City, University of London

BSc Computer Science

Final Year Project Report

Academic year: 2021-2022

‘Penalty Charge Notice Semi-automated Payment System’ Desktop Application Integrating UI Path for Automation Called ‘PCN-Payer’

By:

Shahbaz Syed

Table of Contents:

Contents

[Abstract 4](#_Toc104858278)

[1 Introduction 5](#_Toc104858279)

[1.1 Problem to be solved 5](#_Toc104858280)

[1.1.1 Problems 6](#_Toc104858281)

[1.3 Objectives 7](#_Toc104858282)

[1.3.1 Primary Objective 7](#_Toc104858283)

[1.3.2 Secondary Objectives 7](#_Toc104858284)

[1.4 Project Beneficiaries 7](#_Toc104858285)

[1.5 Work Performed 8](#_Toc104858286)

[1.6 Assumptions 8](#_Toc104858287)

[1.7 Substantial Changes to Project Definition Document (PDD) 9](#_Toc104858288)

[2 Output Summary 10](#_Toc104858289)

[2.1 Computer/Windows Application 10](#_Toc104858290)

[2.2 Database 10](#_Toc104858291)

[2.2 PCN Payer executable file (.exe) 11](#_Toc104858292)

[3 Literature Review 12](#_Toc104858293)

[3.1 Methodology 12](#_Toc104858294)

[3.2 Similar products 12](#_Toc104858295)

[3.3 Programming languages 12](#_Toc104858296)

[3.4 Solution Architecture 13](#_Toc104858297)

[3.4.1 IDE 13](#_Toc104858298)

[3.4.2 Database/ Database Management System 13](#_Toc104858299)

[4 Method 15](#_Toc104858300)

[4.1 Development Methodology 15](#_Toc104858301)

[4.2 Implementation 16](#_Toc104858302)

[4.2.1 Software Used for Design and Development 16](#_Toc104858303)

[4.2.2 Workflow Plan 17](#_Toc104858304)

[4.2.3 UI Path Design – workflow 1 17](#_Toc104858305)

[4.2.4 Database – workflow 2 19](#_Toc104858306)

[4.2.5 GUI (Graphical User Interface) – workflow 3 20](#_Toc104858307)

[4.2.6 Joining the Separate sections of the application and Verification of data. – workflow 4 21](#_Toc104858308)

[4.3 Evaluation 21](#_Toc104858309)

[4.4 Research and Tutorials. 22](#_Toc104858310)

[5 Results 23](#_Toc104858311)

[5.1 Development Methodology 23](#_Toc104858312)

[5.1.1 Pre Development – Tasks 23](#_Toc104858313)

[5.1.2 During Development – Tasks 24](#_Toc104858314)

[5.1.3 Post Development - Tasks 24](#_Toc104858315)

[5.2 Design 24](#_Toc104858316)

[5.2.1 Automation Path Design – unpaid ticket 24](#_Toc104858317)

[5.2.2 Automation Path Design – paid ticket 25](#_Toc104858318)

[5.2.3 Database Design 25](#_Toc104858319)

[5.2.4 Graphical User Interface (GUI) Design – User Interface 27](#_Toc104858320)

[5.3 Software architecture 28](#_Toc104858321)

[5.3.1 Dependencies on Python libraries for Source Code 28](#_Toc104858322)

[5.3.2 Application Navigation Structure 29](#_Toc104858323)

[5.4 Implementation 30](#_Toc104858324)

[5.4.1 Automation 30](#_Toc104858325)

[5.4.2 Database 32](#_Toc104858326)

[5.4.3 User Interface 33](#_Toc104858327)

[5.4.4 Data Validation 34](#_Toc104858328)

[5.4.5 Application run through 35](#_Toc104858329)

[6 Conclusion and Discussions. 36](#_Toc104858330)

[Appendix A. PDD 37](#_Toc104858331)

[Appendix B. Requirements 37](#_Toc104858332)

[Appendix C. 37](#_Toc104858333)

[Appendix D. UI Design 37](#_Toc104858334)

[Appendix E. Project Timeline 37](#_Toc104858335)

# Abstract

With data showing that over 100,000 Parking Charge Notices being issued in Redbridge council a year. And with these coming with deadlines where if paid in the first 14 days of issue the registered owner could save 50% of the charge. If paid after 28 days the registered owner will pay 50% more and this charge going higher and higher until it is paid off. The aim of this project was to design and develop a system in which would make this process quicker and easier so that people who receive these tickets could save more money. The aim was to create an automation that would complete the online payment form quicker than the user. The automation was developed using UI Path and was designed to have a login system where users could save their payment and personal data so that they wouldn’t have to enter it during the process and could use the system repeatedly.

This Report will outline the design and development of this application going into details of the methods used for the entirety of this project and any changes that were made to the design with justification. Any online resource used and any literature used will have also been mentioned below. This report will conclude with the final product and outcome of this project.

# 1 Introduction

Most people who drive a car in London understand the pain when one sees a parking charge notice pop out when checking their posts. Londoners live a busy life if were not at school or at work were out with friends and family. We are all on a grind living life to the fullest. All that momentum and joy shoots down when we get ready to pay the fines we have. Some fines increase in price because were too busy to pay them and forget. What if there was an application that would automate this task. An application where all you needed to do was login and submit 1 or 2 details rather then filling in the whole form over and over again . In the Parking Annual Report published by Redbridge Council 2017/2018 the number of PCNs issued across London had increased by 9.43% to around 5,616,402. In just Redbridge alone the number of PCNs rose from 131,422 to 164,126 in 2017/2018. There may have been a reduction of tickets issued during the pandemic however now that nearly everything is back to normal the trend me continue. Having a simpler way of paying for these tickets fast would benefit the residents and visitors of Redbridge council roughly around 150,000 times a year.

For the purposes of this report the application will be referred to as PCN Payer.

The contents of all headings labelled with a 1… have been copied from the Project Definition Document (PDD) and have been modified slightly. (Appendix A)

## 1.1 Problem to be solved

For the purpose of this project, the implementation will be limited to only The Redbridge Council and only a few options for each payment method, however this will serve as a proof of concept, and may be scaled to larger establishments and entirely different domains.

As people who are busy with our daily lives, we use social media as a form to keep us all connected, but this distracts us from focusing on the task that we have time limits on. As Londoners who drive we understand how frustrating it has become to avoid paying for charges like the Dartford charge, ULEZ daily charge and the congestion charge in central London. Caught up in the busy Londoner life we sometimes park in places where we shouldn’t and end up getting PCN (Penalty Charge Notice) tickets in the post.

