

# **Treasure Hunt Android Game**

San Francisco State University

CSC 667-01 Spring 2012

Dr. Ilmi Yoon

May 22, 2012

Benjamin Britten

Dennis Feng

Terry Chun Wong

Code Repository:

<https://code.google.com/p/sfsu667-sprint2012-android-treasure-hunt/>

## Table of Contents

1	Introduction.....	4
2	Contributions Made by Group Members.....	5
2.1	Britten, Benjamin.....	5
2.2	Feng, Dennis.....	6
2.3	Wong, Terry Chun.....	6
3	Software Development Platforms and Tools.....	7
3.1	Platforms.....	7
3.2	Development Tools.....	7
4	Server UML Diagram.....	8
5	Server Overview and Design.....	9
5.1	SFSUMapServer.java.....	9
5.2	ServerTable.java.....	9
5.3	Process.java.....	9
5.4	PlayerStats.java.....	12
5.5	PlayerLog.java.....	12
5.6	Encrypt.java.....	12
5.7	Authentication.java.....	12
5.8	Base64Decoder.java, Base64Encoder.java, and Base64FormatException.java.....	12
6	Android Activity Design.....	13
6.1	Activity Flow Chart.....	13
6.2	Login Activity.....	14
6.3	Map Activity.....	17
6.4	Tools Activity.....	21
6.5	Target Activity.....	26
7	Icons.....	27
7.1	Game Icons.....	27
7.2	Location Indicator Icons.....	27
7.3	Tool Icons.....	27
8	Future Developments.....	28
9	Project Learning Experiences.....	29
10	Meeting Logs.....	30
10.1	Week 1: Initial Meeting – April 6, 2012.....	30
10.2	Week 2: Meetings – April 9, 13 and 14, 2012.....	31
10.3	Week 3: Meetings – April 20 and 22, 2012.....	33

10.4 Week 4: Meetings – April 25 and 26, 2012.....	35
10.5 Week 5: Meetings – May 4, 2012.....	37

## 1 Introduction

The Treasure Hunt is an Android “find the treasure” game. It utilizes GPS location information gathered by the Android device, an application server, and the Internet in order to give the player a real world experience of physically deducing a clue and locating the treasure. The experience is heightened by having multiple players all competing to reach the treasure first.

This game is using the San Francisco State University as a playable game area. Starting the Treasure Hunt game will bring the player to the “Login” screen. If the player has previously logged in, the player will have been saved and only a “Start” option will be shown. Upon starting the game, the player will be brought to a map showing their current location. Clicking “Check Here” will place the player into the current active game which provides a written clue of where the treasure is located and changes the player's icon based on their distance from the treasure. The icons are represented as: Blue = Cold, Yellow = Warm, Flame = Hot, Treasure Chest = Win. Tools are available for the player to use in order to help them find the treasure or distract other players from locating the treasure first. Once a player has reached the correct location, that player will receive a message showing that they won and the server will begin a new game.

Points are earned by winning a game. A smaller amount of points are awarded when a player checks their location, and they have reached a distance other than “Cold”. The points determine a player's ranking on the leader board. These same points are also used to purchase tools, forcing the player to decide if they wish to lower their ranking, but truly need the tool to aid them in the game.

## 2 Contributions Made by Group Members

### 2.1 *Britten, Benjamin*

- Architecture
  - Original Server Class Structure and game logic design.
  - Client-Server communication protocol specification using JSON in HTTP Request and Response bodies.
- Client
  - Wrote second version of networking class.
  - Added a networking method to LoginActivity.
  - Added a couple of examples related to networking in MapsActivity.
    - Classes (Client) : HttpClient, LoginActivity, MapsActivity.
- Server
  - Wrote second version of processRequest method in Process class to read the HTTP Request and extract the JSON in the body.
  - Wrote second version of writeResponse method in Process class to include a JSON object in the body of the response.
    - Classes (Server) : PlayerStats, Process, ServerTable.
  - Lots of small adjustments during testing to correct game logic.
- Team Organization and Documentation
  - Coordinated group meetings, maintained weekly meeting logs.
  - Original Project Proposal.
  - Created UI mockups.
  - Code Review.
  - Helped with final project documentation.

## **2.2 *Feng, Dennis***

- Client
  - All Android development including GUI, graphics, and networking.
    - Classes(Client) : LoginActivity, MapsActivity, MyMarkerLayer, SelectTargetActivity, ToolList, ToolListAdapter, Tools.
- Architecture
  - Original Server Class Structure, client and game logic design.
- Documentation
  - Revised Final Project Proposal with Game Scenario, In-Game Tools, etc.
  - Put together the final project documentation.

## **2.3 *Wong, Terry Chun***

- Server
  - Wrote all original and almost all final primary server code.
    - Classes(Server) : Authentication, Base64Decoder, Base64Encoder, Base64FormatException, Encrypt, PlayerLog, PlayerStats, Process, SFSUMapServer, ServerTable.
- Design
  - Provide input on game logic.

### 3 Software Development Platforms and Tools

#### 3.1 *Platforms*

- Android API Level 8 (Froyo) - <http://developer.android.com/sdk/android-2.2.html>
- Google MAPS API for Android - <https://developers.google.com/maps/>
- Java 1.6 - <http://www.oracle.com/technetwork/java/javase/overview/index.html>
- GSON library - <http://code.google.com/p/google-gson/>
- SimpleJson library - <http://code.google.com/p/json-simple/>

#### 3.2 *Development Tools*

- Android SDK - <http://developer.android.com/sdk/index.html>
- Android Virtual Device Manager
- Eclipse 3.7 - <http://www.eclipse.org/downloads/>
- Google Plugin for Eclipse - <https://developers.google.com/eclipse/>
- Android Plugin for Eclipse - <http://developer.android.com/sdk/eclipse-adt.html>
- Subversion 1.6.x - <http://subversion.apache.org/>

[illegible]

Page 8 of 38



## 5 Server Overview and Design

### 5.1 *SFSUMapServer.java*

- This class runs the main server function using an infinite loop that listens for incoming socket requests and starting a new thread when a request is received. This class also initializes the ServerTable.

