VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

FIREFOX OS APPLICATION FOR LEARNING LANGUAGES

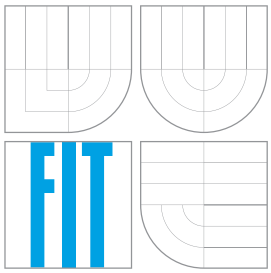
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAKUB CHUDÍK

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

APLIKACE PRO VÝUKU JAZYKŮ PRO FIREFOX OS

FIREFOX OS APPLICATION FOR LEARNING LANGUAGES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAKUB CHUDÍK

VEDOUCÍ PRÁCE
SUPERVISOR

Prof. Ing. TOMÁŠ VOJNAR, Ph.D.

BRNO 2015

Abstrakt

Tato práce se zabývá vytvořením aplikace pro výuku jazyků specificky pro mobilní operační systém Firefox OS. Vzhledem k své povaze, uživatelské rozhraní aplikace se snaží uspokojit ergonomické potřeby aplikací určených pro kapesní zařízení. Aplikuje několik konceptů gamifikace ke zlepšení procesu učení, jehož výsledky jsou prezentovány a vyhodnoceny. Aplikace také přináší své vlastní jedinečné vlastnosti, které jí pomáhají vyniknout mezi ostatními aplikacemi pro výuku jazyků.

Abstract

This thesis deals with creating a language learning application specifically for the Friefox OS operating system for mobile, handheld devices. Because of its nature, the application's user interface attempts to cater specifically to the ergonomic needs of applications for handheld devices. It applies several concepts of gamification to improve the language learning process, the results of which are presented and evaluated. The application also brings its own unique features to make it stand out among existing state of art language learning applications.

Klíčová slova

Firefox OS, Webové technologie, Uživatelské rozhraní, Gamifikace, HTML5, CSS3, Javascript, Výuka jazyků

Keywords

Firefox OS, Web technologies, User interface, Gamification, HTML5, CSS3, Javascript, Language learning

Citace

Jakub Chudík: Firefox OS Application for Learning Languages, bakalářská práce, Brno, FIT VUT v Brně, 2015

Firefox OS Application for Learning Languages

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Prof. Ing. Tomáše Vojnara, Ph.D. Další informace mi poskytl technický konzultant Ing. Martin Stránský z firmy Red Hat Czech. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jakub Chudík
May 20, 2015

Poděkování

I would like to thank my supervisor, Tomáš Vojnar, for his patience, support and constructive criticism regarding the thesis. I would like to thank the company Red Hat Czech for the opportunity to do a thesis on this topic and Martin Stránský for invaluable technical advice and guidance on the project. Lastly, I would like to thank my family for their emotional and financial support – I wouldn't have made it this far without them.

Contents

Chapter 1

Introduction

Firefox OS is one of the latest commercially developed operating systems for smartphones and tablet computers. It was designed as a community-based open-source alternative system for mobile devices that utilizes HTML5, JavaScript, and novelty features such as open web APIs that communicate directly with cellphone hardware [5]. In example, this enables Firefox OS users to use apps without having to install them beforehand. As a relatively recent addition to the smartphone market, Firefox OS is not yet widely available or a popular of operating system for smartphones, but the aforementioned features help to make it stand out among its competition and what ultimately makes or breaks a new mobile platform these days is not the hardware but apps available for the operating system.

The purpose of this bachelor's thesis is to create a language-learning application for Firefox OS. There is a multitude of reasons why such application would be beneficial specifically for this operating system. According to telecommunications companies [13] smartphones with the Firefox OS are most well accepted in the emerging markets of developing countries but perform poorly in the United States. It is explained by their relatively low price thanks to utilizing cheaper hardware that doesn't really compare to today's flagship phones but makes the smartphones more accessible for such economies, as well as viable first-time smartphone owners. Seeing that developing countries are mostly populated by non-native English speakers, it is reasonable to assume that a language-learning application for this particular operating system would be much more helpful than a similar application for other operating systems which are more popular in the West. Ultimately, there's also the fact that currently there are no sophisticated language-learning applications available for the Firefox OS in the Firefox Marketplace and here's hoping that the application produced in this bachelor's thesis will provide a solid freeware alternative.

Chapter 2

Language Learning

This chapter introduces popular and acknowledged existing language-learning software, both commercial and freeware, focusing specifically on their applications for smartphones. These applications, namely *Duolingo*, *Rosetta Stone* and *busuu*, will be used for comparison with the application produced in this thesis. Additionally, it will explain some of the most important terms and techniques with regards to language-learning, such as the forgetting curve, spaced repetition, learning curve and gamification of learning.

2.1 Learning Software Comparison

2.1.1 *Duolingo*

Duolingo is freeware language-learning software. It differentiates quite substantially from other language-learning applications due to the fact that it also serves as a crowdsourced text translation platform. As users progress through the lessons, they simultaneously help to translate websites and other documents. This allows *Duolingo* not to charge students to learn a language. Organizations and businesses provide *Duolingo* with content in need of translating and the users of *Duolingo* are invited to translate these documents and vote on translations. Another unique feature is „The Language Incubator“ which is a tool that allows the community to build new language courses. Volunteers who wish to add a specific language to *Duolingo* can do so if sufficient interest in contributing is detected. After they fully prepare a course it enters the Beta phase which is, once again, open to the public. In the final phase, the Beta testing is considered over but users can still report wrong questions or misleading answers and *Duolingo* allows the volunteering contributors to tweak the course further to fix such issues. Much like other modern learning applications, *Duolingo* utilizes a gamified and a heavily data-driven approach to learning. Language lessons are divided in a „skill tree“ fashion, through which the users can progress to learn new words, which can be then practised in the vocabulary section. Users have limited „lives“ that are lost when they make a mistake, and they earn „experience points“ as they advance through the course. After they complete all lessons associated with it, they earn a particular „skill“. Mistakes made by the user are aggregated and analysed, so that the system can learn and adapt to help the user with the specific parts of the language they find difficult. The main drawback of *Duolingo* compared to other language-learning software is that the lessons are almost exclusively text based. As a result it goes rather light on usage of stock photos for learning and it offers much less speaking practice.

Duolingo is available on the World Wide Web as well as a smartphone app for the iOS,

Android and Windows Phone 8.1 platforms. It is considered by the press to be the best free language-learning app available, completely without charge for additional lessons. A study conducted with native English speakers learning Spanish compared the writing ability of the users of *Rosetta Stone* and *Duolingo* found that *Duolingo* is nearly twice as effective in this regard [15]. Speaking ability was not measured, since *Duolingo* is rather lacking in this area.

2.1.2 *Rosetta Stone*

Unlike *Duolingo*, *Rosetta Stone* is a proprietary language-learning software. It has been released in multiple versions for more than a decade. *Rosetta Stone* itself is available exclusively as a computer-based product on the OS X and Windows XP plus later platforms. However, since Version 4 it has a *Rosetta Stone* TOTALe Online subscription service that offers unlimited Web and app access to all lessons for a specific language. The TOTALe software suite that comes with the service is somewhat chaotic to navigate. The application designed specifically for mobile devices is the TOTALe Mobile Companion, available for iOS and Android devices. It is however only an inferior variant of the *Rosetta Stone* software and it is limited mostly to listening to and repeating sentences. The iPad and Nook exclusive Language Training application (formerly known as Rosetta COURSE) offers lessons identical to the computer-based *Rosetta Stone* program, along with all of its unique features described in the next paragraph. Finally, there's TOTALe Studio HD, an iPad exclusive app for pre-scheduled video chats with native speakers.

