Project Proposal[1]: GotoGro-MRM

Background:

Goto Grocery, a member-based grocery store located in Hawthorn, is currently facing challenges in managing member records and fulfilling their grocery needs efficiently. The store currently relies on a paper-based system to track member details and sales records, leading to difficulties in meeting member expectations and avoiding unnecessary orders. To address these issues, Goto Grocery is seeking a comprehensive software solution that can streamline member management, track grocery needs, and provide valuable insights for better decision-making.

Scope:

Objective:

The objective is to create a user-friendly GUI-based system for GotoGro, enabling efficient management of member records, real-time alerts for necessary stock orders, and seamless data export to CSV format. The main aim is to overcome current challenges from the paper-based system, enhancing customer satisfaction, optimizing stock levels, and modernizing member management for the store's improved operational efficiency.

In Scope:

1. **Member Registration and Management:** Create, edit, view, and delete member profiles, storing contact details, purchase history, and preferences.
2. **Efficient Sales Tracking**: Record and manage sales transactions, providing real-time access to sales data for informed stock management decisions.
3. **Accurate Grocery Needs Analysis**: Utilize historical sales data to predict member grocery needs and recommend stock purchases.
4. **User-Friendly Interface**: Design an intuitive GUI for seamless user interaction and data entry.
5. **Report Generation**: Generate CSV-exportable reports summarizing member information, sales records, and grocery needs insights.

Out of Scope:

1. **Payment Processing:** Payment-related functionalities and integration with payment gateways are not within the scope of this project.
2. **Inventory Management:** Detailed inventory tracking and management are not included in this proposal. Only stock levels and sales information will be in scope to ensure member-preferred products are in stock with they need them.
3. **Third-party Integrations:** Integration with external systems beyond the scope of member management and grocery needs assessment is not part of this project.

Deliverable and Schedule:

Deliverables:

- **Source Code:** Provide the complete source code of the application for potential future modifications and enhancements.
- **Running Application:** Deliver fully functional software, including a user-friendly graphical user interface (GUI) and the necessary backend code.

[1] This document is by no means a “full project proposal”. It has been simplified and customized for the purposes of SWE20001 teaching. The full project proposal includes many other sections which have not been discussed during the first few weeks of SWE20001 teaching.

- **User Manual:** Develop a comprehensive user manual that guides users in effectively utilizing and navigating the application.
- **Training Materials:** Include basic training videos to assist users in getting acquainted with the application's features and functions.

Initial Release Schedule of the Product Backlog items:

The initial release schedule will be divided into 4 sprints:

1. **Sprint 1:** Focuses on database creation and backend processing/functionality.
2. **Sprint 2:** Focuses on UI design and webpage creation.
3. **Sprint 3:** Focuses on Reporting functionality.
4. **Sprint 4:** Dedicated to testing.

No.	Item	Dependencies	Business Value (1 least – 10 most)	Release Schedule (Sprint 1 | 2 | 3 | 4)
F1	Add a member: • Create database tables for new members. • Create functionality for adding new members in the backend.		7	Sprint 1
F2	Add a sales record: • Create database tables. • Backend functionality. • Link Sales records with member records	F1	8	Sprint 1
F3	Edit a sales record: • Create database tables. • Backend functionality	F2	8	Sprint 1
F4	Implement grocery preferences: • UI page/s to allow members to select their grocery preferences. • Backend processing of preferences and storing them against members' records	F1, F2	9	Sprint 1
F5	Build webpage components for member management	F1	8	Sprint 2
F6	Build webpage components for sales management	F2, F3	8	Sprint 2
F7	Add backend functionality to generate CSV reports	F1, F2, F3, F4	9	Sprint 3
F8	Build webpage components for CSV report export	F7	8	Sprint 3
F9	Application Testing	F1, F2, F3, F4, F5, F6, F7, F8	10	Sprint 4

Development of a Member Management System for Goto Grocery

Members:

Enzo Peperkamp - 102895415

I am in full agreement with the team's decision regarding the objectives, scope, and deliverables outlined in this document. The proposed member management system will undoubtedly benefit Goto Grocery by streamlining its operations and enhancing the member experience. It's well-structured and clearly presents our strategic approach to address the identified challenges.

Nelchael Kenshi Turija - 103057559

I agree with the project plan proposed below. The plan addresses Goto Grocery's challenges, and fully explains what the benefits are in addressing these problems. All sections are well-thought out and the entire plan is well-structured. The schedule is well spaced out to avoid overexerting the team in sprints.

Julian Codespoti - 102997816:

I agree with the outlined project plan. The clarity and structure of this document are commendable. I believe that by executing this plan meticulously, we will effectively address Goto Grocery's challenges and substantially enhance the overall member experience. I'm eager to collaborate and bring this vision to fruition.

Alex Kyriacou - 103059830

I believe the below represents a solid foundation for the rest of this project. It represents a flexible but succinct plan as to how we plan to approach this project for the rest of the semester.

Marella Morad - 103076428

I agree with the proposed project plan for Goto Grocery. The plan outlines our strategic approach to addressing challenges and enhancing the member experience, with the schedule ensuring efficient execution.

Table of Contents

Background

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 burdens.

Project Objectives

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

1. Modernises and enhances store operations.
2. Ensures accuracy in record-keeping.
3. Provides timely comprehension of members' needs.
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:

Stakeholders

The primary stakeholders for this project include:

1. **Goto Grocery's management and staff:** As the end-users of the system.
2. **Customers:** The system will enhance their shopping experience.
3. **Project team:** Responsible for delivering the system.

Risks and Mitigation

1. **Risk:** Technical difficulties during development.

   Mitigation: Regular project status checks and having a technical expert on standby.

2. **Risk:** Resistance from staff due to change.

   Mitigation: Providing comprehensive training and demonstrating the benefits of the new system.

Testing and Quality Assurance

The system will undergo rigorous unit testing, integration testing, and user acceptance testing to ensure quality and reliability.

Budget and Resources

The project will require a team of 5 developers and a project manager, with an estimated budget of $100,000, which includes software development, testing, deployment, and staff training.

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. **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 and inventory reports	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

[1] This document is by no means a "full project proposal". It has been simplified and customised for the purposes of SWE20001 teaching. The full project proposal includes many other sections which have not been discussed during the first few weeks of SWE20001 teaching.

Project Proposal[1]: GotoGro-MRM – Solution Design

Table of Contents

[1] This document is by no means a "full project proposal". It has been simplified and customized for the purposes of SWE20001 teaching. The full project proposal includes many other sections which have not been discussed during the first few weeks of SWE20001 teaching.

Solution Direction Description:

Based on the project objectives, scope, and the analyses we conducted in 01P and 02P, both individually and as a group, I find that the most suitable chosen solution direction for Goto Grocery's member management system is to develop a web-based application with a multi-layer architecture. This architecture will consist of the following layers:

- Frontend Layer (User Interface)
- Backend Layer (Application Logic and Functionality)
- Data Storage Layer

This choice is driven by the need for a user-friendly graphical user interface (GUI), the ability to handle real-time interactions, and the project's focus on modernizing and enhancing store operations. By dividing the member management system into distinct layers, the system functions more efficiently, is easier to maintain, and becomes more straightforward to scale if the need arises.

KoST Analysis:

Knowledge:

Problem Domain:

- Understanding the challenges faced by Goto Grocery in member management and the grocery industry. Such as difficulties managing their current paper-based records system which makes analysing records and extracting relevant information harder.
- Knowledge of inventory management, member needs, sales records, and reporting is required to design an effective solution.

Solution Domain:

- Implementing an electronic records management system for easy access and analysis, reducing manual efforts in managing orders and member data.
- Introducing electronic inventory control that synchronizes with records, enhancing stock level tracking and accuracy.
- Utilizing data analysis, including trend and predictive analysis, to ensure preparedness for consumer demands and streamlined stock management.

Skills:

- **UI Design:** Experience in designing easy-to-use user interfaces taking into consideration usability and accessibility standards.
- **Frontend Development:** Experience with frontend development languages (such as HTML, JavaScript, and CSS) and frameworks such as VueJs and React.
- **Backend Development:** Experience with backend development languages (such as C# and C++), design patterns and technologies.
- **Database Management:** Experience with database design, SQL and working with relational databases as the system requires storing member and sales data and extracting that data for analysis and reporting.

Technology:

- No new technologies are needed for the implementation of this design as many existing solutions use the same architecture. All the technologies mentioned above are accessible, for example, frontend frameworks and backend technologies have been used in so many projects and are highly reliable and accessible for the purpose of this project.

Alternatives Considered:

Table 1 Alternative Considered and Justification for Discarding

Alternative	Description	Rationale for Discarding
Desktop Application	Developing a desktop application instead of a web application. This application would be installed on individual computers within the store and accessed locally.	Discarded due to limitations in accessibility and platform dependence. A web application, on the other hand, offers easier access and can be used across different devices and operating systems.
Commercial Software	Purchasing and customizing existing commercial software for member management	Discarded due to the need for a tailored solution to match Goto Grocery's specific requirements as well as added customisation costs.

Rationale for Chosen Solution:

After careful consideration, the above two alternatives (see Table 1) were discarded in favour of developing a multi-layer web-based application for the following reasons:

- Customisation: Goto Grocery needed a solution that could be tailored precisely to its unique requirements, offering functionalities that met the store's specific needs.
- Accessibility and Flexibility: A web-based application provides accessibility from various devices and locations. Staff members could manage the system remotely, ensuring flexibility and convenience.
- Scalability: A web application can be easily scaled to accommodate future growth in terms of users, data volume, and functionalities.
- Cost-Effectiveness: While the initial development of a custom web application involves upfront costs, it offers a long-term cost advantage compared to purchasing licenses and customizing commercial software.
- Alignment with Modern Trends: Web applications align with contemporary technology trends, making them a more appealing choice for users accustomed to intuitive and user-friendly interfaces.

In conclusion, the chosen solution of a web-based application with a multi-layer architecture offered the best balance of customization, accessibility, scalability, and cost-effectiveness, making it the most suitable choice for Goto Grocery's member management system.

High-Level Design:

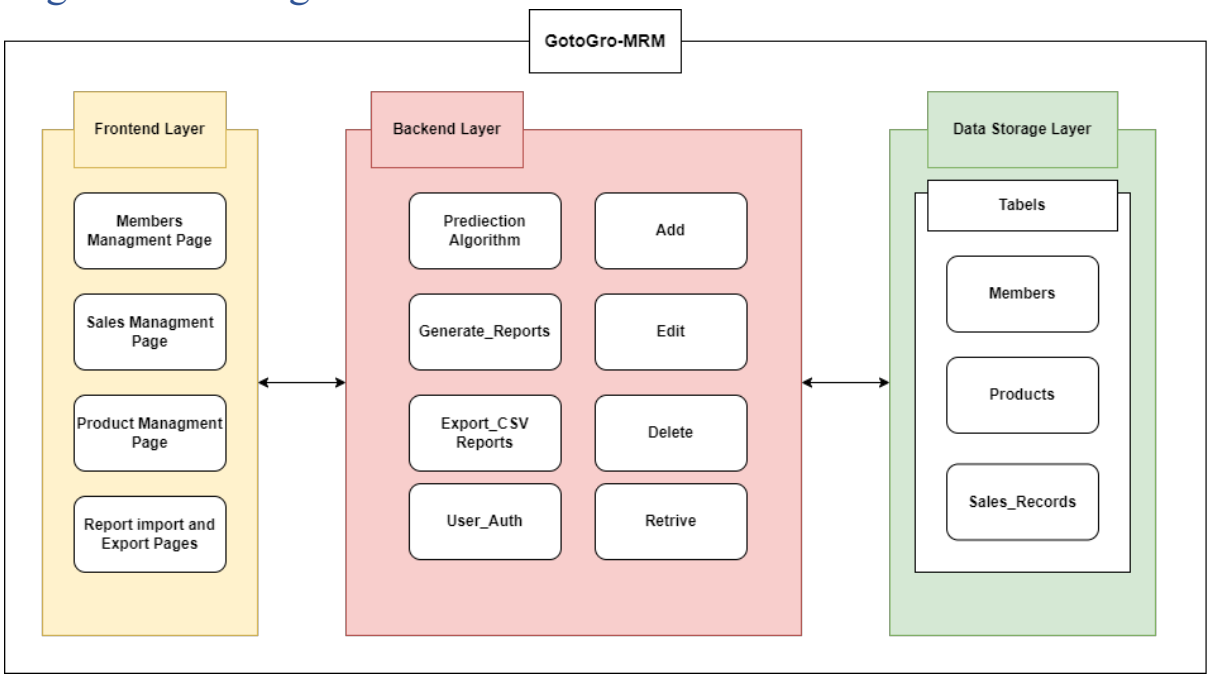


Figure 1 High-Level Design of the Chosen GotoGro MRM

Frontend Layer (User Interface):

This layer is responsible for the user interface components that members and store staff interact with. It includes web pages for adding/editing/removing members, sales records, and products as well as viewing reports and analytics.

Backend Layer (Application Logic and Functionality):

This layer handles the business logic of the system. It processes user requests from the frontend layer, interacts with the data storage layer (for data retrieval, insertion, and deletion), and performs actions such as predicting member grocery needs, generating reports, and managing sales data. It ensures that data flows correctly between the user interface and the data storage. User authentication and account management also reside in this layer.

Data Storage Layer:

This layer manages the storage of data. It includes database tables for members, sales records, and products. The layer is responsible for maintaining data integrity and ensuring accurate record-keeping.

Components, Roles, and Responsibilities:

Frontend Layer (User Interface):

- User Interface Components: Web pages for adding/editing/removing members, sales records, and products.
- User Authentication: Handles user login and authentication, ensuring secure access to the system.
- Account Management: Allows users to manage their accounts, change passwords, and update profile information.

Backend Layer (Application Logic and Functionality):

- Business Logic: Processes user requests, implements prediction algorithms, generates reports, and handles system functionalities.
- Integration: Interacts with the data storage layer to retrieve and store data as needed.
- Data Validation: Validates user inputs to ensure data accuracy and consistency.

Data Storage Layer:

- Database Tables: Stores members' details, sales records, and product information.
- Data Integrity: Enforces data integrity through proper validation and constraints.
- Query Processing: Handles database queries and transactions for efficient data retrieval and manipulation.

Conclusion:

Overall, the chosen multi-layer architecture design (see Figure 1) divides the member management system into three distinct layers that work together seamlessly. The Frontend layer receives input and user requests from the users and sends it to the Backend/Application layer where the data is analysed and processed. The Backend/Application layer then interacts with the Data Storage layer to insert, delete and/or retrieve data as required. The flow is then reversed from the Data Storage layer to the Application layer and lastly to the Frontend layer to present the outcome to the users. Ensuring that each layer has its own specific roles and responsibilities within a multi-layer architecture promotes a structured, organized, and efficient approach to software development. It leads to more a robust, maintainable, and scalable application that aligns well with modern software engineering practices while meeting the project objectives of enhancing store operations, accuracy in record-keeping, and providing an improved shopping experience for customers.

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

Members:

Enzo Peperkamp - 102895415

After thorough analysis and collaboration among team members, the chosen solution for GotoGro's system is the cloud-hosted Model-View-Controller (MVC) architecture, specifically as outlined by Alex. This decision was reached through a comprehensive KoST (Knowledge, Skill, Technology) analysis, validating its alignment with industry needs and business requirements. Opting for a serverless architecture made implementation easier, with less maintenance overhead, fitting the organization's small-scale context, which may lack resources for maintaining its own server infrastructure. Among the alternatives evaluated, including traditional monolithic applications and various server-based solutions, this serverless architecture emerged as the most suitable option. The decision reflects a well-tested response to a common industry problem, allowing for easy extensibility for potential future upgrades, and solidifying it as the optimal choice for GotoGro.

Nelchael Kenshi Turija - 103057559

I wholeheartedly support the team's use of the solution as it perfectly meets GotoGro's demands. Our team’s minimal resources are in line with the serverless nature of this method, and the modular design provides scalability and maintenance will be kept at a minimum. The project's value proposition is further strengthened by the attention to user experience through a React JS application and the implementation of predictive algorithms. This decision displays a carefully thought out plan that best meets Goto Grocery’s objectives and overcomes obstacles that may be presented.

Julian Codespoti - 102997816:

Having closely reviewed the analyses and recommendations put forth by my team members, I am in full agreement with our collective decision to adopt the cloud-hosted Model-View-Controller (MVC) architecture for GotoGro's system. Drawing from my experience with software development and system design, the MVC approach is both efficient and effective, especially when hosted in the cloud. The separation of concerns, inherent in the MVC structure, facilitates a streamlined development process, ensuring that GotoGro's objectives are met without incurring unnecessary complexities. Additionally, leveraging cloud capabilities not only offers reliability and scalability but also ensures cost-effectiveness. This aligns with our goal of ensuring that GotoGro achieves its operational objectives without being burdened by excessive infrastructure costs. I appreciate the rigorous process and collaboration that led to this conclusion, and I am confident that this choice will prove to be the cornerstone of GotoGro's success.

Alex Kyriacou - 103059830

I support the solution outlined below. While the final architecture closely represents the one I proposed individually, it still accurately represents the conclusions we came to as part of our group discussions. The architecture that is seen below is one that is best suited to the GotoGro brief while also considering both the team's strengths and weaknesses (as described in the KoST analysis). We

also ensured to follow architectural best practices as described within the lecture content. Examples of our discussions of the best solution can be seen within the alternatives section.

Marella Morad - 103076428

After extensive discussions with team members, we have concluded that a cloud-hosted Model-View-Controller (MVC) architecture (outlined by Alexander) is the optimal solution for Goto Gro's system. This choice aligns closely with my proposed solution, as both emphasize the separation of concerns into distinct components (i.e., Model, View, and Controller). The decision to opt for a cloud-hosted MVC architecture was driven by its dynamic scalability, enabling efficient performance during peak user loads—unlike the multi-layer architecture. Moreover, cloud hosting ensures inherent security measures and automated backups, directly aligning with the project's objectives of flexibility, security, and modernization.

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:

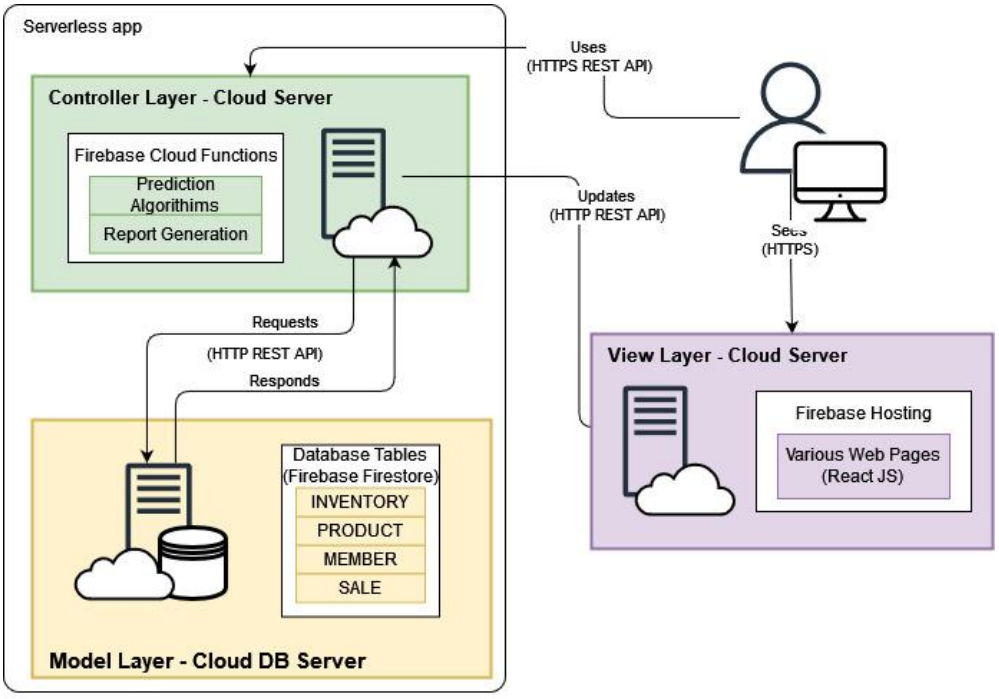


Figure 1: Architecture Of Agreed Solution Direction

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.

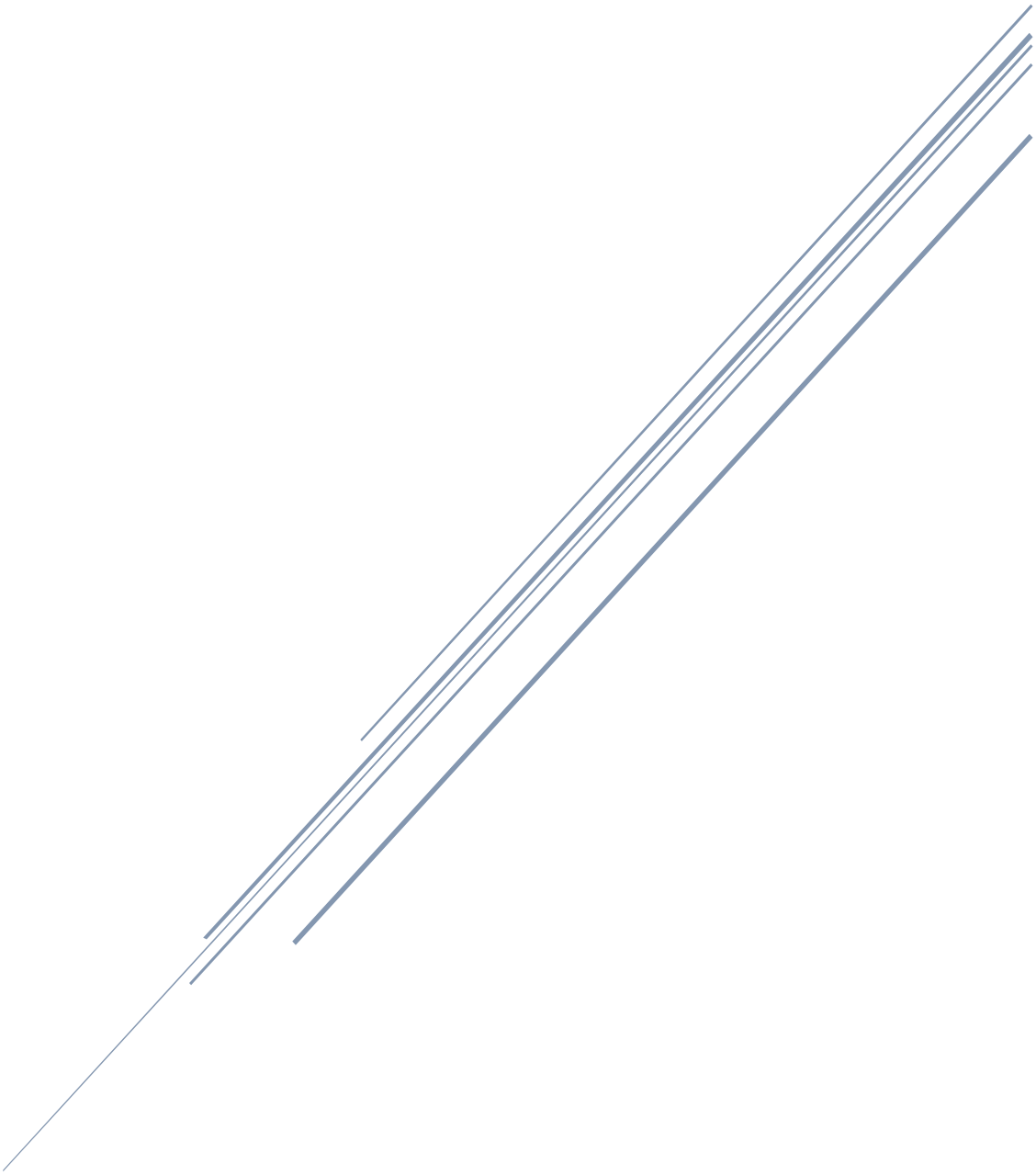
    - **Cloud-Based Infrastructure:**
        - While there's an experience gap in using cloud-based hosting, there is a commitment to learning and implementing basic cloud-based hosting for websites, databases, and servers.
- **Industry Experience:**
    - **Supermarket Retailer Insight:**
        - Experience 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.

PROJECT PROPOSAL: GOTOGRO-MRM

Definition of Done and Quality



SWE20001 – Managing Software Projects

Table of Contents

1 Quality Management:

The “Definition of Done” for this project outlines the conditions that need to be met to ensure that our proposed solution fulfils both the functional and non-functional (quality) requirements as specified in the brief. The following 10 conditions, in alignment with the product quality model (Standards Australia/New Zealand, 2013), provide a more detailed definition of the "Definition of Done".

1.1 Functional Suitability:

1.1.1 Functional Correctness:

1.1.1.1 Number of Defects per KLOC (Thousand Lines of Code):

This metric quantifies the number of defects or bugs found in the codebase of the software per thousand lines of code. The threshold value is set to a maximum of **10 defects per KLOC**. This threshold value considers the trade-off between achieving a reasonable level of quality while acknowledging potential time and resource limitations (such as developers’ experience).

Example: After reviewing the codebase, it was determined that there were 5 defects found in 800 lines of code. This translates to 6.25 defects per KLOC, which falls within the acceptable threshold of 1-10 defects per KLOC.

1.1.2 Functional Completeness:

1.1.2.1 Coverage of Specified Tasks and User Objectives:

This metric evaluates the extent to which the set of functions implemented in the software addresses all the tasks and objectives specified in the project requirements. The threshold value is set to a **minimum of 80%**. Choosing the threshold of 80% provides flexibility and recognizes that software projects frequently involve iterations and potential improvements. This well-balanced approach ensures that the software accomplishes a majority of the intended tasks and objectives, while also permitting refinements and enhancements in subsequent iterations.

Example: The requirements include tasks like registering members, recording sales, managing products, and generating reports. During testing, it was found that the software effectively handles 4 out of 5 tasks, with minor UI improvements needed for product management. The software then meets the above threshold of 80% functional completeness.

1.2 Performance Efficiency:

1.2.1 Time Behaviour:

1.2.1.1 Response and Processing Times:

This metric assesses the speed at which the system responds to user actions and processes tasks. The threshold value is set at **2 seconds**. If the system responds to user actions within this time frame, it meets the quality metric.

Example: One of the key functionalities in GotoGro is adding a new member to the database. This involves capturing member information, validating it, and storing it in the database. The response and processing times for this action are critical to ensure a smooth user experience and efficient member management. After a user has entered the new member's information, the system verifies the input, checks for duplicate entries, and stores the new member's information in the database. If the system responds to the staff member's action and completes the entire process of adding a new member within 2 seconds, then this metric is met.

1.2.2 Resource Utilization:

This metric monitors the utilization of system resources such as CPU and memory. The threshold value is set at **70%**. If the system's resource utilization remains below 70% under normal load conditions, it meets this quality metric.

Example: when generating detailed inventory reports and sales analyses, the system's resource utilization needs to be monitored closely to guarantee that report generation is swift and doesn't hinder other critical system functionalities.

1.3 Usability

1.3.1 User Protection Error:

1.3.1.1 User Task Success Rate:

This metric evaluates the percentage of user tasks that are completed successfully without errors or assistance. The threshold value is set at **95% or higher**. This entails providing clear instructions and easy-to-fill forms for registration. If at least 95% of user tasks can be completed successfully by users without errors or difficulties, the software meets this quality metric.

Example: Out of 100 attempts to update member information, if at least 95 of them are completed successfully without any errors or the need for assistance, the software meets the quality metric.

1.3.1.2 Time to Complete Common Tasks:

This metric measures the time taken by users to complete common tasks within the user interface. The threshold value is set at **1.5 times the average time taken by experienced users**. If users can complete common tasks within this threshold, the software meets this quality metric.

Example: Suppose the average time taken by experienced users to create a new sale record is 2 minutes. In this case, the threshold value would be 1.5 times this average, which is 3 minutes. If most users can complete the sale record creation process within 3 minutes, the software meets the quality metric.

1.4 Reliability:

1.4.1 Availability:

1.4.1.1 System Uptime and Downtime:

This metric tracks the amount of time the system is operational (uptime) and the amount of time it is unavailable (downtime). The threshold value is set to **95% uptime**. The system should be operational and accessible for at least 95% of the time.

1.4.1.2 Mean Time to Recovery (MTTR):

This metric evaluates the average time it takes to restore the system to full functionality after a failure. The threshold value is set at **1 hour**. If the system can be fully restored within 1 hour after a failure, it meets this quality metric.

Example: If a technical glitch makes GotoGro's interface unresponsive, staff and members can't access the system, disrupting member management and grocery services. To meet the MTTR metric, GotoGro's tech team must quickly diagnose and fix the issue, restoring full system functionality within an hour, to achieve this quality metric.

1.5 Security:

1.5.1 Integrity:

1.5.1.1 Vulnerabilities Detected:

This metric identifies critical vulnerabilities found in the software through security assessments. The threshold value is set at **zero** (keeping in mind the size of this system, there shouldn’t be any critical vulnerabilities). If no critical vulnerabilities are detected in the software, it meets this quality metric.

Example: For GotoGro's project, a security assessment is conducted to identify vulnerabilities in the application. During the assessment, the team uses a reputable security scanning tool to analyse the application's codebase and dependencies. The assessment reveals one vulnerability with high severity. This vulnerability is promptly addressed and patched by the development team. Upon second

assessment, if it's found that there are no vulnerabilities, then the system passes this quality metric, otherwise, the development team needs to do more patching.

1.5.2 Confidentiality:

1.5.2.1 Data Encryption:

This metric ensures that sensitive data is encrypted both when at rest and during transmission. A reasonable threshold value is met if **sensitive data is encrypted during transmission** (e.g., using HTTPS) and stored using industry-standard encryption practices.

Example: To ensure the security of sensitive member data, GotoGro's application employs encryption practices. When members submit their information or make transactions, the data is encrypted during transmission using HTTPS, providing a secure connection between the user's browser and the server. Additionally, sensitive data, such as member details and sales records, is stored in the database using industry-standard encryption techniques to prevent unauthorized access. This approach ensures that sensitive information remains confidential and secure throughout its lifecycle.

2 References:

Standards Australia/New Zealand, 2013, *Systems and software engineering - Systems and software quality requirements and evaluation (SQuaRE) - System and software quality models,* AS/NZS ISO/IEC 25010:2013, Sydney/Wellington: Standards Australia/New Zealand, viewed 18 August 2023.

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

Members:

Enzo Peperkamp - 102895415

I strongly agree with the emphasis on Security in the GotoGro-MRM proposal, particularly the 2-factor authentication, aligning with ISO27001 standards. The Performance Efficiency goals, including 60% resource utilization, seem well-balanced. However, I'd recommend a closer look at the proposed average time for system restoration, as 1 hour may be ambitious.

Nelchael Kenshi Turija - 103057559

This document's coherent structure and unified approach reflects our collaborative efforts, skillfully combining individual tasks to eliminate redundancies. The alignment with both our team's goals and industry standards is commendable. I agree wholeheartedly with the presented framework, confident that it will guide us to successful implementation without compromising on quality or efficiency.

Julian Codespoti - 102997816:

The synergy of our collective input is evident in this proposal. By aligning our individual perspectives, we've produced a comprehensive guideline that ensures both functionality and quality. I am confident this will serve as a solid road map for our project's success.

Alex Kyriacou - 103059830

I believe the below document represents a solid foundation for the 2 following sprints. It consists of a measurable framework for each of our team members to be held to on completion of their assigned sprint tasks. While individually there were gaps within our definition of done and quality model, it was through compilation of our ideas that we have the model seen below.

Marella Morad - 103076428

This document encapsulates our shared vision and commitment to excellence. Drawing from our diverse skill sets, we've created a roadmap that possesses a large deal of both quality and functionality. I am optimistic about the milestones we've set and am confident in our team's ability to execute them seamlessly.

Table of Contents

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.

Reference

Standards Australia/New Zealand, 2013, *Systems and software engineering - Systems and software quality requirements and evaluation (SQuaRE) - System and software quality models,* AS/NZS ISO/IEC 25010:2013, Sydney/Wellington: Standards Australia/New Zealand, viewed 20 August 2023.

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:

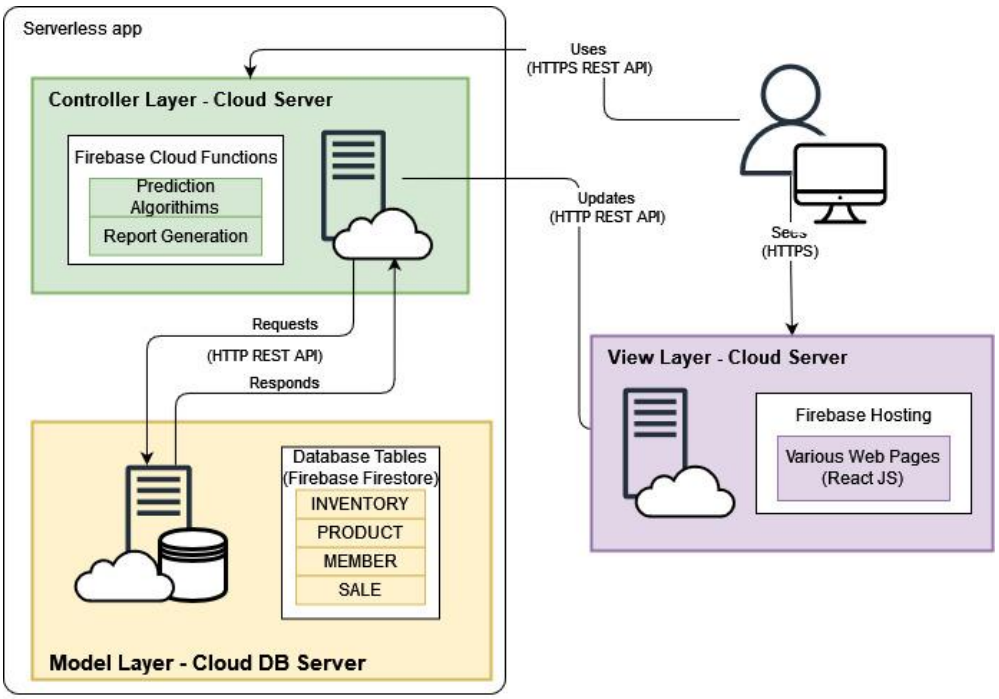


Figure 1: Architecture Of Agreed Solution Direction

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**

Sprint 1 planning: Management System (GotoGro-MRM) for Goto Grocery Inc.

Task 1:

Included factors:

Feature Dependency

This describes any dependencies that a sprint backlog item has. An example of this would be the backlog item of deleting users. This has a feature dependency of Adding new users as it is impossible to delete users if you cannot yet add them.

Why this factor is important

This is a critical factor for this project as without it we have no visibility as to if a backlog item is ready to be worked on. By prioritising feature dependency, we can easily compare and prioritise backlog items.

Development Effort

Development effort encompasses the time, skills, and resources necessary to complete a project. It relies on factors like task complexity and team expertise and plays an essential role in shaping project plans and sprint schedules.

Why this factor is important

Accurate development effort estimation is important for effective resource allocation, realistic timeline setting, and the fair distribution of tasks based on team members' experience and capacity. It ensures that projects are approached with clarity and foresight, reducing risks, enhancing project outcomes, and fostering a more efficient and sustainable development process.

Business Value

This represents a backlog item's overall value to the business, its assets, brand recognition, goodwill and operations. Examples of a backlog item that would be considered to have high business value include:

- A feature that would give the product a competitive edge
- A fix for a bug that customers have been complaining about for some time

Why this factor is important

This is an important factor to consider when prioritising backlog items. By considering the value to the business it ensures that our work stays tightly in scope and focused on the core deliverables of the project. Only once with high value to the business are completed does it make sense to prioritise more quality of life features.

Not-Included Factors:

Date Needed or Timeline

This describes prioritising backlog items depending on the relevant times they are required. If a client required a certain feature by next month then this would be prioritised first.

Why is this not included?

