ModernMigraines: Tracking your Migraine

NICHOLAS FRIES, University of California San Diego, United States
DELIA MCGRATH, Trinity College Dublin, University of Dublin, Ireland
JULIAN STRIETZEL, Karlsruhe Institute of Technology, Germany

Additional Key Words and Phrases: migraine, tracking, alexa

ACM Reference Format:
Nicholas Fries, Delia McGrath, and Julian Strietzel. 2024. ModernMigraines: Tracking your Migraine. 1, 1 (February 2024), 21 pages.

1 INTRODUCTION

The general idea of Modern Migraines is to enable the users to track their Migraines. Through storing and analyzing when a migraine occurred and even more data (e.g. how many steps a day or the weather) the user can analyze which circumstances influence physical health. The application is build to warn the user if a day with a higher risk of a migraine is coming up. Therefor the user can take more care for himself, reduce stress and take his medication if needed.
In section 2 we will introduce our motivation and the background of our project. This includes a brief summary of how migraine effects the life of patients and which parts of population are affected. We will also explain how our application is supposed to help and why existing applications fall short on solving those problems.
In section 3 we will describe our design and the process of designing our application.
In section 4 we will focus on the system we developed including its architecture, the technology we used and the features we implemented. Designing a maintainable and expandable architecture has been a major part of our project as we integrated a lot of different technologies which had to be integrated seamlessly.
In section 5 we will describe how we tested our increments and final product. We will finally evaluate how good our system meets the migraine tracking user's requirements.
In section 6 we will look back on our collaboration and the process of developing the system as a team. We will focus on how we split the work and how we solved the problems that occurred. In the last section we will conclude over our product and look forward to future work that could improve the benefits of ModernMigraines for the users.

2 MOTIVATION AND BACKGROUND

A major motivation for this application comes from one of our own team member's struggles with migraines. Over the past 5 years they have tried various treatments to try and address the migraines. These treatments have ranged from

Authors' addresses: Nicholas Fries, University of California San Diego, San Diego, United States, nfries@ucsd.edu; Delia McGrath, Trinity College Dublin, University of Dublin, Dublin, Ireland, dmcgrat4@tcd.ie; Julian Strietzel, Karlsruhe Institute of Technology, Karlsruhe, Germany, julian.strietzel@student.kit.edu.

changing their diet, to taking medication, and to even undergoing surgery. Of all the methods to prevent migraines, the most helpful tool for them has been tracking and trying to better understand the causes of their migraines. However, due to all the different factors that can trigger a migraine they are still trying to learn how each factor impacts their health even after 5 years of headaches and visual auras.

Approximately 12% of people deal with migraines. Migraine symptoms are very diverse, ranging from headaches to nausea to visual auras, and can prevent someone from being able to accomplish everyday tasks. While there are treatments for migraines, most medicines work as preventative measures but fail to stop a migraine that has already started. This is largely because migraines are not well understood yet. Because of this, it is vital for people who struggle with migraines to know what triggers the start of their migraines and also for them to take medicine at the first sign of the migraine. The aim for our app is to equip the user with a tool that can provide warnings of migraines and help reveal that users specific triggers without causing interruption in the users daily life. If we can successfully warn the user of an incoming migraine we give them the chance to start taking preventative measures early, significantly decreasing the chance of a migraine from happening as well as limiting the severity of migraines that do occur. These preventative measures will differ based on the specific user but could involve them limiting their screen time for the day, taking a nap, keeping their head elevated, taking medicine, and finding ways to limit their stress. This application can especially help a user know when to take medicine. As already mentioned most migraine medicine is preventative, but many migraine medicines also cannot be taken too often. For example, a common migraine medicine currently is named Imitrex [8]. If someone uses Imitrex more than 10 times a month they can become reliant on it and have migraines on days they aren't using the medicine. They also can start having more headaches due to medication overuse. With the help of ModernMigraines the user can have increased awareness about which days they should use medication like Imitrex and which days they can get by without it.

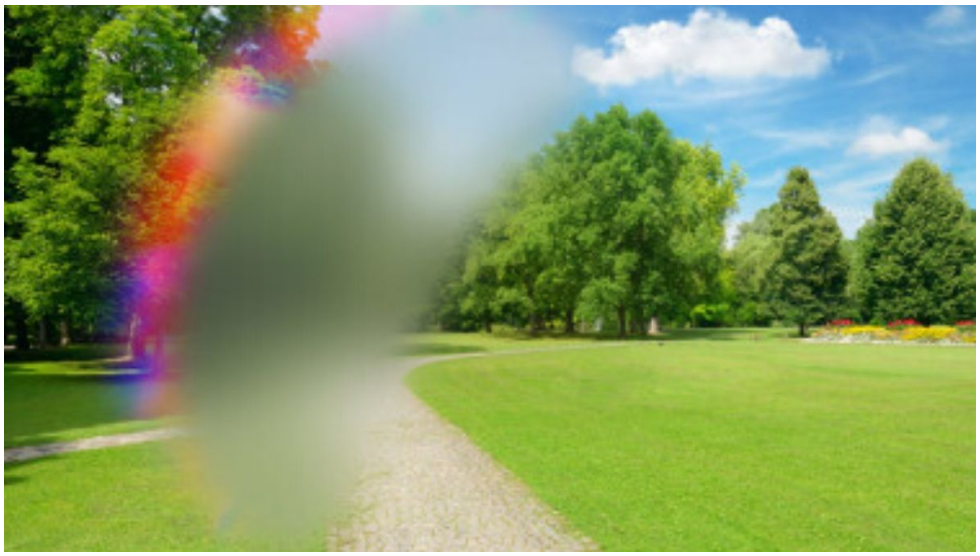


Fig. 1. Example of how a migraine aura affects someones vision. [14]

Many types of migraine trackers currently exist, however many fail to predict migraines and identify triggers while others intervene too much in the users life. Simple migraine diaries help users keep track of when their migraines occur and identify potential triggers. While these can be useful, they are limited due to limited data. All data is given by the user rather than being detected by sensors. This can be a major problem as well because the user may be extremely light sensitive during and after a migraine but needs to look at a screen to input data. Too much screen time can trigger a migraine by itself so limited the user's interaction with their phone is vital. Along with requiring the user to focus on a screen, migraine diaries don't process the data but leave it up to the user to find patterns. On the opposite end, there was a new app released in 2020 that uses AI to help prevent migraines [16]. This app attempts to reduce migraines by showing the user their body signals in an attempt to relax them. It also aims to be able to predict for individual users when they will get a migraine. The drawback to this app is that rather than collecting data constantly through everyday devices (like a phone) it relies on two sensors, one that the user must place on their neck muscles and one that is placed on their fingertip. The user must take 10 minutes everyday to use these sensors and attempt to train themselves to relax in order for the app to help. Along with interfering with the users day this device is also more expensive compared to other migraine trackers due to the required sensors. Another restriction to this app is that while it attempts to predict migraines through body signals and reducing the users stress, it fails to monitor external factors like weather which is one of the biggest triggers of migraines. This is largely due to it having a different focus than our project. It attempts to be a treatment on it's own while ModernMigraines attempts to provide the user with vitale information that can be used to decide on the best treatment.

Both migraine dairies and the AI migraine app fail to track enough factors that trigger migraines. Some of the main triggers of migraines include atmospheric pressure, humidity, and sleep. The AI migraine app does not track these, while migraine diaries rely on the user to manually log all this information which can lead to incomplete and/or inaccurate data. Secondly, migraines are largely triggered by change, making it important to track data everyday and not just on the days that migraines occur. When a user is required to input all the data, they may know the weather and their sleep and screen time for the day but trying to remember this information for the previous day or two can be challenging and cause more inaccurate or incomplete data. By using sensors that are present in our everyday devices to track the data-points everyday we can record more data, be more accurate, and monitor daily changes in the factors the trigger migraines without putting an extra burden on the user. Another advantage of our app is the ability for the user to use Google Assistant. This can help reduce the amount of time a user looks at a screen while also allowing the user to answer questions easily and quickly. Currently, there are no commonly used apps that work with Google Assistant or Amazon's Alexa to help track migraines.

