Developing a game based virtual learning environment

Enda Quigley
BSc.

A dissertation submitted in fulfilment of the requirements for the award of

Master of Science (MSc.)

to the

University of Limerick
Department of Computer Science and Information Systems

Supervisors: Reinhard Schäler
Dr. Dimitra Anastasiou
Dr. Chris Exton

Submitted to the University of Limerick, October 2010
Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of MSc. is entirely my own work, that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

ID number: 0541354

Signed: ________________________________

Date: __________________________________
Abstract

It’s a common misconception that “digital natives” have a short attention span when in fact, they are just bored of the old methods of teaching and prefer to learn through fun, interactive games. These learners crave to be the active participant rather than the passive observer. Educators are slowly beginning to acknowledge the advantages of incorporating video games in the class to complement the old tell-test method of teaching.

The online learning platform that I have developed will provide students with a number of educational games that cover a wide range of localisation awareness topics such as cultural appreciation, linguistic diversity, calendars, currency, and writing systems. Various motivational techniques such as in-game achievements and high score lists will be implemented to promote friendly competition amongst the students and to encourage them to continue playing the games. Our platform has also been developed as a general purpose virtual learning environment (VLE) in an effort to get more Irish primary schools interested in the product.
Acknowledgements

I would like express my sincere gratitude to my supervisors Reinhard Schäler, Dimitra Anastasiou, and Chris Exton for all their help and support throughout the year.

I would like to thank Reinhard for giving me the opportunity to continue this project and for the encouragement to undertake a Research Masters. I would also like to thank Dimitra for getting involved in my supervision and Chris for the creative freedom to continue working on my research.

I would also like to thank the LRC and the CNGL for financially supporting me, Karl Kelly and Ger Harrahill for all of their assistance over the last year and I wish all the PhD students the best of luck completing their research, and in their careers.

Last but not least, I would also like to take this opportunity to thank my parents, family and friends. Your patience and support has been greatly appreciated.
# Table of Contents

1. Introduction ................................................................................................................................. 1  
   1.1 Motivation ............................................................................................................................... 1  
   1.2 Project justification .................................................................................................................. 2  
   1.3 Proposed objectives ............................................................................................................... 3  

2. Research ....................................................................................................................................... 4  
   2.1 Introduction ............................................................................................................................ 4  
   2.2 Analysis of existing products ................................................................................................. 4  
   2.3 Meeting educational professionals ....................................................................................... 6  
   2.4 Curriculum integration ........................................................................................................... 7  
   2.5 Designing a user interface for children .................................................................................. 8  
   2.6 Motivation and encouragement techniques ......................................................................... 10  

3. Literature Review ....................................................................................................................... 12  

4. Design .......................................................................................................................................... 15  
   4.1 Introduction ........................................................................................................................... 15  
   4.2 Requirements ........................................................................................................................ 15  
      4.2.1 Functional requirements ................................................................................................. 15  
      4.2.2 Non-functional requirements ....................................................................................... 18  
   4.3 Design decisions .................................................................................................................... 19  
      4.3.1 Desktop vs. browser ....................................................................................................... 19  
      4.3.2 Client implementation ................................................................................................... 20  
      4.3.3 Classroom competition ................................................................................................. 21  
   4.4 Security features .................................................................................................................... 21  
      4.4.1 Importance of security .................................................................................................. 21  
      4.4.2 Password protection ...................................................................................................... 22  
      4.4.3 Student privacy .............................................................................................................. 23  

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.4.4 Reporting</td>
<td>23</td>
</tr>
<tr>
<td>4.4.5 SQL injection</td>
<td>24</td>
</tr>
<tr>
<td>4.5 Database</td>
<td>25</td>
</tr>
<tr>
<td>4.5.1 Database structure</td>
<td>25</td>
</tr>
<tr>
<td>4.5.2 Multilingual support</td>
<td>26</td>
</tr>
<tr>
<td>4.5.3 Normalisation and referential integrity</td>
<td>26</td>
</tr>
<tr>
<td>4.5.4 Database optimisation</td>
<td>27</td>
</tr>
<tr>
<td>4.5.5 Final schema</td>
<td>27</td>
</tr>
<tr>
<td>5. Implementation</td>
<td>34</td>
</tr>
<tr>
<td>5.1 Introduction</td>
<td>34</td>
</tr>
<tr>
<td>5.2 Implementation technologies</td>
<td>34</td>
</tr>
<tr>
<td>5.2.1 Microsoft Expression Blend 4</td>
<td>34</td>
</tr>
<tr>
<td>5.2.2 Microsoft Silverlight</td>
<td>34</td>
</tr>
<tr>
<td>5.2.3 Windows Communication Foundation</td>
<td>35</td>
</tr>
<tr>
<td>5.3 Software lifecycle</td>
<td>35</td>
</tr>
<tr>
<td>5.4 Teacher dashboard</td>
<td>35</td>
</tr>
<tr>
<td>5.5 Client application</td>
<td>36</td>
</tr>
<tr>
<td>5.5.1 Login</td>
<td>36</td>
</tr>
<tr>
<td>5.5.2 Homepage</td>
<td>37</td>
</tr>
<tr>
<td>5.5.3 Plugin downloader</td>
<td>37</td>
</tr>
<tr>
<td>5.5.4 Profile</td>
<td>38</td>
</tr>
<tr>
<td>5.5.5 Friends</td>
<td>39</td>
</tr>
<tr>
<td>5.5.6 Student search</td>
<td>40</td>
</tr>
<tr>
<td>5.5.7 Achievements</td>
<td>41</td>
</tr>
<tr>
<td>5.5.8 High score list</td>
<td>41</td>
</tr>
<tr>
<td>5.5.9 Message centre</td>
<td>42</td>
</tr>
<tr>
<td>5.6 Plugin API</td>
<td>43</td>
</tr>
</tbody>
</table>
List of Figures

Figure 1: Poorly designed user interface.................................................................5
Figure 2: Teachers use-case diagram.........................................................................16
Figure 3: Students use-case diagram...........................................................................17
Figure 4: Database table inheritance..........................................................................25
Figure 5: Student database schema ............................................................................29
Figure 6: School database schema ............................................................................31
Figure 7: Platform communication schema .................................................................33
Figure 8: Sample web service method .........................................................................47
Figure 9: Analysis of the in-game achievement system................................................55
Figure 10: Students interest in the educational games..................................................56
1. Introduction

1.1 Motivation

In partial fulfilment of the requirements for the degree of Computer Systems in the University of Limerick, it was necessary to complete a final year project (FYP) in 4th year. I designed and developed a working prototype for a "Primary School Localisation Toolkit" under the supervision of Reinhard Schäler. Reinhard contacted me during the summer offering me the opportunity to continue working on the project for another year as a research master student. I had learnt a lot from my previous experience and was glad to have the opportunity to work on this project again and to improve on the overall concept, design and implementation.

I was also motivated to undertake this project as I am a true believer in game-based learning. As an avid gamer, I have no doubt that games can be an effective learning tool and can motivate children to learn more, on their own. Although most games don’t necessary teach anything, learning happens in a rule based world. The player has to constantly learn and adapt to new conditions in order to survive.

I also noticed that a lot of developers who claim to use game-based learning techniques don’t fully understand what makes games fun to play in the first place. Children should not need detailed instructions on how to play a game. Developers should give children the opportunity to challenge facts, form their own hypothesis and try them out until they solve the problem on their own. Educational games should first and foremost be entertaining, only then can education material be woven into the fabric of the game. Assessment is another area which game developers often fail to implement successfully. Students should never be presented with a list of questions to test their knowledge on a given subject. Embedded assessment should be used in its place and be an integral part of the game design, so, if the student manages to beat the game, they have passed the test.
1.2 Project justification

As the number of students in primary school from different ethnic backgrounds has rapidly increased over the last few years, it is important that the topics covered by the National Council for Curriculum and Assessment (NCCA) evolve to meet the demands of our changing society. Over the last ten years, primary schools have been adjusting to the new challenges set by the influx of foreign nationals into the country. Given the increasing diversity within the classroom regarding race, culture and religion, it has never been more important to teach localisation awareness topics to children at an early age.

We also need to address the way in which children are thought. Children these days are raised on rich sources of media such as the TV and PC where everything is so vivid, fast-paced and intense. Based on the latest scientific research and evidence in neurology, it’s clear that exposure of various kinds of stimulation actually changes the brain structure and affects the way people think, learn and interact (Prensky, 2001). Children are starting to think differently from the generations gone by. They want to be the active participants, create their own hypothesis, and learn through trial and error. These “digital natives” have a short attention span for the old tell-test method of teaching but not for interactive learning. Our educational platform attempts to teach children through fun and interactive games – the holders of their attention.

A recent article in the Irish Independent also shows that “Ireland trails world in the use of school computers.” (Donnelly, 2009) The article describes a report carried out by the Operation of Economic Cooperation and Development (OECD) which claims that almost one in three Irish 15 year olds have never used a computer in school – more than double the average in the developed world. This project would introduce children to computers at an early age while teaching them localisation awareness topics at the same time.

Irish primary schools do not have a VLE (virtual learning environment) of choice. Many of the popular UK and US VLE’s do not cater to the Irish Primary School curriculum. The only current alternative is to install a bare VLE on a school by school basis and expect the teachers to gather and upload their own information.
These packages are often too complex, poorly implemented and time consuming. The lack of a roadmap and joined-up thinking are the main causes for failure when games are introduced into the classroom. Ireland needs a solution that can be used by all primary schools around the country, one that can be easily rolled out, maintained, updated and designed specifically for the Irish primary school curriculum.

### 1.3 Proposed objectives

The main objective for this project is to build a reliable and extensible educational platform specifically for use within Irish primary schools. Other proposed objectives include:

- **Designing a suitable user interface for children.**
  
  Analysing current software products designed for children currently on the market and researching user interface design. (See 2.5)

- **Researching motivation and encouragement techniques.**
  
  Learning is not an easy task as it requires time, patience and a great amount of effort. I will research various methods used in order to encourage children to use a software product, keeping their attention and motivating them to continue. (See 2.6)

- **Design an API for developers to integrate their own games.**
  
  Once the core platform framework is complete, I will design an application programming interface (API) for developers to seamlessly integrate their games within the educational platform and gain access to high score lists, achievements and other platform functionality. (See 5.6)
2. Research

2.1 Introduction
This chapter outlines the analysis undertaken of educational learning platforms currently on the market, discussions with educational professionals and how our platform was specifically designed with the Irish primary school curriculum in mind. This chapter also discusses the research undertaken in user interface design and motivational techniques for young children.

2.2 Analysis of existing products
Before starting development on our educational platform, I decided to analyse the existing products on the market. After extensive research online, it would appear that no localisation awareness package currently exists. Any game that deals with such issues is often poorly designed with no built-in reward system, motivation factor or consistency between gaming experiences.

Many of the existing educational software packages available were built for one-off deployment and do not offer new modules or software updates. As these products were designed to implementation, they do not offer an extensible framework in which to provide new content and material.

I also examined the user interface characteristics of many of the games available at prongo.com, primarygames.com and performed a heuristic evaluation of various educational software products currently on the market. The results show that the user interfaces for many of these products were poorly designed, thus affecting the child’s tolerance level.

Figure 1 is just one example of a poorly implemented user interface. It is taken from the educational suite “Young Einstein Mathematics”\(^1\). This user interface is difficult to understand and to navigate since initially, it is not clear what the numbers along the road represent or why many of the signposts are repeated a number of times. There is almost no consistency between forms and the language used throughout the system has not tailored for its target audience.

I attended the London BETT (British Educational Training and Technology)\(^2\) show in January 2010 to gain a better understanding of the current products on the market. This annual three day exhibition allows hardware and software developers from all around the world to demonstrate their latest products and services on-site to educational professionals. The United Kingdom currently has a number of well-established, commercial virtual learning environments on the market, most noticeably the Frog learning platform\(^3\), DB Primary\(^4\), UniServity\(^5\), I am learning\(^6\) and Manga High\(^7\). While not all of these educational platforms adopt a game-based learning strategy, it was encouraging to see that the basic architecture and feature set was similar to the one outlined in this thesis.

\(^2\) BETT homepage - http://www.bettshow.com
\(^3\) Frog learning platform - http://www.frogtrade.com
\(^4\) DB Primary virtual learning platform - http://www.getprimary.com
\(^5\) UniServity connected learning communities - http://uniservity.com
\(^6\) I am learning Games based revision and assessment - http://www.iamlearning.co.uk
\(^7\) Manga High games-based-learning - http://www.mangahigh.com
2.3 Meeting educational professionals

As this educational platform is being designed for primary school children, I thought that it would be beneficial to arrange a meeting with a primary school teacher for research purposes. I called to my local primary school to meet up with one of the teachers, Margaret McCabe, to discuss the project in detail and to ascertain if localisation awareness is currently being thought within the classroom.

She explained how computers are being utilised within the classroom and provided me with a copy of the National Information and Communications Technology (ICT) in Schools Inspectorate Evaluation\(^8\). This document focuses on the development of ICT in the classroom and its contribution to teaching and learning in primary schools amongst other topics.

As the discussion progressed it became apparent that localisation awareness is not taught in class as it is not part of the official primary school curriculum set by the National Council for Curriculum and Assessment (NCCA) in Ireland (Curriculum and Assessment). This meant that I was unable to design the educational games based on any exiting course material and that I would have to research various multicultural topics for inclusion within the platform.

Margaret also suggested that I get in contact with John Hurley, the owner of H2.ie. This website assists organisations design and implement innovative curriculum based projects that utilise ICT and digital media effectively. After meeting up with John and having a long discussion about the factors that motivate children to learn and design techniques he suggested that I purchase a copy of “Digital Game-Based Learning” (Prensky, 2007). This book explains how “digital natives” have a short attention span for the old methods of teaching, but not for learning through the medium of fun, interactive games. The author also goes into great depth explaining how games teach, why they work and how to successfully implement your own educational games.