We decided that this was not distinct enough for our project brief to be identified as a seperate factor. This is because of its similarity with the feature dependency factor. It is very likely that the dates needed for a certain backlog item directly correlate with that item's dependencies.

Risk involved

This refers to the potential issues or threats associated with implementing a backlog item. Risk can arise from various factors such as technical challenges, security vulnerabilities, operational disruptions, or factors external to the project like market changes. Understanding the risks associated with each item allows the team to plan and allocate resources more efficiently, ensuring smooth delivery.

Why is this not included?

- ➔ **Internal Nature of the Project**: Given the project is internal, the risk exposure is inherently low. The system isn't outward-facing, which means it isn't a high-priority target for potential external threats. This allows us to work with a certain level of ease without being overly concerned about external vulnerabilities.
- ➔ **Security/Privacy**: Thanks to our reliance on trusted entities with recognized certifications such as ISO/IEC 27001, ISO/IEC 27017, ISO/IEC 27018, ISO/IEC 27701, SOC 2, and SOC 3, we are confident in the robust security and privacy measures in place. These measures align with industry best practices, as outlined in the Google Cloud security overview whitepaper (Google Cloud, n.d.). This foundation negates the need to further emphasise or allocate resources towards this factor.
- ➔ **Task Dependency at Runtime**: When our system goes live and enters its runtime phase, most tasks are built to function independently. Their modular nature ensures smooth operations, with minimal risk of one task's failure or delay affecting the rest. This lack of interdependence at runtime is a significant advantage, ensuring that our system can remain robust and functional even if isolated issues arise.

- ➔ **Stable Development Environment**: Our team operates in a stable development environment with established tools, protocols, and a skilled team. This environment minimises the risk of unforeseen technical challenges or disruptions, further ensuring a smooth developmental progression.

Other factors

Why are other factors not considered?

- ➔ **Simplicity and Clarity**: By focusing on just the three main factors, we can ensure that our ranking system remains simple and clear. Introducing more factors would not only complicate our decision-making process but could also create confusion and disputes among team members about the relative importance of each factor.
- ➔ **Defined Scope**: Our project has a very defined and limited scope. As such, introducing more factors might not be entirely relevant or beneficial. It's essential to maintain a clear vision and direction, and not get sidetracked by factors that, while they might be important in other contexts, do not necessarily add value in the context of our specific project.
- ➔ **Efficiency in Decision Making**: Every additional factor we consider requires time to assess, debate, and weigh against others. By limiting our focus to three primary factors, we streamline our decision-making process, allowing the team to quickly and efficiently prioritise backlog items.
- ➔ **Avoiding Paralysis by Analysis**: Too many factors could lead to over-analysis, where the team spends more time debating the importance of backlog items than actually working on them. In essence, we're avoiding the pitfall of being "stuck in planning" and ensuring that we move into the action phase more seamlessly.
- ➔ **Uniformity in Understanding**: With fewer factors to consider, there's a higher chance that all team members have a uniform understanding of what's important. This uniformity ensures everyone is on the same page, reducing the likelihood of misunderstandings or disagreements later on.

Task 2:

We have decided that the criteria for prioritising the backlog items will occur in the following order:

1. Feature Dependency
    - Defined by future backlog items dependencies on the completion of this item
2. Development Effort
    - Defined by the estimated number of hours of developer effort required to complete a backlog item to the standard defined within the agreed DoD
3. Business Value
    - Defined by a backlog items relevance to the defined scope of the project and stakeholders demand for its completion

It is important to note that the ordering of these factors is specific to the project's scope, goals and stakeholders. In another project, such factors may be omitted or be prioritised differently. Our team

decided that the dependencies that a particular backlog item has outweighs the other factors listed. This is due to the greenfields nature of the project resulting in many backlog items having large dependencies on each other. For example, due to the lack of existing infrastructure to build on, the team will have to carefully prioritise the first sprint to ensure there are minimal blockers throughout while completing enough to unlock backlog items for future sprints.

Development effort has been prioritised second. Due to the small size of the team it is crucial that we effectively manage the little development time we have available. As such, the team will have to thoroughly evaluate the time commitments of each of the backlog items to see if the time should be better spent elsewhere. This was prioritised underneath feature dependencies. This decision is a natural choice considering a backlog item with many dependencies would need to be prioritised regardless of the development effort it requires. In other words, it would be required to do the 'hard work' first in order to unblock the higher value items later on.

Lastly we have prioritised business value as a third factor. This acts as a litmus test for any items that are deemed to be equivalent for all the preceding factors. Where two items are equivalent, the team will discuss the value a given backlog item aims to deliver to the overall vision of the project. This prevents the team from working on anything that may have feature dependencies but ultimately of little value to the overall project. This has been prioritised last due to the delivery of this project being a single large showcase rather than iterative delivery of features. If features were delivered iteratively business value would make sense to prioritise higher as the goal of the sprints would be to get a minimum viable product working as quickly as possible. However, given this is not the case, the team has the ability to spend time instead laying solid foundations in the unblocking of dependencies, enabling fast growth in later sprints

Task 3:

Ranking	Layer	Epic		No.	Item	Feature Dependency	Development Effort	Business Value	Estimated points*
1	Back-end	DBMS	Create database tables	F1	Create database table for members	★★★★★ (Core foundation feature)	★★★★☆ (High data sensitivity)	★★★★★ (Digitizes member records.)	3
2				F2	Create database table for sales records	★★★★★ (Core foundation feature)	★★★★☆ (High data sensitivity)	★★★★★ (Tracks member purchases.)	3
3				F3	Create database table for products	★★★★★ (Core foundation feature)	★★★★☆ (High data sensitivity)	★★★★★ (Centralizes product inventory.)	3
4			Member API endpoints	F4	Create endpoint for adding members	★★★★☆ (Vital for user registration)	★★★☆☆ (Standard CRUD operation)	★★★★☆ (Simplifies member registration.)	3
5				F5	Create endpoint for editing members	★★★★☆ (Crucial for data accuracy)	★★★☆☆ (Standard CRUD operation)	★★★★☆ (Enhances member data management.)	3
6				F6	Create endpoint for removing members	★★★★☆ (Necessary for data management)	★★★☆☆ (Standard CRUD operation)	★★★★☆ (Supports member removal.)	3
7				F15	Create endpoint for retrieving members	★★★★☆ (Key for accessing member info)	★★★☆☆ (Standard CRUD operation)	★★★★☆ (Fetches member information.)	3
8			Sales API endpoints	F7	Create endpoint for adding sales records	★★★★☆ (Vital for sales tracking)	★★★☆☆ (Standard CRUD operation)	★★★★☆ (Streamlines sales data entry.)	3
9				F8	Create endpoint for editing sales records	★★★★☆ (Crucial for data accuracy)	★★★☆☆ (Standard CRUD operation)	★★★★☆ (Facilitates sales record edits.)	3

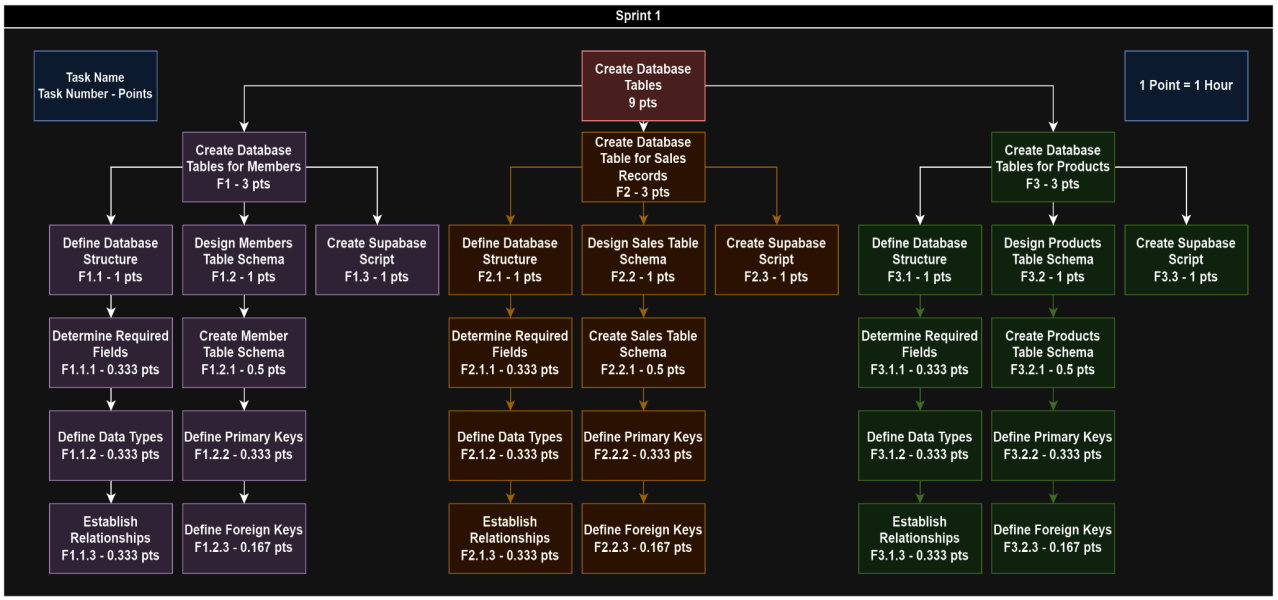
10				F9	Create endpoint for removing sales records	★★★★☆ (Necessary for data management)	★★★☆☆ (Standard CRUD operation)	★★★★☆ (Manages sales record deletion.)	3
11				F14	Create endpoint for retrieving sales	★★★★☆ (Key for sales insights)	★★★☆☆ (Standard CRUD operation)	★★★★☆ (Accesses sales data.)	3
12			Product API endpoints	F10	Create endpoint for adding products	★★★★☆ (Supports inventory growth)	★★★☆☆ (Standard CRUD operation)	★★★★☆ (Enables product additions.)	3
13				F11	Create endpoint for editing products	★★★★☆ (Ensures product data accuracy)	★★★☆☆ (Standard CRUD operation)	★★★★☆ (Assists in product edits.)	3
14				F12	Create endpoint for removing products	★★★★☆ (Maintains product integrity)	★★★☆☆ (Standard CRUD operation)	★★★★☆ (Aids product removal process.)	3
15				F13	Create endpoint for retrieving products	★★★★☆ (Key for viewing product range)	★★★☆☆ (Standard CRUD operation)	★★★★☆ (Retrieves product listings.)	3
16		Report and Analytics		F17	Generate sales reports	★★★★☆ (Provides sales overview)	★★★☆☆ (Complex data aggregation)	★★★★☆ (Provides insights & CSV compatibility.)	8
17		Report and Analytics		F32	Generate inventory reports	★★★★☆ (Vital for inventory management)	★★★☆☆ (Data gathering & formatting)	★★★★☆ (Provides insights & CSV compatibility.)	8
18		User Authentication		F18	User authentication and account management	★★★☆☆ (Key for secure access)	★★★★★ (High complexity due to security concerns)	★★★★☆ (Secures user access.)	8

19	Front-end	Member management web page	F33	Build web page to allow users to add new members	★★★☆☆ (Enhances member addition process)	★★★★☆ (Front-end design & functionality)	★★★★☆ (Enhances member management UX.)	5
20			F34	Build web page to allow users to edit existing members	★★★☆☆ (Ensures data accuracy & usability)	★★★★☆ (Front-end design & functionality)	★★★★☆ (Enhances member management UX.)	5

*1 point = 1 hour

Task 4:

Figure 1 - Work Breakdown Structure



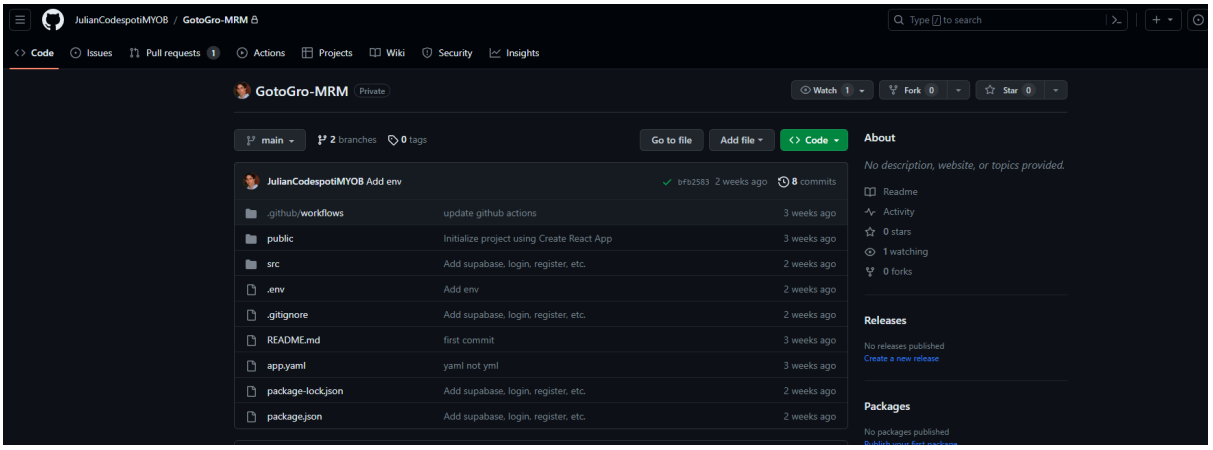
References:

Google Cloud, n.d. *Google security overview*. [online] Available at: https://cloud.google.com/docs/security/overview/whitepaper [Accessed 4 September 2023].

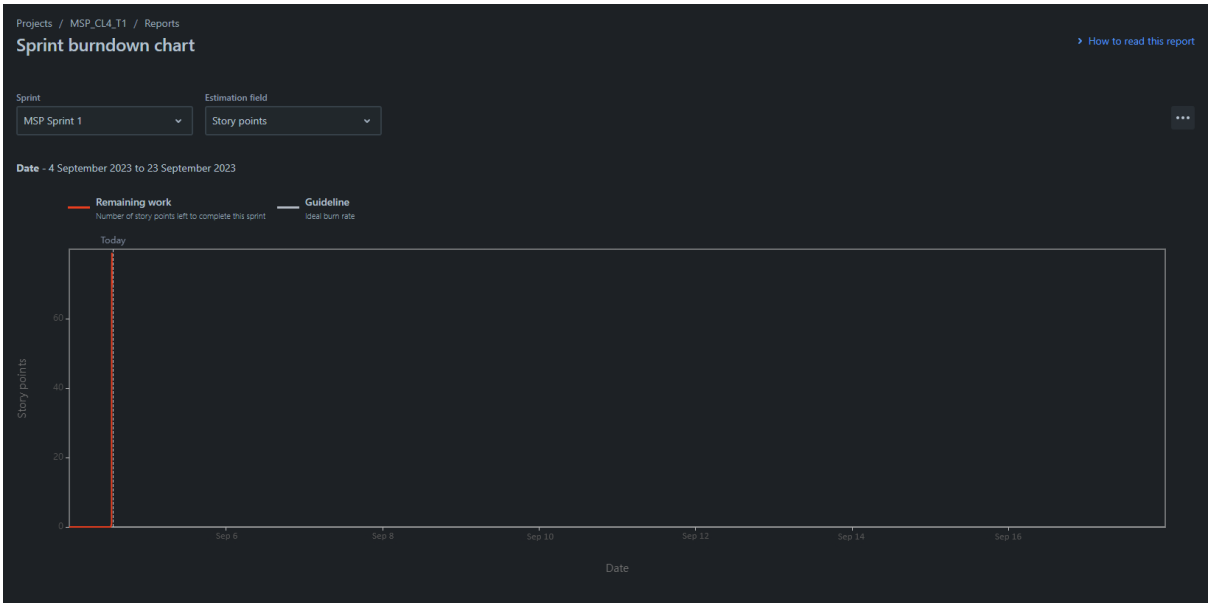
09P Project Initiation

Members	
Name	**ID**
Enzo Peperkamp	102895415
Nelchael Kenshi Turija	103057559
Julian Codespoti	102997816
Alex Kyriacou	103059830
Marella Morad	103076428

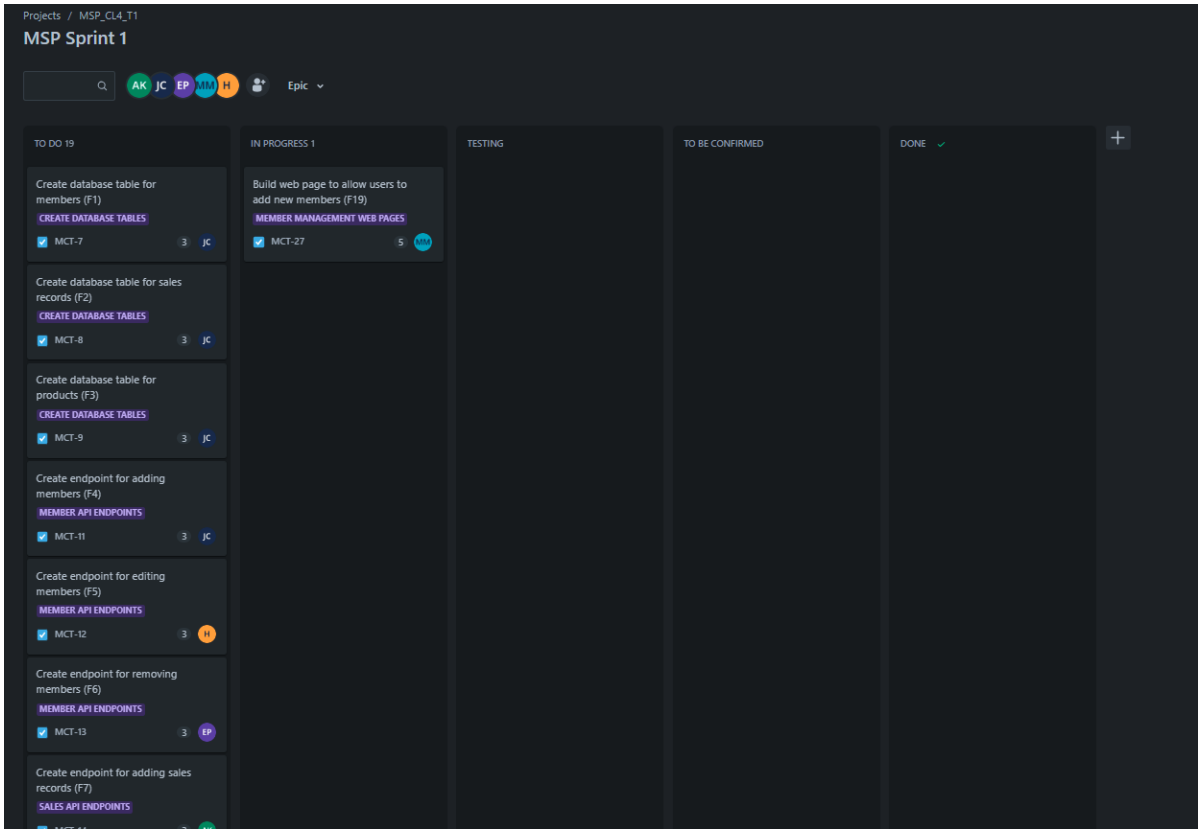
Github Repo Initial Screenshot



Sprint Burndown Chart Initial Screenshot



Task Board Initial Screenshot



10P MidWeek Check - Group

Team: MSP_CL4_T1	
Name	**ID**
Enzo Peperkamp	102895415
Nelchael Kenshi Turija	103057559
Julian Codespoti	102997816
Alex Kyriacou	103059830
Marella Morad	103076428

Table of Contents

Task Boards:

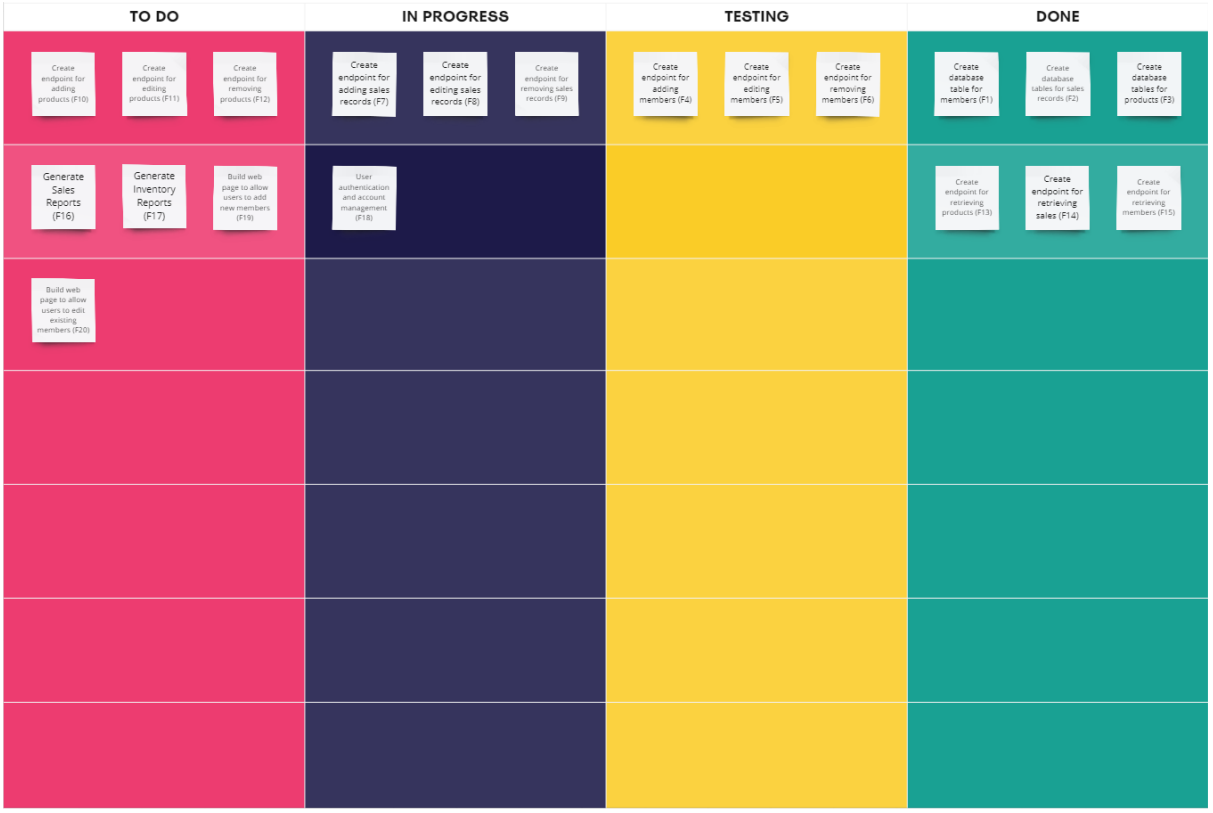
In our first attempt to use Jira for Sprint 1, we faced some challenges since we were new to the platform. This resulted in a delay in recording our work for the first few days and a missing week on our burndown chart. To make sure our progress was accurately reflected, we manually recreated what happened in the first week of Sprint 1. This involved taking screenshots of a manually maintained task board since our Jira task board was not a true reflection. We did this to ensure that our reports show a true picture of our work and progress.

Date: 04/09/2023

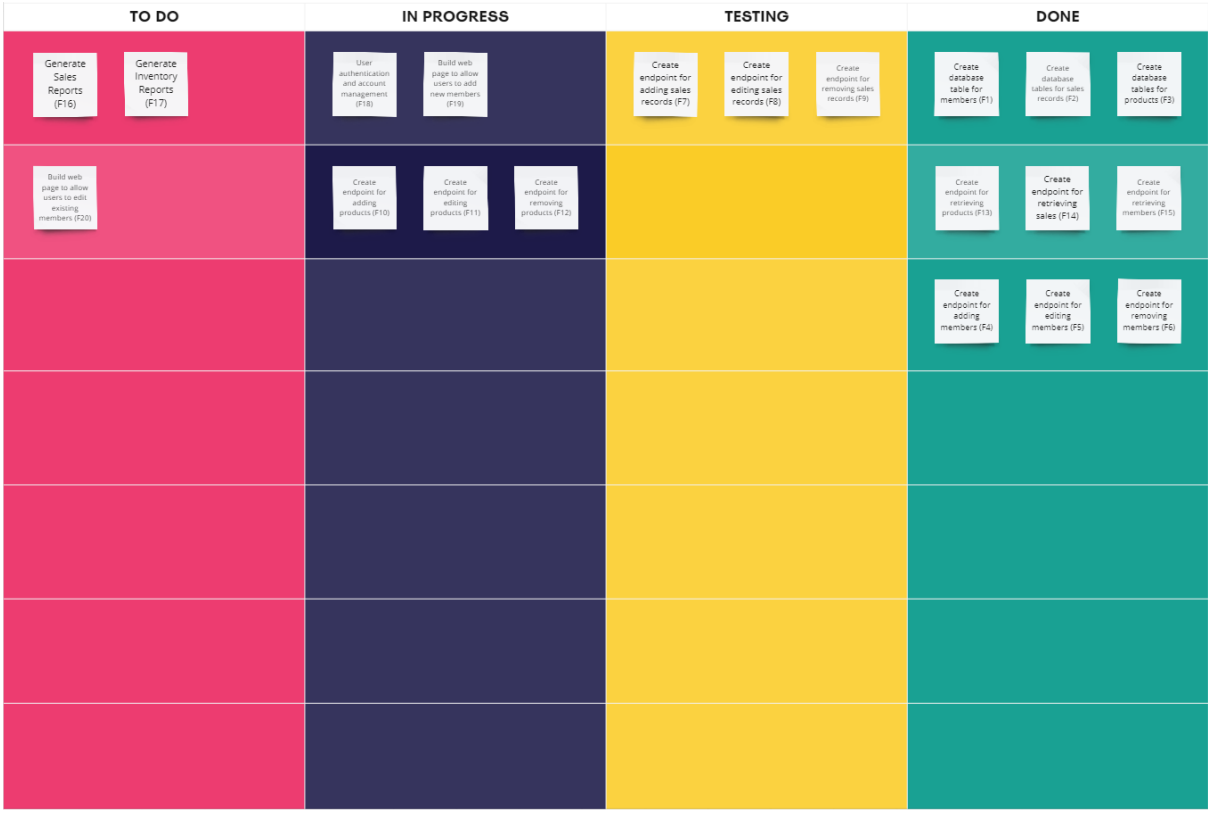
Date: 05/09/2023

Date: 06/09/2023

Date: 07/09/2023



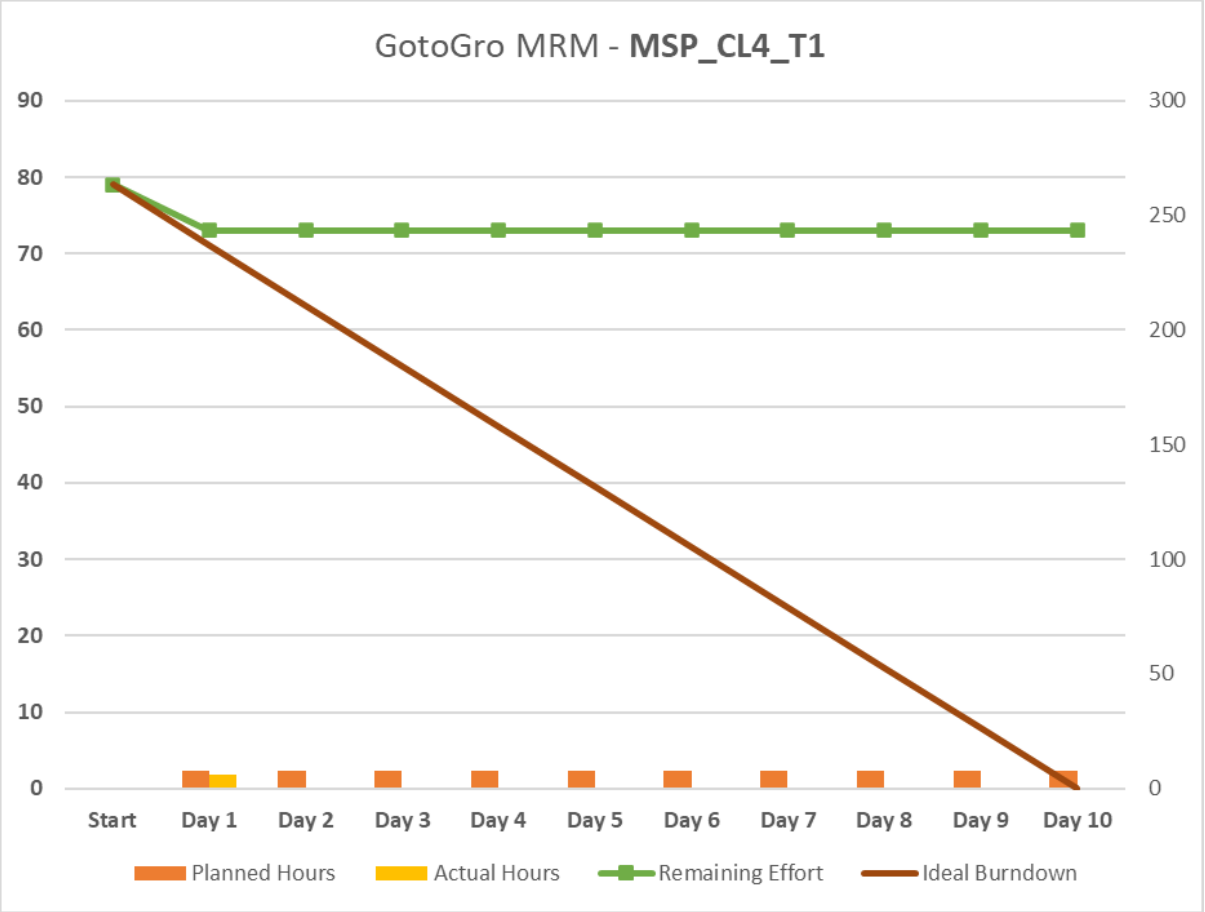
Date: 08/09/2023



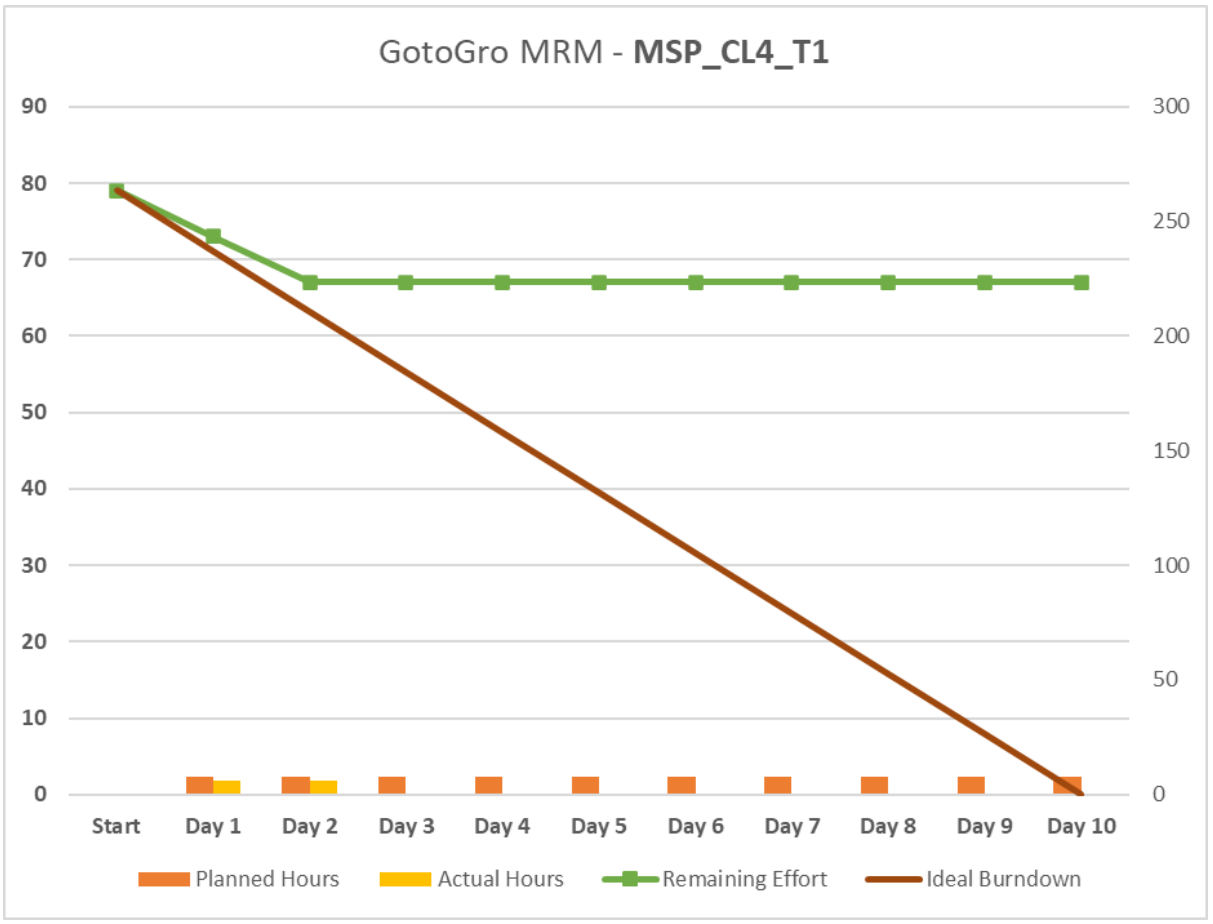
Burn-down Charts:

We opted to manually generate our burndown chart for Sprint 1 rather than relying solely on the auto-generated one provided by Jira. Our decision stems from the learning curve we experienced with the platform during our initial setup, which led to a delay in recording the first week's data accurately. By manually constructing the burndown chart, we were able to incorporate the missing data and ensure a more precise representation of our progress. This approach allowed us to account for the initial hiccups and maintain transparency in our reporting, providing a clearer and more accurate reflection of our team's achievements during the sprint.

