Team HomeAide

Technological Feasibility

October 23, 2020

Project Sponsors: Kelly Roberts, PH. D and Jill Pleasant, MA

Team Faculty Mentor: Fabio Santos

Team members: Seth Borkovec, Ethan Donnelly, Courtney Richmond, Noah Baxter

TABLE OF CONTENTS

I. INTRODUCTION 2

II. TECHNOLOGICAL CHALLENGES 3

III. TECHNOLOGICAL ANALYSIS 4

A. Cross-Platform Mobile App Development 4

B. Database System 8

C. Web Framework 12

D. Cloud-Based Server 15

IV. TECHNOLOGICAL INTEGRATION 19

V. CONCLUSION 22

VI. REFERENCES 23

I. INTRODUCTION

We are team HomeAide from Northern Arizona University's Computer Science Capstone course 2020 to 2021. Our team includes leader Seth Borkovec with members Courtney Richmond, Noah Baxter, and Ethan Donnelly. We are working on project AT@Home with our sponsors, Dr. Kelly Roberts and Jill Pleasant from Northern Arizona University's Institute for Human Development. Our sponsors are involved with enabling people with disabilities to fully participate in all of life's experiences. They do this by influencing policy and impact people at all life stages, including individually and socially.

In the United States, there are over 65,000,000 Americans that are over the age of 60. Of these 65,000,000 at least 23,000,000 have at least one disability that they are living with. Finding ways to sustain a comfortable lifestyle when someone is aging or disabled can be quite the challenge and they might not know where to go to find the tools they need to make their life easier.

The problems include the following:

- There are so many assistive technologies that it is difficult for a potential user to find what they need on their own.
- There is no centralized database for assistive technologies which can make them difficult to find.
- Potential users often lack the experience or resources to find available technologies.

In order to solve these problems, we are going to be developing a cross-platform mobile application which provides the following services:

- Gives the user tailored recommendations for assistive technologies based on their specific needs.
- Uses a database to inventory available assistive technologies in a single location.
- Provides the user with the information they need on the recommended assistive technologies, as well as contacts to local resources.

This document explores the feasibility of the technologies that will be used in our implementation. We will begin by first discussing the technological challenges involved with this project in Section 2. In Section 3 we will analyze each of these challenges by comparing and measuring the alternatives, how the alternatives were explored, and

providing reasoning for choosing a particular solution. Section 4 will detail how our chosen solutions will integrate with each other to provide our overall project solution.

II. TECHNOLOGICAL CHALLENGES

In this section we will outline all the major technological challenges related to our project as determined from our early meetings with the clients and initial project description. Based on these inquiries, we have determined that the major needs in this project include a cross-platform mobile application development framework, a database system, cloud-based server, and a web framework.

A. **Cross-Platform Mobile App Development**. Our client has indicated a need for a mobile application for the end users to interact with. The rationale in using a mobile application instead of only a website includes the fact that a mobile application can have some offline functionality which a web application would not have.

B. **Database System**. This project requires managing an inventory of assistive technologies (AT) including the different characteristics of each, possible uses, areas of the home in which they affect, as well as cost and possibly other factors which have not yet been identified. There will also be a need to store some user information in order to recommend new products to them and to facilitate communication with support staff.

C. **Web Framework**. Our clients will need a public facing for the project, as well as a way to manage the database so a web framework is needed.

D. **Cloud-Based Server**. The database and web framework will need to be hosted in an easy-to-manage location on a cloud-based server. Being cloud-based allows the project to scale and unloads the responsibility of server management from the client.

These are the four major challenges that we have determined in our project. In the next section, we will explore these challenges in detail and analyze the available alternatives.

III. TECHNOLOGICAL ANALYSIS

Here we introduce our analysis of the four challenges. Each challenge will first be described in the context of our project. Then we discuss the desired characteristics for the proposed alternatives and introduce the alternatives. Next we describe how the analysis was performed and detail how each criteria is scored. After scoring each alternative, we provide a comparison of the alternatives, including a table to display the results, and choose a solution based on the results. Having chosen an alternative, we talk about how it will be proven to work with this project. In the next section, Technological Integration, we detail how all the chosen alternatives will work together in this project.