3 DESIGN

The initial design for the app was to implement a seamless network between devices that would be able to gather data in multiple ways and combine them for the user. Initially, the idea was to separate this by creating a watch section, a voice section, and a mobile application section. It was thought, initially, that the voice and mobile sections may overlap, however, as a person may prefer to track on mobile rather than the voice system, or vice versa.

While the separation for this app is correct, this diagram based a belief that an application on the watch would have to be built in order to collect data about the person. As further research was done on the health app, it was decided instead to use a software that scraped data from apple health instead. This meant that instead of having to manually get the data for each thing, the software simply asks to use the data that is already gathered in Apple Health. This is

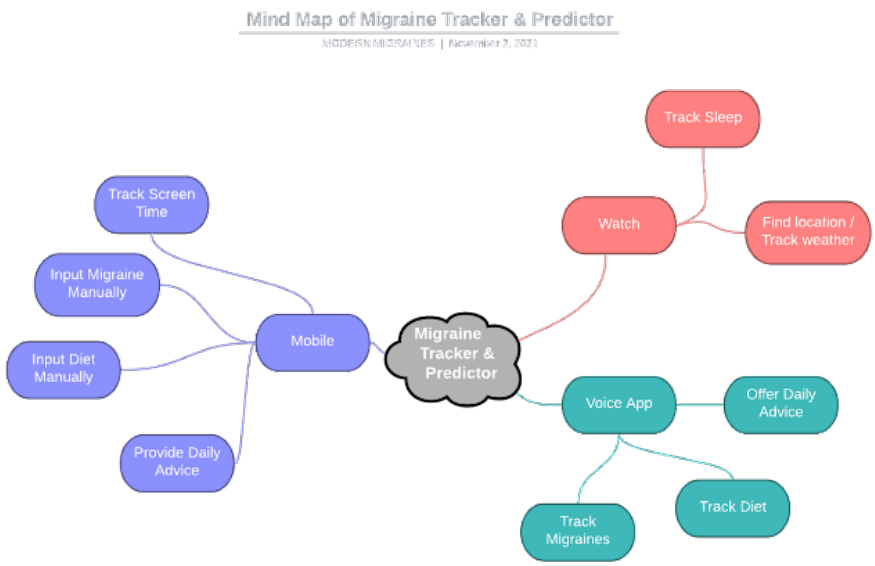


Fig. 2. Initial design interaction of our system.

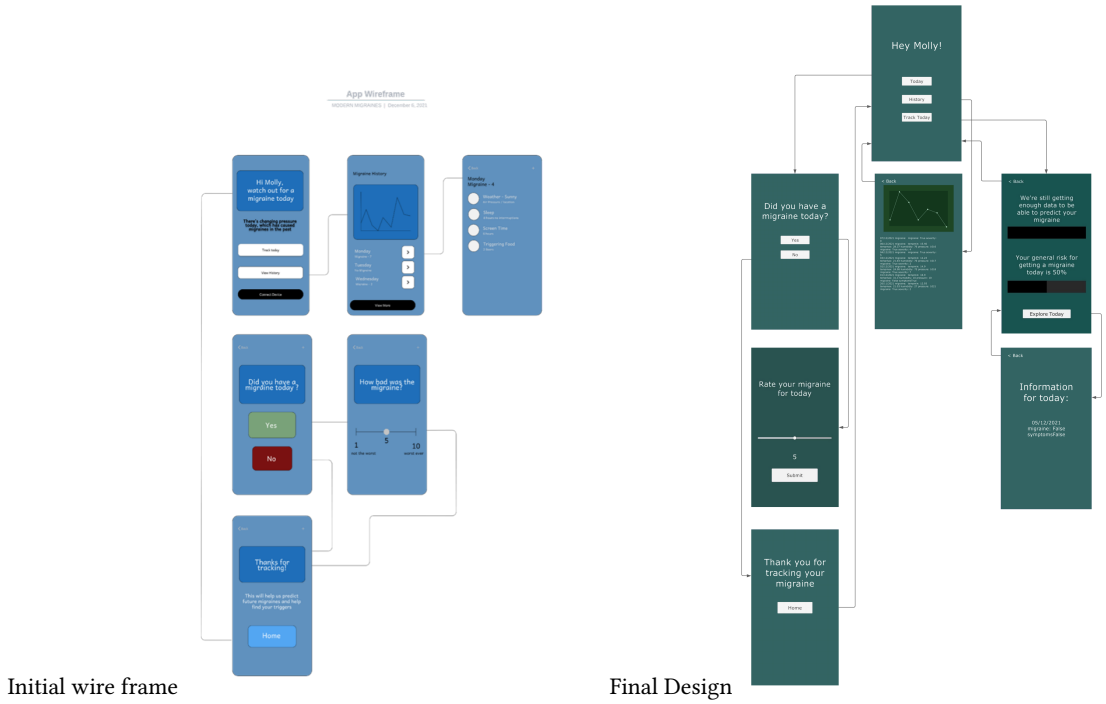
particularly useful as many people may not want to add an additional app to their watch, which makes our migraine predictor have a more seamless integration to everyday life.

The initial prototype of the voice functionality came from working with Voiceflow in the early stages of the course, where we were experimenting with different products. It was designed in such a manner to ask the person first if they had a migraine, then ask them to rate it. However, after further thought to the project, a question about whether the person had migraine symptoms was added to best see what the triggering events are. Especially because some people that experience migraines, may at times experience the symptoms of a migraine, without having an actual migraine.

For the mobile application, it was initially thought that it would provide more features, such as the ability to track certain foods. However, as the database evolved into storing apple health data and migraine data in a way that was simple, it felt as though designing an app with the ability to track food may take an excessive amount of time and take away from the core functionality of the app because what oftentimes the foods are person-specific, which makes the database more difficult to track.

When prototyping the initial wire frame for the app, the goal was to make something that would be easy to use for the user. While many apps have many buttons and icons strewn throughout the app, that would not be beneficial to those that may be prone to migraines. In particular, if someone were to want to track their migraine as they had it, they would be less inclined to go to the app if it would be hard to use and required lots of input. In order to counteract this, the app was made with easy to follow pages and large text and buttons from the start.

As the app was constructed, research was done concurrently when trying to find the color the app should be and what font should be used. When looking at the font, Verdana was chosen due to studies about font and type-spacing done by Hojjati and Nafiseh, which demonstrated that Verdana was the least straining on eyes and the most easy to



Initial wire frame Final Design

Fig. 3. Maturing of the Designs

read on computers [19]. As eye-strain could worsen migraines for some, the font that was suggested in the study was implemented within the application. In other research done by Noseda and others, it was shown that green light has the ability to alleviate migraines slightly [22], while in research done by Bernstein and others it was shown that darker colors were better for those with migraines[15]. Both of this research was noted and combined to make the dark green backdrop of the current app.

Beyond the particulars, it was decided to make bar charts and line graphs of the data gathered, so the user could see visually how their migraine was. While the bar charts were an addition, much of the format was similar to the initial wire frame because the initial wire frame focused on simplicity and ease, which was something that wanted to be kept throughout the development of the app.

When developing the application, there was a process followed, so that the time would be spent on the most valuable parts of the application, and so that there would be regular deadlines for people to follow in order to stay on track with the project.

Because of the size of the team, there felt a need to make deadlines for both individuals and the team. Certain parts of the application relied on only one person, but integration relied on the whole team as it relied on work and expertise from the multiple areas. Thus, the timeline was effective because it was made with the workflow of the team in mind.

