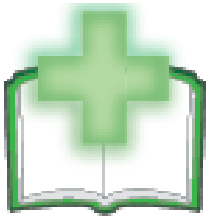

Technological Feasibility Analysis

Presented By: Team HealthLit

Devin O'Neal

Leslie Gurrola

Brendan Tjeerdsma

Ahmir Hughes

Project Sponsor: Dr. Olivia Lindly

Team Mentor: Han Peng

Project Title: Parent Health Literacy Mobile App

November 15th, 2021

Contents

1 | Introduction:

When faced with the task of taking medicine, it can be hard to do correctly, especially with how complicated medicine labels can be. This can lead to providing an incorrect dose of medicine, which can be very dangerous, especially for children. The knowledge of correct medicine dosing and many other concepts is encompassed by the field of health literacy. Health Literacy is the degree to which people can find and use information to make better health decisions for themselves and others. Health literacy is especially a problem for parents of young children, and these parents find it difficult to care for their children and administer medications properly and with the correct frequency and dosage.

Our client is Dr. Olivia Lindly, a professor and researcher at Northern Arizona University. Dr. Lindly studies child and maternal health, focusing primarily on the behaviors relating to child health. She has noticed problems in the quality and dissemination of information related to health literacy and child health practices. For example, community doctors will often provide parents with information on how to care for their children in the form of pamphlets and other physical media. This media usually contains fairly accurate and comprehensive information, however most parents will read over the information and then quickly forget it, or discard it altogether. Additionally, this information can be technical and hard to understand for parents with a lower level of health literacy. All of these problems lead to higher rates of child illness and mortality.

Looking at how to address this problem, we realized that there is one thing that most parents have access to: a smartphone. Our solution is to create a mobile app designed around increasing the health literacy of parents. This app will provide access to interactive modules that parents can complete to educate themselves about child health. We will include modules such as determining the correct dose of medication for a specific child given their weight, tracking

expected growth milestones for children ages one to five as well as access to digital versions of credible health information resources that a parent might expect to find at a doctor's office. Parents will be able to create accounts with an email and password to save and track their progress through these activities, so busier parents can progress at their own pace. Our goal -- as well as the goal of our client -- is to effectively disseminate accurate information and increase the health literacy of parents of young children so that they can provide better care. This process will not come without challenges, however.

2 | Technological Challenges:

Our Mobile App is designed to help parents learn and assess their knowledge in both general Health Concepts and Health Literacy. We intend to do this through interactive modules and games because the gamification of these aspects can help parents not only learn better, but become more engaged with what they are learning. With our given objective, we have found four major challenges that we will be facing.

1. **Mobile Framework** - We want a framework that can make our UIs intuitive, efficient communication with our backend and database systems.
2. **Hosting and Database Systems -** We want a hosting and a database system that is easy to implement, customize, and maintain.
3. **Gamification -** We want a system that can easily gamify the lessons while being easy to implement within our system.
4. **Modeling -** In order to assist with the development of the App, we want a modeling language that can help our group and our client understand the processes behind the App

5. **Administrative Portal -** In addition to the app, we will be making an Administrative Portal to test how the app works and how users interact with it.

3 | Technological Analysis

While getting to know the technology that we must get familiar with for our mobile app there were a lot of issues that we needed to take into account, such as what type of coding platform was the best, what type of database we wanted to use on our backend and how we wanted to implement the graphics of our app's games. First looking at the various coding platforms, we needed something that was easy to manage, while at the same time would let us create a professionally developed app. To first acknowledge the problem of compatibility, we needed a platform that is compatible with IOS and android. One key detail about these different platforms is that some are easier to work than others. Looking into the databases that seemed to fit for our app we needed one that holds the user data, text documentation, and contained good graphics for our future games. Some databases were better than others depending on how we wanted to use the back end, but most of them were better used for transactions, search engines, and other items that we did not need for the app. This led us to dive into our app back-end purpose. We needed something that can hold the client health lit modules, the games that we needed to implement as well as the user profiles and data. With this back-end in mind we also needed a way to use this to the maximum where it was compatible with our modules and games.

