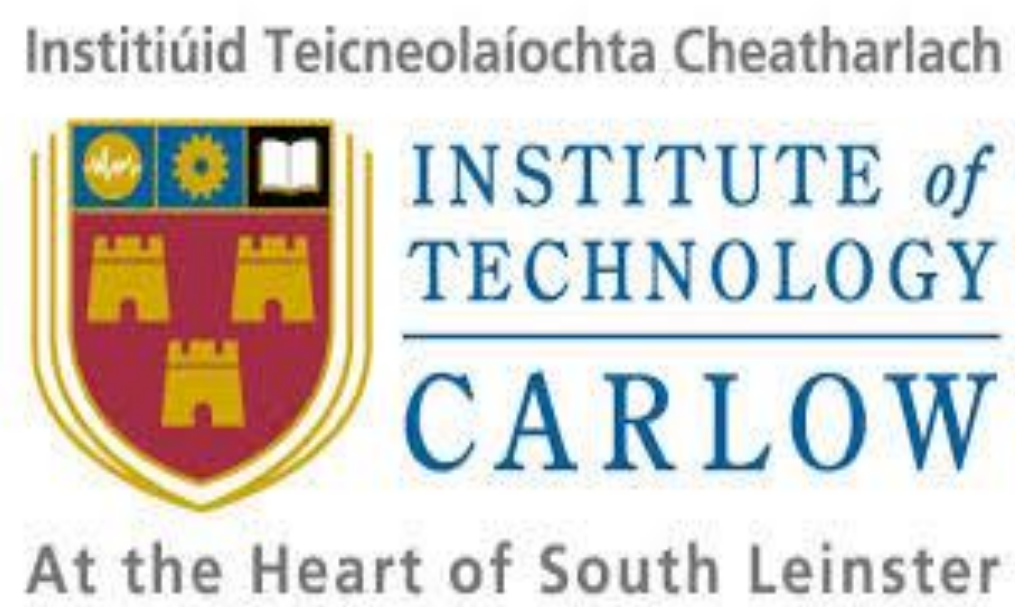

Final Year Project

Final Report

Project Title: ClickNWin

Student: Geoffrey Atkinson

Student Number: C00184861

Project Supervisor: Greg Doyle

Date: 05/04/2017

Abstract

The purpose of this report is to give a detailed description of the work that went into completing the ClickNWin project. The document will outline the work that was done, the project's functionality and descriptions of the various obstacles and challenges that were overcome in the making of the application. The document will also explore what could have been done differently to achieve the goal and the work that did not make it into this version of the application. Advice to future participants in the project will be given and a detail of the learning that was achieve during the work will be described

Table of Contents

1.0 Introduction

This report is the final document for the ClickNWin project which is a web application to be used for the sale of electronic scratch cards. This report will detail the work that was done to complete the project and the challenges that were overcome during the project's development. There were features that were considered for the project but did not make the final implementation due to time and other constraints so a detail of these features will be provided so that they can be examined for consideration in a future implementation. As the project progressed, ideas that were had during the initial phase had to be modified due to more information being discovered or design decisions being made so a summary of the project changes will be given which will be followed by a description of the different Python modules that were used in the finished product. Finally, there will be a description of the testing performed on the application, what would be done differently if restarting the project from scratch, and a summary of the learning that was achieved throughout the course of the work

2.0 Project Idea

The initial idea that began the ClickNWin project was to create an online seller of electronic scratch cards. The application should allow users to store their details on the site, make payments to increase their balance and then use the balance to buy scratch cards for themselves or for friends or family who are also registered with the site. The purchased cards could then be redeemed to see if a prize was won and if it was, they would then be able to redeem the prize back to their balance. This balance could then be used to purchase more cards or be paid to the user's accounts. To facilitate this, basic login and registration functions were required as well as the ability to store user payment details on the site. As the site was dealing with personal user data as well as sensitive credit card information, the site should be highly secure and make use of encryption techniques to store all data. This was achieved by using HTTPS to secure all transmissions between the client and the server. The contents of the ClickNWin database were also encrypted using AES 128 which uses a 128 bit encryption key. If malicious attackers were to compromise ClickNWin's database, they would still not be able to use the information without decrypting it first. It is estimated that it would take 3.4e + 38 years to successfully brute force attack AES 128. It was also necessary to use PayPal in the site as their API for

making and receiving payments was a crucial piece in allowing users to top up their site balance as well as receive payments when they wished to redeem their balance.

The web application will also allow administrators to perform some basic tasks within the application. They will be able to add new administrators, add new scratch card games to the database and modify the ones that are already there. This will allow the prices, prizes and winning chances to be easily controllable by the admin staff of ClickNWin.

3.0 Iteration Descriptions

3.1 Iteration One

The initial iteration of the project began in October 2016. The idea was then expanded through discussion and critical thinking. Work began by researching the various technologies and techniques that would be required to build the application. The different choices that could be made were weighed up and compared against each other. The initial choice for the project's main technology was C# and the ASP.Net MVC framework. These would be supplemented with the usual web technologies of HTML, CSS and JavaScript.

However, while still early in the project, before any work had been started, the decision was made to switch to the Python language and use the Flask framework. Research that had been conducted concluded that Python and Flask would be more suited to the project due to Python's efficient handling of data and the free hosting cloud provider PythonAnywhere where a python application could be hosted for free. Most of the rest of the iteration was spent on the design of the application. A functional specification was drawn up to define what the application should do and for whom. A design document was then created to model the various components of the application, how they fit together and how they could be interacted with.

In the final two weeks of the iteration, work began on building the first screens for the site so they could be prototyped at a project presentation to receive feedback and gain some insight into how the project would proceed. The basic screens for the homepage, login and registration were built and some basic routing with Python was done to allow switching between them.

Overall, the iteration was quite successful as a deeper understanding of the project requirements was gained and the initial designs were done and ready. This meant that the next two iterations could focus solely on the building of the application.

