Project Proposal: Members Record Management System (GotoGro-MRM) for Goto Grocery Inc.

Members:

Enzo Peperkamp - 102895415

I'm in full agreement with everything discussed so far, from objectives to technological choices. One aspect that further encourages me is the balanced distribution of roles within our agile team. Having a Scrum Master, Tester, two Developers, and a Product Owner strikes an excellent equilibrium for skill and responsibility distribution. This setup ensures that each facet of the project is given focused attention, from development to testing and product management. I'm confident that with this level of expertise and specialisation, our team is well-positioned for a successful project execution.

Nelchael Kenshi Turija - 103057559

I am fully aligned with the comprehensive project plan created by the team. The plan's focus on developing a user-friendly member management system, its well-structured scope, clear objectives, risk management strategies, and budget allocation demonstrates a strategic approach which I am in full support of. I believe that the project plan aligns with Goto Grocer’s requirements as outlined in the scope and project objectives.

Julian Codespoti - 102997816:

I believe this collaborative report encompasses our thoughtfully crafted project proposal for GotoGro MRM, reflecting diverse expertise in addressing system requirements, quality goals, and role distribution. The plan's strategic approach, user-centric system focus, and balanced roles within our agile team inspire confidence in its success.

Alex Kyriacou - 103059830

I believe the following document represents a faithful combination of all our team's work thus far. It is from this plan that we will be able to effectively plan our following sprints and the work required. We have effectively set out roles throughout the team that will allow for a solid foundation in the delegation of work and responsibilities moving forward.

Marella Morad - 103076428

I agree with everything documented in this report. We have worked together as a team to come up with the most suitable project proposal for GotoGro MRM. We have carefully considered all the system requirements and the backlog items that will enable us to meet these requirements. We thoroughly discussed and agreed upon the quality model and goals we would like to achieve within the given timeframe. Having a diverse team from different backgrounds and various technical skills will enhance our chances of creating a successful project.

Table of contents:

Background / Problem Description

Situated in the Hawthorn region, Goto Grocery operates as a member-centric grocery store facing challenges in meeting its members' expectations and satisfying their diverse grocery needs. The store currently utilizes an outdated paper-based system to record member details and sales transactions, an approach that has repeatedly demonstrated inefficiency and a propensity for errors. Consequently, Goto Grocery has encountered issues such as ordering unnecessary items, leading to wastage and financial implications due to unnecessary item orders.

Project Objectives

To develop and implement a comprehensive, user-friendly member management system that:

1. Modernises and enhances store operations.
2. Provides timely comprehension of members' needs.
3. Ensure accurate and efficient record-keeping.
4. Offers a more sophisticated approach to inventory management.
5. Is ready for deployment within a year of project commencement.

Scope

This project encompasses the design, creation, and execution of a sophisticated software application tailored to serve as a member management system for Goto Grocery. The software will present a graphical user interface (GUI) to ensure user-friendly data entry, management, and analysis. The key functionalities of the application are detailed below:

- Offering a centralised digital platform for member record management.
- Integrating a sophisticated graphical user interface (GUI) for easy data entry, management, and analysis.
- Facilitating efficient inventory management by providing real-time data analytics on product demand.
- Supporting multi-user access levels for different staff roles, ensuring data security and integrity.
- Incorporating a responsive design to allow access on various devices, ensuring accessibility for staff both on-site and off-site.

Stakeholders

The primary stakeholders for this project include:

1. **Goto Grocery's management and staff:** As the end-users of the system, benefiting from the system's data analytics for strategic planning.
2. **Customers:** Beneficiaries of improved in-store experiences and personalised offers.
3. **Project team:** Tasked with system development, deployment, and maintenance.
4. **Suppliers**: Indirect beneficiaries from efficient inventory management and ordering processes.

Risks and Mitigation

1. **Data Breach**: Implement advanced encryption methods and regular security audits.
2. **System Downtime**: Adopt a robust backup and disaster recovery solution.
3. **Incomplete Data Migration**: Ensure thorough data validation during the migration process.
4. **Scope Creep**: Regularly review project objectives and maintain clear communication with stakeholders.
5. **Budget Overruns**: Adopt a phased development approach, prioritising essential features.