### 5.2 *ServerTable.java*

- This class contains global information related to the game. All the winning geolocations and their necessary information is entered here. As such, ServerTable handles setting a new winning location for a new game and has methods for returning the current winning location.
- This class also maintains the list of current players, and includes methods for reading and writing to a file the player's information.
- This class also maintains the stealer and lockout status, since these two tools are globally set for all players.
- There are numerous other methods related to the above functions as well as other methods for future features (such as dealing damage to other features).

### 5.3 *Process.java*

- This class is where most of the work of the thread takes place. There are a few other methods in the Process class but are not listed here because they are for unsupported features.
- readRequest()
  - The readRequest method reads past the headers of the POST request to get the body and then creates a JSON object from the body.
  - Format of JSON Request :
    - {“playerID”:”Ben”, “currentLocation”:”0.0,0.0”, “option”:”varies”,

```
“password”:”pass”,“tool”:”toolname”, “targetPlayer”:”Dennis”,  
“message”:”tauntMessage”}
```

- JSON uses key-value pairs and depending on the function being performed by the Android client what is included will vary. The key that affects the functionality of the server is “option”. The values for “option” are “signIn”, “getClue”, and “setTool”.
  - readRequest stores any of the values sent for later use by processRequest.
- processRequest()
  - The processRequest method runs different operations based upon the value sent in the JSON for the “option” key. If the option was “signIn” the system loads the player as if they are already a listed player. Otherwise, the system generates a new player in the PlayerLog and creates a new instance of PlayerStats.
  - For option “getClue”, the system checks to see how close to the player is to the winning location, gets the clue for the current game, and sets their indicator. For “getClue”, the setIndicator method is called which determines the indicator. Additionally, the method checks if the player has been targeted by any of the game tools.
  - Finally, the option “setTool” lets the server know the player has purchased a tool. The value for key “tool” will give the toolName. For the *Taunt* tool a value for key “message” will be sent that will override the clue sent to the player. All tools, excluding *Stealer* and *Lockout*, will have a value sent for key “targetPlayer”. In the case of the *Clear Sky* tool, the target player is the purchasing player.
  - Tools *Stealer* and *Lockout* are set in the ServerTable if no existing *Stealer* or *Lockout* are in place.
  - All other tools interact with the PlayerStats objects for the current player and target player.
- writeResponse()
  - The writeResponse method creates an HTTP response to send to the Android client with

a JSON object in the body of the response. The JSON format varies based on what was sent by the client in the request.

- If the option in the request was “signIn” then the response is :
  - {“signIn”:[*Good / Bad*], “playerPoints”:"00000"}
- If the option in the request was “getClue” then the response is :
  - {“clue”:"clue details", “distance”:[*numeric distance from goal*],  
“goalLocation”:(geolocation of goal), “indicator”:[*cold / warm / hot / smoke*],  
“elapsedTime”:[*currently unused*], “playerPoint”:"00000"}
- Finally if the option was “setTool” then the response is :
  - {“status”:[*OK / BAD*], “tool”:"tool name", “distance”:[*numeric distance from goal*], “goalLocation”:[*geolocation of goal*], “elapsedTime”:[*currently unused*],  
“playerPoint”:"00000", “targetPlayer”:[*name of target player*]}
- Currently, in each response, the playerPoints is used by the client to update any changes.
- For option “signIn” and “setTool” the response contains the keys “signIn” and “status”, respectively, that are used as flags for success or failure of processing the command.
- There are a few currently unused values included in the responses.
- setIndicator()
  - This is the last major method in Process. setIndicator is called in the processRequest method, and it determines whether the player has reached the winning location. If not it sets the indicator either *cold*, *warm*, or *hot* based on how close they are to the winning location. setIndicator then checks to see if a *Stealer* or *Lockout* are in place and will adjust accordingly. The method also checks to see if you are the target of a dizzyMonkey or smokescreen and sets your indicator accordingly. Finally, setIndicator is responsible for checking to see if you have stolen a win using the *Stealer* tool.
- writeToLog()

- This updates the files used to store the players' game information.
- `computeDistance()` and `computeElapsedTime()` :
  - Both utility methods used in `Process`.

#### **5.4 *PlayerStats.java***

- This class is for active players. `PlayerStats` maintains information about a player in the game. The method deals with the logic of a tool that has been used by or against a player and keeps track of any tools that have been used against the player.

#### **5.5 *PlayerLog.java***

- This class deals with maintaining a log of the players, their passwords, and their current award total in the file “`htpasswd.log`”.

#### **5.6 *Encrypt.java***

- For encrypting passwords. Currently unimplemented.

#### **5.7 *Authentication.java***

- Checks to see if the username and password matches the stored username and password. Supports encryption.

#### **5.8 *Base64Decoder.java, Base64Encoder.java, and Base64FormatException.java***

- Helper classes dealing with encryption. Currently unused.

## 6 Android Activity Design

Map Activity is the primary activity. From the Map, either the Login activity or the Tools activity can be called. In the Tools activity, the Target activity will be called when the purchased tool requires a target.

