Diverse Makers

Technological Feasibility

Version 2.0

04-05-2024

Project Sponsor: Dr. Jared Duval

Faculty Member: Michael Leverington

Project Mentor: Vahid Nikoonejad Fard

Team members:
Daniel Minichetti (Team Lead)

Kane Davidson

Eduardo De La Rosa

Elleana Negrelli

Aaron Ramirez

Overview

The purpose of this document is to outline the main technological challenges and decide on technologies that will be used in development. This feasibility study and analysis will guide key design challenges

Table of Contents

1.0 INTRODUCTION

Diversity is an important subject in science and technology. Leveraging different backgrounds, experiences, and points of view enriches STEM and brings unique insights to solving problems and making breakthroughs. Over 40 million Americans have a disability; however, research shows that disabled people are severely underrepresented in STEM fields. So much so that only 3% of people in the STEM workforce have a disability. People with disabilities tend to learn best in hands-on and individual learning environments, which can be difficult to accommodate in today's education system. Makerspaces can provide the resources to create this environment for STEM learning.

It is apparent from the workforce and college statistics that there are clear underrepresentation and support systems to aid those with disabilities in a career in STEM. Dr. Jared Duval aims to increase awareness and opportunities for those with disabilities with a social app designed to connect makerspaces to people of all disability backgrounds. By prioritizing accessibility, we can empower those with disabilities to explore their passions and aspirations in STEM by creating a collaborative and supportive environment built around makerspaces.

Since we are still in the early stages of the project, we are analyzing key technological challenges, researching alternatives, and deciding on a preliminary solution that best suits the needs of the project. The purpose of this document is to cover the bigger picture and technical approach while considering potential challenges that may arise in the process. We will analyze the major technological challenges in developing an inclusive platform, such as choosing which libraries and languages we will use to provide an easily accessible product. Ahead in subsequent chapters will be a careful examination of alternative programming languages and libraries, and from the analysis, determine the best plan for integration.

2.0 TECHNOLOGICAL CHALLENGES

There are numerous technological challenges that we will face throughout the development of our crowdsourced mobile application. For example, some of these challenges include: discovering a way to develop the application for iOS, Android, and possibly the Internet. After discovering a development platform, there will be several challenges specific to developing the application. For instance, we must find a way to host the learning resources on the application itself. If we do choose to host these resources in our system, we must build a database that integrates efficiently with our application. In addition to building a storage/hosting platform, we must find a method that allows users to submit possible learning resources. This could involve a group of moderators who approve, moderate, and manage any learning resources that are available on the application. After these resources are made available on our application, users must be able to interact and communicate with each other regarding these resources. This could be done with a combination of a comment board attached to each learning resource and a general forum for each specific topic.

After we overcome these back-end-specific challenges, there are several design challenges we must overcome. For instance, we must come up with a method for allowing users to manage their accounts. These account features would include signing up, logging in, managing their learning resources, submitting new resources to the application, and managing their application settings. Application settings inevitably lead to more technological challenges, such as accessibility options for users with disabilities, which could hinder their ability to use our application. These accessibility challenges could include things such as designing an intuitive UI for those with limited sight, captions for users with limited hearing, and an overall application layout that is welcoming to users who are not familiar with maker spaces.

Now that we have discussed the major technological challenges we may face throughout development, we will focus on discussing specific solutions and ways to overcome them.

3.0 TECHNOLOGY ANALYSIS

It is now time to delve into the specifics of the technological issues based on the demands of this project and early explorations with our client. We've identified the most prominent challenges being issues surrounding the back-end development, developing the app for both IOS and Android devices, ensuring the design of the app is as accessible as possible, and maintaining the learning resources on the app. For each technological challenge, we will first delve into what an ideal solution would look like for each issue, identifying key characteristics that are important to our project. Next, the alternatives section will cover the different possible approaches to solve each issue. In the analysis section, the best solution will be determined. After this section, the chosen approach will be selected. To conclude, we will outline our plans for further testing/validating our choice.

