Wallie - Location Based Credit Cards Offers

Epics and Features Estimation

1. Technologies Used

1.1 iOS Development

Our app initially will be released for Apple iOS. The iOS developer team at Wallie will use Xcode to develop iOS code.

- **Build vs Buy :** Our plan is to develop all the iOS code in house only. We do not plan to outsource any iOS related work.
- **Team:** We plan to have 2 iOS developers initially to start with. We will hire developers who are well versed with iOS development(Swift programming language specifically)
- **Technology Cost:** Using Apple Xcode is free. IDE (Developer tools like Github) environments will incur some cost and this cost is provided in the cost spreadsheet.

1.2 Android Development

After developing iOS app as part of our first release R1, we will move on to developing Android App which will be part of release R2. The Android development team at Wallie will use Android studio to develop the application.

- **Build vs Buy :** Our plan is to develop all the Android code in house only. We do not plan to outsource any Android related work.
- **Team:** We plan to hire 2 android developers initially to start with. We will hire developers who are well versed with Android Development(Java programming language specifically).
- **Technology Cost:** Using Android studio is free. IDE (Developer tools like Github) environments will incur some cost and this cost is provided in the cost spreadsheet.

1.3 Testing Infrastructure

- **Build vs Buy:** The engineer responsible for coding the feature will also be responsible for testing that feature. There will not be a separate testing team. Hence we will be doing our own in house testing. Most of the testing infrastructure will be hosted on AWS (Amazon web services).
- **Technology cost:** The cost will be calculated according to AWS pricing which will be discussed in AWS section.

1.4 Design Tools

- **Build vs Buy:** Instead of developing in house design tools, we will be buying design tools from Balsamiq. Using Balsamiq's advanced design tools will help our design team to quickly design Wallie app.
- **Team:** We initially plan to have one UX/UI designer in our Wallie team.
- **Technology cost:** Costs related to using Balsamiq are provided in the cost spreadsheet.

1.5 Technology Infrastructure - Our own Data Center vs Hosting on AWS

The main components needed in our technology infrastructure will be as follows:

1. Compute - Physical Servers and Virtualization.
2. Security & Networking - Switches, Routers and Firewalls.
3. Storage - Storage servers.
4. Backend technologies (including Database systems)

- **Build vs Buy:** If we were to build our own Data center, then we need to buy our own Servers from vendors like HP or IBM and we need to buy Server Virtualization software from Vmware or Citrix. Similarly, we will need to evaluate Networking vendors like Cisco, Arista and Juniper to buy routers and switches for our data center. When it comes to Storage, we need to evaluate vendors like EMC and NetApp. We believe it will take lot of time for us to evaluate which vendor to go with for our compute, networking and storage requirements. Even if we take time to evaluate vendors, we will need to hire a small Data center team to start with to manage compute, networking and storage resources in our Data center. **We believe going ahead with Amazon web services** (**AWS)** is a better option for us when we are a start up. So, we'll go ahead and host our application on AWS. AWS will also provide all the required backend technologies also.
- **Team**: We will have one Devops Engineer to start with who will take care of hosting our application on AWS.
- **Technology Cost:** Another incentive for us to go with AWS is that if we build our own data center, then we might end up with lot of unused resources and for these unused resources we would have already paid the cost upfront. With AWS, we don't need to pay any money for unused resources as AWS is based on elasticity concept which means AWS will increase or decrease the resources for our app based on the user requests so that we don't pay for unused resources. While we have also looked at Google cloud and Microsoft Azure as other potential cloud hosting providers, we have decided to go with AWS because at this point of time AWS provides much more comprehensive set of hosting services than any other hosting provider. Detailed AWS costing is provided in cost Spreadsheet.

1.6 Mobile Marketing tools like Push Notification and A/B testing tools:

There are vendors like Leanplum who provide Mobile Marketing tools for push notifications and A/B testing. Apple iOS and Google Android platforms have some free built in tools that can be used for push notifications. Similarly, Apple iOS and Google Android platforms have free A/B testing tools. We plan to make use of these free tools. Based on our research, these free tools seem to scale well. So, we will use these free mobile marketing tools.

