Contents

1 Domain Analysis

1.1 Technical Analysis

The product will include a user account system, where users can create accounts which store personal information, and can log in to their account to access resources. Implementing this system will involve solving several security and design problems, but thankfully the user account model has been tried and tested in thousands of online applications.

Almost all websites with account registration require the user to validate their email address in order to complete the account creation process. This is generally implemented using an activation email sent to the address they gave while registering. The user must either use an identification code inside the email or click a link in the message to activate their account. This will prove that the user has access to that email address. Forcing the user to validate an email address can help deter users creating many accounts because each account needs a unique email address assigned to it. This can also counter the problem of spam accounts being created by an automated system.

Another problem is account security, and making sure a malicious user cannot gain access to another user's account. One aspect of this is requiring users to register with more complex passwords. A few websites will not accept passwords which do not conform to certain constraints, such as having at least one digit and a mixture of upper and lower case. This prevents users from using simple and easy to guess passwords. However, even the most complex and hard to guess password is useless if an adversary can intercept it when being sent to the server. To protect against this, passwords are usually hashed with a hashing function that is very difficult or impossible to reverse . The server will never see the user's actual password, only a hash of it. This has the added benefit of protecting every users password if the server database is compromised. While it is true that if the password is intercepted the adversary can use it to authenticate themselves into that particular service, because the password is hashed they can not find the original password and use it with other services. This therefore protects users which use the same password for many websites.

A user's authentication credentials are more likely to be intercepted if they need to send them frequently. This is conventionally prevented by implementing authentication sessions, where the user only needs to provide their credentials once per session. The server then creates a hash code which identifies the session, and this code is sent to the client. For subsequent requests which require authentication the client only needs to send the session hash. The session usually expires after a certain amount of time after it was initiated, or since the last action during that session was made. To add even more security, a session may be associated with the IP address of the client which requested its creation. If an adversary were to listen in and obtain the session hash, they would be unable to use it to authenticate because their IP would differ from the legitimate user.

The Data Protection Act (1998) requires that organisations holding personal information must remove that information when the subject of the data requests it[1]. Usually this is complied with by allowing the user to delete their account through an automated process, without any intervention from an administrator.

2 Hardware and Software

The main product shall take the form of an "Android app" - a self contained program running on an Android mobile device. This is what the user will directly interact with. Our product will also require a central server in order to function. The Android app shall be able to transfer information to and from this sever.

2.1 Android Application

2.1.1 Hardware

The device provided for testing the Android app is a HTC Desire C, and the application should perform well on similar devices. The relevant hardware capabilities of the Desire C are listed below[2]:

- Screen Resolution: 320 x 480 pixels
- Processor Speed: 600MHz
- Memory: 512MB
- SD Card Storage: 4GB +
- Battery Life: 10 - 20 hours of active use
- Internal GPS Antenna

The app is unlikely to require more capable hardware than what is provided by the device because any intensive processing of data can be offset to the central server. The most that the device will need to handle is some trivial real-time graphical operations, and for that the hardware is more than adequate. Memory should not be a concern because the device has a relatively large amount for a hand-held device, and this product will not require excessive amounts of data to be stored. The app should not exhaust the battery supply of the device to a prohibitive extent, and some effort should be made during development to limit the usage of battery draining resources. The product requires the ability to find its current position using a GPS system, which is an ability of the testing device. This device is considered a very basic model, and the majority of handsets in circulation are at least as capable. If an application functions well on the Desire C, it will function well on most devices.

2.1.2 Software

The application will be written in the programming language Java using the development environment Eclipse, and will target the Android operating system version 4.0 as a minimum as specified by the client.

Java will be used because it is the language the application developers are most proficient in, and Eclipse was chosen for its superior support for developing Android applications using the Android Developer Tools plugin.

