Technological Feasibility Analysis

Inclusive Solutions

Members:

Ricardo Chairez, Connor Kilgore, Olive Price, Monika Beckham, and Ethan Green

Sponsor:

Susan Purrington

Mentor:

Italo Santos

Nov 10, 2022

Overview:

This document highlights the analysis of our development to the software system, challenges with the product, and possible technological solutions to each issue.

Table of Contents

Introduction

Though over one billion people are living with some form of disability, the world is universally designed for the able-bodied person (1). There is a great need to initiate mass awareness among developers, architects and local bodies regarding accessibility of public spaces. Beyond accessibility, there is the issue of discrimination. Having a safe environment is important for anyone, but that need is not always met depending on the location or business. As of now, more than any current resource, the social platforms brought on by mobile devices have had the greatest impact on the representation of minorities and those living with disabilities. However, there still remain substantial flaws in the structure of our daily environments that limits those who may lie in marginalized groups.

Many assistive technologies exist to support disadvantaged groups, including platforms that work as a vessel for reviewing the suitability of public places. The dominant issue with these resources is that they are often incomplete, offering information only beneficial to few individuals, rather than a full-range of data that accommodates *every* individual. What our client, Susan Purrington is looking for, is a way to provide accurate and updated information for people of all disabilities and backgrounds about businesses nearby. Our client and her company have been gathering data in other ways of sourced information, but they are currently lacking the reliability and range that crowd-sourced data can supply.

Inclusive Solutions is partnering with Welcomed Here, a non-profit organization that seeks to create a world where no one struggles to encounter accessibility, acceptance, and support. In order to move closer to this goal, we will be developing a mobile application that provides *every* person a means to find spaces that meet their individual needs. Our vision is to maximize the use of crowd-sourced information surrounding businesses to provide relevant information for our client as well as other users about businesses before they make the risk of visiting them personally.

To accomplish that vision, our client has asked us to create a mobile app to review businesses on the basis of certain accessibility criteria. A user would be able to create an account, mark the criteria that they are concerned about, and search our reviews for a place that best meets their needs. This will make it easier for people with disabilities and other concerns to be able to know if a place is able to accommodate them and their specific individual needs. They will also need to be able to rate how well a place they visited was able to meet their needs.

In the development of this application, we need to consider the specific technical challenges that we will be facing. In the following sections, we identify the challenges we will be facing. Then we analyze each challenge, providing a chosen approach as well as alternatives, this is accompanied by a proof of feasibility by demonstrating that we can work with the software researched. Finally in our technology integration section we will be bringing all our technical challenges together to explain how they will fulfill ours and our clients goal for this project.

Technological Challenges

a. Server backend

We will need to consider what programming language or platform works best to perform server-based calculations and operations for our product. This includes considerations of the difference of operations during server runtime and processing power requirements. For instance, one language may be far easier to implement than another, but will have far more overhead that may overextend our hosting server's capabilities.

b. Database System

Much like mapping API, there exist many databases to choose from today. We will need to consider which database platform will synergize best with our chosen implementations for our backend and frontend systems.

c. Mapping System

Many mapping application programming interfaces (API) exist today: Google Maps, Bing Maps, OpenStreetMaps, etc. We will need to decide on a mapping API that both meets our client's projected budget while providing robust mapping and geocoding information for a wide variety of regions. We will also need to ensure that we can easily integrate the API we choose with our existing systems.

d. User Account System and Authentication

Our product will need to use some form of storing user information and allowing the user to access and revise this information on demand. The approach we choose will need to have effective security measures in place to prevent any form of security leak. We will also need to ensure that our account system is easy for any user to access and use.

e. User Interface

Being the aspect of our product that end users will interact with directly, it is crucial that we choose an interface technology that will be complex enough to create a good looking application while also integrating well with our existing workflow.

Technology Analysis

Server backend

Introduction

Our crowdsourcing application will be used for both for the benefit of every user as well as for research purposes conducted by Welcomed Here. By relying on a crowd of devices, we must implement a language that performs swift calculations which automatically update the collaborative documentation of each business or public space. The language component must be reusable, modular, and offer a range of activity performed by the user. The carrying out this software must prove to provide accurate overviews of every local space to the user, particularly on a large scale. The language chosen will call for a substantial collection of data, whilst considering the capability, background, and identity of each individual user that submits a review. The server backend must support an algorithm that allows for a customized reporting system based on user characteristics and needs.

