Team Six Deliverable Three

Andrew Clifford (19798632) Anzac Morel (66261880) Connor Macdonald (41647584)
George Stephenson (97277741) Hamesh Ravji (43832772) Rchi Lugtu (89933448)
Taran Jennison (67420293)

October 13, 2019

Contents

1 Executive Summary

2 Business and System Context

3 Stakeholders

This section aims to outline all the stakeholders the project has, and their respective concerns and what the team can do to relieve them. This helps the team see the point of view of the potential users and adjust course to best suit them and keep the project on track. Below there is a table which outlines the types of stakeholders, their needs, possible impact, priority, and action required.
Types of stakeholders:

- SP: Sponsors. People who commissioned project to be completed, including the intended people using the application.
- IN: Internal stakeholders. Employees/managers creating the product.
- EX: External stakeholders. People not directly involved but affected by outcome.

ID: SP0
Priority: High
Impact: High
Persona: Front of house employee
Front of house employees will be working directly with customers and the order management part of the application.
Stakeholders concerns:
Ease of use and speed when selecting options or possibly doing slightly customised orders.
Solution:
Create a clear GUI without unnecessary information.

ID: SP1
Priority: High
Impact: Medium
Persona: Truck manager
Manages the employees in the truck. Will likely update menus and stock levels in the system. Possibly ordering more stock when necessary.
Stakeholders concerns:
Ease of use when updating items in the application. Clarity of information shown so that they can make informed decisions based on stock levels, sales, etc. They might also want the ability to export/import recipes/stock for use in multiple food trucks.
Solution:
Create an easy to use GUI for displaying information and adjusting stock levels/recipes/prices.

ID: SP2
Priority: Medium
Impact: Medium
Persona: Truck Owners
Truck owners who are not also truck managers will not necessarily use the software but will rely on their employees to use it.
Stakeholders concerns:
Truck owners will need the software to work reliably for their employees to ensure their business has little downtime.
Solution:
High coverage of automated tests to ensure high quality and stability.

ID: SP3
Priority: Medium
Impact: High
Persona: Stock managers
Stock managers will only be interacting with the management side of the system to view and change stock levels.
Stakeholders concerns:
Ease of use with the management system. Access to financials, and stock. If the system fails the stock management will likely become inaccessible.
Solution:
Create an easy to use GUI for displaying information and adjusting stock levels/recipes/price. High coverage of automated tests to ensure high quality and stability.

ID: SP4
Priority: Medium

Impact: High
Persona: Chef
The chef will be relying on the system to give them orders. Likely won't interact with the GUI directly.
Stakeholders concerns:
Ability to see orders easily whether by notes or screen. If by screen, then an ability to dismiss or mark as complete is necessary.
Solution:
Create an output after each order is finalized that can be sent to the chef and displayed or printed in a given form.

ID: EX0
Priority: Low
Impact: Medium
Persona: Event Managers
Stakeholders concerns:
Event managers want the food truck to ensure everyone gets their order so reliability is their priority.
Solution:
Ensure orders are as easy to manage and consistent.

ID: EX1
Priority: Low
Impact: Low
Persona: Council
Stakeholders concerns:
The council wants the food truck to abide by food standards.
Solution:
Add a feature which allows tracking of food expiry dates.

ID: EX2
Priority: Low
Impact: Low
Persona: Stock suppliers
Stakeholders concerns:
Reliable stock order requests.

ID: EX3
Priority: Low
Impact: Medium
Persona: Accountant
Stakeholders concerns:
Access to organised financials.
Solution:
Make the management side easy to read and simple to export.

ID: IN0
Priority: Medium
Impact: High
Persona: The team
The team working on the project
Stakeholders concerns:
Meeting required standards and creating a product that fills the needs of the clients.
Solution:
Frequent interaction with stakeholders and good project management.