A. Cross-Platform Mobile App Development

With this project we need to find an app development framework that will allow us to create code that will be able to work for both android and iOS devices as per our clients request. Finding the right framework for our team is crucial to our project's success as we need a framework that has all our desired characteristics.

1. Desired Characteristics

When it comes to finding which framework for app development would work best for us it is first important to consider what characteristics we are looking for in a framework. For our project we are looking for something that would allow us to be able to have compatibility so that our app can be used on virtually any device. Another thing we want to consider is finding one that can focus on UI as that is going to be a crucial part of our app as we target it towards an older audience. We also wanted to focus on a framework that we would all be able to pick up on and learn without too many struggles. Finally we want to consider database integration as we need a framework that can be integrated with whatever database we chose to use.

2. Alternatives

a) Flutter[1]

Flutter is a fairly new app development framework that was developed by Google and utilizes the Dart programming language. Flutter has many benefits to it that we are looking for in an app development framework such as a hot

reloading feature which is very helpful in developing the software. Another thing Flutter has that is important to us is portability as it can run on virtually any device with a screen on it. Following this Flutter also has accessibility options which are very important to us so we can make the UI as user friendly as possible for our audience. Flutter also has easy to use database integration with cloud firestore which can make for easy database use. Flutter is also one of the highest performing frameworks when compared to Xamarin and React Native.

b) React Native[2]

React Native was another framework that we looked into which is a popular framework that many popular applications today use. This framework was created by Facebook and primarily uses Javascript as is language to code in. Some of the benefits of using React Native would include the following, it focuses on UI to create a highly responsive interface. It has built in debugging and a hot reloading feature. It also has an open source Facebook library, and contains native controls and native modules to improve performance. Some downsides are that it has poor documentation and can sometimes have instability and compatibility issues.

c) Ionic[3]

Ionic is based on the Angular framework and uses HTML5 , CSS, and Javascript as its primary technologies for app development. Ionic has a wide range of integration capabilities and plugins which can be very beneficial to us. Ionic also has an extensive choice of UI elements that can be used to create quick prototypes. Another thing to note is that Ionic has precise documentations and testing methods that can be very beneficial to our development. Some cons with Ionic is that it is a plugin dependent system, it does not have hot reloading, and that there are some possible security issues with the framework.

d) Xamarin[4]

Finally we have Xamarin which is a framework that is based on the .NET framework and uses C# for coding. This is another very popular app development framework that utilizes a single tech stack with performance close to Native. Some benefits include the fact that it has full hardware support which eliminates compatibility issues and it also has its own IDE

within the Visual Studio App center. Another thing to note is that it has Xamarin.Forms which can be used to create simple apps and prototypes. However, some downsides is that it can be expensive to use for enterprises as well as the fact that it is not recommended to be used for apps that demand heavy graphics. Another thing to note is that creating the UI with Xamarin can be time consuming and only offers a limited amount of libraries that are needed for mobile app development.

3. Analysis

The method in which the team went over the possible options for choosing an app development framework is that we took into consideration the desired characteristics that we were looking for in a framework. We then selected a few frameworks that fit into that description and following this a list of pros and cons was created for each framework. Next we carefully went through the list of options weighing the benefits that each framework can bring to the table for us while also considering how each one would integrate with the other technological sections such as the database. From there we eliminated one framework at a time that we felt would not best serve us and found one that felt overall would work best for our project.

4. Chosen Approach

Each alternative that was looked at had a little bit of something that we wanted but some of the alternatives lacked more in areas where we felt were more important than others. For example when looking into documentation which we thought would be very important to us, we found that out of our considered alternatives, React Native had the worst documentation which would make development difficult for us.

Another point that was heavily considered was how well a framework could focus on the UI since our app is meant for older people who might have a hard time navigating. After looking at the alternatives we found that most of the alternatives focus on proper UI implementation but found that Xamarin can be the most time consuming, and that Flutter has built in widgets that can create visually appealing and easy to create UI coding.

