School of Electronic Engineering
and Computer Science

Final Report

Programme of study:
Software Engineering for Business
BSc

Project Title:
Social Welfare Helper

Supervisor:
Dr. Raul Mondragon

Student Name:
Salman Saeed Filli

Final Year
Undergraduate Project 2022/23

Date: 02/05/2023

Queen Mary
University of London

Abstract

There are many applications which are designed to improve the lives of others in specific niches by finding relevant opportunities. There are also applications that allow you to access different types of help and support from different personnel. However, to my knowledge, none of the applications out there have been able to successfully combine these elements of social welfare and improving people's lives in a multitude of areas as supposed to one specific area.

The objective is to develop a fully functioning web application which gives users such as former offenders, or people who have been out of touch in society for a long term and may not know how to access certain opportunities, the ability to easily find different types of programmes, and support from experts in order to improve their lives and re-integrate them back into society. The application will allow social workers and different types of staff members to connect with users through appointment booking (online or in person) and they will be able to post valuable information for users to see in a social media like format as well as release programmes in fields such as mental health, career related etc. Upon the user booking an appointment with a staff member of their choosing, if their appointment is in person they are given the directions from their location to the location of the appointment as well as the time and distance to the appointment.

These are for users who may find the abundance of information that is on the internet overwhelming and are looking for a central system that can satisfy all their needs in different sectors and be able to have access to experts in different fields.

Through testing and user feedback, the conclusion was gathered that the resulting web application fulfils the main objectives and requirements.

Contents

Table of Figures

Chapter 1: **Introduction**

1.1 Background

We live in an age where there is an abundance of information on the internet. It has been estimated that there are 5million terabytes of data (Brendan, 2022). Although this can be seen as a good thing, it also creates a lot of distractions when it comes to finding what you are looking for. It also makes it easy to get swayed and lose focus, which can actually hinder productivity. The philosopher Alain de Bottom describes the internet as “overwhelming and asphyxiating”. (Barrett, 2016).

If this is the case for average user, this would be even worse for the people for who have been out of society for a long time, such as offenders, and haven’t had as much access to technology and the internet. If they are overwhelmed with the internet they are less likely to use it and will not be able to benefit from the Information that is out there and get the relevant support from people such as social workers, officers etc. This could have a negative impact on their lives and lead to things such as re offending.

Re-Offending is a common trait among offenders that have completed their sentence. Studies have shown that within the UK, “75% of ex-inmates re-offend within 9 years of release and 39% reoffend within 12months” (University of Birmingham, 2013).

There is plenty of support, programs and opportunities that help with these sort of issues to improves their lives, which I will discuss in the literature review section, that can be can be found on the internet. However, for people who have not been in touch with the internet and technology, this will be a difficult task to achieve therefore It is important that these programs and support are easy to find and access. This could improve their lives and potentially decrease the likelihood of them being re-incarcerated (Monika, 2022).

1.2 Problem Statement

As a Software Engineer, my goal is to develop a system which users, such as offenders or anyone that has been out of touch with society for a long time and doesn’t have much experience with navigating through internet, can use to find programs and support that they are looking in a user friendly, easy to use application and get the relevant support that they need.

The prison population is projected to increase “to 98500 by March 2026, this includes prison population of adult males, adult females and children”. This also includes the prison population that are aged 50 and above (Ministry of Justice, 2021). This is a large increase, especially if you consider the current re-offending rates. As the world becomes more and more digital, these groups of people may become out of touch with the digital world and will be difficult to navigate the internet in order to get find what they are looking for. This makes it more important to develop an application that allows support and programs to be accessed more easily, and make it easy to use with a friendly user interface. I believe that having this kind of system will increase their ability to access various types of opportunities and support and could therefore reduce the problems that have been mentioned (Justice Committee, 2019).

This makes it more important to develop an application that allows support and programs to be accessed more easily, and make it easy to use with a friendly user interface. I believe that having this kind of system will increase their ability to access various types of opportunities and support and could therefore reduce the problems that have been mentioned (Justice Committee, 2019).

1.3 Aim

The aim of this project was to develop a web application that allows users to have accessibility to different programmes, support and experts in different fields at their fingertips through a user interface that easy to use for offenders or people that have been out of society for a long time and are not as well versed in using technology. I aim to do this be by filtering the vast amount of information that is already out there and tailoring it too the users of the application based on what they are looking for from a particular sector such as careers, legal advice, mental health etc. Social workers will be able to post valuable information based on their expertise and post programmes for different sectors for the users to see and have access to. Users can also book appointments with the relevant staff such as counsellors, social workers etc and are given the exact directions to the location of the appointment, as well as the time and distance it takes to get there. The combination of these features aimed to provide users with an engaging, easy to use, effective social welfare application

1.4 Objectives

- Learn about the social welfare system and the sub- sectors within it and choose what you want to include the application to specialise in (if you don't want to include all the sub sectors)
- Program the application to take the necessary input and information from the user and the appropriate user interface to output what they are looking for.
- Implement a booking system so they can book appointments with the social workers or counsellors that are already on the application.
- Implement this social welfare application as a web application.
- Learn about the challenges users normally go through when using technology and aim to remove them to make the application easier to use.
- To implement an algorithm that can allow users to get the directions to their appointments as well as distance/time to get there.
- To perform user testing to conclude the effectiveness of the application
- To utilise the results from user testing to evaluate the application's effectiveness against existing applications

1.5 Research Questions

- How does having digital skills help offenders or people that have been out of society for a long time?
- How does accessing support through technology impact the lives of former offenders or people that have been out of society for a long time, and otherwise wouldn't have access this kind of support?
- What is the impact of digital technology in improving people's lives?

1.6 Report Structure

Chapter 2- explores the research conducted within social welfare and how technology and different programmes was used to improve the lives of different types of people such as therapy patients, offenders, medical patients etc. I evaluated similar applications to reflect on how I will approach the development of my web application

Chapter 3 details the functional and non-functional requirements that the system needs to fulfil, as well as the use cases, risks and design.

Chapter 4 demonstrated the implementation and the results of the implementation, which includes the methodologies, architectures and design patterns used.

Chapter 5 discussed the different types of testing involved during the development and testing phase.

Chapter 6 involves a reflection over the results of the application, including the challenges faced and what I could have done differently. It also gives ideas for future work to expand on this project.

Chapter 2: **Research**

2.1 Literature review

There are plenty of very informative and helpful studies that have been done revolving around technology and programmes supporting people in different avenues. These studies show the role that technology has played in improving people's lives and what is currently in development.

2.1.1 Effect of digital skills on improving people's lives

We now live in an era where digital skills are in demand and highly rewarded. We also know that people such as offenders generally don't have access to such devices, so upon release they will naturally steer away from learning digital skills and getting into this industry. This can mean they would be missing out on plenty of opportunities that they may not be aware of, and is causing a digital skills gap, which not having can heavily impact one's life by decreasing the ability to find a job and then increasing the chance of them re offending. (Innovative Criminal Justice Solutions, 2022)

There are plenty of programs that aim to tackle this problem by providing the right digital skills and opportunities to offenders, one of them being Code400. Founded by Micheal Taylor, the aim of this organisation is to prepare offenders with the necessary digital skills training to find work in the tech sector. Once training has been completed they will have access to further employment, or further training. 18 have graduated so far from this program and none have re offended. Another program includes the digital poverty alliance partnering with Intel to launch the Tech4PrisonLeavers programme to help released offenders gain digital skills and access through Trailblazers mentoring. Intel granted 150000 for this pilot and Trailblazers has said that "re offending could fall to 8% within one year of release and 10% within 2 years". (Sophia, 2022)

This goes to show the effect of having access to these opportunities, and that opportunities like this exist. The goal of this project is to develop an application where opportunities like this will be accessible to the users at their fingertips where they may not have been able to access them previously.

2.1.2 Role of digital applications for support in social welfare and improving people's lives

There are plenty of applications that are designed to support users in different aspects of social welfare. One of the main applications being the NHS app. The NHS App allows patients that are within the NHS to access a 'range of medical services' that they may be looking for. They are able to get advice regarding their 'conditions and treatments, book appointments with their GP surgery, order prescriptions, view their records, register as an organ donor', and much more. (Google Play, 2022).

This has had a big impact on people's health and has proved to be very popular with over 12million new users since the NHS Covid Pass was added on 17th May, and 1.5million people using the application to' manage their organ donation decision'. 150,000 of these new registrations have now been able to take place in the span of 4 months. This shows the applications ability to increase the speed and smoothness of the organ donation process which saves lives. There has also been '3.2million repeat prescriptions' ordered and over '268,000 appointments' booked through the app. (NHS, 2021). This shows the impact of using technology to increase the productivity in the health and medical industry and improves people's lives.

Another study has been done analysing the willingness of Finish prisoner's willingness to use digital health care and social welfare services. From the results "62.3% of the participants totally or partially agreed that they are likely to deal with social and health services in electronic form in the future". (Teemu, 2021). Although there was some opposition, shown such as only "30% agreeing to prefer remote meetings with a professional", it gives us the general idea that the use of digital technology for social welfare is something that prisoners and offenders would be open to. Another study was also done on the usage of job search applications and the results had shown that "Average DAUs for the month of June have grown 57% year-over-year" (Adam, 2022) which further re iterates the growth in the usage of mobile apps for people who searching for jobs.

Another application that has been designed by John Burton is the Inside Connections Application. This application is designed for the family of offenders which gives 'guidance on accommodation, drug addiction and employment'. Paperwork such as work history and qualifications can be uploaded. There is also a CV creator which uses 'the work experience offenders gained' while at prison that is stored in the app to create a CV which they can use to find employment. (Russell Webster, 2017)

This shows the direction that we are heading in as more and more people head towards a digital infrastructure and means to access a range of services, and the goal of this project is to implement a similar strategy to develop a web application so that, finding access to all support, opportunities and programmes is easier.

2.1.3 Counselling post sentence for offenders

There are plenty of professionals who do counselling for different kinds of people for different aspects of their life. There are also studies which have been done to show the impact of this type of counselling. One of these studies was to determine if a 12-week career counselling intervention would increase the career maturity of juvenile offenders. The group used as part of this study had their career maturity percentile ranking 'increased from the 27th percentile to the 62nd percentile'. (Katie, 2010) This will allow them to make good educational and career decisions to enhance their lives, and this study shows the importance of making this kind of support easy to access, which Is what I hope to achieve through the development of this application where counselling is easy to find for offenders or people that have been out of society for a long time, as well as social workers or people involved in counselling

2.1.4 Conclusion

From this literature review, I have come to the conclusion that there are many programmes in place to support offenders, but there is not one central point of access where they can have access to these kind of programs. I have also come to the conclusion that people are resorting to digital means to improve their lives and this will only increase as the technology becomes more and more sophisticated and common to use. I think it is inconvenient to constantly be on the search for these programs and support all over the Internet where there is an overwhelming amount of information, when you can find them at your fingertips through one application.

The aspects I have included in this literature review include the importance and effect of gaining digital skills, using digital applications to improve different aspects of your life, general support programs for offenders in different categories, and the counselling of offender's post sentence. I have broken down my research into these aspects as I plan to combine these elements in the development of my project. I will be developing a digital application which will give users such as offenders access to different kind of programmes to improve social welfare, which will improve their social, digital, and other skills, give them access to counsellors and support.

2.2 Existing applications

There are numerous applications at the moment that are designed to improve the lives of users that make different types of support easier to find. These range from job finder applications to medical applications where you can access a range of services from your local GP. In this section I will be explaining each application, talking about their main features, pros and cons, and how I will differentiate and expand on what was has already been done for my project.

2.2.1 NHS Application

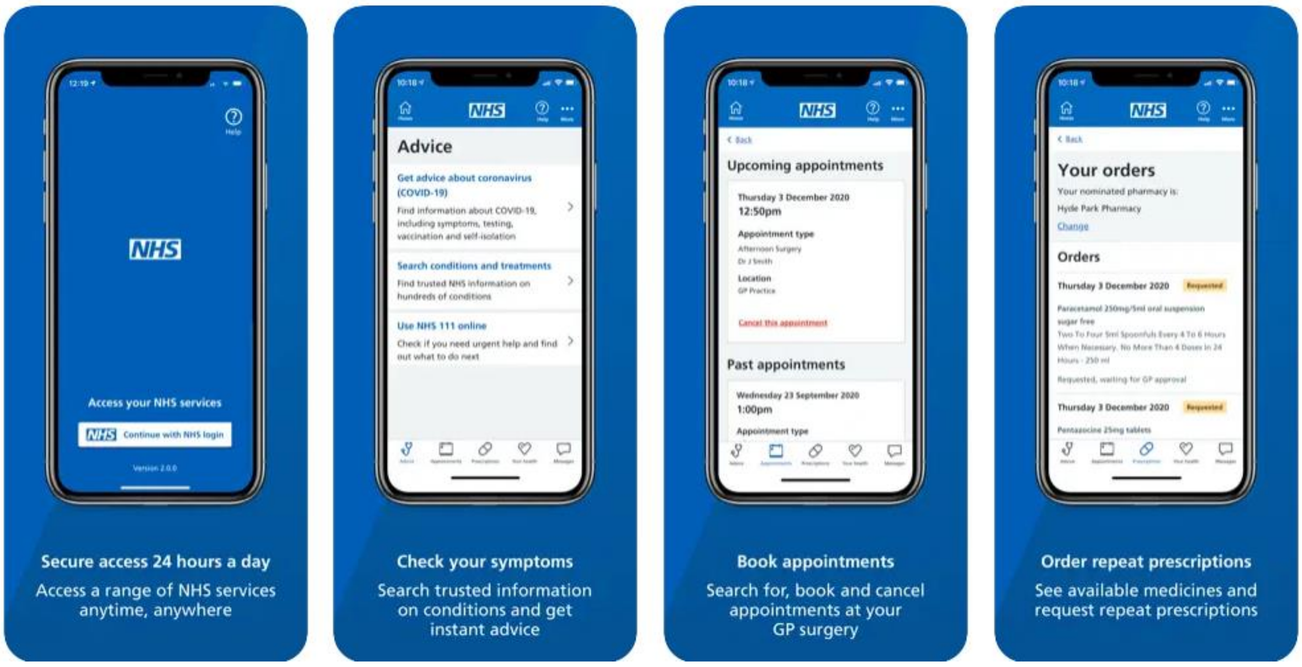

Secure access 24 hours a day
Access a range of NHS services anytime, anywhere
Check your symptoms
Search trusted information on conditions and get instant advice
Book appointments
Search for, book and cancel appointments at your GP surgery
Order repeat prescriptions
See available medicines and request repeat prescriptions

Figure 1: NHS Application

The NHS app is an Android/IOS app that allows users to access a wide range of services in a safe and secure way. These services range from searching for medical advice/treatment to booking appointments with doctors, to ordering prescriptions. (app store 2023)

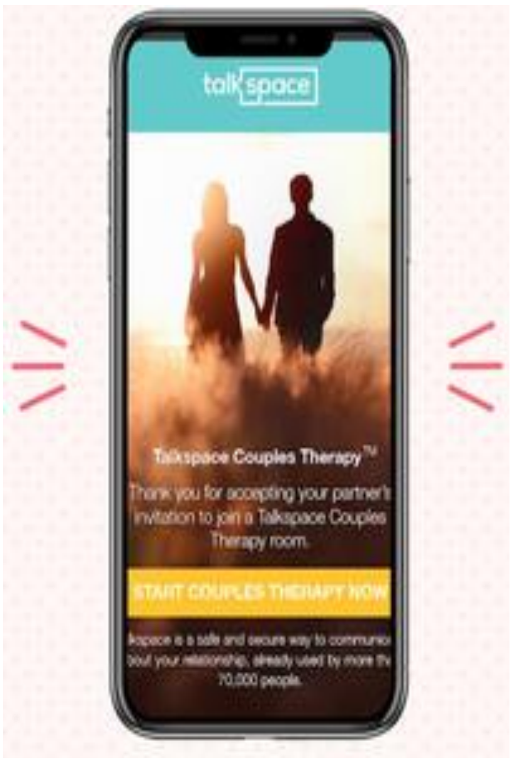
Although the app has excelled in developing a clear interface to present information well, this app is restricted to those who already have access to a GP and doctors from that particular GP. The goal my project is to make expert support accessible to anyone therefore increasing the range of my target users. I also plan to add to this in my project by including the directions to their appointment should it be in person and also increasing the range of sectors that users can get help in.

2.2.2 Talkspace application

Talkspace is an IOS and web application platform that aims to help improve the mental health of users by giving them a brief assessment, then matching them with a therapist based on their answers. (Talkspace 2023

Although there is a clear user interface, the application mainly focuses on mental health issues and the appointments are conducted online. In order to expand the type of support available through my project, my aim is to allow the options for the user to have appointments either online or in person, as well as being able to help in other industries such as career, legal help, etc.

There are also numerous steps when it comes to getting an appointment, which involves answering over 6 questions and revealing their medical history. This may be tedious for the user, particularly if they are not as well versed in technology, and the goal of my project is to remove this process and making the appointment booking process as convenient as possible.

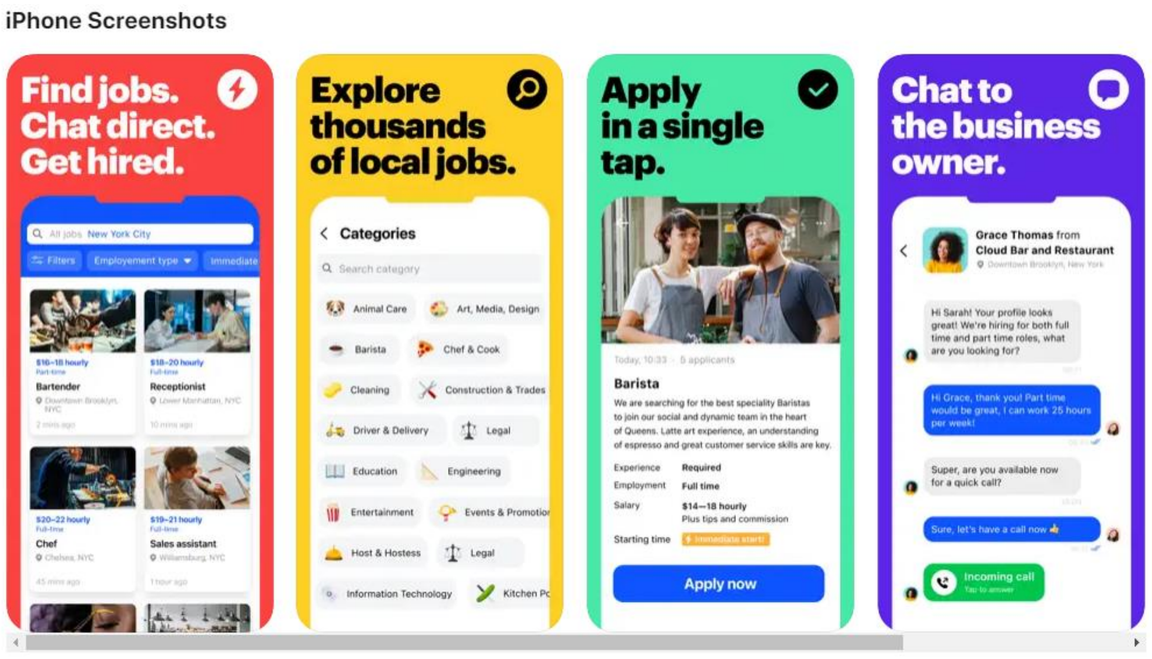


Thank you for accepting your partner's invitation to join a Talkspace Couples Therapy room.

…lkspace is a safe and secure way to communic… …bout your relationship, already used by more th… 70,000 people.

Figure 2: Talkspace Application

2.2.3 JOBTODAY



Barista

We are searching for the best speciality Baristas to join our social and dynamic team in the heart of Queens. Latte art experience, an understanding of espresso and great customer service skills are key.

Experience	Required
Employment	Full time
Salary	$14–18 hourly Plus tips and commission
Starting time	Immediate start!

Grace Thomas from Cloud Bar and Restaurant
Downtown Brooklyn, New York

Hi Sarah! Your profile looks great! We're hiring for both full time and part time roles, what are you looking for?

Hi Grace, thank you! Part time would be great, I can work 25 hours per week!

Super, are you available now for a quick call?

Sure, let's have a call now 👍

Figure 3: JOBTODAY application

Job Today is an application that connects employers and job seekers. Employers can post jobs and job seekers can search for jobs by filtering them based on what kind of job they are looking for. They have the ability to communicate with employers to arrange interviews and the app for them to “Get a job in 24 hours!”.

The app does not have a lengthy process in terms of gathering data from the user. The user can “create a short profile in seconds” and can apply with a single click”. (App Store JOBTODAY, 2023)

This does a great job in connecting employers with job seekers in allowing them to find what they are looking for within their location, and gives them the ability to schedule interviews in a user friendly many. My project aims to expand on this concept when it comes to accessing different kinds of programs for mental health and social welfare, and also for booking appointments with social workers and counsellors etc.

2.2.4 Betterhelp

BetterHelp is another therapist based application where users are given a questionnaire and are matched based on the answers they have given. One good feature about this application is the ability for the users to remain anonymous, and that they also have the option of switching counsellors if they are not happy with the person they have been matched with.

This is a concept that my project idea expands on, and that users booking an appointment will have the option to choose their own staff members to have an appointment with based on factors such as their needs, locations etc.

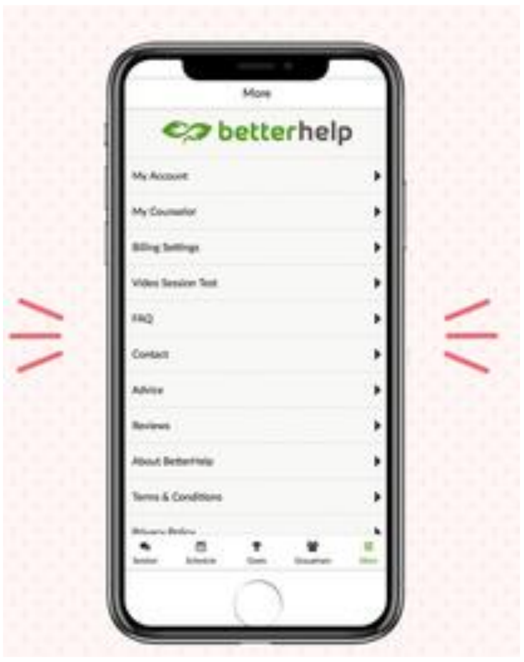


Figure 4:Betterhelp application

2.3 Research conclusion

The research and the studies that have been done have been very insightful in giving me an idea on what is out there to support offenders, and the impact that it has had on them. I have looked into what aspects are there that offenders can receive support in, what the impact is and how I hope to include this within my application to make it more accessible and easy to find.

From the literature review, I have come to the conclusion that there are many programmes in place to support offenders and people who have been out of society for a long time, but there is not one central point of access where they can have access to these kind of programmes. I have also come to the conclusion that people are resorting to digital means to improve their lives and this will only increase as the technology becomes more and more sophisticated and common to use. I think it is inconvenient to constantly be on the search for these programs and support all over the Internet where there is an overwhelming amount of information, when you can find them at your fingertips through one application.

From looking at existing applications, I have come to the conclusion that there are many existing applications that aim to allow user to find the relevant support and guidance that they are looking for and use various approaches to do so. Some used things like chat boxes, while some leveraged appointment booking systems. These examples gave me an idea of what to include and exclude in my project, and what elements to combine in order to create a well-rounded user friendly application.

Chapter 3: **Analysis and design**

The requirements have been split into functional and non-functional requirements in order to gauge how the system must work, and also how the system to perform. The requirements are then used in the design and implementation.

3.1 Requirements Analysis

3.1.1 Functional Requirements

1. The system should allow social workers and other relevant staff members to post relevant information for users to see, as well as a commenting functionality where other staff members can comment on any posts to add further information.
2. System should allow users to view posts from social workers as well as the social worker's profile on the post so they can book appointments with them.
3. Social workers should be able to add programmes from different categories given such as workshops, mental health related etc and users should be able to view them using the dashboard or using a search functionality if they are looking for something more specific.
4. Staff members should be able to create an account/profile for users to see which includes a description as to what they do, what they specialise in, and any useful links to find out further information about them. This should be saved onto the database for them to log in whenever they want to use the application.
5. User's should be able to book appointments with staff members, and the staff members can either accept, decline or re-arrange the appointment. The appointment information should be stored onto the database securely and upon completion the staff member should be able to delete the appointment which will result in the removal of the appointment from the database.
6. Users booking an appointment should have the option of either having an in person or an online appointment. If they choose for an online appointment the staff member will be taken to the zoom site from the page to generate a meeting link, to give to the appointee. If they choose an in-person appointment, the appointee should be given directions to the locations as well as the distance and time it takes to get there.

