[**6CC552: Research Project**](https://udol.derby.ac.uk/webapps/portal/frameset.jsp?tab_tab_group_id=_3_1&url=%2Fwebapps%2Fblackboard%2Fexecute%2Flauncher%3Ftype%3DCourse%26id%3D_62269_1%26url%3D)

**CW2- Shift Roster**

**University of Derby**

**by Qamar Ishaque Chaudhry**

**Introduction**

**1.1 Purpose**

This project aims to develop an intelligent, collaborative, customisable and easy to use shift planning application. It will enable organisations creating, publishing, and editing schedules all in one place. The intended users are UK companies. The application will eliminate the use of manual rosters hung over notice boards. The employees would be able to check their rosters via internet.

**1.2 Project Scope**

The scope of the project is to develop an application which enable employees to receive their roster for week, fortnight or month depending on which way the customer operates their rostering system.

**1.3 Objectives**

Main features of the project are:

* Register/Login – Employers should be able to register and login to the system to create rosters for their employees to use.
* Add/Edit/Remove Employees – Allow the manager/admin to easily edit their employees on the UI so that managers can manually add/remove or alter shifts.
* A shift roaster may cater for full time and part time employees, time off taken out due to social obligations, scheduled working hours must ensure work life balance, payroll should be visible for both managers and employees.
* Each shift in a schedule can be viewed weekly, bi-weekly, monthly moved with drag and drop ease and employees may be grouped with varied experience range and skill sets if needed.

**1.4 Technologies**

To develop this application ASP.NET Core has been chosen as it is deployable on cross platforms (Windows, Linux, MacOS). Applications developed on ASP.NET Core may also be deployed even on cloud giving it more flexibility. Visual Studio 2017, Entity Framework Core,C#.Net has been used in creating this application on ASP.NET Core.

**2** **Requirements**

**2.1.1 Project Plan**

* Understanding requirements and class design.
* Creating Solution and adding projects it.
* Creating the business objects.
* Creating Data access layer in EF. In EF Core only the code first approach is used to create classes for the application which are called business entities or business objects.
* Creating Business logic layer in C#.Net.
* Creating Presentation logic layer in Core MVC.
* Designing models, controllers and views.
* Business rules validation.
* Securing your app.
* Feedback from project supervisor.

**2.1.2** **Functional requirements**

**Management Requirements**

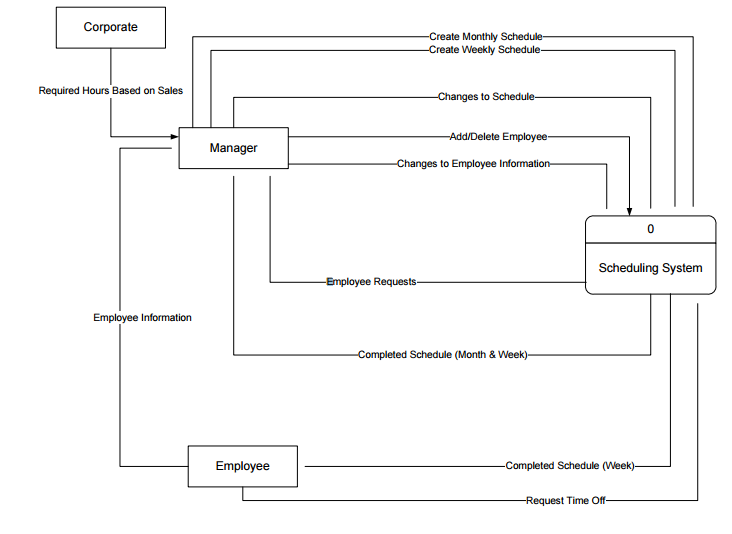
* The system will allow managers to create a weekly schedule
* The system will allow managers to create a monthly schedule
* The system will allow managers to input employee information (i.e. name, phone, email, hours per week, salary)
* Managers will have the ability to edit employee details
* Managers will be able to remove employees that have left the company
* The manager can upload an excel file with employee shifts to generate a roster on the UI
* The manager can add shifts to the roster manually
* The system will allow managers to adjust shifts on the rosters
* The system will allow shifts to be manually removed from the roster
* Managers will receive shift swap requests from employees that have been entered through the system