2 Engineering Organization

We plan to hire engineers for the following roles

Engineering Roles	**Number of Engineers**
iOS Engineer	2
Full Stack Engineer (includes Database expertise)	3
Android Engineer	2
DevOps Engineer	1
UI (Design) Engineer	1

3 Release and Version Considerations

3.1 iOS Release

We plan to work on iOS release first. Our estimate is to take 10 months (Jan 18 to Oct 18) to work on this and release to the users. This time period of 10 months includes testing and integration. Once the first iOS version is released, our engineering team will continue to work on future iOS releases. This will include following:

1) Future iOS releases with bug fixes
2) Future iOS releases incorporating user feedback
3) Future iOS releases improving security, scalability and performance
4) Future iOS releases with new features

All the above releases will be provided in the form on updates to the users. The users can update the app and get the new iOS release. We do not have any plans to release this app for ipad.

3.2 Android Release

Once the iOS release is done, our engineering team plans to work on Android release. Our estimate is to take 7 months (Nov 18 to May 19) and release to the users. This time period of 7 months includes testing and integration. Once the first Android version is released, our engineering team will continue to work on future Android releases. This will include the following:

1) Future Android releases with bug fixes
2) Future Android releases incorporating user feedback
3) Future Android releases improving security, scalability and performance
4) Future Android releases with new features.

All the above releases will be provided in the form of updates to the users. The users can update the app and get the new Android release. We do not have any plans to release this app for Android tablets.

3.3 Release plan for the next 21 months

We plan to have the following major releases in the next 21 months as explained below.

1) R1 - V1 for iOS - This is the first major release for Apple iOS. Target release date : Oct 2018
2) R2 - V1 for Android - This is the first major release for Android. Target release date : May 2019
3) R2 - V2 for iOS - This the second major release for iOS with feature enhancements. Target release date: May 2019
4) R3 - V3 for iOS - This the third major release for iOS with feature enhancements. Target release date: Sep 2019
5) R3 - V2 for Android - This is the second major release for Android with feature enhancements. Target release date: Sep 2019

After we release the first major iOS release (R1-V1 for iOS), we plan to reuse major chunk of backend code for Android. After releasing the first major iOS release in Oct'18, we will also hire Android Engineers. At this point of time, we will have both iOS and Android Engineers in our engineering team. Hence, we plan to do simultaneous development of both iOS and Android platforms.

In Between the major releases, our engineering team will continue to work on intermediate releases to fix bugs, improve performance and scalability. These will be released as updates.

4 Epics and Feature Estimation

We have divided our app’s functionality into different epics and under each epic there will be features that will be implemented. Each feature is further divided into individual components and engineering estimate in terms of number of days has been estimated individually for each component. A group of features in turn completes an Epic.

In this section, we have estimated the time needed for each individual component under every feature. But work for some of the components will happen parallely which means the actual number of days needed to implement the complete feature is actually less than the number of days needed to individually implement the components in that feature. The actual number of days needed to implement a feature is provided in Road map document. This section estimates the number of days needed to implement each component to implement a feature. For each feature, we have also discussed about potential blockers.

4.1 Epic: User Account Management

This includes initial setup of the user account and signing in. The user can sign up using Facebook or Email/Password. Once the user signs up, user needs to add cards. The user can choose the bank and card types from the lists the system has. The user is then asked to setup his/her preferences about location, notification and deals. The notifications preferences include user's choices for frequency of the push notifications and if she/he wants to get aggregated notification or individual store notification. The location preferences include if he/she is okay to share the location always or only while using the application and within what distance of store/place he/she wants to receive notifications. Similarly, the user can sign in when returning to the app and can modify the cards and the preferences.