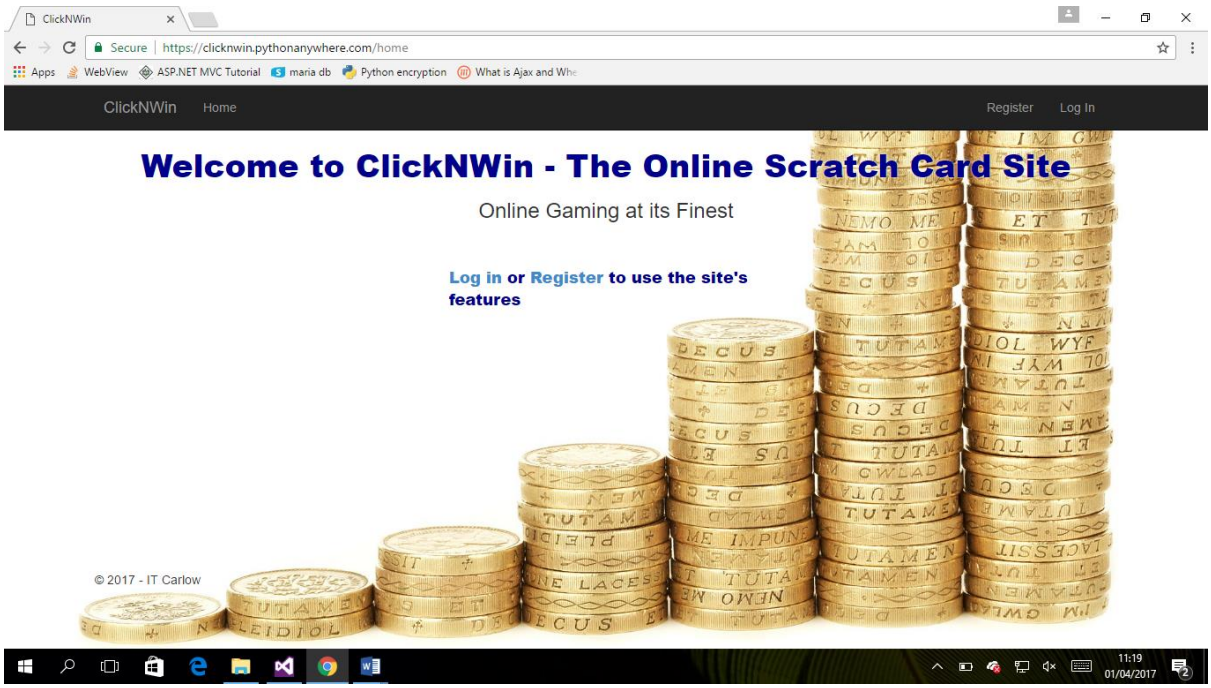

Fig 1. Initial homepage design for ClickNWin

3.2 Iteration Two

Iteration two began in January 2017. With the application's design now confirmed, work began on the basic functionalities that would be required. Login and registration facilities were the first to be added. To allow for the correct use of these functionalities, the application's database was also built. The database was built in MySQL using the MySQL Workbench GUI.

Once registration and login were complete, work progressed to the core functions of the application. A form for adding payment cards to the user account was added. Following this, the functionality for topping up the user's account balance was created. At this time, the only option for a user to top up their balance was to use a stored credit card which was sent in an Application Programming Interface (API) call to PayPal. Functionality for buying the scratch cards was then added with the scratch cards being created and stored immediately after

creation. The algorithm to decide if the cards have a prize or not runs during the card's creation, so whether a prize was won or not is decided immediately although the user will not know until they redeem the card. This was the project's major algorithm so a large portion of time was spent on testing it after the implementation to ensure it worked correctly and would output some winners of various prizes while most cards would be losers.

The main core piece for displaying and redeeming the scratch cards could now start. A data table with a list of all the user's unredeemed cards could be visited and the user could then pick one of their cards. The card was then displayed on the screen with six panels on it. As each panel was clicked, it would disappear and a prize amount displayed. If the user matched three prize amounts on a card, they would win that amount. The winning amount would then be added to the user's balance. The final function for iteration two was then added which was to allow users to redeem their balance back to their bank account. This was achieved by taking an email address for the user which was linked to a PayPal account. A call to the PayPal API was made which passed over the email and amount and the payment could be made.

There was another feedback session at the end of iteration two where the focus was the restriction imposed on users that they must use the applications functionality for storing credit cards if they wished to use the site. This was cited as an application drawback which could alienate some users. The feedback from this session was taken on board for evaluation and implementation in iteration three.