3.1 Back-end Challenges

The back-end of our application will be responsible for handling data, managing user authentication, and providing users access to the accessible Makerspaces. We must choose the best platform that will support us and allow us to scale if the app grows.

3.1.1 Desired Characteristics

An ideal back-end solution for our project should possess the following characteristics:

- **Scalability:** The platform should handle a growing number of users and resources without compromising performance.

- **Authentication and Security:** Since our application is based largely on community engagement, security and authentication are high-priority. User data must be well-protected to maintain the privacy of our users, and security must be ensured when

accessing the application. The framework should support user authentication as well, for example, signing up and logging in via email and password.

- **Ease Of Use:** Given that our team has never been an app of this scale before, the platform we choose must be beginner-friendly and popular enough for learning resources to exist. Ideally, there should be documentation on the platform for members to reference throughout the development process.

- **Cost-effectiveness:** The solution should be an affordable platform to develop our app on since we do not have an expendable budget.

3.1.2 Alternatives

- **Firebase:** Firebase is a back-end application development platform that uses NoSQL and supports development frameworks such as React Native and Flutter. Some of the features offered by Firebase include real-time databases, authentication, hosting, and cloud storage.

- **Supabase:** Supabase is an open-source alternative that offers similar functionality to Firebase and also supports popular app development frameworks.

3.1.3 Analysis

Both Firebase and Supabase offer advantages and disadvantages.

- **Firebase**
    - Pros: Firebase is easy to set up and uses a platform with a substantial amount of documentation to reference. On top of this, Firebase offers a wide variety of built-in features and cloud functions. As mentioned previously, some of the features required by our application will be authentication, storage, and real-time

database. Firebase is relatively affordable and offers a generous free tier along with a pay-as-you-go pricing model.

    - Cons: The main downside of Firebase is vendor lock-in. If we decided that Firebase would be the backend service, it would be very difficult to change our minds down the road. Firebase differs from other providers, and major changes would need to be made to the codebase and architecture to switch platforms. Depending on the magnitude of resources we will need to store and host, there may also be capacity limitations for cloud storage.

- **Supabase**
    - Pros: Supabase is very similar to Firebase, but the biggest advantage of Supabase is that it is open-source, offering more flexibility and customization compared to Firebase. Thus, it is designed for scalability. Besides this, the pros of using Supabase remain very similar to those of Firebase in that it offers key features such as authentication and a real-time database.

    - Cons: Supabase is relatively new and requires more technical expertise. It also has a much smaller community and ecosystem in comparison to Firebase, which means there are fewer resources and documentation available for members to reference. This may make it difficult for us to resolve any issues we come across. However, the Supabase community is rapidly growing.

3.1.4 Chosen Approach

Since each back-end platform has its pros and cons, we must carefully evaluate them based on our requirements. Both are very similar in the features they offer with the biggest difference between them being the fact that Supabase is open-source while Firebase is not. Given that our team is inexperienced, we wouldn't benefit from the advantages of open-source software, as we

don’t have the expertise to extend Supabase’s functionality. As a result, the open or closed-source nature of either platform will not play a role in our evaluations.

The following table compares our alternatives based on the characteristics that are important to us. The alternatives are compared on a scale from 1-5 based on how well the alternative accommodates each characteristic.

Characteristics	**Firebase**	**Supabase**
Scalability	5/5	5/5
Security	5/5	5/5
Ease Of Use	5/5	2/5
Average Score	5/5	4/5

As mentioned previously, ease of use is very important to us, and based on our metrics, Firebase is superior in that regard. As a result, we’ve tentatively chosen it as our back-end platform of choice.

3.1.5 Proven Feasibility

Since we have ultimately chosen Firebase as the most feasible platform, we must ensure that it will work with the other aspects of our application. Based on preliminary research, Firebase supports a multitude of frameworks, but we will need to validate this through various means of testing such as performing integration from the database into the application with test data (i.e. mock learning resources).

