Project Proposal: GotoGro-MRM

Team Details

Team Name:	MSP 14
Tutorial:	Tue 2:30 ATC325
Tutor:	Dr Kaberi Naznin

Members:	
Dylan Jarvis	102093138
Rabya Tayal	103144215
Simon Tran	103602807
Thomas Babicka	103059885
Cody Cronin-Sporys	103610020
Nicholas Dyt	101624265

Background / Problem Description

Goto Grocery is a membership-based grocery store like Costco. They have trouble managing their inventory and often overorder. They currently use a paper-based management system but are looking to upgrade. Goto Grocery have commissioned us to create a basic application that will track their sales data and allow for better inventory prediction.

Scope

To address the main issues above the client has stated two primary objectives the software should fulfil:

Table 1. General Client Objectives

No.	Item	Description
1	Manage member records	Provide a visual interface to search for and retrieve membership details of each member
2	Generate Reports	Display visual reports on sales data for each member as well as create exportable csv reports, potentially for further processing

To support this goal the following specific requirements also need to be met:

Table 2. Specific Objectives

No.	Item	Description
1	Creation of database	The database will hold membership records and can be queried to present details for the interface
2	Creation of an application layer	The application layer will provide the GUI for the company to interact with the database

In order to best meet the client's needs, reasonable extensions beyond these goals were agreed upon by the development team:

Table 3. Extension Objectives

No.	Item	Description
1	Ability to manage inventory within database	To easily track sales data, the database must have a record of all the items Goto Grocery sells. In addition, the capability to add more item to the database would be desirable
2	Ability to manage members within database	Member details should be recorded in the database and displayed when relevant. The ability to add more members is a desirable addition
3	Virtual sales terminal	This interface will allow the team to add sales records as if they were recording a real-life transaction. In future this would ideally be linked to their POS system

Tables 1-3 capture the list of achievable tasks, however there are some tasks that lie outside the scope of the project. Ideally, the items shown in Table 4 could be easily implemented in future, built upon the base program:

Table 4. Out of Scope Objectives

No.	Item	Description
1	Ability to manage stock within database	This would entail the ability to enter in stock changes to the system. This would pair with the sales records to even more effectively manage inventory
2	Low stock trigger	This would entail the ability for the system to automatically detect when stock for a given item reaches the low threshold. This would trigger an automatic alert to help manage inventory
3	Connecting POS to database and application	As outlined above, it is out of scope to implement a physical connection between a scanner and the application, however this would be ideal in fully digitising the system and making the solutions as robust as possible

Stakeholders

Table 5 outlines the main stakeholders and their interests in the project.

Table 5. Stakeholders

Stakeholder	Interests
GotoGro Management	GotoGro management is the primary stakeholder in the project. They have requested the project and determine the features they need in the software. Most importantly, the continued success (or ideally improvement) of their bottom line depends on the success of the software so there is a very large and real stake in the matter. The management team also provides knowledge on the problem domain and brings a business perspective to the development process.

GotoGro Customers	Customers have an obvious stake in the software, though cannot directly contribute to its development. Customers are the ones who will receive the benefits of the correctly working program in the form of better item availability as well as more accurate member rewards. In some cases the customers could provide feedback to the company and this feedback could be used for future improvements.
Development Team	The development team has a stake in the project. They have the technical skills to develop the software and their reputation and career opportunities will be impacted by the success or failure of the project. The developers provide the most realistic feedback when considering how long a feature will take to implement.
GotGro Employees	The employees of GotoGro will be the primary users of the software and half of its success hinges on the data entry being correct. To this effect the GotoGro employees can provide the usability perspective to help determine whether the software UI is easily useable, makes sense, and meets business needs.

Deliverables and schedule

The physical deliverable will be in the form of an exe file which will house the interface and the database. It is not **within** scope to provide a server; thus, it is reasonable to house the database within the program to demonstrate proof of concept and meet the needs of the client.