3.1.2 Non-functional Requirements

1. The system should provide a simple, easy to use interface. There should be separate sections for different functionalities of the application. The design should be able to facilitate usage that requires minimal explanation and experience in using technology.
2. The user input should validate all user input where required. Where users must input a valid value, the system should be able to ensure that users have inputted a valid value before moving on, and where invalid the system should display an error message accordingly.
3. The system should respond to user input within a reasonable amount of time and with as most accuracy as possible. The applies to functionalities such as appointment information, or information regarding directions and other results from API's used.
4. The system should be secure in terms of storing the user's information and should be able to access and edit their own data if need be.

5. The system should be reliable, available 99% of the time and easily maintainable.

3.2 Design

3.2.1 Use Cases

There are several use cases in this project, which were dependant on the actors. Some were given extra use cases to maximise the benefit of the application. The aim was to minimise the amount of input/work needed to be done by general users to allow a user friendly experience. Details regarding some of the use cases can be viewed in Appendix A.

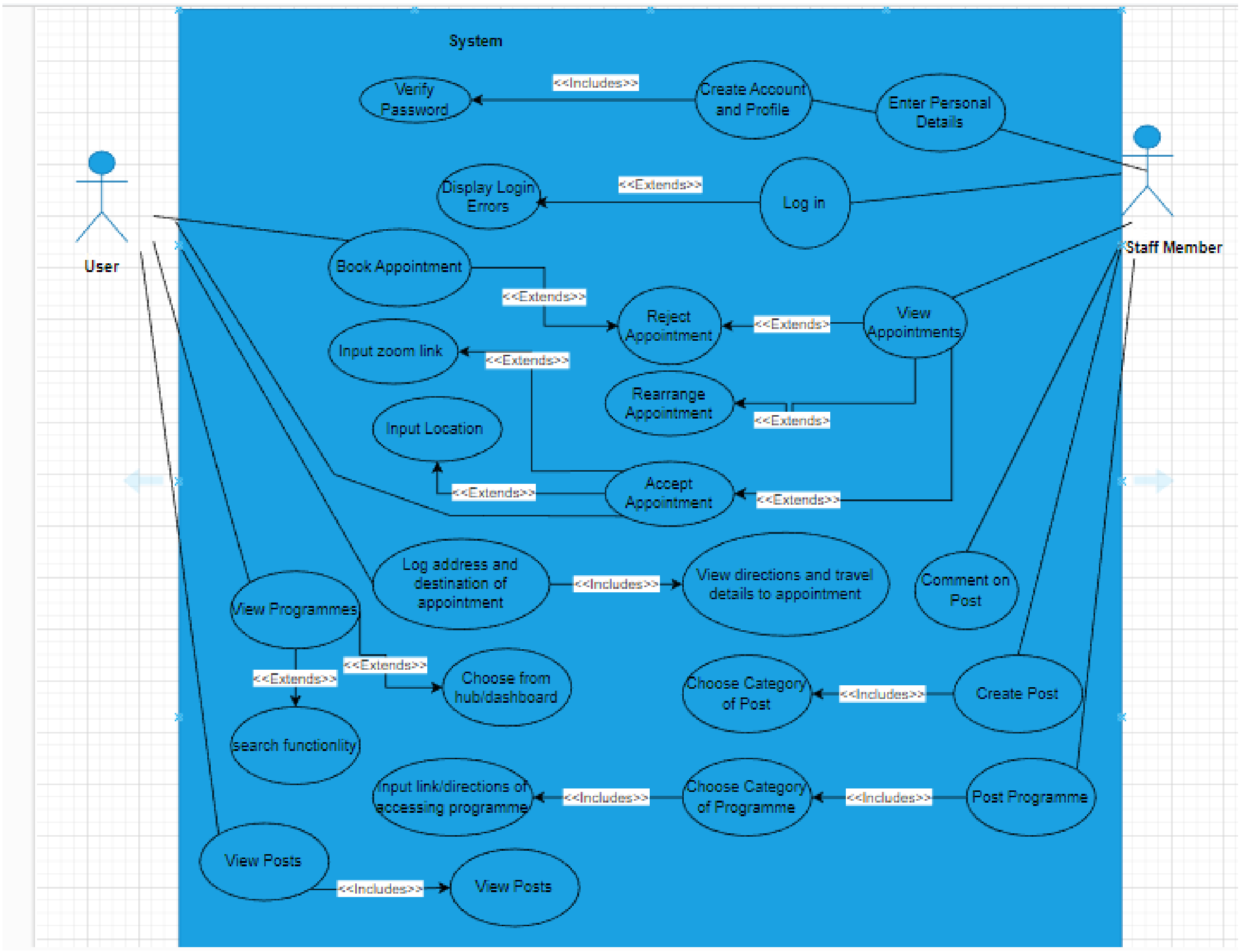


Figure 5: Use Case Diagram for Social Welfare Helper Web App

3.3 SDLC

3.3.1 Development Environment

Django is a framework that is based on the python programming language that facilitates the development of web apps. Django creates an emphasis on security, and scalability, and versatility so it can be used to create sites with different types of functionalities. (Django, 2023). Django gives you the ability to work with database, as well as HTML templates to develop a user friendly front end. More information on how this was implanted for this project will be discussed in chapter 4 the implementation stage.

3.3.2 System Architecture

As I developed this project as a web application, I decided to follow a commonly used architecture when it came to developing web apps using Django. This was the Model-View-Template design pattern. (MVT). This will split the web app into 3 separate entities: the model, the view and the template.

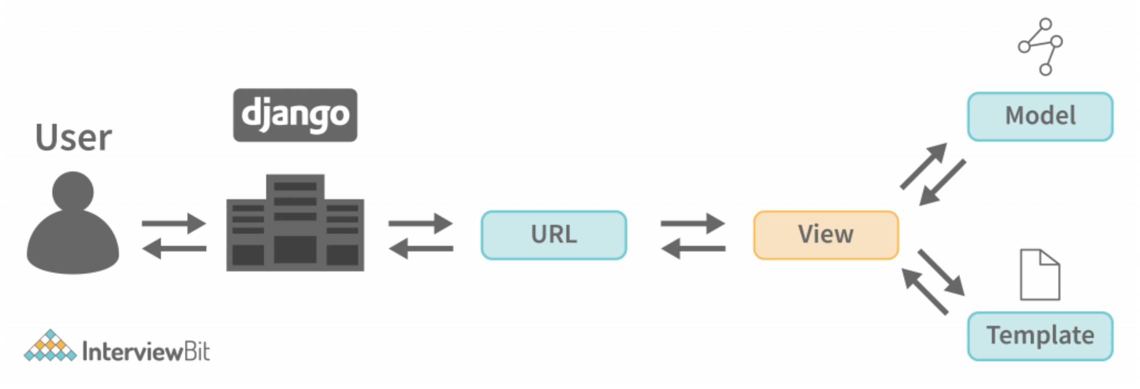


Figure 6: Diagram demonstrating MVT architecture

The model is normally what is used as an object that defines a certain piece of data in the application. The model normally has attributes that will be added/ updated/ read/ deleted during the usage of the web app and is presented as a table in a database. The model takes data from where it is stored and will send it to the view.

The view is responsible for receiving HTTP requests these requests can be to load a particular page, submit a form etc. It will retrieve and process the necessary data that is within the request to fulfil the request. It then renders the results on the user interface. In Django this is done using view functions and the results are posted on the interface using templates.

The template is a HTML template text file that defines the layout of a particular page. It uses placeholder texts that represent actual attributes that were created from the model. Models are used to populate the HTML page with the relevant information as they are able to be bound together. It accepts data from the view to be rendered. (InterviewBit, 2022)

The process of the user interacting with this architecture involves initially using a URL that is passed onto the architecture. A mapper then redirects the request to the appropriate view based on the request URL. Once the view is found it will interact with the model to get the necessary data from the model for processing. Once the data is retrieved and processed it renders the given template populated with the data it gets from the model. (Shubham, 2023)

There are many benefits to using this kind of architecture, including rapid development as it allows different components to be worked on at the same time because as they are very separated. This aspect of separation also makes it easier to modify without affecting other parts of the application, which makes it easier to maintain and increases the quality of the code.

3.3.3 View

The screenshots below show some wireframes that I have designed to give a visual representation of what the application should look like. I will be using these to discuss the flow of the application, and describing the factors that went into designing the web application in this way such as usability, visibility etc.

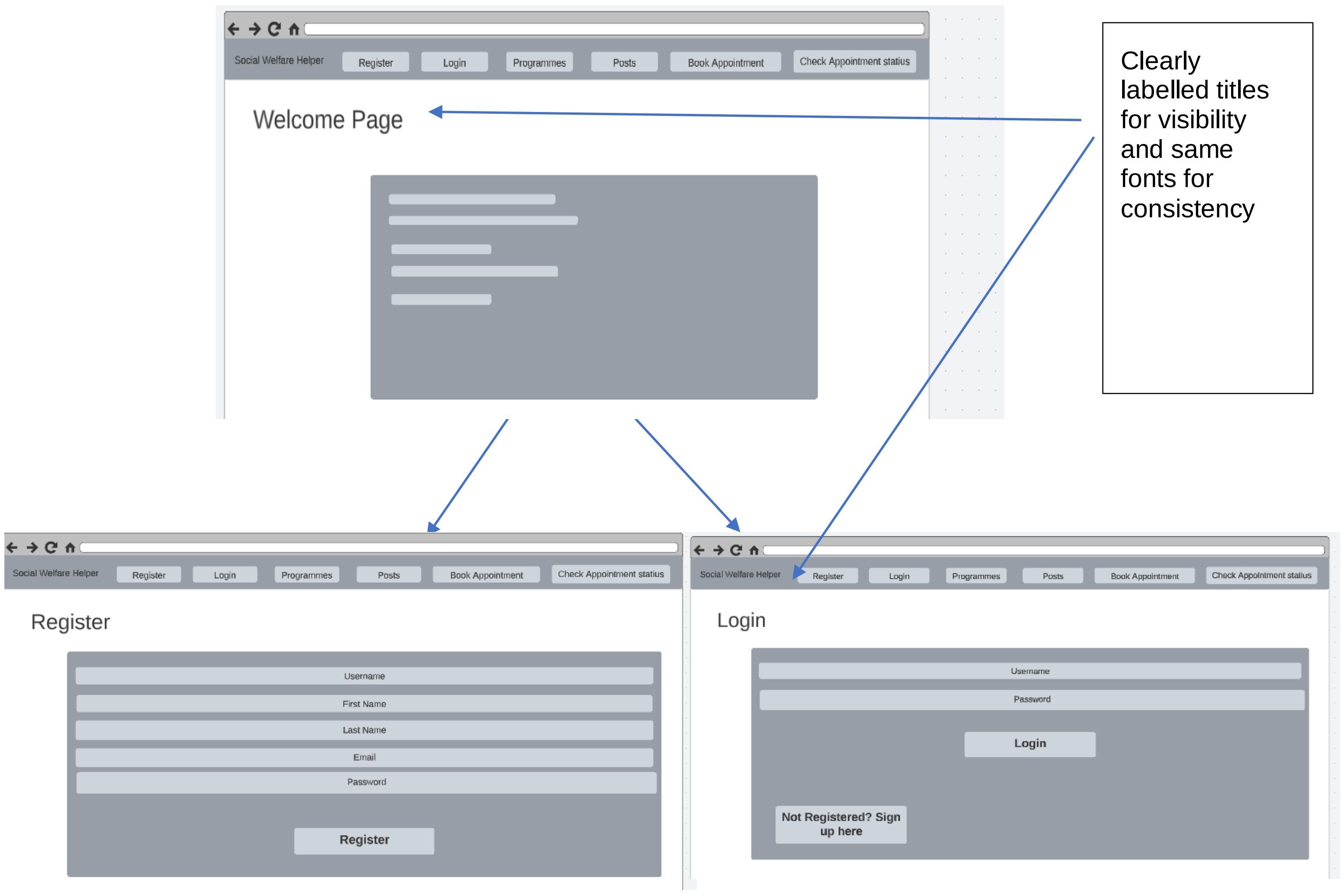
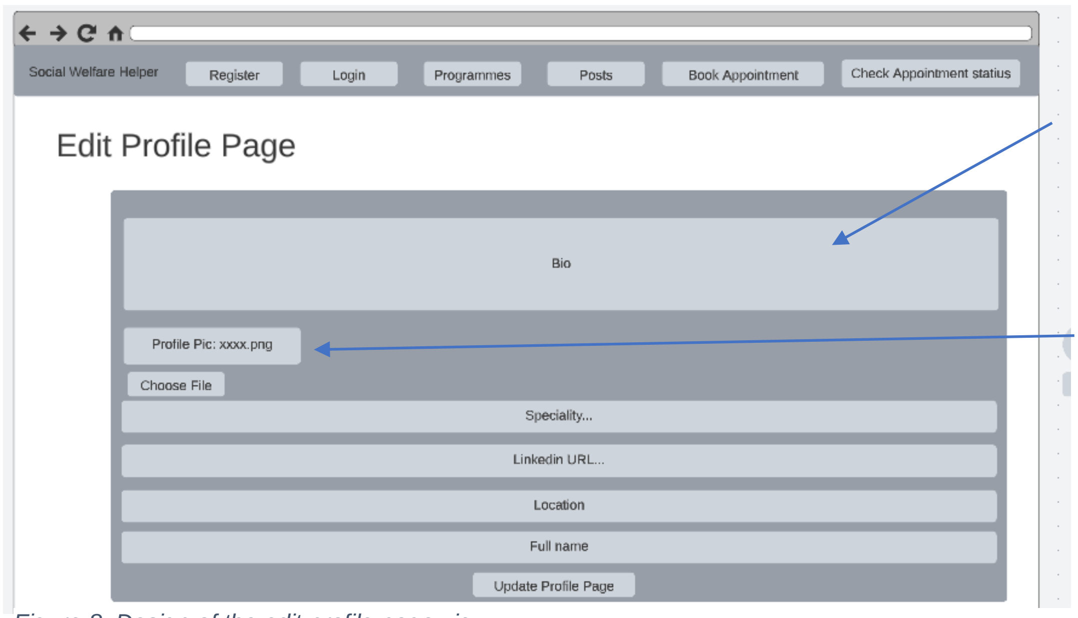


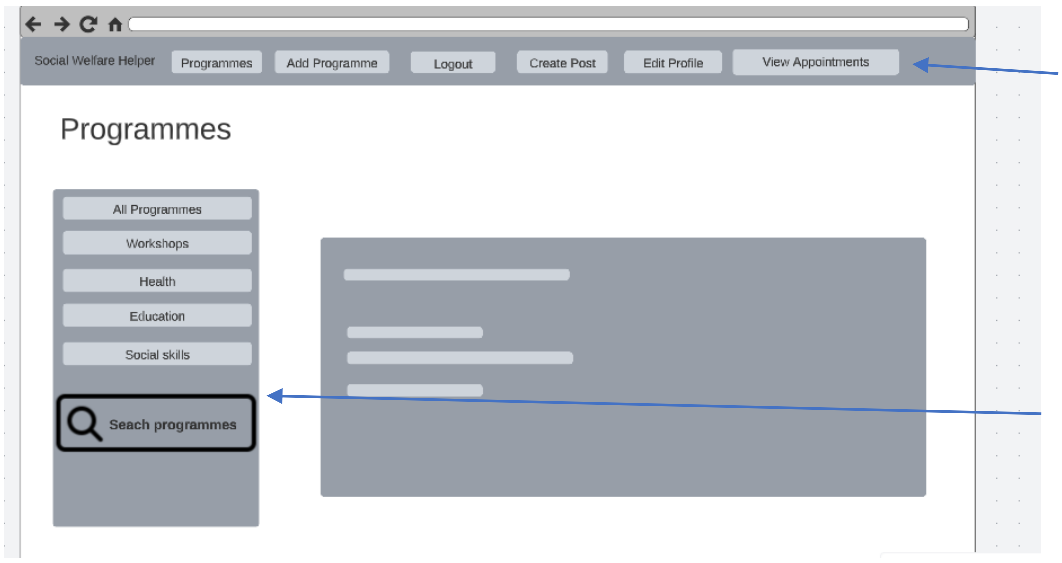
Figure 7: Design of the welcome, login, and register page. And navigation bar for user to explore their options



Clear labelling of fields for more visibility

Profile Image

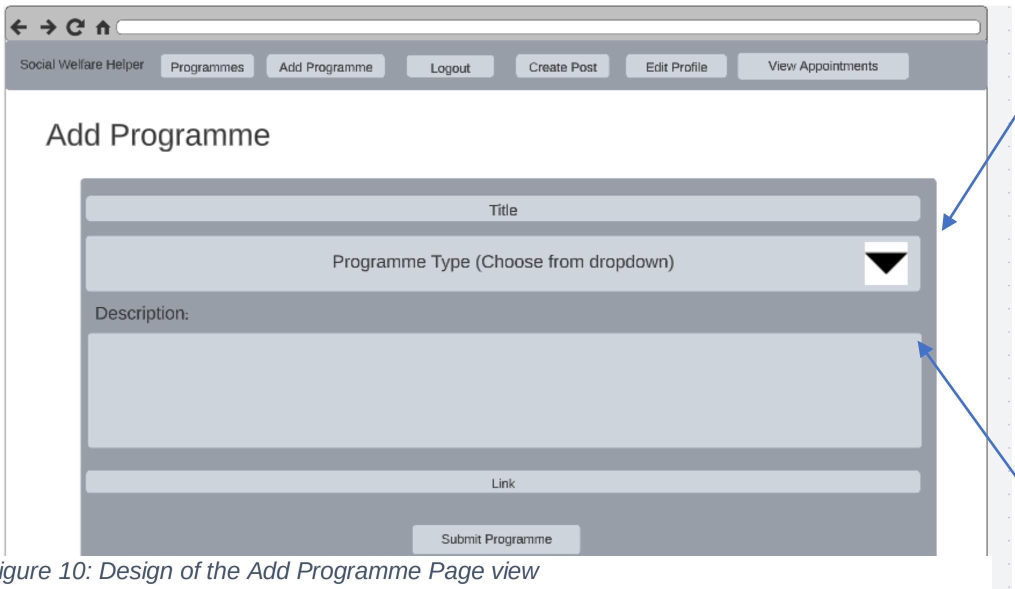
Figure 8: Design of the edit profile page view



Dashboard consistent throughout usage of web app to increase ease of usage and allows the user to navigate to different areas easily.

Each programme type explicitly labelled as well the search bar icon so user its aware of its purpose.

Figure 9: Design of the Programmes page view



Each field labelled for visibility purposes and differ in size. This is so the user understands the length of input that is required.

Dropdown Menu icon used so user is aware of its functionality and can easily choose from the options given. This is an example of natural mapping where the icons resemble what we want out of the program, which in this case is a dropdown of specific options for the user to choose

Figure 10: Design of the Add Programme Page View

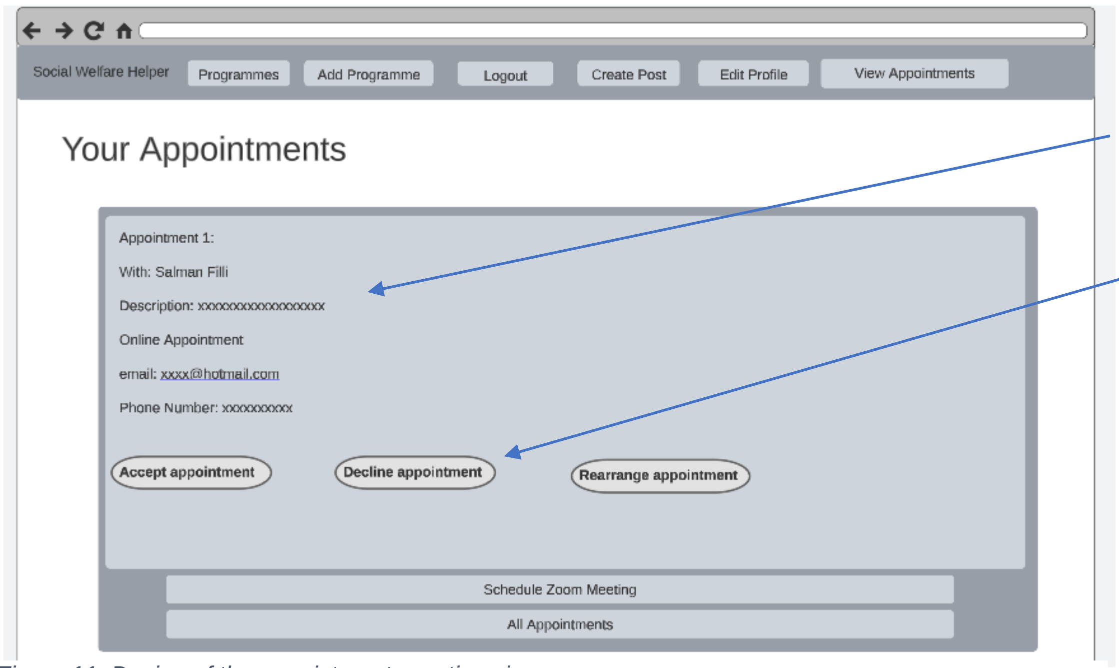


Figure 11: Design of the appointments section view

Each detail of appointment labelled separately for visibility purposes and buttons are made visual so the staff member is aware of each functionality.

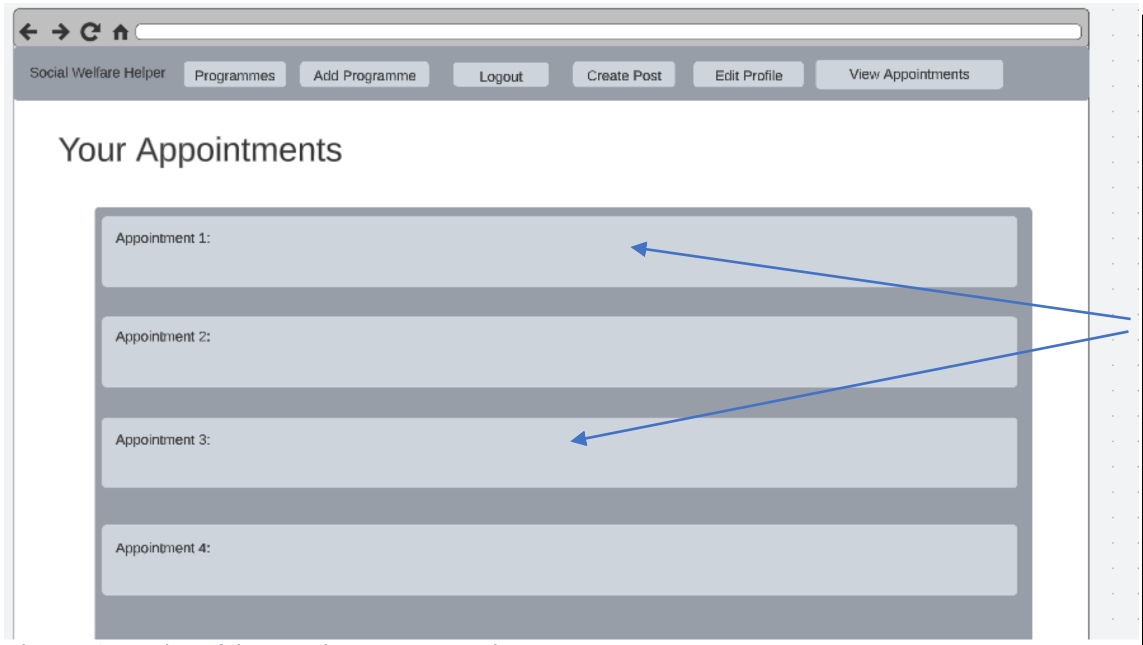


Figure 12: Design of the appointments page view

Appointments are labelled in separate containers so the staff member can differentiate one appointment from the other and manage them accordingly. The staff would also have the option of deleting an appointment if it has been cancelled/completed.