### 6.1 Activity Flow Chart

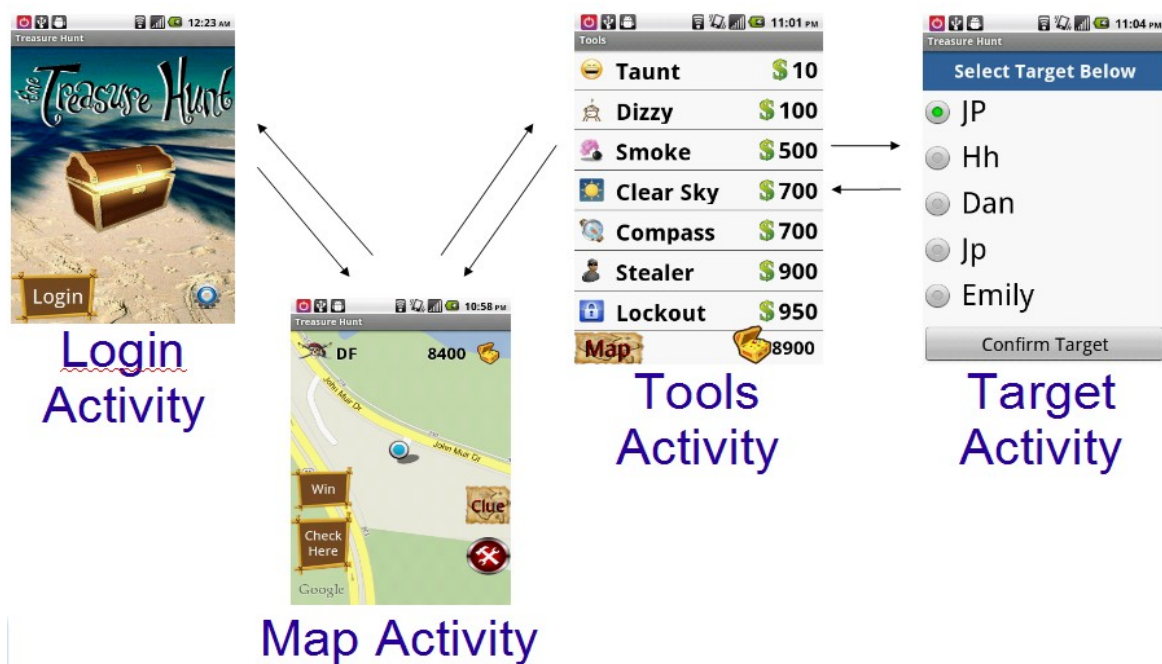


Figure 6.1.1

## 6.2 Login Activity

The Login screen has multiply functions. First, if the game has no saved user, then the player must log into the server. Clicking the “Login” button will bring the user to the login screen.

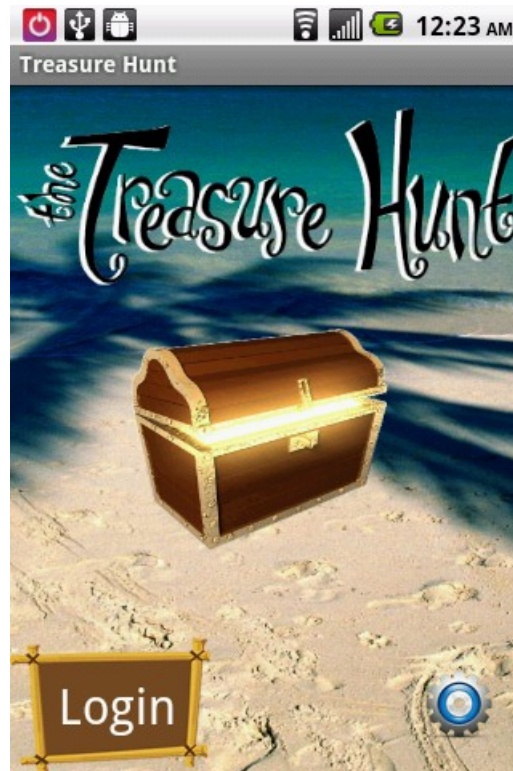


Figure 6.2.1

Here at the login screen (Figure 6.2.2), the user must enter a user name and password. Should the user name not be on the server, the server will accept it as a new user and will save the password for future use. The “Cancel” button will stop the login process and return the user to the previous screen (Figure 6.2.1).



Figure 6.2.2

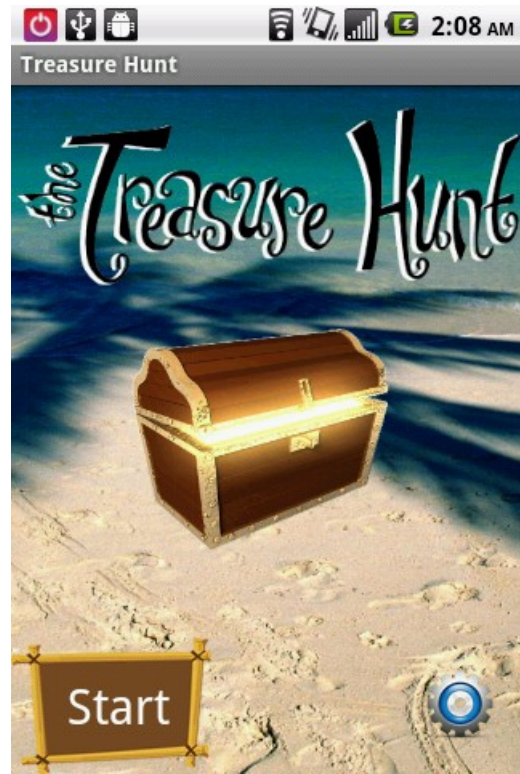


Figure 6.2.3

Once the user has successfully logged into the server, the “Login” button will switch to a start button that will bring the player to the Map activity.



Figure 6.2.4

The “Option” button at the bottom right of the screen (Figure 6.2.4) will bring up the options screen. Here the user has the option to logout of the game, look at current rankings, and run through a tutorial of how to play the game. The later two options are currently under-development on the Android client. (See Section 8, Future Developments).