2.2 Central Server

2.2.1 Hardware

The central server program will reside on a conventional computer, and because this one machine will handle all client requests it will need much more advanced hardware than the mobile devices. This server shall have internet connectivity in order to communicate with the Android app clients. The server will also need to be almost constantly active, with downtime only occurring at times of the day when few clients will want to connect. The machine running the server requires a large upload bandwidth in order to service many clients in a small time frame, and this should be complemented by a high enough CPU and memory access speed to reduce the processing time of client requests. A lower bandwidth can be compensated for by reducing the size of data being sent to clients. The server should have enough free storage space to allow for expansion of the product to cover more area of the world. At a minimum, the server must have at least two processor cores to utilize multi-threading in the server software, 2 GB of memory to be able to process a lot of data simultaneously and also support the operating system, 16GB of available storage space to hold account and game data.

2.2.2 Software

The server will be written in the programming language C♯, and built on the .NET Framework version 4.5 using the Visual Studio 2012 development environment. C♯ was chosen for its superior performance relative to Java, and because it is the language the primary server developer is most experienced with. Visual Studio 2012 will be used for its support of .NET Framework 4.5, and many tools to aid development.

2.3 Website

2.3.1 Hardware

The web server will need to be hosted on the same machine as the game server to allow rapid transferral of data between them. This means the server will need the same hardware capabilities.

2.3.2 Software

The web server software will need to serve HTML pages with dynamic content pulled from the game server's data. Because the game server is also servicing HTML requests, they should be merged into one program. This means the software requirements are the same as for the central server.

For clients accessing the website, they will need a web browser that supports the HTML 5 Canvas element. This includes the latest versions of Chromium, Firefox, Opera, Safari and Internet Explorer. We will be using Canvas to draw game related graphics.

3 Solution Requirements

3.1 User Accounts

3.1.1 Account Registration

Type	Functional
Description	New users must be able to create an account which is stored on the server.
Priority	High
Input	The user will give a username, email address and password.
Operations	The server will check that the username and password conform to any length and composition restraints, the email is structured correctly, and that the username and email address are both unique.
Expected Results	If the given information is valid, a new account with the details entered by the user will be stored.

3.1.2 Account Activation

Type	Functional
Description	New accounts must have their email addresses verified with a verification email. The account will not be 'active' - it will not provide access to restricted resources - until the user verifies their email.
Priority	Medium
Pre-conditions	Account Registration
Input	The user will click a link on the verification email to activate the account.
Operations	The server will check to see if the account has already been activated, or if the account has expired because it had not been activated for an amount of time.
Expected Results	The account will be marked as active if the account exists and has not yet been activated. If the account is not activated within a set amount of time of the email being sent, the account is removed from the system.

3.1.3 Administrator Accounts

Type	Functional
Description	It must be possible to mark an account as being owned by an Administrator. These accounts will be authorised to perform more actions than normal user accounts.
Priority	Medium
Pre-conditions	Account Registration
Expected Results	An account marked as being an Administrator account will have additional abilities as defined by later requirements.

3.1.4 Resend Activation Email

Type	Functional
Description	A user must be able to request that the activation email is resent. This will invalidate the previously sent email.
Priority	Low
Pre-conditions	Account Activation
Input	The user will supply their email address, which is where the activation email will be sent.
Operations	The server will check that the email address is registered to an existing account that is not already active.
Expected Results	If the email is registered to an inactivated account, an activation email is sent to the user with a link to activate the account.

3.1.5 User Authentication

Type	Functional
Description	A user with an activated account must be able to authenticate themselves in order to access resources restricted to account owners.
Priority	High
Pre-conditions	Account Registration
Input	The user will give their username and password.
Operations	The server will look for an account in the database which matches the username and password provided.
Expected Results	If a matching account is found, the user is authenticated as the account owner and they may access some otherwise restricted parts of the system. Otherwise, the user will be informed that the credentials that they supplied are incorrect.

3.1.6 Account Removal

Type	Functional
Description	An authenticated user must be able to request the deletion of that account along with all personal data. This request must be confirmed by the user through a confirmation email within a week.
Priority	Medium
Pre-conditions	User Authentication
Input	The user will need to be authenticated and also click on a link in a confirmation email.
Operations	The request will only be fulfilled when the user has clicked a link in a confirmation email that was sent when they initiated the request and it has been under a week since it was sent.
Expected Results	When the user has confirmed the action, their account will be deleted from the server along with any related personal data.

