**Requirements Analysis and Specification**

**3.1 Requirement Elicitation**

Requirement elicitation is the process of discovering and understanding the user needs and constraints for the purposed system. It provides developers with a helping hand in understanding what problems may arise and what the projects end goal is. During this stage the product requirements are decided, and these can take the form of both functional and non-functional. Sommervile (2016), describes functional requirements as what a system should do with non-functional requirements not being concerned with what the system does. Instead, they are constraints on the services or functions offered by the system such as cost, performance and reusability. During the first supervisorial meeting the functional requirements were discussed and this led to the creation of user’s stories. Each user story consists of a short description written from the user’s point of view with natural language. It focuses on what the user needs instead of what the system should deliver and are a very effective technique in requirements elicitation because their narrative structure helps users to describe what happens in different processes in the system (Sommervile, 2016), (Britton and Doake, 2005). With the creation of the user stories this enabled a product backlog to be developed based upon these user requirements.

The main aim of the system is to provide a functional and easy to use hospitality and employee management product. As stated, these requirements were discussed in the first meeting. However, as the Agile method was used in development new functional requirements were discovered and new user stories created. This meant that these requirements were added to the existing were added to the product backlog and were completed in the next sprint.

**3.2 Product backlog**

As the user stories were created, they helped to develop the product backlog. Each of the task was then given a priority level using the MoSCoW method. Using this method there is 4 levels of prioritisation “must have” (1), “should have” (2), “could have” (3) and “won’t have” (4) with must have being the highest level and won’t have being the lowest (BABOK Guide, 2009). Using this method, it meant that tasks given the must have or high priority level were completed in the first sprints, and if time allowed tasks with a low level or wont-haves could be implemented later in the development stage. Table 2 shows the product backlog with corresponding priority levels that was followed throughout the development of the system.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | MoSCow Priority Level \* | Sprint | User Type | Wants to… | So that… |
| 1 | 1 | 1 | User/Admin | Login into the system | They can use the application |
| 2 | 1 | 1 | User/Admin | Logout of the system | End the ‘session’ and remove all unsaved changes |
| 3 | 1 | 1 | User/Admin | Reset their password | They can access the application if they have forgotten their password |
| 4 | 1 | 2 | User/Admin | View their upcoming and previous shifts | They can view their upcoming shifts for each week |
| 5 | 3 | 2 | User/Admin | View their total hours for the week | They can view their total hours for each week |
| 6 | 2 | 2 | User/Admin | Add a holiday | Add a holiday so that they are not scheduled to work |
| 7 | 3 | 2 | User/Admin | View their upcoming and previous holidays | View their upcoming and previous holidays |
| 8 | 1 | 2 | User/Admin | View their profile | Can view their information |
| 9 | 1 | 2 | User/Admin | Edit their information | They can make changes to any incorrect information |
| 10 | 2 | 4 | User/Admin | Change their password | They can edit their password if forgotten or want to change it |
| 11 | 3 | 4 | User/Admin | View their training level | They can see their training levels |
| 12 | 4 | NI ^ | User/Admin | Create a profile picture | It makes it easier to remember employees |
| 13 | 1 | 2 | Admin | Create weekly shifts for all employees | They can easily create a weekly rota for all employees |
| 14 | 1 | 2 | Admin | Edit an individual shift | Any mistakes or changes can be corrected |
| 15 | 1 | 1 | Admin | Create an employee | A new employee can be included in weekly shift creation |
| 16 | 3 | 3 | Admin | View employee information | Gain a better understanding when creating rotas |
| 17 | 3 | 3 | Admin | Edit employee information | Any mistakes or changes can be corrected |
| 18 | 2 | 3 | Admin | Edit individual employee pay | Timesheets and weekly reports can provide more in-depth information |
| 19 | 1 | 2 | Admin | Change week and day when creating rotas | Rotas can be created for upcoming days/week and past and future rotas can be viewed |
| 20 | 2 | 3 | Admin | Approve employees’ hours | Payroll reports can be created |
| 21 | 2 | 3 | Admin | Edit employee's hours | Any mistakes or changes can be corrected |
| 22 | 2 | 3 | Admin | Add an unscheduled shift for employee that wasn't on the rota to work | Payroll reports and user reports are accurate |
| 23 | 3 | 5 | Admin | Change user’s role | Selected user can access to system can be changed |
| 24 | 3 | 5 | Admin | Add employee training | Users training can be view |
| 25 | 2 | 4 | Admin | Create payroll reports for each individual employee or for all employees | Payroll can be easily calculated based on total hours worked and employee pay |

Table 2 – Product backlog table

\*Priority Level: (1) - Must have/Critical. (2) - Should have/High. (3) - Could have/Medium. (4) Won’t have/Low.

^ NI – not implemented

**3.4 Non-functional Requirements**

As stated, non-functional requirements are not directly concerned with the specific services delivered by the system to its users. Instead they usually specify or constrain characteristics to the system as a whole. Non-functional requirements are often more critical than functional requirements as failing to meet a non-functional requirement can mean the whole system is unstable (Sommerville, 2016). Non-functional requirements can be divided into a number of categories including performance, reliability and usability. During the first supervisorial meeting the non-functional requirements were discussed and a list of these can be found below:

* *Usability* – This measures the characteristics such as consistency and aesthetics in the user interface. It is the ease at which the users operate the system and make productive use of it (Paradkar, 2017):
  + *It must be easy for a user to navigate and pleasant to use.*
  + *The system must be easy to remember for the casual user.*
  + *The user must understand what the system does.*
* *System –* This includes software and hardware responses:
  + *The system must be designed to be user on a web browser, more specifically Google Chrome.*
* *Performance –* A measure of how quickly the system responds to changes:
  + *All pages must not take longer than 10 seconds to load.*
  + *The system must not take longer than 10 seconds to log a user in.*
  + *All buttons must respond when they are clicked.*
* *Security* 
  + *Only registered user can use the system*
  + *To store a password in the database it must be hashed*
* *Reliability -* The ability of the system to function under stated conditions for a specific period
  + *All created features must work all of the time and error messages should show if part of the system fails.*

Paradkar, S., 2017. *Mastering Non-Functional Requirements*. Birmingham: Packt Publishing.