To measure the best technology to use for the application, we separated the technologies into categories explaining how efficient they are. All of the alternatives were rated out of 15, and the highest out of 15 was chosen for app development.

3.1 | Mobile Framework

3.1.1 | The issue

One of our main interests was which app development platform would be the best for our app. Eventually our app is going to need to be cross platform and we needed to investigate which app platform would work best in implementing this. Cross-platform compatibility is something of importance but so is each platform having enough stability to allow good usability..

To pick the mobile platform, we needed to compare many options. Each alternative was measured by ease of use and UI interface/categories. Since the measurement of the chosen one was out of 15, one category needed to be weighted more compared to the others. In this case ease of use was weighed out of 10 and UI interface/categories was out of 5.

3.1.2 | Alternatives

There are many app development platforms that we can use that have their own beneficial UI for user confidence. The user of the mobile app should be able to create their personal profile and select their preference of module. The app should be able to communicate with the back end at a responsible response time. These options are listed below:

Option 1: Reactive Native

React Native is an open source framework that was designed by Facebook and uses JavaScript libraries. This is a high-performance platform to implement efficient UI. It can be used to create a single codebase shared between platforms (iOS and Android).

- *Ease of use* - React Native has the ability to be cross-platform, but it is not as easy to use as others.
- *UI interface and libraries* - React Native requires us to build personal UI, which can be beneficial for customizability. Its libraries are more easily usable and have a wider range.

Option 2: Android Studio

Android Studio is an IDE that is used for android apps and offers more features. Android Studio uses both Java and Kotlin, it also provides faster turnaround time for coding and workflow. This platform can be used to create cross-platform apps, but its focus is native Android apps.

- *Ease of use* - This is a platform that wec are most familiar with and is coded in Java, but the platform it would not be as compatible with iOS as others.
- *UI interface and libraries* - Android Studio's libraries include a variety of sources for building an app, but only for Android platforms. Android Studio allows pre-made UIs that are easy to implement into the application**.**

Option 3: Flutter

Flutter is a mobile app SDK that helps app developers design and build modern apps for both iOS and Android. Flutter was developed by Google and uses Dart as its main programming language. This is our main choice for app development because it is cross-platform and allows us to connect with our games a well. Below is Table 1, showing how we came to this decision as a team and some of the things we considered.

- *Ease of use* - Since one of the core requirements is for the application to have cross platform capabilities, Flutter is a main interest. Flutter requires users to be somewhat familiar with Dart.
- *UI interface and libraries* - Libraries are easy to implement and learn about. There are specific libraries that can be used to help with game creation.

	React Native	**Android Studios**	**Flutter**
Ease of Use (Rating out of 10)	***Score: 7***	***Score: 7***	***Score: 9***
UI interface and Libraries (Rating out of 5)	***.Score: 2***	***Score: 4***	***Score: 3***
Total Score	9/15	11/15	12/15

Table 1: This table shows our thought process in choosing our mobile framework

This table allows clearly numerical representation of Ease of use and UI interfaces and libraries that were mentioned in each alternative.

3.1.3 | Chosen Approach

Collecting the total score mentioned in Table 1, the chosen approach is Flutter. Flutter has the capacity to be multi-platform and allows better integration with the app we plan to create. Flutter is also known to reduce code development time with its “hot reload” feature so we can see changes of our app. It also makes designing easier and it is easy to connect with the UI framework.

3.1.4 | Proving Feasibility

For feasibility the first thing to consider is multi-platform capabilities, due to the games we wish to implement and our client wanting an app for both iOS and Android. We also wanted something that was easy to work with and has the best UI framework for our users. With the different frameworks that it provides we wish to use its recommended hosting system (Google Firebase) to have ease of use with both. Making sure that we include different tests with the front-end of the app to see which one has the best performance, we will choose the best that we see fit.