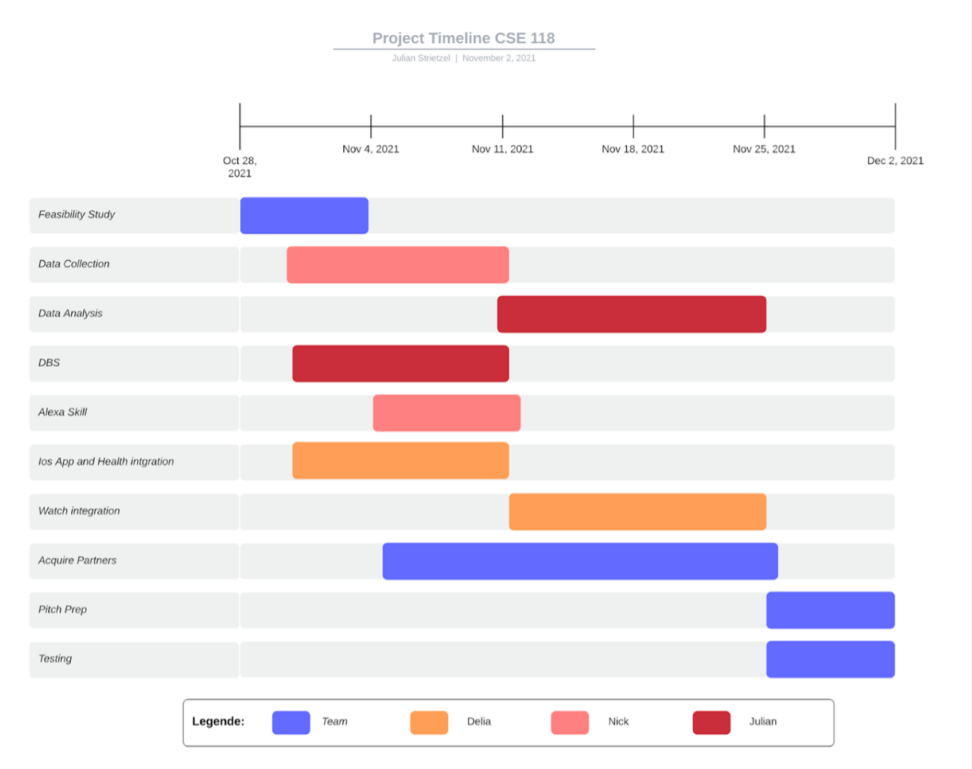


Fig. 4. Timeline for developing the application.

4 SYSTEM DEVELOPMENT

To design a system that allows the integration of all the heterogeneous components we planned to use has been a challenge. We needed to find a reliable, usable, scalable and cheap database system which on top of that had to be compatible to all our components. Preferably this database should be accessible via the internet.

We also needed to find a user interfaces which allowed the user to track the migraines. As the accuracy of our system would rely on this data we had to ensure that the user would use this on a daily basis. Therefor this device had to be deeply integrated in the users daily routines. Priorities for the interface were usability and reliability.

To improve the accuracy even more we planned to integrate as many data sources as possible. The usability of our product relied on this data not being manually tracked by the user. As there is already a lot of data generated by our smartphones we planned to read this data and integrate it to our predictions and analyzes.

Last but not least we needed a central application on which the user could see his risk scores and visualize the data provided.

4.1 Architecture

4.1.1 Client-/Server-Architecture. As a basis we are implementing a classic Client-/Server-Architecture [20]. This means that the system is build around one central server in our case the Google Firebase Realtime Database. This Server is

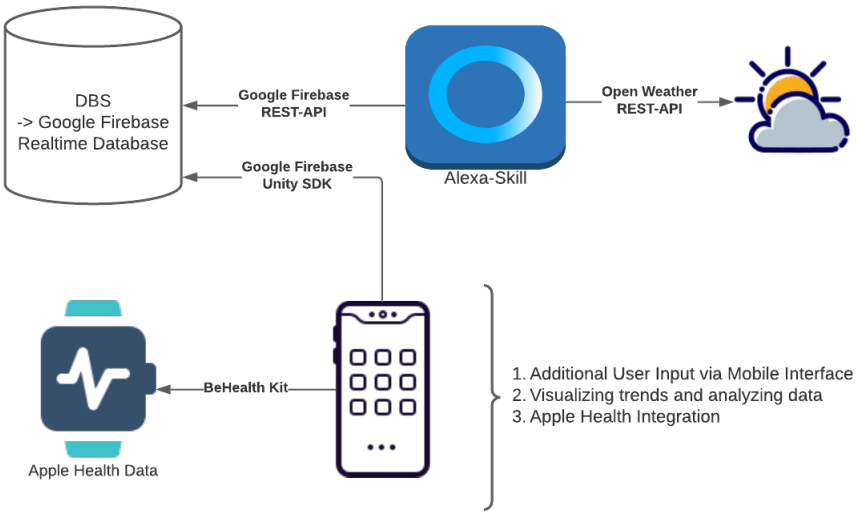


Fig. 5. Architecture and components of ModernMigraines

always running and a central instance for all clients. Those are only communicating via the server to each other. The clients around it can access it to write or read data.

The clients to our server are the entities of the VUI and our application. Those generate and/or use the data about our users migraine, the weather and apple health. To do so they send requests via API-Calls or the SDK to write and read information to and from the database. You will find more about that at the Technology Used section 4.2.

4.1.2 Input. We have two main sources for input. On a daily basis our VUI asks the user about his physical health and the severity of his symptoms. The VUI also adds the weather to our data by using openweather map. This way we have the necessary external data in our dataset independent whether the user is opening the app that day. We still rely on the user speaking to his voice assistant every day though. This problem could be solved in future work by the app updating that info in the background or using a historical weather API.

The second input is our Apple HealthKit integration. As the iPhone of our user constantly tracks steps, screen time and more relevant data. Apple Health step count data has proven to be reasonably accurate [9]. We can fetch this data every time the user opens the app and store that to our local app files. We are pulling that data from the Apple HealthKit via the BeHealthKit for Unity.

4.1.3 Output. VUI and Database are mainly responsible for generating and storing the data for our migraine tracker. Afterwards the application is responsible to analyze the data and show relevant information. We chose this part of our system to show the output of our data on purpose. While the voice interface is useful for asking the user easy questions and storing that to our database it has it's restrictions in providing complex information (reading out all the data we tracked would not be useful).

The output can be separated in two categories: The risk assessment and the data visualization.

The main goal of our application is to calculate risk scores for personal and general daily risk of a migraine. These scores represent a combination of factors that enhance the risk of the user getting a migraine, as pressure, temperature, steps and trends in migraines.

The general score looks at the features and analyzes how they effect migraines in general. The personal score includes individually marked triggers and there impact to the migraine risk of the specific user. More information about risk assessment can be found in the prediction section 4.3.2.

Feedback gave us the insight that providing the user with his risk scores via the google assistant could be useful. This way the user does not have to look at his screen but is automatically provided with the information. The risk could be presented by the VUI in simple low, medium, high, extreme risk categories.

We are also providing our user with graphs showing the variation of different features over time. This will help our user to understand patterns in his health history and extract the key features that are responsible for giving him a migraine.

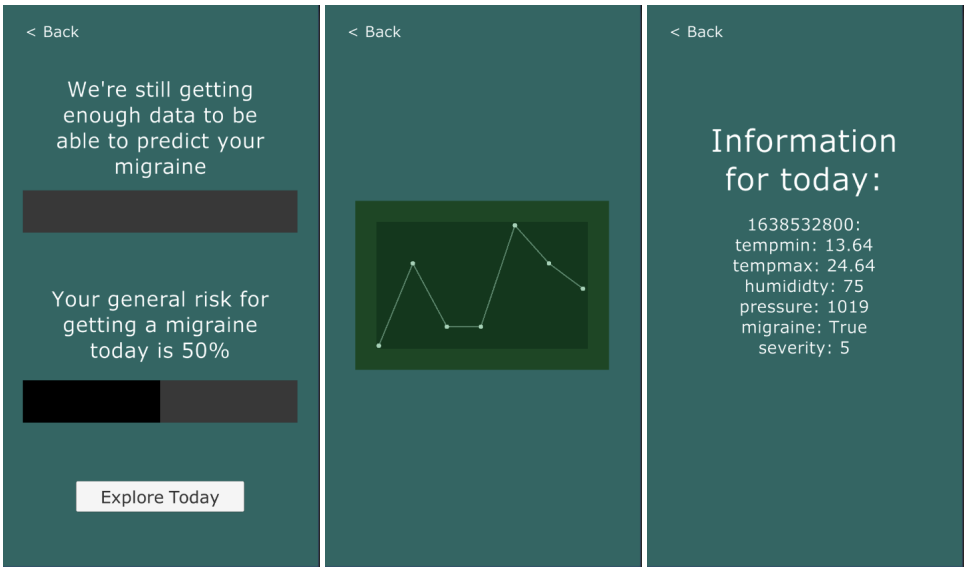


Fig. 6. screens in application

4.1.4 Processing. As our server is only a database server and the data is only coming together completely in the application all our processing is done locally. With growing performance in mobile devices this will not be a problem. Further information in how we are using the data to extract triggers and predict risk scores can be found in 4.3.2 the prediction section.

