Durham University
School of Engineering and Computing Sciences

Design Document

2012-13

Project Name: Geocaching

Team Number: 3

James Camden

James King

Emma Nugee

Alice Smiddy

Charles Wilson

0 Document Information

0.1 Version History

Ver. No	Ver. Date	Revised By	Description
1.0	21/01/2013	Alice Smiddy	Design Document Created
1.1	30/01/2013	Alice Smiddy	Created Introduction
1.2	05/02/2013	Alice Smiddy	Created Interprocess Dependencies

0.2 Changes to Requirements

2.4 - Cache Balance

Type	Functional
Description	A cache must have a point balance associated with it, which must be visible to the owner of the cache when requested.
Priority	High
Pre-conditions	2.3 Cache Ownership
Input	A user makes a request to view the point balance of a cache.
Operations	The server checks to see if the user is authorized to view the point balance of the cache.
Expected Results	The balance of a cache will be displayed to the owner when requested.

2.9 - Cache Scouting

Type	Functional
Description	If a user physically visits the location of a cache owned by a different user, they can choose to use points to scout a cache, and possibly find the point balance of that cache.
Priority	Medium
Pre-conditions	2.4 Cache Balance, 2.1 Account Balance, 2.2 Account Transactions
Input	The location of the user and the number of points they wish to scout with are supplied.
Operations	The distance of the user is checked to ensure they are sufficiently near the cache and the number of points used to scout is checked to ensure the user has at least that number of points in their account. The server will decide if the scout is successful.
Expected Results	If the request was valid, a scout is initiated on the cache by that user with the specified number of points.

2.13 - Battle Breakdown

Type	Functional
Description	After an attack on a cache, the attacking user will be able to view a breakdown of the results of the attack.
Priority	Low
Pre-conditions	2.10 Cache Attacking
Input	An attack on a cache has concluded.
Operations	The server will provide information such as the initial size of each army, the result, how many survivors the user has and how many deserters they've gained.
Expected Results	A breakdown of the results of the battle will be displayed to the attacking user.

2.19 - Scouting Non-Player Caches

Type	Functional
Description	If a user has attacked a non-player cache and the minimum delay between attacks on a non-player cache has not elapsed for that cache, the user will not see the cache on the map so cannot interact with it.
Priority	Medium
Pre-conditions	2.18 Attacking Non-Player Caches, 3.2 Nearby Caches
Input	A user attempts to view a non-player cache on the map.
Operations	The server will check the amount of time since that user last attacked the cache (if at all).
Expected Results	If the user has never attacked that cache, or the time since the last attack is greater than the amount of time a user must wait between attacks on a non-player cache, they are able to view the cache as normal. Otherwise they cannot.

3.2 - Nearby Caches

Type	Functional
Description	The application must be able to show the user, on the map, the locations of caches near to the user's location.
Priority	High
Pre-conditions	3.1 Display Location
Input	The user will request to see the locations of caches near to their location.
Operations	The application will check that the device's GPS antenna is enabled and there is internet connectivity. If there is a connection and the antenna is enabled, the application will send a request to the server for a list of coordinates of caches near to the user's location.
Expected Results	The application will mark on the map, provided by the Google Maps API, all caches near to the user's position.

0.3 Table of Contents

Contents

1 Introduction

1.1 Purpose

The purpose of this document is to provide a thorough guide for developers to implement the virtual geocaching application Fortitude. It has a strong basis in the requirements document such that the clear diagrams and high and low level designs contained within can be used to create software that fulfils the specification laid out therein.

From this document a mobile application and supporting website will be produced that encapsulate an exciting game based on creating, defending and conquering caches. The system realised will allow users to create accounts that can be used for playing the game Fortitude, and will be able to interact with this game through an easy to use and appealing interface.

1.2 Scope

The system designed within this document covers users creating accounts that can be used in conjunction with the game Fortitude, in which a user directs an army to conquer and defend caches (playing the role of forts in the game). The aim of the game is to own the most caches; an army can be increased by attacking outlaw camps, which cannot be owned but provide soldiers when defeated. The user gains caches by placing new caches in available locations or using their army to conquer those owned by other users, and must also leave soldiers at their caches to defend against attacks against them. This core section of the game is conducted solely through the application; the website serves a supporting role allowing users to manage their accounts and plan battle strategies. In a future release of the game, the website will include a forum or message board system to allow users greater interaction with each other and therefore more complex strategies and alliances to be formed; in the current version, users communicate via a private messaging system accessible through both the app and the website. Administrators of the application work through the website, and are able to place or delete caches remotely, delete user accounts, and act on user requests including questions or reporting a cache, a particular user communication or another user.

