout the project.
- Nico - One challenge we encountered was figuring out how to collaborate using GCP while maintaining our database at the same time. Since we each ran an instance on GCP on our own local machines, we had to use some sort of version control, which was Git. But, the GCP interface led to a very slow workflow, since editing using the GNU nano panel was quite tedious. Removing a large chunk of code or refactoring what we had took much longer than it could have taken on an everyday IDE. Some of us wrote code locally, but this was also suboptimal since the database was stored on the cloud, so we copied and pasted our code into the GCP instance to test it. One piece of advice for future teams would be to code their frontend and server locally and also have access to their database locally. This would make testing significantly easier and less time consuming.

- Are there other things that changed comparing the final application with the original proposal?
    - One thing we changed was the removal of salting and hashing of passwords. Although this would have added more security for our platform, there were a few difficulties when it came to this and we found it somewhat unnecessary for the

level of our application. We also ended up removing the graphic visualization of the user's past diagnoses. Since all of us were pretty inexperienced with frontend development and how to store things in databases, it was difficult to actually implement this feature. We didn't include a FAQ page or an "About Us" page as seen on the mockup of our application. This was because we wanted to keep the application solely focused on diagnosing patients and their symptoms rather than adding fluff material.

- Describe future work that you think, other than the interface, that the application can improve on
    - Our team originally planned to apply several additional ideas to our project but due to lack of experience and time, we decided to hold off. One idea we had was to integrate our MySQL database with our chatbot, so the chatbot could have access to our database and provide information to the user based on the data. We felt that this would provide a more significant meaning of having the chatbot beyond answering various questions the user had. Another idea we planned to implement was providing the purchase link to the medicine associated with the disease; in our case, we were able to provide image links of the medicine but a "cooler", more helpful component would be to have a hyperlink that takes the user directly to the site to purchase it (for the ones that are over-the-counter at least). The last component our team believes we could improve on is collecting better datasets. While our datasets had hundreds of diseases and symptoms in our database, we felt that some diseases had missing symptoms that should have been linked, which may give incorrect predictions to the user, not because of our querying algorithm but because the data itself may be insufficient.

- Describe the final division of labor and how well you managed teamwork.
    - We managed teamwork very well as a group since we planned designated meeting times each week (or several times within the week). Everyone provided their availability and we scheduled meetings based on those times. Since our team had people that were experienced in a variety of areas, we worked together as well as split the workload based on each of our strengths. Nico had experience on the UI/UX side so he was responsible for designing the clean graphics and buttons on the homepage as well as the other pages. Caleb's experience in frontend and query filtering/searching allowed our project to easily search and filter diseases and symptoms based on text inputs, enhancing user-friendliness. Pavi's experience in backend and databases meant that he was able to efficiently create backend API's and integrate the advanced queries and stored procedures from stage 3 into our project seamlessly. Luke's experience in backend helped him create backend APIs for adding new symptoms and diseases to the database and also integrating a

chatbot based on OpenAI's GPT 3.5 model. In general, whenever one of us was struggling, others could step in and help achieve the desired goals and there was constant communication/participation from everyone.