**Authors**: Anthony Kiesel, Hira Ahmed, Jon Petersen, Shawn Fairbourn, Kenny Fryar-Ludwig, Kris Mygrant

Urban Hunt: *The Final Project Plan*

# Project Overview

Urban Hunt will allow users to connect with others in their area to play games in real time based on the GPS location of their mobile phone. Many different game modes will be provided, revolving around users interacting with each others’ location data. Some example game modes for the application are: Capture the Flag, GPS Tag, scavenger hunts, “Where’s Waldo”, races, and team relays. Many of our anticipated users would be interested in fitness, however the finished product may include less physically intensive games based on the GPS location

In order to implement this system, we will focus our design on two major components. The first component is a mobile client and UI that allows users access to our system (the app), with some users acting as “hosts” and others as “clients” to facilitate game creation and setup; this satisfies our project requirement for mobile development. The other major component will be a web-hosted MYSQL server that stores and maintains player location data for the duration of the game, and sends location update information to all relevant players of a specific game. In addition to our users there are other actors with a bearing on the outcome of our project, such as the Google maps API used to track player movements and connect users to others in their area, as well as the database used to store user data through the term of the game. The database component satisfies the data persistence requirement for our project, and the communication framework between these two components satisfies the network communications requirement. The primary audience for this application will be mobile phone users (specifically Android users) interested in connecting with their friends through real world games. Due to the mobile nature of the system it may attract users looking to improve their fitness as well, as our game provides an active alternative to other popular mobile games.

This project will also significantly contribute to each team member’s professional development, as we have no previous experience with building such an application and team members will be expected to gain the fundamental skills needed to develop mobile and web-based applications while completing the project.

**Actors:**

* Host User
* Client Users
* Google Maps
* Third-party “Multiplayer” server
* Android™ Smart Device (Phone/Tablet)

**Users:** Fitness Junkies, Android users, “Night Games” people

**Connected Systems:** MySQL server, Google Maps, Android Devices

**Use Cases:**

1. Play a variety of group games (Users)
   1. Track team locations
   2. View tags/Player status
   3. Send/recv messages to/from team
   4. Send/recv messages to/from all players
   5. View digital boundaries
   6. View street map
   7. View satellite map of local area
   8. Update group on current objectives
   9. Initiate game
   10. Save game state for future replay
   11. Replay Mode
   12. Save list of commonly connected friends
2. Track independent GPS stats (Google Maps)
   1. Provide location data
   2. Track player movements
3. Provide Sensory Feedback (Phone)
   1. Vibration Feedback
   2. Play sound
   3. Display Visual data
4. Server
   1. Store data
   2. Send data
   3. Receive Data

# Team Organization

Our team will be organized loosely following Agile principles, with members accepting work on a short-term goal-oriented basis. We will allow room for adjustments to the process, however we believe that an Agile philosophy will be best given the time constraints to our deliverables. During our weekly meetings we will be able to assign out tasks to team members based on current skills and interests. As we continue to progress through the semester, it is expected that some team members will focus in on specific components of the application, though all members are expected to maintain familiarity with the whole project.

**Rough Outline of responsibilities**

|  |  |
| --- | --- |
| Responsibilities and Team Roles | Team members responsible |
| Leader | Kenny |
| Documentation | Hira, Kenny |
| Team Communications | Anthony, Kenny |
| Communications with domain experts | Anthony, Kenny, Jon |
| User Interface Design | Kenny, Shawn, Anthony |
| Specialized Android (sounds, vibrations, etc..) | Kris |
| Database Design/Management | Jon |
| Server/DNS management | Anthony, Jon |
| GUI Implementation | Hira, Shawn |
| Logic/Algorithms | All |
| Network/Server Communications | Kris, Jon |
| Maps API | Kenny, Hira |
| File Structure Management/Repository | Kenny, Anthony |
| Testing | Hira, Kris, Shawn |

# Software Development Process

The development will proceed using a very simple process model that uses elements of Boehm's spiral process [Boehm-1988], iterative process models, and agile software development. The following table shows the entrance and exit criteria for these phases.