Rosetta's trademarked approach to language-learning is called „Dynamic Immersion". It utilizes a combination of images, text, audio and video to teach words, sentences and grammar by spaced repetition completely without any translation. In a typical exercise, the student gets to either hear a sound or see text in the foreign language they're learning. They are then provided with a variable number of images on screen from which they are supposed to pick the correct option. In example, a native speaker makes a statement that describes one of the pictures and based on the context the student choose the one that is associated with the statement the most. In more advanced courses, the student is instead given a picture and they're supposed to complete or give a textual description of the photograph. As an additional feature, the software is also capable of evaluating word pronunciation, if the user has a microphone. It is useful as it compares the student's flawless imagined accent that comes from repeating sentences in their head with their actual accent in reality. The voice recognition function has trouble operating in crowded spaces though, diminishing its benefit for mobile applications. *Rosetta Stone* uses a mundane scoring system for its user's progress in a language course.

In terms of reception from both the media and language experts, *Rosetta Stone* is notable for having over 34 language courses with various degrees of learning depth, including courses for endangered languages. Its institutional usage (in example, a special military version of Arabic courses from *Rosetta Stone* is available to all United States Army personnel) crowns *Rosetta Stone* as the most popular commercial language-learning software. Criticism is focused on overusing the same stock photos, with only four different picture sets designed to be „culturally relevant" towards specific groups of language courses. It has also been for its lack of a pedagogical foundation and focus on marketing to create an economically viable product.

2.1.3 *busuu*

In many ways, *busuu* is kind of the middle road between *Duolingo* and *Rosetta Stone*. *busuu* is described as a social network for learning languages. It is neither freeware, nor proprietary software. Its business model is described as freemium. It offers courses in twelve languages which are initially free but the user must subscribe to get a time-limited access to further, more advanced lessons. Similarly, the mobile apps offered by *busuu* are free to download with a set of basic learning units with additional content available for purchase. Subscribers get all mobile app content as well. *busuu*'s applications are designed for iOS and Android devices.

Much like *Rosetta Stone*, *busuu* offers audio-visual courses. The courses are not based on arbitrary skill trees or levels, but an actual guideline for foreign language learners – the Common European Framework of Reference for Languages (CERF). Study material is divided into units. Each unit consist of multiple-choice questions, speaking and writing assignments and multimedia material. Similarly to *Duolingo*, the learning process is also affected by the community. Users can act as both student and tutor, with the ability to correct another's work, or converse via chat, or a audio or video connection. The program is overall received as well-designed pedagogically with its alternating between different kinds of exercises and the ability to be corrected by fellow *busuu* users who are native users. In practice, however, less than half requests receive a request from the community. Another stand out feature of *busuu* is more of a convenience than a language-learning speciality. *busuu* downloads its exercises to your mobile device in advance, which can be useful for studying in areas that lack reliable Internet access or any access at all.

2.2 Learning Concepts

2.2.1 Forgetting curve

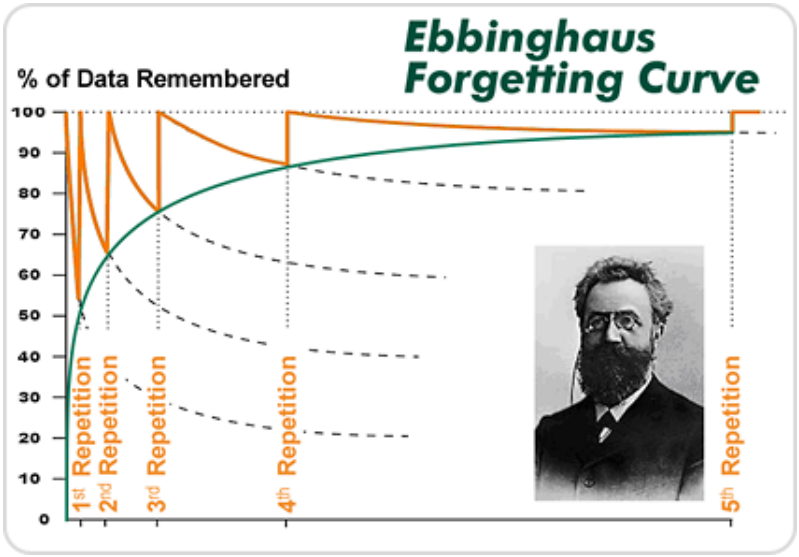


Figure 2.1: Depiction of the forgetting curve from [7].

In the field of psychology, the forgetting curve is a term that refers to a hypothesis for the decline of memory retention over time. The hypothesis was first extrapolated by

Hermann Ebbinghaus in 1885. It describes how the ability of the brain to retain information decreases as time progresses. A plot of a typical forgetting curve (shown here 2.1) purports to show that at the beginning, the exact point when you actually learned a particular piece of information, retention is exactly 100%. As time goes on the retention drops sharply down to around 50% in the first couple of days. The forgetting curve is exponential. Half the memory of newly learned knowledge is lost in matter of days and as time goes on, the forgetting continues but at a rate that is much, much slower. This shows the importance of reviewing learned material. It also supports transience, the process of forgetting that occurs with the passage of time, from one of the seven kinds of memory failure.

The conclusion drawn from the effects of the forgetting curve are that each repetition in learning increases the optimum interval before the next repetition is needed. Spending time each day to remember information greatly decreases the negative effects. This essentially changes the shape of the curve for the repeated information. Another related concept is the strength of the memory, which affects the durability a memory traces in the brain. The stronger the memory, the longer can a person recall it. Ebbinghaus asserted that the best methods for increasing the strength of a memory are either using mnemonic techniques to better represent the memory or repetition based on active recall, which leads us to the concept of spaced repetition.

2.2.2 Spaced repetition

Spaced repetition is a learning technique that incorporates increasing intervals of time between subsequent review of previously learned material in order to prevent the negative effects of the forgetting curve. It exploits the psychological phenomenon known spacing effect, whereby humans remember and learn more easily if information is studied a few times over a longer timespan rather than repeatedly in a short timespan (also known as „cramming“ or massed presentation). The long-term effects of spacing have been specifically assessed to have a major, positive impact in the context of learning a foreign language. Since the 1960s, multiple systems, such as the Pimsleur method for language-learning or the all-purpose Leitner system, which use flashcards, have been devised. Flashcards were simple pieces of paper with a word on one side and its translation on the other.

Naturally, with the advancements of personal computers, this advanced into modern spaced repetition systems for computer-assisted language learning and software-based solutions [3]. While staying true to the original flashcard models, these solutions allow for automated scheduling and most importantly automatic statistic gathering where the software is able to adjust its spacing intervals for specific questions that the student found difficult. In example, every time the student is unable to produce a correct response, the material is considered harder than the rest and will appear more often than the the subjectively easier questions that the student had no problem with. Other beneficial effects of computer-assisted spaced repetition are automatic question-answer plus meaning generation (data has to be put in only once), availability of additional information such as examples in sentences and most importantly audiovisual alternatives to written pairs. The aforementioned *Duolingo* language-learning software makes great use of the spacing effect.

2.2.3 Learning curve

The learning curve represents graphically the increase of learning with experience. Once again, it was first described by Hermann Ebbinghaus. The term itself is used when the same task is repeated over and over again in a series of tests, or for learning a body of knowledge

over time. The learning curve is closely related to the experience curve effect. Experience shows that the more times a task is performed, the less time and effort is required on each subsequent repetition of the task [14]. In a broader sense, the learning curve is used not only for learning itself, but also things such as the natural limits for resources, production costs or technologies in general.

In terms of language-learning, the learning curve supports a collection of methods that support learning at a higher rate without reduction of understanding the material and without negatively affecting the memory retention. The most relevant method in this case is visualisation of the learned knowledge. It purports that new foreign language terms can be learned more easily if the foreign term is paired with a picture, which the term describes, rather than the classic term-translation pair. Such approach also enables another method to enhance the speed of learning – removing the context in the user's native language and letting them get through the test purely on the context, or simply trial and error (referred to in in *Rosetta Stone* as „Dynamic Immersion").