Feature	Priority	Release
1-1 Account Management		
- iOS	P1	R1 - V1
- Android	P1	R2 - V1
1-2 Card Management		
- iOS	P1	R1 - V1
- Android	P1	R2 - V1
1-3 Referral System		
- iOS	P1	R2 - V2
- Android	P2	R2 - V1

Feature 1-1: Account Management

- Login System - Sign-up, Sign-in, Facebook Authentication, Forgot Password?, Change Password, Log Out, Remember Me
- Account Settings and Preferences for notifications, location
- Contact Support - Push a request to contact support
- Feedback - Give some feedback for the application or report a bug

Components	Estimation (Days)	Blockers

A. User System (Backend)	30	
B. iOS Development	25	Epic 4.1 It requires integration with user system, hence backend is a dependency
C. Android Development	25	Epic 4.1 It requires integration with user system, hence backend is a dependency

The backend estimation includes the database, integration with database, implementing the business logic of the system though APIs. iOS or Android development includes designing the screens, getting the fields from the user inputs, getting data from backend through APIs and saving the data to backend. It also includes setting up the Facebook login SDK.

Feature 1-2: Card Management

- Card Details - Add, Edit and Delete card information from the system. This only includes the bank and type of the card.

Components	Estimation (Days)	Blockers
A. User System	5	
B. iOS Development	5	Epic 4.1 It requires integration with user system, hence backend is a dependency
C. Android Development	5	Epic 4.1 It requires integration with user system, hence backend is a dependency

The backend estimation includes the APIs and setting up the database for the cards types we will be offering to the users. iOS or Android development includes the integration with the backend APIs and screens designs.

Feature 1-3: Referral System

- Referral system - The users will be able to provide referral to their friends/family with a referral code to unlock an exclusive deal or offer.

Components	Estimation (Days)	Blockers
A. Referral Engine	10	
B. iOS Development	5	It requires integration with referral engine, hence backend is a dependency
C. Android Development	5	It requires integration with referral engine, hence backend is a dependency

The referral backend engine will be responsible for generating referral codes and APIs to send and receive the requests. iOS or Android development includes screens where the user can request for referral code or enter the code he/she has for referral and access the exclusive deal or offer.

4.2 Epic: Merchant Account Management

The merchants will have a web-based system to manage the deals and offers and publish to the deals system. They will have account set-up and sign-in features. And once they have set up ready, they can add, modify and delete the deals. They will have a system to add the advertisement, and pay using the application.

Feature	Priority	Release
2-1 Account Management	P1	R1 - V1
2-2 Offers Management	P1	R1 - V1
2-3 Ads Platform	P1	R2 - V2

Feature 2-1: Account Management

- Sign-up process and verification System
- Sign-in, Forgot Password?, Change Password, Remember Me, Log Out
- Account Settings such as Name of merchant, location, phone
- Contact Support
- Feedback

Components	Estimation (Days)	Blockers
A. Merchant User System	25	
B. Web Application	25	Merchant user system backend should be ready to to integrated with the user facing web interface

The Merchant user system estimation consists of setting up the database, and the business logics. The web application included the integration with backend Get/Post calls and designing of screens.

Feature 2-2: Offers Management

- Add/Update/Delete deals and offers

- Search Deals and Offers Added

Components	Estimation (Days)	Blockers
A. Offers Engine	25	
B. Web Application	15	Web interfaces requires Offers Engine to be implemented first for integration

Offers Engine includes integration with existing merchants system (if they use something already) and business logic to access database.The estimation for web application consists screen designs and integration with backend service.

Feature 2-3: Ads Platform

- Add/Update/Delete Advertisement and Pricing
- Integration with 3rd party Payment Gateway

Components	Estimation (Days)	Blockers
A. Ads System	30	
B. Web Application	20	Web interfaces requires Ads System to be implemented first for integration
C. 3rd Party Payment Gateway	10	For Payment Gateway, it is required to have a web interface and backend ready; legal and account setup with 3rd party providers

The ads system will include adding advertisement banner, editing the ads and promos for merchants. The web application will provide the platform to support these and show the pricing of advertisements. Payment gateway includes integration with payment system to pay for the ads they have put.

Perceived Technology Risks:

The technical risk involved with this system is only related to 3rd Party Payment Gateway. Its integration with the system and making sure about the security aspect of the platform.