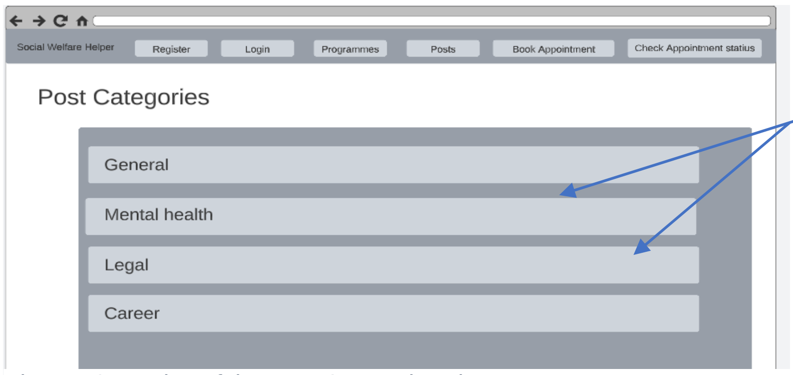
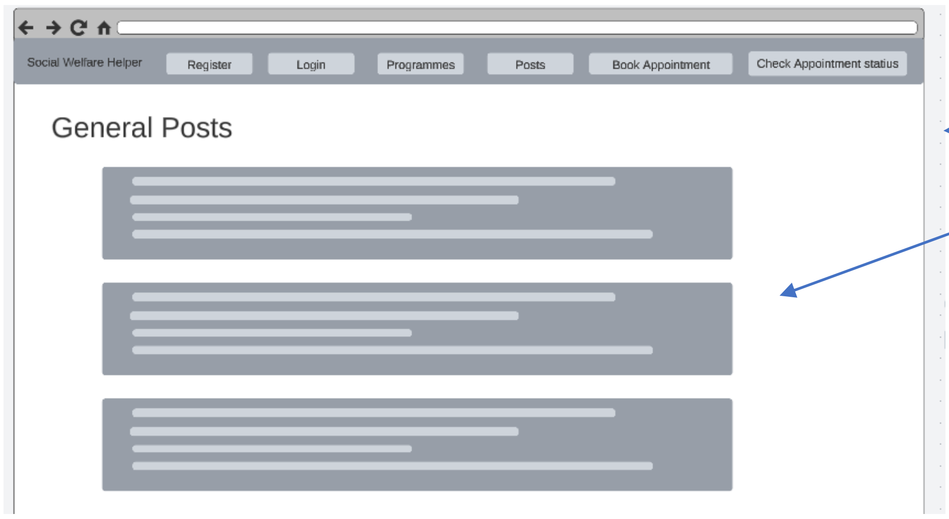


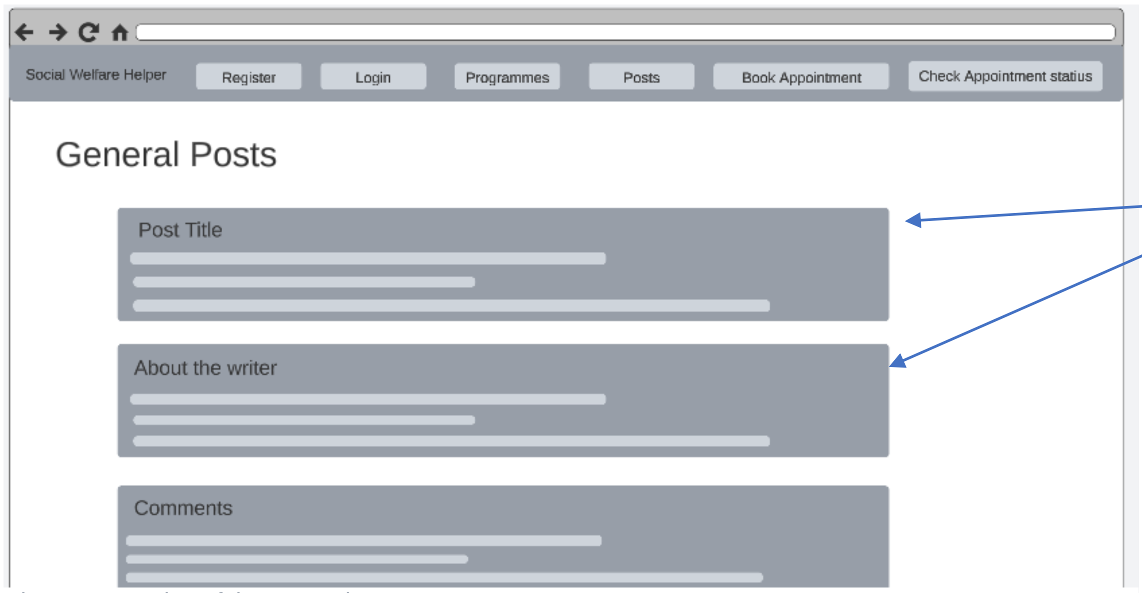
Figure 13: Design of the Post Categories view

Each post category explicitly labelled and categorised so the user can access posts based on the sector of interest. Also increases visibility as all posts are organised properly and separated.



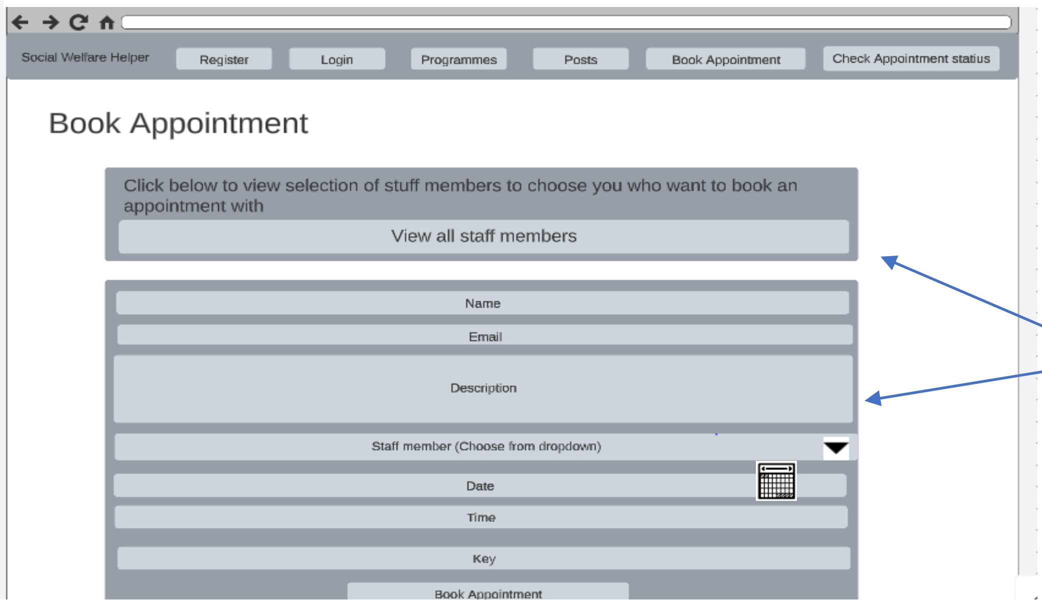
Posts are separated using containers for easier differentiation between them

Figure 14: Design of the posts view



The post content, the information about the writer of the post and comment section are all created in separate containers so user can differentiate between them. This increases visibility and consistency

Figure 15: Design of the posts view

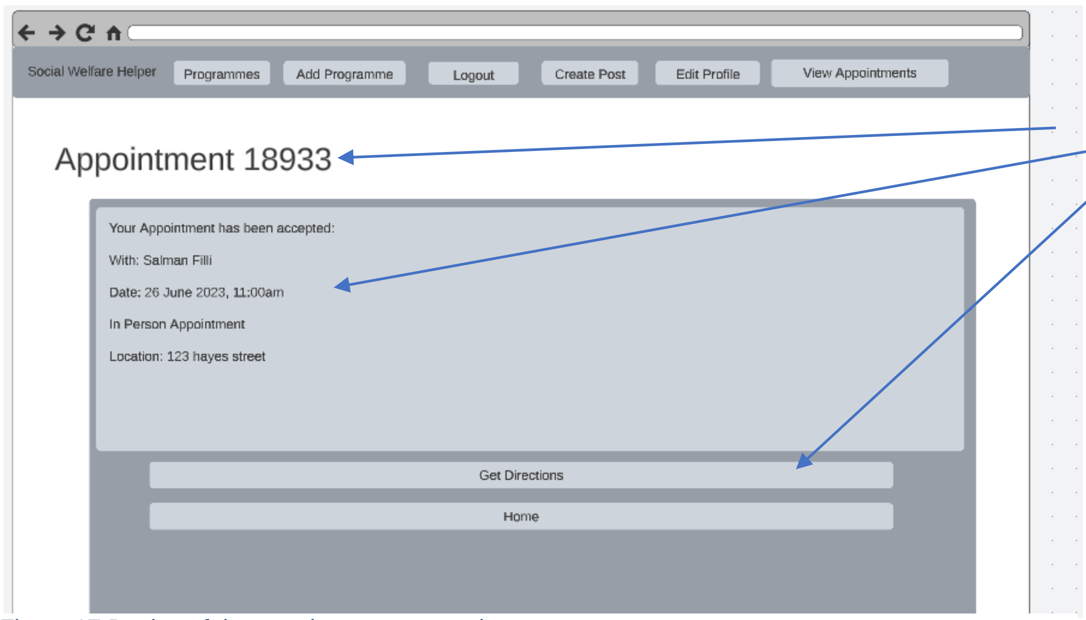


Appointment fields labelled clearly to increase ease of use.

The date has an appropriate, commonly recognised calendar icon so the user knows it represents and can choose a date accordingly (natural mapping)

A different container is used to output the view staff member section to increase its visibility

Figure 16:Designing of the Booking Appointments Page



Appointment 18933

Your Appointment has been accepted:

With: Salman Filli

Date: 26 June 2023, 11:00am

In Person Appointment

Location: 123 hayes street

Figure 17:Design of the appointments page view

Appointment with the key explicitly labelled so user knows these details correspond to their appointment.

Appointment details labelled based on the fields explicitly so the user can take note of the details.

Separately labelled buttons allow the user to find them more easily and navigate accordingly

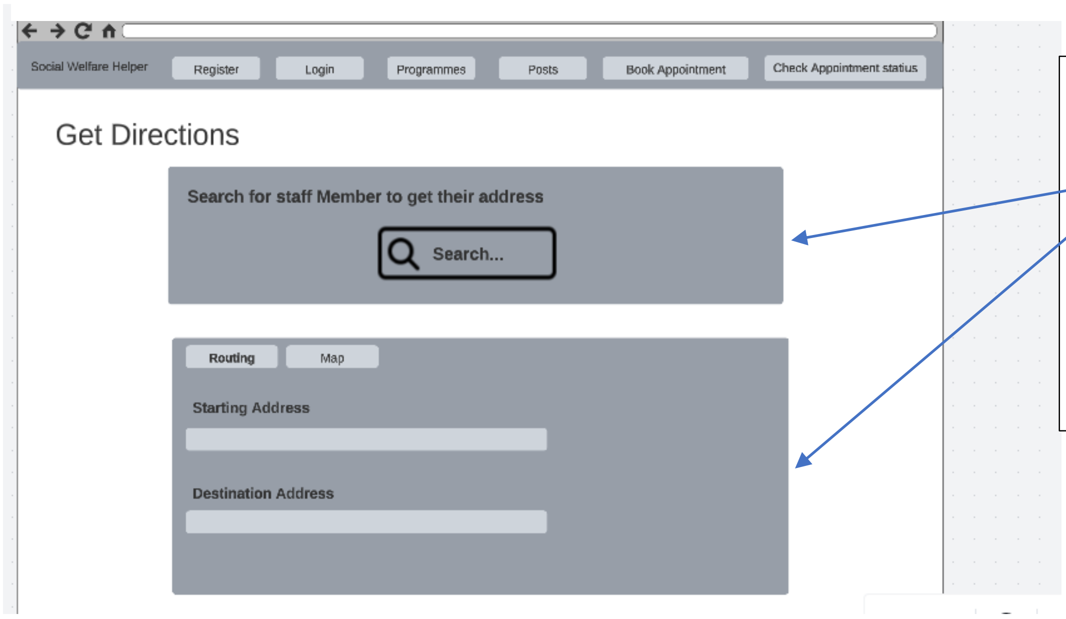


Figure 18: Design of the directions page

- Search bar used for to search for the staff members address by name.
- Input fields explicitly labelled so user knows what to input in order to get the correct directions to their appointment

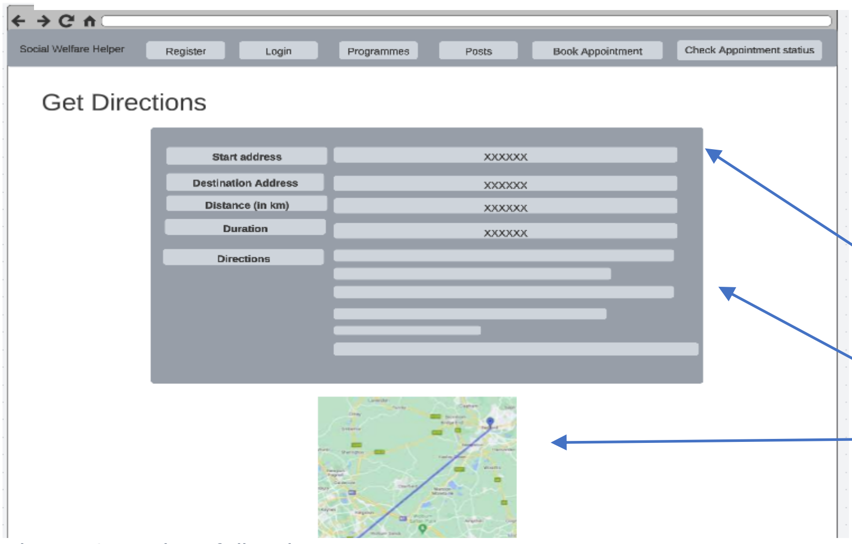


Figure 19: Design of directions page

- Fields relating to directions labelled for clear visibility of travel info.
- The directions are output as a step by step guide with each line representing one step.
- This is further enhanced through using the map as the user has a more visual representation of the directions.

3.3.4 Database Models

Django gives you the ability to create tables and fields for different entities using SQL. Below is a table of the main entities used within the application and the relationships between them. The model diagrams were built using a class diagram visualisation.

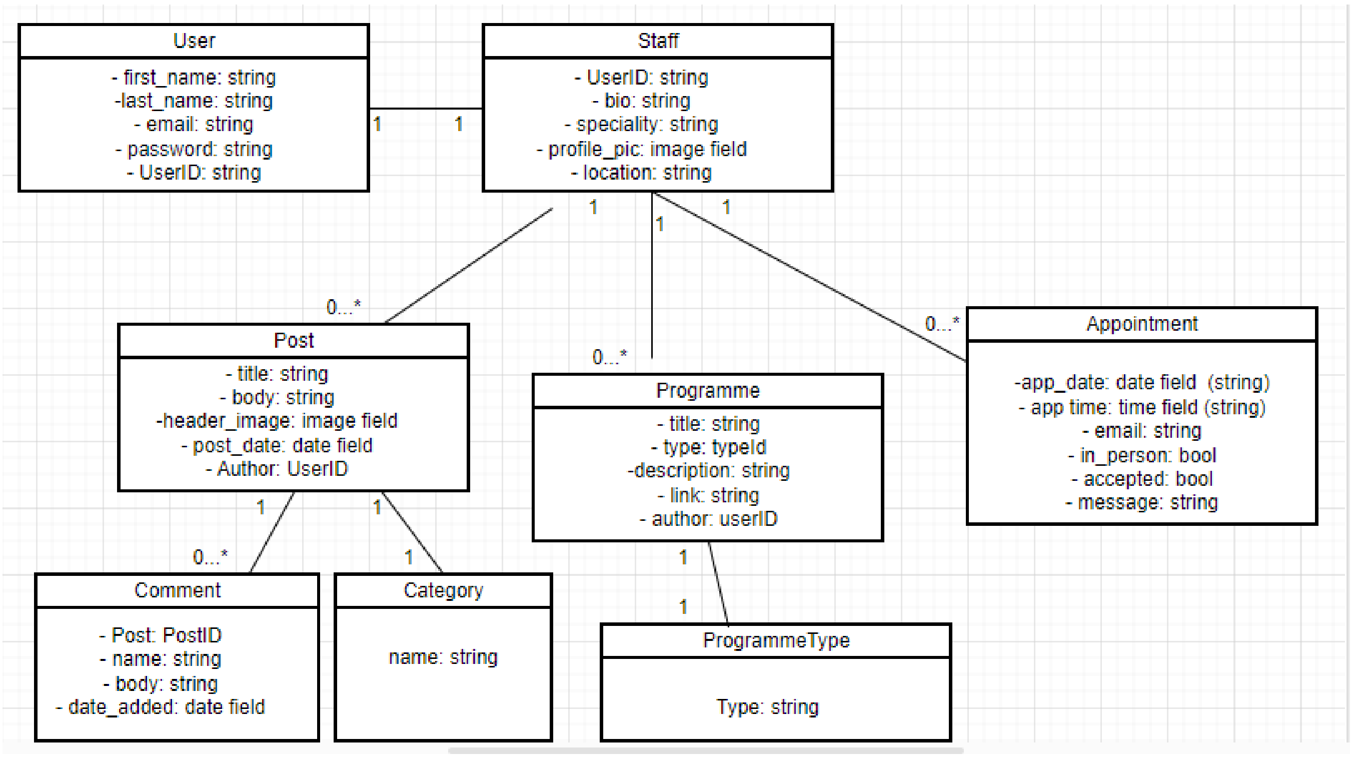


Figure 20: Class Diagrams for different database models

Django comes with an object relational mapper so once you write out the entities and their respect fields using python, it creates databases to store all the relevant fields, (Wikipedia Object-relational mapping, 2023). Django also has a built in user model (shown as User on the table) which already contains default values such as first_name, last_name etc. Because of this I was able to create a model for staff members to inherit from the User model. This way the Staff model had the same values that the default User model had, and I could add extra fields that would be specific to the Staff Member. This saves time and increases maintainability.

Chapter 4: **Implementation and Results**

4.1 Platforms and Technologies used for implementation

4.1.1 Django

Social Welfare helper is a Django web application. I decided to go with a web app due to the flexibility that was provided as they can be accessed by any browser on mobile or desktop. However, a mobile app is mainly restricted to a mobile phone. Using a web app would allow the platform to be accessed by more users through different device types. Also, web apps don't require installations which save memory and data. This reduces the barrier to entry so will make users more inclined to using it. Furthermore, web apps don't have to be updated regularly so it reduces the mainainance required. (Daragh Tuama, 2023).

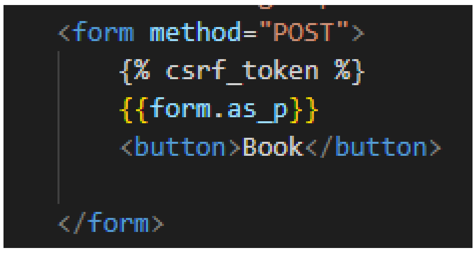
Django is a framework based off python that facilitates the development of web apps. Django creates an emphasis on security, and scalability, and versatility so it can be used to create sites with different types of functionalities with high level maintenance (Django, 2023).

Security in Django

Django excels in web security through numerous security measures.

Django web apps provide security is through cross site scripting protection. XSS attacks are when users inject code into the browsers of other users so they can execute certain scripts on the user's website for malicious intent. Django provides protection against this by escaping certain types of HTML code that could by harmful to the website. This is done by converting harmful HTML characters to unharmful ones. (Django Security 2023)

Django also provides security is by preventing XXS attacks, which is where an attacker will complete malicious actions using the credentials of another user without their knowledge. This is protected against by using A CSRF middleware and creating a CSRF token. This is a secret value that is sent to the client side from the user, and is checked everytime the client makes a request. This prevents the attack because the attacker would have to know this specific value for the user if they want to replay a form. (Brian Myers 2021)



```
<form method="POST">
    {% csrf_token %}
    {{form.as_p}}
    <button>Book</button>

</form>
```

Figure 21: Usage of csrf tokens in django

Django also provides security for the databases, which stores all the information in the program. Attackers can attack a database through a SQL injection, where they execute additional commands on the database to change the way the app functions. This can lead to leakages, and even completing wiping of your data. Django Prevents this by allowing developers to implement an authentication function (StackHawk 2021).

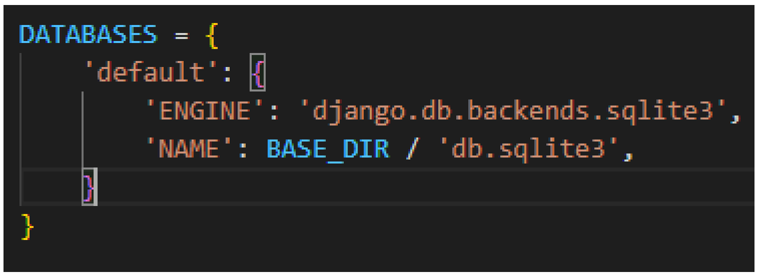
Overall, security was a big concern in this project, because user information was being stored for things such as logging in, signing up, booking appointments etc. Therefore, it was important to choose a technology where I could facilitate strong security.

Architecture in Django

The architecture that I used for this project was the MVT architecture. This was explained in depth in section 3.3.2. This was done to allow optimal maintenance and quality.

Databases in Django

Django supports numerous database types including PostgreSQL, MYSQL, SQLite and others. For this project SQLite was used. In the settings file we define the desired database to implement and what engine as can be seen in the figure 22 below. (Django Databases 2023)



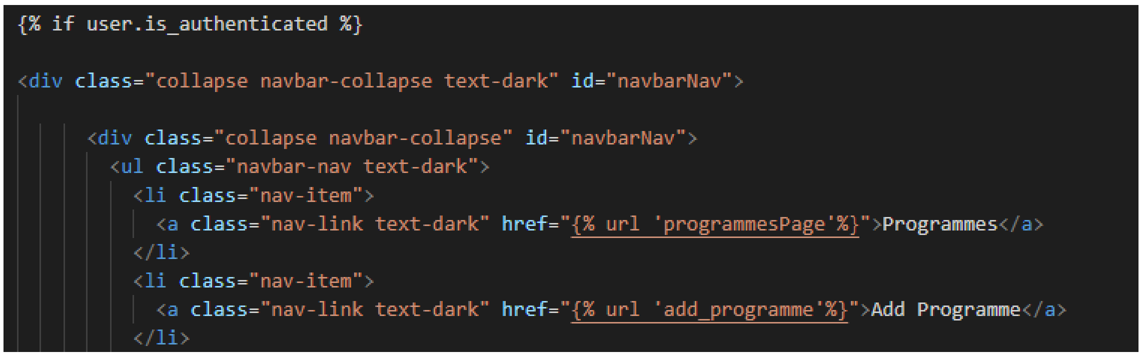
```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```

Figure 22: Implementation of databases in Django

Django uses models which is a single source of information about your data which contains fields and behaviours. Each model will correspond to a database field, which is migrated onto the database before running the site. Once a model is defined it will create a database table with the values attributed to that model. Databases were used in this application to store information regarding the Staff, Appointment, Posts, Programmes, and Comments, log in/sign up etc. You can also create relationships between different models in the application e.g. the relationship between a Staff and Post can be a one to one relationship where a staff member can create one post, or one to many where one staff member can create many posts. (Django Models and databases 2023)

4.1.2 HTML templates

I decided to use HTML templates for the front end. This is because it allows Django to view HTML dynamically. It allows you to combine Django and python logic, for e.g. if statements and for loops to render content onto the web page in a way that the developer intends, whilst meeting certain conditions. An example of how this was implemented in my web app can be seen in the figure 23 below, where we check for user authentication before outputting certain content. Through using the {% if user.is_authenticated %} tag.



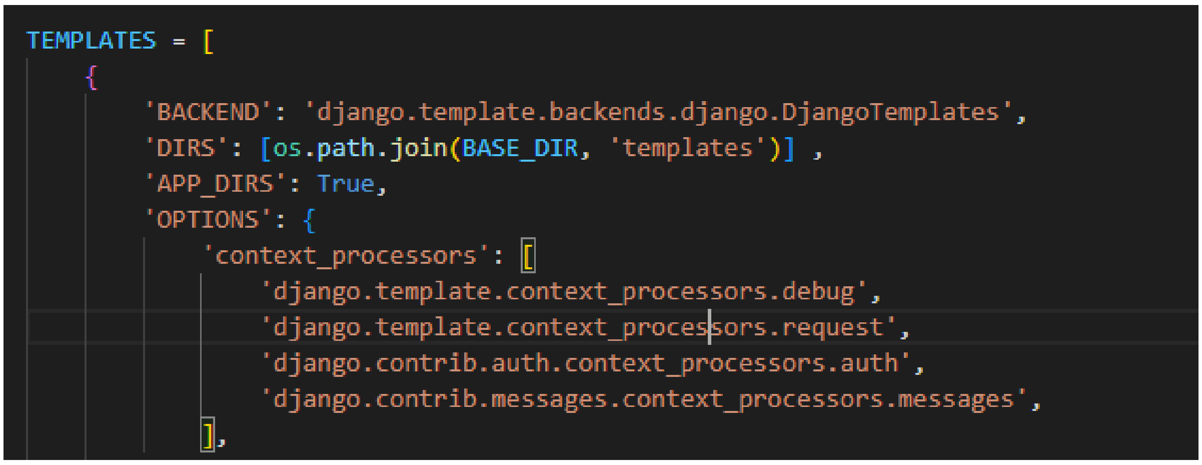
```
{% if user.is_authenticated %}