**Employee Requirements**

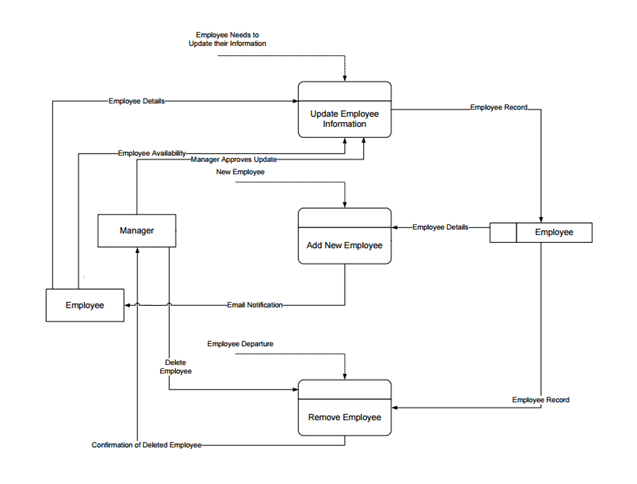
* The system will allow employees to check their roster for the upcoming week/month
* The system will allow employees to request a shift swap with a colleague
* The system will allow employees to view other employees details i.e. email and phone number in case they need to get in contact with them

**2.1.2 Data Flow Diagrams**

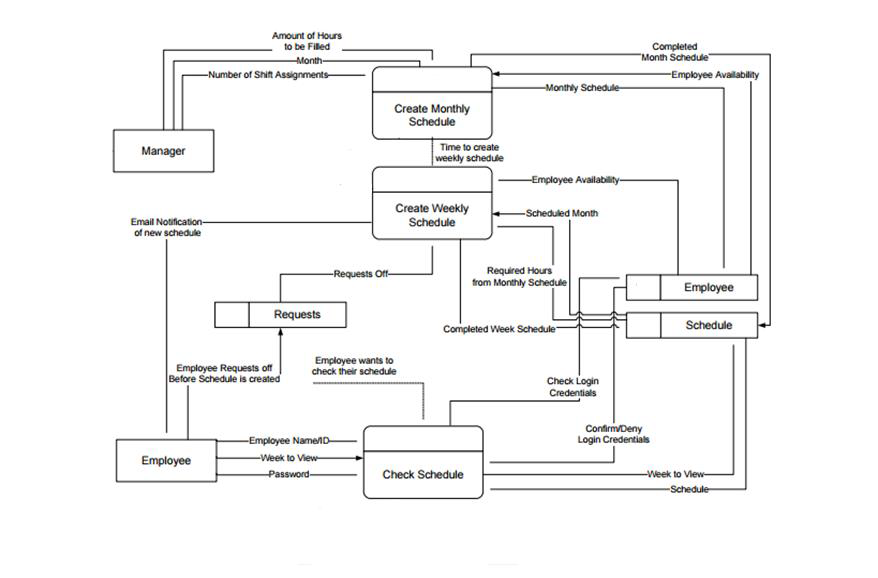
**Context Diagram**



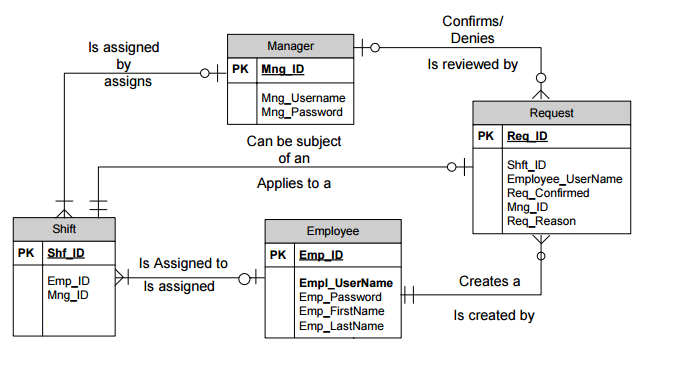
**Data Flow Diagrams Start Phase**



**Data Flow Diagrams Final Phase**



**Entity Relationship Diagram**

**2.1.3 Database requirements**

A database will need to be created and connected to the application through SQL Server as the application will be developed in ASP. Net Core. The databases created in SQL Server 2016 encrypt the data for higher security within the system hence the data would be protected. It would have the following entities:

**Employee**

The Employee entity is uniquely identified by an employee ID. The Employee entity contains important information about an employee such as EmpId ,Username, and Password. Employment Date (In order to cater for mix experienced shift as per business logic), Job Title, Gender, First Name, Last Name, Email, Phone.

**Request**

The Request entity is uniquely identified by a request ID. It needs to keep track of the Shift ID and the Employee who wants a shift swap. It also needs to keep track of who approved the request and whether it was approved. Only managers can approve a request.