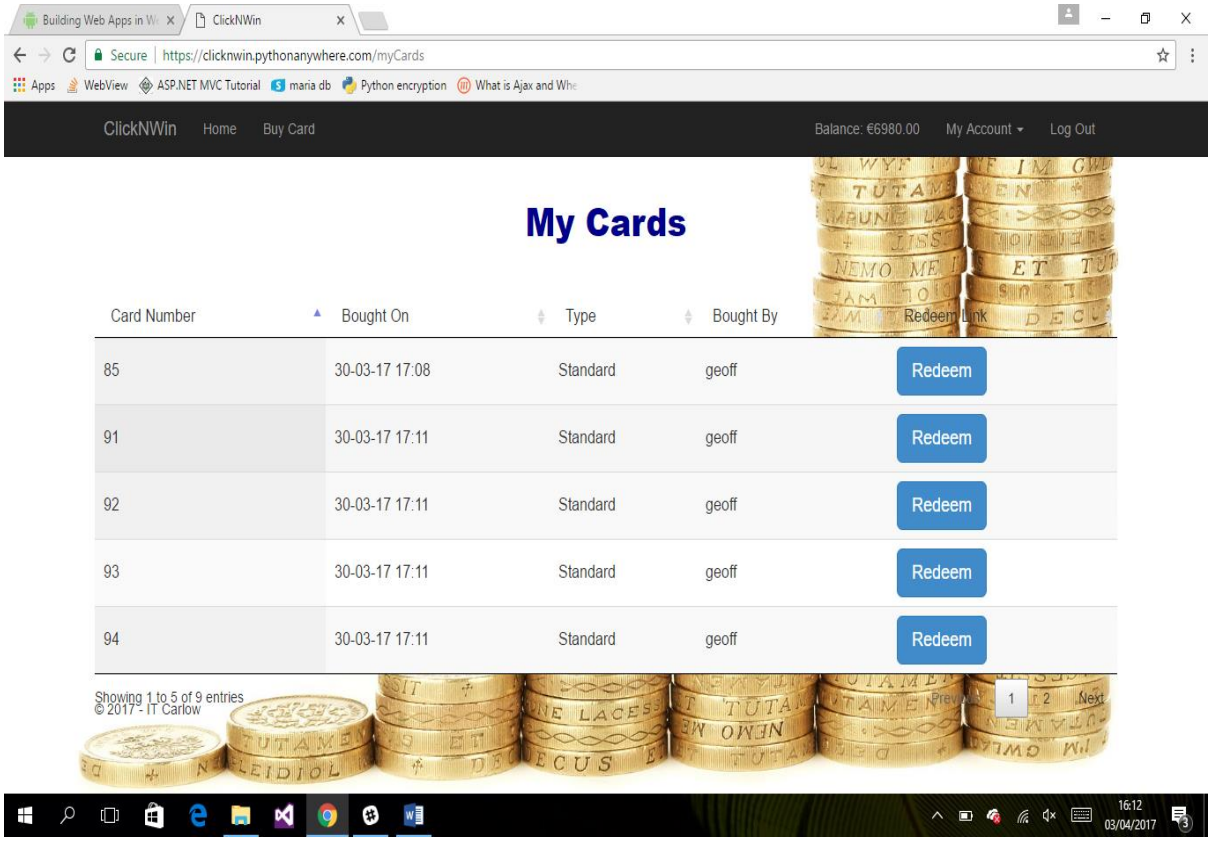

Fig 2. Display of a user's current cards

3.3 Iteration Three

Iteration three began in March 2017. The focus for this iteration was to implement features suggested by the previous feedback session and to clean up and refactor the code base in preparation for the final release. The first implemented feature was to encrypt all data being stored in the MySQL database. While this had been identified as a key feature from the start, it had not yet been implemented. Following this, work began on allowing users to top up their balance using their PayPal account instead of having to store their credit card details with ClickNWin. This was achieved with a PayPal API call which returned a URL to the PayPal website which the user could be redirected to and would then send the user back to ClickNWin after the payment was confirmed.

Following this, it was decided to prioritise the remaining work and the final piece of functionality that was worked on was an administrator back end. This added the functionality for admins to be able to add new admins to the system, create new scratch card games and modify the existing ones. When an admin is creating the games or modifying them, both the

prizes and the chances for winning those prizes can be changed as well as the price of the cards. This was the final piece of functionality added to the project. Once this was complete, the code was thoroughly read through and refactored and commented where necessary. A final deployment was then done to PythonAnywhere and the application was tested.

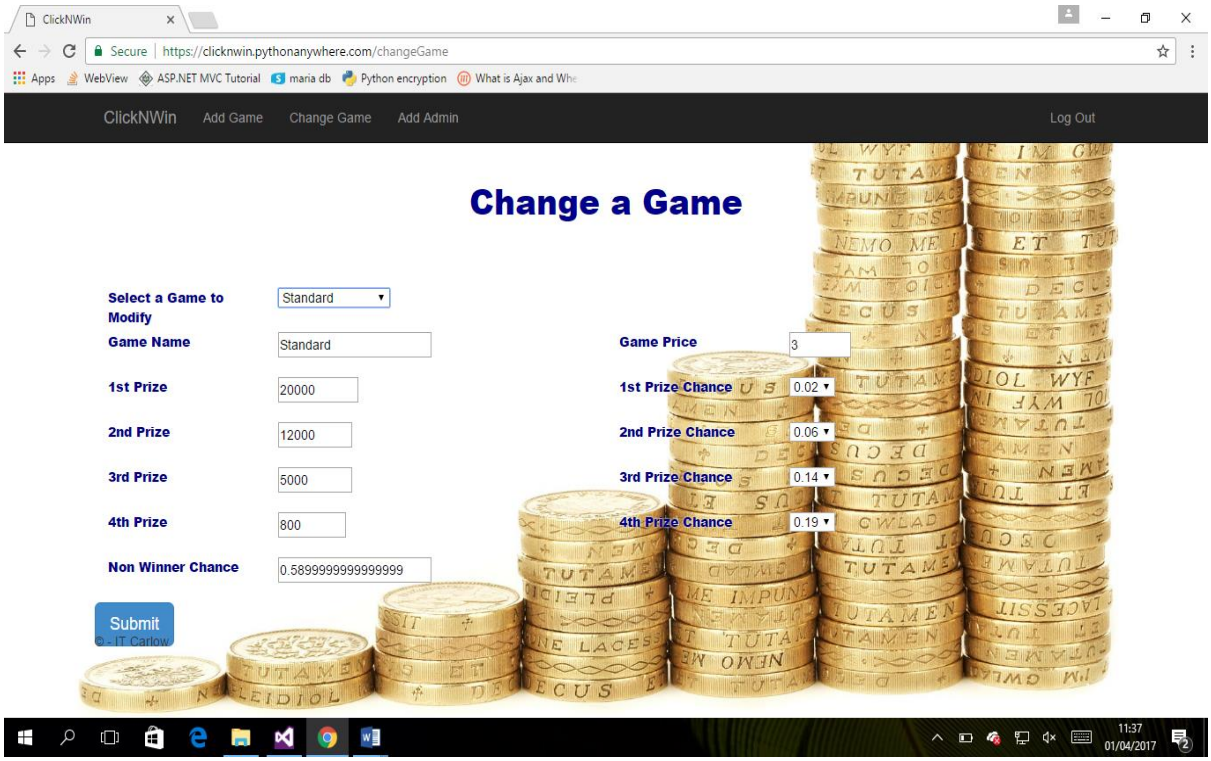

Fig 3. Admin page for modifying an existing game.

4.0 Challenges Encountered

4.1 PayPal Issues

One of the initial issues that was encountered in the project was during the initial implementations of the PayPal API. The PayPal sandbox website which was used for setting up test accounts and generating credentials for the different applications that were using PayPal to process their payments is difficult to get transaction errors from. During the initial testing of the API call to receive payments from user credit cards, the payments were not being successfully created. The PayPal websites transaction tracker showed that the payments were

an error but it was difficult to pinpoint the reason. The only hint at the error was in a log on the website where a reference to the card's type could be seen. After experimenting with the input into the JSON that was sent with the API call, it was discovered that the credit card type i.e. Visa, MasterCard, had to be sent in all lowercase letters whereas the ClickNWin database had the type of card stored with the first letter capitalised. The problem was overcome by formatting the credit card type values before they were stored to ensure it was in all lower case.

The issue was not immediately resolved as the API call still did not work with the PayPal transaction logs now pointing to the entered top up amount as a problem but still not giving a reason. More input experimentation was required before it was discovered that the API cannot handle floating point numbers that start with the decimal point. They require a leading zero. It was also discovered that there must be exactly two numbers after the decimal point not one. These problems were solved by creating the formatCurrency function that resides in the utils.py file. This function was called on all currency amounts sent to the API to ensure they were properly formatted before being sent to PayPal.

