1. Introduction

Preliminary requirements

Our preliminary requirements are sufficient to a prototype of the application. The scope of the project will be limited due to the technical difficulty of the application components. Therefore our preliminary requirements are sufficient to implement a proof of concept application.

Further requirements will be identified once the application is closer to a usable state.

Technical issues

Our project will include many different module, and therefor will be very technically difficult. These components include:

- Implementing a android app.
- Implementing a server.
- Creating (or using) algorithms for image matching.
- Using an API that interfaces with the current ualberta web application framework.

Technical issues will arise due to the many different pieces of the application, and will have to be recognized early to avoid hold-ups late in the project.

Personnel issues

There are currently no technologies that we will be using that we do not have a team member with some experience using. However, this could end up being an issue if one component falls behind and there are not enough group members available with expertise to accelerate that component. Therefore we will us pair programing to ensure that expertise i balanced during the project.

Resources required

We have a few project requirements:

- A server for back-end image recognition.
- Android phones for development and testing

For the server we are using an amazon ec2 ubuntu image. We made this choice so we could easily scale are project.

Dependencies between issues

This project can be split into three components, the web-server, the android app and the image comparison component. Only web-server communicates with the other two. All issues with these parts can be dealt with individually, however integration testing must be done to ensure proper communication.

Risks

In this document we analyse risks that could cause our project to fail. The number one risk is “Scope creep”, we mitigate this risk by clearly defining scope in our

project documentation.

7. **Breakdown of development tasks.**

The development tasks are broken down into increments and int requirements. The breakdown is detailed below, in the "Planned Estimates" section.

2. RISK ASSESSMENT

Risk	Potential Impact	Likelihood	Mitigation Strategy
Scope			
Scope creep	**High:** With a short time frame complete a finished product expanding requirements would impact testing/ project quality and chance of success.	**Low**	We identified scope with our user requirements and submitted them to our clients for review. By creating the project in iterations we are minimizing affect potential increase in scope.
Technology			
lack of experience with technology we are programming with	**Medium:** Miss estimation of time required to complete tasks. Bulky code with poor testing	**High**	Take advantage of the expertise of our clients with the images recognition. Work in pairs, to help avoid running into dead ends and to take advantage of each other's strengths. Test driven development much refactoring
Lack of experience with project tracking tools	**Low:** Lack of experience causes wasted time and frustration	**High**	We opted to use a light weight project tracking to so that we would have less to learn
Repository Failure	**Low**	**Low**	
Personnel Risk			
Weak involvement from group members	**High:** In order for us to complete on time and deliver a well done project we need involvement from all of us	**Low**	Pair programming well help keep us all involved and focussed Creating a good team atmosphere will make people feel committed personally to each other and not want to let the team down
Personality issues	**Medium:** leads to lack of corporation, reduced peer review	**Low**	Our team is committed to egoless programming.

Missing expertise	**Medium:**	**Medium**	Use our support network. We have access to many experts including a grad student who has built a very similar project.
Schedule			
Unrealistic schedule	**Medium:**	**Medium**	By scheduling the most critical section first and using agile practices to determine velocity we can adapt
Client			
Weak user participation	**Medium:** Lack of user participation causes poor prioritizing and possibly late changes in requirements	**Medium**	Keep the user involved by frequently looking for user support and keeping the user updated of our progress
Client feedback not representative of the end user	**Medium:**	**Medium**	We have made connections with the U of A web development team to get end user feedback

3. Project Structure

The Mr Bear project is supposed to deliver a high quality finished app, however the primary concern is to explore the use of image matching, in the application of building recognition on campus. The results of this project will be used in possible future projects, which will expand image recognition to other types of objects. In order to complete best project possible, we are creating light weight agile structure.