Something else that we found was that Flutter seemed to have the least amount of downsides when we looked at each framework overall. The biggest downsides to

Flutter was that it is still fairly new and had only a few database options to choose from, but it does support the database we chose to use.

	Database integration	UI experience	Accessibility	Compatibility	Learnability	Total
Flutter	3	5	5	5	4	22
React Native	4	4	4	3	4	19
Ionic	4	3	3	4	4	18
Xamarin	4	3	3	4	3	17

Table 1: Mobile application development

In Table 1 above we see how we have compared the four frameworks with the various characteristics that we are looking for. With database integration the scores were based on the amount of databases that they can be integrated with as well as how easy it will be to integrate with the database.

With the UI experience we gave a ranking based on how well a nicely created UI can be made as well as how easy it can be to create a UI that our client is looking for. For accessibility the ranking was given based on the ease of use a framework would supply for the user to be able to adapt their setting so they can better navigate the app for the best user experience.

For compatibility the ranking was given based on how well we can transfer the app to different devices and how well it would run on those devices.

Lastly, the grade for learnability was given based on how easy it would be for us to learn the development language that the framework uses. We also need to consider how quickly we can understand the framework to get the coding process underway. With all of the factors considered and the resulting rankings, we found Flutter to come out on top with the overall best score in the categories we found most important.

5. Proving Feasibility

With all this taken into consideration we have decided to move forward using Flutter as our app development framework. We believe that this will be our best choice as it has the most beneficial features for this project and is well-documented. It also has accessibility options which are crucial for this project including large font support, screen readers, and sufficient contrast. It also has high compatibility which is exactly what our client wants for this app so that it can reach as many individuals as possible.

Flutter also has convenient tools to create a UI for users that is not confusing. We really want our main focus to be on UI as we feel it will have the most importance with our targeted audience. Some demos that we will be developing with this framework is a basic user registration page as well as a homepage to show that this framework is capable of doing exactly what we need in a clean and user friendly way.

B. Database System

This project will need to store a large number of entries for assistive technologies as well as their features. A database is required in order to store all of this information. It will also be used for other functions such as storing user information and providing a means to communicate with the app. There are four important qualities we require from a potential database including mobile application integration, connecting to a cloud server, security, and reliability.

1. Desired Characteristics

a) Integration with mobile application

It is preferred that the database of choice has a swift and easy integration with mobile applications. Since there is a possibility of developing a web application alongside the mobile application, it is also preferred if the database of choice can integrate with both a mobile application and web services.

b) Connection to a Cloud based service

The database will need to have some type of storage backup in the event of a local storage failure. Utilization of the Cloud service will allow for this

possibility. This will also help users with transferring information between multiple devices as their information is stored on the Cloud service.

c) Security

It is critical that personal user information be kept private and is HIPAA compliant. Databases need to be secured not just from outside attacks, but also from internal misuse as well.

d) Reliability

To maintain a positive customer experience, the database should have good uptime to make sure users have uninterrupted service.

2. Alternatives

a) Oracle[5]

Oracle is a database that uses both MySQL and NoSQL databases. Many customer reviews state that it is a reliable database because devices do not fail to connect to it very often. It is also relatively secure in that breaking into the database from the outside is also unlikely. A beneficial quality about Oracle is that it has a built in Cloud system, referred to as the Oracle Cloud Infrastructure. This means that the database and Cloud services can be achieved in one package through Oracle. However, it is quite difficult to integrate Oracle (either the database or its Cloud) for mobile applications because it may require additional API's. There are also higher level functions provided by Oracle that cost more. It is unknown at this time what those functionalities are as it is difficult to determine what functionalities will be important until development begins. It is possible that the API that will integrate the database and Cloud with a mobile application may be one that requires payment.

b) PostgreSQL[6]

PostgreSQL is an open source database so there is no need for licensing or additional fees. This database supports SQL and JSON querying. It is an ideal database environment for a relational type database and is common in web and mobile applications. However, it does have frequent timeout errors in which devices disconnect from the database frequently. The user interface itself also has a moderate learning curve for beginners.