### 6.3 Map Activity

Map activity shows the player's name, current points, an icon identifying the player's current location on the map, and the “Check Here”, “Clue”, and “Tools” buttons. The “Win” button is only for debugging purposes. This button allows the tester to send to the server the winning location thus emulating that the player is at the winning location. Upon first entering the game, the current location icon is designated as a “?”. Touching the icon will bring up a message box explaining the meaning of the icon (as shown in Figure 6.3.2).

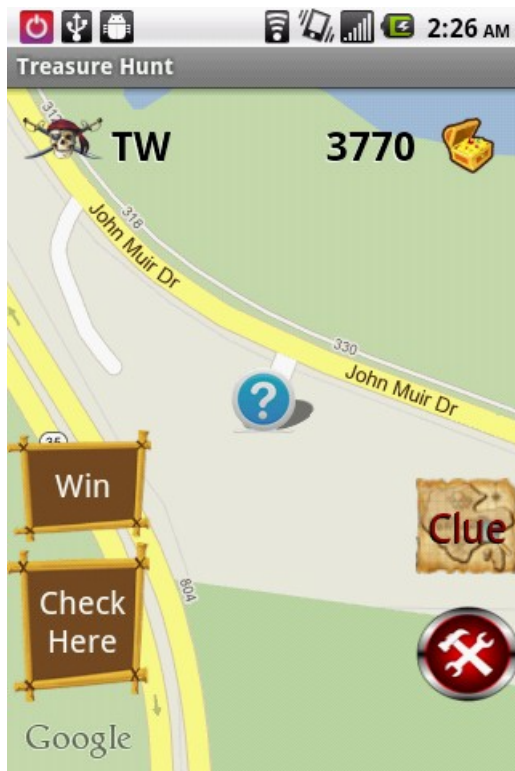


Figure 6.3.1

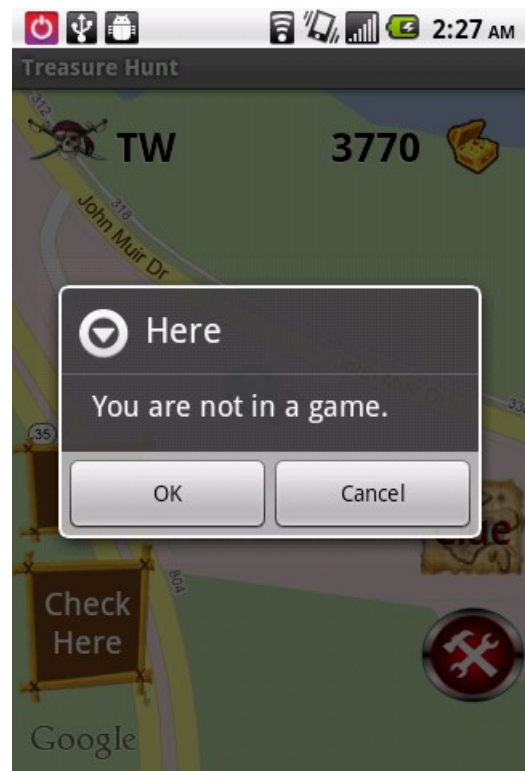


Figure 6.3.2



Figure 6.3.3

Clicking the “Check Here” button sends to the server the players current location as gathered by their Android device via either GPS or Wi-Fi. The server determines the player's distance from the treasure and replies to the player with the proper response. One response is if the player won or lost the game. The other possible response is to return the distance the player is from the treasure and the adjoining clue. If the player did not win or lose the game, then the player will see the clue screen appear with the clue to the location of the treasure (as shown in Figure 6.3.3).

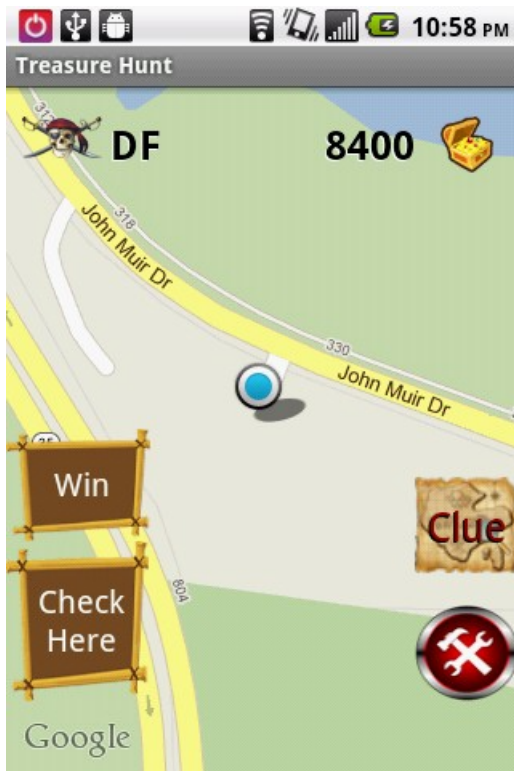


Figure 6.3.4

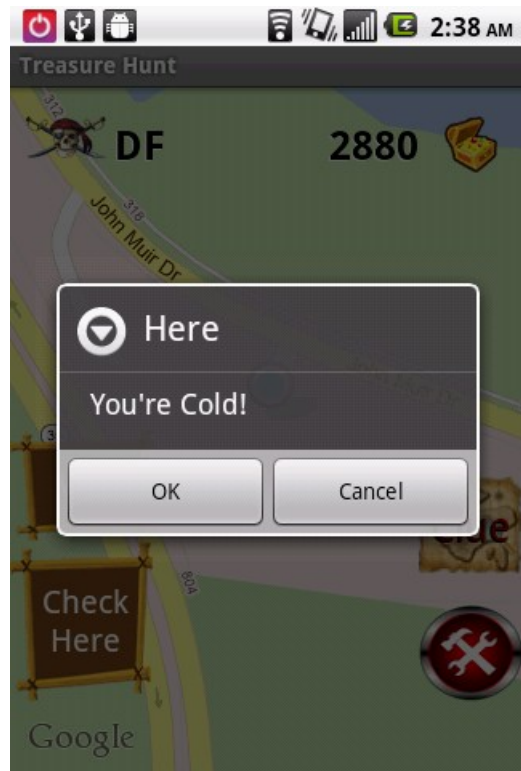


Figure 6.3.5

Pressing the “Clue” button will toggle the clue screen between visible or invisible. With the clue screen gone, the current location icon is visible. Again, touching the icon will display the meaning of the icon. For all possible icons refer to Section 7.2, Location Indicator Icons.