It was also necessary to find a school to test out the finished product with. As the current primary school curriculum does not teach many of the localisation awareness topics that the platform was originally designed to cover, we decided to focus on the

---

\(^8\) National ICT in Schools Inspectorate Evaluation
independent “Educate Together” multi-denominational schools around Ireland where “all children having equal rights of access to the school and the social, cultural and religious background of each child to be equally respected”. There are currently 52 Educate Together schools around Ireland. Information regarding this project was published in their December 2009 newsletter and a number of schools contacted us expressing interest in the final product. We decided to focus on local Educate Together schools and got into discussion with two primary school teachers from the “Limerick School Project” over the course of the year.

2.4 Curriculum integration

From the very start it was clear that these educational games should be explicitly designed to complement the current primary school curriculum rather than being an extra classroom activity covering extra-curricular subjects. Teachers need to benefit from our platform just as much as the students do. If these games are outside the scope of the curriculum or do not present the material in an easier to understand format, then it’s unlikely that the software will be used in schools.

SPHE (Social, Personal and Health Education) was identified as one of the key curriculum strands where the platform could be utilised in the classroom. SPHE covers “prejudice, discrimination, developing and sustaining relationships based on mutual respect, take pride in their national, European and global identities and come to an understanding of what it means to be a citizen in the widest sense.” However, the recommended time allocation for SPHE is just one class period per week. This is not enough time to justify using a complete software package, so it was decided that the platform should be pitched as a general educational tool. For example, the pen-pal correspondence tools (see 5.5.9) may be used as an educational English activity, cultural diversity and appreciation games may fit into SPHE lessons and the Europe explorer game (see 5.8.1) could be used in geography class to teach the map of Europe as its primary goal and to offer extra cultural information on all of these countries as a secondary goal, not necessarily outlined in the national curriculum.

---

9 Educate Together - http://www.educatetogether.ie
10 Limerick School Project - http://www.limerickschoolproject.ie
ICT is another subject that is often neglected in many Irish primary schools. The official NCCA curriculum guidelines\(^{13}\) explain how ICT can be used to support the learning and teaching of SPHE in the classroom. “Children can develop self-confidence and motivation through using and becoming familiar with computers. Other aspects of SPHE such as learning to take turns, to share and to co-operate can be developed as children use computers in a planned and appropriate manner. Computers can also be particularly helpful in enhancing children’s decision-making skills and in helping them to become discerning and judicious users of various technologies.

Children can use computers in SPHE to gather information on specific topics or to collate and present data. While there are many CD-ROMs available on health and social issues, the internet also provides a wealth of up-to-date information and can indicate further avenues of investigation. Exploring the internet, and using e-mail or video conferencing can enhance children’s sense of global citizenship and foster a wide range of communication skills.”

2.5 Designing a user interface for children

The user interface for this educational platform was one aspect of the project that I spent a lot of time researching. The user interface is often one of the main factors that affect the acceptance of software product, especially those aimed at young children. Designing a user interface for children is far more complex than designing one suited for adults. Children are far more critical and expect an intuitive software experience. In general, children find it hard to concentrate, have a short attention span and tend not to read instructions presented to them on-screen. They also like a certain amount of realism in the product.

I also researched the one laptop per child\(^{14}\) (OLPC) initiative before I started to design my own user interface. The OLPC initiative is a non-profit organisation whose goal is to introduce children in third world countries to computers for the first time. As these children have no preconceived notions of computers or operating systems it was the perfect opportunity to re-design the user interface from the ground-up to cater for the needs of children. The operating system they designed is

---

\(^{13}\) NCCA curriculum guidelines - http://www.curriculumonline.ie

\(^{14}\) One laptop per child initiative - http://laptop.org
widely praised for being extremely visual and powerful while remaining intuitive. Many of the metaphors that adults have become accustomed to over the last number of years such as the desktop and recycle bin were replaced with those that children can relate to.

The following techniques were adopted throughout the platform:

- **Interactive content**
  Children are curious and enjoy exploring the user interface for interactive elements. To aid the discovery process, I set the cursor to a hand symbol whenever the user is hovering over an interactive element. This encourages the student to click on the element and find new functionality within the platform without relying on text labels.

- **Tooltip labels**
  “Highly visual menus and icons appear to be appealing to children and easy for them to understand and use.” (Wilson, 1988) I avoided using text labels throughout the platform as research has shown that young children don’t take the time to read labels, instead they use visual imagery. Each interactive element has an associated tooltip attached to it. This allows the student to find out more about the element by hovering over it and reading the text that pops up.

- **Single mouse click**
  Young children often have difficulty interacting with a mouse especially when it comes to complicated actions such as drag and drop. The point and click approach should be adopted instead whenever necessary. To remove any confusion and to promote consistency, all actions throughout the platform are assigned to the left mouse button by default.

- **Avoid cartoon imagery**
  Children tend to like realism. “Oftentimes adults, and especially designers, underestimate children and their intelligence. Just because they are young doesn't mean they want to constantly be associated with cartoons or simple
design.” (Jones, 2008) Studies have also shown that cartoon figures and animations are known to be potentially distracting for users.

### 2.6 Motivation and encouragement techniques

Achievements are used as a form of motivation in our platform. The primary reason for providing motivation is because learning is not an easy task; it requires a lot of time and effort. Each plugin has the ability to award achievements for completing certain set goals. The title, description and accompanying value of each achievement is assigned by the plugin developer and stored in the platform database. “A trophy presentation is not only a great idea, but it will motivate the receiver to do more, to do better.” (Selvon, 2006) Achievements motivate the students to repeat a given task until they succeed. They can be used for a variety of purposes such as unlocking additional content, rewards or new areas in the game.

In-game achievements are an evolution of the high score concept. High score lists were popular on classic arcade machines back in the 80’s when gaming was a more social experience. Since the advent of cheap home game consoles such as the Nintendo Entertainment System (NES) and the Sega Mega Drive, gaming had become a more solitary past-time and high score lists quickly lost all purpose. Once home consoles got the ability go online and connect with each other, natural competitiveness came back in the form of in-game achievements, a global high score list that highlights the gamers overall skill and experience, rather than their rank in one particular game.

Many of the game consoles currently on the market have adopted a similar reward system. The Xbox 360 offers achievements while the PlayStation 3 rewards the user with trophies and allows them to display their trophy cabinet online. Over the last few years there has been a rapid increase of the number of social gaming providers. OpenFeint\(^\text{15}\), the leading social gaming platform for the iPhone OS allows game developers to add social gaming features to their own games such as friends, leader boards, achievements, matchmaking, forums and live chat. Since the initial release back in February 2009, their software has been bundled with over 1000 iPhone applications - a true testament to the popularity of social gaming. Other platform providers include Plus+, Crystal, Gameloft Live and Scoreloop for the iPhone OS, OpenFeint social gaming network - http://www.openfeint.com

\(^\text{15}\)
Steam for the PC and Mac and Games for Windows Live for Microsoft Windows operating systems. Apple has also ventured into this territory by integrating a dedicated game centre application into iOS 4.1, the latest version of their mobile operating system used on iPhone, iPod touch and iPad devices.
3. Literature Review

In this chapter I will review the critical points outlined by several publications.

For a general introduction to the concept of game-based interactive learning, I picked up “Digital Game-Based Learning” (Prensky, 2007). This book provides in-depth analyses of the old method of tell-test teaching and explains why it is not as effective as it was before the advent of technology. The book also explains how stimulation of various kinds “actually changes brain structure and affects how people think” and that these transformations go on throughout life. The author, who coined the phrase “digital natives” to describe students of the 21st century, claims that although today’s students have a short attention span for the old methods of teaching, they have a thirst for knowledge through the medium of fun interactive games. Using Sesame Street, the TV show, as an example to demonstrate that if you can keep children’s attention for long enough then you can educate them. The author explains how these “digital natives” crave to be the active participant rather than the passive observer. “We have to stop telling because almost nobody is listening.” The educational system is out of date, designed for a previous generation and needs a radical overhaul. “The learner has no choice in the curriculum, and has no choice in the way it is presented.” Every student is treated equally, and obviously this approach is not working. “Even with a computer in every classroom – or even a laptop for every learner – these computers, even with the internet, will not by themselves produce a better educated population” Software is the key which is the driving force for education. (Prensky, 2007) goes on to explain how games can replace the old tell-test method of teaching. A fun game will keep a child’s attention and for an educational game to be successful, it must first be fun and then only educational. This type of game will be learning that the child wants to do – stealth learning is the key. “When we enjoy learning, we learn better.” The author also expands on his theory of how and why digital game-based learning works and explains the complexity of designing a good educational game. “Not enough emphasis on learning and we risk falling into being just a game. Not enough emphasis on engagement and we risk sliding into computer based training.”
“Their Space” (Green, Hannon, 2007), a recent article published by DEMOS in 2007 attempts to dispel common myths about youth culture and explore the “real value behind digital interaction that are part of everyday life.” This article investigates what young people are learning outside the classroom, how schools can build on top of it and suggestions for further research. The finding of their report shows that “the use of digital technology has been completely normalised by this generation, and it is now fully integrated into their daily lives”. Their research has also identified a number of different user ‘types’ such as the “Information gatherers - Google and Wikipedia addicts” and the “Creative producer - those who build websites, post movies, photos and music to share with friends, family and beyond.” This article re-enforces many of the points raised by (Prensky, 2007). In order to see change in our educational system, “there needs to be a shift in thinking about investment from hardware towards relationships and networks.” We need to listen to the students if we are to develop a bridge between learning strategies and fun activities. “There is a ‘disconnect’ between the people who make the decisions and those who are experiencing the results in schools” and we are at a risk of alienating a generation of young people. Rather than harnessing the technologies that are already fully integrated into young people’s daily lives, schools primarily have a “battening down the hatches” approach. This article explains what motivates young people to learn and how “informal digital activities” has changed their learning styles. Children learn from each other and share knowledge on a horizontal basis. They feel comfortable blurring the lines between teacher and student. The article ends by explaining how the school structure can be changed and analyses the reasons why we haven’t made the necessary changes already. Although many schools have identified the existence of the problem, they have gone about fixing it in the wrong way. Delivering the latest hardware is only the first step in dealing with this problem. The authors offer a few simple changes that could easily be introduced such as abandoning the 20th century concept of homework and focusing on the “essential skills of evaluating information” instead of memorising and recalling.

Before starting to develop our platform, it was necessary to understand how Irish primary schools currently operate and how computers are currently utilised within the classroom. I obtained a copy of the “ICT in schools” inspectorate evaluation study (Department of Education and Science, 2006) from the department website.
This report documents how ICT is integrated with the curriculum, outlines the investment made in hardware and software over the last number of years and analyses the use of computers in schools – by teachers, students and staff.

Another interesting Irish article was “Teaching Multiculturalism in Three Schools in the Derry City Council Area” (Delargy, 2007). This article is based on a six month research project looking at the teaching of multiculturalism in three primary schools in Derry. The purpose of the author’s research was to examine the various issues faced by teachers and students in attempting to the teaching of multiculturalism in schools. As part of the research project, (Delargy, 2007) made a number of visits to the participating schools to observe classroom activities and to see how children of different nationalities and denominations interacted with each other. The author examined the multicultural activities and lessons taught within the class, religious education, and multicultural reading material available in the school libraries and identified a number of instances where “different traditions and cultural practices were woven into the curriculum.” The author believes that the Irish curriculum needs to be updated to embrace the concept of equality and multiculturalism within the classroom.

These are only a handful of authors that influenced my research over the last few months. (McKenna, 2009) for instance provided a vital overview of the three main hazard areas for primary school children: personal privacy, data protection and child abuse. This report, commissioned on behalf of the Barnardos children’s charity influenced the security aspects of the system. (Ross, 2006) wrote an interesting paper which compares “primary school children’s views on cooperation and competition in England, Slovenia and Hungary”. His research forced me to re-examine the social aspects of our platform while (Selvon, 2006) went on to confirm my overall belief that achievements, trophies and high-score lists are a great form of motivation for primary school children.
4. Design

4.1 Introduction
This chapter outlines the requirements elicitation process, explains the conscious design decisions, the security features and documents the final database schema.

4.2 Requirements
“Requirements Elicitation is the process of discovering the requirements for a system by communication with customers, system users and others who have a stake in the system development.” (Madigan, 2001)

Before starting development on our platform, I compiled a concise list of functional and non-functional requirements by identifying the stakeholders, taking their needs into account and creating a list of requirements.

These requirements helped me focus during the software development stage. Without a clear and unambiguous list of requirements, it is difficult for the software developer to know what to build or how the modules should function and interact.

4.2.1 Functional requirements
Functional requirements define the intended behaviour of the system or an individual component. They describe the internal workings of the software and explain how Unified Modelling Language (UML) use-case scenarios are satisfied.

The UML allows developers to create a graphical representation of a software system. UML provides 13 different types of diagrams, each covering a particular aspect of the sub-system. A use case is one specific UML diagram that outlines the functionality provided by the system. The main purpose of a use case diagram is to demonstrate what roles each system user (represented as actors) can perform.

Figure 2 is a simple use-case diagram that describes the basic tasks that teachers can perform on our platform. If the teacher provides valid login credentials, he/she can login, manage students, manage classes and view all online reports submitted by other platform users.
Figure 2: Teachers use-case diagram

Figure 3 is a use-case diagram that outlines the tasks that student platform users can perform on our platform. If the student provides valid login credentials, he/she can login, view their profile, view their friend list, search for other platform users, manage their platform messages, view achievements and launch plugin games.
Figure 3 also has two extension points which “describes a point in a use case where an extending use case may provide additional behaviour.” For example, logged-in users can only edit their own profile page. Likewise, platform users cannot send friend requests to themselves or to those already on their friends list.

![Students use-case diagram](image-url)

Figure 3: Students use-case diagram

---

16 UML extension points - http://argouml-stats.tigris.org/documentation/manual-0.26/ch17s04.html
4.2.2 Non-functional requirements
Non-functional requirements encapsulate all user requirements that are not explicitly functional. These requirements can be seen as constraints for implementing the functional requirements defined in the use case diagram and are often called “quality requirements”. Non-functional requirements are gathered by identifying the stakeholders and identifying their needs. Primary school students, aged eight to twelve years old are the main stakeholders in our platform. The following are the three main non-functional requirements identified:

- Usability
- Performance
- Extendibility

Usability
Usability was identified as one of the main non-functional requirement for this project. Young children are impatient and expect their software to be simple, intuitive and fun to use. Adults are much more tolerant with software and can learn to adapt, whereas children usually get frustrated and move on to another product.

