Table of contents

[**Introduction**](#_ipyj9hkx93ud)**……………………………………………………………………... 1**

[**Project Description**](#_3ad9eqs5m6xl)**…………………………………………………………….. 1**

[**Project Functionality**](#_4agw3364t5y3)**…………………………………………………………... 2**

[**Core functionality**](#_5esnseqa753b)**………………………………………………………………. 2**

[**Stretch Functionality**](#_o7x02g3un8cj)**…………………………………………………………... 3**

[**Intended Users**](#_n6gii8ilb03x)**………………………………………………………………….. 5**

[**Platforms**](#_4znb0mt4ug4)**……………………………………………………………………….... 6**

[**Expected Costs**](#_6guq6xo4bin2)**…………………………………………………………………. 6**

[**Technical Challenges**](#_aw697dn6zcq3)**………………………………………………………….. 7**

[**Maintenance**](#_6qm838y6lg0u)**…………………………………………………………………….. 9**

[**Product Backlog Items Priority**](#_8ezh4uffxpyw)**…………………………………………….. 10**

[**Stretch Goal Product Backlog Items Priority**](#_tm3ptifr07in)**……………………………. 11**

# 

# **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 application 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 as anything of note 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, parks, and various other points of interest. We intend to provide information on smaller scale landmarks as well, but this will depend on the quality of our sources.

In order to discern between landmarks near and far, we will implement an “aiming” feature. Some mobile screen UI will assist the user, likely with some augmented reality to assist the user and to help dial them in. 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 and identify the landmark in question if any exists. The algorithm specifics are not yet defined, but it will be based off the location and direction from GPS.

We intend to store a user’s historical landmark information for future retrieval. We’ve got numerous features that we plan to implement, and many of them will rely on some stored data of a user’s landmark history. More technical details will follow in this document, but it’s important to note that the ability to store landmark data is an

important feature. It opens the door for many of the extra features we’re considering implementing.

# **Project Functionality**

Our project has a very specific task to accomplish, and it must be implemented well if we want users to continue using our app. With that being said, we have a wide range of additional features that need to be fleshed out into core functionality or stretch goals. We’ve attempted to eliminate “grab bag” features that don’t serve a useful purpose to our users, but we still have not completely decided where to draw the line between strictly core and stretch functionality. On a positive note, we’ve got tremendous stretch potential for this project. Most of these features are small, and won’t weaken the core functionality if they don’t get implemented. However, if we have the development time to devote to stretch goals, we believe our app will provide a very satisfying user experience.

# **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

e) Allow the user to add text notes or details about a landmark they’ve visited

A) 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. Besides just the name of the landmark, the app will also give a brief description of the landmark. This may include, elevation, height, distance from user, and a short history of the landmark. It makes sense to give users the option to opt out of even a brief description, but we feel strongly that if a user cares enough about a landmark to toggle it, they’re probably interested in a bit of information.

B) By saving user’s history, we allow them to return to the landmarks they have been to. The history will be stored on a database through an internet connection. This allows users to automatically retrieve their visited landmarks at anytime. 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.

C) 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. This would be best done locally as it is up to the user to have the space to store their landmarks. The basic information stored would include GPS coordinates (latitude, longitude), and elevation.

D) 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.

E) We want the user to be able to associate *private* notes, reminders, details or any other text with landmarks they’ve visited. There may be details that are relevant to the trip that need to be recorded so that they aren’t lost. For example, if a user visits a national park, they might need to remember that the parking can only be paid for with coins. They might have visited several sites in one day, and want to record memories from each site before they’re forgotten. There are many good reasons to allow the user to add text notes to a landmark they’ve visited. If adding personal text to a landmark is too awkward or obtrusive, this could very easily be implemented as an opt-in feature. What we want to avoid is a user having to use a seperate app, or some other form of recording things when we can easily implement it ourselves.

# **Stretch Functionality**

a) Providing historical or relevant information about a landmark

b) A user ranking system for landmarks

c) Popularity of landmarks, number of people visited

d) Expand scope of landmarks to include more variety

e) AR Integration

f) Social media integration

g) Optimized path for tourists in a new city

A) 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.

B) 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.

C) By having the popularity of landmarks tracked, it allows users two options. First, to discover popular landmarks around them which is best for users in unfamiliar territory. Second, for users who enjoy the peace and quiet, to find a more secluded landmark to explore and enjoy.

D) Earlier, we defined landmarks as locations of natural phenomenon. This could be expanded to include man-made structures such as the Eiffel Tower. Because of the subjective nature of landmarks, they are everywhere. In our cities, in our forests, everywhere humans go we leave landmarks. Many of these landmarks are more common or accessible to people in urban areas that can’t easily get to more remote natural phenomenon. By expanding the scope of our supported landmarks, we can be more inclusive to those that don’t have the luxury of wilderness ventures.