4.3 Epic: Deals in Map View

This includes in map mode, how can the user find nearby retail stores and the deals they have according to the user information. The user can enter the map page (which is also the default page after login and initial setup as a new user) and the map will locate where the user is (first get permission for location) and find from vendor database the retail stores nearby. The user can quickly check how far a particular store is, what the accurate address is and the best deal in this store (credit card deal). If the user need to compare and check more information, s/he can further enter into the detail page to see all the deals suitable for this store. If the user then want to change preference, s/he can also change and save here. If s/he have a new card available for compare, s/he can also set here. If the user want to check a particular store which is not nearby, s/he can search from the map to locate the place, and check then compare deals . After the user knows where s/he wants to go, s/he can choose navigation option which will invoke Google map in which the destination is already be set.

Feature	Priority	Release
3-1 Map View		
- iOS	P1	R1 - V1
- Android	P1	R2 - V1
3-2 Deal Details		
- iOS	P1	R1 - V1
- Android	P1	R2 - V1
3-3 Search		
- iOS	P1	R1 - V1
- Android	P1	R2 - V1
3-4 Navigation		
- iOS	P2	R3 - V3
- Android	P2	R3

Feature 3-1: Map View

- Map - Get permission for accessing nearby, Zoom-in and zoom-out, load nearby shops, Simple summary of distance and address for a shop

- Store List - List of stores ordered by distance, the best deal for each shop according to the preference

Components	Estimation (Days)	Blockers
A. Map Engine (Backend)	20	Epic 4.1 and Epic 4.2 - It depends on User System backend and Merchant System backend
B. iOS Development	15	A
C. Android Development	15	A

The backend estimation includes permission for access, integration with Google map API. The iOS and Android development includes UX/UI design of the screen, Map zoom in and out, getting merchant data from database through APIs for the list, getting user data to match shops.

Feature 3-2: Deal Details

- Store Info - Basic info of the chosen store (photo, address, distance)
- Deal Compare - All the deals the user has that can apply to this store (best as default)
- Add - Add a new card
- Change Preference - Change the comparison preference
- Report - Report invalid deals

Components	Estimation (Days)	Blockers
A. iOS Development	10	Epic 4.1, Epic 4.2 and Epic 5.1 - It depends on User System backend, Merchant System backend and Deal System backend
B. Android Development	10	Epic 4.1, Epic 4.2 and Epic 5.1 - It depends on User System backend, Merchant System backend and Deal System backend

The iOS and Android development includes UX/UI design of the screen, getting store data from database, getting user data to match the shop deals, user fields for adding and saving new cards into database, widgets for changing preferences, report through email, usability improvement for interaction.

Feature 3-3: Search

- Search: Search Field for any shop available

Components	Estimation (Days)	Blockers
A. iOS Development	10	Epic 5.1 - It depends on the Deal Engine backend
B. Android Development	10	Epic 5.1 - It depends on the Deal Engine backend

The iOS and Android development includes UX/UI design of the search field, integration with search engine.

Feature 3-4: Navigation

- Navigation Invoke : Google map App

Components	Estimation (Days)	Blockers
A. iOS Development	2	-
B. Android Development	2	-

The iOS and Android development includes UX/UI design of the navigation widget, integration with merchant database, invoke Google map with the address of a store.

Perceived Technology Risks:

The technical risk involved with map view is the dependency systems. In this epic, we use Google Map API to request for the location of our users, which might cause scalability problem when the amount of users increases. The accuracy of location is another risk that might influence the user experience since we are supposed to provide the accurate recommendation for a user based on user card profile, merchant deal information and the location of the user.

4.4 Epic : Push Notifications

This epic covers location based push notifications. Based on the user location, push notifications will be sent to the user. These location based push notifications will be push messages that will be sent to the user based on his/her preference settings. By location, we mean within one mile radius of the user's current location. This one mile radius is the default setting. The user can change the distance settings and see offers by increasing the radius. The user account management epic will let the user set his/her preferences where the user can set what kind shop offers he/she wants to receive, what kind of credit card offers he/she wants to receive etc. Similarly user account management epic will also have setting preferences to let the user turn on/off the notifications.

Implementing this epic will especially be useful to the user when he/she is at a shopping mall with lots of shops. Based on the user preference settings, this feature will push notification with different offers from relevant shops(based on the preference settings) within the shopping mall.