A few of us are great at quickly paying it and reap the benefits of paying only half price of the charge but there are many of us who miss the 14-day window and end up having to pay the full price and in some cases 50% or more on top of the original fine. The tiresome task of paying a fine could be avoided if there was an application to do it all for us. Its purpose would be to simplify paying a parking fine so that we could save money by not paying our tickets late. It will also allow us to see where we are getting the tickets and what we can do to avoid them. There are no current Applications that I had found on the market to help users pay for there tickets quicker than to directly pay on the council website. This gap in the market which could benefit from over 100,000 potential use-cases alone in the recipients of Redbridge issued PCNs

### 1.1.1 Problems

|  |  |
| --- | --- |
| Problem | Solution |
| Some people don’t have the time to pay for the ticket | The app will reduce the time needed to pay for the ticket drastically making it easier and simple to pay the fines |
| Paying the fine late ends up costing more | The app can use just a photo to pay for the fine so no need to keep the paper with you and as it is simple and easy as soon as you see it you could pay it under 2 minutes |
| The task is boring and sometimes people just want to put off paying it | The app will allow for quick payment so it would be perfect for people who are very busy |
| No one website for all different boroughs | The app will simplify this through either image recognition of with a simple drop-down tab so all websites in one place. |

To achieve these solutions external / third party technologies maybe needed. An online Information storage system / online database may also be needed.

1. Python’s newest version or same version as the programme must be installed on the user’s Computer / Laptop.
2. Using PIP or manual download, install the following:
   1. Tkinter
   2. MySQL. Connector
   3. Selenium
   4. Chromedriver\_autoinstaller
   5. Re
3. Have the latest version of Google Chrome downloaded and set up.
4. Have MySQL downloaded and setup :
   1. MySQL Workbench 8.0 CE
   2. MySQL Shell
   3. MySQL Installer – Community
   4. Set up the MySQL Database used by the application by running the provided SQL file with all structural setup code on MySQL Workbench 8.0 CE.
5. Have The PCN Payer Setup application downloaded; used to download the PCN Payer desktop application.

## 1.3 Objectives

### 1.3.1 Primary Objective

1. To provide a quick and easy to use application to allow people to pay for their parking fines/ penalty charge notices all in one place.
2. To simplify the process of paying these fines and provide a way for users to see all previously paid for tickets.
3. Remove the need for users to repeatedly input their personal and payment details
4. Email the users a copy of their receipts to confirm the payment.
5. Allow the user to view the automation so they can see the process so that they are certain of the completion of the process and any errors along the way.

### 1.3.2 Secondary Objectives

To provide the functionality of other tasks that could be done on the payment websites through the app. For example:

1. Be able to send an appeal for the penalty charge
2. Be able to see the images of the vehicle in question through the application.

## 1.4 Project Beneficiaries

2 beneficiaries have been highlighted below:

Users will be able to pay their parking fines quicker which would save them their hard-earned money by paying before the 14-day period, so they only have to pay 50% of the fine. Users will be introduced with an application that allows them to pay for any London based penalty charge without needing to look through the documents sent in the post for a website.

The Councils will benefit from this as they will receive their payments quicker. The intended use of this application is to simplify the payment process for users so if they use it during the first 14 days of the issue of the ticket the council will revive this money quicker and may serve as a partial-deterrent to users to reduce the number of times they commit PCN ticket worthy acts.

## 1.5 Work Performed

Throughout the planning, design and development of PCN Payer, the agile methodology was adopted. There were 3 workstreams in which I developed the separate sections needed for the application to work. The first workstream focused on the User Interface (UI) Path which was mainly the back-end part of this application. This was the section that took the longest duration to work on as I had to do a lot of research to make sure that I understood the ‘selenium’ library, I already knew how to code in python but I was very new to the syntax and functions of this and other libraries used. The second workstream focused on the other back-end section of the programme, the database. I had past experience in SQL however setting up the server at data storage tables to work with the UI Path code was fairly tricky. I had started to merge these workstreams towards the end however I decided it would be easier to start the third workstream first as it would be easier and would require less changes. The third workstream was the Graphical User Interface (GUI) which also took time to learn as I wanted most of the code to be in python I chose to use the Tkinter library as this is the standard GUI library for python. Messing around with the GUI code so the correct pages would appear at the correct time and that they were all structured correctly was difficult to manage as the library doesn’t have the same flexibility as working with website design.

## 1.6 Assumptions

I had a different structure in mind for the application as I thought that it would be better to use automation through mobile devices. This was backed up by the fact that there were a plethora of articles and videos on the internet to explain each stage step by step. I had decided to change the platform on which my application was based on from an android application to a desktop/windows application as I realised it would be better if the user could see the automation in front of them as they could see that it was working and correct any errors as it went on. This approach meant that the application would not need any online based servers at the time.

I had also assumed that merging the MySQL database on the local MySQL Server written in SQL with the UI Path written in Python would be hard to merge as I didn’t have any prior knowledge of this and how it was done. The hours spent on research opened my mind to how easy this was and all it required was to install a new python library called ‘MySQL. Connector’ which enabled the python programme to connect to the server and to then send and receive data used to create new users and store user data.

## 1.7 Substantial Changes to Project Definition Document (PDD)

The most substantial change made in comparison to the PDD was that I changed the platform in which the application would run. Initially I had the idea to design it in a way that my laptop would work as a server where a mobile application would send and receive data through the internet but this was tricky as there was less security in this method as users would be sending personal data to an unsecure server through the internet. This was the integrity of data security would not be maintained and for this reason I decided to create a local version of my application where users could download and set up the application in under 5 minutes and use at home their selves. Although this system may seem less useable the idea was that most people would receive the PCN’s at home or would be driving to or from work/home where once they reached they could go on to their laptops/computers and quickly pay for their tickets most people have access to a laptop or computer at home or work. The functionality where users could see the photos taken by the ‘Ticket Enforcement Agents’ would still be maintained as users have a few seconds to have a quick glance at them during the automation where I have coded a very brief pause for the benefit of the user.

Another change made was the amount of functionalities the application provided as my main aim would not have been to the best standard if I was focused on the other smaller parts of the application. Functionalities like an option for users to appeal their PCN’s and or for users to be able to change their details that were stored on the database. I decided that this would all be added If I chose to update the current version and make a version 2.0. I believed that limiting the scope of this project allowed me to focus on the smaller things important for the main functions of the programme like selective data validity checking so that users would not add incorrect data when creating a new account. I had started on some back end functionalities which would mean that only a small update to the python code would be needed to introduce new functionalities; I have added a PCN-ticket table in the database already so that users would only need to update the application and not the local MySQL servers in their laptops.

I also decided not to introduce the feature for users to receive emails quarterly/annually as most people receive less than 12 tickets a year and most of them know where they got their tickets. Instead using the already provided functionality of the council website I decided to add the functionality where once the user paid for the ticket the confirmation would be sent via email to the user provided email when creating an account.