4.1.5 Security & Privacy. As we are handling critical and private health data security and privacy is an important consideration in our system architecture. Luckily we are using established standards, connections and operating systems which have security and encryption implemented by design. Firebase Realtime Database has the so called Firebase Realtime Database Rules integrated which can define a security layer on our server based on authentication and structure of our database. These rules make it easy to enforce user permissions. For example the following rule would

```
{
  "rules": {
    "users": {
      "$uid": {
        ".write": "$uid === auth.uid",
        ".read": "$uid === auth.uid"
      }
    }
  }
}
```

Fig. 7. Rules for the realtime database providing access to subcollection for only one specific authenticated user

make each user's collection only accessible by the authenticated user himself 7. We are planning to integrate the Firebase authentication to manage users and authentication methods in the future. Firebase security has been evaluated and certified by externals [10]. The SDK and API calls are verified by an individual API key in combination with the bundle identifier which has to be recognized by the Firebase server. Rules and authentication in combination with the certified security of Firebase infrastructure will be a strong mechanism to protect the private data in our database from attackers.

Furthermore we don't really have to worry about local security of our data on the iPhone [3]. All iOS versions since iOS 4 have a built-in security feature called Data Protection. It allows an app to encrypt and decrypt the files stored in their app directory. The encryption and decryption processes are automatic and hardware-accelerated. Data Protection is available for all kinds of files and enabled by default. So we can be sure that our data stored locally on the iPhone is secure.

So far we are not using any identification method in our application nor our voice user interface. We can never be sure whether we are really providing access to the right person or only the right device. On the phone most people use biometric identification or a code/password to lock the access to their private data on the phone. Nevertheless, in the context of voice interaction privacy and security concerns rise [23]. Personal responses of voice assistants based on voice matching [2] technology has been an interesting field of development. This could be implemented in future versions to regulate the access to personal migraine tracking.

Privacy is another important concern for us as we are handling critical data. Ensuring that no unauthorized access to our data is possible is not enough. We also have to ensure that we don't store unnecessary data and the user knows how we handle his data. Till now we have a minor problem with that privacy aspect. Our cloud service provider (Google) can access all our data in the Firebase and we are limited to the privacy regulations of Google which were often criticized [18]. On top of that privacy in cloud services in general has been an issue [17]. In our case this is a trade off between security, convenience, reliability and compatibility and privacy. In the limited time provided we were not able to integrate better privacy in our product. Nevertheless, we were already trying to implement Privacy by Design our application. We do store as less data as possible in the cloud but prefer storing it locally on the device. This is a reason why we do not push all the added HealthKit data to our Firebase. In the future we could additionally delete all the data we pulled from the Firebase afterwards. The Firebase would only be used as a buffer between input and output client. Then there would not be the complete health history of our users in the cloud which would be better concerning

privacy. This again will be a trade off between privacy and convenience as this will increase the risk of data loss and make switching between devices more difficult. However, this is something to be focused on as privacy within apps becomes increasingly more important [21].
Another idea could be to switch to another cloud service with better privacy.

4.2 Technology Used

4.2.1 Google Firebase - Realtime Database. Google Firebase is a Toolkit for cloud based app and web development. The Firebase Realtime Database is a cloud-hosted NoSQL database that lets us abstract from the servers and infrastructure behind it and focus on the implementation of our functionality. Realtime Database ships with mobile and web SDKs which makes it easy to use with the rest of our system. There is a SDK specially for Unity and the documentation is very helpful. As parts of our team had already worked with it this was an easy decision to not setup our own servers and self-host a SQL-Database but just use the server-less google cloud infrastructure.

In our database we will store the weather and migraine data that our VUI generates. This can later be pulled from the application to be used for analyzing. The phone does only store new data to the Firebase if the user decides to track his migraine not by the VUI but in the app. The health data does not need to be stored to the cloud as it is managed by the operating system of the device.

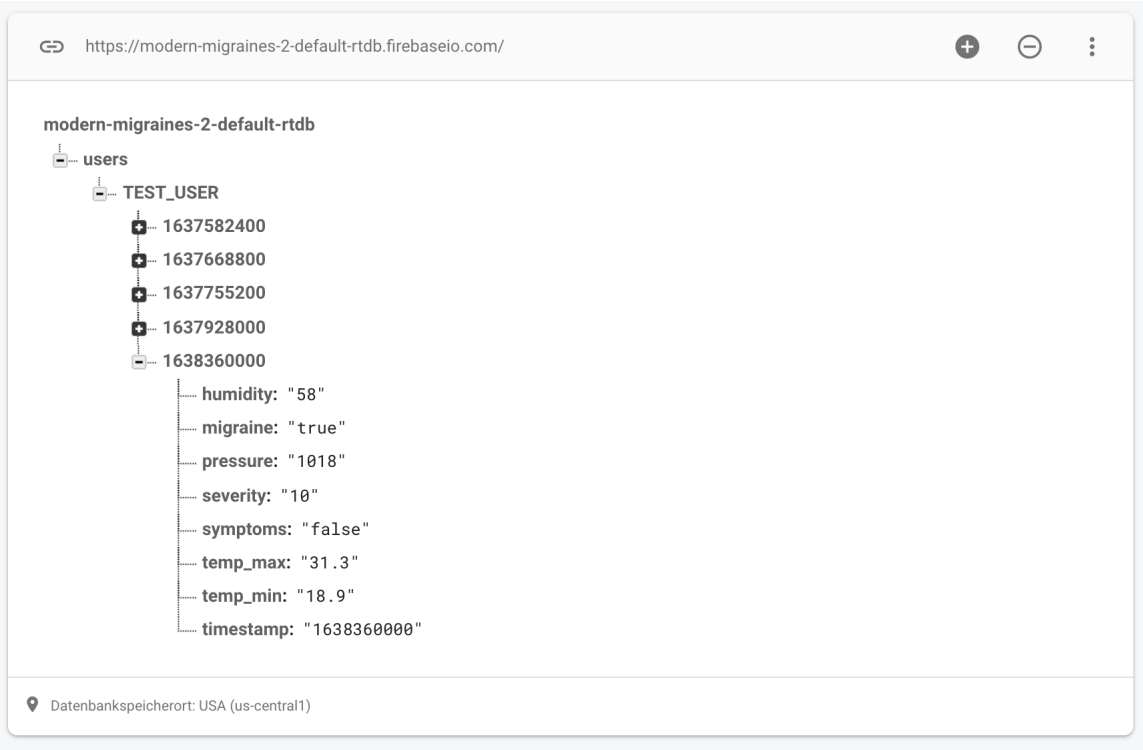


Fig. 8. Database structure with demo data for a test user

We implemented two ways of interaction with our Firebase: The first is the Google Firebase API which provides a REST interface to GET, PUSH and PATCH json files to/from a specific position in the database. REST stand for REpresential State Transfer and is an architectural style for distributed hypermedia system [13]. REST APIs are build to include all information necessary for executing a task in the request itself without the necessity for the caller to store any state data. This makes it very handy for our purpose as we can include it as a lightweight call in our VUI and we are using this to patch our tracking data from the VUI to our database. This has in general proven to work fast, easy and reliable. After deploying from voiceflow to Alexa we had some problems with compatibility as the API did not work in the Alexa Developer Console so we had to rebuild our voiceflow for Google Assistant from scratch. That fixed our problems as we were now able to write to our Firebase.

The other way of interacting with our database is the Firebase SDK for Unity. We used a childAdded- and a childModified-listeners [refs] to fetch the data from RTDB. This allowed us to get updates from the database in real time without manually refreshing the app. This is relevant as we based our risk scores on that data. We also push data via the SDK to the Firebase when our user is tracking his migraine in the app and not via the Google Assistant. Integrating the Firebase Project to Unity was still not as easy as advertised as some problems with mysterious "CocoaPods" impeding our building process.