Testing and Quality Assurance

1. **Unit Testing:** Each module/functionality will be tested individually for correctness.
2. **Integration Testing:** Ensuring seamless interaction between different system components.
3. **User Acceptance Testing (UAT):** Staff members will test the system in a real-world scenario before full-scale deployment.
4. **Security Testing:** Identifying potential vulnerabilities and ensuring data protection measures are effective.
5. **Performance Testing:** Ensuring the system performs optimally under expected loads.

Budget

- **Development Team:** 3 developers (Front-end & UI/UX, Back-end & Database, and a Full-stack developer)
- **Testing Team:** 2 testers (1 for manual testing and 1 for automated testing)
- **Project Manager:** Responsible for project timelines, stakeholder communication, and budgeting.
- **Budget Breakdown:**
    - **Software Development:** $60,000
    - **Testing and QA:** $15,000
    - **Deployment and Training:** $15,000
    - **Contingency Fund:** $10,000
- **Total Estimated Budget:** $100,000

Out of Scope

1. **Payment and Financial Transactions:** The application will not handle payment processing or financial transactions. It will focus solely on managing members' records, grocery needs, and inventory.
2. **Automated Ordering and Restocking:** While the software will track inventory levels and members' needs, it will not directly place orders or manage restocking operations. These activities will continue to be handled manually by store staff.
3. **Integrations with External Systems:** The application will not integrate with other third-party systems, such as accounting software or point-of-sale systems.
4. **External System Integrations:** No direct integrations with third-party POS or financial software.
5. **Barcode Scanning:** While beneficial, barcode scanning for inventory management is excluded in the initial release.

6. **Online Shopping Platform:** The software will not provide an online shopping platform for members. It will serve as a tool to assist the brick-and-mortar store in better catering to its members' needs.

By staying focused on the specified objectives and scope, the project will provide Goto Grocery with a powerful and efficient solution to streamline its operations, improve member satisfaction, and reduce unnecessary costs.

Deliverables and Schedule

1. **Software Application:** A fully functional member management system with a user-friendly interface.
2. **Source Code:** All developed source code.
3. **User Manual:** A comprehensive user manual.
4. **Training Program:** A training program for the staff at Goto Grocery.

Initial Release Schedule of the Product Backlog Items

No.	**Item**	**Dependencies**	**Business Value (1 least – 10 most)**	**Release Schedule (Sprint 1 | 2)**
F1	Create database table for members	-	9	1
F2	Create database table for sales records	-	9	1
F3	Create database table for products	-	9	1
F4	Create endpoint for adding members	F1	8	1
F5	Create endpoint for editing members	F1	8	1
F6	Create endpoint for removing members	F1	8	1
F7	Create endpoint for adding sales records	F2	8	1
F8	Create endpoint for editing sales records	F2	8	1
F9	Create endpoint for removing sales records	F2	8	1
F10	Create endpoint for adding products	F3	8	1
F11	Create endpoint for editing products	F3	8	1
F12	Create endpoint for removing products	F3	8	1
F13	Create endpoint for retrieving products	F3	8	1
F14	Create endpoint for retrieving sales	F2	8	1
F15	Create endpoint for retrieving members	F1	8	1
F16	Create a prediction algorithm to predict member’s grocery needs based on previous sales	F1, F2, F3	7	2
F17	Generate sales report	F16, F1, F2, F3	7	1
F18	User authentication and account management	F4, F5, F6	5	1

F19	Build web page to allow users to add new members	F4	8	2
F20	Build web page to allow users to edit existing members	F5	8	2
F21	Build web page to allow users to remove members	F6	8	2
F22	Build web page to allow users to add new sales records	F7	8	2
F23	Build web page to allow users to edit existing sales records	F8	8	2
F24	Build web page to allow users to remove sales records	F9	8	2
F25	Build web page to allow users to add new product	F10	8	2
F26	Build web page to allow users to edit existing product	F11	8	2
F27	Build web page to allow users to remove product	F12	8	2
F28	Build web page to allow for sales data to be imported into backend system via CSV	F7	5	2
F29	Build web page to allow for data to be exported to CSV	F13, F14, F15	7	2
F30	Build web page to allow users to view sales reports and analytics	F17	7	2
F31	Implement search functionality on the stock management page with filters to let users browse products	F13	5	2
F32	Generate inventory reports			1