Limited training will also be provided to brief the client on how to use the product before handoff. This will be delivered during the final sprint in the testing phase.

To complete the project a series of tasks need to be completed. The predicted project backlog is shown in the table below.

Table 6. Initial Release Schedule

No.	Item	Dependencies	Business Value (1 least – 10 most)	Release Schedule (Sprint 1 | 2)
1	Inventory Table	-	8	Sprint 1
2	Item Table	-	8	Sprint 1
3	Member Table	-	8	Sprint 1
4	Sales Record Table	1,2,3	10	Sprint 1
5	Add new member interface	3	4	Sprint 2
6	Add new item interface	2	2	Sprint 2
7	Add sales record	(1-4)	10	Sprint 2
8	View member sales record	7	9	Sprint 2
9	Predict required inventory	8	9	Sprint 2

10	View visual report of sales record	8	8	Sprint 2
11	Export report in csv format	10	8	Sprint 2

Solution Direction

In a realistic setting, the project calls for a web hosted database management system (DBMS) with desktop clients on all relevant point of sale (POS) machines. Together these constitute and enterprise framework. However, given the timeframe, we can simplify the implementation to either be entirely web hosted – accessible through a browser – or entirely desktop hosted, where the database exists directly on the host machine. Table 1 shows a comparison between the two alternatives:

Table 7. Comparison of Solution Directions

Desktop Application	Web Server
Database is stored locally on machine	Database is stored on the web on a server
Program is self-contained	Program must be stored a separate web server
Program is not easily accessible from mobile devices	Program can run on any device with internet access
No additional hardware required	Server required, either through cloud-hosting or physical storage
Generally cheaper	Generally more overheads
Requires coding skills only	Requires coding skills, also skill with markup languages HTML, CSS etc.

To make a distinction between the two, we must consider the problem domain, the solution domain as well as personal familiarity with relevant business knowledge, skills, and technology:

Table 8. Knowledge of the Problem Domain

No.	Item	Description
1	Customer needs	Knowledge of what a customer needs from the business, specifically, why are they using a member-based retail chain
2	Business needs	Knowledge of business needs, why they chose to use a member system, what benefits this provides
3	Current Technology	What kind of physical systems do they currently have in place, how do these function, how can they be used/repurposed
4	Network Architecture	How does the business currently manage its network, security concerns, how can this be leveraged to solve problems

Table 9. Knowledge of the Solution Domain

No.	Item	Description
5	Solution Technology	Knowledge of what technologies exist already to deliver digital reporting, how are these used in industry
6	Server Management	How to set up and manage a physical/cloud-hosted server to store company information
7	Cybersecurity	What are the threats to the system and how can these be managed

8	Desktop Applications	Competency in a coding like C#, C++, Python, Ruby, enough to deliver the desired functionality
9	UI	Competency in linking raw code to a user interface which can be used by the intended users without issue
10	Database Management	Competency in working with and extracting data from a database
11	File Management	Competency in extraction data from software or database and saving it into a file, especially csv format
12	Web Technologies	Competency using HTML, CSS, PHP and Javascript to create a web interface

Table 10. Team Competencies

No.	Item	Dylan	Rabya	Cody	Thomas	Nicholas	Simon
1	Customer Needs	Medium	Low	Medium	Medium	Medium	Low
2	Business Needs	High	Low	Medium	Medium	Medium	Low
3	Current Business Technology	High	Low	Low	Medium	High	Low
4	Network Architecture	Medium	Medium	Medium	Low	Medium	High
5	Solution Technology	Medium	Low	Medium	Medium	High	Medium
6	Server Management	Low	Low	Low	Low	High	High
7	Cybersecurity	High	Low	Medium	Low	High	Medium
8	Desktop Applications	High	Low	High	Medium	Medium	High
9	UI	High	Medium	High	High	Medium	High
10	Database Management	High	High	High	High	High	High
11	File Management	Medium	Medium	Medium	Medium	Medium	Medium
12	Web Technologies	High	High	Medium	High	Medium	Medium

GAP ANALYSIS

Table 4 demonstrates the overall competencies of our team in the skill areas required for the project. Overall, our strongest competency is in database management, which is ideal because both solution directions require a database.