Desired Characteristics

Our backend language should enable us to manage data changes coming from the frontend to create frameworks capable of handling many concurrent requests. Choosing a language that **team members are already proficient in** is preferred. The three desired characteristics of our language are as follows:

a. <u>Event-driven architecture</u>
   This design pattern enables events to be detected which act instantly to replace the current “state” of documentation for a business. Also known as asynchronous communication, the information flow between devices, backend servers, and the user interface is updated consistently in real-time, rather than periodically.

b. <u>Scalable</u>
   In backend, scalability refers to the way in which a system's performance increases or decreases in response to the amount of queries, requests, and other processing demands. For example, a program that is highly scalable would perform well on both small and large sets.

c. <u>Contain both object-oriented and functional constructs</u>
   Object-oriented programming (OOP) centers program design around data or objects. It emphasizes the organization of code into vital variables which each accomplish different operations. This method allows for modularity, making both development and troubleshooting much more efficient. The generalized criteria for OOP is *polymorphism, encapsulation, and inheritance*. Functional programming is different in the sense that it uses immutable and predefined data to tell a program exactly what to do.

Alternatives

1. Kotlin
   Officially released in 2016 by Jetbrains, Kotlin is modern, statically-typed, and targets several different platforms. It is the preferred language for Android development. It is interoperable with Java and is extremely concise, allowing for easy and safe development.

2. Javascript backend development frameworks such as Node.js
   Being one of the most used programming languages in the world, JavaScript is used for both frontend and backend development. It was invented by Brendan Eich in 1995 and became extremely popular in the early 2000s. Node is reliable for developing platforms that require a persistent connection from client to server; they are often used for real-time applications such as push notifications, chat, and news feeds.

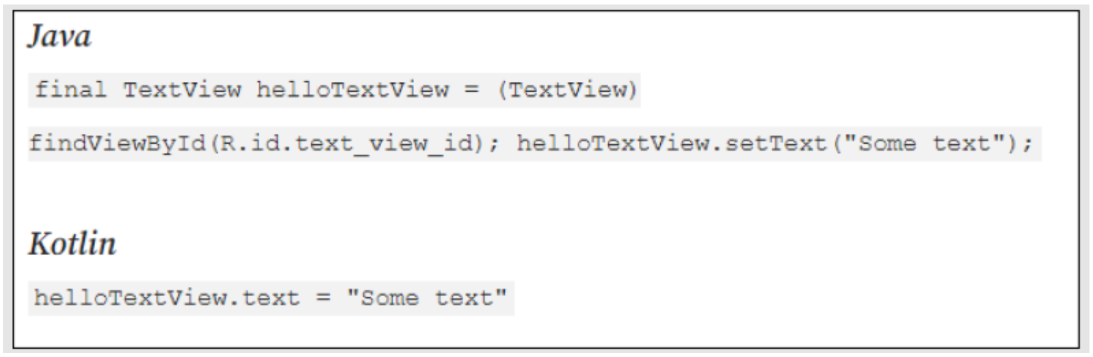
3. Java backend
   Java is a high-level language developed by James Gosling in 1991. There are four types of applications that can be created with Java, and one of them is Android app development. It is a simple, robust, and secure language.

4. PHP
   This is a server-side scripting language, meaning it is used to manage dynamic content, databases, session tracking, etc. It was created by Rasmus Lerdorf, primarily geared towards web development in 1993. It can be used to take care of web-service requests from the Android App to the server.

Analysis

1. **Kotlin**
   Due to its succinct and intuitive nature, Kotlin is known to increase team efficiency. Work is able to be done with fewer lines of code, as seen in Figure 1. It is also consistent with Java-related tools and frameworks. However, rough estimates show a 40% cut in the number of lines of code compared to Java.

    a. It should be noted that Kotlin supports event-driven architecture.
    b. Kotlin is often used to build server-side applications that scale to massive numbers of clients, i.e Google, Pinterest, Netflix, Tinder, and more.
    c. Both object-oriented and functional structures are supported.



Java

```
final TextView helloTextView = (TextView)
findViewById(R.id.text_view_id); helloTextView.setText("Some text");
```

Kotlin

```
helloTextView.text = "Some text"
```

Figure 1. Java and Kotlin textview comparison (2).

2. **JavaScript**
    a. Node.js is asynchronous in nature, allowing events emitted cause listener objects to be executed.
    b. As Node.js is an industry leader, it is one of the most scalable languages available. It is known to be highly efficient for the creation of high-end applications.
    c. Though JavaScript is not a class-based object-oriented language, it still supports polymorphism and encapsulation, as most dynamic languages do.

3. **Java**
    a. There are several event listeners available, making Java an event-driven language
    b. Java is known to have a rich set of caching solutions to aid scalability.
    c. Java is one of the most widely used languages for object-oriented programming.

