Final Year Project - 2015/2016



Dublin Institute of Technology

Kevin Street

D8

**Online Generic QR Coding System**

Paul Moran | C12370211

DT211/4

Computer Science (Infrastructure)

Supervisor : Mark Foley

**Project Declaration**

I hereby declare that the work described and presented as part of this project and dissertation is entirely my own work (except where specifically stated) and has not been submitted as an exercise for a degree at this or any other university.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Paul Moran

Acknowledgements

I would like to thank my supervisor for this project, Mark Foley for his encouraging advice throughout the year, and his guidance through tough aspects of the project. His advice paved the way to additional material which significantly aided in the completion of the project.

I would also like to thank Damien Bourke for his guidance from the start of the project.

I would also like to express my gratitude to Charles Alexandre Gagnon, Seamus Kilmartin, and my family for their individual contributions in helping to test various aspects of the project.

Thank you all very much as I would have struggled significantly without your help during the process of the project development.

Table of Contents

[1. Project Introduction 5](#_Toc447562544)

[1. Project Statement 5](#_Toc447562545)

[2. Project Objectives 6](#_Toc447562546)

[3. Structure of Project Report 9](#_Toc447562547)

[4. Conclusion 10](#_Toc447562548)

[2. Research 11](#_Toc447562549)

[1. Introduction of Background Research 11](#_Toc447562550)

[2. Existing Systems and Solutions 12](#_Toc447562551)

[3. Technologies Researched 19](#_Toc447562552)

[4. Other Research Performed 32](#_Toc447562553)

[5. Resultant Findings/Requirements according to Research 32](#_Toc447562554)

[3. Design 33](#_Toc447562555)

[1. Introduction to Design 33](#_Toc447562556)

[2. Approach and Methodology 33](#_Toc447562557)

[3. System Architecture 35](#_Toc447562558)

[4. Design of User Interface 36](#_Toc447562559)

[5. Design of System Structure 38](#_Toc447562560)

[6. Design of Source Code 39](#_Toc447562561)

[7. Project Features 40](#_Toc447562562)

[8. Other Design Documents 41](#_Toc447562563)

[4. Development and Implementation 43](#_Toc447562564)

[1. Introduction of Implementation 43](#_Toc447562565)

[2. Development of Database 43](#_Toc447562566)

[3. Development of MVC – Model, View, Controller 46](#_Toc447562567)

[4. Development of HTML Templates 47](#_Toc447562568)

[5. User Interface 48](#_Toc447562569)

[6. Development of Functions and Factories 49](#_Toc447562570)

[7. Implementation of QR Code Features 50](#_Toc447562571)

[8. Use of Other API’s and Software 54](#_Toc447562572)

[9. Conclusion 55](#_Toc447562573)

[5. Project Validation 55](#_Toc447562574)

[1. Introduction 55](#_Toc447562575)

[2. Testing 56](#_Toc447562576)

[3. Demonstration of Project Features 56](#_Toc447562577)

[4. Problems and Issues 56](#_Toc447562578)

[6. Project Plan 61](#_Toc447562579)

[1. Introduction 61](#_Toc447562580)

[2. Deviations from Initial Plan and Design 62](#_Toc447562581)

[3. Changes to be addressed if Project was repeated 62](#_Toc447562582)

[4. Future Development of Project 63](#_Toc447562583)

[7. Conclusions 66](#_Toc447562584)

[1. Introduction 66](#_Toc447562585)

[2. Report Summary 66](#_Toc447562586)

[3. Project Summary and Heuristics 70](#_Toc447562587)

[4. Learning Outcomes 70](#_Toc447562588)

[5. Personal Reflection 71](#_Toc447562589)

[6. Final Project Conclusion 72](#_Toc447562590)

[Bibliography 72](#_Toc447562591)

Table of Figures

[Figure 1 - EventBrite Logo 15](#_Toc447566399)

[Figure 2 - EventBrite Interface 16](#_Toc447566400)

[Figure 3 - EventBrite Promo Logo 19](#_Toc447566401)

[Figure 4 - Parking Tag(Site Failure) 20](#_Toc447566402)

[Figure 5 - Big3 : Browsers 22](#_Toc447566403)

[Figure 6 - Sample Development Tools 26](#_Toc447566404)

[Figure 7 - NCPS Printed Parking Ticket 27](#_Toc447566405)

[Figure 8 - Quick Response Code 27](#_Toc447566406)

[Figure 9 - Code39 Barcode 28](#_Toc447566407)

[Figure 10 - Standard Irish Postal Stamp 29](#_Toc447566408)

[Figure 11 - Franking Stamp 29](#_Toc447566409)

[Figure 12 - Paypal Payment 31](#_Toc447566410)

[Figure 13 - Agile Development 37](#_Toc447566411)

[Figure 14 - Technical Architecture Diagram 38](#_Toc447566412)

[Figure 15- Standard MVC w/Factories 40](#_Toc447566413)

[Figure 16- MVC Venn Diagram 41](#_Toc447566414)

[Figure 17 - Top Comment Box 42](#_Toc447566415)

[Figure 18 - Use Case Diagram 44](#_Toc447566416)

[Figure 19 - Condensed Gantt Chart 44](#_Toc447566417)

[Figure 20 - Expanded Gantt Chart(w/o Report Development) 45](#_Toc447566418)

[Figure 21 - DAO Folder Contents 47](#_Toc447566419)

[Figure 22 - DBManager Class 48](#_Toc447566420)

[Figure 23 - index.php 49](#_Toc447566421)

[Figure 24 - Model Construct 50](#_Toc447566422)

[Figure 25 - View Construct 51](#_Toc447566423)

[Figure 26 - Controller Construct w/Switch 51](#_Toc447566424)

[Figure 27 - Login Screen 53](#_Toc447566425)

[Figure 28 - Customer Template 53](#_Toc447566426)

[Figure 29 - Employee Template 54](#_Toc447566427)

[Figure 30 - Admin Template 54](#_Toc447566428)

[Figure 31 - Sample Table(Customers Events) 54](#_Toc447566429)

[Figure 32 - Switch Snippet 55](#_Toc447566430)

[Figure 33 - Method Called By Switch in Controller 55](#_Toc447566431)

[Figure 34 - Authentication/Validation Construct 56](#_Toc447566432)

[Figure 35 - Logged In User Check 57](#_Toc447566433)

[Figure 36 - Get Generated QR 58](#_Toc447566434)

[Figure 37 - Read Excel Spreadsheet w/Scanned Data 60](#_Toc447566435)

[Figure 38 - Get Employee's Service (readExcel.php) 60](#_Toc447566436)

[Figure 39 - Get Employee's Service from DAO (Model.php) 60](#_Toc447566437)

[Figure 40 - Return SQL Query with Employee's Service (employeesDAO.php) 60](#_Toc447566438)

[Figure 41 - Perform Stamp Check on Scanned Data 61](#_Toc447566439)

[Figure 42 - Perform Parking Ticket Check on Scanned Data 62](#_Toc447566440)

[Figure 43 - Perform Event Invite Check on Scanned Data 62](#_Toc447566441)

[Figure 44 – Data Loaded to Database 63](#_Toc447566442)

[Figure 45 - Scanned Data Table(Employee w/ All Privileges) 63](#_Toc447566443)

[Figure 46 - cURL command to send SMS via. Clickatell 64](#_Toc447566444)

[Figure 47 - Call SMS API (Controller.php) 64](#_Toc447566445)

[Figure 48 - MailGun API to send Email 65](#_Toc447566446)

[Figure 49 - MailGun API being called (controller.php) 65](#_Toc447566447)

# Project Introduction

## Project Statement

The idea behind the Generic Online QR Coding System is to simplify some everyday tasks for users by removing some inconvenient aspects of those tasks.

Essentially it will allow the user to generate a barcode which will be used for a specific purpose in which the system will recognise, and when scanned the system will handle the appropriate data accordingly.

This barcode will be unique to each use, each user and to each specific task.

The applications which will be implemented to demonstrate the use of this system will be as follows:

* Postal Stamp Generation
* Car Parking Ticket Generation
* Event Creation

The use of this system could be developed and implemented on a much broader and general scale. It could be used a general identification system, or validation system in various infrastructures, such as transport, postage systems, essentially anywhere that needs proof of payment, validation or identification.

## Project Objectives

#### Proposed System Utilities

The purpose of the Generic Online Barcoding System is quite simple. The main aim is to make QR Codes which can be used in almost any situation to replace old fashioned, inconvenient activities which we may or may not have to perform every day.

The targeted solution is a generic website which will allow registered users to perform various tasks in which the use of Visually Encrypted Codes will make these tasks a lot easier and more controllable.

On registration of the website the user will receive an SMS message notifying them of registration. This will be used as a part of identifying the valid attendee or driver who the barcode represents.

In order to prove this concept, it has been decided upon to focus on 3 aspects (Postal Stamping, Event Creating, and Car Parking).

The user will have the option, within the website, to make use of these 3 utilities.

**Postal Stamping** – Should the user decide they wish to send a letter, but have no conventional stamps in their possession, this system will generate a barcode containing the required information to simply identify the purpose of the barcode, the user, and the date it was generated.

The user simply has to print this barcode and stick it to an envelope/package and post it.

When the code is scanned, the appropriate stamp duty will be deducted from the users account. This process can be completed by using the users’ personal banking details, or a third party such as Pay-Pal or Stripe. Stripe's third party simulator will be used to fake a monetary transaction.

The process eradicates the personal inconvenience of an individual having to buy a conventional postage stamp, but yet they still manage to send their letter and pay the stamp duty with very little effort.

**Event Creating** –

This aspect of the system will show how the use of how this generated code can be used to provide a sufficient amount of control as well as ensuring validity of the entrants to the event. For this aspect the event creator will have the option to create the event and specify the names of each person attending in which on arrival they will be given their appropriate barcode/ QR code tag to as proof of ID and validity. The other option available for the creator will be to create the event, and publish it to a number of email addresses who are also members of the system. The guests will receive notification of this event both online and as an SMS and choose to *attend/reject.*

Should the guest decide to choose 'attend', a name tag will be generated for that individual user. I.e., when this user (guest) arrives at the event, they will be given the name tag which will contain their information embedded in a barcode. Before that user receives this name tag the provider will be able to know if this person is who they say they are due to the fact that their account contains a picture of them. (In a real life situation, Naming identification may also be used).

Again, systematic control, and validation is provided with the use of this system in the given scenario.

**Car Parking** -

Although it may seem that this is three systems in one, effectively it is one system which can be used for multiple use and purpose. In this final implementation of its use we will see its effectiveness to allow people to park their car without having to worry about purchasing parking tickets.

Firstly, a user simply goes onto the website, and chooses that they wish to generate a parking ticket. The ticket is generated and is printed off. The user can then use their mobile to top up the amount on the ticket. The amount they choose to top up by determines the duration of the ticket.

Approaching the final 15 minutes of the duration, the user will receive a text notifying them to either return to their car or to top up to extended their stay.

#### System Requirements

* User Roles – Customer, Employee, Admin

Provide access to the system for different type of users. In the case of this system, the users present will be:

1. Customer – Generates the QR codes for various applications
2. Employee – Scans and Validates QR codes for different aspects of the system depending on which service they provide.
3. Admin – Can access almost all aspects of the system and perform CRUD operations where needed.

* Customer Self-Registration to System

Any user with access to the system can register with the system to access and use the generation capabilities of the system. The user may only generate a QR code is they have a linked Stripe account to the system.

* Generation Of Stamp

Each registered user(customer) can generate a new postal by providing the necessary parameters to determine the price of the stamp when scanned into the system. The pricing of the stamps are based on the following criteria:

* + - Destination
    - Weight
    - Postage Type
* Generation Of Car Parking Ticket

Registered customers can generate their own parking ticket in which they should have the option a pay during generation for a specified duration, or they may generate a ticket in which is not valid until they go onto their account and *top-up* their parking ticket by a choice of either:

* Duration
* Amount
* Generation Of Event and Invites

Registered users must also be capable of creating an event in which they can specify the number of invites to send out. Invites should be sent out by email. This email will also be used as an identifier for other users on the system in which if they receive an invite to their email account, the event will also be added to their list of events on the system.

* Automatic SMS distribution

When certain operations or criteria are triggered, an SMS message should be sent to the users provided mobile number to notify them of a particular event regarding a QR code linked to their account.

* Automatic Emailing of Event Invites

When a user creates an event and provides the email addresses of the invited people, the emails should be sent out automatically to each invited person automatically during the creation of the event in the system.

* Automatic Payment method using Stripe

When a QR code validation requires a payment from the users Stripe account, the payment should happen in the background automatic preferably. An alternative to this would be to notify the user of a payment about to occur in which they can *confirm* or *deny* this payment which is to occur in which they may wish to change their mind.

* Scanning QR Data into System using Android Application

For the employee user of the system, they must be able to use an Android device to scan a particular QR. When the QR is scanned into a file in the system. They may next load this data to the database if they wish. The data which is loaded to the database will determine the validity of the scanned barcode based on various criteria such as payment, expiry dates and ownership.

* Manipulation of Data scanned into the System

When the data is scanned it is important that multiple accurate checks are performed in order to make sure that a conclusive decision is made by the system as to whether or not the data scanned is valid or not.

* Administration capabilities such as adding new employees, new admins, and perform CRUD operations on all data tables in the database.

As an administrator of the system, they must have the capability to create, read, update and delete almost all relevant data in the database.

## Structure of Project Report

In this report we will be looking at each and every aspect during the research, planning, design and development of this project.

The chapters throughout are as follows:

#### Chapter 1 – Project Introduction

In the first chapter the idea behind the project will be outlined by mentioning some of the problems in which it may be a suitable solution to. The project goals and objectives will also initially be mentioned here to outline some of the common features expected in the project. Finally the structure of the project report will give glimpse into the content of each other chapter.

#### Chapter 2 - Research

Within the Research chapter every aspect of research performed either prior or during the design or development of the project is outlined here. Topics mentioned include background research into the purpose and proposal of the project idea, evaluation of similar systems and technologies and other researched sources such as questionnaire results, possible user’s opinions, and advice given by other members of the IT sector.

Finally the chapter will conclude with the overall given results and findings of the researched data and information.

#### Chapter 3 – Design

In this section of the report, all design and structural aspects will be portrayed. Firstly, the approach, methodology and system architecture is expanded on based on the research results given in the previous chapter. For the next three sections to this chapter the physical design of the User’s Interface, the System Structure and the Source Code will all be explained here. Following on from the design of some of structural coding parts, details about other feature’s uses are described. Lastly, detailed design documents including use cases, ERD’s etc. are provided.

#### Chapter 4 – Development and Implementation

The Development and Implementation chapter will outlined the process regarding the building and development of the system. The chapter has been broken up into separate sections outlining the development of each aspect. Those aspects include:

* Database
* General MVC Framework
* HTML templates
* Functions
* QR Code Features
* Other API’s and Software.

Finally the chapter will conclude on the development while giving a short summary on the development phase.

#### Chapter 5 - Project Validation

Testing analysis and evaluations will primarily be outlined in the Project Validation chapter. Testing methods such as White/Black box testing will be mentioned and described where used. Brief results will be outlined before the last section in which all results and findings will be evaluated concluded. After the Testing section, simple individual demonstrations will be provided given examples of the various aspects of the system. Links and screenshots are also provided. Problems and issues which occurred throughout the development phases are also outlined here. This topic includes issues and solved and unsolved as well as describing the used solution.

#### Chapter 6 – Project Plan

Within the second to final chapter, aspects and possibilities of the finished system will be described. Firstly, deviations of the initial proposed idea will be outlined as the system did not turn out fully as expected. If the project was to be repeated, changes and new approaches are proposed here as well new ideas and solutions to unresolved issues. But to conclude this part of the report, the topic outlining possible future plans for development uses of the system will be portrayed in detail.

#### Chapter 7 – Conclusions

In the final chapter of the report, a summary of the report will be provided given a short reminders of the chapters read. Most importantly, learning outcomes for the developer will be described along with some personal reflection regarding experiences, choices and preferences throughout the participation of this project.

Lastly a final project conclusion will be presented.

## Conclusion

As described in the previous sections, we have looked at the possible scenarios in which this project will be able to be applied. The goals and objectives outlined will be discussed further in the report in almost each chapter where relevant. After given a brief overview of the chapters and topics to come, next we will look at the research executed and how certain results and evaluations were made in order to decide various technologies and approaches to be used.

# 2. Research

## Introduction of Background Research

This section briefly outlines the following information which will be expanded on progression through the chapter:

* Why my system is relevant
* Various uses of my system as well as brief similarities
* Simplicity and Flexibility of its use
* Inconvenient aspects of other systems in place

Considering the proposed idea various research was carried out in regard to how this system may be deployed. It was decided that a website using a stable, flexible framework would be sufficient to demonstrate and implement the system. After having some basic experience with MVC (Model, View, Controller) it was decided that that would be the foundations of the system.

Companies such as An Post which was developed in 1984 have been using the same conventional postage stamp system since it was erected. It methods, though effective, are quite old-fashioned in terms of ordinary conventional postal letter. Whereas with the Generic Barcoding System users would not need to leave the comfort of their own home to purchase a postal stamp. Ref. [1]

Parking Tag is another company who have similar systems in place to that of the system in question. Although its methods are very modern, with the Generic Online Barcode Generating System users can make their own parking tickets and top up when needed all within one site. Ref. [2]

Finally, another site which reflects some similarities is Eventbrite (2006) which allows users to create, promote and host events of their choosing. This site charges a rate based on the amount of tickets sold through the site as well as a service fee. Whereas with the Generic Online Barcode System, self-generated barcodes/QR codes will be used to identify each type of ticket sold to each individual person effectively with control and sufficient validation. Ref. [3]

Before taking aspects such as design and approach into consideration, a list was drawn up of solutions and ideas to a number of everyday tasks which this system may be able to improve in one way or another. After speaking to number of friends, colleagues, classmates, family and lecturers and explaining some of these simple applications of the proposed system, one thing everyone agreed on was that with the implementation of this project, it will reduced or completely eradicate some very inconvenient everyday tasks such as purchasing a postal stamp Ref. [4], or returning to your parked car to buy a new parking ticket or even as a control mechanism to validate entrance to a venue etc. It is almost a known fact that the purchasing of postal stamps in today’s world is rather out-dated and old-fashioned. With the exception of stamp collectors Ref. [5], having to buy a stamp in order to send a letter in this ever growing technological world is very inconvenient.

After speaking to two family members and two neighbours, who are approximately between the ages of 40-60, they too agreed that they would be willing to register and use the system to give them access to simplified tasks such as generating their own postal stamp, generate their own parking ticket when needed so they would not need to leave the house, or return to their car.

Although improvements in this regard cannot be outlined for applications such as Eventbrite, the inclusion of event management using this system will prove yet another application this systems use.

Next we will look deeper into the existing systems in which have just been mentioned.

## Existing Systems and Solutions

#### Introduction

In the following section some of the research aspects of the project will be outlined in regard similar systems and software which are available. After extensive research gathered about multiple companies, software, and systems have been found which provide similar services to the project in question. During this chapter the key similarities and differences between these systems and software are described and evaluated.

In order to achieve such a comparison, a list of heuristics (similar to Nielsen’s Heuristics Ref. [6]) have been drawn up to compare each system to and created a rating system. Based on the results of this chapter, an accurate evaluation will be drawn up.

#### Development Heuristics

As mentioned before, some systems researched contained similar concepts, goals and principles as this project. In this section the positives, negatives, similarities and contrasts between each of these systems including the Generic Online QR Code System will be detailed. Based on some of the results gained from research gathered, some faults and improvements will be outlined which may affect the design of this project.

***Login/Registration***

The process in which a new user is added to the system. However, this process should not be in any way a struggle to a new user. Questions asked and information provided should be straight forward for both an experience as well as a less technically advanced user. In terms of logging in, should false parameters be provided whilst a user is logging in, sufficient error messages should be displayed? Logging in should also be clear and user friendly.

***Site Navigation***

The website should be easy to navigate and tasks should be easy to perform without additional help from guides or internet assistance.

***Scanning System***

Labels, stamps and barcodes, when scanned return accurate, valid and correct information regarding the sender of the parcel/letter/parking etc.

***Labelling System***

The system in place marks each package with the necessary token to distinguish one package/letter from another as well as providing minimal, yet sufficient information to allow for successful postage. This mark/stamp/barcode will successfully establish a sufficient relationship with the appropriate sender and the system in place.

***UI and Aesthetics***

The User Interface is pleasing to the eye and template details are aligned correctly. The font and sizing throughout the website is clearly readable and each detail which is necessary for the user is easily perceived and understood.

***Help and Documentation***

Help, information and sufficient guides are provided throughout each part of the website should the user of any experience need it.

***Suitable form constraints***

Each form throughout the website has sufficient data handling techniques, regular expressions are used where needed, and frontend error handling is in place to prevent erroneous data from being received by the system.

***Accurate Data Handling***

When data is read into the system, is it handled appropriately giving back accurate results and information?

#### Similar Systems for Evaluation

###### System 1 – EventBrite



Figure 1 - EventBrite Logo

Eventbrite is an event creation website which allows a user to create, organise, host and promote events of their choosing. Charging applies based on the number of tickets sold, with the exception of free events.

* Registration & Login – 8/10
* Site Navigation – 9/10 Simple, Lists other events on throughout the coming weeks
* Scanning System – 7/10 Scanning System works as supposed to, but the ticket can be used by anyone
* Labelling/Addressing System – 10/10 Very similar to that of my own proposed idea. Barcode is present on a conventional ticket to be scanned.
* UI and Aesthetics – 9/10 Easily Readable website, with white background.
* Help and Documentation – 10/10 Explains what to do one the first page on entering the website.
* Suitable form constraints – 8/10 When creating an event, the User has the option to accurately detail the event being created.
* Accurate Data Handling – 8/10 On approach with a valid ticket, it is scanned returning the validity of that ticket, although the ticket is unique to other tickets but is not unique regarding the ticket holder.