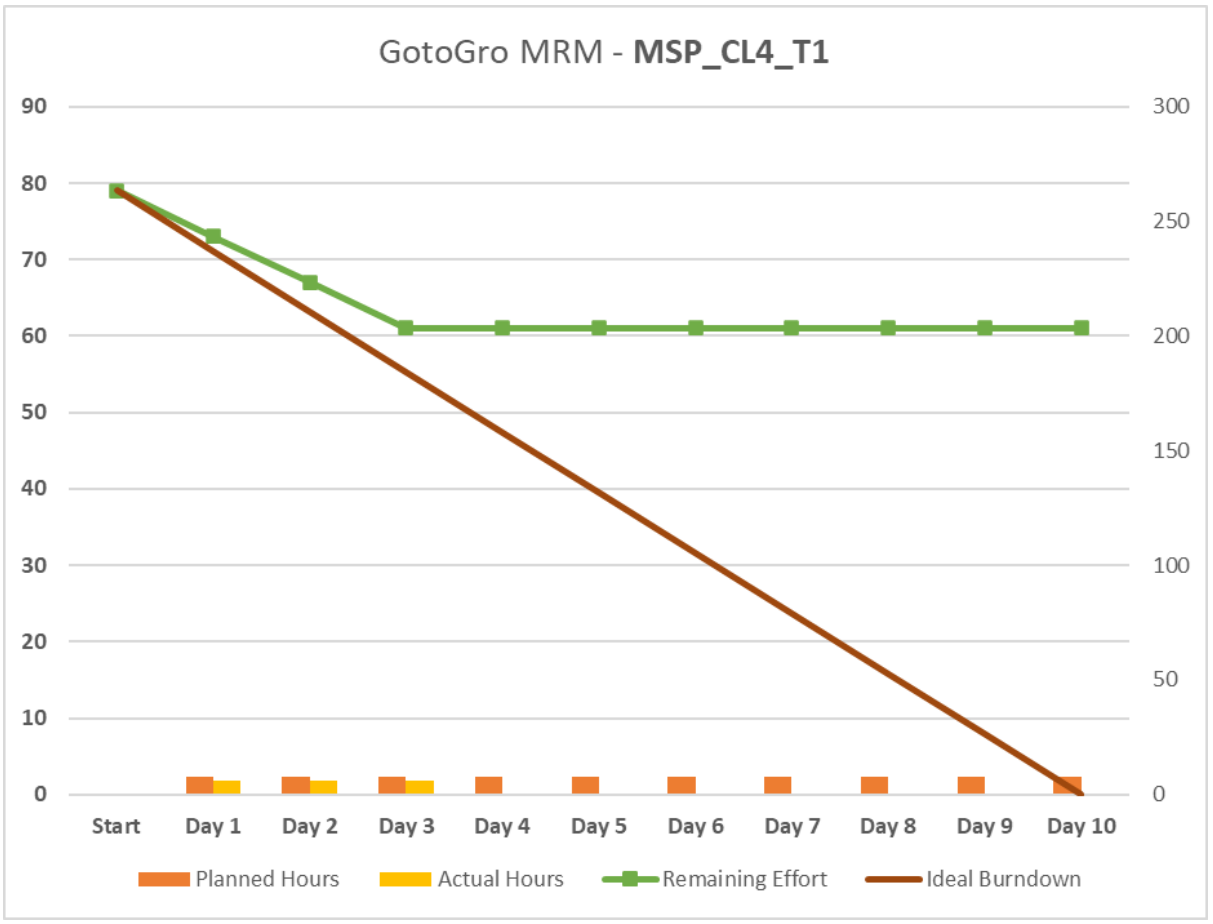
Date: 04/09/2023



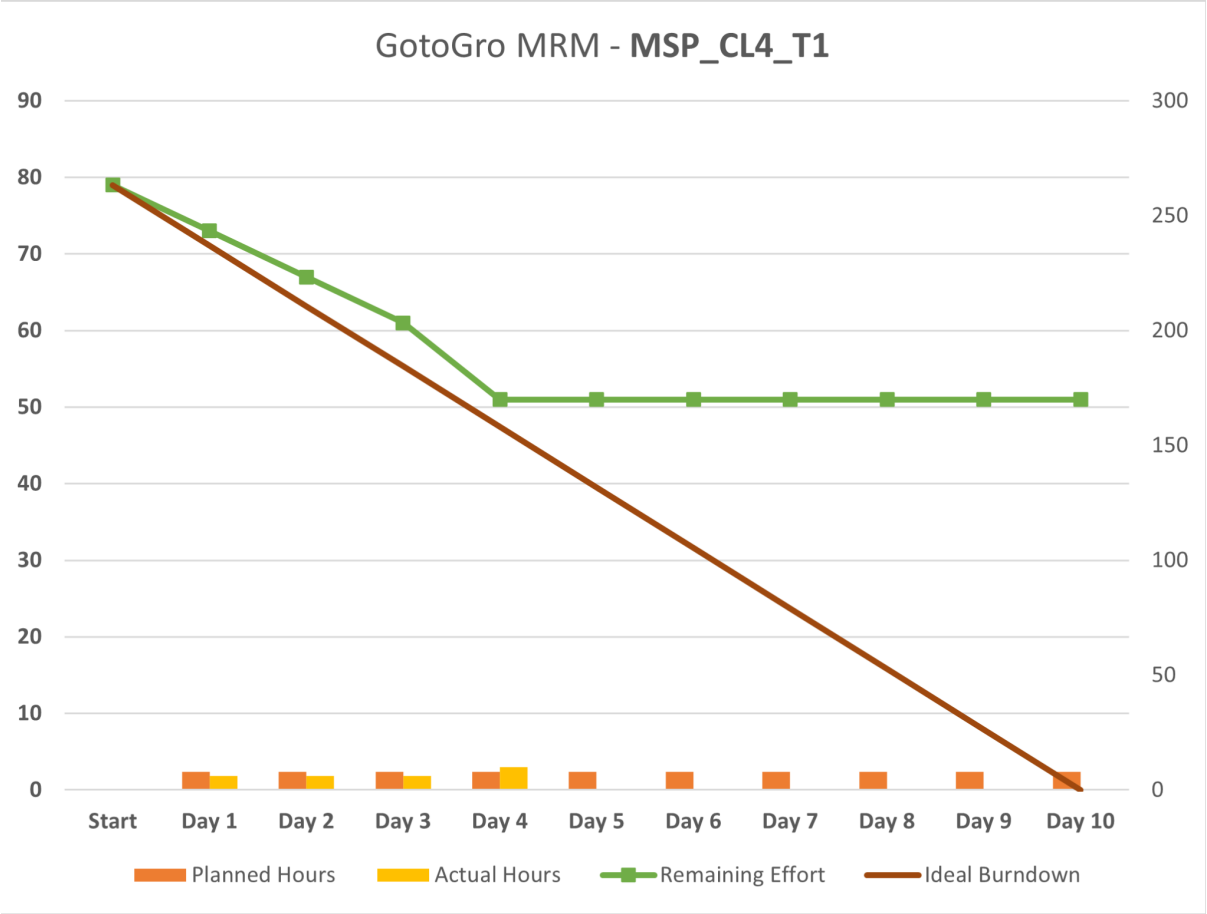
Date: 05/09/2023



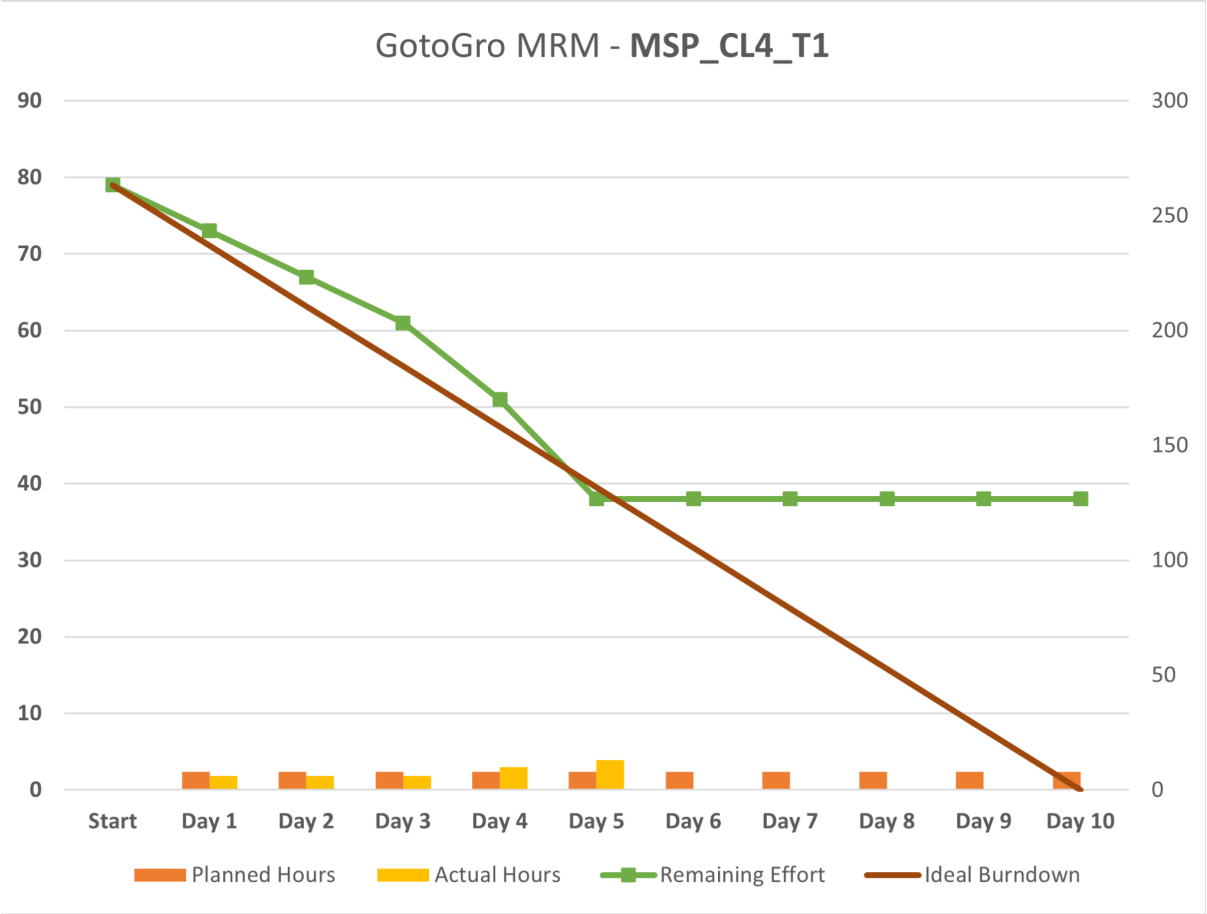
Date: 06/09/2023



Date: 07/09/2023

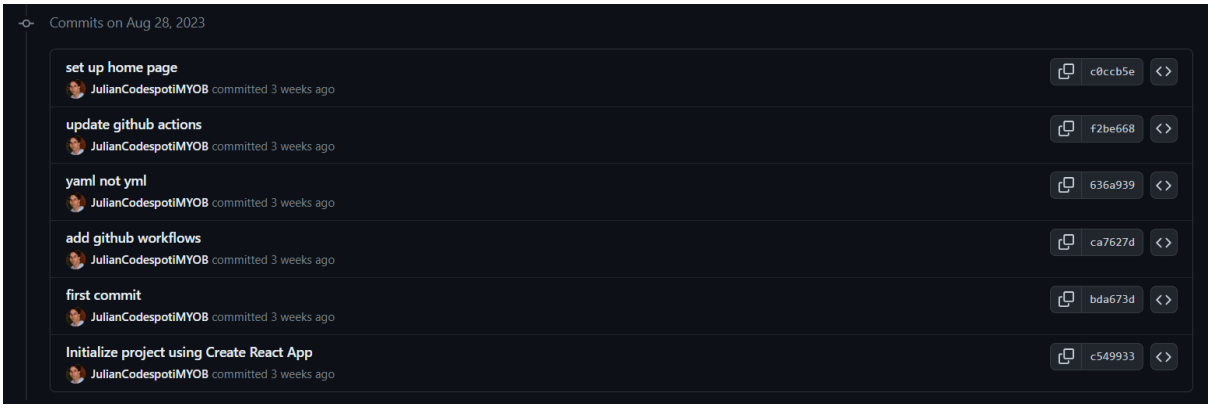


Date: 08/09/2023

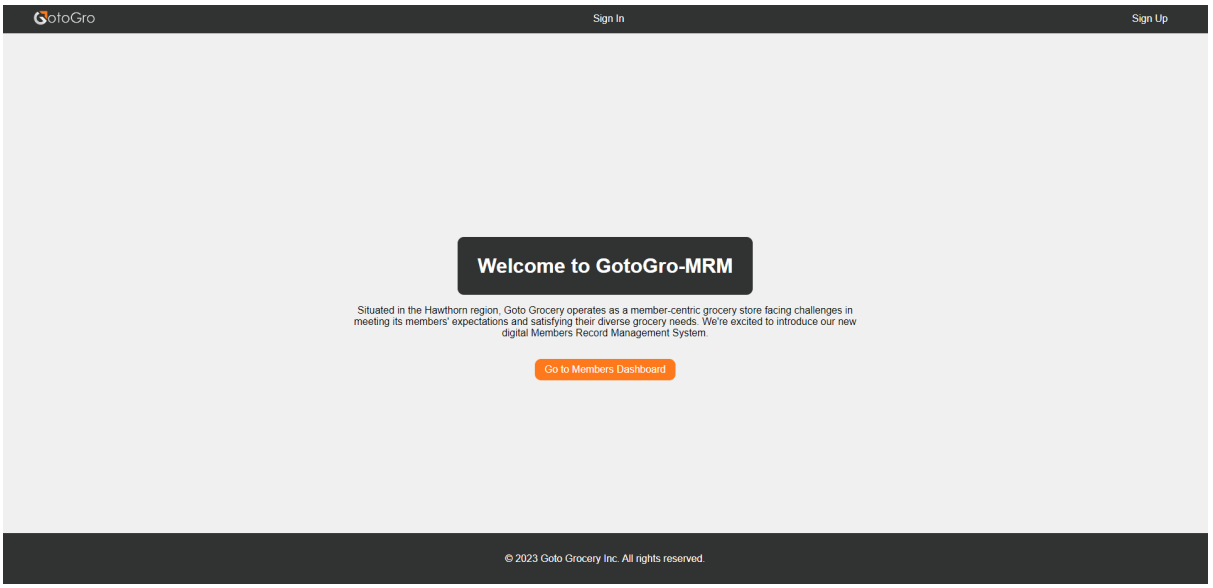


Project Repository Status:

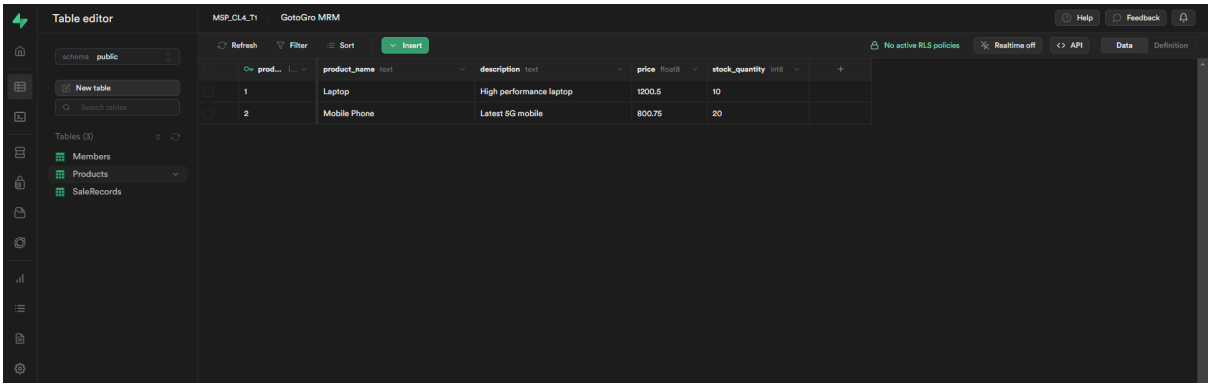
Before the official commencement of Sprint 1, we took the initiative to lay the foundation for our project. On August 28th, we created our GitHub Repository, initialised our web app using Create React App as we had discussed earlier and included it in our proposal. We then created a placeholder homepage to test app deployment.



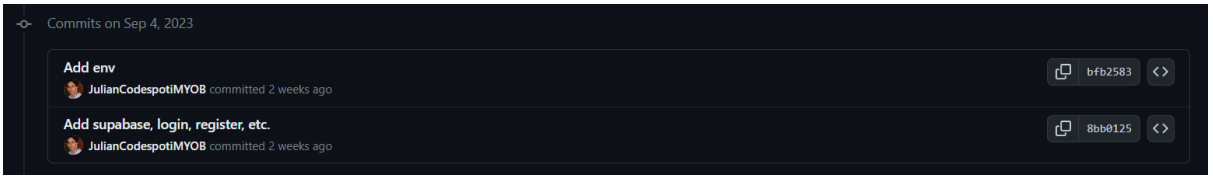
Furthermore, we streamlined our project deployment process by integrating Google Services with our workflow. We established a seamless connection between our GitHub repository and Google Services, enabling automatic deployment each time a merge into the 'main' branch occurred. Our app is accessible through this link - https://msp-cl4-t1.ts.r.appspot.com/



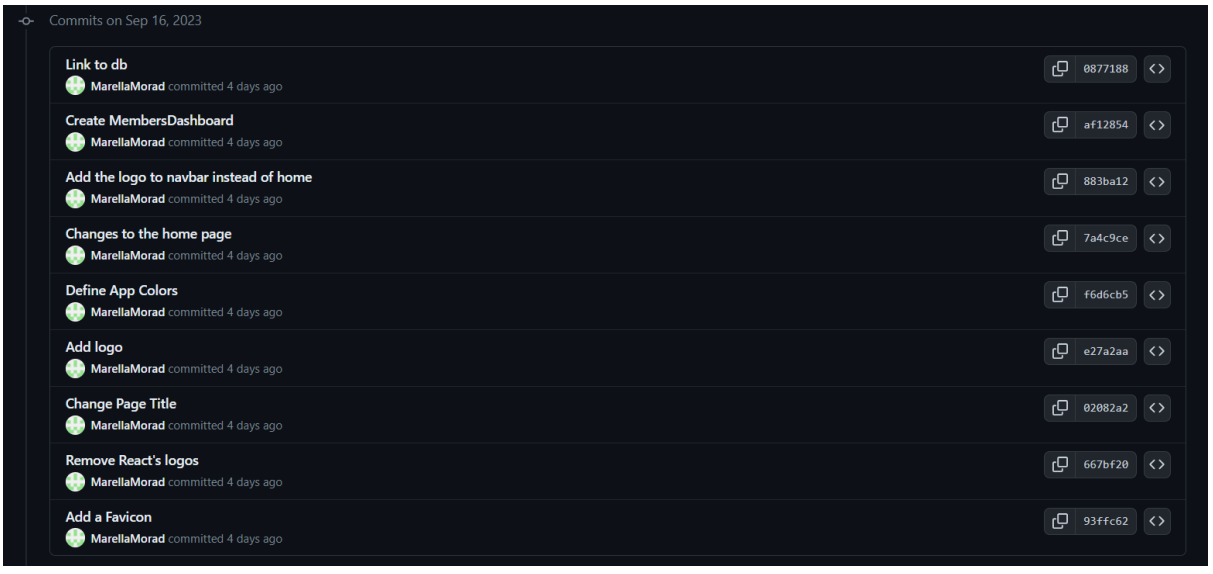
Subsequently, we conducted extensive research to identify the most suitable database solution for our project needs. After a comprehensive evaluation, we strategically chose to implement Supabase as our database solution. Supabase's robust feature set provided us with accelerated development capabilities, especially when it came to implementing endpoints. It allowed for seamless integration of SQL syntax directly into our React application, which streamlined our interactions with the database. This efficient setup enabled by Supabase empowered us to allocate more time to testing and implementing the user interface, aspects that would have otherwise been compromised due to time spent on endpoint configuration. This has proven to be a substantial time-saver in Sprint 1 and promises to continue benefiting our project moving forward.

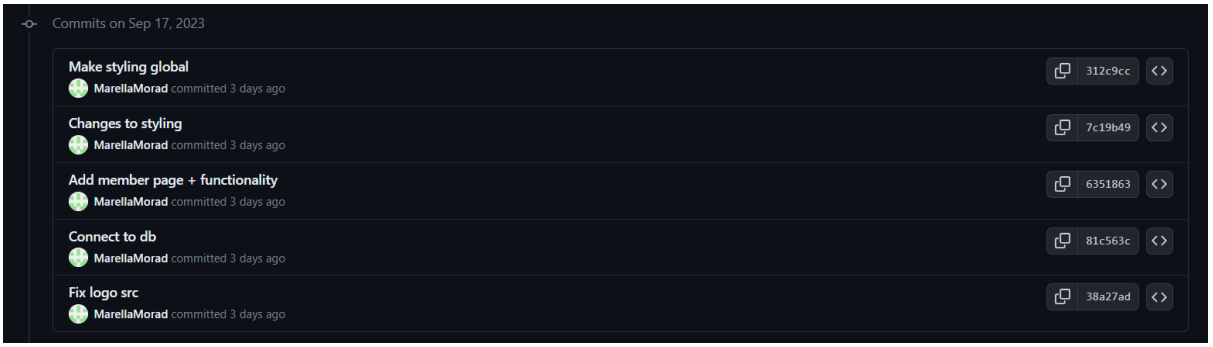


With the successful setup of our repository, databases, and initial application, we managed to overcome all the previously encountered blockers. This cleared the path for us to dive headfirst into the development phase of our web application. Having these foundational elements in place provided us with a solid starting point, allowing us to concentrate our efforts on crafting and enhancing the features and functionalities of our application without further impediments.



Acknowledging the extra week provided by the mid-semester break, we decided to utilize this time for further development of our application and to account for any time lost during our first week of Sprint 1.





In the commit history of our GitHub repository, you'll mainly see contributions from two team members. However, this doesn't capture the collaborative nature of our project. We frequently work together to overcome challenges, particularly after team meetings where we discuss any roadblocks.

Each team member has a unique approach to tasks, affecting the timing of their contributions in the commit history. For example, some tasks, such as compiling reports, appear later to ensure that only complete, error-free features are merged.

Not all crucial tasks result in GitHub commits. Important activities like service setup and database design might not be immediately visible but are integral to our project. Similarly, preparatory tasks like endpoint testing have allowed us to create a library of reusable code. While this groundwork may not appear in the commit history, it enables us to work more efficiently in the future.

In summary, although the commit history offers a limited view, collaborative efforts and strategic planning are fundamental to our project's success.

Meeting Minutes:

Date: 04/09/2023

	Alex Kyriacou	**Julian Codespoti**	**Nelchael Kenshi Turija**	**Enzo Peperkamp**	**Marella Morad**
What are you working on?	Currently working on learning the supabase api's and best practices. As well as familiarising myself with Jira and all the tools required to manage tasks going forward.	Setting up Supabase organization and initiating table creation.	Learning how to utilise Supabase's API and Jira.	Installing and setting up @supabase/supabase-js in the local development environment - 0.5 hour Creating the initial Supabase client for connecting to the database - 0.5 hour Developing "Retrieve" functionality for items using the Supabase client - 1 hour	Design the UI for the Add New Member Page
ETA	Expected EOD but continuous work required	Expected completion within 1 hour.	EOD	EOD	EOD
What's up next?	Will continue to support team with ongoing responsibilities as Product Owner as well as looking at the creation of removing products and adding sales records endpoints	Proceed with GitHub repository and organization creation.	Testing backlog items that have been finished from in-progress	Begin integration of the Reporting component. Addressing potential bugs or issues that arise from the initial setup.	Design the UI for the Edit/Delete Existing Member Page
Any Blockers?	N/A	N/A	N/A	Awaiting SUPABASE_URL and SUPABASE_ANON_KEY for connecting the client.	N/A
Please add here	N/A	Supabase setup	N/A	Supabase setup	N/A

the number of hours still required for each task you worked on(burndown chart)		- 1 hour left.		- Completed Client connection - 0.5 hours left Retrieve functionality for items - 1 hour left	

Date: 05/09/2023

	Alex Kyriacou	**Julian Codespoti**	**Nelchael Kenshi Turija**	**Enzo Peperkamp**	**Marella Morad**
What are you working on?	Worked on developing the endpoints to support the products and sales database tables. Configured my system with npm and all the tools required for development of this project.	Finalizing GitHub repository and organization creation.	Testing create database table for members (F1), create database tables for sales records (F2), create database tables for products (F3)	Integrating the Reporting component into the main App - 1 hour Testing and refining "Retrieve" functionality for items using the Supabase client - 0.5 hour Exploring potential endpoints needed for the Members entity, given the app's progression - 0.5 hour	Design the UI for the Edit/Delete Existing Member Page
ETA	Tomorrow EOD	EOD	EOD	EOD	EOD
What's up next?	Finish development for the database endpoints.	Start developing the home page.	Testing completed back-log items from today	Begin developing endpoints for the Members entity based on findings. Work on refining the Reporting UI components that integrate with Supabase.	Review with the team and product owner for feedback on the design
Any Blockers?	Nothing to report	N/A	N/A	Minor inconsistencies between expected and	N/A

				retrieved data from Supabase. Investigation ongoing.	
Please add here the number of hours still required for each task you worked on(burndown chart)	~1 hour for completion of the endpoints	GitHub setup - 1 hours left.	N/A	Integration of Reporting - Completed Testing "Retrieve" functionality - 0.5 hours left Members' entity endpoint exploration - Completed	N/A

Date: 06/09/2023

	Alex Kyriacou	**Julian Codespoti**	**Nelchael Kenshi Turija**	**Enzo Peperkamp**	**Marella Morad**
What are you working on?	Completed the database endpoints. Begun investigating the best approach to the inventory dashboards including libraries and best practises	Hosting the website on Google Cloud Services.	Testing endpoints: retrieving members, products, and sales	Developing "Add Member" and "Edit Member" endpoints, making use of Supabase - 2 hours	Design a logo for GotoGro
ETA	Ongoing process but will move to next task by EOD	EOD	EOD	EOD	EOD
What's up next?	Next task is to design the UI for the inventory dashboards with the knowledge learned from today's research	Updating GitHub actions for continuous deployment.	Testing completed back-log items from today	Further refining and testing of the newly developed Member-related endpoints. Start planning for Products entity and the associated endpoints.	Review with the team and product owner for feedback on the design
Any Blockers?	Nothing to report	N/A	N/A	No major blockers. Minor UI challenges	N/A

				during testing.	
Please add here the number of hours still required for each task you worked on(burndown chart)	EOD so ~1-2 hours	Hosting - 4 hours left.	N/A	"Add Member" and "Edit Member" endpoints - 1 hour left	N/A

Date: 07/09/2023

	Alex Kyriacou	**Julian Codespoti**	**Nelchael Kenshi Turija**	**Enzo Peperkamp**	**Marella Morad**
What are you working on?	Currently working on Designing the UI for the inventory dashboard.	Developing the home page and working on user authentication.	Testing endpoints: adding,editing and removing members	Finalizing and refining "Add Member" and "Edit Member" endpoints - 1 hour Starting on "Retrieve Products" and planning other necessary endpoints for Products entity - 1 hour	Create Endpoints to manage members
ETA	Hoping to finish by the end of tomorrow	Tomorrow EOD.	EOD	EOD	EOD
What's up next?	Once this is done i can start integrating it into the final application	Developing login and signup pages.	Testing completed back-log items from today	Finalize the "Retrieve Products" endpoint.	Develop Add Members page
Any Blockers?	None	N/A	N/A	Some delay due to changing requirements for the Members entity.	N/A
Please add here the number of hours still required for each task you	Hoping to finish EOD tomorrow ~2-3 hours	Home Page development - 2 hours left.	N/A	"Add Member" and "Edit Member" endpoints - Completed	N/A

worked on(burndown chart)				"Retrieve Products" and planning - 1 hour left	

Date: 08/09/2023

	Alex Kyriacou	**Julian Codespoti**	**Nelchael Kenshi Turija**	**Enzo Peperkamp**	**Marella Morad**
What are you working on?	Didn't have much time yesterday and so still working on the UI of the inventory dashboard	Finalizing login and signup pages and initiating connection with Supabase.	Testing endpoints: adding, editing and removing sales records	Finalizing the "Retrieve Products" endpoint - 1 hour Starting UI component development for displaying and testing the management of Products - 1 hour	Develop the Add Members page
ETA	Hoping to finish by end of tomorrow	EOD	EOD	EOD	EOD
What's up next?	Integration into the final application	Hosting the website on Google Cloud Services.	Testing completed backlog items	Test and refine the "Retrieve Products" functionality. Explore potential requirements for a "Sales" endpoint.	Finalise the Add Members page
Any Blockers?	Assuming no more delays, nothing to report	N/A	N/A	Minor issues integrating the newly developed "Products" UI component with Supabase.	N/A
Please add here the number of hours still required for each task you worked on(burndown chart)	~2-3 hours	Login/Signup pages - 1 hour left.	N/A	"Retrieve Products" endpoint - Completed UI component for Products - 1 hour left	N/A

Final Check: Management System (GotoGro-MRM) for Goto Grocery Inc.

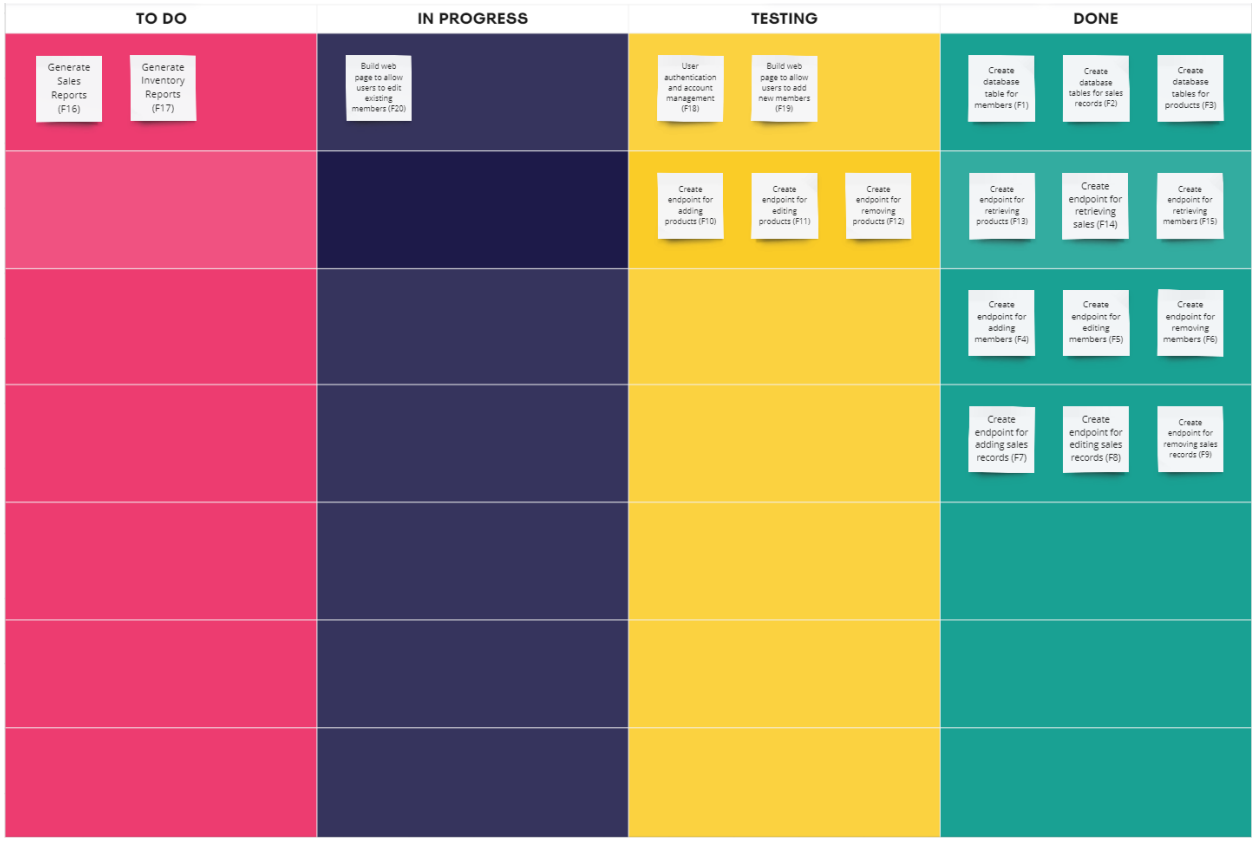
Team: MSP_CL4_T1	
Name	**ID**
Enzo Peperkamp	102895415
Nelchael Kenshi Turija	103057559
Julian Codespoti	102997816
Alex Kyriacou	103059830
Marella Morad	103076428