4. **PHP**
    a. There are libraries, such as ReactPHP, which allow for event-driven, non-blocking I/O.
    b. Many developers prefer not to work with PHP because they do not find it scalable, though there are ways to make it more dependable
    c. PHP is an OOP and functional language.

we compare Kotlin, JS, and Java, as these languages fulfilled the three desired characteristics described above.

	Kotlin	**Java**	**JavaScript**
Benefits	● Reduces lines of code by 40% ● Safe from “billion-dollar mistake” or null-safe ● Interoperable	● Each team member is proficient in this language ● Extremely well-established; many libraries available ● Rigid structure makes application less prone to bugs	● Provides simplification of client-server interaction ● Reduces demands on servers
Challenges	● Because language is new, there are limited libraries and solutions ● Slower compilation speed	● Cumbersome syntax/readability issues ● More memory-consuming/slower compared to other languages	● More prone to bugs ● Client-side security often overlooked

Chosen Approach

Java will be the chosen approach for this case. Though this language has the disadvantage of tedious syntax, its structure is what our team is most experienced in. The benefit of a well-established language provides more learning tools and solutions than Kotlin, which only has limited resources available due to being relatively new.

	Event-driven	Scalable	OOP and functional constructs	Average
Java	5	5	5	5
Kotlin	4	5	4	4.3
JavaScript	5	4	3	4
PHP	3	2	5	3.3

Proving Feasibility

To further prove that this solution will fully combat our specified challenge, a series of client server programs will be written in Java.These programs will handle requests made by the client, performing operations similar to those we will need for our reporting systems. This will be demonstrated later in the semester.

Database System

Introduction

One of the main purposes of this project is to gather information to provide demographic analytics to our client. The answer to how we store this information is through a database that holds that information and organizes it such that we can make conclusions based on what the data is showing.

Desired Characteristics

- **Affordability**: having a server host that runs the database, though initially free, will eventually have an upkeep cost. It is important that the long term pricing is as low as possible while also fulfilling the needs of our client.
- **Scalability**: In early development, the expectation is that only a few people will make use of the services that the database provides, but in time the power and size of the database will need to grow, ultimately having a service that can scale with that anticipated growth is critical.
- **Usability**: Different server host services include varying libraries, api's and what capabilities they have. It is important that the services provided are able to fulfill our needs as developers to make other technologies possible. This especially applies to how we intend to manage the data collected into our database on the backend.

Alternatives

The first service that comes to mind is **Amazon Web Services (AWS)**, launched in 2006 by Amazon. This service is known through previous projects for members on the team and was very easy to set up. AWS is used by a lot of diverse big name companies such as Netflix, McDonalds, Samsung and Lyft.

Firebase is another service that can be used to store our database. Firebase was launched in 2011 by James Tamplin and Andrew Lee, this service is now owned by Google. This alternative was recommended to us by the group mentor, information was then found on the internet in various forums and pages. Firebase is utilized by companies such as Instacart, Twitch and Square App.

Analysis

	Amazon Web Services	**Firebase**
Usability	● MySQL server integration which means relational data storage model ● User authentication api ● Pre-existing familiarity among team members.	● Real time object based data storage (does not have mySQL server integration) ● User authentication api
Scalability	● Pay what you use model ● Capable of upgrading service model as the database requires more resources	● Pay what you use model ● Capable of upgrading service model as the database requires more resources
Affordability	● 12-month free tier to use a server with 750 hours per month. ● Billed automatically after trial period. ● Future pricing varies depending on the instance used as well as the size of web service needed.	● Bases the free tier based off of a monthly allotted amount of read-write queries to the database that will be blocked after the capacity is reached. ● There is no limit to how long the free tier is used. Once the project scales up, The pricing will be based on how many queries are made to the database, as well as the required storage size needed to withhold all data ● Queries are blocked after the free limit capacity is reached when using the free tier

Chosen Approach

Ultimately the best approach is to go with Amazon Web Services. The reasoning for this is the use of a relational database. Our client will be expecting to have data from lots of different datapools to be interacting with each other to create demographic information. Firebase's object based data storage can heavily limit the ability to do such functions. Additionally AWS will be a cheaper service in the long run when the project scales up.

Here is a table analysis of why AWS is the best technology to use:

Service	**Usability**	**Scalability**	**Affordability**	**Average**
AWS	5	5	3.5	4.5
Firebase	2.5	5	4.5	4

Proving Feasibility

In order to prove feasibility of this approach, a demo will be used later this semester that will demonstrate a simple upload of data to a SQL server that is hosted on an Elastic Compute Cloud (EC2) instance followed by a simple download of data from that database in our technology demo.