Solution Direction

Description of chosen solution direction

After careful consideration and assessment between all group members' proposed solution. The team has decided to go with the solution proposed by Alex. Namely, we have decided to develop a cloud hosted Model-View-Controller (MVC) architecture. This architecture is a natural choice given the brief. GotoGro is a small organisation that is likely unable to maintain its own server infrastructure. Therefore maintaining a simple serverless architecture allows for easy extensibility if future upgrades are required. This solution is a well tested solution to this common industry problem and consists of the following architecture:

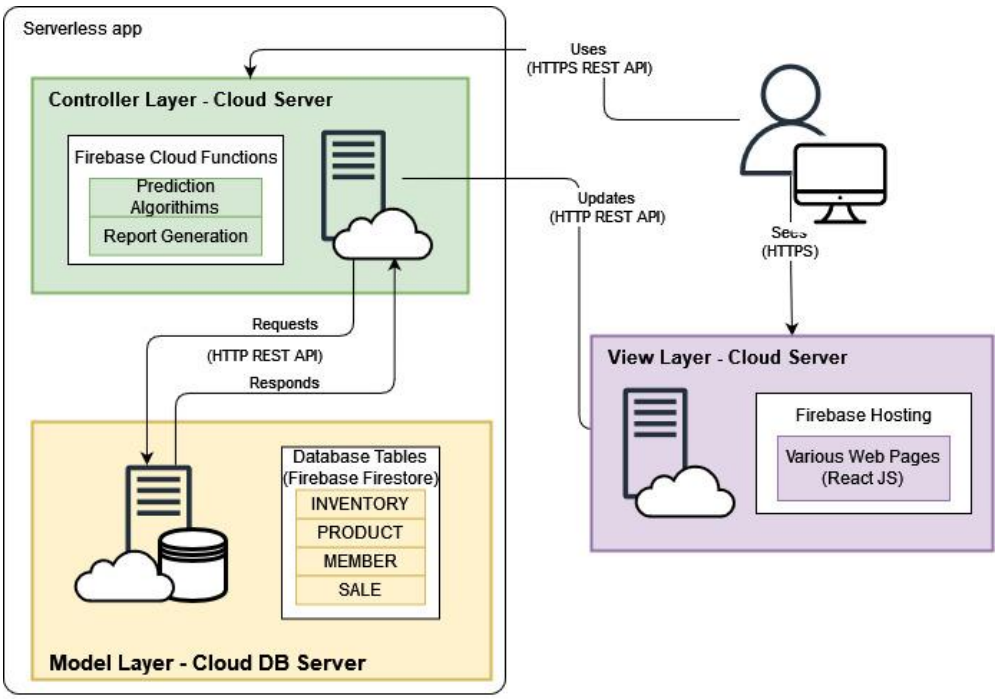

Figure 1: Architecture Of Agreed Solution Direction

Model Layer

This consists of a cloud hosted relational database service that will store all the database entities required for the system. This layer will interact with the controller layer in order to update records as well as retrieve relevant data from the database for The UI, report generation, algorithm prediction etc. Given this is using a serverless infrastructure, this will expose an endpoint that can be queried by the controller.

View Layer

This layer is what the end-user interacts with through their browser and represents the GUI. This consists of a cloud hosted React JS application that will display the relevant data, reports and UI to the user. This will query the controller to retrieve and update any data that is required.

Controller Layer

This consists of a server that serves as the intermediary between the View Layer and the Model Layer. The Controller serves to handle all requests from the View Layer and translate them into database queries and software processes before returning the final result. This includes generation of inventory reports, execution of a prediction algorithm or retrieval of a certain member's sales. The Controller Layer will be created by a number of cloud hosted

serverless functions that will act as a robust REST API that can be queried by the View Layer to perform required tasks.