3.1.7 Administrator Account Removal

Type	Functional
Description	A user authenticated as an administrator must be able to delete any non-administrator account.
Priority	Medium
Pre-conditions	Administrator Accounts
Input	The administrator will select which account they wish to delete.
Operations	The server will check that the selected account exists and the user making the request is an authenticated administrator.
Expected Results	The selected account will be removed from the system.

3.1.8 Password Recovery

Type	Functional
Description	A user must be able to have their password sent to their email address.
Priority	Low
Pre-conditions	Account Registration
Input	The user will enter their email address.
Operations	The given email address will be checked to see if it is registered to a user account.
Expected Results	If the email address does not match an account, the user is informed with a message. Otherwise, an email will be sent to the given email address containing the corresponding account's username and password.

3.2 Game Mechanics

3.2.1 Account Balance

Type	Functional
Description	Each user account must have a point balance associated with it. When authenticated, the user that owns the account must be able to view the balance at any time.
Priority	High
Pre-conditions	Account Registration
Operations	The server will check that the user is authenticated.
Expected Results	An authenticated user will be able to view their account balance.

3.2.2 Account Transactions

Type	Functional
Description	It must be possible for points to be added and removed from a user's account balance by other components of the system.
Priority	High
Pre-conditions	Account Balance
Input	The amount of points to be added or removed is specified.
Operations	A withdrawal transaction will not occur if the number of points to remove exceeds the number of points stored in the account.
Expected Results	When a withdrawal or deposit occurs, the updated amount of points in the account will be updated immediately.

3.2.3 Cache Ownership

Type	Functional
Description	A cache is able to be owned by at most one user at a time. Users must be able to see which caches are owned, and who owns them.
Priority	High
Pre-conditions	Account Registration
Input	A user makes a request to view the owner of a cache.
Operations	The server will check to see if the cache has an owner.
Expected Results	If the cache has an owner, the current owner will be displayed.

3.2.4 Cache Balance

Type	Functional
Description	A cache must have a point balance associated with it, which must be visible to authorised users, including the owner of the cache, when requested.
Priority	High
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 authorised users when requested.

3.2.5 Cache Transactions

Type	Functional
Description	It must be possible for points be added or removed from the cache by other components of the system.
Priority	High
Pre-conditions	Cache Balance
Input	When points are being added or removed, the amount is specified.
Operations	A withdrawal transaction will not occur if the number of points to remove exceeds the number of points stored in the cache.
Expected Results	When a withdrawal or deposit occurs, the updated amount of points in the cache will be visible to authorised users when requested.

3.2.6 Cache Withdrawal

Type	Functional
Description	The owner of a cache must be able to transfer points from the cache to their account when they physically visit the location of the cache. An owned cache becomes unowned if the owning user withdraws all points from the cache.
Priority	High
Pre-conditions	Account Balance, Cache Balance
Input	The user will specify how many points to withdraw.
Operations	The server will check to see if the user performing the transaction is within a given radius of the cache. The server will also check to ensure the point balance of the cache has at least the number they wish to withdraw.
Expected Results	If the user is not within a given radius of the cache or the cache has less points stored in it than the requested amount to withdraw, the user will be informed and the transaction will not occur. Otherwise, the cache and user account balances are updated after the transaction. If a withdrawal leaves a cache empty then the cache is marked as unowned.

3.2.7 Cache Depositing

Type	Functional
Description	When they physically visit the location of a cache that is owned by them or does not have an owned, a user must be able to transfer points to the cache from their account. After transferring one or more points to an unowned cache, the unowned cache will become owned by the user.
Priority	High
Pre-conditions	Account Balance, Cache Balance
Input	The user will specify how many points to deposit.
Operations	The server will check to see if the user performing the transaction is within a given radius of the cache. The server also checks to see if the user has enough points.
Expected Results	If the user is not within a given radius of the cache or the user owns less points than they requested to deposit, the use is informed and the transaction does not occur. Otherwise the cache and user account balances are updated after the transaction. If a deposit leaves a previously empty cache populated then the cache is marked as owned.

3.2.8 Cache Placement Cost

Type	Functional
Description	When placing a cache, a number of points must be deducted from the user's account balance.
Priority	High
Pre-conditions	Account Balance, Cache Placement
Input	A user attempts to place a cache.
Operations	The server will check to see if the user has enough points in their account to place the cache.
Expected Results	If the user can afford it, the cache is placed and points are removed from the placing user's account.