Performance
Both speed and performance are important non-functional requirements for this project. Most children are impatient when it comes to waiting for content to load, especially educational content. Children are constantly making the payoff vs. patience decision (Prensky, 2007). If the payoff is not seen to be worth the wait then the children quickly abandon the idea and move on to something else for instant gratification. Every attempt must be made to improve speed and performance. For this reason, the size of the core platform framework should be kept to a minimum and all plugin content should be cached by the Microsoft Silverlight framework, resulting in faster subsequent loading times.

Extendibility
One of the main goals of this project is to create a platform framework that facilitates effortless expansion in the future. It is necessary to design a simple, yet powerful application programming interface (API) to facilitate platform extendibility. It is also
important that no modification to the source code of the core platform framework is required to facilitate expansion.

4.3 Design decisions

4.3.1 Desktop vs. browser

Before deciding what language to develop this educational learning platform in, it was first necessary to identify what environment this application would be built on.

Desktop applications have the advantage of always being available. The user is not burdened with software updates, loading screens or waiting for content to buffer. They are guaranteed a fast and satisfying experience every time as the content is stored locally on the desktop machine. Unfortunately, this also means that the user’s settings and progression are also stored locally on the host machine. This limits the user’s interaction to just one machine and inhibits portability and accessibility. The content associated with desktop applications is often out of date as the software has to be manually updated.

I considered using the Microsoft XNA framework\textsuperscript{17} for this purpose. This framework consists of a set of tools with a managed runtime environment provided by Microsoft that facilitates computer game development. The XNA framework is arguably more suited for the creation of games but as the end-user is required to download and install the additional XNA runtime, it did not seem like a practical solution for this project. Using the XNA runtime would also limit the games to the Microsoft Windows platform.

Web based applications have the advantage of being able to provide the latest content directly to the user. These types of applications store user progress in a centralised location allowing the user to access their account on any machine connected to the internet. In contrast to desktop applications, speed and availability are the main deterring factors of web based applications. The user will only be able to access the application in the presence of an internet connection and even then, they’re not guaranteed a quality of service. Congestion or contention at either the client or server side can result in a slow and highly frustrating end-user experience.

As this Masters is a continuation of my final year project, I had accumulated a large amount of code from the previous year in the form of a hybrid web-aware desktop

\textsuperscript{17}Microsoft XNA - http://en.wikipedia.org/wiki/Microsoft_XNA
application. Taking into account the feedback from those who evaluated the original software product and the evolution of the user requirements since the original inception, I decided that it would be best to start again, pick a more appropriate development environment and learn from my previous mistakes. This educational learning platform is built using the Silverlight development framework (see 5.2.2). It is a small-footprint, multi-platform solution that offers the best features of both the web and desktop.

4.3.2 Client implementation

After deciding what environment this application would be built on, it was then necessary to decide which language our platform would be written in. The three main options for web-based development are Flash, Silverlight and HTML 5.

Adobe Flash is the leading platform for creating rich internet applications (RIA) online. Flash uses an object-oriented scripting language called Action Script and delivers its content using a cross-platform browser plugin. Since this project is a continuation of my final year project, originally written as a desktop program in C# using the Windows Presentation Foundation (WPF) graphical sub-system, it would be necessary to re-write the whole application again if flash was adopted. Although Flash has a 96% penetration rate across all major desktop operating systems\(^{18}\), it is considered to be buggy, archaic and extremely resource intensive, often consuming 100% of available CPU cycles.

HTML 5 was the next technology I considered using. HTML 5 is the latest revision of the HTML (Hyper-Text Mark-up Language) web-standard. HTML, in conjunction with other standard web technologies such as JavaScript, CSS and XML could essentially eliminate the need for various third party browser plugins such as Adobe Flash, Microsoft Silverlight, Sun Java runtime environment and Google Gears. Although many of the features of HTML 5 have been implemented into modern browsers, the specification has not yet been finalised and use of this technology does not guarantee a consistent experience across all operating systems and web browsers.

Microsoft Silverlight (see 5.2.2) was the development platform I adopted to implement this game-based virtual learning environment. As Silverlight is a strict subset of WPF, much of the work undertaken during my final year project could

\(^{18}\) Rich Internet Application Statistics, August 2010 - http://www.riastats.com
easily be ported to the new platform. It is a small-footprint (less than 6mb in size), multi-platform solution that works natively on Windows and Mac OS X.

4.3.3 Classroom competition
I had always envisaged highly interactive scenarios while designing this educational platform where the students could form relationships, communicate and compare results, high scores and achievements with one another. However, one of the main criticisms I got from teachers and professionals was in relation to the promotion of competition within the classroom environment.

After some research, I found out that the United Kingdom and much of Europe are moving towards a more formalised education system in which interaction between students is closely monitored and controlled.

I found an article which compares “Primary school children’s views on cooperation and competition in England, Slovenia and Hungary” (Ross, 2006). The statistics for this report were gathered by interviewing children between eight and nine from four different schools in each country. Each group were asked the same questions in order for cross-national comparison to be achieved. The results revealed that competition still exists in UK Schools however teachers “discourage competition to prevent arguments and so that there isn’t any fighting.” They will also discourage competition in order to spare the children’s emotions so they “don’t feel bad.”

In the end I decided to keep the highly interactive, social gaming features such as high scores, achievements and profiles as I feel that they are vital to the success of this platform. Children aren’t afraid to look foolish or make mistakes in front of the computer. They just shrug it off and try again. As Ollie Bray19, the national adviser for ‘Learning and Technology Futures’ says, “Good teachers use good tools, play and interaction was always used in classrooms.”

4.4 Security features

4.4.1 Importance of security
In a report commissioned by the Barnardos children's charity, (McKenna, 2009) focuses on three primary hazard areas for primary school children.

19 Ollie Bray homepage - http://olliebray.typepad.com
• Personal Privacy
  “The policy and means through which children are individually identified, and can maintain selective data privacy.”

• Data Protection
  “The security and administration of stored data about children, and the policy and physical means through which that data is accessed, and transported physically or through electronic networks.”

• Child Sex Abuse Imagery (CSAI)
  “Restricting access to the means through which such unlawful media is available to consumers.”

This virtual learning environment attempts to address many of the issues highlighted in the report. However, the security of our platform or any other software system cannot simply be declared; rather it must be proven over time. To reduce the chances of a security exploit, recognised industry security standards, best practices and guidelines were adopted and implemented throughout the platform.

4.4.2 Password protection
The passwords for all user accounts on our platform are stored securely in the database using SHA-1, the industry standard secure hash algorithm. As this is a hashing algorithm, the encryption is one-way. The original plain-text password cannot be derived from the 160-bit hash. In order to validate login credentials, the system must generate a new hash for the password provided and compare the hash to the one stored in the database for the corresponding username. If both hashes match then the user is authenticated.

Although SHA-1 is one of the most secure and widely used hashing algorithms, it's still prone to common hacking techniques such as dictionary attacks. This brute force form of attack compares the user’s hashed password against a large list of pre-compiled SHA-1 hashes comprising of large multi-lingual dictionaries, names and commonly used passwords. If both hashes match, then the hacker is able to determine the original password used to generate this hash. To combat this problem, extra information is added to the plain-text password before the hash is generated.
This process, known as salting, will ensure that the password will not appear in any prebuilt SHA-1 dictionary and consequently will not suffer from this form of attack. Even the slightest variation in the input such as character capitalisation will produce a completely different hash.

However, if a static salt is used for all passwords and discovered at a later date, then the security advantage of adding extra information to the password is lost. A custom dictionary hash table could easily be generated, thus exposing all weak passwords on the system. The most robust solution is to create a unique salt for each user. In this case, if the random information is ever exposed, a complete custom hash table would need to be generated for each user on the system.

4.4.3 Student privacy
To protect the privacy of all students online, communication should only occur between classmates and friends. Before students become mutual friends, a friend request must be sent by one student and accepted by the other. The recipient of the friend request is also free to deny the request or block any further communication.

To improve the overall experience of this collaborative educational platform, profile privacy is set to “public” by default. Those students who are more concerned about their privacy have the option to opt-out and set their profile privacy settings to “private”. This will ensure that their profile is only accessible to existing friends and classmates. It also prevents their profile from appearing in public student searches, other students viewing their profile, comparing high scores and achievements, friend requests and communication through private messages.

4.4.4 Reporting
In April 2010, the Child Exploitation and Online Protection (CEOP) centre proposed the idea of a "panic button" to be installed on all popular social networking sites such as Facebook, Bebo and MySpace to protect children from "cyber bullying, hacking, viruses, distressing material and inappropriate sexual behaviour" (Hogan, 2009). Although they were unsuccessful in their attempt to integrate the panic button on Facebook, they have since adapted their solution to integrate with the external application facility provided by the Facebook SDK (Software Development Kit).

20 CEOP Facebook application - http://apps.facebook.com/clickceop
Each student using this educational learning platform can create a detailed profile, upload material and communicate directly with teachers, friends and classmates. With all open systems comes a certain amount of abuse. To limit the effects of this abuse, a dedicated reporting system has been built into the platform framework. Manual reports can currently be raised by platform users once they come in contact with unsavoury material online. Although this feature is highly requested by educational professionals, it is ultimately a reactive form of protection as the damage has already been done. It may take a few hours or even days for the teacher to login, notice the report and take appropriate action. The reporting functionality could be expanded in the future to include pre-emptive reporting strategies (see 7.3.4).

4.4.5 SQL injection
Structured Query Language (SQL) injection is a “code injection technique that exploits a security vulnerability occurring in the database layer of an application” (Watson, 2006). The risk of an SQL injection exists when an SQL query accepts data from the end user without first sanitising the input or defining appropriate parameters for the data. If the database schema is obtained, then malicious code can be added to the original query as part of the user input. SQL injections often occur when standard table and field names are adopted. For example, assuming that user details are stored in a "users" table and that a "username" and "password" field exists, a malicious user could add a query in the username input field to change the password of the "administrator", obtain all user details or drop (delete) all information in the database.

Another common method of circumventing SQL–based login systems is to provide additional logic that always equates to true. For example, instead of just entering a password in the login field, a malicious user could enter any random information followed by an equation that is guaranteed to be true. Without proper parameterisation, this string is merged with the SQL query and no distinction is made between the literal values provided by the user and the executable code defined by the software developer. In this case, the password supplied is more than likely incorrect, but the query returns a true value as 1 is always equal to 1.

SQL injections are prevented by constructing SQL statements using type-safe SQL parameters for each input field. SQL parameters are non-executable, literal values
with field length validation. If malicious code is entered as part of the parameterised statement it will be treated as a literal string value, unable to execute, having no effect on the integrity of the database. In this project, as Silverlight applications are unable to communicate directly with the database, SQL prevention is implemented at the web service layer. All information supplied by the user is inserted into pre-defined parameterised SQL queries and executed by the web service.

4.5 Database

4.5.1 Database structure

After constructing the use cases diagrams for our platform (see 4.2.1), a number of different user types and roles were identified. Many of these entities such as administrators, school, teachers, students and guardians share common attributes, so to reduce database redundancy and complexity, a schema based on table inheritance was designed.

Figure 4: Database table inheritance
Figure 4 is a hierarchical representation of the user type tables. Login credentials, name, gender, date of birth, etc are all stored in the "users" base table as these common attributes apply to all platform users. Sub-type tables need to be created for every defined user type. These tables only contain information directly associated with each of the entities. For example, the "guardians" table requires extra contact information that is not relevant to all platform users.

This inheritance structure drastically reduces the complexity of the schema along with the number of associated web service methods required for common tasks. For example, we only need one web service method to update a user's last login: UpdateLastLogin, rather than four separate methods that perform similar tasks: UpdateSchoolLastLogin, UpdateTeacherLastLogin, UpdateStudentLastLogin and UpdateGuardianLastLogin.

4.5.2 Multilingual support
As our platform is designed to cater for a number of different languages, it is necessary that the client application, web service and database environment all adopt the same character set and encoding. The Unicode character set was used for this project as it contains all major living scripts and is the global standard for information interchange over the internet. UTF-8, the variable length character encoding was used to encode all information in the MySQL database. As UTF-8 is a strict superset of ASCII, it can store and represent English and most European languages using fewer bits per character, drastically reducing the size of the database as the platform expands over time.

4.5.3 Normalisation and referential integrity
Database normalisation is the process of designing a database schema with minimal redundancy in order to avoid update and deletion anomalies. Redundancy occurs when the same information is stored in multiple relations, or when derived data is stored in the database. Update anomalies occur when information is modified in one relation and is not updated in associated relations. Deletion anomalies occur when information is inadvertently lost because of the deletion of other attributes. Both of these characteristics are eliminated when the schema is normalised, more specifically, when the schema is normalised into 3rd Normal form (3NF).
Referential integrity is another key aspect of schema development. It ensures that data remains consistent between all relations by enforcing business rules at the database level. For example, information about the platform users is distributed over multiple relations. If a user’s account is deleted, then we also want all associated information to be removed from the database. Many developers try to enforce these rules at the application level; however, as the project evolves over time, this manual method of enforcement increases the chances of data irregularity. As MyISAM, the default MySQL storage engine is unable to enforce referential integrity through the use of constraints; the InnoDB storage engine was used in its place.

4.5.4 Database optimisation
To improve the overall performance of the database, many of the larger database tables were broken down into smaller, logical tables. Decoupling the information in the database, extensive of primary keys (which are naturally indexed) and use of pivot-tables greatly increase the performance of standard SQL queries.

To retrieve information from a MySQL database, one or more SQL queries must be executed against it. The .NET framework currently offers three methods of executing queries – ExecuteReader, ExecuteNonQuery and ExecuteScalar. For optimal database performance and increased scalability, it’s important to choose the most efficient execution method based on the expected query results.

`ExecuteReader` should only be used when a query is expected to return one or more rows due to the high overhead involved in the setup and disposal of a database reader. `ExecuteNonQuery` should be used whenever the application is performing Create, Read, Update and Delete (CRUD) functions and only requires to know the number of rows affected by the operation. Finally `ExecuteScalar` should be used when the application is only interested in a single value, from one column, in one row of a database table. This method returns a single object. If the object has a null value then the query has failed, otherwise it can be cast into the desired data type.