3.2 | Hosting and Database

3.2.1 | The issue

Getting to more of the back-end desired characteristics, we need to focus on making sure that any data that we are trying to implement or collect from our user will be easy to manage. Each time we wish to include a module we need it to be cost efficient so it can be integrated into the app without affecting the current modules and the user profile/settings. Since our client wants to collect data from our users, we need to make sure that any type of data she needs is safely collected and is accessible to us and her if she wishes. One thing that might be a problem is overloading the back end to the point where performance is affected. The back end can be affected by incorrect game implementation and the data collection intrudes upon it.

To pick a hosting and database platform, we needed to compare between each alternative. Each alternative was measured by ease of use, customizability, and maintainability. We described each alternative's capabilities with each measured category and from there we scored them in table 2. Since the measurement of the chosen one was out of 15, this means that one category had to be scored out of 5, with the chosen approach being the highest out of 15.

3.2.2 | Alternatives

Option 1: Amazon Web Service (AWS) Amplify

AWS Amplify is the most compatible hosting platform for Flutter. AWS Amplify has a set of tools and services that enables mobile developers to gain access to open-source libraries and UI components. It allows the app to use elements like signing in/out and tracking user analytics. Since it is connected to Flutter it is compatible with both Android and iOS and uses Dart as well.

- *Ease of use* - AWS is comprehensive and is easy to use. It has many co-database hosting sites that we can choose from so it is more compatible with our back-end needs. This also means that it is limited to the services that Amazon provides
- *Customizability* - Not very customizable due to Amazon guidelines but offers other hosting sites.
- *Maintainability* - It is maintained by Amazon, so we are limited to that but it also has its benefits.

Option 2: Back4App

Back4App is a low code back-end that helps developers build mobile apps at a rapid pace. It provides a parse server and helps optimize applications. It is known for being compatible with GraphQL and RestAPIs. It has a low learning curve and flexible support. Back4App is created by developers for developers.

- *Ease of use* - Back4App offers a wide range of features and if you are familiar with developing apps it comes with ease. Therefore for some of us it is easier to implement compared to others**.**
- *Customizability* - Back4App allows developers to customize their app depending on the features they want to include.
- *Maintainability* - It is maintained by the version of its parsing server making the app more robust.

Option 3: Google Firebase

Google Firebase can be used with our mobile application, it is secure and can deploy commands. It is the ideal for monitoring our database with the real-time backend and API it holds. It has straight forward hosting but limits our user to have a Google account. However, we can use this to our advantage since users are likely to have an existing Google account already. This will make it easy for users to sign into our app with an existing Google account or create a new one to use. This will also allow us to gather metrics about our app's performance and how users are navigating and using it, allowing us to improve our systems. Because it is owned by Google, we do not have to worry about scalability and performance. It will be easy to integrate with our chosen platform Flutter since both are from Google. Below is Table 2, showing how we came to our decision as a team and some of the things we considered.

- *Ease of use* - Mainly made for simplicity, allows app developers to integrate UI features seamlessly.Can be easy to use with its need of front-end logic and offers crash reportings to fix bugs.
- *Customizability* - Is able to implement custom authentication and database connections.
- *Maintainability* - Similar to AWS it is maintained by its creator Google.

	AWS Amplify	Back4App	Google Firebase
Ease of Use (Rating out of 5)	***Score: 4***	***Score: 2***	***Score: 5***
Customizability (Rating out of 5)	***Score: 2***	***.Score: 3***	***Score: 4***
Maintainability (Rating out of 5)	***Score: 4***	***Score: 2***	***Score: 4***
Total Score	10/15	7/15	13/15

Table 2: This table shows our thought process in choosing our hosting and database service

This table shows a clearly numerical representation of ease of use, customizability, and maintainability as mentioned in each alternative for choosing hosting and database platforms.