E) AR Integration can help us take our UI to the next level. By including some reference markers on the screen as a user aims the phone, we can simplify the process of honing in on a particular landmark. This will be especially useful in situations where landmarks are located very close to each other. We have no specific UI plans yet, but to visualize what we mean here, picture a modern car that comes equipped with with a back up camera. The AR overlay is generally quite simple, but extremely effective at helping you get within inches of danger. Ours could also be quite simple but give our users a very powerful visual aid. Let’s face it: if it’s hard for a user to toggle a landmark, they aren’t going to use our app for very long.

The second application of AR we’re considering is a view identical to the one you would see when running your camera app, except the overlay would consist of visual representations of nearby surrounding landmarks. When standing at the base of a mountain there might not be a practical reason for this feature, but if a user is trying to reach out beyond their vision, this feature could give another great visual representation of what’s around them. This feature isn’t necessary, but we feel that it help users associate our app with a professional grade app.

F) We’re not creating a social media app. However, if we allow our users an easy way to create social media content based on our landmark data, we will expose our app to anyone in the user’s circle to our app with the endorsement of the user. If we decide to move forward with real world production, this has potential to be a free, highly effective marketing strategy.

G) Creating optimized routes for landmark discovery in minimal time is a feature that’s likely only going to be implemented if we are able to really expand our scope of landmarks. Tourists could deploy a Dijkstra’s algorithm style feature to help them plan a route that will lead them on a highly efficient path of discovery in a small area. Even with our core landmark scope, this feature could be useful in situations where many natural landmarks are packed in tightly.

# **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.

These are broad categories of users, but we’ve intentionally planned our user cases to be broad. Anyone who’s travelled on the highway has looked at some landmark and said “Wow! That’s incredible! What is it?”, whether they’re avid adventurers or not. The fact that you’re not out every weekend exploring new and interesting areas doesn’t mean that you don’t have a legitimate reason to be interested in our app, and we want to keep it that way.

# **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 two seperate apps, one for Android and one for iOS. To avoid making two mediocre apps, we’ll focus our resources on creating one *great* app, and our progression will determine whether or not we explore an iOS app. Amazon Web Services look promising for potentially hosting our services. At this time we’re doing our best to avoid locking ourselves into any specific tech. Still, the list of viable cloud services isn’t that long and amazon seems like a strong contender. Another stretch goal involves integrating our app with desktop software, and that platform would likely be limited to windows. Now that we’ve got a better idea of what we’re trying to accomplish we’ve been researching various development platforms. They’re all going to have strengths and weaknesses, and we’ve got firsthand details from last year’s project teams on this topic (Xamarin anyone?). We’re focused on avoiding falling into a trap where we’ve prioritized a development platform over the project. After we bring our research to light, we’ll pick the platform that we feel is best for our needs.

# 

# **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. Even at just a few dollars apiece, we’ll probably rack up quite a few dollars this way in the interest of being thorough in development and debugging for a wide range of phone models.

Our Git repository is also a cost to be considered. Currently it’s being provided by a team member, but should that change we’ll need to account for the monthly subscription fee in order to keep Git, or consider moving to TFS to avoid the cost.

Alongside hosting our servers on the cloud, we’ll probably maintain complete backups of our work on our local machines. Since some cloud services charge based on the amount of data used, we may do some testing on our own machines. Ultimately we’ll need to produce a product that runs completely on the cloud, but we can avoid excessive data cost by handling some testing and debugging locally. The cost to run our own server machines is low, but again, greater than zero.

We’re also anticipating some small budget for any incidental costs we incur along our project progression. These types of incidentals are mostly related to pay tiers of software that will help us develop and communicate more efficiently. We have at our disposal many free options for software, so we’re not necessarily planning on using anything that requires payment. Still, we’ve talked about it and prepared ourselves for the possibility that pay tier software maybe something we consider down the road if we find that our free options are lacking. Because everyone’s budgets are unique and personal, we’ll consider this option really only in a situation where we *need* it.

# **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. The most best UI and additional features in the world won’t be worth anything if we screw this up. The scariest part about this challenge is that if our sources don’t end up doing what we need them to, we have nowhere to turn. The list of API’s we can call sources isn’t that long. Alot’s riding on being able to pull meaningful information from *somewhere.*

It is likely that the application will require some kind of internet connection to be able to access information about landmarks. In times when phones are out of service, it might not be possible to gather the information about the landmark which renders the app useless. A way around this issue would be to locally store information about landmarks in the vicinity as it is unlikely a user will quickly move large distances without going into an area with service. Likewise, storing the history of landmarks in the database will also require a connection. A solution would be to locally store the information in a queue until a connection is restored where the queue would then be uploaded to the database.

