**6 Evaluation**

As defined in chapter two the aim of the project was to create a web application that makes employee management, rota creation and payroll management for hospitality managers easier. It can be seen that the created application provides all the functionality that was drawn up during the requirement elicitation stage. All the requirements that were given a “must have” or “could have” priority level where implemented and all the functionality based upon the user’s stories have also been implemented. Furthermore, from user acceptance testing it also showed that the non-functional requirements in terms of usability have also been met. Users said that the interface was easy to understand, and the design of the system and its features meant the it was easy to learn as a whole. Any problems that were found by users were fixed and meant the system become more reliable and increased the user’s confidence in the system. In terms of meeting the main aim of the project, it can be seen that the system meets those aims created at the start of the project, and as such can be considered a success.

As a whole the project can be considered a success, however, there are a number of downfalls or areas within the development of the system that could be improved. Given the time constraints associated with the project these downfalls may not have existed as more time could be given to solving them. Furthermore, the lack of experience in working with chosen technologies as this was the developers first time taking on a project of this size also introduced some weakness. In future development of a project of this size it will be implemented and designed differently due to the now required knowledge and experience.

More experience in using the Angular framework would enable the shared functionality of components to be separated into the created services. Instead, the majority of the component’s logic takes place with the component and should instead be placed within the designed services. The services have in the most part only been used for calling the API and making requests to the backend, as a result there is a large amount of code repetition within some components. Some components could share the same methods and functions but instead of placing them into services these methods have been repeated within each component that needs it. For example, the rota creation component has four separate areas, then manager-area, bar-area, floor-area and chef-area. Within these four components they all make use of the same methods and functions, however, these functions have been produced 4 times, when they only needed to be produced once. Looking back, it would have made sense to make use of Angular’s services and place all these methods into one service that each component could make use of. However, given that this knowledge was gained at a later date and due to the time constraints, this use of services wasn’t implemented and the already designed components were left untouched. Making use of the services would have reduced the amount of code and the size of the application and if future development of the project was to take place this is one of the main improvements that would be made straight away.

In terms of testing the application, only user acceptance testing, regression testing and testing of use cases was only carried out. More comprehensive testing would have included using the Jasmine test framework that is included with Angular upon each download and install. Unit testing of each module, component and service in project would have allowed more defects to be shown to the developer and as such, the system would lead to a much more reliable system. However, due to the aforementioned time constraints associated with the project, testing of this nature was not able to take place with only user testing being implemented. Again, in the future this is something that would be looked into and all modules, services and components would be tested to provide a more reliable system.

The way in which error’s that are returned from the server and dealt with by Angular or displayed to the user could also make use of some improvement. Upon each method call to the API the component that subscribes to the results handles the error. This again leads to a lot of duplicate code that could be reduced by introducing the ErrorHandler hook provided by Angular that creates centralised error handling. This error handler class would handle any errors returned in one global class that can be used by all components. However, this was not made use of instead each component handles its own errors and again this led to a lot of code duplication. As this error handler class was discovered at a later stage it was not implemented to the already existing code. Again, in any future development this feature will be made use of.

In terms of the technology picked to develop the application, it was the correct choice. The use of Angular allowed for an easy to use front end and through its features allowed for the creation of a dynamic web application. The backend server creation meant the functionality could be provided and all information associated with the application could be stored, edited and deleted. As all technologies shared the use of JavaScript in some capacity it meant the application could be quickly developed to a high standard. As experience in using the technology grew, more dynamic features could be added, and already existing features could be improved upon based upon user feedback.

While the application was only designed to be a web application to be used on computers an increased amount of web users are using a smartphone device, with mobile accounts accounting for approximately half of the web traffic worldwide (Clement, 2020). Using the application on a mobile device is near on impossible as many features become useable once the screen size goes below 400px in width. Some pages don’t even display the features if it detects that the screen size is below a certain size, instead it displays a message indicting that the feature is not available on a screen of that size. While the application was never designed to be used on mobile, the growing popularity of using mobile devices to browse the web means that any future development of the system should look at making the application mobile friendly.

Although there are downfalls in the created application, the project can still be considered as success as it meets all the user requirements. The downfalls exist in the organisation of the code and features that could have implemented however, which were discovered at a later date. These do not have any impact on the system itself and it still does what it was designed to do. Given more time, experience with the technology would develop and the problems shown would be changed. However, at this moment in time, the designed application meets the specification of creating a dynamic web application that makes rota creation, employee and payroll management easier. As well as this all the functional and non-functional requirements have been met, and as such, the project can be considered successful.

Talk about:

1. How it meant the requirements – functional and nonfunctional both met through user testing
2. Then talk about the weakness
   1. Testing
   2. Code organisation -
   3. Error handler
3. Although these exist it should be noted that it was your first time using the language and taking on a project of this size. While it doesn’t effect the code and the all the functions work as designed, in future the application would be structured differently now all this knowledge has been developed
4. Right technology chosen?
5. Mobile development
6. Future development.