3.2 Development for Multiple Platforms

In our mission to create an inclusive and easily accessible platform for connecting individuals with disabilities to makerspaces, offering our application on the most popular systems (IOS & Android) will be a priority to ensure we can provide as many users as possible learning opportunities to support their STEM aspirations.

3.2.1 Desired Characteristics

The desired outcome for this app is that it must be supported for both IOS and Android devices and be easily accessible for anyone with an Android or IOS device that is still supported. There are specific requirements that need to be met in order to accomplish this task.

- **Programming Languages and Frameworks:** In the process of developing an app for both IOS and Android, choosing the right languages and frameworks will be important to have our code deployable on both platforms. Ideally, these languages will compile our application with exceptional performance.

- **App Store and Distribution Requirements:** For both the Apple App Store and Google Play Store, separate developer accounts will be needed for deploying our app. Each respective store has its own fees and submission guidelines, so ideally, we would want our app to comply with all the rules and regulations set by Apple and Google.

- **Cross-platform Consistency:** It is desired that both versions of the app on Android and IOS are as consistent with each other as possible while adhering to individual platform-specific requirements. As mentioned earlier, a solid framework will ease this challenge.

3.2.2 Alternatives

- **Native App Development:** Native app development is the creation of software programs that run on specific devices and platforms. Native app development requires different technologies than cross-platform options that are generally built around specific mobile OSs to deliver maximum performance. The main benefit of native development frameworks is that they have flawless behavior and compatibility with their respective OS. However, to accomplish the desire to distribute to multiple platforms, two separate codebases would need to be created for each respective native framework. Examples of apps that traditionally use a native development framework include mobile banking apps, language apps, or social media-based apps. [1]

- **Hybrid App Development:** This design approach starts with a cross-platform framework such as Flutter or React Native but later implements and embeds platform-specific constraints where needed. This approach also allows you to reuse design elements across multiple platforms which would benefit user experience and decrease variability between them. Maintenance would also benefit here as updates and bug fixes can be more easily applied universally across all platforms. Some well-known apps that use this design approach include Instagram, Twitter, Uber, Gmail, and Amazon [2].

- **Progressive Web App Development (PWAs):** Progressive web development involves building apps using web platform technologies that still provide the same user experience as using a native-built app for their respective platform. PWAs offer flexibility in that they combine the best features of traditional websites and platform-specific technologies. This means a PWA can be installed from the platform's app store or directly from the web, but can also still be installed like a platform-specific app. Some well-known apps that utilize PWA benefits include Pinterest, Trivago, AliExpress, and Spotify [3].

3.2.3 Analysis

Gauging what development approach we think would align best with our desired characteristics was found during further analysis of frameworks and looking at applied examples. It's important to reiterate that the main concern here is reaching as many individuals as possible, which will be found on IOS and Android, so ensuring we can efficiently accomplish that and find the approach that allows for the most seamless integration and maintenance was the priority going into our analysis.

- **Native App Development**
    - Performance: Since native app development involves creating and optimizing an app for a specific platform, the app will offer high performance since it will be using a framework, languages, and libraries that are built for that platform's specific OS.
    - Maintenance: In the context of our requirements, the maintenance for Native apps would be time-consuming as we would be running two separate code bases for each respective platform we are looking to support (IOS & Android).
    - User Experience: Generally, the user experience when using a natively built app will feel as optimized and consistent as possible because it is built around one specific platform.

- **Hybrid App Development**
    - Performance: Hybrid App development performance generally will achieve the same high level of performance found in native app development, arguably in less time.
    - Maintenance: Since with hybrid apps you will mainly write the app functionality in a single codebase, maintaining and updating the app as it evolves is generally an easier process as your code is cross-compatible from platform to platform.
    - User Experience: Similar to native apps, the user experience will be consistent and seamless but, in certain cases, not as optimized as it can be.