4.5.5 Final schema
The following figures represent the MySQL schema designed for this project. As the final schema contains over 30 interconnected tables, it was impossible to represent the all of the tables and their associated relationships in one clear diagram. So, for
the purpose of the thesis, the schema was broken down into three distinct views – student, school and platform communication.

Figure 5 is a visual representation of all the database tables associated with the student and the user base class entity. The *users* table contains the main information about all platform users. Each user has a *user_id* which uniquely identifies the user within the database. The *users* table also stores values such as the user’s forename, surname, date of birth and their hashed password value. This table also keeps foreign key references to other values such as the *gender_id*, *privacy_id* and *nationality_id*.

There are a number of reasons for storing foreign key references rather than plain-text literal values. First of all, this reduces duplication within the database; the real data is stored in another table and is assigned a unique ID. If the data needs to be modified at a later date it only needs to be updated in one location. Also, as the data has been decoupled and is represented by an auto-incremental integer value, this allows information to be indexed by the database management system and retrieved much faster than the conventional method of parsing every relationship record.

The *students* table is a sub-table of the *users* table. This table contains information relevant to the student entity that does not apply to the *users* base-table. In this case, all student entities are bound to one particular school. To find out more information about a student, such as their username, forename or surname, it is necessary to join the *students* and *users* table on a common attribute – the *user_id*.

The *user_relationships* table records the relationship between two platform users. All friendships must be confirmed before they can interact with each other. The *user_id* of the user who initiated the friendship is stored in the *user_id_one* field. The *relationship_types* table stores the different types of relationships two users can have (confirm, friends, blocked). Once the recipient user represented by the *user_id_two* field confirms or denies a friend request, the *user_relationships* table is modified to reflect the change.
Figure 5: Student database schema
Figure 6 outlines the tables associated with the school setup process and plugin allocation. Our platform has a number of plugin games to choose from. The information for each plugin is stored in the `plugins` table. Each plugin has a unique `plugin_id`, name, description, assembly filename, image filename and a foreign key reference to the developer who designed the plugin. Each plugin also has the ability to offer in-game achievements for completing certain set goals. These achievements are stored in the `plugin_achievements` table. This table keeps a foreign key reference to the plugin that the achievement is tied to, the name, description and value of the achievement along with the achievement image filename. This table does not store the criteria for unlocking the achievements; this logic must be implemented within the plugin game itself by passing the `plugin_id` and `student_id` to the web service whenever an achievement has been unlocked.

The `guardians`, `schools`, `teachers` and `students` tables store the information relevant to each entity. Each student and teacher is linked to one particular school through the foreign key reference - `school_id`. The table `student_guardians` is an example of a pivot table where one student can have a number of guardians. Likewise, a parent or guardian can be linked to a number of students within the platform.

Teachers are responsible for setting up online classrooms. This information is stored in the `classes` table. Each class has a `class_id` for identification purposes, a class name and a foreign key reference to the class level. The pivot tables `class_teachers` and `student_classes` allows for a number of teachers to manage the online class and for multiple students to be enrolled in the same class.
Figure 6: School database schema
Figure 7 outlines the tables associated with all forms of communication within the platform – private messages and reports. The hierarchical structure of the database and the extraction of common attributes in the form of a user entity allows for communication between all platform users, not just limited to individual sub-entities such as teachers, students or guardians.

Each private message is stored in the messages table, uniquely identified by a message_id. Each message stores a foreign key reference to both the sender and recipient, has an optional subject and body field and is given a DATE_TIME stamp. All private messages have a foreign key status_id reference to indicate its status - read or unread.

Reports are the second form of communication within the platform. All reports are stored in the reports table. Each report is uniquely identified by a report_id and stores a foreign key reference to both the user who lodged the report and the user who the report was filed against. Each report is filed under a particular category and has a severity level associated with it. This allows teachers to prioritise report resolution by addressing the most severe reports first. All reports have a foreign key status_id reference to indicate its status - read or unread.

See Appendix C for the full MySQL schema definition.
Figure 7: Platform communication schema
5. Implementation

5.1 Introduction
This chapter outlines the tools and technologies used during implementation along with a brief description of the newer one. This chapter also describes the implementation of the major learning platform components.

5.2 Implementation technologies
- Microsoft Visual Studio 2010
- Microsoft Expression Blend 4
- Microsoft Silverlight
- Windows Communication Foundation
- N-Unit testing framework

5.2.1 Microsoft Expression Blend 4
Expression Blend is just one of the applications in the Microsoft Expression Studio suite. It is a user interface design tool that offers a lot of the same functionality as Microsoft Visual Studio 2010 but specializes in animation and the creation of rich user interfaces for Windows Presentation Foundation (WPF) and Silverlight applications.

5.2.2 Microsoft Silverlight
Silverlight is a relatively new development platform used to create rich interactive applications for consumption on the internet. The Silverlight runtime is a cross-platform plugin that runs on Macs and PC's with all modern web browsers such as Internet Explorer, Firefox, Opera Safari and Chrome. The latest runtime offers many of the features of the .NET framework in a lightweight package less than 5 megabytes in size. An open source implementation of the runtime called Moonlight allows Silverlight applications to run natively on Linux machines. Although Silverlight has only been available for a few years, the latest rich internet application
penetration statistics show that approximately 65% of all operating systems already have Silverlight installed on their machines.\textsuperscript{21}

User interface elements in Silverlight are defined using a structured, XML based language called XAML (Extendable Application Mark-up Language). C# is the object-oriented programming language used to interact with the XAML interface and to define the business logic of the Silverlight application.

\section*{5.2.3 Windows Communication Foundation}
Windows Communication Foundation (WCF) is an amalgamation of various existing services and technologies into a unified library as part of the .NET framework. These services include XML web services, queuing and .NET Remoting. Although WCF services are hosted on the Windows Server environment, they can be consumed by a wide range of languages due to compliance with standards such as SOAP (Simple Object Access Protocol) and the ability to offer multiple service endpoint interfaces.

\section*{5.3 Software lifecycle}
“Agile software development is a group of software development methodologies based on iterative and incremental development, where requirements and solutions evolve through collaboration between self-organizing, cross-functional teams.”\textsuperscript{22}

As this project has been a learning process from start to finish, the agile development methodology was adopted once the basic platform requirements were obtained. This allowed me to be more flexible with the design decisions and adapt faster to changing requirements. Sketches and mock-ups were designed at different stages during development, providing instant feedback and clarification.

\section*{5.4 Teacher dashboard}
The dashboard provided for the teachers to monitor and control students, classes and assigned plugins is written completely in HTML and PHP. As the dashboard does not require rich graphics, animation or highly interactive components, a conscious decision was made to build a simple, highly accessible interface that requires no external player or runtime to operate. This implementation enables teachers to log on and use the dashboard from any internet connected device without having to worry

\begin{footnotesize}
\begin{itemize}
\item \textsuperscript{21} Rich Internet Application Statistics, August 2010 - http://www.riastats.com
\item \textsuperscript{22} Agile Software Development - http://en.wikipedia.org/wiki/Agile_software_development
\end{itemize}
\end{footnotesize}
about missing runtimes, user account limitations or incompatible Silverlight devices such as mobile phone and PDA’s.

The teacher dashboard offers the following functionality:

- Create student account;
- Create a new virtual classroom;
- Modify class details, assign plugin games, manage class list.

Each virtual classroom can have a different set of plugin games to cater for the class level, weekly agenda and specific curriculum requirements. This enables teachers to control exactly what material the students have access to rather than loading all of the plugin games available for the educational learning platform.

Teachers are also responsible for creating student accounts. This is normally a once-off task undertaken at the start of the year and does not take long to complete. I experimented with various other methods of student account registration but after talking to a number of primary school teachers and educational professionals they all agreed that the teacher should be responsible for account creation. This ensures that the students online identity is accurate, prevents the signup of malicious user’s masquerading as young children and provides a solid hierarchical responsibility link.

5.5 Client application

5.5.1 Login

The login UserControl (see Figure B1) is the first object presented to the student when they visit the educational learning platform website. Once the student enters their username and password, the password is salted and hashed at the client side (see 4.4.2) and a request is made to the web service’s AuthenticateUserAsync method to validate the login credentials supplied. If the username and password are valid, an instance of the StudentPlatform class is instantiated and an asynchronous call is made to the GetStudentDetailsAsync and GetAssignedPluginsAsync method.

The GetStudentDetailsAsync method queries the MySQL database at the server and returns basic information about the platform user such as their StudentID, Forename, Surname, AvatarFilename and LastLogin. The GetAssignedPluginsAsync method returns a list of PluginDetails objects. These objects contain the PluginID,
PluginName, ImageFilename and AssemblyName of all the plugin games assigned by the teacher.

Once the student details and assigned plugins are acquired from the server, their encapsulating objects are passed to the StudentPlatform object and the current login object is disposed.

5.5.2 Homepage

Once the student has successfully logged in, they are brought to their custom homepage (see Figure B2) where all of the platform activities are displayed. The homepage contains three permanent links, one to their profile (see 5.5.4), friends list (see 5.5.5) and achievements page (see 5.5.7). The homepage will also contain a link to all of the games that the teacher has selected for use within the classroom.

For each assigned plugin, the homepage creates a new ExternalPlugin object passing in the PluginDetails as part of the objects parameter list. The ExternalPlugin object keeps a reference to the PluginDetails object, downloads the plugins thumbnail image from the ImageFilename location and creates a MouseButtonButtonDown event listener ready to launch the plugin once the student clicks on the thumbnail image.

5.5.3 Plugin downloader

Each platform plugin is an external Silverlight assembly saved on the server. To reduce overall loading times, these plugins are downloaded on demand rather than being bundled with the core platform assembly. This modular structure enables the platform to easily expand in the future and reduces the overall bandwidth required to operate and maintain the service.

Once the student clicks on a plugin thumbnail from the homepage, a PluginLauncher object is instantiated and is passed the PluginDetails reference associated with that plugin. The PluginLauncher object downloads the plugin assembly from the location specified in the AssemblyVersion field using an instance of the WebClient class. While the assembly is being downloaded, the student is kept updated on the progress through the use of a ProgressBar. Once the plugin assembly has been fully downloaded from the server, an attempt is made to cast it as an IPlugin object to allow it to fully interact with the platform framework. If the cast is successful then the platform will call the GetSplashScreen() method from the IPlugin object and the
game will begin. However, if the attempt to cast the assembly fails due to a transmission error, the user is returned to their homepage. Our platform also takes advantage of application storage feature provided by Silverlight, so, if the same version of an assembly has been downloaded before then it will load the cache rather than downloading the assembly again.

5.5.4 Profile
The students profile is a key social aspect of this game-based educational platform. Each student has their own profile page and has the ability to customise it and upload information for all their friends and classmates to see.

The profile UserControl (see Figure B3) is a polymorphic class that is capable of adapting to two similar use-case scenarios. The first being when the logged-in user views his/her own profile, and the second being when the logged-in user views any other profile page. As the structure and functionality of both use-case scenarios are very similar, it makes sense to combine them into one class rather than two separate classes. If the functionality needs to be modified at a later date, it only needs to be updated in one class. Overloaded constructors were used to differentiate between the two use-case scenarios.

The first constructor accepts a reference to an IStudentPlatform interface and a StudentsDetails object. The StudentsDetails object contains some basic information about the user, most importantly, their assigned StudentID. The second constructor does not take a StudentsDetails object as one of its parameters. This assumes that the profile to be fetched is the user’s own profile. Once the basic student details have been obtained, the class will then request the full profile from the web service by calling the asynchronous methods GetGamercard and GetStudentProfile.

The GetGamercard method returns the students gamer score, latest achievements and the name of the last plugin game played. The GetStudentProfile method returns a StudentProfile object containing the following information: full name, date of birth, nationality, latest activity, biography, avatar and list of interests and spoken languages. The StudentProfile object also contains a ProfileType enum (enumerated type) which determines the relationship between the logged-in user and the current student profile.
The `ProfileType` enum can have the following values:

- Personal
- Stranger
- Friend
- Confirm
- Request

If the `ProfileType` enum has the “Personal” value, then the current student profile is that of the logged-in user. In this case, the profile page will display a button to enable the user to modify their profile. If `ProfileType` is set to “Stranger” then the logged-in user has no specifically defined relationship with this student. In this case, the send friend request and block student buttons are displayed on-screen. The `ProfileType` enum is set to “Friend” if both students are friends. The logged-in user then has the option to either send the student a private message or remove them as a friend. The last two enum values relate to pending friend requests. If the logged-in user initiated a friend request, then the current student’s `ProfileType` will be set to “Request”. The logged-in user will not be able to directly contact this student until their friendship is confirmed. The last possible enum value is “Confirm”. If the student (whose profile the logged-in user is currently viewing) initiated a friend request, then the logged-in user has the option of either accepting or rejecting the friend request directly from the profile page.

5.5.5 Friends
This virtual learning environment was designed to be a social gaming experience. Interaction with classmates and friends are a key component of this vision. The “friends” sections allows the students to view all their friends, check when they were last online and gives them access to common tasks such as, viewing their profile, comparing achievements and sending private messages.

While the friends `UserControl` (see Figure B5) object is loading, a list of the student’s friends is requested by calling the asynchronous web service method, `GetFriendsAsync` and providing the `studentID` integer as the methods parameter. This method will return a list of `StudentDetails` objects from the web service. If the number of items in the list is zero, then the student has no friends. However, if the
list is not empty, a new StudentGrid object is created for each of the StudentDetails objects in the list and added as a child to the FriendsStackPanel object displayed on-screen.

5.5.6 Student search
From researching various localisation awareness topics it appears that pen-pal correspondence is an activity that can benefit all age groups. Keeping in contact with a pen-pal helps expand a child’s knowledge of the world. This activity helps teach children about different countries, cultures, race and opinions. Although children these days regularly keep in daily contact with each other through various social networking websites such as Bebo, Facebook or MySpace, research has found that they tend to use these facilities to keep in contact with existing friends and rarely find new friends using these services. In fact, many of these sites discourage this activity by only allowing you to browse your friends profile pages and in Bebo’s case, encouraging you to “only add people as friends if they are actually friends”.