2.2.4 Gamification

Gamification is a term used to describe the usage of video game mechanics and game thinking in a non-game context [16]. Broadly defined, gamification defines and takes the elements from games that make them fun and motivate the players to continue playing and uses these elements in a different kind of software in order to influence said software's users to behave similarly. Gamification has been studied and successfully applied in several domains and the majority of reviews and studies on gamification found it to have positive effects. It is argued that video games themselves often do a better job of teaching than de-contextualized, skill-and-drill instruction. Students can use educational games to engage in difficult tasks without embarrassment when they fail, and teachers can use educational games to build problem-solving skills and help students see the meaning in their lessons. In the context of education, gamification can potentially influence the students' focus on meaningful learning tasks, attendance or even them taking the initiative when it comes to learning [11]. Gamification is not to be confused with game-based learning or outright educational games, where the learning agenda is encompassed within the game itself. Gamification methods don't even necessarily demand to be in some sort of software and can be applied to a real life education process (in example, even a classroom that incorporates these methods can be considered „gamified") [10].

In terms of language-learning software, *Duolingo* makes the most out of gamification elements in their approach to learning. The elements relevant to this field (explained on the *Duolingo* example) are:

- **progress mechanics** – the users progress is being tracked all the time with „experience points" that increase linearly as the user invests more time in the application (and answers the questions correctly) and „skills" that are awarded for when the user completes all tasks associated with it. This does not actually differ much from other language-learning software that also awards the user with points for completing tasks or even certificates when they fulfil the specific requirements. Such things are almost a necessity for computer-assisted learning to begin with. Although it's basically the same system, the difference in the narrative, simply giving the reward system a new, glorified name and directly presenting it to the user has a positive impact compared to hiding these statistics away or treating them normally.

- **player control** – the „skill trees“ in *Duolingo* branch at certain points. This gives the user the illusion of choice where they can progress the way they want rather than the way the application expects them to go. Ultimately, they have to go through the entire tree to advance to the next level but with this approach the user feels less rail-roaded and less forced to learn specific aspects of the language that they perhaps would not wish to at that moment.

- **immediate feedback** – the users immediately lose „lives“ for giving a wrong answer and gain „experience points“ for giving the correct one. The constant risk and reward can make the learning process potentially addictive. On its own this can have a negative effect where it rewards the user for „meta-gaming“ such as doing menial tasks over and over again to get their instant gratification with minimal thought input and they learn nothing new. However, when other elements of gamification are incorporated as well, it can have a huge impact on the user's motivation and their willingness to keep using the application in future as well. Much like a video game, this can engage the user for hours more than standard, mundane learning software would.

- **scaffolded learning with increasing challenges** – as mentioned beforehand, *Duolingo* has multiple levels, each dealing with increasingly complicated parts of the language being taught in them. Its data-driven approach that analyses the user's mistakes also allows it to increase the difficulty by focusing more on the parts of the language that the user had issues with in the past and such questions are bound to occur more often due to the spacing effect.

- **opportunities for mastery, and levelling up** – the levelling up part happens obviously when the *Duolingo* user progresses through the actual levels into which the language and its „skill trees“ are divided into. Its „skills“ are the representations of actual mastery. Even before the user gets on a specific level (from an actual language proficiency perspective – as in beginner, intermediate, native, etc.) they can get these achievements as rewards for mastering specific parts of the language. It can be in example a reward for completing the vocabulary section or the grammar section and so on. This can be used to differentiate users on the same level proficiency-wise giving them still a sense of achievement for having accomplished something that others did not.

- **social connection** – possibly the most subtle but also the most important part of *Duolingo* as well as the most resonating gamification method. It strives to leverage people's natural desire for socializing, self-expression within a group and competition. *Duolingo* offers its users the option of creating a profile that not only saves their progress (if using the Web application rather than the smartphone app) but also to upload their results, progress, acquired „skills“ and so on to the Internet and actually compare them with other users. This makes their achievements visible and can potentially encourage the user to compete with others that they know who also share this information about them. In a broader sense, it is potentially problematic as it can consequently lead to unethical behaviour, low cooperation, collaboration or even disadvantaging certain user demographics but such issues are relatively a non-factor in language-learning applications as the first and foremost goal for the user is still their own betterment in the foreign language, for practical reasons. *Duolingo*'s

community driven and created language courses also cleverly take advantage of this particular gamification method, as the volunteers and contributors are credited (along with the percentage of work they did in comparison to others) and visible for all other users to see on the language course itself.

busuu's social network approach to learning is also considered an element of gamification – giving an opportunity for collaborative problem solving, something that can potentially freshen up the learning process but it relies heavily on the willingness and the participation of other users.

Criticisms of gamification methods claim that the most popularly implemented strategies are not actually fun and merely create an artificial sense of achievement. Gamification was described as a „populist idea that actually benefits corporate interests over those of ordinary people“ [2] as it exploits human psychology to first and foremost keep the users invested into the software. As pointed out in the previous paragraphs, it can also encourage unintended behaviour. Nonetheless, from a purely computer engineering standpoint, it is overall beneficial to implement at least some of these methods as it certainly helps the software to stand out and survive on the app market, especially when it comes to independently developed language-learning software without a huge marketing budget to promote it or without professional pedagogic experience behind the design of the learning process.

Chapter 3

Application Design

This chapter describes the actual design of the application produced in this bachelor's thesis. Many of the design choices rely on the language-learning methods described in the previous chapter and the overall intention is to create an application that is inspired by the already mentioned popular mobile applications for language-learning, taking „the best of all worlds“, whilst also contributing with something further, something of its own.

3.1 The Learning Process and Analysis

3.1.1 Language Division

One of the biggest assumptions that the application makes in regards to languages that are available for learning is that the language can be split as follows:

- **Language Levels** – the levels represent big milestones achieved by the user in progress of learning the language. A level can be divided into multiple sections. Once all of the sections at a certain level are finished, the user proceeds to the next level.
  - **Level Sections** – a section represents some particular skill required to master the language, at the current language mastery level. Sections are independent by default, so you can proceed in multiple sections at once. However, if necessary, sections can be set to be dependent on one another. A section contains multiple questions, which usually share at least one common grammatical tag.
    - **Section Questions** – self-explanatory. Questions can be either purely textual, or they can come with images that visually represent the foreign language term that's being tested.

3.1.2 Evaluating Language Skill Level

As soon as the user makes their decision as to what language course they want to study, they proceed directly by jumping into a test. The point of this test is to evaluate the user's current proficiency with the language.

The alternative, letting the user rate themselves is a poor design choice. It is highly unlikely that the average user is able to correctly ascertain their skill level and it may lead to over- or underestimating themselves.

Another important point of the introductory test is that the user does not actually know that it's assessing their initial rating, as it is not mentioned to them. This is to prevent

dishonest behaviour where the user may be inclined to use outside help with the test if they know about its importance. The intention is for them to assume that this is just how the application always works like. The test starts with the most basic and the easiest questions available in the language for the first few tasks. At some points, however, the difficulty may jump to a much higher level, even relatively early in the test. This is to check whether the user is significantly more skilled than a user who might have had more trouble with the relatively easier questions.

Naturally, answering just one of these questions correctly does not make the application assume that the user is an expert – perhaps they managed to do so just because of plain luck. To deal with such situations, the test not only manages an initial skill rating but also a uncertainty factor for the user's rating. The more unpredictable the user's answers get (in example, they are able to answer hard questions, yet fail on the most basic ones), the higher the uncertainty factor. Similarly, the factor decreases when the user's results start becoming consistent when compared to a certain, predefined skill rating level. Once the uncertainty factor is dealt with and the user answers a minimal amount of questions, the test is done. If they continue to behave chaotically, the test may take significantly longer. Consequently, the amount of questions the user has to answer is the smallest if they completely fail the test. The test is also rather short if the user completely aces it.