# 2 Output Summary

## 2.1 Computer/Windows Application

|  |  |
| --- | --- |
| Description | The Output of this project source code which could be compiled by most computer/ laptop text editors if python is installed on the computer however I compiled it in IntelliJ Idea Community Edition. Is an computer/Laptop based application which can be downloaded using a provided installer that I have set up for recipients to download and install PCN Payer through but will require the database to be set up for use. The application is an .exe executable file which once clicked opens up the application without the need of any text editor. Only the mentioned libraries and python is required for it to run. This will allow users |
| Output Type | This is the Application consisting of around 660 lines of Python code using 5 libraries (not counting comments and a few spaces in between). The 660 lines includes 11 interconnected functions needed for function of the application. Of the 660 lines of python code around 640 lines of code were written by me and around 20 lines copied using online sources. Research was used to write the remaining 640 lines which have influence from some research sources such as YouTube tutorials. The research / external sources are mentioned in the results section below. The complete version will be submitted via Moodle with all external sources mention here and all files needed to set up and run the Application. |
| Intended recipient | Currently any internal testers who own a windows laptop/computer with the installed application and are willing to set up the MySQL Server using provided SQL set up file. In the future after some tweaks any person who lives in , works at or drives through the Redbridge council. They will benefit from this if they get tickets often as this is a faster and easier way to pay for PCN’s once set up. |
| Results Link | More details can be found : |
| Appendix Link | More details can be found: |

## 2.2 Database

|  |  |
| --- | --- |
| Description | A Database used to save user data for running the automation. The database is altered through the application where the application reads and writes data to authenticate existing users and or create new ones. |
| Output Type | SQL Code; 94 lines of code which was auto-generated by MySQL Workbench 8.0 CE from the existing database I created in sections using the help of a tool in Visual Paradigm 16.3. Which I then altered to fit my needs. |
| Intended recipient | Will be used as part of the data store for the complete windows laptop application. This is needed for the application to run |
| Results Link | More details can be found : |
| Appendix Link | More details can be found: |

## 2.2 PCN Payer executable file (.exe)

|  |  |
| --- | --- |
| Description | An executable file which will after downloading allow the user to set up part of the application. This is so that users will interact with the application as like any other application. |
| Output Type | An executable file that is used to install and set up part of the application. This was generated using a third party application called Inno setup complier. |
| Intended recipient | Will be used as the set up executable file to download the application and partially set up its necessary files on the computer/laptop |
| Results Link | More details can be found : |
| Appendix Link | More details can be found: |

# 3 Literature Review

## 3.1 Methodology

Not having the best experience in using software development methodologies I initially spent time researching the different types of methodologies to see which one would work the best for me and for what I had intended my application to be capable of. I had some knowledge about the waterfall methodology as this was the easiest to understand and followed a step by step process. The methodology has a simple structure and understandable but creates a rigid framework for development and causes some resistance to make any changes during the development of the project.

The Agile development methodology provides more flexibility to change parts of the project that don’t fit in well with what I have intended for the project and also allows me to change and or remove some parts of the project which create difficulties when developing and testing. This cyclical methodology was the one I chose because of its flexibility. The development of this project relied on no tester/user feedback and therefore was mainly focused on improving the application through numerous test to make sure there were no major errors and or problems with the back end.

With the advantages of agile methodology with changes to fit my project needs the methodology allowed me to make numerous changes in the project with every development and test cycle.

## 3.2 Similar products

From my research I could not find any products that had similar functionality to the functionality I had envisioned for my project. I searched for numerous application on the apple store and on the play store and could not find any. I tried looking for web apps as well but they also did not have any similarity in my proposed functionality. Most applications I did find were about paying for parking where the companies either owned or managed the parking spaces. My idea was to create an application which allowed users to pay for parking fines as the councils were still stuck on the old fashioned way of paying by phone or website both requiring users to take a lot more time out of their day to do so. The market was nowhere near saturated with similar products and my product would be one of one.

## 3.3 Programming languages

When developing an application developers have to decide which programming language to use. Mainly Java or kotlin is used for android applications however for desktop applications java is most commonly used. Previous use of java for developing games led me to believe this was the best choice but when researching UI path the modules available in python seemed much more easier to use. Python was much more easier to learn and code in as there was a plethora of YouTube videos and online articles explaining how to use UI Path which was the backbone of my project.

Python has many benefits as it is dynamically typed which means variables don’t have to be assigned, it is automatically assigned a data type during execution and has improved productivity as users don’t have to spend too much time on learning syntax and rather work on the problem at hand. Its vast library support and ease to download and import libraries means that it is almost certain that you could find functions to fit your task. Using pip (Python Package Manager) is really easy and simplifies the exhausting tasks of downloading libraries and changing system variable like java requires users to do.

## 3.4 Solution Architecture

### 3.4.1 IDE

UI Paths can be created using various IDEs and they all work very similarly but as I had past experience with IntelliJ community edition. I chose it to be the one that I would use to develop my code. The coding assistance with its easy to navigate menu feature really stands out but the game changer is where its error analysis comes to play. With just one search with the clear in-depth error analysis the user could find a web article or YouTube video explaining the cause of the problem and how to fix it. Having used this IDE before meant that this was my go to for this project. The internet is mostly unanimous with agreeing that PyCharm is a better IDE to use when programming in Python however ‘stackshare.io’(2022) also agrees that the ‘on-the-fly Code analysis’ of IntelliJ IDEA and its ‘advanced refactoring’ is reason why some people chose it over PyCharm. I had the past experience and really liked the way my IntelliJ IDEA was set up and therefore it was a simple choice at the end.

### 3.4.2 Database/ Database Management System

With the plethora of options available both for online and offline use the choice wasn’t easy at the start. I had some past experience with using MySQL Workbench last year which made me lean slightly in that direction. The main functionality that I required from my database choice was that I could connect my Python code to it and or receive or send data from my python code fairly easily and fast. When researching I saw that MySQL community edition came with its own server application and a few extras which made me think that it was a well-rounded software. Upon Further investigation I realised I was looking for a database management system rather than the database its self and the large number of options out there was making it harder for me to choose one. I started researching through YouTube tutorials and saw that there were many using MySQL Workbench. For that reason I decided to stick with it and started using it for the creation and management of my database.

# 4 Method

## 4.1 Development Methodology

Continuing on from section 3.1 the Agile methodology was beneficial for the project as ‘the agile development methodology minimises risks’(Synopsis.com 2017). As the software was developed in iterations adding new features step by step this meant that I could easily pin point any new errors as I would not move onto the next functionality without completing the previous one. My approach was somewhat different to this methodology as I had started the development of the software with very loose plans as I knew that there were going to be complications where it would be easier to change the requirement rather than to keep working on the complications and thus wasting valuable time.

