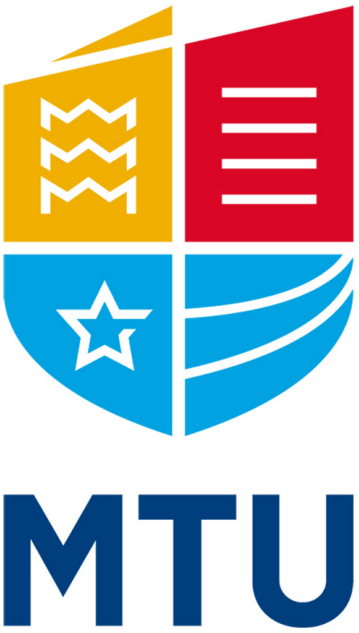

Intercepting Survey Click Fraud using Deep Reinforcement Learning

by

Gearóid Sheehan

This thesis has been submitted in partial fulfillment for the degree of Bachelor of Science in Software Development

in the
Faculty of Engineering and Science
Department of Computer Science

May 2021

Declaration of Authorship

I, Gearóid Sheehan, declare that this thesis titled, 'Intercepting Survey Click Fraud using Deep Reinforcement Learning' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for an undergraduate degree at Munster Technological University Cork.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at Munster Technological University Cork or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this project report is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: 

Date: 17/05/2021

MUNSTER TECHNOLOGICAL UNIVERSITY CORK

Abstract

Faculty of Engineering and Science
Department of Computer Science

Bachelor of Science

by Gearóid Sheehan

The anonymity offered by an online presence is frequently used by individuals looking to increase their own capital gain through various methods of fraud.

While click fraud can be committed by anyone with a computer, more sophisticated methods have also been developed over the years. Often utilizing social media as well as cheap labour in developing countries, many fraudsters are creating swarms of fake online profiles for use in 'click farms'. These click farms can be used to initiate survey fraud, social media page growth, advertisement tampering and even election swinging. They are utilized for either the direct gain of the controller of the farm, called the 'farm master', or else hired out to complete services for others, often with prices as low as 1 dollar for 1000 'clicks'.

Survey fraud in particular is often a target of click fraud due to the payload offered for the completion of certain surveys. In the realm of online surveys, a number of companies currently exist providing services for the creation and hosting of surveys for casual surveying. Currently, very few of these companies offer protection from malicious entities. While other companies which solely provide this protection exist, their services are designed for large scale data collection and require previous knowledge of market-research. This renders them completely unsuitable for casual surveying. The need for protection of data collected from casual surveys should be a high priority no matter the size of the survey. Institutions such as work places and universities frequently collect data using casual surveys. These surveys can easily be targeted by click fraud, where the surveys are completed untruthfully for either payload gain or to just tamper with the collected data. The resulting data collected is fraudulent, and for surveys offering a reward payload their budget is depleted on bad data.

This project involves the creation of a web application which provides a straight forward survey creation and hosting platform. The solution entails a full-scale web application, where users of the application can register and login to their accounts, and begin creating survey projects for hosting. Most importantly, each survey is protected from malicious entities using a deep reinforcement anomaly detection algorithm, flagging surveys which it deems to have been answered untruthfully as well as documenting the survey performance. All statistics from the users past and present survey projects are available to the user via detailed graphs for further use.

The solution is to be built using a mixture of technologies. These include Angular framework for the client-side application, ASP .NET Core for the back-end and Microsoft SQL Server Management Studio for the database. For the cloud-platform services AWS will be used, and in particular AWS EC2 for the hosting of the application and database, with AWS S3 Buckets being used for the hosting of surveys. All machine-learning will be done in TensorFlow and will be hosted on AWS Sagemaker using the Tensorflow SDK.

Acknowledgements

This project would not have been possible without the help and support from a number of different individuals. Firstly, I would like to thank both of my project supervisors - Dr. Mubashir Husain Rehmani and Cliona McGuane. Their knowledge and advice was hugely beneficial in both the research and implementation phase of this project, and I consider myself very lucky to have benefited from having them as my supervisors.

Secondly, I would like to thank all the team at the Cork city office of OpinionRoute LLC, where I completed an eight month intern-ship from January 2020 until August 2020. I greatly appreciate all the efforts the team put into furthering my education in the field of software development. In particular I would like to thank the director of engineering Gavin Wills, who presented me with the base idea for this project, and Tony Moffat, who was always at hand with advice.

Lastly, I would like to thank both my loving parents who have supported me through and through, guiding me into adulthood and and giving me every opportunity possible to succeed in life ...

Contents

List of Figures

List of Tables

Abbreviations

AAD	**A**ctive **A**nomaly **Detection**
API	**A**pplication **P**rogrammable **I**nterface
AWS	**A**mazon **W**eb **S**ervices
BC	**B**efore **C**hrist
CORS	**C**ross **O**rigin **R**esource **S**haring
CSS	**C**ascading **S**tyle **S**heets
CSV	**C**omma **S**eperated **V**alues
DQN	**D**eep **Q** **L**earning
ER	**E**ntity **R**elationship
HTTP	**H**yper **T**ext **T**ransfer **P**rotocol
IEEE	**I**nstitute of **E**lectrical and **E**lectronic **E**ngineers
ICCC	**I**nternational **C**onference on **C**ommunications in **C**hina
IFPC	**I**nternational **F**raud **P**revention **C**onference
IDE	**I**ntegrated **D**evelopment **E**nvironment
LINQ	**L**anguage **IN**tegrated **Q**uery
MIT	**M**assachusetts **I**nstitute **T**echnology
QAAS	**Q**uality **A**s **A** **S**ervice
REST	**R**epr**E**sentational **S**tate **T**ransfer
SAAS	**S**oftware **A**s **A** **S**ervice
SCSS	**S**assy **C**ascading **S**tyle **S**heets
SDK	**S**oftware **D**evelopment **K**it
SQL	**S**tructured **Q**uery **L**anguage
TPS	**T**oyota **P**roduction **S**ystem
UML	**U**nified **M**odeling **L**anguage
US	**U**nited **S**tates

Dedicated to my family.....

Chapter 1

Introduction

1.1 Motivation

The integrity of data gathered from online surveys is constantly under threat due to fraudulent activity. Both click fraudsters and in general dishonest users are taking advantage of paid surveys to make money online.

Much of the time answering in a systematic manner, they complete the survey without reading or answering the questions truthfully so they can claim the reward as fast as possible before moving to the next survey. Often lying about details such as their credentials, these malicious users sometimes access online surveys which are intended to gather data from individuals with a certain kind of expertise, such as a doctor, due to the larger payload.

For the first eight months of 2020 the author completed an intern-ship with market research company OpinionRoute. Based out of Cleveland, Ohio, they are the leaders in insights process management. One of the services provided by them is survey data protection, through technologies such as profile screening, IP blacklisting and duplicate identification. While learning the technicalities of market research and fraud detection there, they detected a gap in the market with regards survey fraud protection. There is currently a range of web applications which offer survey creation and hosting facilities for casual surveying, however there are very few that also offer protection services from both click-fraudsters and in general dishonest users. Sites such as SurveyMonkey and SurveyLegend provide services such as email and location verification, but this is a very limited form of protection. Sophisticated solutions are offered by companies such as OpinionRoute for third party survey fraud prevention, but their services require an in depth knowledge of market research and survey insights and are completely unsuitable for casual surveying.

This realization impelled the author to create a user friendly application which allows for straight forward survey hosting and scalable protection without the need for a prior skill set from the user. This project would result in an excellent tool for those using the application to sharpen the quality of their survey data without having to research deep into the field of market research. In addition, a project which consists of such a broad technology stack would be extensive in honing the authors knowledge as a developer.

1.2 Contribution

The contributions to this project were established from an array of aspects, with two in particular being the driving forces. The learning outcomes which the author achieved from college modules provided them with much of the base knowledge required for the technologies used in this project. Two in particular which were monumental in the preparation of their skill set were the distributed systems programming and data analytics programming modules. Writing code which could communicate across distributed systems using industry practised design patterns was a major turning point for them, as the transition from being a novice to a capable programmer was becoming apparent. Their fascination with the endlessly possible applications for machine-learning was established when learning the core fundamentals of data analytics. The understanding gained of the various machine-learning algorithms and their use-cases were a key element to the overall functionality of this project.

While the said modules provided them with an excellent base of knowledge to be used in conjunction with this project, they maintain that the bulk of the skills which they regard as the main contribution were those which they acquired while on work placement with OpinionRoute. Employed as an intern developer, they were provided with the opportunity to work on a full technology stack, with their code being released in the companies end product. In doing so, they became competent in component based front-end architecture and back end processing, learning the complexities of the Angular and ASP .NET Core frameworks. The authors understanding of the agile approach to software development along with the organisation of multiple different technologies in a stack was refined to the point where this project would be within their capabilities of producing.

1.3 Structure of This Document

The rest of this document is structured as follows:

Chapter 2 - This chapter begins by describing the area within computer science in which the project is based on. It describes each of the technologies and topics it touches on in detail. It then proceeds to give a review of the current state of the art of the technologies.

Chapter 3 - This chapter discusses the problem at hand, listing out the objectives, functional requirements and non-functional requirements.

Chapter 4 - This chapter explains how the implementation of the project will be undertaken in detail.

Chapter 5 - This chapter documents the steps taken to implement the proposed application from the ground up.

Chapter 6 - This chapter presents an objective evaluation of the final system, describing the applications functionality, metrics, system testing and results

Chapter 7 - This chapter reflects on the project and report, discussing what could have been done differently as well as what may be done in the future.

Chapter 2

Background

2.1 Thematic Area within Computer Science

The subject matter of this project is to investigate whether an entity committing click fraud can be detected using a machine-learning algorithm through a user-interface and a cloud platform. In this section the author investigates the theory behind click fraud and in particular the danger it poses to the integrity of survey data. The theory and various means of implementing anomaly detection algorithms are also be investigated thoroughly, as the methods chosen will have a profound effect on the success of the application. Also discussed are the technologies that will be employed to provide a full-scale web application, with some background on how they have evolved over the years into advanced frameworks and services.

2.1.1 Click Farming

A form of online fraud, click farming in recent years has grown into profitable business for cyber criminals, which mainly operate in developing countries. The essence of a click farm is to create large amount of fake online accounts and profiles on platforms such as social media pages and e-commerce stores. The accounts can then be used as a swarm for a multitude of uses. The owner of the click farm, known as the 'farm master', advertises services which can be bought online, such as the sale of Facebook likes and Instagram followers. Click farms operate using manual labour, bots, or sometimes a mix of both.

FIGURE 2.1: Inside a human operated click farm [1]

- **Manual labour click farms** - These farms rely on human workers who manage an arrangement of either phones or computers. Often the areas in which they operate in are unfit for work due to the dark lighting and heat from the electronics. They iterate over the devices, completing mundane tasks such as creating profiles, liking posts, and filling out surveys. Manual labour click farms are more expensive than bot operated ones but are necessary when actual human interaction is needed, such as when filling out a CAPCHTA [2]. In many areas of the world there are no laws condemning click farms directly, despite how unethical in nature they are. Workers are paid, on average, one US dollar for a thousand likes or for following a thousand people on Twitter [3]. Most manual labour click farms operate using the following process.

    1. A warehouse is rented out by the farm master and populated with a number of either hand held devices or computers, typically in a developing country.
    2. Workers are hired to operate the devices in the click farms.
    3. The click farmer advertises the services available online, with most taking credit cards as payment.
    4. When a service is purchased, the workers use the swarm of devices and profiles to complete the task

- **Bot operated click farms** - These farms are created by farmers with advanced programming skills, using automated swarms of bots as opposed to manual workers. Using bots is cheaper, faster and does not require the physical space which is needed when using manual workers. These bots run scripts created by the farmer and are directed towards the target. Bots are thought to account for around half of all internet traffic, with others putting the figure as high as 60 percent [1].

While click farms are used to target a multi-varied range of targets, there are three areas in particular of interest for this paper which are commonly effected.

- **Online Surveys** - One of the most popular areas attacked using click farms, and the main subject of this paper, is online surveys. Online surveys often offer rewards for gathering the opinions of internet users. In order to access the payload from these surveys, experienced programmers create bot swarms which they distribute across the internet using web-trawling, filling out surveys untruthfully and gathering the rewards. The individual hosting the survey ends up with dirty data, and to make matters worse, they have paid for it in some form. Due to the fact that the nature of surveys is to gather a limited number of instances, survey fraud can also be carried out just to use up the survey instances so that the end data received will be mainly be dirty, and as a result skewed. In August 2020, market researchers OpinionRoute discovered a large click farm in Texas using their CleanID technology [4]. This goes to show that click farm fraud as an industry is not only prevalent in developing countries, and is a worldwide issue.

- **Social Engineering** - Social engineering has become a widely discussed topic in recent years, and click farms have had a major role to play in many high profile cases where social engineering was used. It is widely believed that many multi-national companies are utilizing click farms for online social media growth, escaping detection by authorities by employing 'middle-men' to seek out and hire the click farms. Recently Hasbro, the parent company of the popular board game 'Monopoly', were forced to denounce a casino company who was sub-licensing the Monopoly brand after it was discovered the casino has used click farms to boost their Facebook pages [5]. According to the Times, during the Brexit debate Russian click farm bots tweeted more than 150,000 pro-Brexit tweets in the build-up to the vote [6].

- **Advertisement Campaigns** - Click farming is also used to attack advertisement campaigns, which ties in with survey fraud due same goal of 'drying up' the victims resources. It is carried out using one of the following methods. The first is by hiring competitor fraudsters to deplete the advertising budget of the competitor so that they will be able to have their ads shown in higher pay-per-click rankings at a lower cost. In this case, the competitor is weakened instead of being outbid in the pay-per-click bidding system. The investment on the click farm made by the fraudster is only a very small fraction of the amount lost by the competitor. The second is by hiring the click farm workers to click on ads on the click farmer's own site. This way, the money lost by the advertisers is gained by the click farmer, rather than by the search engines and content networks as in the first method.

Over 300 billion dollars a year is spent worldwide on digital advertising, and it is believed that as much as one in four dollars spent is lost to ad fraud [7].

2.1.2 Anomaly Detection

In data mining, anomaly detection is referred to the identification of items or events that do not conform to an expected pattern or to other items present in a data-set. Typically, these anomalous items have the potential of getting translated into some kind of problems such as structural defects, errors or frauds [8]. In this project we are using anomaly detection to uncover anomalies in the completion of online surveys in order to detect the presence of a click fraud entity.

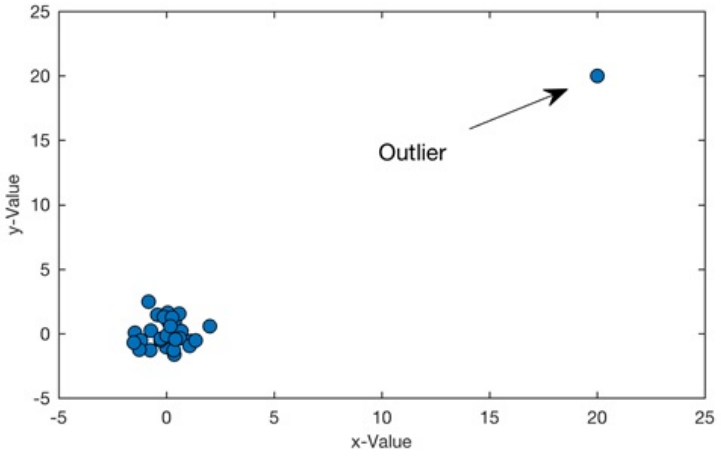

FIGURE 2.2: Anomaly in a Data-set [8]

The use of anomaly detection in computer systems was first coined by the North American cyber-security researcher Dorothy Denning in 1986. Originally being proposed for use in intrusion detection systems, it became an important tool for data pre-processing in machine-learning. In the year 2000, data scientists began looking at anomaly detection as a viable form of collecting information to solve real world problems [9]. This was in comparison to using previously favoured techniques such as clustering and classification. It was determined that anomaly detection algorithms could be a profound tool for solving problems in areas such as fraud detection and disease prediction

Anomalies can be generalized into three different classes.

- **Point Anomalies** - A data instance which is anomalous due to its distance from the general population of data-points in a set.
- **Contextual Anomalies** - Data which is an anomaly due to the context of the data collected. A common example is in a time-series, it may be typical for a shop to sell lots of coats in Winter but not during the Summer.
- **Collective Anomalies** - A collective set of data instances which detect anomalies as a cluster.

2.1.3 Reinforcement Learning

Reinforcement learning is a branch of machine-learning where the agent learns through trial and error. Similar to the way a human learns to ride a bike, a machine-learning algorithm using reinforcement learning improves its through its experiences. It involves communication between the agent and its environment using states, actions and rewards [10]. It follows what is known as a Markov decision process. Reinforcement learning is suitable for this project as there is no training data available for an unsupervised or supervised anomaly detection algorithm. This project will incorporate deep reinforcement learning for unknown anomaly detection, using an approach cited in a paper from Texas A&M University [11].

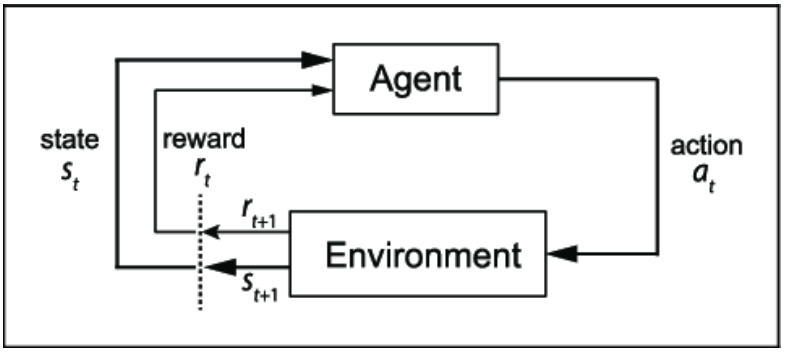

FIGURE 2.3: Markov Decision Process of Reinforcement Learning [12]

FIGURE 2.4: The Cloud [13]

2.1.4 Cloud Platform

A cloud platform, or cloud computing service, delivers on-demand computing services online using a pay for what you use approach. These platforms are based on the trait of flexibility. Their computational hardware is stored in massive data centres around the world which can be accessed via connecting to a central server. This allows users to access deployed websites, machine-learning algorithms, databases and even robotics applications from anywhere with an internet connection. The services they provide are elastic in nature; this is the dynamic allocation of cloud resources to projects, workflows and processes [14]. They are also highly scalable, meaning the platform will handle the alternation of the applications requirements over time within the confines of the infrastructure via statically adding or removing resources to meet applications demands if needed [15]. The cloud platform market is largely dominated by three providers in particular; Amazon Web Services (AWS), Microsoft Azure and Google Cloud [13]. AWS will be used as the cloud platform for the purpose of this project for several reasons. Firstly, the author has experience with the platform as they used it in the months prior to completing this project on an internship. This resulted in the author becoming comfortable with the AWS console and the services it provides. Using services such as S3 buckets, elastic beanstalk and Sagemaker resulted in a large amount of knowledge being gained about them and this factor could not be over looked. Also, the fact that AWS Sagemaker is currently the lead machine-learning service in cloud computing is important to note.