Our weakest skill appears to be in server setup and management, closely followed by customer needs. Extra time must be assigned to these aspects to ensure they are up to standard.

In terms of experience with creating desktop applications versus web applications, we have a strong split, with a slight lean towards web proficiencies. Three of the team members opted for a web-based architecture, while the other three provided justification for a modified enterprise structure. In this model, the database would be hosted on a web server and business logic would be handled on another, but instead of providing access through webpages, a desktop client would be used to interface with the logic layer.

Ultimately, we decided to go for this model. The members of the team more comfortable with desktop applications will be responsible for creating the client software, while the team members with stronger web proficiency will be responsible for the database and business server setup.

HIGH LEVEL DESIGN

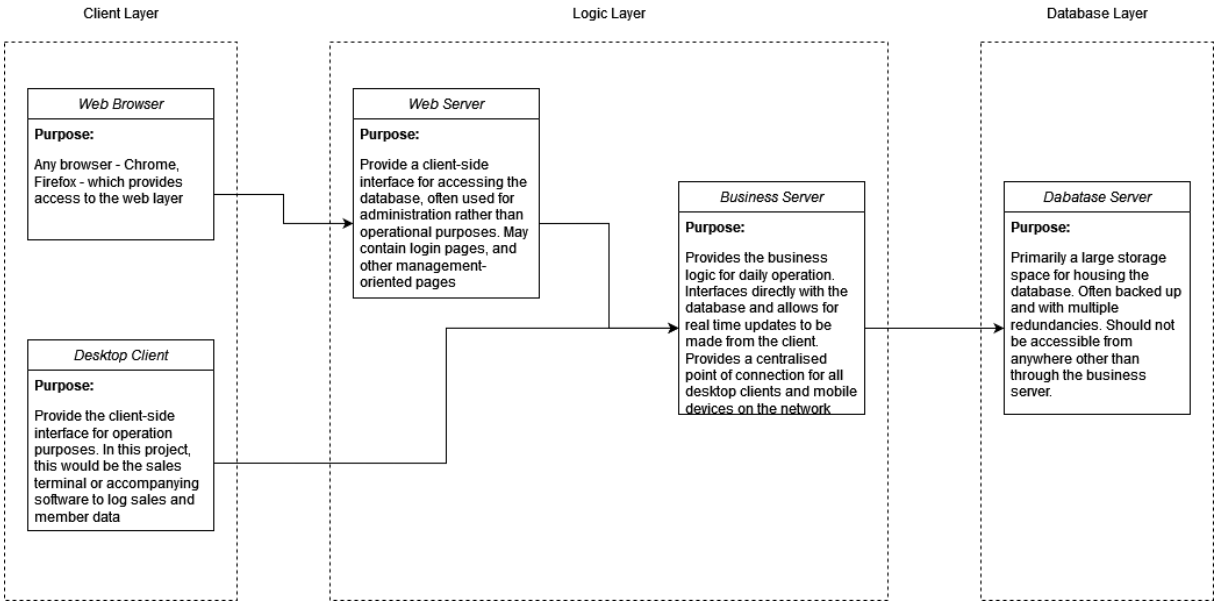
The high-level design is broken down into three layers:

The client layer (application layer), which provides a UI and a visual method of interacting with the system.

The logic layer, where business logic takes place and drives things like calculation and reporting.

The database layer, which houses the raw business data.

Together they create an architecture as follows:



Quality Management

Quality management on a software product relies on creating quantifiable test benchmarks and ensuring the software performs according or exceeding the criteria. Each item should be developed considering the user/client requirements. The items shown in green are SMART

goals the team can realistically achieve during the project timeframe, while the items shown in red are made to address the ISO standards and will not actually be carried out.