<div class="collapse navbar-collapse text-dark" id="navbarNav">

      <div class="collapse navbar-collapse" id="navbarNav">
        <ul class="navbar-nav text-dark">
          <li class="nav-item">
            <a class="nav-link text-dark" href="{% url 'programmesPage'%}">Programmes</a>
          </li>
          <li class="nav-item">
            <a class="nav-link text-dark" href="{% url 'add_programme'%}">Add Programme</a>
          </li>
```

Figure 23: Implementation of HTML templates to authenticate users

Template engines are used to configure a Django project. There is a settings file where we define the template engine (Figure 24) where we define the path to the engine class implementing the template backend API. We also define a list using the 'DIRS' variable to define where the engine should look for template source files. In our case the files were in a folder called 'templates'. We use the APP_DIRS variable which determines whether the engine should look for templates inside installed apps which in our case we wanted it to so we set it to true. (HTML Templates 2023)



```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates')] ,
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
```

Figure 24: Implementation of template engines to define a template

There are many benefits of using html templates including the ability to combine it with Django to produce more dynamic content. You can also combine it with libraries such as bootstrap to increase the aesthetics of the website.

4.1.3 Google Maps API

For this project it was important that the relevant support and appointment was easy to find so this project can distinguish itself from others. The best way to do this was to give directions to their appointments upon booking them to make it that much easier. After doing some research the best way to implement this was through the the google maps API. This allows you to embed Google maps into web pages and mobile apps as well as retrieve data. It does this through giving the user a key which then gives them access to numerous map functionalities. This was the perfect for our example as the Google Maps platform provided API's for different map related functionalities including API's for routing, directions, and maps. These API's were used in this application for users to input their address and the destination address of the appointment, and the distance, directions, time of journey would then be outputted. The exact implementation details of how this was done for this project is discussed in section 4.3. (Google Maps Platform 2023)

4.2 System Architecture

The system architecture for my project has been designed below in figure 25. Here we see how different platforms, technologies and functionalities integrated together to develop a full scape web application. This includes front end, back end and usage of the relevant API's. It is important that they are integrated properly so the application works.

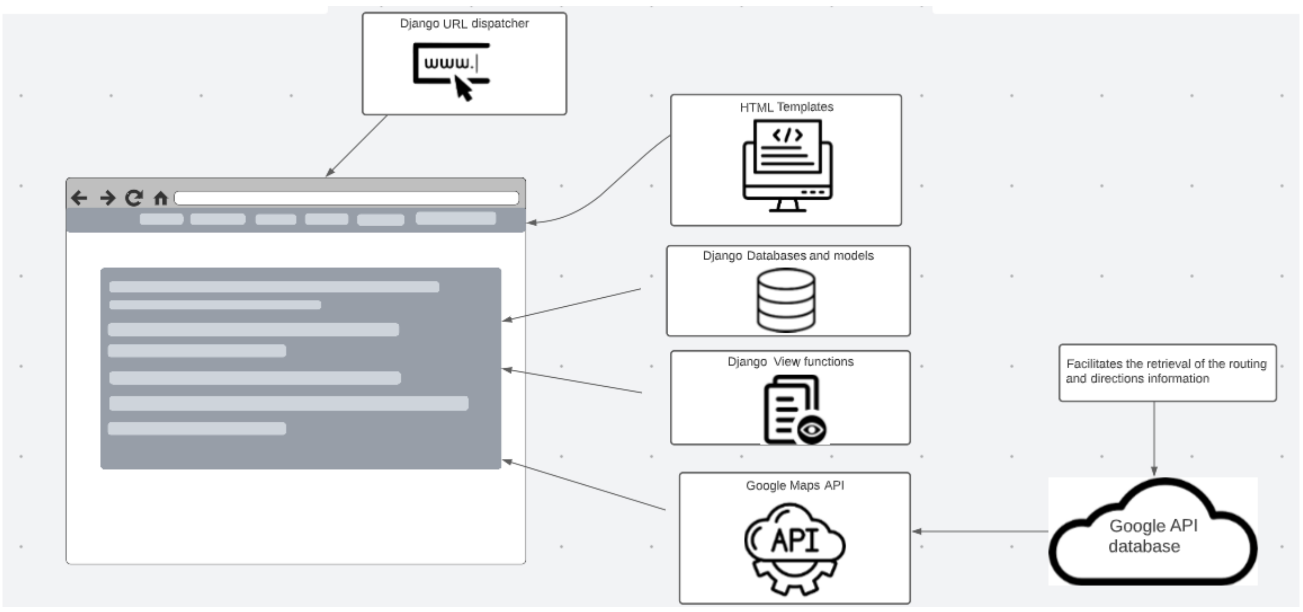


Figure 25: Diagram showing different architectures implemented in the project

4.3 Implementation details

4.3.1 Application Model and screen flow

The social welfare helper application has 23 screens in total with majority of them being follow on screens from the main areas being the Programme, Post, Appointment, and Account related screens. The flow of the application which involves the users navigating through various screens is shown in figure 26. The screen that will always be started with is the home screen containing a navigation bar pointing to different areas depending on the type of user. As can be seen from figure 26, the screens shown differ between a logged-in and non-logged-in user, as they would be using the application for different purposes. However, the staff member can navigate some of the screens that a regular user can navigate that may not have been explicitly labelled in the diagram. Some of these include the Programme, Posts, and the Comment screen.

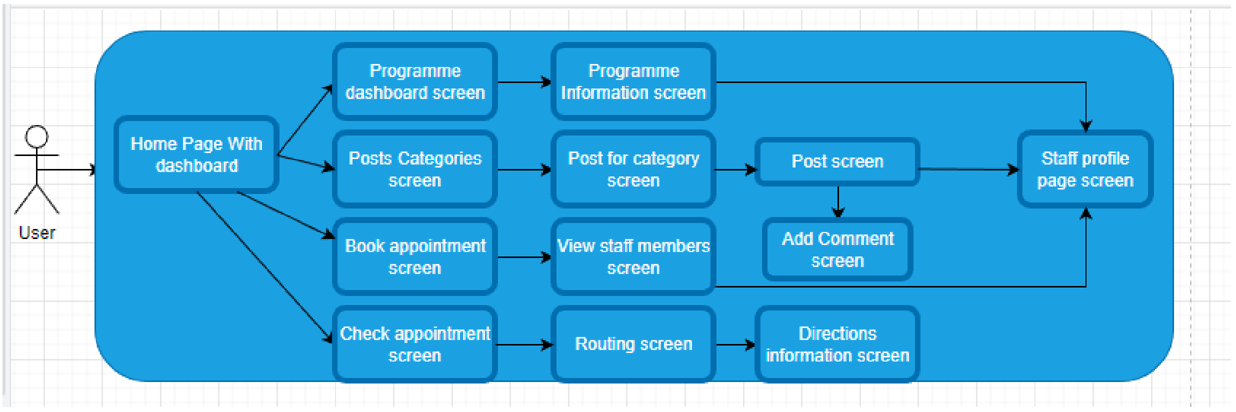


Figure 26: Diagram showing flow of application from general user's point of view

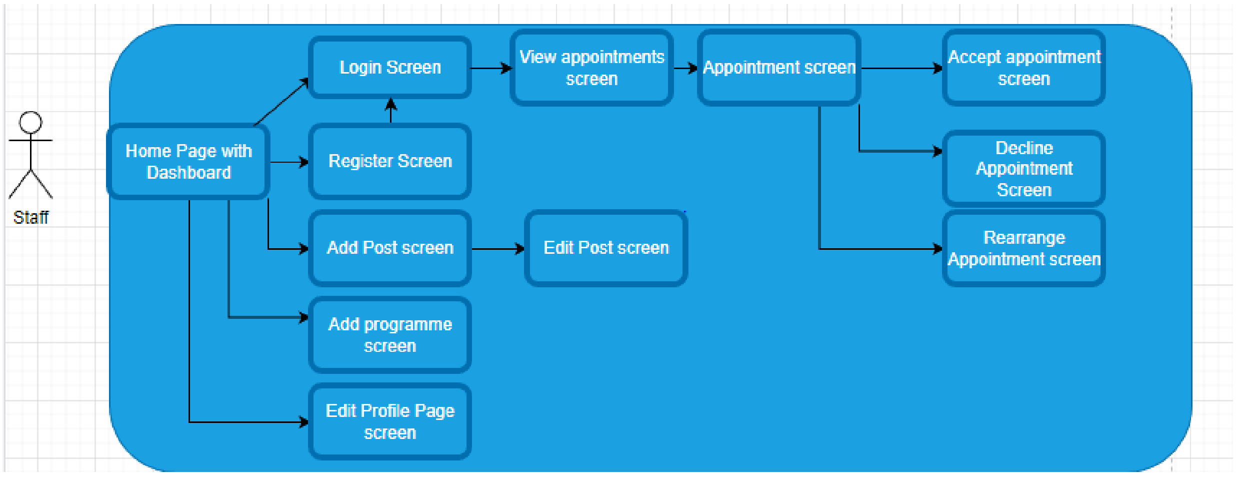
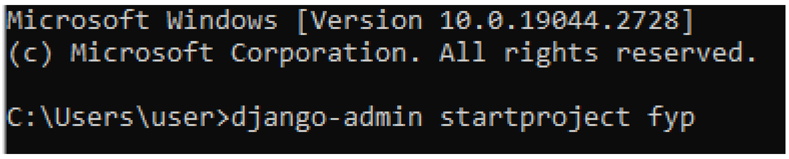


Figure 27: Diagram showing flow of application from staff members point of view

4.3.2 Project Setup

To start implementing the web app, we had to set up the Django project. This was done using the command line interface (CLI). First, the Django framework was installed. Once Django is installed you direct the CLI to the folder/directory where you want your code to be stored. Then you create a project using the following command (figure 28)

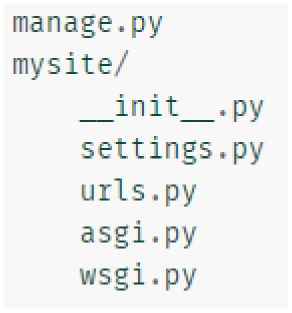


```
Microsoft Windows [Version 10.0.19044.2728]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user>django-admin startproject fyp
```

Figure 28: Shows setting up the Django project using the command line interface

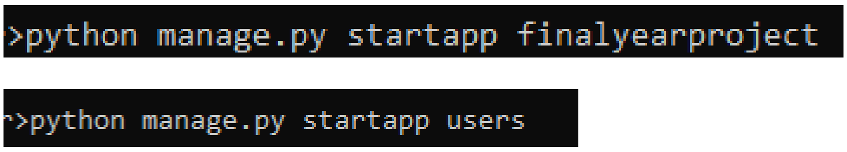
This creates some python files that will be necessary including the settings file, URL files etc as seen in figure 29.



```
manage.py
mysite/
    __init__.py
    settings.py
    urls.py
    asgi.py
    wsgi.py
```

Figure 29: Shows organisation of different files once the project has been created

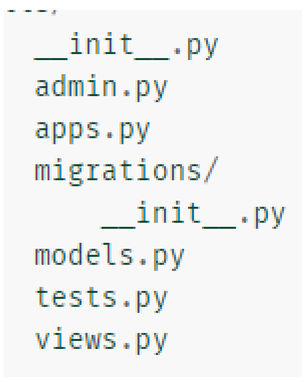
This project will act as an environment where you can then create one or more apps. The app refers to the web app that will carry out functionalities and the project refers to the collection of apps for a particular website. 2 apps were created in this project, one for the website and one for the users which takes care of profile pages, login, registering etc. Figure 30 shows commands used to create the app and leads to the creation of other files as seen in figure 31. These files allow database creation, URL dispatchers, view functions, migrations for when new databases are created and admins which are used to register the models you create onto the website. (Django apps 2023)



```
>python manage.py startapp finalyearproject
```

```
>python manage.py startapp users
```

Figure 30: Shows Process of starting a Django app using the CLI



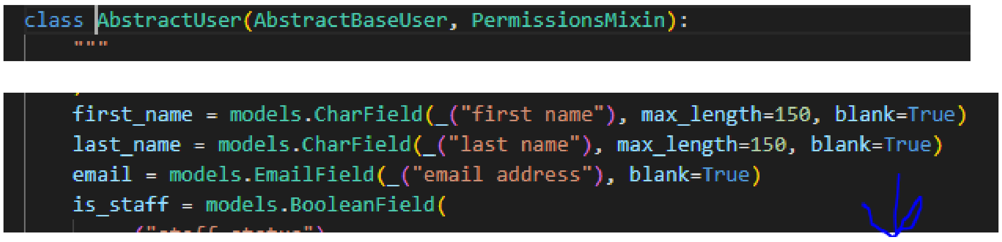
```
  __init__.py
  admin.py
  apps.py
  migrations/
      __init__.py
  models.py
  tests.py
  views.py
```

Figure 31: Shows organisation of files once the Django app has been created

The actual workspace was now created and the actual implementation could begin.

4.3.3 Account Setup

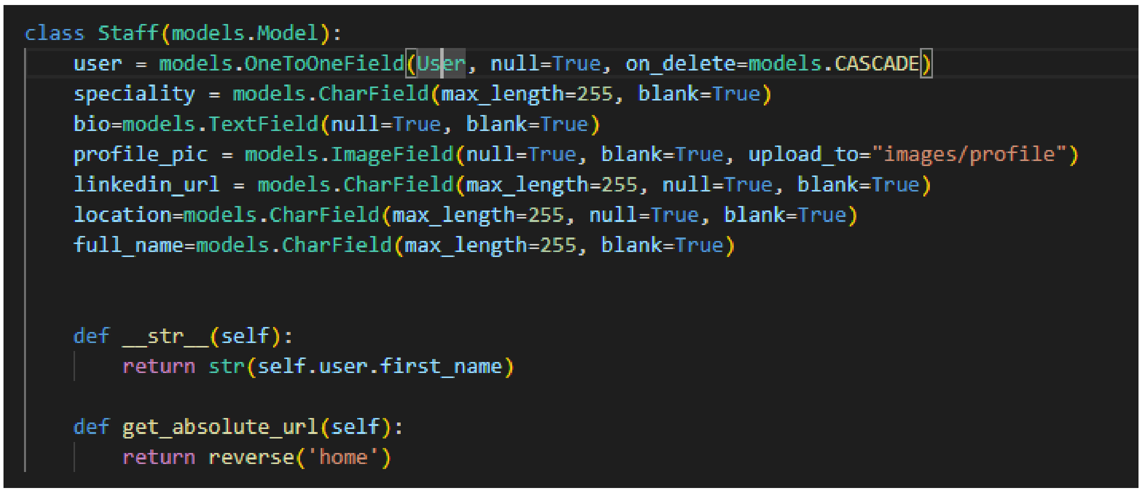
It is always an important stage during the development of a web application to develop the entities of data that you expect to be stored in the app, this is done through models. One of the main models of the application was the Staff model. This is what stores the information of each staff member so they can add posts, programs and accept appointments. Django has a generic User model which inherits from the AbstractUser model (figure 32). This models stored commonly used attributes such first names, last names, emails etc. This means that whenever you make a new model in Django you don’t have to define new fields for these attributes, you can simply inherit them from the generic Django User model using the OneToOneField functionality (figure 33). This was done in this project to develop the Staff, which we then added some extra fields of our own including the Bio, speciality, location, linkedin_url etc which can be included by the staff member themselves when they create and edit their profile page.



```
class AbstractUser(AbstractBaseUser, PermissionsMixin):
    """
```

```
    first_name = models.CharField(_("first name"), max_length=150, blank=True)
    last_name = models.CharField(_("last name"), max_length=150, blank=True)
    email = models.EmailField(_("email address"), blank=True)
    is_staff = models.BooleanField(
```

Figure 32: shows the Abstract user Class to be inherited from



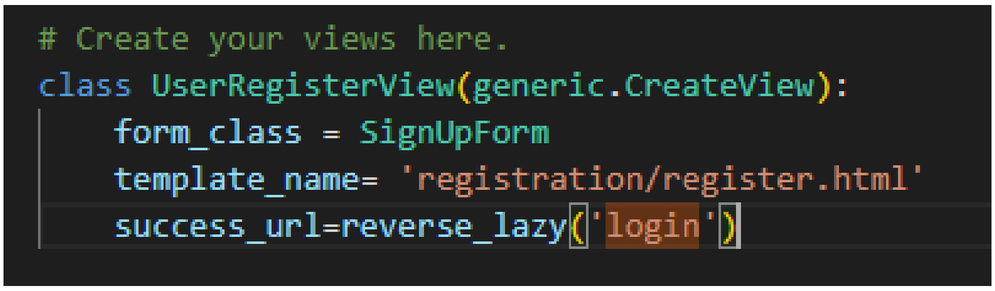
```
class Staff(models.Model):
    user = models.OneToOneField(User, null=True, on_delete=models.CASCADE)
    speciality = models.CharField(max_length=255, blank=True)
    bio=models.TextField(null=True, blank=True)
    profile_pic = models.ImageField(null=True, blank=True, upload_to="images/profile")
    linkedin_url = models.CharField(max_length=255, null=True, blank=True)
    location=models.CharField(max_length=255, null=True, blank=True)
    full_name=models.CharField(max_length=255, blank=True)


    def __str__(self):
        return str(self.user.first_name)

    def get_absolute_url(self):
        return reverse('home')
```

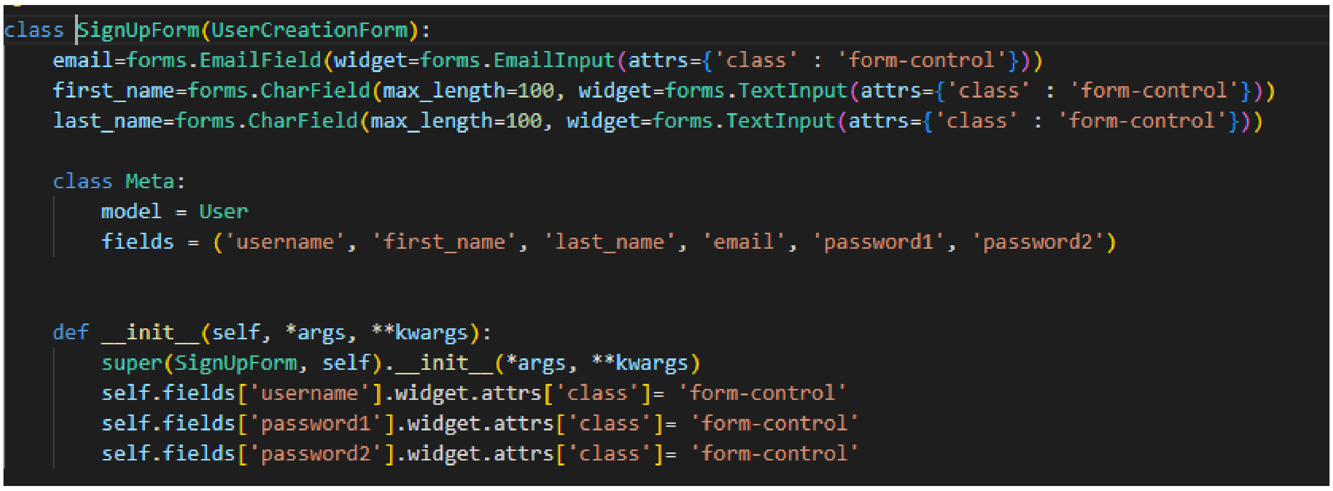
Figure 33: Shows staff model and the different attributes belonging to the staff member

Once the model was created the login, sign up and profile page functionalities was implemented. This was done through view functions and Django forms. Figure 34 shows the usage of Django generics to create a class for the view as supposed to a normal function. This made it easier to implement the forms and render the template. We assign the form_class variable to Signup Form which is a form that we create using Django forms as well as the directory for the template (register.html) so that the program knows what template to render when this view function is called. Once the view function is created we create the forms in a separate file (forms.py) for readability purposes. We created the sign up forms as a class (figure 35). Django has a generic UserCreationForm which stores common form attributes including username and password fields. This way we were able to inherit the UserCreation form and also adding our own extra fields such as email, first_name, last_name which the user can then input when signing up. The Meta class is used to structure the behaviour of the form by determining the order of the fields showing up on the form. We also use it to state the model which will be effected upon the completion of this form.



```
# Create your views here.
class UserRegisterView(generic.CreateView):
    form_class = SignUpForm
    template_name= 'registration/register.html'
    success_url=reverse_lazy('login')
```

Figure 34: Shows implementation of a user registration page in Django

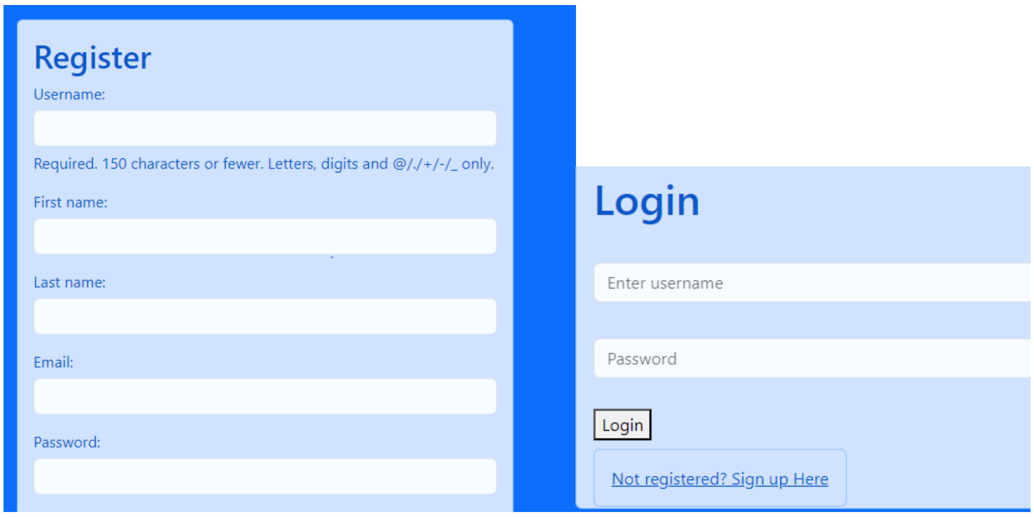


```
class SignUpForm(UserCreationForm):
    email=forms.EmailField(widget=forms.EmailInput(attrs={'class' : 'form-control'}))
    first_name=forms.CharField(max_length=100, widget=forms.TextInput(attrs={'class' : 'form-control'}))
    last_name=forms.CharField(max_length=100, widget=forms.TextInput(attrs={'class' : 'form-control'}))

    class Meta:
        model = User
        fields = ('username', 'first_name', 'last_name', 'email', 'password1', 'password2')


    def __init__(self, *args, **kwargs):
        super(SignUpForm, self).__init__(*args, **kwargs)
        self.fields['username'].widget.attrs['class']= 'form-control'
        self.fields['password1'].widget.attrs['class']= 'form-control'
        self.fields['password2'].widget.attrs['class']= 'form-control'
```

Figure 35: Shows implementation of a sign up form in Django



Register

Username:

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

First name:

Last name:

Email:

Password:

Login

Enter username

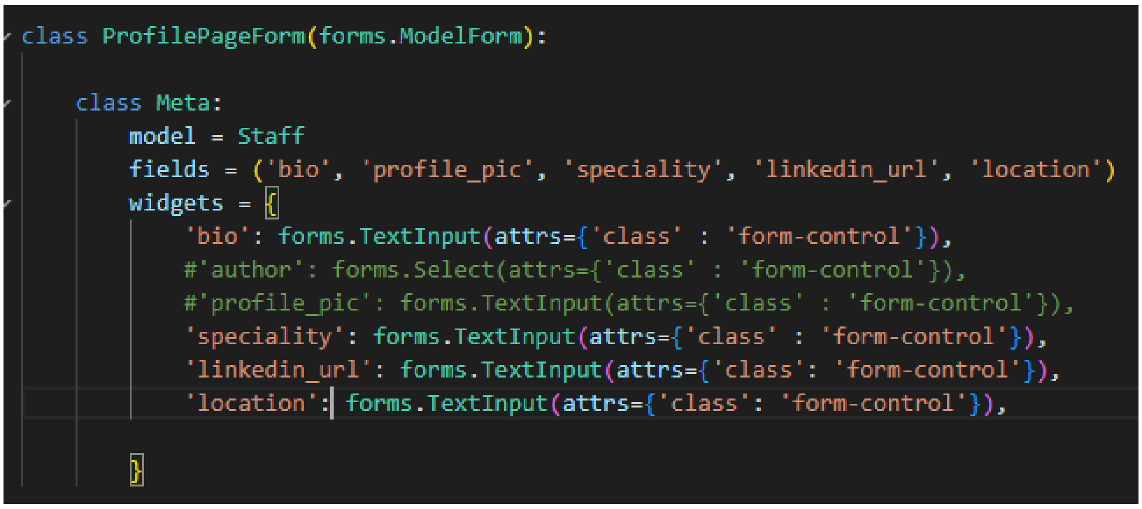
Password

Login

Not registered? Sign up Here

Figure 36: Shows result of implementation of sign up and login form

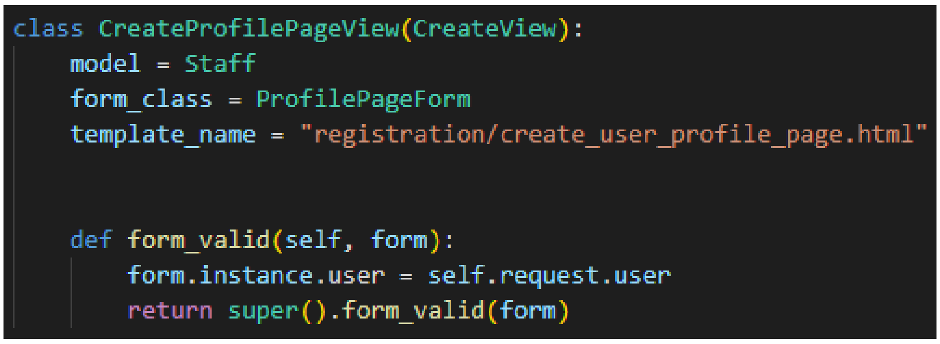
This same process was used when it came to creating and viewing the profile page of the staff members. We use a class based view for the staff member to create their own profile page which includes information such as the bio, profile pics etc. I created a form for the profile page (figure 37) with the extra fields mentioned. This will be what the users see when they see their profile page which is important for events such as appointment booking. We then assign this to the createprofilepage view function to output the form when called to.