Mapping System

Introduction

A key component of our MVP is the ability to search for places that meet your selected needs by distance and list of needs met. Unless we have a way to find the distance from your current location and what places would be within a given distance of you, that functionality would be impossible for us to implement in the 8 months we have to design a solution. Fortunately, many online mapping sites already have this information and provide APIs to developers to leverage their existing solutions to these problems, but they each have their own drawbacks that we need to address.

Desired Characteristics

- **Geocoding**
  We will need a system that is able to encode an address into latitude and longitude coordinates, and that would be able to decode coordinates from the phone's location provider into an address. Ideally, such a tool would be able to decode the address of a given set of GPS coordinates directly into a predefined data structure that we can easily manipulate to get the data from the API to the database without too many intermediate function calls or cumbersome nesting.

- **Place Data and Search**
  It would simplify the review and upload process greatly if we could leverage our map provider's existing database of business names, addresses, categories, and other information instead of requiring the user to manually enter everything themselves.

  To be able to do that effectively, however, we'd need to have a mapping system that provides high-quality, up-to-date information about businesses with as few businesses missing as possible. The business data should also have as few closed businesses as possible, to prevent users from leaving a review or searching for information about a business that no longer even exists. Once again, it would be ideal for our chosen solution to provide this data in a native data structure, but another format would work provided we can get the data we need easily.

- **Cost Structure**
  Because the client will need to continue to pay for the mapping system access long after the project is completed, we should seek to find a solution that is cost-effective and has a simple, consistent cost structure which will make it easier for them to budget for, while still allowing the client to scale our solution up easily as more users start using the app.

Alternatives

1. **Apple MapKit JS**
   Apple MapKit JS is a JavaScript library used to access and display data from Apple's Apple Maps service. While Ethan was still trying to decide if Google Maps Android SDK or the Bing Maps API were better, Ricardo informed the team that Apple MapKit is included in the cost of an Apple Developer Account subscription. Later research revealed the existence of a JavaScript version of the MapKit libraries which could theoretically be hooked into an Android app.

   Apple wrote this framework in 2019 as a way to allow developers to use Apple Maps in their websites without having to switch the user into the Apple Maps app on iOS and Mac devices. It has since been adopted by the DuckDuckGo search engine as their main mapping and place info provider.

2. **Google Maps Android SDK**
   Google Maps is the leading navigation and point-of-interest information provider in the market, and they provide a set of SDK libraries that can be used in an Android app. The team had known of the existence of this framework before starting the project.

   The first version of this framework was created by Google in 2008, with the launch of Android as the sole navigation and location data provider. Due to its long history as a navigation and geospatial data provider, Google Maps is used by the vast majority of Android apps that need maps or location data.

3. **Bing Maps REST API**
   The Bing Maps REST API is a web request system that Microsoft created to allow programmers to get data from Bing Maps that matches with their queries. Ethan discovered the existence of Bing Maps as a geocoding solution early on, before the map needs became more complex than simple geocoding.

   Bing Maps is used largely by smaller organizations that don't want to pay as much as Google Maps costs for a location service provider.

Analysis

	Apple MapKit JS	Google Maps SDK	Bing Maps API
Native Language Objects	No, this framework is JavaScript only, so it won't be able to give us Java or Kotlin (Android's 1st class languages) objects without a conversion function. Will require a WebView in the app to run	Yes, the Android libraries define the return type class in both Java and Kotlin, in addition to the query functions we'll need.	No, but it returns JSON that can be serialized into an object through the GSON library
Location Data Quality	Less than ideal: Most locations are where they should be, but there are a few duplicate locations and locations in the wrong place.	High quality: Most businesses self-report their own locations and Google verifies information submitted by users about businesses.	Acceptable: most places exist in their maps in or around NAU campus, although some are miscategorized.
Search Command Structure	A function that takes in a SearchOptions object and executes a search based on those options, and returns a list of places matching the description	A function that takes in individual search options and returns a native object based on those options. It lacks a "radius" field, so we'll have to calculate corners of a box from the radius	A URL with fields that are populated with parameters for our requests. This makes automatically suggesting a place from the search difficult, if not impossible
Geocoding	Create a mapkit.geocoder object with the current location, then call its lookup method with the search string, or call its reverse lookup method with a coordinates object	Create a geocodeListener object to get the results, then pass it into either the appropriate functions to run a forward or reverse geocode	Request using a url with a SpatialFilter(lat,long,distance) or SpatialFilter(address_string, distance) to reverse or forward geocode from Bing's data
Cost Structure and Scalability	You get up to 25,000 service calls and 200,000 map instantiations per day with the $99 a year Apple Developer membership. More uses require negotiation with Apple, and it is unclear if nonprofit discounts are available.	Each type of transaction has its own associated cost, but you can set usage limits per day to keep your bill from spiraling out of control. Each developer/organization gets $200 free every month, with nonprofit discounts available after meeting with Google.	Developers and nonprofits get free usage, developers being limited to 125000 billable transactions a year, and nonprofits being limited to 50,000 per day. If more billable request transactions are needed, you must purchase a volume license from Microsoft or a distributor.

