Individual Report

1) A part of our code that I wrote and feel proud of is the login system.

```
def login_user(request):
    if request.method == "POST":
        #username and password is taken as input
        username = request.POST["username"]
        password = request.POST["password"]
        #checked to see if it exists in the database
        user = authenticate(request, username=username, password=password)
        #if user exists then redirect them to the staff page
        if user is not None:
            login(request, user)
            return redirect('staffPick')

        #if user doesn't exist redirect them back to login page with error message
        else:
            messages.success(request, ("Error logging in"))
            return redirect('login')

    else:
        return render(request, 'login.html', {})

def logout(request):
    #if logout button pressed user redirected to login screen
    return render(request, 'logout.html')
```

The way this piece of code works is that it takes the username and password input from the user when they insert it into the website and stores it into their respective variable names. The authenticate method is then used to see whether the user exists or not inside of the database and the user is then stored in the variable ‘user’. An if statement is then used to check that the user isn’t None (doesn’t exist) where they’ll be redirected to the correct URL ‘staffPick’. Else an error message is output, and the user gets redirected to the login page again. The else statement on the outer loop is for any invalid inputs the user enters into the form when they type their details, if invalid they will be taken back to the login page. I also made a logout function which is linked to a button on the webpage where, if clicked, it will redirect the user to the login page. My code contributed to the success of the project as it provided functionality to the final project which covered an important user story being the staff should be authenticated when they enter the site.

		Complexity	Function Point Value
EQ	Waiter viewing current orders	H	7
ILF	Client database	A	8
ILF	Order_Status database	A	3
ILF	Order database	A	4
EI	User registration input	A	3
EO	Manager viewing total spent	H	4

UFP is 7 + 8 + 3 + 4 + 3 + 4 = 29

3) What I have learnt about developing software as a group using Scrum is how important frequent communication between the members is. I acted as the Scrum master over the entirety of the project, so it was my job to organize meetings, discuss what others were doing and to also set work for members to do. If I had not hosted frequent Scrum meetings, I could say for certain that we would have been very lost on where to take our work, it was only through these meetings that I found out where members were struggling on parts of the project that they needed help on. Communicating ensured that our team complied with the Codes of Conduct and Practice, an example of this would be how members were able to voice their concerns and opinions on how we could go about the project which led to valuable contributions that improved the overall quality of our final product, this falls under section 2e of the code of conduct. By getting frequent updates from each person on the team and having them showcase their work I was also able to reassign them to different parts of the project where I felt as though they operated more comfortably and could produce work more efficiently. An example of this was after the first sprint, having seen what work was produced by the back-end team, I had to reshuffle some of them over to the front end which ended well.

4) Robust code played an integral role in the development of our program as it allowed us to make our code more reliable which led to it being able to account for numerous errors and circumstances. Having code that adapts to misinputs or errors made the final product a lot more reliable and cleaner, if we had not coded in this way team members would have encountered difficulties testing code that others have made. An example of this was with the development of the staff login system, I had to create the code in a way that accounted

for the user not existing, making the username and password case sensitive and redirecting the user when they either successfully logged in or didn't. Without this if a user tried to login but they didn't exist in the database then the page would just crash which isn't useful to anyone. Implementing robust coding encouraged team members to thoroughly test their code to see what ways the user could possibly break it and I am content with the final product regarding how it handles error handling and the number of errors it can pick up on.