Table 11. Definition of Done

Condition	Description
Functional Suitability	Functional suitability concerns the physical abilities of the software and if they satisfy the requirements. For this project the program must be able to perform the functionality explained in Task 02P, specifically: - Users must be able to add, search for and retrieve details of active members by using a member ID lookup graphical interface. - Users must be able to generate reports in csv format summarising sales data over user-defined timeframes. - No more than 2 minor (still allows program to complete task) issues discovered over testing time of one week. - Able to run on computers with minimum system specs of 1GB of RAM and intel i3 or equivalent.
Performance Efficiency	Performance efficiency is a measure of how much time it takes for the program to perform key functions. No benchmark was specified by the client, but we can make a reasonable assumption to arrive at the following parameters: - It should take no more than 2 seconds for the program to add a record to the database once the button is pressed. - It should take no more than 2 seconds to display the result when searching for a member by their ID. - It should take no more than 1 second to add an item to the sales tab of a given member, this is essential so that the sales provider can continue scanning items in a timely manner. - It should take no more than 5 seconds to compile and publish a daily report in csv format. - It should take no more than 5 minutes to compile and publish a monthly report in csv format.
Compatibility	Compatibility refers to the program's ability to be run on machines with different architecture and operating systems: - No specific platform has been specified by the client. The program should therefore be functional on the 3 most common operating systems: Windows, Linux and Mac. - Cannot noticeably slow down other programs running on the target machine.
Usability	Useability concerns the ease of use of the program considering the average training of the user. Though hard to quantify, the following parameters will be used to satisfy this condition: - Users should not be able to enter any incorrectly formatted member or item ID's. - Interface should be similar to currently existing POS systems. - Users should be able to use the interface to the same degree as other aspects of their job, ie. if someone with a given disability is capable of working at GotoGro, then they should be able to use the program interface. - An operator familiar with the company should be capable of using the interface with <5% mistakes in 2 days or less. - An operator new to the company should be capable of using the interface with <5% mistakes in a week or less.
Reliability	Reliability refers to the program's ability to perform its function correctly and accurately: - To consider the software successful it should have an internal error (not caused by user error) rate of no higher than 0.5%.

	- Worded differently, only 1 error is permissible in every 200 item or member entries. - All database data should be preserved in the case of an outage or failure.
Portability	Portability refers to the ability of the software to be used in multiple environments and still perform as expected: - System must completely replace paper-based ordering system.

Team Discussion

Many values were consistent across the team, especially in relation to functional stability and performance. The definitions mostly contested related to defining useability, reliability, and portability. Tables 2-6 summarise the discussion and ultimate justifications that led to selecting the conditions above.

Table 12. Justification for Functional Suitability

	Detail
Chosen Definition	Users must be able to add, search for and retrieve details of active members by using a member ID lookup graphical interface. Users must be able to generate reports in csv format summarising sales data over user-defined timeframes. No more than 2 minor (still allows program to complete task) issues discovered over testing time of one week. Able to run on computers with minimum system specs of 1GB of RAM and intel i3 or equivalent.
Alternate Definition	None.
Deciding Factors	Not applicable. The team had already agreed on functional scope meaning there was no alternate opinions to be chosen between.
Justification	No justification needed; these are the functional components specified by the client.
Conformation to ISO Standards	Our requirements fulfill the “functional completeness” subheading, as meeting them satisfies the complete design brief provided by the client. Functional correctness is not explicitly addressed however we discussed using a testing window (of a week) during which no more than 2 minor issues (cosmetic or database related) can be discovered, else the program would be considered incorrectly functioning. Specifying the RAM required on the operating machine obeys functional appropriateness as POS systems are likely to be lightweight and potentially old, therefore the program must run as intended on systems with the specified specifications.