4.2 Encryption

The second issue that arose was due to a choice made earlier in the project. It was in relation to the encryption of the database contents. Earlier in the project, it was decided that the encryption would not be implemented at the start of the project as it aided the testing of the application to clearly see the values that were being stored in the database. It was mistakenly believed that it would be a simple matter to plug the encryption and decryption into the database functions and that nothing would change except for the values being stored in the database. However, towards the end of iteration three, when work began on implementing the encryption, it was realised that this was not the case. As the AES encryption algorithm that was being used from the pycrypto library could not encrypt values that were not strings, the database structures had to change to accommodate this. With the values that came back from the database now being different, pre-processing had to be added to some database functions to convert the values to how they were expected in other modules so as more work did not have to take place in these modules to facilitate the change in data type.

4.3 Balance Redemption

Another issue encountered towards the end of the project was when users were redeeming their balance they could go to the redeem balance page and redeem some funds back to their account, use the browser's back button to go back to the page and then redeem more than they had in their balance. This was due to the client side validation using the amount for the clients balance from their balance amount in the top navigation bar which would be cached when the page reloaded and still show the amount from before the previous redemption. This was a potentially serious issue as the application could have been defrauded and lost a lot of money due to this bug. To prevent this, server side validation for the amount users had in their balance was added to work alongside the client side validation so if one was bypassed the other could still pick up the issue. If the client side validation failed, due to the cached value, the server would pull a fresh copy of the user's balance and compare it to the amount sent for redemption. If the amount sent was greater than the value pulled from the database, an error message would be sent to the user and the transaction cancelled.

5.0 Future Features

During the work on ClickNWin, there were some features that did not make this version of the application due to time constraints. These features will be detailed here to help provide a priority list for the next implementation of the application. These features should be the focus of future iterations.

5.1 Mobile Version

A major feature not implemented in ClickNWin was a mobile version of the application. Time constraints did not allow this feature to be added during the three iterations of the project. Initial research suggests that it may not be too difficult to use a WebView class in Android to display the application as if it was a native mobile app while it would actually be displaying the web application without a browser around it [1]. However, as ClickNWin was not designed with mobile in mind, and therefore does not have a responsive design, this could make the pages that are displayed appear disjointed and possibly hinder the user experience. To counter

this issue, the front end of ClickNWin could be redesigned as a native mobile application and use RESTful calls to the pre-existing Flask API to update the database and other processing. This would eliminate the need for a full redesign but it is unclear yet whether this is possible and would require more research. This should be considered the priority of any future iteration of ClickNWin. While the market that can be reached by the web application is reasonable, there is the potential to open up a much bigger user base if the application was ported to mobile devices.

As part of the initial research, a working mobile version of ClickNWin, using the WebView wrapper, was created. While this is not a finished product, it demonstrates the proof of concept for creating a mobile version of ClickNWin. The mobile version could be fully launched by either redesigning the web application UI to be more compatible with mobile or by designing a native mobile front end that could potentially integrate with the Python back end. To demonstrate the mobile version, some screenshots are included below.