Feature	Priority	Release
4-1 Location based push notifications		
- iOS	P1	R1 - V1
- Android	P1	R2 - V1

Feature 4-1: Location Based push notifications

Components	Estimation	Blockers
A.iOS Development	20	Epic 4.5 - Deals Database should be ready before this feature can be implemented on iOS.
B.Android Development	20	Epic 4.5 - Deals Database should be ready before this feature can be implemented on Android.

The estimation period mentioned in the above table also factors in any unforeseen issues while integrating with push notifications. It might take developers a few days to get used to the push notification APIs and integrate with our Wallie app. The estimation period also includes testing efforts needed to integrate this feature in our Wallie App.

Perceived Technology Risks:

With respect to push notifications feature we do not expect to hit any serious issues with respect to technology integration. We will be making use of free tools provided by Apple iOS and Google Android to integrate push notification functionality. One thing we might come across is if these free tools from apple and google don't scale well. In case where we cannot scale with the free tools, we need to look at commercial tools like Lean Plum to scale the push notification functionality.

4.5 Epic: Recommendations

Wallie has two types of recommendations for the user - deals recommendations and new credit card recommendations. Deals recommendations can be delivered either when the user searches for specific stores manually or automatically (through push notifications) when they are near the stores. New credit card recommendations will be shown to the user within the app based on their usage/behavior data. Both of these recommendations will rely on Wallie's recommendation engine on the backend.

Feature	Priority	Release
5-1 Deals Recommendation		
- iOS	P1	R1 - V1
- Android	P1	R2 - V1
5-2 Credit Card Recommendation		
- iOS	P1	R2 - V2
- Android	P1	R3 - V2

Feature 5-1: Deals Recommendations

For deals recommendation, we will need user's data and at minimum:

- User's added card types (refer to epic 4.1)
- Their notification preferences (refer to epic 4.1)
- Location data (just the map engine component*; refer to epic 4.3)
- Deals data (just the merchant offers account management*; refer to epic 4.2)

Components	Estimation (Days)	Blockers
A.iOS Development	30	-
B.Wallie's recommendation engine on the backend	40	Epics: 4.1, 4.2* and 4.3*
C.Android Development	30	-

Feature 5-2: Credit Card Recommendations

For credit card recommendations, we will need:

- User’s data (not an exhaustive list)
    1. All credit card types added by the user
    2. User’s location history
    3. Store types when notifications were triggered
    4. Manual store type searches performed by the user
    5. Type of notifications user interacted/responded

- Credit card affiliate data which will have details of reward programs for the credit cards in the affiliate network which includes (not an exhaustive list):
    1. APR
    2. Rewards category structure
    3. Sign-up bonus
    4. Credit score requirement range* (a grade/estimate is fine)

Components	Estimation (Days)	Blockers
A.iOS Development	20	Epic 4.5 - for recommendation engine backend. Credit card database
B.Integration with 3rd party service such as Plaid	5	Legal & Subscription setup with 3rd party provider
C.Android Development	20	-

Perceived Technology Risks:

The deals recommendation engine and credit card recommendation engine, while doesn’t necessarily have any specific technology risks, their effectiveness is dependant on the amount of data we have in our system. This data comes from various stakeholders such as users themselves, stores using our merchant account system, our credit card rewards database. Since these engines would integrate with multiple internal (and in some cases, external) services, a closer collaboration will be necessary to avoid any blockers. Also, since this is the core of Wallie’s functionality, anything less than desired performance would greatly affect our ability to market Wallie.

This Epic includes managing the online offers ecosystem throughout end-to-end shopping experience of a user. This includes providing best offers and discounts to users based on their preferences. The user is first asked to add a web browser extension and enable it to find best deals for their favorite shopping brands. This also asks for the user's permission to always enable the extension to constantly look for the best offers for that user's needs. The user can also set up the account for email notification delivery for exciting offers on their preferred shopping brands.