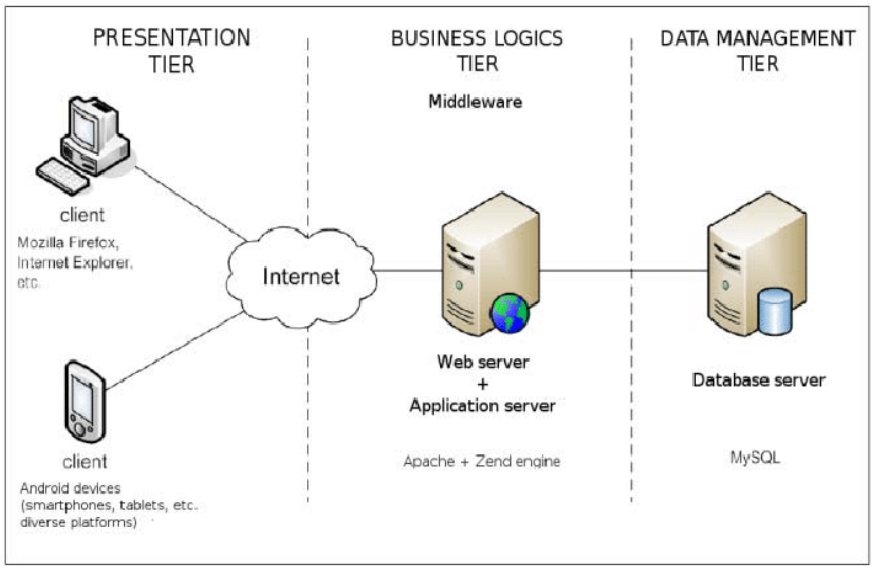

FIGURE 2.5: Three-Tier Web Architecture

2.1.5 Three-Tier Web Architecture

Three-tier web architecture is a form of multi-layered web architecture, with the complete stack being made up of a client tier, application tier and a database tier. This is opposed to the more traditional two tier architecture which consisted of only a client tier and database tier. The complexity of websites and the processing involved with the services many of them offer has advanced at a rapid pace. The advantage of three tier architecture is that we can separate the client from the business logic, offering better scalability, performance, availability and complexity to web applications. This added complexity has also been facilitated by the emergence of software frameworks. Providing a foundation for the software being written, frameworks encourage code recycling and good practice from the developer, as well of the use of predefined libraries and classes which are available from the chosen frameworks repository. There are a large number of web frameworks available for both front-end and back-end development, however a select few dominate the industry. For this project we will be using the following frameworks in the technology stack.

- **Angular** - Angular is unique with its two-way data binding feature. This means there is a real-time synchronization between the model and the view, where any change in the model reflects instantly on the view and vice versa. One can also use this framework to develop multi-page as well as progressive web applications. Companies like BMW, XBOX, Forbes, Blender, and others deploy applications built with Angular. It is a component-based framework, which is a hybrid approach of layered and feature-based architecture.

- **ASP .NET Core** - A member of the .NET family, ASP .NET Core is developed by Microsoft and was first released in 2016. It is a free and open-source framework and it is the successor to ASP .NET [16]. It supports MVC architecture, however for the purpose of this project we will be using it as a back-end web API application for processing and database access. Within the application, we will be making extensive use of the ClosedXML and OpenXML libraries for the parsing of excel spreadsheets.

2.1.6 Application Programming Interfaces

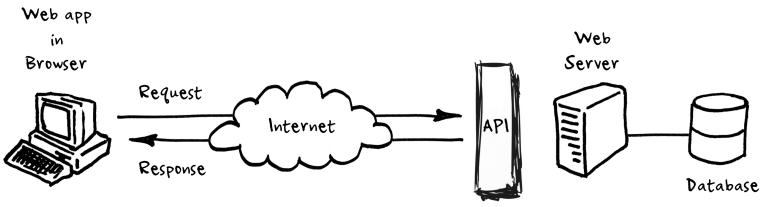

FIGURE 2.6: API Sketch [17]

An API is concept in software technology that essentially refers to how multiple applications can interact with and obtain data from one another [17]. It is the messenger which takes requests as an input and outputs the requested data back to the user. API's are called using HTTP requests. HTTP is the underlying format that is used to structure request and responses for effective communication between a client and a server [18]. The use of API's will feature heavily in this project.

2.2 A Review of Survey Click Fraud

The desired outcome of this review is that the author will gain knowledge of how click fraud occurs in the current world and what technologies offer with regards to survey protection from click fraud.

Conferences and Journals

- IEEE International Conference on Communications in China (ICCC) - This annual conference brings professionals and experts from around the world together to discusses current and emerging technologies in the field of communications, with the subject of click fraud being a major topic [19].

- International Fraud Prevention Conference (IFPC) - This is an annual conference which is held in Dublin, Ireland. Some of the top cyber security and fraud expert from around the world attend every year and discuss the latest cutting edge technologies in cyber security [20].
- Identification of Click Fraud and Review of Existing Detection Algorithms, by Shubhangi Jain, Falguni Jindal, Anmolika Goyal, Savy Mudgal [21]
- Towards a Machine Learning Approach for Detecting Click Fraud in Mobile Advertising, by Riwa Mouawi, Mariette Awad, Ali Chehab, Imad H. El Hajj, Ayman Kayssi [22]
- Fake reviews tell no tales? Dissecting click farming in content-generated social networks, by Haizhong Zheng, Neng Li, Minhui Xue, Suguo Du, Haojin Zhu [23]

Books and Texts

- Internet Fraud Casebook: The World Wide Web of Deceit, by Joseph T. Wells [24]
- Social Engineering: The Science of Human Hacking, by Christopher Hadnagy [25]
- Fraud Analytics Using Descriptive, Predictive, and Social Network Techniques, by Bart Baesens, Veronique Van Vlasselaer, Wouter Verbeke [26]

Companies which may be interested in this project

- OpinionRoute LLC - The leaders in market research and insights processing, OpinionRoute provide first-class data collection services and QAAS technology solutions [27].
- SurveyMonkey - Currently the worlds most popular survey creation website, SurveyMonkey allows users to create and host surveys online with ease [28].
- Qualtrics - Qualtrics a simple to use web-based survey tool to conduct survey research, evaluations and other data collection activities [29].
- Forter - Forter is SAAS company that provides fraud prevention technology for online retailers and marketplaces [30].
- DataDome - DataDome is a SAAS bot protection solution for e-commerce and classified ads businesses [31].

Wikis, Forums, Blogs and Videos

- Fraud: How They Steal Your Bank Account [32]
- Google: The Truth About Click Fraud [33]

2.2.1 Current State of the Art

With the ever expanding Internet of Things, the world is becoming increasingly more data driven. Scammers and online fraudsters have in turn taken advantage of this world-wide change. Click fraud is on the rise and the emergence of click farms is increasingly more evident. Security against this kind of fraud has grown into an industry itself, with companies such as Google and Facebook taking massive steps to reduce the footfall of fake profiles and the influence they can have, from fake restaurant reviews to election swinging.

Click farms have been the subject of several investigations from high profile news outlets such as Yahoo Finance, France 24 and ABC News. Many popular YouTube channels have also covered the topic, such as TODAY and Technical Guruji. The report done by Yahoo Finance in particular states that click farms in China alone are a market worth 50 billion [34]. Even U.S president Donald Trumps success against Hillary Clinton was massively due to the current click economy and global digital labour market [35]. It is also worth mentioning that this was done on half the budget used by Clinton's Democratic party. Italian security researchers and bloggers Andrea Stroppa and Carla De Micheli discovered in 2013 that 360 million dollars to date were earned from the sale of and the potential benefits of buying fake Twitter followers. 200 million dollars a year is also earned from fake Facebook activities [3].

Online advertising is another sector which is currently under substantial fire from click fraud. About 40 to 80 percent of Facebook advertisements are bought on a pay-per-click basis. Advertisers have claimed that about 20 percent of Facebook clicks are invalid, and some advertisers have tried to seek refunds. This could cost Facebook 2.5 billion dollars of their 2014 revenue.

Survey fraud is a sector is often targeted by click fraud due to the payload that many surveys offer for their completion. In general terms, the term survey refers to systematic data collection about a sample drawn from a specified larger population [37]. The oldest known survey ever discovered recorded land register in Egypt in 3000 BC. It is believed the survey was undertaken to re-establish farm boundaries after a significant flooding of the river Nile [38]. It can also be said that this was one of the first known instances of data collection.

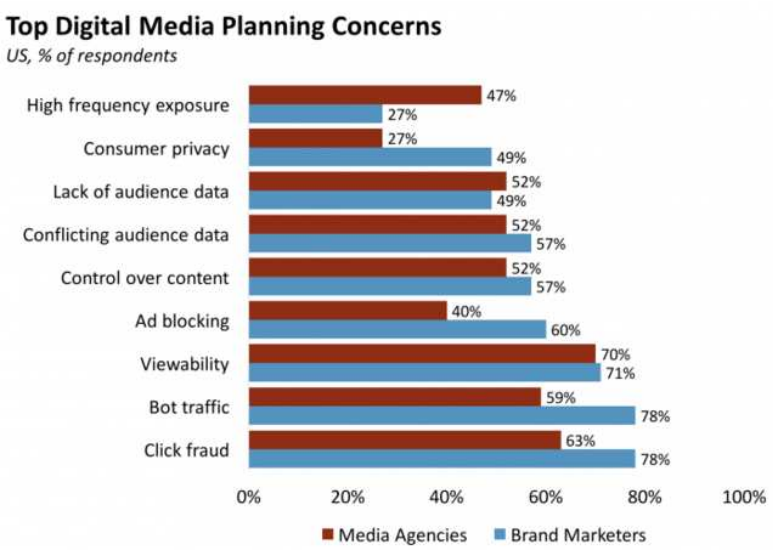

FIGURE 2.7: Business Insider Report on Digital Media Planning Concerns [36]

While the idea of a survey has not shifted drastically since then, there have been a number of changes to how they are conducted. The invention of the telephone brought on the idea of conducting telephone surveys in the 1960s [39]. Their success was limited however, as people answering the phone often mistakenly assumed the person wanted to sell them something, and hung up before providing information [40]. The collection of survey data took a huge leap with the emergence of the worldwide web. The jump from the physical into the virtual world resulted in the value of data sky-rocketing. This is also known as the 'Data Revolution' [41]. Due to the logging of nearly everything which is uploaded online, institutions now had direct access to data which could be utilized to further knowledge in nearly every area possible. Not only did the internet provide data, it could also be used for data collection, and the process of carrying out surveys rapidly moved online. Conducting surveys online, for the most part, proved to be superior in nearly every aspect.

1. Surveys could be deployed quicker and cheaper. The time and cost of creating, distributing and collecting physical surveys was now a fraction of what it was formerly.

2. Surveys could be deployed to a much larger population, allowing for the mass collection of data for large survey samples.

3. Surveys became less hassle for respondents to fill out, resulting in more survey quotas being achieved quicker.

Those creating the surveys began to further push to increase the response tallies by offering rewards to online users for filling out surveys. These rewards are usually presented in the form of cash, store credit or sweepstakes [42]. Websites which offered to advertise reward based surveys for institutions now began to appear online. For a fee, an institution can send their survey to a website offering this service. The website then informs their users that a new reward based survey is available for completion, and provides them with the survey.

The underlying issue of these reward based surveys is that fraudsters soon began to exploit them. Due to the payload, click farms were established to complete surveys as fast as possible so that they can claim the payload and move on to the next survey [43].

FIGURE 2.8: Example of a Website Offering Paid Surveys

As a result, the industry surrounding online surveying grew further, with market research and anti-fraud companies quickly providing solutions. Companies such as Qualtrics began providing survey hosting services with the option of using protections such as preventing ballot box stuffing and page indexing [44]. Another company called Opinion-Route took data protection further with their CleanID technology, which detects survey respondent markers such as oscillating IP address or user agent, and spoofing of browser signature.

Anomaly detection has been utilized for anti click fraud in sectors such as online advertising and social engineering [45], but is not implemented into a technology that protects survey data from dishonest respondents. The author intends to capitalize on this gap in the market and build such a service.

Chapter 3

Problem - Intercepting Survey Click Fraud using Deep Reinforcement Learning

"Data is the new oil" - Clive Humby. The collection of data comes at a cost, and companies in 2015 spent 68 billion US dollars on market research alone [2]. A vast amount of data collected for market research is from widespread opinions and experiences of general populations, while some data is collected from a particular subject matter. Regardless, in order to coax individuals into providing this data companies offer rewards for filling out surveys and questionnaires, much of the time in the format of online rewards and virtual currencies.

Individuals carrying out click fraud are exploiting this reward system by completing surveys dishonestly in rapid times using fake profiles. In order to receive payments from the surveys as quickly and often as possible, either human or bot operated click farms are used. This is a major threat to the integrity of the data being collected in these surveys, as the results will not be a true reflection of an individuals opinions or experience. Not only being a waste of money for the company, the data will become skewed overall and will present inaccurate end results.

Institutions with large budgets and market-research expertise have the option to protect the quality of their data collection over-time using software from data security services. But many smaller institutions wish to carry out surveys and questionnaires on an infrequent basis, and currently there is no high-level solution available to protect their data quality from fraudsters.

3.1 Objectives

Collection of Data and Investigation

- Investigate the existence of patterns which both human and bot operated click farms follow when fraudulently filling out surveys.
- Acquire suitable training and test data for the initial training of the chosen machine-learning algorithm.
- Determine which anomaly detection algorithms portray the highest accuracy when detecting click farms and fraudulent users in machine-learning.
- Investigate the legal proceeding involved in the handling and storage of user data.
- Investigate suitable technology stacks and cloud resources available to build and host the application.

Design and Build the Application

- Decide how the data will flow through the application. Design the database using ER diagram. Design the front and back end applications using UML and class diagrams. Design the UI using wire frames.
- Set up the environments required. Create AWS account and prepare the necessary cloud services for the application such as elastic beanstalk, S3 buckets and Sagemaker.
- Write the python script for the chosen machine-learning algorithms. Train, test and deploy using AWS Sagemaker.
- Code the front and back end applications.

Testing and Validation of Success

- Quality test for any defects or unprecedented activity in the application.
- Test the outcome and user experience of the application.
- Validate the success of the application and its ability to protect survey data from click farms and fraudsters.

3.2 Functional Requirements

The functional requirements for this project have been defined as the following.

- **Register Account and Login** - A user should be able to register an account and login with the application in order to use the services. The system should validate that the registration details entered have met the predefined standards, such as a an aptly secure password and a legitimate email address. Once an account is created, the user should be able to login to their account using the correct credentials. User data should be protected using correct authentication methods including a token service and password hashing/salting.
- **Create and Host New Survey** - Logged in users should be able to create a new survey for hosting. The application should take the user input of the surveys details and have a button to upload excel sheets with the survey data. The uploaded survey should be deployed as a website using S3. The user should have the option to deploy straight away or schedule for the survey to be deployed on a certain date.
- **Protect Hosted Survey** - The application should monitor the deployed survey and make a decision whether the survey is being filled out maliciously using the deep reinforcement anomaly detection algorithm. The application should detect malicious entities and document them in the database.
- **View Survey Performance** - The user should be able to view the real-time data on how their survey is performing. The application should send a daily report email to the email the user registered with, including all relevant data gathered on activity on the survey in the last 24 hours.
- **Download Survey Performance Statistics** - The user should be able to download all graphs created by the application.
- **Edit Hosted Surveys Details** - The user should be able to make edits to a survey which is currently deployed. The length of time for which the survey is deployed and the excel file with the survey data should be able to be updated. The user should also be able to end a surveys deployment prematurely.

3.3 Non-Functional Requirements

The functional requirements for this project have been defined as the following.

- **Simple UI** - The application should have a clean and descriptive user interface so that users have no trouble navigating the application.
- **Responsive UI** - The application should be quick and responsive, so that the user can carry out tasks as easily as possible and on a range of different devices.
- **Descriptive Alert Messages** - The application should have success and error messages when the user is executing tasks such as logging in and uploading files, so that the user is informed whether they have done so correctly or not. If the latter, the message should provide a description as to why the task was unsuccessful.
- **Edit User Details** - The application should allow the user to edit their personal data, such as credentials and profile image.

Chapter 4

Implementation Approach

Developing a full-scale web application capable of carrying out the desired functionality mentioned in chapter three will incorporate a large number of technologies in its stack.

4.1 Architecture

The architecture of the stack used for this project will be as following:

- **Angular 11** - Front-End
- **ASP .NET Core** - Back-End
- **Microsoft SQL Server** - Database
- **AWS EC2** - Application Hosting
- **AWS Sagemaker** - Machine-Learning Hosting
- **AWS S3 Buckets** - Deployed Survey Hosting
- **Swagger** - API Interface

4.1.1 Software Development Tools

The integrated development environments which will be used for the project are Visual Studio Code, Visual Studio and TensorFlow. These environments were chosen due to their rich features and the presence large online communities associated with them which offer technical support. The API documentation tool Swagger will also be used.

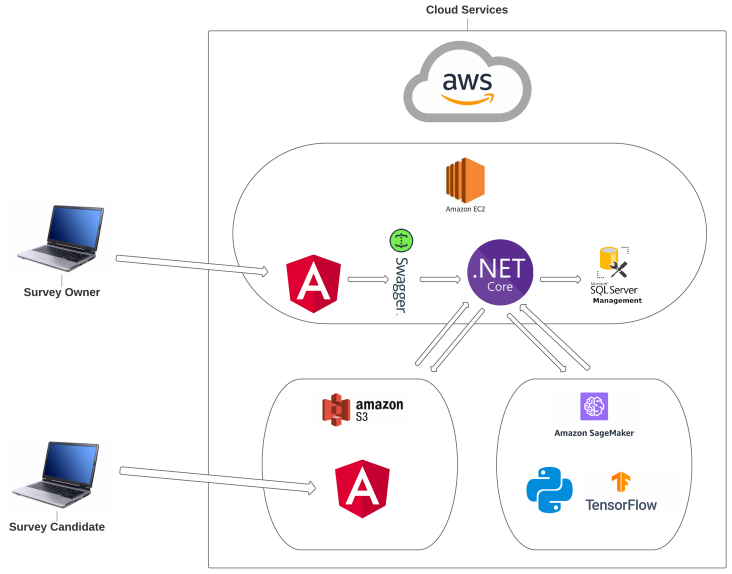

FIGURE 4.1: Complete System Architecture

- **Visual Studio Code** - Visual Studio Code is a feature-rich and lightweight source code editor developed by Microsoft Corporation. It has an extensive collection of libraries and tools which make it an ideal solution for front-end development. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code re-factoring, and embedded Git [46]. It is often described as a 'front-end out of the box IDE'.