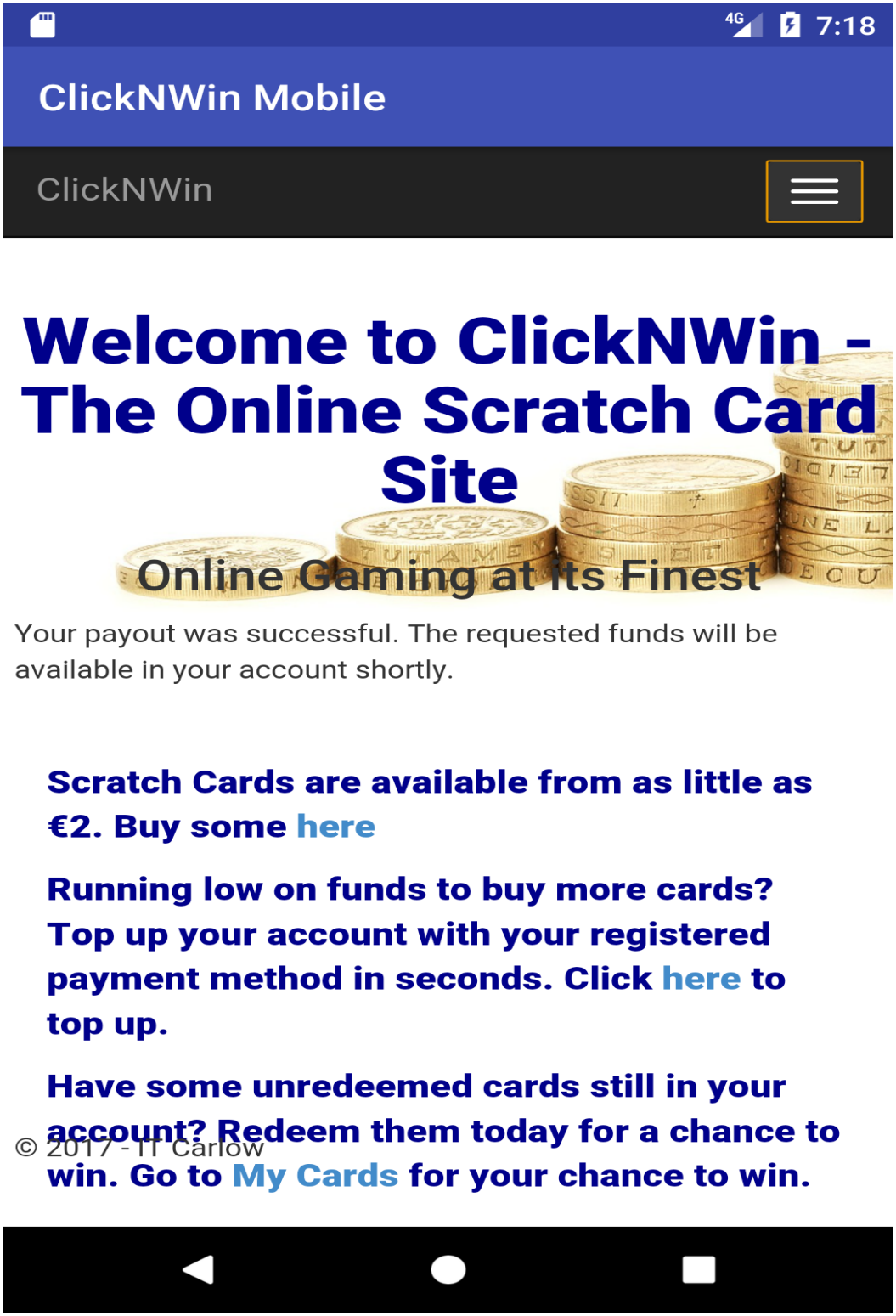

Fig 4. ClickNWin mobile homepage

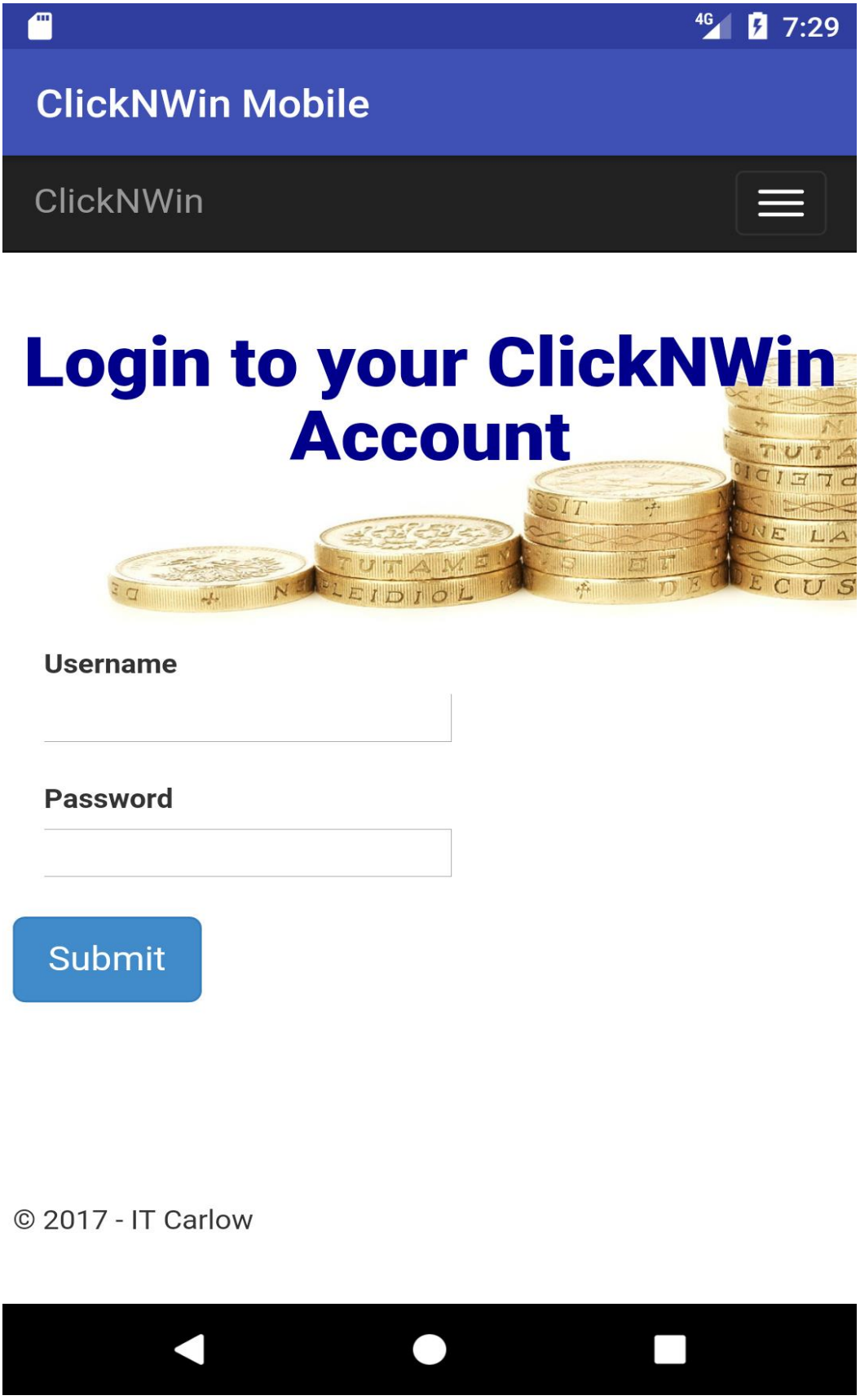

Fig 5. ClickNWin mobile login page

5.2 New Games

The second feature of ClickNWin that was not implemented but should be high on the priorities list for another iteration is the ability to handle different types of games. In its current state, ClickNWin has a basic game type that involves drawing the card with six panels on it and getting the user to click them to make them disappear and trying to match three prize amounts. While new games with different prizes can be added, the games will still be mostly similar and could eventually become stale for the user. ClickNWin could be modified to add different types of games, such as a crossword based game or even a lotto system with different numbers at given points in the week. This would require new design work, possibly new database tables and extra development which was not in the scope for the current version of ClickNWin but could be considered for its future developers.

6.0 Project Changes

Over the course of the project changes had to be made to some aspects due to new information, feedback or design choices. The major changes that were made will be listed here to provide some insight into how the project could have turned out if these changes had not been made. The changes that were made have been documented in the design and functional specification documents to represent the new state of the project.

6.1 Technology Change

The first major change that was made to the project was mentioned previously. This was the decision to switch the main server language and framework from C# and ASP.Net MVC to Python and Flask. While ASP was not a bad choice for the development of ClickNWin, the lack of free hosting options for C# projects compared against those available for Python frameworks necessitated the move as the application needed to be easily hostable for the purposes of the project. Python is also very good at working with data. During the project, the various data structures that are available and the libraries that can be found for Python

were extremely useful for the project. One of the project's major algorithms, cumulative probability for deciding if cards were winners or not, originally had a C# implementation and this was tested to ensure its accuracy. Included below are the results of a test run on the corresponding Python algorithm for comparison purposes. The original results for the C# version of this test are contained in the project's research report. The algorithm was run 1000 times and the results compiled.

Prize	**Probability**	**Expected Results**	**Actual Results**
1st	1%	10	9
2nd	5%	50	42
3rd	11%	110	115
4th	15%	150	150
Not a Winner	68%	680	684

Fig 4. Cumulative probability algorithm test results for Python

These results compare favourably with the implementation of the C# algorithm. There is variation on the expected results which is not abnormal behaviour due to the random nature of the algorithm but the actual results are not so distant to the expected results that the algorithm could be considered unusable. The results that were seen in the C# test are quite like the ones from this Python test. This was the project's major algorithm and the favourable results show that there was no negative impact on the project from the change in technologies.