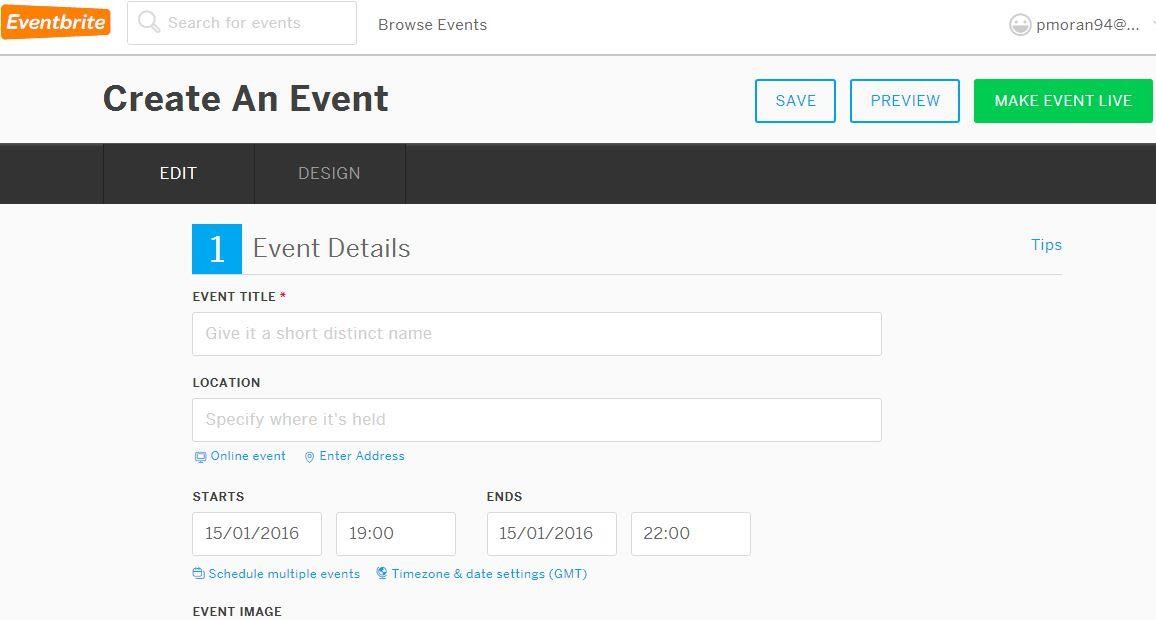


Figure 2 - EventBrite Interface

###### System 2 – Franking Machine

Similar to this project idea, Franking is a process where a company can register a stamp or logo, which is printed onto a postal package. This stamp contains the *price paid, date, ID number* and *logo.* Most Franking Machines have a built-in scale to calculate the price of the package. The Franking stamp is essentially proof that the postal package has been paid for by the company directly from their account.

Ref. [7]

* Registration & Login – N/A
* Site Navigation – N/A
* Scanning System – 10/10
* Labelling System – 10/10
* UI and Aesthetics – N/A
* Help and Documentation – N/A
* Suitable form constraints – N/A
* Accurate Data Handling – 10/10

###### System 3 – An Post

An Post is the current postal system in place in Ireland. They use standard postal stamps on their letters and packages where a sender must purchase a standard stamp prior to sending the letter. They also use the concept of the Franking Machine as seen above for bigger packages sent by businesses and companies.

* Registration & Login - N/A
* Site Navigation – 8/10 (Small website, Less is more)
* Scanning System -9/10
* Labelling System – 3/10 Standard Postal Stamp
* UI and Aesthetics – 4/10 Light Green font with white background -Not clearly visible(Even with contrast option)
* Help and Documentation – 7/10 Most is covered
* Suitable form constraints – 6/10 Not much interactive content
* Accurate Data Handling – 7/10

###### System 4 – Parking Tag

Parking Tag is a convenient way to pay for the parking of a vehicle by text. The user simply registers their car registration and the time and place they have parked.

Sample Text would contain the word Park, with a two letter area identification code and the number of minutes in which the user wishes to park. Like so:

Park DA 60 to 53311 = Parking in zone DA for 60 minutes Ref. [8]

The authorities then scan the car registration to determine the validity of the payment. This system acts in a similar way to the Car Parking aspect as the Generic Online Barcoding System as it requires registration, topping up and scanning. The same concept is to be incorporated into the development of the system in question.

* Registration & Login – 10/10 Very quick efficient straight forward
* Site Navigation – 9/10 Clear (not over developed)
* Scanning System – 8/10
* Labelling System – 8/10 Car Registration
* UI and Aesthetics – 7/10 (3 main colours)
* Help and Documentation – 8/10 Short concise , and answers the most relevant questions
* Suitable form constraints - 6/10 Any value can be entered for County(), Mobile number is not limited
* Accurate Data Handling – 8/10

#### Application of Heuristics

**Unweighted Evaluation of Systems**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Heuristic** | **System 1** | **System 2** | **System 3** | **System 4** |
| **Registration & Login** | 8/10 | -- | -- | 10/10 |
| **Site Navigation** | 9/10 | -- | 8/10 | 9/10 |
| **Scanning System** | 7/10 | 10/10 | 9/10 | 8/10 |
| **Labelling System** | 10/10 | 10/10 | 3/10 | 8/10 |
| **UI and Aesthetics** | 9/10 | -- | 4/10 | 7/10 |
| **Help and Documents** | 10/10 | -- | 7/10 | 8/10 |
| **Form Constraints** | 8/10 | -- | 6/10 | 6/10 |
| **Accurate Data Handling** | 8/10 | 10/10 | 7/10 | 8/10 |
| **Total** | 69/80 | 30/30 | 44/70 | 64/80 |

*Table 1 – Unweighted Heuristics*

**Weight Evaluation of Systems**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Heuristic** | **Weight** | **System 1** | **System 2** | **System 3** | **System 4** |
| **Registration & Login** | 10 | 8\*10=80 | -- | -- | 10\*10=100 |
| **Site Navigation** | 10 | 9\*10=90 | -- | 8\*10=80 | 9\*10=100 |
| **Scanning System** | 15 | 7\*15=105 | 10\*15=150 | 9\*15=135 | 8\*15=120 |
| **Labelling System** | 20 | 10\*20=200 | 10\*20=200 | 3\*20=60 | 8\*20=160 |
| **UI and Aesthetics** | 15 | 9\*15=135 | -- | 4\*15=60 | 7\*15=105 |
| **Help and Documents** | 5 | 10\*5=50 | -- | 7\*5=35 | 8\*5=40 |
| **Form Constraints** | 10 | 8\*10=80 | -- | 6\*10=60 | 6\*10=60 |
| **Accurate Data Handling** | 15 | 8\*15=120 | 10\*15=150 | 7\*15=105 | 8\*15=120 |
| **Weighted Total** | 100 | 660/100 | 500/50 | 535/90 | 805/100 |
| **System Score** | .out of  10 | 6.6 | 10 | 5.94 | 8.05 |

*Table 2 – Weighted Heuristics*

#### Evaluated Results

1. System 1 – (Event Brite)

All of the heuristics were applied to this system. The website is very easy to navigate and easy to read.

Forms and event creation is extraordinarily easy for the user. This aspect is one in which will have to be taken into high consideration whilst developing the Generic QR Coding System.

The labelling system is very similar to the system being developed although the system in question will be making the QR codes unique to both the event and the user not just to other individual tickets. Below is an image of the introduction page which clearly identifies the options/steps in which a user can perform with the system.

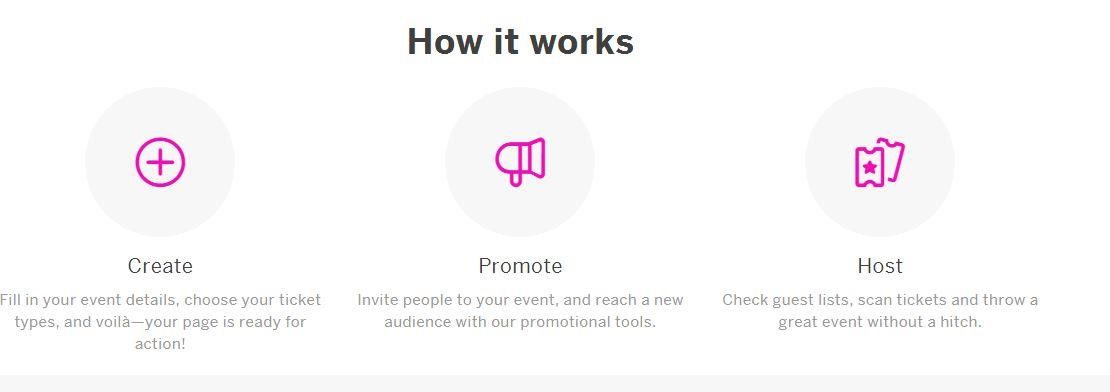


Figure 3 - EventBrite Promo Logo

1. System 2 – (Franking)

Franking uses just 3 of the heuristics listed in this exercise. However in terms of its score and use it is perfect according to these heuristics.

This proves that the purpose of the Franking machine is perfect for the job it does. It is simple, effective and very efficient for companies and businesses. The process of this technology is something which I will be taking into consideration during further design aspects of this system

1. System 3 – (An Post)

7 out of 8 of the heuristics listed are applied to this system.

Website was small and easy to navigate, although the fonts and colours used did not contrast with each other well. Some formatting was lost after the *change contrast option* is pressed.

The used labelling system, is a single postage stamp which the sender requires in order to send a post. Postage stamps are purchasable from the post office and participating newsagents and retailers. Fault is that the postage stamp needs to be purchased prior to sending, which is rather inconvenient for a conventional sender.

The website provides good help and FAQ's for the user.

Overall the system is very standard but the method in which is used by a user to send a postal letter is very out-dated and inconvenient.

4. System 4 – (Parking Tag)

This system uses all of the heuristics. Overall the website is very short, concise website with good aesthetical features. Although during the duration of research this website was inaccessible for users on multiple occasions.



Figure 4 - Parking Tag(Site Failure)

But when the site is accessible it is rather easy to navigate, and easy to understand.

I have realised the importance of correct form controls as the forms within this website didn’t have proper controls and should the user make a mistake, they would have to reset the form. This can be very inconvenient for any user.

Similar scanning system as the idea in my project. Scans tag for car, and will be notified of the parked cars validity. The concept is based on the use of car parks providing a specific code which represents that car park area.

Overall the service has a good website, and the system uses quick convenient methods to check the validity of the parked cars.

#### Conclusion

So after researching these systems it is clear that the weighted heuristics demonstrates how each system is similar in its own way. Although not all characteristics were applied or necessary to each system. This was due to the fact that each of these systems provided different services to each other, but provided some similar concepts to that of this project. A *System Score* was given based on how well that system does its own job. In System 2(Franking), although it does not use or required many of the heuristics mentioned, it however does what it's supposed to do very effectively based on the weighted totals given to each topic.

Therefore, after seeing the results of each system valuable evidence has been gathered in order to evaluate, improve and include various ideas and aspects to incorporate into the development of the system in question.

## Technologies Researched

#### Introduction

As mentioned in the previous chapter there were various software, technologies and systems which were discovered which provided a similar service as the Generic Online Barcoding System whether that be its overall purpose or partial.

In this chapter various technologies, software and tools are listed which are available and suitable for use in order to develop this project.

Pros and cons, advantages and disadvantages to each of these technologies and architectures are described in this section, as well as hardware aspects which may have been required.

Furthering the software aspects which may be used, various platforms and programming languages will be discussed which may be used in order to complete the goals and objectives that have set for the development of the system.

#### Platforms/Browsers

-Considering the BIG 3 (Chrome, Firefox, Internet Explorer)

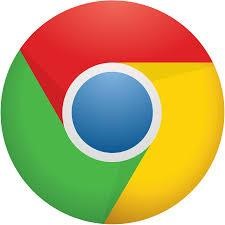


Figure 5 - Big3 : Browsers

* ***Google Chrome***

Overall the dominant internet web browser. Almost all newer systems are using it for one purpose or another. It has a number of positive aspects such as the feature to allow for non-responsive pages to be killed without shutting down the entire browser**.** It also has the translation function to detect a foreign language within a web page and allow the user to convert the text. Ref. [9]

It also allows the user to download new content to a page without having to restart the browser.

Although all these gadgets provide the user with control, it simply does not make for the better browser available.

* ***Firefox***

Firefox is next in line to Google Chrome. It has a huge library of add-ons available for the general user. Allows for great customization of the browsers making it not one of the easiest to use.

One major fault of Firefox, is that it can automatically allow hidden add-ons to attach to your browser causing it to run slower. Although it has some good security features, it has a slow page load time on image-heavy pages. Ref. [9]

* ***Internet Explorer***

Known for being slower than the rest of the browsers, it also has greater security features. It allows the user to block sites from cataloguing behavioural habits on the web.

This can be automatic or sites can be added manually. Ref. [9]

#### Methodologies

#### Programming Languages

One troubling aspect of design was deciding what language to use when building my framework. I personally have much more experience using PHP, but after some brief research I was also persuaded towards the idea of learning Python Django.

So essentially, the debate was between using PHP or Python's Django as a backend framework. Other languages and possibilities were also researched. Ref. [10]

#### Backend Languages

* ***Java (JVM Languages)***

Java is often used by the bigger companies, but this does not necessarily mean it is the best choice. It is primarily based around reliability but can be rather expensive due to over complicated web solutions. Nowadays, using Java as a backend solution programming language is been more or less replaced by the newer, fresher languages such as *Play, Scala etc.*  Although I have sufficient experience in general Java programming I personally only have basic experience with it as a backend language. Ref. [11]

* ***Python***

Python is known for being very fast for development and prototyping. Coding is easier to maintain and refactor. Python is one of the languages in which most of the web is designed, along with PHP and Ruby etc.

A lot of people argue that the likes of Python is rather error-prone, although in comparison to Java based languages, it is easier to control and develop with whereas with Java languages development is slowed due to its hierarchy and makes refactoring hard. Python's newest popular web-framework, Django is growing ever so fast and in a positive way. It is open-source and is constantly being developed almost every day.

According to my research using Django as a backend framework would work well with my system, although unfortunately I have not used Python in quite some time therefore I can only put it under consideration. Ref. [10]

• ***C***

C is one of the oldest programming languages and yet it's still holding firm in the Top 10 Ref. [11]. C is a cross-platform development language and is compatible with compilers available for almost all machines and operating systems. C is a very well structured language, although this does not provide good flexibility for a developer when trying to achieve particular functionality. But because of its smaller vocabulary it is easier to read and maintain.

In my personal experience of C Programming, it was the first programming language which I became accustomed to using. Although its vocabulary is quite smaller, it doe's reduce flexibility during development and can be very time consuming trying to figure out some particular solutions.

• ***PHP***

PHP, is possibly the most popular backend language although this is debated. Personally, I have some strong general knowledge with developing with PHP in particular areas. It has a huge support of frameworks and tools compared to Python. It is also very simple and fast to find and debug errors within a system.

PHP was solely designed for web-development. One of the most popular websites which used PHP during its development was, and still is, Facebook.

One of the unnoticed advantages of the use of PHP is its ability for rapid reloading. This is where simple changes the PHP code you get immediate feedback, without server restart, whereas with Python there is a slight delay. Ref. [10]

Although I have only been introduced to PHP in the past year, I have become quite accustomed to it. I have had some regular experience in web development in PHP during my participation of college assignments. From some of the feedback I have researched about PHP, it seems perfect for the development for my system.

* ***Go***

Go is one of the newer languages is stream at the moment. It's basically formulated around the best aspects of both Java and Python. It provides a simple yet powerful hierarchy.

One of the known downsides to developing in Go, is the fact that there are plenty of tools surrounding it, although some of these tools are not as mature as some of the ones which we are more familiar with in other languages.

Overall, according to my research, teams who make the switch with involving Go in there projects are often happy with the results.

* ***Hack***

Hack is a language which is formally a superset of the functionality provided by PHP. It was developed by Facebook in order to improve reliability within the PHP world of programming.

Programming languages such as *Hack* are mainly proven as solutions to some particular company programming solutions.

Since I have had some experience with PHP, I will be having a look into *Hack* as the following weeks commence. With general PHP, Hack may prove quite useful during the development of my system.

#### Frontend Languages

* **JavaScript**

JavaScript is very simple and versatile, and is used to extend multiple functionalities in a website. One positive aspect of JavaScript is that it executes on the client-side of the website and does not fully need server connection. It is fast and effective to get a suitable frontend UI for the user. Some downsides to the use of JavaScript include security issues. Ref. [12]

* **JQuery**

JQuery, essentially is a god send for JavaScript. In essence it condenses blocks of JavaScript syntax into smaller, easier blocks of code to perform various tasks.

It has a very simple syntax and requires a lot less code than that of JavaScript. Although it is generally a much simplified version of JavaScript, it can be very limited in terms of functionality. Ref. [13]

* **HTML**

HTML(Hypertext Mark-up Language) is the programming language which allows for the creation of websites. It uses tags as identifiers throughout its syntax. It uses hyperlinks to move from page to page or to determine the output on the screen. It is relatively simple to learn and is used for website templating and layout. It controls the layout, format, and links throughout a webpage. Should java script, CSS etc., be used they will all be included or be linked using HTML tags in order to interact with the page. Ref. [14]

* **CSS/Bootstrap**

Essentially CSS(Cascading Style Sheet)/ Bootstrap is the styling of a webpage. Height, width , colour, shading, style, transition, position, margins etc. can all be defined in CSS file and called in separate file such as HTML page to apply certain styling where needed.

* **ActionScript**

ActionScript is OOP Language designed specifically for web-animation. ActionScript is often used for user – keyboard interaction. It allows for the development of onscreen environments to be made a lot simpler. It is an *event-based* language Ref. [15], which means that actions are triggered by events.

Newer versions of the language have been released which bare some similarities to JavaScript. Ref.

[16]

#### Development Tools and Software

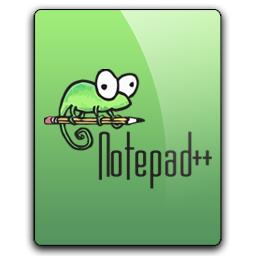


Figure 6 - Sample Development Tools

* ***Sublime Text***

One of the more powerful text editors, and known for being perfectly suited with keyboard savvy users. It supports multiple languages. It is a general purpose text editor which provides some useful built in tools to navigate, specific selection, quick edit a particular document. It is very useful should the developer need multiple windows open during work. Ref. [17]

* ***Notepad++***

Notepad++ is very similar to Sublime described above. It is a lightweight tool, with plenty of add-ons and plugins to satisfy the user. It supports over 50 languages. Ref. [18]

One negative aspect is that official download is exclusive for Microsoft Windows systems, as well as the fact that some earlier versions have been known to crash, although this has been fixed in later versions and updates.

* ***Eclipse***

Eclipse is known for being a rather heavy editor, which comes in multiple forms and flavours depending on what the developer wishes to use it for. The UI is very configurable and provides a good folder structure for your project. Similar to the other two editors mentioned above, it too has plenty on plugins and add-ons, including Android SDK, ADT etc. Ref. [19]

Its main advantage is that it is well suited for a larger scale project, with multiple collaborators.

But because of its bulky nature, it can be very slow and resource heavy. Since there are so many different plugins and “flavours” it can be confusing choosing which ones to choose based on what goals may have been set. Java is the basic and most supported language with Eclipse.

##### Conclusion

The overall results gained from the research into these three code development tools it has been concluded that because of its simplicity, flexibility, ability to make light of work it was decided that *Sublime Text* should be used during the development of the system. It is simple to download, simple to use, has multiple built-in tools for development and has a very aesthetically pleasing interface.

Eclipse proved to be quite “buggy” from time to time and very heavy to implement. Notepad++ is quite nice to use to it did not contain all the same built-in tools which sublime text contained.

#### Labelling Systems (based on Similar Systems)

• ***Parking Ticket***

Parking Tickets in Ireland are simple and basic but the user is required to purchase one from a machine every time they wish to park somewhere. A parking ticket is usually left in clear visibility under the windscreen to be checked by the inspector. The ticket will display the time of purchase, amount paid for, and the time of expiry.



Figure 7 - NCPS Printed Parking Ticket

 ***QR Code (Quick Response Code)***

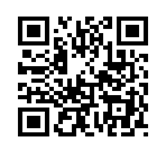


Figure 8 - Quick Response Code

*Figure 3 – QR Code*

A QR Code is a two-dimensional barcode matrix. A QR Code uses 4 standard encoding modes to store data within this visual 2D barcode (Numeric, Alphanumeric, Binary, and Kanji). A QR Code is arranged in a square image populated with black modules and can be read using most imaging devices such as a scanner or a camera. Their use has grown due to its fast readability and it's capability to store more data than that of a standard UPC barcode

* ***Barcode (ExtendedCode39)***

Code 39 Barcodes are one of the more used, and simpler barcode styles. It contains a character selfchecking method, making it simpler to develop. It can represent numbers, upper-case, lower-case, the space character, as well as some special characters. Ref. [20]



Figure 9 - Code39 Barcode

* ***Conventional Stamp(Sticker)***

A conventional Postage Stamp, is generally a simple sticker like visual proof of payment of stamp duty in order to allow for a package/ parcel to be sent through a postal service. By law, a postal stamp can contain the following: Ref. [4]

* Country Emblem
* The word Postage(in relevant language)
* The Expiry Date
* Year of Issue
* Personal Stamps can contain image of person(alive)
* Logo of Private Company
* Copyright Symbolism
* Registry Mark



Figure 10 - Standard Irish Postal Stamp

* ***Franking***

Franking is a process used usually by bigger companies to send a package or letters. Franking in general is the application of stamps or markings to identify the validity of a posted letter or package. A franking machine is used to apply an individual stamp which can bare the personalized logo of a company along with the date of application, price paid and other baring marks. It is a simple and effective concept but it can be costly to purchase a larger franking machine. But should a general user wish to own a franking machine it too need to be purchased and refilled. Ref. [21]



Figure 11 - Franking Stamp

##### Conclusion

According to research gathered, it has been found that for this particular system the use of barcodes and QR Codes are perfect for their purpose. They are both simple to generate, and can encrypt enough information to store for multiple purposes.

They also require less production cost as they do not require coloured ink, special paper or special design like the conventional postage stamp, franking machine or parking ticket. They are both simple and effective.

#### Database Technologies

* ***Oracle***

The Oracle DBMS is one of the most trusted and widely used relational database engines. Data is structured to allow users to access it through the use of SQL. It is built on a relational database framework. It also contains a component to allow cross network compatibility. Oracle runs well on most operating systems. One of the main features of Oracle is that is allow for robustness, which means there is no single point of failure which will bring the database down. Ref. [22]

* ***SQL Server***

Microsoft's answer to Oracle. It supports standard ANSI SQL languages. Over the years SQL Server has been focussing on advanced featuring options, and security. It bares the very same architecture as Oracle which is a plus for developers. Ref. [23]

* ***MySQL***

Another competitor to Oracle and SQL Server, is owned by Oracle Corp. Its source code is freely available as it was originally developed as freeware. It was developed in C and C++ and is compatible with major Operating Systems. Ref. [24]

MySQL is mostly known for web-hosting applications, due to its web-optimizing features as well as the fact that it is freely available. It is part of the structure which is known for delivering advanced web applications (Linux, Apache, MySQL, and PHP (LAMP)). MySQL is often used as a backend data handling system, used in sites such as Google and Facebook.

#### Third Party Payment Method

* ***Pay-Pal***

Pay-Pal is a well-known third-party payment company where users register an account with PayPal pay for goods and services using its utilities rather than paying directly from their personal banking accounts using their banking details. Other websites and companies often allow their good/services to be paid for through Pay-Pal when it is built into their website.

Pay-Pal also provides a demo simulation of payments into company accounts from made up accounts to give a user/developer a taste of what they may be expecting should they use or implement the service in their website.