3.2.9 Cache Scouting

Type	Functional
Description	If a user physically visits the location of a cache owned by a different user, they must be able to view the current point balance of that cache.
Priority	Medium
Pre-conditions	Find Location, Cache Balance
Input	The location of the user is supplied.
Operations	The distance of the user from the cache is checked to ensure they are sufficiently near the cache.
Expected Results	If the user is within a given radius of the cache, they will be shown the number of points that is stored in the cache.

3.2.10 Cache Attacking

Type	Functional
Description	After scouting a cache, a user must be given the option to trigger an attack on that cache using points from their account.
Priority	High
Pre-conditions	Cache Scouting
Input	The attacker chooses how many points from their account they will attack with.
Operations	The server checks that the attacker's distance to the cache is less than a given radius, and has input a number of points between one and the number of points in their account. The server will decided which party survives the encounter, and how many points they lost in the conflict.
Expected Results	If the request was valid, an attack is initiated on the cache by that user with the specified number of points.

3.2.11 Successful Attack

Type	Functional
Description	If the attacker wins, the ownership of the cache must pass to them. All defending points will be lost, and the surviving attacking points will transferred to the balance of the cache.
Priority	High
Pre-conditions	Cache Attacking
Expected Results	The surviving points from their attack will be moved to the cache's point balance, and the cache becomes owned by the attacker.

3.2.12 Successful Defence

Type	Functional
Description	If the defender wins, the cache must remains theirs. All attacking points are lost, and the surviving defenders remain in the cache.
Priority	High
Pre-conditions	Cache Attacking
Expected Results	The surviving defenders remain in the cache, which remains owned by the defending user.

3.2.13 Point Generation

Type	Functional
Description	Points must be periodically supplied to each user based on their current performance in the game.
Priority	High
Pre-conditions	Account Balance
Input	The point allocation system will use a user's current cache number, the events of recent battles involving the user, and other relevant data.
Operations	The server will use the given information to decide how many points to give the user.
Expected Results	Each user will periodically receive points based on their performance in the game.

3.2.14 Administrator Placed Caches

Type	Functional
Description	It must be possible for administrators to place caches without needing to be at the location. These caches behave as normal, and the administrator can place it with as many points in them as they wish, including none.
Priority	Medium
Pre-conditions	Cache Attacking
Input	Administrators will specify the longitude and latitude of a new cache to place, along with how many points will be stored there.
Operations	The server will validate the location given and if the number of units placed is non-negative.
Expected Results	The cache will immediately be placed at the given location with the specified number of points in its balance.

3.2.15 Non-Player Caches

Type	Functional
Description	It must be possible for administrators to place caches which may be attacked by users, but not claimed after a victory.
Priority	Medium
Pre-conditions	Cache Attacking
Input	Administrators will specify the longitude and latitude of a new cache to place, along with how many points will be stored there.
Operations	The server will validate the location given and the point balance given is non-negative.
Expected Results	The cache will immediately be placed at the given location with the specified number of points in its balance.

3.2.16 Attacking Non-Player Caches

Type	Functional
Description	A user attacking a non-player cache must receive a number of points if they are victorious, but it will not become theirs. After the victory, the number of points in the cache balance should be reset.
Priority	Medium
Pre-conditions	Non-Player Caches
Input	Users may trigger an attack in the same was as they would on a user controlled cache.
Operations	The process for deciding the outcome of an attack on a non-player cache will take the same form as one on a user cache. Each attack is treated as a separate instance and each user will have to wait a period of time before they can attack the same non-player cache again.
Expected Results	The attacking player will receive a point reward directly to their account if they are deemed to have won the battle, and the point balance in the cache will be reset.

3.2.17 Special Event Placement

Type	Functional
Description	It should be possible for administrators to define areas by the MAC address of a nearby wireless network. This area will define a collection point for a one-time-only reward which will be limited to a given number of users.
Priority	Low
Pre-conditions	Find MAC Address
Input	Administrators will specify the MAC address of the new cache, the reward, and how many users may claim that reward.
Operations	The server will validate that the given address is a valid MAC address, and that the reward and number of users which can claim it are greater than zero.
Expected Results	A new special event area will be available for users to be notified of and claim rewards from.