3.2.3 | Chosen Approach

Looking at Table 2, the chosen approach is Google Firebase due to both Flutter and Google Firebase being the most compatible with each other. The security and monitoring it provides for our user is an interest as well based on the client's interests. It is also capable of hosting the game we are going to design for the client's modules.

3.2.4 | Proving Feasibility

To make sure there is feasibility for Google Firebase hosting we need to be able to sample some of the modules and profiles through it. Making sure that the user can create their profile and selecting the module they wish to enter. We can do this by creating fake users and testing modules. In the first phases of our app there is the chance that the user will be selecting a text module rather than a game module. Therefore in our future test we need to be able to select the game module.

3.3 | Gamification

3.3.1 | The issue

For part of our app, the client wants the assessments to be games with computer generated imagery and controlled input by the user. This means that we will need a program that can render computer graphics and have controlled user input so the games can be played without issue. We will also need to ensure that our mobile framework is compatible with whichever software we choose.

To pick the gamification platform, we needed to compare each of our alternatives. Each alternative was measured by ease of use and implementation. Since the measurement of the chosen one was out of 15, this means that one category needs to be weighted more compared to the others, in this situation ease of use was weighed out of 10 and implementation out of 5. The highest out of 15 became our chosen approach for the application.

3.3.2 | Alternatives

Option 1: HTML5

HTML5 is a markup language that handles the structure of web pages. Compared to its predecessors it can handle computer graphics and audio easier which is useful for the game development element of our app. HTML5 utilizes Cascading Style Sheets and JavaScript for styling and “behind the scenes" algorithms respectively. HTML5 is also compatible with our chosen mobile framework so it can be wrapped around our app.

- *Ease of use* - HTML5 can be displayed in Flutter with some modifications.
- *Implementation* - Need to implement CSS and JavaScript to properly make the games which requires more work and time.

Option 2: GameMaker studio 2

GMS2 is a game engine that can render and allow for 2-D based games to function. It utilizes its own programming language while allowing for many different tools and packages to allow for many different types of games since it was designed specifically for game development. GMS2 projects can be exported in HTML5 format so it can be compatible with our framework.

- *Ease of use* - Same reason as HTML5 since projects can be exported to that format.
- *Implementation* - GMS2 is a beginner friendly engine and is provided with scripts and packages to further simplify implementation.

Option 3: Flame

Flame is a game engine that allows for games to be built quickly while utilizing the infrastructure of Flutter. This is especially useful since we already plan to use Flutter for our framework, so portability will be easy here. Flame is useful for making simple games, but more complex games may prove to be a challenge for this engine. Below is Figure 3.3.2, showing how we came to our decision as a team and some of the things we considered.

- *Ease of use* - Flame is directly built on top of Flutter making it the easiest engine for displaying with our mobile framework
- *Implementation* - Flame is not a beginner-friendly language, especially if users are not familiar with game design.

	HTML5	GMS2	Flame
Ease of Use (Rating out of 10)	***Score: 9***	***Score: 9***	***Score: 5***
Implementation (Rating out of 5)	***Score: 3***	***Score: 5***	***Score: 2***
Total Score	12/15	14/15	7/15

Table 3: This table shows our thought process in choosing a framework to develop our games in.

This table shows a clear numerical representation of ease of use and implementation as mentioned in each alternative for choosing a gamification platform.

3.3.3 | Chosen Approach

Table 3 shows that the chosen approach is to utilize GameMaker Studio 2 in order to develop the application's games. This is mostly due to GMS2 having a more powerful system to make games and some of our members in our group having experience with the GameMaker programming language.

3.3.4 | Proving Feasibility

We can test the feasibility of the games by first making a more simplified version of them which should be just the basic concepts in GMS2. Then we will export to HTML and perform modifications to see if it will communicate with our framework.

3.4 | Administrative Portal

3.4.1 | The Issue

In order to test and do studies with our app, we will need to develop a Web-Based Administrative Portal. This portal needs to be user-friendly and intuitive.