6.2 PayPal Website Feature

During the second feedback session, the point was raised that ClickNWin was restricting its users to topping up their account balance by storing their credit card details with the application only. As previously mentioned, this could have the potential of alienating a portion of the user base who may not feel comfortable with allowing a new and potentially untrusted application to have access to their payment details. These users may prefer to use a more trusted and well known method of payment such as PayPal. This had not been considered previously during the design phase of ClickNWin but was a very valid point. Therefore, research went into how to use the PayPal API to redirect users to the PayPal website to complete their payments to top up their balance. It was discovered that the payments call used to make credit card transactions

could be modified to send two return URL's and the payment amount to PayPal which would then return a redirect URL that the user could be sent to where they could authorise the payment and then be returned to ClickNWin [2]. Attached below is the page where the user confirms their payment to ClickNWin. Selecting the confirm button will bring the user to a receipt page back on ClickNWin with the payment details while selecting the cancel option brings the user to ClickNWin's home page.

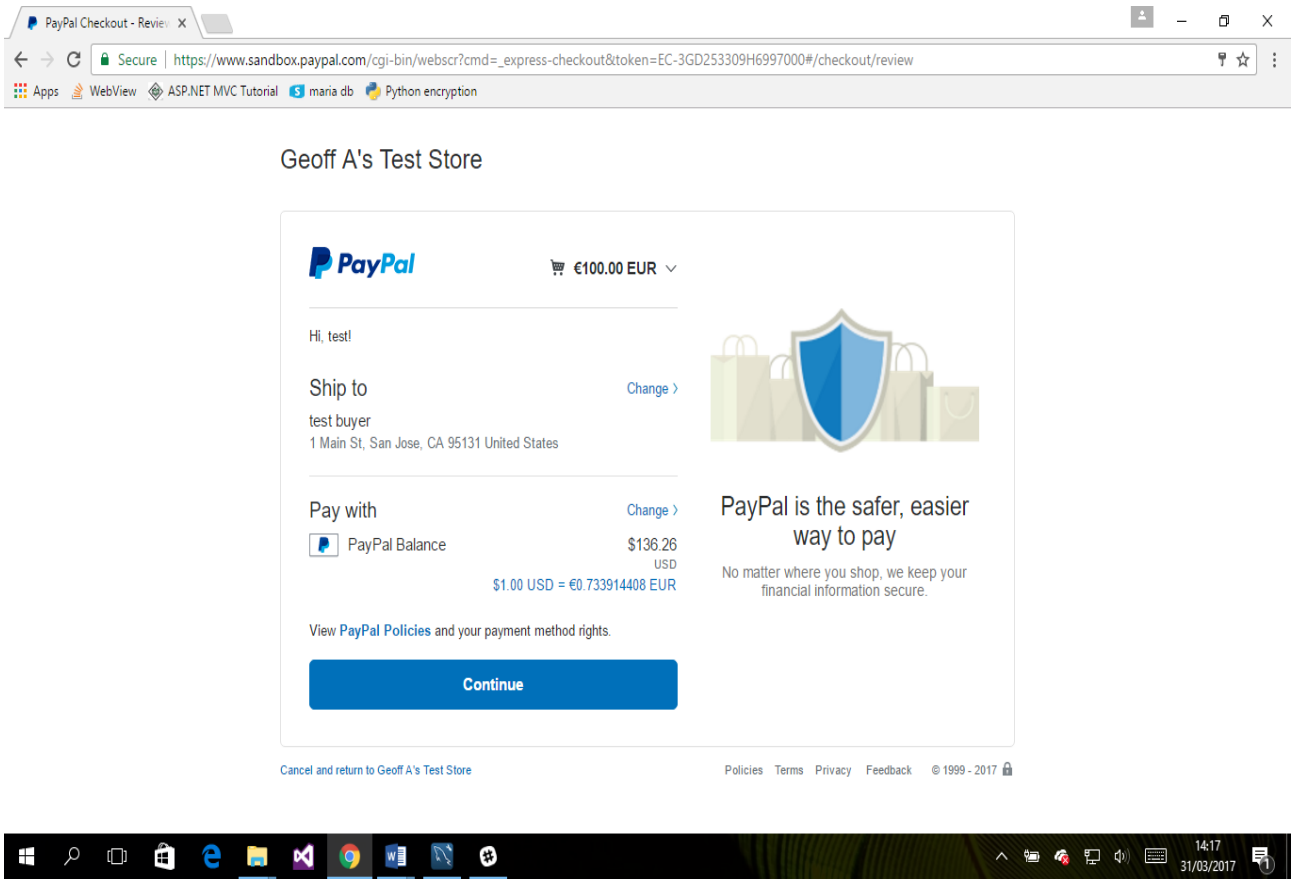

Fig 5: PayPal payment confirm page

This was a good addition to the project as it helped ensure that users not comfortable with storing their card details with the site now had a way to still use the site and not have to find an alternative.

6.3 Database Changes

Over the course of the project, the database model that had been defined in the initial design document had to be revised as new information was discovered or new ideas were generated. While the database model did change drastically, there were some new fields added which had not been anticipated during the initial design and were needed to bring the project to completion. The changes made were adding the boughtOn field to the scratch cards to provide better sorting of the cards when displayed and splitting the card holder name on payment cards into the first and last names as this how PayPal requires it to be sent.

Below is a diagram of the completed database model which was used in the finished product. All data types are strings of various lengths except for the id fields which are integers and the redeemed variable on scratch cards which is a tiny integer. PK denotes the primary key for the table and FK denotes a foreign key.

