**4. Design**

**4.1 Angular design.**

In Angular applications, modules are the basic building block. Each Angular application has a root module, named *AppModule* and provides the bootstrap mechanism that launches the application (Angular.io, 2020). Within the *AppModule* it contains child modules that are hierarchal in nature. While a small application many only contain the one module, larger applications, like the one developed, have a number of feature modules. A module groups together components, directives, pipes and services that are related so that they can be combined with other modules to create an application.

As well as having at least one module, every Angular application has at least one component. In the application this is *AppComponen*t which is the root component and connects a component hierarchy with the page object model, or DOM. Each component defines a class that contains application data and logic and is associated with an HTML template that defines a view to be displayed (Angular.io, 2020). Shared functionality between components are written into services. Services are injected into components and a component can delegate tasks, such as fetching data from the server or validating user input to the services. **FIGURE NUMBER** shows how all these main building blocks of an Angular application are related.

A picture containing food

Description automatically generated

Figure – Angular design (Angular.io, 2020)

The organising of code into distinct functional modules with components and services helps to develop complex applications with reusable code, allowing applications are lean and efficient (Angular.io, 2020). This modularity can be seen throughout the designed application for example, the *profile module* uses a number of components such as *user-information* and *user-shifts* which each have their own class that contains the logic for that component and their own HTML template. These individual components are then combined together to give the *profile* page in the application. Furthermore, the majority of components make use of the created shared services. For example the *user-welcome-component* uses the *getEmployeeDetails()* method that contains a HTTP GET request inside the *employee.service* to gather user information from the database and display it in the relative HTML template. A sequence diagram found at **FIGURE NUMBER** shows this process in more detail.

A screenshot of a social media post

Description automatically generated

Figure

**4.2 Application Design Structure.**

Following from the requirement elicitation, the structure of the application was designed. Using the Angular module and component design and the product backlog, the application was spilt into a number of key components that provide similar functionality. Through the process of the requirement elicitation and the problem specification it was clear that the application would provide different functionality and components depending on what the status of the user was. If a user was assigned admin status the system would contain a dashboard component, a rota component, an employee overview component, a timesheet component, a report component and a profile component. Whereas, a general user’s system would contain a dashboard and a profile component. Each of these components design will discussed later in this chapter. The application structure for both type of user can be found in **APPENDIX NUMBER**

**4.2.1 General User Interface**

As one of the non-functional requirements that was proposed was that the system should be easy to use so the user interface was designed with this in mind. It should be easy for users to use the system and be memorable so that once they learn how to use a feature once they don’t need to learn how to use it again. As well as this, the user interface was also designed in way that made it simple to navigate between functions and various components quick and easy. As a simple design and easy to learn features were needed, a CSS framework was used. The framework used is called MDB-Bootstrap.

By using a framework, it meant that the overall design, in terms of typography, colours and input controls was consistent and looked similar throughout the application. The framework also provides many UI elements that general web users are already familiar and so doesn’t require large amounts of time to be dedicated to learning how to use the system and its controls. The general user interface uses a blue navigation bar present at the top of every page once a user is logged. Furthermore, the main body of the application has a white background with black text making the information easy to read. Buttons are coloured according to their action, for example a delete button is given the colour red as this colour naturally makes user pause and think about whether or not they want to perform the action. As well as this, any error messages are shown in red with success message being given the colour green. Feedback from user tests showed that these designs to the general user interface makes it quick to learn and use the system as a whole.

**4.2.1 Sign in screen.**

**A screenshot of a cell phone

Description automatically generated**The sign in screen is the landing page to the application. It is a simple design and contains a simple form were registered users enter their email and password and click the button to sign in. If the credentials are correct the system navigates them to the relevant user home page. ***FIGURE NUMBER*** shows the initial design with ***FIGURE NUMBER*** showing the actual design. It can be seen that there isn’t a huge difference between each, this is because a sign screen follows a similar pattern no matter what application you use.

**A picture containing text, photo, looking, light

Description automatically generated**

Figure

Errors will show in red if either input field is touched and left empty. Furthermore, if the email input doesn’t match a standard email format an error will also show. If the user enters the details and it doesn’t match the stored credentials another error will show indicating which input doesn’t match the systems records. At the bottom there is a forgot password link that navigates a user to the forgot password page where they can reset their password. **FIGURE NUMBER** shows a flow chart for the process of events when a user uses the sign in screen.

Figure

A close up of a device

Description automatically generated

Figure

**4.2.2 Navigation bar**

Once a user has logged in there every page features a navigation bar, or ‘navbar’, at the top of the screen that and present on every page within application. This navigation bar is different depending on a user’s status and allows for easy navigation between features and pages throughout the application.**FIGURE NUMBER** shows the general user nav bar with **FIGURE NUMBER** showing admin navigation bar. A navbar was chosen as allows users to navigate easily through the application features. When the user is on a selected page the item associated within the navbar is given a CSS class of active to ensure that a user knows what page and feature they are using. It also contains a log out link that logs the user out of the application and then they are navigated to the sign in screen.