```
class ProfilePageForm(forms.ModelForm):

    class Meta:
        model = Staff
        fields = ('bio', 'profile_pic', 'speciality', 'linkedin_url', 'location')
        widgets = {
            'bio': forms.TextInput(attrs={'class' : 'form-control'}),
            #'author': forms.Select(attrs={'class' : 'form-control'}),
            #'profile_pic': forms.TextInput(attrs={'class' : 'form-control'}),
            'speciality': forms.TextInput(attrs={'class' : 'form-control'}),
            'linkedin_url': forms.TextInput(attrs={'class': 'form-control'}),
            'location': forms.TextInput(attrs={'class': 'form-control'}),

        }
```

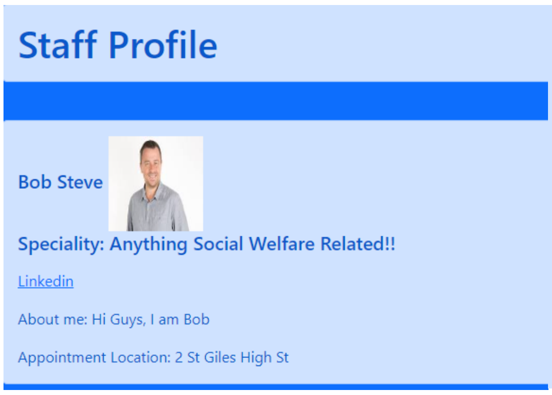


```
class CreateProfilePageView(CreateView):
    model = Staff
    form_class = ProfilePageForm
    template_name = "registration/create_user_profile_page.html"


    def form_valid(self, form):
        form.instance.user = self.request.user
        return super().form_valid(form)
```

Figure 37: form and view function for the profile pages and v

Once the profile is created they then have the ability to edit their profile page. This functionality was done using the similar concept as stated above. However, as you can see in the figure 38 below we used *updateview* as a parameter as this page's main role is for the user to edit the values. If someone wants to see the profile page of a staff member we have created the *ShowProfilePageView* view function. *DetailView* is a parameter to show that this page's responsibility is showing the information of the profile of the staff member.



Staff Profile

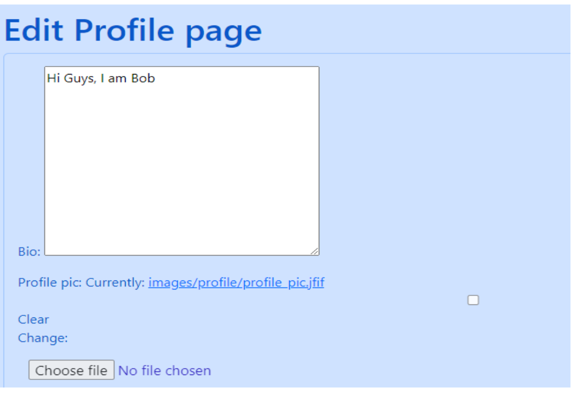
Bob Steve

Speciality: Anything Social Welfare Related!!

Linkedin

About me: Hi Guys, I am Bob

Appointment Location: 2 St Giles High St



Edit Profile page

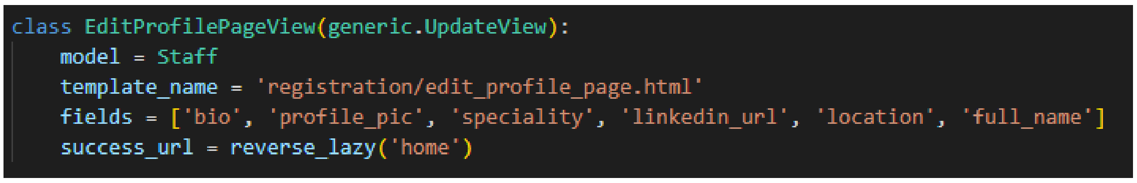
Bio: Hi Guys, I am Bob

Profile pic: Currently: images/profile/profile_pic.jfif

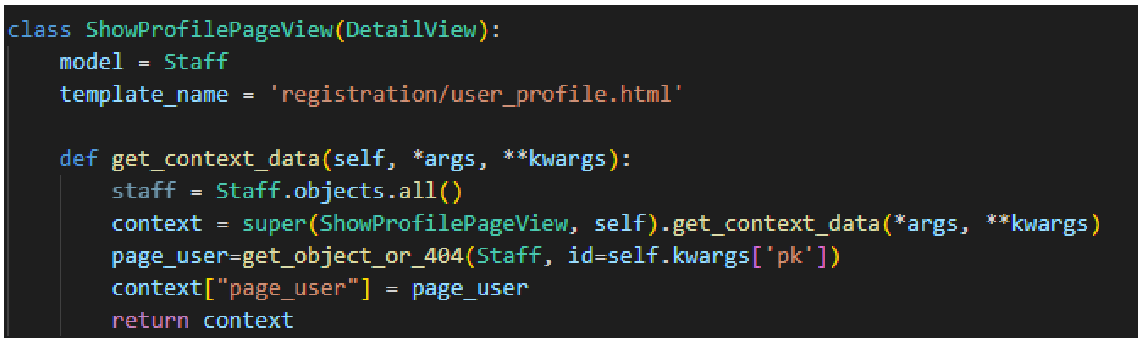
Clear

Change:

Choose file No file chosen



```
class EditProfilePageView(generic.UpdateView):
    model = Staff
    template_name = 'registration/edit_profile_page.html'
    fields = ['bio', 'profile_pic', 'speciality', 'linkedin_url', 'location', 'full_name']
    success_url = reverse_lazy('home')
```



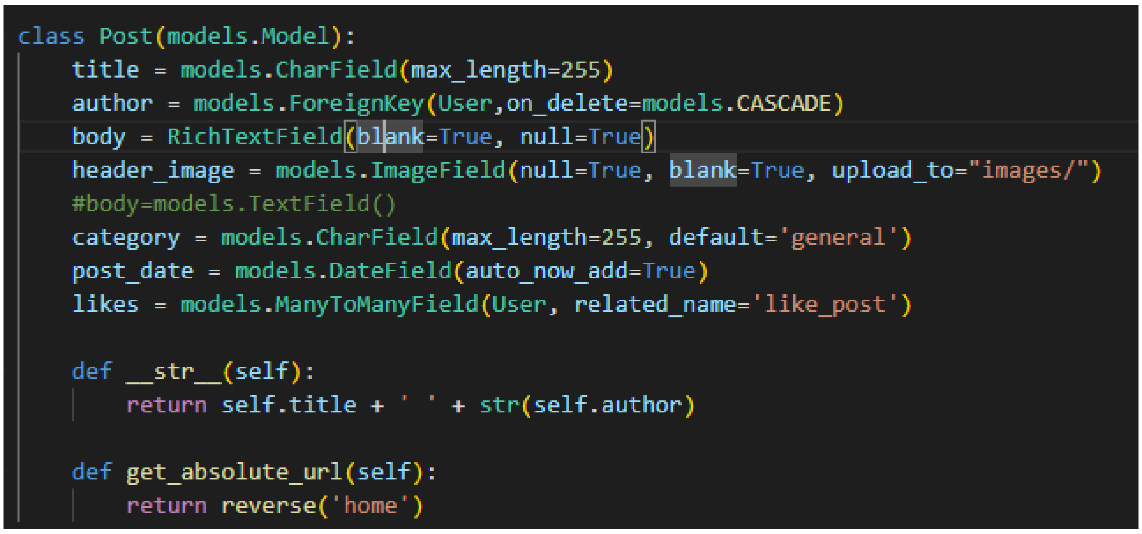
```python
class ShowProfilePageView(DetailView):
    model = Staff
    template_name = 'registration/user_profile.html'

    def get_context_data(self, *args, **kwargs):
        staff = Staff.objects.all()
        context = super(ShowProfilePageView, self).get_context_data(*args, **kwargs)
        page_user=get_object_or_404(Staff, id=self.kwargs['pk'])
        context["page_user"] = page_user
        return context
```

Figure 38: view functions and results of the profile pages

4.3.4 Post setup

Another main aspect of this project was to allow the staff members to post valuable information for the other users to see and comment on. The Post mode created stores information regarding the post which includes the title, body etc. A foreign key for the author attribute aligns each post to the relevant user model who posted it. We also created a Category model which stores the category of the post. The staff member chooses which category their post is relevant for upon making the post. Each post has comments and to facilitate this we create a Comment Model which stores the name and body of the comment and a foreign key which creates a relationship with the post model where each comment corresponds to a particular post.

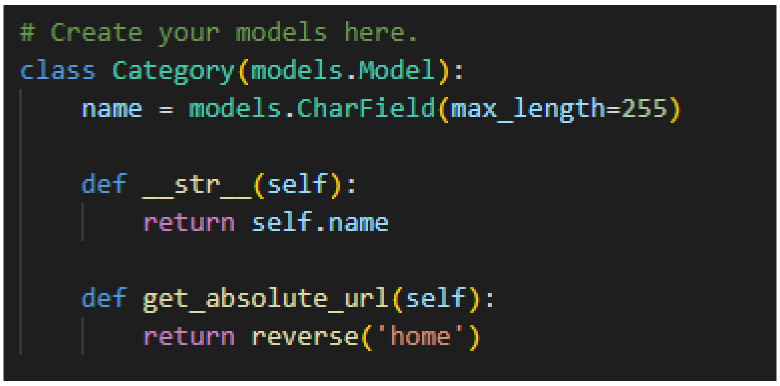


```python
class Post(models.Model):
    title = models.CharField(max_length=255)
    author = models.ForeignKey(User,on_delete=models.CASCADE)
    body = RichTextField(blank=True, null=True)
    header_image = models.ImageField(null=True, blank=True, upload_to="images/")
    #body=models.TextField()
    category = models.CharField(max_length=255, default='general')
    post_date = models.DateField(auto_now_add=True)
    likes = models.ManyToManyField(User, related_name='like_post')

    def __str__(self):
        return self.title + ' ' + str(self.author)

    def get_absolute_url(self):
        return reverse('home')
```

Figure 39: Shows implementation of a Post model in Django

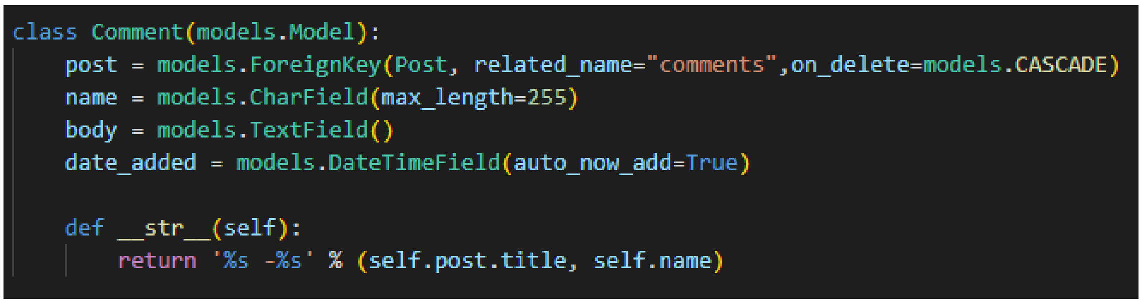


```python
# Create your models here.
class Category(models.Model):
    name = models.CharField(max_length=255)

    def __str__(self):
        return self.name

    def get_absolute_url(self):
        return reverse('home')
```

Figure 40: Shows Implementation of the Category Model

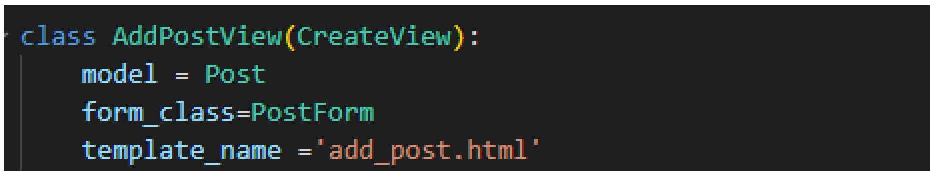


```python
class Comment(models.Model):
    post = models.ForeignKey(Post, related_name="comments",on_delete=models.CASCADE)
    name = models.CharField(max_length=255)
    body = models.TextField()
    date_added = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return '%s -%s' % (self.post.title, self.name)
```

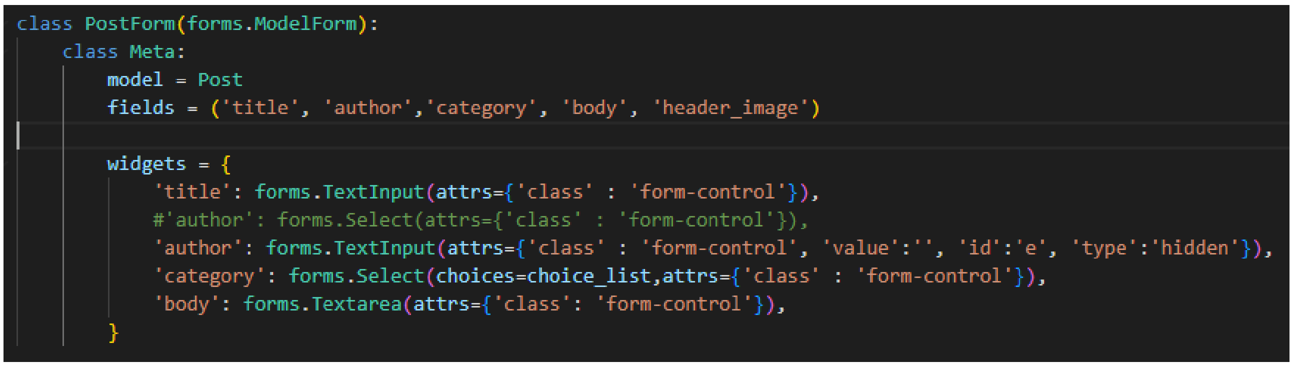
Figure 41: Shows implementation of a Comment model

Once these models were created, the view functions with their respective forms were created (figure 43) as well as the view function to show the actual detail of the post and the view to output the form. This was used in the same way for the comments.



```python
class AddPostView(CreateView):
    model = Post
    form_class=PostForm
    template_name ='add_post.html'
```

Figure 42: Shows implementation of for the view function for adding a post

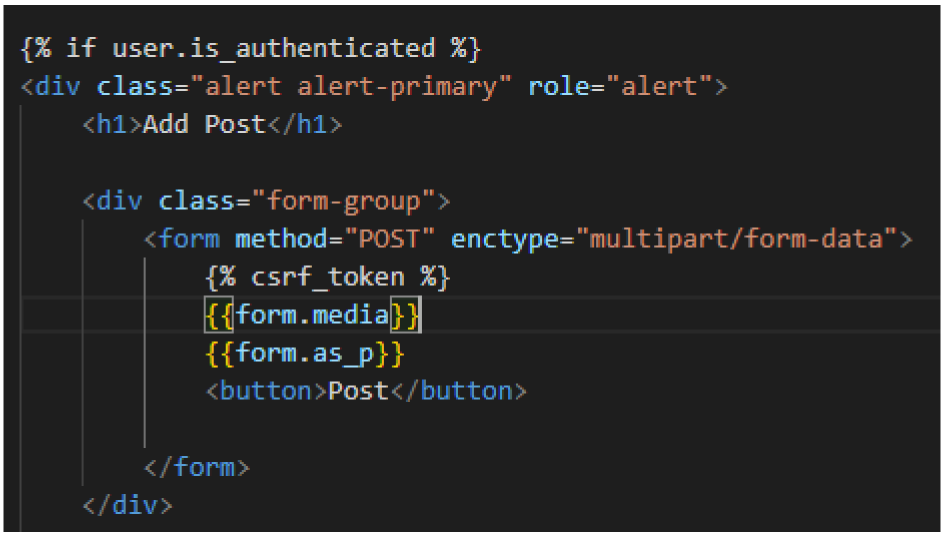


```python
class PostForm(forms.ModelForm):
    class Meta:
        model = Post
        fields = ('title', 'author','category', 'body', 'header_image')

        widgets = {
            'title': forms.TextInput(attrs={'class' : 'form-control'}),
            #'author': forms.Select(attrs={'class' : 'form-control'}),
            'author': forms.TextInput(attrs={'class' : 'form-control', 'value':'', 'id':'e', 'type':'hidden'}),
            'category': forms.Select(choices=choice_list,attrs={'class' : 'form-control'}),
            'body': forms.Textarea(attrs={'class': 'form-control'}),
        }
```

Figure 43: Shows implementation of a Django form to create a post

Once the backend was created we could then implement this in the front end through the use of HTML templates. We first check that the user is authenticated before being able to add a post, then we use the csrf token for security purposes (see section 4.1.1 for further details on this) and we then used placeholders to output the Django form with the appropriate styling. (figure 44)

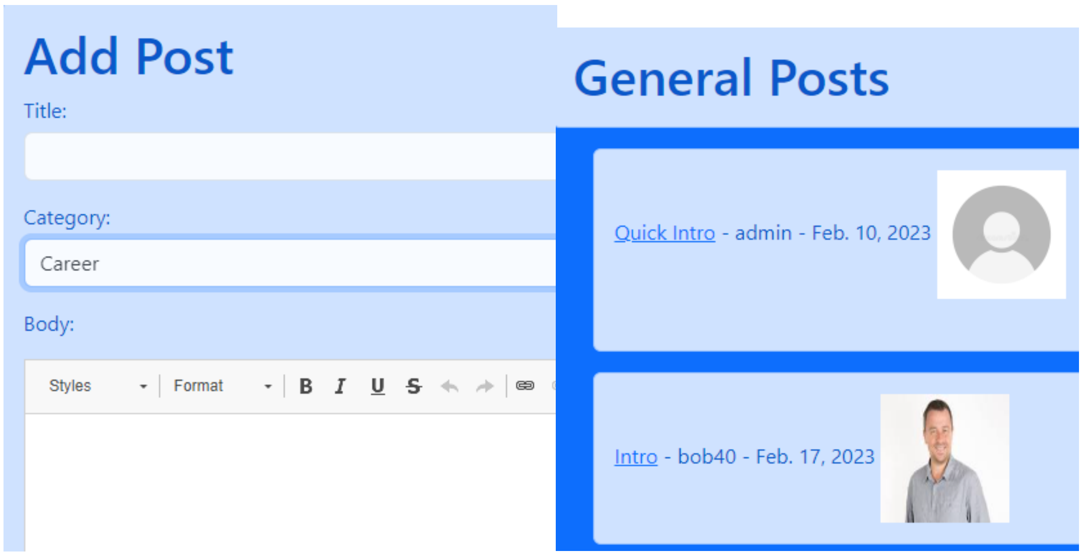


```
{% if user.is_authenticated %}
<div class="alert alert-primary" role="alert">
    <h1>Add Post</h1>

    <div class="form-group">
        <form method="POST" enctype="multipart/form-data">
            {% csrf_token %}
            {{form.media}}
            {{form.as_p}}
            <button>Post</button>

        </form>
    </div>
```

Figure 44: Shows implementation of using HTML templates with Django for user authentication



Add Post

Title:

Category:

Career

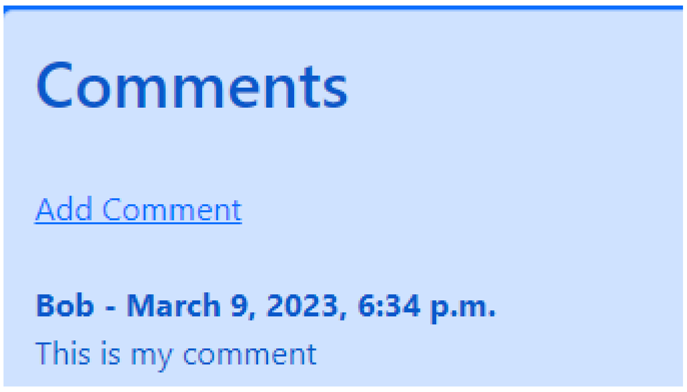
Body:

General Posts

Quick Intro - admin - Feb. 10, 2023

Intro - bob40 - Feb. 17, 2023

Figure 45: shows results of adding a post and viewing posts



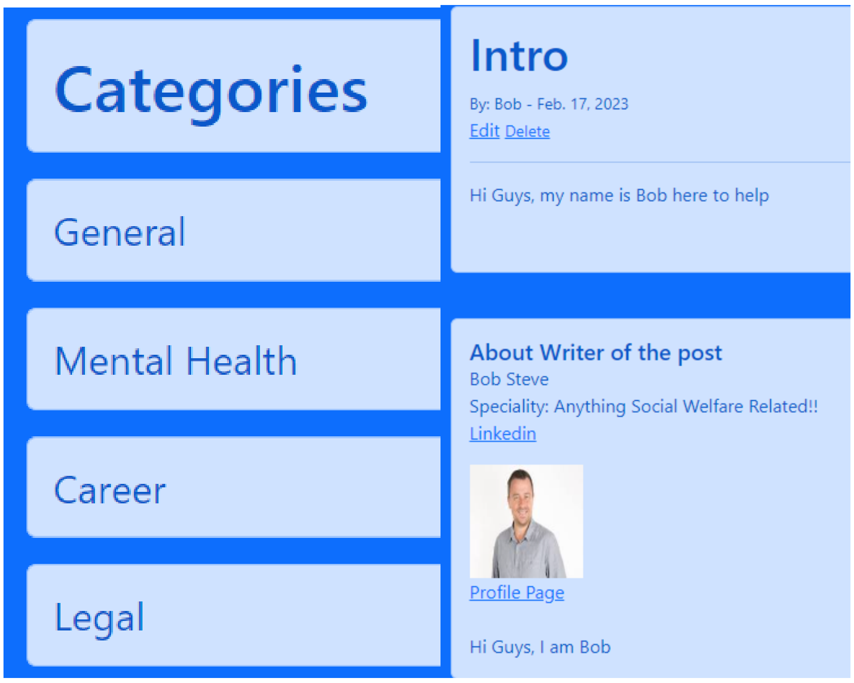
Comments

Add Comment

Bob - March 9, 2023, 6:34 p.m.

This is my comment

Figure 46: Results of implementing a comment section



Intro

By: Bob - Feb. 17, 2023

Edit Delete

Hi Guys, my name is Bob here to help

About Writer of the post

Bob Steve

Speciality: Anything Social Welfare Related!!

Linkedin

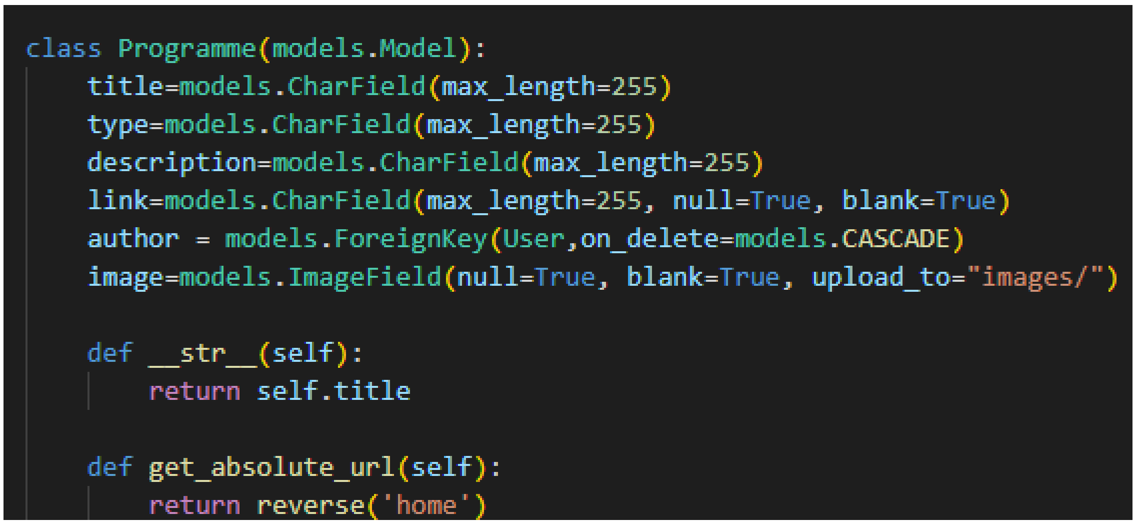
Profile Page

Hi Guys, I am Bob

Figure 47: Result of implementing a categories section as well as viewing a post

4.3.5 Programme setup

Another aspect of this project was allowing staff members to post different types of programmes with the necessary information on how to benefit from these programes fir the users to potential partake. A similar concept with the Posts set up was used for this including the model, the view function and integration with the front end. It was key to include a link so that they can for navigate to the web page where they can access a particular program as well as the staff member who posted the program. This is because if they need more information/advice about a particular program they can book an appointment with the person who posted that programme.



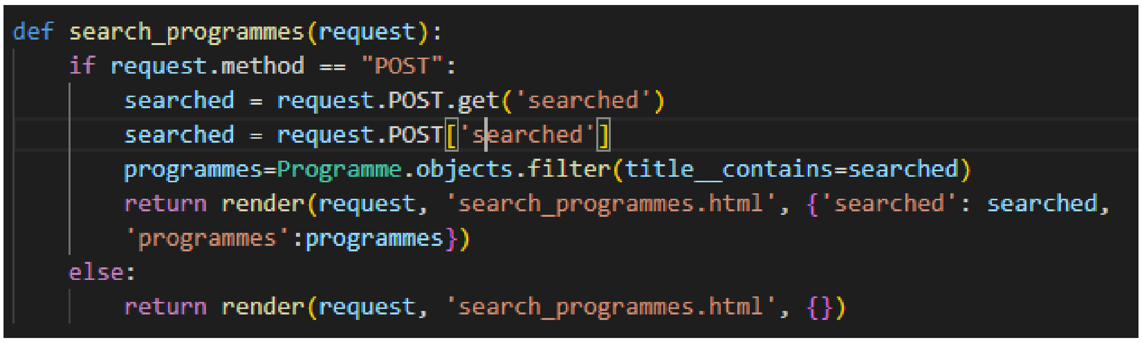
```
class Programme(models.Model):
    title=models.CharField(max_length=255)
    type=models.CharField(max_length=255)
    description=models.CharField(max_length=255)
    link=models.CharField(max_length=255, null=True, blank=True)
    author = models.ForeignKey(User,on_delete=models.CASCADE)
    image=models.ImageField(null=True, blank=True, upload_to="images/")