Table 13. Justification for Performance Efficiency

	Detail
Chosen Definition	It should take no more than 2 seconds for the program to add a record to the database once the button is pressed. It should take no more than 2 seconds to display the result when searching for a member by their ID. It should take no more than 1 second to add an item to the sales tab of a given member, this is essential so that the sales provider can continue scanning items in a timely manner.

	It should take no more than 5 seconds to compile and publish a daily report in csv format. It should take no more than 5 minutes to compile and publish a monthly report in csv format.
Alternate Definition	Various small changes between 1-3 seconds for all record-based additions and manipulations. Generating a daily report should take no more than 5 minutes to compile.
Deciding Factors	There was a discrepancy between the expected time it would take for a report to be generated. The 5-minute suggestion was made under the assumption that reports would only be generated at standardised off-peak times like end of the shift or end of the month making time efficiency not a priority. Contrastingly, it was discussed that the amount of data needed to generate even a daily report should be able to be done in under 5 seconds in csv format.
Justification	Taking a standard daily estimate for supermarket sales as roughly $70,000, and the fact that the average shopper in Australia spends $160 on groceries per week (across all trips, but considered as a single trip here), we estimate that GotoGro serves approximately 440 members a day. Generating a csv file with 440 lines is easily achievable in under 5 seconds, even if each item the customer bought was itemised (assume 30 items per customer on average, yielding 13,125 entries). Additionally, it was discussed that a shift supervisor might want to quickly make a report halfway through the day to gauge their sales performance, in this case it is ideal to return this report as quickly as possible to not interrupt service.
Conformation to ISO Standards	All parameters come under the ISO subheading “time behaviour”, as they all define time specific benchmarks which the software must meet. Resource utilisation was not considered in depth given the overall spec limitation of the kind of devices that would be used (POS machines). Capacity is difficult to quantify given the client did not provided details of the number of customers currently in their database, however yielding a search result in under 2 seconds provides a soft benchmark on capacity, as meeting this requirement depends on the current database size.

Table 14. Justification for Compatibility

	Detail
Chosen Definition	No specific platform has been specified by the client. The program should therefore be functional on the 3 most common operating systems: Windows, Linux and Mac. Cannot noticeably slow down other programs running on the target machine.
Alternate Definition	None.
Deciding Factors	Not applicable. The team agreed on platform compatibility.
Justification	It is industry standard to create programs with capability of running on at least the big three OS’ being Windows, Mac and Linux. Similarly to resource capacity limitations, noticeable slowdown of other programs is a more qualitative metric but one of the most obvious to detect during testing.

Conformation to ISO Standards	ISO specifies co-existence as a subheading which is covered by our metric of ensuring no noticeable slowdown of other programs. The ISO specification of interoperability is not exclusively OS based, but also speaks to the interoperability of the program with other programs on the same device and alternative entry points such as mobile apps etc. Given our program is self-contained and will be built to only talk to a specific single database, interoperability is a negligible consideration.

Table 15. Justification for Usability

	Detail
Chosen Definition	An operator familiar with the company should be capable of using the interface with <5% mistakes in 2 days or less. An operator new to the company should be capable of using the interface with <5% mistakes in a week or less. Users should not be able to enter any incorrectly formatted member or item ID's. Users should be able to use the interface to the same degree as other aspects of their job, ie. if someone with a given disability is capable of working at GotoGro, then they should be able to use the program interface. Interface should be similar to currently existing POS systems.
Alternate Definition	Operators should be able to use the interface with less than 5% error within a day.
Deciding Factors	Since members of the team have retail experience using similar systems we discussed and drew from those team members' experience to determine what timeframe was most reasonable to expect from a staff member.
Justification	Mistakes are damaging to a business; therefore, the priority is to ensure that the mistakes being made are minimised due to error checking. Obviously, user error is unavoidable (entering in the incorrect member ID), but this is where ease of access and learnability plays a key factor in helping team members to get comfortable with the interface as quickly as possible.
Conformation to ISO Standards	Appropriateness recognisability, and user interface aesthetics are both mentioned in ISO and are covered her under the line concerning designing the interface similar to currently existing systems. Learnability and Operability are covered by the considerations speaking about the timeframe and error margin that workers should be expected to uphold using the system. Accessibility is directly addressed with our requirement that any current GotoGro worker should be able to use the interface. The final ISO requirement is user error protection. Given that items will be entered via barcode scan there is no need for error protection, the best error protection we can feasibly implement is ensuring correct formatting for member ID input.