Another problem we could encounter are inaccurate or unreliable GPS components. They could be fine, or they could be terrible. We’re planning on having to deal with this particularly in the cheap phone market. Without any experience on the team working in this area, we don’t know what successes or failures we’re going to encounter when 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. If we find that no GPS components are that trustworthy, we can compensate by narrowing the scope of our landmark definitions. Obviously this would be a blow to the overall mission of our project, but unlike the issue with relying on our sources we can still do *something* with bad GPS data.

Differentiating between multiple landmarks that are located close to one another will be a huge problem we need to solve.. Users are inevitably going to encounter landmarks that are located side by side, lined up in the direct the phone is pointing, or otherwise close enough that we have to figure out how to tell them apart and *figure out which one the user actually wants.* The good news here is that unlike the sources and GPS problems, we can do something about this problem. We may have setbacks or lose time developing the algorithms we’ll be using, but we’ll be relying on ourselves vs. relying on components and sources we can’t change. At worst this issue will likely cost us time instead of becoming a barrier to successfully completing our app.

Another challenge will affect all of us: none of us have developed Android apps before. None of us have worked with GPS components. None of us have experience working from the cloud. The API’s we plan to use are new to us, and so are databases. No matter what technology we decide to use, there’s a good chance we haven’t ever used it before. We look forward to this challenge; we’ve even spoken about specifically choosing platforms we know nothing about. It’s going to be a steep learning curve to try and tackle numerous development tools that are new to us. This includes the design process we will be following. We have some experience in this area, but for a first time, real world project, it will be trial by fire in how well we adapt.

# **Maintenance**

Our long term maintenance plan is going to depend completely on the quality of our finished product. In an ideal scenario, we overcome our challenges and Landmarked ends up being something we’re really excited about. We wrap up our Junior Project Sequence and shift towards real world production, and users fall in love with our app.

Unfortunately, it's too early in the game for us to know how well we will adapt to problems. Right now we have a clear vision and specific goals, but we don't know if our end product is going to reflect them. We don’t know how many of our stretch goals we’re going to implement, and we don’t know even if our core functionality implementation is going to be practical for real world use.

One concern we have is keeping up to date with any changes in the underlying API we will use to gather map data as they will affect the core functionality of our application.

We’ve worked hard to break down as many aspects of our project into tangible pieces and timelines as possible, but unfortunately long term maintenance is going to be put off until we’ve got an app we can experiment with. We will have to determine whether Landmarked is something we're interested in continuing at the end of our project.

If we decide we want to continue production, were going to have to decide *how* we're going to delegate and manage maintenance. With so many unknown factors that will only become known in time it doesn’t make sense to dedicate time or resources now to planning maintenance. We've decided to focus our efforts on making the best app we can. When were done, if we've achieved our goals, we can reward ourselves by figuring out how to prepare Landmarked for real world production.

# **Product Backlog Items Priority**

A) Generate landmark information from phone orientation

1. Find way to access gyro/compass information
2. Access latitude/longitude coordinates
3. Create algorithm to gauge distance from landmark
4. Test algorithm to ensure accuracy
5. Interface with API’s to pull information about landmark

B) Store local data on phone

1. Design data structure to store landmarks
2. Implement data structure for storage
3. Store data

C) Store data in database

1. Design database
2. Implement database
3. Implement interface for database
4. Test database

D) Create UI for application

1. Design GUI for user
2. Implement GUI
3. Test GUI

E) Create custom landmarks

1. Mark point of interest from user’s location
2. Store information into database
3. Test custom landmark retrieval
4. Allow users to input information about personal landmark
5. Add GUI screen that lists all personal landmarks

F) Add landmark notes

1. Design note storage
2. Implement note section on GUI
3. Test note upload and retrieval

# **Stretch Goal Product Backlog Items Priority**

A) Providing historical or relevant information about a landmark

1. Explore API to pull historical/relevant information
2. Integrate API into existing program

B) Expand scope of landmarks to include more variety

1. Reevaluate definition of landmark
2. Integrate new landmark information with existing data

C) Popularity of landmarks, number of people visited

1. Design popularity system
2. Implement system into project
3. Implement interface for GUI
4. Test popularity system

D) A user ranking system for landmarks

1. Design ranking system
2. Integrate ranking system into project
3. Implement ranking system into GUI
4. Test ranking system

E) Social media integration

1. Explore API’s from big name social media
2. Integrate user’s social media accounts to Landmarked through these API’s
3. At users request, generate compact content that we push to user’s media pages

F) AR Integration

1. Draw rough mockups of what we want the user to see on screen in aim mode
2. Explore API’s that will allow us to implement our design for aim mode
3. Explore API’s that will allow us to render semi realistic representations of landmarks
4. Populate the user’s “camera view” with 3D renderings of landmarks within distance user is aiming for

G) Optimized path for tourists in a new city

1. Generate local points of interest within x units of distance from user
2. Ask for approval from user of generated points
3. Implement Dijkstra’s algorithm or similar shortest path algorithm on points
4. Present generated path to user for final approval
5. Explore API to give user turn by turn navigation to next point in path