**Stroke Rehabilitation App**

Assignment 5 - Reflection

Stroke of Genius

Ted Krimmer - 532970

Contents

[1 Introduction 2](#_Toc105958413)

[2 Usability Testing and Usability Goals 3](#_Toc105958414)

[3 Development Experience Comparison 4](#_Toc105958415)

[3.1 IDE experience 4](#_Toc105958416)

[3.2 Language features and syntax 4](#_Toc105958417)

[3.3 Framework features 4](#_Toc105958418)

[3.4 Firebase implementation 4](#_Toc105958419)

[4 Application Comparison 5](#_Toc105958420)

[5 Future Directions 6](#_Toc105958421)

[5.1 Future features 6](#_Toc105958422)

[5.2 Google App Store 6](#_Toc105958423)

[5.3 IOS App Store 6](#_Toc105958424)

[6 Conclusion 7](#_Toc105958425)

[References 8](#_Toc105958426)

# Introduction

Stroke rehabilitation can be life-changing; however, it can also be complex and slow. There is no one size fits all with stroke rehabilitation, so I have decided to build this app focusing on hand mobility and improving brain cognition and hand-eye coordination. Stroke rehabilitation often involves doing thousands of repetitions of simple tasks to rebuild the mind-muscle connection(neuroplasticity**)**. This goal will provide the user with an environment to complete a helpful task while making the app easy to use and enjoyable and allowing users to track their progress. The American Stroke Association reports that “10 percent of stroke survivors make an almost complete recovery, while another 25 percent recover with only minor issues.” (Holland, 2021). In the United States almost 800,000 people a year suffer from a stroke according to the Centre for Disease Control and Prevention (CDC) (Centres for Disease Control and Prevention, 2021). I believe with such a large number of people; the app must focus on being easy to navigate and efficient whilst making it worthwhile for their rehabilitation.

During the development of this app, I have created it three times in different environments. The app was built in the three settings: Android studio, XCode (IOS) and Flutter. Each time was working with other constraints and guidelines. Each of these development cycles changed the design immensely. The final product of the application looks nothing like what I had initially designed. This was for two reasons: I became more confident in programming the app and the feedback provided.

# 2 Usability Testing and Usability Goals

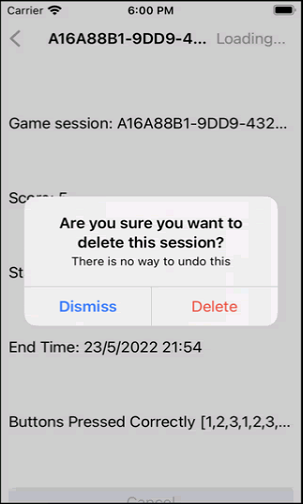
Graphical user interface, application, Teams

Description automatically generatedIn the first stage of development, I build a low-fidelity prototype. Then I got feedback through a series of user participation. Based on the testing results, I made some changes to the app. Most were functional fixes with adding features and making movement around the app more accessible. I also added a few decorative elements which aren’t related to the testing I feel added to the app. The feedback I got from marking this first assignment spoke about the lack of customisation settings and the app relying too much on the default UI elements. While this feedback was taken on board, I often struggled to complete the app's basic functionality in time, which led to a lack of decorative elements in the following builds. These changes continued to develop as I worked through building the app on each platform.

Figure

When designing this app, I considered 6 Usability goals: Learnability on first use, Memorable for repeat users, efficient (user efficiency), failure-resistant, forgiving, and satisfying. I believe that the most important of these goals will be Learnability on first use and enjoyable. This project’s scope states that we can assume the general app function will be controlled by a clinician or carer and that the patient will only interact with the application during the actual exercise. However, I have tried to build the app with the idea of patients using it independently.

In the first build of the app on Android, I focused on making it easy to navigate and learnability on first use. I did this by trimming the number of options while making the button more prominent than usual. As seen in figure 1, The homepage of the app is fundamental. The buttons are big enough to account for any reduced mobility, range of motion, or ability to control specific muscle movements of the users. In comparison, you can see an apparent lack of customisation, which I think would hurt how satisfying the user’s experience might be.