    def __str__(self):
        return self.title

    def get_absolute_url(self):
        return reverse('home')
```

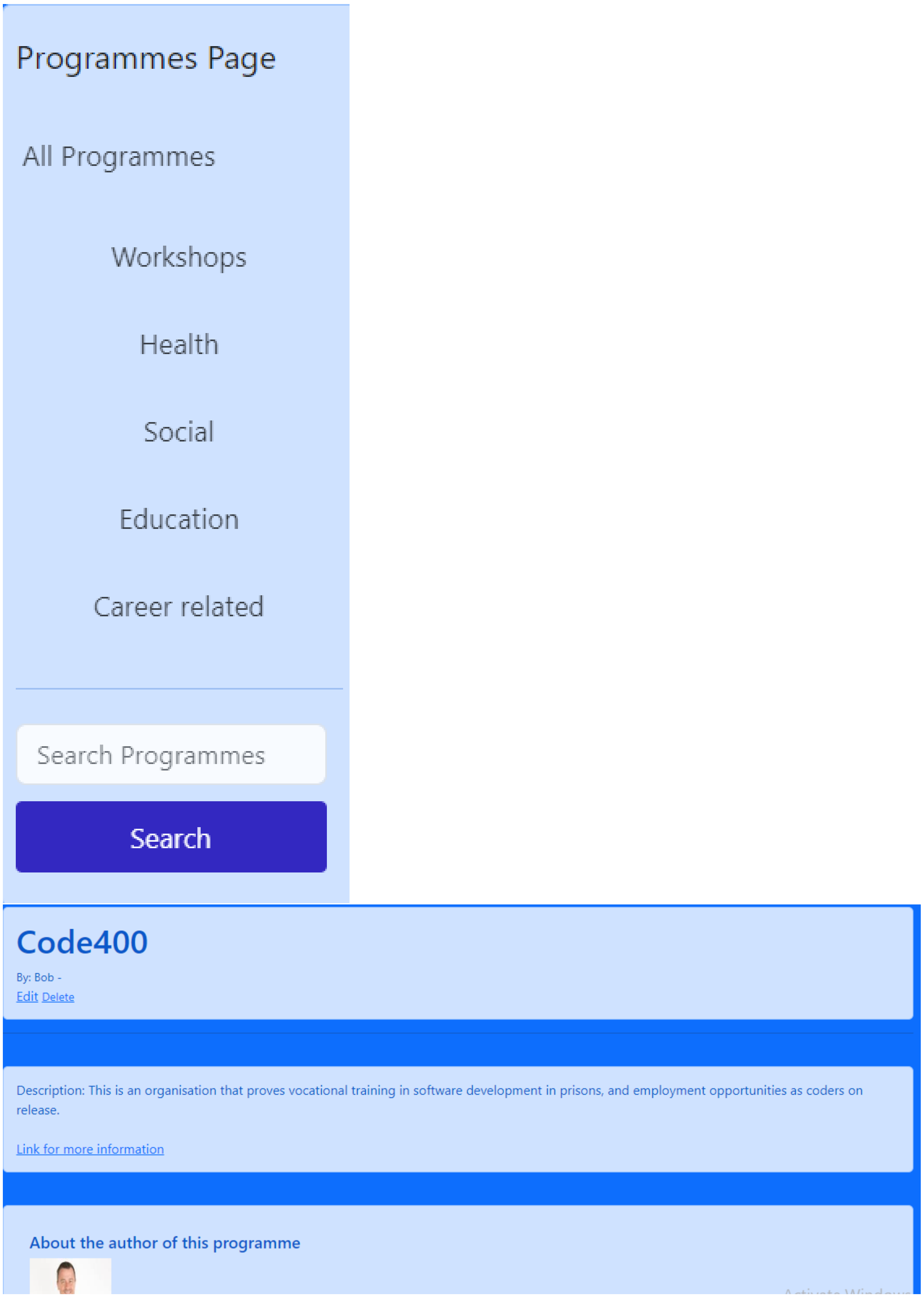
Figure 48: Implementation of a programme model

Another distinct feature of this section is for the user to search for programmes on their own using a search bar that is located on the page. We extract what they have inputted on the search bar using the get() method (figure 49), and then filter all the programmes on the system and output only the programmes that are within the constraints of what the user has searched for. After this a new html page will be rendered, outputting the information filtered accordingly. The user can then navigate each programme for more details accordingly.



```python
def search_programmes(request):
    if request.method == "POST":
        searched = request.POST.get('searched')
        searched = request.POST['searched']
        programmes=Programme.objects.filter(title__contains=searched)
        return render(request, 'search_programmes.html', {'searched': searched,
        'programmes':programmes})
    else:
        return render(request, 'search_programmes.html', {})
```

Figure 49: Implementation of the search bar functionality to search for programmes



Code400

By: Bob -

Edit Delete

Description: This is an organisation that proves vocational training in software development in prisons, and employment opportunities as coders on release.

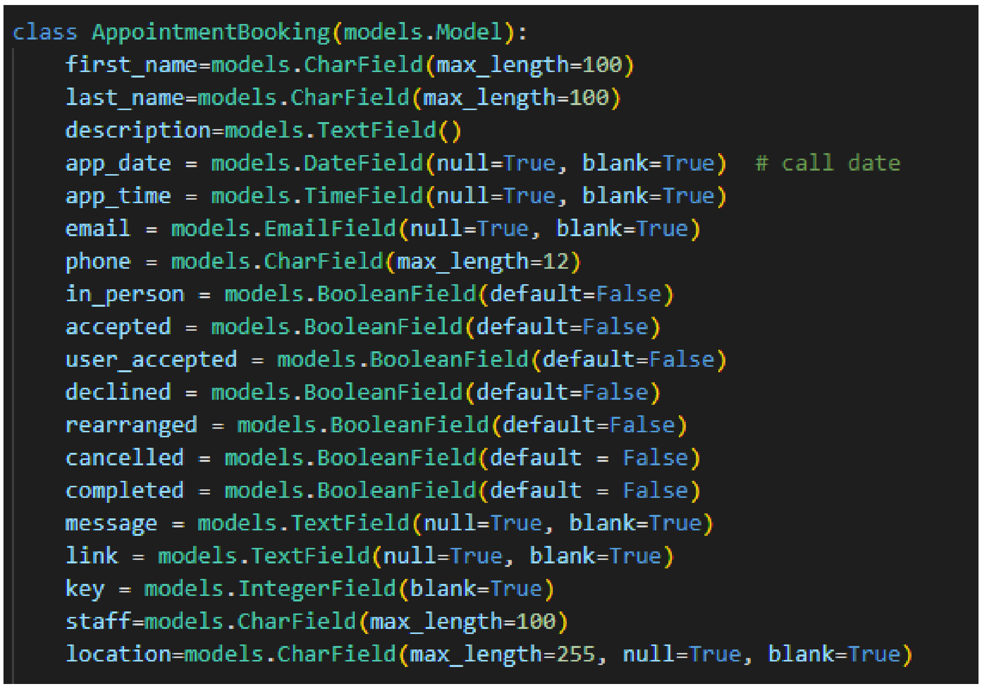
Link for more information

About the author of this programme

Figure 50: shows result of implementation of the programmes section

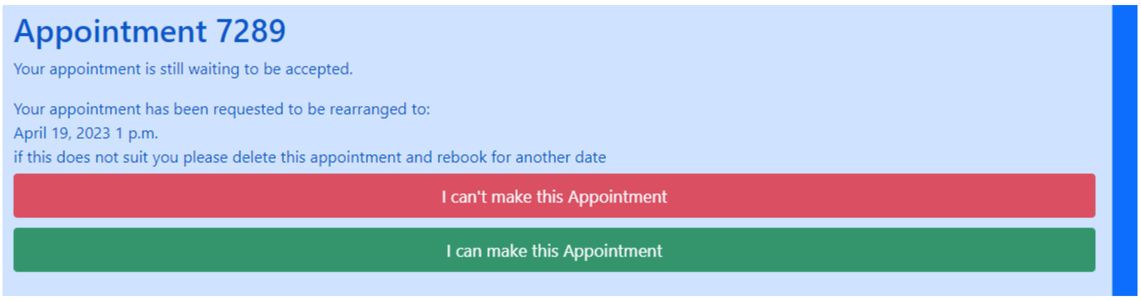
4.3.6 Appointment setup

The ability to book appointments with the relevant staff was the main key aspect of this project. Once they have booked an appointment, the staff member can accept/decline or rearrange with the appointment and the user would have the option to either make the appointment in-person or online. There are many fields used in the model ranging from user info, location etc (figure 51). If the appointment gets changed/cancelled this information is shown to the user (figure 52). The message field is used so that if the staff member for e.g. declines an appointment, they can write a message as to why they are doing so and give any recommendations to the user. As the user doesn't need to create an account, the way they are able to use a key that they are given. They simply search for the status of their appointment using a randomly generated key and they are given the information regarding their appointment. (figure 52)



```
class AppointmentBooking(models.Model):
    first_name=models.CharField(max_length=100)
    last_name=models.CharField(max_length=100)
    description=models.TextField()
    app_date = models.DateField(null=True, blank=True)  # call date
    app_time = models.TimeField(null=True, blank=True)
    email = models.EmailField(null=True, blank=True)
    phone = models.CharField(max_length=12)
    in_person = models.BooleanField(default=False)
    accepted = models.BooleanField(default=False)
    user_accepted = models.BooleanField(default=False)
    declined = models.BooleanField(default=False)
    rearranged = models.BooleanField(default=False)
    cancelled = models.BooleanField(default = False)
    completed = models.BooleanField(default = False)
    message = models.TextField(null=True, blank=True)
    link = models.TextField(null=True, blank=True)
    key = models.IntegerField(blank=True)
    staff=models.CharField(max_length=100)
    location=models.CharField(max_length=255, null=True, blank=True)
```

Figure 51: Shows implementation of a model that represents the booking of an appointment



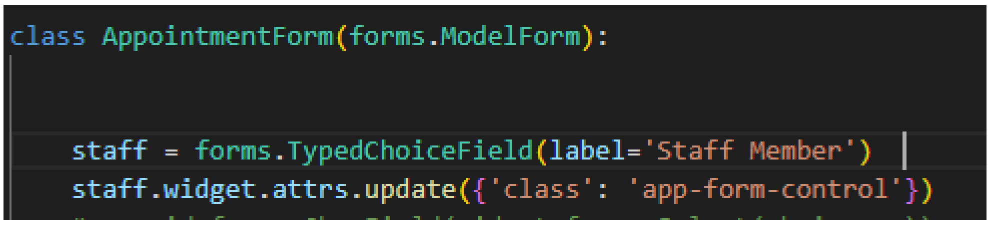
Appointment 7289

Your appointment is still waiting to be accepted.

Your appointment has been requested to be rearranged to:
April 19, 2023 1 p.m.
if this does not suit you please delete this appointment and rebook for another date

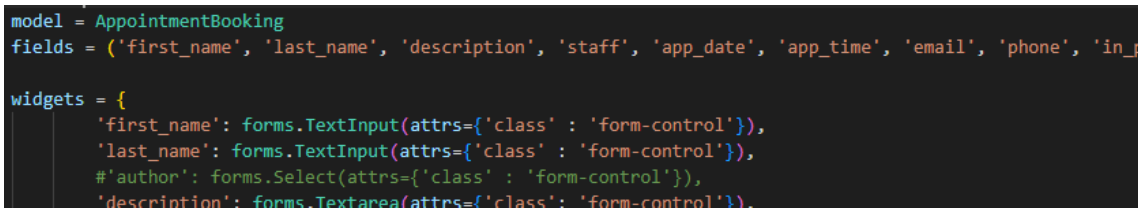
Figure 52: Results of appointment booking and rearrangement

I then proceeded with creating the view function to book appointment. This involved creating the form (figure 53) where I imported the form created into the view function. In the form we define the fields for the user to input and style them appropriately using the 'form-control' attribute. (figure 53)



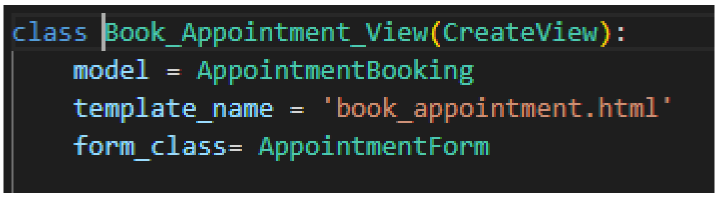
```
class AppointmentForm(forms.ModelForm):


    staff = forms.TypedChoiceField(label='Staff Member')
    staff.widget.attrs.update({'class': 'app-form-control'})
```



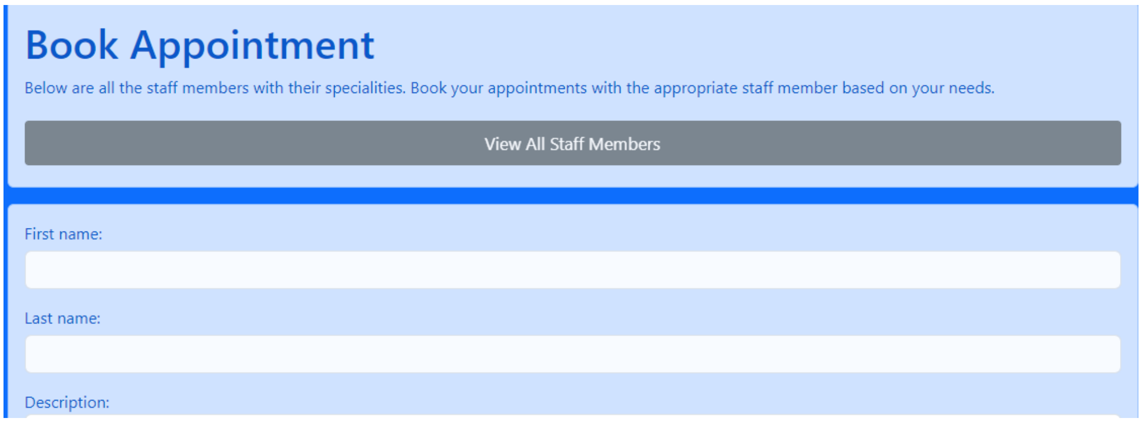
```
model = AppointmentBooking
fields = ('first_name', 'last_name', 'description', 'staff', 'app_date', 'app_time', 'email', 'phone', 'in_

widgets = {
        'first_name': forms.TextInput(attrs={'class' : 'form-control'}),
        'last_name': forms.TextInput(attrs={'class' : 'form-control'}),
        #'author': forms.Select(attrs={'class' : 'form-control'}),
        'description': forms.Textarea(attrs={'class': 'form-control'}),
```



```
class Book_Appointment_View(CreateView):
    model = AppointmentBooking
    template_name = 'book_appointment.html'
    form_class= AppointmentForm
```

Figure 53: Shows implementation of a form and view function to book the appointment



Book Appointment

Below are all the staff members with their specialities. Book your appointments with the appropriate staff member based on your needs.

View All Staff Members

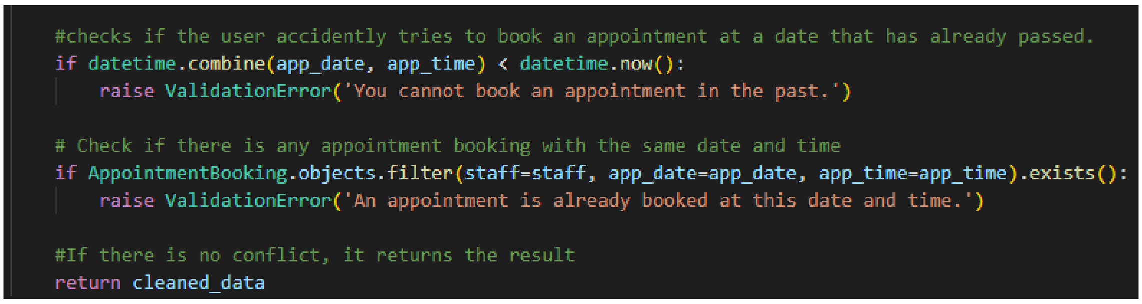
First name:

Last name:

Description:

Figure 54: Result of appointment booking form

To prevent clashes and booking appointments that are in a past date I included some validation (figure 55). If these conditions are met it returns an error using the ValidationError() method and prevents the appointment from being booked

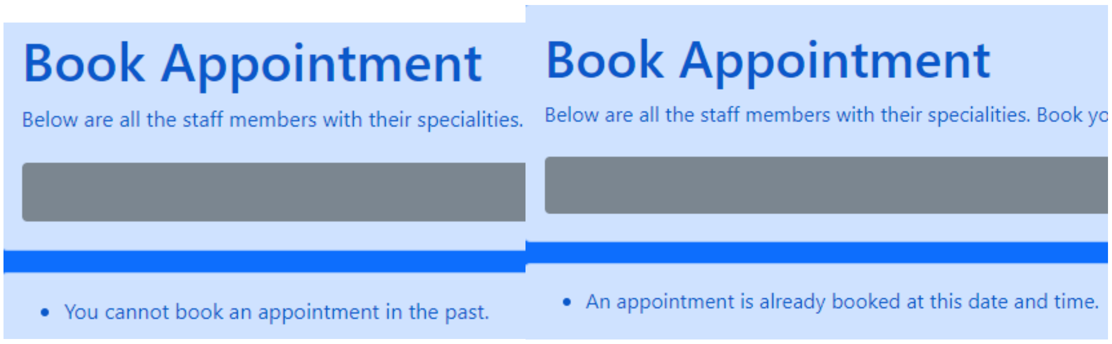


```
    #checks if the user accidently tries to book an appointment at a date that has already passed.
    if datetime.combine(app_date, app_time) < datetime.now():
        raise ValidationError('You cannot book an appointment in the past.')

    # Check if there is any appointment booking with the same date and time
    if AppointmentBooking.objects.filter(staff=staff, app_date=app_date, app_time=app_time).exists():
        raise ValidationError('An appointment is already booked at this date and time.')

    #If there is no conflict, it returns the result
    return cleaned_data
```

Figure 55: Shows implementation of validation for the appointment booking form to prevent clashes



Book Appointment

Below are all the staff members with their specialities.

- You cannot book an appointment in the past.

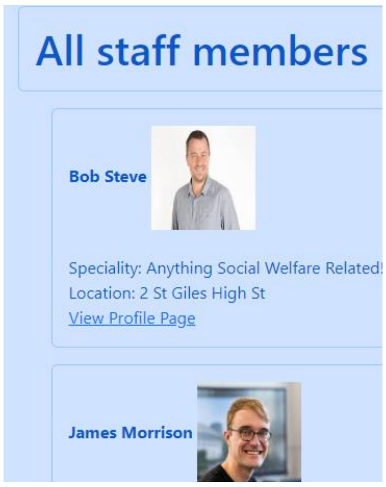
Book Appointment

Below are all the staff members with their specialities. Book yo

- An appointment is already booked at this date and time.

Figure 56: Shows result of form validation implementation

There is also an interface that allows the user to access a list of all the staff members before booking an appointment which is in the “all_staff_members.html” file.



All staff members

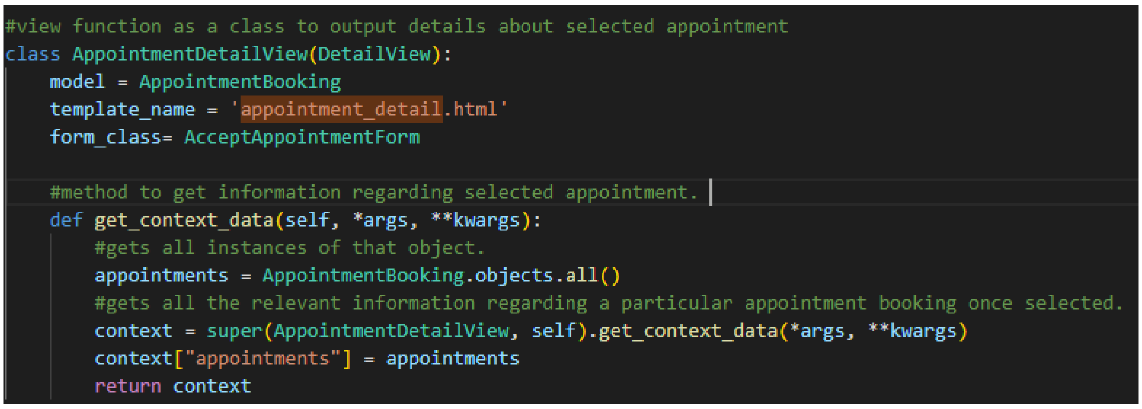
Bob Steve

Speciality: Anything Social Welfare Related!
Location: 2 St Giles High St
View Profile Page

James Morrison

Figure 57: Shows all staff members being shown to the user

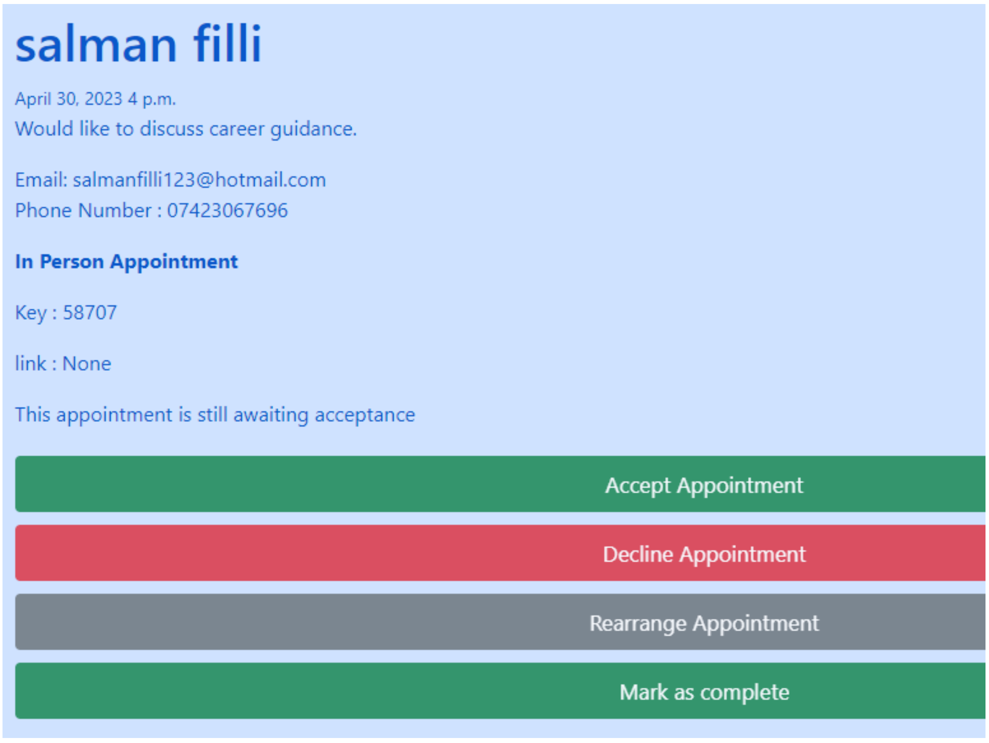
Once the appointment has been booked the staff member can view all their appointments and navigate them one by one. Once the staff member clicks on a certain appointment they can accept/reject/rearrange this appoint by filling in the relevant details such as location, the link etc. Appointment information is outputted using the view function (figure 58).