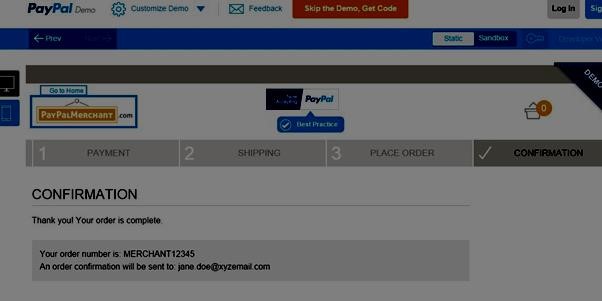


Figure 12 - Paypal Payment

* ***Stripe***

Stripe is a very similar service as Pay-Pal regarding how it may be used in this project. It too provides a demo service to allow both consumers and developers to experience its service within their purchases and websites.

It was developed in Ireland and provides powerful tool-kits, and is perfectly scaled for any website or infrastructure.

##### Conclusion

Ref**.** [25]

|  |  |  |
| --- | --- | --- |
| **Characteristics** | **Pay-Pal** | **Stripe** |
| **Transaction Fees** | Charges 2.9% + 30c, but this rate is discounted as higher valued purchases are made. | Charges a flat rate of 2.9% + 30c per successful charge. This rate does not change. Service fees tend to be generally cheaper than Pay-Pal |
| **Security** | Now stores cards in vaults, but data still has to go through the servers, which puts a burden on security. | Use Stripe.js, which means the transaction data is sent directly to stripe and not servers. Server breaches not disclose details. |
| **API** | RESTful PayPal API is now more developed and documented since Stripe. Whereas it was quite buggy before Stripes release. | The Stripe API is clean, welldocumented, and easy to use therefore making other API's up their game. |
| **Data Portability** | Pay Pal has a severe lack of data portability in regards to registered customers. | Stripe on the other hand has great data portability and allows its customers to move and transfer important card data easily. |
| **Customer Service** | Pay Pal provide both a phone service and an emailing service should a customer need to contact , although recently they have been known to have very poor responsive rates | Stripe only provide an emailing service, but this emailing service is a 24hr live, free service, open to both engineers, and Stripe themselves. |

*Table 3 – Third Party Payment Comparisons*

Therefore based on the following research found which notes the key differences between *Pay-Pal* and *Stripe* I have decided to implement Stripe's demo API to handle money transactions within my system.

#### Deployment Method

Deployment of the system is a very important aspect, although the intended approach is to develop the majority of the system locally, then later deploy it to ensure that it is in fact responsive to other browsers.

Methods available for deployment are as follows:

* Microsoft Azure Virtual Ubuntu Virtual Machine

Microsoft Azure requires a subscribed account using a key which is provided by the head of the school of computing in DIT. This gives me certain allowances as to what I can deploy using their service. I was also advised to use it by other students who were successful using it in the past. It also provides constant up time and database storage. Ref. [42]

* Okeanos Virtual Machine

Okeanos Virtual Machines are free to use but are often very unreliable in terms of their up time. Therefore deploying on this would prove very inefficient considering some automatic scripts using crontab are to be used within the system. Ref. [38]

* Heroku

Heroku provides a PHP and MySQL deployment base using ClearDB. This is set up using a Heroku Toolbelt which can be downloaded from their website and used within the Windows Command Prompt. Using GIT commands the website repository is created and pushed on the Heroku server. Next you simply connect the database to the ClearDB database.

Although this method seems rather effective the choice of using Microsoft Azures seems more reliable and maintainable. Ref. [39]

#### Methods of Contact

As part of the system it is important to notify the user of a particular event regarding certain occurrences which relate to the QR tickets they generate.

The methods of contact which were shortlisted are as follows:

* + - Email

By using an Email account as a means of contact for this system would mean that the user would need to be connected to the internet on a regular basis. Although emailing is a rather effective method of contact it is not as instant or convenient as other methods listed. By using PHP’s built-in mail method it will be possible to easily send email once the system is deployed.

An alternative to this would be to use a third party API such as MailGun’s Emailing API Ref. [43]. The use of this API would allow for emails to be sent when the system is still being developed locally.



Figure - MailGun API/SMTP

* + - SMS Messaging

SMS Messaging is the most convenient method of contact for this system, as it means that as long as the user has their mobile in their possession they will receive a notification indicating a certain event regarding their account. It is fast, reliable and the most convenient method listed.

This method would require the use of a third party API such as Clickatell. The downside to this is that in order for it to be implemented in a professional environment it requires paid subscription.

For this project, if this method is to be used a test account will be created. Therefore no payment is needed although the content of the message which is sent is restricted to that of a test message.

* + - Online Account Notifications

This method is by far the most inconvenient method as it would require the user to keep logging in and out of the system to check if they have received any new notifications on the system. It would also mean that the user would require internet access to receive any notification regarding expired tickets, approaching parking ticket expiry time, or upcoming events.

As a result of these factors it has been chosen that SMS messaging is the most ideal for the system given the fact that it is convenient to the user, does not require constant internet connection and is just as effective. It also means that if this system was to be made public for professional use it would simply just require an account update on Clickatell’s web service.

#### Scanning Devices

After looking at multiple Android applications which provide the use of a QR code scanner, the one which showed the most promise was the Barcode Scanned provided by ZXING Ref. [47].

By using this application along with TEC-IT’s GetBlue Application it will allow for data to be transferred from the portable medium to the system via HTTP.(or Bluetooth for local use).

TEC-IT also provide a desktop application called TWedge which allows the data which is received from the scanned to be inserted to a Microsoft Excel Spreadsheet. Ref. [45]

From here the data can be read using a PHP file and manipulated accordingly.

This method was chosen initially due to the fact that I have had personal experience using it before and was satisfied with the result.

## Other Research Performed

• Interview with Seamus Kilmartin To be concluded.

* Interview with Charles Gagnon Alexandre

## Resultant Findings/Requirements according to Research

So after extended research on these technologies and weighing up some of the pros and cons of some of the technologies as well as considering personal preference the following technologies have been chosen to develop this system and website:

|  |  |
| --- | --- |
| **Technology Type** | **Technology Name** |
| Browser : | Google Chrome |
| Backend Programming Language: | PHP |
| Frontend Programming Language: | HTML, JavaScript, CSS |
| Development Tool: | Sublime Text |
| Labelling System: | Barcoding/QR Coding |
| Database Technology: | MySQL |
| Third Party Payment: | Stripe |
| Scanning Applications: | Android Device/Get Blue/TWedge |
| Deployment Method: | Microsoft Azure Virtual Machine |
| Contact Method: | SMS Messaging |

*Table 4 – Technology Conclusions*

#### Hardware Requirements

To provide a sufficient proof of concept it has been decided to use a hand-held barcode scanner as part of the system. This will be used by the “employee” to scan the barcode in order to read in the relevant data to be handled or manipulated with an accurate result.

A fully functional laptop will also be required in order to run the system and website as the client side.

The file structure and system content will be stored on a server. Therefore access to a server will be required to deploy the website files and folders.

To demonstrate the effectiveness of the QR Code in the project as a substitution of a barcode should the user wish, an Android Device with a QR Reading application will be used such as Androids QR *Droid Code Scanner*. Ref. [26]

# 3. Design

## Introduction to Design

The following chapter will outline all of the planning and design aspects of the project in detail. Further details are provided regarding the methodologies used throughout the project development as well as the system architecture of the proposed project. Following that, both the structural and visual design elements are outlined before looking at design documents including use-cases, ERD’s etc.

Timetables and Gantt charts are also provided in this chapter to outline the time allocated to the development phases of the project.

## Approach and Methodology

### Approach

For this project, the most important part of development was making sure the structure of the system and website was sustainable. MVC was the chosen structure for the system, as it is very simple to implement, control, expand and develop. Ref. [27]

Getting started on the project, firstly a pen-written design was drawn out of the folder structure, database tables and connection, then followed by some simple functionality to get the website up and running. This process included listing all templates, html forms, constraints for each form, and a breakdown of the accessibility of each type of user.

After that, rather than focusing on all design at once or all development at once for the whole project throughout, a step-by-step process was established. By doing this it makes it possible to design, implement and test each individual aspect/function one at a time (similar to TDD (Test Driven Development) Ref. [46] ). This also gave a possible estimate of output based on some assumed inputs.

The View was to be populated with specific data and templates based on the type of user logged on. By doing this a simple User Interface was designed allowing for sufficient testing of basic data and functionality. Since the idea of this project is generic barcoding, it was clear that the sooner development of a proof of concept was developed, the easier the progress of the system would become.

The approach has been followed since the beginning of the project, and progression has been evident throughout.

### Design Methodology

#### • *Agile Methodology*

The Agile Methodology is typically used in software development, but can be used in multiple fields of study and development. It allows for quick response to unpredictability by using incremental, iterative work cadences which are known as *sprints.* Ref.[28]

*Agile Methodology* is an effective alternative to sequential development and waterfall model method.

Essentially, the *Agile Methodology* has been applied throughout the project thus so far. Firstly, a proposed idea/function is designed and given sample data on paper to estimate a possible output. Should this not prove accurate during implementation, then a new approach and design to the same function is drawn up and tested.

So far, this has been used effectively and giving back accurate results mainly regarding database inputs and outputs.

Agile Methodology provides flexible, accurate testing on a regular basis throughout development in each of the Model, View and the Controller of the MVC framework. Although the MVC structure does provide some stability when it comes to the development of the system including each individual function, when applying the Agile Methodology it allows separate design, implementation and testing when needed during development.

All aspects of development will be implemented and tested using the Agile Methodology, such as:

* The Database(Tables, Constraints, Relationships)
* MVC Structure(Layout, Folder Structure, Factories, Connections)
* Registrations and Sample Data Handling within MVC and Database(CRUD operations, error handling, validation methods)
* QR Code Implementation
* Individual HTML Templates/Forms/visuals.

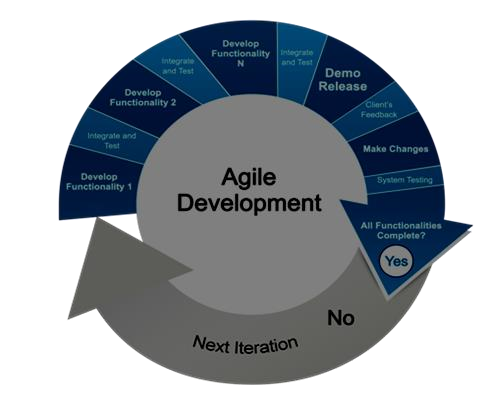


Figure 14 - Agile Development

In the next section we will outline the system architecture used for the project being deployed.

## System Architecture

### This system will consist of a *Three-Tiered Architecture* which will be made up of a *front-end user interface, a data modelling system and a data storage.* Ref. [29]

For the frontend a View will be used to populate the content of a template, which too will have an MVC structure made up of frontend languages such as HTML, JavaScript and JQuery.

As for the data-modelling of the project an MVC structure will be design and implemented using PHP and MySQL queries. Data will be controlled here and passed to and from the database. This file structure will be saved in a server such as Microsoft Azure Ref. [42] .

The data will be then be stored in a MySQL database server.

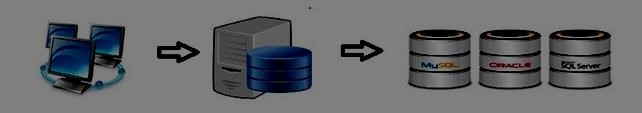


Figure 15 - Technical Architecture Diagram

Presentation Tier 🡪 Application Tier 🡪 Data Tier

## Design of User Interface

For the UI (User Interface) Design it was essential to keep it ‘clean and simple’. By using the utility provided by *Palleton.com* Ref. [44]I was able to find a suitable colour scheme for the interface. Paletton provides accurate well-looking colour contrasts and shades. Other tools and API’s were used in order to provide essential graphical elements such as fonts, structure, colour, and graphics. These tools include:

* Twitter Bootstrap
* CSS
* Other Glyphicons provided by Jan Kovarik Ref. [40]

Testing multiple structures and UI elements gave an insight as to how the website should look. Bootstraps *‘col’* attribute allowed for specific columns and margins to be defined where needed. Each operation in which the user interact with the system is to be centred on the screen. For example:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| col-lg-12 | | | | | | | | | | | |
| col-lg-6 | | | | | | col-lg-6 | | | | | |
| col-lg-3 | | | col-lg-9 | | | | | | | | |
| col-lg-4 | | | | col-lg-4 | | | | col-lg-4 | | | |
| col-lg-3 | | | col-lg-3 | | | col-lg-3 | | | col-lg-3 | | |
| col-lg-2 | | col-lg-8 | | | | | | | | col-lg-2 | |
| col-lg-2 | | col-lg-2 | | col-lg-2 | | col-lg-2 | | col-lg-2 | | col-lg-2 | |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Figure - Breakdown of col for bootstrap templating

By centralising all visual templates and forms on the screen contributes to the ease of access for the user.

Paper diagrams were first drawn up of various possibilities for the UI structure. After some consideration and the webpage structure chosen was designed as follows.

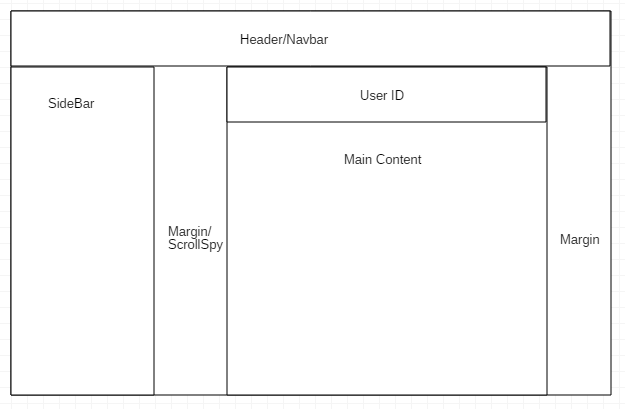


Figure - Chosen UI Structure

The following candidates were also strong possibilities:

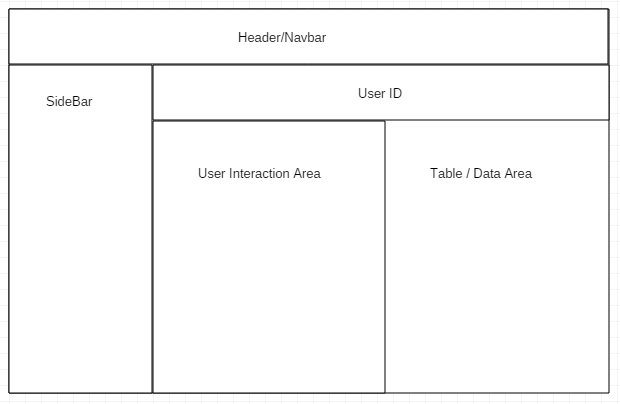


Figure - Possible UI Structure (1)

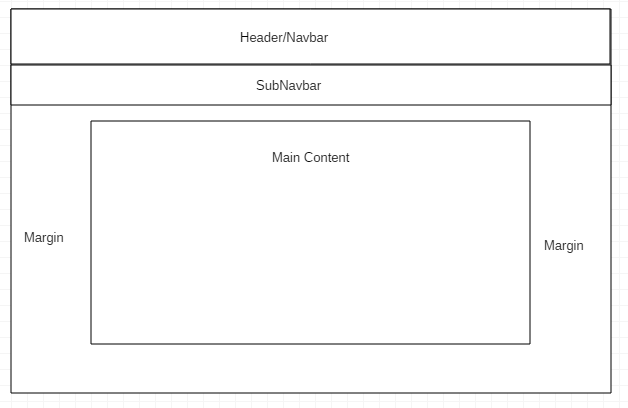


Figure - Possible UI Structure (2)

I decided to add a “side-menu” to provide hidden options for each user type. Although each user type is been given its own html template, the structure of the templates are prominently the same.

Contrast in the UI between each user type:

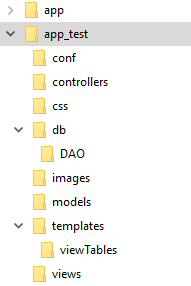
|  |  |  |  |
| --- | --- | --- | --- |
| **Element** | **Customer** | **Employee** | **Admin** |
| **Side Menu** | * Top Up Car Ticket * View My Events * Track Sent Stamps * Get Expiry Time of Current Parking Ticket | * View Issues * Search Customers * Report Issue | * View Customer Issues * View Employee Issues * View All Customers * View All Employees * View Admins * View All QR ID’s * View All Events/Invites * View All Stamps * View All Parking Tickets |
| **Color** | Dark Grey,Light Grey, Grey Fade, Blue, Amber | Dark Grey,Light Grey, Grey Fade, Blue, Amber | Dark Grey,Light Grey, Grey Fade, Blue, Amber |
| **ScrollSpy** | 3 Tabs | N/A | N/A |
| **Navigation Bar** | * Profile(Drop Down) * MyStripe * Report Issue * Change Password | * Profile(Drop Down) * Load Scanned Data * View Scanned Data * Change Pin | * Profile(Drop Down) * Change CPark Rate * Add Employee * Change Pin |
| **Table Views** | Read/ Delete | Read/ Delete | CRUD \* |
| **Header Bar** | QR Image / Title | N/A | N/A |

The next section outlines structural development for the System.

## Design of System Structure

While considering the structure of the system various elements were taken into thought such as the MVC framework, QR implementation, use of API’s, folder structure etc.

The following was the initial folder structure of the proposed system.



Although this is the starting point for the folder structure, later in this report the changes to this structure are outlined and described in detail.

The following figure describes the relationship of the components of the core MVC, as well as one which includes attached elements.

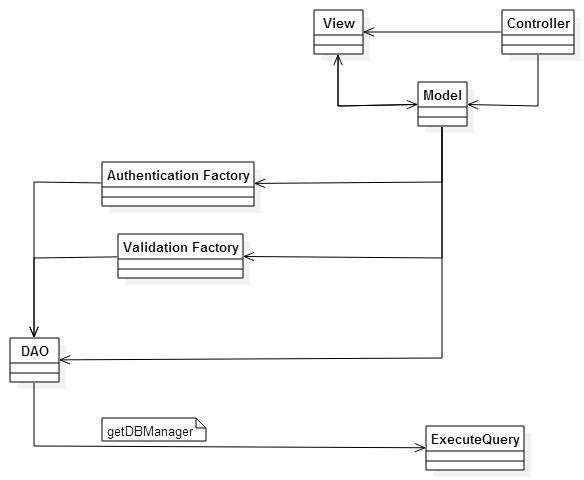


Figure 20- Standard MVC w/Factories

Figure 12. – MVC General Structure

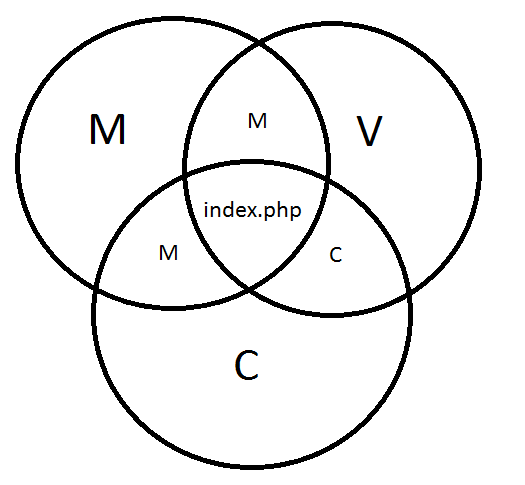


Figure 21- MVC Venn diagram

Here is a short breakdown of the content of each folder:

|  |  |
| --- | --- |
| **Folder** | **Description/Contents/Examples** |
| Conf | config.inc.php : contains all the constant variables used throughout the system |
| Db | Will hold the database connection files and the DAO’s for user interaction with the database. |
| Phpqrcode | Will hold all the code necessary to generate the relevant QR code for the user. |
| Models | Will holder the model class, validation and authentication factories classes. |
| Controllers | Will hold the controller class and the cronController to ensure scheduled executions are carried out appropriately. |
| Views | Will hold the view.php which will be populated by the HTML files and templates in the *templates* folder. |
| Templates | Will hold all HTML forms, user templates and views which are used to populate the view.php based on user interaction with the system. |

## Design of Source Code

In terms of the structure of the coding elements most of it is structured based on the MVC framework.

Rather than commenting on each line of code, a comment box will be provided at the top of each coding file. Every function will be described by a small comment box located above the method definition.

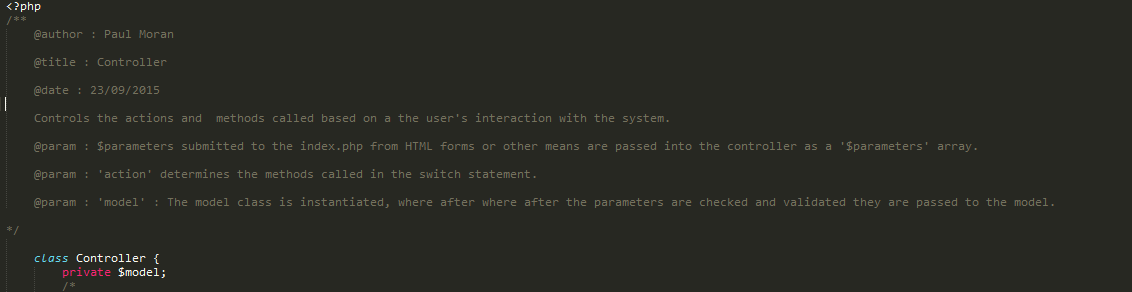


Figure 22 - Top Comment Box

## Project Features

#### SMS Messaging

For the SMS Messaging element of the system, the Clickatell API proved rather effective. This API is simply called from a separate PHP file which executes a cURL Ref. [41] command passing in the appropriate parameters for the API. Unfortunately for this project, I am limited to the amount of texts I can send. The content of the message is also unable to be changed due to the account being used being a test account.

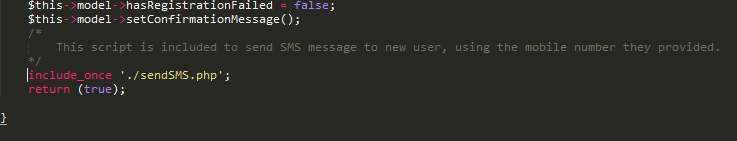


Figure - Send SMS calling - Design

#### Automatic Emailing

As part of the system, the event creation aspect was to simply create an event which would generate individual QR Codes for each invite, and email this QR to the invitee email address.

In order to achieve this, PHP’s mailing API was considered as the simplest and most effective solution.

#### QR Code Generation

For the QR Code generation an external 2D-QR generator was used. This generator is a free-to-use software and licensed to Dominik Dzienia Ref. [36] . It provides accurate data readings for multiple lengths of data as well as multiple input characters.

The idea of creating a new QR generator would’ve been rather extensively time consuming. Therefore, I manipulated this code and library to suit this projects requirements.

Again, in order to perform the QR generation a cURL command is called from within an external PHP file which is triggered where needed.