After the test is finished, the user gets the appropriate Language Levels unlocked. If they did better than a certain skill level, all the exercises considered to be required to obtain that skill level are already considered finished and are not necessary to be done by the user. If the user starts at the most basic level, the fact that their language skill was rated in the initial test (and the fact that they did not do very well) is completely obfuscated to them – they merely managed to unlock the first level. This prevents negatively affecting the user's self-esteem, if they had perhaps thought that they were better at the language than the results showed. After all, positive reinforcement is one of the main pillars of the application.

3.1.3 Regular Progress

After the evaluation process, the analysis of the user's progress continues in a slightly different way even in normal tests and exercises. The application keeps track of whenever the user makes a mistake. As described in the spacing effect, such functioning is desirable because it enables to identify which questions the user has the worst difficulties with. In a spaced repetition system, such questions would repeat a lot more often than others to help the student overcome their specific weak points in the language they're learning. However, to make things more sophisticated, the method used for the spacing effect in the application does not keep „mistake records“ for the specific question, which would make the exact same question appear over and over again.

A better solution is to apply certain tags which identify the questions and group them together based on what part of the language they are associated with. In example, if the user has trouble answering a question that is associated with the „preposition“ tag, they are more likely to get more questions from this tag in general, rather than this specific question only.

A single question may even be associated with multiple tags and as the user gets more and more questions wrong, the spaced repetition system can see patters within the wrongly answered questions simply due to the common denominator tag standing out. With this approach, the application could potentially find out that (in more complicated questions)

the crux of the matter was not actually the most obvious part that was being predominantly tested by the question, but something else within the wording of the question. The user may make a mistake in an exercise related to, say, present perfect (in English) but over time the algorithm finds out that they did not actually have a problem with the grammatical part of the question, but the vocabulary used in it.

Of course, it is also necessary to repeat questions that the user had no trouble with, occasionally, to help the learned information stay in the user's memory. However, they appear less frequently than the new and the troublesome ones.

3.2 A Community-driven Multimedia Pool

This part of the application design section describes the perhaps least language-learning related, but also the most standout, original idea of the application itself. The thought behind the idea was to come up with a comfortable and easy way how to update the images that are used in some of the exercises within the language-learning process. Surely, it would get tiresome to look at the same set of stock pictures overused mercilessly. At the same time, it might be too time-consuming for the developer to find multiple variant pictures for the same term and it could quickly bloat the space requirements necessary to keep all these images on the memory card, limiting the applications potential to be deployed on cheaper devices.

The most interesting solution to the problem is to make use of Firefox OS open approach to Web APIs. The application can allow the user the download images from a suitable image hosting platform if they wish to do so. Multiple candidates for the service were considered, and the resulting service is described in the implementation chapter. The most important part is that they allow some way that can be used to describe the images. A specific album reserved for the application would be used. The images are not downloaded permanently but only used temporary (if the cache is enabled on the operating system) by accessing the album and getting the image from there only when it's needed.

There is also the option that the users of the application could suggest images to be used in the learning process by uploading them into another album. This „download image pool“ could be maintained by a developer or chosen community members and contributors, who filter out the inappropriate images and check if the uploaded images are actually representative for the term that the uploader themselves meant them for. The images that fulfil all the necessary requirements get to be uploaded in the actual albums used for the multimedia questions in the language learning process and basically shared with all the other application users in the world.

3.3 User Interface

This section of the Design chapter talks about how the application looks like and most importantly, it gives the reasoning why it looks like that.

3.3.1 Design Philosophy

The main „philosophy“ behind the design of the user interface is to take advantage of the fact that the application is being designed specifically for Firefox OS, an operating system for handheld devices (primarily smartphones). This means, that everything about the user interface design can cater specifically to the controls of such devices and the practical

demands of their users. The design also attempts to keep everything simple, so that once the user gets a basic understanding of how the controls work or where all the important control elements are, they can safely assume that there will be no „surprises“.

3.3.2 Context Action Menu

The Context Action Menu is accessed by the „plus“ sign in the top right corner of the user interface, on its header. Basically, every „action“, everything that triggers progress in the application is controlled through here. The user presses the relatively small button in the corner, and a menu that shows all the actions available to the user. The actions vary depending on what context the user is in – the inner state of the application (in example, if the user is currently being evaluated or not) and the page that the user is currently on (that they can visually see). Whilst this design may seem too simple, or even primitive, it is actually very practical. It allows the user to hold the device in one hand and to control pretty much everything that happens in there with just the thumb of the same hand. It is quite the ergonomic solution as well. The user can relatively safely rest their fingers on most of the phone's screen area without having to „lock“ the device to avoid tampering with whatever is on the screen when this application is running. This also means that there are no text input control elements which would allow to perhaps answer questions by writing the answer instead of choosing it. Although that may sound like an unfortunate limitation, I personally do not believe that having a different way of answering question really does anything beneficial for the learning process. If anything, it is detrimental, because it messes with the user's rhythm and, especially on typically cheap Firefox OS handheld devices, using the keyboard can be a rather unpleasant experience.

3.3.3 Preferences Menu

The preferences menu (pictured here 3.1), where all the settings of the applications are, is on the horizontally opposite side of the user interface. It is meant to be the most rarely clicked control element of the application and hence it is the most further away from the user's right thumb. The most notable setting in this menu is the one that allows the user to flip almost the entire interface horizontally. Basically, it flips the position of all the control elements. This allows left-handers, or people who prefer to hold their device in the left hand, to enjoy the same ergonomic comfort as everyone else. The install button (if necessary) also appears on the top header, as is usual with Firefox OS applications.

3.3.4 Main Page

The main page refers to the body of the application where the majority of information is displayed. It is rather simple and straightforward, since the only real control elements here are expandable thumbnails of images and a footer/toolbar at the bottom of the user interface which is used to switch between the interface display pages, which are referred to as follows:

- **Main** – initially, this page displays the splash-screen that the user sees when they open the application. During the learning process, it is then used to display the questions and answers, whether they're textual or visual in nature. Visual questions only come in the form of image-question and text-answers. Originally, it was planned to also have the opposite of that, text-question and multiple image-answers, however I was