This is where parts of the Rapid Application Development (RAD) methodology where starting to show. ‘This RAD process allows our developers to quickly adjust to shifting requirements in a fast-paced and constantly changing market’(Synopsis.com 2017). The constantly changing of requirements was needed to be able to produce a high quality application in a short period of time. The application had no user feedback; I had to manage with the resources I had and therefore depended on thoroughly testing and re-testing every section of the application to provide a robust application.

Using Agile Scrum Methodology meant that the workload was manageable as the development of the project was broken down into sprints. ‘Scrum is one of the many types of agile methodology, known for breaking projects down into sizable chunks called “sprints.” Agile scrum methodology is good for businesses that need to finish specific projects quickly’(Businessnewsdaily.com, 2021). Here I was able to develop the application incrementally as I had broken the workload down into 3 separate yet interdependent workflows. The first being the UI Path used for the back end of the application where all the automation would be done. When this was possible incrementally developing the second workflow, the database to accommodate for the information that would be needed to be stored for the automation to work fast. Then when both were working fine and at a stage where no changes were necessary I started to work on the GUI workflow. This was where Increments had to be broken down into smaller increments as small changes could make or break the application. After the application was working I was able to start the fourth sprint where I could add additional features like adding a new account so that the application could be well rounded overall. Each Development sprint lasted an average of around 20-30 days and the last sprint to make final stage tweaks took around 10 days.

These sprints saved me a lot of time as I did not have to waste as much time finding bugs and fixing them as every time I found an error I would simply revert to my last save and work back up from there. When a sprint was completed I was rigorously tested to make sure no errors/bugs were transferred over to the next development sprint.

## 4.2 Implementation

### 4.2.1 Software Used for Design and Development

|  |  |
| --- | --- |
| Software | Why and how it was used during the project |
| GitHub Desktop | GitHub allowed me to keep online versions of my application. During the iteration when I made changes if the application found bugs and I couldn’t fix it I would have an online save which I could retrieve. This also allowed me to see my progress and keep track by giving a title and description to every GitHub push. The ease of use from this application allowed for simple uploads to my private online GitHub account. |
| Microsoft Word | Used to write up this report and document any questions I may have had for my supervisor. Allowing me to keep track of a application to do list where I could amend the list as I went on developing the application. |
| Inno Setup Compiler | Used to create a executable set up programme that would set up the application onto a user’s computer. This would simplify the storage of the application when another person would try to use it. |
| MySQL Workbench 8.0 CE | This was used to create and monitor the database created for the application. I would be able to read and write to the database using this tool and was able to keep track of any writes the application would make to the database. This software allowed me to create and manage the database stored on my local drive and was the only thing I used for database related work. |
| Visual Paradigm 16.3 | This software was used to design the database in the early stages of development where I was making numerous tweaks to the database. This software also was used to generate SQL code used in MySQL Workbench 8.0 CE to create the database as only the structure and dependencies were needed to be made on this software. |
| IntelliJ IDEA Community Edition 2021.3.3 | This software was used to write the python programme needed to create the application . The UI Path was written in python using this software and so was the GUI. The access to the database from the Python script was also made using this application. |
| Google Chrome | This software was used to access pdf’s related to the project like the project brief. This was also used to carry out hours of research into methodologies, design tools, YouTube tutorials and etcetera. |
| Proto.io  / Fluid UI  (Online Websites) | An online tool was used to create design prototypes to use as a guide to how that application should look like. This also had the tools to show which button when clicked would go to which page.  Fluid UI was used to design mobile applications and this was used before I decided to change my application to a computer /laptop based application. |

### 4.2.2 Workflow Plan

The plan was based on the Methodology mentioned is section 4.1. The Idea was that I would start working on the main parts of the project such as the UI and Database as if there were problems with these and I would have to change the structure or language of my application I could do so without going too deep into the project. I started with the plan of having 3 workflows/sprints where I would work on them first however I realised that towards the end of development I had done 4. Each workflow began with planning then research then development and finally testing. These 4 stages were necessary as without the help and criticism of testers I would not be able to catch bugs without thoroughly testing the sections individually. Each sprint varied with duration and in those sprits time spent on the 4 stages also varied. The UI Paths section taking the most time for research and development; whereas the GUI sprint taking the most time on testing and making fixes.

### 4.2.3 UI Path Design – workflow 1

The UI Path of the application was the section that would automate the processes that a normal person would do when trying to pay for a ticket online. This knowledge of UI path was something I spoke to my supervisor about which he had informed me about as I didn’t know that this was the way moving forward. The UI Path is a programming way of creating paths for the script to follow to be able to automate a repetitive process. This way the same task could be done again and again with just one set up automation. Web scraping allows the automation that is set up to extract data from these websites without any manual labour which would take much more time. Using this data like where tickets where issued, the price of the PCN and the other details regarding the PCN could be extracted to the python script and be used for the application. The benefits of web automation are that it is cost-effective, easy to implement, low maintenance and speed, accurate with data and an effective way of managing data(raluca-p.medium.com, 2021) .

#### 4.2.3.1 UI Path Research

##### 4.2.3.1.1 Decisions

I started off with researching which was the best language to write the application in and which language would be the easiest for me to learn the new syntax and manage to build the project without spending too much time on learning the language. The 2 options were Java and Python and from my research I had found that Python had a dedicated Library which was widely used for web automation and that It would have been a very cost effective way of creating the automation side of the application. I had hesitations with python and when it came to making a GUI for the application however Python also had a library for that as well which I had later found out. For these reason and what is mentioned above in section 3.3 I decided that Python would be the main language I would use for coding and that there would be less issues than coding in multiple languages.

##### 4.2.3.1.2 Selenium Library

From my research I had gathered that the best way to use web automation through python was to use a python module called Selenium which worked very simply. The way It was designed was that you could tell a python script to find something on a webpage by just using the ‘XPath’ of that specific thing. This was easy to do using the ‘developer tools – inspect elements’ tool on google chrome. From finding the XPath I could either retrieve data, type into a input box or click buttons. With this simple tool I could learn the process manually first and then begin to automate it. ‘Selenium is a strong set of tools that firmly supports the quick development of test automation of web applications’ (geeksforgeeks.org, 2022). Using this to my benefit I used the tools provided to automate the entire process from entering the ‘PCN Number’ to the point where after the ticket is paid for the council website would send the receipt via email.

##### 4.2.3.1.3 Redbridge Council Website.

After understanding the tools and techniques I would use to automate the website I needed some time to study the website and how it worked. Using tickets paid for in the past and the new one that I had received I repeatedly went through the process marking down which button to click and where to write the details so I could have a written order before I started to programme it.