4 Quality Requirements

Operational / Management		
ID	**Description**	**Acceptance Tests**
Availability		
QR1	Software service needs to be available to the employees at all times when food truck is open / operational. No fatal errors.	System can run for 12 hours without an issue (12 hours because food trucks / small food businesses normally operates for around 10 hours or so).
Reliability (Robust)		
QR2	Software system should not crash or in general not be erroneous for the duration that the software service is being used.	If customization were made on a food by a customer (e.g no pickles in the burger), once the order has been processed, it should always notify the food staff on that customization.
Usability / ease of use		
QR3	GUI should be visually clear, and simple to use by end users. The application must be user-friendly	User should not be presented with more than 7 interactions at a time. User should not require any technical knowledge to use the app.
QR4	User interface should be easy to remember, and user should know how to use it on subsequent visits.	When the owner adds stock items, it should take < 5000ms to do it. User should not be presented with more than 7 interactions at a time.
Speed		
QR5	Software functionalities should not take a considerable amount of time to process.	Use case operations should process < 500ms, depending on the type of action the user performs, unless prompted otherwise. E.g When a customer orders food, the order processing time should take < 2000ms.
QR6	Some functionalities of the software should not take forever to process. However, it doesn't necessarily need to be super fast as well.	When the user checks the sales for the day, the information generated should take < 1000ms to be presented on the screen.
Maintainability		
QR7	Low coupling high cohesion. Code should easily be understood, repaired, or enhanced by other developers for future feature additions..	All methods (excluding some GUI methods) should have JavaDoc descriptions
Flexibility		
QR8	This is the ability on how easy our software can adapt to changes made in the future due to requirement changes etc.	When creating a newer and updated version for the software. The modules should have low interdependence with each other. Having a low dependence for software modules makes it easier for developers to add more functionalities etc, to the current version of the system
Portability		
QR9	This is the ability for our software to be accessed, deployed, and managed regardless of what platform it runs on.	System can be run on different platforms such as lab machines, or home desktop etc. As well as being able to import and export data.
Scalability		
QR10	The ability for our software to adapt to sudden changes in customer requirements.	System can be run on different platforms such as lab machines, or home desktop etc. As well as being able to import and export data.

5 Use Cases

6 Functional Requirements

UC1	Add money to the cash register	$200 in the cash register
UC2	Remove money from the cash register	$200 in the cash register
UC3	Too much money is removed from the cash register	$30 in the cash register
UC4	Add an item to the current order	One burger in the current order
UC5	Remove an item from the current order	One burger and one chips in the current order
UC6	Cancel the current order	The current order contains one burger and one chips
UC7	Refund the total cost of an order	The cost of the order that is to be refunded was $15 and there
UC8	Customise a menu item	The current order contains one burger
UC9	Add a new menu to the system	There are zero menus in the system
UC10	Add an item to an existing menu	A menu contains zero items
UC11	Remove an item from an existing menu	A menu contains one item, Burger
UC12	Edit an item	Burger doesn't contain tomatoes
UC13	Add a recipe to a menu item	Burger doesn't have a recipe
UC14	View the recipe for a menu item	Recipe for a burger needs to be viewed
UC15	Add stock items	5 buns are currently in stock
UC16	Update stock when stock is used	2 chips are currently in stock
UC17	Item with no stock is ordered	0 chips are currently in stock
UC18	View available stock	Levels of stock need to be viewed
UC19	View daily sales	Daily sales need to be viewed
UC20	Generate sales report	Sales report is needed
UC21	Change price of item	Price of burger is $10
UC22	Save menus to external file	Menus need to be exported
UC23	Load menus from an external file	Menus need to be imported
UC24	View sales from previous days	Previous days sales need to be viewed
UC25	Confirming an order	Current order contains one burger and one chips, total cost is $

7 Acceptance Testing

8 Deployment Model

9 Data Modelling

10 Technical Design

11 GUI Prototypes

Hello i am gui

Table 1: Summary of exposures to different risks that could occur during development

Team Risks				
ID	Description	Likelihood %	Impact 1 - 10	Exposure
R-01	Team Conflict	10%	1	0.1
R-02	Unfamiliar Development Tools	20%	2	0.4
R-03	Unfamiliar APIs/ libraries, various programming skill levels	90%	5	4.5
R-04	Miscommunication with lecturers	80%	5	4
R-05	Loss of data, problem with import of data	20%	6	1.2
R-06	Product does not agree with stakeholders expectations	90%	10	9
R-07	Code written by individual team members not readable by others	70%	4	2.8
R-08	GitLab, Google Drive, etc. become unavailable	2%	7	0.14
R-09	No Internet	5%	6	0.3
R-10	Bug in software e.g. bug says something gluten free when actually it isn't	90%	10	9

Table 2: Summary of consequences to different risks that could occur during development