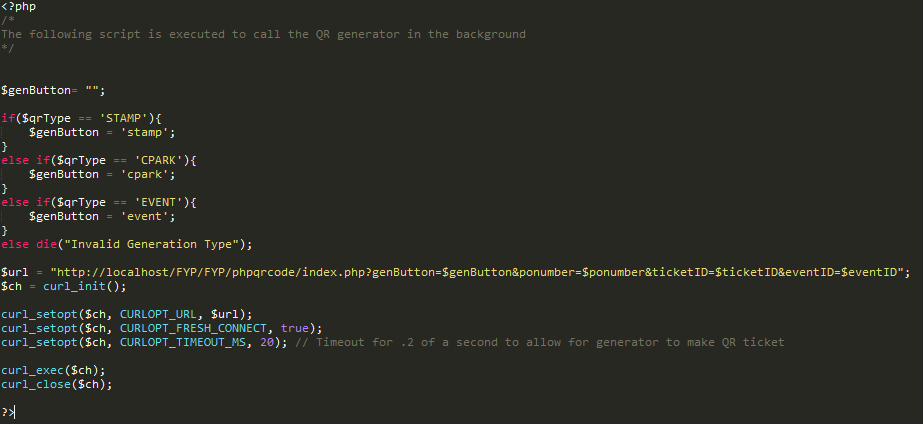


Figure - QR Gen cURL -Design

#### Stripe Payments

The Stripe API will be used to carry out dummy payments made by the users of the system. If the API is not registered with a particular users account on the system, then that user will not have the permissions to generate QR codes which require payments.

The API will be called in the background using a cURL command which will be populate with the valid parameters based on the user’s interaction with the system. For example:

*If the user chooses to generate a postal stamp, they will have to provide the necessary information about their postal package: destination, approximate weight and postal type. Once each of these parameters are provided then the valid amount will be deducted from the user’s Stripe account when the QR code is scanned for the first time.*

|  |  |  |
| --- | --- | --- |
| Money Transactions in System | | |
| Ticket Type | When | Amount During Transaction |
| STAMP | Payment occurs when a valid ticket is scanned for the first time provided the parameters provided are correct. | The amount to be deducted is based on the destination, weight and package type. |
| PARKING TICKET | Payment occurs if an initial expiry time is provided during generation. If not, the payment is not made until the ticket is topped up. | The amount is based on the duration between the time of payment and the expiry time as well as a standard hourly rate being applied. The user also has the option to top the ticket by a choice of duration or specific amount. |

#### Scanning QR Codes

In order for the system to take action with any of the data given from the QR code they must first be read into the system after generation by a customer. In order for accurate readings of the data, a suitable Android application can be used to read the data onto a particular device.

Next step is to feed that data into a readable file in the system. From here a PHP script can be used to perform relevant validity checks on that data and then push it to the database.

<IMG – DIAGRAM SHOWING THE DATA ROUTE BEING PASSED FROM ENTITY TO ENTITY>

## Other Design Documents

#### Use Case Diagram

 Users: User, Admin, Employee, Third Party

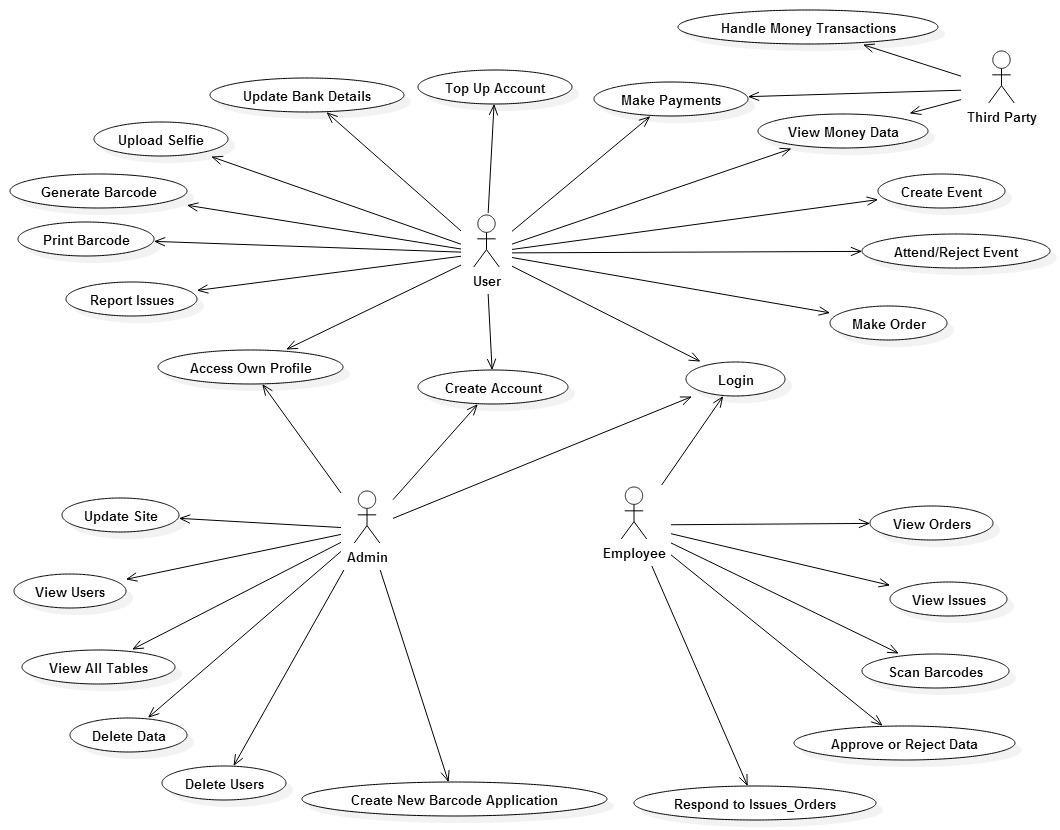


Figure 25 - Use Case Diagram

#### Gantt Charts

* **Gantt Chart (Unexpanded)**

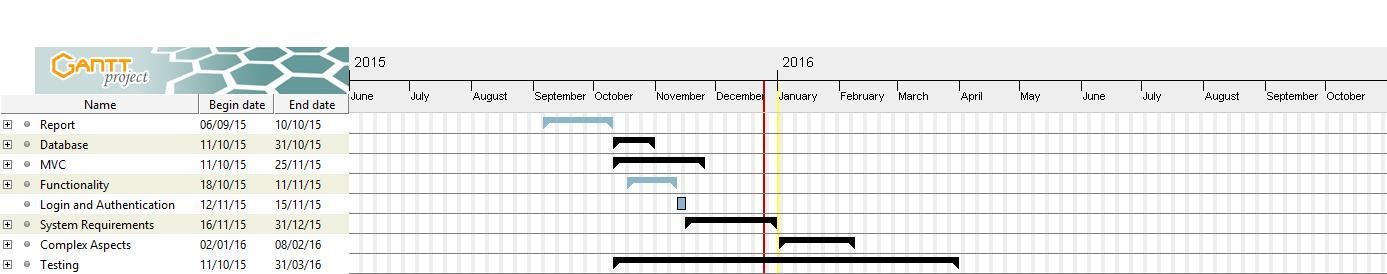


Figure 26 - Condensed Gantt Chart

* **Gantt Chart (Expanded)**

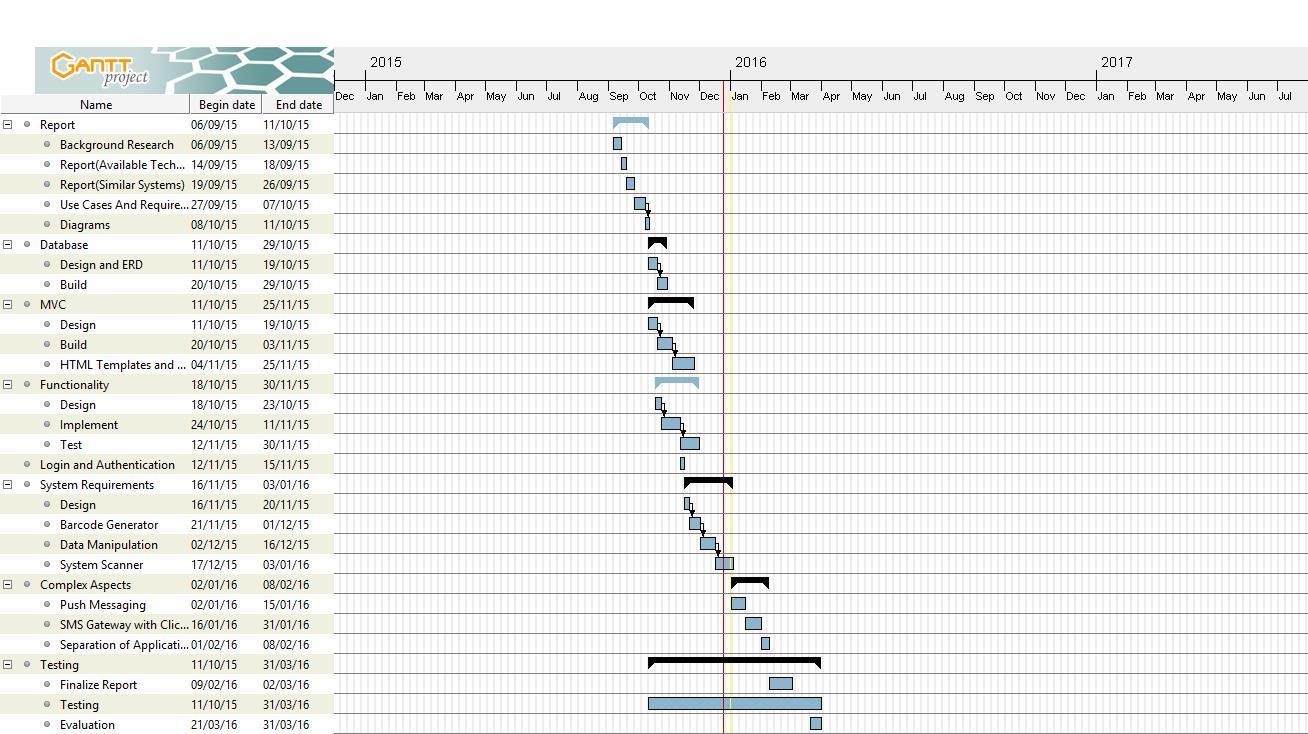
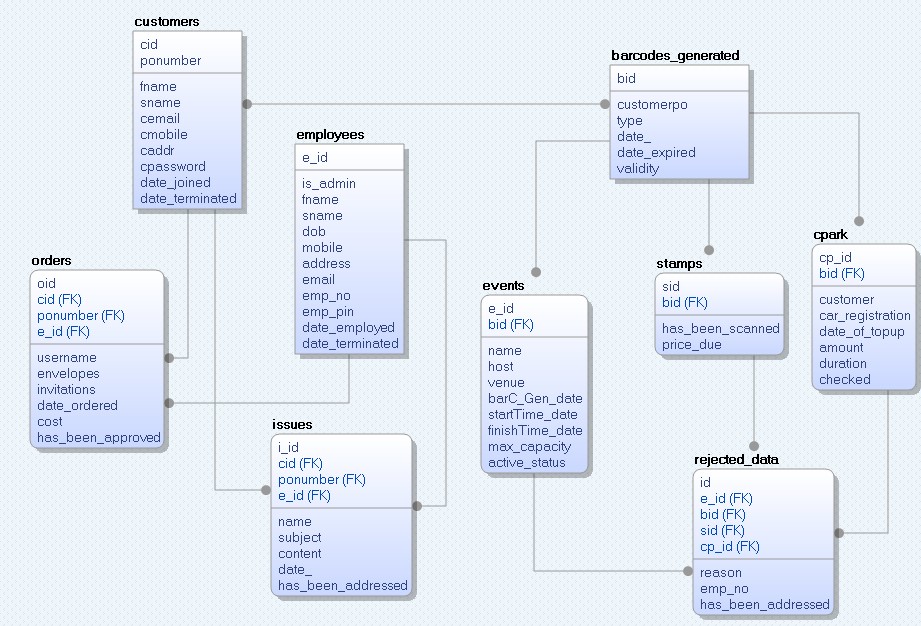


Figure 27 - Expanded Gantt Chart(w/o Report Development)

#### ERD Diagram



*Figure 13 - ERD Diagram*

# 4. Development and Implementation

## Introduction of Implementation

In this chapter the full implementation of the system is described. Majority of the implementation is based on the evaluation provided from the design chapter. The components which are outlined throughout this chapter are as follows:

* Database
* The MVC Framework
* HTML Templating
* Further Development of CRUD operations
* QR Coding Features
* API use and External Software

## Development of Database

The development of the Database was predominantly based on the ERD described in the Design chapter. In this section we will be outlining the step-by-step development of the database, and its relationships.

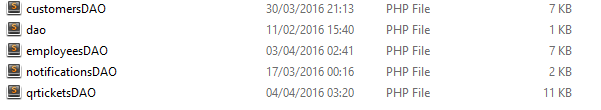


Figure 28 - DAO Folder Contents

*Figure 14 - Developed DAO's*

Designing and implementing the database was the first key priority. It was majorly important that data used at any given time, was stored correctly, using the appropriate software, hash functions, data types and structure. In order to test the each column and table, sample data (both correct and incorrect ie. Using null in a non-null column) was inserted.

1. Firstly, all the database tables were designed and developed that were needed in order to complete the structure of the database as well as listing them under the appropriate user-type (i.e., General User, Employee, Issues, etc.).
2. Access to each table was based on the SQL Queries and templates provided to each user type. For example.

*A customer would not be given the option to view other customer’s details within their own template.*

1. Next, all indexes for each table were added including Primary Keys, Foreign Keys, and Unique Constraints etc. Following this step it was then important to create the appropriate relationships for each table.
2. Now that each table was almost fully designed in my database, sample data was tested using some manual CRUD operations using the localhost/PHPmyAdmin server. If a different database was to be used at a later stage. The current build script of this database can be “’*exported*’ for use on a different database.
3. Finally activity and validity columns were added and tested. The purpose of these columns is to keep track of various validations factors in particular generated data. This would include columns to ensure that a customer has paid for a particular service, or making sure that a ticket is still within its validation date before being accepted into the system.

For example:

*Should a customer create a new parking ticket, this would in turn make the old one redundant due to the concept that the car can only be parked in the one spot at any given time. This in turn prevents others from using a lost ticket if that ticket was still within its validation date*.

Following the creation of the database, the next essential step for the database was to write out a list of the queries in which would be each of the DAO’s. In order for each of these queries to be executed to affect the database, a PHP file (simple\_db\_manager.php) was created which contained the methods needed to connect to the database and execute the queries. See Figure 15.

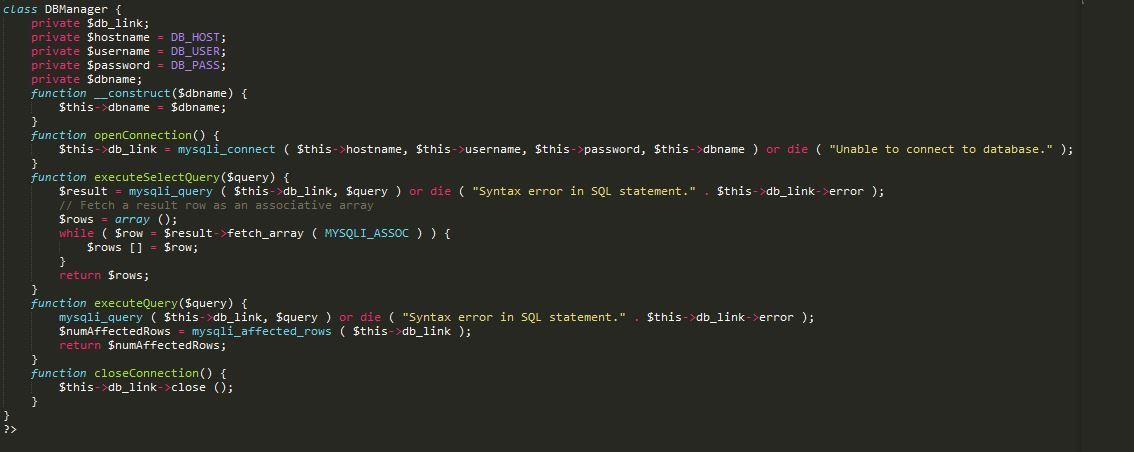


Figure 29 - DBManager Class

Essentially this DBManager is the access point to the database.

All database related files are contained within the FYP/db folder. This folder is structured as designed in the Design chapter, see Figure []. The main DAO\_Factory class was then created to allow other classes to access the individual DAO’s methods and the DBManager as seen in Figure [].

Now that the methods were created to return each of the DAO’s, the next step was to develop each of the DAO’s themselves.

* BaseDAO

The BaseDAO simply extend its uses to each of the other DAO’s which in this case is the use of the DBManager.

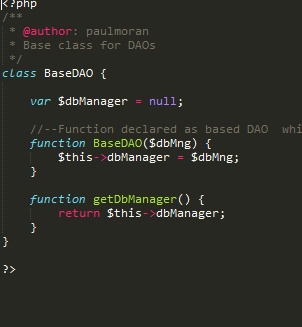


Figure - baseDAO code

* CustomersDAO

The CustomersDAO contains all of the queries which manipulate the customers table in the database. Below is the list of all queries in which affect the customers table in the database.

* EmployeesDAO

The EmployeesDAO is more or less the same as the CustomersDAO except we are dealing with the employees table in the database. This DAO relates to both employees and admins of the system.

* NotificationsDAO

The NotificationsDAO handles all queries regarding issues from both employees and customers. Issues are used as a simple non-instant messaging service between the customer, employee and admin of the system.

* QrticketsDAO

The QrticketsDAO is the largest DAO, which is connected to the qrcodes, parkingtickets, events, stamps, parking\_price, invites tables. This DAO handles all queries which are related to the generation, scanning or manipulation of the QR data.

In the next section the implementation of the Model, View and Controller is detailed. It will describe how the model as well as other classes interact with the methods defined in each of the DAO’s.

## Development of MVC – Model, View, Controller

Following the design of the MVC structure, the Model, View and Controller were implemented in accordance with the database in place. As part of the MVC structure, the necessary factories were implemented in order to get the index, controller, model, database connections and view to work together effectively. This also included connections between the Model and the factories to the necessary DAO’s.

Like every website, an *index* file was created which was to link the *[model to controller]* and the *[model and controller to the view].* This was to handle any user input to the system using a PHP *$\_REQUEST*. See Figure [23].

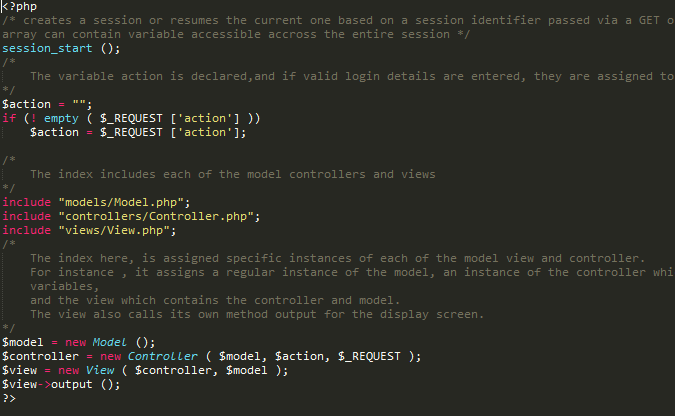


Figure 31 - index.php

Next was the file and folder structure was designed and created for MVC framework, starting with the *model, view and controller* folders. These were followed by the *authentication factories, validation factories* within the model folder*.* Once some of the functions in the DAO's were tested with some sample data it was important to move ahead with initialising the model, view and controller files.

#### MODEL

The model file handles any data transactions, methods or functions which are called in the controller. The model is also instantiated with a copy of the necessary data checking factories (*authorisation, validation*).

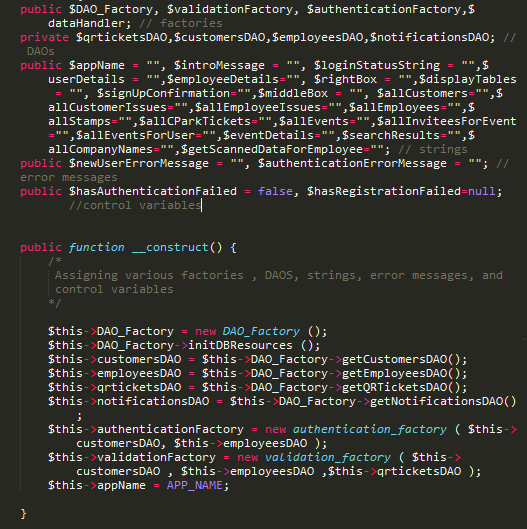


Figure 32 - Model Construct

#### VIEW

The view is simply a control to determine what is being output on the screen for the user to see. The view is used to determine what user is logged on. Therefore determines which template to be applied for the user to ‘view’. The view is instantiated with a copy of both the model and the controller. This is so values may be taken from the model or factories to be displayed for the user such as a *username* or *numeric value* from the database. Whereas the controller is also instantiated in order for specific set values to be displayed such as the confirmation and errors messages.

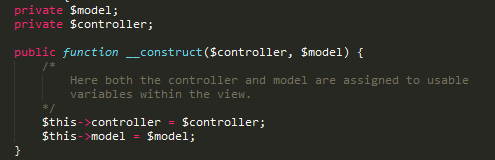


Figure 33 - View Construct

#### CONTROLLER

The controller receives all user input from the *index.php* file. This is instantiated with a copy of the model in order to pass various values to the model and then receive results such as Boolean values from the model to set certain variables such as confirmation/error messages.

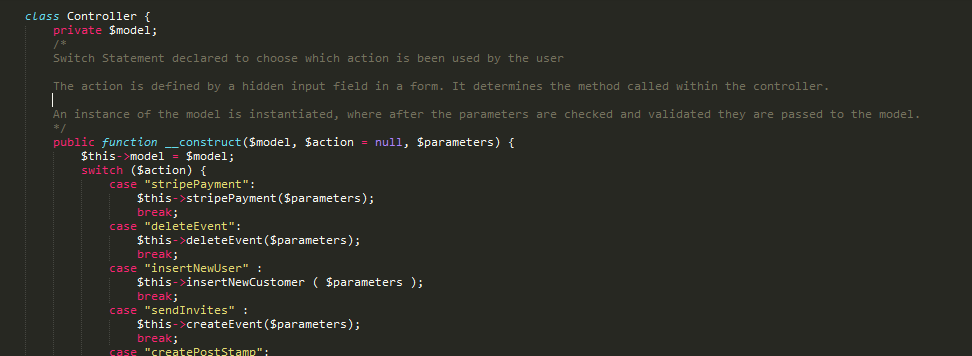


Figure 34 - Controller Construct w/Switch

#### OTHER NECESSARY FOLDERS

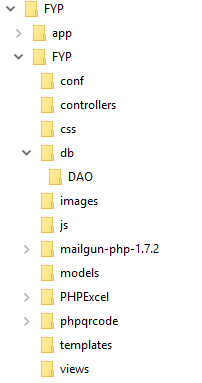
On creating the MVC files the configuration (*conf*) folder was then created which contained the *config.inc.php* file to hold the constant values throughout the website such as the database connection variables and error messages.