Figure 6.3.6

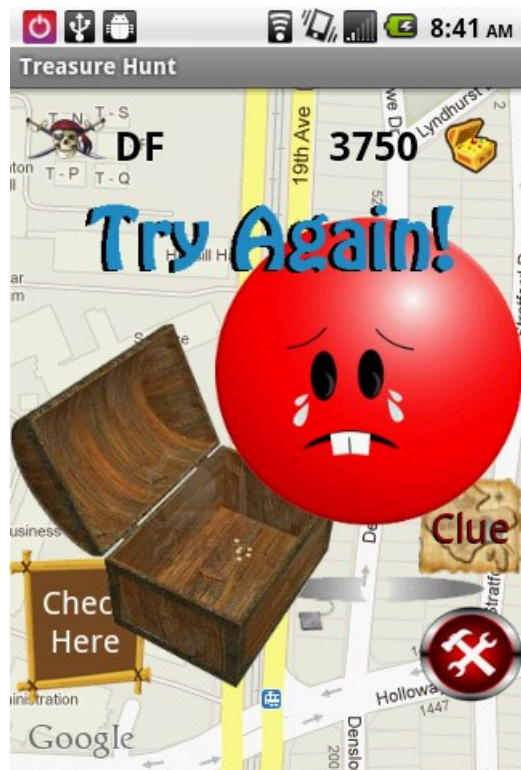


Figure 6.3.7

If the player won or lost the game, two different images will appear. The win screen (Figure 6.3.6) appears if the player is at the location of the treasure when they pressed the “Check Here” button. The server will award points to the player and start a new game with a new clue. If the player lost then the lost screen (Figure 6.3.7) will appear. The player will have to click “Check Here” button to get the new clue for the new game.

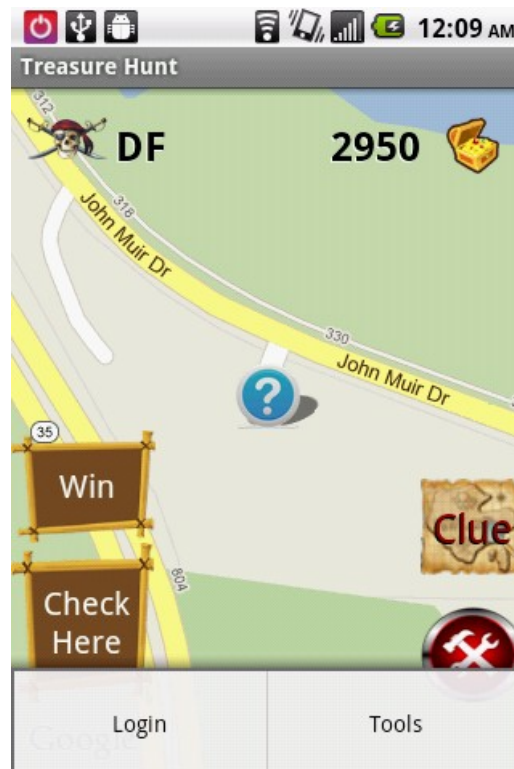


Figure 6.3.8

Navigation between the Login and Tool screen is also available by pressing the physical menu button on the individual's Android device. This brings up two menu options for the user (Figure 6.3.8). Since there is no icon for returning to the Login screen, this is the only way to access the Login screen from the Map screen.

## 6.4 Tools Activity

Clicking the red “Tools” button at the bottom right of the Map activity screen will bring the player to the Tools activity (Figure 6.4.1). The Tools menu has a list of available tools for purchase, the player's current points balance, and a “Map” button to return the player to the Map activity screen.

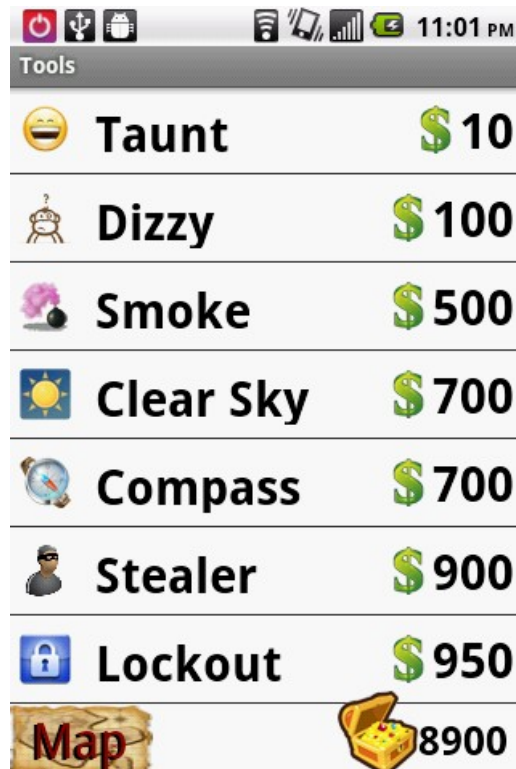


Figure 6.4.1





Figure 6.4.2

Touching the line that the tool is on will bring up the tool's help screen which includes the tools' cost, description, and a “Buy” button to purchase the tool. Touching anywhere on the tool's help screen will make the it disappear.