- **Visual Studio** - Visual Studio is a suite of component-based software development tools and other technologies for building powerful, high-performance applications [47]. Its powerful debugging capabilities make it an ideal solution for the development of back-end applications, and is the primary IDE used for development of applications using the .NET framework.

- **TensorFlow** - TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.

- **Swagger** - Swagger is a RESTful API documentation tool. It uses the OpenAPI specification and generates interactive documentation which can be viewed on a browser.

4.1.2 Frameworks

A total of four software frameworks are to be used in this project stack. Each framework was chosen with careful consideration, taking into account their distinctive features, design patterns, available external libraries and in the case of the client-side frameworks, their visual aspects.

- **Angular** - Angular was chosen as the front-end framework to be used for the web application as its component based architecture can provide a smooth and straight forward experience for the user. This was essential as one of the main characteristics of the application is to be as user-friendly as possible. Due to the nature of the application having to take a large amount of user input, Angulars reactive forms feature makes it an ideal choice.

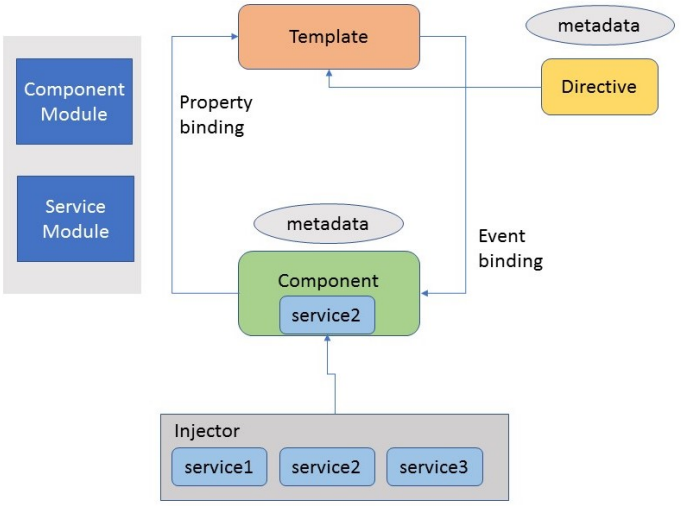

FIGURE 4.2: Angular Application Architecture

- **ASP .NET Core** - ASP .NET Core was chosen as the back-end for all the applications processing due to its RESTful API, existence of the ClosedXML library and its overall power and speed. As its environment is cloud-ready, no extra configuration will needed in order to deploy it to the AWS EC2 instance it will run in. The Angular client will communicate with ASP .NET Core using a RESTful API. Its architecture will follow the repository pattern, passing data from the controllers into relevant services where all business logic is performed. All communication with the AWS Sagemaker instance where the real-time machine learning is taking place will also occur in the service layer. Any communication with the database is done through the repository layer. ASP .NET Core uses its own method of querying SQL using what is known as Language Integrated Queries, or LINQ. LINQ

allows the use of C# to write strongly typed queries. It uses a derived context and entity classes to reference database objects. Entity framework core passes a representation of the LINQ query to the database provider. Database providers in turn translate it to database-specific query language, which in the case of this project will be SQL [48].

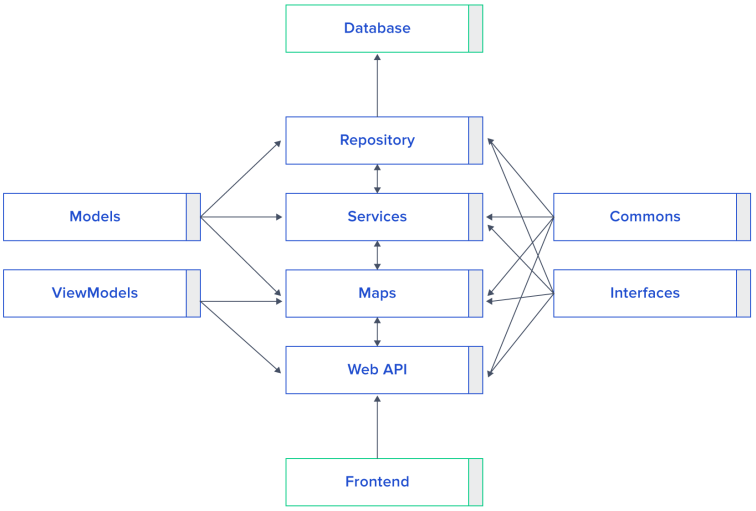

FIGURE 4.3: Repository Pattern in ASP .NET Core Application

- **Bootstrap** - The Bootstrap framework in conjunction with SCSS will be used for the styling in the client-side Angular application. This is mainly due to the benefits which Bootstrap will provide with regards to development speed. Its responsive structures and styles are regarded an excellent match with the Angular framework, and will greatly reduce the amount of vanilla SCSS which would otherwise need to be coded. Its grid system will be used throughout the Angular application, with the login page being created using the Jumbotron and Modal classes. Many other features such as its drop-down class will also be used extensively.

4.1.3 Libraries

- **ClosedXML** - ClosedXML is a .NET library for reading, manipulating and writing excel files. It is used to parse data from excel files into C# classes such as lists. It does so using the various streams classes in C# and XLWorkbook object as provided by the library. It will part of the core functionality of the whole system, being used to parse the excels files passed in when a new survey is being created. The data extracted is then used to create static web-pages for that new survey, hosted in an S3 bucket.

- **AlertifyJS** - AlertifyJS is a JavaScript library which provides color coded pop-up messages on button clicks. Each message is completely customisable. The library will be used for all success and error messages in the project.

- **Apex Charts** - Apex charts is a free to use JavaScript library which is under the MIT licence. It is a highly responsive, interactive and dynamic library which is renowned for its high performance and ease of use. It will be utilized heavily in this project for the visualization of each surveys performance across a multitude of graphs.

- **Font Awesome** - Font Awesome is a font and icon tool-kit which was created using CSS and Less. The majority of icons are free to use and it complements Bootstrap very well.

4.1.4 Cloud Platform

The cloud platform chosen, as mentioned earlier, is Amazon Web Services, or AWS. Three particular services provided by AWS will be utilized in this project. These include EC2, S3 and Sagemaker.

- **AWS EC2** - AWS EC2 is a virtual server which offers computational power in the cloud. It works off the basis of instances, with each instance type varying in power. It is completely elastic and allows capacity to be increased or decreased in a matter of minutes, depending on the growth of the application. This is where the projects client-side and back-end applications will be deployed. EC2 also provides cloud services for Microsoft SQL Server, which will be utilized for the projects database.

- **AWS Sagemaker** - AWS Sagemaker is a fully managed cloud service which allows for machine-learning algorithms to be built, trained, implemented and accessed all on a cloud server. Ready to go machine-learning models can be used from the AWS marketplace, however for this project the TensorFlow SDK will be used and deployed using Sagemaker. This is because a custom model is being used for the machine-learning. Data is passed to and from Sagemaker using API endpoints.

- **AWS S3** - AWS S3, also known as Amazon Simple Storage, is a scalable object based storage service. It stores objects using the concept of buckets. For this project, S3 will be used to host the static web-pages of newly created surveys. When a new survey is created, a service in the back-end will add it to the S3 bucket via the S3 API.

4.1.5 Application Structure

The client-side Angular application allows the user visiting the page to login to their account or register if they do not already have an account. Both the login and register functionalities use Angulars reactive forms. The client sends the completed forms to the back-end using HTTP requests, where the passwords are hashed and salted and authentication is confirmed.

Once the user is logged in, a navigation bar appears and this will remain on show on all the pages which require user login. This navigation bar has a Bootstrap drop-down which contains links to navigate to the view profile and edit profile pages, as well as the option to log out. The first page a user is navigated to after login is the home page. This home page shows cards which represent each of the surveys which that users account currently has hosted on S3. A user can choose between deleting, editing and viewing a survey from each card. A singular button which allows a user to create a new survey also exists on this page.

When the create survey option is chosen, the create survey page which consists of a reactive form opens. Here, the user can create a survey using the provided form. The user provides the amount of questions and answers which will be in the survey, and clicks the generate template button. A template excel file with the necessary amounts of rows and columns for the questions and answers is then made available. The user can download this template and fill the questions and answers they want to use in their survey into the template. The completed file can then be uploaded back up to the client using the upload survey data button. The create survey button can then be pressed, creating a new survey for hosting with the given form data and excel data. This survey data is created in S3 using HTML. The survey is now hosted on S3, and the an alert box opens displaying the link which the survey can be accessed at.

Now that the survey is hosted, it can be completed by any respondents who have the link for it. As the respondent fills out the survey, the machine-learning tracks the speed and patterns occurring in the answering. If an anomaly occurs, a counter increases in the algorithm. If the counter reaches a certain number, the algorithm decides that the respondent is filling out the survey incorrectly and they are kicked from the survey.

4.2 Risk Assessment

Risk 1

- **Risk** - User Uploads Bad Data
- **Risk Category** - Frequent Minor
- **Explanation** - The data uploaded in the excel files must correlate to the template downloaded. As it is excel, it is possible for the user to add extra columns to what is in the template and upload the excel file with extra columns.
- **Mitigation** - When the file is uploaded and sent to the back-end for parsing, there will be a try/catch error handler which prevent the system from attempting to parse an excel file with incorrect columns. An error message will be shown on the client side if this occurs.

Risk 2

- **Risk** - Browser Specific Bugs
- **Risk Category** - Occasional Major
- **Explanation** - Due the differences across browser when rendering client-side applications, a user may visit the website using a browser which does not render some of the application correctly. This may also occur due to browser updates.
- **Mitigation** - Create functionality in the application where a user can alert the developers of an issue with the website via email.

Risk 3

- **Risk** - General Bugs in System
- **Risk Category** - Remote Critical
- **Explanation** - Some of the websites functionality may linger after some time. This can be caused by issues from browsers being updated, libraries depreciating and other internal problems in the application.
- **Mitigation** - Do a full regression test of the application once every month. The alert button mentioned in risk 2 is also a solution.

TABLE 4.1: Initial risk matrix

Frequency/ Consequence	1-Rare	2-Remote	3-Occasional	4-Probable	5-Frequent
4-Fatal					
3-Critical		Risk 3			
2-Major			Risk 2		
1-Minor					Risk 1

4.3 Methodology

In order to complete this project successfully and as efficiently as possible, a number of preconceived measures must be taken. These measures will ultimately effect how the project is researched, how the necessary computer science skills are learned and which core project management approach is used.

- **Research Method** - The majority of research was conducted by using articles from Google Scholar and IEEE, but several other online sources were also used. It was accomplished using the seven step research process as described by Cornell University [49].

  1. Identify and develop your topic.
  2. Find background information.
  3. User catalogues to find books and media.
  4. Use indexes to find periodical articles.
  5. Find additional internet resources.
  6. Evaluate what you find.
  7. Cite what you find using a standard format.

- **Learning Necessary Computer Science Skills** - This project includes a large variance of technologies in its stack. The author intends to gain the base of the required skills by completing online tutorials from established sources before attempting to implement each technology. A 'learn as you go' approach will then be taken during each implementation, referencing back to the tutorials and documentation regularly to avoid making major mistakes.

  With regards to the web application itself, the author is confident that their programming skills in Angular and ASP .NET Core are of a proficient level. For the most part, other than general referencing of documentation and websites such as

StackOverflow the author should not need to undertake a large amount of study for these elements of the project.

All the necessary learning in relation to the cloud platform will be done using AWS official online tutorials. These tutorials are extensive and detailed as they are created by Amazon themselves, and should be all the author needs to gain the required knowledge.

The machine-learning implementation will be the most technical and difficult area to complete. A combination of using online tutorials and guidance from the project supervisor will be needed. Advice from a machine-learning expert in the university will also be sought out before implementation.

- **Core Project Management Approach** - The core project management approach taken when implementing this project will be an approach mixing the Kanban and Scrum methodologies, using the Kanban based web application Trello.

  Kanban is a subsystem of the Toyota Production System (TPS), which was created to control inventory levels, the production and supply of components, and in some cases, raw material [50]. Its use has expanded to nearly all sectors as its interpretation is narrowly restricted. It is widely used in the field of software development. Kanban involves the use of a board and cards, with each card representing a different item in the product backlog. On the board, the cards are placed in the column which represents their importance in the work-flow. These columns have the titles new, in-progress, testing and complete. As the state of a backlog item changes, it moves to the right along the board, and when it enters the complete column the item can be assumed as being done.

  Scrum is a subset of the agile approach to software development, and is planned by a scrum-master. After initial planning, a series of short development phases, or sprints, deliver the product incrementally. A sprint typically lasts one to four weeks [51]. A product backlog of all the tasks which must be implemented is also made, with each task being given a difficulty rating. For this project there will be a total of four sprints, with each sprint being 3 weeks in length. Tasks in the backlog are spread across the sprints equally.

  Mixing both these methodologies a 'Scrumban' board will be created in Trello. This will require creating sprints with time and task limits as per usual in scrum, using the layout of a Kanban board.

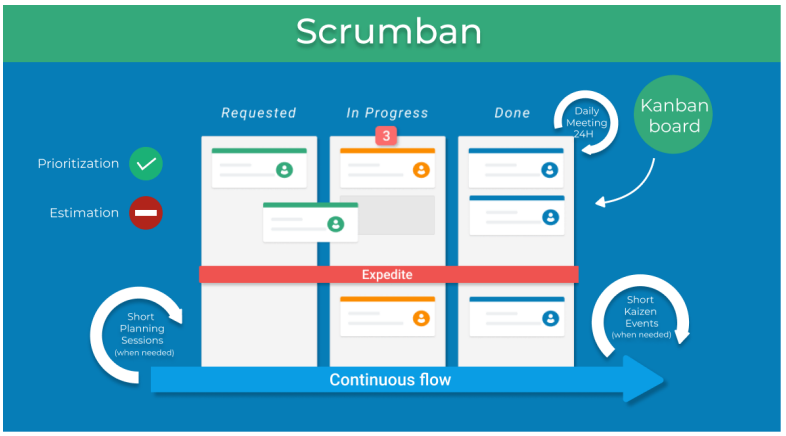

FIGURE 4.4: Scrumban Methodology[52]

4.4 Implementation Plan Schedule

For the implementation plan schedule of this project, the workload can be split into four sprints, each with their own sub-tasks to be completed. These tasks will be completed in accordance with the Scrumban methodology as described in section 4.3.

4.4.1 Web Application

Building the web application will not be the most technically challenging aspect of the project, but it will be the most time consuming. Therefore it will be tackled in the first sprint.

1. Create the database in Microsoft SQL Server Management.
2. Scaffold the ASP .NET Core application in Visual Studio and connect to the SQL database.
3. Populate the database with some dummy data for testing.
4. Generate the Angular project in Visual Studio Code.
5. Add ApexCharts, AlertifyJS, Bootstrap and FontAwesome libraries to the Angular project.
6. Add CORS support in the API.
7. Create the user login and registration components on the client side, implement security such as password hashing and salting on the back-end.

8. Develop the client-side further, creating skeletons for each page using Angular components. Connect them via services to the relevant back-end controller points with HTTP calls.

9. Create the back-end service layers for each controller and retrieve the dummy data using LINQ queries from the repository layers.

4.4.2 Machine-Learning Algorithm

The steps to create the machine-learning algorithm are short, but will require a great deal of study and time in order to code the model and ensure the results are satisfactory.

1. Create new TensorFlow project.

2. Develop the ML model in TensorFlow.

3. Test and tweak the ML model, documenting results.

4. Revert back to previous step until satisfactory results have been gained.

4.4.3 Cloud Platform

The setting up of the cloud services and deploying the applications to them should be relatively straight-forward, however complications may occur and ample time will have been allocated for this in the sprints.

1. Sign up for AWS with an account.

2. Create a key pair.

3. Create and launch a new EC2 instance.

4. Create an S3 Bucket.

5. Deploy Application to EC2 instance.

6. Deploy the TensorFlow project to Sagemaker for hosting using the TensorFlow SDK.

7. Configure the endpoints in Sagemaker for data to be passed to and from the machine-learning algorithm.

4.4.4 Connecting Technologies and Final Touches

It is worth noting that in the final few stages of the project will involve working on all technologies in the stack. This is due to many technologies depending on the progress of other technologies in the stack to be completed.

1. Connect back-end to the S3 Bucket for survey uploading and hosting.
2. Connect the back-end to the Sagemaker endpoints.
3. Finish styling the Angular application, such as making it responsive and reactive.

4.5 Evaluation

As the author is the only member of the team in this project, they will act as the scrum-master as well as the developer and tester. Each day during a sprint, a daily stand-up is held. Here the completion of the tasks in that sprint is reviewed. Tasks are moved around the Kanban board depending on progression and importance. At the end of each day, the author will check for tasks which have been moved to the testing column and act the role of tester. Using regression tests, they will decide whether the task can be moved to complete or flagged as having a bug, depending on the results of the tests. Regression testing is of utmost importance. It will prevent scenarios later in the development cycle where bugs are found in features with no time to fix them.

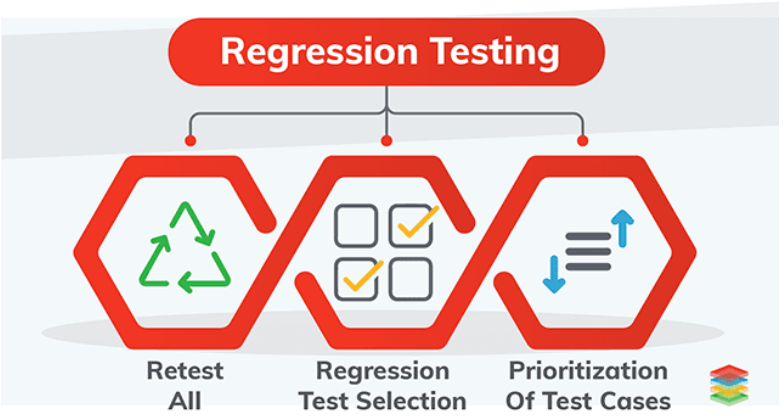

FIGURE 4.5: Regression Testing Cycle

4.6 Prototype

Prototyping for the Angular client-side application was done using the Pencil Project [53]. This is a graphical user interface prototyping tool which is simple to use and has an excellent library of elements drawn in a pencil sketch style. It is worth noting that filling out each input of a form on a given prototype page below is mandatory. If the incorrect format of data or no data at all is entered into any input of a form, an error message will appear under the given input. The buttons to progress for that page are also greyed out and non accessible until each of the required fields is filled out correctly. The home page can be accessed at all times by clicking the 'Survey Scrub' logo in the navigation bar. Also, after each button click either a success or error message will be displayed on the screen informing the user of the outcome of the button click. This will be done using the Alertify JS library. All wire-frame images can be found in Appendix B.