- **Progressive Web App Development (PWAs)**
    - Performance: Performance on PWAs are typically going to be slower than that of hybrid or native apps due to them commonly being lighter in size. Browser dependency can also result in performance issues.
    - Maintenance: Since you are mainly using web technologies, maintenance can be easier than both hybrid and native apps; however, there are some nuances, such as PWAs' heavy reliance on advanced native features that can be difficult to reach through web APIs.
    - User Experience: Overall the user experience was the most egregious aspect of using PWAs, as users are heavily dependent on either the browser they are using, which will lead to a largely inconsistent experience across the user base.

3.2.4 Chosen Approach

Although each approach presents different pros and cons, we must choose which one will best suit the requirements of our application. Native and Hybrid Development seem to offer the best user experience, while PWA Development would be easier to maintain. However, the overall goal is to pick the development approach that checks the most boxes for the desired characteristics. The table presented compares the alternatives based on how well they accommodate each characteristic on a scale of 1-5 with 1 being “Low/Poor” and 5 being “High/Great”.

Characteristics	**Native Development**	**Hybrid Development**	**PWA Development**
Programming Languages & Frameworks	5/5	4/5	3/5
App Store & Distribution Requirements	5/5	4/5	4/5
Cross-Platform	2/5	4/5	2/5

Consistency			
Average Score	4/5	4/5	3/5

3.2.5 Proven Feasibility

In the interest of building an app that is both scalable and accessible by as many users as possible, designing a hybrid app that is developed for both IOS and android platforms will be the most efficient approach to offering cross platform support. Since this approach will allow us to build one application for both platforms, it will cut down on time and resource costs, and make it easier to manage and update the app without having to update two separate codebases. This in turn will make the user experience from platform to platform seamless.

3.3 Design Challenges

Design is a key component of our application, seeing as our goal is for users with disabilities to be able to easily access and utilize the features of the application. This means ensuring resources for creating community partnerships, sharing project ideas, STEM learning resources, and entrepreneurial training for creating businesses owned by disabled people are presented in a manner that allows optimal user engagement. To cater to the diverse needs of our users, we will need to incorporate accessible features within our design.

3.3.1 Desired Characteristics

The desired outcome for this app is that all content, such as text, images, and videos, will be as accessible as possible for individuals with disabilities., various characteristics will be incorporated fundamentally into the app's design To satisfy this requirement.

- **Compliance with Accessibility Guidelines:** Our design will need to comply with established web/mobile application accessibility standards to ensure an extensive range of disabilities are being accommodated. This will ensure all users are being accommodated and identified.

- **Intuitive User Interface:** The user interface of our application must be straightforward and easy to navigate. This will be essential so that users can easily find what they need without unnecessary complexity.

- **Alternative Text for Images:** All of the images on the application should have alternative text descriptions so users who rely on screen readers can access the various digital materials. This is important so that users will be able to navigate through the application and understand the images being displayed on the screen.

- **Closed Captioning for Videos:** Video content must include closed captions to accommodate those who are hard of hearing. This is especially important as videos provide a plethora of information that will need to be understood in order to complete a tutorial or learn about a specific topic. With the inclusion of closed captions, users will be able to engage with the content effectively.

3.3.2 Alternatives

To address these design challenges, we have considered various UI/UX frameworks and tools. These technologies will help us leverage visual elements in the user interface to make content on the platform as accessible as possible for individuals with disabilities. Four possible options to achieve this include Material Design 3, Bootstrap, ARIA, and React Native.

- **Material Design 3:** Provides guidelines for designing accessible applications and websites.

- **Bootstrap:** Widely used library for HTML, CSS, and JS, with various plugins to create better accessibility for applications.

- **React Native:** Offers many built-in design components that are accessible by default. This can also be extended with additional libraries for further accessibility.

3.3.3 Analysis

- **Material Design 3**
    - Compliance with Accessibility Guidelines: Using the WCAG guidelines we found that Materials Design provides a comprehensive method for accessibility. This is evident in its high-contrast UI elements.
    - Intuitive User Interface: The interface of Material Design is easy to navigate with native UI components for each platform.
    - Alternative Text for Images: Includes sufficient guidelines for implementing alt-text for images. This can be validated with screen reader technology, which fits with our customer use case.
    - Closed Captioning for Videos: Material Design does not directly offer a built-in method for video captioning. This would entail integration with a third-party tool.