Chosen Approach

With all the factors considered, it appears that the Google Maps SDKs are our best choice. The search and autocomplete APIs are already in Java, so we won't need to do much (if any) parsing, making it superior to MapKit JS. Google Maps also has Java libraries, so we don't need to format query strings using cumbersome string.Format() calls. Their cost structure is also the most scalable, allowing the client to keep the app running while the user count grows without needing to renegotiate their contract with the map provider. We'll also be able to trust the data we receive about what type of business it is and where it is located more than we would with our other options.

	Native Language Objects	**Data Quality**	**Search Command Structure**	**Geocoding**	**Cost Structure and Ease of Scalability**	**Average**
Apple MapKit JS	0	2.5	5	5	2.5	2
Google Maps SDK	5	5	3	5	4	4.2
Bing Maps REST API	2.5	3.5	3.5	5	6	3.2

Proving Feasibility

In order to prove the feasibility of this approach, we will write a small demo program that gets the user's location from around campus, gets the names, addresses, and categories of the buildings, and displays them in a list. It will then get the GPS coordinates of the SICCS and Engineering buildings and display those on a map.

User Account system and Authentication

Introduction

Central to our client's vision of our product will be the ability for any of our users to maintain personal information and preferences for easy retrieval and revision as needed. An account system would allow all that as well as the ability to access this information from as many devices as one needs. Account systems and implementations exist all over the internet, so we have many implementations to choose from. A key issue we will have to address is the threat of user information being compromised in one way or another. The information we will be handling could be very dangerous in the wrong hands, so any interface for this information must be as secure as possible. Our solution will ideally have a strong link between our frontend and backend for seamless authentication.

Desired Characteristics

- **Secure**
  - Account system related information should be as difficult as possible for malicious actors to access. Part of this is ensuring that login credentials are encrypted.
- **Easy to implement**
  - Considering the amount of time we'll be spending on the system as a whole, we should choose an approach that can be easily implemented in our chosen backend and frontend languages.
- **Easy to license**
  - The technology we choose to implement must have compatible licensing to be used in our project. This includes low or free of cost solutions in particular.

Alternatives

1. **Firebase Authentication**
   Firebase is an application development platform owned and operated by Google. It includes many features for building, releasing and monitoring applications. Of primary interest to our project is the authentication libraries and platform included with Firebase.
2. **pac4j**

pac4j is a security framework developed for Java. It includes many libraries with support for a variety of authentication methods as well as different client methods to access them.

3. **JWT**
   JSON Web Tokens (JWT) are a format for transferring specific information in a secure way between client and server. Specifically, authentication data is sent as encrypted JSON data.

Analysis

1. **Firebase Authentication**
   a. Credentials must be run through a given method to attempt login. This method encrypts credentials and sends them directly to Firebase. While we cannot see what Firebase exactly does with the given credentials, it is safe to assume that they are handled appropriately due to Google managing them.
   b. Implementing a sample Android application using provided Firebase libraries took very little effort. Firebase libraries have intuitive method names such as “signInWithEmailAndPassword” as well as extensive documentation for creating accounts, logging in, and accessing account data.
   c. Free Firebase plans allow unlimited access to non-phone verification methods and a limit of 50,000 monthly active users. This plan is free to use for any purpose, with a pay-as-you-go scale for active users beyond this limit.
2. **Pac4j**
   a. The libraries automatically encrypt sent data when using IP address or Lightweight Directory Access Protocol connections. When using SQL-based authentication, password data must be manually encrypted using an external library.
   b. Implementing pac4j in a sample Android application took a lot of management of individual URLs and locations to query for authentication. It seems that due to the flexibility of the libraries, there is much overhead to account for the difference between an Android client and pure Java.
   c. Pac4j is licensed under the Apache License 2.0, which permits royalty-free use and redistribution for any purpose.

3. JWT

a. JWT is inherently secure as the RFC 7519 defining the standard for JWT states that it must include an encryption algorithm. This will, however, require that we maintain a "secret" or other encryption keys.

b. Since JWT follows a JSON format, it is very easy to parse all information passed in by a JWT. From there, it would be trivial to compare username and password data against database information. We would, however, have to make an implementation for maintaining stateful connections.

c. Since JWT is just a public standard, we would completely own our implementation of the standard.