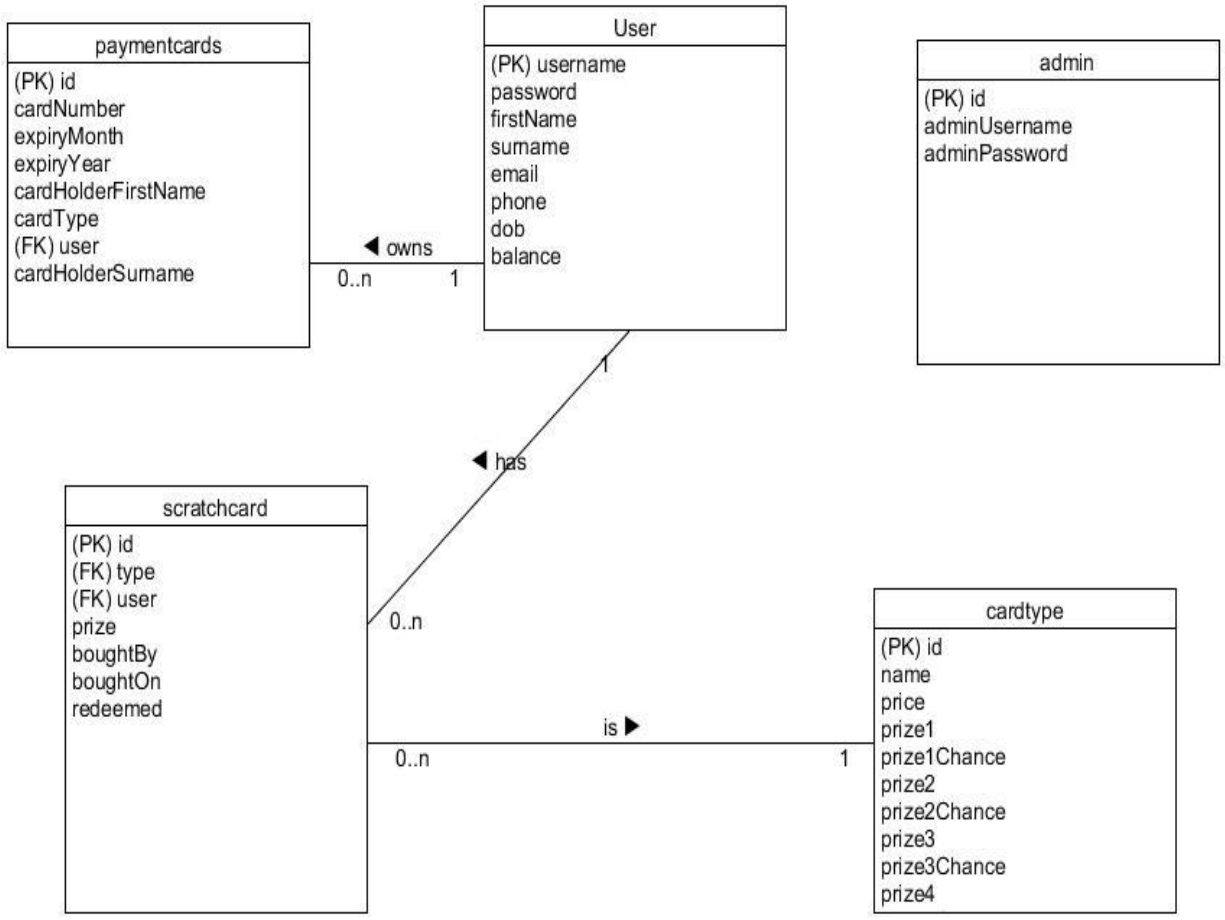

Fig 6. Final database model for ClickNWin

6.4 New Admin Functionality

The design document for this project described only one administrator functionality for the application. This was the ability to modify the existing scratch card games. Towards the end of iteration three, when this functionality was about to be implemented, it was decided to add two additional administrator functions to make the administrator section less bare and provide important functionalities for the project that could leave it lacking if they were left out. The extra two functionalities were the ability to add new admins and the ability to add new scratch card games. Allowing new games to be added meant that they would not have to be manually added by allowing admins to modify the database and therefore a high degree of control over the new games could be maintained by enforcing the values that could be put into the games using form validation.

7.0 Module Descriptions

The following section will give a brief description of the functions for the different Python modules that were created for ClickNWin.

7.1 Views

The views module provides the main routing and viewing for the application. All the main pages for the application are rendered from this module. Most of the pre-processing for any data that is being rendered on a page or being sent for storage is done in this module along with database calls that are necessary to retrieve or insert the data. The Flask library is the main dependency for this module.

7.2 Admin

The admin module is used for routing and viewing the pages that serve administrator functionality. There is a decorator function which prevents non-logged in admins from

accessing the pages and there are database calls to retrieve and store the game data that admins will be changing.

7.3 API

The API module contains routes for AJAX calls that are made from the client. There are several functions here which can asynchronously retrieve and return data to the web pages. The most important function can check if usernames already exist for the registration and card buying forms.

7.4 Database

The database module contains all calls to the MySQL database. These can store, update and retrieve data for the application. All database functions in this module use the encrypt module to encrypt and decrypt the data that is flowing through them to keep the application database secure. The connection data for the database is stored as a global variable within the module. The module uses the DBcm library as a dependency.

7.5 Encrypt

The encrypt module uses the AES encryption algorithm to encrypt and decrypt all data that is being stored or retrieved from the database. The secret key and initialisation vector that the algorithm uses are stored as global variables in the file. This module uses the pycrypto Python library as a dependency.

7.6 Utils

The utils module contains some useful utility functions. It was decided to separate these functions from the other modules due to them not fitting the category for being in the other

modules. There are functions for creating new scratch cards, processing card payments and formatting currency for the PayPal API.

7.7 PayPalAPI

The PayPal API module contains the functions that make calls to PayPal for credit card payments, redirects to the PayPal website and pay outs when customers want to redeem their balance. Much of each function is written in JSON which is then sent to PayPal for processing with necessary variables plugged in when the calls are made to the PayPal module by the application. There is a global variable which contains the client keys necessary to make the calls. The module uses the Python library PayPalRestSDK as a dependency.

8.0 Testing

Early in the project, it was decided that automated testing would not be included in the scope for this project. The main form of testing that has performed throughout the project is manual testing. Testing was conducted constantly during the development process. As new features and functions were added to the application, thorough testing was conducted to ensure that they were as bug free as possible and were correctly performing the functionalities that were required of them. The testing was performed by multiple users who came from both a technical and non-technical background so that a wide variety of opinions could be sought and proper feedback be gotten for the application.

Before the application moved onto the commercial market, it would be extremely beneficial to put together a full testing group for the application to get feedback and insights into how it would be perceived after release. This focus group would be made up of potential users of the application and would involve them spending some time testing all functionalities of ClickNWin and giving their feedback. This feedback could then be used to refine the application before a full release.

9.0 Project Advice

With the project now complete, I can look back and critically analyse how I approached it and the advice I might give to myself or someone else doing a similar project. While mistakes were made throughout the work, each one of these is an important learning point that can be reflected on and becomes important experience when doing future projects.