c) SQLite[7]

SQLite is the most commonly used database for mobile applications. Thus it will have easy integration with mobile application development environments. SQLite also supports an Android/Java framework for mobile applications. SQLite does have an external API that allows for encryption of the database itself. However, a lot of database security is related to managing who has administrative access to the database.

d) Firestore[8]

Firestore supports a wide variety of languages such as Node.js, Java, Python, Unity, and C++. Like Oracle, it also has built in Cloud services thus will be easy to integrate the Firestore database with its Cloud services. Although Firestore is free, after a certain amount of storage has been exceeded, the application will require payment for further use or for a greater storage capacity. This applies to both the database itself in terms of how much information is stored in the database, and the Cloud services in terms of how much information is stored in the Cloud.

3. Analysis

The method of analyzing potential databases was based upon how easy it is to integrate the database with a mobile application, how secure the database is, the cost of the database, the potential learning curve of the languages known by the database, and how reliable the database is.

4. Chosen Approach

For selecting the database for this project, SQLite is the best option. Although it does not have built in Cloud services like Oracle or Firestore, it does have easy integration with outside Cloud services and with mobile application environments. SQLite also has a lower learning curve as opposed to the other database options.

Price is also a major concern for this project because our client relies on grants for funding. PostgreSQL and SQLite are the only options on the above list that do not have a price tag attached to functionality. SQLite was chosen over PostgreSQL because it is more reliable (doesn't timeout frequently) and is easier to integrate in a mobile application environment.

	Mobile integration	Security	Price	Languages	Reliability	Total
Oracle	3	5	3	4	5	20
PostgreSQL	5	3	5	3	3	19
SQLite	5	4	5	5	4	23
Firestore	4	4	3	5	5	21

Table 2: Databases.

Table 2 above quantifies each of the potential databases that we considered. Each category is ranked from 1 to 5 where 1 is the lowest in that category and 5 is the best in that category. Each database scored pretty close to each other with none of the databases receiving a score less than 3 in any category. PostgreSQL did receive the lowest overall score with hits to its security, the learning curve of the languages supported by the database, and reliability. Oracle and Firestone are nearly tied with only having differences between how easy they are to integrate with mobile applications, security, and the language learning curve. SQLite scored better than the rest primarily due to its mobile integration and price.

5. Proving Feasibility

We believe that SQLite will be the best fit for the database development environment. The mobile application development technology, Flutter, supports SQLite giving it another advantage over the others. To test out SQLite, we will create a simple database with a few test entries. We will also test the encryption API, as well as develop internal security for administrative management.

C. Web Framework

An easy way to manage the database is needed, along with a public location for information about the application. Thus a website needs to be created. The public side of it will advertise the application, and perhaps host a web-app version as well. More importantly, the website serves as the client's interface to the database, allowing them to manage the inventory of assistive technologies (AT), and to interact with users of the application.

1. Desired Characteristics

The ideal solution for this project needs to have strong security features to protect access to the database and possibly user accounts. It also needs to have good compatibility with the database to reduce the chance for bugs and errors. It's also important to work well with the database to reduce the work of maintenance and updates. Taking into account the skill level of the team, the ideal solution will need to have well-developed documentation and use a language that isn't too difficult to learn.

2. Alternatives

a) Django[9]

Django is a popular free open-source web framework based on Python. A search query for web frameworks often includes results that place Django in their top five best web frameworks. It was initially released in 2005 and is now supported by an independent non-profit organization called the Django Software Foundation. Some well-known websites built using Django include Disqus, Pinterest, Instagram, and Quora.

Database Compatibility: PostgreSQL, MariaDB, MySQL, Oracle, SQLite

b) Flask[10]

Flask is another free web framework based on Python. It was initially released in 2010 and is sometimes among the top ten web frameworks in comparison lists. We found it attractive because it is based in Python which is easy to learn. The creator is Armin Ronacher and is maintained by David Lord and Markus Unterwaditzer. Issues are managed by Adrian Mönnich. A couple of well-known websites using Flask include Pinterest and LinkedIn.

