**GUDE Assessment**

Section 1 35%

**1) Explain concept model for the application, explicitly describing how the domain and the locational/geocaching model is used**

2) **Identify usability concepts**

**3) Operationalise usability concepts - how to measure these concepts**

4) **Define task allocation across the systems**

**Write in 3rd person / we actively**

**Getting Usability into Development Environments**

**Section 1**

**The Purpose of the System**

The domain for this application is language learning. The aim is that users will be able to learn languages contextually via geocaching and memory techniques with flashcards. To make the process of language learning more efficient, the app will allow users to unlock vocabulary in their target language based on locations they visit. This idea allows the user to more efficiently learn the words that they will regularly come across in daily life.

The user will also gain experience points for each place they visit, encouraging them to learn more if they want to and therefore visit more places and gain more caches. As users study, they will also gain experience points so that users who study more gain more experience and higher levels. Certain rarer locations that will contain harder words will be level locked, so that only users who have gained enough experience points can unlock and learn them.

The overall purpose for the user is to be able to learn languages in an efficient yet fun way, as if the entire learning process was a game.

Within the app they should be able to:

* Create, edit and view account/profile.
* Use a map to find local geocached vocabulary flash card sets.
* Learn and review individual and combined sets of flash cards.
* Edit and delete seperate flash cards in case of issues with difficulty.
* View their level and experience points.
* View history of completed geocaches (possibly use some form of achievement system).

The target users could be anyone who wants to learn a language and as such could be of any age. For usability testing purposes people between the ages of 18-50 will be selected for prototype testing.

**How Caching is Used in This System**

The app will display a list of categories of locations, such as “Train station” or “University”. The user can then click on one and a list of locations will appear along with coordinates and distance from the user. If the user clicks on one of these options then more information such as a map and information about the words they can learn will appear. The user then heads to the location using the information supplied by the app. When they enter the radius of the geocache, a notification will automatically appear on their phone and will tell them that they have gained a new set of cards and some experience points. This will then get updated to the user’s profile and synced to the cloud. These updates will then be sync across devices when the user logins in (on a device).

**Usability Concepts**

Usability needs to be a high priority as users are expected to be of varying ages and will be using a variety of different devices. The app should be easily usable to a point that the user does not make to many mistakes to get to a certain point or to do a certain task. Jakob Nielsen has invented several usability evaluation heuristics to help improve the usability of systems fast and cheaply, in hope to solve issues like these for software designers. He has 10 heuristics on usability for User Interface Design which will be used in this system, along with suggestions from other experts, to identify and measure usability.

This system has two main concepts. Find new flashcard decks via geo-caching, and studying found flashcard decks. Having two main concepts means that the system should be designed in a way that keeps the user informed as to what they are doing all the time, so that they do not get confused. Nielsen (1995, para 2) states that users should always be kept informed about what is going on via appropriate feedback and within reasonable time. For this system this includes such things as confirmation messages, possible loading screens and possible help documentation. If documentation is included then Nielsen (1995, para 11) also suggests that it should be easy to find, focused on what the user’s task, a simple list of instructions and not too large. This system will be designed in a way that should mean that help and documentation is made redundant, however, as an added extra it can’t hurt the user if it is added in.

When considering content in the app, the terms and language that is used must be simple to the user rather than complex system terminology as Nielsen (1995, para 3) states. For example, mentioning “caches” in a geo-caching app is not always that helpful, as most users may not realise what geo-caching is or even what a cache is. For simplicity, it is better to use phrases that any user, new to the subject area or not, will be able to understand. In this systems case there is a good chance that if a user is learning a language, they may wish to change their display language. This will make the app much harder for them to use, but lots of people do this to help learn languages. Due to this, the system must use simple language with a combination of buttons and images. This also has an effect on how they will use the app as well and as such will affect the order in which information will appear. Content will need to be displayed in a logical order to prevent confusion.

As Nielsen (1995, para 5) says in “Consistency and standards”, you should not confuse users by using different terminology, situations or actions when they actually mean the same thing. This is also suggested by the Android User Experience Team (n.d., para 13) where they say that if something looks the same then it should always work the same. Consistency across the system is important to prevent confusion for users.

The app should have an easy learning curve and should be very simple to cater for all ages. Shneiderman (n.d., para 10,11) states in his Heuristics that reducing short-term memory load is a must as you want to avoid interfaces where users must remember information between different displays. Now, this will be in that GUIs to certain extent as the design of the flashcard quiz requires users to have to remember information. However, this does not mean that use of the rest of the system should be taxing on the user. By reducing short-term memory load in this way throughout the rest of the system, users should find learning new words or grammar rules when studying a lot easier. This is because they will have less things to remember, meaning they have more potential to learn more.

**Operationalise usability concepts**

In Usability Metrices (2001, para 5) Nielsen lists the most basic measures of usability which he lists as follows:

* Success rate (can users actually perform the task?)
* Time taken to complete a task
* Error rate
* User satisfaction

He also mentions other possible metrics such as the amount of times users need to backtrack to find the correct window/page. He then goes on to discuss comparing two designs and how to quickly tell if a new design has improved in usability. He recommends taking tasks that the user has to do then to record how long it takes the user to complete the task.

To do this you would gather the data for each design first then, for each one, add up the time taken so you get what he calls a “how long it takes users to do stuff” for each design. From here you can simply calculate a percentage improvement based on the time difference. He also means though that this can be misleading as if certain tasks are not performed as often but improve in efficiency then these tasks could skew the results. When some tasks are performed more than others, it is better to work out percentage of improvement separately first and then get a geometric mean of the tasks percentages. By doing this you then get a fair “usability score” which can tell you just how higher or lower usability is for a new design.

