Due: Monday, February 5, 8:00 AM (submit to BlackBoard, under Assignments)

**Team Name:**

Team 15

**Team Members and email addresses:**

|  |  |  |  |
| --- | --- | --- | --- |
| Yarden Tamir  ytamir18@gmail.com |  |  |  |
| Shawn Parkes  shawndparkes@gmail.com |  |  |  |
| Stephen Fulton  cardman4782000@gmail.com |  |  |  |
| Rebekah Manweiler  b.manweiler0105@gmail.com |  |  |  |
| Alec Knutsen  aqknutsen@gmail.com |  |  |  |

**Team Meeting time:**

Wednesday 1:40

**Lab Meeting time:**

Wednesday 1:40

**Contact:**

Yarden Tamir

**Project Description:**

This project takes the audacious task of combing the real and the virtual world. We plan to create a mobile application written for Android to bring the age of mobile gaming into the new augmented reality world. Following in the footsteps of the wildly successful Pokemon Go, we plan to create a mobile application using Google’s new ARcore augmented reality API. The game that we will create will be a scavenger hunt game in which a user or group of users will be able to pick a location or use the one that they are currently at and have their world transformed into a new realm where a winner will be decided based on the type of game that they decided. We are choosing to undertake this task because we see that this is the future of the mobile application landscape and wish to be on the frontier of innovation.

**Project Milestones**

First Semester

* Finalize project idea and complete Initial Project Description **October 2, 2017**
* Complete Project Proposal **October 23, 2017**
  + Documentation
    - Description
    - Workflow
    - Gantt Chart
  + Images
    - Use-Case Diagram
    - Application Design Mock-ups
* Complete Project Proposal and Video **November 10, 2017**
* Software Organization and Design **November 20, 2017**
  + Class Diagrams
  + State Diagrams
  + Edit workflow and work plan
* Development and Testing of Software Capabilities **December 11, 2017**
  + Unity AR and Geolocation services (Google CORE API)
  + Multiplayer connection and communication (Google Play API)
  + Server connection and communication (Google Play API)
  + Object Models and Animations (Unity and Blender)
* Evaluations and End-of-Semester Requirements **December 15, 2017**

Second Semester

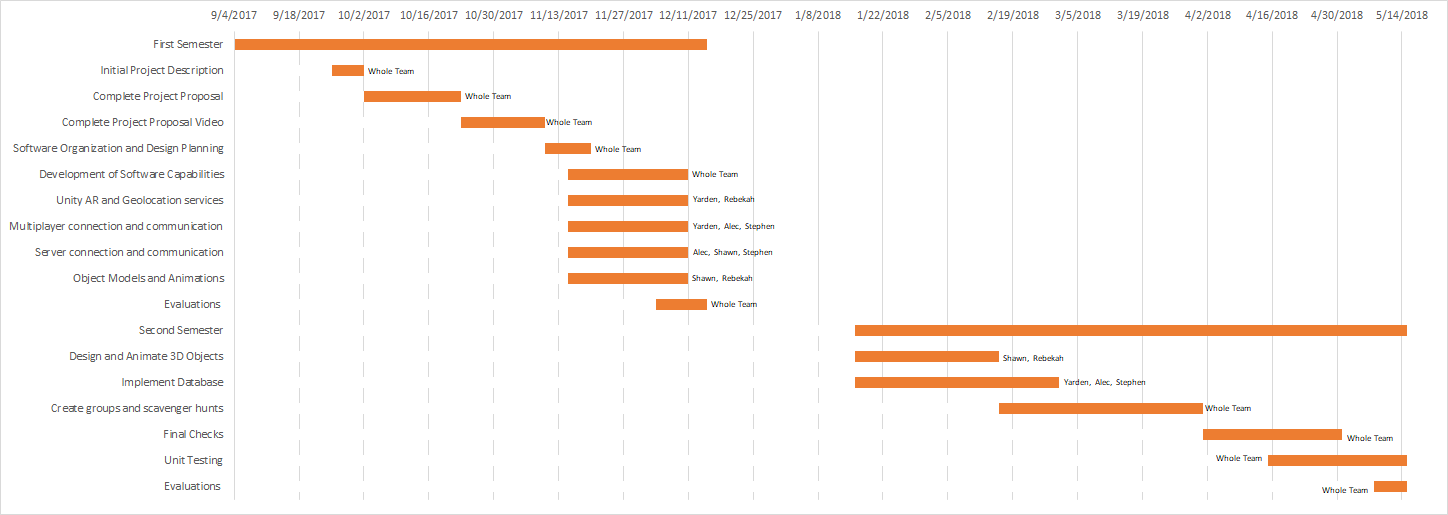
* Designing 3D Objects
  + Design objects **January 28, 2018**
  + Covert objects into transferable form **January 31, 2018**
* Implement Database
  + Create Schema for Data Elements **January 31, 2018**
  + Create database elements **January 31, 2018**
  + Test Database elements **February 5, 2018**
  + Implement Database elements **February 5, 2018**
* Creating groups and scavenger hunts
  + Create networking relationships among devices **March 10, 2018**
  + Implement Google API for interphone networking **March 11, 2018**
  + Configure networks to work for game modes **March 13, 2018**
  + Deploy networking into the application **March 15, 2018**
* Finalize Features
  + Clean code **March 17, 2018**
  + Inspect algorithms and mechanics **March 19, 2018**
  + Debug all issues in the backlog **March 20, 2018**
  + Add any cool or unique features **March 21, 2018**
  + Code addition freeze **March 17, 2018**
* Unit Testing
  + Write Unit Tests **March 31, 2018**
  + Fix issues that arise from unit testing **April 15, 2018**
* Evaluations **April 18, 2018**