1. Process model

An agile process model best fits the challenges of our project. We are committed to continuous collaboration, refactoring and integration. To achieve this we will:

- Use pair programming
- Not type cast people into one role so that they do not understand how the pieces must be integrated.
- Use full software development model in every iteration
- Meet face to face often to update each other on what we have recently completed
- Complete unit testing for all parts of our project

2. Team management and anticipated roles

The ego-less programming model is well fitted for complex tasks. We will use this management model to ensure high quality code as a result of peer review. Our belief is that self-organizing teams will keep our team agile capable of adapting to particular

challenges, while minimizing the overhead and division cause by a top down team lead. Certain roles will naturally exist but we do not believe that assigning titles is not the solution. As hard working students we all want to be productive and successful so naturally we will play to our strengths.

3. the project-monitoring mechanisms

Our progress will be measured by the creation of working and tested software units. This will be monitored by velocity, burn down charts and time lines. We are using Trac on xp-dev.com which provides easy to use agile monitoring tools.

4. Major Scheduled Items

In order to reduce risk we have ordered our iteration based on technical complexity and perceived difficulty.

1. Iterations

Increment 1 - Image matcher

A system that matches a photo taken by user on a smart-phone and matches it to high definition photos of campus. The software will provide a ranking of photos based on how well they match. At this stage a low priority will be placed on usability, and GUI will not be detailed.

Increment 2 - Building Identifier

The building identifier will use an image to attempt to identify which building the image is of. A set of possible buildings is sent to the user, along with information about the buildings. This will also require the creation of an API for managing the building profiles on the server side. Mr. Bear system administrators should be able to access this API through a personal computer GUI, possibly a web app.

Increment 3 - Mr. Bear University Guide

Fully integrated end user software, this software will allow users to find their way to any building on campus using all of the features in the previous increments. A user can search for a building by name or photograph, and use the system to help them locate where they are and guide them to where they are going. In this stage the GUI will be polished, creating a viable mobile app.

Increment 4 - Augmented Reality

Include feature to augment reality by identifying buildings while using the camera in Mr. Bear. Buildings in the viewer will have markers on them that display information about the building

2. Planned estimates

Activity	**Planned/ Estimated**			**Actual**		
	Start Date	***End Date***	***Effort (person hours)***	***Start Date***	***End Date***	***Effort (person hours***
Iteration 1	Feb. 11	Feb.27	120h			
Implement image matching algorithms using ***open CV***	Feb.14	Feb.22	20 - 40h ???			
Deploy an EC2 server	Feb.11	Feb.19	20h			
Shrink and format the images to be used with our image matching	Feb.11	Feb.19	4h			
Complete naive testing on image recognition on images without sorting based on geo-tags	Feb.20	Feb.27	4h			
Create a web server using the python ***twisted matrix*** library that can communicate with a basic android app	Feb.20	Feb.27	20h			
Develop a unit test framework for the web server using the ***Trial*** library	Feb.20	Feb.27	10h			
Make a prototype android app with image send and receive functionality to test our server and image recognition with	Feb.11	Feb.23	40h			
Iteration 2	Feb.24	Mar.14	150h			
Create a database of images matched with geo-tags and locations						
Create a database query that chooses possible images based on GPS and compass direction						
Build an API for building profiles						
Add functionality or rebuild the android app so that it can send an image and receive multiple building suggestions, which to pick from						
Complete unit testing for the android app						

Evaluate the efficacy of the image recognition tweak algorithms as needed						
Create a learning algorithm so that the image matcher increases its accuracy based on user response						
Use basic prototype to get end user feed -back in the form of a survey						
Iteration 3	Mar.15	Mar.22	50h			
Fully integrated android app						
Complete tests on scalability of the server architecture						
Carry out tests of application with sample end users						
Provide simple scripts for adding building information and images to the server						
Add building text search features to the server side						
Iteration 4	Mar 23	Mar 30	50h			
Complete tests on scalability of the server architecture						
Fix scaling issues						
Add an augmented reality feature so to show case the app						
Increase the usefulness of the app by adding more complicated images recognition						
Complete full testing of all functional requirements						