```
#view function as a class to output details about selected appointment
class AppointmentDetailView(DetailView):
    model = AppointmentBooking
    template_name = 'appointment_detail.html'
    form_class= AcceptAppointmentForm

    #method to get information regarding selected appointment.
    def get_context_data(self, *args, **kwargs):
        #gets all instances of that object.
        appointments = AppointmentBooking.objects.all()
        #gets all the relevant information regarding a particular appointment booking once selected.
        context = super(AppointmentDetailView, self).get_context_data(*args, **kwargs)
        context["appointments"] = appointments
        return context
```

Figure 58: implementation of a process to output appointment information for the staff member



salman filli

April 30, 2023 4 p.m.
Would like to discuss career guidance.

Email: salmanfilli123@hotmail.com
Phone Number : 07423067696

In Person Appointment

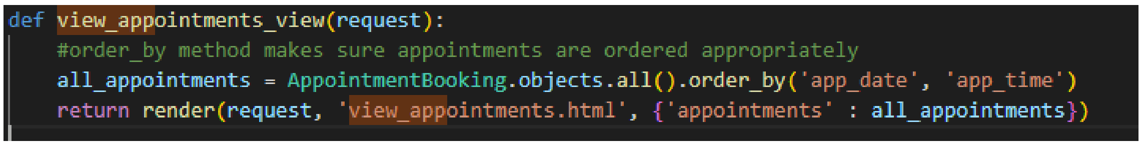
Key : 58707

link : None

This appointment is still awaiting acceptance

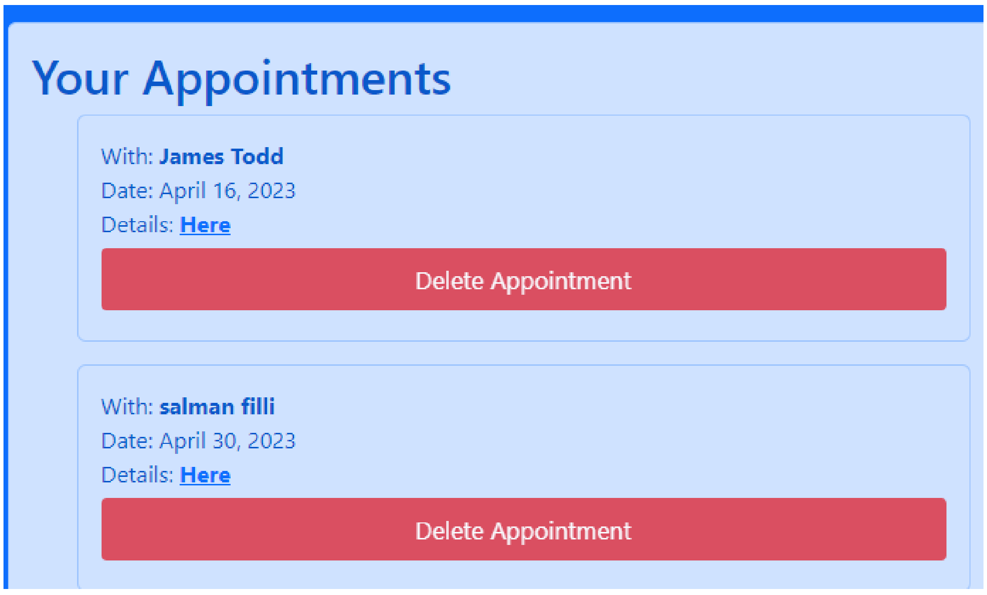
Figure 59: resulting implementation of outputting appointment details

For the appointments to be ordered appropriately, we use the order_by method (figure 60) in the view function with the date and time fields so the system knows how exactly we want to order all of the bookings. All the appointments are stored as a variable called all_appointments and are then passed onto the template and rendered.



```
def view_appointments_view(request):
    #order_by method makes sure appointments are ordered appropriately
    all_appointments = AppointmentBooking.objects.all().order_by('app_date', 'app_time')
    return render(request, 'view_appointments.html', {'appointments' : all_appointments})
```

Figure 60: Implementation of functionality for staff members to view appointments



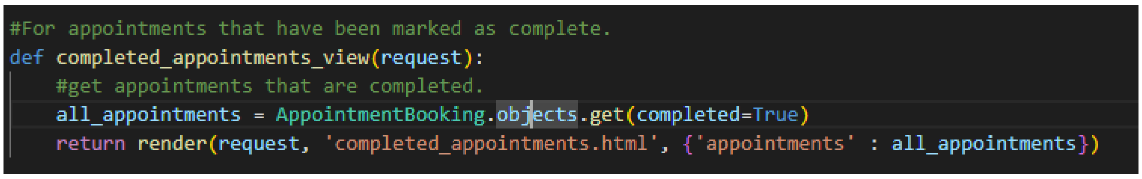
Your Appointments

With: **James Todd**
Date: April 16, 2023
Details: **Here**

With: **salman filli**
Date: April 30, 2023
Details: **Here**

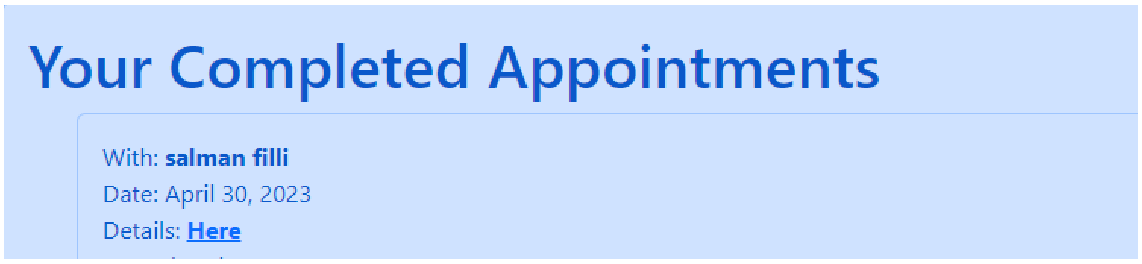
Figure 61: Result of staff member being able to view there appointments

Once the appointment is completed staff members can mark the appointment as complete and view completed appointments in selection section to track progress. To do this we use a similar method as above but we use a get() method to fetch appointments that have the complete field as true, which means the appointment has been completed.



```
#For appointments that have been marked as complete.
def completed_appointments_view(request):
    #get appointments that are completed.
    all_appointments = AppointmentBooking.objects.get(completed=True)
    return render(request, 'completed_appointments.html', {'appointments' : all_appointments})
```

Figure 62: Implementation of functionality to view all appointments that have been completed.

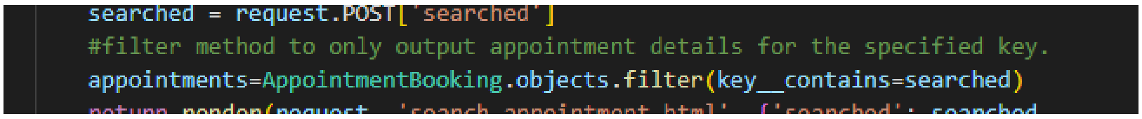


Your Completed Appointments

With: **salman filli**
Date: April 30, 2023
Details: **Here**

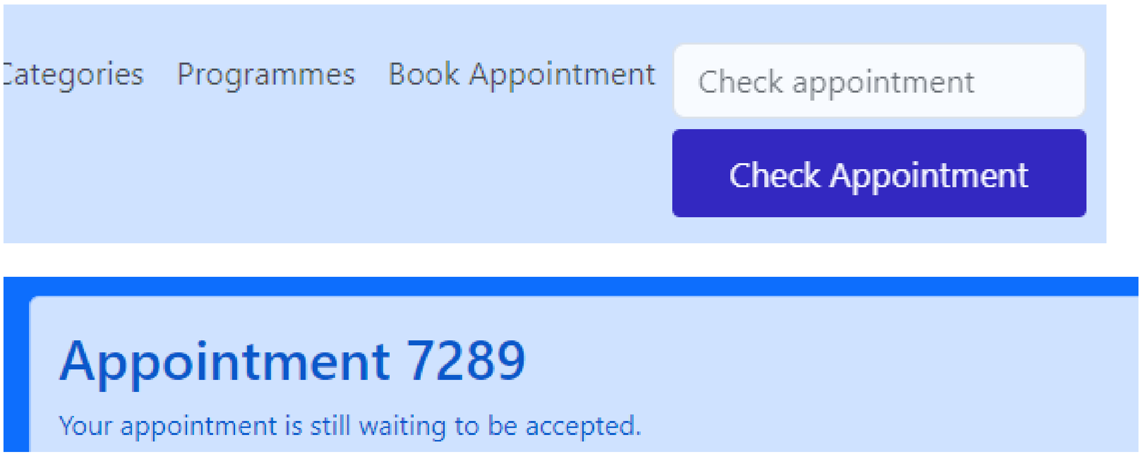
Figure 63: Implementation of completed appointments section

Users search for the status of their appointments by typing up the key given during booking and the system outputs the information for that particular appointment. We use the *filter* method and the *contains* attribute so we know what is being search matches with the key for that appointment (figure 64).



```
        searched = request.POST['searched']
        #filter method to only output appointment details for the specified key.
        appointments=AppointmentBooking.objects.filter(key__contains=searched)
        return render(request, 'search_appointment.html', {'searched': searched
```

Figure 64: Implementation of search functionality of appointments

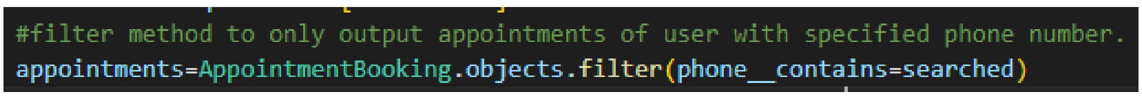


Appointment 7289

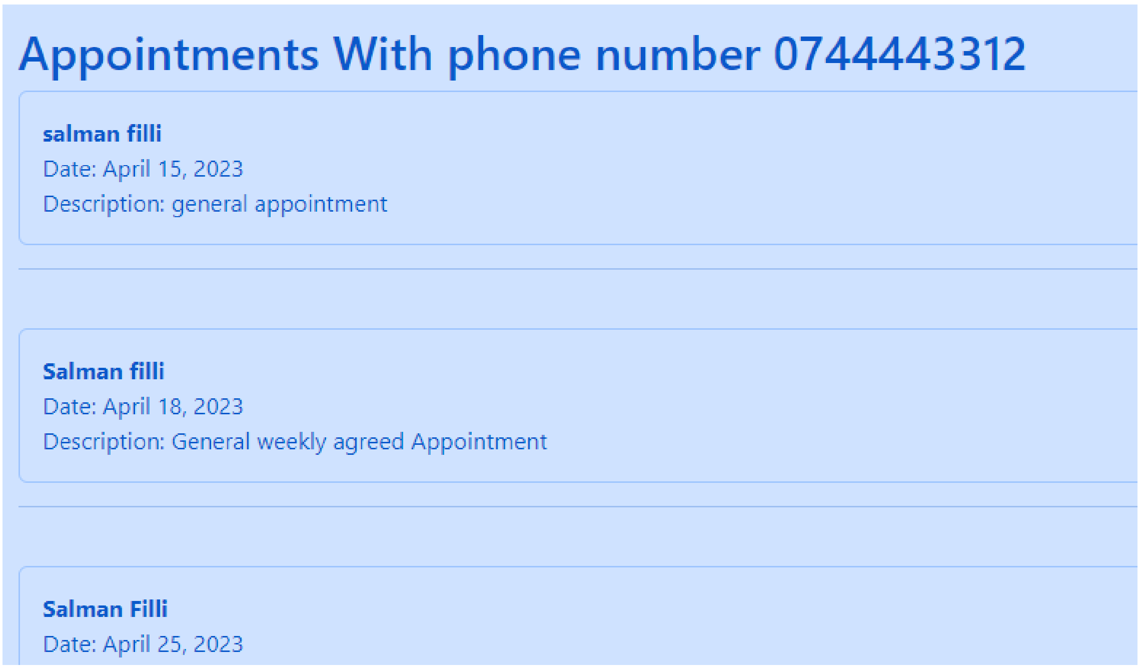
Your appointment is still waiting to be accepted.

Figure 65: Result of search functionality for appointments

This is replicated when users search for all their appointments by inputting their phone number. This is generally unique to one person so allows the user to view their own appointments.



```
#filter method to only output appointments of user with specified phone number.
appointments=AppointmentBooking.objects.filter(phone__contains=searched)
```



Appointments With phone number 0744443312

salman filli
Date: April 15, 2023
Description: general appointment

Salman filli
Date: April 18, 2023
Description: General weekly agreed Appointment

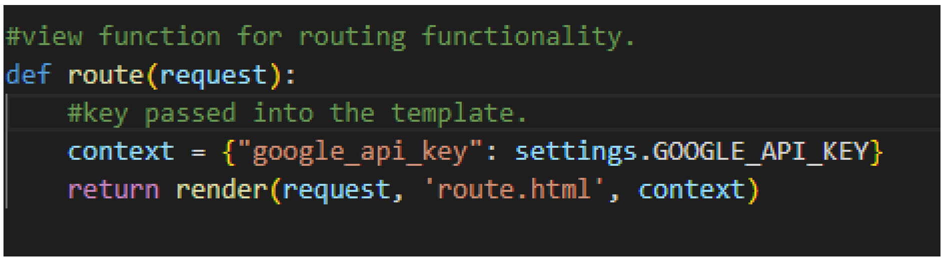
Salman Filli
Date: April 25, 2023

Figure 66: Functionality of searching for all of your previous appointments to track

Whether the staff chooses to accept/deny/rearrange an appointment, this information will be displayed to the user, and they can delete an appointment if they can no longer make the rearrangements.

4.3.7 Directions setup

A key aspect of this project is the ability to get directions and travel information to your appointment, including time to get there, the distance as well as each step by step directions. To facilitate this, we used the Google Maps API which stores the addresses and information throughout the country so we can output this information for the user. To do this, we are given a Google API key which is stored in the settings.py file. We create the view function for the routing section, which is where the user inputs the start and end destination, which takes the API key and passes it onto the template for rendering (figure 67).

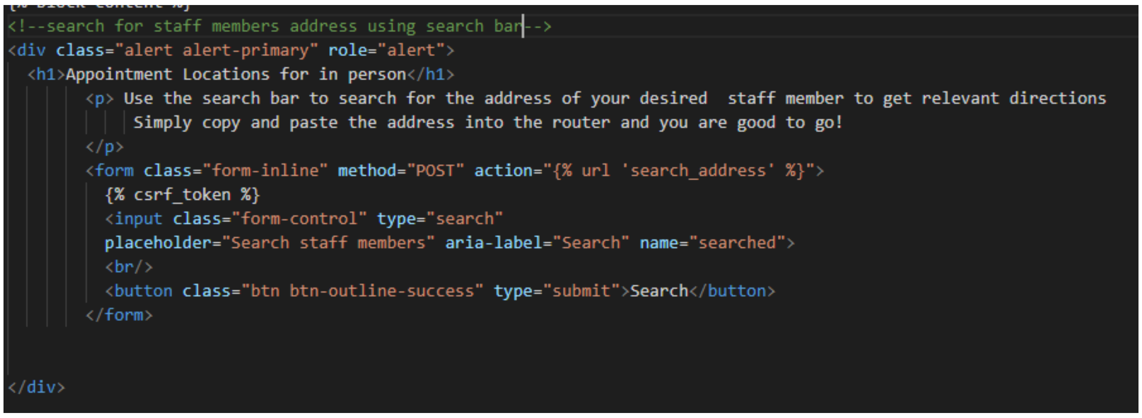


```
#view function for routing functionality.
def route(request):
    #key passed into the template.
    context = {"google_api_key": settings.GOOGLE_API_KEY}
    return render(request, 'route.html', context)
```

Figure 67: View function for routing functionality

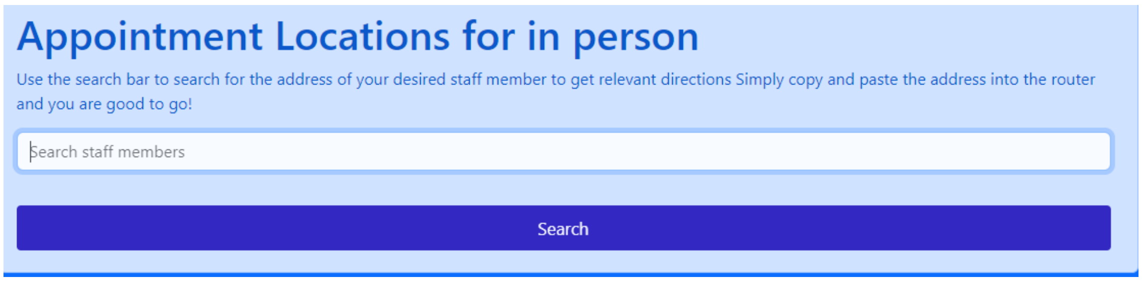
This is also used in the maps view function which is the view function used to output the results being the distance, directions etc.

In the template I also include a search functionality for users to search the address of a staff member by searching their name (figure 68).



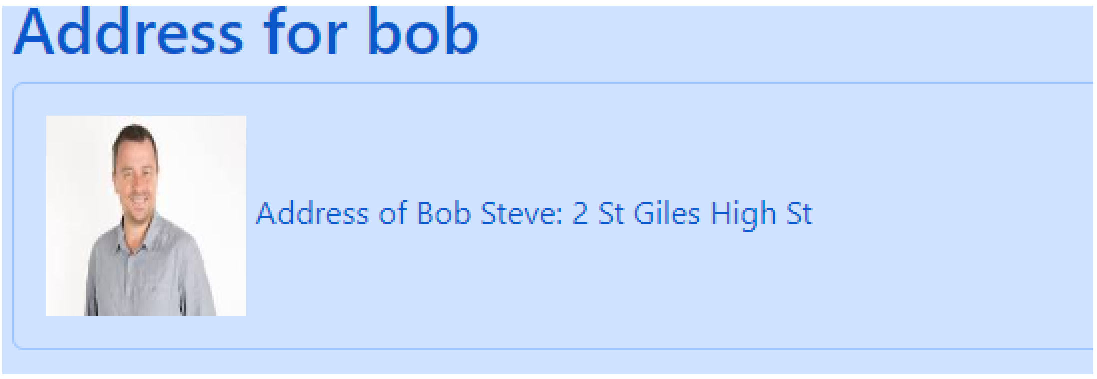
```
<!--search for staff members address using search bar-->
<div class="alert alert-primary" role="alert">
  <h1>Appointment Locations for in person</h1>
      <p> Use the search bar to search for the address of your desired  staff member to get relevant directions
           Simply copy and paste the address into the router and you are good to go!
      </p>
      <form class="form-inline" method="POST" action="{% url 'search_address' %}">
        {% csrf_token %}
        <input class="form-control" type="search"
        placeholder="Search staff members" aria-label="Search" name="searched">
        <br/>
        <button class="btn btn-outline-success" type="submit">Search</button>
      </form>


</div>
```



Appointment Locations for in person

Use the search bar to search for the address of your desired staff member to get relevant directions Simply copy and paste the address into the router and you are good to go!

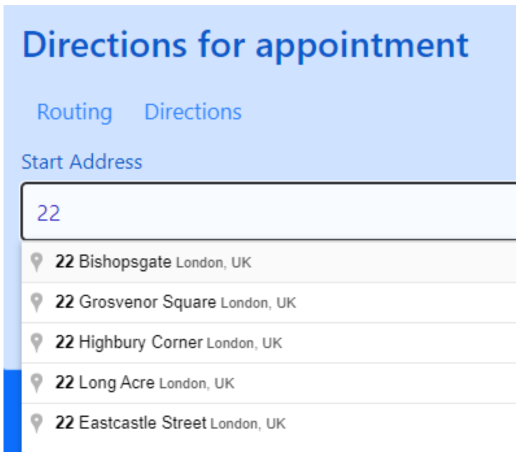


Address for bob

Address of Bob Steve: 2 St Giles High St

Figure 68: Implementation and result of outputting the address of staff member searched for

In this HTML template we also execute the code for google_places.js which will autocomplete what the user is typing with one of the addresses from the API that resembles it the most (figure 69).

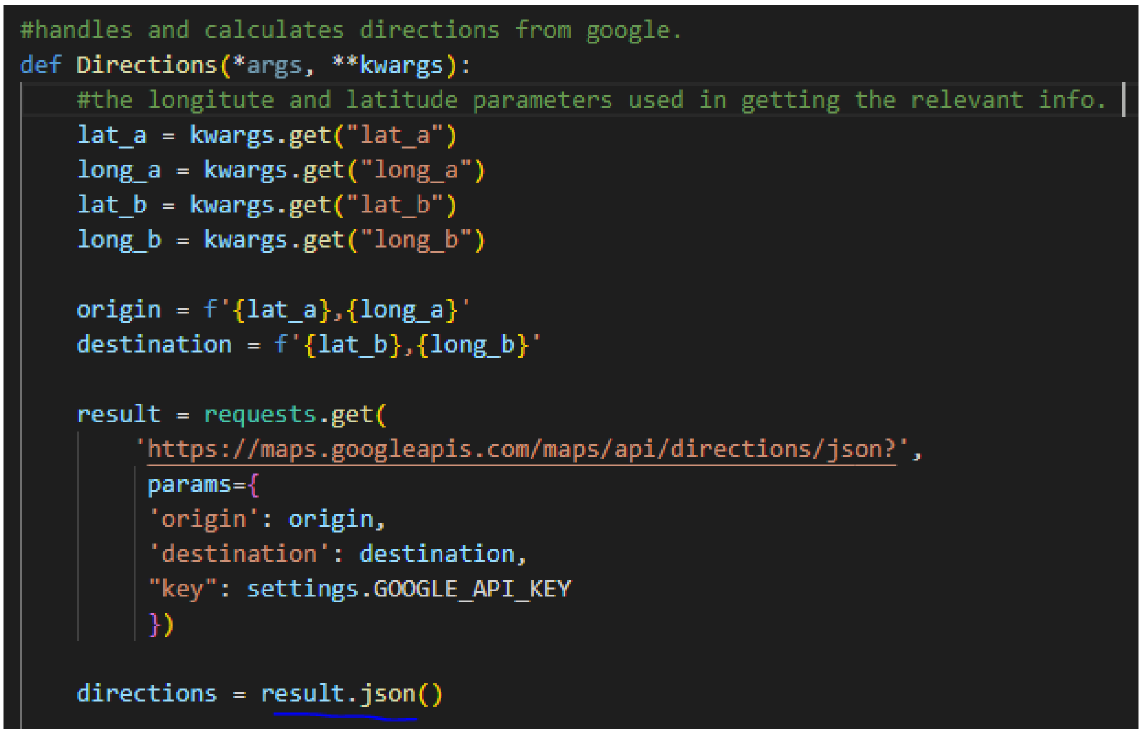


Directions for appointment

22 Bishopsgate London, UK
22 Grosvenor Square London, UK
22 Highbury Corner London, UK
22 Long Acre London, UK
22 Eastcastle Street London, UK

Figure 69: Implementation of auto completion of address for the routing

Once the start and destinations are given the travel information is displayed. We first create a directions method to make the relevant calculations and API calls (figure 70). The view function then uses this method along with the parameters to output the result using the relevant templates. (figure 71)



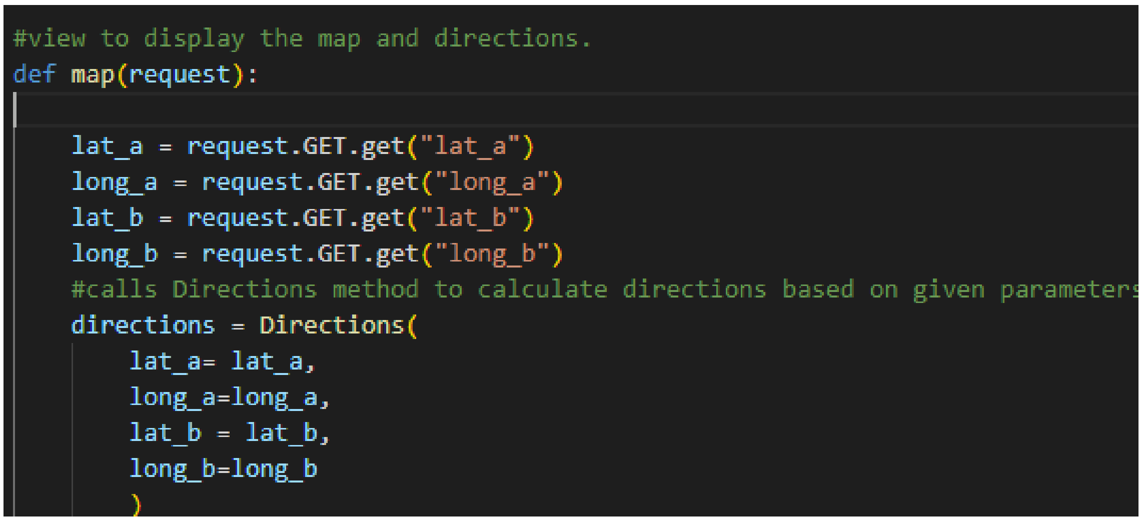
```
#handles and calculates directions from google.
def Directions(*args, **kwargs):
    #the longitute and latitude parameters used in getting the relevant info.
    lat_a = kwargs.get("lat_a")
    long_a = kwargs.get("long_a")
    lat_b = kwargs.get("lat_b")
    long_b = kwargs.get("long_b")