An *images* folder which holds all external images and graphics used.

A *js* folder was created to hold some of the *JavaScript (js)* files which are used in the client side views.

Then the Cascading Style Sheet (*css*) folder to hold all the styling code as well as some of the bootstrap variables.

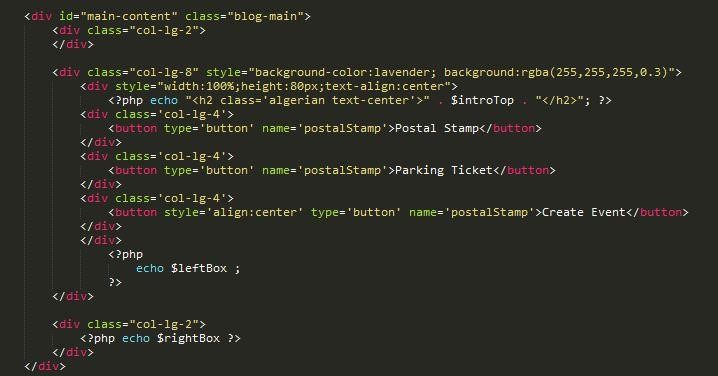
Now that the folder structure has been implemented, the next step was to begin populating them with appropriate coding files such as PHP, JavaScript, CSS, HTML files.



Next we will have a look at the development of them html templates and forms. These are primarily used by the View to populate part of a given HTML template. (NOTE : HTML templates which contain tables populated by values from the database are called specifically in the HTML template itself.)

## Development of HTML Templates

One of the folders which was added at a later stage was the *templates* folder. This held all the user forms, index-structures for each type of user and html tables which were used to display the data taken from the database tables.



*Figure 20 - User Template Main Content*

1. Firstly the template index for each user (*4:customer, employee, admin, login-page*).
   1. *Template\_index.php*
   2. *Template\_index\_employee.php*
   3. *Template\_index\_admin.php*
   4. *Template\_index\_login.php*
2. The appropriate forms for each user type were listed by hand See Figure[]. After being listed the next step was to build each of them, and provide sample data to test each submission. A list of all forms and templates is listed below. See Figure[]
3. Next was to list all required data tables to be displayed for each user. This was an important step as each user should only have access to particular data, so knowing what data was being displayed on each table, for each user is important.
4. Now that all the templates, forms and tables have been created it is now possible to include each of them in the view were required. The structure of the webpage is as follows:

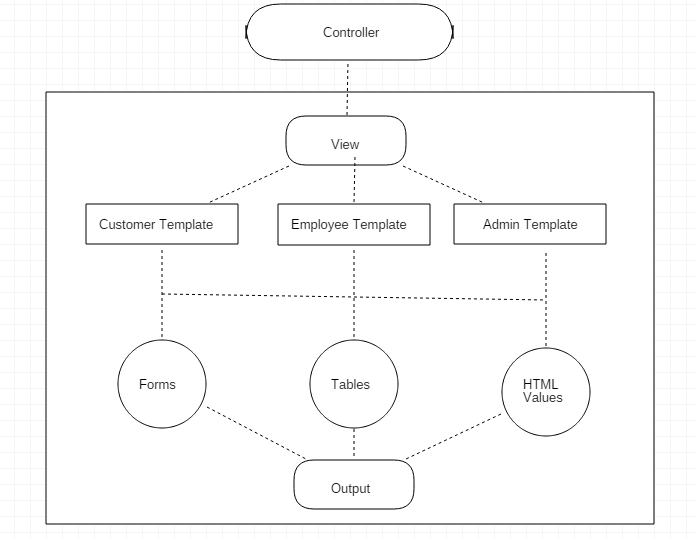


Figure – View, Templates, Output Breakdown

## User Interface

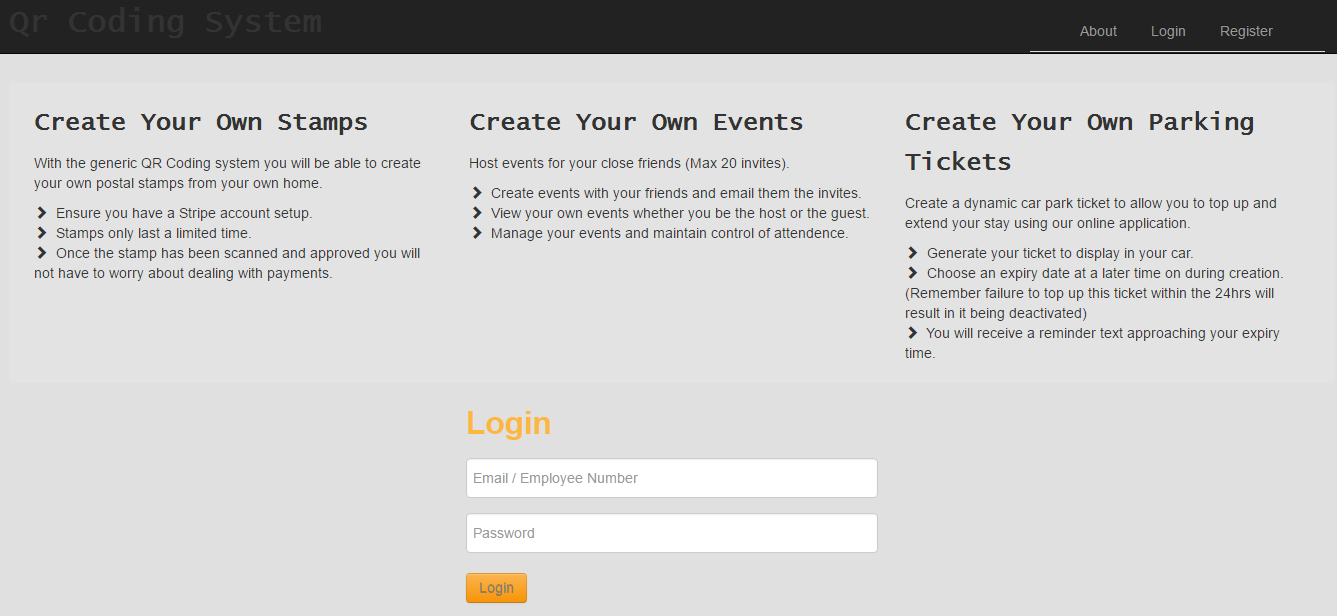


Figure 36 - Login Screen

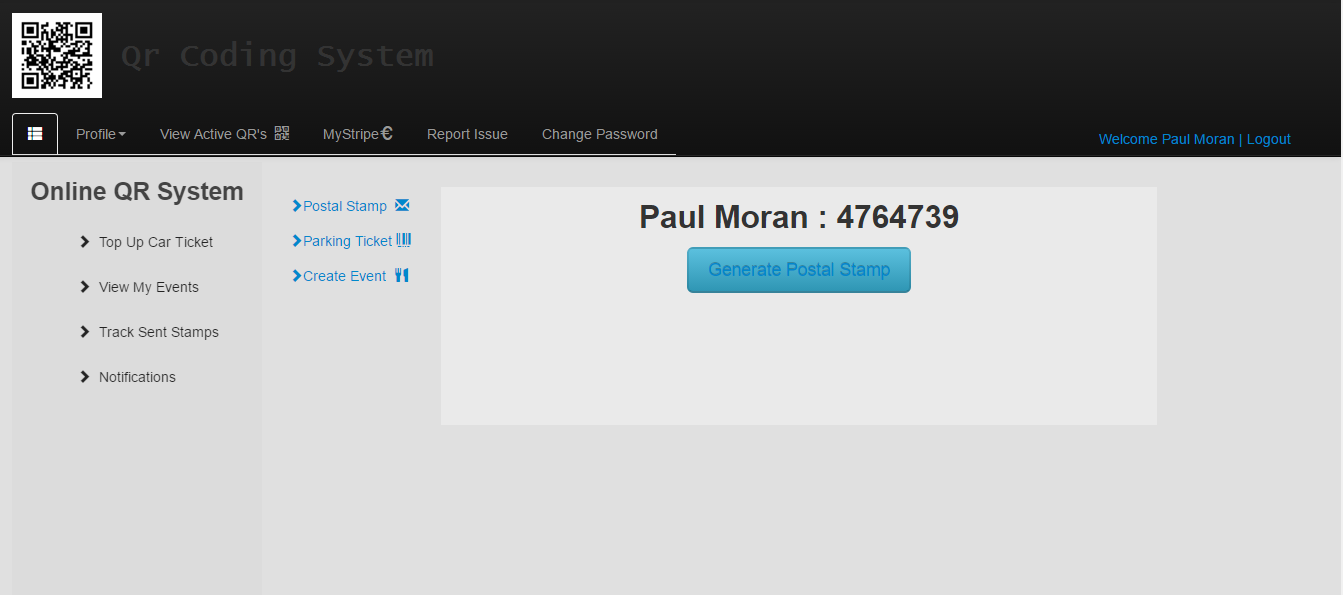


Figure 37 - Customer Template

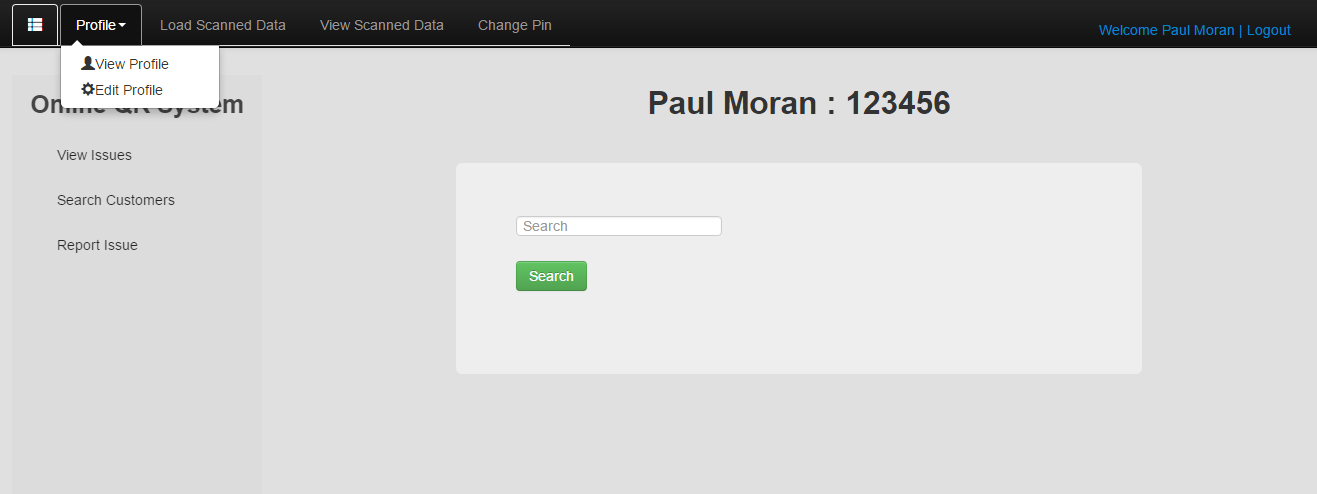


Figure 38 - Employee Template

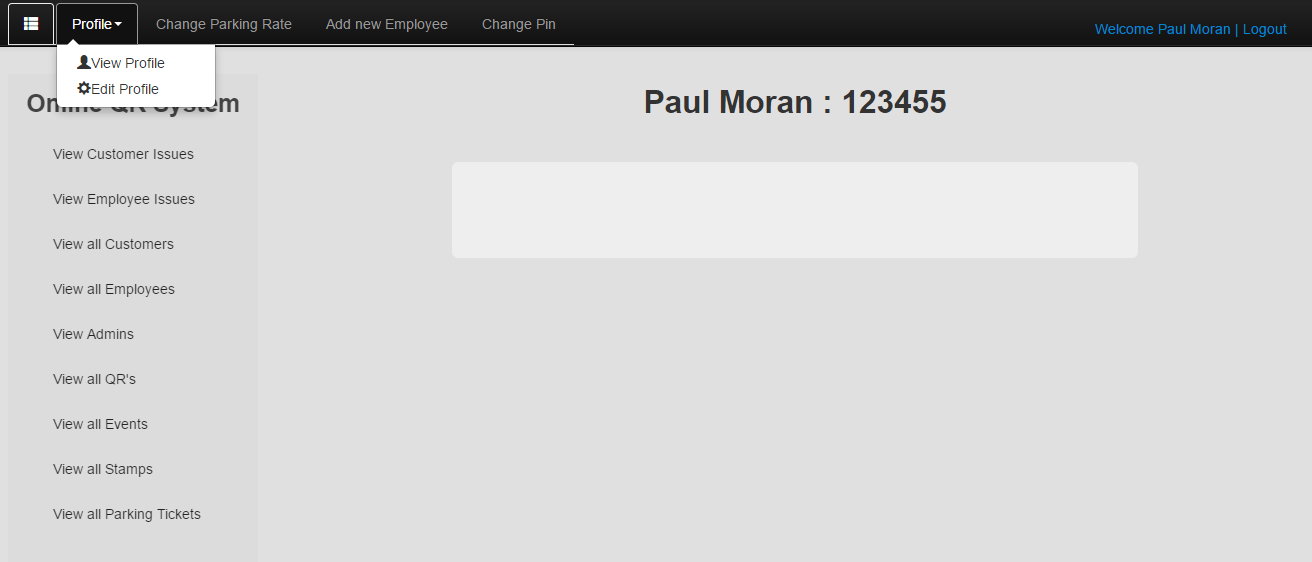


Figure 39 - Admin Template

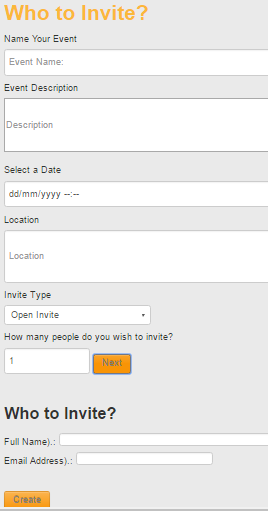
Each field in the form is check for valid input using both HTML on the frontend and PHP data validation in the controller.

Figure - Sample form (Event Creation)

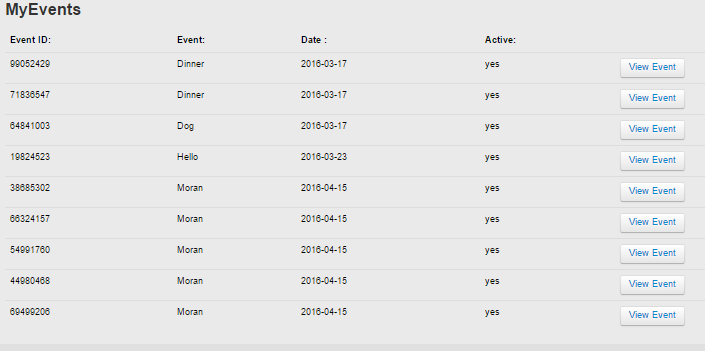


Figure 41 - Sample Table (Customers Events)

Each of these templates are shown in the view based on a particular user’s interaction with the system.

## Development of Functions and Factories

The functions and methods proved to be one of the most time consuming and most difficult aspects of creating the project. By beginning with the functions called in the *controller,* it was possible to quickly map each of them to the appropriate functions in the model.

To start with building the functions in the controller, the first aspect to be create was the *switch statement* which in this case was to check the value of the *action* parameter passed in from the *index.php* as described in Section 2 of Development and Implementation. This was then followed by the appropriate methods being created in the same script which were called after a case in the switch statement had been triggered.

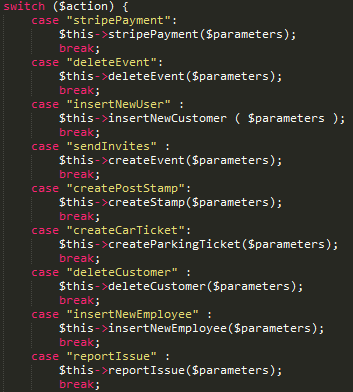


Figure 42 - Switch Snippet

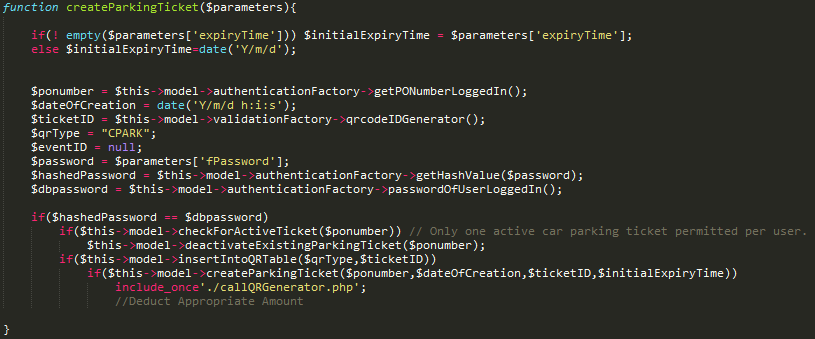


Figure 43 - Method Called By Switch in Controller

On triggering a statement within the switch, it would call the required function as mentioned above. From here this function would then call the opposing method inside the model, whilst passing any necessary parameters.

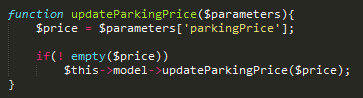


Figure - Controller Method (Update Parking Price)



Figure - Model Method (Update Parking Price)

Once the pattern is formed of the connections for each function within the controller matching the ones the model/factories, it was then easier to map data movements throughout the system. The controller requires the methods in the model in order to access the database and the factories. This is because the only entities which have access to the DAO’s are the model and factories as seen in Figure[].

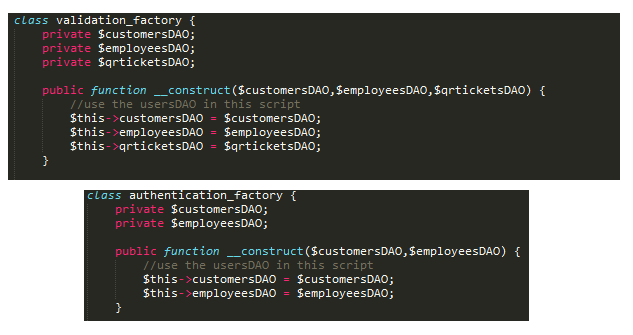


Figure 46 - Authentication/Validation Construct

After creating the necessary functions within the controller, and model, as well as some in the factories which are described later in this chapter. It was then decided to begin mapping the methods from the model/factory to the appropriate DAO. See this sample scenario to understand the passing of the data between each entity

Scenario:

*If the user wished register, they would be required to enter in their necessary details in the HTML form. These values are then submitted to the index.php, which redirects them to the controller, and based on the ‘action’ passed from the HTML form, a method is called in the controller. After the method is called in the controller, the values are checked using the factories See Figure[] . If all of the values be acceptable they are then passed to method in the model, which calls the opposing function in the correct DAO and echoes the parameters to that method. From here the SQL statement is executed using the entered parameters.*

The DAO would handle the SQL statement required to execute the insert statement passing in the parameters containing the new user's details.

5. It is evident that a simple task is quite drawn out, but yet rather effectively. This was the process for almost all tasks and user interactions. Anything that may needed to be changed or laid out slightly differently was done so as required. Although this is the main structure and process used throughout my MVC website.

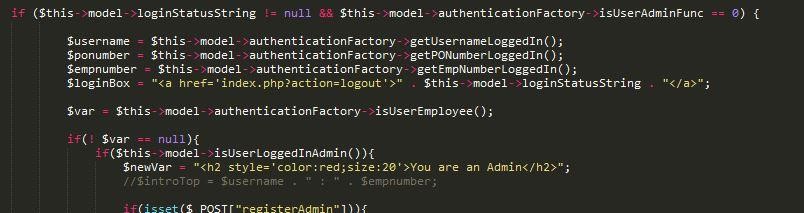


Figure 47 - Logged In User Check

## Implementation of QR Code Features

In this section the implementation of all QR features are portrayed. This also explains the use of the external QR generator code which is implemented from a third party developer. Although the generation of the QR codes is based on the work by Ref. [36] a significant amount of changes were made in order to satisfy the needs of this system. Some of these changes will be outline in this section. The main aspects of this section are as follows:

#### QR Code Generation

For the actual generation of the QR Codes themselves, a software provided by Dominik Dzienia Ref. [36] is used. The use of this software as oppose to developing my own generator proved rather beneficial in terms of the quality of the QR Code. After a number of attempts to replicate some aspects of the code it was established that by simply using the majority of the source code, would provide more accurate readings to my system. It also gave a good indication to the generation of QR Codes and how they stored data.

Since the code provided by Mr. Dzienia simply read in data as input and produced a PNG image as the QR Code, it was simple to manipulate the data which was passed to the required methods. By simply calling the *index.php* within a cURL script for the QR generator inside the *controller* following the confirmation that the user does in fact wish to proceed with the creation of the image, it was possible to use the necessary data provided from the controller to be passed to the generator. See a breakdown of this process is figure[44] and figure[45].

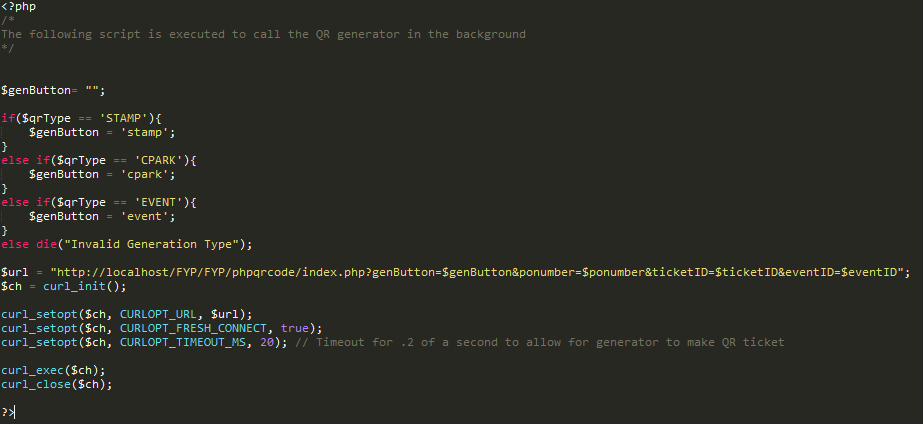


Figure - Call QR Gen - cURL

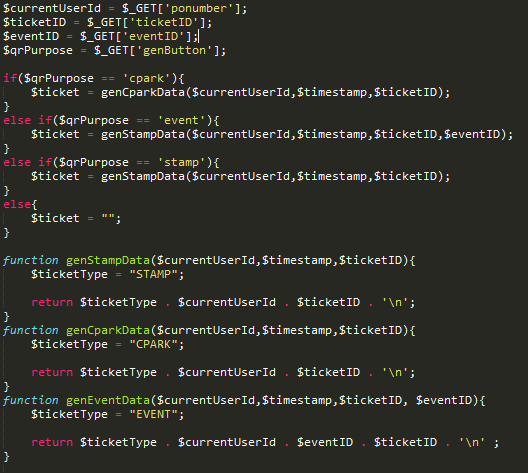


Figure - QR Generator Index (Data)

Each type of QR code receives its own distinguishable data. The breakdown of the data for each type of QR Code is as follows:

* PARKING TICKET – CPARK<PONUMBER><TICKETID>
* POSTAL STAMP - STAMP<PONUMBER><TICKETID>
* EVENT INVITE - EVENT<PONUMBERofHOST><EVENTID><TICKETID>

When scanned, each ticket will be differentiated by the system. It will recognise the purpose of the ticket, then later check for certain validation criteria which will be highlighted in the *QR Data Manipulation* section.