Database Compatibility: PostgreSQL, MySQL, SQLite

c) Ruby on Rails[11]

Rails is a free web framework built on Ruby and often competes with Django for the top spot in lists comparing web frameworks. It was first released in 2004 and is used by well-known websites such as Twitch, GitHub, Hulu, and Zendesk.

Database Compatibility: PostgreSQL, MySQL, SQLite, SQL Server, Oracle (except DB2)

d) CUBA Platform[12]

CUBA Platform is a framework using Java and Kotlin. It offers both a free version as well as a paid version. It is compatible with the popular IDE, IntelliJ IDEA, but also comes with its own CUBA Studio IDE as an alternative. Development on CUBA can be done mostly in the user interface. We could not find examples of well-known sites developed on CUBA Platform, but it remains attractive because Java is a familiar language on the team.

Database Compatibility: PostgreSQL 8.4+, MySQL 5.6+, SQL Server (2005, 2008, 2012+), HSQLDB, Oracle 11g+, MariaDB 5.5+

3. Analysis

In order to provide a reasonable analysis of each alternative, we installed each one and created a "Hello World" page for each. The benefits of this approach include exposure to the documentation, and checking the ease of use. The documentation was used for both installation and creating the test page. We also considered the availability of examples and resources from third-parties to be included in the documentation criteria. The security of each was determined by the availability of authentication APIs and built-in administration. Database compatibility was scored on the number of types of databases natively supported by each platform without using third-party tools since we would prefer to avoid relying on extra tools for integration.

4. Chosen Approach

Each alternative offered adequate documentation, as well as being supplemented by third-party sources. Of these, Rails was the most complicated to install, but there are enough resources available to help get it working. Creating an initial "Hello

World" page was mostly straightforward on each except CUBA Platform. Django and Rails both include a default test page that is displayed when the installation is complete and the server is run the first time. Flask required writing a simple Python file to serve HTML. CUBA Platform needed some initialization before displaying a page.

With regard to security, Both Django and CUBA Platform include a built-in web administration portal when you run the server. There are also authentication APIs included in both. Rails also has built-in authentication, but does not include a convenient built-in web administration interface. However, there are plugins for Rails which take care of this. Flask is a very minimalistic framework that does not come with any authentication built-in. Authentication in Flask must be created by the developer which is not very convenient for our project.

Most of these offered built-in API's supporting common databases. Flask had the least support for databases while the others supported five different varieties.

	Security	**Database Compatibility**	**Documentation**	**Ease of Use**	**Total**
Django	5	5	5	5	20
Flask	1	3	4	4	12
Ruby on Rails	4	5	5	3	17
CUBA Platform	5	5	4	4	18

Table 3: Web frameworks.

Table 3 above displays the results of our analysis of each alternative. Each criteria is graded 1-5 where one is the lowest score and five is the best score possible. The alternatives all scored pretty close to each other in each category except in security. Since Flask does not have any built-in authentication, it got the lowest score. Rails falls behind a bit in ease of use because it uses a language that none of us have used before. Django edges out CUBA Platform in documentation and ease of use because it is much easier to find examples of Django implementations than of CUBA Platform. Tallying up the total scores leaves Django as our preferred solution for the web framework.

5. Proving Feasibility

In the analysis for web frameworks, we chose Django as the best fit solution. In order to confirm this choice, we will set up a test site connected to an instance of the chosen database to demonstrate that it integrates well with the database solution. To show the security features, we will implement the test site so that a user must be logged in order to modify records in the database. An anonymous user will only be able to read the records from the database.

D. Cloud-Based Server

For our project we will need a web hosted cloud server which will be used to host the database and web framework. The cloud server is important in providing secure and reliable access to our services.

1. Desired Characteristics

There are several characteristics that our group is looking for in the cloud hosting service. One of the most important characteristics to our client is reliability. While all services require some downtime in order to perform updates, it is important that this down time is as minimal as possible. Another important aspect of cloud hosting service is its cost. While most cloud services offer a free trial period, our client has indicated that they would prefer a lower cost option since their funding is mostly from grants. Going along with cost it is also important that our chosen service has adequate storage space for the price as the database that the cloud server will be hosting will likely store many images. Lastly it is important that the cloud service offers the option for quick and easy scalability in the event our client needs to upscale in the future.