Our Database stores the data organized in collections for every user (8). This ensures that user-data is not confused and each user can only access his personal data. In every users collection there is a document for each day the user has used the migraine tracker. This document contains the weather data from the openweather-API, the fact whether the user had a migraine this day, whether he had symptoms and the severity of his migraine. The identifier for each document is the UNIX time stamp for noon at that specific date. This id gives the documents a natural order and is sufficiently identifying in as we only want to store one data point for each tracked day. If the user uses the tracker more than once the data will be overwritten for that specific date. So far our database does only store data for one user as authentication is not implemented in the application nor in the VUI.

4.2.2 Voiceflow. Voiceflow is a program that is used to create new skills for voice assistants to use. You can design for multiple different voice assistants we decided to use Google Assistant. Originally we planned to use an Amazon Echo, however the Echo failed to send patch requests to our database. This could be due to the database we are using being Google's Firebase.

To start the skill on a Google Assistant, the user will say "Hey Google, start migraine tracker." and call our start tracking intent. (While the action is still in development we have to say "Talk to Migraine Tracker." Or "Talk to demo.") Migraine tracker prompts the user asking if they had a migraine today. This question opens a slot for a new migraine "true" or "false" variable. If they respond yes, it asks the user to rate the severity on a scale from 1 to 10, ten being the worst, 1 being a mild migraine. If they respond no it asks if they had any symptoms at all. After receiving the user's responses a patch request is sent to the database and adds the new information.

For each answer there are several utterances implemented. The user does not have to say exactly "Yes, I had a migraine today." "Yes", "I did" or similar will also be recognized.

Furthermore we integrated shortcuts for the experienced user. When somebody gets used to the migraine tracker he probably does not want to walk through the complete conversation every time and he does not have to. At the first choice block the user can also choose the yes_skip, the no_no_skip or the no_yes_skip intent. For example if the user

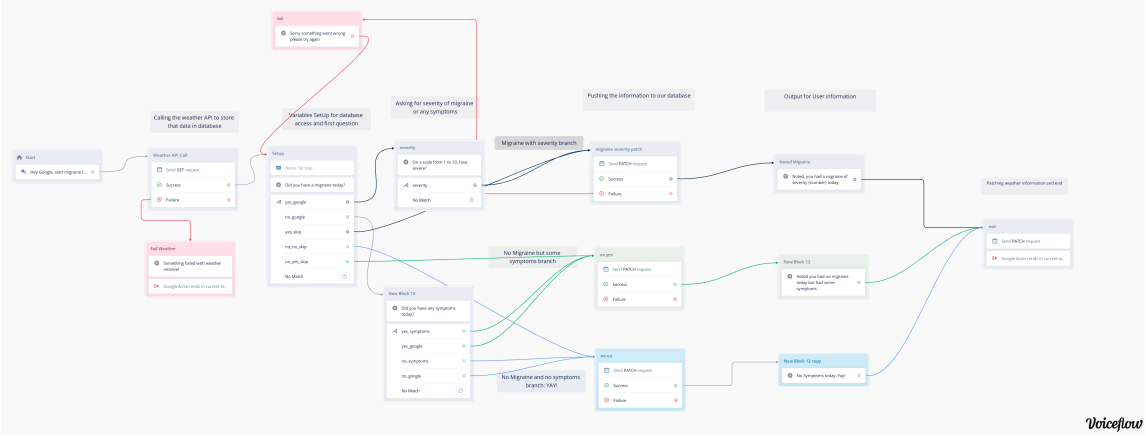


Fig. 9. Voiceflow diagram of the voice assistant skill.

says "Yes 10", the system will skip the question for severity and will directly push the information. This enhances the usability of our VUI.

By using the voice assistant in general, we help the user answer the questions that ModernMigraines needs to know without the need for the user to look at a phone screen. This enhances the usability of our system. Furthermore we call our weather API within the voiceflow application and push that data to the Firebase, too. This ensures, that we have the weather data for every day, even when the user does not open the app on his phone at all.

4.2.3 Google Assistant. Google Assistant is Google's virtual assistant and pendant to Amazon's Alexa and Apple's Siri. It is primarily available on smartphones and smart speakers but also in android auto. Via the Google action console developers can extend the functionality of the product with Google actions. Those are similar to Alexa skills. Intents are used to let the user do something, utterances are different ways the user could ask for something and slots are variables that can be filled by the user input [5].

4.2.4 openweathermap API. We are using the OpenWeather API to get current weather information for the users location. OpenWeatherMap is an online service, owned by OpenWeather Ltd, that provides global weather data via API, including current weather data, forecasts, nowcasts and historical weather data for any geographical location [6]. The are a team of weather and IT experts. They are using a convolutional neuronal network and provide more than 2 million users around the world with weather data since 2014.

We are focusing on the aspects relevant for a migraine. Those are especially humidity, pressure and temperature [12]. The API provides us with weather data all around the world based on coordinates or the names of over 200,000 cities. In our current state of the application restricted the usage to the area around San Diego and are only fetching local weather. The data is provided in a json file (see below).

Accessing historical data costs per API call. So we can't add weather data to historical tracking data. We have to call the current weather data every time a user tracks his information and add that data to our Firebase for future usage. This is done via our voiceflow application.

Example request resulting in 12.

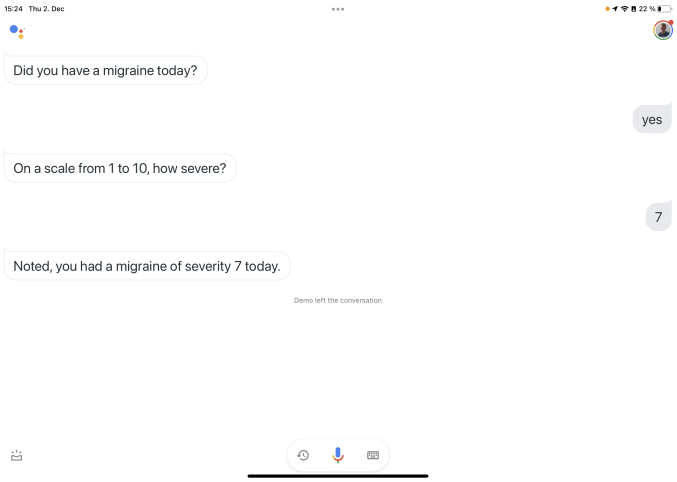


Fig. 10. Voiceflow Migraine Tracker on Google Assistant for iPad

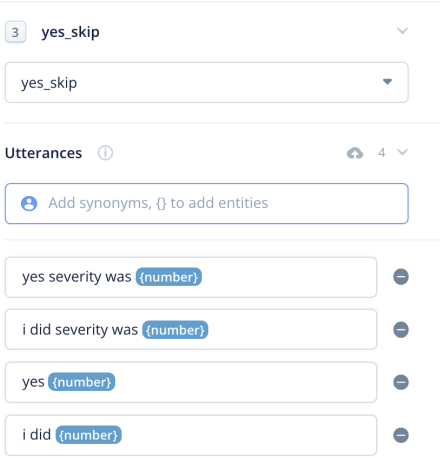


Fig. 11. Utterances for yes_skip intent in first choice block

```
https://api.openweathermap.org/data/2.5/weather?lat=32.71574&lon=-117.1611&appid=f5029e208...&units=metric
```

4.2.5 Unity. Unity is a development environment specifically for 3d games. In the context of mobile gaming they make it possible to build to iOS and Android from the same code base. This is an outstanding feature as most other environments for mobile development are specifically for one operating system (see Android Studio or xCode).

Unity is structuring an application in a lot of game objects. Those represent each text, block or player object on the screen. They are organized in Scenes which represent one specific screen in the application. Scripts, imports, objects and more are categorized as assets and can be attached to game objects. Coding scripts is in C.