Following the generation of each ticket, it will be displayed within a HTML script called *displayGeneratedQR.php* which will provide a 5 second delay in order to allow the generation to complete. See FIGURE[].

In order to attain the appropriate QR code which has been generated, a script has been wrote which matched a particular sub-string within the name of the PNG file generated. This substring is in fact the ticket ID which has been incorporated into the QR code itself. The code for this script is as follows.

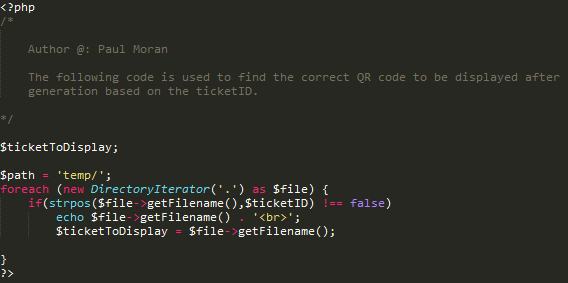


Figure 50 - Get Generated QR

Once generated the customer then has the option to download and save the PNG QR code to print.

#### QR Code Scanning

Although there are multiple applications available to read a barcode/QR Code, it was necessary to choose one which would scan the image and instantly upload the data to readable source on the system. Here is a quick recap on the applications that were used (see the Research chapter for further details about these applications):

* Barcode Scanner by ZXING

Android app to quick scan barcode data.

* GetBlue by TEC-IT

Receives data scanned by the Barcode scanner and pushes it to another entity using either, HTTP, TCP, Bluetooth, Textfile etc. (Bluetooth in our case). This data is pushed to the TWedge application on the desktop system.

* TWedge by TEC-IT

Receives each reading of the data forwarded by GetBlue. This data can then be added to an Excel Spreadsheet.

From the Excel Spreadsheet this data can then be pushed to the database using a PHP script.

###### Steps to Setting up the Scanner

* 1. Download *Barcode Scanner* by ZXING from the app store onto an Android device.
  2. Download GetBlue by TEC-IT onto the same Android device.
  3. Download TWedge by TEC-IT onto the Computer Desktop (Main) System.
  4. Adjust the Data-Source in the GetBlue App to ‘*Camera Scanner*’ and the Data-Sinks in the app to either ‘HTTP’ or ‘Bluetooth’ (Bluetooth for local use).
  5. Start the GetBlue Application (Camera Scanner should appear)
  6. Connect the Android Device to the Desktop System (By Bluetooth for local use, otherwise HTTP would be used)
  7. Start the TWedge application on the desktop system.
  8. Go to ‘Configuration’ on the TWedge application, and click ‘*Interface*’. Change from Serial port to Bluetooth, and then ‘Devices’->Click your Android Device.
  9. On Exit these changes can be saved.
  10. Next scan a barcode/QR code with the Android device and notice the entry being inserted to the TWedge interface.
  11. If all is well so far, open the Excel Spreadsheet from the main directory of the system.
  12. Highlight the first free cell in column 1.
  13. Scan the QR codes once again and notice that each data entry is read into the spreadsheet.
  14. Once the spreadsheet is saved the data can be read by the PHP script and inserted into the database.

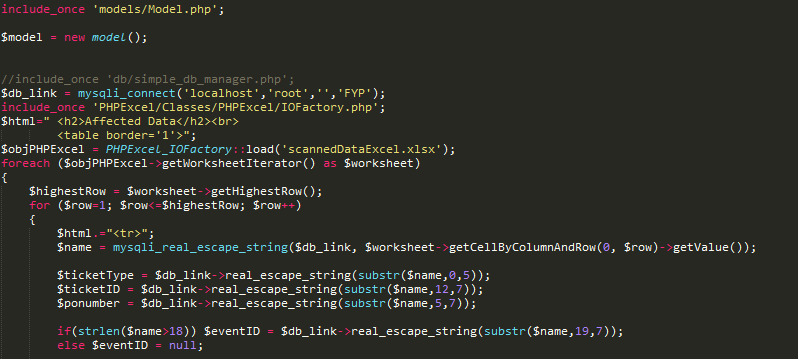


Figure 51 - Read Excel Spreadsheet w/Scanned Data

From the database this data can now be checked for validity for whatever purpose the data was generated. Another option would be to check validity prior to the insertion to the database.

Measurements have also been taken to ensure that the service in which the employee that is logged in is valid to scanned QR code. This prevents a worker in a post office scanning a QR code which represents a Parking Ticket.

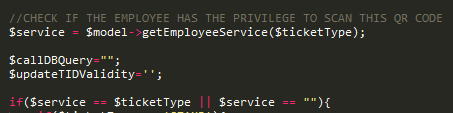


Figure 52 - Get Employee's Service (readExcel.php)

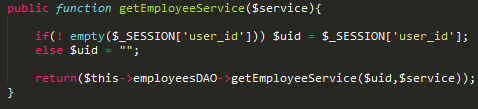


Figure 53 - Get Employee's Service from DAO (Model.php)

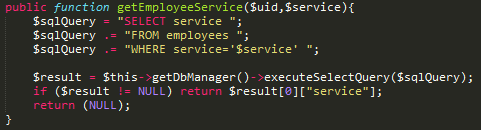


Figure 54 - Return SQL Query with Employee's Service (employeesDAO.php)

#### QR Code Data Manipulation

This part is the most important aspect in regard to the scanning of the data. It makes sure that the data scanned is usable and valid.

Regardless of whether the data is read prior to the database insertion, or from the database itself it needs to be passed through a variety of checks. In this section we will be looking at what checks each QR type will have to pass through in order to be deemed valid.

In order for each ticket to be read, the data for each entry is broke up individually as seen in the *QR Generation section.* If the first 5 characters read STAMP/CPARK/EVENT then the system will recognise this. Next is to identify the creator of these tickets by using the P.O. Number (Next 7 characters) to see if the character exists etc.

The next steps are individual for each ticket type.

###### Postal Stamp

* Check the date validity for the stamp. If the ticket is not scanned for the first time within 2 weeks of being made it will be rejected.
* Check that the destination matches with the zone which was selected during generation.
* Check the payment has been made following the scanning of the ticket for the first time.
* Update the data in the database, regarding that entry. The updated data will depend entirely on whether or not the stamp passed all validity checks.

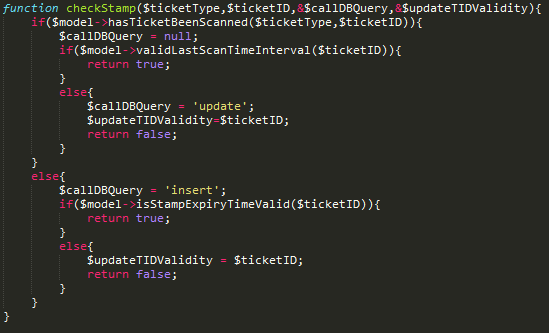


Figure 55 - Perform Stamp Check on Scanned Data

###### Parking Ticket

* Check that the expiry has not been reached.
* Check that a valid payment has been made should the time of generation have been greater than half an hour ago.
* Update the data in the database, regarding that ticket. The update data will depend entirely on the validity of the ticket which has been scanned.

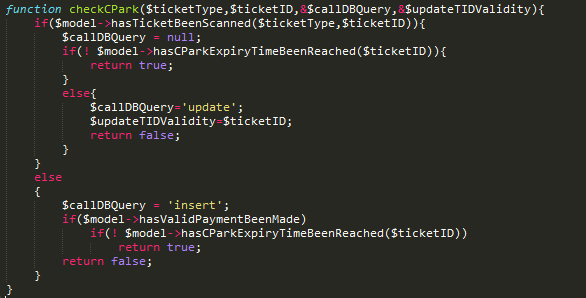


Figure 56 - Perform Parking Ticket Check on Scanned Data

###### Event Invite

* Check that the Event ID exists and is active.
* Check that the Ticket ID exists and is an invitation to that event.
* Check that the user’s ID matches the name related to the Ticket ID in the database.
* Update the data in the database. The updated data will depend on the validity of the invite ticket which was scanned.

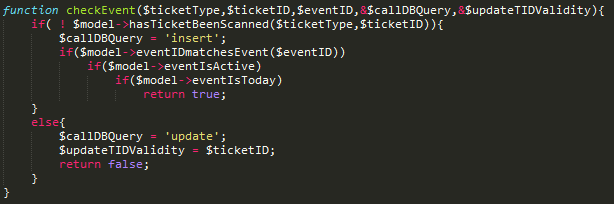


Figure 57 - Perform Event Invite Check on Scanned Data

Following the loading of the data to the database See Figure[], the *employee* user now has the option to view the data. Measurements have been taken to ensure that an employee can only view data which they have privileges to. For example, there is no need for a worker in a post office to view the details of an event in the system.

See FIGURE[] to see the scanned data of an postal system being displayed.

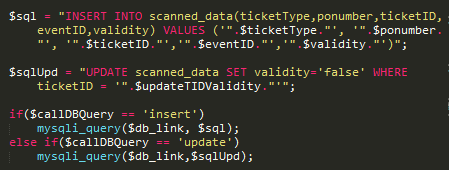


Figure - Insert/Update Scanned Data

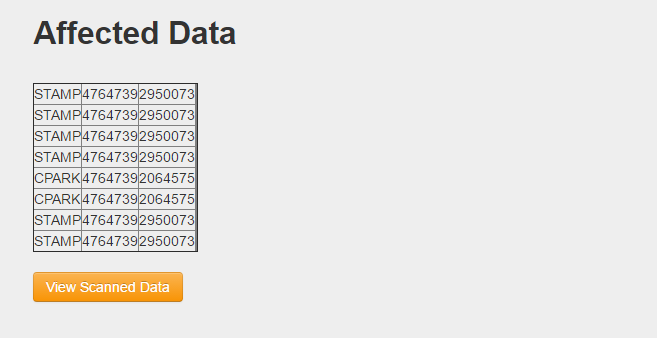


Figure 59 – Data Loaded to Database

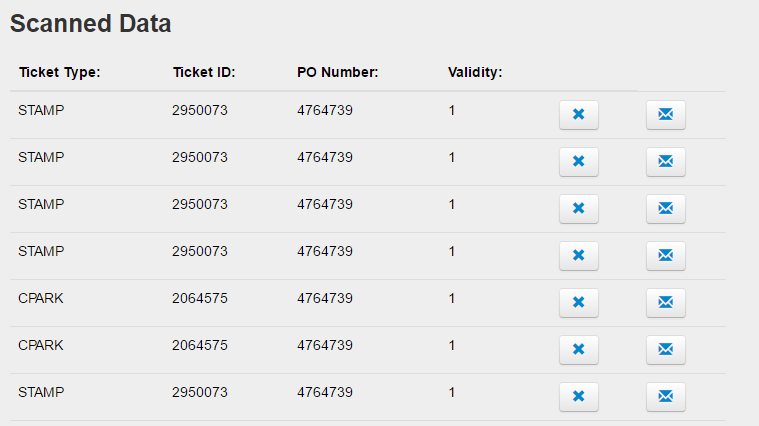


Figure 60 - Scanned Data Table(Employee w/ All Privileges)

## Use of Other API’s and Software

For this section the use and integration of other API’s and software is outlined. These include the use of Clickatell’s SMS Messaging and the PHP emailing gateway for localhost.

### Clickatell’s SMS API

Clickatell’s SMS service allows for SMS text messages to be sent to multiple phone numbers when needed. For this project the purpose of the text message service is as follows:

* + 1. To send SMS after successful registration of a customer.
    2. To send SMS to customer when parking ticket to close to expiration time.
    3. To send SMS to a customer two hours prior to an event start time.
    4. To notify a customer of a rejected stamp, to notify a customer of an invalid parking ticket.

The script send the SMS message is only invoked in the controller based on some user interaction or timing schedule of a particular action. The API to send the SMS is called using a cURL command.

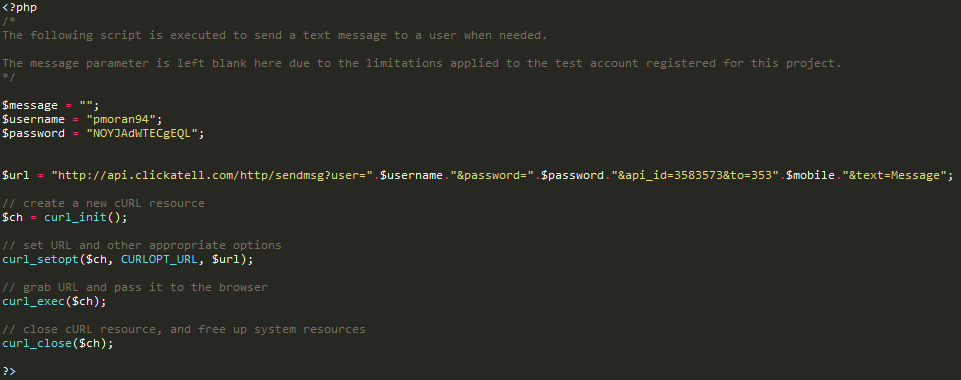


Figure 61 - cURL command to send SMS via. Clickatell

As seen in Figure[] the same code is called for each text message to send the same message to each receiver. This is due to that fact that Clickatell’s free subscription having limited functionality until a paid subscription is created.

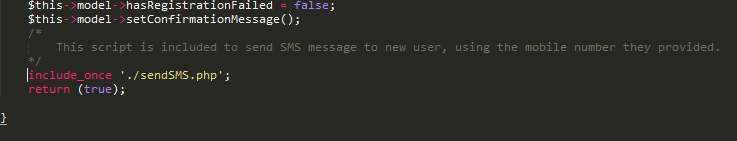


Figure 62 - Call SMS API (Controller.php)

### MailGun’s Emailing Service

Similar to Clickatell’s SMS API service, MailGun provides a Emailing service which in the case of this system is used to send out the invites to the appropriate address along with an attachment bearing the QR Code PNG file.

The emailing script is only invoked once in the controller although it is in a loop to allow it to send to all the necessary email addresses who have invited to this particular event. See Figure[].

The MailGun Emailing code is called using its own built in API. See Figure[].

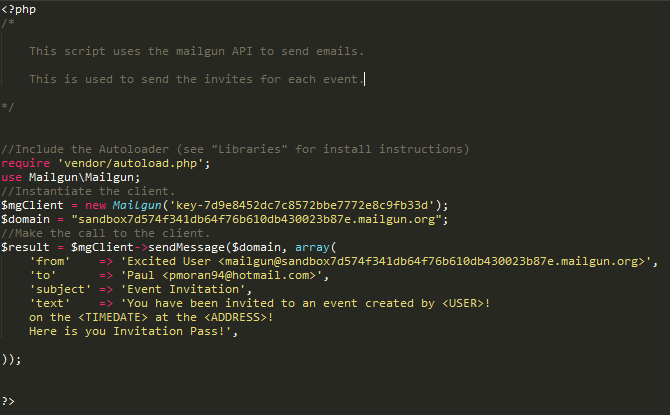


Figure 63 - MailGun API to send Email

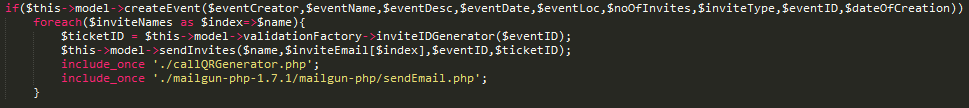


Figure 64 - MailGun API being called (controller.php)

## Conclusion

From this chapter, the overall majority of functioning software and features has been outlined. Although there are multiple approaches and aspects which may have been included to improve the content of the system, the features explained in this chapter provided useful, conclusive solutions to some problems which were brought up during the interim phase of the project. To conclude this chapter here is a summary of the topics which have been discussed.

1. The development and structure of the database.
2. The implementation of the MVC framework.
3. The development of the HTML templates and how they populate the view.
4. The progression of the User Interface and Structure.
5. Further development of required functions and methods throughout the system.
6. The generation, scanning, and manipulation QR codes are their data using various applications mentioned in the research chapter.
7. The use and implementation of other API’s within the system such as Clickatell SMS and MailGun Emailing.

In the next chapter the testing and validation aspects regarding the implemented software is evaluated and results concluded.

# 5. Project Validation

## Introduction

Since the MVC Framework is used, it will be ensuring appropriate structure to both my system and website. Each aspect and component will be tested through both, *Black Box and White Box Testing*. In order to complete a full software examination both types of testing are required.

This chapter will also provide some demonstrations of the features in the system during operation. Some of the features being demonstrated are Clickatell’s API, MailGuns service, the generation of each QR code type, some HTML templates etc. Links will also be provided to videos of various features in operation.

Issues encountered during the course of the project are also highlighted including descriptions and solutions (this includes proposed solutions should the issue have not have been resolved.).

## Testing

***“A process of analysing a software item to detect the differences between existing and required conditions (that is defects/errors/bugs) and to evaluate the features of the software item” -* ANSI/IEEE 1059 standard**

Testing is vital to a systems development to ensure that the system does not fail at various point of use where the user may not understand what may have went wrong. It contributes to the completeness of the system/software.

### White Box

White Box testing is performed where by the tester has extensive knowledge of the code behind the system. By knowing and understanding the code within the system, the tester is hereby able to estimate sample data to test within the system. The tester is hereby also able to evaluate the results, to determine if the program diverges from the intended objective. Ref. [30]

For this project I will be the main white-box tester, by listing all possible inputs based on my code and constraints I will be taking note of point in which the system has failed, and ensuring that as much invalid input error handling is managed in both my factories and my controller.

* Unit Testing

This is carried to by the developer to test various inputs on the system based on the developer’s knowledge of the code.

### Black Box

Black Box testing is testing which is performed by a tester who has no prior knowledge of the internal code. The tester (who may or may not have any knowledge of programming code) can only assume what the possible output may be based on the type of system and whatever input they may have tested. Unfortunately, not every aspect/path or possible input can be tested, therefore some gaps maybe left unattended. Ref. [31]

In terms of my project, to gain a sufficient evaluation on the functions, design, and controls I will need to extensively perform both of these forms of testing. As mentioned above, during Black Box testing, not all aspects can be covered 100%, therefore I have decided to perform Black Box Testing at monthly intervals by three separate testers. This approach will allow a variety of users with different levels of computer understanding to test the system. All results will be recorded, analysed and updated as each test is performed.

### Unit Testing

#### Customer User Type

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Customer Login and Registration** | | | |
| **Test No.** | **Test Name** | **Description** | **Result** | |
| 1.1.1 | Customer Self - Registration | Insert completed successfully using appropriate hash functions. SMS API triggered and SMS sent to mobile registered by new user. | | **PASS** |
| 1.1.2 | Customer Login | Successful login using the new user’s credentials. The SESSION variables have been set using the user\_id and username. | | **PASS** |
| 1.1.3 | Customer Logout | The SESSION variables have been successfully unset throughout the system. | | **PASS** |
|  | **Customer Navbar Options** | | | |
| 1.2.1 | View Profile Info | Customer can successfully view their profile data individually. | | **PASS** |
| 1.2.2 | Edit Profile Info |  | |  |
| 1.2.3 | Change Password | Customers password changes successfully provided the previous password Is given. | | **PASS** |
| 1.2.4 | Report Issue | Report successfully issued to the database. | | **PASS** |
| 1.2.5 | MyStripe Access |  | | **FAIL** |
|  | **Customer Side-Bar Options** | | | |
| 1.3.1 | Top Up Parking Ticket (Duration) |  | |  |
| 1.3.2 | Top Up Parking Ticket (Amount) |  | |  |
| 1.3.3 | Auto - Payment After Top Up |  | | **FAIL** |
| 1.3.4 | View Events | Customer can view all the events they have created as well as the ones in which they have been invited to. | | **PASS** |
| 1.3.5 | View Individual Event | Customer can view the full details of a given event. | | **PASS** |
| 1.3.6 | Remove Event |  | |  |
| 1.3.7 | Track Sent Stamps | A customer can see the ‘status’ of each postal stamp in which they have created. | | **PASS** |
|  | **Customer Backend System** | | | |
| 1.4.1 | Generate Post Stamp | Generated postal stamp data is successfully inserted in the database where needed. | | **PASS** |
| 1.4.2 | Correct QR Displayed |  | |  |
| 1.4.3 | Generate Parking Ticket | Parking ticket data is generated and inserted to the database where needed. | | **PASS** |
| 1.4.4 | Auto-Payment on Predefined Expiry |  | | **FAIL** |
| 1.4.5 | Correct QR Displayed |  | |  |
| 1.4.6 | Generate Event | Appropriate event data generated in inserted to the database. | | **PASS** |
| 1.4.7 | Invites Generated | Individual invite data is inserted to the database and passed to QR generator. | | **PASS** |
| 1.4.8 | Invites Sent (Email) |  | | **FAIL** |
| 1.4.9 | QR Generation for Each Invite |  | | **FAIL** |

#### Employee User Type

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Employee Login** | | | |
| **Test No.** | **Test Name** | **Description** | **Result** | |
| 2.1.1 | Employee Login | Employee successfully logs in using the data provided during registration through the admin page. Appropriate SESSION variables are set. | | **PASS** |
| 2.1.2 | Employee Logout | All required SESSION variables are successfully unset. | | **PASS** |
|  | **Employee Navbar Options** | | | |
| 2.2.1 | View Profile Info | Employee can view all their profile details. | | **PASS** |
| 2.2.2 | Edit Profile Info |  | |  |
| 2.2.3 | Change Pin | Employee can successfully change login in pin once old pin is provided. | | **PASS** |
| 2.2.4 | Load Scanned Data | Employee can load appropriate data which matches their own service. The data is read from the Excel Spreadsheet. | | **PASS** |
| 2.2.5 | View Scanned Data | Employee can view the relevant scanned data which matches the service they provide. (STAMP, CPARK, EVENT) | | **PASS** |
| 2.2.6 | Delete Scanned Entry |  | |  |
| 2.2.7 | Approve Scanned Entry |  | |  |
| 2.2.8 | Contact Customer |  | |  |
|  | **Employee Side-Bar Options** | | | |
| 2.3.1 | View Issues | Employee can see the issues reported by customers. | | **PASS** |
| 2.3.2 | Delete Issue |  | |  |
| 2.3.3 | Search Customers | Employee can successfully search for a customer. | | **PASS** |
| 2.3.4 | Display Search Results | Search results are successfully displayed of the customers. | | **PASS** |
| 2.3.5 | Delete Customer |  | |  |
|  | **Employee Backend System** | | | |
| 2.4.1 | Data Scans to Excel File | When the scanner is on, data is scanned to the excel file successfully and inserted row by row. | | **PASS** |
| 2.4.2 | Data Scans to Particular Excel File Automatically and Saves |  | | **FAIL** |
| 2.4.3 | Auto-delete Content from Excel after Scan |  | | **FAIL** |
| 2.4.4 | Data Is Validated Appropriately when Scanned | Data is passed through validation criteria before being manipulated or inserted to database. | | **PASS** |

