Salvatore Canale  
Mengyi Chen  
Adonis Davis  
Joshua Diehr

Mental Health System Report

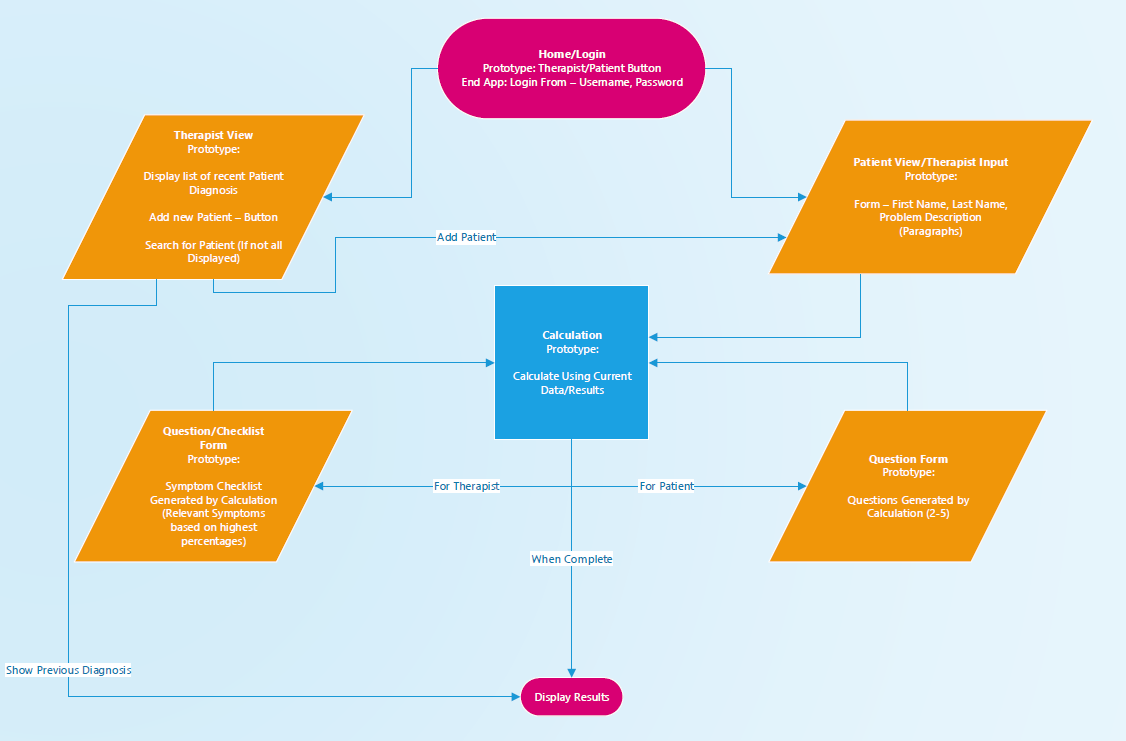
**Project Overview:**

The intent of this project was to make a tool to assist mental health professionals with a psychiatric diagnosis. The current most accepted method of diagnosing mental health disorders is interviewing patients and associate symptoms with disorders described in the DSM manual(which is the Diagnostic and Statistical Manual of Mental Disorders). The interview process is commonly referred to as the “Initial Psychiatric Assessment”. The most commonly accepted method of performing the initial psychiatric assessment is with a SCID interview guide built from the DSM. The SCID is the Structured Clinical Interview for DSM-5 Disorders (SCID). Both tools used together allow the psychiatrist or therapist to take the symptoms from the patient and make an educated diagnosis of the most closely described disorder(s). Currently, this is all done manually by the mental health professional. The intent of the new tool is to automate the diagnosis process and build upon a collective set of real-life diagnoses to create a better tool for diagnosing and potentially grow the knowledge of these disorders.