- **Bootstrap**
    - Compliance with Accessibility Guidelines: With the bootstrap accessibility plugin, we are able to test against many WCAG criteria and have found many integrations with ARIA and keyboard navigation. However, it requires some modifications to fully meet all the guidelines.
    - Intuitive User Interface: Bootstrap's grid system and design elements are very responsive and meet our requirements for intuitive navigation across a variety of devices. However, these interfaces have a generic look and feel with not as many native platform elements.
    - Alternative Text for Images: Bootstrap supports alternative text attributes but requires more intensive implementation as descriptive text will need to be added to image tags in the code itself.

- Closed Captioning for Videos: Similarly, Bootstrap requires third-party tools for closed captioning. Integration can lead to good results, with captions being displayed accurately on different platforms.

- **React Native**
  - Compliance with Accessibility Guidelines: React Native has optimized accessibility features, such as screen reader support and accessible components for mobile devices. We found the framework to be compliant with WCAG high and offered a very good foundation for accessibility within an app like ours.
  - Intuitive User Interface: The framework creates very intuitive and engaging interfaces with elements from a user's native operating system. We also found this to have smooth navigation throughout the interface
  - Alternative Text for Images: React Native supports alternative text for images through the AccessibilityLabel property, which offers effective usage with many screen readers.
  - Closed Captioning for Videos: React Native requires external libraries to support closed captioning but has extensive documentation available for implementation. With the integration of these libraries, we found the process to be direct and effective to achieve the correct results.

3.3.4 Chosen Approach

Based on our analysis, we have chosen to use React Native for our mobile application. This will provide a reliable framework for achieving the design goals we have set. React Native provides extensive documentation and community support for its accessibility features. This makes sure our app's UI is as intuitive as we would like it to be. In conjunction, we will also utilize the ARIA standards and reference all of our designs to ensure they comply with WCAG guidelines.

Characteristics	**Material**	**Bootstrap**	**React Native**

	Design 3		
Compliance with Accessibility Guidelines	5/5	4/5	5/5
Intuitive User Interface	5/5	3/5	5/5
Alternative Text for Images	5/5	5/5	5/5
Closed Captioning for Videos	2/5	2/5	4/5
Average Score	4.25/5	3.5/5	4.75/5

3.3.5 Proven Feasibility

To ensure that our user interface and design are accessible throughout the entire development process we will conduct user testing sessions with disabled individuals in makerspaces local in our community, to gather feedback on the UI/UX of the mobile prototype. Based on this user feedback, we will then implement iterative design changes to the app that focus on enhancing accessibility. We will also make use of tools in our development process that allow us to determine if our application runs into any WCAG compliance issues. This will help pinpoint what areas we need to improve so all of our users are addressed.

3.4 Maintaining Learning Resources

Maintaining software tends to be one of the most difficult and expensive components of software development. Our application is not an exception, we will need to build an efficient system, which can help keep costs down while at the same time providing an easy way to maintain the learning resources.

3.4.1 Desired Characteristics

- **A form of moderation of learning resources:** Building an efficient moderation solution is crucial for our application to ensure only relevant and appropriate content is made available to users. Content should be suitable for all audiences and contribute to improving STEM learning outcomes.

- **An intuitive process for uploading new resources to the application:** Users should easily be able to upload their own resources to our application. If the process for uploading is not intuitive, users will not feel obligated to contribute to the community of other users.

- **An intuitive process for modifying learning resources:** Learning resources should be able to be easily modified if needed. For example, if any content inside of the learning resource is no longer relevant, a moderator or user should be able to change it if necessary.