Table of Contents

MSP_CL4_T1 Monday 2:30 PM - 4:30 PM
BA708

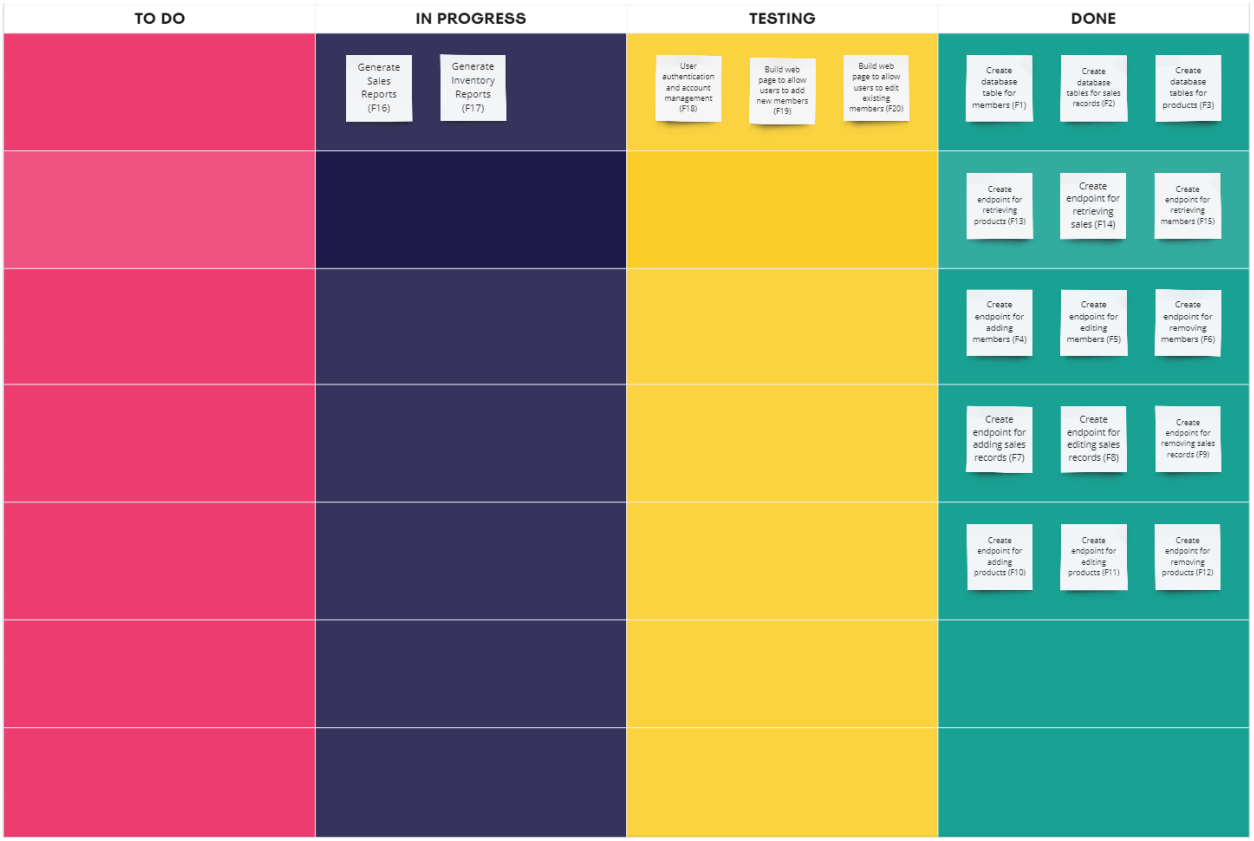
Task Boards

Date: 18/09/2023



MSP_CL4_T1 Monday 2:30 PM - 4:30 PM
BA708

Date: 19/09/2023



MSP_CL4_T1 Monday 2:30 PM - 4:30 PM
BA708

Date: 20/09/2023

MSP_CL4_T1 Monday 2:30 PM - 4:30 PM
BA708

Date: 21/09/2023

MSP_CL4_T1 Monday 2:30 PM - 4:30 PM
BA708

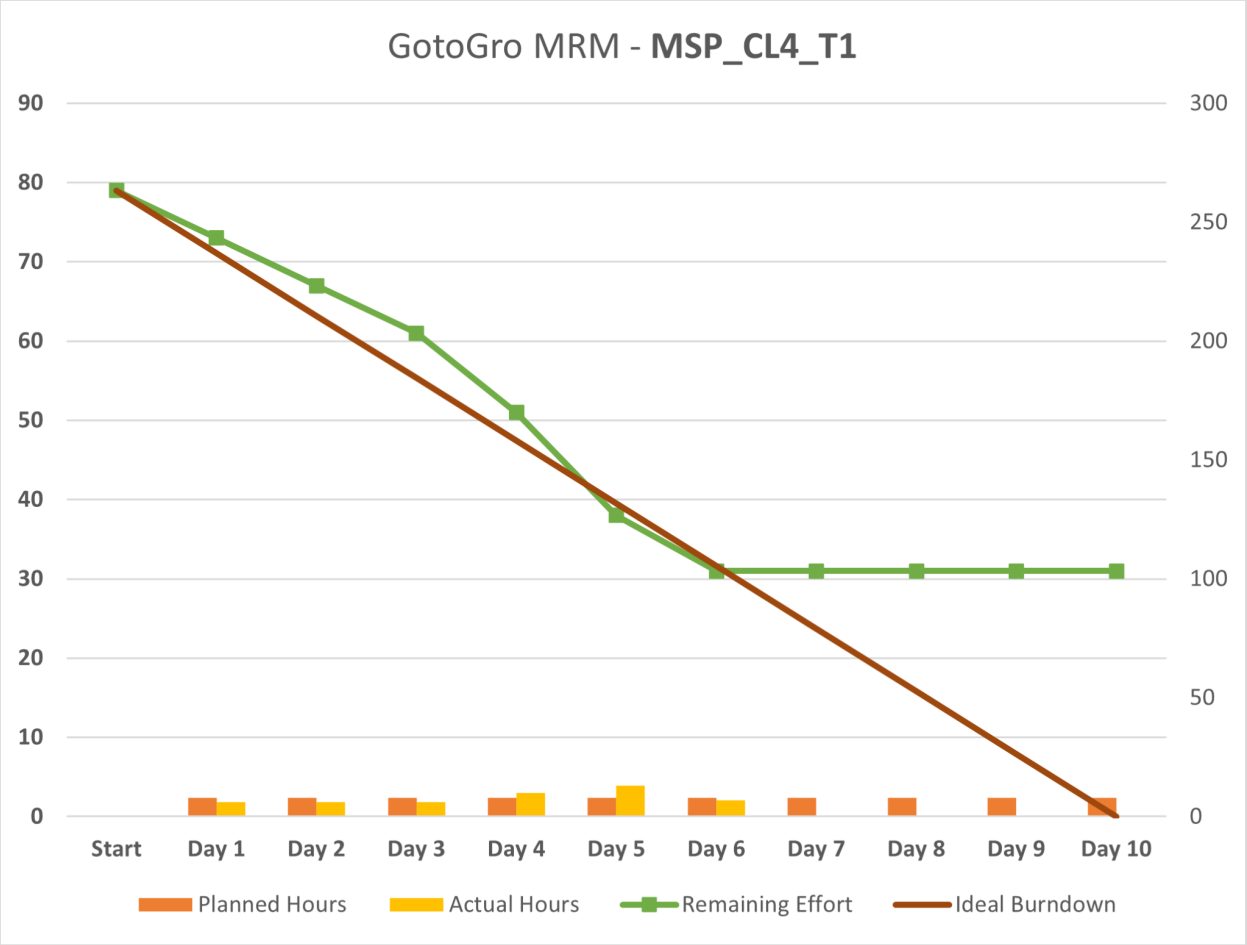
Date: 22/09/2023

MSP_CL4_T1 Monday 2:30 PM - 4:30 PM
BA708

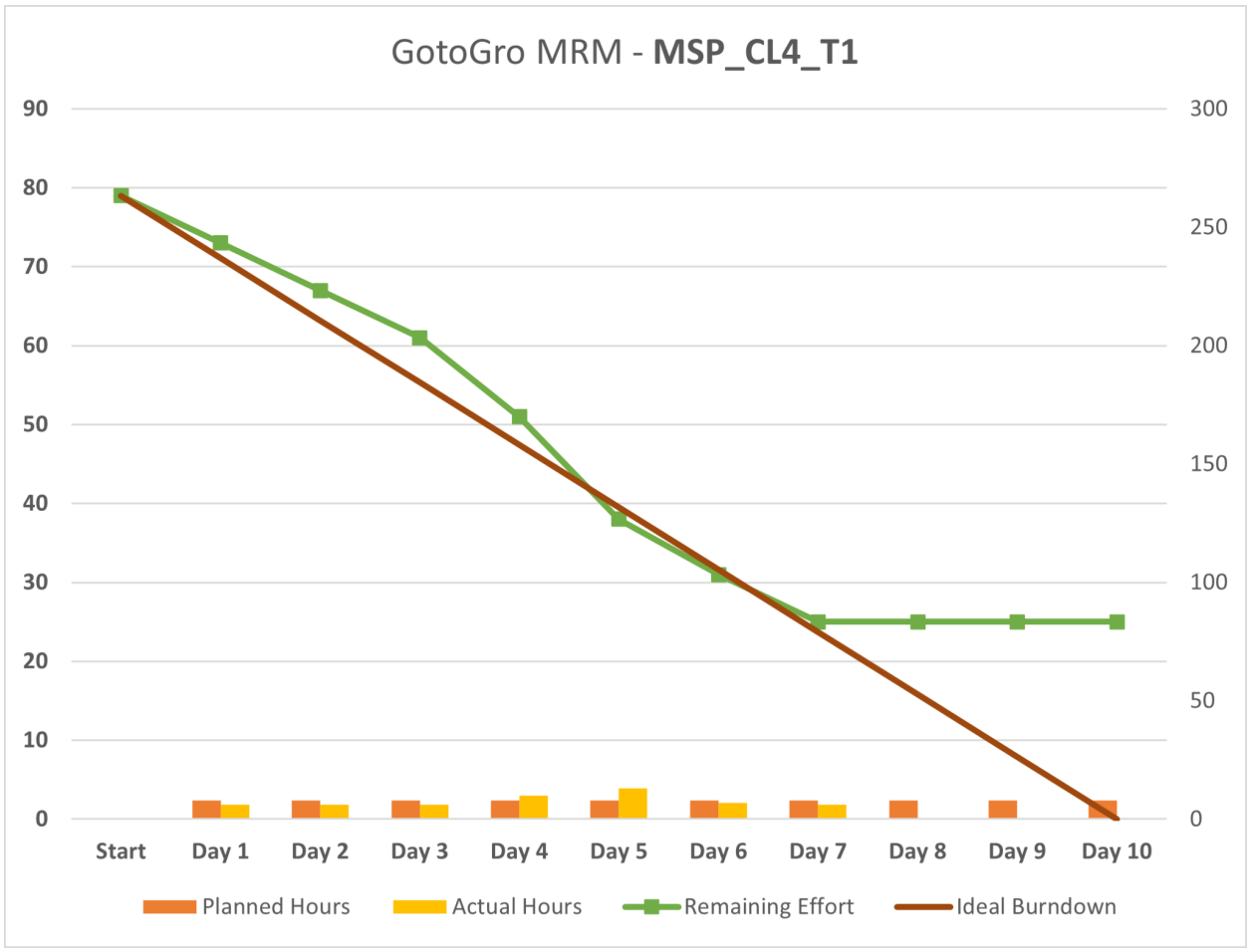
Date: 23/09/2023

Burndown Charts

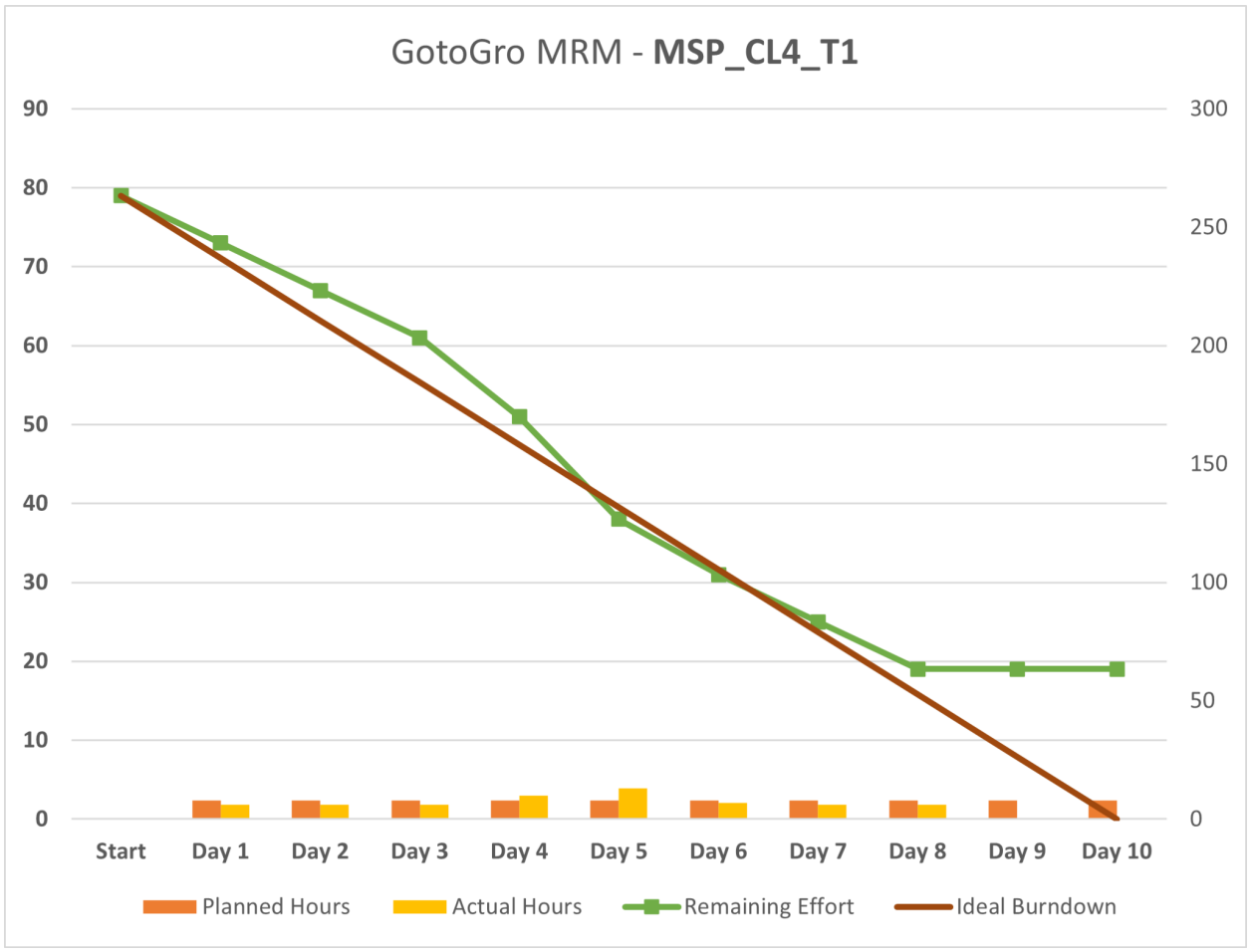
Date: 18/09/2023



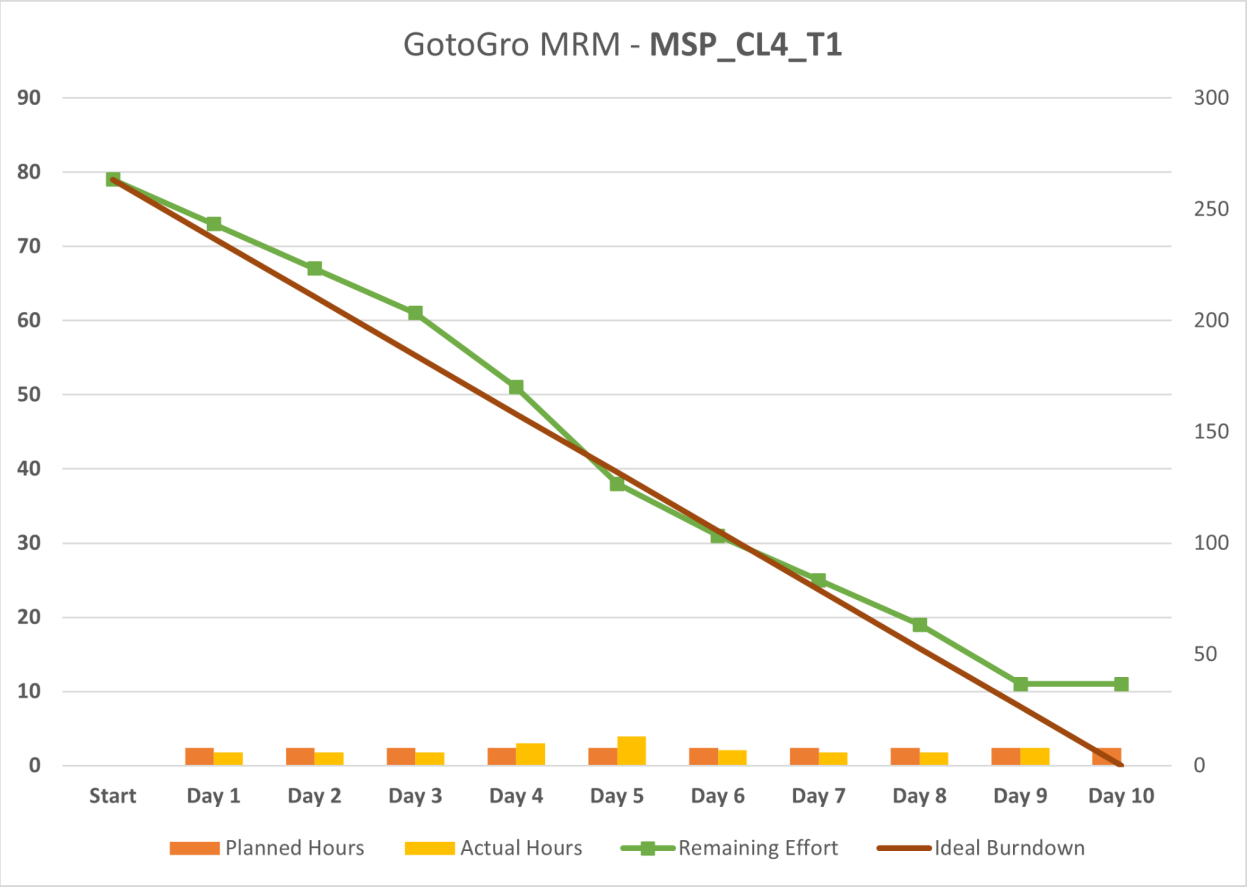
Date: 19/09/2023



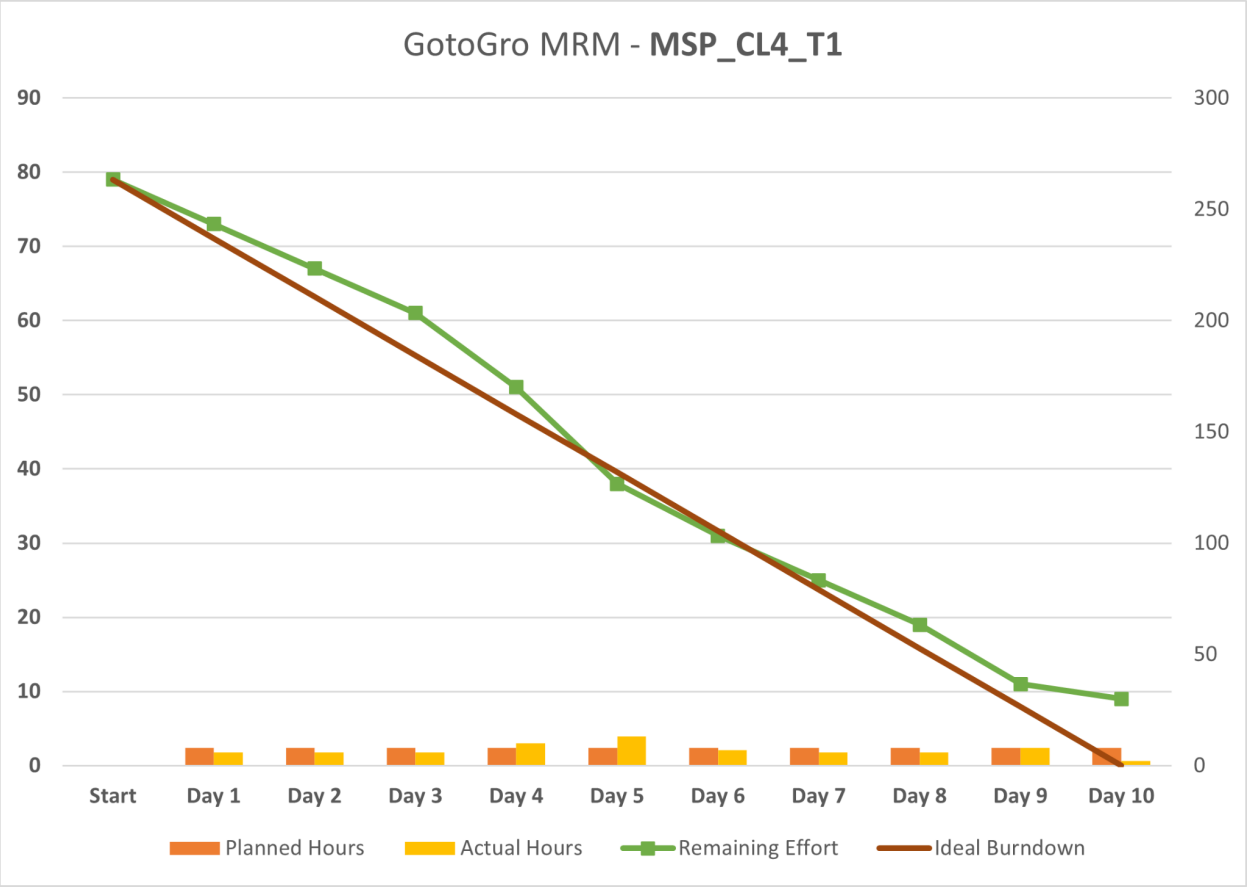
Date: 20/09/2023



Date: 21/09/2023

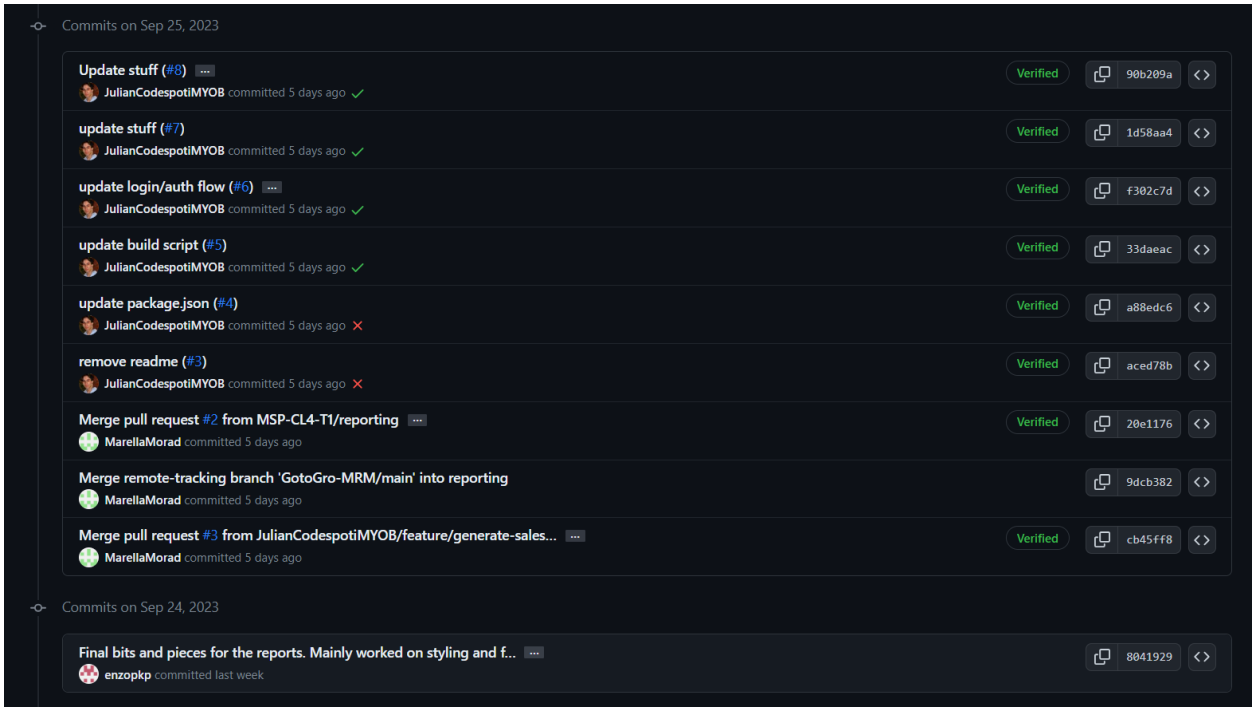


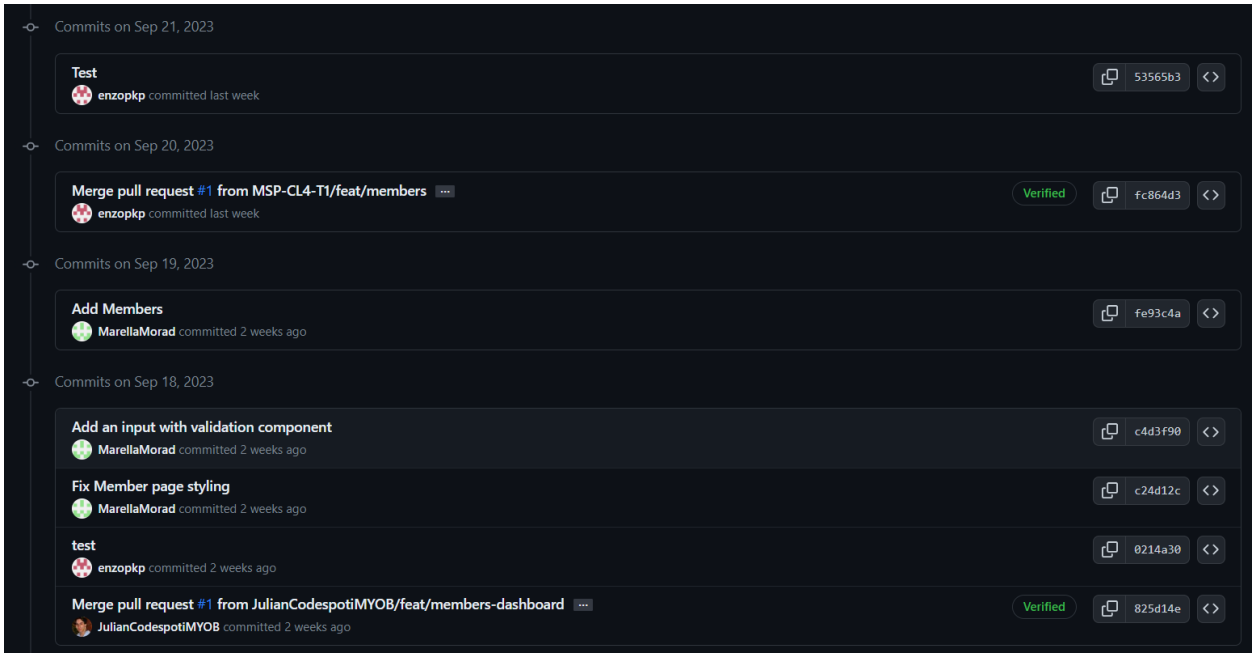
Date: 22/09/2023



MSP_CL4_T1 Monday 2:30 PM - 4:30 PM
BA708

Project Repository Status





MSP_CL4_T1 Monday 2:30 PM - 4:30 PM BA708

Meeting Minutes

Date: 18/09/2023

	Alex Kyriacou	**Julian Codespoti**	**Nelchael Kenshi Turija**	**Enzo Peperkamp**	**Marella Morad**
What are you working on?	Given the time remaining in this sprint, I am currently working on a MVP version of the inventory reports that will allow for querying the database and viewing the raw data. In sprint 2 i will then work on building a dashboard view that give more in depth insights	Updating GitHub actions for the deployment pipeline.	Testing endpoints: adding, editing and removing products	Testing and refining the "Retrieve Products" functionality - 0.5 hour Starting on the development of "Retrieve" endpoint for Members (F15) as mentioned - 1.5 hours	Yesterday, I was able to finish the Add Members page so today I'm starting with the development of the Edit Members page.
ETA	Hoping to get done by thursday this week	By mid-day.	EOD	EOD	EOD
What's up next?	Once this is completed as per the DoD, this will mark the end of the sprint 1 tasks	Testing the entire setup and ensuring smooth deployment.	Testing completed backlog items	Integrate the "Retrieve Members" endpoint with the existing Members UI. Initial exploration of generating a Sales report (F17).	Meet with the team to see where help is required regarding development, testing and/or documentation.
Any Blockers?	N/A	None	N/A	No major blockers. Small configuration issues with the new endpoint.	No blockers
The number of hours still required for each task you worked on	~4-5	GitHub actions update - 5 hours left.	N/A	"Retrieve Members" endpoint (F15) - 0.5 hours left Testing "Retrieve Products" - Completed	N/A

MSP_CL4_T1 Monday 2:30 PM - 4:30 PM BA708

Date: 19/09/2023

	Alex Kyriacou	**Julian Codespoti**	**Nelchael Kenshi Turija**	**Enzo Peperkamp**	**Marella Morad**
What are you working on?	Continued development of the inventory reports as per yesterdays stand up.	Conducting comprehensive testing of the entire setup.	Testing User Authentication and account management	Finalizing the "Retrieve" endpoint for Members (F15) - 0.5 hour Kickstarting development for "Generate Sales report" (F17) - 1.5 hours	Yesterday, I was able to finish the Edit Members page. After meeting with the team, we identified that the most assistance is required in documentation and tasks 10P and 11P. To address this, I'll start by manually creating the Sprint 1 burndown chart and setting up the Task Board to visualize our progress. Then, I'll proceed to draft sections 10P and 11P, integrating the necessary screenshots.
ETA	Still on original schedule, hoping to finish thursday	EOD	EOD	EOD	EOD
What's up next?	Once development is completed testing and documentation remains as per the DoD	Addressing any issues discovered during testing.	Testing completed backlog items	Delve deeper into the requirements for the Sales report. Collaborate with the team for front-end integration.	Review the tester's feedback on the Add and Edit Members page and fix any bugs.
Any Blockers?	N/A	None	N/A	Clarifications are needed on specific metrics to be included in the Sales report.	Testing of the Add and Edit Members pages to be completed
The number of hours still required for each task you worked on	~3-4	Testing - 2 hours left.	N/A	"Retrieve Members" endpoint (F15) - Completed "Generate Sales report" (F17) - 5.5 hours left	N/A

MSP_CL4_T1 Monday 2:30 PM - 4:30 PM BA708

Date: 20/09/2023

	Alex Kyriacou	**Julian Codespoti**	**Nelchael Kenshi Turija**	**Enzo Peperkamp**	**Marella Morad**
What are you working on?	Almost finished the inventory reports MVP. Given similarity with Enzo's sales reports. We will merge the code so that codebase is consistent and scalable in the next sprints upgrade of adding graphics to the dashboard	Adding additional security features in Supabase and refining user authentication.	Testing the Add Members page	Continued development of the "Generate Sales report" (F17) - 2 hours Initial integration tests of the report generation with mock data - 0.5 hour Collaboration session with the team on how the report should be rendered - 0.5 hour	Since testing is not yet done, I'll wrap up task 10P for submission by Monday, gather the necessary screenshots for 11P, and provide assistance to fellow team members as required.
ETA	Tomorrow EOD	EOD	EOD	EOD	EOD
What's up next?	Once given to Ken for testing and a final review is completed this will mark the end of the sprint	Implementing role-based access controls in Supabase.	Testing completed backlog items	Further testing of the Sales report with real-world data. Feedback and iteration session based on initial tests.	Review the tester's feedback on the Add Members page and fix any bugs.
Any Blockers?	N/A	None	N/A	Some challenges visualizing certain data points in the Sales report.	No blockers
The number of hours still required for each task you worked on	~2	Security and Authentication - 1.5 hours left.	N/A	"Generate Sales report" (F17) - 3 hours left	N/A

MSP_CL4_T1 Monday 2:30 PM - 4:30 PM BA708

Date: 21/09/2023

	Alex Kyriacou	**Julian Codespoti**	**Nelchael Kenshi Turija**	**Enzo Peperkamp**	**Marella Morad**
What are you working on?	Today was my birthday meaning that i did not get any time to work on the repo, only touch ups are left. Ken is ready to test meaning that everything should be ready by tomorrow.	Implementing role-based access controls in Supabase for refined user permissions.	Testing the Edit Members page	Refining and making adjustments to the "Generate Sales report" (F17) based on feedback - 2 hours.	Reviewing the tester's feedback on the Add Members page and fixing raised bugs
ETA	EOD Tomorrow	EOD	EOD	EOD	EOD
What's up next?	Final look over reporting code with Enzo, ensuring that codebase is in a state of readiness for reporting/dashboards in sprint 2	Optimizing Google Cloud Services configurations for better performance.	Testing completed backlog items	Final testing phase for the Sales report. Documentation and comments for the sales report generation process.	Review the tester's feedback on the Edit Members page and fix any bugs.
Any Blockers?	N/A	None	N/A	Some inconsistency in data metrics were raised during the feedback session that needs to be addressed.	No blockers
The number of hours still required for each task you worked on	1 hour left	Role-based access controls - 1 hour left.	N/A	"Generate Sales report" (F17) - 1 hour left	N/A

MSP_CL4_T1 Monday 2:30 PM - 4:30 PM BA708

Date: 22/09/2023

	Alex Kyriacou	**Julian Codespoti**	**Nelchael Kenshi Turija**	**Enzo Peperkamp**	**Marella Morad**
What are you working on?	Inventory reports are finished alongside Enzo's sales report. Enzo is ready to push the code to master meaning that task can be considered done.	Optimizing configurations and resources in Google Cloud Services for the application, while also refining and updating GitHub Actions to enhance the deployment pipeline's robustness and reliability.	Testing the Generate Sales Report and Generate Inventory Reports pages	Initiating documentation for both the Sales report and week's endpoints (1 hour) and conducting code review and housekeeping tasks to ensure code compliance with project standards (1 hour).	Review the tester's feedback on the Edit Members page and fix any bugs. Close all completed tasks in this sprint and see where help is needed
ETA	EOD	EOD	EOD	EOD	EOD
What's up next?	Need to complete a sprint review with the team as well as plan for what will be included within the sprint 2 task board	Finalizing and validating the entire setup, focusing on security and efficiency.	Work with the team to brainstorm and plan for Sprint 2	Take a brief break before launching into the next sprint. The team will then engage in a collaborative session to brainstorm priorities, gather requirements, and set the initial plan for the upcoming sprint.	Collaborative session with the team to brainstorm the next sprint's priorities.
Any Blockers?	N/A	None	N/A	None, all the main tasks for this sprint have been addressed.	None
The number of hours still required for each task you worked on	N/A	Google Cloud Services Optimization - 1.5 hours left. GitHub Actions refinement - 1 hour left.	N/A	N/A	N/A

Sprint Review: Management System (GotoGro-MRM) for Goto Grocery Inc.

Team: MSP_CL4_T1	
Name	**ID**
Enzo Peperkamp	102895415
Nelchael Kenshi Turija	103057559
Julian Codespoti	102997816
Alex Kyriacou	103059830
Marella Morad	103076428

Sprint Review Meeting Minutes

Date: 25/08/2023

Team Introduction:

Alex - Product owner
Enzo - Scrum Master
Ken - Tester
Marella - Developer
Julian - Developer

Goals:

Create database tables - successful.
Create endpoints to manage Members, Products, and Sales Records - successful
Setup User Authentication - successful
Build Member management Dashboard and pages - successful
Reporting - Sales and Inventory Reports - basic functionalities successful

Demo:

Demoed user authentication.
Demoed sign up.
Demoed email confirmation.
Demoed login.

Stakeholder comments: You've done a good job with the user authentication page, however you need to add the same title (GotoGro) to the login page. Just to be clear of what the actual website you're signing up for.

Stakeholder question: Does it automatically log you in?
Answer: Yes, you're authorized for 24 hours. You'll only see that login page after 24 hours of being logged in.

Demoed front page
Demoed google cloud hosting

Stakeholder question: So it's deployed on cloud?
Answer: Yes

Stakeholder question: Are you paying for this service?
Answer: No, we're using the free tier. As long as it doesn't hit 200 dollars, estimated costs, we don't need to pay yet.

Demo'd supabase

Developer: We've hosted our backend on supabase. Database that exposes its endpoints and authentication built in. member table, products table, and sales records table. On our front end we can connect directly via supabase.

Stakeholder question: Can we actually add stuff?
Answer: Yes we can.

Demoed dashboard
Demoed authentication-end of supabase.

Taskboard:

Product owner comments: I like how we can query and add records to the databases from the front end but from the user experience, GotoGrocer, I want the ability to view pie charts, analytics as to what is going on rather than seeing raw data.

Feedback on task board:
2 options with the dashboards (Reporting tasks):

1. Consider as not finished, and to move from sprint 1 to sprint 2
2. Close sprint 1 and add new items on sprint 2

The team decided to consider the Reporting task as not finished, move it to Sprint 2 and close Sprint 1.

Sprint Review Meeting Minutes

Date: 25/08/2023

Sprint Review: Management System (GotoGro-MRM) for Goto Grocery Inc.

Team: MSP_CL4_T1	
Name	**ID**
Enzo Peperkamp	102895415
Nelchael Kenshi Turija	103057559
Julian Codespoti	102997816
Alex Kyriacou	103059830
Marella Morad	103076428

Sprint Review Meeting Minutes

Date: 25/08/2023

Alex started the meeting off by introducing himself as a product owner. The rest of the team then introduced themselves in order. Enzo as the scrum master, Ken as a tester and Marella and Julian as developers. Alex then handed it off to Enzo to discuss the team's goals.

Enzo discussed the main goals of Sprint 1, which are the following

- To create the necessary database tables
- Create endpoints to manage the members, products, and sales records database tables
- To setup user authentication
- To build the member management dashboard and pages
- And to add a reporting page for sales and inventory.

Overall, Enzo and the team are happy since the team managed to achieve all the goals set out for this sprint. The only feedback Enzo gave was that he would like to add more advanced functionality to the two reporting pages, as what is present is basic functionalities.

Enzo then handed it over to Marella and Julian, who then demoed the application. They started off by showing how user authentication works. A user can sign up and receive an email confirmation plus login to prevent unwanted users from being able to use the app.

The stakeholder then commented that we would need to add a title to the login page to signify what the user is actually logging into.

The stakeholder then asked if the page automatically logs you in. Julian responded by saying that any user who logged in is authorized for 24 hours. This will prevent them from being asked to login again until 24 hours has passed.

Julian and Marella then demoed the home page of the app as well as where is it hosted (Google Cloud Services)

The stakeholder asked if it's deployed on the cloud? Julian responded with an affirmative, and that we have our own link that anyone can access on any device, even through a phone.

The stakeholder then asked if the team is paying for this service. Julian answered that the team is only using the free tier. As long as the total estimated cost of hosting the website doesn't exceed 200 dollars, the team doesn't need to pay for using their service.

Afterwards, Julian demoed the database by showing the supabase website. He then went on to explain that the team has hosted the entirety of the backend on supabase. This means that the database fully exposes

its endpoints and has authentication built in. He then shows off our database tables, the member table, the products table, and the sales records table. He then demonstrated that the front end connects directly to the back end utilizing supabase.

The stakeholder then asked if we can add records to the tables. Marella and Julian then proceeded to demonstrate that the app in its current stage, can add records to the database tables. Afterwards, Marella demonstrated the member dashboard that they created.

Afterwards, the product owner, Alex, commented that he likes how we can query and add the records to the databases from the front end but from a user standpoint he wants the ability to visually see data and the ability to view graphs and analytics rather than raw data.

After demoing the app, Marella then proceeded to show the taskboard and the progress we made during sprint 1. The stakeholder then asked why there is one last task that is considered “in progress” during the last day of the sprint. The team then explained that due to wanting to add advanced functionalities to the reporting pages, we are unsure if we would like to mark it as complete and add a backlog item towards sprint 2.

The stakeholder then provided us with the following options regarding the uncompleted backlog item:

1. Consider as not finished, and to move from sprint 1 to sprint 2
2. Close sprint 1 and add new items on sprint 2

The team decided to consider the Reporting task as not finished, move it to Sprint 2 and close Sprint 1.

Sprint 1 Goals Assessment and Reflection

In this section, we delve into a detailed assessment of the goals we set out for Sprint 1. Understanding what we've accomplished and the areas where we aim to improve is crucial for our team's iterative progress. Here's a breakdown of our achievements and areas of growth:

1. Creation of Database Tables

- Status: **Met**
- Details: The foundational database tables, crucial for the project's backbone, were set up seamlessly. These tables not only serve the present needs but are also flexible enough to accommodate future requirements.