#### Admin User Type

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Admin Login** | | | |
| **Test No.** | **Test Name** | **Description** | **Result** | |
| 3.1.1 | Admin Login | Admin login successful using correct credentials. | | **PASS** |
| 3.1.1 | Admin Logout | Appropriate SESSION variables are unset and user is signed out. | | **PASS** |
|  | **Admin Navbar Options** | | | |
| 3.2.1 | View Profile Info. | Admin can view their own profile information. | | **PASS** |
| 3.2.2 | Edit Profile Info. |  | |  |
| 3.2.3 | Change Parking Rate | Admin can successfully change the parking rate of the system. | | **PASS** |
| 3.2.4 | Add New Employee | Admin can successfully register a new employee for the system. | | **PASS** |
| 3.2.5 | Change Pin | Admin user can change their own login pin once the previous pin is provided. | | **PASS** |
|  | **Admin Side-Bar Options** | | | |
| 3.3.1 | View Customer Issues | Admin can view the reported issues from customers. | | **PASS** |
| 3.3.2 | Delete Customer Issues |  | |  |
| 3.3.3 | View Customers |  | |  |
| 3.3.4 | Search Customers |  | | **FAIL** |
| 3.3.5 | Delete Customers |  | |  |
| 3.3.6 | View Employees | Admin can view all employees in the system. | | **PASS** |
| 3.3.7 | Delete Employees |  | |  |
| 3.3.8 | Make Employee Admin | Admin can successfully upgrade a regular employee to admin. | | **PASS** |
| 3.3.9 | View All QRs | Admin can view all QR data rows generated which are In use to the system. | | **PASS** |
| 3.3.10 | Delete QRs |  | |  |
| 3.3.11 | View All Events | Admins can view all events created within the system. | | **PASS** |
| 3.3.12 | View Invites for Events |  | |  |
| 3.3.13 | View All Stamps | Admins can view all the relevant information for each stamp in the system. | | **PASS** |
| 3.3.14 | Delete Stamps |  | |  |
| 3.3.15 | View all Parking Tickets | Admins can successfully view all the relevant information for each parking ticket in the system. | | **PASS** |
| 3.3.16 | Delete Parking Tickets |  | |  |

#### Other System Features

|  |  |  |  |
| --- | --- | --- | --- |
| **Test No.** | **Test Name** | **Description** | **Result** |
| 4.1.1 | Appropriate checks for Stamp | The system run multiple checks to make sure that it is in fact a valid stamp to this system. | **PASS** |
| 4.1.2 | Appropriate checks for Parking Ticket | The system ran multiple checks to make sure that the parking ticket was valid and had not yet expired. | **PASS** |
| 4.1.3 | Appropriate checks for Event Invite |  |  |
| 4.1.4 | Appropriate Data Manipulated for Stamps | Data is changed in the system and database based on the correct validity of that ticket. | **PASS** |
| 4.1.5 | Appropriate Data Manipulated for Parking Tickets | Data scanned into the system is manipulated appropriately based on its validity and payments made. | **PASS** |
| 4.1.6 | Appropriate Data Manipulated for Event / Invites |  |  |
| 4.1.7 | Emails Sent for Invites |  | **FAIL** |
| 4.1.8 | SMS sent where required by API. | The SMS API is called throughout the system when required. | **PASS** |

## Demonstration of Project Features

### Use of Clickatell’s SMS API

### Use of MailGun’s Emailing API

### Generation of the QR Codes

### Scanning of QR Code

### Validation of Scanned QR Code

### HTML Templating

### Login

## Problems and Issues

#### Introduction to Problems and Issues

In this section the issues and problems which were encountered during the duration of the development of the project are discussed. Each issue is given its own section describing the issue itself, the status in terms of the imposed system and the solution either given or proposed solution which would be applied if more time was given or if new approaches to the project were used during the planning and design stages.

#### Problems and Issues

|  |
| --- |
| **Issue/Problem:** Time Management |
| Description: After spending too much time focusing on certain aspects during the implementation phase, some aspects including the deployment of the system was delayed. Overall this meant that some features were either only partially completely by the end or even some features were not implemented due to poor time management in other areas. |
| **Status:** N/A |
| **Solution / Proposed Solution:** If the project was to be repeated more time would be spent on planning and prioritizing some elements which may have been overlooked this time round. |

|  |
| --- |
| **Issue/Problem:** SMS Messaging |
| **Description:** One of the issues raised during the design of the project was the feature to send SMS messages to a particular user when required by the system. |
| **Status:** Resolved |
| **Solution / Proposed Solution:** The solution to this issue was to invoke the use of Clickatell’s API which executes a cURL command to send the required data ie, Mobile Number and Text Content to receive the message from Clickatell’s API number. Ref. [34] |

|  |
| --- |
| **Issue/Problem:** Scheduled Execution of script the check data in the database after a repeated set time. |
| **Description:** One of the harder tasks of the project was to make the system continually recognise certain dates and times within the database. For example, the system would need to know that the expiry time of a parking ticket was nearly reached in order for the SMS script to be triggered to notify the user to either top up or return to their car. |
| **Status:** Partially Resolved |
| **Solution / Proposed Solution:** The recommended solution to this issue was to use a Crontab operation to trigger the script to check the data in the database without a user’s interaction, however since this system was not deployed on a server an alternative solution was drawn up. This was to use Windows Sched-Task to call the script since the system was developed locally on a Windows machine. |

|  |
| --- |
| **Issue/Problem:** Sending Emails from localhost |
| **Description:** Due to the delay caused by the system not being deployed sending emails from my localhost raised a number of problems which required adjustments to configurations and settings within the php.ini file regarding SNMP etc. |
| **Status:** Resolved |
| **Solution / Proposed Solution:** The quickest solution to handle this problem given the amount of work and time, was to use an API provided by MailGun. This API allows for multiple emails to be sent by the system where needed. |

|  |
| --- |
| **Issue/Problem:** Header Problem (1) |
| **Description:** One of the more time consuming errors which was found during the implementation process was in regard to the inclusion of an external PHP file. When executed the HTML was populated with incomprehensible error code. |
| **Status:** Resolved |
| **Solution / Proposed Solution:** Solving the first problem in regard to the external header file was an exceptionally simple solution which would never have been discovered had it not been researched.  As it turns out, in order for an external header file to be recognised, there is a common error called the *white space problem.* In order to solve this problem, the white space (if any) before the PHP tags where the header is being called, i.e., <?php> , must be removed. |

|  |
| --- |
| **Issue/Problem:** Header Problem (2) |
| **Description:** After solving the white space problem, now when the external PHP was executed, the header file of that page would overwrite the header of the source page. Therefore the main page was no longer in the view but the external page was. This meant the page was no longer interactive and simply only display the content of the external file. |
| **Status:** Resolved |
| **Solution / Proposed Solution:** The reason for the external PHP page being included within the source page was to display the full HTML surrounding the PNG file created. A simple solution to this was to evade including the full HTML of the external file and simply extract the newly generated PNG image (QR Code) from the necessary folder to be displayed where necessary within the source HTML. |

|  |
| --- |
| **Issue/Problem:** Multiple User Login / Template Cross-Over |
| **Description:** Another problem which arose during implementation was dealing with views for each type of user. At first the idea and approach was to create one single index-template for all users and populate that template with content based on the type of user which was logged in. Unfortunately a problem kept arising from this approach where after any entity on the page was clicked after logging in, the content of the view/template would swap from user to user. |
| **Status:** Resolved |
| **Solution / Proposed Solution:** To overcome this simple problem, a different approach was attempted, in which requires using multiple index-templates (one for each type of user), where depending on the type of user that is logged in, the template would vary. Unfortunately at first this raised another issue where multiple templates where being called at once. This was due to a simple coding fault in which two *if-statements* where true at the same time. Following the full solution this proved rather effective towards the completion of the project. |

|  |
| --- |
| **Issue/Problem:** Internet required for bootstrap and other libraries |
| **Description:** Unfortunately for a one week period of developing my website, my internet at home was down. Only then was it realised that some of the bootstrap was using was being referenced using a live link. An attempt to download a later version of the bootstrap file was tried but with no luck. |
| **Status:** N/A |
| **Solution / Proposed Solution:** Although this was not so much an error as it was a setback. It was important to focus on other aspects of the project which did not require an internet connection. |
| **Issue/Problem:** Event Self-Invite |
| **Description:** This issue was more caused by poor planning and time management. The issue is that every invite sent it sent direct to a particular email, which prevents others from attending/ knowing about other *open-invite* events which may be of interest. |
| **Status:** Unresolved |
| **Solution / Proposed Solution:** A proposed solution to this problem would be to simply set create the option for a user to search certain “tags” which would relate to their interests regarding a particular event. When searched the results would display a table of *open-invite* events in which they would request an invite. The host of that event would *approve/reject* the request for the invite simply because it is their event. Otherwise once there is still spaces/tickets available for a “paid” event a user may purchase a valid ticket for that event. |

|  |
| --- |
| **Issue/Problem:** Backing up material during development |
| **Description:** One simple issue I which occurred during the development of this project was the breakage of the machine in which it was being built on. Since the work had not been backed up within two days prior, the recent work was lost. This is more of a personal practice issue than technical. |
| **Status:** N/A |
| **Solution / Proposed Solution:** After purchasing a new machine to continue the project development it was possible to continue from the latest backed-up material. |

|  |
| --- |
| **Issue/Problem:** Deployment |
| **Description:** The single biggest, and most time-consuming issue was the process of deploying the website. The main attempt was using a Microsoft Azure Ubuntu Virtual Machine, where I was getting the error regarding database connection issues. Other attempts of deployment using similar methods included:   * Deployment on local virtual machine using Oracle Virtual Box. * Deployment on an Okeanos Virtual Machine. * Deployment using Heroku PHP & MySQL   Each deployment method proved equally disappointing. |
| **Status:** Unresolved |
| **Solution / Proposed Solution:** |

|  |
| --- |
| **Issue/Problem:** Creating a separate model for QR behaviour. |
| **Description:** If a separate model was created which handled all QR data generation, scanning and manipulation it would make the system a lot more sustainable. This issues was raised at a late stage in the development of the project due to a poor choice of structure regarding the QR features. |
| **Status:** Unresolved |
| **Solution / Proposed Solution:** By invoking an extra model for the QR features within the controller and view, it would structure the code into more organised and maintainable fashion. |

|  |
| --- |
| **Issue/Problem:** Displaying the correct QR Code after generation |
| **Description:** Following the generation of the QR Code it was important to display the correct PNG file on screen to the user. The issue arose where the PHP and HTML used to get and display the correct file were not doing that, instead raw code was being displayed in its place. |
| **Status:** Unresolved |
| **Solution / Proposed Solution:** Although code has been provided to attain the correct file as well as the HTML used to display the PNG image the issue still remains that raw source code is being displayed. However, the solution to this issue is clearly something simple due to a syntactical error. |

|  |
| --- |
| **Issue/Problem:** Manipulating Date/Time Variables in the Database |
| **Description:** This system strongly relies on accurate readings of date and time elements of the system. For example, the expiry time of a parking ticket needed to be recognised in order for certain criteria to be matched. The issue was that the updating, and comparing of these time variables was rather tricky and very inconsistent event with the built-in methods that PHP provides. |
| **Status:** Partially Resolved |
| **Solution / Proposed Solution:** Although the majority of date/time comparisons worked some of the updating methods for the new times were rather inconsistent. By using multiple methods to gain a more accurate solution to update the variable I was able to make beneficial adjustments to get the time as accurate as possible. |

#### Conclusions

Although not all problems and issues which occurred were resolved, a solution has been drawn up for each of them. Some of these issues will be discussed further in the Project Plan chapter regarding aspects which may be changed if the project were to be repeated.

As discussed, some issues were simply down to poor design and planning choices. To conclude this issue alone it is evident that further planning and design is worthwhile in any project title.

# 6. Project Plan

## Introduction

In the following chapter further future development and uses of the system are outlined. Many aspects and approaches would applied if this project was to be repeated. In this chapter some of these changes are discussed in detail regarding almost all phases of the project duration including research, design and implementation. But first the section regarding changes which occurred during the duration of the project, from the proposed system/design and the finished product are discussed in detail outlining why these changes took place and what benefits came from these changes.

## Deviation from Proposed Idea

The initial proposed idea was simply to create the postal stamp system in which a user would not need to be purchased from a post office or newsagent.

This simple idea was expanded to a more generic idea which incorporated the QR system containing a QR code generator and handling scanned data methods. This system proved to be rather more sustainable and useful as it could be applied to multiple utilities where needed. Although it proved an awful lot more complex and time consuming. If fully completed it could be expanded to other uses which are mentioned later in this chapter.

## Changes to be addressed if Project was repeated

On completing the majority of the project based on the tasks and objectives which were set out from the start, it was evident that there were multiple areas in which would require improvement or a new approach if the project was to be repeated.

Although there are other areas not mentioned below which may require more work, the main reason for this was due to poor application of other areas. Personally, I believe that by being more attentive to the factors below this would have had a knock on effect on the more practical side of the project.

In this section these factors are outlined given reasons and alternatives as to why they would be changed during a repeat of this project.

#### Further Planning and Time Management

Although extensive planning was carried out prior to the development aspects of the project, I personally believe that by applying some of the knowledge gained over the duration of the project regarding planning and research better structure would have been applied the management aspects of the project.

Attempts were made of making time management issues more flexible during the project which resulted in poor project management in other areas as assigned times began to overlap into other processes.

If the project was repeated time management would become a priority in itself. This is simple a lesson learned during this projects first attempt.

#### Extended Research into the Software and Objectives of the Project

Many of the aspects and features described in the earlier stages of the project are rather simple in theory but due to lack of research in regard to how these features would be developed/ applied resulted in inaccurate and delayed completion.

If the project were to be repeated it would be essential to ensure that each and every aspect and feature which was relevant to the requirements must be researched and tested prior to design and implementation of the feature.

Since this issue was not addressed at an earlier stage some features which are embedded into the system took longer than anticipated to implement.

#### Further Design

Although the design of the system and framework was rather accurate compared to the proposed system and the finished product, some areas of the design were altered during the process. These aspects include:

1. The ERD structure which was applied to the database.
2. Elements of the Interface which provided some extra simple functionality to the user.
3. Folder structure in which some features needed extra repositories since they were not discovered until later in development.

#### Prioritization towards System Requirements

One of the major setbacks of the system was due to project management in regard to system requirements. Although plenty of aspects were added to the system to make it a more complete system, some important system requirements were delayed during implementation due to less critical elements being developed prior to them.

This issue prevented a number of key requirements from been complete before the deadline.

As a lesson learned from this experience it is now understood to prioritize major elements of a given system over less important features.

## Future Development of Project

Essentially, the idea of the completed system is to allow the user to use this unique generated barcode for multiple purposes as proof of validation on payments and as a point of payment also.

Example 1,

Postal System – User Registers with system, and is given their unique id code. Should that user wish to send a letter, they can simply, select from the website the option to *Generate Stamp*. They will then be shown a *barcode on – screen* which will be made up from the following data.

1. 6 character ID for Barcode Type i.e., STAMP
2. Personal Unique ID Number i.e., 1234567
3. Ticket ID ie, 1234567

Therefore the Barcode generated will contain the following data:

STAMP12345671234567

When scanned by the system it will recognise what the barcode is for – i.e., Postal Stamp,

Then it will recognise the user account to deduct the appropriate amount from that user account.

As a security feature a time limit from the time of generation from which the barcode is valid was implemented. This would be to prevent unauthorised users using someone’s generated barcode for more than the expiry time should they manage to attain the barcode.

Example 2,

Car Parking – Same as above, after registration the user will be given their unique personal id. Should that user wish to pay for a parking, they can simply select from the website the option to *Generate Parking Ticket*. They will then be shown a *barcode on – screen* which will be made up from the following data.

1. 6 character ID for Barcode Type i.e., CPARK
2. Personal Unique ID Number i.e., 1234567
3. Ticket ID i.e., 1234567

Therefore the Barcode generated will contain the following data:

CPARK12345671234567

From here the user can top up the amount they wish to pay from their account. They can specify the time of parking, and the duration of which they wish to stay parked. The amount deducted from their account will be determined based on the duration they have specified. Should the user wish to stay longer than they initially opted for, they will have the option to update their parking status to extend their duration. Again based on this the amount deducted will also be updated.

As seen from the above examples of how the system can be used based on the type of barcode which is generated, essentially the system is an all-in-one way in which a user can simplify various everyday activities which require various identification and validation through the use of a scanning system.

This is the type of system which could be implemented into multiple aspects of everyday life all within one system. Ranging from postal stamp system, ticket validation, and user identification it could also be expanded at a later phase to implement a simple limited point of sale utility.

For example,

Similar to *Pay-Pal,* where people can top up their Pay-Pal account and purchase things online using that utility rather than providing personal banking information. This system could be expanded to an over the counter point of sale system, where the user could carry a copy of their barcode with them within their wallet or as a wristband, in which a shop keeper will scan the barcode and the amount will be deducted from their personal account with this system or from a third party account such as PayPal or Stripe rather than their personal bank.

Essentially, this is system could be expanded to cover a wide range of applications across various uses incorporated with NFC (Near Field Communication), Bluetooth etc. Ref. [37]

#### Incomplete Intended features in current System

1. **Stripe Implementation.**

As one of the more important aspects of this system, the payment mechanisms was to be used to pay for the postal stamp duty and parking ticket fare automatically during either top up, generation or on scan by a member of the system(employee).

1. **Self-invite to open invite events in a particular area or region.**

One feature which was decided on at a later stage would be to allow users to view ‘*Open-Invite*’ events which have been created but which they may not have received an invite to. They can simply view these events and choose to ‘Request’ an invite in which the host can respond be issuing the invite to their email address or reject the request.

1. **Deployment of System.**

One of the main concepts of the system, was to deploy it. Due to deployment issues other aspects of the systems were delayed such as the method in which data was to be scanned into the system, and the crontab.

1. **Crontab/Easycron Implementation.**

As mentioned in the previous point, the crontab utility was unable to be used due to the system not being deployed. This was going to be used for multiple purposes such as:

* Checking if an event date had been passed in which it would trigger the event to become inactive which would prevent tickets issued for that system from being scanned into the system.
* Notifying a user by SMS that they have an event on that day.
* Notifying a user that their car parking ticket was approaching its expiry time.

1. **Adjustments to Admins Capabilities.**

Some functional features for the admin user were left out. Another aspect which was to be added was the ability of an admin to specify an area in which they were in control of. This would be affect the pricing range of parking from one area to another.

1. **Further aesthetical development in the UI.**

It was intended to provide confirmation and error messages throughout the system which something was either successfully affected in the database or if a failure occurred.

1. **(Event) Scanning for tickets relevant to particular event by EventID.**

It is important to make sure that when a ticket for an event is scanned into the system, that the ticket scanned was generated for that particular event. To address this is issue, it would simply mean that when the employee (with EVENT privileges) displayed the table of scanned information, they would have to provide the ID for that event and the table would be populate with the tickets which were issued for that particular event.

1. **Prevent QR generation if NO Stripe account is registered to a particular account.**

Unfortunately since the Stripe payment feature was incomplete by the deadline, it prevented the option to ensure that a user did in fact provide the correct credentials for a Stripe account to be linked to the system. This would then be used to ensure that QR tickets were not being generated by any registered user unless they had the account provided to be charged. (Applies to POSTAL STAMP and CAR PARK TICKET)

1. **Accurate pricing of stamps based on 3 column matrix.**

This feature was also based on the payment aspects of the system. It would simple be used to calculate the correct price which would be applied to the postal stamp. The criteria in which the Irish postal stamp is based on is: Destination, Weight, Postage Type.

# 7. Conclusions

## Introduction

This chapter will provide an overall reflection of the project. As well as outlining some personal benefits and reflections it will finally give an absolute conclusion to the report and project standings by applying Nielsen’s Heuristics to the finished system. But firstly a quick summary of the chapters provided in this report will be outlined.

## Report Summary

The following section outlines a summary for each chapter described in this report.

#### Project Introduction

This chapter outlined the issues and purposes related the project being built.

###### Project Statement

The project statement described the initial uses and benefits of developing the proposed system. It gave a quick insight to some everyday tasks in which would be made a lot simpler to the end user by using the system described in this report.

###### Project Objectives

The project objectives section described the necessary functional requirements to enhance and complete the system. Each objective had its own sub-objectives in order to break down each task individually.

###### Structure of the Project Report

The structure of the report section simply described to the reader the following content of the report in which they would be reading should they wish to proceed.

###### Conclusion

To conclude this chapter, the conclusion gave a simple summary of the sections described prior to it.

#### Research

###### Introduction to Background Research

The introductions section to this chapter essentially outlined the common purpose to which the project was necessary. It further described the issues in society which the proposed system may be able to adjust positively in some way.

###### Existing Systems and Solutions

In this section of the research similar systems were compared and contrasted to gain a better insight as to how the proposed project should be designed and approached. By looking at the positive and negative aspects of these systems it was easier to know what design approaches to avoid and to know where other negative aspects could be adjusted.

###### Technologies Researched

Since there is thousands of possible technologies which may have been used to build this project, and hundreds more variations for each, it was important to compared, test and evaluate which technologies should be used to build this project. The technologies looked at included: Databases, Programming Languages, Development Tools, Browsers and Deployment Mechanisms.

###### Other Research Performed

In this section, other research which was carried out is outlined as well as providing a short summary of the findings. This other research included interviews with possible users, questionnaires filled out by other IT professionals and some advice given by those IT professionals.

###### Resultant Findings/Requirements according to Research

The final aspect of this chapter saw the evaluation and conclusion to the research section. The technologies and approaches that were to be used were finally described here and outlined for there use.

#### Design

###### Introduction of Design

Firstly, an introduction was given outlining the content of the entire design chapter.

###### Approach and Methodology

This chapter further describes the methodology chosen from the research chapter, and further outlined its description and its use. In the case of this project we looked at *Agile Methodology.*

###### System Architecture

The system architecture shows the basic fundamental structure in which the system was to be built. In the case of this proposed project a *three-tier* architecture was used. This consisted of a frontend, backend, and database.

###### Design of User Interface

This chapter provided initial structural drawings and layouts for proposed templates for each user type in the system. It also outlined some of the tools and features which were used to enhance the UI such as Twitter Bootstrap, Palleton etc.