|  |  |  |
| --- | --- | --- |
| **Phase** | **Iteration** | **Tentative Exit Criteria** |
| **Phase 1** | Project Planning | Project selected, plan completed, team organized, configuration management in place |
| **Phase 2** | Requirement Capture and Analysis | A common understanding of requirements and system analysis captured in meaning documentation  A complete prototype that will help improve understanding of requirements |
| **Phase 3** | Architectural, UI, and DB Design | A system design that satisfies the requirements  A prototype that explores a design alternative |
| **Phase 4** | Detailed Design, Implementation, and Unit Testing | Initial implementation (60% - 80% of the functionality) with some unit test cases) |
| **Phase 5** | More Implementation and Testing, Packaging, and Deployment | Final implementation, testing, packaging, and deployment |

For analysis and high-level design activities, we will use a conceptual-model language, called *Unified Modeling Language* (UML). The UML is expressive and semi-formal. Its expressiveness allows it to describe a wide range of concepts in both the problem and solution domains. Its formalism is rooted in a meta-model that describes its syntax and some of its semantics.

# Feature Priority

## High Priority

1.1 The system will provide a screen that allows users to host a new game

1.2 The system will allow users to join an existing game

1.3 The system will keep track of who is currently active in a given game

2.1 The system will provide a set of rules for specific types of games

2.2 The system will provide an engine to provide that set of rules to current players

2.4 The system will record the current location of players from their phone (Unimplemented)

2.5 The system will display the location of other players

2.7 The system will determine when a game is finished (Unimplemented)

2.9 The application will maintain a network connection component to talk to the location server

2.10 Google Maps will determine actual location from GPS coords

2.11 The application will determine how users are interacting (collisions)

4.1 The location server can authenticate component to prevent cheating

4.2 The location server can store data for Player Location

4.3 The location server can store data for Player Teams

4.4 The location server can store data for Games being played

## Medium Priority

2.3 The system will provide verification that users play by the rules (Unimplemented)

2.6 The system will display the objectives of the current game

2.8 The phone will provide sensory feedback (Visual, audio, haptic) based off of game status (Unimplemented)

4.5 The location server can store data for messages / other misc information about a game (Unimplemented)

6.0 The system will provide an AI component to simulate other players

## Low Priority

2.12 The application will have the capability to save game data to resume play later (Unimplemented)

3.1 Fitness users can keep stats on local storage of distance travelled (calories burned?) (Unimplemented)

3.2 Fitness users may keep stats to replay the locations of users after completion of a game (Unimplemented)

5.0 The player’s applications will receive messages from the host of the game

6.0 The application will provide a splash screen to keep already whipped user engaged (Unimplemented)

# Communication policies and procedures

As a team we have agreed to conduct most of our communications through weekly team meetings and by email. Our weekly meetings are currently scheduled for Wednesday afternoons in the CS 3450 classroom, with an additional meeting on Friday as necessary. We have also established a Google Drive folder containing project documents that all team members are free to edit. This shared folder will allow us to collaborate on documentation without needing to find times where we are all available.

During the planning phases of the project, most issues will come before the whole team. As we begin development and team members start to specialize in their roles, we may choose to designate specialists to handle tasks within their specializations before bringing them before the group if we begin to run out of time during our weekly meetings. Another major component of our weekly meetings will be examining the current state of our project on BitBucket and reviewing any major changes made.

# Peer Review

Our peer review will consist of each member of the team individually filling out their own customized peer review form. We will be conducting peer reviews after each project deliverable is finished. The peer review will consist of evaluating each team member on their dependability, willingness to work, and their actual contributions to the project.