Figure 6.4.3

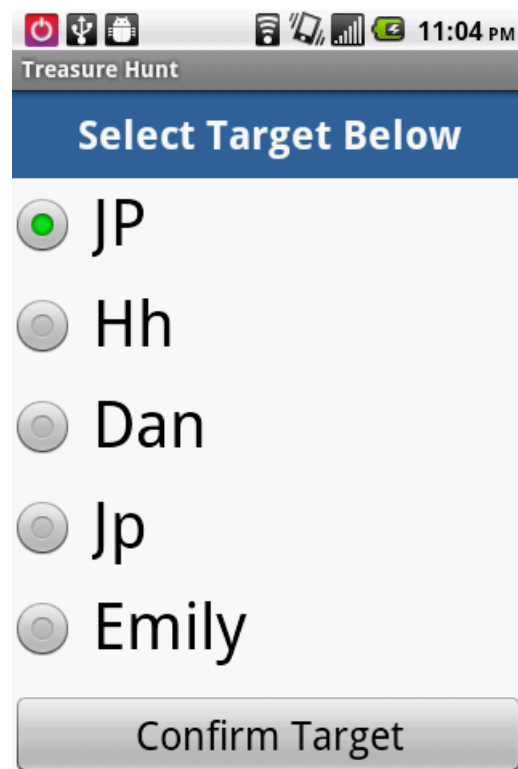


Figure 6.4.4

Clicking the “Buy” button will send the player to a confirmation screen (Figure 6.4.3). The player can “Confirm” to purchase the tool which deducts the cost of the tool from the player's current points balance. Some tools require a target. When necessary, a Target Activity screen (Figure 6.4.4) is called where the player can select from the players list of whom to target.



Figure 6.4.5



Figure 6.4.6

A good purchase (Figure 6.4.5) will show if the tool purchase was accepted by the server. A bad purchase (Figure 6.4.6) can occur for various reasons. Currently the server only allows a player to hold one negative effect, so if the player attempts to target another player that already has a negative effect on them, they will receive a bad purchase. Additionally, global effect tools like “Stealer” and “Lockout” can only have one owner. Attempting to purchase a “Stealer” when someone else has already purchased it will result in a bad purchase.



Figure 6.4.7

Should the player attempt to purchase a tool without the proper funds, then a “Not Enough Funds” screen (Figure 6.4.7) will appear and the purchase will not go through to the server.



Figure 6.4.8

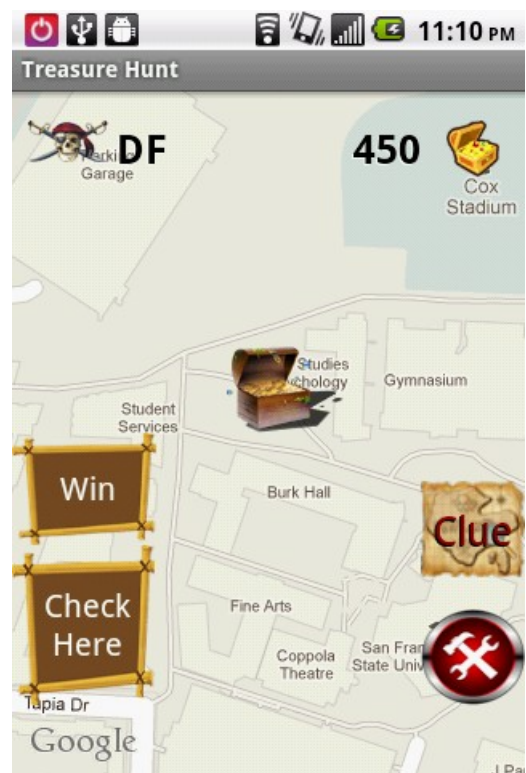


Figure 6.4.9

The “Compass” tool has a unique effect that is not noticeable until the player returns to the



Map activity screen. After a good purchase for the “Compass” tool, upon returning to the Map activity screen, the screen will move to the location of the treasure and show the treasure icon designating the winning location.

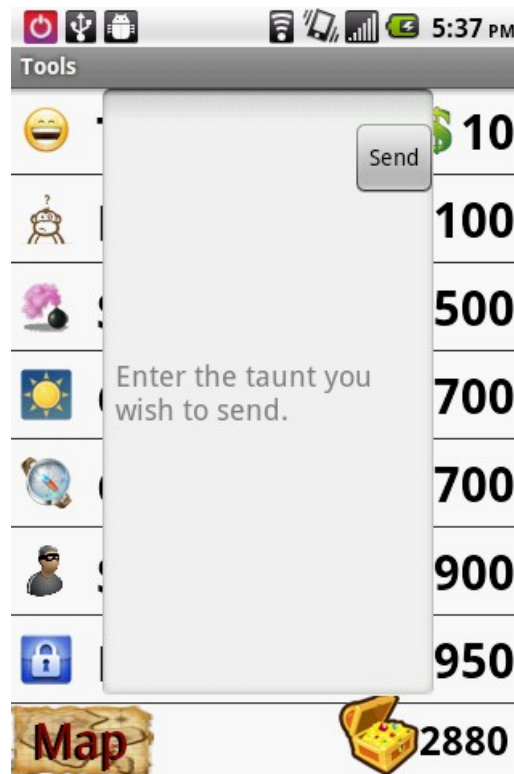


Figure 6.4.10

Taunt also has a special screen after being purchased. A text box will appear (Figure 6.4.10) and allow the player to enter a specific taunt that will replace the target's clue. Upon clicking the “Send” button, the player is brought to the Target activity to choose a target.