2. Development of Endpoints for Members, Products, and Sales Records

- Status: **Met**
- Details: With the aid of Supabase, we developed efficient and responsive endpoints. This efficiency afforded us the opportunity to reinvest our time into enhancing the platform's overall user experience, leading to a more interactive and intuitive system.

3. Implementation of User Authentication

- Status: **Met**
- Details: We rolled out a robust user authentication system, safeguarding our platform from potential security threats. Supabase's tools streamlined this process, ensuring users have a secure and smooth login experience.

4. Construction of Member Management Dashboard and Pages

- Status: **Partially Met**
- Details: While we established a functional member management page, there's room for growth. The current version serves as a foundation upon which we plan to build, integrating more comprehensive features and a polished interface in subsequent sprints.

5. Reporting - Sales and Inventory Reports

- Status: **Partially Met**
- Details: We initiated the incorporation of reporting mechanisms for sales and inventory with elementary filtering. The initiation of an export function is another step in the right direction. Still, given stakeholder feedback, we recognize the need to augment these features in the upcoming sprint.

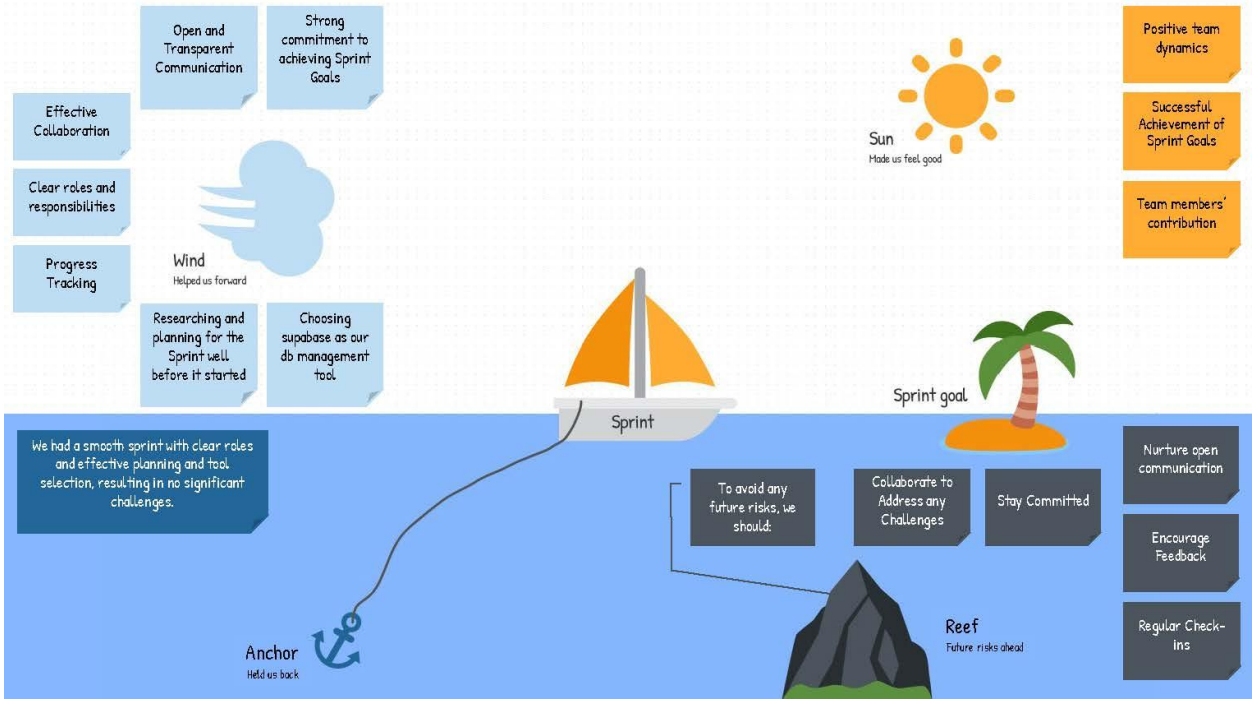
In summation, Sprint 1 saw a commendable momentum, setting the stage for an even more productive and innovative Sprint 2.

Sprint Retrospective: Management System (GotoGro-MRM) for Goto Grocery Inc.

Team: MSP_CL4_T1	
Name	**ID**
Enzo Peperkamp	102895415
Nelchael Kenshi Turija	103057559
Julian Codespoti	102997816
Alex Kyriacou	103059830
Marella Morad	103076428

Table of Contents

Sailboat Retrospective



Meeting Minutes

Team's velocity

Ideal (from your ideal burn-down chart) vs actual (from your final burn-down chart)

Our team's journey towards project completion can be best illustrated by comparing our ideal velocity, as depicted in our ideal burn-down chart found in 11P, with the actual progress reflected in our final burn-down chart. Throughout the project, we meticulously followed the ideal trajectory with slight deviations, aligning our efforts with the planned tasks and timelines. However, as the finish line drew near, a singular task remained incomplete and that was to generate the inventory and sales reports, casting a shadow on the last day of the burn-down chart. This is due to the fact that we wanted an inventory and sales reports that are more advanced than what is currently developed. Despite our best efforts, this final task reminds us that perfection may remain elusive in the dynamic landscape of software development. Nevertheless, our ability to closely mirror the ideal velocity for the majority of the project demonstrates our team's commitment and agility in navigating challenges and making continuous strides towards our goals.

Overestimating Or Underestimating?

Did your team overestimate your ability? Or Did you under-estimate the effort required to complete the tasks?

Reflecting on our project's performance, it becomes evident that our team's estimations leaned towards overestimating our abilities rather than underestimating the effort required for task completion. Specifically, our initial assessments were influenced by the assumption that generating endpoints, particularly those associated with Supabase, would entail a significant amount of effort. However, as we delved deeper into the project, it became apparent that Supabase had already provided much of the necessary code to seamlessly gather the required details from the database. This unexpected efficiency in leveraging Supabase's resources led to a notable discrepancy between our initial expectations and the actual effort required. In hindsight, this experience serves as a valuable lesson in the importance of reevaluating assumptions and continually refining our estimation processes to ensure a more accurate assessment of future tasks.

Assess Complexity, Improve Estimates

What can you do to get a better understanding of the "complexity" of the tasks required? Or What can you do to get better time estimates next time?

To enhance our comprehension of task complexity and improve our time estimates for future projects, there are several proactive steps we can take. Firstly, conducting thorough task analysis and breaking down large tasks into smaller, more manageable subtasks can help us grasp the intricacies involved. Collaborative brainstorming sessions with team members, drawing upon their diverse expertise, can provide valuable insights into potential challenges and complexities.

What is working? Why?

Our team dynamic is currently running smoothly, and it's important to acknowledge what's working well and why. The synergy among team members is highly effective, with open and transparent communication fostering collaboration and problem-solving. This cohesive environment has allowed us to efficiently tackle tasks and achieve our goals. Additionally, our well-defined roles and responsibilities contribute to the team's success, ensuring that everyone knows their contributions matter and align with our collective objectives.

What is not working? Why not? Any suggestions to improve the situation if this occurs in the future?

At this juncture, there aren't any significant issues or challenges that need addressing. However, to maintain this positive momentum in the future, it's essential to continue nurturing open communication, encourage ongoing feedback, and remain adaptable to evolving circumstances. Regular check-ins and retrospectives can help us quickly identify and address any potential hurdles or areas for improvement. Ultimately, sustaining our current success will depend on our collective commitment to maintaining these positive team dynamics and continually seeking ways to enhance our collaborative efforts.

Software Design: Management System (GotoGro-MRM) for Goto Grocery Inc.

Team: MSP_CL4_T1	
Name	**ID**
Enzo Peperkamp	102895415
Nelchael Kenshi Turija	103057559
Julian Codespoti	102997816
Alex Kyriacou	103059830
Marella Morad	103076428

Table of Contents

Design of the Software Components

Model Layer

In our current architecture, Supabase serves as the Model. It stores and manages all data, handles database operations, and enforces data integrity and consistency. Supabase consists of a cloud-hosted relational database service that will store all the database entities required for the system. This then interacts with the view/controller layer to create, read, update and delete records as well as retrieve relevant data from the database for The UI and report generation. This layer is also responsible for all the user access control and authentication within the application including 2FA management.

View/Controller Layer

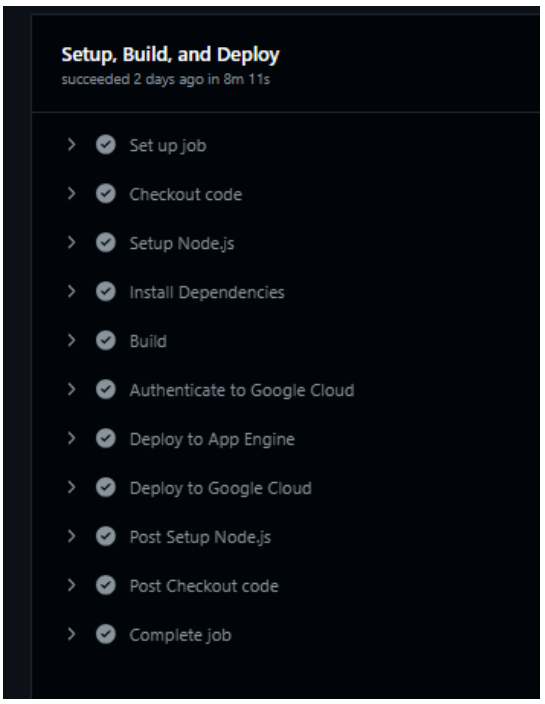
In our current architecture, we have a site powered with React JS hosted on Google Cloud platform that acts as a dual View and Controller layer. In a traditional MVC pattern, the controller and view act as separate entities, with the controller handling input, requests and updates and processing them for the view layer to be presented. This was not required for our project as with modern frameworks such as React JS, the controllers' responsibilities are now able to be handled among various components and libraries. These are able to handle all state management while also handling the presentation of the final site.

CI/CD Pipeline Architecture

In addition to the project architecture, we have also spent considerable thought on the architecture of our CI/CD pipeline. This has made the development of the GotoGro codebase seamless and deployment instant.

GitHub

We have used GitHub Actions alongside the GCP plugins to set up an automatic deployment pipeline to ensure that our production server always reflects the most up-to-date version of the master branch.




```yaml
name: Build and Deploy to GCP

on:
  push:
    branches:
      - main

jobs:
  setup-build-deploy:
    name: Setup, Build, and Deploy
    runs-on: ubuntu-latest

    steps:
    - name: Checkout code
      uses: actions/checkout@v2

    - name: Setup Node.js
      uses: actions/setup-node@v2
      with:
        node-version: 20

    - name: Install Dependencies
      run: npm install

    - name: Build
      run: npm run build

    - name: Authenticate to Google Cloud
      uses: google-github-actions/auth@v0.4.0
      with:
        credentials_json: ${{ secrets.GCP_SA_KEY }}

    - name: Deploy to App Engine
      uses: google-github-actions/deploy-appengine@v1
      with:
        project_id: ${{ secrets.GCP_PROJECT_ID }}
        credentials: ${{ secrets.GCP_SA_KEY }}

    - name: Deploy to Google Cloud
      uses: google-github-actions/deploy-appengine@main
      with:
        project_id: ${{ secrets.GCP_PROJECT_ID }}
        credentials: ${{ secrets.GCP_SA_KEY }}
```

Jira

In addition to our GitHub integration we have also set up Jira to assist with our development efforts. Within sprint 2 we will be able to synchronise with GitHub to perform some of the following actions automatically:

- When a pull request is merged, move the relevant issue to done
- When a branch is created, move the issue to in progress
- When a task is moved to in progress, create a branch in GitHub

By having this pipeline in place, it assists the team in both working on the project, as well as knowing with confidence what the state of the project is at any given time. This last point also ensures that our burndown chart stays accurate at all times as it automatically reflects the state of each task within the sprint.

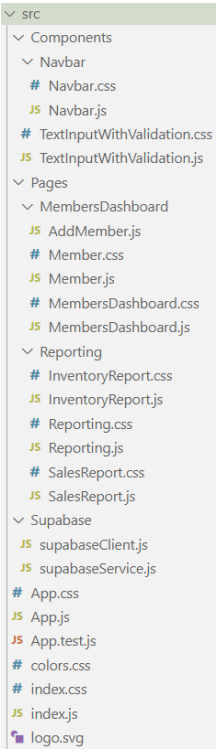
Design Justification using Design Principles

Naming Conventions:

Before starting to develop our web app, we agreed on some naming conventions. Naming conventions are essential for code readability and maintainability. It's important to have consistent and descriptive names for variables, functions, and components to make working on our product a lot easier and not require explanation from each team member every time they code something new.

Component Names:

We are following the camelCase naming convention to name our components, for example, `SalesReport`, `Reporting` and `MembersDashboard`.



Variable and Function Names:

We made sure our variable and function names are descriptive and follow the PascalCase naming convention such as `searchMembersByName`, `updateMember`, and `softDeleteMember`. Having such descriptive functions and variable names makes our code self-explanatory and reduces the number of comments we need to add to explain it.

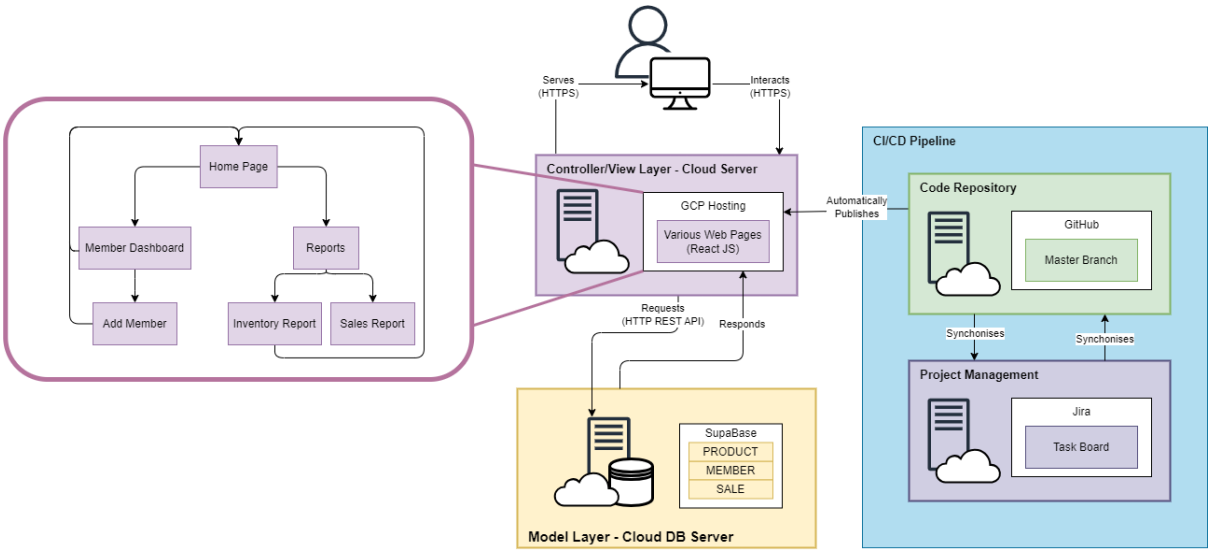
Code Organisation:

We've organized our code files into folders to enhance modularity, readability, and maintenance, facilitating developers' comprehension, modification, and software expansion. Beyond this, code organisation helps foster collaboration, scaling our project, and ensuring its long-term maintainability. This structure keeps our project flexible and efficient as it evolves and grows in complexity.

Design Principles:

Model-View-Controller Design Pattern:

The Model-View-Controller (MVC) design pattern is a software architectural pattern that separates an application into three interconnected components: Model, View, and Controller. As mentioned earlier, in our design, Supabase serves as the Model Layer, responsible for data storage, database operations, and user access control, while React JS, hosted on Google Cloud, serves as the dual-purpose View/Controller Layer. Unlike traditional MVC patterns, where controllers and views are separate, this design leverages modern React components and libraries to manage both user interface presentation and state control. Supabase acts as the backend database service, enforcing data integrity and handling user authentication, including 2FA management, while React takes care of site presentation and user interaction, blending the roles of controllers and views for an efficient and contemporary architectural approach. This is shown in the figure below.



Object-Oriented Principles:

We've structured our code to align with object-oriented principles such that:

- Each React component represents a cohesive unit of functionality
- Each component encapsulates its behaviour and rendering logic
- State variables and props are used to manage data flow and component interactions

Strong Cohesion and Weak Coupling:

We've organized our code to exhibit strong cohesion by structuring it into separate modules, each with a specific responsibility for handling particular functions. This approach enhances the comprehensibility and maintainability of our code. For instance, instead of consolidating all member functionality into a single Member component, we opted to divide it into three distinct components:

- MembersDashboard: Responsible for searching, retrieving, and displaying members.
- AddMember: Responsible for adding new members.
- Member: Responsible for editing existing members.

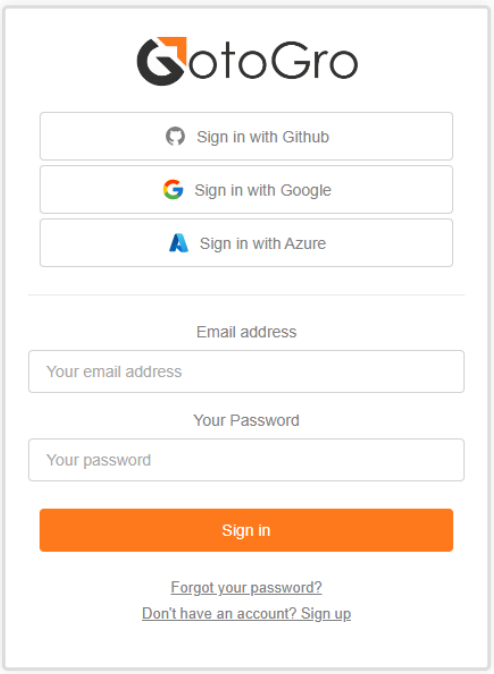
Furthermore, we aimed to achieve weak coupling, ensuring that these modules interact with one another with minimal dependencies. In simpler terms, they don't need to have an in-depth understanding of each other's internal workings to collaborate effectively. This concept is clearly exemplified in how our frontend components interact with Supabase APIs without necessitating extensive knowledge of how Supabase functions or how the database is structured. This approach streamlines the process of making modifications or adding new functionality without affecting other parts of the codebase.

Designing the Logo and UI

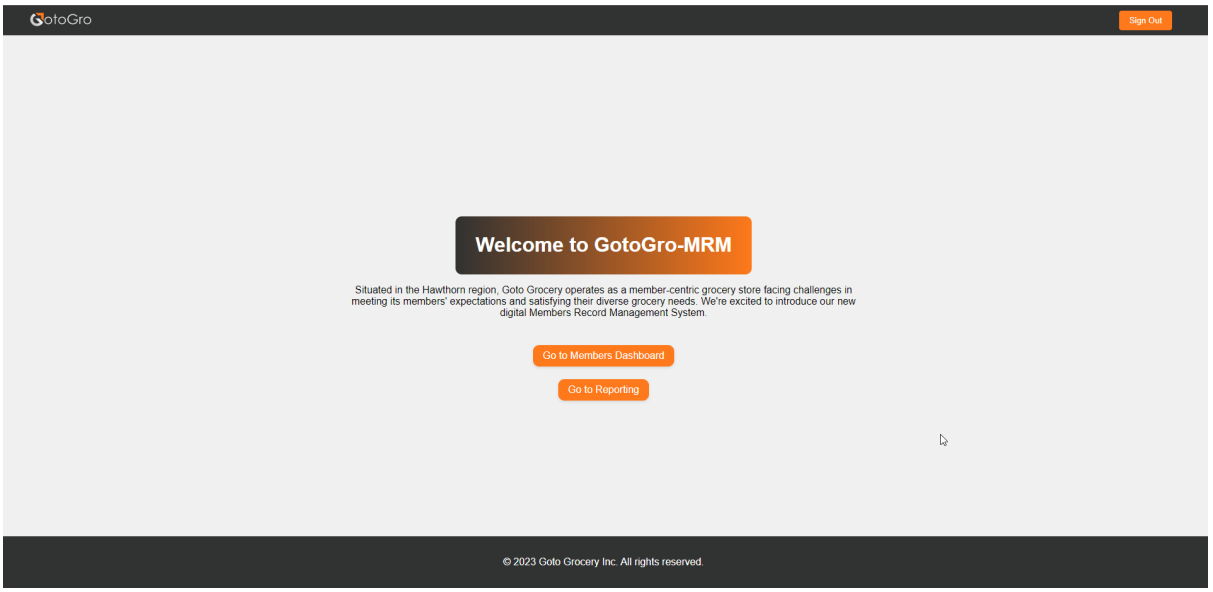
We made the decision to create a logo and a color palette for our web application. We began by crafting a suitable logo that embodies a sense of warmth and professionalism. After several design iterations, we settled on the logo displayed below. We have developed two variations of the logo to seamlessly integrate with both dark and light backgrounds.

GotoGro GotoGro

For example:
Light background in our sign-up and sign-in pages:



Dark background in our navbar:



With our logo in place, we proceeded to define our colour palette, ensuring that visual consistency would be a hallmark of our project. We also paid meticulous attention to maintaining a high level of colour contrast to enhance website accessibility.

colors.css

src > colors.css > ...

```
...
/* colors.css */
:root {
    --primary-color: #313232;
    --secondary-color: #fe791b;
    --dark-secondary-color: #bf5d17;
    --tertiary-color: #007BFF;
    --dark-tertiary-color: #0056b3;
    --dark-shade: #484848;
    --light-shade: #8c8c8c;
    --lighter-shade: #e1e6e1;
    --lightest-shade: #f0f0f0;
    --text-color: #000000;
    --info-color: #17a2b8;
    --warning-color: #ffc107;
}
```

SWE20001 Managing Software Projects
Self and Peer Review Assessment Form [Sprint #1/2]

Date: 1/10/2023_____
Your Team: MSP-CL4-T1_____ Your Name: Marella Morad_____

Use the instructions (see below) to fill in scores for each category A to J.

Team Members (Name)	A	B	C	D	E	F	G	H	I	J	Total
Self	4	4	4	5	5	5	4	5	5	5	46
Enzo Peperkamp	4	4	5	5	5	5	4	5	5	5	47
Nelchael Kenshi Turija	4	4	4	5	5	5	4	4	5	4	44
Julian Codespoti	4	4	5	5	5	5	4	4	5	4	45
Alexander Kyriacou	4	4	5	5	5	5	4	5	5	5	47

Your Reasoning / Justification (You must write a paragraph about each team member below. Incomplete reviews will not be accepted.)

Name, student number	**Comments (complete sentences required)**
Self	I consistently demonstrate a high level of commitment and enthusiasm in my work. With a score of 46, I am both productive and efficient, often going beyond the call of duty, which most of my teammates also do. My communication skills are satisfactory, and I maintain a harmonious atmosphere within the group. I always contribute to assignments and group discussions and always put my best effort towards the success of this group. I have also witnessed outstanding work from my teammates, which has enabled us to consistently excel in completing every task we're given.
Enzo Peperkamp	Enzo consistently performs at a high level, earning a total score of 47. He demonstrates outstanding productivity and delivers high-quality work, consistently accurate in all areas of contribution. Enzo is also a great team player with excellent communication skills. He is a self-starter and highly dependable, making him a valuable asset to the group. Enzo regularly reminds us about assignments and any outstanding work, and he excels at reviewing documents, addressing any issues, and ensuring our submissions align with the given rubric.
Nelchael Kenshi Turija	Nelchael received a total score of 44, indicating strong performance in most areas. He consistently meets productivity and quality standards, maintaining satisfactory communication and interpersonal relations within the team. Nelchael demonstrates initiative, is highly dependable, and is very responsive.
Julian Codespoti	Julian earned a total score of 45, indicating a solid performance across the board. He consistently exceeds productivity standards and communicates effectively with the team. Julian's interpersonal relations are satisfactory, and he displays a positive attitude. He shows great initiative and is a valuable team member, consistently completing work in a timely manner. Julian is always willing to help and highly dependable, thanks to his technological skills.
Alexander Kyriacou	With a total score of 47, Alexander consistently stands out in various aspects. He is highly productive, delivering high-quality work consistently. His communication skills are satisfactory, and he maintains a positive attitude. Alexander shows a strong sense of initiative and dependability, making him a reliable and valuable team member.

Self and Peer Assessment Form
The main purpose of this form (on Sheet 2) is for all Group members, including yourself, to reflect on its interactions, but it may be also be helpful in resolving disputes over the relative contributions of Group members.
Using the spreadsheet **Self and Peer Assessment Form**
1. List the members of your Project Group
2. Enter a score between 0 and 5, for categories A to J for all members of the group including yourself.
3. You will be asked to take a newly completed form to Group meetings with your supervisor: your supervisor will tell you which meetings.
S. Winger-Haunty (1990). University of Wisconsin-Stout Modified by Pheroza Daruwalla and Ian Knowd, 1994

A. Quantity of Work
0 - Did nothing - uninvolved
1 - Does enough to get by
2 - Occasionally exceeds standards- needs improvement
3- Satisfactory. Does more than what is required
4 - Very industrious. High Quality. Consistent
5. Always exceeds productivity standards. Outstanding

B. Quality of Work
0 - Careless. Makes frequent mistakes. Assignment suffers.
1 - Mistakes frequent enough to question results.
2 - Work is basically correct.
3 - Accurate when and where it really counts. Satisfactory.
4 - Almost always accurate in all areas of contribution
5 - Outstanding. Perfect quality. No mistakes.

C. Communication Skills
0 - Blunt, discourteous, does not listen, antagonistic, distant, aloof.
1 - Sometimes tactless. Approachable and friendly once known by others.
2 - Agreeable and pleasant. Warm, friendly , sociable, listens.
3 - Always very polite and willing to help. Very sociable and outgoing. Listens and understands.
4 - Courteous and very pleasant. Excellent at establishing good will.
5 - Inspiring to others. Artful listener. Really understanding.

D. Initiative
0 - Displays no self starting characteristics. Acts without purpose.
1 - Puts forth little effort. Requires prodding - sets no speed records.
2 - Puts in minimal effort to get task completed.
3- Strives hard. Desire to achieve.
4 - High desire to achieve. Always puts in a solid days work.
5 - Sets high goals. Self starter with high motivation. Constantly goes beyond call of duty.

E. Efficiency
0 - Work is invariably late.
1 - Work occasionally completed on schedule.
2 - Work usually complete on schedule. Some contribution to minor problem solving.
3 - Work always complete on schedule.
4 - Work complete. Consistent in defining and resolving major problems.
5 - Work invariably done ahead of schedule. Imaginative.
Can be counted on to make major contributions.

F. Personal Relations
0 - A very disruptive influence
1 - Is source of some friction
2 - Causes no problems
3 - Satisfactory, harmonious
4 - Is a positive factor
5 - Respected by others. Presence adds to environmental stability

G. Group Meeting Attendance
0 - Never attended any meetings. Showed no interest.
1 - Occasionally attended. Would commit and then not show.
2 - Sometimes uncooperative in planning schedule. Hard to get in touch with.

3 - Would attend. Usually late
4 - Could be counted on to attend.
5 - Never missed a meeting. Always on time

H. Attitude and Enthusiasm
0 - Poor disposition, uninvolved, indifferent
1 - Unenthusiastic, blase
2 - Half hearted
3 - Positive demeanour
4 - Positive attitude and spirited.
5 - Exuberant and eager. Positive influence. Inspiring to others. Team builder.

I. Effort
0 - Puts forth no effort. Expects others to carry the load.
1 - Puts forth some effort.
2 - Displays enough effort to get by.
3 - Solid contributions
4 - Strives very hard. Energetic.
5 - Self starter. Consistently goes beyond call of duty.

J. Dependability
0 - Uninvolved. Unreliable
1 - Unsteady, but tries somewhat.
2 - Occasionally would come through. Inconsistent.
3 - Needs some improvement. Suitable.
4 - Very trustworthy. Could be counted on to take responsibility.
5. Always responsible. Kept the group together and in the right direction. Steady influence

Sprint 2 planning: Management System (GotoGro-MRM) for Goto Grocery Inc.

Team: MSP_CL4_T1	
Name	**ID**
Enzo Peperkamp	*102895415*
Nelchael Kenshi Turija	*103057559*
Julian Codespoti	*102997816*
Alex Kyriacou	*103059830*
Marella Morad	*103076428*
Lachlan Martin	*103067448*

Enzo Peperkamp - 102895415

After scrutinizing the proposed plan for Sprint 2, I find the emphasis on feature dependencies particularly insightful. Acknowledging the intertwined relationship between features, such as the dependency of deleting users on adding new users, establishes a solid foundation for systematic progression. I am aligned with the direction we're taking and feel that our estimations, especially using the analogy technique, are well-justified.

Nelchael Kenshi Turija - 103057559

I am particularly impressed with the multi-pronged estimation techniques we've employed this sprint. The meticulous analysis and clear rationale provided for each task further instill confidence in our strategy. The emphasis on business value resonates with me, ensuring we prioritize tasks that bring about tangible benefits.

Julian Codespoti - 102997816

The careful delineation between feature dependencies, development effort, and risk speaks to a mature understanding of our project's intricacies. Drawing from my software development background, I believe the separation of concerns in our report aptly mirrors the MVC approach. Additionally, I appreciate the foresight in planning for increased velocity with Lachlan's inclusion, optimizing our resource allocation.

Alex Kyriacou - 103059830

Having been closely involved in discussions around the chosen architecture, it's heartening to see its merits reflected so comprehensively in our sprint plan. The detailed estimations, especially the by “experts” / “Delphi” techniques, underscore the collaborative spirit of our team. I'm confident that this sprint will pave the way for successful future iterations.

Marella Morad - 103076428

Our team's in-depth approach to identifying the relationship between features, especially noting the importance of business value even in the face of risk management, is commendable. I believe our approach in Sprint 2, backed by robust estimations and a clear direction, will prove instrumental in Goto Gro's growth.

Lachlan Martin - 103067448

As a recent addition to the team, I'm impressed with the clarity and precision of our sprint plan. The emphasis on risk, especially understanding its ties with each backlog item, resonates with my experience. Since, I’m a bit behind on the overall development process on this project, I will instead help in areas where I find myself really strong at which is Website Automation Testing. I'm eager to contribute and collaborate in ensuring our planned approach for Sprint 2 is actualized seamlessly..

Table of Contents

Task 1

Feature Dependency

This outlines any dependencies a sprint backlog item may have. For instance, the backlog item of deleting users is dependent on the feature of Adding new users, because users cannot be deleted without first being added.

Why this factor is important for Sprint 2

Feature dependencies still play a vital role in this sprint. Without clear insights into these dependencies, there's potential risk in misallocating resources or initiating development on features prematurely. By continuing to prioritize feature dependency, we ensure an organized and systematic approach to our sprint backlog.

Development Effort

This captures the various factors like the time, expertise, and resources essential to realize a project, with a keen focus on task intricacy and the expertise of our team.

Why this factor is important for Sprint 2

While we're focusing more on risk in this sprint, development effort remains paramount. An accurate estimation allows for a pragmatic allocation of resources and task distribution, directly impacting risk management. A miscalculated development effort can result in overcommitment, potentially jeopardizing the sprint’s objectives.

Business Value

This criterion evaluates the intrinsic value of a backlog item to the enterprise. A high business value can be attributed to items that either offer a competitive advantage or address critical issues prevalent among users.

Why this factor is important for Sprint 2

Incorporating risk assessment does not diminish the significance of business value. Aligning our priorities with items that offer tangible value ensures that we're not just mitigating risks, but also driving growth and satisfaction for the business and our customers.

Risk Involved in Developing an Item

This encompasses potential challenges or threats linked to the implementation of a backlog item. Risk factors can range from technical to external market-driven changes. In Sprint 2, we’re emphasizing this criterion to aid in our resource planning and delivery efficiency.

Why this factor is important for Sprint 2

Given the project's evolution, understanding and mitigating risks becomes indispensable. Despite the inherent advantages of our current project structure, such as an internal project nature and a stable development environment, there are still risks linked with each backlog item. By evaluating these, we can anticipate challenges and make informed decisions about which items to include in the sprint backlog.

Task 2

This ranking takes into consideration the importance of feature dependencies, development effort, business value, and risk involved in developing each backlog item. The frontend sales records management web pages are given the highest priority as they are essential for basic data management and likely have dependencies on other features. Products management and reporting web pages follow as they are crucial for inventory and reporting needs. The search and prediction algorithm, dashboards, and reports and analytics are also vital, but they may not possess as direct dependencies. Finally, the member management web page is ranked lower as it involves testing and may not directly impact core functionality in Sprint 2.

Ranking	Layer	Epic	No.	Item
1	Frontend	Sales Records Management Web Pages	F22	Build web page to allow users to add new sales records
2			F23	Build web page to allow users to edit existing sales records
3			F24	Build web page to allow users to remove sales records
4		Products Managment Web Pages	F25	Build web page to allow users to add new products
5			F26	Build web page to allow users to edit existing products
6			F27	Build web page to allow users to remove products
7		Reporting Web Pages	F28	Build web page to allow for sales data import via CSV
8			F29	Build web page to allow for data export to CSV
9			F30	Build web page for viewing sales reports and analytics
10		Search and Prediction Algorithm	F16	Create a prediction algorithm
11			F31	Implement search functionality with filters
12		Dashboards	F33	Create Dashboard for Inventories
13			F34	Create Dashboard for Users
14		Reports and Analytics	F35	Improve Inventory Reporting Function
15			F36	Improve Sales Reporting Function

In Sprint 1, our team of 5 planned for a total of 79 hours of work, which translates to roughly 16 hours for each team member over the course of the two-week sprint. This planned velocity equated to an average of 7.9 hours of work per day for the team.

However, when we look at our actual performance in Sprint 1 (see the table below), our team achieved an average velocity of 7 hours per day.

Setting	Start	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10	AVG
Planned Hours		7.9	7.9	7.9	7.9	7.9	7.9	7.9	7.9	7.9	7.9	7.9
Actual Hours		6	6	6	10	13	7	6	6	8	2	7
Remaining Effort	79	73	67	61	51	38	31	25	19	11	9	
Ideal Burndown	79	71.1	63.2	55.3	47.4	39.5	31.6	23.7	15.8	7.9	0	

In Sprint 2, with the addition of a new team member, Lachlan, it's reasonable to anticipate an increase in our team's velocity. This is in line with the typical expectation that more team members can potentially lead to increased productivity.

To ensure a safe and flexible planning approach, it would be prudent to plan for a total of 96 hours of work for the sprint, which equates to an average velocity of 9.6 hours per day. This planning target takes into account the potential positive impact of the new team member while allowing for a comfortable margin of flexibility, aligning with the principle of being conservative in our estimations.

By setting a planning target of 96 hours, we’re taking a cautious yet realistic approach, ensuring that we have the capacity to handle the work effectively while maintaining a commitment to delivering quality outcomes.

Task 3

Estimating by Analogy:

The estimation of the following tasks can be accomplished using the **Estimating by Analogy** technique, primarily due to the common functionalities they share with the **Member Management Web Pages** epic completed in Sprint 1. Each of the tasks in question involves building web pages for Sales Records Management and Products Management. While the Member Management web pages took approximately 4 points to develop, it's important to note that these Sales Records and Products web pages have distinct properties and requirements specific to their respective domains. Consequently, a reasonable estimate for each of these tasks would be approximately 5 points per web page. This approach leverages the analogy of development effort from the Member Management epic while accounting for the unique aspects of Sales Records and Products Management.

Epic	No.	Item	Estimation
Sales Records Management Web Pages	F22	Build web page to allow users to add new sales records	5
	F23	Build web page to allow users to edit existing sales records	5
	F24	Build web page to allow users to remove sales records	5
Products Management Web Pages	F25	Build web page to allow users to add new products	5
	F26	Build web page to allow users to edit existing products	5
	F27	Build web page to allow users to remove products	5

Tasks F35 and F36 can be estimated through analogy, considering they are enhancements to the two Reports and Analytics tasks accomplished during Sprint 1, each taking 8 points to complete. Since these tasks build upon existing functionality, we propose an estimate of 5 points per task.