3.4.2 | Alternatives

Option 1: ASP.NET

ASP.NET is a server-side scripting language released by Microsoft as part of the .NET developer platform. It's open source, cross platform, and it specifically focuses on developing Web-Applications. ASP.NET specializes in large-scale products and security-related protocols. There are a lot of features tied to ASP.NET, however, because it's tied to .NET, it's outside compatibility and customizability are limited.

- *Ease of use* - ASP.NET includes the flexibility to add and remove features as needed.
- *Compatibility* - Difficulty for ease of making changes, documentation gaps, lack of supporting tools and lack of basic features since fairly new.

Option 2: PHP

PHP is a server-side scripting language released in 1994 as a free and open-source Web Development tool. PHP is a simple tool designed for customizable user interfaces and small-scale functionalities. While PHP has limited functionality, it has a lot of compatibility with outside software.

- *Ease of use* - Great for beginners since it allows simple and easy coding techniques. It is also well rounded enough for professional coders to use as well.
- *Compatibility* - Is hase a rapid development that is very compatible with mobile applications. Many PHP developers make sure to use the framework as it is meant to be used to ensure application's security

	ASP.NET	**PHP**
Ease of Use (Rating out of 10)	***Score: 6***	***Score: 9***
Compatibility (Rating out of 5)	***Score: 3***	***Score: 5***
Total Score	9/15	14/15

Table 4: This table shows our thought process in choosing the administrative portal framework.

Table 4 shows a clear numerical representation of ease of use and compatibility for choosing the best administrative portal framework for the application.

3.4.3 | Chosen Approach

Table 4 clearly indicates that PHP is best fit for the application's Administrative Portal. The PHP framework specializes in making a UI for the portal, but its customizability, compatibility, and accessibility allows it to be a better fit for the smaller-scale testing.

3.4.4 | Proving Feasibility

We can test the functionality of the language by making a simple Test UI and seeing how it works with our chosen database, Google Firebase. This will allow us to evaluate how cohesive PHP and Firebase are.

3.5 | Modeling Languages

3.5.1 | The Issue

We needed a modeling language to communicate key ideas to our client about the structure of our mobile app and to provide a framework with which to build this structure.

3.5.2 | Alternatives and Chosen Approach

Looking at popular modeling languages, one immediately stood out to us: UML. UML is easily the best and most popular modeling language for modeling systems, and it is one that everyone on our team is familiar with. Additionally, there exists a large number of applications for using UML to model systems. For these reasons, we have chosen to use UML as our modeling language.

3.5.3 | Proving Feasibility

We can prove the feasibility of UML as our modeling language by creating simple class diagrams of our system. These class diagrams will act as an outline of how the system will function overall, taking into account all of the individual systems and how they work together. This will be instrumental in keeping visible a clear image of our system as a whole and will facilitate easy integration of related systems as needed.

4 | Tech Integration:

Using UML, we can easily show how all of the systems that we have previously analyzed will fit together. Figure 1 shows that Flutter will be the main framework for the application which the user will see and interact with. This will allow for interactivity for both the iOS and Android operating systems. The GMS2 games created will utilize the export feature to be converted into HTML5 format which will then be displayed and be interacted with from Flutter. Our backend will contain our database maintained by Google Firebase and will contain all of the user info needed in order for the app to function. For our administrative portal, PHP will

communicate with our database so we can access the appropriate data for the portal. Our modeling language (UML) will allow us to communicate information about the app with our client so that information can be clearly conveyed. Below is the flowchart diagram of how all of the software will be integrated.