- **Login Page** - Figure B.1 shows the login page, which is compromised of a Bootstrap Jumbotron and a form with input fields for the user name and password. There are three options a user can take to navigate from this page. A register button loads the register component so that the user can register for an account if they do not already have one. The login button sends the form with the user name and password input to the back-end for validation, and if the login is successful the application will navigate to the users home page. If login is unsuccessful, it will remain on the login page. A link with the text 'Forgot Password' exists at the bottom of the Jumbotron. When clicked, this link generates a Bootstrap modal on the login page.

- **Forgot Password Modal** - The forgot password modal shown in figure B.2 is compromised of a form with input fields for an email and a password. Clicking the reset button sends the form to the back-end for validation. If the email does not exist in the database for an account, an error message will appear. If the email does exist in the database, a success message will appear and the modal closes. An email containing a link to confirm the password change will then have been sent to the given email address.

- **Register Page** - The register page in figure B.3 also contains a Jumbotron with a form with input fields. In order to register for an account with the application, the form must be filled out with several pieces of information. The forms inputs include a user name, password, email, institution name, subscription type, institution category and the uploading of an image for a logo. When the register button is clicked, the form is sent to the back-end for validation, and if the registration is

successful then the user will be navigated to their new home page. If the registration is not successful an error message appears and the user stays on the register page.

- **Home Page** - The home page in figure B.4 is the page navigated to after a successful login. Now that the user is logged in, a navigation bar at the top of the page appears and a drop-down to access areas of the users profile appears. This drop-down contains links to view profile, edit profile and logout. At the top of the main window of the application, there is a button called create survey. Clicking this button takes us to the create survey page. As can be seen by the cards on display, the user currently has six surveys created. Each card will have a small amount of detail about the survey such as the dates and sample size, and three buttons. The first button is to delete that survey from the cloud and the users account. The second button is the edit survey button, which navigates to the edit survey page for that survey. The third button is the view survey button, which navigates to the view survey page for that survey.

- **Create Survey Page** - The create survey page in figure B.5 contains a form with input fields. These input fields are for the name of the survey, start and end dates via a calendar, category of the survey, sample size and an upload for the excel files with the survey data. When the deploy survey button is clicked, the form is sent to the back-end for validation. If the survey is created successfully, the application navigates to the view survey page for that survey.

- **Edit Survey Page** - The edit survey page in figure B.6 is almost identical to the create survey page, however includes an edit survey button as opposed to a create survey button. When this edit survey page is loaded, the fields are already filled with the surveys present data. The user can edit these fields and save the changes by pressing the edit survey button.

- **View Survey Page** - The view survey page in figure B.7 contains all the data about the that survey. To the left of the page, details such as the dates, category, sample size and respondents numbers are displayed. To the right of the page, graphs such as bar charts, pie charts, scatter plots and line graphs give a visualization of the surveys performance. These graphs will visualize data such as percentage of respondents who have been flagged as click fraud entities. The data for these graphs will be derived from the machine-learning algorithm in AWS Sagemaker.

- **View Profile Page** - The view profile page in figure B.8 is reached through a link in the drop-down in the navigation bar. Here, a users profile information can be viewed.

- **Edit User Page** - The edit user page in figure B.9 is reached through a link in the drop-down in the navigation bar. When the page loads, all the form values are populated with the users current profile data. All the values in the form can be changed and saved using the edit details button.

Chapter 5

Implementation

This chapter documents the steps taken to implement the proposed application from the ground up. The first section discusses the difficulties encountered and the alternate solutions which were used in order to overcome them. The second section discusses the actual solution approach, how the project as a whole was tackled and how the development stage was implemented.

5.1 Difficulties Encountered

The following section lists the difficulties encountered during the implementation phase, along with their impact on the end solution and how they effected the time scheduling structure of the project. Each issue is categorized as being either easy, medium or hard, depending on its overall impact on the project.

Table 5.1: Ranking of difficulties encountered

Issue	Difficulty	Effect on Schedule
Calling REST API from client	Easy	Mediocre
Adding data persistence	Medium	Substantial
Adding RL model to Sagemaker	Medium	Severe
Creating endpoints in Sagemaker	Hard	Substantial
Running RL model in Flask	Hard	Severe

5.1.1 Calling REST API from Client

Angular services were not retrieving data from REST API despite not returning errors or null values. When first creating methods in the Angular projects service layer, they were failing to hit the .NET Core REST API endpoints. Debugging the issue was a struggle, as no error message was being returned and the address appeared to be correct after checking several times. Eventually it was discovered that the code calling the service layer was not subscribing to the service methods in the Typescript component, and therefore not receiving any data back from the service.

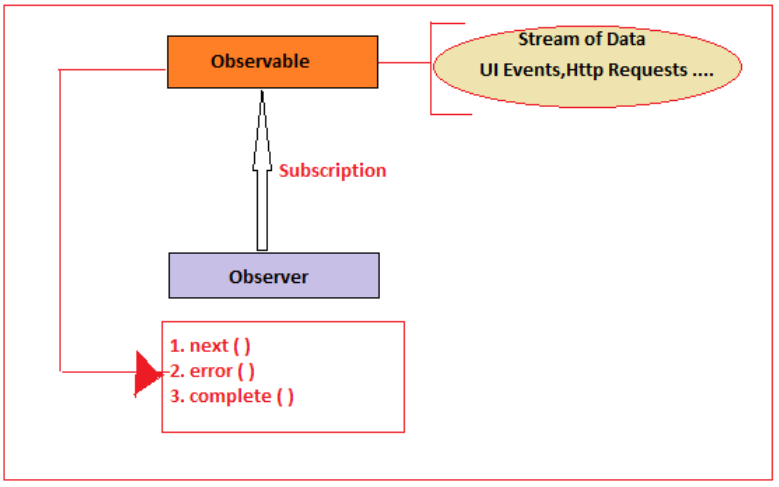

FIGURE 5.1: Subscribing to observables in TypeScript

5.1.2 Adding Data Persistence

Components in the Angular application did not have persistence and were losing data on page refreshes and routing. While the data was now loading correctly, the issue of data persistence now became apparent in the front-end. Page refreshing and using certain routes resulted in null values being returned in functionalities such as as logging in and retrieving survey data. In order to create login persistence, I used the browsers local storage to retain the users login data. For the components and routes, I modified the architecture to call the REST API each time a new one was loaded. However, the result had a detrimental effect on the applications performance, and did not prove to be an optimal solution. Instead, I used local storage to store the data on the applications primary load, and then a subscription service to retrieve the data when a new component or route was loaded.

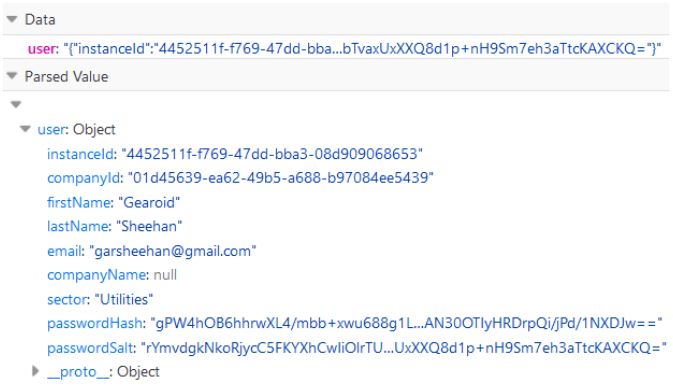

FIGURE 5.2: Storing the login data in local storage

5.1.3 Adding Reinforcement Learning Model to Sagemaker

As anticipated when investigating solutions earlier during the design phase, the reinforcement learning proved to be the most problematic component of the project. Firstly, the issue of depreciated code became a time consuming and tedious task when attempting to run the code on Amazon Sagemaker inside a Jupyter Notebook. The reinforcement learning was based on the model described in a paper from the year 2020 from Texas A&M University [11], however strangely all of the Tensorflow related python code was for Tensorflow version 1.6. This created a sizeable complication, as Sagemaker only provided Tensorflow kernels for versions 3.6 and up. The result was a vast amount of time was spent combing through the model, updating the code to comply with Tensorflow 3.6.

5.1.4 Creating Endpoints in Sagemaker

Once the model had been updated to include the required code for the provided Tensorflow kernel version, it was then time to test the model using the CSV data. The model proved to be exemplary in locating the anomalies in the test data, and was satisfactory for the task at hand. The next task was to link up the model with the ASP .Net Core back-end. As this is an elementary task when using Sagemakers predefined models, I presumed when researching for the project this would be the case for my reinforcement learning model. Unfortunately, it became apparent Sagemaker does not support creating endpoints in Jupyter Notebook instances. Instead, the reinforcement learning model would have to be reconstructed as a Flask application. As a virtual environment could now be used when creating the Flask application, the versions of Python and Tensorflow available for use no longer became restricted. This rendered the time and effort updating the models code to work with Tensorflow 3.6 a waste.

5.1.5 Running Reinforcement Learning Model in Flask

Despite the reinforcement learning model successfully training and detecting anomalies when deployed as a Jupyter Notebook on Sagemaker, once it had be reconstructed into a Flask application a significant error appeared. While the training of the model appeared to run smoothly as it had done before, when attempting to detect anomalies in unlabelled data-sets the reward for each data-set was returned as a list of zeros. The terminal displayed the message 'Loading a model without an environment, this model cannot be trained until it has a valid environment'. After consulting my supervisor and another lecturer who had experience with machine-learning models, I decided to opt for a different approach to detecting the fraudulent surveys, using a reinforcement learning classification model based on the Keras library.

```
WARN: gym.spaces.Box autodetected dtype as <class 'numpy.float32'>. Please provide explicit dtype.
WARN: gym.spaces.Box autodetected dtype as <class 'numpy.float32'>. Please provide explicit dtype.
Loading a model without an environment, this model cannot be trained until it has a valid environment.
```

FIGURE 5.3: Error occurring when running Meta-AAD RL algorithm

5.2 Actual solution approach

This section discusses the final solution of the application, taking into account the differences between the proposed solution and the solution which was implemented. The implementation schedule is also discussed, as the route taken differed slightly to the one originally planned due to the issues mentioned in 5.1.

5.2.1 Final architecture

The final architecture of the application is similar to the one proposed in the research phase, but has three modifications modifications made which are discussed below.

- **Cloud Hosting the Reinforcement Learning Model** - The first deviation from the original architecture was the removal of Amazon Sagemaker due to the issue of not being able to create custom endpoints. Instead, as seen in the new diagram, the reinforcement learning model was launched as a Flask application which is hosted in an Amazon elastic beanstalk instance in the cloud. This Flask application receives the excel data from ASP .NET Core using a REST API controller, and passes the data to the reinforcement learning model.

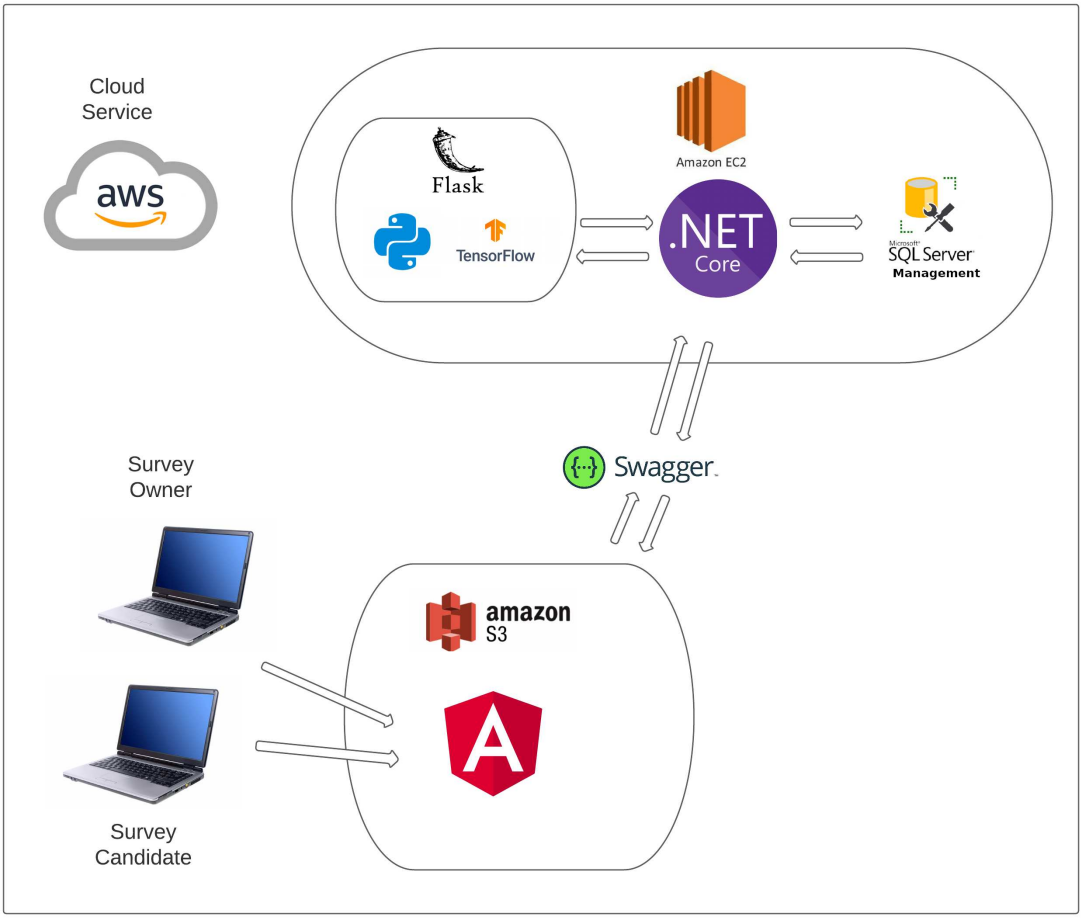

FIGURE 5.4: System Architecture of Developed Solution

Angular Application Architecture - The second deviation from the original architecture was changing the architecture of how the surveys are hosted. Originally the plan was to create two separate Angular applications - the SurveyScan application and a separate application to host the surveys. Instead, the final design used one Angular application, and added the survey hosting as a separate route and component which did not need user authentication to access. This change was made after considering the lack of benefits and time implications that went with creating a separate application for the survey hosting.

Survey Hosting - The final deviation from the original architecture was changing the cloud hosting environment for the Angular application to an S3 bucket as opposed to an EC2 instance, which the back-end and database were deployed on. This was due to a misunderstanding in the research phase. As an Angular application is static, an EC2 instance is unnecessary, and therefore can be deployed successfully using an S3 bucket and Cloudfront.

5.2.2 Use Cases

1. A user access the SurveyScan login page. They can enter their email and password and login to the application.
2. A user can access the register for account page and register for SurveyScan.
3. A logged in user can access the home page and view their open surveys and delete surveys.
4. A logged in user can access the create survey page and create a new survey.
5. A logged user can access the survey insights page for each open survey, view the survey metrics and download PNG, SVG and CSV data on the survey.
6. A user can complete a survey given the link, and submit the survey.

5.2.3 Risk Assessment

Risk 1

- **Risk** - User Uploads Bad Data
- **Risk Status** - Encountered and mitigated successfully
- **Explanation** - This risk was expected to occur, as even when uploading sample data there was occasions where the file uploaded was not formatted correctly due to user error. The result was when the files were being parsed from S3 on a surveys completion, unexpected behaviour such as questions with no answer option began to occur. This was solved by implementing the try/catch functionality in the back-end to prevent a survey from being created if the excel file was not formatted correctly.

Risk 2

- **Risk** - Browser Specific Bugs
- **Risk Status** - One occurrence not mitigated
- **Explanation** - An issue began to appear when using Firefox as the browser where an error stating '[WDS] Disconnected!' was displayed in the console. While the error was not causing issues in the application, it caused concern all the same. After researching online, the error was related to not having the correct SSL certificates installed. Unfortunately, time constraints preventing this issue from being resolved.

Risk 3

- **Risk** - General Bugs in System
- **Risk Status** - All known bugs mitigated
- **Explanation** - Several bugs presented themselves in the development process of the project. Through manual regression testing after implementing each piece of functionality, these bugs were identified and removed. At the time of completion of the project, there are no known bugs existing in the projects functionality.

5.2.4 Methodology

As the application was built by a single developer, each piece of functionality was fully built across the stack where possible before moving onto the next piece. Bugs which became apparent along the development process were added in as Trello tasks and were tackled as soon as the functionality being worked on at the time of discovery was completed. This was done to ensure a backlog of bugs which needed fixing did not build up, as this would effect time management as well as possibly create issues further down the line when new functionality was being worked on.

5.2.5 Implementation Schedule

As mentioned above, the implementation schedule diverged minimally off course at times due to running into issues and bugs, but kept to the Kanban methodology proposed in the research phase. Overall the schedule was adhered to for the most part, and was completed on time. The four sprint Kanban structure defined in the research phase proved extremely efficient for the completion of the project. In particular, the allocation of the fourth sprint for working on functionality which caused issue or bugs was paramount in completing the project on time.

- **Sprint One** - The projects development began with the scaffolding of the necessary frameworks and databases. Once these skeletal structures had been created, the ground works for the project were then laid. Beginning with the back-end, all the necessary dependencies such as CORs compatibility and adding Swagger as the API interface were implemented. An SQL database was then created, and using entity framework core the back-end was connected to the database.

Implementing the first piece of functionality then commenced - the user login. This was done in parallel to creating the registration page, as both pieces of functionality were interwined. At the time of development, no issues had appeared.

Once login and registration was completed the home page was created. This was a long and tedious task, as dummy data had to be created in order to create a structure for the card grid system. Here the issue of data persistence with the login became apparent, and a new Trello card was opened so that the bug could be fixed later on.