Creating an account with a username, password and email address may be done through the GUI on the phone application or the website, and is authenticated through an email sent to the user. At any time, the user can request a new activation email or a password reset email. Once authenticated, the user can build their army of soldiers, which act effectively as a point score, and interact with caches if physically present at the cache s location. The locations of caches are displayed on a map, which can also show the user s location and a route from the user to a specified cache. Interactions with caches include placing a cache if no other cache is within 300m of the location; adding or withdrawing soldiers from a cache the user owns to increase or decrease the defensive army of that cache; adding soldiers to an empty, unowned cache in order to become that cache s owner; or scouting and/or attacking an enemy cache. At any time a user can see basic information about a cache, such as who owns it.

Scouting a cache is the only method of seeing how large that cache s defensive army is, and is done by sending a number of soldiers (each of which has a chance of failing and dying, in which case the user s army will decrease accordingly) to scout the cache. A user can chose to attack the cache after or without scouting, and will specify the number of soldiers to attack with. If they are victorious, they become the new owner of that fort, and the surviving soldiers of their attacking army become the new defending army of the cache, and the soldiers of the defeated army die or surrender and become owned by the victors. If the attackers lose, then every attacking soldier dies and the

surviving defenders will continue to defend the cache. After the battle, the user will be displayed a breakdown of results including the initial number of soldiers in each army, the winner of the battle, the survivors of the attacking side and the amount of soldiers that surrendered to the winning side.

Outlaw camps are non-player caches which cannot ever be conquered, but can be attacked in the same way as user owned caches. If the user defeats the outlaw camp then their army increases as the defeated outlaws join the army, however if the user loses then the soldiers in the attacking army are lost. There are limits on the number of times a user can attack an outlaw camp within a given time period.

The final type of cache is the special event caches (or treasure areas) found by entering the range of a wireless network with a specific MAC address. These reward the user on finding them with a number of soldiers, and may in a future version of the game include other treasure such as weapons or advanced soldiers that would strengthen the user s army. Each special event cache will only reward a user once and has a limited supply of treasure, meaning that after a certain number of users visit it the cache will become empty and removed from the game.

Caches cannot be interacted with on the website, but users can view caches on a map and filter the view such that, for example, only enemy caches are displayed. They can also view information about their own caches and account including a record of their history in the game. A user can also manage their account through the website, including updating their details, changing their password or requesting their account be deleted.

The application and website are run off a database storing each cache and user which is accessed and updated by a server. The server also provides the functionality behind both the app and the website, as well as checking the validity of data and commands to reduce errors or the possibility of cheating the game; this is particularly important in authenticated an account or ensuring that location specific actions are performed at the correct location. Certain restricted features, such as promoting a user to an administrator, can only be accessed directly through the server and not through the website or app.

1.3 Definitions, Acronyms and Abbreviations

1.4 References

1.5 Overview

The remainder of this document will be a thorough guide for developers of this system. It will be split into Interface Descriptions and Element Descriptions. In the Interface Description section there will be mock ups of the user interface for both the app and website and descriptions and diagrams of how a user might use these. The Element Description section will design the system architecture and core parts of the system, and explain decisions along the way. It will do this through design elements each with a specific function, shown through block diagrams. Class diagrams will be used to show the static structure of the system and dependencies between components. Activity diagrams and sequence diagrams will be used to show the dynamic behaviour of the system.

2 Interface Descriptions

While it may seem far removed from the core programming aspect of software engineering, the design of a good user interface is key to the success of any system. Indeed, Douglas Bell (1987) went so far as to claim that the user interface is the "yardstick by which a system is judged," and certainly it plays a crucial role in attracting users to the software.

This section will describe what factors are taken into consideration in the design of the interface for the Fortitude application, and how the final product will be suitable, appealing and fulfil the criteria set forth in the requirements document (23 Cats, 2012).

2.1 Design Methodology

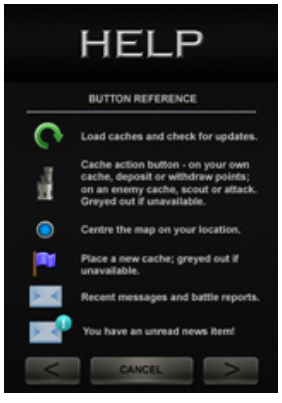

Figure 1: Button reference in the help

While designing the interface, it was important at all times to consider ease of use for users of various levels of familiarity with the system. Our domain analysis had highlighted the importance of intuitiveness in a system and of required functions being easily accessible (23 Cats, 2012:5-7); for this reason we have tried at all times to make the interface clear, and have also included a help screen should the user ever require it (see fig 2.1). Bell (1987) highlights the role of a well designed user interface in reducing errors, and to this end several of the features of our app were tested on potential users during development. In particular, it had originally been decided to disable the android back button within the app to give us greater control over which screens the user could access at any point in time. However, our users found the lack of a back button difficult to get used to, and so its functionality has been incorporated into the application wherever possible.

To further the ease of use of the Fortitude Application, buttons which perform a specific action will be greyed out when the action is not available. These include, for example, the 'place cache' button which would be greyed out if the user was too close to an existing cache, shown right in fig 2.2. The greyed out flag would

Caches	Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.
Users	Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.