Alternatives

Architecture	Description	Reason for Discarding	Potential Use Cases for GotoGro
Traditional Monolithic Application	Single-tiered software where GotoGro's UI, business logic, and data access are all in one codebase, likely hosted on a single server.	While suitable for initial GotoGro setup with a single store, it lacks scalability. Updates would affect the entire application.	If GotoGro remained a small local business with no plans to expand and only required basic features.
Frontend + Serverless Backend	GotoGro's web interface connects to serverless functions that execute backend tasks, potentially leveraging platforms like AWS Lambda.	While it allows for auto-scaling, the unpredictable latency might disrupt GotoGro's real-time inventory or billing operations.	For specific event-driven features, like sending notifications to customers about offers or processing one-off bulk orders.
SPA with Integrated Backend	GotoGro as a Single Page Application (SPA) with dynamic content loading from an integrated backend.	Scalability concerns arise as the SPA grows. SEO might be a challenge if not properly optimised.	If GotoGro decided to launch a special promotional website with limited-time offers and wanted smooth user interactions without full page reloads.
PWA + API Backend	Progressive Web App (PWA) version of GotoGro providing native-like capabilities, interacting with a backend via APIs.	Complexities related to service workers, cache management, and potential pitfalls in offline data syncing.	If GotoGro decided to create an offline-capable app for locations with spotty internet, like pop-up stalls or kiosks in remote areas.

Micro-frontends with Microservices Backend	GotoGro's frontend and backend split into smaller, independently deployable units. Each store or department could have its own micro-frontend.	High operational overhead and intricate deployment pipelines.	Ideal for a future where GotoGro has multiple distinct departments or franchises, each wanting autonomy in their tech stack and deployments.
Full-stack Desktop Application	A desktop application tailored for GotoGro staff, managing everything from inventory to billing, and syncs to a remote database.	Challenges in cross-platform compatibility, deployment, and updating across stores.	If GotoGro planned to provide a highly specialised in-house software tool for staff, focusing on intensive tasks like 3D product visualisation or complex inventory planning.

Chosen Solution Analysis

KoST Analysis

Knowledge (K):

- **Problem Domain:**
  - **Paper-Based Records Management:**
    - GotoGro's current manual management, akin to traditional retailers, is prone to errors. Transitioning to an electronic system will streamline tracking and analysis.
  - **Brick and Mortar Retail:**
    - Sales at GotoGro, like other physical stores, require an electronic system for transaction entries, as records are not created automatically.
  - **Understanding of Grocery Store Operations:**
    - Comprehensive grasp of industry standards, including member management, sales records management, inventory management, and member-based business models, similar to those in prominent grocery chains.
- **Solution Domain:**
  - **Electronic Records Management:**
    - Implementation akin to modern e-commerce platforms, for tracking orders and members, reducing manual effort seen in GotoGro's current process.

    - **Electronic Inventory Control:**
        - Integration with systems similar to ERP solutions that manage stock levels.
    - **Data Analysis and Processing:**
        - Usage of predictive algorithms and ETL practices, comparable to leading data analytics tools, for trend analysis and data cleaning.
    - **Software Engineering Expertise:**
        - Familiarity with .NET languages like C#, C++, JavaScript (including frameworks like Angular, React), PHP, HTML, CSS, Python, and various DB systems like MongoDB, SQL, NoSQL, and Firebase.
    - **User Experience Design:**
        - Applying principles used in successful web applications to ensure a user-friendly interface.
    - **Alignment with Business Needs:**
        - Tailoring the solution to meet GotoGro's specific needs, leveraging understanding of grocery operations and member management, mirroring best practices in the industry.

Skill (S):

- **Web Development:**
    - **E-commerce Systems:**
        - Experience in building basic web-based e-commerce systems with inventory management, similar to some small-scale online retailers, using .NET, JS, and other related technologies.
        - Willingness to mentor the team and fill experience gaps in this area.
    - **Cloud-Based Infrastructure:**
        - While there's an experience gap in using cloud-based hosting, there is a commitment to learning and implementing basic cloud-based hosting for websites, databases, and servers.
- **Industry Experience:**
    - **Supermarket Retailer Insight:**
        - Experience working within large supermarket retailers like those found in major chains, providing valuable insights into inventory management, usability, and UI decisions of competitors.
- **Data Analysis:**
    - **Gap in Predictive and Trend Analysis:**
        - Acknowledgment of a need to enhance skills in predictive and trend analysis, required for inventory prediction algorithms, mirroring sophisticated data science applications in other industries.