Epic	No.	Item	Estimation
Reports and Analytics	F35	Improve Inventory Reporting Function	5
	F36	Improve Sales Reporting Function	5

Estimating by Size Comparison:

Task F37 is a testing task, which, in comparison to previous tasks, is relatively smaller in size and complexity. Since the web pages have already been developed during Sprint 1, the process of writing tests should be straightforward. However, it's worth noting that setting up the testing environment for Unit testing may require some time. Therefore, we are estimating 3 points for writing unit tests, 1 point for writing usability testing cases, and 1 point for executing the usability testing cases.

Epic	No.	Item	Estimation
Member Managment Web Page	F37	Write Test Cases for the Member Managment Web Pages - Write Unit Tests - Write Usability Testing Scenarios - Execute Usability Testing Cases	5 = 3 1 1

Task F16 involves the development of a prediction algorithm, representing a task of moderate to high complexity. Consequently, we have assigned it an estimated effort of 8 story points. In the case of implementing search functionality with filters (F31), this task also presents a high level of complexity, meriting an 8-point estimate. When it comes to creating a dashboard for inventories (F33), it entails the complex task of handling and visualizing a substantial volume of data. In light of this, a 10-point estimation is deemed appropriate to account for the task's size relative to the previous two. In parallel, the creation of a dashboard for users (F34) is anticipated to encompass the management of a substantial amount of user-related data, and therefore, it is also estimated at 10 story points.

These estimations are formulated with consideration for the complexity and size of the tasks, yet it is important to note that they remain approximate and subject to further refinement based on more detailed requirements and team expertise.

Epic	No.	Item	Estimation
Search and Prediction Algorithm	F16	Create a prediction algorithm	8
	F31	Implement search functionality with filters	8
Dashboards	F33	Create Dashboard for Inventories	10
	F34	Create Dashboard for Users	10

Estimating by “experts” / “Delphi” techniques (including Planning Poker):

During our sprint planning, we leveraged the collective expertise of our team members, drawing upon their past experiences and insights. Through comprehensive discussions, we presented the task details, including technical considerations, and utilized the Delphi technique, coupled with Planning Poker, to elicit individual story point estimates from each team member for the three tasks related to Reporting Web Pages (F28, F29, and F30). By aggregating and averaging these estimates, we arrived at a well-informed and consensus-driven assessment of the effort required for each task.

Epic	No.	Item	Estimation
Reporting Web Pages	F28	Build web page to allow for sales data import via CSV	5
	F29	Build web page to allow for data export to CSV	5
	F30	Build web page for viewing sales reports and analytics	5

Overall Sprint Tasks and Estimations

Ranking	Layer	Epic	No.	Item	Estimated Points
1	Front-end	Sales Records Management Web Pages	F22	Build web page to allow users to add new sales records	5
2			F23	Build web page to allow users to edit existing sales records	5
3			F24	Build web page to allow users to remove sales records	5
4		Products Managment Web Pages	F25	Build web page to allow users to add new products	5
5			F26	Build web page to allow users to edit existing products	5
6			F27	Build web page to allow users to remove products	5
7		Reporting Web Pages	F28	Build web page to allow for sales data import via CSV	5
8			F29	Build web page to allow for data export to CSV	5
9			F30	Build web page for viewing sales reports and analytics	5
10		Search and Prediction Algorithm	F16	Create a prediction algorithm	8
11			F31	Implement search functionality with filters	8
12		Dashboards	F33	Create Dashboard for Inventories	10
13			F34	Create Dashboard for Users	10
14		Reports and Analytics	F35	Improve Inventory Reporting Function	5
15			F36	Improve Sales Reporting Function	5
16		Member Managment Web Page	F37	Write Test Cases for the Member Managment Web Pages	5

*1 point = 1 hour

Sprint 2 planning: Management System (GotoGro-MRM) for Goto Grocery Inc.

Team: MSP_CL4_T1	
Name	**ID**
Enzo Peperkamp	*102895415*
Nelchael Kenshi Turija	*103057559*
Julian Codespoti	*102997816*
Alex Kyriacou	*103059830*
Marella Morad	*103076428*
Lachlan Martin	*103067448*

Enzo Peperkamp - 102895415
Our focus on refining the sprint workflow this time, especially ensuring Jira's meticulous update, stands as a testament to our commitment to continuous improvement. The tools served us well last sprint, and I believe their effective use this time will further streamline our process.

Nelchael Kenshi Turija - 103057559

It's evident from our prior experiences that setting up and diligently updating our tools, especially Jira, is paramount. Our recent discussions on sprint workflow highlighted these areas of improvement, and I'm confident we're heading in the right direction.

Julian Codespoti - 102997816

Taking our lessons from Sprint 1, I value our collective decision to prioritize keeping our Jira board accurate. The systematic workflow we've established for Sprint 2 should ensure clarity and efficiency throughout.

Alex Kyriacou - 103059830

Seeing the inclusion of Lachlan in our toolset is a positive step forward. Coupled with our renewed dedication to upkeeping the taskboard and learning from our past sprint mistakes, I anticipate a smoother workflow this time.

Marella Morad - 103076428

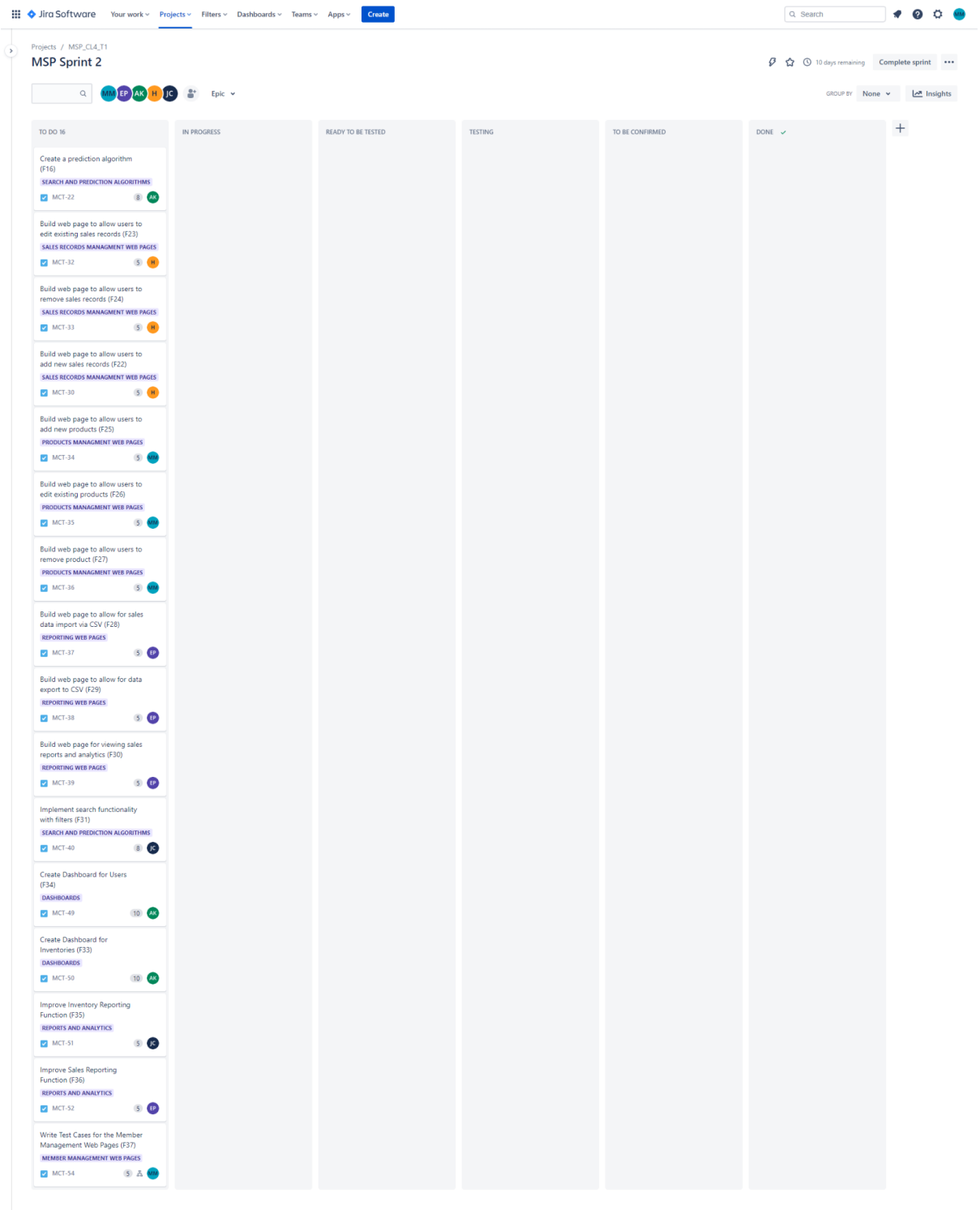
Our team's proactive approach, as demonstrated in rectifying the Jira oversight from Sprint 1, showcases our adaptability. The workflow, clearly defined and mapped out, promises to keep us on track and organized.

Lachlan Martin - 103067448

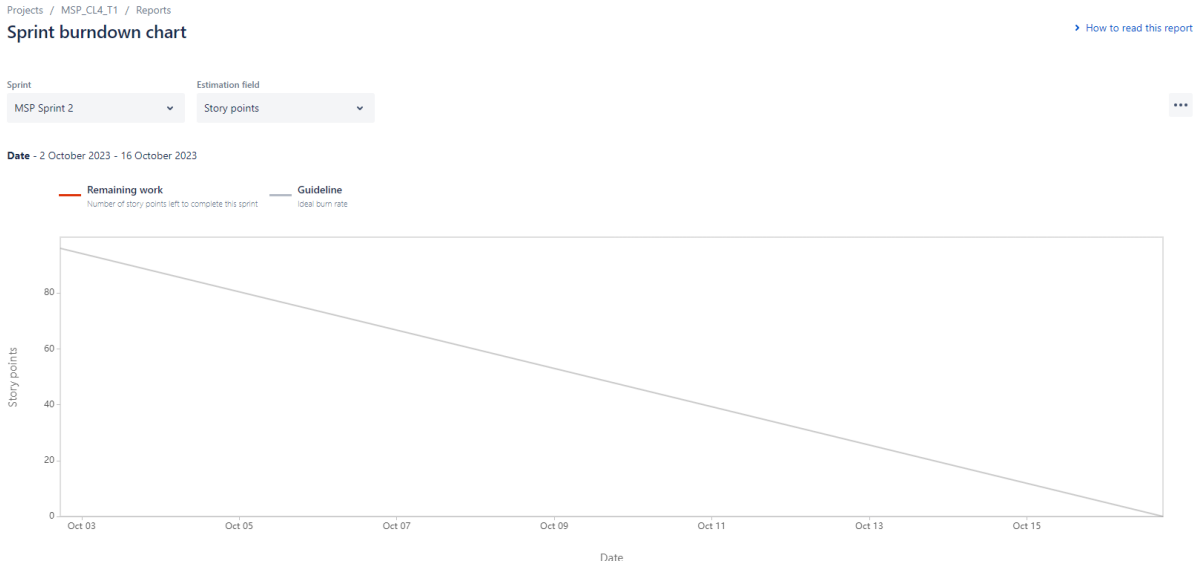
It's been fulfilling to hit the ground running with the team. I was able to contribute immediately by helping define a clear sprint workflow, addressing a previously existing gap; lack of automated testing. Let's remain diligent with our Jira updates and ensure our processes remain smooth throughout Sprint 2.

Table of Contents

Task board updated accordingly for Sprint 2

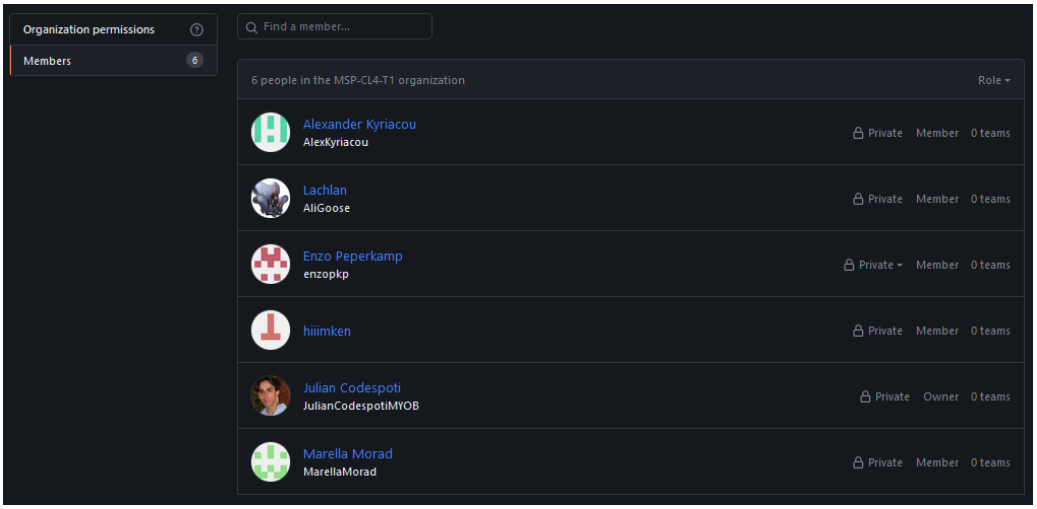


Sprint 2 ideal burndown chart

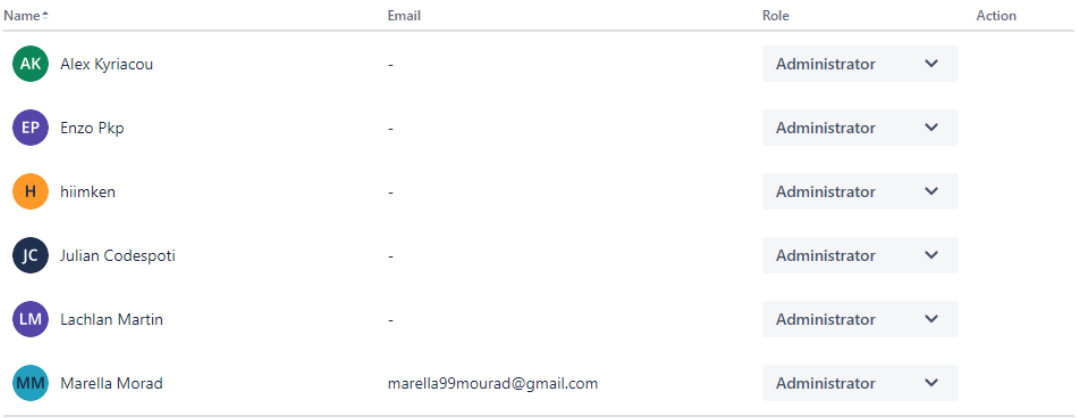


Add new member(s) to our different tools

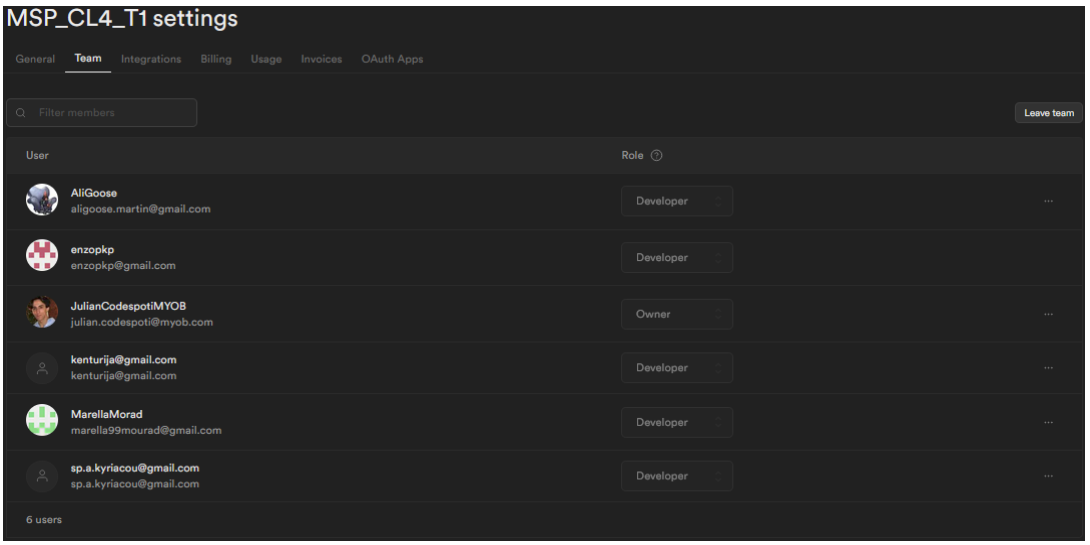
GitHub



Jira



Supabase



Sprint Workflow

- Sprint starts and sprint backlog items are moved into the **“To do”** column.
- Developers pick up tasks for development and move them into the **“In progress”** column. They will create a new branch from the *main* branch and name it after the User Story ID/Code.
- Once development is complete and the changes are merged into the production server. The item will be moved into the **“Ready to be tested”** column.
- Testers will pick up items for testing and move them into the **“Testing”** column.
- Once testing is complete, they will move the item into the **“To be confirmed”** column and provide all the test evidence they collected.
- The Product Owner / Scrum Master will review the test evidence and move to **“Done”** when they’re happy with the development and testing of the item.

Links

Please let us know if you run into any issues accessing these links:

- Website: https://msp-cl4-t1.ts.r.appspot.com/
- Jira: https://msp-cl4-t1.atlassian.net/jira/software/projects/MCT/boards/1
- GitHub: https://github.com/MSP-CL4-T1/GotoGro-MRM

Sprint 2 MidWeek Check: Management System (GotoGro-MRM) for Goto Grocery Inc.

Team: MSP_CL4_T1	
Name	**ID**
Enzo Peperkamp	*102895415*
Nelchael Kenshi Turija	*103057559*
Julian Codespoti	*102997816*
Alex Kyriacou	*103059830*
Marella Morad	*103076428*
Lachlan Martin	*103067448*

Contribution Statements:

Enzo Peperkamp - 102895415
The evolution of our tasks, from initiating the CSV export to enhancing our Sales Reporting Function (F36), has showcased the dynamism of our project. Each stage teaches something new, and I'm eager to wrap up F36, hoping it adds significant value to our end product.

Nelchael Kenshi Turija - 103057559

Progressing through the Sales Records Database functionalities, from editing to adding records, has been a hands-on experience. Each step requires meticulous attention, ensuring that our end-users have a seamless experience. Looking forward to wrapping up these functions and ensuring our database interactions are fluid.

Julian Codespoti - 102997816

Confronting challenges like the data consistency issue and then moving onto comprehensive testing of our dashboards epitomizes the fluidity of our development process. Every hiccup serves as a lesson, making our product more robust. I'm eager to address any issues that arise from our testing phase, knowing it will only refine our final output.

Alex Kyriacou - 103059830

Navigating through personal commitments like job interviews while managing our sprint tasks has been a balancing act. As I begin to delve deeper into dashboard functionality, laying down the foundation with testing plans and library selections promises a streamlined development process ahead. I'm gearing up for a productive second week, with the end goal being a dynamic dashboard.

Marella Morad - 103076428

Shifting from writing unit tests to design and development tasks has provided a holistic view of our product's lifecycle. With the product dashboard complete and the "add product" page in the pipeline, I am optimistic about providing a comprehensive experience to our users. Our commitment to quality and functionality stands tall.

Lachlan Martin - 103067448

Diving into different testing frameworks, finally settling with Selenium IDE, and promptly moving to test various user stories is a testament to the agile nature of our work. As tasks move to the testing phase, my focus is sharpened, ensuring each function is foolproof.

Table of Contents

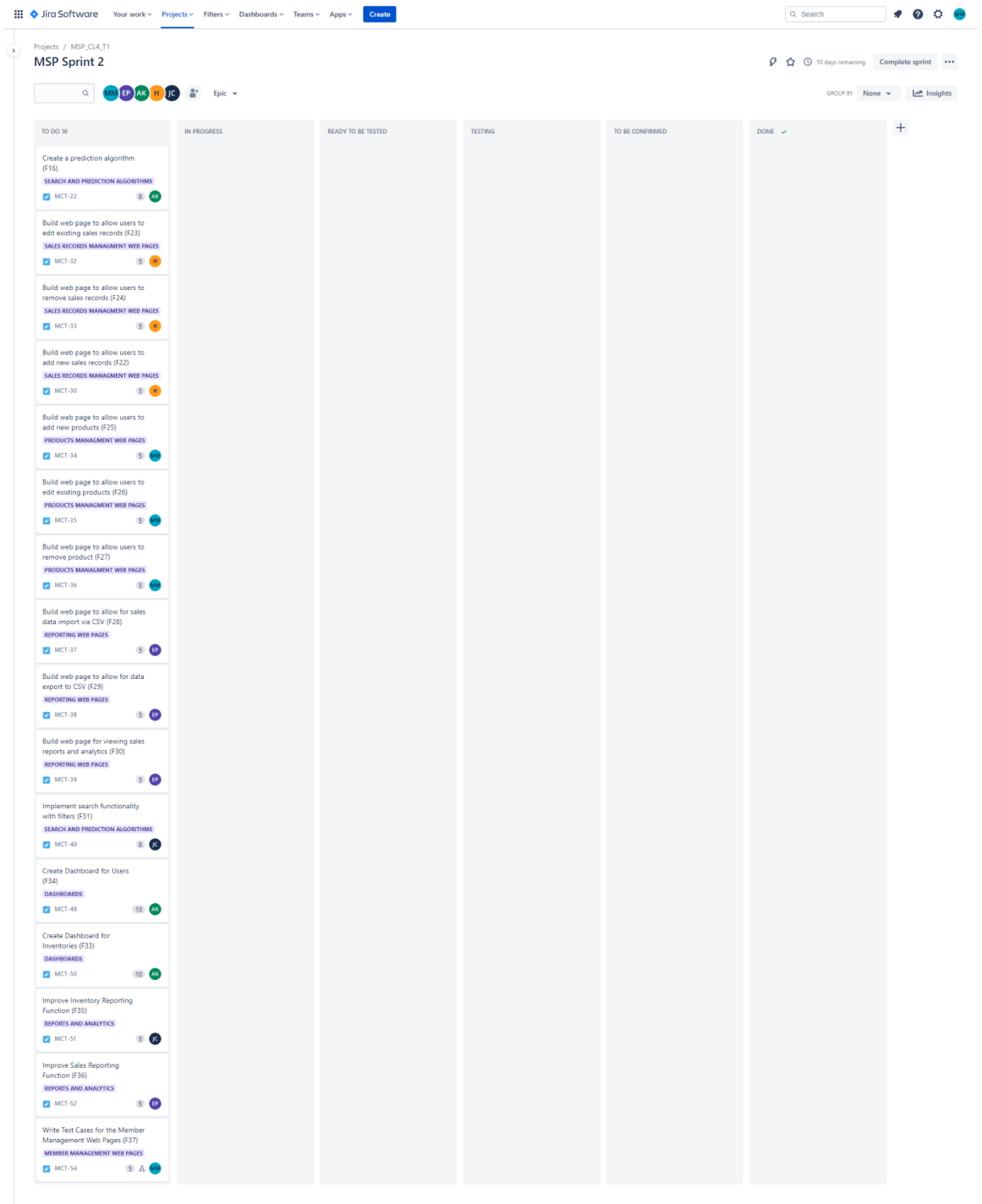
Resources and URLs:

JIRA Board Link
Burn-down Chart Link
Github Repo Link

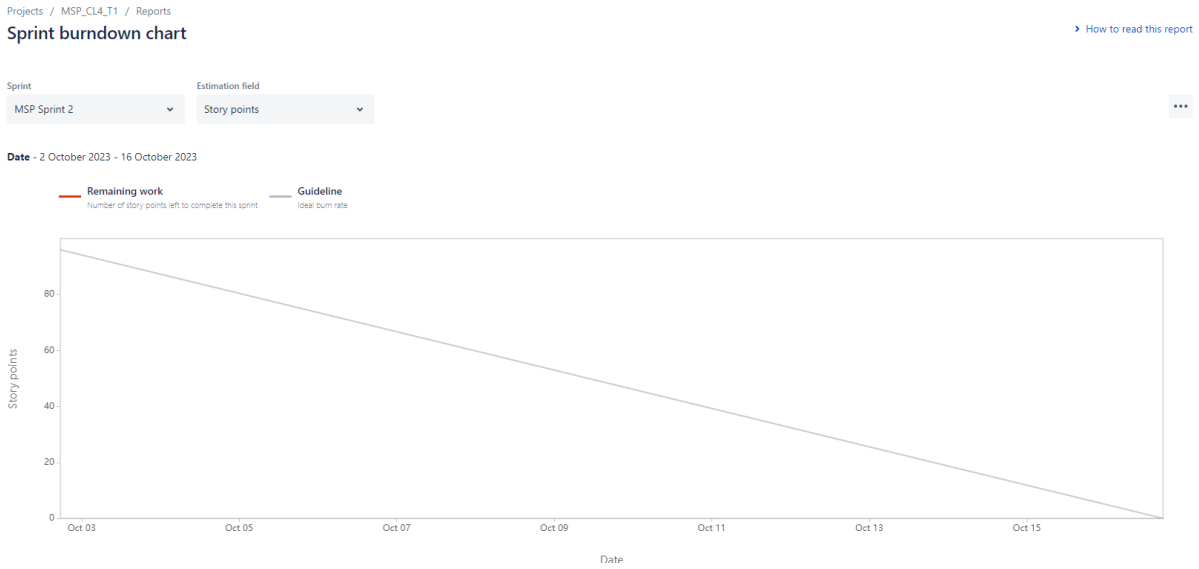
Date: 02/10/2023

On the first day of the sprint, our primary focus was on research and design, which is why the task board currently doesn't display any tasks in progress. You can find additional details in the meeting minutes.

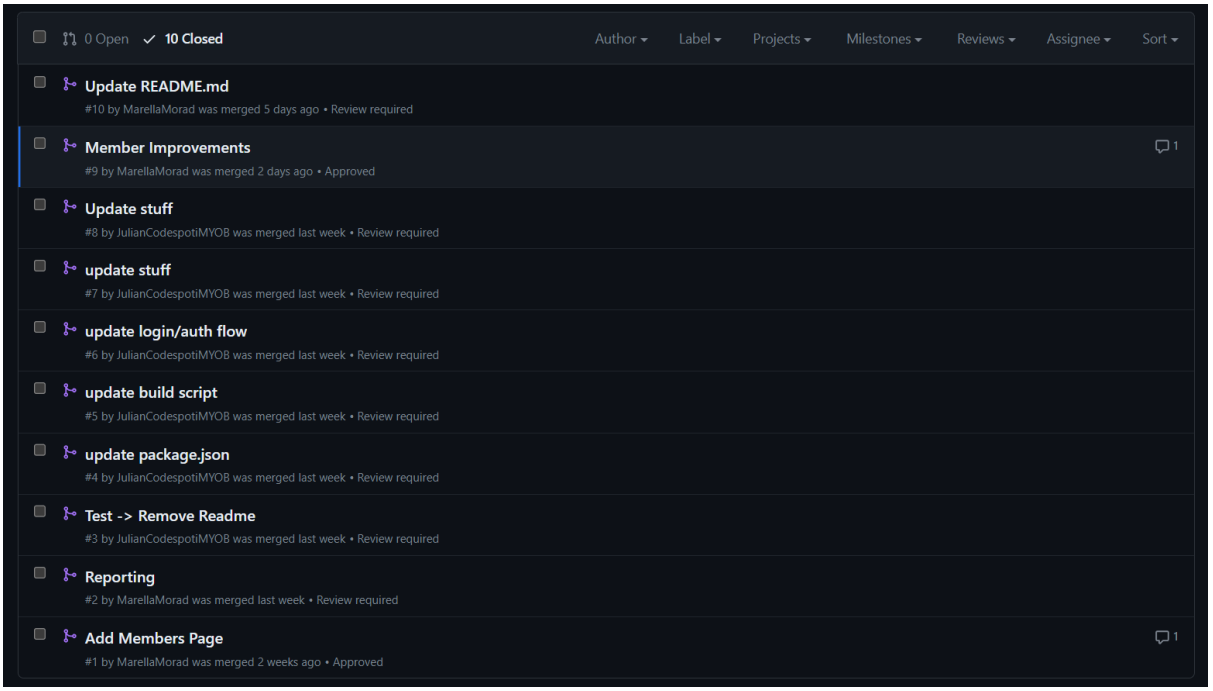
Task Board:



Burn-down Chart:



Project Repository Status:



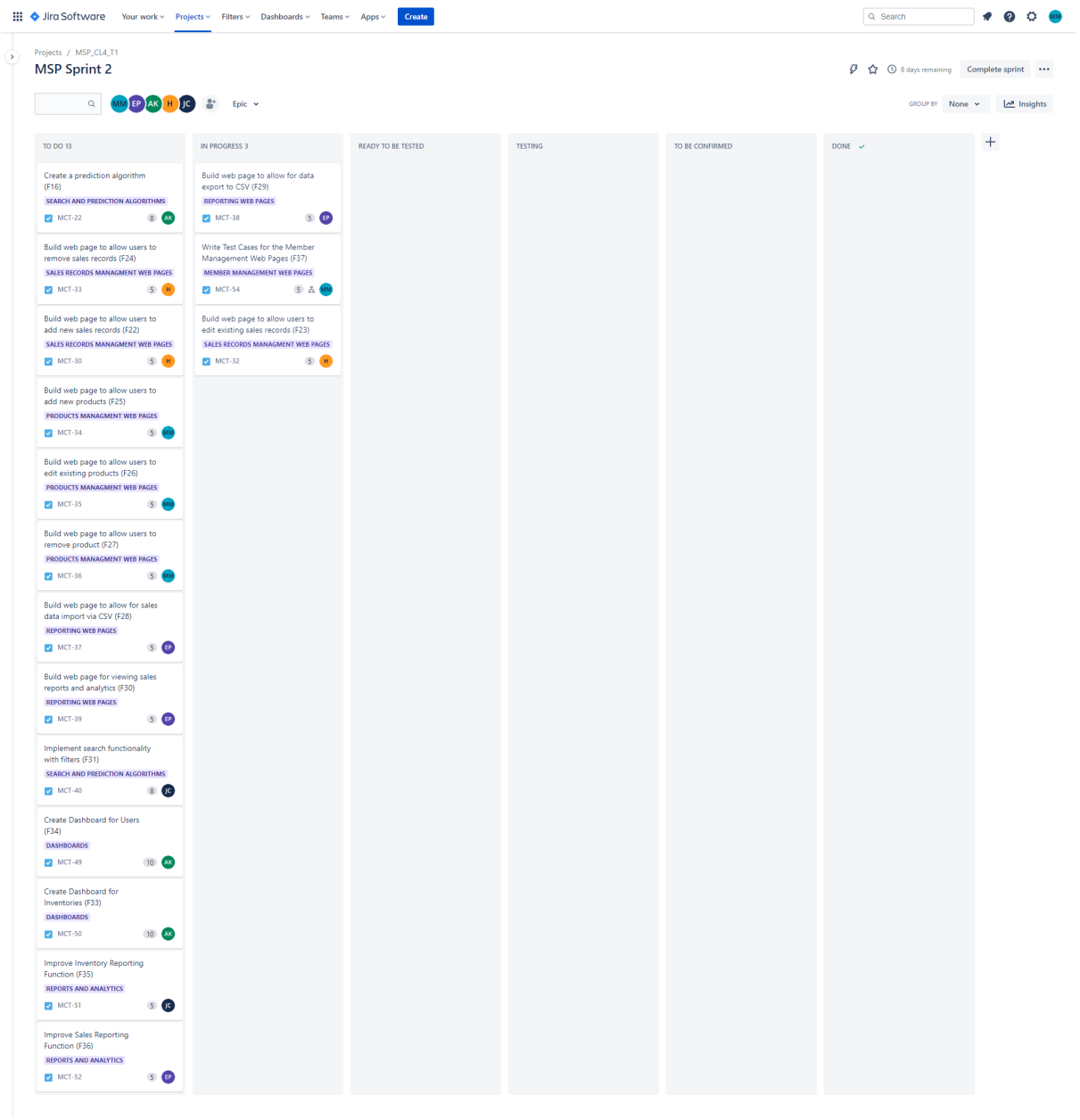
Meeting Minutes:

	Alexander Kyriacou	**Julian Codespoti**	**Nelchael Kenshi Turija**	**Enzo Peperkamp**	**Marella Morad**	**Lachlan Martin**
What are you working on?	Worked with greater team to do sprint planning and allocating the appropriate tasks for the relevant sprint	Focused on the sales dashboard, implementing graphs and analytics based on sales data. Additionally, commenced the setup and preliminary design for products dashboard.	Sprint initialization stage. Assigned tasks equalling 15 hours for the whole sprint	Collaborated with the team on sprint planning. Began the implementation of the CSV export feature for the sales report. (F29)	I will start researching a unit testing framework called Jest to write unit tests for components developed in Sprint 1	I will begin working on setting up a Java Project utilising Selenium for Automation Testing.
ETA	N/A	EOD + 2 Days	N/A	EOD	N/A	N/A
What's up next?	Will begin working on the dashboard tasks, including looking at appropriate frameworks and approaches to testing	Progressing with the products dashboard and further enhancing the sales dashboard based on preliminary feedback.	Create web pages for editing existing sales records	Complete the CSV export function. (F29) Initiate the development of a webpage for viewing sales reports and analytics. (F30)	I will start writing unit tests for the Member Management Web Pages.	Will await development to be complete for User Stories and begin testing them.
Any Blockers?	N/A	N/A	N/A	N/A	N/A	N/A
The number of hours still required for each task you worked on	N/A	Sales Dashboard: 5 Hours Products Dashboard: 8 Hours	N/A	F29: 3 hours	N/A	N/A

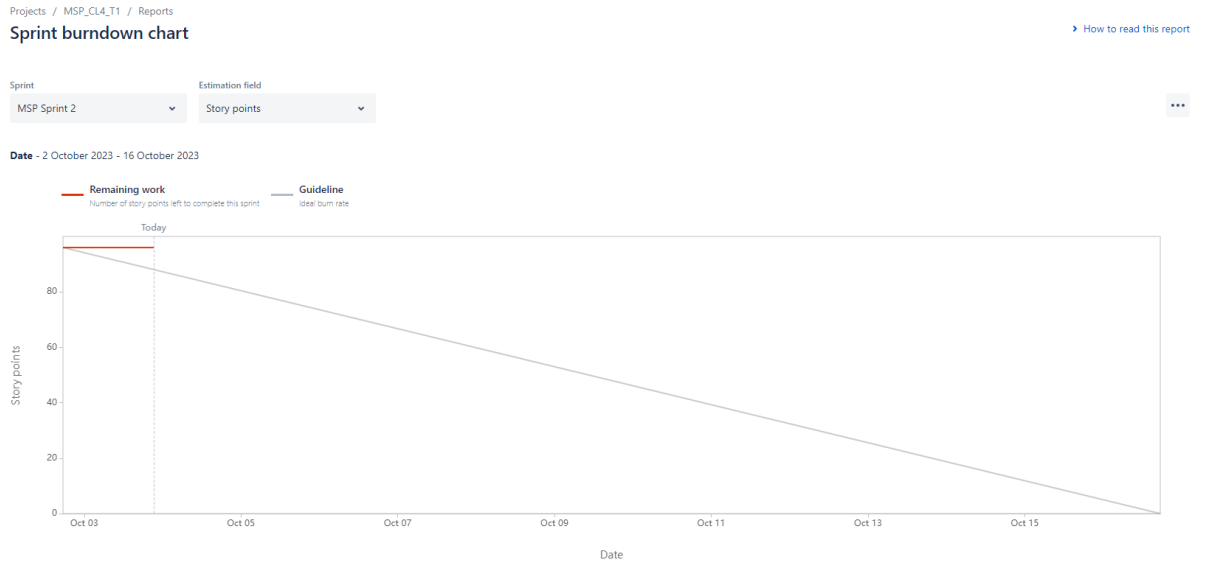
Date: 03/10/2023

We've begun the development of web pages to enable CSV data export and the editing of existing sales records. In parallel, we've also initiated the member management web pages testing tasks.