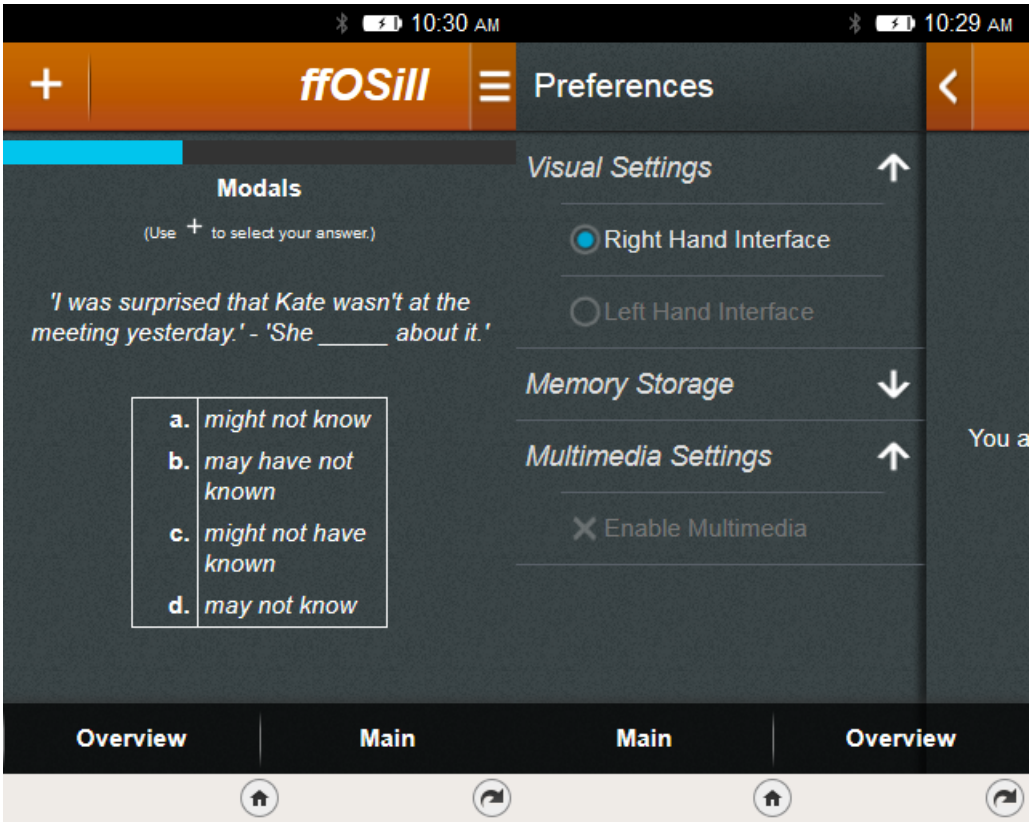


Figure 3.1: Picture of the UI with an active question and a left-hand interface on the left and a picture of its preferences menu on the right.

unable to find a way how to display the images comfortably, and more importantly it would bring some technical limitations due to the way the images are represented in the application. These limitations are explained in the Implementation chapter of the thesis. On this page, the context menu is used to select the user's option (answer) to the presented question, most of the time. An example of the question page can be seen here 3.1.

- **Overview** – just like the name says, the purpose of this page is to show the overview of the user's active language courses, what level they are on, what levels they have already finished and their progress through the active level sections. The progress bars (or „experience bars“) are shown for each section in the section overview display page. The context menu actions for this page are used to navigate between choosing a language, a level and a section (the former being available from the start, while the latter two are only available if an already calibrated language was chosen).

- **Images** – this page displays instructions how to upload and properly share images that the user would like to add to the multimedia pool of the application. The context action menu is used to select an image, whose expandable thumbnail is then presented on the page, and afterwards it leads to the selection of the category tag that the user would wish to upload their image as. An example of the image upload page can be seen here 3.2.

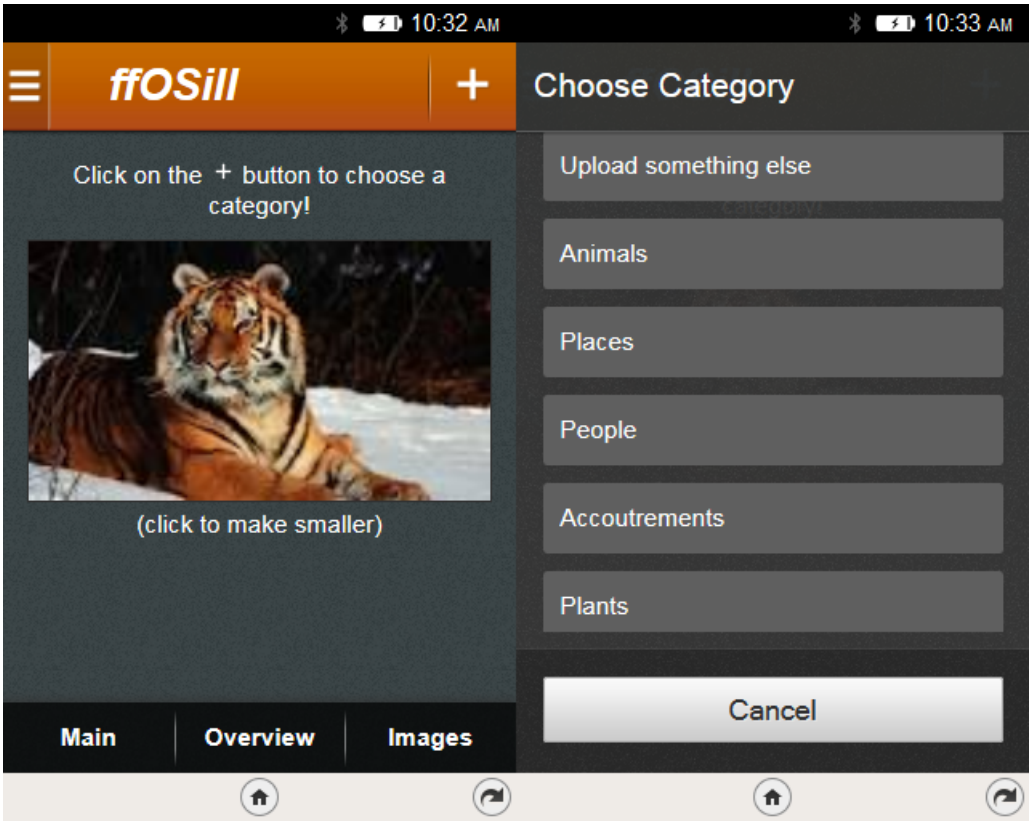


Figure 3.2: Picture of the UI with enabled multimedia and a picture selected for upload on the left and a picture of its context action menu on the right.

Chapter 4

Implementation

This chapter describes how, and with what technologies, the design mentioned in the previous chapter was implemented. As far as (computer) languages go, the application is programmed exclusively in HTML5, CSS3 and Javascript. It uses „pure“ Javascript – though the popular jQuery or AngularJS libraries would certainly be useful, they weren't used to avoid potential issues in the testing phase caused by some arbitrary restrictions based on the imperfect implementation of (some versions of) the Firefox OS simulator. The application is considered a „Packaged app“ – a type of an Open Web App that has all of its resources (the source code, the manifest etc.) packaged in a zip file, rather than having resources on a Web server. Such apps are distributed for the Firefox OS end users strictly through the Firefox Marketplace.

For the sake of the implementation, the application's project codename is *ffOSill* – an abbreviation for *Firefox OS Interactive Language Learner*. Whist the name is certainly catchy, it may cause some trademark problems when trying to get a place in the Firefox Marketplace.

4.1 Interface Implementation

I used *Gaia Building Blocks* as a foundation for the user interface of the application. Specifically, *ffOSill* is built upon an already pre-edited, simplified form of Building Blocks with some basic added functionality by Pierre Richard from `hacks.mozilla.org` [12]. Additionally, the generic `install.js` and `dispatcher.js` scripts for installing applications and dispatching events were also taken from him. Whilst retaining much of the visual design of the Building Blocks original (such as the color scheme and general proportions) some UI elements were significantly modified for the application's needs. The Building Blocks version used is the same as the one recommended for the 1.3 version of the operating system, to ensure maximum backwards compatibility. They were also chosen because they already fulfil the recommended visual standards for Firefox OS applications [5], so I wouldn't need to trouble myself with making my own graphics matching their requirements.

4.2 Local Storage

Since *ffOSill* is a Packaged app, it needs some way to store its memory between user sessions locally. Otherwise, the user's progress would be lost every time they quit the application or turn off the device. The most obvious way is using `localStorage` – a new HTML5 feature

which allows web applications a new way of storing persistent information [8]. Unlike cookies, it allows for a lot more storage space and it isn't necessary to transmit it to the server. Since the application has no server of its own and doesn't use cookies, this would seem like an ideal solution. However, it has its issues:

- accessing information in `localStorage` is a synchronous operation, meaning it blocks the main thread and can potentially make for an unpleasant, laggy experience for the user, especially if it's accessed too often and to retrieve big chunks of data.
- its size is limited to 5MB and there is no way of selectively deleting records or telling how full it is.
- a single call to the `localStorage.clear()` method can wipe the memory of all applications using it.
- all data has to be stored as strings.