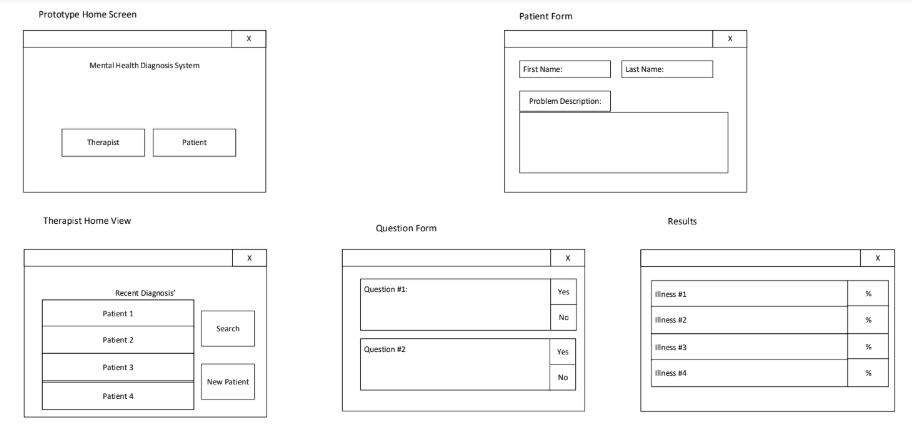
**Project Workflow:**

The project was divided into four main parts: GUI, Text-Mining, Deduction Engine, and the Database. Each member was to work on one or two of these parts. There were some initial struggles we had to overcome as a team before work could even begin. We divided up the work into these four equal parts and allowed everyone to select one or two areas of interest. We initially thought we had Bonita as a resource and that she already had a start on a usable database. After a few weeks of communication with her, we then found out she wasn’t going to work with us and didn’t have any database for us to use. Josh took the lead and built an access database and built all the java code to interact with the database. While this was going on, Mengyi was also working on the problem of how to import the text from the DSM into a usable format in the database. The requirement was to make this a fully automated process, not manual entry from the DSM to the database. This proved to be a much more involved problem and the final solution we ended with was still a compromise. The compromise is that we do have a programming solution to the problem but it imports more than just the wanted keywords into the database. We did not think that was ultimately the best implementation but it is a good first step to a better solution to the problem.

Now that those two crucial steps were completed, we began solving the other two main components of the project. How to design an interface? How to calculate the diagnoses? We built all of the project using Java. The GUI specifically was built with Java Swing libraries. This is a universally accepted and clean looking graphical user interface. Before we began programming the GUI, we needed to know what screens were needed, how those screens should look, and what would the workflow be for every user within those screens. The first step was to design and layout the workflow. View workflow below.



After we knew the workflow, we needed to know what screens were needed. This was easy to know since the workflow dictated which screens were necessary. The harder part was designing how the screens should look. What information was needed and helpful to the user at which step in the diagnosis. View screen mockups below.



Now that workflow and screen design was completed, we were able to begin coding the GUI.

The previous three components needed to be completed prior to the deduction engine development. The deduction engine is what we are referring to the code and algorithms responsible for doing the actual diagnosing. We are deducing down from the master list of disorders to the disorders matching the patient’s description and showing what percentage of that disorder the patient matches with.

The three main components within the deduction engine:

* Patient description of questions
  + Takes the input of the patient description in the form of a single string object.
  + Runs the description string through a set of code that cleans the string and outputs only keywords matching the keywords in the database and associates them with their value from the database
  + Value and keywords passed back to the deduction engine
  + Takes the values and keywords and adjusts scores for each disorder based on provided keywords
  + Ranks disorder scores highest to lowest
  + Writes disorder score and rank to database
  + Reads in the table of questions from the database
  + Decides which questions to ask user based on keywords that were not used by the user
    - Example: User entered keywords sad and depressed. In order to adjust the score, the question is related to the keyword lonely. Asking a question about being sad or depressed would be redundant.
  + Passes questions to ask the user back to the GUI
* Question/answer to summary
  + Takes the input of the user's answers to the previously described questions
  + Uses the answer keyword values to adjust the patient's disorder scores
  + Recalculates the rank of the user disorder scores
  + Writes the new disorder score and rank to database
  + Passes the new disorder score and rank to the GUI
* Summary to final diagnoses
  + Takes the input of the diagnosed disorder from the GUI
  + Writes the diagnosed disorder to the database

**Project Conclusion:**

Thus far, this project has produced a working prototype of a program that utilizes Microsoft Access to save, add patients, and input patient diagnosis. Although the deliverable for the project meets the requirements for setting a rough shell, it is by no means a comprehensive program. There are still many improvements that can be made with the project.

**Project Follow-up Improvements and Future Work:**

Here are some of the things that the team thinks the program can be further improved:

* The project was a concept, so we only included three disorders
  + Add in all disorders from DSM based on the blueprint of how we implemented a project for three disorders
* Split up disorders and sub-category of disorders within a disorder
  + Example of Attention Deficit Disorder. There is a sub-category disorder of Attention Deficit Hyperactive Disorder within the Attention Deficit Disorder description.
* Further, develop code that automatically took DSM write-up and converts to keywords in database
  + Part of the problem is the way the DSM is written. It is written to be read like a book and less like a diagnostic manual. There are long paragraphs describing a particular illness or symptom, not a bulleted list or something more conducive for our application.
  + Also was an issue that we were using a photocopied DSM converted into a text document and not a true electronic copy of the DSM manual.
* Add functionality to record other symptom keywords that we do not have in our database of keywords and associate them with the disorder for future diagnoses
  + Example: If every time someone is diagnosed with depression they include the keyword “sleepy” but we don’t have the keyword sleepy in our database, either allow the doctor to add that word to the database or automatically add it based on some form of the machine learning algorithm. This will allow the program to truly become smarter as it diagnoses more people.
* Look into other ways of managing a database. Limited by making it a local database - cloud, server, etc...
  + We used Access for a local database that would be simple. It could be investigated in making the project more scalable to the real world with a server or cloud database. This would also allow more background functionality to run that could use the real-world data to help improve the diagnosis using machine learning.
* Have a method of access restriction.
  + Adding an element of anonymity would be needed. For example, when anyone else other than the doctor is accessing patient information (for example a pharmacist or any other user) should only be able to view data on a need-to-know basis. Any other information that is not necessary should be hidden.