Apart from the cross platform compatibility we expected unity to work seamlessly with Firebase our database and Apple HealthKit. There is a unity SDK offered from Google Firebase with integration compatible both for Android and

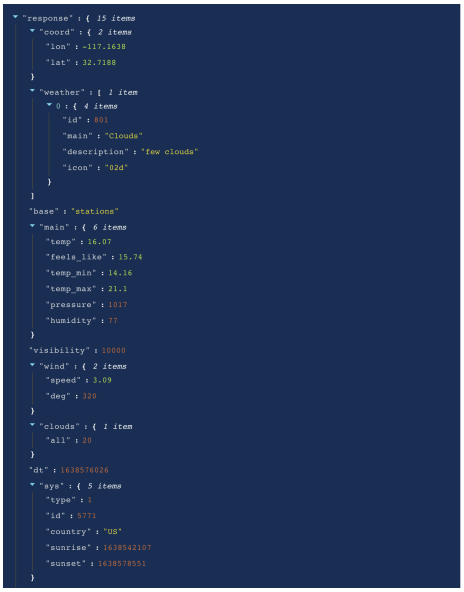


Fig. 12. json response to API call

iOS [1]. There is also the BeHealthKit which allows us to access the complete HealthKit Data within our application [4]. Both integrations where easy to install and well documented. Nevertheless, in development and building we had a lot of problems with both of them.

One major flaw of unity mobile development is that you can not directly build to iOS. You always have to build a xCode project and afterwards build from xCode project to the phone. This workflow is kinda inconvenient as it takes a lot of time and throws a lot of errors. We especially fought with a package manager called "Cocoapods" associated with the Google Firebase. It was failing from Unity and preventing us from building a working xCode project. Weirdly it worked when run separated in our project folder which has become our workaround after hours of testing. The BeHealthKit worked fine in our Unity project at first. When building to xCode we realized that the HealthKit dependencies were not properly configured. On top of that we realized that our xCode accounts were not licensed to work with HealthKit and even the University License did not include that. For this reasons we had to abandon the idea of integrating HealthKit to our demo on the iPhone. In Unity it does work though, as the BeHealthKit contains demo data for testing and debugging.

We see the general advantages of developing with Unity even for mobile applications. Nevertheless, when you are not relying on the 3d features of it but want to develop a simple application, we'd suggest to do so with an IDE especially for app development as Android Studio or xCode. If you really want to develop for multiple platforms with the same code basis you might have a look into flutter or similar. Sadly Unity has not proven to be intuitively usable for our use case even for people who are familiar with mobile development.

4.2.6 Apple HealthKit. Apple Health is an application integrated to the apple iOS ecosystem which tracks and analyzes personal health data like step count, calories, nutrition and heart rate (depending on external devices like apple watch). This App is supposed to be a central health hub integrated to all health and fitness tracker apps on the iPhone to make the experience more seamless.

The Apple HealthKit is framework for applications to access and write that information to Apple Health. For us the step and sleep data were interesting as movement and sleep have a major impact on migraine risk. We used BeHealthKit to integrate Health access to our Unity project.

4.2.7 BEHealthKit. BeHealthKit is a Unity plugin to access the Apple HealthKit framework from Unity apps for iOS. As Unity does not natively offer HealthKit integration we had to use this plugin to access any data from it.

The plugin costs $40 in the unity asset store and enables the developer to read and write quantities like steps, sleep, heart rate, calories, nutritional data and blood pressure. This is done by attaching the class HealthStore and HealthKitDataTypes to a game object and authorizing them by calling HealthStore.authorize(HealthKitDataTypes). In the last we defined all the data types our application would like to access. This should start a popup on the iPhone asking for the permissions to access the specific data. As we had problems building the HealthKit part we could not test whether it worked like that in praxis. Afterwards the HealthStore could be called to access all authorized data types from HealthKit. We integrated that to our demo in Unity with the integrated demo data from BeHealthKit. This data is very handy as it let's you test the general working of your application in Unity even though no real data from the phone is available.

4.3 Features

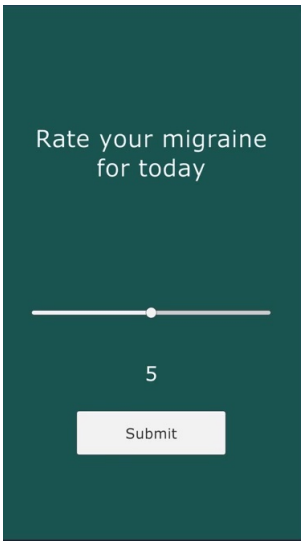


Fig. 13. Tracking the migraine in the app

4.3.1 Recording. The key feature of our app is the migraine and trigger tracking. Tracking can be accomplished via the Google Assistant or our application. When tracking a migraine the user is also asked for a severity (see fig 13). If

they did not have a migraine they will be asked for any other symptoms. Tracking their personal health will allow the user to analyze trends and recognize migraine triggers (see fig 6). This is beneficial because it helps the user better understand how to limit their migraine severity and frequency. All the triggers that are currently tracked are done with sensors in the phone. This is an important feature because there are too many triggers for a user to easily track them all on their own. Having a single application that tracks all the important data for a migraine makes migraine management easier for the user.

The Google Assistant adds current weather data to the data points. This data is fetched from openweather api (4.2.4). Our application also automatically adds relevant data from Apple Health (4.2.6) to our tracked data to increase accuracy of our risk assessment.

4.3.2 Prediction. In order to assess the users risk for a migraine, ModernMigraines examines the averages of the various data-points. It gives the user a personal risk rating and a general risk rating. If the everyday average of a specific factor has a significant difference from the average on days where a migraine was reported, that factor is flagged as a trigger for that user. Everyday, ModernMigraines compares the data for the day with what the averages are for each factor. If there is a significant difference and that factor has also been flagged as a trigger then the user's personal risk increases. The final personal risk rating is based upon how many total triggers have been identified for the user and how many of those triggers are active that day. There are situations where ModernMigraines will not give the user a personal risk rating. This happens if the user has used the app for less than 10 days or has reported less than 3 migraines. This is because ModernMigraines needs data from the user to assess their risk, and if there is not enough data to reliably do this we do not want to give misleading information. Also, if no factors have been flagged as a trigger there will be no personal risk rating given because no triggers for the individual user have been identified. Regardless of how many days the app has been used ,how much data has been collected, and how many triggers have been identified the user will always be given a general risk rating. This rating is not based on the specific user but rather is given based off of all factors recorded for that day even if they have not been marked as a trigger. The purpose of giving this general risk rating is to provide service to the user even when personalized results are not available. The longer a user has ModernMigraines the more accurate it's risk ratings will become.

4.3.3 Weather and Health Data Usage. As explained earlier weather and activity have an effect on the migraine risk of the user. We are including our tracked data together with the migraine tracking and severity in our calculation for the risk scores. This data includes temperature, change in temperature, humidity, change in humidity, pressure, change in pressure, sleep, step-count, and screen-time. All of this data is recorded with help from sensors in the phone. GPS is used to retrieve the users location which is then used to get weather information. Apple Health uses various sensors, including the accelerometer and gyroscope, to retrieve health data about the user. This information is vital for the user to retrieve an accurate risk assessment for a migraine.

5 TESTING AND EVALUATION

The testing of our system began first with testing the voice flow as that was the first part of the project to be completed. This was done by initially started by having different members of the team talk with the voice flow application. It was quickly found that many more cases had to be added to the voice flow as many people have different expressions and ways of saying the same thing. For instance, when some people talk they began to rate their migraine at the beginning, or instead of saying yes would say yeah. Although these were simple for other people to understand, the voice flow struggled. While testing was continually done on the voice flow until the end of the project, when it began to work

with most cases, it was put to the side so that other products could be worked on and improved within the project. It was also put to the side because of the difficulty many voice systems have in understanding conversations. Voice systems are notoriously difficult because it requires planning input on what the user may say, and even then sometimes the interpreter misunderstands the user's input and thus stores incorrect results.

The Firebase system was also tested throughout its development. Once the Firebase initially began to ran it was tested to see whether the data was actually stored within the system and stored in the correct locations. The database is quite difficult to tell if it is working correctly by simply staring at the code, and thus needed to constantly be run when slight changes were made to see if it still worked as expected. In some ways, the testing of the database helped to build the software of the database as when the testing failed, the code would be modified accordingly. On that same note, when each feature began to record, the system had to be tested again to see if it would load correctly. Thus, through the extensive tests, the database could be evaluated as to whether it was effective or not.