Figure - General User Navigation Bar



Figure 7 – Admin Navigation Bar

**4.2.3 User Dashboard/Homepage.**

**A close up of text on a white background

Description automatically generated**Following a successful login, the user is presented with the user dashboard. The admin and the user dashboard are both the same. During the design phase it was decided that the user home page should show an overview of the users shifts for the current week. **FIGURE NUMBER** shows the original design thought up during the initial design phase. By using the user details that have been entered during the login in phase an HTTP GET request sends a call to the API. This then returns the relevant information for the user and they are presented with their current shifts. From user testing it was made clear that it would be beneficial for a user to be able to view who is also working throughout the week. A button was added under each day and when clicked it shows a more detailed view of who is working, what time they are working and the area in which they are working. **FIGURE NUMBER** shows the actual design of the user home page.

Figure

**A screenshot of a cell phone

Description automatically generated**

Figure

**4.2.3 User Profile Component**

Another component in which the user and admin share is the profile component **FIGURE NUMBER** shows the original design sketch and **FIGURE NUMBER(A, B)** shows the actual design. The profile shows all the general user information, their shifts, their holidays and their training. The general user information shows details such as the users name, email, address and phone number. It was decided that this should be editable so that if any information regarding the user can be changed and the system can be updated according. If the user wants to edit their information, they click the edit button which launches a modal in which they can change their details. Form controls were used to ensure that no field was left blank and information was filled in according. This helps to ensure data integrity with regards to the database.

A close up of text on a white background

Description automatically generatedBelow the general user information is the *“*your shifts” section. This is used to show the users upcoming and past shifts. Each shift is shown with the date, start time, finish time and area in which the user is working. Moreover, it shows if the user is working today. If they are it displays the start time, finish time and area. However, if the user is not working today, it will display a message of “You’re off today, have a nice day!”. The original design intention in regards on how to display the shifts was to show every upcoming and past shift. However, after using the system it became clear that this idea would lead to the page being cluttered and messy as a user could have a large number of upcoming and past shifts. Instead, the upcoming shift area is limited to show 5 upcoming shifts and the past shifts were moved into side modal in which the user can scroll through and view their pervious shifts.

Figure

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedThe ‘unavailability’ section contains all the users upcoming and past holidays. As well as being able to view the dates there are unavailable, they can use it add a date in which they cannot work. The ‘+’ icon opens a dropdown containing a calendar in which they can select a date in the future to request off. This is then added to the upcoming holidays. If the user tries to add a date that they have already requested off an error message will show stating that this day has already been booked off. As well as this a user can delete an upcoming holiday by clicking the red ‘trash-bucket’ icon. One item that was not included in the original design was the “holiday count”. This was added because normally employees are limited to a set number of holidays and this ensures that the user will not exceed this amount.

Figure

Figure

Lastly the training section shows the users level of training. The original idea was that a user could add their own training, such as having a first aid training certificate. However, this was decided against because it could lead to false or misleading being inserted by the user. Instead, the admin can add training to the user in the employee component.

**4.2.3 Employee overview component.**

A close up of a logo

Description automatically generatedThe employee overview component was designed so that it would list all the employees registered to use the system. An admin can use this to create an employee, as well as being able to view and edit an individual employee, see **FIGURE NUMBER**. Furthermore, employees that no longer work for the company can be deleted.

Figure

The employee page gives an overview of all employees and individual employee information can be view by clicking the view button beside the user. Upon clicking of this button, the system navigates to user and loads all their information based upon the users ID. Once on the individual employee page a tabular button navigation is located at the top of the page. During the original design sketch the tabs are located on the left however, in the actual design they are located at the top **FIGURE NUMBER AND FIGURE NUMBER**. This change in design was because after some experimentation and use of the system the tabs worked better and were more visual pleasing if they were located at top of the page. Each tab will load the information regarding the individual employee and includes their general information, information regarding their shifts and holidays. As well as this an employee’s, training can be view and added. The training area is divided into the areas and other training. This was done as the area training corresponds the areas within the rota creation area. Other training includes other training such as first aid or health and safety that an individual employee may have completed.

A close up of text on a white background

Description automatically generated

Figure

The employee profile allows for the editing of the employee information. This is done through a form and the save changes button located at the button only becomes active if the system detects that any changes to the employee information have been made. Furthermore, as with all other forms in the system error messages will show if any input field is left blank, ensuring no null information will be inserted into the database. As well as this, to change the user status a drop displays the various roles in the system and if the user inputs information that does not match the created roles then an error will show and stop the role from being updated. Deleting a user will cause a pop up to open indicating that the user has been removed and the system navigates back to employee page in which the user has been removed and the employee count has been updated.