At first low fidelity prototypes will be created. Then, following this simple but effective method for measuring usability, the prototypes will be tested with sample users and results will be recorded. During the testing the users will be given set tasks and the time taken to complete these tasks will be measured. After this process is complete, analysis will be done on how to improve the results of the tests that have been conducted. This should be done with some form of control test results to give some idea as to whether a task has taken a long time or not. This may be hard to do as the majority of software is very different and has various tasks, therefore the likeliness that a “standard” time for tasks exists. In this case, simply making two different low fidelty prototypes is probably the best solution. Once the most efficient solution has been found, the high fidelity prototypes can be created and also tested. Once a high fidelity has been made, tests for tasks will be conducted and results recorded. Changes will then be made and updates to the prototype to increase usability will be considered, implemented and the system will then be tested again.

* *learnability*
* *efficiency*
* *memorability*
* *errors*
* *satisfaction*

*utility is also very important*

* *Definition of Utility = whether it provides the features you need.*
* *Definition of Usability = how easy & pleasant these features are to use.*
* *Definition of Useful = usability + utility*

*Consistency*

*The system needs to support:*

* *human cognitive architecture*
* *human perceptual architecture*

**Task Allocation**

The following table shows the tasks that will be performed throughout the system. Each task will be caused by an action caused by user input or due to an action in the system itself.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Task | Mobile | Desktop | Server | User |
| Look up possible caches | Yes | Yes | No | Yes |
| Check achievements | Yes | Yes | No | Yes |
| Sync data across devices | Yes | Yes | Yes | No |
| Store data | Yes | Yes | Yes | No |
| Check map | Yes | Yes | No | Yes |
| Review flashcards | Yes | Yes | No | Yes |
| Obtain flashcards from cache | Yes | No | No | Yes |
| Obtain experience points from cache | Yes | No | No | Yes |
| Get data from google maps (e.g. whether the current location is a shop, bank, park etc) | No | No | Yes | No |
| Sign in/Create account | Yes | Yes | No | Yes |
| Create/edit account info | Yes | Yes | No | Yes |
| Change display language | Yes | Yes | No | Yes |
| Change learning language | Yes | Yes | No | Yes |
| Authorise account/sign in | No | No | Yes | No |

Section 2 45%

1) Initial Interface Specification

2) Low fidelity prototypes for both Mobile and Desktop applications

3) Quick and dirty initial empirical evaluation

4) Interface Specification modification

5) High fidelity prototypes for both Mobile and Desktop applications (must use Visual Basic for the Desktop prototype)

6) Quick and dirty empirical re-evaluation

7) Interface Specification modification

**Section 2**

*Describe the process of creating the prototypes*

**Initial Interface Specification**

The Android User Experience Team (n.d., para 4) states that “Real objects are more fun than buttons and menus” and go on to say that allowing users to directly touch and manipulate objects in an app reduces cognitive effort needed to perform tasks and is also more emotionally satisfying. They suggest that this is better than traditional buttons and menus. As they suggest that using this technique “reduces cognitive effort”, the system should be designed in a method that implements this well so that the user can easily use the app for learning. Reduced cognitive effort when it comes to using the app will make it easier for the user to concentrate during actual learning.

The Android User Experience Team (n.d., para 7) also observe that using “short phrases with simple words” is very important as “people are likely to skip sentences if they’re long.” This is an incredibly simple rule but is important to remember when writing any form of explanation on how to use certain features. Small sentences, especially in a foreign language, are easier to read and understand than longer sentences. As the system is aimed at getting users to learn foreign languages with individual sentences flashcards (gained from caches) this rule should also be taken into consideration for the apps cache content.

In the next paragraph the Android User Experience Team (n.d., para 8) also claim that “pictures are faster than words”, telling us to “consider using pictures to explain ideas” as “they get people’s attention and can be much more efficient than words.” This guideline is a important as it is true that an app with the right balance of text and pictures will be more easily understandable than just an app with pure text. Take an icon button for example. Using a clearly understandable icon in place of a text button could make it quicker for the user to understand the function of the icon button. This is something to definitely consider when thinking about menu items, flashcard content and especially for the locations of caches. Based on this guideline I think that a visual representation of cache locations, such as an image of a map, will be a better idea than explaining locations via text. Taking into consideration the first guideline, mentioned earlier from the Android User Experience Team, an interactive map would be even more beneficial for the user.

There will not be much security besides on the server itself so potentially the user’s data could be modified. Therefore what private data is stored by the user in their account settings will need to be considered.

The app could have a huge variety of accessibility issues, especially for the mobile version. There are many different types of disabilities out there and many are affected by "small screens", for example, partial blindness. In this case the user would prefer a large screen to see the app on and may own a tablet for this purpose. However, if this app is not optimised for accessibility and the buttons are still small or the text is just as small as a normal mobile version, then the user will have a hard time using it.

I don't think the app will have any environmental issues.

Section 3 20%

1) Critical evaluation of, and reflection on your process, and the role of prototyping in empirical evaluation.

We expect you to upload

(as a single archive in ZIP format, named with your student ID, max size <100Mb)

1) Report

2) Software developed (demonstration in class)

3) Evidence of paper prototypes and evaluation

Works Cited

"10 Heuristics for User Interface Design: Article by Jakob Nielsen." *10 Heuristics for User Interface Design: Article by Jakob Nielsen*. Web. 01 Mar. 2017.

"Android Design Principles." *Android Developers*. Web. 27 Feb. 2017.

"Ben Shneiderman." *Ben Shneiderman*. Web. 01 Mar. 2017.

"Ben Shneiderman." *Ben Shneiderman*. Web. 03 Mar. 2017.

"Usability Metrics." *Usability Metrics*. Web. 03 Mar. 2017.