    origin = f'{lat_a},{long_a}'
    destination = f'{lat_b},{long_b}'

    result = requests.get(
        'https://maps.googleapis.com/maps/api/directions/json?',
         params={
         'origin': origin,
         'destination': destination,
         "key": settings.GOOGLE_API_KEY
         })

    directions = result.json()
```

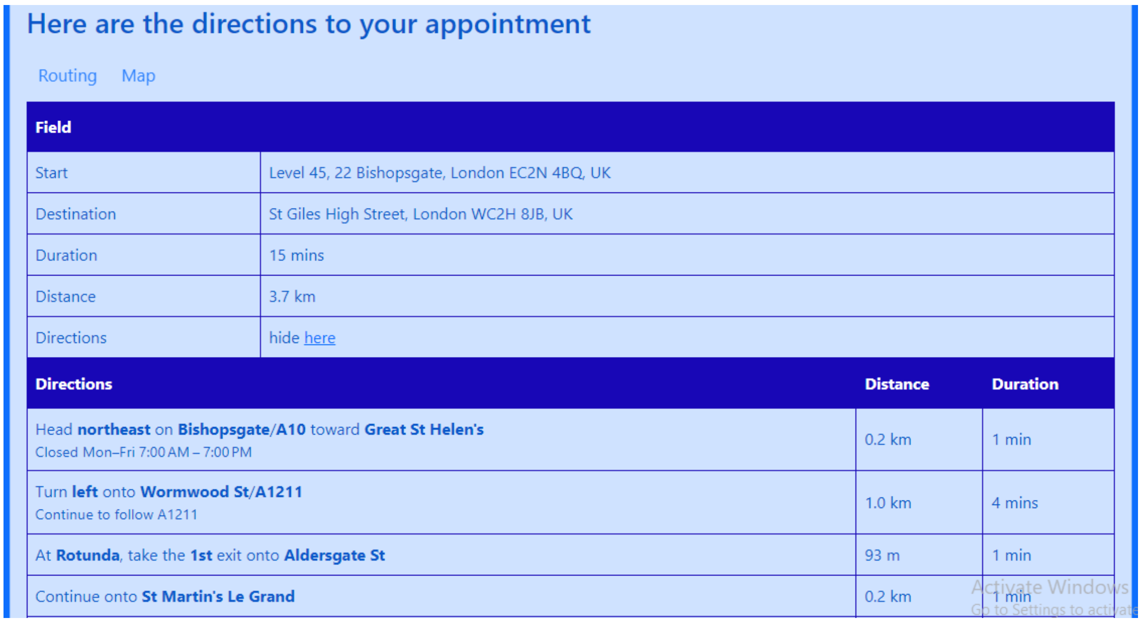
Figure 70: Functionality to fetch the directions based on user input of addresses



```
#view to display the map and directions.
def map(request):

    lat_a = request.GET.get("lat_a")
    long_a = request.GET.get("long_a")
    lat_b = request.GET.get("lat_b")
    long_b = request.GET.get("long_b")
    #calls Directions method to calculate directions based on given parameters
    directions = Directions(
        lat_a= lat_a,
        long_a=long_a,
        lat_b = lat_b,
        long_b=long_b
        )
```

Figure 71: view function to output the directions



Here are the directions to your appointment

Routing Map

Field	
Start	Level 45, 22 Bishopsgate, London EC2N 4BQ, UK
Destination	St Giles High Street, London WC2H 8JB, UK
Duration	15 mins
Distance	3.7 km
Directions	hide here

Directions	Distance	Duration
Head **northeast** on **Bishopsgate/A10** toward **Great St Helen's** Closed Mon–Fri 7:00 AM – 7:00 PM	0.2 km	1 min
Turn **left** onto **Wormwood St/A1211** Continue to follow A1211	1.0 km	4 mins
At **Rotunda**, take the **1st** exit onto **Aldersgate St**	93 m	1 min
Continue onto **St Martin's Le Grand**	0.2 km	1 min

Figure 72: Output of directions functionality

4.4 Summary

This chapter involved describing the implementation done in order to meet functional, non-functional requirements and use cases. The methodologies used to satisfy the requirements were explained as well as the step by step process of doing so. I described the technologies, libraries and API's that were used to help me implement this web application. Overall, the web application is fully functional and satisfies the use cases.

Chapter 5: **Validation and Testing**

Once the functionalities had been implemented, I carried out different types of testing to determine whether the system works as intended, and to evaluate areas of improvement.

5.1 Unit testing

Unit testing involved writing test cases on individual components. For this I wrote test cases for the urls which tests to see that the correct urls get resolved to the correct view function. I also wrote unit tests for the view functions, which tests the view gets opened with the correct response code (200=successful request). I also write test cases for the models which tests the creation of the database objects. For all of these tests I had to create mock data to simulate different scenarios to test how the program would react to being fed different pieces of data. This works well as you don't have to open the web app and manually create scenarios yourself so it saves a lot of time in the testing process. Doing unit tests has helped improve the quality of the code and made it easier to spot bugs.

5.2 Efficiency Testing

One of the main non-functional requirements was that the system should respond to user input and queries within a reasonable amount of time. A google study stated "a satisfactory load time as two seconds or less, and 53% of them will leave a mobile site if it takes more than 3 seconds to load." (Jakub Niechial 2017). In order to satisfy this requirement, I tested all the functionalities of the application and timed it accordingly. majority of the functionalities took less than 1 second to load and work. Button clicks were almost immediately responsive. The address matching functionality and the output of the directions took around 1 second to load. This is due to it involving an external call to an API to calculate the data for the directions. Therefore, Overall, the system is efficient.

5.3 User acceptance testing

I showed the application and its features to people with different backgrounds to test my application. The full scripts can be seen in Appendix B

From the feedback I was able to derive that the users were happy with the application and that there is room for improvement.

5.4 Non Functional Requirements Testing

As can be seen in figure below, I have described each non-functional requirement and how I was able to satisfy each one.

Non -functional requirement no.	How I met the functional Requirement

1	As seen on the web app, the system is very to use, by implementing common web app features such as navigation bars to navigate to different areas as well as search bars to search for whatever you are looking for.
2	Using Django forms, input was validated where required such as not logging in with the correct information, or invalid appointment booking. In these scenarios the user will not be able to progress to the next stage of usage.
3	See 5.1.4
4	Through many security features that can be used in Django. See figure 4.1.1 security In Django
5	Upon running the system, a numerous amount of times, it was available 100% of the time, with the only time the system being unavailable was when there was an error in the code which was fixed.

5.5 Summary

This chapter described the testing strategies used and how I was able to test my project to test that all use cases were implemented and all requirements were satisfied.

Chapter 6: **Conclusion/Evaluation**

This chapter describes the overall evaluation and outcome of this project. I discuss what I have achieved and learnt, as well as reflect on how well this project turned out, any problems I have faced my future plans.

6.1 Learnings and achievements

Overall, the development of the Social Welfare Helper web application has been a success. I developed all the functionalities within the constraints given to the best of my ability. As I stuck given deadlines, and balanced the development of the web app with other responsibilities, I believe this was a project success as well as a project management success.

I am grateful for the ability to apply knowledge learnt from the web programming module, which taught me the foundations of developing web apps. I was then able to use this foundation to do further independent research to enhance my skillset in this field to develop my project. I also believe that the Project Risk management module played a big role to the success of this project as it gave me the ability to plan and consider different aspects such as feasibility, risk etc so it gave me the tools needed to manage the delivery of the project successfully. The software development and software engineering module gave me the tools to carry out factors such as requirement gathering and the design of the class diagrams, database diagrams, flow diagrams etc.

Using the MVT pattern, made the application more loosely coupled, meaning the components of the application were relatively independent and changes in one aspect didn’t affect others. This made it easier to fix any bugs and made the overall application code easier to read, organize, and improve.

6.2 Challenges that I faced

One of the challenges faced was lack of knowledge on the front end side of web development. I didn’t know how to utilize HTML templates to combine it with the Django backend. So this involved having to do some research to learn the fundamentals of this technology.

Something I had hoped to implement was the output of an actual visible map that shows the directions. This was something I could implement using the API but had trouble using the templates to output the map so was only able to output the directions in written format which is still beneficial to the user.

Another challenge faced figuring out how to make the application as easy to use as possible whilst still delivering value. For e.g. When it came to appointments I was originally going to use email for the appointment functionality, meaning users would receive their appointment acceptance from the staff member by email. But after realizing that this would make things harder for the user as they would have to switch devices/screens to access email, I decided to remove this and simply include a key identifier of their appointment so they can view the status of their appointment on the app itself.

Another challenge faced was approaching the storage of user information. Users don't giving away information without necessity as it has been stated that 'users will weigh the effort it takes to complete your form against the value they'll get from using your site. If the value they get is little, they'll pass on your form.' (Anthony, 2012). This means it was imperative that the user wasn't deterred from using the application due to unnecessary requests to hold their data. This is why if it is a regular user using the application and not a staff member they don't have to create an account to view posts or find programes. I only include data collection during appointment booking as the staff member needs to know about the appointee to make the appointments as helpful as possible. It also only asks the user regarding their location when it comes the directions functionality. This is necessary as without this the system cannot plan a route for them to their appointment.

6.3 What would I add or do differently in the future?

- **Background Check:** when staff members sign up to the application, I think it would have been good to facilitate some kind of automated background checks to determine if they are qualified to use the application and conduct appointments.
- **Design:** Although the user interface is clear and easy to use. I would have invested more time in making it look more professional as I spent the bulk of my time making sure that the project satisfies the requirements given.
- **Recommendation enhancement:** Users are currently able to find different sorts of programmes in different sectors or search whatever kind of programme they are looking for. However, if I had more time I would have added onto this functionality by asking them a few questions and using this give more personalised recommendations on programmes.

6.4 Summary

This section has been a reflection on how I felt about this project and have evaluated how I believe it went. Overall I learnt a lot about the various stages of web application development in this project and look forward to using these newly learnt skills in future projects and endeavors.

References

Brendan McGuidan. (2022). How big is the internet? [online] Available from https://www.easytechjunkie.com/how-big-is-the-internet.html

Barrett Holmes Pitner (2016). The struggle against Internet overload is real [online] Available from https://www.dailydot.com/unclick/internet-overload-the-struggle-againstis-real/

University of Birmingham. (2013) Breaking the cycle? Prison and visitation and Recidivism in the UK: [online]. Available from https://gtr.ukri.org/projects?ref=ES%2FK002023%2F1#/tabOverview

Monika Grierson, Delvin Varghese, Mitzi Bolton, Patrick Olivier (2022) Design Considerations for a digital service to support prison leavers. [online] Available from https://dl.acm.org/doi/10.1145/3532106.3533567

Ministry of Justice (2021) Prison Population projects 2021 to 2026, England and Wales. [online] Available from https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/1035682/Prison_Population_Projections_2021_to_2026.pdf

Justice Committee (2019) Prison Population 2022: Planning for the future- report Overview. [online] Available from: https://publications.parliament.uk/pa/cm201719/cmselect/cmjust/483/reportoverview.html#heading-3

Innovative Criminal Justice Solutions (2022) Digital skills – a path to improve offender social (re) integration [online] Available from: https://prisonsystems.eu/digital-skills-apath-to-improve-offender-social-reintegration/

Sophia Waterfield (2022) How digital skills can help ex-offenders stay out of prison. [online] Available from https://techmonitor.ai/leadership/workforce/ex-offenders-digitalskills-prison

Google Play. (2022). NHS App [online]. Available from: https://play.google.com/store/apps/details?id=com.nhs.online.nhsonline&hl=en_GB&gl=US&pli=1

NHS. (2021). Milestone hit with over 16 million NHS App users. [online]. Available from https://transform.england.nhs.uk/news/milestone-hit-with-over-16-million-nhs-appusers/

Teemu Rantanen, Eeva Järveläinen and Teppo Leppälahti (2021). Prisoners as Users of Digital Health Care and Social Welfare Services: A Finnish Attitude Survey. Chapter 3.3. [online]. Available from https://www.mdpi.com/1660-4601/18/11/5528

Adam Blacker (2022) Job search apps have more engagement than ever, remote work of high interest [online]. Available from https://blog.apptopia.com/jobsearch-app-engagement-high-alongside-remote-work

Russell Webster (2017) Service User Launches New Prison Family Support App [online] Available from https://www.russellwebster.com/service-user-prison-familysupport-app/

Katie M. Rhode (2010) The Effect of Career Counselling on the Self-Efficacy and Career Maturity of Residential Juvenile Offenders. Chapter 4 Submitted to the Graduate Faculty of Texas Tech University in Partial Fulfilment of the Requirements for the Degree of DOCTOR OF PHILOSOPHY [online] Available from https://ttuir.tdl.org/bitstream/handle/2346/ETD-TTU-2010-12-1067/RHODEDISSERTATION.pdf?sequence=2&isAllowed=y

App store (2023) [online] Available from https://apps.apple.com/gb/app/nhs-app/id1388411277

Talkspace (2023) [online] Available from https://www.talkspace.com/#how

App Store JOBTODAY (2023) [online] Available from https://apps.apple.com/us/app/job-today-easy-job-search/id981163277

Django (2023) [online] Available from https://www.djangoproject.com/start/overview/

InterviewBit (2022) [online] Available from: https://www.interviewbit.com/blog/django-architecture/

Shubham (2023) [online] Available from: https://www.educative.io/answers/what-is-mvt-structure-in-django

Wikipedia Object-relational mapping (2023) [online] Available from: https://en.wikipedia.org/wiki/Object%E2%80%93relational_mapping

Daragh Tuama (2023) [online] Available from: https://codeinstitute.net/blog/web-app-vs-mobile-app/

Django Security (2023) [online] Available from: https://docs.djangoproject.com/en/4.1/topics/security/

Brian Myers (2021) https://www.stackhawk.com/blog/what-is-cross-site-request-forgery-csrf/

StackHawk (2021) [online] Available from: https://www.stackhawk.com/blog/sql-injection-prevention-django/#what-is-sql-injection

Django Databases (2023) [online] Available from: https://docs.djangoproject.com/en/4.1/ref/databases/

Django Models and Databases (2023) [online] Available from: https://docs.djangoproject.com/en/4.2/topics/db/

HTML Templates (2023) [online] Available from: https://docs.djangoproject.com/en/4.1/topics/templates/

Google Maps Platform (2023) [online] Available from: https://developers.google.com/maps/faq#whatis

Django apps (2023) [online] Available from: https://docs.djangoproject.com/en/4.1/intro/tutorial01/

Jakub Niechial (2017) [online] Available from: https://www.netguru.com/blog/11-reasons-why-your-web-app-is-slow#:~:text=According%20to%20a%20Google%20study,factors%20for%20Google%20mobile%20searches.

Anthony (2012) [online] Available from: https://uxmovement.com/forms/8-reasons-users-arent-filling-out-your-sign-up-form/

Appendix A – Use case Diagrams

Use Case	View Directions and direction details to appointment
Actor/s	User
Pre-Condition	Appointment must have been booked and accepted by the staff member. The appointment must also be an in-person appointment.
Post-Condition	User has viewed the directions as well as the distance, and time it should take to get to the appointment.
Basic Path	1. User clicks on book appointment page 2. They then input the staff member to book the appointment with 3. The user then checks if their appointment has been accepted. 4. If the appointment is accepted they are given the location of the appointment as well as well as a link to get the directions 5. The user inputs their starting point and the location of the appointment 6. The system then displays the distance, and the time to the appointment as well as detailed directions of how to get there.
Alternative path	At step 5 the user may input an invalid address which may cause an error which should be displayed accordingly.

Use Case	Create/post Programme
Actor/s	Staff Member
Pre-Condition	Staff Member must be logged in
Post-condition	The programme should be displayed for users to see and access and stored in the database
Basic Path	1. Staff member clicks on the ‘add programme’ section on the home dashboard 2. They then input the details regarding the programme including the category, description, link to access the program etc. 3. They then post the programme which allows it to be saved. 4. The system then displays this programme on the section of the category that the staff member has posted it as. For e.g. if it is a career related programme, it should be displayed in

	the careers section. Or if the user search matches what they have posted that programme should come up.
Alternative path	At step 2, if the user has skipped inputting any of the required information, the system should return some sort of message or prompt for them to go back and input the necessary information.

Use Case(s)	View/Reject/Rearrange/Accept appointment
Actor/s	Staff Member
Pre-Condition	- Staff Member must be logged in - User must have booked an appointment for them to view
Post-condition	- The Appointment should be displayed - Staff should be able to reject, rearrange or accept appointment
Basic Path	1. Staff member logs in and clicks on the view appointments tab. They can now view the appointments. 2. They then click on the button that says “accept appointment”, “reject appointment” or “rearrange appointment”. 3. They then fill in the details needed for that section e.g. for decline they input the message as to why they have declined. 4. The system then updates the fields of the appointmentBooking model based on the staff members action.
Alternative path	At step 3, if the user has skipped inputting any of the required information, the system should return some validation or prompt for them to go back and input the necessary information.

Appendix B - User Feedback scripts

6.4.1 Script 1

Me: “Can you please give some feedback on the application?”

Test user: “Nice application, well put together. I like that there are different categories of programmes to view and users can search for different programmes. I particularly think the appointment booking feature is very beneficial.”

Me: “Thanks for your feedback”

6.4.2 Script 2

Me: “Can you please give some feedback on the application?”

Test User 2: “Great application! Everything is organised accordingly and I particularly like the directions functionality. Would have been great if you there was a visible map showing the directions though!”

Me: “Thanks for your feedback!”

6.4.3 Script 3

Me: “Can you please give some feedback on the application?”

“Although the design could have been improved aesthetically, the app is very easy to use particularly for the target demographic and the functionalities are very beneficial Everything works accordingly.”

Me: “Thanks for your feedback!”

Appendix C – Project milestone plan

Task	Expected Start Date	Expected End Date	Time/Deadline
Research project ideas and review existing ideas	26th September 2022	7rd October 2022	10 days
Write project definition and planning	7th October 2022	17th October 2022	10 days
Project definition Submission	**-**	**-**	**17th October 2022**
Research different aspects of social welfare and functionalities to include in my application.	18/10/2022	29/11/2022	2 Weeks
Write the Interim report including diagrams, risk assessment, requirements etc	02/10/2022	29/11/2022	8 Weeks
Learn and Improve on my Web app related coding ability. (ongoing and parallel to other tasks)	18/10/2022	06/12/2022	7 Weeks
Interim Report Submission	**-**	**-**	**29/11/2022**
Progress Slide Submission	**-**	**-**	**06/12/2022**
Progress Presentation	**-**	**-**	**17/12/2022**
Full implementation of the application	01/01/2023	06/03/2022	9 Weeks
Draft write up of final report	07/03/2023	21/03/2023	2 Weeks
Draft Report	**-**	**-**	**21/03/2022**
Complete necessary testing, make code improvements and fix any remaining bugs	21/03/2023	20/04/2023	4 Weeks

Complete Final Report, video and presentation	21/03/2023	29/04/2023	5 Weeks (parallel to task above)
Final Report Submission	**-**	**-**	**04/05/2022**
Project Video Submission	**-**	**-**	**06/05/2022**
Viva	**-**	**-**	**09/05/2022**

Appendix D: Risk Assessment

Description of Risk	Description of Impact	Likelihood Rating	Impact Rating	Preventative action
Users would be uncomfortable with sharing personal information	Negates the point of the application as the point was we staff members can't support users if they don't know who they are.	Medium	High	Assure the user that their information won't be shared publicly and design the system so that it is unable to do so anyway, allowing their information to be secure. Also only ask information from users when it's absolutely necessary like for appointment booking, directions etc.
Users being given wrong advice regarding programmes and support	Users would be misled which could affect their lives and would no longer want to use the application	Low	High	Make sure that staff members are able to update information that they share or remove anything that is no longer necessary to keep up to date.
Loss of user information	The user may have to input their information again taking up time and reducing user satisfaction.	Low	high	Make sure there is a correct data infrastructure and security so that this does not occur.
Delay in project delivery due to other commitments/ circumstances	This may cause the deadline to not be met or the applications quality to be reduced in order to stick to the deadline.	Medium	Medium	Make sure to manage time and workload accordingly and avoid procrastination. This way you have time to solve any errors that come up later on in the project development.
Poor quality code	This will cause the application to have bugs and logical errors which can affect the functionality of the application and therefore the quality	Medium	High	Frequently test and debug code, resolve bugs and logical errors as soon as they are found, make sure design before implementation is up to a high standard and use best coding practises.
Scope Creep	Can cause deadlines to get surpassed as well as extended project timeframes. It can	Low	High	Break down the project into manageable sections and frequently review the scope. Any scope changes should be documented and the project must

	also add extra complexity to the project			be rescheduled in a way that it will still be able to fulfil this scope change.
Loss of code/project files and reports	I would no longer have a project to submit and would have to restart development which will take up time and could lead me to miss deadlines.	Medium	High	Keep more than one copy of everything and always makes sure it is backed up accordingly.
Inability to implement certain features correctly	Reduces the quality of the project if not developed, can cause delays in delivery which affects sticking to deadlines.	Medium	Medium	Make sure to begin develops early to account for extra added on by these issues. Also make sure to do substantial research before developing a feature so you will know how to approach it.

Appendix E: Social Welfare Helper Unit test

The table below shows some of the test cases written as well as their results

Feature	Test case	Unit test function	Expected output	Result
Model	Test valid creation of staff model	`test_staff(self)`	Boolean of True, as the values tested are equal to the value that were used to create the staff member.	Success
	Test valid creation of Category model	`test_category(self)`	Boolean of True as the values tested are equal to the value used in the creation of the category	Success
	Test valid creation of appointment booking	`test_appointment_booking(self)`	Boolean of True as the values tested are equal to the value used in the creation of the category	Success
Views	Test valid template being used for ‘home’ view function and successful	`test_home(self)`	Boolean of True as the request will be successful due to the 200 response	Success

	request of view function		code and the template being tested should be valid.	
	Test valid template being used for ‘Programmes page’ view and successful request of view function	`test_ProgrammesPage(self)`	Boolean of True as the request will be successful due to the 200 response code and the template being tested should be valid.	Success
	Test valid template being used for ‘book appointments’ view and successful request of view function	`test_book_appointment(self)`	Boolean of True as the request will be successful due to the 200 response code and the template being tested should be valid.	Success
URLS	Test URL for programme detail page being successfully resolved	`test_programme_detail_url_is_resolved(self)`	Returns true as the view function of the url equals the view function inputted as part of the test case	Success

	Test URL for the add post page is successfully resolved	`test_add_post_url_is_resolved(self)`	Returns true as the view function of the url equals the view function inputted as part of the test case	Success
	Test URL for the booking appointments page	`test_bookappointment_url_is_resolved(self)`	Returns true as the view function of the url equals the view function inputted as part of the test case	Success