###### Design of System Structure

This chapter simply described the folder structure of the software which in turn outlines the paths defined throughout the code where needed. This ensured for a more sustainable and maintainable system.

###### Design of Source Code

This chapter provided an insight to the structure of some coding elements by outlining the purpose of the comments within the code as well and description of each file by the use of comment boxes.

###### Project Features

This chapter describes various features and assets in which were to be used throughout the system. This included API’s, external software and other protocols.

###### Other Design Documents

In this section design documents such as use cases, ERD diagrams etc. were displayed to provide further relevant detail for the project.

#### Development and Implementation

###### Introduction to Implementation

This section introduced the reader to the Development and Implementation phase. It briefly described the entire content of the Development and Implementation chapter.

###### Development of Database

This section outlines the structure and development of the database to be used for the system. It was based on the ERD diagram described in the *Other Design Documents* section of Chapter 3.

###### Development of MVC – Model, View, Controller

This chapter describes in detail the full implementation of the main structural framework of the system, which in this case is *MVC.*  Each element was described and the step by step process to its implementation was given. The connections between each class was also summarized.

###### Development of HTML Templates

This chapter gave the full view on each of the index templates used throughout the system for each user type. It also provides the full detailed list to the forms and HTML elements used for the User Interface. The integration of these files was not shown until the following section.

###### User Interface

The User Interface sections describes the integration and use of the templates, HTML forms and the *View.*  The combination of each of these aspects, builds up the majority of the User Interface.

###### Development of Functions and Factories

This section provided the details about the functions, methods and factories used within the system. Each main function was given a description and a use. It also describes the way in which each function is used. The factories provided as part of this project are : *Validation Factory,* and *Authentication Factory.*

###### Implementation of QR Code Features

One of the main aspects to the system is the QR Code Features. This section outlines the implementation of the generation components, the reading components and the manipulation of the data within the QR Code during creation and scanning.

###### Use of Other API’s and Software

Other API’s such as *clickatells SMS messaging* is described in full detail regarding its use and description. Any other external software used within the system is outlined in this section.

#### Project Validation

###### Introduction

In this section an introduction to the project validation chapter is provided. It gave a brief description to the following sections of the chapter.

###### Testing

In the testing section of the project validation, different test cases are examine on each of the functional, and some non-functional requirements throughout the system. The section also provides a summarised version of conclusive evaluations of the testing phase.

###### Demonstration of Project Features

This section will gave a description of the use of some of the features and functionality within the system. It also provides some useful links the videos of the system being used.

###### Problems and Issues

This section outlines all of the main problems encountered throughout the participation of the project. It gives a description of each problem both solved and unsolved as well as giving a full descriptive approach to how some other problems which were not solved would be tackled.

#### Project Plan

###### Introduction

This section of the Project Plan chapter simply provided an introduction to the following sections of the chapter regarding the future evaluations of the project.

###### Deviation from Proposed Idea

In this section, the main change between the proposed plan and idea and the finished product are highlighted. Reasons for the change, how they were changed and other relevant aspects are also outlined.

###### Changes to be addressed if Project was repeated

This section looks at other approaches which would be taken, or adjusted if the project was to be repeated. Since not all aspect of the proposed design were completed during implementation, other approaches which may have enabled each of these aspects to be fixed are outlined in this section.

###### Future Development of Project

This section discusses some future possibilities and development ideas for the completed project including new possibilities and application uses for the system.

## Project Summary and Heuristics

#### Key Completion Components Include

* Aesthetically Pleasing Web Interface
* Good Form Controls throughout the Website
* Successfully Implemented MVC Framework and backend
* Effective Barcode Generation to successfully identify the unique ID of the person, and uniqueness of the ticket.
* Areas of effective data manipulation.
* Proof of concept regarding use of QR Generator and scanner.
* SMS Gateway Successfully Established
* Successful Validation of Each Ticket when Scanned
* Login and Registration Constraints

#### Backup and Storage

Throughout the development process of the project, the use of backup software was essential. Two platforms were used to back-up the software and material of the system.

* DropBox

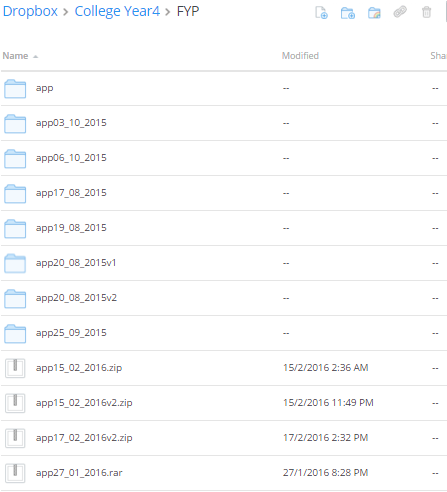


Figure - DropBox Usage

* Github

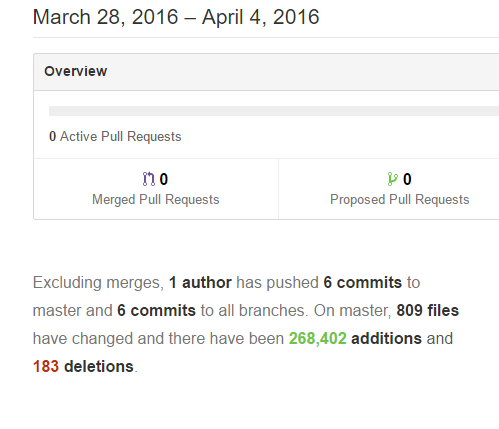


Figure - Interaction Statistics

This image explains the git repository interactions that were made during the final week of development.

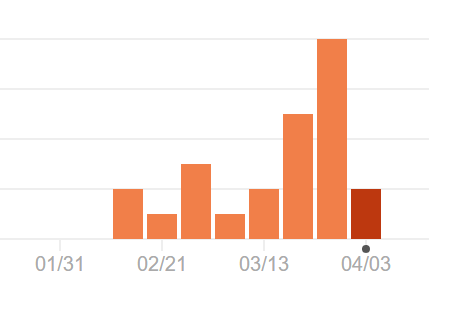


Figure - Git Commits

This is a represent of the commits into Github between Mid-January and the project deadline.

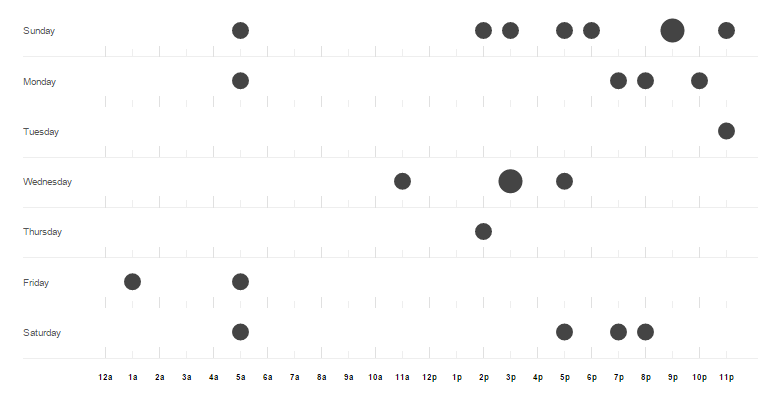


Figure - Commits Punchcard

#### Nielsens Heuristics of Completed System

|  |  |  |
| --- | --- | --- |
| **Heuristics** | **Rating** | **Description** |
| **Visibility of System Status.** | 3 | Not much use of confirmation or error messages throughout the system. |
| **Match between System and Real World** | 8 | Easy to understand what the system does, although not all features presently work. |
| **User Control and Freedom** | 6 | Confirmation is required to proceed with the generation of the main feature in the system. (QR generation). Although database usage does not feature undo and redo. |
| **Consistency and Standards** | 8 | The system is rather straight forward to user for each user. |
| **Error Prevention** | 2 | Error messages are all but present. Although some operations seem like they worked to the user, the backend operation may have failed at some point. |
| **Recognition Rather than Recall** | 7 | Most information is available to the user. The only this is not the case, is when an ID is displayed in the place of a Name, or when a user may wish to know a particular value within the database. |
| **Flexibility and Efficiency of Use** | 3 | Employee User has to scan data in a particular order to make sure that the data in captured in the Excel Spreadsheet. |
| **Aesthetic and Minimalist Design** | 8 | The interface is made up of good contrasting colours and is requires little effort to make out the elements on the page. |
| **Help users recognize, diagnose and recover from errors** | 2 | Lack of error messages. Means that a user may have a problem or issue but may not be aware of it. |
| **Help and Documentation** | 5 | The system is rather straight forward to use. But a lot more help and FAQ could be provided. |
| **TOTAL** | **52/100** |  |

## Learning Outcomes

Throughout the participation of all phases of this project, I have developed my skills in a wide range of areas both technical and non-technical. Here I will outline these new skills as well as improved ones.

#### Technical Learning Benefits

###### General Coding

Considering the quantity of coding aspects of this project it is evident that my own coding skills have improved since I began the project. Due to the practice and application of my coding abilities throughout the implementation of the system, the standard of my code improved immensely along with my knowledge and understanding of other source code libraries and elements available to use. This is something which I can apply to other languages, and protocols.

###### Use of APIs and cURL

Personally, I had never really used API’s much throughout my experience of computer science. By using various API’s along with the cURL command in PHP it gave me a good insight to their benefit to any given system which may requires its use. It has also given me the idea of which the system that is implemented into this project may be improved by changing its standard and making it an API itself for use in other systems.

###### Understanding of Data Interpretation of QR Codes

Working with various data coding mechanisms such as QR codes and barcodes has taught me how to encrypt data into these visual representation of specific data. I now have a strong understanding of their uses and benefits as well as their simplicity. Developing this system with these aspects showed me how software may interpret these images and manipulate the data where required.

###### Use of Microsoft Azure Virtual Machine

Unfortunately when attempting to deploy the system using a Virtual Machine on a Microsoft Azure web server, the system would not connect to the database. However the use of the Microsoft Azure gave me an indication of its uses in a wide range of applications. This also gave me a stronger understanding of web deployment on web servers.

#### Non-technical Learning Benefits

###### Importance of Research and Design

Due to the incompletion of various elements of the system, I later learned the importance of accurate design and planning. Overall many elements of the system worked well to contribute to the completion of the project, this gave me invaluable experience in the field of required research and design.

###### Project Management

One of the main skills developed during the project process was the ability to manage and schedule the phase of the project development. Although some areas of management were overlooked, this is simply something to improve on in other projects and tasks in the future. This project required high amounts of management in multiple areas such as planning, structure, time, and development. Experiencing project management in each of these areas thought me valuable lessons and improve my management skills as a whole.

###### Working under Stress

The overall workload of this project was somewhat overwhelming. Various tasks were tougher than others, which made me adjust preparations and planning in order to prioritize each system requirement. This is something which will hold to me a work environment and other projects which may require intense commitment.

## Personal Reflection

Overall I enjoyed making this project. I learned a lot of new skills as well as developing older ones. Computer Science and software development is a favourable subject of mine and to build a full system which contains multiple complex aspects gave me a strong insight to the type of projects and source code which I may be dealing with at a later stage. I was also happy to overcome some stressful situations and downfalls during the making of this project.

Some areas of the project proved more difficult than others in almost all areas. I was disappointed that I struggled to deploy the system which would have been a strong benefit to how the system would have worked. An area which let me down the most was time management, where some aspects of development overlapped other in terms of the time assigned to them.

A project like this gave me good experience working independently as well as recognising areas in which I needed help when required. The overall experience of building this project from start to finish provided me with an invaluable evaluation of my own skillset in this sector.

Personally if I could start the project again I would, but by using my newly gained knowledge which I picked up throughout the duration of the project I would make specific adjustments to the planning and management phases.

## Final Project Conclusion

As described throughout this report there were both positive and negative outcomes of this project.

The overall building of the project was rather stress-inducing, and time consuming but thought me many new lessons as an individual and developer.

Unfortunately, the project itself had many issues which had yet to be resolved. This was something which also thought me about responsibility and accountability when dealing with projects and objectives.

The interface of the system was rather a success as it was responsive on a number of different screen sizes and browsers. The concept of the system was delivered appropriately in terms of generating codes and data which may be used in multiple applications/utilities which would benefit the general user.

It demonstrates how a user can generate their own QR code to be used for one of the following utilities.

* Postal Stamp
* Car Parking Ticket
* Event Creation

The Postal Stamp generation was a success. The stamp is generated and available for the user to save and print. When the stamp is scanned by the correct employee user who has permission to the postal stamp service provided within the system. The data is read from an Excel file and checked for validity. Unfortunately, not all checks were successful during the implementation phases due to errors when handling date and time variable manipulation.

The Car Park Ticket generation was also a success. It is ensured that once a new ticket is generated by a particular user, any older tickets issued to that account are made inactive. The user has the choice to provide an expiry time on generation, or to add the expiry time at a later stage. This ticket, unless topped up will only be active for maximum 24 hours before it is deemed inactive and cannot be used again. Similar to the stamps, when the parking ticket is scanned the data is read to an Excel file initially. Before the data is pushed to the database, checks are made to make sure that the data is either valid or invalid.

The deciding factor for a valid car park ticket is that a payment has been made and that the current time/date has not exceeded the expiry time assigned to that ticket based on the amount paid.

As for the generation of an event, it is handled quite differently. The event itself does not in fact have a QR code but does require a unique ID. The QR codes that are generated for the event are the individual invitations. Each invitation is unique to a particular event. These invitations baring the QR code are to be sent to the provided email addresses of the invited persons.

Unfortunately for this application of the system, the generation of the QR code is delayed and very inconsistent. I believe this is due to the method which is used to call the QR generator. This second downfall to this application is that the invitations which are sent out are also rather inconsistent. This may also be a result from the same issue.

When scanning for event tickets, the employee user must provide the event ID to view the correct invites that particular event. Again, each ticket is checked for valid credentials to ensure that the ticket is valid, and the event is active.

The issue which was raised regarding the scanning of invite QR tickets, was too, the time/date manipulation.

Since the deployment was also unsuccessful, this prevented some aspects from also being completed such as the use of crontab, and PHP standard emailing service.

I found the experience of this project both very enjoyable and rather stressful. If anything was learned from this, it is that planning, design and time management should highly prioritized and carefully organized.

# Bibliography

1. ToastieCoastie, “Why are postage stamps necessary?,” 2015. [Online]. Available:

https://www.reddit.com/r/explainlikeimfive/comments/2a2519/eli5\_why\_are\_postage\_stamps \_necessary\_they\_just/. [Accessed 10 11 2015].

1. Parking Tag, “How it Works?,” 2015. [Online]. Available: https://www.parkingtag.ie/. [Accessed 12 11 2015].
2. EventBrite, “Fees and Pricing,” 2015. [Online]. Available: https://www.eventbrite.ie/fees/. [Accessed 20 11 2015].
3. Arekibo, “The Post Office and Stamps,” 2015. [Online]. Available:

http://www.anpost.ie/AnPost/History+and+Heritage/Learning/The+Post+Office+and+stamps/. [Accessed 15 11 2015].

1. 2-clicks-stamps, “What is a stamp collector?,” 2015. [Online]. Available: http://www.2-clicksstamps.com/article/what-is-a-stamp-collector.html. [Accessed 10 11 2015].
2. J. Nielsen, “10 Usability Heuristics for User Interface Design,” 2014. [Online]. Available: http://www.nngroup.com/articles/ten-usability-heuristics/. [Accessed 02 11 2015].
3. R. Miller, “The evolution of franking: different ways to indicate postage was paid,” 2015.
4. DLR, “Pay and Display Parking Methods,” 2015. [Online]. Available: http://www.dlrcoco.ie/aboutus/councildepartments/transportation/applyforit/parking/payand displayparkingmethods/. [Accessed 10 11 2015].
5. Apica, “Pros and Cons of Popular Web Browsers,” 4 August 2014. [Online]. Available: https://www.apicasystem.com/blog/pros-cons-popular-web-browsers/. [Accessed 19 10 2015].

[1 R. Zaremba, “Which programming language you should use for a web backend,” 08 03 2013.

0]

[Online].Available: http://rz.scaleit.pl/2013/03/08/which\_programming\_language\_should\_you\_use\_for\_a\_web\_backend.html. [Accessed 01 11 2015].

[1 R. H. Technology, “C, C++ or Java — Which Is Best for Your Backend Developer Career?,” 2015.

1] [Online]. Available: http://www.roberthalf.com/technology/blog/c-c-or-java-which-is-best-foryour-backend-developer-career. [Accessed 09 11 2015].

[1 JScriptors, “Advantages and Disadvantages of Javascript,” 2015. [Online]. Available:

2] http://www.jscripters.com/javascript-disadvantages-and-advantages/. [Accessed 10 11 2015].

[1 JScriptors, “Advantages and Disadvantages of JQuery,” 2015. [Online]. Available:

3] http://www.jscripters.com/jquery-disadvantages-and-advantages/. [Accessed 10 11 2015].

[1 R. Shannon, “What is HTML?,” no. 1, 2012.

4]

[1 M. Rouse, “Event-Driven Application Definition,” 2012. [Online]. Available:

5] http://searchcloudapplications.techtarget.com/definition/event-driven-application. [Accessed 15 11 2015].

[1 M. Rouse, “ActionScript Definition,” 2005. [Online]. Available:

6] http://searchsoa.techtarget.com/definition/ActionScript. [Accessed 15 11 2015].

[1 Sublime Text, [Online]. Available: http://www.sublimetext.com/. [Accessed 02 11 2015].

7]

[1 R. Boudreaux, “Open source NotePad++ is a solid contender among text editors,” 2011.

8] [Online]. Available: http://www.techrepublic.com/blog/web-designer/open-source-notepadplus-plus-is-a-solid-contender-among-text-editors/. [Accessed 15 11 2015].

1. PhilKH, “Is Eclipse the best IDE for Java? If not, is there something better?,” [Online]. Available: 9] http://stackoverflow.com/questions/152691/is-eclipse-the-best-ide-for-java. [Accessed 19 11 2015].
2. Denso Wave Incorporated, “2D Barcodes,” 2014. [Online]. Available: https://archive.is/IuLU. 0] [Accessed 16 11 2015].

[2 Marketing VF, Expertmarket, “What is a Franking Machine?,” 2015. [Online]. Available: 1] http://www.ie.expertmarket.eu/franking-machines. [Accessed 29 10 2015].

[2 Janalta Interactive Inc., “Oracle Database (Oracle DB),” 2015. [Online]. Available:

2] https://www.techopedia.com/definition/8711/oracle-database. [Accessed 20 11 2015].

[2 Janalta Interactive Inc., “SQL Server,” 2015. [Online]. Available:

3] https://www.techopedia.com/definition/1243/sql-server. [Accessed 20 11 2015].

[2 Janalta Interactive Inc., “MySQL,” 2015. [Online]. Available:

4] https://www.techopedia.com/definition/3498/mysql. [Accessed 20 11 2015].

[2 Jestro LLC., “Stripe vs PayPal: Who should you choose?,” 2015. [Online]. Available:

5] https://memberful.com/blog/stripe-vs-paypal/. [Accessed 12 11 2015].

[2 DroidLa, “Droid QR Code Scanner,” DroidLa, 2015.

6]

[2 T. Dalling, “What is the advantage of MVC?,” 2015. [Online]. Available:

7] http://www.tomdalling.com/blog/software-design/model-view-controller-explained/. [Accessed 29 10 2015].

[2 admin, “Agile Methodology,” 2015. [Online]. Available: http://agilemethodology.org/. [Accessed 8] 12 11 2015].

[2 Technopedia, “Three-Tier Architecture,” 2015. [Online]. Available:

9] https://www.techopedia.com/definition/24649/three-tier-architecture. [Accessed 24 11 2015].

[3 QuinStreet Inc., “White Box Testing,” 2015. [Online]. Available:

0] http://www.webopedia.com/TERM/W/White\_Box\_Testing.html. [Accessed 24 11 2015].

[3 QuinStreet Inc., “Black Box Testing,” 2015. [Online]. Available:

1] http://www.webopedia.com/TERM/B/Black\_Box\_Testing.html. [Accessed 23 11 2015].

[3 The PHP Group, “header,” 2014. [Online]. Available:

2] http://php.net/manual/en/function.header.php. [Accessed 23 10 2015].

[3 Y. Ryu, “PHP Push Notification Android,” 2015. [Online]. Available:

3] http://www.phpclasses.org/package/8987-PHP-Send-push-notifications-to-Android-and-iOSdevices.html. [Accessed 28 11 2015].

[3 Clickatell, “SMS-Gateway,” 2015. [Online]. Available:

4] https://www.clickatell.com/developers/sms-gateway/. [Accessed 28 11 2015].

[3 Denso Wave Incorporated, “QR Codes,” 2014. [Online]. Available:

5] http://www.qrcode.com/en/qrfeature.html. [Accessed 09 11 2015].

[36] Dzenia, D., n.d. *PHP QR Code* [Online]. Available at: http://phpqrcode.sourceforge.net/ [Accessed 28 Dec 2015].

[37] Faulkner, C. Nov 2015. What is NFC? Everything You Need To Know, Britain: techRadar.

[38] GRNET S.A., 2011-2015. *Conecting to a VM.* [Online] Available at: <https://okeanos.grnet.gr/support/user-guide/cyclades-how-do-i-connect-to-a-vm/> [Accessed 04 Jan 2016].

[39] Heroku, 2016 What it is?. [Online] Available at: <https://www.heroku.com/what> [Accessed04Jan2016]

[40] Kovarik, J., 2015. *Glyphicons.* [Online]   
Available at: http://glyphicons.com/  
[Accessed 15 Jan 2016].

[41] Linux, 2008. *What is cURL?.* [Online]   
Available at: https://curl.haxx.se/docs/manpage.html  
[Accessed 15 Feb 2016].

[42] Microsoft, 2016. *Microsoft Azure Standalone Virtual Machine.* [Online]   
Available at: https://azure.microsoft.com/en-us/documentation/scenarios/virtual-machines/  
[Accessed 23 Jan 2016].

[43] RackSpace, n.d. *MailGun - How to Start Sending Email.* [Online]   
Available at: https://documentation.mailgun.com/quickstart-sending.html#how-to-start-sending-email  
[Accessed 02 Feb 2016].

[44] Stanicek, P., 2002. *Paletton.* [Online]   
Available at: http://paletton.com/  
[Accessed 02 Dec 2015].

[45] TEC-IT Datenverarbeitung GmbH, 2015. *TWedge - Keyboard Wedge Software.* [Online]   
Available at: https://www.tec-it.com/en/download/data-acquisition/twedge/Download.aspx  
[Accessed 01 Mar 2016].

[46] Wambler, S., 2003. *Introduction to Test Driven Developement.* [Online]   
Available at: http://www.agiledata.org/essays/tdd.html  
[Accessed 25 Nov 2015].

[47] ZXING, 2016. *Barcode Scanner.* [Online]   
Available at: https://play.google.com/store/apps/details?id=com.google.zxing.client.android&hl=en  
[Accessed 23 Jan 2016].