Encouraging students to get in contact with children of different nationalities helps them appreciate the differences in cultures first hand. It also helps them to discover that all “children are the same for the most part. Games and music are but two topics in the international language of children.” (Brooker, 2005) Setting up a pen-pal programme helps children gain a better understanding of the world in which they live in, at a level they can relate to.

Students can use the built-in student search feature (see Figure B6) to find other platform users with similar interests. The search page initially loads a list of all of the students who have used the platform in the past (whose privacy settings are set to public) by invoking the GetSearchProfilesAsync web server method. From this list the user can instantly check out the students profile page, compare achievements and send friend requests. To cater for platform expansion and to provide more relevant search results, the user can filter the list of students by a number of different categories. These categories include interests, spoken languages, nationality, class level, gender and last active. Using a combination of these filters, students can easily find others that they would be interested in talking to. Once a filter is selected, the GetSearchProfilesAsync web server method is invoked again and passed an SQL sub-query containing the users filter clause. This query is appended to the end of the
GetSearchProfile’s generic SQL statement and the filtered list of students is returned to the Silverlight client.

5.5.7 Achievements

Our platform allows you to create an online identity and implements an achievements reward system (see Figure B7 and B8) that work consistently across all platform games.

When one of the educational plugin games wants to award an achievement, it calls the plugin API method AwardAchievement. This method takes a single integer parameter which indicates the achievementID to award to the student. Once this core platform framework method is invoked, it calls the asynchronous web service method AwardAchievement providing the current studentID (obtained from the CurrentStudent reference variable) and achievementID as method parameters. If the web service returns a null value then the student has already earned this achievement and no further action is taken. However, if the method returns an AchievementDetails object this indicates that the student has not previously earned this achievement and that the database has been updated successfully. The AchievementDetails object contains information about the achievement such as its image, title, description and value, which in turn will be displayed on-screen in the form of a pop-up toast notification.

As a number of achievements can be awarded at the same time, all achievements earned are delegated to the platforms notification manager. The notification manager maintains a queue of notification toasts and will display each of them on the screen for a brief period of time - one at a time.

5.5.8 High score list

Each plugin game for this educational learning platform has the ability to offer a high score list to record student performance (see Figure B11). As this is a common component in a lot of the platform games, the functionality was built into the core platform framework rather than being implemented in each individual plugin assembly.

When an external plugin wants to display the current high score list, it calls the ShowHighScoreList method defined by the IPluginHost interface. This method,
implemented by the StudentPlatform class consults the currentPluginDetails reference variable to obtain the PluginID of the current plugin. Using the PluginID as the only parameter, an asynchronous call is made to the GetGlobalHighScoreList web service method to obtain a descending list of all the students who have played the plugin and their associated high score value. Once this information is obtained, a new HighScore object is instantiated and pushed to the front of the breadcrumb navigation stack.

5.5.9 Message centre

The purpose of the message centre (see Figure B13) is to consolidate all platform messages into one, easy to manage area. These types of messages include friend requests and private messages from other students. The message centre can be accessed at any time by clicking on the envelope icon on the top right hand side of the screen.

The core platform framework is constantly probing the database to check for new messages. The CheckForStudentMessages web service method just returns a Boolean value (true or false) indicating the presence of new messages. If a new message is detected on the server, the message centre icon will change colour in an attempt to subtly grab the user’s attention. As this operation is repeated every five seconds, a lightweight query was designed to reduce the load on the server. The full body of each message is only downloaded once the user enters the message centre. Once the message centre is loaded the user is taken to their inbox and two asynchronous web server methods are invoked - GetPendingRelationships and GetMessageDetails.

The GetPendingRelationships method returns a list of friend requests that other students have made. The user then has the option of looking at the students profile, accepting the friend request, rejecting the friend request or blocking any further communication from this user. The GetMessageDetails method simply returns a list of private messages from the database. Before these messages are displayed on-screen, the unread messages and pending friend requests are moved to the top of the list and the other platform messages are sorted ascending by date.

The message centre is also where students go to send new messages (see Figure B14). Once the student clicks on the “Compose Message” button they are brought to
the message editor. Here they can choose the recipient and fill in the message subject and body like a regular email. For security reasons, students can only send messages to pre-existing friends. Once the student has completed all of the mandatory form fields, a MessageDetails object is instantiated and initiated with these values. This object is then passed to the SendPrivateMessage web service method. Once the WCF web service receives this information, it will create a new entry in the database and notify the recipient the next time they login to the system.

5.6 Plugin API

5.6.1 Introduction

To facilitate the implementation of a modular platform framework, all plugins must use the custom platform API (Application Programming Interface) in order to interact with the core platform framework. “APIs are a tool for programmers to make one piece of software talk to another piece of software” (Goel, 2010). The API provides a limited set of platform functions that can be called by the plugin through the use of shared interfaces. These interfaces are encapsulated in an external .dll file, separate from the platform implementation.

All interfaces created in this project use the common Java and C# interface naming convention which is to prefix all interface names with the capital letter ‘I’. This helps to quickly distinguish them from classes and abstract classes.

The plugin API contains three main interfaces for interacting with the platform: INavigationItem, IPluginHost, and IPlugin. Any plugin that fail to implement these interfaces will not be compatible with the platform framework.

5.6.2 INavigationItem

The INavigationItem interface is implemented by any UserControl object which can be inserted into the breadcrumb navigation bar. This interface defines four methods.

- GetControlName()
- ControlAdded()
- ControlRestored()
- ControlClosed()
The GetControlName method is called when a UserControl object is being added to the breadcrumb navigation bar. If no alternate name is provided during insertion, then the platform will call this method to get the UserControl’s default display name.

The ControlAdded method is called on the currently active object in the breadcrumb navigation bar whenever a new navigation item is added. As this object will no longer be visible on-screen, this method offers the object the opportunity to react to this event. For example, pausing the game or turning off all background music.

The ControlRestored method is called when a non-active UserControl object in the navigation bar is selected. As the object is once again the active navigation item and visible on-screen, this method allows the object to resume where it had left off.

The ControlClosed method is called on a UserControl object once it has been permanently removed from the breadcrumb navigation bar. This method can be used for a variety of purposes such as saving the state of the game, object cleanup and deallocation of system resources.

5.6.3 IPluginHost

The IPluginHost interface is implemented by the StudentPlatform class. This interface gives the plugins access to a limited subset of functions present in the platform framework. Seven methods are defined in this interface:

- Push(INavigationItem item)
- Push(INavigationItem item, String title)
- Pop()
- PopAll()
- PopBackTo(INavigationItem item)
- AwardAchievement(int achievementID)
- AwardHighScore(int score)

The Push method is used when a plugin wants to add a new object into the breadcrumb navigation bar. A reference to an INavigationItem object is passed to the IPluginHost. The overloaded version of this method also allows the plugin to specify
the name of the INavigationItem object displayed in the navigation bar. A generic class can be given a more specific display name using this overloaded method.

The Pop method is used to go back to the previous INavigationItem object on the breadcrumb navigation bar. This method will only execute successfully if the number of INavigationItem objects is greater than one as the home screen must always remain on the navigation bar.

The PopAll method is a faster way to remove all INavigationItem objects from the breadcrumb navigation bar and to return to the home screen. This method will continue to call the Pop method while the number of items in the navigation bar is greater than one.

The PopBackTo method is designed to keep popping off INavigationItem objects from the navigation bar until the current INavigationItem object is the same as the object specified in the methods parameters.

The AwardAchievement method is called whenever a student earns a plugin achievement. The unique achievementID is passed as the methods parameter. The IPluginHost then proceeds to make an asynchronous call to the platforms web service to see if the student has already earned this achievement. If the AwardAchievementAsync method returns nothing then the achievement has already been earned. However, if an AchievementDetails object is returned, this means that the achievement has not been earned before. In this case, a notification toast is shown on-screen with the information provided by the AchievementDetails objects.

The AwardHighScore method is called whenever a student obtains a score from one of the plugin games. The score integer is passed as the methods parameter. The IPluginHost then proceeds to call the AwardHighScoreAsync method from the platform web service providing the studentID, pluginID and score. If the database has no record of the student playing this plugin before, then the score provided is set as the highscore field in the student_high_scores table. However, if the student has already played the game before, then both values must be compared. If the score provided is greater than the current highscore value, then the database is updated. Otherwise, if the score provided is less than or equal to the current highscore value, then no change is made to the database.
5.6.4 IPlugin

The IPlugin interface is implemented by all platform plugins. This interface defines the methods necessary to initiate the plugin and for the platform framework to interact with the plugin.

This interface contains the following methods:

- RegisterPlatform (IPluginHost platform)
- GetSplashScreen()

The RegisterPlatform method is called by the IPluginHost object whenever a plugin is loaded into the platform framework. The IPluginHost object passes a reference to itself to the IPlugin object. This reference allows the plugin to interact with the platform framework via the methods defined in the IPluginHost interface.

The GetSplashScreen method is also called by the IPluginHost object whenever a plugin is loaded into the platform framework. This method returns an INavigationItem object which is added to front of the breadcrumb navigation stack.

5.7 Web service implementation

5.7.1 Introduction

Web services are online applications that enable the exchange of data between computers connected to the internet. They are commonly used as an interface for database management systems (DBMS). Silverlight client applications execute in a sand box, within the web browser. In order to interact with the MySQL database, the client needs to connect to the WCF web service and invoke predefined methods. These methods can be used to implement additional business logic and to limit how clients can interact with the database. Login validation is an example of one method implemented in the WCF web service. This method does not grant the client full access to the database, instead the SQL validation query is executed remotely by the web service and the result is returned to the client in the form of a Boolean return type.
5.7.2 Exception handling

“An exception is an event that occurs during the execution of a program that disrupts the normal flow of instructions during the execution of a program.”

Exceptions are generally unforeseen errors caused by invalid input, memory leaks, or other undesirable system conditions. Exception handling is the process of recovering from these errors and getting back to the normal flow of execution. The WCF web service is a vital element of the virtual learning environment. As each instance of the student platform depends on the web service for client-server negotiation, downtime must be kept to a minimum. Each web service method is encapsulated in a `try-catch-finally` block to deal with system exceptions allowing individual methods to fail gracefully instead of crashing the entire system.

```csharp
public Boolean UsernameExists(string username)
{
    Boolean exists = false;
    if (username != null && username.Length > 0)
    {
        try
        {
            this.connection.Open();
            MySqlCommand command = new MySqlCommand(
                String.Format("SELECT username FROM students WHERE username = '{0}'", username),
                this.connection);
            
            if (command.ExecuteScalar() != null)
            {
                exists = true;
            }
        } catch (Exception)
        {
            exists = false;
        }
    }
    finally
    {
        this.connection.Close();
    }
    return exists;
}
```

Figure 8: Sample web service method

The main logic of each WCF method is contained and executed in the `try` block. This block is generally followed by a number of `catch` blocks to deal with a number of different `Exception` types such as `IOException` or `InvalidCastException`. As all of the WCF exceptions for this web service are handled the same way, all exceptions are caught using the generic `Exception catch statement.`

---

23 micro5.mscc.huji.ac.il/~aries/Exception.doc
If an exception does occur, the normal flow of execution is halted and a catch block is executed. Catch blocks attempt to recover from error states. For our platform, the clean-up process normally involves deleting any temporary information we had collected or returning a negative value to the client. In this case, the method has failed and the client is prompted to try again.

Each try-catch statement also contains a finally block. These blocks are guaranteed to execute whether an error has occurred or not. As a connection to the database is opened in all of the web service methods, it’s important to close these connections again once the method has finished executing. Explicitly closed database connections are returned to the connection pool for optimal web service performance.

5.8 Platform games
Designing games is a difficult task. Designing educational games is even harder due to all of the imposed constraints and limitations. It’s often a delicate balancing act between fun gaming concepts and learning objectives. If too much emphasis is placed on the educational content then children are unlikely to play the game. On the other hand, if the game is fun but lacks sufficient educational content then the educational game has failed its primary task – to teach the children.

Game design is a highly iterative process. It’s unlikely that a successful game will be designed and implemented from start to finish without making a few changes along the way. Many professional educational game developers that attended the Games for Change Festival 24 recommend launching a game at 25% completion. Build the basic game mechanics, maintain a working model and get feedback. It’s important to be open, flexible and willing to change while at the same time maintaining an overall vision for the final game.

Ideally educational games should be built around a fun game concept. Only once that is achieved should educational content be added to the game. Although researching localisation awareness topics and designing educational games were not the primary objectives of this project, I decided to design and implement a few plugin games to effectively demonstrate the features of the educational platform.

24 Games for Change Festival homepage – http://www.gamesforchange.org
5.8.1 Europe explorer

This plugin game (see Figure B9 and B10) aims to help primary school children correctly identify European countries in a fun and educational manner. This game is designed to complement the current primary school geography curriculum and provides sufficient motivation to encourage the students to keep playing the game. The student is given the name of a country, its national flag and asked to identify each country on the map.

This plugin offers five achievements - testing both speed and accuracy:

- 100 points - Beat the game in less than 2 minutes.
- 150 points - Around the world in 80 seconds.
- 200 points - Correctly identify 10 countries in a row.
- 250 points - Correctly identify 20 countries in a row.
- 300 points - Correctly identify 30 countries in a row.

Once the main module has loaded, an array of 40 European countries is populated. The plugin uses the Random class to pick a random country from the list of country. Once the user has correctly identified the country, the PickRandomCountry method is invoked again to pick another random country from the array and update the country details panel. This random element prevents the students from memorising the order in which countries are displayed.

The map of Europe used in this game was derived from a SVG (Scalable Vector Graphics) image taken from the European Wikipedia article. Once I had the SVG image, I was able to load it into Microsoft Expression Design and convert the vector image into a set of paths. These paths function as the boundary for each country. Once the MouseLeftButtonDown event listener was attached to every path I could determine what country the user had selected. As each path uses the same generic event handler, once a country is selected, the event handler casts the sending object to a path, examines the String value stored in the paths optional tag field and compares it to the name of the country the player is currently looking for. If the two values match then the user has correctly identified the country. This plugin also keeps track of users score as they progress through the game. The DispatcherTimer

---

is set to elapse every second. Every time it does, it checks to see if the player has run out of time and records how long they have spent on the current question. These records, along with various other statistics are analysed at the end of the game to calculate the final score and to determine what achievements to award to player.