Hence, its usage for storing data of Firefox OS applications is not recommended. Fortunately, there have been improvements (or „hacks“ if you will) to improve the situation. The one chosen for this application is `async_storage.js`, an edited version of the original Gaia project version, by Asier Arizkuren [1]. It is basically an asynchronous version of the `localStorage` API based on an `IndexedDB` database, which uses the same relatively simple methods of storing and loading information that the former did. Additionally, it can also store entire objects instead of just strings, allowing me to store the entire Language object created for each language course in `language.js` without having to serialize it for storing and parsing it for loading information.

4.3 Language Data Format

The language data is encoded using the Extensible Markup Language (XML) format. It mirrors the design mentioned in 3.1.1. An example of how it's encoded follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<data>
 <languages>
  <language name="LANGUAGENAME">
   <tags>
    <tag>TAGNAME</tag>
   </tags>
   <replace_string>QTEXT_STRING</replace_tag>
   <questions>
    <level id="LEVELID">...</level>
   </questions>
  </language>
 </languages>
 <multimedia>
  <available_category_tags>
   <category name="CATEGORYNAME"/>
   ...
  </available_category_tags>
```

```
  <category in_LANGUAGENAME="CATEGORYNAME">
   <album>LINK</album>
   <picture in_LANGUAGENAME="TERMNAME">LINK</picture>
  </category>
  ...
 </multimedia>
</data>
```

The XML definition of a Language Level:

```
<level id="LEVELID">
 <subsection name="SECTIONNAME">
  <info>
   <main_tags>
    <tag>TAGNAME</tag>
   </main_tags>
   <dependency>SECTIONNAME</dependency>
   <requirement>NUMBER</requirement>
  </info>
  <question id="QUESTIONID">
   <qtext>QUESTION_TEXT</qtext>
   <answer id="correct">CORRECT_ANSWER</answer>
   <answer id="false">INCORRECT_ANSWER</answer>
   ...
   <additional_tag></additional_tag>
   <img_alt>
    <ref_term>[CATEGORYNAME.]TERMNAME<ref_term>
    <ref_link>LINK</ref_link>
    <qtext>...</qtext>
    <answer>...</answer>
    ...
   </img_alt>
  </question>
 </subsection>
 ...
</level id="LEVELID">
```

Most of the language definition is rather self-explanatory. As you can see, the data `.xml` file includes all the language info as well as the multimedia info which are split in the two main XML nodes. The `<replace_string>` element defines what string (that should be found in all question strings of its parent element language) is visually replaced with the correct answer upon answering. The `<requirement>` element in the section info is used to enable the language creator to put emphasis on certain sections and the `<dependency>` element is optional and it says what section must be completed before this section is unlocked to the user. The `<img_alt>` element is optional but highly recommended, since it enables usage of visual question types. It is used as the alternative (but preferred) version of the question if multimedia access is enabled. The visual question is replaced with a thematically similar standard, text question if multimedia are unavailable, so the non-visual question part is obligatory. The inside of the `<multimedia>` tag is explained in the Multimedia Implementation section of this chapter (4.6).

The main reasoning behind the XML structure is that potentially anyone can add new language records, new sections and questions to existing languages or new multimedia, even with relatively little understanding of computer engineering. It means that a separate individual, who is an actual professional expert on the language subject, can edit and add questions, independently of the software engineer.

4.4 Calibration Algorithm

This section describes the concrete algorithm of the skill evaluation method designed in 3.1.2.

Algorithm 1: Language Skill Evaluation Algorithm Pseudocode

Data: initial MMR, defined MMR brackets for each Level
Result: final MMR which defines the user's Language Level