- **User Interface Design:**
    - **Intuitive Design:**
        - Proficient in designing intuitive and user-friendly graphical user interfaces, applying principles found in successful digital platforms.
- **Database Management:**
    - **Designing and Managing Relational Databases:**
        - Skillful in managing relational databases, utilising popular DB systems like SQL, MongoDB, Firebase, in line with best practices in database design.
- **Programming Skill:**
    - **Wide Range of Languages:**
        - Proficiency in languages including .NET languages (C#, C++), JavaScript (including Angular, React), PHP, HTML, CSS, Python, ensuring the development of required functionality.
- **Project Leadership:**
    - **Agile Model Leadership:**
        - Experience in leading teams under the Agile model, reflecting a strong understanding of Agile principles, and a readiness to take ownership in team development.
- **Alignment with Business Needs:**
    - Leveraging the above skills to precisely meet GotoGro's specific needs in terms of web development, UI design, inventory prediction, and member record management.

Technology (T):

- **Existing Systems:**
    - Platforms like Woolworths and Coles' websites demonstrate existing technology for online groceries, but they are not tailored to Goto Grocery's specific needs, especially regarding sales recording and member management.
- **Customization Gap:**
    - While similar member and sales record management applications exist, the supermarket industry has generally lacked systems that can be customised to individual retailers' unique needs and operations.
- **Technological Accessibility and Innovation:**
    - The proposed application for Goto Grocery leverages existing and well-established technologies such as web development frameworks and database technologies like MongoDB, SQL, Firebase.
    - This approach ensures accessibility without the need to introduce unique or new technology, in line with current industry standards.

- **Historical Gap in Supermarket Industry:**
    - Historically, the supermarket industry's focus has been on in-store experience and product availability rather than detailed member management and predictive inventory control.
    - This can be likened to the gap in the car industry where sensors were absent, leading to a lack of assistance in tasks like backing a car.
    - Addressing Goto Grocery's problem through a tailored Members Record Management System (MRM) fills this gap by emphasising member preferences, inventory prediction, and seamless sales recording.
    - Just as sensors revolutionised safety in cars, a custom-built MRM can transform member satisfaction and operational efficiency in supermarkets, setting a precedent for industry evolution.
- **Alignment with Business Needs:**
    - By focusing on Goto Grocery's specific requirements and industry-wide gaps, the technology chosen for this project represents a blend of innovation, accessibility, and customization, designed to elevate Goto Grocery's operations and potentially reshape technological norms within the supermarket industry.

Quality Management

Definition of Done

The "Definition of Done" for the GotoGro MRM application serves as a critical and comprehensive checklist for conditions that must be met to ensure the successful delivery and release of the product. This definition aligns with both the ISO 25010 standard and specific project objectives, guaranteeing that the application not only meets but exceeds the quality standards necessary to fulfil the project's functional and non-functional (quality) requirements. The definition consists of the following items:

1. Code Standards & Quality:
    - Any code must be checked in.
    - At least one other team member must review code.
    - Any functions and classes within the code must have XML (or equivalent) docstring justifying its reason for existence, arguments, and return types.
    - The project must compile/build without any errors.
    - Any refactoring must be completed.
2. Documentation & User Guides:
    - Any relevant documentation must be updated.
    - The application is thoroughly documented, including user guides and technical documentation as a deliverable.
3. Data Migration & Reporting:
    - All member records are accurately migrated from the paper-based system to the application database.

- The application can generate reports detailing member and sales records.
- Records can be exported to CSV format without data loss or formatting issues.

4. Testing & Performance:
   - User acceptance testing is incorporated and completed, and feedback is assimilated for improvements.
   - Performance testing is performed to ensure the application can handle expected usage loads.
5. Deployment & Accessibility:
   - The application is deployable and accessible to all relevant stakeholders.