#### 4.2.3.2 UI Path Development

When I started the development of the automation I realised that retrieving data like the contravention details and imaged were pointless as I wanted the user to see the entire automation in front of them. This way the user could see the images and the details of the PCN charge that differed from the physical notice that they would have at hand. The second problem that arose was the authentication of payment that if this had failed there would be no way of the user knowing as different cards work differently and sometimes require users to authenticate the payment using their mobile devices. From this I realised that I had to put a stop in the automation if an error like that arose as this would then allow the user to work on the reason at their own pace and then let them continue the remained of the process manually as this would make it easier for the application script and also the user as they would not have to fight the automation to stop and allow them to change any details. As I had mentioned in my PDD I wanted to introduce ways in which the user could also appeal the claim but I chose not to do this as this would expand the scope of the project making more room for errors and the main functionality of the application. I also decided not to integrate image recognition as this would mean that either the application would have to become a mobile one or that the user would have to upload photos onto their laptop. Either way increasing the number of steps which would go against the whole point of the application which is to make the boring process automated so that it would be faster than doing it manually.

#### 4.2.3.3 UI Path Testing

After writing the complete script for the automation I ran multiple tests to develop the code more. Tests like wrong PCN number and wrong car registration number were some of the first test I did to see if the application could handle it. As I had not coded the error handling I had to spend more time researching how to handle the errors thrown which would terminate the automation. After those bugs were fixed I started working on the data validation section of the automation and then I had realised that this could all be done when users would make their own accounts so no to put much strain on the automation itself I should validate the data entries from the user before they went into the automation.

### 4.2.4 Database – workflow 2

#### 4.2.4.1 Database Design

At the start of the project I started off with noting down the information needed from the user before the automation started so that there would not be any need of interruptions during the automation. I then began to focus on the login aspect of my application and what details from the user would be needed then. This meant that if the user was able to login to access their private details I could have multiple users on the same computer so that multiple people could use the application. Before commencing on development I used software Visual paradigm to design the database as mentioned in section 4.2.1 .

#### 4.2.4.2 Database Development

Using the design software I could understand and create the connections between the tables in the database and then using the tool provided turn the diagram into SQL code which I would run in MySQL Workbench 8.0 CE to create the database. After creating the database I ran some test data through the database using the SQL file editor provided so that I could see how data would be saved in the database and if the correct connections were being made.

#### 4.2.4.3 Database Testing

I had finished the development of the database but know needed to focus on testing the data entry points. U sing the tools provided I changed the data type to accommodate for integer entries that would start with zero as the database would accept a number like ‘012’ and make it into the number ‘12’ cutting out the ‘0’. The simplest fix for this was to enter the data like a String as so that the database would not change the entries. Other tests where done to make sure that the database would be able to find user details correctly or not to which there were no issues. I also needed to save the syntax for data entry and data retrieval as this would be needed for the data read and write section of the automation.

### 4.2.5 GUI (Graphical User Interface) – workflow 3

#### 4.2.5.1 GUI Design

Using the tools mentioned above in section 4.2.1 I used Proto.io to create prototypes of how the applications should look. The Idea was to use simple colours and a clear layout for the structure of the application’s GUI as this would not distract the user from the intended purpose of the application and not to overcomplicate it. My aim was to have max 4 pages where all functions of the applications would be present on which would provide a fast and simple use of the application to the user. Using the online tool I prototyped the 3 pages that I thought were necessary and then worked my script around that so that I would be very similar to those designs with only making necessary changes where it would benefit the user or make the script simple.

#### 4.2.5.1 GUI Development

I started with the prototypes and then found tutorials online on how and what to use when coding the applications GUI to its backend. I found a few YouTube tutorials where the GUI was made using a Python library called ‘Tkinter’. Using Tkinter I found that the library was straightforward and simple to use. I had to avoid going overboard with the GUI so I stuck to the prototype designs I had made earlier. From here onwards I designed the second page as well on a different file as I was yet to learn how the buttons communicated with the script and for this reason I started on the third as well.

The pages were used as follows:

1. Users Login with Email Addresses and Passwords with access to make a new account page.
2. Users input PCN number and car Registration also with access to make a new account page.
3. Users can create new account with a form to fill in with personal and payment details.

From here I realised that there was much more coding to do for the verification of the details the Users would input and how the users would know to check which details were wrong without redoing the entire form again.

### 4.2.6 Joining the Separate sections of the application and Verification of data. – workflow 4

#### 4.2.6.1 Connecting GUI to Python Script

From The point all the GUI pages were done I started working on connecting the GUI pages to the script by implementing the communication between the GUI/Script and the database. This was to avoid writing more and more lines of code I would integrate functions into the buttons to communicate with the database through and back to the script. Realising that this was the easy park as all the communication was to send a SQL request to the database and compare the details the users wrote with the details in the database in the script itself. After all the buttons were set and all the functions were made I started with testing to see what user input details needed which type of verification.

#### 4.2.6.2 GUI / Python Script – Testing

During the testing of the application I realised that there was a lot more Python script needed to verify the user given data and secondly a lot more code to show to the user that the details were wrong. I started this by changing the stucture of the GUI so that it could accomodate for boxes that would highlight correct and incorrect details given to the user in comparison to the format required for the automation. From here I wrote more functions to verify the data when users created new accounts so that when the automation ran it would do so with correct data. More tests were run to see if there was a need to verify the data better than how it was and accordingly changes were made. These quick tests were the best way of developing this projects source code as the incremental changes that the Agile Scrum Methodology allows for made it that there were no major errors towards the completion of the source code and that there were improvements made overall this was also mentioned in section 4.1 .

## 4.3 Evaluation

During the development of this project the code used for connecting the Python script to the database was copied from an online YouTube tutorial and this was because I was struggling to understand the connection ports and code needed to connect and read and write data.

Overall the Scrum Agile methodology worked fine for me however I worked with it fairly loosely as mentioned in section 4.1 I decided to follow a mixture between RAD methodology and Agile methodology as I wasn’t releasing sprints that worked independently rather was creating sections of the code incrementally with multiple test in-between to reduce the occurrence of bugs. The RAD Methodology came in hand towards the end where I had to change the stucture of the GUI to accommodate for data validation and also at the beginning where I had differed from my PDD where I decided to create a computer/desktop based application rather than a mobile application so that the user could simply see the automation. The stucture of how the system as a whole was also changed as I did not have to make an online database for a mobile application so these changes of requirements were allowed through the RAD Methodology.

## 4.4 Research and Tutorials.

All Tutorials used throughout the project are mentioned below in the research link and where they have been used have been mentioned above in section 4. Where code has been copied from Tutorials it has been mentioned above and will also be found in the research link below. The time code was copied was where I was struggling to connect the script with the database for date transfer.