## 6.5 Target Activity

When a called from the Tools activity, the Target activity requests the list of players from the server (Figure 6.5.1). Once the targets are load, the player will be able to scroll through the list and select one player with which to target. The player can not select themselves as their name will not appear on the list. Self targeting tools like “Clear Sky” automatically target the player purchasing the tool.

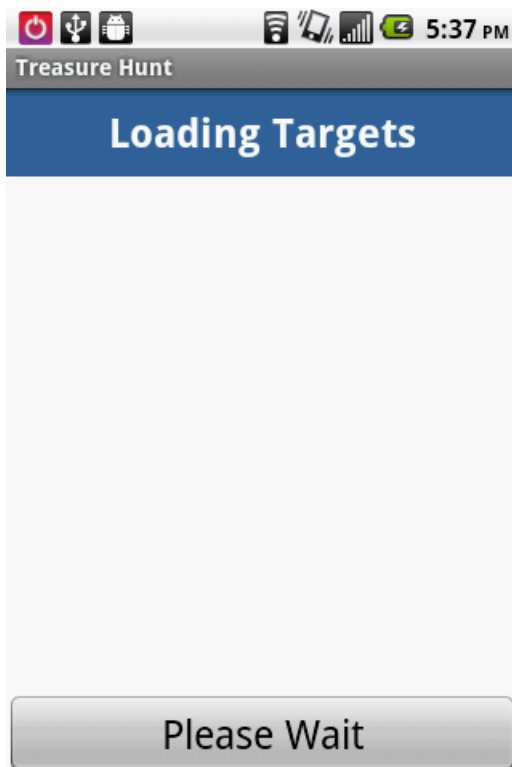


Figure 6.5.1

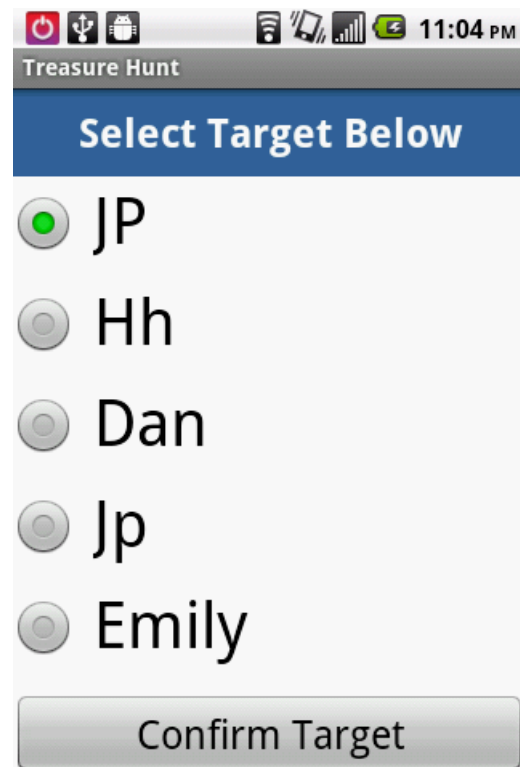














Figure 6.5.2

## 7 Icons








### 7.1 Game Icons

	Brings up the “Login” screen.
	Brings up the “Options” menu.
	Checks your location in the current game. Gives you a clue and sets your indicator based on how close you are to the treasure. If you are at the location of the treasure, you will receive a “Winner” message.
	Toggles showing the clue on the screen.
	Takes you to the “Tools” menu.
	Returns you to the game map.

### 7.2 Location Indicator Icons

	Player is not in a game. Also occurs when “Lockout” is active.
	Cold
	Warm
	Hot
	Treasure location.
	Smoke. A “Smoke Bomb” has been used on the player. Indicator is lost until duration of “Smoke Bomb” completes or player purchases “Clear Sky” to remove effect.

### 7.3 Tool Icons

	Taunt: Removes the clue from the target by replacing it with a taunt that you send.
	Dizzy Monkey: The target's location indicator will always show “Cold”.
	Smoke Bomb: The target shows smoke for their indicator.
	Clear Sky: Removes all negative effects on you.
	Compass: Location of the treasure will appear on the map until the next “Check Here”.
	Stealer: If any player claims a win, the holder of the “Stealer” wins.
	Lockout: No player can win the game.

## 8 Future Developments

- **Rankings:** The server supports both player rankings based on points and holds a time counter for each player that records the amount of time the player has played the game. Currently, the Android client is not set up to read either ranking or time played information from the server.
- **Tutorial:** Would like to setup screen shots and instruction on how to play the game, menu functionality, and tool use.
- **User Defined Game:** Allow the user to create their own defined playing area, goal locations, clues, and allowable players.
- **Winning Badge System :** Once in the winning region the player receives a collectible badge.
- **Complete Puzzle to Win :** Once in the winning region the player must complete a puzzle in order to unlock the collectible badge.

## 9 Project Learning Experiences

Communication is key when doing a project in such a short time frame. A lot can happen in one meeting so not having everyone present causes precious time delays in bringing everyone up to speed. Also, the missing person may have key insight into an issue being discussed. Such was the case in formatting our response-request communication, where one member had key information that would have saved everyone time in setting the formatting standard. Additionally, lack of communication led to deviations from the original design structure that was not passed on to the other team member. This resulted in adjustments that needed to be made on both the server and client sides during testing phases.

Clear, concise, and modular code became an apparent necessity when different people needed to work on code. Hours could be spent diagnosing how a method is utilized, especially without proper code documentation. Having good modular code allowed for quick additions to the code especially during testing phases when certain quirks on both the server and client appeared.

One of the best experiences was the knowledge that passed between team members. Each member brought their own expertise to the team. Things learned by all member included Android programming, JSON, GSON, networking protocols, Apache API, Eclipse, and using the various SDK's and API's associated with Android development like Google Maps.