```
1:  uncertainty = 100%
2:  questionsCount = 0
3:  while uncertainty > minimumThreshold or questionsCount < 20 do
4:      questionsCount++
5:      get random question from the random range based on current MMR bracket
6:      if answered correctly then
7:          check consistency
8:          if consistent good performance then
9:              decrease uncertainty
10:             increase MMR by a function of uncertainty
11:             modify random range based on current bracket upwards
12:         else
13:             increase MMR by a function of uncertainty
14:             increase uncertainty
15:         end if
16:     else
17:         check consistency
18:         if consistent bad performance then
19:             decrease uncertainty
20:             decrease MMR by a function of uncertainty
21:             modify random range based on current bracket downwards
22:         else
23:             decrease MMR by a function of uncertainty
24:             increase uncertainty
25:         end if
26:     end if
27:     recalculate MMR bracket
28: end while
```

The MMR acronym used in the algorithm stands for „Matchmaking Rating“. It is one of the gamification influences on the overall conception of the application (the MMR term itself generally comes from the Elo rating system method for calculating relative skill levels of players). The idea behind this part of the application is that the whole point of the language evaluating is to find the right skill bracket for the user who just started

a new language course. While they aren't competing against anyone per se, the concept of „defeating" stronger opponents (more difficult questions than they were expected to be able to handle) giving more rating than others still applies. Consistence is a very important factor in here – answering just one in ten difficult questions is nothing more than a stroke of luck and should be treated as such by the algorithm as well.

The values itself that are used to define the initial MMR and the MMR brackets were implemented empirically, based on observations of how different skilled users fared in an actual placement test done by pedagogic professionals. The main advantage of this method is that users' language skill is determined much faster than simply copying existing tests with pre-determined question counts and gradually difficulty progression. It comes at the cost that there is also a possibility that the user is placed in a higher or lower skill level than he should be in. To lessen the impact, the algorithm's MMR placement takes a pessimistic approach towards the user's progression – it simply assumes the worst about them. It is much harder to climb back up in MMR once the user already made some consistent mistakes at low level questions. This is done simply by having the MMR bracket ranges biggest for the lower level and allowing only a fraction of the top MMR range for the highest bracket. The decision comes from a common sense assumption that, despite the fact that all „unlocked" levels are still retroactively available for revisiting, it is quite unlikely that the user will click on them, even if they're having a tough time in the current level (as a result of being placed higher that they should be). On the other hand, if a highly skilled user gets accidentally placed in a lower level than he should be, he should have no trouble getting to the next one within a short period of time. At worst, it hurts their pride, rather than significantly impeding their progress. The majority of the time spent testing this application was spent towards proving or potentially disproving this assumption, as described in the Testing chapter.

4.5 CALL Implementation

This section briefly describes how the computer-assisted language learning (CALL) methods based on general learning concepts, which were described in 2.2, were implemented in *ffOSill*. The forgetting curve and spaced repetition concepts are enabled by making use of the short and long term memory of the application. Once they got past the calibration process, the user's progress through individual sections is much slower and based on small „experience points" (XP) rewards for each correctly answered question. These XP are effectively synonymous with progress points. Users never lose XP for answering a question wrong (*ffOSill* focuses on the positive). However, if a user keeps getting the same question over and over again in the same session, they keep getting less XP for it, down to a minimal amount. Such behaviour, tracked through the short term memory of the application, would indicate that the section is either low on questions or that the user is genuinely trying to „grind it out" by putting lots of time into the same section within the same application session. As explained in 2.2, this is not an efficient way of learning, and is punished by slowing it down.

On the other hand, when the user makes a mistake, the tags of the question he made a mistake on are recorded in the long term memory, to aid the spaced repetition concept. Once some tag's errata counter reaches a certain threshold (based on the level the user is in, or more precisely the total amount of questions in the level), the tag is „popped" from the queue of tags waiting to be repeated. If no question with the appropriate tag is found in the currently active section, other sections on the level are searched through. Testing

how well this works on real users is rather difficult, since it relies on the assumption that the questions are well inter-tagged to begin with, which may require more pedagogic input than the application had at the time of testing. Furthermore, the users themselves may find it difficult to tell that the forgetting curve triggered question they get was actually the one they had trouble with, because they may have, ironically, by that time forgotten that they had trouble with the material.

Finally, it should be noted that since there are no pre-determined tests to go through in the application, some degree of randomness to freshen things up is necessary. When a section is selected, question order (outside of questions triggered by CALL methods) is determined by all available questions in the active sections from a randomly shuffled queue (once it's empty, it's shuffled again). Similarly, every time a question is generated, the correct and incorrect answer positions on the screen are randomly shuffled. To implement this, the application makes use of a standard variation of the Fisher-Yates shuffle (aka Knuth shuffle) algorithm.

4.6 Web API Usage for Multimedia Implementation

The main tool for implementing the visual question content of the application is the Imgur API Version 3 [4]. The Imgur on-line image hosting service has been chosen because of all the currently popular image sharing sites, it was the most suitable one that fulfilled all the conditions for usage by *ffOSill*. As a service, it allows its end users to create multiple albums with easily editable album names and image titles. Its API allows easy and fast access to the created albums. A simple `GET` request can be used to get information about all images on the album through an XML (or JSON, if preferred) response, and through there it is easy to find an image (or images) inside the album based on the title the application is looking for.

In the context of the application it means that it can search for an appropriate image for an image type question which describes a referential term that consists of a category and a tag from said category. In example, the category name can be „animals“ and the tag that belongs to this category might be „dog“. From the implementation perspective, this means that *ffOSill* searches through the XML data if the category and tag combination exists and, if so, if there is a direct picture link for it. However, it can also find an album link instead of a direct picture link. If it does that, the application uses the Imgur API to check if an image with a title that corresponds to the tag exists in there. If yes, the link(s) to the found picture(s) can also be used as the image pool for the category and tag combination. One image is randomly chosen from this pool and used for the image type question that requested the term.

Basically, for every term, there is either a direct link provided in the image question's definition in the XML data, or the application uses a term to find a combination of albums and direct links for the term and randomly chooses one of them. It could be said that category tags are synonymous to albums and terms are synonymous to direct picture links. However, the use case scenario where the XML data was defined as available categories with no albums is also not unexpected. It could mean that the term category is available to create albums for in the future – perhaps with the contribution of the application's end users.

This brings us to the other benefit of using the Imgur API. It was the only service that could provide a feasible way for the users to contribute images without the application actually needing its own server to control the flow. Other than creating regular albums,

which can be controlled through the Imgur website after signing in, The Imgur API Version 3 also allows creation of so called „anonymous albums“. Unlike the regular ones, which require OAuth 2.0 authentication – in other words, to be logged in – in order to be manipulated with, anonymous albums require nothing but a Client ID (the to which the application using the API is registered) and a „deletehash“ which is a random string that is return on creation of an anonymous album. Knowledge of the deletehash is basically the authorization for editing the anonymous album. The only downside to this is that the anonymous albums have to be controlled fully through the Imgur API (the service provider does not give this option to regular registered accounts on their websites).

To make matters even simpler, Mozilla Firefox provides its own API called *Web Activities* (documentation available here [5]) which is supposed to be available to all installed code running on Firefox for Android, but is currently only enabled on Firefox OS. Activities are something that the user wants to do on the device. Typically, send an e-mail, save bookmark, or most relevantly for this application – pick an image. This activity enables the application to let the image picking handling completely to Firefox OS. It offers the user to choose an image from gallery, resize it, or to go to the in-built Camera application and take a picture which will be then selected for uploading. The only downside to *Web Activities* is the fact that it is necessary to use slightly different code for Firefox OS 1.3 and below and Firefox OS 2.0+.

As far as usage limitations go, the Imgur API is free for non-commercial usage. *ffOSill* has been planned to be released under a free license from the start (one that is compatible with / considered a *Good License* by Fedora) so this is no problem. Heavy free usage comes with certain bandwidth limitations, but the application is not expected to exceed any of them, especially since the idea of having multiple image answers (instead of just image type questions) abolished. All that's necessary to use the API is that the application is registered (which is a simple and free process) after which the developer is granted their own `Client ID` and `Client Secret`.

Chapter 5

Testing

This chapter describes the entire testing process, all the input necessary for testing, information about the test subjects and feedback from them. Finally, it evaluates the outcomes of the testing phase from both the technical and language skill perspective and how *ffOSill* compares to the existing language learning application software.

5.1 Prerequisites & Preparation

The main prerequisite for testing this application is, of course, having a thoroughly implemented language for testing. Despite the fact that the application is localized in English, the language used for testing purposes was English as well. In order to make the experience for the language learners at least somewhat professional and productive, and make the test results as accurate as possible, most of the questions and sections used were taken from the Cambridge book *English Grammar in Use* [9]. Since publicizing the questions would be copyright infringement, my personal best interpretation of what an English language course should look like was used for the purposes of demonstrating the language learning aspect of the application.