	Firebase Authentication	**pac4j**	**JSON Web Tokens**
Benefits	● Very easy to integrate into mobile applications. ● Native support for a large variety of authentication methods. ● Well documented.	● Diverse plethora of libraries for most implementations ● Available under a permissive free-use license.	● Well documented, universal standard. ● More flexibility for individual functionality. ● Very easy to parse input information.
Challenges	● Depends on an external service. ● Costs of service will increase as our user base increases.	● Will require the use of external libraries for encryption. ● The lack of overhead will require much more manual implementation on our part.	● Much more work to be done for implementati on. ● More management of each individual auth variant.

Chosen Approach

Considering our desired characteristics and overall impression from the analysis of each alternative, we constructed the following table summarizing the benefits and challenges of each approach:

	Secure	Easy to implement	Easy to license	Average
Firebase Authentication	5	5	3	4.3
pac4j	3	3	4	3.3
JSON Web Tokens	4	3	5	4

Given the constraints of the scope of this project, as well as our analysis of each approach, we have chosen to use **Firebase Authentication** for our User Account System solution. For a project with a larger scope or a team dedicated to this aspect of the project, the other approaches could potentially yield a very robust and powerful system, but for the purposes of our project we must ensure we do not spend unneeded resources. The limits of the free license for Firebase are extremely lax for a project in development and provide many tools to reduce our workload while not compromising security.

Proving Feasibility

In our analysis, we constructed a simple authentication test using Firebase Authentication. Going forward, we will expand upon our simple test to include far more authentication methods, and establish a consistent session with confirmed credentials.

User Interface

Introduction

Our app is made for accessibility and usability, so we want our user interface to reflect that. There are many frameworks to choose from when it comes to mobile development. Some are exclusive to Android or iOS, while some are cross platform. When choosing a framework to build our user-interface, our main concern is that it can integrate well with our chosen map framework, Google Maps. Map integration is a large part of our app, so we need a framework that works well with it. The user interface is a key part of our app, so we need to choose a UI framework that will allow us to create an experience that is accessible, seamless, and intuitive for all users.

Desired Characteristics

- **Integrates with Android and Google Maps SDK**
    - Map integration is a large part of our app, so we need a UI framework that can incorporate it. We are creating an Android app and using the Google Maps SDK, so the framework must be compatible with both.
- **Easy to implement**
    - With little time we have, we need to choose a UI framework that allows us to create something intricate while being intuitive to use.
- **Robust/Feature heavy**
    - Our user interface is going to be relatively complex. We need a UI framework that is not limited in its potential. Not only are we planning on adding a map, but the app needs to be accessible to all users. We need as much flexibility as possible when creating the user interface.

Alternatives

1. **XML/Layout Editor**
   XML is a markup language similar to HTML. For native Android development, XML is used to define the layout of the UI elements. The layout editor is a tool built into Android Studio that allows developers to modify the UI with a GUI instead of working directly with the XML. As of 2022, XML is the most popular way to create user interfaces in native Android.
2. **Jetpack Compose**
   Jetpack Compose is Android's recommended UI framework for building UI in native Android. It is a very new framework, having been released in 2021. It uses

a declarative API that requires less code than XML. Compose is written exclusively in Kotlin and incompatible with Java.

3. **SwiftUI**
   SwiftUI is a UI framework for building UI in native iOS. It is a very new framework, having been released in 2019. It uses a declarative API that requires less code than the imperative alternative, UIKit. SwiftUI is written exclusively in Swift and is only for iOS applications. Jetpack Compose is very similar to SwiftUI in that they are both new declarative UI frameworks that are written in their native language.

Analysis

1. **XML/Layout Editor**
   a. XML is the tried and true way of creating user interfaces in native Android. It is directly compatible with Google Maps, so integrating it into our app would be seamless and intuitive.
   b. Out of all the UI frameworks considered, XML has the most documentation and familiarity among Android developers. While Jetpack Compose and SwiftUI are newer frameworks, XML has been around much longer and benefits from having years of documentation and experience among developers. This could make our experience much easier, as we will have more resources at our disposal and not have to worry about the quirks that come with using a newer technology.
   c. XML is the most capable and robust way of making user interfaces in Android. Being around as long as it has, there is a tutorial or documentation for almost anything that needs to be implemented.
2. **Jetpack Compose**
   a. Jetpack Compose is exclusive to native Android development. It is capable of integrating Google Maps, however it is newer and does not have the flexibility XML has with its integration. Compose is capable of integrating Google Maps, however it does not directly support it so there is some hacking involved to get it to work properly.
   b. Android claims Jetpack Compose to be an easier and more intuitive way to create user interfaces. However, Compose lacks sufficient documentation because it has only been out for a year. Using this newer technology could be tricky, as we would have to deal with all the quirks and lack of information inherent in choosing to use a solution that has only been out for a year.
   c. Jetpack Compose is a newer technology, similar to SwiftUI. Like SwiftUI, it is lacking in features and capabilities that its predecessor (XML) has had

for years. The most crucial capability being direct compatibility with Google Maps.