**Project Budget**

We will currently host our database services on the EECS department system, though that is subject to change to a cloud based service. Our specific expenses:

* Hosting/Database:
  + Cost: Free (Varies with higher usage)
  + Vendor: Amazon Web Services
  + Time: Throughout the development process
* Software Licenses:
  + Cost: Varies
  + Vendor: Android SDK, Unity (Plugins + Assets), Third party programs, APIs and libraries
  + Time: As needed through development cycle.
* Unity Subscription:
  + Cost: Free ($35/month pro version)
  + Vendor: Unity
  + Time: As needed through development cycle.
* Github Subscription:
  + Cost: $5 per month
  + Vendor: Github
  + Time: As needed through development cycle.

**Gantt Chart:**



**Work Plan:**

* Initial Project Description **Whole Team**
* Complete Project Proposal **Whole Team**
  + Documentation
    - Intellectual and Ethical Issues **Yarden**
    - Software Description **Alec**
    - Gantt Chart **Rebekah**
    - Change log **Whole Team**
  + Images
    - Use-Case Diagram **Shawn**
    - Workflow **Stephen**
    - Application Design Mock-ups **Stephen**
* Complete Project Proposal Video **Whole Team**
* Software Organization and Design Planning **Whole Team**
  + Class Diagrams
  + State Diagrams
  + Edit workflow and work plan
* Development and Testing of Software Capabilities **Whole Team**
  + Unity AR and Geolocation services **Rebekah, Yarden**
  + Multiplayer connection **Stephen, Alec, Shawn**
  + Server connection and communication **Alec, Yarden, Stephen**
* Designing 3D Objects
  + Design objects **Shawn, Rebekah**
  + Covert objects into transferable form **Rebekah, Yarden, Alec**
* Implement Database
  + Create Schema for Data Elements **Yarden, Alec**
  + Create database elements **Yarden, Alec**
  + Test Database elements **Yarden, Alec**
  + Implement Database elements **Whole Team**
* Creating groups and scavenger hunts
  + Create networking relationships among devices **Stephen**
  + Implement Google API for interphone networking **Stephen**
  + Configure networks to work for game modes **Whole Team**
  + Deploy networking into the application **Whole Team**
* Finalize Features
  + Clean code **Yarden, Shawn**
  + Inspect algorithms and mechanics **Alec, Rebekah**
  + Debug all issues in the backlog **Whole Team**
  + Add any cool or unique features **Whole Team**
  + Code addition freeze **Whole Team**
* Evaluations **Whole Team**

**Github link:**

https://github.com/ytamir/ScavengAR

**Final Project Design:**

* How the software works
* 1,000-1,500 words
* 3-5 graphs, charts, or illustrations

This project will involve a mobile application scavenger hunt game that will involve Android, Google APIs, Amazon Web Services, and new augmented reality technology. Besides the Android front-end, the main Google APIs in this project will include Google’s real-time multiplayer API, Google’s service location API, Google’s maps API, and Google’s new AR Core API. Furthermore, on the backend, we will be using Amazon’s Web Services Elastic File System along with SQL.