# 5 Results

## 5.1 Development Methodology

During the planning and development of the project I had smaller to do list written down which would be used as dynamic requirement list where changes were made to fit the intention of the project. This is because before the project has started understanding exactly what the programme requires is not easy as after further research and understanding new techniques and skills are picked up to enhance the project. This was complimented by mainly using the Scrum Agile Methodology with aspects of RAD Methodology. The main aspects of these 2 methodologies that I needed was the ability to change designs and requirements on the go and have a constant cyclic development process where after developing a certain amount of Python script I would go back to thoroughly test it to avoid any errors coming up later on. By pairing these 2 methodologies efficiency was increased and the lack of testers was not that problematic.

Using the Scrum Agile methodology as it is intended to be used would then later allow me to make improvements and structural changes for a updated version release whilst having a complete product already ready to use.

### 5.1.1 Pre Development – Tasks

The intention was to start with a plan in place as this would provide structure and improve of efficiency as I had to plan the meetings with my supervisor and the deadlines I had been given.

Here are the steps that I had taken to ensure that I would be well prepared before development of the project mentioned briefly in section 4.2 :

1. Gathering the research that I had on the database software and automation software.
2. Using the council website understand the data needed from the user.
3. Design the database data structure accordingly.
4. Splitting the workload into 3 workflows to have a better plan of action.
5. Create a workflow requirements list for the automation process and plan the order of the actions.
6. Create a workflow requirements list which would have space for changes for the database.
7. Make prototypes for the GUI to have a better understanding of how application will look.
8. Create a workflow requirements list for the GUI using the prototyped design I made earlier.
9. Gather and save the research that I had used and was planning on using for the report and to be able to look back on for any help.

### 5.1.2 During Development – Tasks

1. Start working on the requirements list for the automation.
2. Using the automation re run valid details to ensure no bugs
3. Run tests on incorrect data entries and note down for improvements in next work flow
4. Make database and test if data read and write is possible.
5. Connect script to database and run queries.
6. Incorporate some communication between the automation and script when running the automation.
7. Work on the GUI Code separately and develop the pages first.
8. Create functions for the buttons to work when GUI is incorporated into the automation script.
9. Join the GUI/database script with the automation/database script and connect the automation to start when initiated by GUI/Database code.
10. Improve of validation of the new user data entries to remove any data validation when accounts are in use.
11. Re run tests and write down any bugs needed to be fixed for next workflow.

### 5.1.3 Post Development - Tasks

1. Test application for any bugs.
2. Improve on any verification of data entries .
3. Develop any page of the GUI for ease of use for the user.
4. Make notes on any changes made and improvements that could be made in the future.

## 5.2 Design

The design of this project was split into 3 sections. One was the design of the GUI which was how the application would look like to the user. The other was to the design of the database which would not be seen by the user but would store the user data. The third was the design of the automation and which path the automation would take to complete the payment for the PCN. Designing these 3 sections were important as they allowed me to plan and execute them with ease as I already had an idea of what was needed from the application. Changes could have been made for these designs however only a few were made when deemed necessary.

### 5.2.1 Automation Path Design – unpaid ticket

1. Start on Redbridge Council Pay PCN website = 'https://my.redbridge.gov.uk/parkingpcn/'
2. Enter PCN Number in PCN Number textbox
3. Enter Vehicle Registration in ‘vehicle registration’ textbox
4. Click on Pay my PCN button
5. Click on ‘pay for your pcn’ button
6. Click on ‘pay for your penalty charge notice online’ button
7. Click ‘checkout now’ button
8. Enter Name of cardholder in ‘name of cardholder’ textbox
9. Enter Card number in ‘card number’ textbox
10. Enter Card expiry date
11. Enter CVC number in ‘cvc’ textbox
12. Click on ‘continue to checkout summary’ button
13. Click on ‘confirm payment’ button
14. Enter email address in ‘email address’ textbox
15. Click on send email button

### 5.2.2 Automation Path Design – paid ticket

The process is the same up to point 4 and from there onwards there is no button to move forward in the process as the status would say fully or partially paid with contravention photos and details, maybe even a CCTV recording.

### 5.2.3 Database Design

The database was designed using the requirements from the research into the council website and what data entries it required. I also used some of the requirements for the automation to design the database as I needed to store email addresses and passwords for users to login. I had also left space for future upgrade by introducing a tickets table where I could have a page showing the user the past tickets they had paid for.

The Entity Relationship Diagram below shows the 1 to many relationship that the accounts table has to the number of payment details; This was done on purpose to allow users to have more than 1 payment card in future updates.

The one to many relationship between the accounts table and Ticket tables shows that one user could have paid for numerous tickets and this table was made for the future expansion of the application where users could see a list of all tickets paid for through that account. Although the Diagram shows that the accounts table has a one to zero or more relationship with the payment details table this is in practice as a one to one or more relationship as a user cannot make an account without payment details but can make an account.

I wanted to keep the database simple for now as not to complicate myself and focus on any changes / upgrades in the next update when I chose to do so. This was to limit the scope of the project so that I could do the main requirements better with no time wasted on fixing bugs.

Graphical user interface, diagram

Description automatically generated

### 5.2.4 Graphical User Interface (GUI) Design – User Interface

The GUI was developed on top of my initial Idea to develop an android application. The design for that was more towards a mobile application look which I thought wouldn’t look the same for a desktop application. For this reason I redesigned the original design to a much more simple yet clear design so that I would be less distraction and very easy to use. This would make the application more popular to use and would in no way waste more time than paying for a PCN manually.Graphical user interface, application

Description automatically generated

This was the original design for the android application however the changes I made to the design makes the application look more tailored to a laptop/computer based application. The android application design was made on Fluid UI

The application / GUI design was made on proto.io and this was better for desktop application design. The mobile application was used as a base to improve on. The light ‘baby’ colours stood out making the application look easy to use and simple as well as the small window size and low number of buttons and textboxes emphasized the ease of use factor for this application.

Chart

Description automatically generatedGraphical user interface, website

Description automatically generated

Page 2 – after logging in users can start automation by giving pcn number and vehicle registration number

Page 1 – used for logging in and or creating a new account

Graphical user interface

Description automatically generatedThe design of the application was made between the colours light blue and light pink. This was to lighten the colour and put more focus on the automation rather than the application. The light blue and light pink contrast each other like ‘blue and red’. I designed all pages to have an exit button so that they could terminate the application whenever they felt like however decided that the new user account page’s exit button would only exit that page as to make the user go back to the main menu.

Page 3 – used for creating a new account so that users can use the automation.

The plan was to develop the GUI in Python using the Tkinter library which would have enough customizations to create pages like these.

The GUI was entirely designed in proto.io an online tool to design applications and was developed in its entirety on IntelliJ.

I learned through my research on Microsoft website that ‘A great desktop application is powerful and, at the same time, simple.’(docs.microsoft.com,2021) and this is why one of my main focuses was to simplify everything at try to declutter as much as possible. I learnt on this website what makes applications powerful and what makes the user experience simple.