3. **SwiftUI**
    a. SwiftUI is exclusive to iOS development and incompatible with Android. It is capable of integrating Google Maps, however MapKit would be the better choice for SwiftUI as it is native to iOS (and free). Being incompatible with Android eliminates SwiftUI from consideration.
    b. SwiftUI is the most intuitive and easiest UI framework listed. It is similar to Jetpack Compose in how it works but has the advantage of being out for two years longer. It has more documentation and tutorials than Compose, but still not as much as XML.
    c. SwiftUI was built as a successor to Apple's other UI framework, UIKit. While Apple claims SwiftUI to be the future of creating user interfaces for Apple devices, it is simply incapable of doing many of the things UIKit is capable of.

	XML/Layout Editor	**Jetpack Compose**	**SwiftUI**
Benefits	● Similar to HTML, which we are all familiar with ● Most tried and true way of creating user interfaces in Android ● Lots of documentation and tutorials ● Directly compatible with Google Maps	● The future of creating UI in Android, more future proof ● Can integrate Google Maps ● Less code and files than XML	● Easiest UI framework to use ● Newer technology, future proof
Challenges	● Older technology, might be phased out by Jetpack Compose in the future ● Requires more code and files than Jetpack Compose	● Does not have extensive documentation ● Newer technology, lacking capabilities XML has ● Is capable of integrating Google Maps, but not directly compatible.	● Incompatible with Android ● Limited in its capabilities

Chosen Approach

	Integrates with Android and Google Maps SDK	Easy to implement	Robust/Feature heavy	Average
XML/Layout Editor	5	3	5	4.3
Jetpack Compose	3	4	3	3.3
SwiftUI	0	5	3	2.6

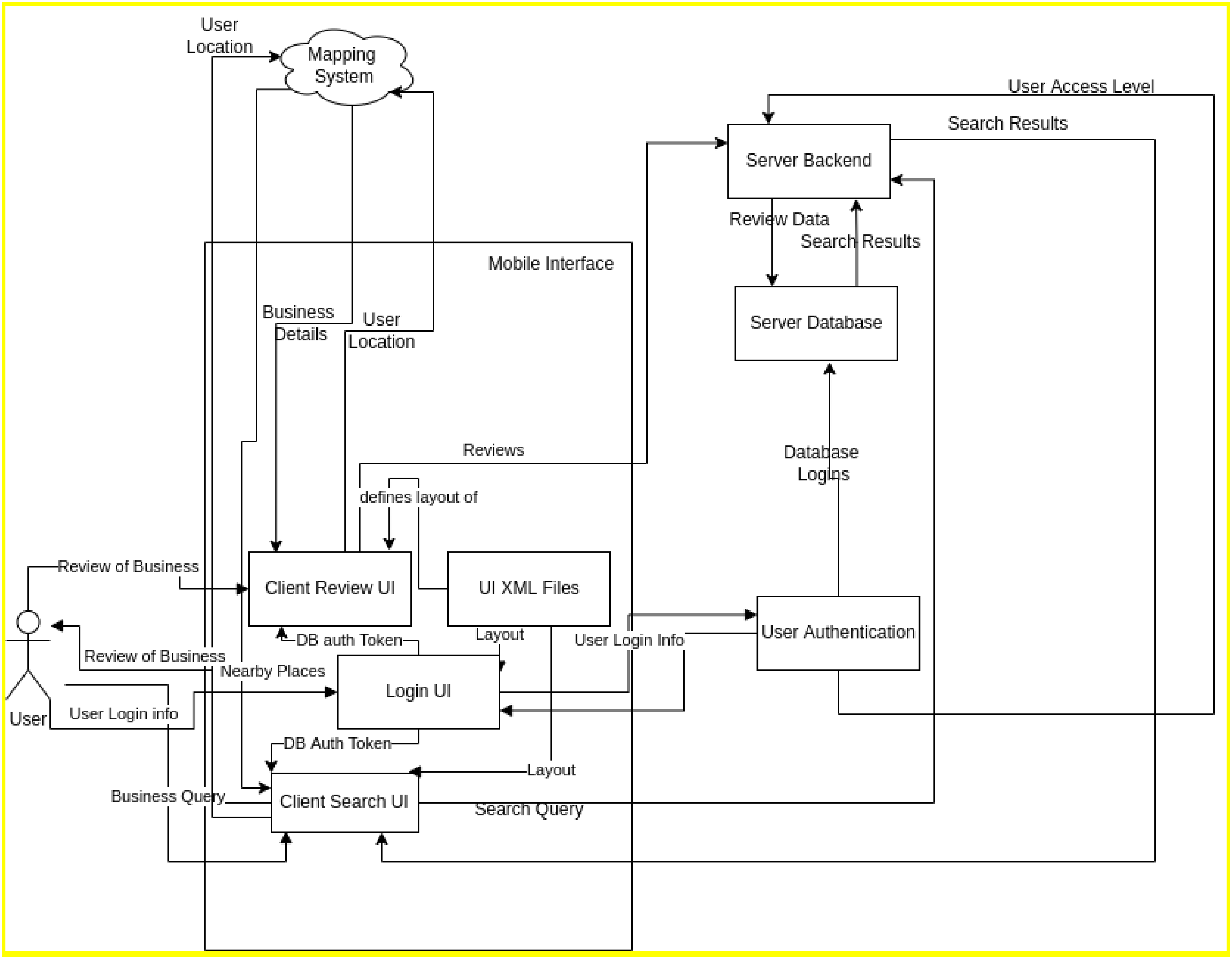
Given the presented UI frameworks and tools, **XML/Layout Editor** is our chosen approach. SwiftUI, while better overall than both XML and Jetpack Compose, is just not feasible because it is exclusive to iOS and we need something that works with Android. XML and Jetpack Compose are the only two realistic options we have. Jetpack Compose is a newer and more intuitive UI framework that is recommended by Android, however its indirect compatibility with Google Maps and lack of documentation hurt its consideration. Google Maps is a large part of our app and we want something that is tried and true, so we believe XML is the best way to go.