From an end user’s perspective, the app will feature an initial screen for starting a new scavenger hunt game. The two primary options on this page will be the game size and the number of players in the game. User’s will have the option to choose a small, medium, or large game size, and the appropriate radius from the user’s phone will be selected for placing objects. In the case that a user decides to play multiplayer, this project will use the real-time multiplayer API in google play to handle the invitation of players and multiplayer functionality throughout the game. This API will allow communication between players, a user interface for inviting players to games, and storing game information throughout the duration of the game.

After all invitations are accepted and all users set their status to ready, a random scavenger hunt map will be created based on the phone location of the user that prompted the game. First, this will involve using Google’s service location API to obtain the current location of the phone. Then, using the appropriate radius for the user inputted game size along with Google’s maps API, a circle will be generated from which we can generate a viable set of locations. Using a random latitude, longitude generator, locations for the scavenger hunt objects can be chosen within the given boundary. Furthermore, Google’s maps API can help detect invalid locations, such as a body of water, and replace the location with a valid one. If time permits, we will consider generating locations specific to a particular topic. For example, in this scenario, a user could select coffee shops only, and we could use Google’s maps API to generate a coffee shop scavenger hunt. Once the algorithm selects locations, the timer and game will begin.

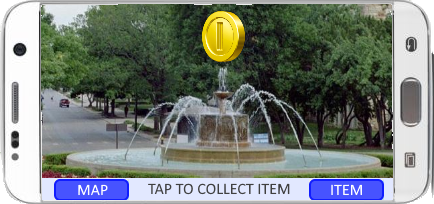
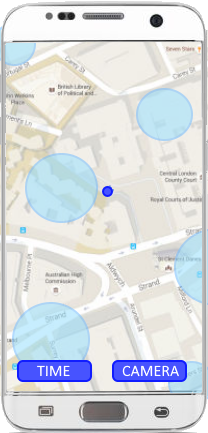
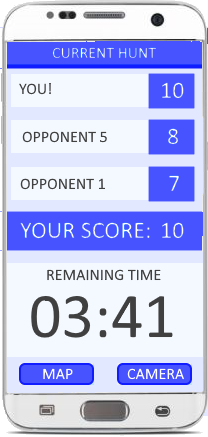
The in-game portion of the application will be the most comprehensive consisting of three main parts: a map, progress screen, and camera screen. Android fragments will allow us to split up the UI into these three main components. The map screen’s purpose will be to guide the players to the objects without making it totally obvious where the objects are located. For each location, a circle will be circumscribed around the object and drawn on the map; Google’s maps API has a circle interface that will work perfect with this design. The size of the circumscribed location will depend on the game size (smaller circle for smaller games and larger circle for larger games). Additionally, the map will display a distance to the nearest scavenger hunt object. Once the user is in the defined mini-map circle, which we can detect through Google’s service location API, we will allow the user to enter camera mode if so desired.

The second UI screen involves the multiplayer aspect of the game. This screen will contain the time left in the game along with the progress of other players on the screen. Again, using the real-time multiplayer API in google play, we will be able to handle communication between players and the display of other’s progress. Each time a payer captures an object, a notification will be sent to all players. Furthermore, each player will be able to see the object’s their opponents have captured at what specific location.