Feature	Priority	Release
6-1 Browser Extensions	P1	R3 - V1
6-2 Targeted Emails	P2	R3 - V2

Feature 6-1: Browser Extension

- Shopping intent - Capture items viewed, items in cart and items purchased by brands
- Real-time shopping engagement - Offers used, purchasing behavior and pattern
- Push Notification - Live display of benefits for intended purchase

Browser extensions require some key information from other databases:

- User data
    1. User's preferences and account data
    2. Shopping history and patterns by brands and cards
- Deals data
    1. All the current deals recommended to the user
    2. The deals user accessed/showed interest

Components	Estimation (Days)	Blockers
A. Online Offers System (Backend)	20	-
B. Safari Extension	20	Epic 4.6, 4.5 and 4.1 - This requires integration with Online, Deals engine and User Systems database

C. Chrome Extension	20	Epic 4.6, 4.5 and 4.1 - This requires integration with Online, Deals engine and User Systems database
D. Firefox Extension	20	Epic 4.6, 4.5 and 4.1 - This requires integration with Online, Deals engine and User Systems database

The Online Offers System estimate includes developing the database and setting up the business logics for the entire system factoring in the different offers provided by Wallie. The estimates for each of the browser extensions consists of integration with Online Offers system, Deals engine and User Systems database. It includes getting user account, preference, active deals data from the backend, saving user search and real-time shopping behavior patterns into the database along with live push-notifications per users preferences. This also factors in the APIs to request and verify user choice to always enable the browser extensions.

Feature 6-2: Email Notifications

- Devices - The type of devices used by the user to browse and shop
- Online Shopping Offers - The type of offer and categories
- Offer details: Discounts, gift cards, time period

For developing targeted email notifications, we will need:

- User data
    1. User’s preferences and account data
    2. Device used by user for current browsing request
    3. User’s location and search history
    4. Frequency of shopping for specific items/brands
- Deals data
    1. Hit ratio of deals recommended to the user

Components	Estimation (Days)	Blockers
A. Email Notification System	25	Epic 4.6 and Epic 4.1 - This requires integration with Email Notification system, Online Offers systems, and User Systems database

The Email Notification System includes setting up the database and the business logics. Template customization consists of integration with Email Notification system, Online Offers systems and User Systems database. It also involves creating APIs and screen designs for different types of devices.

Perceived Technology Risks:

The browser extensions run the risk of facing technical compatibility challenges due to changes in parent browser standards and other usage terms for those browsers.

4.7 Epic: Website Creation

Website for Wallie’s launch, support and marketing. The website needs to be responsive and work across all device sizes and across major browsers such as Chrome, Firefox and Safari.

Feature	Priority	Blockers
7-1 Browser	P1	

Feature 7-1: Browser

Components	Estimation (Days)	Blockers
A. Chrome	5	-
B. Firefox	5	-

5. Non-functional Implementation Considerations

5.1 Performance

- Response Time

  There are several scenarios that we have to consider with respect to response time. One of the scenarios is when the user searches for a particular store. In this scenario, the user is proactively looking for a potential purchase, which is important for the entire user experience which decides retention rate.

  To ensure response time meets the users psychological expectation, we have to pay attention to the integration quality with Google maps API and find alternative prevention solutions such as adding a cache layer in case Google map service is down or the user is in a bad internet environment.

- Resource Utilization on User's phones.

  We need to consider that we are not consuming too much memory in the user's phone while ensuring our app interoperates well with the host operating systems Android and iOS.

  We also need to make sure that the release size of the periodic updates we provide to the app is not huge so that the user can update the app easily and quickly.

- Availability

  We also have to consider the availability of the App, that is, to make sure it is reliable with an uptime of close to 100%. In case, the App hits an exception and becomes unresponsive, we should make sure the user can restart the App and is able to use again.

  As we are hosting on AWS, distributed DNS and caching will be taken care of by AWS which means that our app can be reliable with an uptime of close to 100%.

- Scalability

  Our business has a large range of users who have the needs to look for best deals. Multiple users can search and locate at the same time. And as the business grows, the user base will become bigger as well. We have to consider how to scale the business without affecting the user experience. We have to ensure that a large amount of users can access different functionalities within the App simultaneously.

As we will be hosting our App on AWS, we expect AWS to scale based on the number of user requests to our app. AWS is known to elastically grow compute and storage resources based on the number of requests.