ID	Consequences	Justification˜of
R-01	Reduced Productivity	Team has rules
R-02	Reduced Productivity as time spent learning how to use dev tools	Members of the
R-03	Some members limited to certain tasks, may mean some membes have to do more work than others	This is the tean
R-04	Doing tasks incorrectly and will have to redo or get a bad mark	The team has v
R-05	Have to re-complete work / manually import	All members of
R-06	Software won't sell (fake world) / bad mark from lecturers (real world)	It is very unlike
R-07	Reduced Productivity / Code has to be re-written	With new tools
R-08	Reduced Productivity, Can't commit new changes, may have to switch platforms	These services a
R-09	Can't commit changes to GitLab	Internet is prov
R-10	People could get sick	It is very unlike

12 Risks

12 Risks

The risk assessment module analyzes a number of different risks that both the team as well as the operator (user of the software) must be aware of during development and use of the software. Each risk is analyzed by multiplying its likelihood of occurring by the impact of the consequences on the group/user. This allows (low-likelihood, high impact) risks to be compared to (high-likelihood, low impact) risks. Most importantly, the last column of the table indicates how the risk can be avoided altogether, so this table should be referenced regularly.

12.1 Team Risks

12.2 User Risks

12.3 Risks Discussion - talk about what risks we have encountered and how we prevented / mitigated their effects, and justify why some values changed

Based on the feedback from the 1st deliverable and the 2nd deliverable it was clear that the risks section was not as extensive as it should have been, and some of the values were not correct e.g. We had the likelihood of team members being unfamiliar with libraries at 30% when really it should have been at 90%. In hindsight, we should have had more than one person deciding on the values for the risk assessment module as it resulted in biased and less thought through values. However time constraints near the end of the deliverable didnt allow for this. Time management was something that definitely held our grade back in the 1st and 2nd deliverable and this is a clear example of that.

For deliverable 2 we changed some of the likelihood values as you noticed and added a 'Justification of liklihood percentages' column too. Below are some examples of some of the risks we actually encountered and how they affected the development of the project.

R-03 - Unfamiliar libraries: This became clear to us very early on in deliverable 2 when using libraries such as JavaFX. Not many members of the group were familiar with it to begin with, even after having completed the JavaFX Lab. This hindered development in some areas where basic GUI functionality actually took a lot longer than expected to get up and running without any bugs.

Table 3: Summary of exposures to different risks that could occur during use of the software

User Risks				
ID	Description	Likelihood˜%	Impact˜1 - 10	Ex
R-11	Human error (misuse of software)	80%	9	7.2
R-12	Program freezes while processing customer's orders	20%	10	2
R-13	Screen showing the cooks what orders to make is inconsistent with actual order	20%	10	2

Table 4: Summary of consequences to different risks that could occur during use of the software

ID	Consequences	Justification˜of likelihood˜percentages
R-11	People could get sick	It is very likely that a user makes a mistake as mistakes happen frequentl
R-12	Angry customers lines get long, lose order	There is a low chance that the system will have a bug that will crash the
R-13	Angry customers	There is a low chance that the system will have a bug with such an integ

R-02 - Unfamiliar development tools: Most members of the team had only used Eclipse from SENG201 for Java projects. Switching over from Eclipse to IntelliJ was hard for some members of the team as the Project and Module SDK settings were playing up, however once we got it working there were no more problems and we concluded that IntelliJ is a lot better than Eclipse. SceneBuilder was also very new for most people, Taran did a lot of the design work for our GUI, so he had to learn how to use it by himself but he got the hang of it pretty quickly and produced an appealing GUI. We used Google Drive to store all of our design documentation as everyone was familiar with it, however we eventually had to switch over to LaTeX which was a new development tool for all of us. As this switch happened towards the end of deliverable 3, in hindsight we should have used LaTeX from the beginning as it is better than Google Drive.

R-08 GitLab, Google Drive, etc. become unavailable: We had the likelihood of this risk at 2%, as it seemed so unlikely as the development tools we were using such as GitLab and Google Drive are run by large companies. We were proven wrong when Google Drive crashed on us. Our design doc was 60 pages and when we had seven people trying to edit it at once, it crashed. Hence we switched our design documentation over to LaTeX which was much more friendly and has much better formatting tools. Connor's laptop also crashed two days before deliverable 2 was due. This was unfortunate, however we mitigated its consequences by making sure we always met where there was a lab computer available for Connor to work on. Hamesh also had a spare laptop that he kindly lended to Connor when we had to meet where there were no lab machines.

13 Testing Protocol

1	RowCol1	Row1Col2
2	Row2Col1	Row2Col2
3	Row3Col1	Row3Col2

Table 5: First Table

14 Project Timeline

15 Document Changelog

Appendices

A Class Diagram