## 5.3 Software architecture

### 5.3.1 Dependencies on Python libraries for Source Code

The source code in its entirety was written by me however there were libraries that I had used mentioned in section 4.2.3 and section 1.1.1 . Without these libraries the source code will not work and therefore I have included them as imports at the top of the source code and listed them in section 1.1.1 .

I used PIP or manual download to install the following as the script is dependent on these libraries:

1. Tkinter – For GUI .
   1. \* = Everything – this was used for making the GUI / User Interface
   2. Messagebox from Tkinter – used to allow pop ups in the application .
2. MySQL. Connector – To connect the Python script to the database.
3. Selenium – this was used for the web automation.
   1. Web Driver – allows the automation to take place on Google Chrome .
   2. By – used to find textboxes and buttons on the website using the ‘XPath’ of that object and in turn allows the script to input data to the website textboxes.
   3. ‘NoSuchElementException’ – used when an element can’t be found the script won’t cause a crash and the application will continue to run smoothly.
4. Chromedriver\_autoinstaller – this makes sure that the correct Chrome Driver is installed to allow the automation to run.
5. Re – this was used for data validation when a user would input data to the application.

The model of this application doesn’t really fit into a set structure as the First window is the main window/First page which when a user logs in changes to the Second page. There was no need for a logout function as a user could just restart the application. The third page was a secondary window which had no impact on the first window. This could be opened from either page 1 or 2 and when closed did not close the primary window regardless of if it was on page 1 or 2. However if the secondary window was open and the user terminated the first / Primary window using the ‘exit’ button the whole application would be terminated. The purpose of this is that a user could change their mind on making a new account or not however they can’t change their mind on using the application or not as they would have to restart it afresh. The reason I designed the application like this was to improve fluidity and transfer of data between function rather than making all data global and having pages access that data. The pages only had access to data set to them and not to anything else. This way the use of data is broken down and distributed.

### 5.3.2 Application Navigation Structure

Diagram

Description automatically generatedThe Diagram below will outline the navigation between the pages. The order was not kept as normal where a user can go back and forth between pages as to disrupt the flow of the application and therefore the application takes on a more unconventional forced application flow. This builds on the simplicity of the programme and its east of use. The Diagram on the left shows the cyclical structure the end of the application takes as a user could pay for multiple tickets one after the other and if the user entered wrong ticket details they could run the automation again. The diagram shows that a user could make a new account from either page 1 or 2 and the pop up window to make a new account will appear from here a user could terminate the new user account window or complete it and go back to where they left off from.

## 5.4 Implementation

This Section will outline the implementation of the project, how the source code was developed, any problems I came across, any major decision made, code copied from external sources, an in-depth overview on how to run the application and any information needed to understand the development and or use of the application.

### 5.4.1 Automation

Using IntelliJ IDEA Community Edition 2021.3.3 I wrote the source code for the Automation without starting on the development of the GUI or the Database this was because I wanted to understand how the automation will work so I have an idea on how I would build on it to add the GUI and access to the database.

#### 5.4.1.1 Setup

After downloading Python I downloaded the IDE. The IDE was IntelliJ Community Edition 2021.3.3. Using the Python Package Manager (PIP) I installed the libraries I needed for my python script. PIP handled the environment variables and path for me so I don’t have to do all that manually.

Using PIP I installed pyinstaller, selenium, chromedriver-autoinstaller, Tkinter, RE, mysql-connector and mysql-connector-python. This has also been mentioned above in section 5.3.1 , 4.2.3 and 1.1.1 .

#### 5.4.1.2 Python Project Setup

Using IntelliJ I followed the project wizard to make myself a python project. From here I started by creating a python file called Redbridge.py where I would start coding the automation.

#### 5.4.1.3 Automation Script

driver = webdriver.Chrome()  
driver.get(startURL)#startURL is start website set at the top  
inputPCNBox = driver.find\_element(by=By.XPATH, value='//\*[@id="Pcn"]')  
inputPCNBox.send\_keys(x)  
inputRegNoBox = driver.find\_element(by=By.XPATH, value='//\*[@id="Vrn"]')  
inputRegNoBox.send\_keys(y)  
searchButton1 = driver.find\_element(by=By.XPATH, value='//\*[@id="formsubmitpcnsearch"]/div/div/div/div/div/div/div[3]/input[2]')  
driver.implicitly\_wait(1)  
searchButton1.click()

The snippet of code from the automation above shows how to start the automation on a website the driver is needed and it starts with a given website which I had set above. The inputPCNBox is a variable which would store the location of the text box so that using ‘.send\_keys’ I can input data into the website. The ‘driver.implicity\_wait(1)’ makes the automation wait; this is necessary as the website has delays and when clicking on buttons to fast the automation would halt. The ‘.click’ allows the script to send clicks to specific buttons which allows the automation to move forward onto the next page / section.

try:

driver.find\_element(by=By.XPATH, value='//\*[@id="Pcn"]')

except selenium.common.exceptions.NoSuchElementException:  
 pass

Code like this with a try except clause and the script embedded inside of it was a very important as this was the only way to manage the automation where some parts of the website were missing. The ‘NoSuchElementException’ would come up when a certain element is not present on the website but according to the automation path it should be there. This would happened when the user would input the wrong details and instead of the page moving onto the next one the webpage would throw an error. The automation would look for textboxes and buttons on the next page but as the page was the same and those buttons were not found the application would throw this error. This is why these try clauses were used to prevent any halts and allow the automation to run smoothly.

The rest of the automation would follow this trend of data entries, button clicks and try clauses for common errors that could be cause by wrong data entries using the users given incorrect details.

Common errors like this are:

1. Incorrect PCN Number
2. Incorrect vehicle Registration
3. Incorrect Format for any formatted data entry for example card number or card expiry month, 16 or 2 digits only.
4. Incorrect card details
5. Ticket already paid for
6. Expired card
7. Missing data entry

#### 5.4.1.4 Automation Error Checker

The error checker function would compare the error with a few pre set errors with pre set outcomes however if the error checker did not know the error then it would show that error on the application. There was an error that was repetitive and this error did not fit in the window so I amended the code so that if that error was found the window would change dimensions and become wider. The automation would halt and the user could continue and make changes on the website if they choose to do so.

### 5.4.2 Database

As the database was generated using the entity relationship diagram I had made in visual paradigm there wasn’t any need for me to write any code however for anyone else to use this application they would need to set the database on their own in their own local server and local drive therefor using MySQL Workbench 8.0 CE I used a tool to generate SQL code which would make the tables and structure for any user who choses to use this application for testing or for their own personal use. This source code for the database will be shown in the appendices and will be given with the project submission.