9.1 Make Changes

The technology change previously mentioned in this document was a modification that would have had an impact on the outcome of the project. While the change was necessary to make the project a success, I still regret not doing an in-depth project with C#, ASP and MVC. These technologies are not only very interesting, they are also still very relevant. The current Tiobe index ranks C# ahead of Python in its rating system with the two languages coming in at four and five respectively [3]. Using these technologies would have required a much different approach to the project as they have different ways of building web applications and handling databases. It would have been a different kind of challenge to use these technologies in the project and I believe I would be coming out at the end of the work with a very different skillset.

C# requires far more proficiency with classes and objects than Python and I would have had to increase my knowledge of these programming concepts to make the project a success, had I used these technologies. However, while I do have a sense of regret over the change, I am still happy with the decision to change technologies. Python is still a highly relevant language and I had some prior experience with C# with the possible opportunity for more within my working environment in Unum. The chance to gain the knowledge to really go into detail in a new language and produce a project of reasonable scope could not be turned down and I was happy with my decision.

To anyone in a similar position, I would advise them to think hard about what they would want to do with the project. The project's entire design does not have to change in the last couple of weeks but decisions made early in the project especially before any major coding has begun can still be reversed and new opportunities explored and inserted into the project. I would advise someone doing a project not to feel bound by design choices made in the first few weeks or think that the original design and specifications for the project have to be rigidly conformed.

Some of the best ideas for the project will most likely occur while coding the application and should be explored and experimented with if they can bring value to the application.

9.2 Listen to Feedback

At all stages of the project I received feedback from a variety of sources. My supervisor, the other supervisors, classmates and even family and friends who I discussed the project with. When building a widely accessible application like ClickNWin, feedback is an invaluable source of ideas that can potentially be incorporated into the application. It can range from simple features that can improve the user experience to unthought of issues that could pose serious problems for your project. As previously mentioned, some important ideas for the project came through discussions after I had explained the application at project demos or in conversation. The idea to allow users to buy scratch cards for their friends was an idea that came up through some initial discussions I had about the application. The potential issue that could have arisen by forcing users to store their credit card details within the application was raised at a project presentation and made me realise the importance of always trying to provide the user with options rather than forcing them into what I though as the best solution.

Ideas for the project can come from a variety of sources so I would advise to always try discuss the project with different people. Explain the idea, get opinions and see what people think. The idea they have may not be something that can be put into your project at that time or it could become the cornerstone of your application. Therefore, in the spirit of agile, it is always important to seek and analyse feedback.

9.3 Documentation

When I began the project, there was a whirlwind few weeks of constant writing and research for the documents but as soon as this was finished, I dived straight into the coding and only looked back at documents to remind myself of some design or functionality. I never updated them during this time with any new information I had discovered or the changes I made to my design. It is key to keep the documentation of the project as up to date as possible at all stages of the work. Whenever new ideas, functionalities or just a different way of creating an existing

functionality are being explored, always update the relevant documentation as this will save a lot of time during the final weeks of the project when the focus should be on the final documents and testing.

10.0 Learning Achievements

10.1 Technical Achievements

After completing this project, I feel I have gained a good knowledge and understanding of how to define, design and build any kind of web application project in the future. I was able to use the skills learned throughout my time on this software development course to create a fully functional and well-designed piece of software that with the right funding could be put out to the commercial market. While I was working with very specific technologies, Python and Flask, most of the work that was done on this project is easily transferable to any other similar project using any kind of web application framework. The thought processes for creating web applications are fundamentally the same and the only major difference would be learning the new language and framework. The experience with using the core web technologies of HTML, CSS and JavaScript are also transferable to almost any web project and the new knowledge of using AJAX I could put into practice in several places throughout this project is also an extremely useful skill to have as AJAX is becoming a more common feature in modern web applications [4].

While it is good to have the overall understanding of the application development process, I was also very satisfied with the level of competency I gained in using Python and Flask. If I moved onto to doing a project with a different language, there would almost certainly be a learning curve for me at the start of the project. However, I think that if I was to begin a new project using even a different Python framework, I would almost certainly be able to get going a lot quicker and be far more comfortable with the work.

During the work, especially the coding work, I came across many small challenges which would probably be trivial to an experienced developer but were new to me. These ranged from small oversights in functions I was coding to larger errors like the previously mentioned error around users being able to redeem more than they had in their balance. Making these kinds of mistakes throughout the project has thought me some of the simple things to look for when

creating applications like ClickNWin. Small oversights, like not checking what happens when a user clicks the back button, can quickly turn into application breaking bugs which could sink a project very quickly. I am grateful that I have had the chance to make these mistakes now and learn what to look for in the future as this will give me a good grounding for when I move forward with new projects in the future.

10.2 Personal Achievements

Apart from the technical learning achieved throughout the project, I have also learned skills that will be useful away from the software development process. The research process and writing I had to use when undertaking this project is transferable across any number of disciplines as is drafting of formal documents which has been used throughout this project.

My greatest learning throughout the process is the cultivation of the mental discipline and strength that is required to complete a project like this. Much effort over a long period of time must go into a project such as this one and it is very easy to lose motivation during the process. I am happy with the way I approached the project and the work that was accomplished. I believe that the discipline learned over the last six months of this project will stand to me in good stead in any job or role I will have going forward.

Acknowledgements

First, I would like to thank my project supervisor, Greg Doyle, for his guidance and help throughout the project. His inputs into my ideas and thoughts on how I should proceed with the proceed with the project helped shape the outcome of my project.

I would also like the rest of the project supervisors. Their input during the presentations and willingness to help whenever I had questions about my project after classes allowed me to form new project ideas and make my final work far better.

Finally, I would like thank my classmates who were always on hand to help with questions or problems and were always willing to drop their work for a few minutes to look at mine with no complaint.

References

[1] – Chris Ching/Code with Chris. 2016. How To Make an App For Your Website In Less Than 30 Minutes. [ONLINE]. Available at: http://codewithchris.com/make-an-app-from-website/#webviewapp. [Accessed 29 March 2017].

[2] – Payments API. (No Date).[ONLINE]. Available at: https://developer.paypal.com/docs/api/payments/. [Accessed 24 March 2017].

[3] – Tiobe. 2017. Tiobe Index March 2017. [ONLINE]. Available at: https://www.tiobe.com/tiobe-index/. [Accessed 31 March 2017].

[4] – Jake Rocheleau/VanDelay Design. 2016. What Is Ajax & How Is It Used In Modern Web Development? [ONLINE]. Available at: http://www.vandelaydesign.com/what-is-ajax-webdev/. [Accessed 31 March 2017].