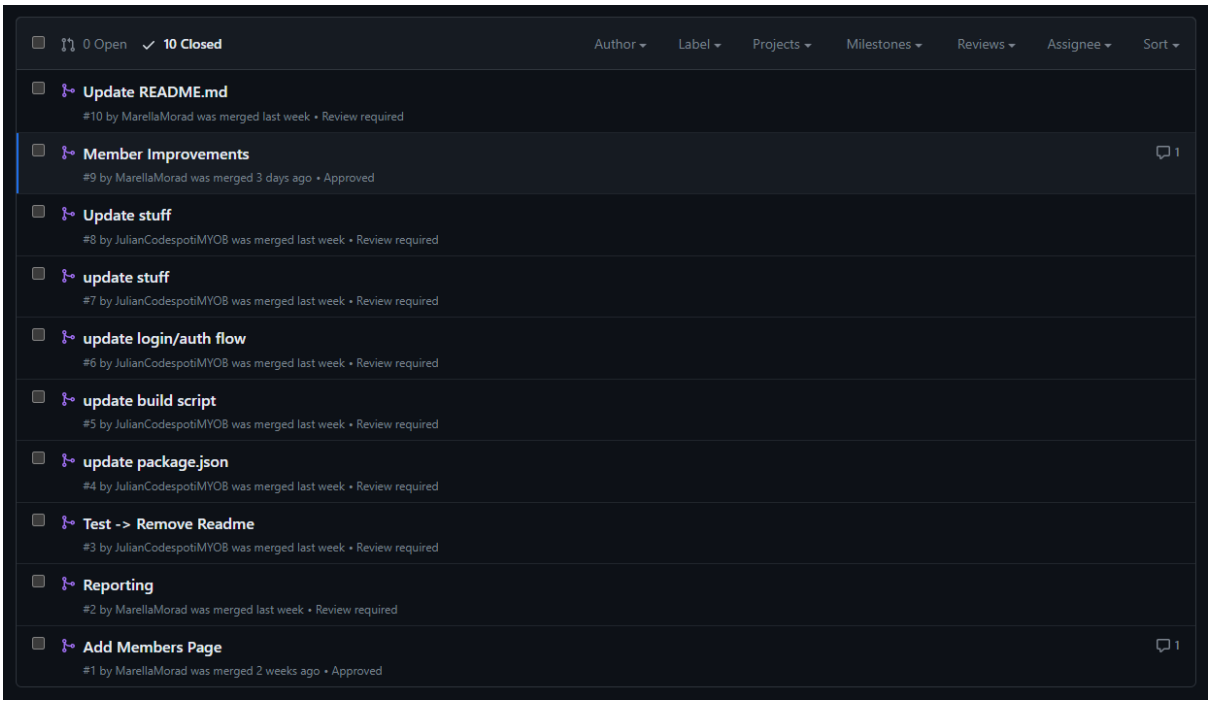
Task Board:



Burndown Chart:



Project Repository Status:



MSP_CL4_T1 Monday 2:30 PM - 4:30 PM BA708

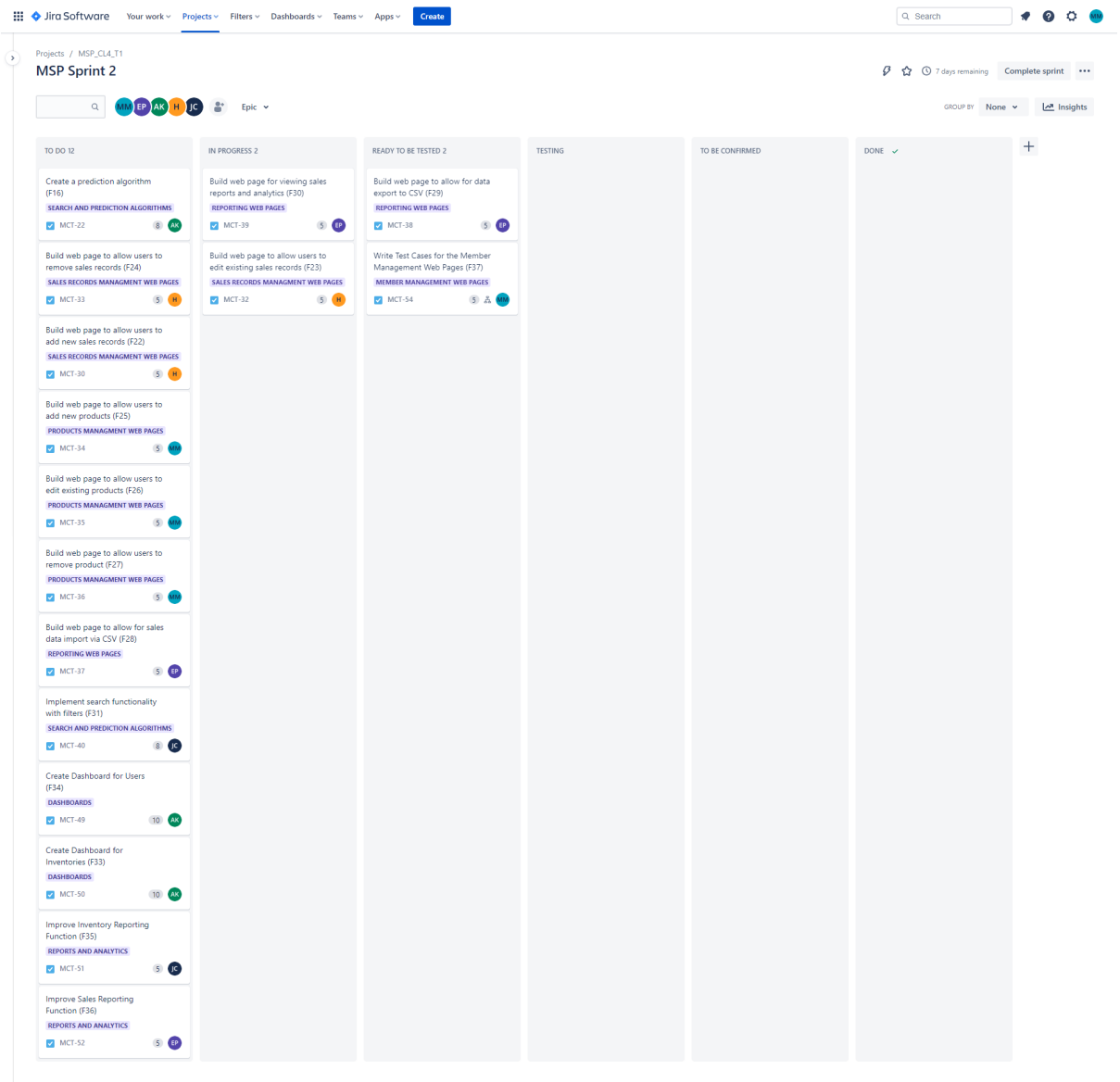
Meeting Minutes:

	Alexander Kyriacou	Julian Codespoti	Nelchael Kenshi Turija	Enzo Peperkamp	Marella Morad	Lachlan Martin
What are you working on?	I work on Tuesdays and Wednesdays and so didn't get much time to work on it today. Still tasked with working on the dashboard for the websites which will be a large task	Continued developing the products dashboard, focusing on graph visualizations for displaying diverse product information.	Create web pages for editing existing sales records	Completed CSV (F29). Initiated the development of a webpage to view sales reports and analytics (F30).	I finished my research about Jest yesterday and was able to set up a sample test file. Today, I will write the unit tests for MembersDasboard and AddMember components	I had a test to study for last night so I didn't have much time. Still looking at creating Java Project with Selenium Framework.
ETA	10 points still allocated	EOD + 1 Day	Total of 5 hours	EOD	EOD	3 hours
What's up next?	Work on the testing of the dashboards as per my distinction task	Moving towards infrastructure tasks, planning to update Google platform services infrastructure and modify the pipeline.	Continuing the task listed above	Conclude work on F30.	Writing the Unit tests for the Member component as well as writing the usability testing cases.	After Testing Project is set up, await US for testing.
Any Blockers?	N/A	N/A	I was able to create a barebones page that can be entered through the homepage. However, when entering a search query in the text box, it comes up with an unexpected error. Will continue working on this tomorrow.	N/A	N/A	N/A
The number of hours still required for each task you worked on	10	Products Dashboard: 5 Hours	3 hours.	F30: 4 hours	N/A	N/A

Date: 04/10/2023

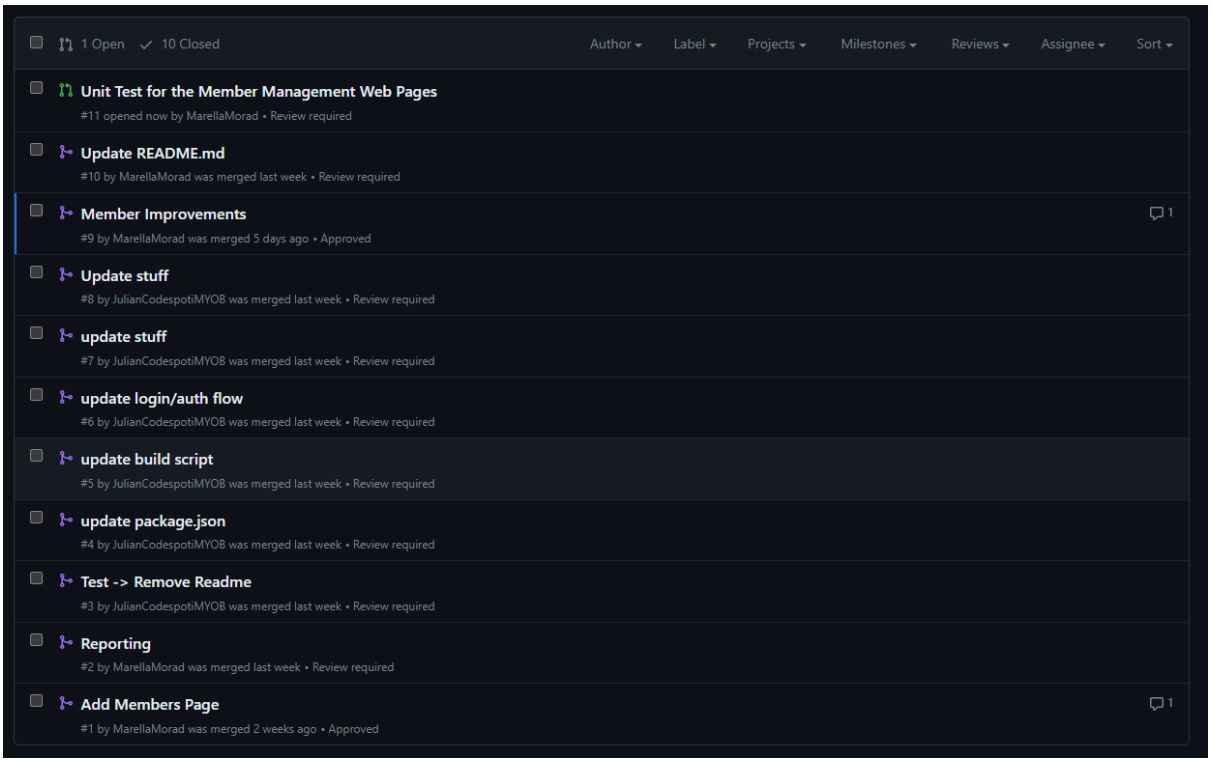
Several tasks were ready for testing, including the CSV data export and the member management test cases. Additionally, we've initiated work on new tasks, such as developing the web page for viewing sales reports and analytics. Meanwhile, the editing of existing sales records continues to make progress.

Task Board:



Burndown Chart:

Project Repository Status:



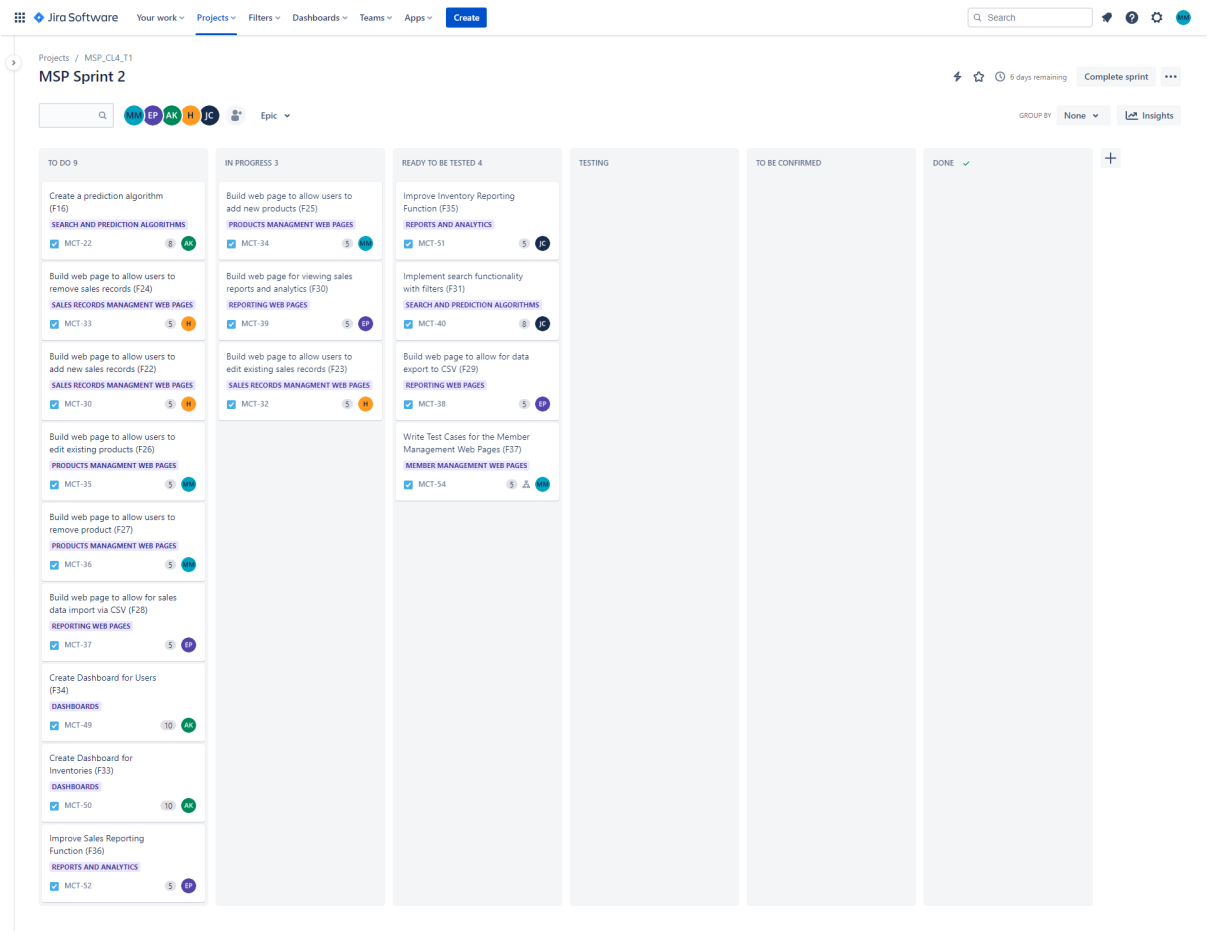
Meeting Minutes:

	Alexander Kyriacou	Julian Codespoti	Nelchael Kenshi Turija	Enzo Peperkamp	Marella Morad	Lachlan Martin
What are you working on?	Given i also worked all day today i was unable to work on the sprint today, i have a job interview all day tomorrow and so am hoping to get a good chunk of my sprint items items done after then when i have more time	Addressed updates on the Google platform services infrastructure and made crucial alterations in the pipeline to modify resources and CPU allocations.	Figuring out how to fix the error when searching records within the SaleRecords database	Ongoing development of the web page for sales report and analytics (F30).	I finished writing the unit tests for all Member Management Web pages as well as the writing usability testing cases.	Change of plans, I will stick with Selenium IDE. The basic setup is complete.
ETA	N/A	EOD	EOD	EOD	EOD	EOD
What's up next?	Beginning work on the dashboards task.	Finalizing the sales dashboard and initiating testing phases for implemented functionalities.	Editing functionality	Conclude work on F30 Begin enhancement of the Sales Reporting Function (F36).	Design and Develop the web page to add new products	A couple of items have moved to testing, will pick it up and complete automated testing.
Any Blockers?	Currently my only blocker is myself as my current schedule is quite busy	N/A	N/A	N/A	N/A	N/A
The number of hours still required for each task you worked on	N/A	Infrastructure Update: 3 Hours	1 hour to implement the editing functionality	F30: 1 hour	N/A	N/A

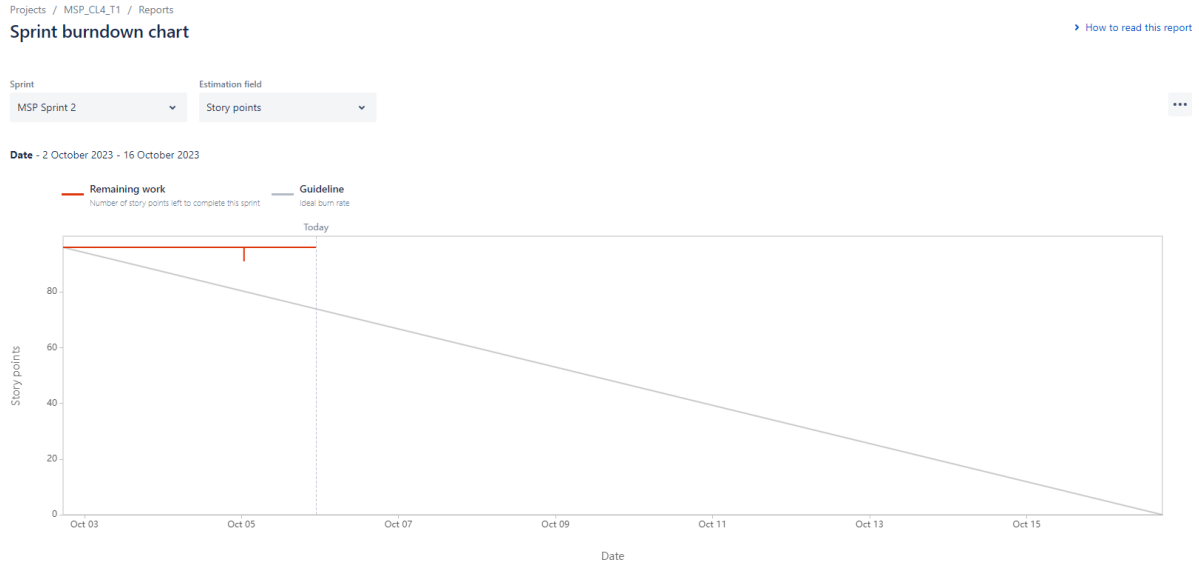
Date: 05/10/2023

We continued working on the web pages that enable users to edit existing sales records. Additionally, we initiated the development of a web page for adding new products. Furthermore, several other tasks have been started, and the development work for these tasks has been completed. These include improving the inventory reporting function and implementing search functionality with filters. These tasks are now marked as "ready to be tested".

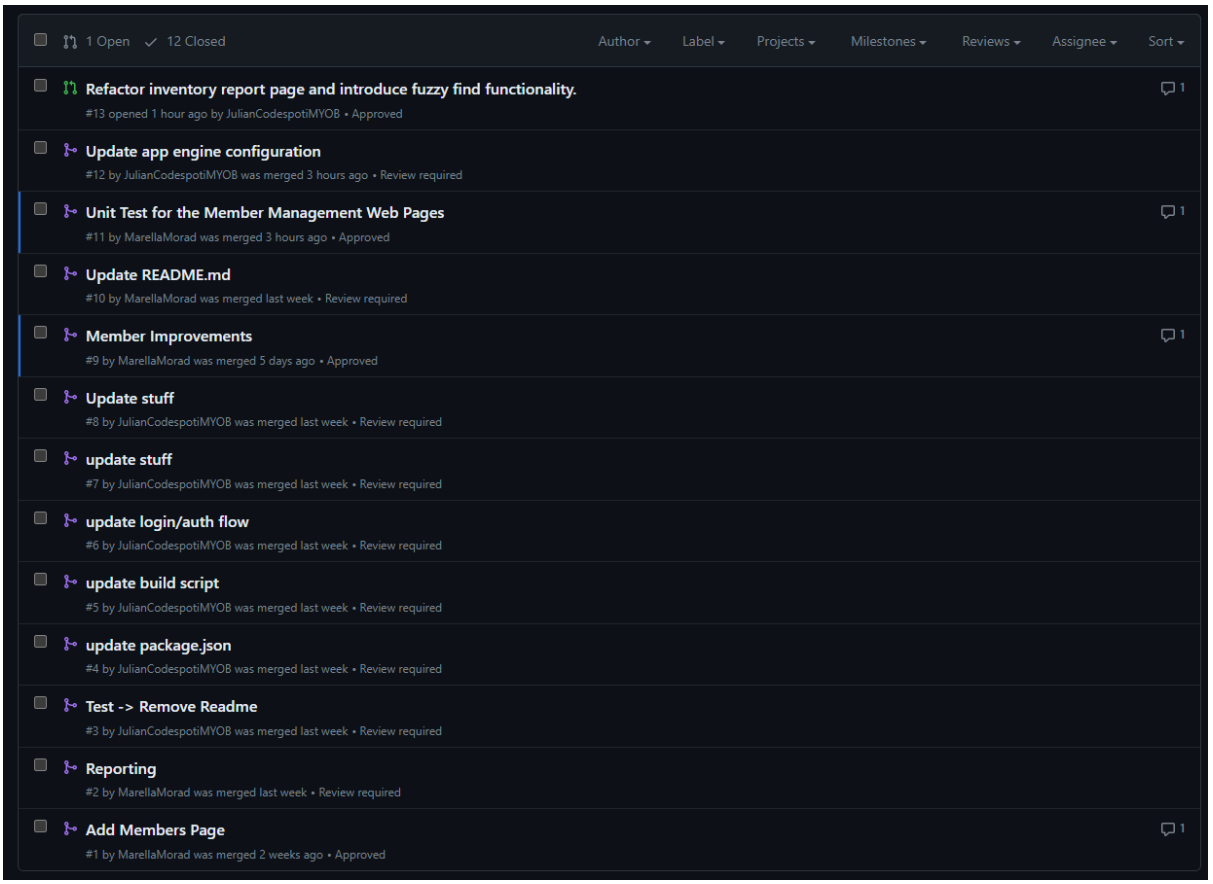
Task Board:



Burndown Chart:



Project Repository Status:



Meeting Minutes:

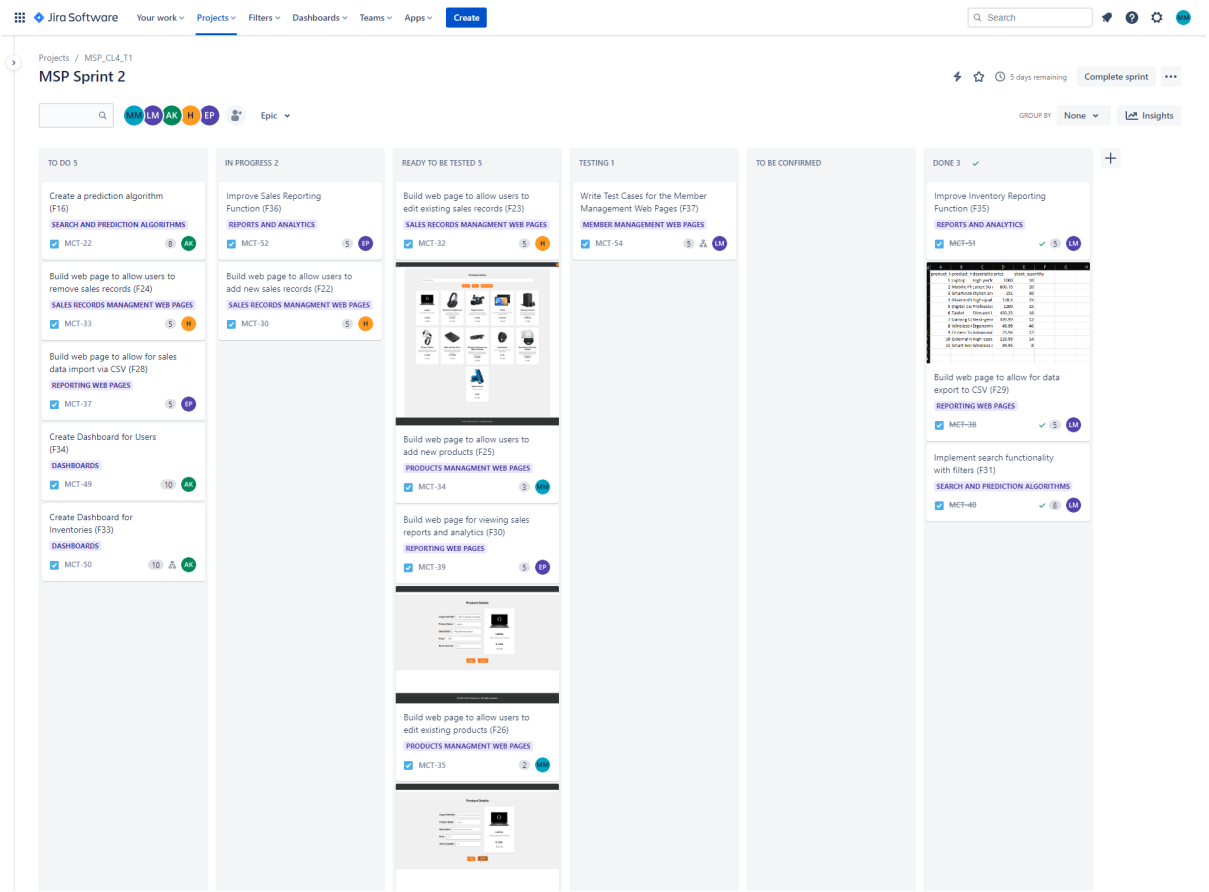
	Alexander Kyriacou	**Julian Codespoti**	**Nelchael Kenshi Turija**	**Enzo Peperkamp**	**Marella Morad**	**Lachlan Martin**
What are you working on?	As discussed yesterday, I had a job interview for most of today, leaving me with little time to work on the group project.	Finalizing the sales dashboard and initiating preliminary testing to ensure functionality and data accuracy.	Finishing the editing functionality and starting the adding records functionality to Sales Records Database.	Experienced delay in constructing the web page. Was unable to make progress today due to unforeseen work commitments.	Today I designed and developed the product dashboard webpage, and also started developing the add product page.	Testing for MCT-38, MCT-51, MCT-40.
ETA	N/A	EOD	EOD	EOD	EOD	EOD
What's up next?	Beginning work on the dashboards task.	Delving deeper into testing phases, ensuring all elements are functioning as intended and making any necessary adjustments.	Continuing to work on adding functionality	Begin enhancement of the Sales Reporting Function (F36).	Finishing the add product page	MCT-54
Any Blockers?	Currently my only blocker is myself as my current schedule is quite busy	Encountered a minor issue with data consistency in graph visualizations, working towards resolution.	N/A	Current workload prevented any progress today.	N/A	N/A
The number of hours still required for each task you worked on	N/A	Finalizing and Testing Sales Dashboard: 4 Hours	4 hours	F30: 1 hour	2 hours on the add product page	3 hours on MCT-38, MCT-51, MCT-40.

Date: 06/10/2023

As we wrap up week 1 of Sprint 2, we've been diligently working through our backlog. We've introduced some new tasks, including improving the sales reporting function and constructing the web page for adding new sales records. Notably, the web page for editing existing sales records and all the pages within the Product Management Web Pages epic are now ready for testing.

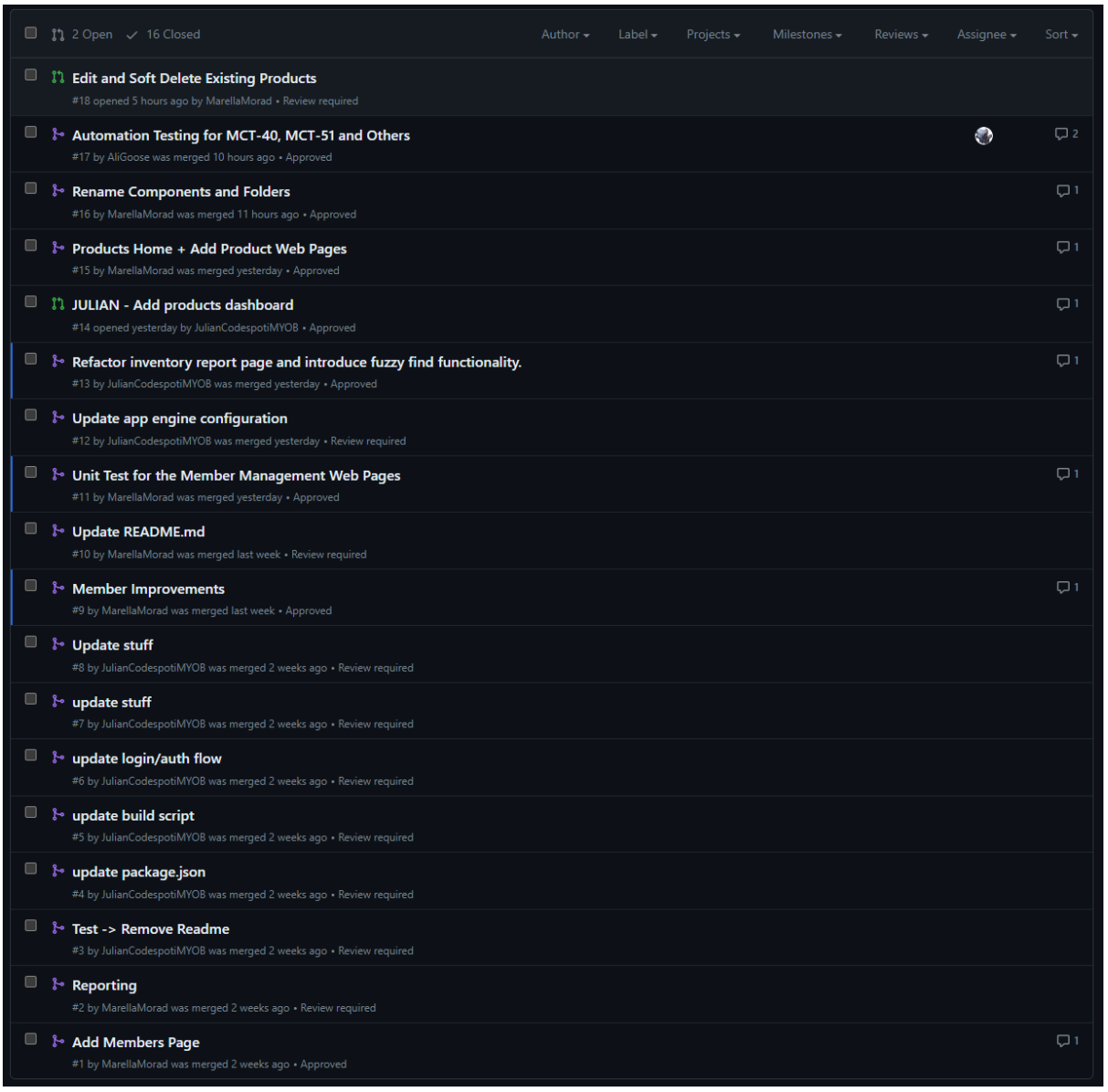
Furthermore, as part of our workflow, after completing development and testing on select tasks, they progress to the "to be confirmed" stage. Upon approval by the Scrum Master and/or Product Owner, they are then marked as "done."

Task Board:



Burndown Chart:

Project Repository Status:



Meeting Minutes:

	Alexander Kyriacou	Julian Codespoti	Nelchael Kenshi Turija	Enzo Peperkamp	Marella Morad	Lachlan Martin
What are you working on?	Today I began work on the dashboard functionality. This initial day has mostly been about planning the test plan going forward as well as the libraries that will be used, and will begin the actual development over the weekend.	Resolved the data consistency issue in the sales dashboard and moved forward with comprehensive testing of both the sales and products dashboards.	Adding records functionality	Initiated enhancement of the Sales Reporting Function (F36).	Finished the add product page	MCT-54
ETA	Hoping to be done by Monday next week	EOD	EOD	EOD	EOD	3 hours
What's up next?	Product dashboard will be next	Addressing any issues arising from the testing phase and preparing for a review and feedback session.	Continuing to complete the adding records functionality	Finish F36. Begin development of a web page for importing sales data via CSV (F28).	I'll wait for test results and any issues raised by the tester	Testing backlog has about 4 US related to product
Any Blockers?	N/A	N/A	N/A		N/A	N/A
The number of hours still required for each task you worked on	6 hours	Addressing Feedback and Final Adjustments: 3 Hours	1 hour	F36: 2.5 hours	N/A	N/A

Sprint 2 Final Check: Management System (GotoGro-MRM) for Goto Grocery Inc.

Team: MSP_CL4_T1	
Name	**ID**
Enzo Peperkamp	*102895415*
Nelchael Kenshi Turija	*103057559*
Julian Codespoti	*102997816*
Alex Kyriacou	*103059830*
Marella Morad	*103076428*
Lachlan Martin	*103067448*

Contribution statements:

Enzo Peperkamp - 102895415

Reflecting on our sprint, the completion of our sprint despite university and work commitments acting as a roadblock, signifies our adaptability and persistent drive. Balancing academia and project responsibilities has fostered a nuanced understanding of time management and prioritization. It's gratifying to look back at the diverse challenges and new learnings each stage brought us, all contributing towards a product that we hope will bring significant value to our users.

Nelchael Kenshi Turija - 103057559

I believe that everyone has contributed equally for sprint 2. The effective communication and collaboration the team has maintained throughout the sprint were instrumental in the achievement of our goals. I also successfully completed the development of the sale records editing, deleting and adding page functionality. Additionally, I provided support in the creation of the necessary reports that were due during the Sprint 2 timeframe.

Julian Codespoti - 102997816

In reflecting upon Sprint 2, the progress and camaraderie within our team have been uplifting. Tackling the inventory and sales dashboards with Alex, we exploited Chart.JS to create both user-friendly and functional displays of crucial data. Assisting with bug fixes and team reports also provided a window into collaborative troubleshooting and enhanced our project's robustness. Through hurdles and triumphs, this sprint has been a mosaic of collaboration, innovation, and relentless pursuit toward delivering a user-centric project. Working as a developer for the team is a refreshing role, not deviating too far from the processes undertaken at my workplace.

Alex Kyriacou - 103059830

I believe that each of our team members has been able to contribute a fair amount within this sprint. This marks almost the end of the semester for this project and I think we have managed to work together very effectively throughout. I have done my best to support the rest of the team in both the assignments as well as the development work. In addition, my work as the Product Owner has meant that I have been responsible for the scope of the last two sprints as well.

Marella Morad - 103076428

I believe our team operated more effectively during this sprint. The workload was distributed more evenly, and everyone contributed equally. This improvement can be attributed to the valuable experience gained from Sprint 1.

In this sprint, my primary focus was on the web pages, particularly the Product management pages. I also dedicated time to creating reports, responsible for capturing screenshots over the two-week duration of the sprint. Additionally, I conducted usability testing on the web application during the last two days and addressed any issues that arose, leveraging my skills as a developer.

Lachlan Martin - 103067448

I believe I made a valuable contribution to the team. I took a large workload of setting up and developing our automation testing system. The team was quick to implement fixes to defects I had detected. I hope the team appreciates the automation testing I provided for their web app. I enjoy working as a quality engineer for the team.

