Trey Thompson, Brady Underwood, Isabel Nelson and Kait Flinn
ESOF 423 - Final Project Portfolio

DxMood

Section 1 - Program: Despite advancing discussion, mental health remains to be an urgent nationwide problem. Mental healthcare is widely inaccessible, especially in remote areas, like Montana, with significantly fewer resources. After working in the medical field for many years, Dr. Philip Bain thought of a solution: an algorithm to diagnose mental illnesses. A patient would take multiple screening tests from the comfort of their home, and Bain's algorithm would compare each test score to result in a diagnosis.

We have taken Bain's algorithm, and created a web-app for doctors to input these test scores, resulting in a diagnosis and medication recommendation. These results are stored in a database accessible for health professionals, eliminating much of the tedious paperwork for both patients and medical professionals. Doctors will always have discretion over a proper diagnosis, but the algorithm aims to save time, delivering faster results and solutions. This will also allow doctors to treat more patients more effectively.

Section 2 - Teamwork:

- Kait - Primarily focused on setting up a database for patient and doctor information. After looking into different options, Firebase (by Google) seemed the most feasible option. As a group we used a code-based approach, which created some roadblocks in connecting Firebase to our backend. We decided Firebase was not the best option, and switched to setting up a database using SQL. Kait also observed most of the user testing as well.
- Brady - Assisted in adding Axios to the frontend and beginning to make api calls to the C# backend through GET and PUT. Also worked on writing javascript and to make the html interactive.
- Trey - The primary contributor to the backend C# development of the full-stack DxMood application, utilizing API calls and creating models. He also implemented the DxMood Algorithm for the diagnosis and suggested medication.
- Isabel - Primarily focused on the frontend design of the project. Started by making a low fidelity prototype on paper and sketched out what we wanted the application to look like. Then made a high fidelity prototype through Figma adding font and color to the design. After both prototypes were complete, used html and css to create and design a new patient page, patient page, and doctor account page.

Section 3 - Design Pattern: In the DxMood project, the Model-View-Controller (MVC) pattern is used, which can be viewed in the Models and Controller Directories. This pattern was selected because it is easier to maintain, test, and modify the application's components independently. It also allows for easy integration with other frameworks and tools that are commonly used in .NET web development. By separating the application into the Model, View, and Controller components, the code can be organized in a way that makes it more modular and easier to understand. This makes it easier to work on different parts of the codebase independently without affecting other parts of the application and in turn allows for better scalability.

***Section 4 - Technical Writing*:**

User documentation
Instructions for using application:

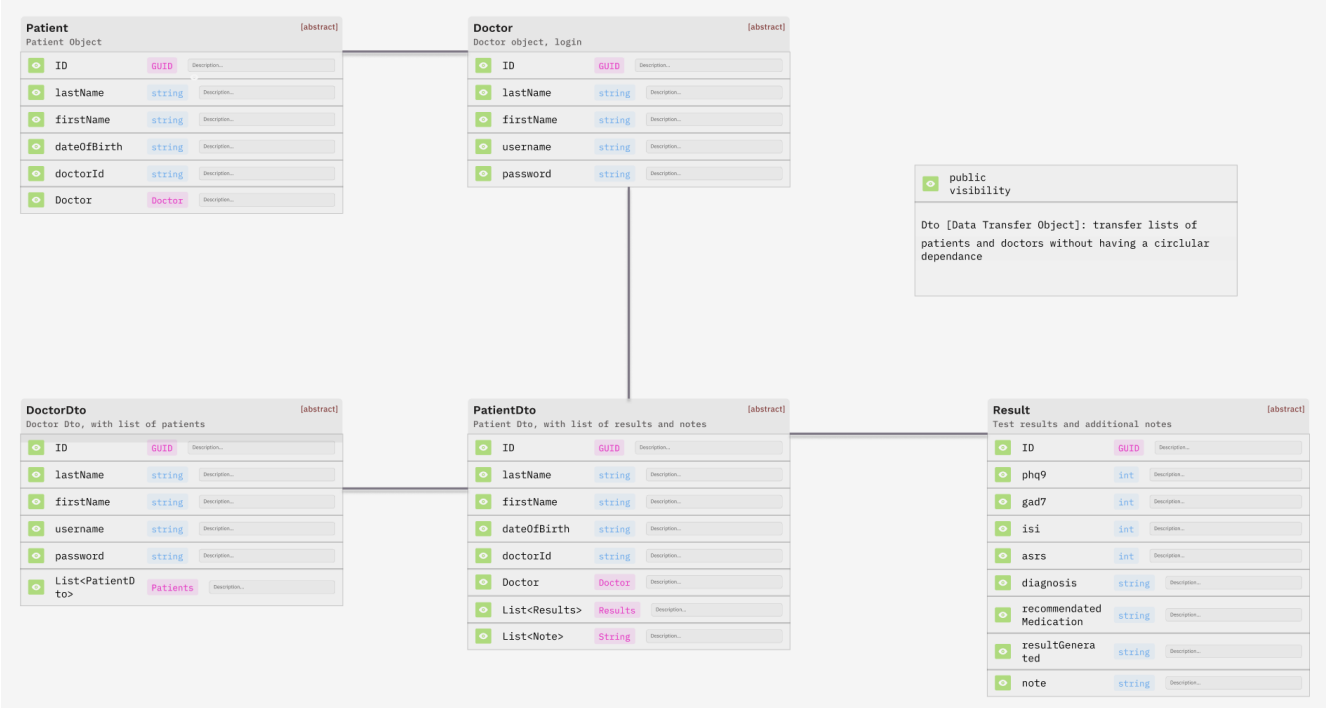
- The application is stored in a zipped folder which you can unzip to reveal an executable, you can run the executable by double clicking
- After opening the application in your browser, you will see a welcome page along with two buttons directing you to “Log In” or “Sign Up”.
- You can play around in the navigation bar and see where different features will be.
    1. **DxMood button** - takes you to the login/signup page
    2. **New Patient button** - takes you to a screen that will allow you to add a new patient. Here you will be able to fill out a new patient's information, as well as enter their scores for the various tests. Once the information is filled out, you will be able to hit the “diagnose” button. This is also where you will be able to see their diagnosis based on Dr. Bane’s DxMood Algorithm.
    3. **Patients button** - takes you to a list of the doctor’s patients. Here you will be able to click on a patient’s name and pull up their information and history within the DxMood software. A doctor will be able to see the patient’s diagnosis, their list of medications, and any notes that have been created under their name. Here, the doctor will also be able to create new notes and add updates to the patient’s information.
    4. **Account button** - takes you to the doctor's account for the DxMood software. A doctor will be able to see their account information and how many patients they have.

Developer documentation:

- DxMood Brief Overview:
    - A diagnosis assistance and recording tool for doctors
    - A full stack application with a C# .NET backend and HTML and Javascript frontend
    - API can be exposed at /swagger/index.html
- Install:
    - All that is needed for installation for the project is .NET framework
- Clone:
    - `$ git clone https://[LASTNAME][FIRSTNAME]-admin@bitbucket.org/esof423-drrx-bitk/dxmood-bitk.git`
- Start:
    - `$ cd DxMood`
    - `$ dotnet run`
- File structure:
    - wwwroot: Static Frontend Code
    - Controllers: C# Backend Code
    - Services: C# Backend Code

- API
    - We use Swagger API to help visualize our data models
    - When running a localhost this can be accessed at /swagger/index.html
- Link to Github: https://github.com/423s23/G5-DrRx
- Link to Product Website: https://dxmood.mystrikingly.com/
- Database:
    - Initial Product was set up with a public database for presentation and testing purposes **ONLY**. To fully implement this product for it to work in the future, a new server and database will need to be set up. Here are the instructions to add a new Azure Database to the backend. **Future Authentication steps may need to be taken to properly connect the database.**
        - Create a new server on Azure Services
        - Create a new database in that Server
        - Copy the connection string that is given to you and paste it into "appSettings.json" in the DxMood Project folder.
            - Replace MultipleActiveResultSets = **False → True**
        - Run the following migration commands in terminal
            - `Dotnet ef migrations add init`
            - `Dotnet ef database update`