3.2.18 Special Event Rewards

Type	Functional
Description	When a user enters the effective range of a wireless network whose MAC address has been designated as a special event placement, they must be able to claim a reward from it. If it is still available, they will receive points directly to their account balance.
Priority	Low
Pre-conditions	Special Event Placement
Input	A user claims a reward at a given wireless network.
Operations	The server will check to see if the reward is still available.
Expected Results	If the reward is still available, a user can claim the reward. The value is credited directly to their point balance and the special event area is removed after the reward has been claimed by a designated number of users.

3.2.19 Cache Reporting

Type	Functional
Description	Users should have the ability to mark a cache as being placed unfairly. Administrators will be alerted to the reported cache.
Priority	Medium
Pre-conditions	Cache Placement
Input	A user will select a cache that they wish to report.
Operations	The server will check that the cache is not owned by the user themselves, has not already been reported, and is not a non-player cache.
Expected Results	If the report is valid, an administrator will be alerted to the reported cache, with information including the owner, location, and who reported it.

3.2.20 Administrator Cache Deletion

Type	Functional
Description	Administrators must have the ability to delete any cache from the system.
Priority	Medium
Pre-conditions	Administrator Accounts
Input	An administrator will select a cache that they wish to remove from the system.
Operations	The server will check that the selected cache exists and that the user making the request is an authenticated administrator.
Expected Results	The selected cache will be deleted from the server and will no longer be visible to users.

3.2.21 Account Deletion Cache Removal

Type	Functional
Description	When a player account is removed from the system, all caches owned by them must also be deleted.
Priority	Medium
Pre-conditions	Cache Ownership, Account Removal
Input	The process will be triggered by a user account being removed
Operations	The server will find all caches owned by that account.
Expected Results	All caches owned by the deleted player will be removed from the system.

3.3 Application

3.3.1 Display Location

Type	Functional
Description	The application must allow the user to have displayed to them their current location in the context of a map provided by the Google Maps API.
Priority	High
Pre-conditions	Find Location
Input	The user will request for the Google Maps API powered map to navigate to their current location.
Operations	The application will check to see if the user has an Internet connection and that their GPS antenna is enabled.
Expected Results	If the user has an internet connection and their GPS antenna is enabled, the application will show the user's location on a map provided by the Google Maps API.

3.3.2 Nearby Caches

Type	Functional
Description	The application must be able to show the user, on the map, the locations of caches near to a specified location.
Priority	High
Pre-conditions	Find Location, Display Location, Server Connectivity
Input	The user will request to see the locations of caches near to a specified position.
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 specified position.
Expected Results	The application will mark on the map, provided by the Google Maps API, all caches near to the specified position.

3.3.3 Map Zooming

Type	Functional
Description	The application should allow the user to view a larger or smaller area of the map.
Priority	Medium
Pre-conditions	Nearby Caches
Input	The user will be able to specify the 'zoom level'.
Operations	The application will ensure that the specified zoom level is within defined limits, and if so will request from the server a list of caches within the portion of the map that is currently visible.
Expected Results	The map is displayed at different levels of zoom, as specified by the user, along with any caches that should be visible at that level of zoom.

3.3.4 Path Finding

Type	Functional
Description	The application should allow the user to view a path between the user's current position and a target cache they specify.
Priority	Low
Pre-conditions	Find Location, Display Location, Nearby Caches
Input	The user will select which cache they want to navigate to.
Operations	The application will use the Google Directions API to find a path to the target cache.
Expected Results	A path will be drawn on a map to show which route to take.

3.3.5 Cache Placement

Type	Functional
Description	The application must allow the user to request the placement of a cache at their current location.
Priority	High
Pre-conditions	Find Location, Server Connectivity
Input	The user will trigger a request to place a cache at their current position.
Operations	The server will validate the request to place a new cache.
Expected Results	The application will send a cache placement request to the central server, and if it is successful all users will be able to see the new cache.