5.9 Server configuration

This game-based virtual learning environment was hosted on a desktop machine provided by the Localisation Research Centre (LRC) for the duration of my research. The server was given a static, publically accessible IP address and the domain name www.endaquigley.com/platform was set to redirect to this address. Windows Server 2008 R2 was the operating system used on the machine. It’s a highly modularised version of the Windows operating system that can be optimised for a variety of purposes by installing additional system features and roles.

The primary purpose of this server is to:

- Run the back-end MySQL database;
- Provide a WCF web service to facilitate database interaction;
- Host the platform framework and various plugin games.

The MySQL Community Server\textsuperscript{26} version 5.1 was the DBMS (Database Management System) used to host the platform database. This software is free to download and completely open-source. Once the DBMS was installed, the default username and password for the system was modified for security purposes. The final step was to create a new database and import the full schema structure.

The WCF web service is required for the client Silverlight application to interact with the MySQL database. The application server and IIS (Internet Information Services) web server roles have to be enabled in the operating systems server manager to run WCF web services. The web service solution needs to be placed in the default IIS root directory, converted to an IIS application and setup to use the integrated ASP.NET v4.0 application pool. If the main WCF web server class is using the same MySQL DBMS credentials then all web service operations should succeed once invoked.

\textsuperscript{26}MySQL Community Server - http://dev.mysql.com/downloads
The main platform framework is also located in the default IIS root directory. The Silverlight solution is pre-compiled into a .XAP file and placed into the ClientBin directory. The various avatars, national flags, and plugin game assemblies are also located in this directory. By isolating these files from the core platform framework we can significantly reduce the size of the main Silverlight application by downloading these resources on demand.
6. Testing and Evaluation

6.1 Introduction

“Software Testing is the process of executing a program or system with the intent of finding error.” (Myers, 1979)

This chapter outlines the steps taken in order to ensure that the implementation of the core platform framework works as expected. Unit testing, user acceptance testing and student evaluation were the main forms of testing adopted for this project.

6.2 Unit testing

Unit testing is the process of isolating the individual components of an application to verify the correctness of each. Unit tests are written by the software developer preferably before execution and provide a strict contract which the code must adhere to. Like all forms of testing, unit tests can only show the presence of software errors but not the absence of errors.

I used the NUnit testing framework27 for running automated unit tests. NUnit is the Microsoft .NET equivalent to the popular JUnit Java testing framework28. After installation, I created a new class library and included this as part of my project solution. As all of the test classes are stored in this assembly it is easy to remove the test cases from the final platform release. The Platform and NUnit_Framework assemblies had to be included as project references before testing could take place.

All methods with the Public access modifier were tested. Trivial getter and setter methods can be ignored as the tasks they perform are simple in nature. Private methods are also excluded as the developer only intended one class to interact with these particular methods.

Various assert statements are created within each test case to verify that the correct results are obtained. The unit testing framework shows a visual representation of the

---

27 NUnit official website - http://www.nunit.org
28 JUnit official website - http://www.junit.org
test results. A green icon is assigned to those tests that passed and a red icon indicates that an error has occurred.

Automated unit test cases also help facilitate refactoring and regression testing at a later date. As these test cases are executed frequently they provide rapid feedback for the developer and prevent class functionality from being accidentally harmed.

6.3 User acceptance testing
User acceptance testing is performed to get instant feedback from the end-users. As a select group of users are testing the functionality of the product, user acceptance testing is best performed once development has been complete. In a commercial environment, user acceptance testing is often one of the last stages of a project and often occurs before the client accepts the current implementation of the system. The main purpose of user acceptance testing is to verify that the products functional and non-functional requirements have been satisfied. The software product should also be fully tested by other means such as unit testing and black box testing before starting user acceptance testing. However, as any amount of testing can never be considered one hundred percent comprehensive, user acceptance testing may also highlight hidden bugs during the process.

Two CTYI (Centre for Talented Youth, Ireland\textsuperscript{29}) lecturers, Naoto Nishio and Dimitra Anastasiou agreed to help me test out the platform by allowing me to participate in the “Japanese Language, Webpage Localisation and Machine Translation” course that they were running in the University of Limerick. This summer course is aimed at primary school students between eight to twelve years of age and provided the perfect opportunity to evaluate the project. Two Japanese themed games were designed specifically for the course and used on the final day. The students were given a short list of objectives but were ultimately free to do whatever they wanted within the platform.

The objectives were as follows:

- Login using your username and password.
- Go to your profile page and personalise it as much as you want.
- Go to your friends list and search for a new friend.

\textsuperscript{29}Centre for Talented Youth, Ireland - http://www.dcu.ie/ctyi
- Play the “Japanese typing practice” game on the home screen.
- Play the “Pokémon translation” game and see who gets the highest score.
- Play the “Europe explorer” game and see who gets the highest score.
- Check out the achievements screen to see what ones you have yet to earn.

I observed the students as they were using the platform for the first time. A number of the students were impatient and quick to ask questions, but given enough time or help from classmates they eventually solved their own problems. I noticed that most of the students enjoyed the competitive nature of the games. Teachers may discourage competition in the classroom but from observing the students behaviour, it’s clearly an effective motivating factor. None of the students seemed to have any difficulty using the platform. They were aware of the many of the concepts applied throughout the platform such as the breadcrumb navigation bar, profile pages, friend invites and in-game achievements.

A questionnaire was handed out at the end of class. It comprised of nine short questions (see Appendix B) for the purpose of gathering anonymous information from the students. Unfortunately many of the students were too occupied with the games to adequately answer the questions and a few of the students complained that the questions were too open-ended. Overall it would appear that they enjoyed the platform experience.

When asked if they would rather learn through the use of interactive games, one student wrote “Yes, bored [sic] to read and listen, like to do stuff”. Many of the students agreed that it was a more enjoyable experience.

The next question asked them if they were motivated by the high score lists. Out of every student participating in the questionnaire, they all agreed that the high score list motivated them to play the games again. As one student put it, he wants just wants to “get more than his friends”.

The next form of motivation examined was the in-game achievements system. When asked whether they were a motivating factor, it was a split decision. Figure 9 presents the data obtained from the students. Those that were motivated by the in-game achievements said that they “wanted to complete the set goals” and collect “more and more” achievements. Some students just didn’t care about this feature.
stating that they “just want to have fun” and the main reason why students didn’t like the achievements was because of the difficulty level associated with them.

Figure 9: Analysis of the in-game achievement system

When asked what they liked about the system, a lot of students said that “it was fun”. Others said that they liked the games and the ability to “have friends” and “talk to people”. On the other hand, when asked what aspects of the system they did not like, many of the students claimed that they didn’t like some of the educational games. With more effort put into the content and overall quality of these games I’m sure a lot of the students would love playing them.

Question eight asked the students if, given the chance, would they play educational games like these in their free time? Over sixty percent of the students said that they would play these sort of games in their free time, one guy adding, only if he “got bored enough”. Figure 10 presents the data obtained from the students. A number of students complained that the Japanese games focused too much on education and were not as fun as they could have been. With better designed games I am certain that the number of students willing to play these games in their free time would increase.
The final question asked the students to list a number of features that they would like to see implemented in the next version of our platform. While many of the students left this section blank, online videos, multiplayer and more games were the main features requested.

For the full list of questions, see Appendix B.

6.4 Minimum specifications

The main goal of this project was to build a powerful, lightweight educational platform that runs well on older computers, as many schools in Ireland tend to have slow computers in the classroom. (Donnelly, 2009)

The minimum system requirements for running the platform are as follows:

- 1 GHz 32-bit processor
- 256 MB of memory
- 32 MB of graphics memory
- 100 MB free disk space (for optimal caching)
- 800 x 600 screen resolution
- 56k modem or higher
- Windows XP, Vista, 7, Mac OS X 10.5 or later.
- Internet Explorer, Firefox, Chrome, Safari and Opera.

Although our platform has been designed to operate under these conditions, no attempt has been made to calculate the minimum specification for the plugins. As these can vary in both complexity and size, it would be impossible to outline a realistic minimum specification to comprehensively cover every platform plugin. One can only imagine how resource intensive a 3D plugin may be or how much disk space a large application may occupy if the developer decides to embed a lot of visual imagery and audio assets.
7. Conclusions

7.1 Introduction
This chapter deals with the project conclusion, lessons learnt and outlines a few suggestions for further work on this software product.

7.2 Summary
In section one I provided justification for undertaking this research and outlined the primary objective for this project which was to design and develop a fully extensible virtual learning environment, based on game-based learning strategies. Section two covers the research undertaken over the course of the year. This research includes the analysis of existing educational products currently on the market, meetings with educational professionals and an attempt to integrate our software product with the current primary school curriculum. A lot of effort also went into designing an appropriate user-interface for primary school students and the investigation of motivation and encouragement techniques within the platform through the use of in-game achievements and global high score lists.

Section four covers the architectural design of the platform, from outlining the functional and non-functional aspects of the system, explaining the various conscious design decisions such as whether the platform should benefit from the full desktop experience or be a portable web-based application. Section four concludes with an in depth review of the security aspects of the system along with a visual representation of the final database schema. The next section outlines the tools and technologies used during implementation along with a brief description of the newer one such as Microsoft Silverlight and Windows Communication Foundation. Section five also describes the implementation of the major platform components and how they all come together to form a cohesive product.

Section six covers the testing and evaluation of the final software product. N-Unit testing was used to verify the integrity of each individual software component and user acceptance testing was undertaken with a class of eight to twelve year old primary school students to evaluate the complete platform experience. Section six
concludes by outlining the minimum specifications needed to run the educational platform.

7.3 Future research and development

This thesis implements a fully functional game-based virtual learning environment, ready for use in primary schools around the country. However, I believe that a lot more work can be done in this area to improve the existing product and overall concept. Many of these ideas were formed late in the software’s lifecycle. Given the strict time constraint, I decided to document these features but exclude them from the final specification. As these features may be implemented in a future version of our platform, a lot of effort was put into designing the platform with expansion in mind. I have outlined the main features below.

7.3.1 Greater admin control

One of the concerns that educators had with the platform was extent of the freedom that students had within the platform. Although students cannot create their own account, they have access to the platform from any location and are free to interact with other students, modify content and upload their own material. Although reporting functionality is currently built into the platform, it is essentially a reactive method of dealing with irresponsible student behaviour and may take a few days to rectify the situation. Greater admin control may involve a more proactive approach where teachers have to manually approve content before it is uploaded onto the platform server.

7.3.2 Platform localisation

If our platform intends to reach a large audience, the software needs to be adapted to cater for a wide variety of cultures and languages. Many of the resources such as graphics, text and help files would have to be extracted from the source and re-integrated in a translatable format such as XLIFF (XML Localisation Interchange File Format).

7.3.3 Educational games

Designing educational games should be a co-operative effort between game developers and educational professionals. Quest 2 Learn\textsuperscript{30}, a game-design inspired

\textsuperscript{30} Quest 2 Learn - http://www.q2l.org
school in New York City for example has a dedicated curriculum development and design team to “work with students and teachers, providing opportunities to work with game designers and other media specialists on professional development and curriculum projects”. One of their main goals is to “design learning tools and toolkits for use in the school and within the digital media and learning network”\(^{31}\). They hire three full-time game developers, pedagogical teaching specialists, an assessment specialist and a project manager for the development of in-house content. This multi-talented team allows each member to focus on their own strengths and benefit by working together.

**7.3.4 Content filtering**

Develop a content filtering system that can detect pornography and report abusive behaviour before the student has the opportunity to upload the offending material. For example, if a student tries to bypass the curse filter or bully another student online, their current action will be cancelled and the teacher will be notified immediately. This reporting strategy would not depend on direct user intervention; rather it would prevent the incident in the first place. A third party content filtering solution such as “eSafe protect” \(^{32}\) could be implemented which, along with its extensive anti-pornography features offers protection against online bullying, gambling and self-harm.

**7.3.5 Parental reporting and engagement**

As the teacher, student and guardian roles have already been defined in the database schema, it provides the ideal opportunity for parents to engage in their child’s education. Daily attendance, calendars, announcements, reports and personalised letters could be uploaded and maintained by teachers and viewed by parents through a separate online interface.

With the contact details of all parents and guardians stored in the database, this could also enable schools to easily get in contact with parents to notify them of commonly occurring events such as sickness, suspension or poor attendance. These messages could be stored as a template and translated into a variety of different languages. This system would be extra beneficial for multinational schools where, in some

\(^{31}\) Q2L Mission Lab - http://www.q2l.org/missionlab  
\(^{32}\) eSafe protect homepage - http://www.esafeeducation.co.uk
cases, parents may have little or no English and may rely on their children for direct communication with the school.

7.3.6 Secure socket layer certificate
Secure socket layer (SSL) is a cryptographic protocol that provides end-to-end encryption for communication over the internet. Although the password for each user is currently protected through the use of salted hashes (see 4.4.2), all other communication on the platform is sent over the internet in the clear. Student details, private messages and other sensitive information could be obtained by a malicious user on the network using various eavesdropping techniques.

An SSL certificate, purchased from a reputable certificate authority and applied to the server would ensure that all communication between the client and the server is securely encrypted using 128-bit public-key encryption, the legal standard enforced by BECTA\(^{33}\), the UK Education body.

Of 76 contributors from IT Management and Corporate Governance who were asked about the security of date in transport by (McKenna, 2009), all 76 responded that “in no case should data belonging to anyone, children or otherwise, be transported in clear unencrypted form.” All 76 of the contributors agreed that this should be legislated for and tightly controlled.

7.4 Conclusion
I initially set out to design and implement a basic educational platform specifically designed for the purpose of teaching localisation awareness topics in Irish primary schools. From analysis of the final product, I believe that I have achieved all of my set goals while at the same time managing to significantly increase the scope of the project. Although the platform currently only has a limited amount of educational, a plugin architecture was adopted and a simple API (Application Programming Interface) was designed to facilitate seamless expansion in the future.