Proving Feasibility

To prove feasibility, we will mock up the user interface in Android Studio with XML and Jetpack Compose respectively. SwiftUI, while a powerful technology, is not worth testing because it is not compatible with Android

Technology Integration

All of the technologies previously mentioned need to work together with a comprehensive strategy that will help our design accomplish what is required from our client. To do so a diagram was made to show how the different components will communicate with each other:



The user sends their reviews and queries to the mobile interface, which is divided into a Client Review UI and a Client Search UI after the user is logged in through the login UI. The authentication system sends the Login code a DB Access Token that the other modules will then use to either search for businesses or submit a review, and sends the server backend the access level of that user. In the process of searching for places or submitting a review, they send the mapping system their current location. Search UI then requests a list of nearby businesses that it can use in its search, and Review UI then gets the ID from Google Maps that gives them the category and ID of the business to use in the review. The Review UI then sends the review to the Server Backend, which then processes it and inserts it into the database. The Search UI submits its requests to the server backend, which then queries the database and passes the data back to the client, which then presents the results to the user.

Conclusion

People have individual needs that must be fulfilled when going to a public space and right now there is not a resource to find information on a given location that fully meets the accommodation needs of every individual. Our team, Inclusive Solutions, is partnering with Welcomed Here to create a mobile app that assists individuals in finding spaces that fulfill their needs. We will accomplish this by building an application that allows users to enter various criteria focusing on inclusivity and rate with these in mind.

In order to create our application, we must first address the following technical challenges:

- Server Backend
- Database System
- Mapping System
- User Account System and Authentication
- User Interface

This document has outlined our research into viable solutions and implementations for each of these problems. Our considerations for each of these challenges focused mainly on the scalability as this project progresses, future costs associated with our solutions, and what will work best for us going forward to ensure we deliver the minimum viable product by the conclusion of this project. As per such, we came up with the following solutions respectively:

- Java
- Amazon Web Services
- Google Map SDK
- Firebase Authentication
- XML/Layout Editor

Moving forward, we will develop proof of concept demonstrations for each of these solutions to show how we will be integrating them in our complete product. We are confident that our chosen solutions will support a complete and intuitive minimum viable product by the conclusion of our work.

References

(1) United Nations. (n.d.). *Factsheet on persons with Disabilities Enable*. United Nations. Retrieved November 7, 2022, from https://www.un.org/development/desa/disabilities/resources/factsheet-on-persons-with-disabilities.html

(2) *Difference between Java and Kotlin in Android with examples*. GeeksforGeeks. (2020, May 27). Retrieved November 7, 2022, from https://www.geeksforgeeks.org/difference-between-java-and-kotlin-in-android-with-examples/

(3)“Firebase Realtime Database.” *Google*, Google, https://firebase.google.com/docs/database.

(4) Daly, Donald J., and Donald J. Daly. “Economics 2: EC2.” *Amazon*, CGA Canada Publications, 1987, https://aws.amazon.com/ec2/instance-types/.