Along with the Firebase, the mobile application was extensively tested. Within the Unity environment that the app was built in there is a play mode that emulates what the app will look like and do when it is running on a phone. Throughout the entirety of the project, this was the most constant form of testing on the app. If a new button or script was added, there would be an attempt to run the app to see if it would compile correctly. Many times, if something new was added the app would not build at all on the screen, thus informing that changes had to be made because it failed the test to build. Along with putting it in play mode to test it, the application was also put on the iPhone by exporting the unity project to xcode and connecting the phone to the computer for it to download. For this, two different iPhones were tested and it was found that the newer iPhone 13 had different display settings and resolutions and thus the app appeared a different shade, and the buttons a different size. However, on the iPhone 11 it displayed the same as it had in the Unity display. The application that was built on the iPhone 11 was later shown to different people that had migraines to gain insight and feedback. Many people appreciated the simplicity of the application, but felt that there could have been more personalization features within the app. Overall, the many tests of the app and system improved the overall functionality and user experience of our final product.

When looking at the initial motivation and comparing that to the system created, it's fair to say that the system provides the fundamental functionality for the initial idea. The initial idea focused on putting those with migraines first, which can be seen throughout the system created. When the user does have to interact with a screen, it's in a color that should help with migraines [15] [22] and a font that should cause the least amount of stress [19]. The user doesn't have to input tons of data into their phones and can use the systems around them to do it instead. Most of all though, it takes the data from people's everyday life and can tell them whether they will have a migraine. This is important because even if the app were to put user's first, it wouldn't matter if the app wasn't functional in the way that it could accurately predict whether they would have a migraine. Thus, it is fair to say that the system created meets the general idea it was based on as it helps people track and predict their migraines without them having to do too much work.

6 COLLABORATION

6.1 Structure of the team

The structure of the team was dynamic, which only made sense due to the team being made up of three people. The roles within the group would often shift from week to week, altering the person taking the lead for the week. The constant shift was also beneficial as there was not enough people to assign labor too, meaning that if only one person were lead in the project, they would have to do all the work of the lead as well as all the work of someone else as well.

Thus, everyone traded off the position of leader at different times which meant no one person was constantly bearing the brunt.

The team made an effort to actively meet up in person and on Zoom. This worked well for the specific team because of the team dynamic they developed in the initial weeks working with each other. Seeing each other face to face during a meeting meant that they could more easily and effectively communicate their ideas and thoughts with one another. This worked because the team was comfortable with each other, but they also used a designated team chat where they could ask questions to each other as well. The ability to message and respond to each other outside of the meetings, allowed for increased communication that accelerated the project.

Every week the team met on Zoom or in person to talk about what had been done and the person to do the write up for that week would alternate from person to person. The team knew what to do each week by keeping track of progress with Git Projects. Git projects allows for an easy way to drag things from to-do, in-progress, and completed. This tool was easy to view because it was on Git and allows for a good visualization of the progress of the app and what was left to be done. Thus, this made Git a helpful tool used within the project.

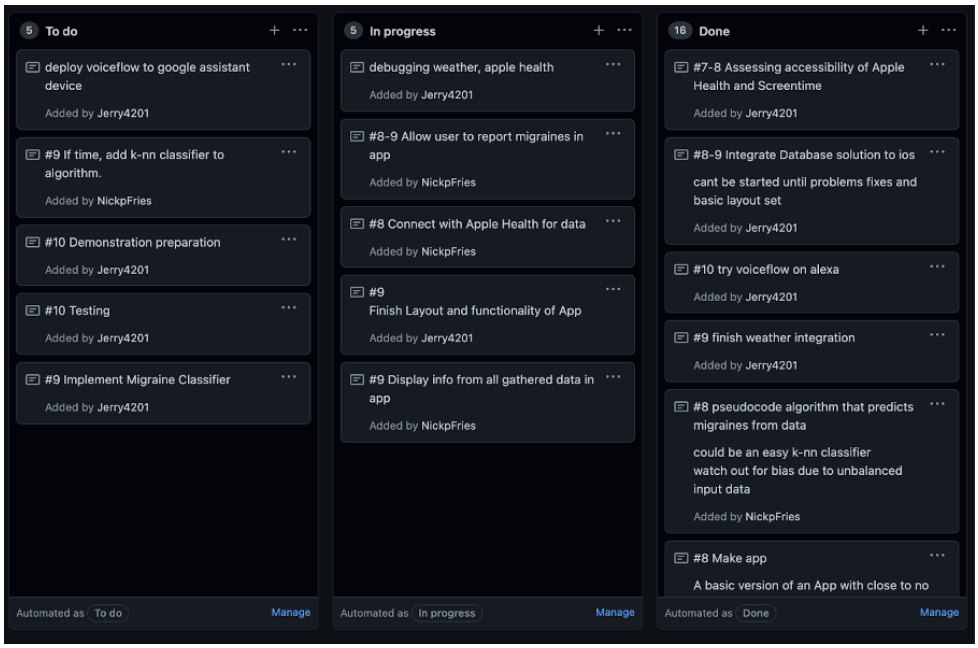


Fig. 14. Git Project view for the system.

6.2 Overall Collaboration and Division of Labor

Within the team, Nick Fries took on the task of using the weather API, voice interaction, as well as creating both the general migraine predictor as well as the personal migraine predictor. It made sense for the same person to create both predictors as they would have gotten familiar with the variables and classes needed in one, that they would most effectively be able to implement it in the other.

Looking at the next member, Delia McGrath took on the design, interface, and integration within the system. After initially creating many of the diagrams for the project proposal and slight familiarity with Figma, it only made sense that she should continue to work on the user display as it seemed that she had a level of comfort-ability with it that could be useful given the short time frame.

Lastly, the final member, Julian Strietzel took on the task of working with Firebase, the voice interaction, as well as the interaction. Before going into the project, Julian had a background working with Firebase, which was taken advantage of by having him work on it again, but with this system.

As previously stated, the reports submitted were on rotation, so that no one person would always be doing them. This also enabled people to get on board with parts of the project they did not know as much about. Before each report, there was a weekly meeting held to discuss each member's contribution to that week, so everyone in the group would be cognizant of what was being submitted on their behalf.

For the final presentation and the final report, the work was divided into sections that they would then have to write on and present on. This made sense for the team as those that had spent time writing up and going into detail about specific parts would be the same ones to explain that part to everyone else. This also worked in favor because if someone then had a question on that part, the same person would easily be able to answer.

6.3 Problems/issues and how they have been solved

Many of the issues that arose in the project had to do with the team's lack of knowledge in the software they were dealing with. For instance, none of them had previously dealt with C the programming language, which meant there was a lot of work to understand how to implement the processes within the particular language. In addition to this, it was difficult to work in the certain environments as they had also not previously been worked in. In particular, understanding the functionality of Unity was a problem at multiple points within the project. Whether it was initially adding scripts to the different GameObjects within Unity to add functionality to buttons, or whether it was integrating the Firebase to Unity so that the app would be functional in the way that it could display to the user their results. However, most of these issues were resolved with the help of Tommy and the internet. Many people have used Unity and C in the past so the documentation found on the web provided feasible solutions to the problems, and also solved them.

The primary issue faced within the system was gathering the data from Apple Health. While a kit was purchased that should have allowed the user to opt in to sharing their data with the app, there was an issue with licensing and Unity that did not allow for this to work. Thus, the app within Unity uses sample Apple Health data that is provided within the toolkit. Although it was a disappointment that the platform used for the app didn't enable for the app to work as intended, if the project continues, the solution to this will be to move away from Unity and try to work directly in XCode under different licensing. While it is uncertain what exactly will be the magic fix to the issue, the continual persistence of the team will solve the issue if it is decided to continue working on this system.

7 CONCLUSION AND FUTURE WORK

Our work on ModernMigraine has created a new and unique tool to use in the fight against migraines. By automatically tracking weather data, retrieving Apple Health data, incorporating voice assistant, and analyzing patterns in data our application is bringing more modern tools to the hands of the user to use against migraines. With the help of ModernMigraines, many users can better predict and address migraines. This application is not a treatment itself, but can help the user determine what treatments are best for them. However, this application does have it's limits. With

migraines not being very well understood and an immense amount of migraine triggers, some users may not find ModernMigraines useful because it may not track the factor that impacts that specific user the most. Our application is not capable of tracking every trigger because there are so many and some simply are not easily trackable with current technology. For example, one cause of chronic migraines is brain trauma. This may be revealed through the use of a CT scan, but ModernMigraines has no way of identifying this itself. We focused ModernMigraines on tracking some of the most common triggers of migraines in order for it to help the maximum amount of people it can. According to the American Migraine Foundation the top 5 most common triggers of migraines includes stress, sleep, and weather [11] which are all trackable with ModernMigraines. ModernMigraines is by no means an end solution to the fight against migraines, but provides valuable knowledge to the user that can be used to help determine the best course of action for treatment.