A screenshot of a cell phone

Description automatically generated

Figure

**4.2.4 Rota creation component**

The rota creation component is used to create the weekly rotas. **FIGURE NUMBER** shows the actual design of the page and **FIGURE NUMBER SHOWS** the initial sketch. From the sketch and the actual design, it can be seen that there isn’t much difference. This is because the initial design was based upon other rota creation systems. As some users may have experience in using the other system it was believed that the design should be as close to it as possible while providing similar features. As a result, it would aid in making the application easy to use as a user will be familiar with how the functions work.

**A close up of text on a white background

Description automatically generated**

Figure

A screenshot of a social media post

Description automatically generated

Figure

Upon loading of the page, the user is presented with the rota for the current day with the date being present at the top of the screen. Any shifts that have already been created for that day will be shown. Beside the date is two buttons which are used to forward and back a week. Below this are the day buttons which are used to navigate between the days of the week. Upon clicking the date will be updated making it easy for the user to remember what day they are creating a rota for. Any created shifts will be shown and if no shifts have been created a message stating that no shifts have been created for the selected date. From user testing it was made clear that being able to navigate quickly through the days and the weeks enabled quicker rota creation.

The main area is divided into a number of smaller subsections. Each corelates with the various areas in which an employee might be assigned to work in during their shift. These areas were thought up from experience working as the designed areas are the main areas that employees are always assigned to work in.

The left-hand side shows the list of employees and they are dragged and dropped into a selected area. A drag and drop method enables simple copying of a user into an area. An icon is located to each employee’s name and once clicked will open a modal that contains useful information related to the user such as upcoming holidays. This aids in the creation of rota’s as it means that a user cannot assign a user to work a day in which they are unavailable to work.

Upon dragging a user into an area, the user is promoted to insert a start time and finish time. If these times are left blank an error message will show and shift cannot be allocated to the user, this ensures no blank information cannot be inserted into the database. If the times are correctly filled in, then the shift can be allocated to the user. Once allocated the user can edit the shift if any mistakes or changes need to made and can completely remove the shift by clicking the remove button. An error message will show if the selected employee is already allocated to work or if the user has requested the day off. This pop-up error is useful as it ensures that employees are not allocated to work multiple shifts for one day and means that a user will not be assigned a shift in which they are unavailable to work.

**FIGURE NUMBER** shows a flow a diagram that indicates the logical steps that were needed in order to insert a shift for a selected employee. By following this earlier created diagram, it meant that the function could be designed accordingly.

A screenshot of a cell phone

Description automatically generated

**4.2.5 Timesheet component**

The time sheet component was designed so that employees past shifts and hours could be viewed and approved, **FIGURE NUMBER** shows the end design. The original concept was to just have an area in which past shifts could be approved for payroll summaries. However, the end design now includes a function in which the hours can be edited or deleted. This was added because often staff members can work for a longer or shorter period of time that what they were originally allocated to work or removed if the employee doesn’t work their allocated shift. For example, an employee might work to 6 o’clock when they were allocated to work to 5 o’clock so the shift will need changed or the employee might be sick and not work. By adding this function, it meant that all hours can be recorded correctly. Furthermore, any unscheduled shifts for an employee can be added and approved.

At the top there is a week control and it displays the week in which is selected. This means that when using the component, the user doesn’t get lost and knows what date frame of shifts they are approving. The component also displays the total amount of shifts that need approval meaning that the user can quickly gauge how many needs approved or edited.

The users are located on the left-hand side and once a user is selected their shifts are displayed along with the associated shift approval count, the employee’s total hours and their total pay for the time period. The card design means that the shifts are separated with each providing information regarding the shift. Upon approval of the shift the card turns green and provides a striking difference in shifts that need approval vs shifts that aren’t approved.

**A screenshot of a cell phone

Description automatically generated**

Figure

**4.2.6 Report component.**

A close up of text on a white background

Description automatically generatedThe report component was designed so that payroll reports could be viewed for all employees or an individual employee over a selected date range. The proposed design, seen in **FIGURE NUMBER** doesn’t differ from the actual design, seen in **FIGURE NUMBER**. There is a start and end date picker that allows to select the date range. Employees are selected via a dropdown that lists all the employees and the “All” dropdown element will get the reports for all the employees. Upon clicking the *Get Reports*button it will load all the shifts for the selected employee and date range. Each individual shift is listed with all the information that is associated with the shift. Furthermore, there is *a total hour* and a *total pay* sum that provides a quick summary for the all the hours the employees have worked, and the total pay indicates the total pay for the selected date range. If the selected employee has no shifts for the date range, the system will show a message specifying that the selected user has no shifts for the selected date range.

Figure

A screenshot of a cell phone

Description automatically generated

Figure