Introduction

We are Logan Francisco, Garrett Hammock, Kelly VanMeter and Junmin Yee. Our project is named Landmarked, but currently this is a working title and subject to potentially be changed. The following is a document that outlines our technical specifications.

Project Description

Landmarked is a mobile based app that will allow a user to point their phone at a landmark and retrieve information about it. Because landmarks are subjective and will vary person to person, let’s consider a landmark anything that you would expect to find on Google Maps. Our intended usage would be for landmarks such as mountains, lakes and other natural phenomenon, historical sites and parks. We intend to provide information on smaller scale landmarks as well, but this will depend on the quality of our source.

In order to discern between landmarks near and far, we will implement an “aiming” feature. The user will be able to tilt their phone upwards and downwards to measure distance and toggle the landmark they want. For example, in a straight line travelling outwards from the direction the phone is pointed, a lake could be sandwiched between a mountain and a park. The user could toggle the lake by tilting their phone upwards beyond the range of the mountain but not enough that the lake is skipped and the park is toggled.

To make this possible, we’ll use a phone’s GPS features to locate the user and the direction they are pointing their phone. Based on the location and direction, we’ll develop an algorithm that will allow the user to toggle the landmark they desire. The algorithm specifics are not yet defined, but it will be based off the location and direction from GPS.

Project Functionality

Our project has a very specific functionality that must be implemented. However, we have a wide range of additional features that need to be fleshed out into core functionality or stretch.

Core functionality

a) Providing the user with the name of the landmark they are toggling

b) Store the landmark in a database for future retrieval

c) Allowing the user to create custom landmark points and storing them in DB

d) Allow landmark data to be stored locally, on the user’s phone, till it can be uploaded

1. At a minimum, we need to be able to provide the name of a landmark for a user. If we can’t deliver a name, then we don’t have an app. Identifying landmarks by name is THE core feature.
2. Storing a user’s landmark history in a database doesn’t inherently benefit our core features. However, because it opens the door for so many of our secondary and stretch goals, it will be treated as a core feature. We can implement as many or few of our stretch goals (outlined below) as we choose, but without a database storing past landmark history many of these secondary goals will be impossible.
3. Because landmarks are subjective, we’ll allow users to create their own landmarks and store them like we would any other landmark. This will be a useful feature for users that discover something they want to remember, or need to remember where something is located. The information we provide on any landmark is only going to be as good as our sources, so we need some mechanism to store locations that are important to the user but not necessarily available from our sources.
4. Because many landmarks are in areas without mobile data connections, we need to store landmark data locally in situations where no internet connection is available. Because some user phone plans don’t include data connections, this feature will allow them to store their landmark history until they have a wifi connection.

Stretch Functionality

1. Providing historical or relevant information about a landmark
2. A user ranking system for landmarks
3. ADD SOME OTHER THINGS HERE I CAN’T REMEMBER SAME WITH CORE FUNCTIONALITY
4. For users that want more information about a landmark than it’s name, we’d like to implement a feature that delivers both the name and some additional Wikipedia style information about the landmark. Because the information we pull and provide will likely be very verbose, this will likely be an opt-in feature.
5. We would like to implement a system that allows users to report that they’ve been to a landmark. This feature would be useful to users that aren’t interested in heavily traveled landmarks, or perhaps are only interested in heavily travelled landmarks. This feature doesn’t improve our core functionality, but will help niche users determine how interested they are in visiting a landmark.

Intended Users

Firstly, it’s important to note that due to the simplistic “point and shoot” nature of our app, there’s no demographic of potential user that won’t be able to use our app because they don’t understand it. Our UI ideas don’t take a high level of technical ability or maturity to use. Some of our stretch functionality does cater to a more niche group of users, but the core feature of our app can be used by anyone that can start the app and point the phone.

Since the entire purpose of Landmarked is to identify landmarks, we’re trying to entice adventurers, travelers and explorers. To use our app, you have to point your phone at a landmark. To point your phone at a landmark you have to be near the landmark. These are the users we want – outgoing people that are likely to be near a landmark to identify.

Our UI goal is to be simple enough that a young child could pick it up and intuitively use the app, as many are familiar with games and apps from a young age. The challenge will be maintaining that simplicity, but refining it to a point that older people feel they’re using a sophisticated app.

Platforms

Our platform target is Android mobile devices. Depending on how our stretch time goes, we could also target iOS mobile devices. With the resources and time we have, it may not be realistic to attempt development on an Android app and an iOS app. Amazon cloud services look promising for potentially hosting our services. Another stretch goal involves integrating our app with desktop software, and that platform would likely be limited to windows.

Expected Costs

Our biggest cost will likely be cloud services. Many of the big names offering these services offer some time of free tier. Unfortunately, the amount of data that comes with the free tiers won’t come close to the amount we’ll need for development, testing, and eventually production. Amazon seems promising, but ultimately we will end up going with the cheapest provider that offers the best tools and support. These costs shouldn’t cripple us, but are important to bear in mind because they are greater than zero.

Since one of our concerns is the quality of the GPS components in cheaper phones, we’re going to have to buy some old, cheap phones aside from any that we can source for free. Luckily, the phones we use are modern so we probably won’t be buying modern Android phones for testing. But, even though older Android phones are usually inexpensive, we need to remember that their cost is also greater than zero.

Technical Challenges

The biggest challenge we’re going to face in successfully implementing Landmarked is sourcing our landmark information. Luckily, we have access to some really great API’s that will probably be great. The problem is that there aren’t that many sources, and the project falls apart if we can’t consistently pull good information from these sources. To be successful we need to be pretty accurate, *most, if not all* of the time. If a user points their phone at a landmark, and we fail to deliver the information the user expects to see, it’s likely that user will write our app off as trash, uninstall, and we will have failed at our most basic goal. Regardless of our efforts, if our sources fail, we fail. The data we get from our sources is all or nothing.

Another problem we could encounter is low accuracy or unreliable GPS components. They could be fine, or they could be terrible. Without any experience on the team working with them, we don’t know what successes or failures we’re going to encounter when working with GPS. Luckily, even if the GPS components are inconsistent or inaccurate, we may have a little wiggle room to fudge the readings in our algorithm. Even if the GPS data we record is less than ideal in terms of accuracy and consistency, we can probably add some tolerances to our calculations to deliver perfectly acceptable results without the user ever knowing there was an issue.

Maintenance

Core functionality

Stretch functionality