Now that a basic homepage was created which mirrored the proposed design on the wire-frame, the create survey page was developed. Only front-end work had been done at this stage for both the homepage and create survey page. This was because the back-end code for both would have to be done simultaneously. Once finished, the back-end was then constructed for both pieces of functionality. The dummy data was removed, and an application which allowed a logged in user to view their homepage and create surveys now existed.

- **Sprint Two** - This sprint began with the implementation of the reinforcement learning algorithm. A Sagemaker notebook instance was launched in AWS, and the algorithm was implemented. As the issues with the versioning and kernels became apparent, this task was abandoned and returned to several times in the sprint. When the critical issue with the endpoints was realized, the task was pushed back to sprint three as too much time had been spent already on the task.

  On the application side, the view survey insights page was developed in this sprint also. The base structure for the page was created, before the Apex JS chart library was implemented with dummy data.

- **Sprint Three** - This sprint focused on creating the component for hosted surveys. This was a particularly challenging sprint as it involved a large amount of code and online research in order to collect the survey answers and data, before parsing to an XLWorbook in the back-end.

  The reinforcement learning task which had been pushed back from sprint two was also completed in this sprint. Once the Flask application had been chosen along with the new learning model, it was time to connect it to the back-end. By the end of this sprint, the application had taken full shape and the remaining sprint could be spent improving, restructuring and tweaking the application.

- **Sprint Four** - As this was the final sprint, the majority of the work done here was either cosmetic or secondary functionality. The few bugs which had appeared over the course of the development were also ironed out here. Elements such as the

charts and dates were modified to no longer use dummy data, and could receive dynamic data on the survey in question from the database. Angular animations were added in order to improve the visual aspect of the application, and the theme and colouring were finalized using SCSS and Bootstrap.

Finally, the application was deployed to the cloud using AWS S3 buckets for the front-end, and EC2 instances for the back-end, Flask application and database.

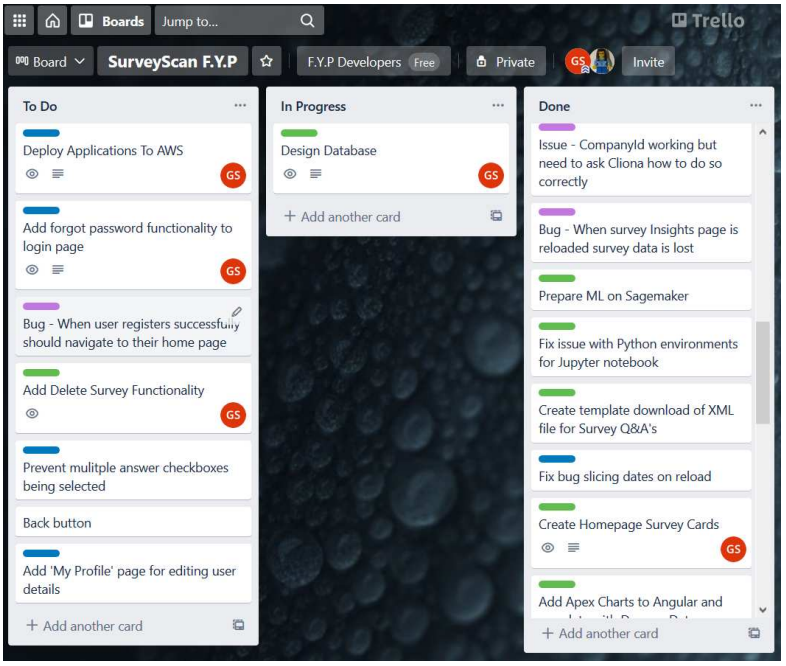

FIGURE 5.5: Trello Board

5.2.6 Retrospective of Functional and Non-Functional Requirements

5.2.6.1 Functional

- **Register Account and Login** - The register account and login requirement was completed to a very satisfactory level and there was no underlying issues. The required security features such as password hashing/salting were used to full effect.
- **Create and Host New Survey** - The create and host survey requirement was completed to a very satisfactory level and there was no underlying issues. The method of uploading surveys changed slightly from the original proposal, where a user now downloaded an excel template, filled the template and uploaded the file, but the underlying functionality was still satisfied.
- **Protect Hosted Survey** - The protect hosted survey requirement was not completed to a satisfactory level. This was due to the issues with the reinforcement

model mentioned earlier in this chapter. While a different model was used, the degree of protection offered was sub-par to the level originally aimed for.

- **View Survey Performance** - The view survey performance requirement was partially completed. While displaying survey statistics and graphing of the surveys performance were achieved, the automated daily email functionality was not implemented due to time constraints.
- **Download Survey Performance Statistics** - The download survey performance statistics requirements was completed to a satisfactory level. This requirement was achieved easily due to the download feature which was part of the Apex charts library.
- **Edit Hosted Surveys Details** - The edit hosted surveys details requirement was not completed. The decision not to complete this requirement was taken in sprint 4, as it was an extraneous requirement in comparison to others which needed to be completed under the time constraints.

5.2.6.2 Non-Functional

- **Simple UI** - The simple UI requirement was completed to a satisfactory level. This was achieved by utilizing the Bootstrap library and its ease of use.
- **Responsive UI** - The responsive UI requirement was completed to a satisfactory level. This was achieved by first completing the simple UI requirement as mentioned above, along with careful planned data flow through the website using subscriptions and minimal HTTP requests where possible.
- **Descriptive Alert Messages** - The descriptive alert messages requirement was completed to a satisfactory level. This was achieved through the developers previous knowledge of the AlertifyJS library.
- **Edit User Details** - The edit user details requirement was not completed. Similarly to the edit hosted survey details requirement, it was not worth the time needed under the given time constraints.

Chapter 6

Testing and Evaluation

The goal of this chapter is to present an objective evaluation of the final system. As the project is a cloud deployed web application, the full process of creating a web application from the ground up and deploying to a cloud provider is described. This is achieved by breaking the process into the following sections - functionality and testing/results. Larger screenshots of code can be found in appendix A using the provided references.

1. **Functionality**
2. **Metrics**
3. **System Testing**
4. **Results**

6.1 Functionality

In this section each of the main functions in the application are explained, along with the technologies used to create them. A combination of screenshots of the user interface as well as the underlying client and back-end code are used to showcase how each piece of functionality was implemented.

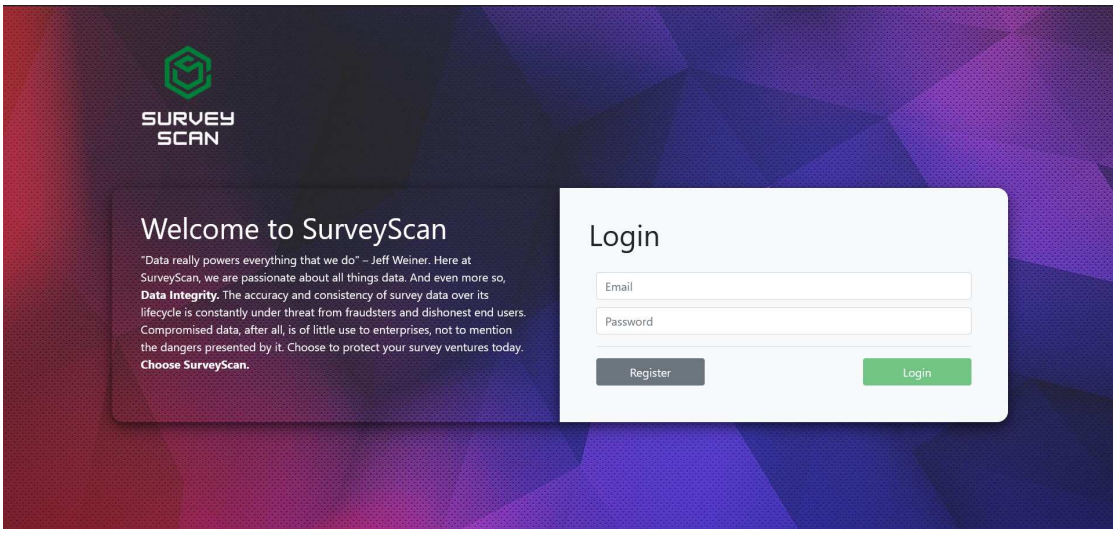

FIGURE 6.1: Login page

6.1.1 Login

Angular - The client side login allows a user to login to an existing account by entering their email and password into an Angular form. This data is then passed to the back-end, and if the login is successful the user is logged in and brought to their home page. The user data is saved to local storage for data persistence, and an RXJS replay subject is used to retrieve and set the currently logged in user from the local storage on each refresh. If the login is unsuccessful in the back-end, an error message appears and the user remains on the login page.

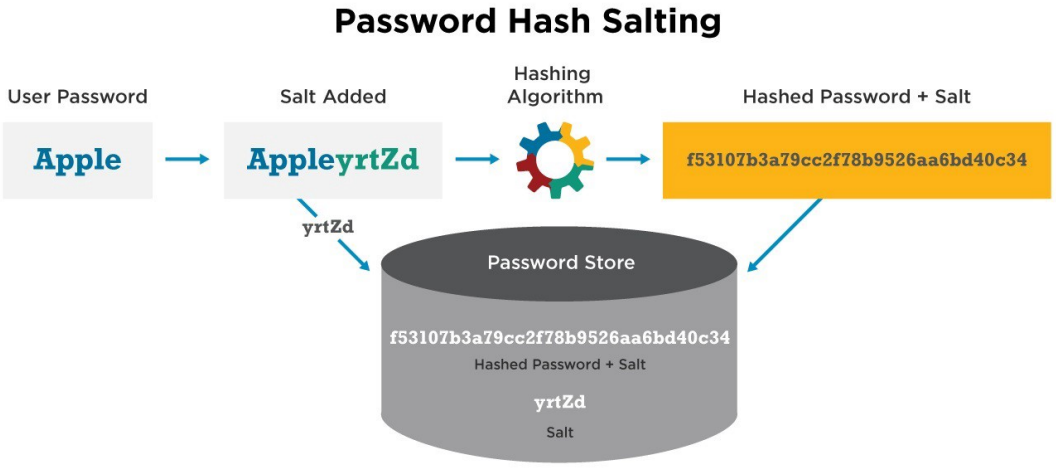

FIGURE 6.2: HMACSHA512 Algorithm

ASP .NET Core - The back-end recieves the data from Angular via its REST API controller. Using entity framework core, the database is first queried to check if the user email exists in the database, and if this is the case then the corresponding record is retrieved into the back-end. Using the HMACSHA512 keyed hashing algorithm, the password is decrypted from the records password hash and salt values and compared with the password entered by the user. If the passwords match, then the login details have been a success and the user data is passed back up to the client.

```
public async Task<User> Login(string email, string password)
{
    var user = await _context.Users.FirstOrDefaultAsync(x => x.Email == email);

    if (user == null)
    {
        return null;
    }

    if (!VerifyPasswordHash(password, user.PasswordHash, user.PasswordSalt))
    {
        return null;
    }

    return user;
}

1 reference | GearoidSheehan, 6 days ago | 1 author, 1 change
private bool VerifyPasswordHash(string password, byte[] passwordHash, byte[] passwordSalt)
{
    using (var hmac = new System.Security.Cryptography.HMACSHA512(passwordSalt))
    {
        var computedHash = hmac.ComputeHash(System.Text.Encoding.UTF8.GetBytes(password));

        for (int i = 0; i < computedHash.Length; i++)
        {
            if (computedHash[i] != passwordHash[i]) return false;
        }
    }

    return true;
}
```

FIGURE 6.3: Decrypting password hash and salt and returning user

6.1.2 Register

Angular - A user can register for an account by entering their details into an Angular form, accessible from the login page. Similarly to the login functionality, the form is sent the the back-end for processing. Due to the extensive validation features on Angular forms, much of the validation is done already client-side. If the registration is successful, the user is logged in and brought to their homepage, and if an error message appears and the user remains on the register page.

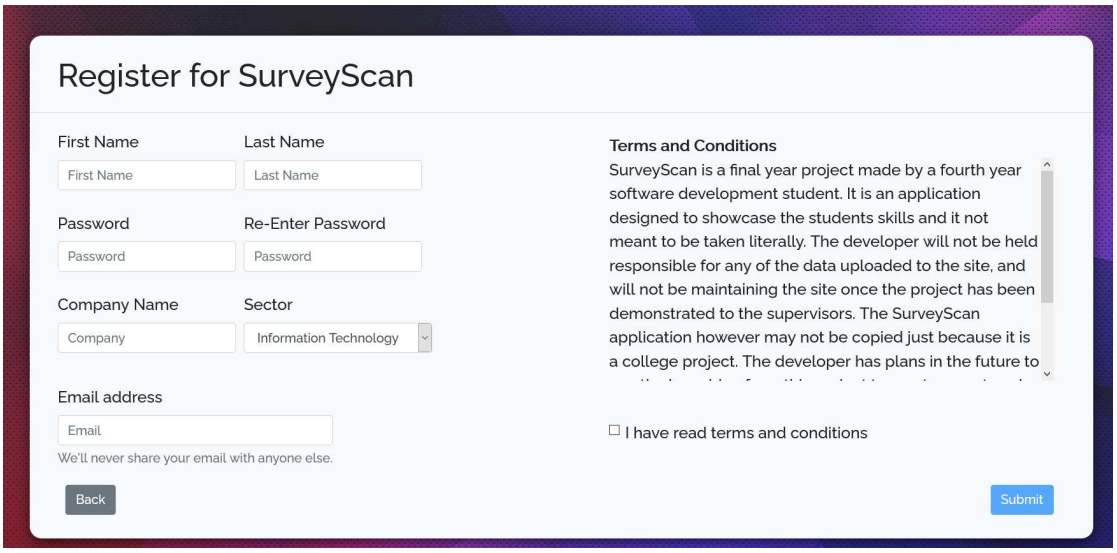

FIGURE 6.4: Registration form for SurveyScan

ASP .NET Core - The back-end retrieves the data from the incoming form and creates a user object and passes that object to the repository layer. Again, the HMACSHA512 algorithm is used, but this time to encrypt the users password to a password hash and salt. These are added to the user object and then saved to the user table in the database.

6.1.3 Home Page

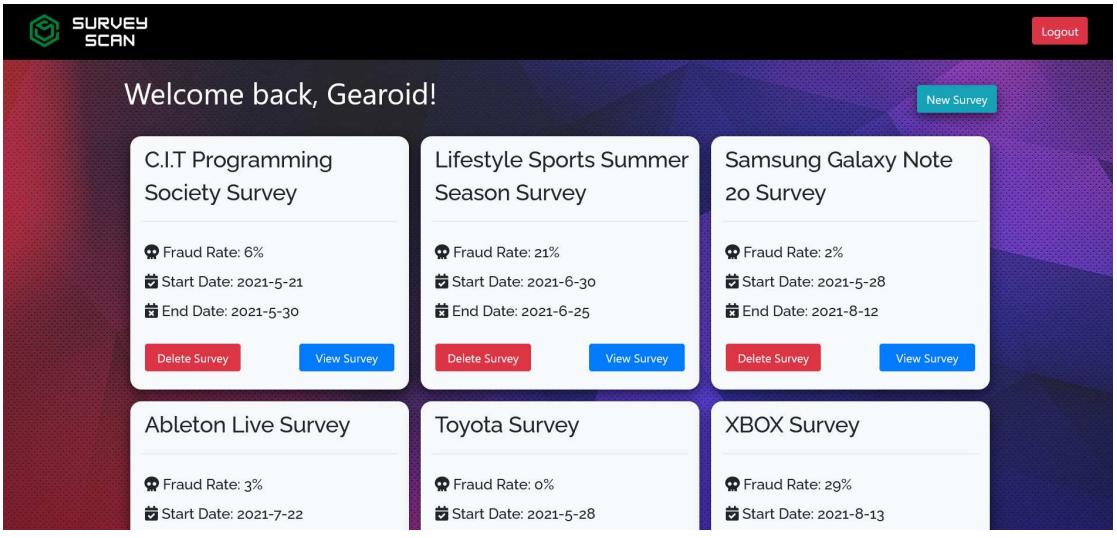

FIGURE 6.5: SurveyScan home page

Angular - When a user logs in, they are brought to the home page. This page consists of all the surveys a user has created, displayed in a grid format. Each survey card has some immediate data about that survey, as well as the options to either delete the survey or view the surveys insights. The list of surveys belonging to the logged in user are gathered from the backed by passing down the users company ID to the back-end. When the list is retrieved, the surveys are displayed in the cards using the Angular ngFor directive.

ASP .NET Core - When the controller receives the request from the client with the company ID, it is passed down to the repository layer and the list of surveys for that user are returned using entity framework core.

6.1.4 Create Survey

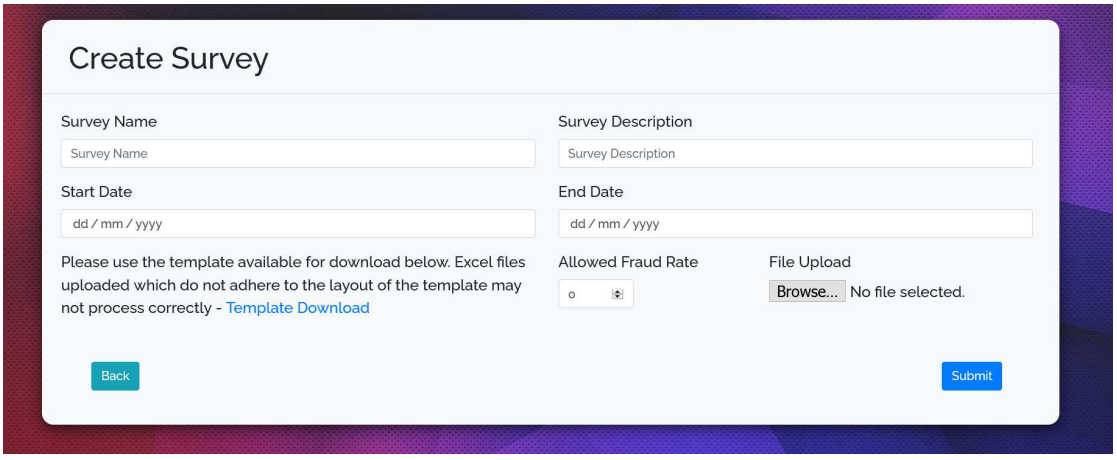

FIGURE 6.6: Create survey form for SurveyScan

Angular - A user can create a survey by filling out the create survey form. Along with datetime and string values, this form requires an excel file to be uploaded, consisting of the questions and answers which will be included in the survey. A template is available for download via a link in the form, which is retrieved from an Amazon S3 bucket. The form is sent to the back-end for processing.