I visited a number of local primary schools in an attempt to get teachers interested in the project. Unfortunately due do their busy schedules many of them were unable to help with the design and development of the platform, more specifically, the educational platform games. As it took the best part of a year to develop the

\(^{33}\) BECTA - http://www.becta.org.uk
software, I did not get the opportunity to try out the platform in any Irish primary school. However, a number of teachers I had contacted over the last few months seemed interested in the concept and are willing to try out the platform at the start of the new semester. Once the software was feature-complete I was given the opportunity to demonstrate and evaluate the software with a class of primary school students during the summer of 2010. Thankfully they all seemed to enjoy the experience and confirmed my belief that social, game-based learning is the way of the future.
8. References


9. Appendices

This report contains three appendix items

- Appendix A: Platform Screenshots
- Appendix B: Questionnaire
- Appendix C: Database Schema
Appendix A:
Platform Screenshots

- Figure A1: Login screen
- Figure A2: Student homepage
- Figure A3: Student profile
- Figure A4: Avatar selection
- Figure A5: Friends list
- Figure A6: Student search
- Figure A7: Achievements overview
- Figure A8: Game achievements
- Figure A9: Europe explorer splash screen
- Figure A10: Europe explorer game
- Figure A11: High score list
- Figure A12: Japanese typing practice
- Figure A13: Message centre
- Figure A14: New message
- Figure A15: Country scrapbook (FYP implementation)
- Figure A16: Common Japanese expressions (FYP implementation)
Figure A1: Login screen

Figure A2: Student homepage
Figure A3: Student profile

Figure A4: Avatar selection
Figure A5: Friends list

Figure A6: Student search
Figure A7: Achievements overview

Figure A8: Game achievements
Figure A9: Europe explorer splash screen

Figure A10: Europe explorer game
Figure A11: High score list

<table>
<thead>
<tr>
<th>Rank</th>
<th>Name</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>bein hel'v (bein.hel'v)</td>
<td>87700</td>
</tr>
<tr>
<td>2</td>
<td>Daniel Arbab (man ltd)</td>
<td>83300</td>
</tr>
<tr>
<td>3</td>
<td>Brandon Cullinan (brandon.cullinan)</td>
<td>84551</td>
</tr>
<tr>
<td>4</td>
<td>Cadwell Rynne (cadwell.rynne)</td>
<td>83900</td>
</tr>
<tr>
<td>5</td>
<td>Ena Quigley (Quigley Guy)</td>
<td>83700</td>
</tr>
<tr>
<td>6</td>
<td>Reese Gavin (nam-old)</td>
<td>79300</td>
</tr>
</tbody>
</table>

Figure A12: Japanese typing practice
Figure A13: Message centre

Figure A14: New message
Figure A15: Country scrapbook (FYP implementation)

Figure A16: Common Japanese expressions (FYP implementation)
Appendix B:

Questionnaire

1) Do you rather learning through the use of interactive games? Explain why.

________________________________________________________________________
________________________________________________________________________
________________________________________________________________________
________________________________________________________________________

2) Does the high score list motivate you to play the games again?

________________________________________________________________________
________________________________________________________________________
________________________________________________________________________
________________________________________________________________________

3) Do the in-game achievements motivate you to complete the set goals?

________________________________________________________________________
________________________________________________________________________
________________________________________________________________________
________________________________________________________________________

4) Did you have any difficulty navigating through / using the learning platform?

________________________________________________________________________
________________________________________________________________________
________________________________________________________________________
________________________________________________________________________

5) What aspects did you like about the learning platform?

________________________________________________________________________
________________________________________________________________________
________________________________________________________________________
6) What aspects did you not like about the learning platform?

________________________________________________________________________

________________________________________________________________________

________________________________________________________________________

7) What type of games do you like to play at home? (How long per day?)

________________________________________________________________________

________________________________________________________________________

________________________________________________________________________

8) Given the chance, would you play educational games like these in your free time?

________________________________________________________________________

________________________________________________________________________

________________________________________________________________________

9) What extra features would you like to see in this game-based learning platform?

________________________________________________________________________

________________________________________________________________________

________________________________________________________________________
### Appendix C:

Database Schema

| CREATE TABLE IF NOT EXISTS user_types ( |
| user_type_id INT UNSIGNED NOT NULL AUTO_INCREMENT , |
| user_type VARCHAR(30) NOT NULL COLLATE utf8_unicode_ci , |
| PRIMARY KEY (user_type_id) , |
| UNIQUE KEY (user_type) |
| ) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci; |

| INSERT INTO user_types VALUES |
| (1, 'Administrator'), |
| (2, 'School'), |
| (3, 'Teacher'), |
| (4, 'Student'), |
| (5, 'Guardian'); |

| CREATE TABLE IF NOT EXISTS genders ( |
| gender_id INT UNSIGNED NOT NULL AUTO_INCREMENT , |
| gender VARCHAR(10) NOT NULL COLLATE utf8_unicode_ci , |
| PRIMARY KEY (gender_id) , |
| UNIQUE KEY (gender) |
| ) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci; |

| INSERT INTO genders VALUES |
| (1, 'Male'), |
| (2, 'Female'); |

| CREATE TABLE IF NOT EXISTS nationalities ( |
| nationality_id INT UNSIGNED NOT NULL AUTO_INCREMENT , |
| nationality VARCHAR(100) NOT NULL COLLATE utf8_unicode_ci , |
| flag_filename varchar(50) NOT NULL COLLATE utf8_unicode_ci , |
| PRIMARY KEY (nationality_id) , |
| UNIQUE KEY (nationality) |
| ) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci; |

| INSERT INTO nationalities VALUES |
| (1, 'Irish', 'irish.png'); |

| CREATE TABLE IF NOT EXISTS avatars ( |
| avatar_id INT UNSIGNED NOT NULL AUTO_INCREMENT , |
| avatar_filename VARCHAR(50) NOT NULL COLLATE utf8_unicode_ci , |
| PRIMARY KEY (avatar_id) , |
| UNIQUE KEY (avatar_filename) |
| ) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci; |

| INSERT INTO `avatars`(`avatar_id`, `avatar_filename`) VALUES |
| (1, 'default.jpg'); |

| CREATE TABLE IF NOT EXISTS privacy ( |
privacy_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
privacy VARCHAR(20) NOT NULL COLLATE utf8_unicode_ci,
PRIMARY KEY (privacy_id),
UNIQUE KEY (privacy)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

INSERT INTO privacy VALUES
(1, 'public'),
(2, 'private');

CREATE TABLE IF NOT EXISTS users
(user_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
user_type_id INT UNSIGNED NOT NULL,
username VARCHAR(20) UNIQUE NOT NULL COLLATE utf8_unicode_ci,
unique_salt VARCHAR(15) NOT NULL COLLATE utf8_unicode_ci,
password_hash CHAR(40) NOT NULL COLLATE utf8_unicode_ci,
forename VARCHAR(30) NOT NULL COLLATE utf8_unicode_ci,
surname VARCHAR(30) NOT NULL COLLATE utf8_unicode_ci,
gender_id INT UNSIGNED NOT NULL,
privacy_id INT UNSIGNED NOT NULL,
avatar_id INT UNSIGNED NOT NULL,
date_of_birth DATE NOT NULL,
nationality_id INT UNSIGNED NOT NULL,
last_login DATETIME NOT NULL,
PRIMARY KEY (user_id),
UNIQUE KEY (unique_salt),
UNIQUE KEY (user_id , user_type_id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

ALTER TABLE users ADD FOREIGN KEY (user_type_id) REFERENCES user_types (user_type_id) ;
ALTER TABLE users ADD FOREIGN KEY (avatar_id) REFERENCES avatars (avatar_id) ;
ALTER TABLE users ADD FOREIGN KEY (privacy_id) REFERENCES privacy (privacy_id) ;
ALTER TABLE users ADD FOREIGN KEY (gender_id) REFERENCES genders (gender_id) ;
ALTER TABLE users ADD FOREIGN KEY (nationality_id) REFERENCES nationalities (nationality_id) ;

INSERT INTO users VALUES
(1, 1, 'endaquigley', 'MyUniqueSalt', '0628d61cc6a3c472dce94088ae47e35d21f8fee6', 'Enda', 'Quigley', 1, 1, 1, '2010-04-17', 1, '2010-01-01 01:00:00');

CREATE TABLE IF NOT EXISTS interests
(interest_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
interest VARCHAR(30) NOT NULL COLLATE utf8_unicode_ci,
PRIMARY KEY (interest_id),
UNIQUE KEY (interest)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

INSERT INTO interests VALUES
(1, 'Games'),
(2, 'Programming'),
(3, 'Hurling');
CREATE TABLE IF NOT EXISTS user_interests ( 
    user_id INT UNSIGNED NOT NULL , 
    interest_id INT UNSIGNED NOT NULL , 
    PRIMARY KEY (user_id, interest_id) 
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci; 

ALTER TABLE user_interests ADD FOREIGN KEY (user_id) REFERENCES users (user_id) ON DELETE CASCADE ; 
ALTER TABLE user_interests ADD FOREIGN KEY (interest_id) REFERENCES interests (interest_id) ON DELETE CASCADE ; 

INSERT INTO user_interests VALUES 
(1, 1), 
(1, 2), 
(1, 5); 

CREATE TABLE IF NOT EXISTS languages ( 
    language_id INT UNSIGNED NOT NULL AUTO_INCREMENT , 
    language VARCHAR(50) NOT NULL COLLATE utf8_unicode_ci , 
    PRIMARY KEY (language_id) , 
    UNIQUE KEY (language) 
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci; 

INSERT INTO languages VALUES 
(1, 'Irish'), 
(2, 'English'), 
(3, 'French'), 
(4, 'German'), 
(5, 'Japanese'); 

CREATE TABLE IF NOT EXISTS user_languages ( 
    user_id INT UNSIGNED NOT NULL , 
    language_id INT UNSIGNED NOT NULL , 
    PRIMARY KEY (user_id, language_id) 
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci; 

ALTER TABLE user_languages ADD FOREIGN KEY (user_id) REFERENCES users (user_id) ON DELETE CASCADE ; 
ALTER TABLE user_languages ADD FOREIGN KEY (language_id) REFERENCES languages (language_id) ON DELETE CASCADE ; 

INSERT INTO user_languages VALUES 
(1, 1), 
(1, 2), 
(1, 3); 

CREATE TABLE IF NOT EXISTS relationships_types ( 
    relationship_type_id INT UNSIGNED NOT NULL AUTO_INCREMENT , 
    relationship_type VARCHAR(20) NOT NULL COLLATE utf8_unicode_ci , 
    PRIMARY KEY (relationship_type_id) , 
    UNIQUE KEY (relationship_type) 
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
INSERT INTO relationships_types VALUES
(1, 'request'),
(2, 'friend'),
(3, 'banned');

CREATE TABLE IF NOT EXISTS user_relationships
(user_id_one INT UNSIGNED NOT NULL,
user_id_two INT UNSIGNED NOT NULL,
relationship_type_id INT UNSIGNED NOT NULL,
PRIMARY KEY (user_id_one, user_id_two));

ALTER TABLE user_relationships ADD FOREIGN KEY (user_id_one) REFERENCES users (user_id) ON DELETE CASCADE;
ALTER TABLE user_relationships ADD FOREIGN KEY (user_id_two) REFERENCES users (user_id) ON DELETE CASCADE;
ALTER TABLE user_relationships ADD FOREIGN KEY (relationship_type_id) REFERENCES relationships_types (relationship_type_id);

CREATE TABLE IF NOT EXISTS administrators
(user_id INT UNSIGNED NOT NULL,
user_type_id INT UNSIGNED NOT NULL DEFAULT 1,
PRIMARY KEY (`user_id`));

ALTER TABLE administrators ADD FOREIGN KEY (user_id) REFERENCES users (user_id) ON DELETE CASCADE;
ALTER TABLE administrators ADD FOREIGN KEY (user_type_id) REFERENCES user_types (user_type_id);
ALTER TABLE administrators ADD FOREIGN KEY (user_id, user_type_id) REFERENCES users (user_id, user_type_id);

INSERT INTO administrators VALUES
(1, 1);

CREATE TABLE IF NOT EXISTS schools
(school_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
school_name VARCHAR(150) NOT NULL COLLATE utf8_unicode_ci,
address1 VARCHAR(100) NOT NULL COLLATE utf8_unicode_ci,
address2 VARCHAR(100) NOT NULL COLLATE utf8_unicode_ci,
address3 VARCHAR(100) NOT NULL COLLATE utf8_unicode_ci,
county VARCHAR(50) NOT NULL COLLATE utf8_unicode_ci,
phone_number VARCHAR(30) NOT NULL COLLATE utf8_unicode_ci,
fax_number VARCHAR(30) NOT NULL COLLATE utf8_unicode_ci,
email_address VARCHAR(150) NOT NULL COLLATE utf8_unicode_ci,
PRIMARY KEY (`school_id`));

ALTER TABLE schools ADD FOREIGN KEY (user_id) REFERENCES users (user_id) ON DELETE CASCADE;
ALTER TABLE schools ADD FOREIGN KEY (user_type_id) REFERENCES user_types (user_type_id);
ALTER TABLE schools ADD FOREIGN KEY (user_id, user_type_id) REFERENCES users (user_id, user_type_id);

CREATE TABLE IF NOT EXISTS teachers (  
  user_id INT UNSIGNED NOT NULL,
  user_type_id INT UNSIGNED NOT NULL DEFAULT 3,
  school_id INT UNSIGNED NOT NULL,
  PRIMARY KEY (`user_id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

ALTER TABLE teachers ADD FOREIGN KEY (user_id) REFERENCES users (user_id) ON DELETE CASCADE;
ALTER TABLE teachers ADD FOREIGN KEY (user_type_id) REFERENCES user_types (user_type_id);
ALTER TABLE teachers ADD FOREIGN KEY (user_id, user_type_id) REFERENCES users (user_id, user_type_id);
ALTER TABLE teachers ADD FOREIGN KEY (school_id) REFERENCES schools (school_id) ON DELETE CASCADE;

CREATE TABLE IF NOT EXISTS students (  
  user_id INT UNSIGNED NOT NULL,
  user_type_id INT UNSIGNED NOT NULL DEFAULT 4,
  school_id INT UNSIGNED NOT NULL,
  PRIMARY KEY (`user_id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