**Manager**

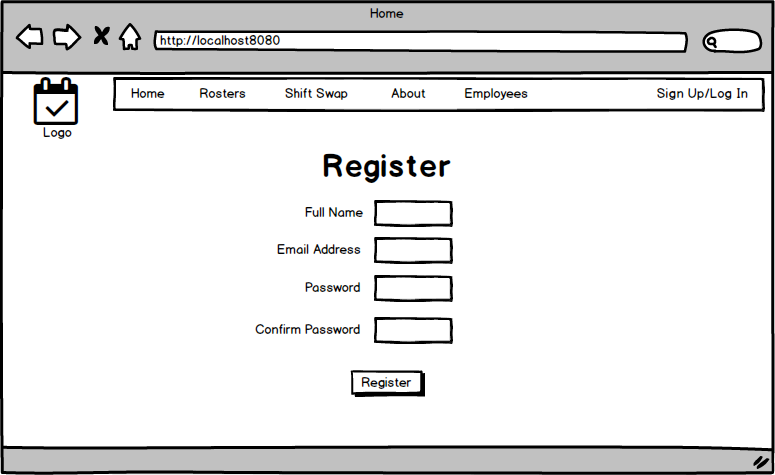
The Manager entity is uniquely identified by a manager ID. It also contains information about a user name and password to log into a system. Their names are kept track of for the user side. Managers generate rosters and review shift swap requests.

**Shift**

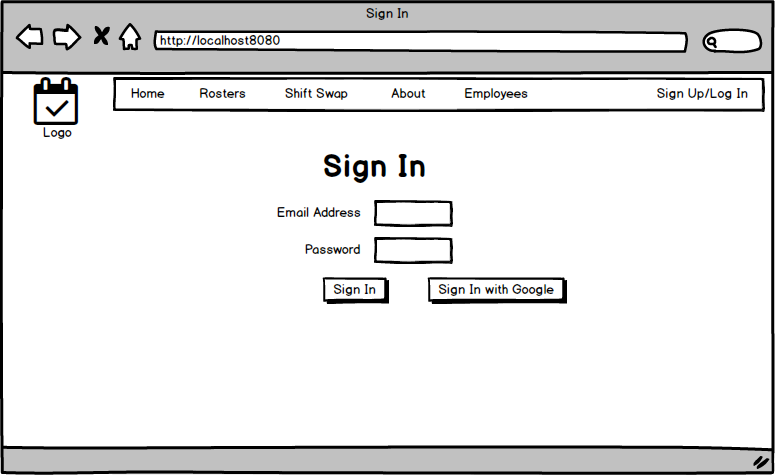
The Shift entity is uniquely identified by a shift ID. The entity needs to keep track of the manager who created it as well as the employee who will fill it. Each shift is part of the overall schedule. It contains Shift ID, Title, Request (best practice is to use some code like F407 for absence due to child sickness), Start At, End At, Location, Payrate, Hours.

**3. Presentation Layer-GUI**

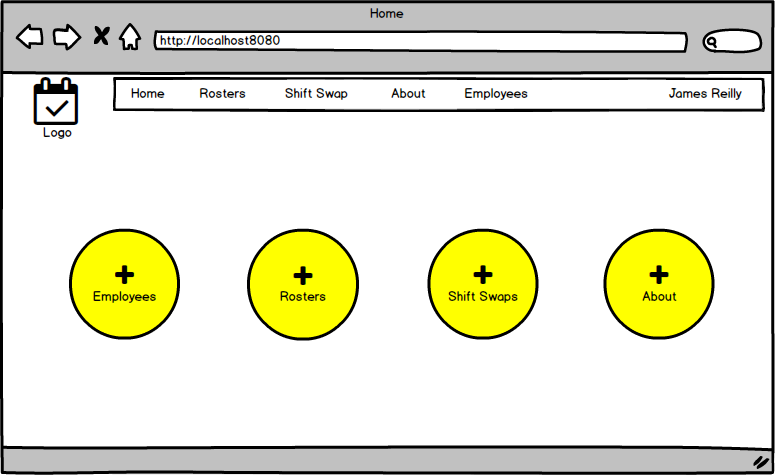
**3.1.1 Register & Login**



This is the register screen where users i.e. the administrator/manager will register employees employees to be added to the system in order for them to access their rosters.