ASP .NET Core - When the form is received by the controller, it is passed to the service layer. The incoming file is renamed with a GUID value, and is uploaded to Amazon S3 using a method from the AmazonS3Uploader class. This class contains methods for uploading and downloading files from Amazon S3 buckets, using file-streams and the AWS SDK for S3. Once the file is uploaded, the surveys details including the files new name are recorded to the database.

```
4 references | GearoidSheehan, 9 days ago | 1 author, 1 change
public class AmazonS3Uploader
{
    1 reference | GearoidSheehan, 9 days ago | 1 author, 1 change
    public async Task UploadFileAsync(IFormFile file, string newFilename)
    {
        var credentials = new BasicAWSCredentials("AKIAIRQ7TSVMGCEGJDNQ", "uFxYkrrGjIc5ni334
        var config = new AmazonS3Config
        {
            RegionEndpoint = Amazon.RegionEndpoint.EUWest1
        };

        using var client = new AmazonS3Client(credentials, config);

        await using var newMemoryStream = new MemoryStream();
        file.CopyTo(newMemoryStream);

        var uploadRequest = new TransferUtilityUploadRequest
        {
            InputStream = newMemoryStream,
            Key = newFilename,
            BucketName = "survey-scrub-xml-files",
            CannedACL = S3CannedACL.PublicRead
        };

        var fileTransferUtility = new TransferUtility(client);
        await fileTransferUtility.UploadAsync(uploadRequest);
    }
```

FIGURE 6.7: AmazonS3 upload function

6.1.5 Survey Insights

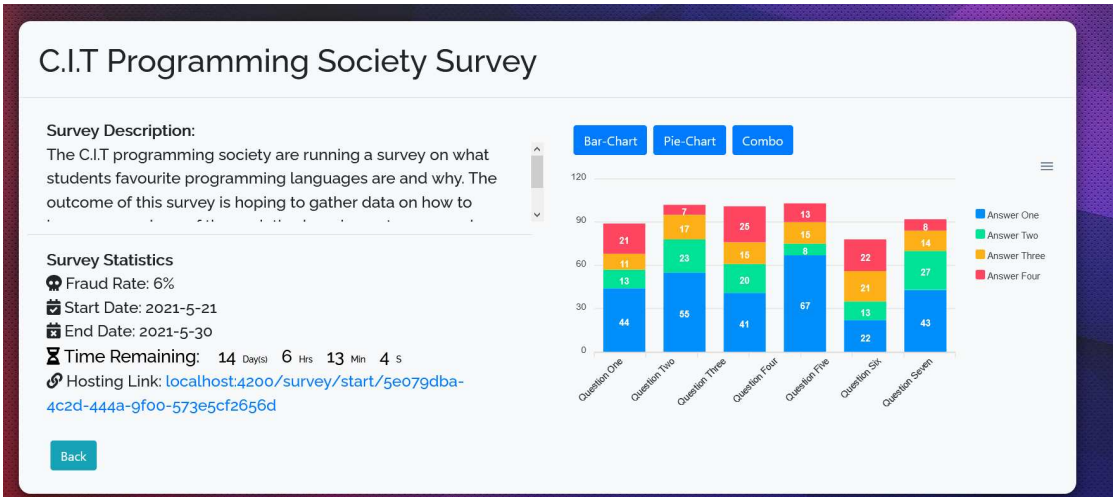

FIGURE 6.8: Survey insights page

Angular - The survey insights page contains extensive data on an individual survey. In addition to the metrics displayed on the homepage card, the surveys description, remaining time, and the link to complete the survey are also available. The surveys performance is graphed to the right, with a selection of three different charts from the Apex charts library to choose from. CSV, PNG and SVG files on each chart can also be downloaded. The data for all the users surveys have already been loaded into memory when the home page was accessed, so the data for a particular survey is passed in from there using a behaviour subject from the survey service. The survey currently being viewed on the survey insights page is then saved to local storage in order to achieve data persistence in the event of a page refresh.

```
ngOnInit(): void {

  this.surveyService.surveyGet()
    .subscribe((property) => {
      this.survey = property;
      this.survey.surveyURL = environment.surveyURL + '/survey/start/' + this.survey.s3Filename;
    })

  this.getChartData();
  this.showChart('barchart');
}
```

FIGURE 6.9: Survey insights Angular code

ASP .NET Core - As the data is passed in from the home page using a behaviour subject in the survey service, the back-end is not accessed in order to load the survey insights page.

6.1.6 Complete Survey

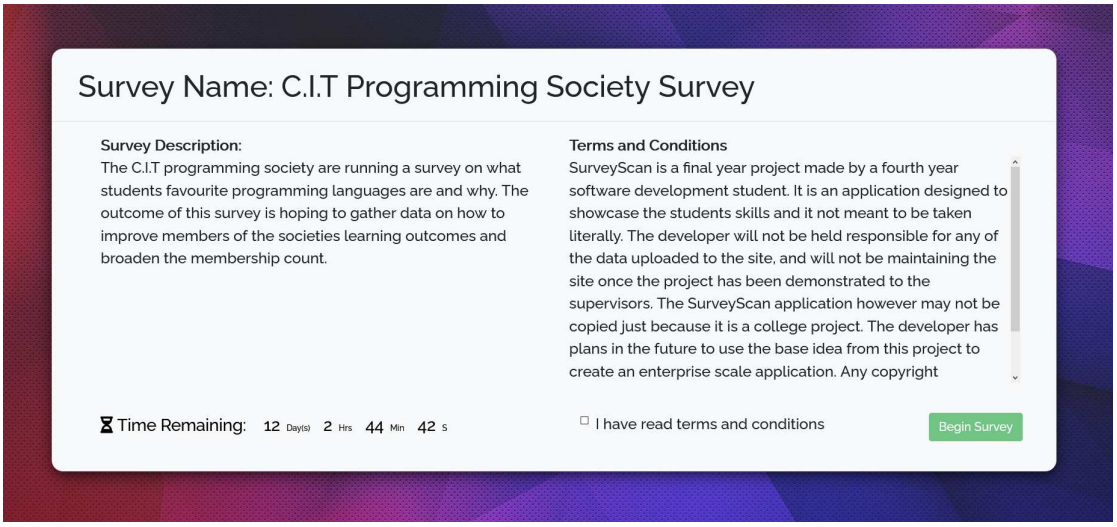

FIGURE 6.10: Complete survey landing page

Angular - In order to complete a survey, the user can access the survey link generated on its created. This link can be accessed without any user authentication, allowing anyone with the link to complete the survey. When the page loads, the survey ID is extracted from the given link using a route snapshot from the Angular activated route service. The survey service then attempts to retrieve the survey data for that survey ID from the back-end. Once the survey is loaded, the description is shown along with the terms and conditions. Once the checkbox which asks the user to confirm they have read the terms and conditions is checked, the user can now begin the survey by clicking the 'begin survey' button.

FIGURE 6.11: Survey question page

A new component is then loaded, displaying the first question and its selection of answers in checkbox form. This form is created using an Angular form group. As soon as the page loads, a timer begins and records the time taken for the user to select an answer and move onto the next question. Also recorded is the answer chosen, its position in the list of answers, the exact time of day and the length of the question in characters. These details are saved to a model called SurveyQuestion, and each for each question answered the instance of the SurveyQuestion model is added to a list in a model called Survey, which represents a completed survey as a whole. On completion of the survey, the Survey model is then sent to the back-end for processing. A component is loaded informing the user the survey has been completed.

ASP .NET Core - When a surveys link is accessed, the survey is originally retrieved from the database using the survey ID passed down from the client. The survey is then passed back up to the client. When the survey has been completed by the user, the back-end recieves it from the client and begins to process it. The data is passed to the survey from the controller, and a data table is created from the DTO.

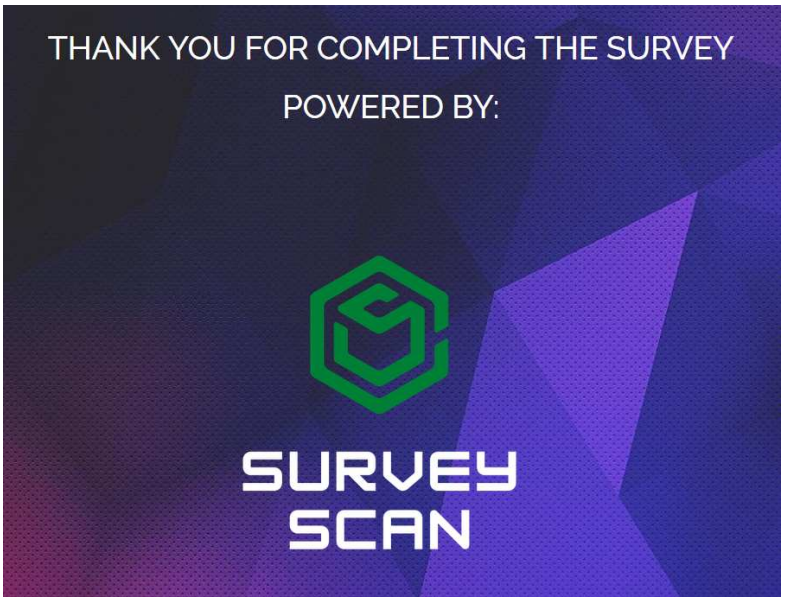

FIGURE 6.12: End of survey page

```
DataTable dt = new DataTable
{
    TableName = "tableone.xlsx"
};

dt.Columns.Add("datetime", typeof(int));
dt.Columns.Add("length_time", typeof(int));
dt.Columns.Add("placing", typeof(int));
dt.Columns.Add("q_word_count", typeof(int));

foreach(var question in completedSurvey.surveyQuestions)
{
    DataRow dr = dt.NewRow();
    long unixTime = ((DateTimeOffset)question.datetime).ToUnixTimeSeconds();

    dr[0] = unixTime;
    dr[1] = question.time;
    dr[2] = question.placing;
    dr[3] = question.questionWordCount;

    dt.Rows.Add(dr);
}

dt.AcceptChanges();
```

FIGURE 6.13: Creating data table from the DTO

The data table is then added to an XLWorkbook object, which is in turn converted into a memory stream. This stream is used to send the survey data in a Multipart-FormDataContent object to the flask application, where is can be processed by the reinforcement-learning algorithm and checked for fraudulent activity.

```
var stream = new MemoryStream();

string filename = "completedsurvey.xlsx";

using (XLWorkbook wb = new XLWorkbook())
{
    wb.Worksheets.Add(dt);
    using (stream)
    {
        wb.SaveAs(stream);
    }
}
```

FIGURE 6.14: Converting XLWorkbook to MemoryStream

Once the reinforcement learning has processed the survey, an instance of each survey as a whole is also saved to the database in a table called SurveyResults. The percentage of the number of questions in the survey which were marked as being answered fraudulently is calculated. If this percentage is greater or equal to the acceptable percentage which the user defined when creating the survey, the 'isFraud' cell is marked as being true.

When the data from the Flask application is received in the response, an instance of each question answered in the survey is saved to the database. This instance consists of the data collected on the survey from the client and the result on whether the question was answered fraudulently from the flask application.

Results Messages

	Id	CompletionInstance	Question	Answer	Datetime	Time	Placing	QuestionWordCount	IsFraud
35	CD0-DC81-4AB8-822C-533133085DAE	395AD2F7-620E-4444-ABCA-0F46EB9BF10B	How long is a normal day of college?	More than 6 hours	2021-05-05 01:28:36.7490000	0	1	36	1
36	CD0-DC81-4AB8-822C-533133085DAE	395AD2F7-620E-4444-ABCA-0F46EB9BF10B	Where do you get your lunch from?	Canteen	2021-05-05 01:28:41.6010000	4	1	33	1
37	CD0-DC81-4AB8-822C-533133085DAE	27E050FF-B7E1-4C5B-BCDF-1C61125DA48C	When do you wake up?	Between 7 and 9	2021-05-05 01:29:34.4500000	12	1	20	1
38	CD0-DC81-4AB8-822C-533133085DAE	27E050FF-B7E1-4C5B-BCDF-1C61125DA48C	Do you eat breakfast?	Always	2021-05-05 01:29:45.9640000	10	1	21	1
39	CD0-DC81-4AB8-822C-533133085DAE	27E050FF-B7E1-4C5B-BCDF-1C61125DA48C	What time do you begin college?	Between 12 and 6	2021-05-05 01:29:54.7560000	8	1	31	1
40	CD0-DC81-4AB8-822C-533133085DAE	27E050FF-B7E1-4C5B-BCDF-1C61125DA48C	How do you commute to college?	Car	2021-05-05 01:30:04.7980000	9	1	30	1
41	CD0-DC81-4AB8-822C-533133085DAE	27E050FF-B7E1-4C5B-BCDF-1C61125DA48C	How long is a normal day of college?	More than 6 hours	2021-05-05 01:30:12.6690000	7	1	36	0
42	CD0-DC81-4AB8-822C-533133085DAE	27E050FF-B7E1-4C5B-BCDF-1C61125DA48C	Where do you get your lunch from?	Canteen	2021-05-05 01:30:16.9660000	4	1	33	0
43	62-D9E3-4DEA-9A78-7C144B7BB710	2D751C70-8825-4090-B95D-28DC50BFD0C7	When do you wake up?	Before 7	2021-05-05 01:44:39.3230000	1	1	20	0
44	62-D9E3-4DEA-9A78-7C144B7BB710	2D751C70-8825-4090-B95D-28DC50BFD0C7	Do you eat breakfast?	Rarely	2021-05-05 01:44:40.7110000	1	1	21	0
45	62-D9E3-4DEA-9A78-7C144B7BB710	2D751C70-8825-4090-B95D-28DC50BFD0C7	What time do you begin college?	Before 9	2021-05-05 01:44:42.2010000	1	1	31	0
46	62-D9E3-4DEA-9A78-7C144B7BB710	2D751C70-8825-4090-B95D-28DC50BFD0C7	How do you commute to college?	Bike	2021-05-05 01:44:43.3310000	1	1	30	0
47	62-D9E3-4DEA-9A78-7C144B7BB710	2D751C70-8825-4090-B95D-28DC50BFD0C7	How long is a normal day of college?	More than 6 hours	2021-05-05 01:44:45.8370000	2	1	36	0
48	62-D9E3-4DEA-9A78-7C144B7BB710	2D751C70-8825-4090-B95D-28DC50BFD0C7	Where do you get your lunch from?	From Home	2021-05-05 01:44:47.2890000	1	1	33	0
49	62-D9E3-4DEA-9A78-7C144B7BB710	F94CF452-ABB1-44F7-82CD-3C9FD4F7DAE4	When do you wake up?	Between 7 and 9	2021-05-05 01:45:50.5900000	1	1	20	1
50	62-D9E3-4DEA-9A78-7C144B7BB710	F94CF452-ABB1-44F7-82CD-3C9FD4F7DAE4	Do you eat breakfast?	Always	2021-05-05 01:45:52.0700000	1	1	21	1
51	62-D9E3-4DEA-9A78-7C144B7BB710	F94CF452-ABB1-44F7-82CD-3C9FD4F7DAE4	What time do you begin college?	Between 12 and 6	2021-05-05 01:45:53.5800000	1	1	31	1
52	62-D9E3-4DEA-9A78-7C144B7BB710	F94CF452-ABB1-44F7-82CD-3C9FD4F7DAE4	How do you commute to college?	Bus	2021-05-05 01:45:56.1270000	2	1	30	1
53	62-D9E3-4DEA-9A78-7C144B7BB710	F94CF452-ABB1-44F7-82CD-3C9FD4F7DAE4	How long is a normal day of college?	Less than 3 hrs	2021-05-05 01:45:57.7390000	1	1	36	1
54	62-D9E3-4DEA-9A78-7C144B7BB710	F94CF452-ABB1-44F7-82CD-3C9FD4F7DAE4	Where do you get your lunch from?	Local Shop	2021-05-05 01:45:59.0080000	1	1	33	1
55	62-D9E3-4DEA-9A78-7C144B7BB710	82130A04-C7E1-48BA-811B-7FA2068B57C8	When do you wake up?	Between 7 and 9	2021-05-05 01:46:59.0500000	1	1	20	1

FIGURE 6.15: Instance of each question saved to SurveyQuestionResults table

Results Messages

	InstanceId	IsFraud	CompletionInstance	SurveyId
1	20CD90E6-E31C-45A4-A373-3CAAD1DBF954	1	1A53179B-805B-496F-8003-09F46FCF32ED	8D0EDCD0-DC81-4AB8-822C-533133085DAE
2	CE760C1D-DE03-4799-B46D-4EF61C5AD623	1	3D16D993-7405-439D-A337-0B351985B0DF	8D0EDCD0-DC81-4AB8-822C-533133085DAE
3	0EE7542B-86D8-4619-8765-344635DBEB1B	1	395AD2F7-620E-4444-ABCA-0F46EB9BF10B	8D0EDCD0-DC81-4AB8-822C-533133085DAE
4	DBB0FA58-AB61-4343-ABB5-E7F017EE8510	1	27E050FF-B7E1-4C5B-BCDF-1C61125DA48C	8D0EDCD0-DC81-4AB8-822C-533133085DAE
5	B9F696CD-65F4-4E9A-B61C-B184D791B440	1	EA174022-1DC1-4C56-993C-4920FBA145CA	8D0EDCD0-DC81-4AB8-822C-533133085DAE
6	01402191-3092-4893-BDF3-8E0AF0B6DF5E	1	D2B1DCF7-69E0-4CC9-8357-973E811E174D	8D0EDCD0-DC81-4AB8-822C-533133085DAE
7	BB646E1E-9847-4CF1-8236-2FFC453C3909	1	291DB6CE-5CBD-493E-A48F-BC1205F275C4	8D0EDCD0-DC81-4AB8-822C-533133085DAE

FIGURE 6.16: Instance of each survey saved to SurveyResults table

6.1.7 Reinforcement Learning

The reinforcement learning is hosted in a Flask application which recieves the excel file from the back-end and passes it to the model for evaluation. The method of reinforcement learning eventually implemented used the Keras API and the DQN algorithm [54]. The approach taken was now based on classification as opposed to anomaly detection.