The second build was made in IOS. I kept the same usability goals and principles in mind. I kept the home page simple with large buttons. My focus in this was to build on the ideas on the first of the first development but to make the app more failure-resistant and forgiving. I added alerts when the user tries to delete a session in their history. This will allow the user to cancel the action before the data is deleted. This is an essential step in any application design. The feedback for this build from the marker was that many requirements were not met.

The final build was done is a flutter with Android as the target device. When building this, I tried to focus on getting as much of the functionality implemented as possible, as this was the only feedback I had gotten. I believe this version was backwards in terms of style; however, it was still learnability on first use, Memorable for repeat users, efficient (user efficiency), failure-resistant and forgiving.

Figure

# 3 Development Experience Comparison

## 3.1 IDE experience

The IDE experience I had as a first-time user was challenging in all three I used to build the app. I spent more time on google trying to work out what everything did than I finished writing the code. XCode was the best, in my opinion. This was primarily based on how easy it was to run the emulator. Simply press build, and away it went. With Android, the emulator had multiple options to set up and would often crash and freeze for no reason. Flutter had even more options. However, I never had any issues. The best part of them was that Flutter had an auto-format option for the code. This made my code not get too messy and disorganised.

## 3.2 Language features and syntax

The language and syntax used were vastly different for each IDE. Each had its way of making some tasks more manageable and some not so easy. For example, in XCode, to make a button do something when used, it had an “IBOaction” linked with it. The use of click and drag made it easy to add and use to the button. Android and nothing like this. You used a “setOnClickListener”. I believe Flutter is the easiest way to add functionality to a button by simply using “onPress”, which made a lot of sense. However, I still have difficulty updating the page on Flutter after an action while having no issues understanding how this works in the other two.

## 3.3 Framework features

Each of the developments of the apps had its challenges and characteristics. My favourite IDE experience was XCode using swift. The storyboard was a great way to visualise how the app was working. The storyboard shows how the screens would interact and move to the next. Android and flutter had nothing like this. The best way to see it is by running both and manually clicking on each screen. However, XCode was not as intuitive as flutter. Flutter was, for the most part, easy to understand the code and was very readable. XCode and Android used separate tabs to customise items such as lists, images, and buttons, which was initially challenging to learn. Flutter, on the other hand, if you wanted to add colour to text, you could write the colour and then give it a value. I found this a lot more intuitive. The transferring of data between screens was easiest on Flutter. Android has a similar approach using context and then selecting the data. When moving the screen, you could pass the data you wanted when building the move. XCode had a complicated way of giving data by using segues. I don’t fully understand how it works, but they were easy to set up with click and drag. I made invisible buttons with the data from the previous screen and wanted to pass in the segues.

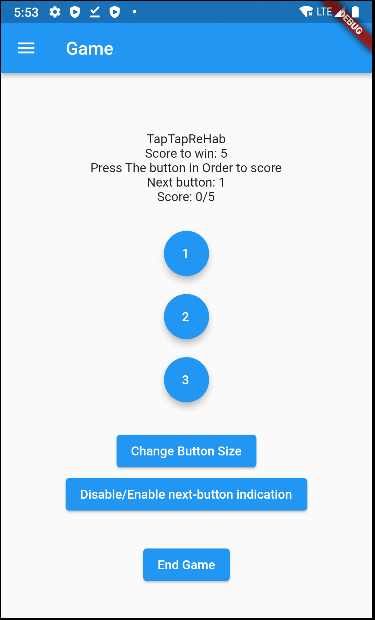
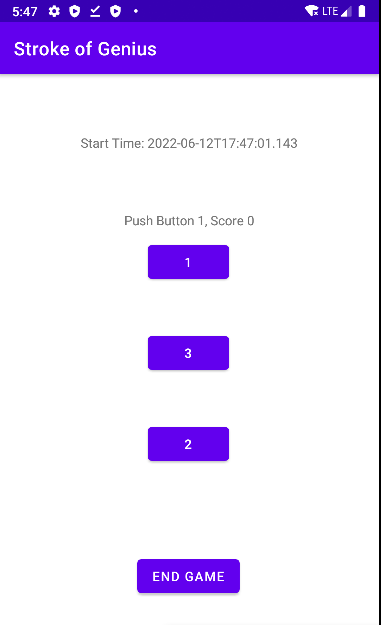
## 3.4 Firebase implementation

The implementation of Firebase was similar for all languages. The installation was different, but the syntax and language were similar. To access an object in the database, for example, the id of a movie, you can use movie.id. XCode was slightly more complex by using document.data(as: Movie.self). Flutter had the most complex, which I still don’t fully understand. You had to map the data to a JSON but then simply call movie.id.