ALTER TABLE students ADD FOREIGN KEY (user_id) REFERENCES users (user_id) ON DELETE CASCADE;
ALTER TABLE students ADD FOREIGN KEY (user_type_id) REFERENCES user_types (user_type_id);
ALTER TABLE students ADD FOREIGN KEY (user_id, user_type_id) REFERENCES users (user_id, user_type_id);
ALTER TABLE students ADD FOREIGN KEY (school_id) REFERENCES schools (school_id) ON DELETE CASCADE;

CREATE TABLE IF NOT EXISTS guardians (  
  user_id INT UNSIGNED NOT NULL,
  user_type_id INT UNSIGNED NOT NULL DEFAULT 5,
  address1 VARCHAR(100) NOT NULL COLLATE utf8_unicode_ci,
  address2 VARCHAR(100) NOT NULL COLLATE utf8_unicode_ci,
  address3 VARCHAR(100) NOT NULL COLLATE utf8_unicode_ci,
  county VARCHAR(50) NOT NULL COLLATE utf8_unicode_ci,
  home_number VARCHAR(30) NOT NULL COLLATE utf8_unicode_ci,
  mobile_number VARCHAR(30) NOT NULL COLLATE utf8_unicode_ci,
  email_address VARCHAR(150) NOT NULL COLLATE utf8_unicode_ci,
  PRIMARY KEY (`user_id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

ALTER TABLE guardians ADD FOREIGN KEY (user_id) REFERENCES users (user_id) ON DELETE CASCADE;
ALTER TABLE guardians ADD FOREIGN KEY (user_type_id) REFERENCES user_types (user_type_id);
ALTER TABLE guardians ADD FOREIGN KEY (user_id, user_type_id) REFERENCES users (user_id, user_type_id);

CREATE TABLE IF NOT EXISTS student_guardians (student_user_id INT UNSIGNED NOT NULL, guardian_user_id INT UNSIGNED NOT NULL, PRIMARY KEY (student_user_id)) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

ALTER TABLE student_guardians ADD FOREIGN KEY (student_user_id) REFERENCES students (user_id) ON DELETE CASCADE;
ALTER TABLE student_guardians ADD FOREIGN KEY (guardian_user_id) REFERENCES guardians (user_id) ON DELETE CASCADE;

CREATE TABLE IF NOT EXISTS plugin_developers (developer_id INT UNSIGNED NOT NULL AUTO_INCREMENT, developer_name VARCHAR(50) NOT NULL COLLATE utf8_unicode_ci, developer_website VARCHAR(150) NOT NULL COLLATE utf8_unicode_ci, PRIMARY KEY (developer_id)) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

INSERT INTO plugin_developers VALUES (1, 'Enda Quigley', 'www.endaquigley.com');

CREATE TABLE IF NOT EXISTS plugins (plugin_id INT UNSIGNED NOT NULL AUTO_INCREMENT, developer_id INT UNSIGNED NOT NULL, plugin_name VARCHAR(50) NOT NULL COLLATE utf8_unicode_ci, plugin_description TEXT NOT NULL COLLATE utf8_unicode_ci, assembly_name VARCHAR(50) NOT NULL COLLATE utf8_unicode_ci, image_filename VARCHAR(50) NOT NULL COLLATE utf8_unicode_ci, PRIMARY KEY (plugin_id), UNIQUE KEY (plugin_name), UNIQUE KEY (assembly_name)) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

ALTER TABLE plugins ADD FOREIGN KEY (developer_id) REFERENCES plugin_developers (developer_id);

INSERT INTO plugins VALUES (1, 1, 'Europe Explorer', 'Plugin Description', 'EuropeExplorer', 'EuropeExplorer.png'), (2, 1, 'Pokemon Translation', 'Plugin Description', 'PokemonTranslation', 'PokemonTranslation.png'), (3, 1, 'Japanese Typing Practice', 'Plugin Description', 'TypingPractice', 'PracticeTyping.png');

CREATE TABLE IF NOT EXISTS plugin_achievements (achievement_id INT UNSIGNED NOT NULL AUTO_INCREMENT, plugin_id INT UNSIGNED NOT NULL, achievement_name VARCHAR(50) NOT NULL COLLATE utf8_unicode_ci, achievement_description VARCHAR(250) NOT NULL COLLATE utf8_unicode_ci, achievement_value INT UNSIGNED NOT NULL DEFAULT 0, image_filename VARCHAR(50) NOT NULL COLLATE utf8_unicode_ci, PRIMARY KEY (`achievement_id`)) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

ALTER TABLE plugin_achievements ADD FOREIGN KEY (achievement_id) REFERENCES plugin_developers (achievement_id);

CREATE TABLE IF NOT EXISTS plugin_achievements (achievement_id INT UNSIGNED NOT NULL AUTO_INCREMENT, plugin_id INT UNSIGNED NOT NULL, achievement_name VARCHAR(50) NOT NULL COLLATE utf8_unicode_ci, achievement_description VARCHAR(250) NOT NULL COLLATE utf8_unicode_ci, achievement_value INT UNSIGNED NOT NULL DEFAULT 0, image_filename VARCHAR(50) NOT NULL COLLATE utf8_unicode_ci, PRIMARY KEY (`achievement_id`)) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
ALTER TABLE plugin_achievements ADD FOREIGN KEY (plugin_id) REFERENCES plugins (plugin_id) ON DELETE CASCADE;

INSERT INTO plugin_achievements VALUES
(1, 1, 'Achievement 1', 'Beat the game in less than 2 minutes', 100, '1.jpg'),
(2, 1, 'Achievement 2', 'Beat the game in less than 80 seconds', 150, '2.jpg'),
(3, 1, 'Achievement 3', 'Correctly identify 10 countries in a row', 200, '3.jpg'),
(4, 1, 'Achievement 4', 'Correctly identify 20 countries in a row', 250, '4.jpg'),
(5, 1, 'Achievement 5', 'Correctly identify 30 countries in a row', 300, '5.jpg'),
(6, 2, 'Achievement 1', 'Beat the game on easy mode or higher', 50, '1.jpg'),
(7, 2, 'Achievement 2', 'Beat the game on regular mode or higher', 150, '1.jpg'),
(8, 2, 'Pokemon Master', 'Beat the game on hard mode', 250, '1.jpg'),
(9, 2, 'Achievement 4', 'Spell the name of a pokemon in Japanese without making a mistake', 100, '1.jpg'),
(10, 2, 'Fast fingers', 'Answer a question in less than 15 seconds', 150, '1.jpg'),
(11, 2, 'Total recall', 'Score full marks in the bonus round', 200, '1.jpg'),
(12, 2, 'It’s over 9000!!!', 'Score over 9000 points in the game', 100, '1.jpg');

CREATE TABLE IF NOT EXISTS class_levels (class_level_id INT UNSIGNED NOT NULL AUTO_INCREMENT, class_level VARCHAR(20) NOT NULL COLLATE utf8_unicode_ci, PRIMARY KEY (class_level_id), UNIQUE KEY (class_level)) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

INSERT INTO class_levels VALUES
(1, '1st'),
(2, '2nd'),
(3, '3rd'),
(4, '4th'),
(5, '5th'),
(6, '6th');

CREATE TABLE IF NOT EXISTS classes (class_id INT UNSIGNED NOT NULL AUTO_INCREMENT, class_name VARCHAR(50) NOT NULL COLLATE utf8_unicode_ci, class_level_id INT UNSIGNED NOT NULL, PRIMARY KEY (class_id)) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

ALTER TABLE classes ADD FOREIGN KEY (class_level_id) REFERENCES class_levels (class_level_id);

CREATE TABLE IF NOT EXISTS class_teachers (class_id INT UNSIGNED NOT NULL, teacher_user_id INT UNSIGNED NOT NULL, PRIMARY KEY (class_id, teacher_user_id)) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

ALTER TABLE class_teachers ADD FOREIGN KEY (class_id) REFERENCES classes (class_id) ON DELETE CASCADE;
ALTER TABLE class_teachers ADD FOREIGN KEY (teacher_user_id) REFERENCES teachers (user_id) ON DELETE CASCADE;
CREATE TABLE IF NOT EXISTS student_classes (  
student_user_id INT UNSIGNED NOT NULL ,  
class_id INT UNSIGNED NOT NULL ,  
PRIMARY KEY (student_user_id, class_id)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

ALTER TABLE student_classes ADD FOREIGN KEY (student_user_id) REFERENCES students (user_id) ON DELETE CASCADE ;  
ALTER TABLE student_classes ADD FOREIGN KEY (class_id) REFERENCES classes (class_id) ON DELETE CASCADE ;

CREATE TABLE IF NOT EXISTS class_plugins (  
class_id INT UNSIGNED NOT NULL ,  
plugin_id INT UNSIGNED NOT NULL ,  
PRIMARY KEY (class_id, plugin_id)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

ALTER TABLE class_plugins ADD FOREIGN KEY (class_id) REFERENCES classes (class_id) ON DELETE CASCADE ;  
ALTER TABLE class_plugins ADD FOREIGN KEY (plugin_id) REFERENCES plugins (plugin_id) ON DELETE CASCADE ;

CREATE TABLE IF NOT EXISTS student_plugin_history (  
student_user_id INT UNSIGNED NOT NULL ,  
plugin_id INT UNSIGNED NOT NULL ,  
last_launched DATETIME NOT NULL ,  
PRIMARY KEY (student_user_id, plugin_id)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

ALTER TABLE student_plugin_history ADD FOREIGN KEY (student_user_id) REFERENCES students (user_id) ON DELETE CASCADE ;  
ALTER TABLE student_plugin_history ADD FOREIGN KEY (plugin_id) REFERENCES plugins (plugin_id) ON DELETE CASCADE ;

CREATE TABLE IF NOT EXISTS student_achievements (  
student_user_id INT UNSIGNED NOT NULL ,  
achievement_id INT UNSIGNED NOT NULL ,  
date_earned DATETIME NOT NULL ,  
PRIMARY KEY (student_user_id, achievement_id)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

ALTER TABLE student_achievements ADD FOREIGN KEY (student_user_id) REFERENCES students (user_id) ON DELETE CASCADE ;  
ALTER TABLE student_achievements ADD FOREIGN KEY (achievement_id) REFERENCES plugin_achievements (achievement_id) ON DELETE CASCADE ;

CREATE TABLE IF NOT EXISTS student_high_scores (  
student_user_id INT UNSIGNED NOT NULL ,  
plugin_id INT UNSIGNED NOT NULL ,  
score INT UNSIGNED NOT NULL DEFAULT 0 ,  
date_earned DATETIME NOT NULL ,  
PRIMARY KEY (student_user_id, plugin_id)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
ALTER TABLE student_high_scores ADD FOREIGN KEY (student_user_id) REFERENCES students (user_id) ON DELETE CASCADE;
ALTER TABLE student_high_scores ADD FOREIGN KEY (plugin_id) REFERENCES plugins (plugin_id) ON DELETE CASCADE;

CREATE TABLE IF NOT EXISTS `states` (  
    state_id INT UNSIGNED NOT NULL AUTO_INCREMENT, 
    state VARCHAR(20) NOT NULL COLLATE utf8_unicode_ci, 
    PRIMARY KEY (state_id), 
    UNIQUE KEY (state)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

INSERT INTO states VALUES  
(1, 'new'),  
(2, 'old');

CREATE TABLE IF NOT EXISTS `messages` (  
    message_id INT UNSIGNED NOT NULL AUTO_INCREMENT, 
    sender_user_id INT UNSIGNED NOT NULL, 
    recipient_user_id INT UNSIGNED NOT NULL, 
    subject VARCHAR(100) NOT NULL COLLATE utf8_unicode_ci, 
    body TEXT NOT NULL COLLATE utf8_unicode_ci, 
    date_sent DATETIME NOT NULL, 
    state_id INT UNSIGNED NOT NULL, 
    PRIMARY KEY (message_id)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

ALTER TABLE messages ADD FOREIGN KEY (sender_user_id) REFERENCES users (user_id) ON DELETE CASCADE;
ALTER TABLE messages ADD FOREIGN KEY (recipient_user_id) REFERENCES users (user_id) ON DELETE CASCADE;
ALTER TABLE messages ADD FOREIGN KEY (state_id) REFERENCES states (state_id);

CREATE TABLE IF NOT EXISTS severity (  
    severity_id INT UNSIGNED NOT NULL AUTO_INCREMENT, 
    severity VARCHAR(25) NOT NULL COLLATE utf8_unicode_ci, 
    PRIMARY KEY (severity_id), 
    UNIQUE KEY (severity)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

INSERT INTO severity VALUES  
(1, 'high'),  
(2, 'medium'),  
(3, 'low');

CREATE TABLE IF NOT EXISTS report_categories (  
    category_id INT UNSIGNED NOT NULL AUTO_INCREMENT, 
    category VARCHAR(30) NOT NULL COLLATE utf8_unicode_ci, 
    severity_id INT UNSIGNED NOT NULL, 
    PRIMARY KEY (category_id), 
    UNIQUE KEY (category)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
ALTER TABLE report_categories ADD FOREIGN KEY (severity_id) REFERENCES severity (severity_id)
;

INSERT INTO report_categories VALUES
(1, 'spam', 3),
(2, 'nudity', 1),
(3, 'impersonation', 1),
(4, 'bullying', 1),
(5, 'swearing', 2),
(6, 'other', 2);

CREATE TABLE IF NOT EXISTS `reports` (report_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
  sender_user_id INT UNSIGNED NOT NULL,
  reported_user_id INT UNSIGNED NOT NULL,
  title VARCHAR(150) NOT NULL COLLATE utf8_unicode_ci,
  user_description VARCHAR(500) NOT NULL COLLATE utf8_unicode_ci,
  report_body TEXT NOT NULL COLLATE utf8_unicode_ci,
  report_category_id INT UNSIGNED NOT NULL,
  date_submitted DATETIME NOT NULL,
  state_id INT UNSIGNED NOT NULL,
  PRIMARY KEY (report_id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

ALTER TABLE reports ADD FOREIGN KEY (state_id) REFERENCES states (state_id);
ALTER TABLE reports ADD FOREIGN KEY (sender_user_id) REFERENCES users (user_id) ON DELETE CASCADE;
ALTER TABLE reports ADD FOREIGN KEY (reported_user_id) REFERENCES users (user_id) ON DELETE CASCADE;
ALTER TABLE reports ADD FOREIGN KEY (report_category_id) REFERENCES report_categories (category_id);