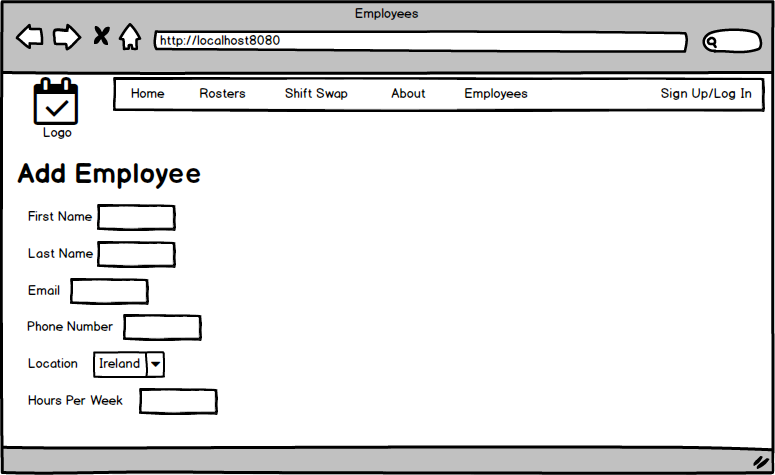


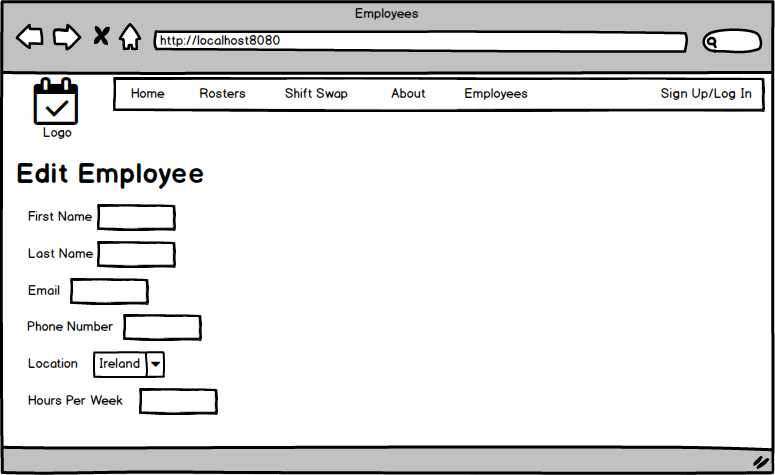
Once users are registered, and their details are saved to the database they will then be directed to the login page when they click the Sign Up/Log In link in the top right corner of the page. Users will then input their information to access the system. System Admins will have more rights than normal employees i.e being able to add an employee to the system.

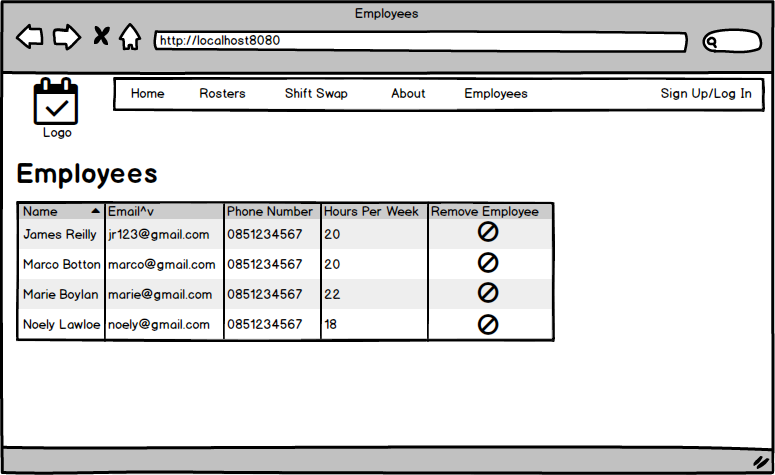


This is the home page where users will be able to access the available sections of the system such as employees, rosters, shift swap and the about page. Employees/users who do not have admin access will be able to access the whole system except for the employees page as they are not permitted to add or remove employees from the system. The system admin is the only person who can access the employees database and make the appropriate changes.