After we have begun coding we will begin a peer review process where at least two other team members will review the code before the code enters the testing branch. From there, the team as a whole will be continually testing the branch, and once we are satisfied that the bugs have been addressed, the code will move onto the production branch.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Work Breakdown Structure (WBS)** | | | | | | | | |
| **Category / Task** | **Three Point Estimation** | | | | | | **\_(Estimate\_)** | **Notes** |
| **Best** | **Likely** | **Worst** | **Exp** | **SD** | **95%** |
| **Create User Interface (app)** |  |  |  |  |  |  | **69.7** |  |
| **Design User Menus** | **2** | **4** | **8** | **4.3** | **2.0** | **6.3** | **6.3** |  |
| **Implement Visual Interface** | **5** | **10** | **20** | **10.8** | **4.9** | **15.7** | **15.7** |  |
| **Populate Google Maps** | **10** | **16** | **24** | **16.3** | **4.5** | **20.8** | **20.8** |  |
| **Create “audio scheme”** | **4** | **8** | **16** | **8.7** | **3.9** | **12.6** | **12.6** |  |
| **Vibrate functionality** | **1** | **2** | **3** | **2.0** | **0.6** | **2.6** | **2.6** |  |
| ***Integration (20%)*** |  |  |  |  |  |  | **11.6** |  |
| **Create the Game Engine** |  |  |  |  |  |  | **127.8** |  |
| **Implement Main Game types** | **15** | **20** | **25** | **10.0** | **3.2** | **13.2** | **13.2** |  |
| **Interface with Maps API** | **5** | **10** | **15** | **10.0** | **3.2** | **13.2** | **13.2** |  |
| **Send info to GUI for “populating map”** | **4** | **8** | **15** | **8.5** | **3.6** | **12.1** | **12.1** |  |
| **Create Polling logic** | **4** | **8** | **12** | **8.0** | **2.5** | **10.5** | **10.5** |  |
| **Create Main Routine** | **15** | **20** | **25** | **10.0** | **3.2** | **13.2** | **13.2** |  |
| **Set up Data Structures** | **8** | **12** | **16** | **12.0** | **2.5** | **14.5** | **14.5** |  |
| **Create networking scheme** | **8** | **12** | **18** | **12.3** | **3.2** | **15.5** | **15.5** |  |
| **Future Implementations** | **10** | **20** | **30** | **15.2** | **5.2** | **17.2** | **18.2** |  |
| ***Integration (20%)*** |  |  |  |  |  |  | **18.4** |  |
| **Register App** |  |  |  |  |  |  | **1.1** |  |
| **Create App Name** | **1** | **1** | **1** | **1.0** | **-** | **1.0** | **1.0** |  |
| ***Integration (10%)*** |  |  |  |  |  |  | **0.1** |  |
| **Project Management** |  |  |  |  |  |  | **12.0** |  |
| **Consult Specialists** | **8** | **15** | **20** | **14.7** | **3.8** | **18.5** | **18.5** |  |
| **User FAQ/Tutorial** | **2** | **5** | **8** | **5.0** | **1.9** | **6.9** | **6.9** |  |
| ***Integration (20%)*** |  |  |  |  |  |  | **5.1** |  |
| **Setup Server** |  |  |  |  |  |  | **52.3** |  |
| **Create Database** | **5** | **10** | **20** | **10.8** | **4.9** | **15.7** | **15.7** |  |
| **Create polling logic** | **4** | **8** | **15** | **8.5** | **3.6** | **12.1** | **12.1** |  |
| **Secure Server/Database** | **5** | **10** | **20** | **10.8** | **4.9** | **15.7** | **15.7** |  |
| ***Integration (20%)*** |  |  |  |  |  |  | **8.7** |  |
| **Project 95% Confidence** |  |  |  |  |  |  | **264.1** |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Risk Analysis** | | | | |
| **Risk / Mitigation Activity** | | **Probability** | **Impact** | **Estimate** |
| **R1** | **Lack of experience** | **High** | **Medium/High** | **60** |
| **1.1** | **Undetectable Bugs / User testing** | **Low/Medium** | **Medium/High** | **20** |
| **1.2** | **Project Complexity / Incremental Development** | **Low** | **Low** | **30** |
| **1.3** | **Incorrect implementation / Code review** | **Medium/High** | **Low** | **10** |
| **R2** | **Server/Network Failures** | **Low** | **High** | **15** |
| **2.1** | **Too Much Bandwith / Buy more Bandwidth** | **Low** | **Low/Medium** | **10** |
| **2.2** | **OS/Server software failure / Daily backups** | **Low** | **Low** | **5** |
| **R3** | **Integration Errors** | **Medium** | **Medium/High** | **50** |
| **3.1** | **Android Version Fragmentation / Build for Targeted Versions** | **Low/Medium** | **Medium** | **30** |
| **3.2** | **Merge Conflicts/Increased Communication and Code Review** | **Medium** | **Medium** | **20** |
| **Total** |  |  |  | **125** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Project Estimate** | | | | |
| **Area / Activity** | **Suggested\*** | **Multiplier** | **Totals** | **%** |
| **Management** |  |  |  |  |
| **Management Activities** | **0.20 - 0.40** | **0.25** | **66** | ***6.6%*** |
| **Analysis** |  |  |  |  |
| **Analysis Activities** | **0.10 - 0.30** | **0.25** | **66** | ***6.6%*** |
| **Development** |  |  |  |  |
| **Software Development** |  | **from WBS** | **264** | ***26.5%*** |
| **Unit Testing** | **0.25 - 2.00** | **1** | **264** | ***26.5%*** |
| **System Testing** |  |  |  |  |
| **System Testing Activities** | **0.15 - 0.5** | **0.5** | **132** | ***13.3%*** |
| **Deployment** |  |  |  |  |
| **Deployment Activities** | **0.11 - 0.14** | **0** | **0** | ***0.0%*** |
| **Risk Mitigation** |  |  |  |  |
| **Risk Mitigation Activities** |  | **from Risk** | **125** | ***12.5%*** |
| **Contingency Reserve** |  |  |  |  |
| **Known Unknowns** | **0.05 - 0.20** | **0.2** | **53** | ***5.3%*** |
| **Management Reserve** |  |  |  |  |
| **Unknown Unknowns** | **0.0 - 0.20** | **0.1** | **26** | ***2.7%*** |
| **Other Expenses** |  |  |  |  |
| **Other Expenses (hardware/software/etc.)** |  |  |  |  |
| **Totals** |  |  | **997** | **100%** |