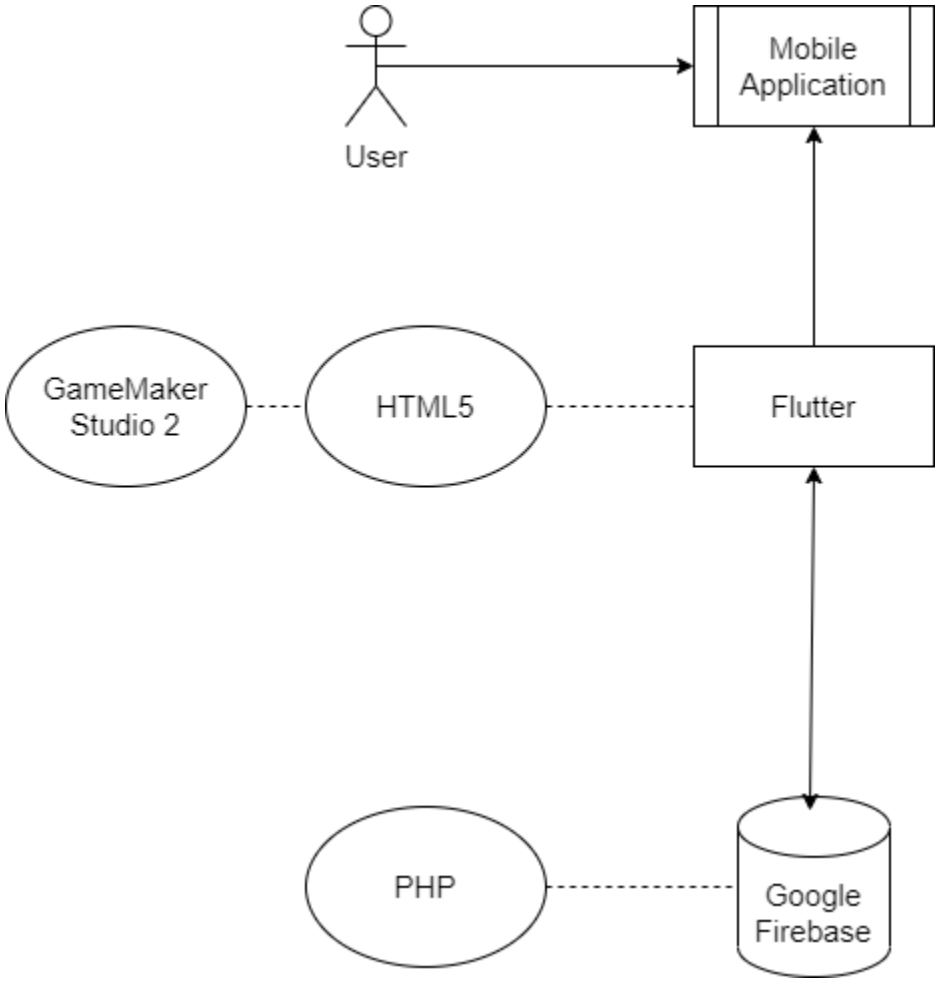

Figure 1: This shows a basic framework for our system, illustrating how some of our chosen systems will work together

5 | Conclusion:

In this document we have outlined our problem, how we plan to solve it, and the technological challenges that we will face in doing so. In summary, low health literacy is a

problem that affects many parents and leads to higher child illness and mortality rates. Our team, with the help of our client Dr. Olivia Lindly, seeks to remedy this issue through the development of a mobile app that aims to increase the health literacy of the parents of young children. This mobile app will teach parents about how to properly care for their children through the use of interactive modules that teach parents essential skills such as proper dosing of medications for children according to their weight. We are confident that we can develop an app that will meet these requirements, and we will work closely with Dr. Lindly to ensure that we are developing the right tools for the job. Our development will be streamlined by the use of technologies such as Flutter and Google Firebase, ensuring that our app runs smoothly, quickly and efficiently. We hope to gamify the process of increasing one's health literacy, leading to higher information retention rates and better care for children. In addition, we hope to make an Administrative Portal that can be used to test the app and collect data that we can use to improve how the app conveys information. Looking forward, our team will ensure that we have outlined the proper requirements that are expected of us when building this app through our requirements specification document.