2. Alternatives

a) Amazon Web Services[13]

Amazon Web Services (AWS) is the largest cloud service out of the three options. It has several data centers worldwide and offers a server up time of about 99.9%. This translates to being down just short of nine hours per year. AWS offers several different cloud hosting options at various prices. One of these is a service called LightSail.

Out of the options AWS provides LightSail seems like it fits our project best. LightSail is intended for sites that anticipate low to medium levels of traffic, and have frequent content changes. LightSail offers a 1-month free trial, and afterword the service costs between $3.50 and $160 per month biased on factors such as storage size. AWS also offers pay-as-you-go plans and partial refund options in the event that the total server downtime exceeds 99.9%[14].

a) Google Cloud[15]

Google Cloud is a relatively small cloud hosting service due to the fact it is relatively new. It offers a variety of different cloud hosting services at a range of prices. One of the services it offers for cloud hosting is called Cloud Ways. Cloud Ways states an uptime of 99.5% and does not have scheduled downtime. This server uptime translates to it being down about two days out of the year. Cloud Ways builds on top of the standard Google Cloud hosting services, and offers several other features such as managed server backups that the other services from Google do not. Cloud Ways offers a 3-day free trial of their service and then their plans start at $33 a month including pay-as-you-go plans. Cloud Ways offers extra 24/7 support for any questions or help that is needed

a) Microsoft Azure.[16]

Microsoft Azure offers a number of features for web hosting. It promises a 99.9% server uptime which means that on average it will be down just below nine hours per year. Microsoft Azure also offers a 12-month free trial which gives access to a number of features including limited storage space (5GB), text pattern recognition, as well as many others. It also includes a list of features that will remain free such as load balances and security measures[17]. After the end of the free trial, prices vary greatly depending on the services used such as storage space, location, and several other factors. At the low-end, Microsoft Azure offers packages in the $10 per month range. Microsoft Azure also allows users to cancel their service mid-month and receive a refund for the time they did not use. Some packages include automatic or manual scalability that can adjust/be adjusted biased on incoming traffic.

3. Analysis

The method used to analyze each of the cloud service alternatives was to look at all the desired characteristics and compare the different cloud hosting options to each

other. How much each one costs, how long of a free trial was offered for, and what features were offered as pricing increased were all taken into consideration. The amount of uptime was also taken into consideration even though all options had relatively comparable service uptime.

4. Chosen Approach

Each of the three options had pros and cons. AWS being the largest of the cloud hosting services offered the most options for different packages and prices even outside of LightSail. It also offered money back should its services be down longer than the promised nine hours per year. Google Cloud had the most support for helping people using their service. Microsoft Azure offered the longest amount of time to try out the service before having to commit to paying. All three options had relatively similar uptimes with Google Cloud having the longest downtime by about a day and a half. The three options also offered comparable scalability, all stating they had auto scalers which could make changes based on incoming traffic as well as making it easy to upgrade to a higher tier should the need arise. In the end the decision came down to cost.

Given that Microsoft Azure offers the longest free trial period of one year, and the fact that neither of the other two options came close, it is the best option for our cloud hosting service. One of the main concerns of our client is that of cost. With a 1-year free trial and some services that are always free our client would have plenty of time to try it out for themselves. Microsoft Azure also has the option of canceling the subscription at any point, allowing our client the freedom to change at will. Finally, much like the other two alternatives, Microsoft Azure offers pay-as-you-go options, only charging depending on what is used.

	Price	**Reliability**	**Scalability**	**Total**
AWS	3	5	5	13
Google Cloud	2	4	4	10
Microsoft Azure	5	5	4	14

Table 4: Cloud servers.