Quality Model

- Functional Suitability
  - Functional Correctness
    - Number of Critical Defects found in testing
      - 0 Critical Defects
    - Number of Defects per KLOC (Thousand Lines of Code)
      - 10 defects per KLOC
  - Functional Completeness
    - % of functions that are completed
      - 90%
- Performance Efficiency
  - Time Behaviour
    - % of functions returning within the specified response time.
      - <= 90%
  - Resource Utilisation
    - % of system resources utilised (such as CPU and memory)
      - <= 60%
- Security
  - Confidentiality
    - All Personally Identifiable Information (PII) and Sales data will be inaccessible until an authorised user has been authenticated.
      - Data will be encrypted adhering to ISO27001 standards.
  - Authenticity
    - All users will require authentication before being allowed access to the system using a 2-factor authentication method.
      - Google Authenticator / Or Text Message Authentication
  - Integrity
    - Number of vulnerabilities found in the software through security assessment
      - Zero (0)
- Usability
  - Operability
    - Average time taken by users to add new member/sales records or to edit new records
      - Average submission time is less than 2 minutes

    - Learnability
        - Amount of training (hours) required for non-technical stakeholders to be able to learn all aspects of the system.
            - 2 Hours
    - User Protection Error
        - % of user tasks that are completed successfully without errors
            - >= 95%
- Reliability
    - Availability
        - % of uptime the system is required to have
            - 99%
        - The average time taken to restore the system to full functionality after a failure
            - 1-hour

Quality Goals

- After submitting a purchase record, the system will allow the users to begin creating a new purchase record within 2 seconds.
- A confirmation message will be displayed within 3 seconds of creating a new user.
- The home page of the system will display within 3 seconds.
- An exported CSV with a file size of less than 2 megabytes will be downloaded within 5 seconds.
- All reports should be generated within 10 seconds of invocation from the user.
- The system shall support up to 5 users concurrently.
- Upon review, the codebase should not have more than 10 defects per KLOC (Thousand lines of code)
- Under normal load conditions, the system should not utilise more than 60% of its resources to ensure other system functionalities remain unaffected.
- Out of 100 attempts to update member information, at least 95 of them should be completed successfully without any errors.
- If the system encounters any technical glitches (i.e. becomes unresponsive), then the system should be fully restored within 1 hour after the failure is identified.
- Upon conducting a security assessment, no critical vulnerabilities should be found.

Resources

Name of student	*Student Id*	*Role*	*Responsibilities*
Enzo Peperkamp	*102895415*	*Scrum Master*	• *Owns the Agile process* • *Aims to motivate, coach and guide team ensuring that team does not go off target* • *Makes all project level decisions*

			• *Ensures that team has all the necessary information to deliver on the vision within the defined constraints*
Nelchael Kenshi Turija	*103057559*	*Tester*	• *Conduct thorough unit testing to ensure individual components and functions work as intended* • *Participate in test case creation and execution to ensure comprehensive test coverage* • *Verify that the GotoGro MRM meets specified functional and non-functional requirements* • *Provide clear and detailed testing documentation*
Julian Codespoti	*102997816*	*Backend Developer*	• *Developing and maintaining server-side logic and databases to support frontend functionalities.* • *Building APIs and endpoints to enable data exchange between the frontend and backend systems.* • *Ensuring data security, authentication, and authorization mechanisms are implemented effectively.* • *Optimizing server performance, including database queries and response times, for efficient backend operations.* • *Collaborating with frontend developers, designers, and other team members to integrate frontend and backend components seamlessly.*
Alex Kyriacou	*103059830*	*Product Owner*	• *Prioritising the product backlog alongside team* • *Understand business level risks and consequences of actions from a business perspective* • *Decides when to release product into production* • *Manages product vision*
Marella Morad	*103076428*	*Frontend Developer*	• *Designing the frontend user interface.* • *Setting up the required frontend technologies and frameworks (such as React).* • *Developing and maintaining the webpages and related frontend components.* • *Collaborating with the backend developer to establish a connection between the frontend and the backend.* • *Collaborating with other team members for feedback on the overall appearance of the project.*

Approval Signatures

Project Team

	Name of student	Student Id	Signature
1	**Enzo Peperkamp**	102895415	Enzo Peperkamp
2	**Nelchael Kenshi Turija**	103057559	Nelchael Kenshi Turija
3	**Julian Codespoti**	102997816	Julian Codespoti
4	**Alex Kyriacou**	103059830	Alex Kyriacou
5	**Marella Morad**	103076428	Marella Morad

Project Sponsor - Harsharan Kaur

Tutor's name (on behalf of the client)	**Signature**