Although it has been said that you only need to test with five users [6], I managed to find seven different test subjects willing to go through an approximately month long test phase. Because most of them don't own an actual device with Firefox OS, to make things as even as possible, they all tested the application using the Firefox OS simulator, running the 2.0 version of the operating system. All the test subjects needed to use an instruction manual to get the simulator running on a Mozilla Firefox web browser, regardless of their computer literacy level, although the least proficient users needed some personal help and instructions as well.

5.2 Test Results

The test results, with all the relevant data listed for each test subject, are here 5.1. A brief explanation of the data in the table:

- The **Computer Literacy** tracks the individuals' ability to use all kinds of computer software in general as well as their experience specifically with handheld devices, such as smartphones. Mostly a subjective, self-reported statistic, but the highest skilled individuals had education to back it up and the lowest rated subjects wouldn't lie about that.

Their	Test Subjects						
	#1	**#2**	**#3**	**#4**	**#5**	**#6**	**#7**
Computer Literacy	High	High	Average	Average	High	Low	Minimal
Achieved Language Skill	BAN4	B2 & FCE	B1	BAN4	B1 → BAN2	N/A	C2
***CEF* Level Equivalent**	B1	B2	B1	B1	B1 → A2	A1	C2
Expected Evaluation Level	Level 3	Level 4	Level 2	Level 3	Level 3 or 2	Level 1	Level 4
Actual Evaluation Level	Level 2	Level 4	Level 1	Level 3	Level 2	Level 1	Level 3
Time To Reach Next Level	5 hours	1.5 hours	1 hour	8 hours	4.5 hours	6 hours	3 hours

Table 5.1: A comparison of the subjects' technical & language skills and their results.

- **Achieved Language Skill** is based on relatively objective standards – skill in English that the individual is actually, officially confirmed to have achieved. The BAN levels are based on the BUT standards for judging English language skills. Some are supposed to directly correlate to a particular *CEF* level. The FCE language skill refers to the *Cambridge English: First* certificate. Although it is supposed to be a practice level of B2, it is differentiated since the B levels listed in this row all come from school state exams.
- The ***CEF* Level Equivalent** row transforms the previous one into the *Common European Framework for Languages: Learning, Teaching, Assessment* guideline used to describe achievements of foreign language learners across Europe. They are split into Basic User (A1, A2), Independent User (B1, B2) and Proficient User (C1, C2). BAN4 is said to officially correlate to the B1 level.
- **Expected Evaluation Level** is the language skill level that I expected the user would get based on their aforementioned official skill and sometimes my personal assessment of the individual if their case was unclear. In general, the rules could be summed up as follows:

  - A1 is always Level 1.
  - A2 might be Level 1, but is most likely Level 2.
  - B1 might be Level 2, but is most likely Level 3.
  - B2 is at least Level 3, but might be Level 4.
  - C1 should be Level 4 and go up from there.
  - C2 should be Level 4 and have no trouble finishing it.

- **Actual Evaluation Level** is the level the test subject obtained after they were done with the calibration process in *ffOSill*.

- **Time To Reach Next Level** is probably the most important statistic, since it tracks the approximate (or estimated) time to advance to the next milestone in the language according to the application. Although it is expressed in hours, it is not necessarily as absolute as presented, since it wasn't measured exactly and was self-reported by the individuals themselves, based on the time they thought they spent with the application. In some cases, the test subject could only provide a relative amount of time they spent on the application overall (as in a third of it, in example).

5.2.1 Explanation

The fact that none of the users actually went above their expected level is appeasing. The only real example of overperforming was subject #2, but that was somewhat to be expected, because they managed to go through both the B2 level state exam and the FCE exam with relative ease, so their real achieved language skill should be most likely at least C1, hence the Level 4. BAN4 holders were expected to live up to the B1 equivalent Level 3. Subject #1 unfortunately did not manage to do so. Since it also took them a relatively long time to go to the next level (which was supposed to be their actual level), the most feasible explanation to me is that their language skill is no longer reflected in the level they've officially acquired – something that is bound to happen without repeated practice. Test subject #3 has also achieved a lower level than expected, and the expectations were already set at the lowest level. They may have had a B1 level according to a school state exam, but their language skills hardly reflected that to begin with and they managed to pass only barely. The most jarring discrepancy is test subject #7. They had officially obtained a certificate that says they are able to use English at a near-native level at some point in time and yet they failed to obtain the highest level in the application. This may be explained by a multitude of factors, but mainly the fact that this user had the lowest expertise with handling handheld devices and also the oldest among the test subjects.

5.2.2 User Feedback

This section contains some of the interesting feedback that the individuals who tested the applications provided themselves regarding the application:

- The user interface was generally well received. Though not deemed as particularly visually attractive, the design was commended for keeping all the control elements clearly at hand at all times. Ironically, the users with the highest computer literacy needed the most time accepting the way the application is controlled, but even those test subjects could see the practical value of the control elements' distribution on the

interface. The idea itself was actually inspired by complaints from test subject #7 in a very early stage of testing.

- The calibration process implementation can be considered a success, since none of the test subjects had any particular, negative remarks regarding it and this phase took them one hour at the most to complete.
- Although its usage during the test phase was very limited, due to the lack of a functional camera to take pictures with in the Firefox OS simulator, the idea of potentially sharing images from the device with the entire user-base was universally welcomed.
- The forgetting curve and spaced retention went mostly unnoticed by the test subjects. This is because it's hard to realize that the method is being applied from an end user's perspective and because I had difficulties coming up with proper tagging that would interconnect the questions in the English language test version.

5.3 Evaluation

	Language Learning Applications			
Features	*Duolingo*	*busuu*	*Rosetta Stone*	*ffOSill*
License	Proprietary	Proprietary	Proprietary	Free (Apache License)
Payment Model	Freeware	Freemium	Payware	Freeware
Available Multimedia	Minimal	Audio-Visual	Audio-Visual	Visual (Images)
Community Input	Create courses	Tutoring	Minimal	Contribute images
Availability	Online platform	iOS & Android	iOS & Android	Firefox OS

Table 5.2: A comparison of the applications mentioned in this thesis and *ffOSill*, the application created for this thesis.

As evident from the previous section, it can be concluded that the application did at the very least succeed in applying it's slightly gamified approach to the language skill evaluation method. Its „experience bars“ and methods of implementing the typical computer-assisted language learning methods are not bad per se, but they do not stand out amongst the competition. The user's reaction to the application was generally positive but that may be expected with cherry-picked test subjects. For a retrospective view, a table that compares the application created in this thesis, *ffOSill* to the rest of the language learning applications

mentioned in this thesis can be found here 5.2. This is ultimately what the average user's decision to use or not use a language learning application would be based on.

Chapter 6

Conclusion

In this Bachelor Project, I have developed an application called *ffOSill*. It is an interactive language learning application designed specifically for the Firefox OS operating system for handheld devices. It was designed with this in mind and attempts to break new ground by utilizing an unorthodox user interface, allowing its users to control it easily and comfortably with just one hand. It applies many of the typical language learning methods used in contemporary software of this kind but it also takes the concept of gamification to comparatively new levels. Its attempts to create a faster language skill evaluating method have been thoroughly tested and the test results proved that it can have a positive contribution towards the language learning process. The application makes smart use of Web APIs to circumvent some of the shortcomings of typical Firefox OS devices and even better use at enabling the application's users to contribute to the language's multimedia pool. The thesis clearly points out how *ffOSill* is different from the competition and that there are valid reasons for choosing it over similar, even commercial software. In the long term vision, there are many useful potential developments for the application, such as making it even more open towards the community by improving on the language creation process and improving its tag definition. The more welcoming to the community it is, the more will the pedagogic language level increase and the application will become gradually more professional. With its dynamic approach to question definition, it could also be modified into software for learning more than just languages.

Bibliography

[1] Arizkuren A. asynchronous version of the localStorage API [online]. https://github.com/aarizkuren/asyncStorage, 2014-11-21 [Accessed 2015-4-3].

[2] Chaplin H. I Don't Want To Be a Superhero [online]. http://www.slate.com/articles/technology/gaming/2011/03/i_dont_want_to_be_a_superhero.2.html, 2011-3-29 [Accessed 2014-11-28].

[3] Goodwin-Jones, R. Emerging Technologies: Mobile Apps for Language Learning. *Language Learning & Technology*, 15(2):2–11, July 2011.

[4] Imgur.com. Imgur API Version 3 [online]. https://api.imgur.com/, 2015-4-2 [Accessed 2015-1-2].

[5] Mozilla Developer Network and individual contributors. The Firefox OS platform [online]. https://developer.mozilla.org/cs/Firefox_OS/Platform, 2005-2015 [Accessed 2014-10-16].

[6] Nielsen, J. Why You Only Need to Test with 5 Users [online]. http://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/, 2000-03-19 [Accessed 2015-05-11].

[7] phase-6 GmbH. phase-6 system of repetition [online]. https://www.phase-6.com/en/vocabulary-tool/scientific-background/scientific-background.html, 2007-2017 [Accessed 2014-11-29].

[8] Pilgrim M. The Past, Present & Future of Local Storage for Web Applications [online]. http://diveintohtml5.info/storage.html, 2009-2011 [Accessed 2015-4-3].

[9] Murphy R. *English Grammar in Use*. Cambridge University Press, third edition, 2006. ISBN 978-0-521-53289-1.

[10] Renaud, C., Wagoner, B. The Gamification of Learning. *Principal Leadership*, 12(1):56–59, 7 2011.

[11] Ricci, K.E. The use of computer-based videogames in knowledge acquisition and retention. *Journal of Interactive Instruction Development*, 7(1):17–22, 1994.

[12] Richard P. A minimalist's working example of the design guide rules for Firefox OS [online]. https://hacks.mozilla.org/2012/12/fxosstub-a-minimalists-working-example-of-the-design-guide-rules-for-firefox-os/, 2012-12-10 [Accessed 2015-02-24].

[13] Shah, J. ZTE finds Firefox OS faring better in emerging markets than the US [online]. http://www.pcworld.com/article/2839579/zte-finds-firefox-os-faring-better-in-emerging-markets-than-the-us.html, 2014-10-27 [Accessed 2014-11-28]. PCWorld.

[14] Sousa, D.A. *How the Brain Learns*. SAGE Publications, third edition, 2006-1-28. ISBN 9781412937382.

[15] Vesselinov, R., Grego, J. Duolingo Effectiveness Study [online]. http://static.duolingo.com/s3/DuolingoReport_Final.pdf, December 2012 [Accessed 2014-11-28].

[16] Werbach, K. *For the Win: How Game Thinking Can Revolutionize Your Business*. Wharton Digital Press, 2012. ISBN 978-1613630235.