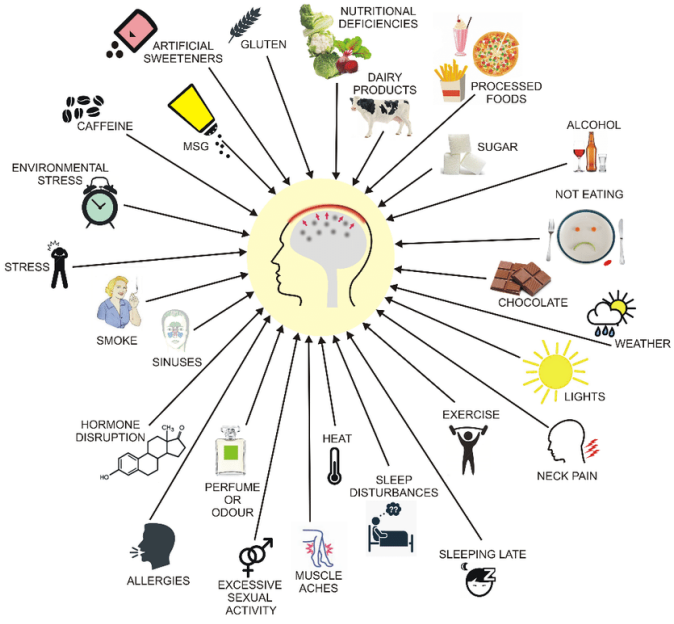


Fig. 15. Shown are just a handle-full of the many migraine triggers. [7]

Due to a limited amount of time, ModernMigraines has not yet reached its full potential but has laid out a foundation that can easily be built upon. For example, it currently tracks days that the user had symptoms without a full-on migraine and also tracks the severity of migraines but it doesn't use this data yet to more accurately predict migraines. Despite there not being use for this information we still ask the user to report this information so if we update the application to take this data into account we will have enough data to immediately put the new feature to use rather than waiting to gather enough new data. There are also other data points we can add into the application like diet. We did not prioritize diet as a factor to track because it would require much more input from the user, but it could be a nice optional feature to implement in order to increase the amount of people that could be helped. Perhaps the biggest improvement that could be made would be with the algorithm for determining the user's risk for a migraine. Ideally, ModernMigraines would use machine learning to predict migraines. Our team is not highly experienced with machine learning and due to our restrictive time frame we did not have enough time to learn and implement it in

our application. If we had extra time we would have attempted to use the K-nearest-neighbors algorithm to increase ModernMigraines accuracy in assessing migraine risk. Finally, another addition we would like to implement is to allow a voice assistant to communicate data and report a daily risk assessment for the user. This would allow ModernMigraines to further submerge itself into daily life and also decrease the user's interaction with a screen. While these potential future developments would increase the functionality and accuracy of ModernMigraines, the application still provides valuable knowledge in it's current state.

ACKNOWLEDGMENTS

To Tommy, for laughing about Irish jokes and helping with all the stuff not working as expected.

REFERENCES

[1] [n. d.]. Add Firebase to your Unity project | Firebase Documentation. https://firebase.google.com/docs/unity/setup
[2] [n. d.]. Amazon.com Help: What Is Alexa Voice ID? https://www.amazon.com/gp/help/customer/display.html?nodeId=GYCXKY2AB2QWZT2X
[3] [n. d.]. Apple Platform Security. ([n. d.]), 219.
[4] [n. d.]. BEHealthKit: Table of Contents. http://beliefengine.com/BEHealthKit/documentation/
[5] [n. d.]. Create Intents, Utterances, and Slots | Alexa Skills Kit. https://developer.amazon.com/en-US/docs/alexa/custom-skills/create-intents-utterances-and-slots.html
[6] [n. d.]. Current weather data - OpenWeatherMap. https://openweathermap.org/current
[7] [n. d.]. Figure 4. Main migraine triggers. Some are well established and... https://www.researchgate.net/figure/Main-migraine-triggers-Some-are-well-established-and-confirmed-by-reports-on-large_fig2_337263193
[8] [n. d.]. Imitrex Oral: Uses, Side Effects, Interactions, Pictures, Warnings & Dosing - WebMD. https://www.webmd.com/drugs/2/drug-11571/imitrex-oral/details
[9] [n. d.]. The iPhone Health App from a forensic perspective: can steps and distances registered during walking and running be used as digital evidence? | Elsevier Enhanced Reader. https://doi.org/10.1016/j.diin.2019.01.021
[10] [n. d.]. Privacy and Security in Firebase. https://firebase.google.com/support/privacy
[11] [n. d.]. Top 10 Migraine Triggers and How to Deal with Them | AMF. https://americanmigrainefoundation.org/resource-library/top-10-migraine-triggers/
[12] [n. d.]. Weather and Migraine. https://americanmigrainefoundation.org/resource-library/weather-and-migraine/
[13] [n. d.]. What is REST. https://restfulapi.net/
[14] 2019. Migraine with aura: an overview. https://migrainecanada.org/posts/the-migraine-tree/roots/migraine-categories/migraine-with-aura-an-overview/
[15] Carolyn A. Bernstein, Rony-Reuven Nir, Rodrigo Noseda, Anne B. Fulton, Shaelah Huntington, Alice J. Lee, Suzanne M. Bertisch, Alexandra Hovaguimian, Catherine Buettner, David Borsook, and Rami Burstein. 2019. The Migraine Eye: Distinct Rod-Driven Retinal Pathways' Response to Dim Light Challenges the Visual Cortex Hyperexcitability Theory. *Pain* 160, 3 (March 2019), 569–578. https://doi.org/10.1097/j.pain.0000000000001434
[16] Arnoud Cornelissen. 2020. App helps counter migraine with artificial intelligence. https://innovationorigins.com/en/app-helps-counter-migraine-with-artificial-intelligence/
[17] Dan Svantesson, RogerClarke. [n. d.]. Privacy and consumer risks in cloud computing | Elsevier Enhanced Reader. https://doi.org/10.1016/j.clsr.2010.05.005
[18] Stefanie Alki Delichatsios and Temitope Sonuyi. [n. d.]. Get to Know Google… Because They Know You. *MIT, Ethics and Law on the Electronic Frontier* ([n. d.]), 33.
[19] Nafiseh Hojjati and Balakrishnan Muniandy. 2014. The Effects of Font Type and Spacing of Text for Online Readability and Performance. *Contemporary Educational Technology* 5, 2 (June 2014). https://doi.org/10.30935/cedtech/6122
[20] Sun Microsystem. 2000. Chapter 7: Distributed Application Architecture. *Distruibuted Application Architecture* (2000), 20.
[21] Mia T. Minen, Eric J. Stieglitz, Rose Sciortino, and John Torous. 2018. Privacy Issues in Smartphone Applications: An Analysis of Headache/Migraine Applications. *Headache: The Journal of Head and Face Pain* 58, 7 (2018), 1014–1027. https://doi.org/10.1111/head.13341 _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/head.13341.
[22] Rodrigo Noseda, Rony Reuven-Nir, Carolyn Bernstein, David Borsook, Catherine Buettner, and Rami Burstein. 2017. Green light alleviates migraine photophobia (S47.005). *Neurology* 88, 16 Supplement (April 2017). https://n.neurology.org/content/88/16_Supplement/S47.005 Publisher: Wolters Kluwer Health, Inc. on behalf of the American Academy of Neurology Section: April 27, 2017.
[23] Ke Sun, Chen Chen, and Xinyu Zhang. 2020. "Alexa, stop spying on me!": speech privacy protection against voice assistants. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems (SenSys '20)*. Association for Computing Machinery, New York, NY, USA, 298–311. https://doi.org/10.1145/3384419.3430727