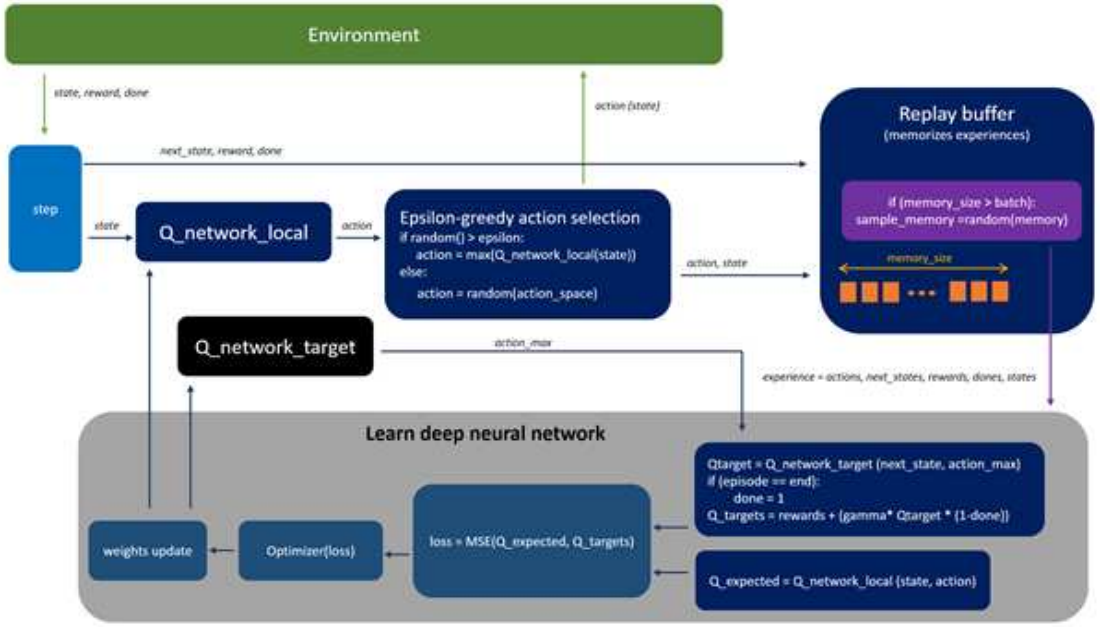

FIGURE 6.17: DQN algorithm diagram

- **Original Approach - Anomaly Detection** - The original approach focused on using anomaly detection to seek out values in each of the rows which represented what the algorithm determined to be an anomaly. Depending on the number of anomalies across all the answers in a given survey instance the survey would be classified as being answered fraudulently or truthfully.

- **New Approach - Classification** - The new approach focuses on using binary classification to classify each row in the data-set, representing a survey question answered, as either fraudulent or not fraudulent. The fraud acceptance rate is defined by the user when creating the survey. The number of questions classified as fraudulent are then calculated as a percentage of the overall number of questions in the survey. If this percentage is greater than the fraud acceptance rate, the survey is classified as being answered fraudulently.

```python
app = Flask(__name__)

ALLOWED_EXTENSIONS = set(['xlsx"'])

def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS

# Get the uploaded csv file
@app.route("/postcsv", methods=['POST'])
def uploadFiles():

      # get the uploaded file
      file = request.files['file']
      if file and allowed_file(file.filename):

            df1 = pd.read_excel(request.files.get('file'), engine='openpyxl')
            return evaluate(df1)
      else:
            return "File is empty!"
```

FIGURE 6.18: Flask controller for RL process

The approach therefore used a semi-supervised model, however this resulted in the recurrence of the issue which led to choosing reinforcement learning in the first place - a lack of data for training. This issue was solved by creating fake data using Mockaroo, a random data generator website.

```python
def mnist_dqn():
    logger.configure(dir='./logs/mnist_dqn', format_strs=['stdout', 'tensorboard'])
    env = MnistEnv(images_per_episode=1)
    env = bench.Monitor(env, logger.get_dir())

    model = deepq.learn(
        env,
        "mlp",
        num_layers=1,
        num_hidden=64,
        activation=tf.nn.relu,
        hiddens=[32],
        dueling=True,
        lr=1e-4,
        total_timesteps=int(1.2e5),
        buffer_size=10000,
        exploration_fraction=0.1,
        exploration_final_eps=0.01,
        train_freq=4,
        learning_starts=10000,
        target_network_update_freq=1000,
    )

    model.save('dqn_mnist.pkl')
    env.close()

    return model

start_time = time.time()
dqn_model = mnist_dqn()
print("DQN Training Time:", time.time() - start_time)
```

FIGURE 6.19: DQN algorithm in Flask application

6.1.8 Cloud Hosting

Now that the applications development had been finished, it was time to migrate to the cloud. This operation was made much easier due to the advice given by a co-worker from last years internship. His expertise in cloud computing was highly beneficial and resulted in a large amount of knowledge being learned about the proper and improper methods of migrating to the cloud.

Environment name ▲	Health ▽	Application name ▽	Date created ▽	Last modified ▽	URL ▽	Running versions ▽	Platform ▽
SurveyScrubAPI-prod	Green	SurveyScrubAPI	2021-01-15 16:08:09 UTC+0000	2021-01-15 16:14:44 UTC+0000	surveyscrubapi-prod.eu-west-1.elasticbeanstalk.com	v20210115155556	IIS 10.0 running on 64bit Windows Server 2019

FIGURE 6.20: Elastic Beanstalk Instance of Back-end

The Angular application was deployed to an S3 bucket on AWS. This is due to the fact that the application is static, and differed from the architecture proposed in the research phase. The back-end and SQL server were deployed to elastic beanstalk instances. The only code change in order to prepare the application for the cloud was to edit the environment variables in the Angular application to coincide with the new URL for the back-ends REST API.

6.2 Testing/Results

As can be seen in the image below, the reinforcement algorithm is successfully training and testing using the fake data-set generated with Mockaroo. Taking into account the datetime, length of time taken to answer the survey, the placing of the question in the survey and the word count in the question, the algorithm classifies whether the question had been answered fraudulently, giving the 'is_fraud' column a 1 in the case of fraud and a 0 in the case of not fraud. The final data-frame in the screenshot from the terminal represent the survey which has just been completed and passed through.

```
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5001/ (Press CTRL+C to quit)
          datetime  length_time  placing  q_word_count  is_fraud
0    1615974054000       4.8270        2            18     False
1    1607340057000       4.3683        1            43      True
2    1593970061000       8.1060        8            90     False
3    1594501835000       4.9008        5            83      True
4    1609228317000       3.9191        7            40      True
..             ...          ...      ...           ...       ...
995  1609436013000      11.9429        5            55     False
996  1602378260000      10.1459        8            14     False
997  1615971378000      14.1564        4            81     False
998  1605768181000       3.9872        9            53      True
999  1603543146000      12.7977        9            38     False

[1000 rows x 5 columns]
     datetime  length_time  placing  q_word_count
0  1621283477            8        1            36
1  1621283486            9        2            35
2  1621283489            2        3            33
3  1621283501           11        4            48
4  1621283503            2        5            41
5  1621283518           15        6            63
DQN TRAIN =  766
DQN TEST =  234
     datetime  length_time  placing  q_word_count  is_fraud
0  1621283477            8        1            36         1
1  1621283486            9        2            35         0
2  1621283489            2        3            33         1
3  1621283501           11        4            48         0
4  1621283503            2        5            41         1
5  1621283518           15        6            63         0
127.0.0.1 - - [17/May/2021 21:31:59] "POST /postcsv HTTP/1.1" 200 -
```

FIGURE 6.21: DQN Reinforcement Learning Results

Chapter 7

Discussion and Conclusions

This chapter reflects on the project and report, discussing what could have been done differently as well as what may be done in the future. In order to achieve full transparency on the final outcome of the project, the chapter is split into a multitude of sections.

1. **Solution Review** - Reflecting on the outcome of the solution itself.
2. **Project Review** - Reflecting on the project as a whole.
3. **Conclusions** - Discussing the conclusions that can be made from the projects results.
4. **Future Work** - Discussing future work.

7.1 Solution Review

The finished solution as a whole worked well as a platform for creating and hosting surveys, however fell short in its main goal to detect when fraudulent entities are answering the surveys. This shortcoming can be attributed to several reasons outlined below.

1. The Meta-AAD reinforcement learning algorithm initially proposed could not be used due to technical difficulties.
2. The classification reinforcement learning algorithm used was not accurate enough in detecting fraudulent activity.
3. A larger amount of data needs to be collected from each survey in order to identify fraudulent activity.

While the outcome of the reinforcement learning side of the project was a disappointment, the opposite can be said for the application itself. The end solution turned out above expectations, and the majority of functionality was developed in the given time frame. A large amount of technical knowledge was gained which will be monumental going forward in a career in software development, some of which are listed below.

1. **Web Architecture** - Developing a full scale web application resulted in excellent experience on how the difference entities and frameworks work an co-operate together.

2. **Cloud Hosting** - Deploying the application to the cloud was an excellent learning curve as this was a first time experience.

   REST APIs - While REST APIs had been used previously, the amount used in the project provided experience on the underlying theory of REST and it works.

   Data Streaming - The extensive use of C# streams such as file and memory streams provided excellent knowledge on the uses and theory behind data streaming.

   Reinforcement Learning - While the reinforcement learning used was based off pre-existing models, the project was began with zero experience in the field of reinforcement learning. A satisfactory amount of knowledge was gained on internal topics such as Markov decision processes and OpenAI Gym.

7.2 Project Review

Overall the attitude towards the outcome of the project as a whole was extremely positive. One element of the project development which went particularly well was the speed in which the web application was developed, given the large amount of tasks which had to be completed. This is down to a combination of adhering to the scrum-ban methodology mentioned in chapter 4, as well as having the project architecture and prototypes well defined in the research phase.

Despite this, there were several challenges faced throughout development, some of which in hindsight could have been approached better. One in particular was the architecture of the reinforcement-learning model. Due to a combination of a lack of research regarding the models cloud hosting and a lack of knowledge on the subject, the end result did not live up to expectations. However, due to the complexity of the subject and given that the project was only being developed at an undergraduate level, there is still a sense of

achievement that the reinforcement learning was implemented to a stage where it was operational.

Another issue which may have been handled better was source control. While a GitHub repository was set up and used at the beginning of the project, falling out of the habit of pushing code to the repository before shutting down the computer resulted in a large amount of working being lost on two occasions during development.

The issues that arose from the reinforcement learning also resulted in a silver lining in terms of how time was managed. As time management was not a skill which was not well implemented in past projects, it came as a welcome revelation how well the Kanban system was worked with throughout development, even when major issues arose.

7.3 Conclusion

In terms of background, problem description and the solution approach which were undertaken during the implementation of this project, several conclusions, both primary and secondary can be made.

Primary Conclusions

1. Survey fraud is a major problem in the market-research industry.
2. Click fraud is one of the biggest threats to the quality of data gathered using online surveys. The statistics I found regarding its impact online were staggering and it is clear it is a problem which is going to be relevant for years to come.
3. While click fraud is a problem for industries involving online survey fraud, there is another more sinister fallout from the practice which is being brushed under the rug. The infringement of click farm workers human rights in the developing world is something I did not expect to come across in my research. I was shocked to read about the poor conditions and low wages workers experience.
4. Machine-learning is a viable solution to help combat click fraud, however an alternative approach to using reinforcement learning may be better suited.

Secondary Conclusions

1. The importance of research, planning and prototyping cannot be underestimated in software design.
2. Using a code first approach to database manipulation using Entity Framework Core and LINQ is preferred.
3. Angular and ASP .NET Core compliment one another well as a client and back-end.
4. AWS as a cloud host is a preferred option due to its extensive features and rich documentation.

7.4 Future Work

Due to the ever growing value of online data collection, I believe this project certainly has potential for future work, even if just to exist as a prototype for a larger, industry standard application. Future work which I can see improving this project would involve both the addition and upgrade of several elements. These upgrades and additions would not only improve the main functionality, but also the look and feel of the application.

A large amount of work could be put into the machine learning side of the project. Ideally, it would a reinforcement learning model which scored a higher accuracy, or else a different form of learning such as supervised or unsupervised. In correlation to this, the amount of data points should be gathered from each survey in order to improve classification.

While the interface and theme turned out well, they are very reliant on the bootstrap library, and therefore would benefit greatly from being restyled by a professional UI designer. Also, the charts may be improved by adding more complexity and interactivity to them.

Bibliography

[1] Oli, "Click farms: What are they what are they for?" [Online]. Available: https://www.clickcease.com/blog/click-farms-what-are-they-what-are-they-for/

[2] ——, "Click farms: What are they what are they for?" [Online]. Available: https://www.clickcease.com/blog/click-farms-what-are-they-what-are-they-for/

[3] N. Perlroth, "Researchers call out twitter celebrities with suspicious followings." [Online]. Available: https://bits.blogs.nytimes.com/2013/04/25/researchers-call-out-twitter-celebrities-with-suspicious-followings/

[4] OpinionRoute, "Us survey farm fraud discovered by cleanid." [Online]. Available: https://www.opinionroute.com/post/survey-farm-frau-discovered-by-cleanid-in-the-usa

[5] ——, "How low-paid workers at 'click farms' create appearance of online popularity." [Online]. Available: https://www.theguardian.com/technology/2013/aug/02/click-farms-appearance-online-popularity

[6] I. Manor, "Amid election fraud concerns: Could robots decide a us election?" [Online]. Available: https://www.jpost.com/opinion/amid-election-fraud-concerns-could-robots-decide-a-us-election-648353

[7] A. P. Team, "Ad fraud and the rise of click farms." [Online]. Available: https://www.clickcease.com/blog/click-farms-what-are-they-what-are-they-for/

[8] N. Joshi, "Machine learning for anomaly detection." [Online]. Available: https://www.allerin.com/blog/machine-learning-for-anomaly-detection

[9] A. K. S. Anitha Ramchandran, "Unsupervised anomaly detection for high dimensional data—an exploratory analysis." [Online]. Available: https://www.sciencedirect.com/topics/engineering/anomaly-detection

[10] Udacity, "Reinforcement learning basics." [Online]. Available: https://www.youtube.com/watch?v=2xATEwcRpy8&ab_channel=Udacity

[11] M. W. X. H. Daochen Zha, Kwei-Herng La, "Meta-aad: Active anomaly detection withdeep reinforcement learning." [Online]. Available: https://arxiv.org/pdf/2009.07415.pdf

[12] Z. C. Isaac Galatzer-Levy, Kelly V. Ruggles, "Data science in the research domain criteria era: Relevance of machine learning to the study of stress pathology, recovery, and resilience." [Online]. Available: https://www.researchgate.net/figure/\Reinforcement-learning-schematic-Reinforcement-learning-RL-can-be-formulated-as-a_fig4_322424392

[13] L. Dignan, "Top cloud providers in 2021: Aws, microsoft azure, and google cloud, hybrid, saas players." [Online]. Available: https://www.zdnet.com/article/the-top-cloud-providers-of-2020-aws-microsoft-azure-google-cloud-hybrid-saas/

[14] D. Hein, "What is elasticity, and how does it affect cloud computing?" [Online]. Available: https://solutionsreview.com/cloud-platforms/what-is-elasticity-and-how-does-it-affect-cloud-computing/#:~:text=Elasticity%20refers%20to%20the%20dynamic,enterprise%20needs%20to%20run%20something.

[15] J. Ben-David, "Cloud elasticity vs cloud scalability." [Online]. Available: https://blog.turbonomic.com/blog/on-technology/cloud-elasticity-vs-cloud-scalability

[16] Microsoft, "Asp .net documentation." [Online]. Available: https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-5.0

[17] J. Chen, "What is an api why does it matter for social media?" [Online]. Available: https://sproutsocial.com/insights/what-is-an-api/

[18] RapidAPI, "Http request methods – what are http requests?" [Online]. Available: https://rapidapi.com/blog/api-glossary/http-request-methods/

[19] ICCC, "International conference on communications in china." [Online]. Available: https://iccc2019.ieee-iccc.org/#:~:text=The%208th%20IEEE%2FCIC%20International,workshops%2C%20tutorials%20and%20invited%20tracks.

[20] S. Rae, "Welcome to international fraud prevention conference 2021 – the eu's fastest growing cybersec fincrime congress." [Online]. Available: https://www.internationalfraudprevention.com/

[21] A. G. S. M. S. Jain, F. Jindal, "Identification of click fraud and review of existing detection algorithms." [Online]. Available: https://ieeexplore.ieee.org/\document/8987878

[22] A. C. I. H. E. H. A. K. R. Mouawi, M. Awad, "Towards a machine learning approach for detecting click fraud in mobile advertizing." [Online]. Available: https://ieeexplore.ieee.org/document/8605973

[23] M. S. D. H. H.Zheng, N.Li, "Fake reviews tell no tales? dissecting click farming in content-generated social networks." [Online]. Available: https://ieeexplore.ieee.org/document/8330469

[24] J. T. Wells, "Internet fraud casebook." [Online]. Available: https://www.amazon.com/Internet-Fraud-Casebook-World-Deceit/dp/0470643633

[25] C. Hadnagy, "Social engineering: The science of human hacking, 2nd edition." [Online]. Available: https://www.amazon.co.uk/Social-Engineering-Science-Human-Hacking/dp/111943338X

[26] W. V. Bart Baesens, Veronique Van Vlasselaer, "Fraud analytics using descriptive, predictive, and social network techniques: A guide to data science for fraud detection." [Online]. Available: https://www.amazon.com/Analytics-Descriptive-Predictive-Network-Techniques/dp/1119133122

[27] OpinionRoute. [Online]. Available: https://www.opinionroute.com/

[28] SurveyMonkey. [Online]. Available: https://www.surveymonkey.com/

[29] Qualtrics. [Online]. Available: https://www.qualtrics.com/uk/

[30] Forter. [Online]. Available: https://www.forter.com/

[31] DataDome, "Real-time bot protection." [Online]. Available: https://datadome.co/

[32] ITV, "Executing a police raid on a fraud gang — fraud: How they steal your bank account." [Online]. Available: https://www.youtube.com/watch?v=MMRIF5vEWuM

[33] KyleSulerud, "Google ads: The truth about click fraud." [Online]. Available: https://www.youtube.com/watch?v=ZNn7dARUrlw&ab_channel=KyleSulerud

[34] Y. Finance, "China has a secret 50 billion dollar click farm problem." [Online]. Available: https://www.youtube.com/watch?v=W8KG9nOPdOo&ab_channel=YahooFinance

[35] Anon, "Never mind the algorithms: the role of click farms and exploited digital labor in trump's election." [Online]. Available: https://www.casilli.fr/2016/11/20/never-mind-the-algorithms-the-role-of-exploited-digital-labor-and-global-click-farms-in-trumps-el

[36] I. Pastoor, "A look behind the scenes of click farms." [Online]. Available: https://www.diggitmagazine.com/articles/look-behind-scenes-click-farms

[37] H. S. Norbert Schwarz, Robert M Groves, "Survey methods." [Online]. Available: https://www.researchgate.net/profile/Norbert_Schwarz2/publication\/232562918_Survey_methods/links/55db515d08aed6a199ac5d02/Survey-methods.pdf