Section 5 - UML: DxMood Class Diagram:



Section 6 - Design Trade-offs: One of the tradeoffs we had in our design was choosing a code first approach. While this allows for the database to always be in sync with our code, it added some overhead when initially setting up the project. The primary benefit of this approach is once the project is set up we can always rely on the database to be correct which speeds up development time dramatically. A different tradeoff we had was choosing how to design the front-end, the options were between using a framework such as React/Blazor or using vanilla HTML and Javascript. With .NET (our backend service) it is easiest to serve up static files and if we used a framework we would have to build our code into static files every time we wanted to serve it with .NET. On the other hand, with static files we are able to update the code and immediately see changes in the backend. The tradeoff is slightly slowing down development time and requiring more javascript to be written.

Section 7 - Software development life cycle model: Our team is using Jira to sprint plan, develop sprint goals, and create a burndown chart. We are using BitBucket to upload our code. During each sprint we create new branches to work on individually and merge them at the end of the sprint. Once the sprint is complete, we merge our project into GitHub.

Jira and Bitbucket have both been very helpful tools for our team to use. After planning each sprint and splitting up tasks, we are able to move our tasks into an "in progress" folder, "code review" folder, and "done" folder. This way we can clearly see who is working on what portion of the sprint and what phase they are at in the process. These software tools have been helpful in holding us accountable for the work that needs to get done as well as has allowed us to track progress amongst group members.

Going forward we will continue to use both Jira and Bitbucket as it is something that has been working really well for our team. We have had no issues with it and all like the model we are using.

Section 8 - User Testing:

Our group conducted user testing with Neeve McCarthy and Macie Hopkins. We started by giving the users our laptop and told them to play around with the application. We watched them go through the app without giving any prompts or helping them when they had questions. We had them do this for around 20 minutes and then proceeded to ask a series of questions.

Given login credentials, both users were able to open the project and login with ease. They then navigated through the site, testing out the different features. After 20 minutes, and we asked the following questions:

- How do you add a patient?
- How do you open a patient's file?
- How do you add a new result to an existing patient's file?
- How do you delete a patient?
- What did and did not make sense while using the site?
- Did you find the site easy to use?
- Is there anything you wish was different about the site?

With thoughtful answers from our participants, here are our biggest takeaways:

- Adding a patient is a straightforward process that the users were able to do easily. "The 'Add Patient' was super easy, I was able to add a new patient with all their information without any problem" (Hopkins).
- Both users had no issues opening a patient's file. They were able to navigate to the patient page, see the list of patients for Dr. Brady, and open the patient's file using the "view" button.
- McCarthy had a hard time adding a new result to an existing patient. When recalling how to do it, she was unsure what button in the navigation bar she was supposed to click on. After going back through the application, she was able to navigate back to the patients and find the button and add a new result. This may be something that users need to get used to as they use the application over time.
- One thing that was challenging for both users was opening the page to see an empty home screen. It was unclear whether they should wait for something to load or if that was the page entirely. They both eventually moved on to add patients and navigate through the site. With this observed confusion, we are changing the opening page to show the logged in doctor's current list of patients. This will hopefully aid in making the site flow better. Overall, operating the site was simple and doable for each participant, "the process of adding and deleting patients was really straightforward" (McCarthy).
- The participants agreed that with an introductory walkthrough, the site would have given them no problem whatsoever. Moving forward, a simple overview of the program or brief written instructions would set up users for the most success.

After testing the application with two separate users, we learned that our web app was ultimately easy to use for non-computer science individuals. With feedback from the participants and observed behavior, we will make a few minor tweaks to ensure the highest quality product.