3.4.2 Alternatives

To address these content management challenges, we have considered the following platforms:

- **Firebase-Created Tools:**
    - Firebase offers several in-house tools such as moderation and database management. For example, a system that allows the use of text and image moderation. This form of moderation utilizes the Cloud Vision API to handle image moderation as well as an automatic moderator function for text moderation. In addition to moderation, Firebase also offers a database management system, which will be able to handle storing and interacting with learning resources that are uploaded to the application.
- **Third-Party Tools:**

- There are numerous third-party tools that integrate with Firebase applications. For example, Moderation API is a service that provides AI-powered moderation to Firebase applications. In addition to this, there are several database management systems that we could use instead of Firebase's integrated services. For instance, MongoDB and Cloudinary both offer storage solutions for applications.

3.4.3 Analysis

There are pros and cons for both technologies. One benefit to using Firebase's included tools is that they integrate seamlessly with our application since it will be created and deployed on Firebase's system. This means we are less likely to run into challenges implementing moderation and storage. While many tools integrate with Firebase, the chances of them doing so seamlessly are much lower if they aren't supported directly by Google. However, as our application grows, these Firebase tools, such as storage, may become much more expensive.

3.4.4 Chosen Approach

After careful consideration, we will choose to implement the built-in tools that Firebase offers. Since our team is fairly inexperienced in deploying Firebase applications, ease of use and compatibility are a major priority for us. Although there are some concerns about the scalability of these, the ease of use along with the fact that they are supported directly by Firebase outweigh our concerns.

Characteristics	**Firebase Tools**	**Third Party Tools**
Compatibility	5/5	4/5
Ease of Use	5/5	4/5
Cost	4/5	5/5
Average Scores	4/5	3/5

3.4.5 Proven Feasibility

Now that we have chosen to adopt Firebase's tools, we will test them throughout our development. Once our database has been created and filled with learning resources and users, we will be able to test how this data can be maintained. One possible test for maintaining these resources would consist of creating a set of sample learning resources. A portion of these resources would require moderation and then be changed as needed. This test would demonstrate the entire flow of uploading a resource, checking for moderation, and changing it as needed.

4.0 TECHNOLOGY INTEGRATION

Having selected Firebase as the back-end framework and React Native as the front-end framework, we will now outline how these technologies will integrate to create a cohesive and functional application that will support the disabled community for years to come.

Firebase will serve as the foundation for the application's back-end, so will provide essential services like a real-time database, user authentication, and cloud storage. The React Native Firebase SDK will be integrated into our React Native application to facilitate communication between the front-end interface and back-end components.

As stated previously, React Native will be used to develop the application's front-end to ensure native designs and cross-platform compatibility. The UI will be built with its built-in components to ensure consistent designs across operating systems. To enhance accessibility, we will use React native's accessibility APIs and adhere to WCAG guidelines.

5.0 CONCLUSION

To summarize, diversity is crucial in STEM fields in order to provide unique insights to solving problems. While there are millions of people with disabilities in the United States, they are still underrepresented in STEM. Because these people tend to work best in a hands-on environment, makerspaces are a vital aspect in enhancing their learning experiences. However, not all makerspaces are accessible to everyone; because of this, our crowd-sourced mobile application will act as a central hub for providing STEM learning resources for those who do not have access to a makerspace.

This document discussed the numerous technological challenges that we must overcome throughout the development of our application. These challenges consist of: back-end challenges, developing for multiple platforms, developing an accessible UI for users, and maintaining learning resources on the application. In order to accommodate back-end challenges, we will adopt Firebase, a Google-owned development platform that handles deployment, storage, and managing our application. Fortunately, Firebase offers a suite of tools that will handle resource management. These tools include text/image moderation and a database system that will be able to store and manage our resources. In order to create an accessible UI, we will adopt React Native, which also integrates with Firebase. Throughout the development of this application, we will communicate with Dr. Jared Duval, people with disabilities, and those who attend makerspaces. This constant communication will ensure that our application will be as efficient, beneficial, and accessible as possible.