5.2 Accuracy and Precision

As a location-based App, our primary value proposition is to provide our users the accurate deal information they need when they shop. The accurate detection of current location is extremely important for the push notification quality. In the initial phase we choose to use Google map API as our primary consideration for external integration due to its good reputation of accuracy. Any other APIs that can provide better experience in the future business will fall into our consideration as we explore more.

Another point with respect to accuracy is that all the deals that we show to the user should be updated deals and not old deals. For this, we need to make sure that we constantly get the updated deals from the stores and merchants. We need to make sure that all the stores and merchants update the deals database regularly (in our mechant system) so that we always have the latest and updated deals.

5.3 Security

As an App that deals with user financial information, we have to consider the user’s sensitive information as No.1 priority. Unlike other competitors, we don’t ask for the actual number and other transaction-related information, but only card types. Proper encryption will be implemented with regard to login. We will also follow the latest standards and meet all the security compliances to secure user information. Private key for database will be changed regularly.As of now, we plan to use 256 bit encryption which is the most advanced method used in modern encryption algorithms.

We also will convey this image to user that the App is safe to use.

5.4 Usability

We also consider usability as a major factor for our success. Whether the user can have intuitive experience when s/he interact with the App, whether s/he can get proper notification when s/he really needs and whether s/he can safely feel this App can protect sensitive information are our major concerns. We will use quantitative and qualitative techniques to evaluate our usability and improve design and algorithms accordingly.

6 Perceived Risks and Mitigation Strategies

6.1 People risks / Employee risks:

As mentioned in section (Engineering Organization) We plan to hire our own engineering team without any outsourcing. We plan to have 2 iOS developers, 3 full stack developers, 2 Android developers, one Devops engineer and one UI engineer.

As we are a small engineering team, There is always a risk that someone leaves and for an engineering organization of this small size, even one engineer leaving can have a big effect. To mitigate this risk, we plan to make sure that we hire engineers with multiple skill sets so that there is no effect to the work even if one engineer leaves. For example, there is only one UI engineer. If this UI engineer leaves our company for whatever reason, one of our Android engineers can take up the role of UI engineer for some time before we get another new UI engineer. When we hire Android engineer we will make sure that the engineer also some basic UI skills apart from superior Android skills.

As we are a start up, we expect people to play multiple roles in times of emergency. As we are also offering equity (details of which are mentioned in cost document) to the engineers, we do believe that engineers will be willing to wear multiple hats in times of emergency.

6.2 Technology Risks

Some technology risks associated with individual features have been discussed in Epics and estimation section. In this section, we will discuss about broader technology risks that we might encounter while implementing the app.

- **Integration with third party APIs :** We plan to make use of third party APIs like Google Maps API, Apple iOS and Google Android free tools to offer push notifications and do A/B testing. It is always possible that the integration with these third party APIs might fail or might not scale. We need to have a mitigation strategy for this. For example, if free push notification tools from Apple and Google do not scale , we plan to use commercial push notification and A/B testing tools like Leanplum.

- **AWS does not scale or AWS goes down**: It is always possible that AWS does not elastically scale for some reason or AWS goes down. While this is unlikely to happen, there is always a very low probability that this might happen. To mitigate this, we plan to continually assess other cloud hosting providers like Microsoft Azure and Google cloud. If we feel our app is not scaling well in AWS environment, then we will shift it to either Microsoft Azure or Google cloud based on our assessment at that point of time.

6.3 Business Risks

As this is an Engineering estimation document , we do not want discuss too many details about business risks but we will just touch upon the business risks that we might encounter as we develop the app.

- As we rely on stores and merchants to use our merchant system to update all the relevant offers, We need to make sure that our partnership with major stores and merchants is handled carefully. It is always possible that our relationship with a major store or merchant gets strained. We need to make sure that we maintain good relationship with all the stores and merchants.

- We need to make sure we don’t run into any legal issues. We have to make sure that we include all the clauses in our “Terms and Conditions apply statement” that users agree to when they install our app.

- We have estimated the engineering expenses in this doc to this best of our knowledge. We need to make sure that we run our engineering organization within this estimated cost so that we don’t run out of money. One of the major issues startups face is cash crunch where in the startups run out of money as they don’t plan the spending correctly. We have to make sure that we spend the money wisely cutting out unnecessary expenses.