Table 16. Justification for Reliability

	Detail
Chosen Definition	To consider the software successful it should have an internal error (not caused by user error) rate of no higher than 0.5%. Worded differently, only 1 error is permissible in every 200 item or member entries. All database data should be preserved in the case of an outage or failure. Only the current transaction will need to be redone.
Alternate Definition	System should be able to be recovered and restored in 30 minutes or less.
Deciding Factors	Team discussion was conducted concerning the feasibility of restoring the system in a given timeframe. Though a good benchmark, it is unrealistic to hold the quality of the program on a timeframe that is largely out of our control. Even the best software can suffer an attack or malfunction that can take days to repair.
Justification	Setting the benchmark for acceptable failure is somewhat arbitrary but based on the estimated number of customers served per day. Taking the number from above (440), our maximum permissible error criteria would allow for 2 issues in data/member entry over the course of a single day. Using team member retail experience, this was deemed acceptable because the rate of mistakes made due to operator error is already much higher and to the customer there is no distinction in their service experience.
Conformation to ISO Standards	ISO standards mention Maturity as the specification for how well the software performs under expected conditions, we have accounted for this with the minimum permissible error criteria. As for availability, since the system is small scale to start with (one shop only), the database access should not hinge on many factors other than the network setup in the store. It is not directly addressed in our conditions, but it is safe to assume that the program itself will not cause any foreseeable downtime. Fault tolerance is another standard that is hard to quantify, we do not benchmark it here as it is assumed that hardware faults will be covered by redundancy (multiple POS systems active). Recoverability is the final ISO standard, though we decided to exclude the timeframe constraint, we still address recoverability by enforcing the condition that no database entries should be lost in the case of system failure.

Table 17. Justification for Portability

	Detail
Chosen Definition	System must completely replace paper-based ordering system.
Alternate Definition	Program should be able to be installed on new sites within a day.
Deciding Factors	The alternate suggestion was excluded here due to scope. Given the client's brief, the program is only expected to function in a single store, rendering the installability and adaptability of the system largely negligible.
Justification	Portability blankets over the subheading replaceability, which refers to the software's success in replacing the system it was designed to replace. It is clear from the client brief that this program is designed to replace a paper-based management system.

Conformation to ISO Standards	As mentioned, adaptability and installability of the ISO specifications can be considered negligible for this project due to them being out of scope. Replaceability is the final specification to address and we have done so be ensuring that no staff members are using the paper based system after the software is implemented.

Resources

The resources involved in a project like this in real life would require both human and hardware resources. Theoretically, the human resources would include a representative from the company to provide better business insight as well as the development team. Without a business rep, we have allocated roles within the team as shown in Table 18:

Table 18. Member roles

Member Name	Member ID	Role
Dylan Jarvis	102093138	Product Owner, Scrum Master
Rabya Tayal	103144215	Team Member
Simon Tran	103602807	Team Member
Thomas Babicka	103059885	Team Member
Cody Cronin-Sporys	103610020	Team Member
Nicholas Dyt	101624265	Team Member (Lead Developer)

Of note, the role allocation in an agile setting is limited since there is flexibility to work within different areas as a team member (developer). When it comes to sprint planning, we will allocate different backlog tasks to the member with the most relevant skillset or experience in that area.

The hardware resources needed would be access to machines such as barcode scanner and the currently business technology, however since this is being demonstrated as a proof of concept, we will use a virtual interface to simulate the client-side actions.