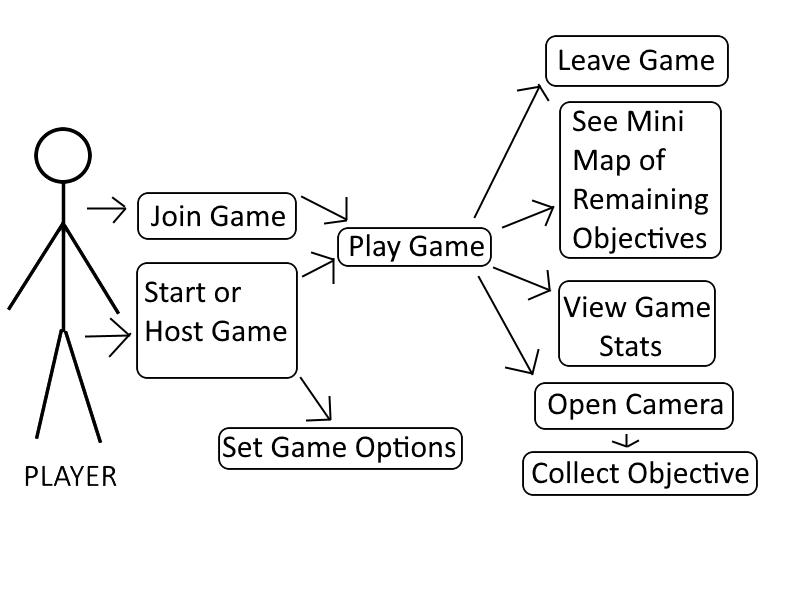
The last portion of the in-game UI involves the phone camera and finding objects. Once the user enters camera mode, we need to place an augmented reality object into the world. Initially, we will have a set of predefined images that will be used for augmented reality objects in our game. The most efficient way to store these images will be in a separate filesystem. Amazon Elastic File System along with Amazon EC2 instances are the perfect tool for this task; they provide file system storage access, and the size of the file system grows elastically with the applications needs. Furthermore, in the future, if we allow users to upload images that they want to use in their games, we can scale our system easily with the Amazon Elastic File System and delete the images after the game with ease. Microsoft SQL will be integrated with our file system to store image paths; thus, once an image needs to be rendered in the real world, we can look up the image path in SQL and retrieve the image from our filesystem. To place the image into the world, we can use Tango augmented reality in the Unity SDK along with Google’s AR Core. Once the users see the object with the camera, they can than tap on the object, and it will disappear from their screen only. Also, a message a notification will be sent to all participants in the game using the real-time multiplayer API.

The end of the game can either occur when the timer runs out, or a user has collected all the objects in the game. At the end of the game, results for each user will be displayed containing the number of objects they found during the hunt. An option for restarting the game will be displayed, allowing any participant in the game to start a new game with the same users, but different locations. We will also collect feedback from the user on the game itself. Feedback will concern whether locations were too close together or too far apart, whether locations were too hard to find or to easy to find, and whether or not the locations were enjoyable. If time permits, we can then implement some machine learning to determine how to improve our location selection algorithm based on this feedback

**Application Mock-Up**



**Use Case Diagram**



**Ethical Issues:**

The primary ethical issue that can arise from our app is one of what is acceptable space for the game to take place. The game spawns objects in random locations around the user. That would mean that if the user was not in a public place then there is a likelihood that the random item spawn will occur over private property that it would be trespassing to go into. This is a legitimate concern because we are very keen on making sure the game is as safe as possible. To work on this issue there is a few potential fixes and ways to improve the experience for all. First we can relay a message at the beginning of a match to make sure you obey all rules and laws of the location and make sure not to trespass on the property of someone else or the roads. Secondly, we can set a time aside before the countdown of the game begins where people despawn items in locations that would not be possible to reach because of any means.

**Intellectual Property Issues:**

The intellectual property issues that we need to be watchful of include the basis of reusing code. For a large part of this project we will be relying on the code of someone else and using their code as a basis of our code. This is fine and protected under fair use and to a larger point even made for our purposes because it is an API to interact with the code and tools available. When we use the API we will need to check the specific rules for fair use and the style guide that the API provides. Additionally, we will need to use objects and images in our game that are either under free use or that we have created. I think the latter is easier and forces us to innovate though we need to make sure that the inspiration we draw from other sources is met with a strict barrier of fair use and not straight copying.

**Change Log :**

* Work plan
  + We have changed some of the things we were going to finish last semester to be this semester as we triaged new requirements.
* Project Milestone
  + We have updated the list to include the full list of tasks we need for completion of the project.