# Development Environment

Because we will be developing an Android application, we will make heavy use of Android Studio (Based off of IntelliJ) for application development. Our primary languages will be Android Java for the application, with SQL and Python on the server. We will be using the Bitbucket client, SourceTree, to interface between Android Studio and our repository. We will also be using BitBucket’s online system for code review. We’ve decided to use MySQL as our database language to interface with our mobile application. From the server we will use a private/public key pair to validate that the application has read/write access to the users specific entry of the database. We will use our own SSH key pairs to connect to the server. In addition we will be using KeePass as our shared password management system and Google Drive/Documents to share non-development documents and collaborations.

# Configuration Management

## Name Schemes

Naming will be determined by the Android style guide. Several notable examples include:

1. camelCase variables (someVar, anotherVar)
2. Member variable prefix (m\_memberVar)
3. Array prefix (aSomeArray)
4. Reference variable (rReferencedVar)

## Change Control

Git will be our version control system, where we will be able to sync our changes across our platforms using the online BitBucket service. To track our changes to the program, our commits will have a header comment that describes what changes were made, when, and by whom. Additionally, these comments will be augmented by “Commit Descriptions” that are built into the Git system. Using these we will be able to assign responsibilities for various features and track what changes cause what bugs.

## Product Building

We will be keeping 3 different branches in our repository. We will start coding in the Development branch and once we feel we have working code, merge those changes to our Test branch. After adequate testing to ensure that the new code is stable, we can publish the final changes to our Production environment.

## Version Management

We are implementing a 3-level version control system with 3 separate Git branches. These branches are Master, Test, and Develop. Develop will be actively under development and is not expected to remain fully stable. Once a feature has been added to *develop* and the branch is considered stable, we will then merge *develop* into *test.* *Test* will then be extensively testing using a predefined list of requirements. Bugs that are found will be fixed in *test* and merged back into *develop.* Once *test* is undeniably stable we will merge that into *master*. *Master* will undergo specific testing related to that feature. *Master* should never fail.

# Change Log

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Description of Change | Review by | Review on |
| 1/28/15 | Initial Draft | Team (All) | 1/30/15 |
| 2/13/15 | Update Use Cases | Hira | 2/13/15 |
| 2/13/15 | Peer Review section added | Kris | 2/13/15 |
| 2/13/15 | Feature Priority section added | Anthony | 2/13/15 |
| 3/20/15 | Added configuration management | Kenny, Anthony | 3/20/15 |
| 4/24/15 | Updated Feature Priorities | Hira, Shawn | 4/24/15 |
| 4/24/15 | Updating Project Plan | Hira, Shawn | 4/24/15 |
| 4/24/15 | Added features to Android App | Shawn | 4/24/15 |
| 4/24/15 | Contributed audio to the App | Kris | 4/24/15 |