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

2.1 System Context

This system, FoodByte, is to be used in either a small café or food truck, by the owners and employees to assist in managing the flow of orders and various day to day operations. Employees will be using the system the majority of the time it is in use so it is important the system will enhance their work and not interfere with it. Managers are able to interact with the system to manage stock, menus, menu items as well as view or refund past order transactions. The system can also communicate to the cooks/baristas to provide them with information on each order including a hierarchical tree displaying items within the order and items if any that correspond to their structure.

2.2 Relevant Business Information

FoodByte is worth developing because there is a gap in the software market for an application which is able to digitally manage day to day operations such as creating orders, managing stock as well as have a count of what is currently in the register all in one small package, and is designed for small businesses. FoodByte is able to provide the owners of the business with a tool to enhance and ease their order systems whilst providing a wide range of backend support for stock and item management.

The application will also allow for owners to import existing data files, such as xml files, containing data on products they offer into the system where the system will update all relevant sections in the application. Having the ability to import data allows the business load different sets of menus and items. The owner is also able to export the menu if they wish which will save the contents to an external file. Having the ability to export data allows the owner to keep track of how their business was doing at a particular time.

FoodByte is unique. One of the key selling points comes when adding new food items to the inventory, the high-level inheritance structure allow the user to have multiple variants of the same item. This allows for on the fly switching between variants of items, such as switching a burger to use gluten-free buns. This would greatly benefit some business as each item is independent of the next where customisations are concerned, for example; A customer may order a Cheese-burger but want to add more cheese, they can with the click of a few buttons.

The target market for the application is small food truck and cafe owners. Suitable customers could be a food truck stall at the Sunday market or a café at the University of Canterbury.

3 Stakeholders

3.1 Stakeholder Surveys

4 Quality Requirements

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

The application runs on a single machine, thus making a deployment model non-essential. Data, such as attributes related to stock instances, items, menu items, menus, and orders will be stored in an XML file which will is stored locally on the owner's machine with the added option to import other XML files formed from the same application. The 4-tier system uses a presentation, system, data, and physical layer where each layer interacts only with the layer beneath. This 4-tier system allows an easier transition from using XML files to store data, to using a database management system.

Originally, the plan was to implement a database into the FoodByte system, though being pressed for time meant that this goal was not achieved. With more time, this 4-tier application allows for simple implementation of a database which would be ideal for larger sets of data though at the moment the FoodByte application just stores data in an XML file called 'data.xml'. The process of saving data checks for corruption by storing data in a 'temp.xml' file then renames this file to 'data.xml'.

The system prints customers receipts and chef order slips via the command line interface when orders are finalised. This is because the development team does not yet have the knowledge or hardware to implement physically using the system with a printer to print receipts on command.

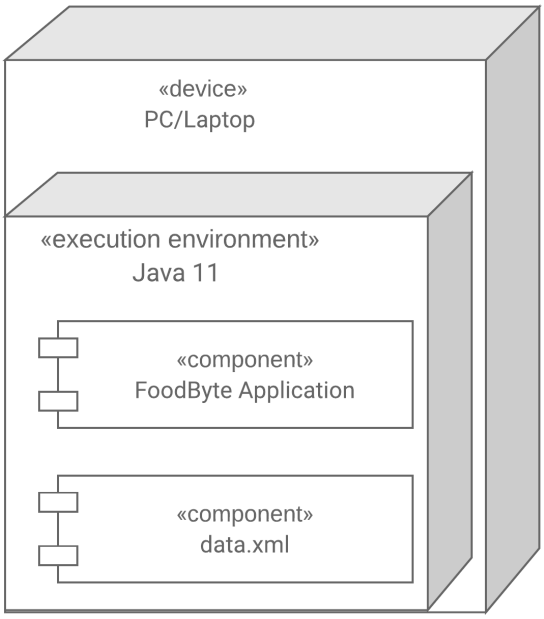

Figure 9: UML Deployment Diagram of the system

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