[38] ICSM, "Evolution of surveying and surveying technology." [Online]. Available: https://www.icsm.gov.au/education/fundamentals-land-ownership-land-boundaries-and-surveying/surveyors-and-surveying-0

[39] R. M. Groves, "Three eras of survey research." [Online]. Available: https://www.uvm.edu/~dguber/POLS234/articles/groves.pdf

[40] Anon, "A quick history of online surveys." [Online]. Available: https://www.irazoo.com/blog/a-quick-history-of-online-surveys/

[41] J. L. Liran Einav, "The data revolution and economic analysis." [Online]. Available: https://www.journals.uchicago.edu/doi/full/10.1086/674019

[42] W. H. Johnston, "Lottery racing sweepstake." [Online]. Available: https://patents.google.com/patent/US5518239A/en

[43] J. Hopper, "Six ways to identify bad survey data." [Online]. Available: https://verstaresearch.com/blog/six-ways-to-identify-bad-survey-data/

[44] Qualtrics, "Security survey options." [Online]. Available: https://www.qualtrics.com/support/survey-platform/survey-module/survey-options/survey-protection/

[45] K. C. I. B. S. S. I. N. R. S. G.S. Thejas, Kianoosh G Boroojeni, "Deep learning-based model to fight against ad click fraud." [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3299815.3314453

[46] Microsoft, "Code editing. redefined." [Online]. Available: https://code.visualstudio.com/

[47] W. Pyke, "Visual studio vs visual studio code: What are the differences?" [Online]. Available: https://stackshare.io/stackups/visual-studio-vs-visual-studio-code

[48] Microsoft, "Querying data." [Online]. Available: https://docs.microsoft.com/en-us/ef/core/querying/

[49] C. University. [Online]. Available: https://olinuris.library.cornell.edu/content/seven-steps-research-process

[50] M. F. MurisLage Junior, "Variations of the kanban system: Literature review and classification." [Online]. Available: https://www.sciencedirect.com/\science/article/pii/S0925527310000198

[51] N. J. L. Rising, "The scrum software development process for small teams." [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=\&arnumber=854065

[52] ——. [Online]. Available: https://www.kanbanize.com

[53] P. Project. [Online]. Available: https://pencil.evolus.vn/

[54] brthor, "Can reinforcement learning be used for classification." [Online]. Available: https://www.brthor.com/yt/?v=td0TjL5tznc

Appendix A

Code Snippets

```
private currentUserSource = new ReplaySubject<User>(1);
currentUser$ = this.currentUserSource.asObservable();

constructor(private http: HttpClient) { }

register(user: FormData) {
  return this.http.post(this.baseUrl + '/register', user);
}

login(user: UserLogin) {
  return this.http.post(this.baseUrl + '/login', user).pipe(
    map((response: User) => {
      const user = response;
      if (user) {
        localStorage.setItem('user', JSON.stringify(user));
        this.currentUserSource.next(user)
      }
    })
  );
}

setCurrentUser(user: User) {
  this.currentUserSource.next(user);
}
```

FIGURE A.1: Methods for logging in and saving to local storage

```
register() {

  const formData = new FormData();

  formData.append('FirstName', this.registerForm.get('inputFirstName').value);
  formData.append('LastName', this.registerForm.get('inputFirstName').value);
  formData.append('Password', this.registerForm.get('inputPassword').value);
  formData.append('PasswordCheck', this.registerForm.get('inputPasswordCheck').value);
  formData.append('Company', this.registerForm.get('inputCompany').value);
  formData.append('Sector', this.registerForm.get('inputSector').value);
  formData.append('Email', this.registerForm.get('inputEmail').value);

  if (this.registerForm.get('inputPassword').value == this.registerForm.get('inputPasswordCheck').value) {
    this.authService.register(formData).subscribe(next => {

      this.userLogin.email = this.registerForm.get('inputEmail').value;
      this.userLogin.password = this.registerForm.get('inputPassword').value;

      this.login();

    }, error => {
      this.alertifyService.error('Registration failed. Please try again.');
    });
  }
  else {
    this.alertifyService.error("Passwords do not match");
  }
}
```

FIGURE A.2: Registration code for SurveyScan

```
public async Task<bool> CreateNewSurvey(SurveyDto surveyDto)
{
    var S3FilenameGuid = Guid.NewGuid();

    Survey survey = new Survey
    {
        SurveyName = surveyDto.surveyName,
        StartDate = surveyDto.startDate,
        EndDate = surveyDto.endDate,
        S3Filename = S3FilenameGuid,
        CompanyId = new Guid(surveyDto.companyId),
        AllowedFraudRate = surveyDto.allowedFraudRate,
        SurveyDescription = surveyDto.surveyDescription
    };

    var file = surveyDto.file;
    string newFilename = survey.S3Filename.ToString() + ".xlsx";

    AmazonS3Uploader amazonS3Uploader = new AmazonS3Uploader();
    await amazonS3Uploader.UploadFileAsync(file, newFilename);

    return await _surveyRepository.CreateNewSurvey(survey);
}
```

FIGURE A.3: Create survey back-end code

```
ngOnInit(): void {

  this.tcForm = this.formBuilder.group({
    acceptTerms: [false,
      Validators.requiredTrue]
  })

  //Retrieves surveyid from the given url
  this.surveyid = this.route.snapshot.paramMap.get("surveyid");

  this.surveyService.getSurvey(this.surveyid).subscribe(Response => {

    // If Response comes function hideloader() is called
    if (Response) {
      this.hideloader();
    }

    this.survey = Response;
    this.dataDisplay = this.survey.data;
    this.survey = Response;

  }, error => {
      this.alertifyService.error(error);
    });
}
```

FIGURE A.4: Complete survey landing code

```
HttpClient client = new HttpClient();

//Prepare request header
client.DefaultRequestHeaders.Accept.Clear();
client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/x-www-form-urlencoded"));
client.DefaultRequestHeaders.Add("Api-Key", "xxxyyyzzz");

//cache
CacheControlHeaderValue cacheControl = new CacheControlHeaderValue
{
    NoCache = true
};
client.DefaultRequestHeaders.CacheControl = cacheControl;

byte[] bytes = stream.ToArray();
HttpContent fileContent = new ByteArrayContent(bytes);
fileContent.Headers.ContentType = MediaTypeHeaderValue.Parse("application/vnd.openxmlformats-officedocument.spreadshee

List<ProcessedQuestions> processedQuestions = new List<ProcessedQuestions>();

try {

    var response = await client.PostAsync(apiUrl, new MultipartFormDataContent
    {
        {fileContent, "\"file\"", "\"" + filename +"\""}
    });

    var responseJSONString = await response.Content.ReadAsStringAsync();
    processedQuestions = Newtonsoft.Json.JsonConvert.DeserializeObject<List<ProcessedQuestions>>(responseJSONString);

}
```

FIGURE A.5: Sending data to Flask application using MultipartFormDataContent

```
Guid completionInstance = Guid.NewGuid();

foreach (CompletedQuestions questions in completedSurvey.surveyQuestions)
{

    SurveyQuestionResults surveyQuestionResults = new SurveyQuestionResults
    {
        InstanceId = new Guid(),
        CompletionInstance = completionInstance,
        SurveyId = completedSurvey.surveyId,
        Question = questions.question,
        Answer = questions.answer,
        Datetime = questions.datetime,
        Time = questions.time,
        Placing = questions.placing,
        QuestionWordCount = questions.questionWordCount,
        IsFraud = Convert.ToBoolean(Convert.ToInt16(processedQuestions[count].is_fraud))
    };

    if (processedQuestions[count].is_fraud == 1)
    {
        fraudCount++;
    }

    count++;

    await _surveyRepository.SaveSurveyQuestionResults(surveyQuestionResults);
}
```

FIGURE A.6: Code saving each question to the database

```
        if (fraudCount > count/fraudPercentage)
        {
            isSurveyFraud = true;
        }

        SurveyResults surveyResults = new SurveyResults
        {
            InstanceId = Guid.NewGuid(),
            SurveyId = completedSurvey.surveyId,
            CompletionInstance = completionInstance,
            IsFraud = isSurveyFraud
        };

        await _surveyRepository.SaveSurveyResults(surveyResults);

        QuestionsDto surveyQuestions = new QuestionsDto();
        return surveyQuestions;
    }
}
```

FIGURE A.7: Code saving survey to SurveyQuestionResults table

Appendix B

Wireframe Models

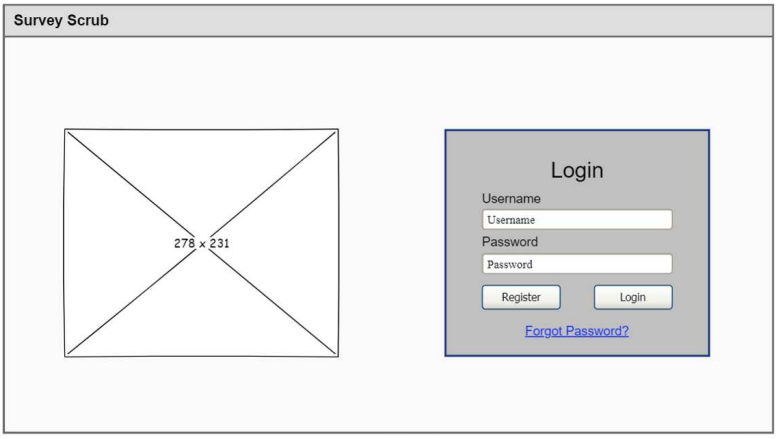

FIGURE B.1: Login Page Wireframe

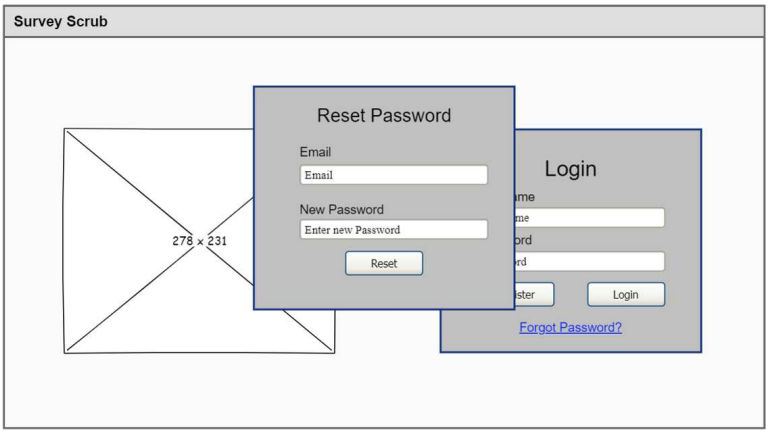

FIGURE B.2: Forgot Password Page Wireframe

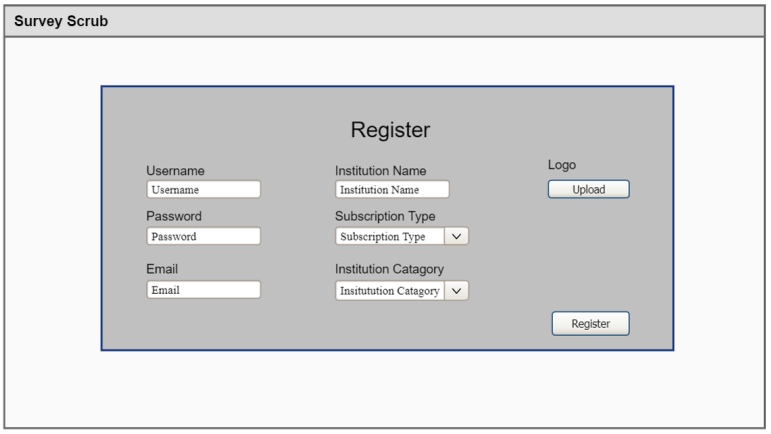

FIGURE B.3: Register Page Wireframe

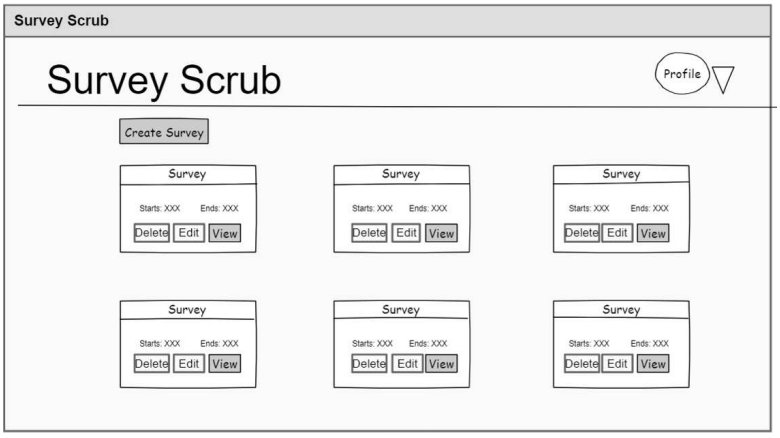

FIGURE B.4: Home Page Wireframe

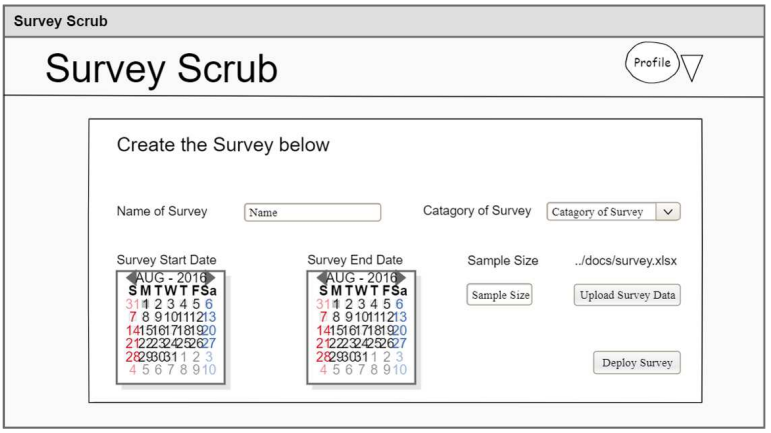

FIGURE B.5: Create Survey Page Wireframe

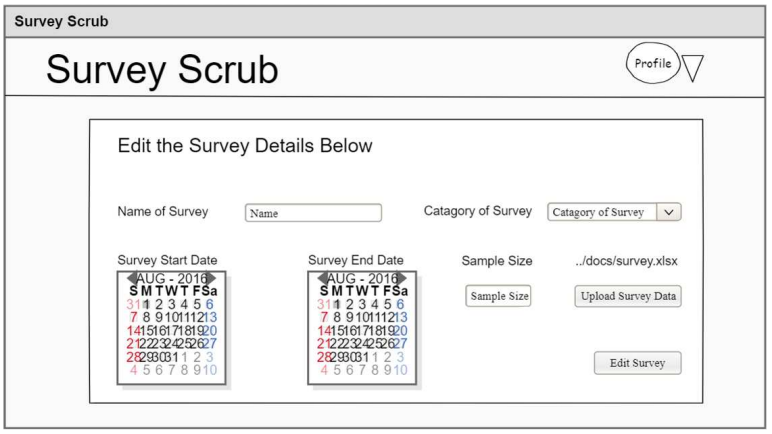

FIGURE B.6: Edit Survey Page Wireframe

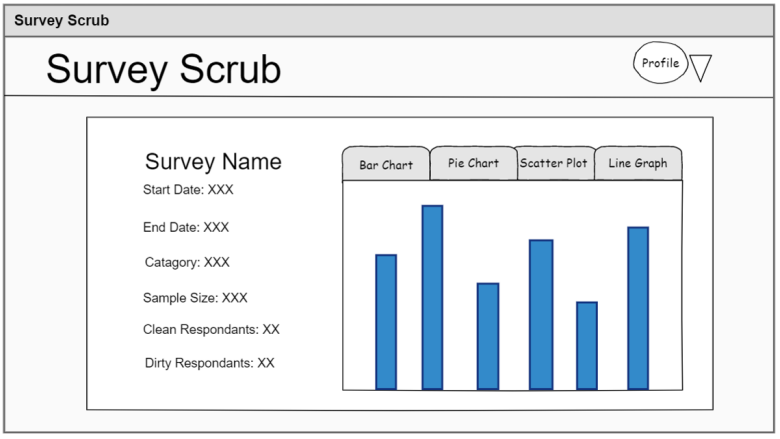

FIGURE B.7: View Survey Page Wireframe

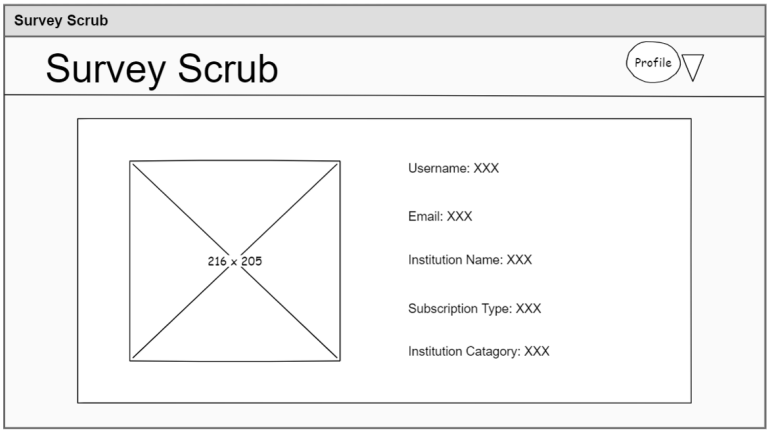

FIGURE B.8: View Profile Page Wireframe

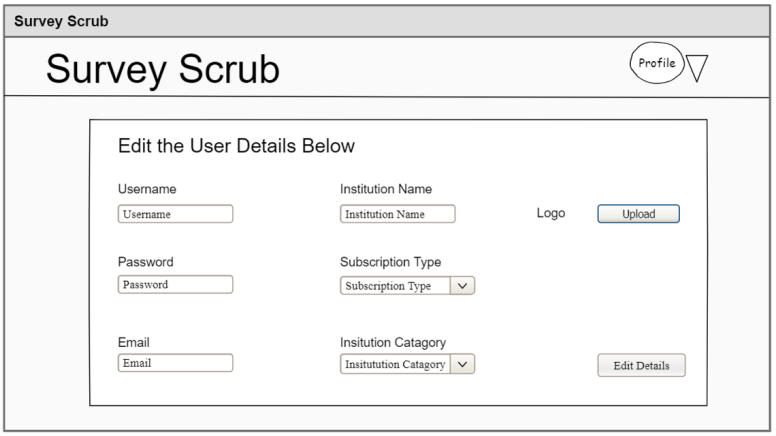

FIGURE B.9: Edit User Page Wireframe