## 10 Meeting Logs

### 10.1 *Week 1: Initial Meeting – April 6, 2012*

#### **Duration:**

4/6 – 2 hours

#### **Present:**

4/6 – Benjamin Britten, Dennis Feng, Terry Wong

#### **Discussed:**

4/6 – Application ideas

- Geolocation Treasure Hunt, the end result.
  - (Professor response) – this is too simple. Last year, a similar term project built a lot more complex ones...
- Started thinking about Basic Structure of App and Server.

#### **Who did what this week:**

Ben:

- Created Google repository.
- Tasked with creating project proposal.

Dennis:

- Wrote outline of project overview.

Terry:

- Installed SDK tools and started working on getting familiar with Eclipse and Android environments.

## **10.2 Week 2: Meetings – April 9, 13 and 14, 2012**

### **Duration:**

4/9 – 1 hour

4/13 – 30 mins

4/14 – 4 hours

### **Present :**

4/9 – Ben, Dennis

4/13 – Ben, Dennis, Terry

4/14 – Ben, Dennis

### **Discussed:**

4/9 – Design issue and milestones.

- Implementation of server–client message passing via HTTP.
- Discussed plans for the week to complete milestone 1 by 4/15.

4/13 – Project proposal and game ideas.

- Discussed final project proposal and assigned tasks.
- Discussed future ideas – Animations tied to in-game tools and exercise function (tied to in game metrics).

4/14 – Game ideas.

- Worked out program logic for Android client and Server.
- Wrote out the class structure for the server.
- Discussed Android GUI logic but still needs to be developed more.

### **Who did what this week:**

Ben:

- Created initial project proposal – 5 hours.
- Worked out project logic – 2.5 hours.

- Worked out server class structure and created google doc – 1.5 hours.

Dennis:

- Completed framework for client–server message passing via http. – 16 hours.
- Final Project Proposal – 1 hour.
- Worked out project logic – 2.5 hours.
- Worked out server class structure – 1.5 hours.

Terry:

- Began work on the server implementation. – 6 hours.



### **10.3 Week 3: Meetings – April 20 and 22, 2012**

#### **Duration:**

4/20 – 1 hour

4/22 – 2 hours

#### **Present :**

4/20 – Ben, Dennis

4/22 – Ben, Dennis

#### **Discussed:**

4/20 – Discussed multiple design issues.

- Discussed message format for passing code between server and client.
- Discussed layout of TOOLS View.
- Discussed multithreading using AsyncTask class.
- Discussed message persistence using SharedPreferences class.

4/22 – Discussed layout and design issues.

- Further refined layout of TOOLS View.
- Layout and control of “HELP” or “?” Popup for more info on various Tools.
- Layout and control for targeting players with Taunt, DizzyMonkey, and SmokeScreen.
- Made a plan for the week.
- Discussed porting Server to App Engine if time allows and adopting Java Servlets.

#### **Who did what this week:**

Ben:

- Initial UI Mockups and control structure – 2 hours.
- Uploaded server code to svn and helped teammate with svn – 30 mins.
- Reviewed initial server code – 30 mins.
- Final UI Mockups – 1.5 hours.

- Preliminary research into Java Servlets and AppEngine – 1 hour.

Dennis:

- Worked on Android client Tools View and Map View – 5 hours.
- Uploaded current Android client to SVN – 15 mins.
- Worked on Android networking – 3 hours.

Terry:

- Continued working on server code – 5 hours.
  - added steal, lockout, encryption, and authentication.

#### **10.4 Week 4: Meetings – April 25 and 26, 2012**

##### **Duration:**

4/25 – 2.5 hours

4/26 – 1 hour

##### **Present:**

4/25 – Ben, Dennis, Terry

4/26 – Ben, Dennis (Phone Meeting)

##### **Discussed:**

4/25 – Discussed various design issues.

- Discussed data format for passing between server and client.
- Discussed targeting Android 2.3 for compatibility issues.
- Discussed the idea of using a Java Servlet for server request and response handling.

4/26 – Discussed switching to JSON for client-server message passing format.

##### **Who did what this week:**

Ben:

- Server – Code and Logic Review – 2 hours.
- Server/Client Debugging and Testing – 2 hours.
- Client – Updated communication with server, created HttpClient class and updated.
- MapsActivity to reflect new communication – 4 hours.

Dennis:

- Worked out SDK version issues – 1 hour.
- Worked with Terry on Client - Server Communication and Message Format – 3 hours.
- Writing and Testing Android HTTP communication – 4 hours.

Terry:

- Added second iteration of Server to SVN (had problems with his computer) – 1 hour.

- Updating request and response formatting – 4 hours.
- Looking into Synchronization needed for server data – 1 hour.

## **10.5 Week 5: Meetings – May 4, 2012**

### **Duration:**

5/4 – 2 hours

### **Present:**

5/4 – Ben, Dennis, Terry

### **Discussed:**

5/4 – Discussed game dynamics

- Clear Skies will remove negative effects affecting user.
- Compass will show winning location on Map.
- Going to try giving player base 1000 points to begin with.
- Rewards for first hot, cold, kind of like achievements.
- Server storing more user data to log (point total, etc).

### **Who did what this week:**

Ben:

- Wrote an Android HttpClient and updated Server communication to get them working – 4 hours.
- Rewrote Client and Server data parsing and creating to use JSON format for passing data – 2 hours.
- Code review – 1.5 hours.
- Updated method for pulling GeoLocation in client – 0.5 hours.

Dennis:

- A ton of updates to Android App GUI – 12 hours
  - custom app icon, login page with artwork, updated maps page, updated tools page with artwork, fixed tools page scrolling issue, added individual tools info popup with purchase button, confirm purchase popup.

Terry:

- Server – 4 hours
  - added logging support to keep track of current point total.
  - refactored some code for efficiency.