Table 4 above shows the scores that each cloud hosting service received on a scale of 1-5 with 1 being the lowest score and 5 being the highest. A score of 5 designates

the alternative as the best option in that category and all other scores are relative to it. All three services scored very similarly in reliability and scalability. Both AWS and Microsoft Azure promise an uptime of 99.9% with Google Cloud being only slightly behind. The main difference among the three is their difference in price which also takes into account the length of each service's free trial. With the shortest free trial and also the most expensive starting options Google Cloud received the lowest score. While AWS does have pricing options that start lower than Microsoft Azure, the fact that AWS only offers a 1-month free trial leads to it scoring lower.

5. Proving Feasibility

In order to prove our choice our group intends to utilize the free trial period in order to start hosting our project during development. Once proven effective, we then plan to continue to use the free trial for as long as we can during development of our project and continue to update all the various components until the project is handed over to our client, at which point it will be there decision on weather or not they intend to continue to pay for the chosen cloud service.

IV. TECHNOLOGICAL INTEGRATION

In this section, we discuss how our chosen alternatives will integrate with each other in order to provide the best solution for our project.

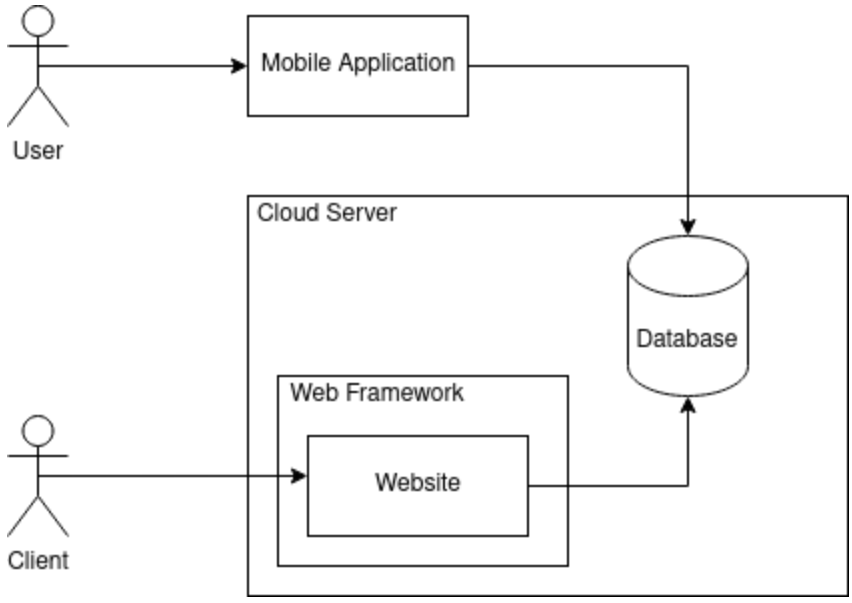

Figure 1: Our project structure.

The diagram in Figure 1 above provides an overview of how the different components need to interact. The website is created within the web framework and connects to the database. Both of these components live on the cloud server where they are easily accessible. The client uses the website to manage the database and the user interacts with the mobile application which gets data from the database. If the client wants to send a notification to the user, he or she can do so by making an entry into the database and the mobile application will find it when it does its periodic database lookup.