3.4 Peripheral Functionality

3.4.1 User Communication

Type	Functional
Description	Users may be able to send messages to other users.
Priority	Low
Pre-conditions	User Accounts
Input	An authenticated user will specify the message subject, and the message to send.
Operations	The server will check that the subject and message body are not empty.
Expected Results	A message will be created which is visible to all intended recipients.

3.4.2 Communication Reporting

Type	Functional
Description	Users should have the ability to report communications sent between users as being inappropriate. Administrators will be alerted to the reported communication.
Priority	Low
Pre-conditions	User Communication
Input	The user will specify which message to report.
Expected Results	An administrator will be alerted to the reported message, including information such as the sender, message content and who reported it.

3.4.3 Activity Recording

Type	Functional
Description	Any game activity performed by a registered user should be recorded and can be viewed by that user when requested. Users will also be able to view activities by other users which have a direct effect on them.
Priority	Medium
Pre-conditions	User Accounts
Input	A user can request the activities that have occurred since a specified time and date.
Operations	The server will locate all activities related to that user since they time they gave.
Expected Results	Any activities that occurred since the given time and date will be displayed to the user.

3.4.4 Website

Type	Functional
Description	There must be a website that is accessible by users.
Priority	Medium
Pre-conditions	User Accounts
Input	A user will make a request to access a certain page of the website through a web browser.
Operations	The server will provide the requested page if it exists.
Expected Results	A user will be able to view a requested web page if it exists.

3.4.5 Website Authentication

Type	Functional
Description	A user must be able to authenticate themselves to their account through the website to be able to access restricted content on the site.
Priority	Medium
Pre-conditions	Website, User Authentication
Input	The user will give their username and password.
Operations	The server will look for an account in the database which matches the username and password provided.
Expected Results	If a matching account is found, the user is authenticated as the account owner and they may access some otherwise restricted parts of the website. Otherwise, the user will be informed that the credentials that they supplied are incorrect.

3.4.6 Viewing Owned Caches

Type	Functional
Description	A user, when authenticated on the website, must be able to view a list of all caches that are currently owned by them, and details about each one, including the number of points stored inside and its location.
Priority	Medium
Pre-conditions	Website Authentication, Cache Ownership
Input	A user will request to view the web page containing a list of the caches they own.
Operations	The server will ensure the user is authenticated.
Expected Results	If the user is authenticated, a web page containing a list of all caches owned by that user's account is displayed.

3.4.7 Viewing Cache Details

Type	Functional
Description	On the website, an authenticated user must be able to view information about a specified cache. This information will include a graphical representation of the location of that cache on a map. If the cache is owned by the user that made the request, further information such as the number of points in the cache and a history of attacks made against the cache while it was owned by that user are also displayed.
Priority	Low
Pre-conditions	Website Authentication, Cache Ownership
Input	A user will request to view a web page containing additional information about a specified cache.
Operations	The server will check that the user is authenticated, the specified cache exists, and if currently owned by the user.
Expected Results	A web page with details about that cache will be displayed, including a graphical representation of the cache's location on a map and the current owner. If the cache is owned by the user that made the request, additional information will be displayed including the number of points in the cache and a history of its recent events.

3.4.8 Overview Map

Type	Functional
Description	An authenticated user must be able to view a map, using the Google Maps API, which displays a subset of all caches. The user will be able to specify a filter to decide which caches are displayed. The user should also be able to use this map to select a cache to view information about it.
Priority	Medium
Pre-conditions	Viewing Cache Details
Input	A user requests to view a web page containing the overview map using a specified filter.
Operations	The server will ensure that the user is authenticated. If they are authenticated, the server will find the collection of caches that matches the specified filter.
Expected Results	If the user is authenticated, they will be provided with a web page containing the overview map. The map will only contain the caches that match the given filter, and will allow the user to select caches on the map to access the cache details page for that cache.

References

[1] HM Government Publishing, *Data Protection Act 1998*.
`http://www.legislation.gov.uk/ukpga/1998/29/contents` Accessed November 2012

[2] HTC Corporation, *HTC Desire C Specs*.
`http://www.htc.com/uk/smartphones/htc-desire-c/#specs` Accessed November 2012