**3.1.2 Employees**

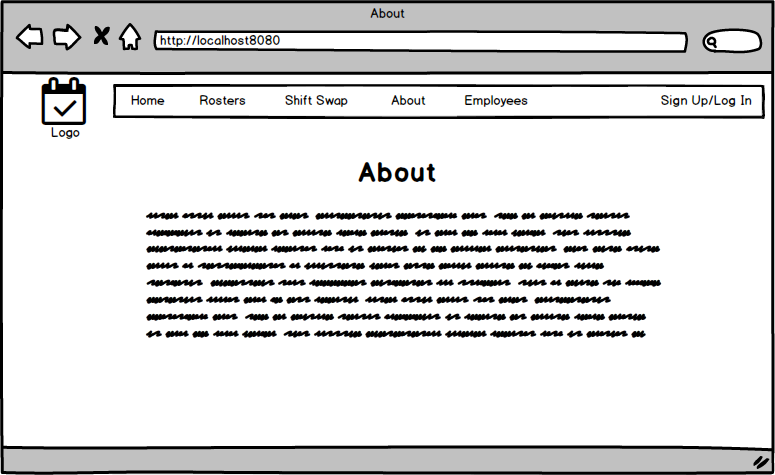


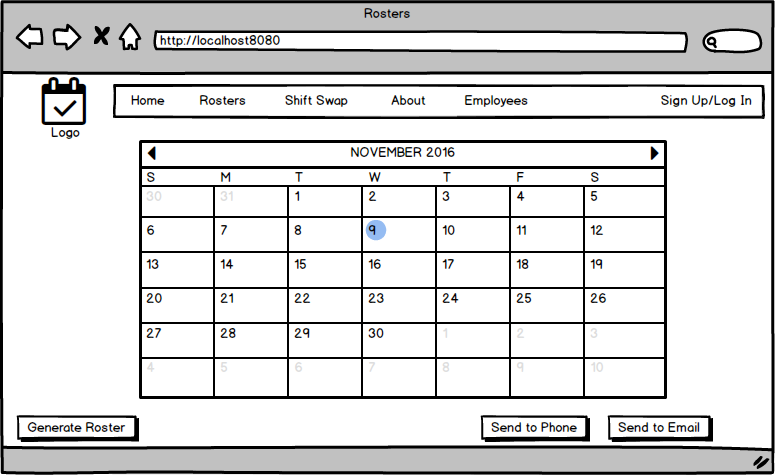




This is the employee screen where the system admin will add users which will save the user and their details to a database to use for generating the roster and for sending the roster to the appropriate email and phone number they supply. The admin can edit employees. As employees are known to leave companies the administrator will be able to remove employees from the database. The employees will be displayed in a grid with the most relative information on display.

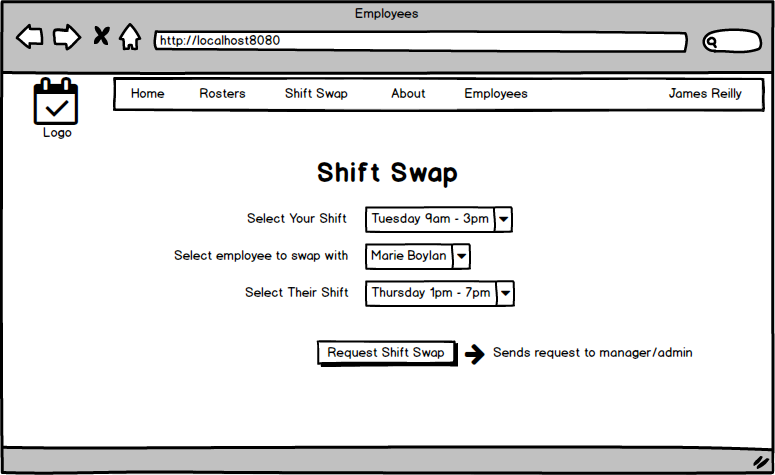
**3.1.3 About**





This is the rosters page where employees rosters will appear once the generate roster button is clicked. The user will be able to send the roster to the employees via phone and email so the rosters are easily seen and accessible.

**3.1.4 Shift Swap**



This is where employees can request a shift swap with a colleague via the system. They will select their shift, the employee they want to swap with and select the other employees shift they would like to take. The request will be sent by email to the manager and they will have the option to appove or reject the request

**3.1.5 Testing**

For testing Microsoft Test Manager, will be used to store tests that will be performed manually on the systems front end. The tests will be on the system to ensure there are no bugs or errors. To ensure overall quality and full coverage with testing these tests will be executed multiple times through an agile methodology of development. Once an element of the application has been implemented it will be tested throughout to ensure overall operational excellence. Exploratory tests will also be performed to get maximum coverage of the system. Testing will need to be sustainable and ran frequently. Testing API calls on the backend is also a viable method of testing. Testing will be performed across all browsers i.e. Chrome, Internet Explorer, Firefox, and Microsoft Edge. The main browser will be Chrome as most internet users use Chrome, but the application will be supported on other browsers for users who prefer Firefox to Chrome for example.

**3.1.6 Customer testing**

Feed back awaited. In progress.

**4. Evaluation**