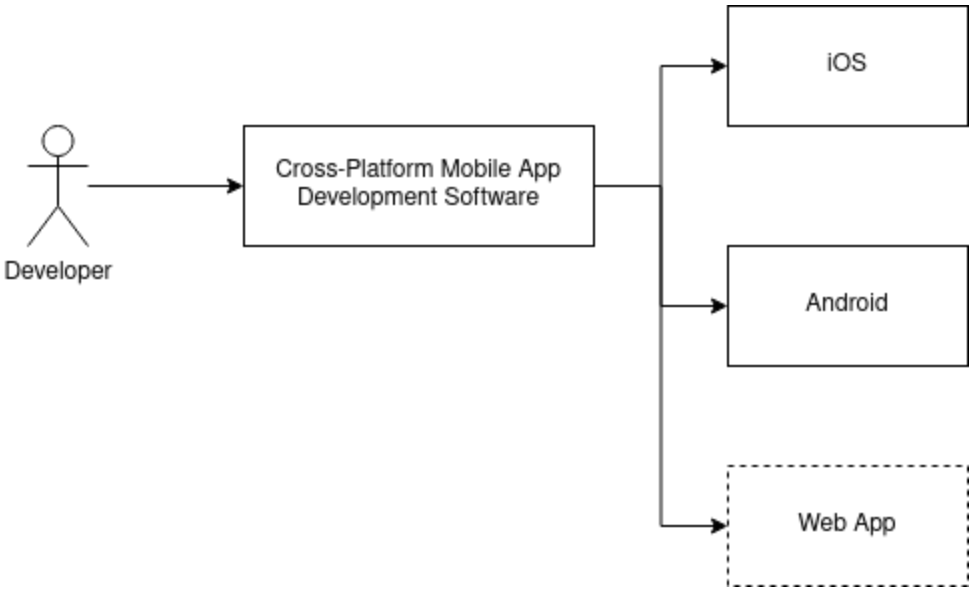

Figure 2: Creation of the mobile application.

This diagram in Figure 2 illustrates how the developer can use a single development software solution to create mobile applications in both iOS and Android using the same codebase. It also gives us the option to create a web application based on the same code.

The mobile application will access the database on a set interval if the mobile device has a connection to the internet. The plugin needed by Flutter that we will use to interact with the SQLite3 database is called "sqflite[18]." This allows the app to regularly check for updates, including any notifications or messages from the client. When the user looks for an assistive technology using the app, the app sends a query to the database and displays the results to the user.

The website provides an easy way for the client to manage entries in the database to keep an inventory of the assistive technologies, and to potentially send notifications to users. The website may also be used to host a web app if the client wants this functionality as well. The website is built using the Django web framework which has a built-in API to interact with the database.

Technology	**Mobile App Development**	**Database**	**Web Framework**	**Cloud Server**
Product	Flutter	SQLite3	Django	Microsoft Azure

Table 5: Our chosen solutions.

Table 5 above shows a summary of our chosen solutions for this project. The website which manages the SQLite3 database will be built using Django. The mobile application will be built with Flutter and interact with the database using the plugin sqflite. Both the database and the website will be hosted in a Microsoft Azure cloud server.

V. CONCLUSION

Our project involves creating a cross-platform mobile application that depends heavily on a database, so finding the right mobile application development software that works well with a secure database was the most important challenge. The client needs an easy way to manage the database and to communicate with the users of the application, so a web framework which integrates well with the chosen database is needed. A cloud service is needed to create a home for the database and web framework, and needs to be scalable according to the client's needs.

The problems we are solving include:

- Iit is difficult for a potential user to find what assistive technologies they need on their own because there are so many.
- There is no centralized database for assistive technologies which can make them difficult to find.
- Potential users often lack the experience or resources to find available technologies.

Our solution will accomplish the following:

- Giving the user tailored recommendations for assistive technologies based on their specific needs.
- Using a database to inventory available assistive technologies in a single location.
- Providing the user with the information they need on the recommended assistive technologies, as well as contacts to local resources.

In conclusion, we are confident that we can build a mobile application which provides the solutions to these problems using the technology we have researched.

VI. REFERENCES

1. https://flutter.dev/
2. https://reactnative.dev/
3. https://ionicframework.com/
4. https://dotnet.microsoft.com/apps/xamarin
5. https://www.oracle.com/database/
6. https://aws.amazon.com/rds/postgresql/what-is-postgresql/
7. https://sqlite.org/about.html
8. https://firebase.google.com/docs/firestore
9. https://www.djangoproject.com/
10. https://palletsprojects.com/p/flask/
11. https://rubyonrails.org/
12. https://www.cuba-platform.com/
13. https://aws.amazon.com/lightsail/
14. https://aws.amazon.com/compute/sla/#:~:text=General%20Service%20Commitment,the%20%E2%80%9CService%20Commitment%E2%80%9D).
15. https://www.cloudways.com/en/#laravel
16. https://azure.microsoft.com/en-us/services/app-service/
17. https://azure.microsoft.com/en-us/free/
18. https://pub.dev/packages/sqflite