Moving forward, we will start communicating with the stakeholders of our application in order to build a suitable plan and foundation. By doing so, we will be able to learn more specifically about certain priorities. While we develop this application, we will continue to communicate with stakeholders to ensure we stay on track, receive relevant feedback, and successfully address the issues discussed previously. We look forward to developing the most accessible, inclusive app to connect makerspace communities with those with disabilities who have struggled to access these materials for far too long.

6.0 REFERENCES

[1] Web apps vs. native apps vs. hybrid apps - difference between types ..., https://aws.amazon.com/compare/the-difference-between-web-apps-native-apps-and-hybrid-apps/ (accessed Mar. 22, 2024).

[2] “The Open Source Firebase Alternative,” Supabase, https://supabase.com/ (accessed Mar. 22, 2024).

[3] “React native accessibility: What, why, and how?,” {callstack}, https://www.callstack.com/blog/react-native-accessibility (accessed Mar. 22, 2024).

[4] R. Flores, “Utilizing bootstrap for ‘mobile First’ development,” Medium, https://medium.com/swlh/utilizing-bootstrap-for-mobile-first-development-f85c8b65118a (accessed Mar. 22, 2024).

[5] C. Chapman, “Why use material design? weighing the pros and cons: Toptal®,” Toptal Design Blog, https://www.toptal.com/designers/ui/why-use-material-design (accessed Mar. 22, 2024).

[6] “Material Design,” Material design, https://m2.material.io/design/introduction (accessed Mar. 22, 2024).

[7] “Content moderation for Firebase,” Company, https://moderationapi.com/integrations/firebase-content-moderation (accessed Mar. 22, 2024).

[8] “Content moderation with cloud functions for Firebase,” The Firebase Blog, https://firebase.blog/posts/2017/06/content-moderation-with-cloud-functions (accessed Mar. 22, 2024).

[9] MozDevNet, “What is a progressive web app? - progressive web apps: MDN,” MDN Web Docs, https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Guides/What_is_a_progressive_web_app (accessed Mar. 22, 2024).

[10] Ionicframework, “What is hybrid mobile app development: Hybrid vs native vs web,” What is Hybrid Mobile App Development: Hybrid vs Native vs Web,

https://ionic.io/resources/articles/what-is-hybrid-app-development (accessed Mar. 22, 2024).

[11] “Native App Development,” Raygun, https://raygun.com/blog/native-app- (accessed Mar. 22, 2024).

[12] “Mobile Operating System Market Share Worldwide,” StatCounter Global Stats, https://gs.statcounter.com/os-market-share/mobile/worldwide (accessed Mar. 22, 2024).

[13] Justin Kek, “Flutter vs react native - A guide from every experience level,” Justin Kek, https://www.justinkek.com/blog/flutter-vs-react-native/flutter-vs-react-native (accessed Mar. 22, 2024).

[14] Ionicframework, “Cross platform,” Ionic Documentation, https://ionicframework.com/docs/core-concepts/cross-platform (accessed Mar. 22, 2024).

[15] “Xamarin: Open-source mobile app platform for .NET,” Microsoft, https://dotnet.microsoft.com/en-us/apps/xamarin (accessed Mar. 22, 2024).

[16] “Write your first flutter app,” Flutter, https://docs.flutter.dev/get-started/codelab (accessed Mar. 22, 2024).

[17] “FAQ,” Flutter, https://docs.flutter.dev/resources/faq (accessed Mar. 22, 2024).

[18] E. G. and S. Deitz, “Diversity and stem: Women, minorities, and persons with disabilities 2023: NSF - national science foundation,” National Center for Science and Engineering Statistics., https://ncses.nsf.gov/wmpd (accessed Mar. 22, 2024).

[19] “Introduction · REACT NATIVE,” React Native RSS, https://reactnative.dev/docs/getting-started (accessed Mar. 22, 2024).

[20] National Center for Science and Engineering Statistics (NCSES). 2023. Diversity and STEM: Women, Minorities, and Persons with Disabilities 2023. Special Report NSF 23-315. Alexandria, VA: National Science Foundation. Available at https://ncses.nsf.gov/wmpd.