#### 5.4.2.1 Database connection

Using mysql-connector mentioned in section 5.3.1 and 1.1.1 I was able to build communication between the python script and the database that I was managing using MySQL Workbench.

mydb = mysql.connector.connect(host="localhost",  
 user="root",  
 passwd="S3y5e3d5",  
 database="pcndb",  
 auth\_plugin="mysql\_native\_password")  
mycursor = mydb.cursor()  
mycursor.execute(  
 "select Account\_ID from pcndb.accounts where Email\_Address = '{}' and Account\_Password = '{}'".format(email, password))

Using the code above I could connect to the database and from there I could send queries to the database and use the results in the python script. ‘mycursor’ would allow me to send queries to my database the one that was accessed through variable ‘mydb’.

for x in mycursor:  
 # print('////////')  
 accountID = x

From there I could get the results back into the script and use them to either validate user login details and or save them for use in the automation.

mycursor.execute(  
 "INSERT INTO pcndb.accounts(First\_Name, Second\_Name,Email\_Address,Account\_Password, Phone\_Number, Postcode, Door\_Number, Street\_Name, County, Country) VALUES ( '{}', '{}', '{}', '{}', '{}', '{}', '{}', '{}', '{}', '{}');".format(  
 fn, sn, ea, ap, pn, p, dn, streetname, county, country)  
)  
mydb.commit()

Using code like this I was able to send data to the database and then using .commit I could save the database to store the new data that I had just sent.

### 5.4.3 User Interface

Using the designs that I had made for how I wanted the application to look like I used the Tkinter library on Python to create something very similar for the pages.

root = Tk()  
root.title('Parking-Charge-Notice Payer')  
root.geometry("475x400")  
root.configure(bg="light blue")

I started off with a simple light blue window with small dimensions of 475 x 400 and a window title.

emailLabel = Label(root, text="Enter User Email: ", bg="light blue")  
emailLabel.grid(row=0, column=0, padx=25, pady=10)  
emailTextBox = Entry(root, width=50, borderwidth=4)  
emailTextBox.grid(row=1, column=0, padx=25)

From there I started adding labels and text boxes where data would be enters by the user.

loginButton = Button(root, text=" Login ", width=15, command=loginFunction, fg="black", bg="light pink")  
loginButton.grid(row=5, column=0, padx=25, pady=15)

Graphical user interface, application

Description automatically generatedGraphical user interface, application

Description automatically generatedUsing the buttons to run functions and textboxes to take in data entry all the tools I needed where done. All I had to do was set the page up correctly using the grid system and the pages were ready to connect with the automation. Using these simple tools I was able to build an application.

A picture containing bar chart

Description automatically generatedAs there was much more details required from the user I felt that it was better to have all the textboxes on one page so I changed the size of this pop up window to accommodate. The benefit of all this on one page makes it clearer and simple to use as for the user there is only 1 page were all details are required for the setup and once that is done the user will never need this again.

### 5.4.4 Data Validation

Using the RE library I was able to custom make the validation code for the different data entries needed in the new account set up page.

if x == "Email\_Address":  
 pat = "^[a-zA-Z0-9-\_]+@[a-zA-Z0-9]+\.[a-z]{1,3}$"  
 if re.match(pat, y):

using code like this and tutorials online I was able to set up formats for specific data entries and then using Boolean expression if a data input was wrong let the user know my changing the User interface to let them know what data entry was wrong as shown in the image below. The example below shows what will happen if the data entry is incorrect

The second image will show how it will appear if the data entry was correct.

Graphical user interface, text, application

Description automatically generatedIf a user tries to make an account using an email that has already been given a pop-up will appear.

This would say something else if the new account was made.

### 5.4.5 Application run through

The application will start and will prompt the user to login with their login email address and password. Assuming this is their first run of the application they would click the ‘New User’ button and go onto the new window that has opened. From here the User will fill out the textboxes with correct details and click submit. This will prompt the application to open a text box with a confirmation or an error and tell the user if their account has been made or not. From here The User would enter their login details and go onto page 2 by clicking the login button. On this page the user will enter their PCN Number and their car Registration and the automation will begin. If there are any errors during the automation it will halt allowing the user to make manual changes to the details and continue manually and or re run the automation. A Video of this whole application will be submitted with the application source code and this report.

# 6 Conclusion and Discussions.

To conclude this project I would like to say that as I have been able to produce a working product I believe that I have been fairly successful at the level the project has been made to. The main objective was to create an application of some sorts that would automate the process of paying for a PCN Ticket issued by the Redbridge Council. I have been able to achieve this in my project however critically speaking I was not able to achieve some of the secondary objectives to the standard I believed I could do it at. Structural changes were made to the application platform, design, and to the way the application interacted with the database however this still enables the outcome that I had aimed for. The project demonstrates the use of UI Path automation, use of databases, database communication with python script, developing GUI for applications and all in all producing a working application that does what its intended purpose is.

To be realistic It is far from being put on the market and there are many upgrades I would like to do on it in the future. Some changes would consists of more secondary objectives such as they ability to change existing account details. Improve the cosmetic look of the applications GUI. Change the Structure of the application to have an online database with encryption. All of these things are Improvements I would like to make to the application in the future. Seeing as there are no competitors in this market It would be possible for me to expand the interests of the application to pay for PCN’s on other Council websites and let the user select from many options. Or maybe allow users to be able to pay for congestion charge and or Dartford charge directly from this application. This application could allow the user to only make an account on this application and not on every council website to have their details saved.

Looking at myself I have learnt many new skills whilst working on this project. I have learnt to research better, plan earlier, design applications and commit all my time and effort to a single project. I have learnt to manage my workload and manage my own project whilst juggling other modules at University. I faced issues with energy and illness but managed to get the work done. I managed stress and anxiety but persevered through for the sake of the project and believe I learnt more about myself and what I am capable of rather than the knowledge gained during this project.

### 6.1 Primary Objective Checklist

|  |  |
| --- | --- |
| Primary Objective | Result |
| To provide a quick and easy to use application to allow people to pay for their parking fines/ penalty charge notices all in one place. | Completed |
| To simplify the process of paying these fines and provide a way for users to see all previously paid for tickets. | Fail – Users can not see past tickets on application |
| Remove the need for users to repeatedly input their personal and payment details | Completed – integrated database for data storage |
| Email the users a copy of their receipts to confirm the payment. | Completed – using the council website to send email as part of automation |
| Allow the user to view the automation so they can see the process so that they are certain of the completion of the process and any errors along the way. | Completed – using webdriver to achieve a automated google chrome window. |

# Glossary

|  |  |
| --- | --- |
| Abbreviation | Definition |
| PCN | Penalty Charge Notice |
| IDE | Integrated Development Envirenment |
| GUI | Graphical User Interface |
| UI Path | User Interface Path |

# Appendix A. PDD

# Appendix B. Requirements

# Appendix C.

# Appendix D. UI Design

# Appendix E. Project Timeline

# 