# 4 Application Comparison

As previously mentioned, my app lacked visual appearance due to the challenges and time restrictions of getting the functionality implemented. However, I believe you can still see the contrast and visible differences between each. Below are examples from each of the application builds of the prescribed games. Figure 5 is a screenshot from the IOS application. I felt this one be the most appealing. The pages feel less intrusive to look at without the use of big header bars on the top. Figure 4 is the cross-platform Flutter build. It has more variety in buttons which is easy to implement. This was only because I found adding different types of controls very easy. Figure 3 shows the Android build. It looks almost identical to the Flutter style because they are targeted for the same platform. Unfortunately, I didn’t spend much time making these look nice or like similar apps on their platforms. I am sure I could make these all look almost identical with a bit more time.

Navigation and style are best displayed on my Flutter application. I could easily add a Navbar or burger menu in the top left. Most users will be able to know instantly what it does without explanation. I also really enjoyed how easy it was to add a back button that fits the IOS style with XCode.



Figure

Figure

Figure

# 5 Future Directions

## 5.1 Future features

Below is a list of features that would make this application more likely to be beneficial for actual stroke patients and their doctors.

* Online chat with a doctor
* Call feature to direct the user to their doctor
* Log in and account
* Exercise program
* Statistics on progress and improvement

## 5.2 Google App Store

You must follow these steps for this project to be available on the Google app store.

1. Make a Google play account. Pay the once fee of $20USD
2. Fill in details of the game. Name, Language, App or game, cost
3. Create and upload an Android App Bundles (AABs)
4. Change Gradle build type
5. Code signing. App Bundles need to be signed before uploading
6. Upload a Binary. AAB needs to be associated with a “release” of an “App” on the Google Play Store
7. Select realise type, e.g., open testing, closed testing, internal testing, or production.
8. Fill out Store listing details.
9. Push to production. If going into Open Testing or Production, Google completes an App Review.

The App can go through a closed beta and an open beta. Closed beta can be done by sending the app to players and gathering feedback. Google Play allows you to invite up to 100k players via email or an *unlimited* number if invited via Google Groups. Open beta is similar. However, players do not need to be asked. Developers can still limit the number of users who can access the game.

## 5.3 IOS App Store

You must follow these steps for this project to be available on the Apple App Store.

1. Create account
2. Pay the yearly fee of $99USD per year for the account.
3. Sign project and code (XCode can do this).
4. Test project with the TestFlight App.
5. Fill in Test information.
6. Send to testers
7. When happy, fill out the store listing.
8. Then push to AppStore

“After you thoroughly test your app in XCode, distribute it to beta testers or release it to users to run on their devices.” ( Apple Inc, 2022) The App can go through a closed beta and an open beta. These can both be done with the TestFlight app. The TestFlight app allows invited users to install, beta test, provide feedback, and get updates on your app.

# 6 Conclusion

I believe making this app have great learnability is the most important goal. Keeping things simple will lean into the rest of the plans. I was disappointed with the result of my apps. They all look fundamental, and some of the functionality wasn’t implemented. I was happy that each version and more functionality than the last. I think the app's concept has a tremendous real word potential and, with adding a bit of functionality and style, could even be used to help patients. Using a cross-platform tool to develop the app is the most useful; however, it can be tricky for new developers to learn without visualising Android and XCode tools.

# References

Apple Inc, 2022. *Distributing Your App for Beta Testing and Releases.* [Online]   
Available at: https://developer.apple.com/documentation/xcode/distributing-your-app-for-beta-testing-and-releases  
[Accessed 12 06 2022].

Centres for Disease Control and Prevention, 2021. *CDC.* [Online]   
Available at: https://www.cdc.gov/stroke/facts.htm  
[Accessed 18 03 2022].

Holland, K., 2021. *Healthline.* [Online]   
Available at: https://www.healthline.com/health/stroke  
[Accessed 15 03 2022].