Table of Contents

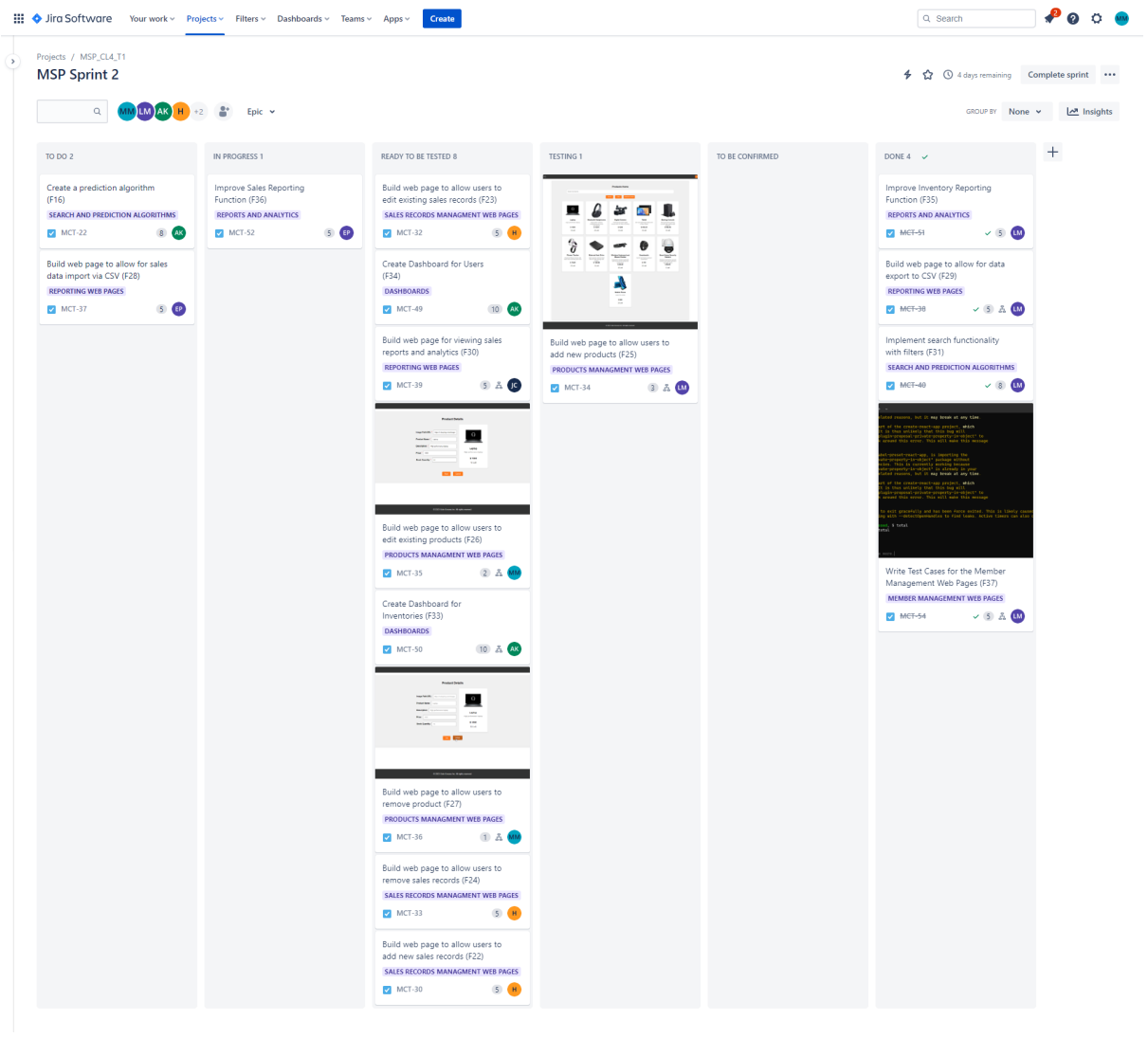
Resources and URLs:

JIRA Board Link
Burn-down Chart Link
Github Repo Link
GotoGro Live Website

Date: 09/10/2023:

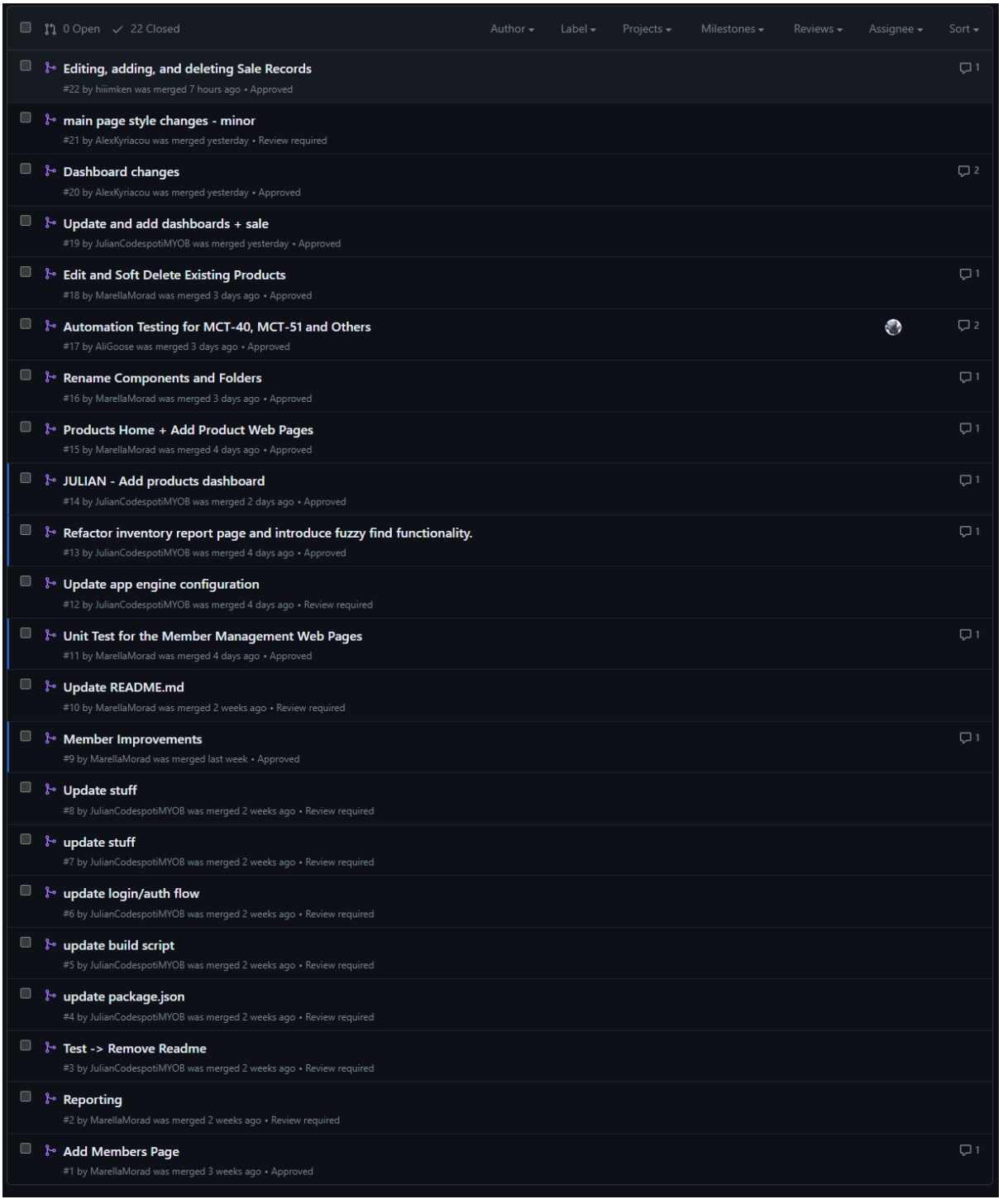
Today, we finished developing the "Edit Existing Sales Records" page, and it's now marked as "ready for testing." We've also made progress on enhancing the sales reporting function, although there's still some work to be done there. In addition, we successfully tested several tasks, which were moved to the "to be confirmed" stage. Our Scrum Master has reviewed and confirmed them, moving them to "done".

Task Board:



Burndown Chart:

Project Repository Status:



MSP_CL4_T1 Monday 2:30 PM - 4:30 PM BA708

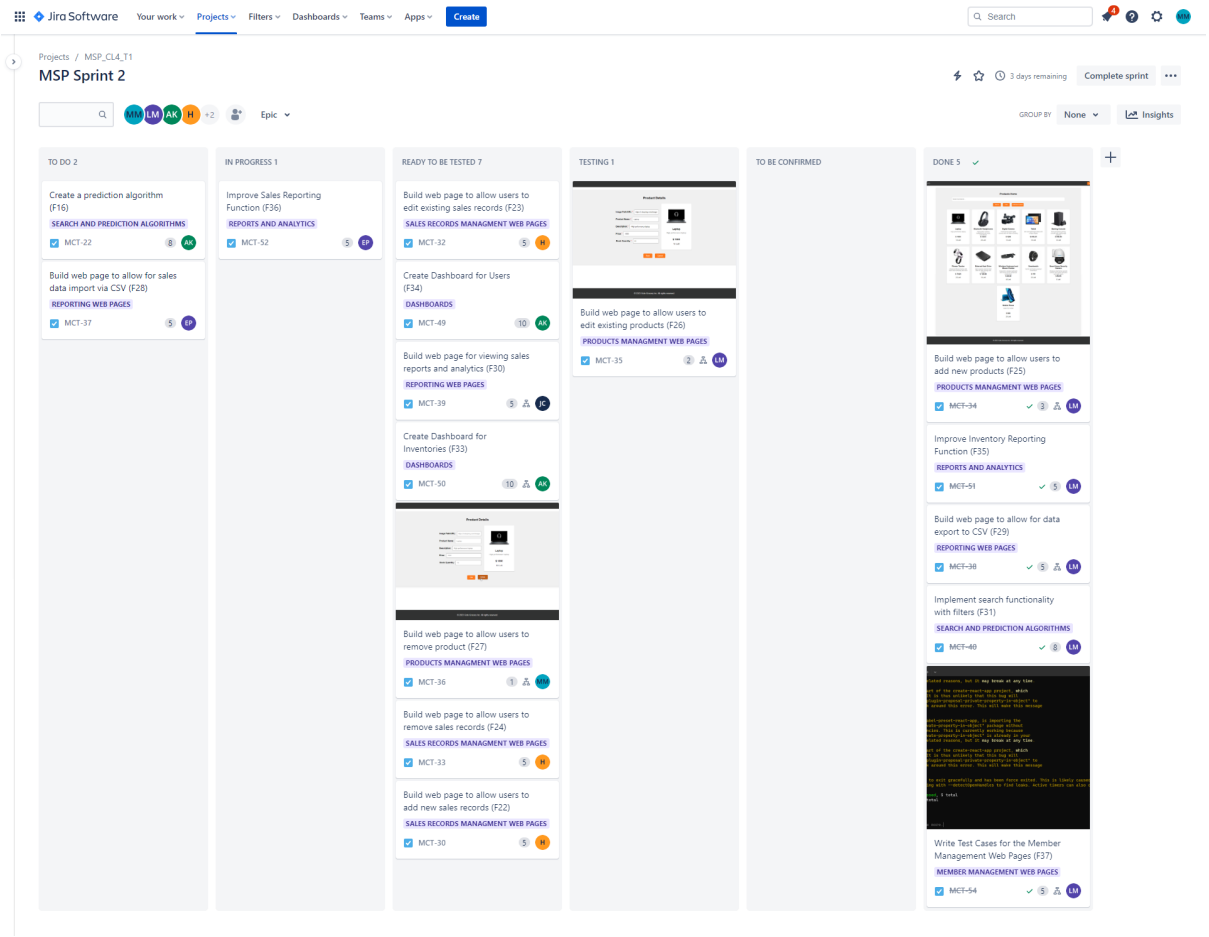
Meeting Minutes:

	Alexander Kyriacou	**Julian Codespoti**	**Nelchael Kenshi Turija**	**Enzo Peperkamp**	**Marella Morad**	**Lachlan Martin**
What are you working on?	Today, both myself and Julian were able to get a large amount of the initial work out of the way for the inventory dashboards. We have decided to use Chart.JS to do this and have an initial draft together that has most of the core information within it.	Similarly to Alex, we have both collectively worked on a large portion of the inventory dashboard works. Utilizing Chart.Js, we have put together a proof of concept that possesses a great deal of the core functionality.	Finishing both adding and deletion of the sales record database	Had to prolong efforts on the Sales Reporting Function (F36) because of unforeseen adjustments. Despite that, F36 has been successfully completed.	Today I worked mainly on getting our reports done and filled out so that they are ready to be submitted	Begen to work on F37.
ETA	We are hoping to get this done by EOD tomorrow	EOD Tomorrow	EOD	EOD	EOD	EOD
What's up next?	Once this is done we can work on getting the sales dashboard also put in place.	Sales Dashboard	Helping team with reports and other tasks that need to be done	Start development of a web page for importing sales data via CSV (F28)	I'm working tomorrow and Wednesday, so I won't be able to do much for this, I will help with testing and fixing bugs if I get a chance	After this, I'll pick up the items related to the product webpage.
Any Blockers?	N/A	N/A	N/A	Unforeseen work needed for F36	N/A	N/A
The number of hours still required for each task you worked on	~2 hours	~2 Hours	N/A	N/A	N/A	2 hours

Date: 10/10/2023:

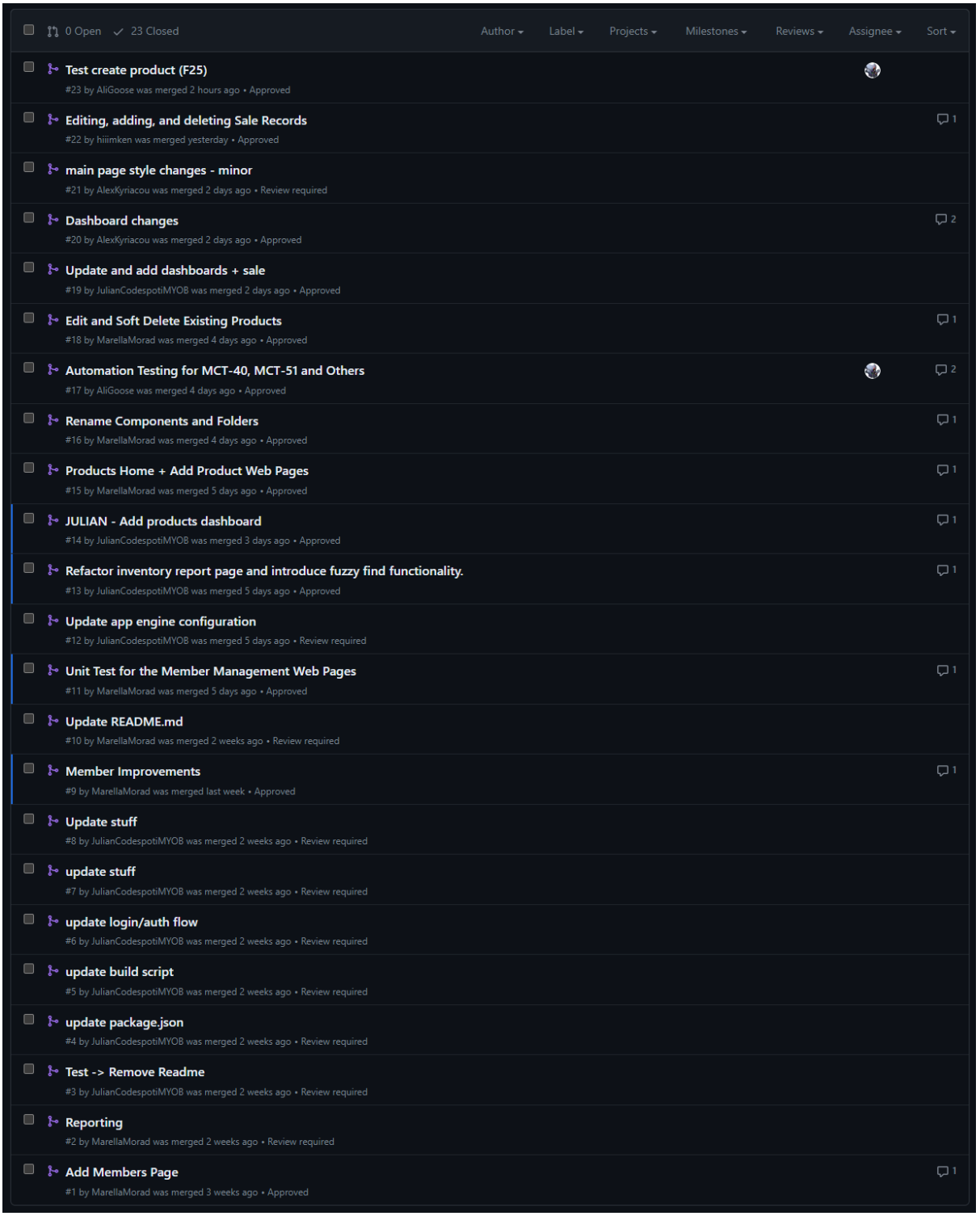
During this sprint, we successfully tested and completed certain tasks, including the evaluation of the "Add New Products" page. We also allocated time to enhance the functionality of our sales reporting feature. In addition to these tasks, we devoted a portion of our efforts to refining the reports required for this unit.

Task Board:



Burndown Chart:

Project Repository Status:



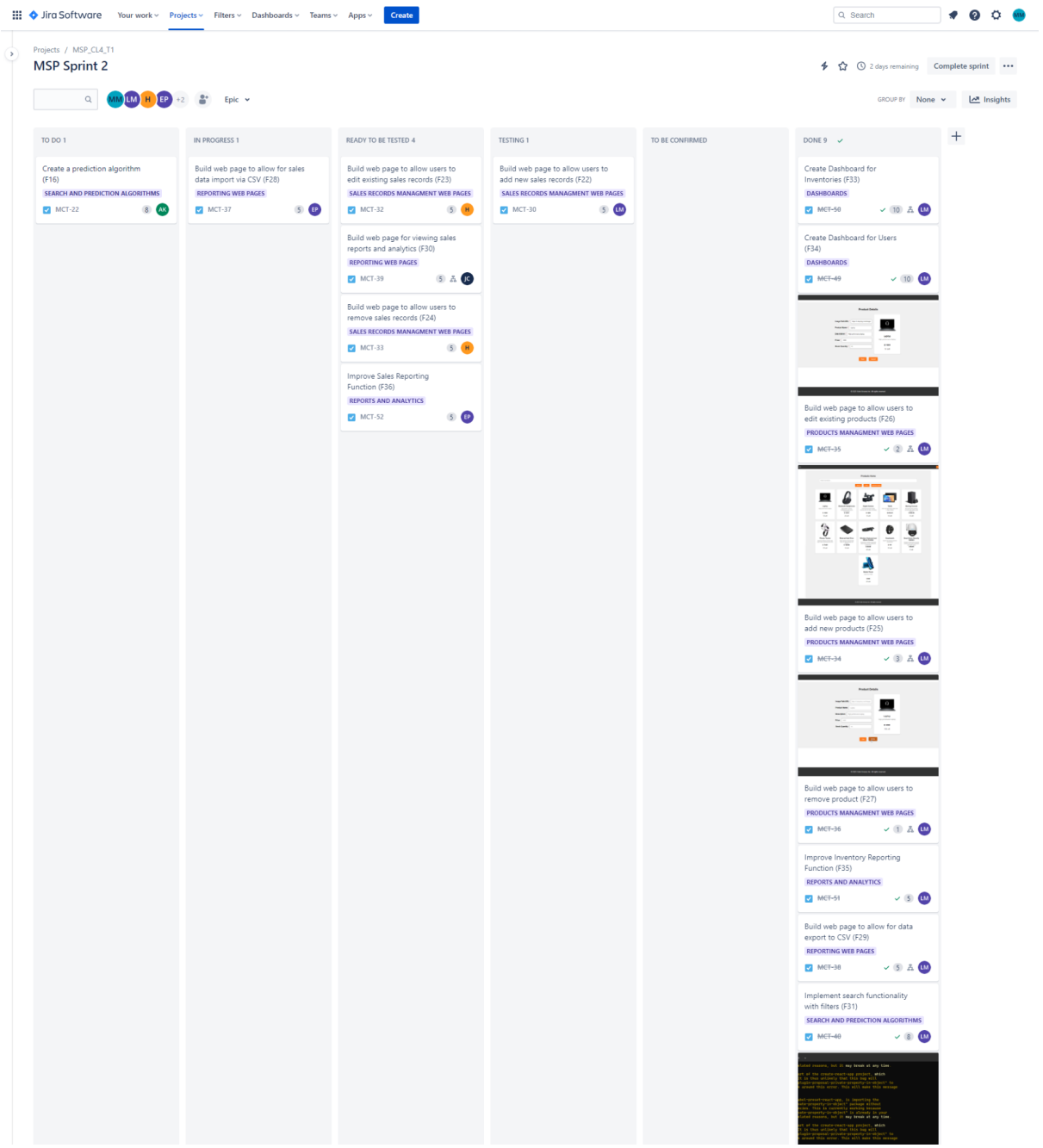
Meeting Minutes:

	Alexander Kyriacou	**Julian Codespoti**	**Nelchael Kenshi Turija**	**Enzo Peperkamp**	**Marella Morad**	**Lachlan Martin**
What are you working on?	Today Julian and I finished the inventory dashboards. All that is left for this is both the functional and usability testing for this task.	Similarly to Alex, we have collectively worked on the inventory dashboards, finishing all of the necessary work for this task.	I've been busy with work and life, so I haven't had the chance to work on this project	Initiated the development of a webpage for importing sales data via CSV (F28).	Today I ran the usability test cases on the member management web pages and I found a few issues with form validation and error handling on which I spent some time fixing	I've picked up F25, F26, and F27 for creating test automation
ETA	Done	Done	N/A	EOD	N/A	EOD
What's up next?	Working on The Sales Dashboards	Working on the Sales Dashboards	Helping team with reports and other tasks that need to be done	Proceed with F28 development.	Fix issues with the Member web paegs	After this I'll pick up the Dashboard stories.
Any Blockers?	N/A	N/A	N/A	N/A	N/A	N/A
The number of hours still required for each task you worked on	N/A	N/A	N/A	F28: 3.5 hours	1 hour	2 hours

Date: 11/10/2023:

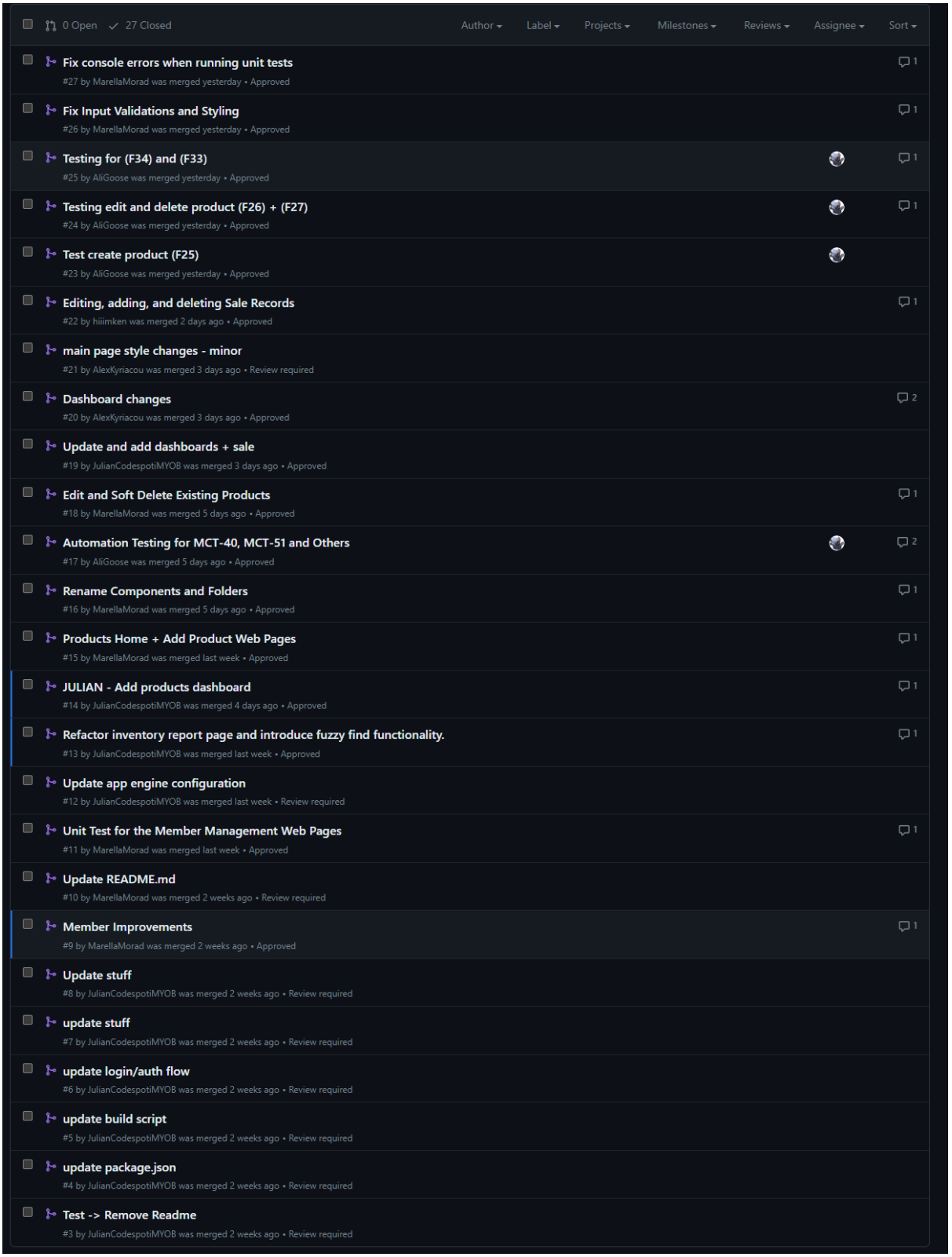
As depicted in the screenshot below, the majority of tasks have been developed and are now in a state where they are ready for testing. In the upcoming days of the sprint, our primary objective is to allocate various team members to test different pages and subsequently provide their feedback and concerns. This collaborative effort will ensure any issues are identified and addressed by the developers promptly.

Task Board:



Burndown Chart:

Project Repository Status:



MSP_CL4_T1 Monday 2:30 PM - 4:30 PM BA708

Meeting Minutes:

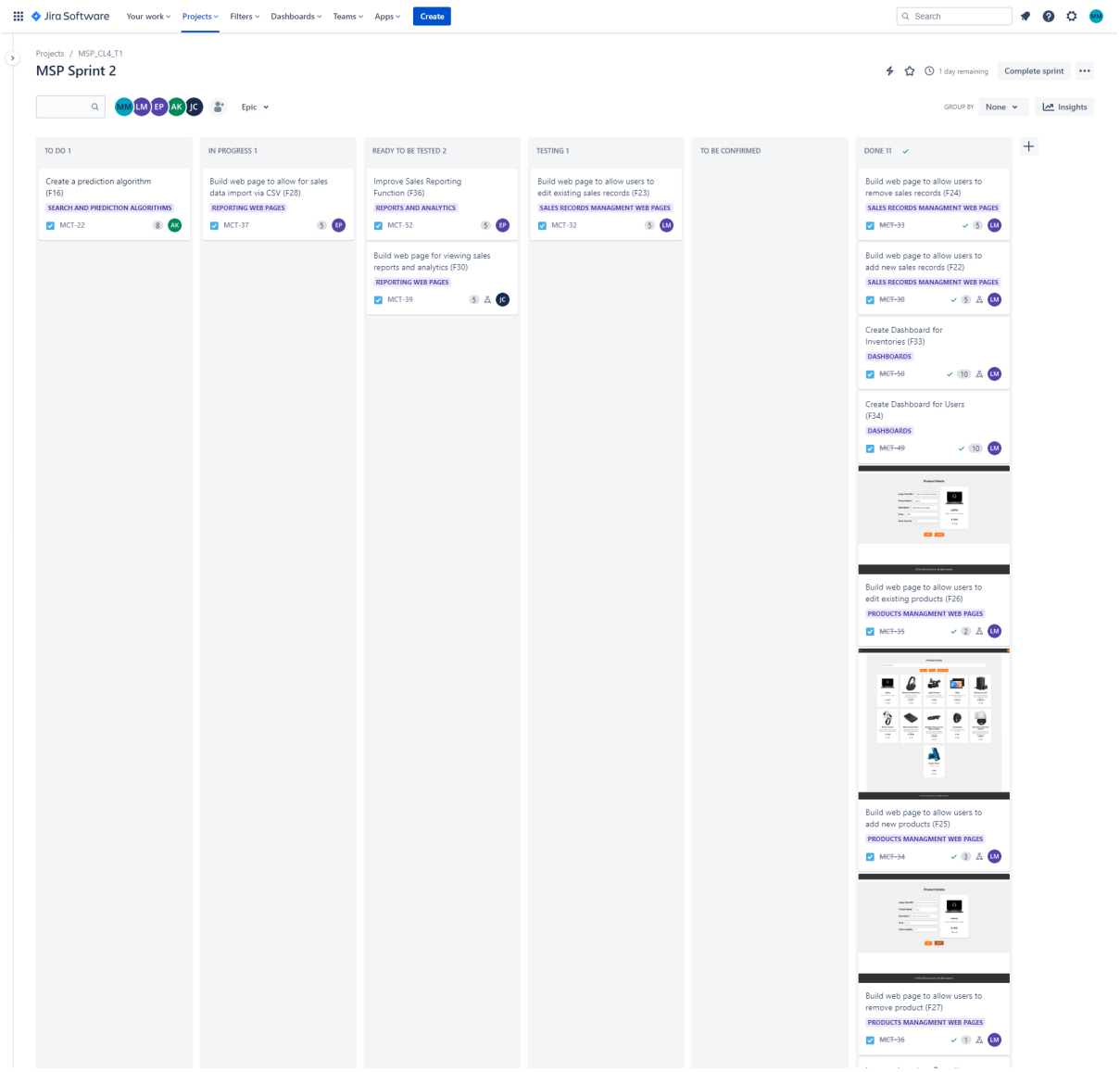
	Alexander Kyriacou	Julian Codespoti	Nelchael Kenshi Turija	Enzo Peperkamp	Marella Morad	Lachlan Martin
What are you working on?	Today Julian was able to both start and finish the sales dashboards. This was quicker than we expected due to the amount we were able to reuse from the inventory dashboards. This is now able to be passed to Lachlan for testing	Similarly to Alex, we have been able to finish the work surrounding the sales dashboards. This was done with ease as the implementation to the inventory dashboards was very similar, and thus, we could reuse a lot of the logic here.	Helping team with reports and other tasks that need to be done	Could not work - other urgent university commitments.	Today I was working and I didn't get a chance to work on this project	Working on F33 and F34.
ETA	Done	Done	EOD	-	EOD	EOD
What's up next?	I need to complete the usability testing of the inventory dashboards as per my distinction task		Helping team with reports and other tasks that need to be done	Proceed with F28 development.	Continue fixing the raised issues	Will work on the sales webpages so F22, F23, and F24.
Any Blockers?	N/A	N/A	N/A	University commitments	N/A	N/A
The number of hours still required for each task you worked on	~2 hours required to perform the usability evaluation as per my distinction task	~2 hours for usability testing.	N/A	F28: 3.5 hours	2 hours fixing bugs	1 hour.

Date: 12/10/2023:

Today, a significant portion of our time was dedicated to testing our web application and addressing any identified bugs. Notably, there were issues related to the sales record and product pages that were promptly rectified by our developers. These fixes, however, may not be reflected in the Task Board, as they were discovered and resolved internally without the need to raise formal bug reports.

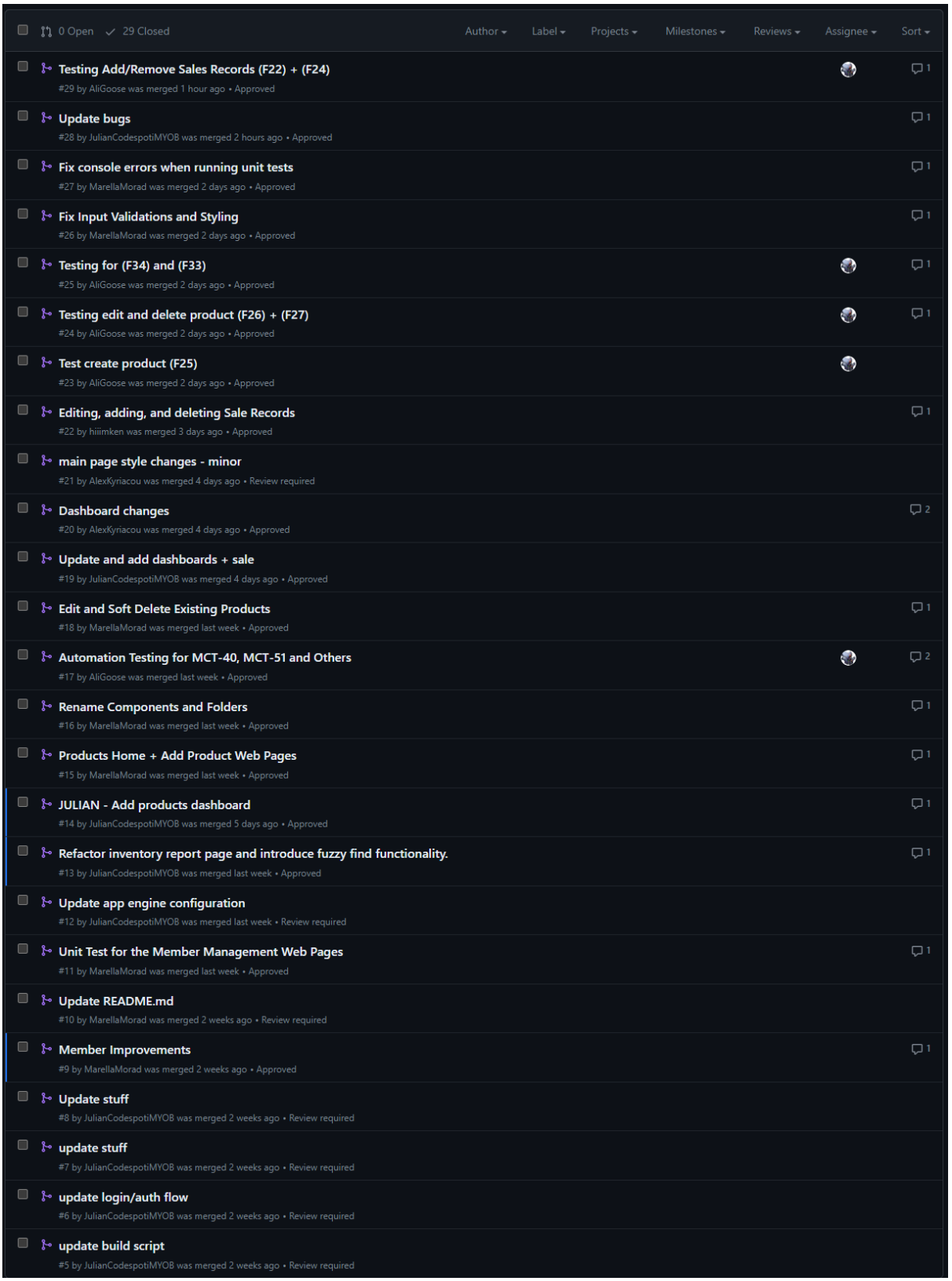
Additionally, we allocated time to further improve our automated testing process. This step is crucial because with each new feature or fix pushed to the application, there is a potential for existing tests to become outdated and fail. This proactive approach ensures that our testing remains up-to-date and comprehensive.

Task Board:



Burndown Chart:

Project Repository Status:



MSP_CL4_T1 Monday 2:30 PM - 4:30 PM BA708

Meeting Minutes:

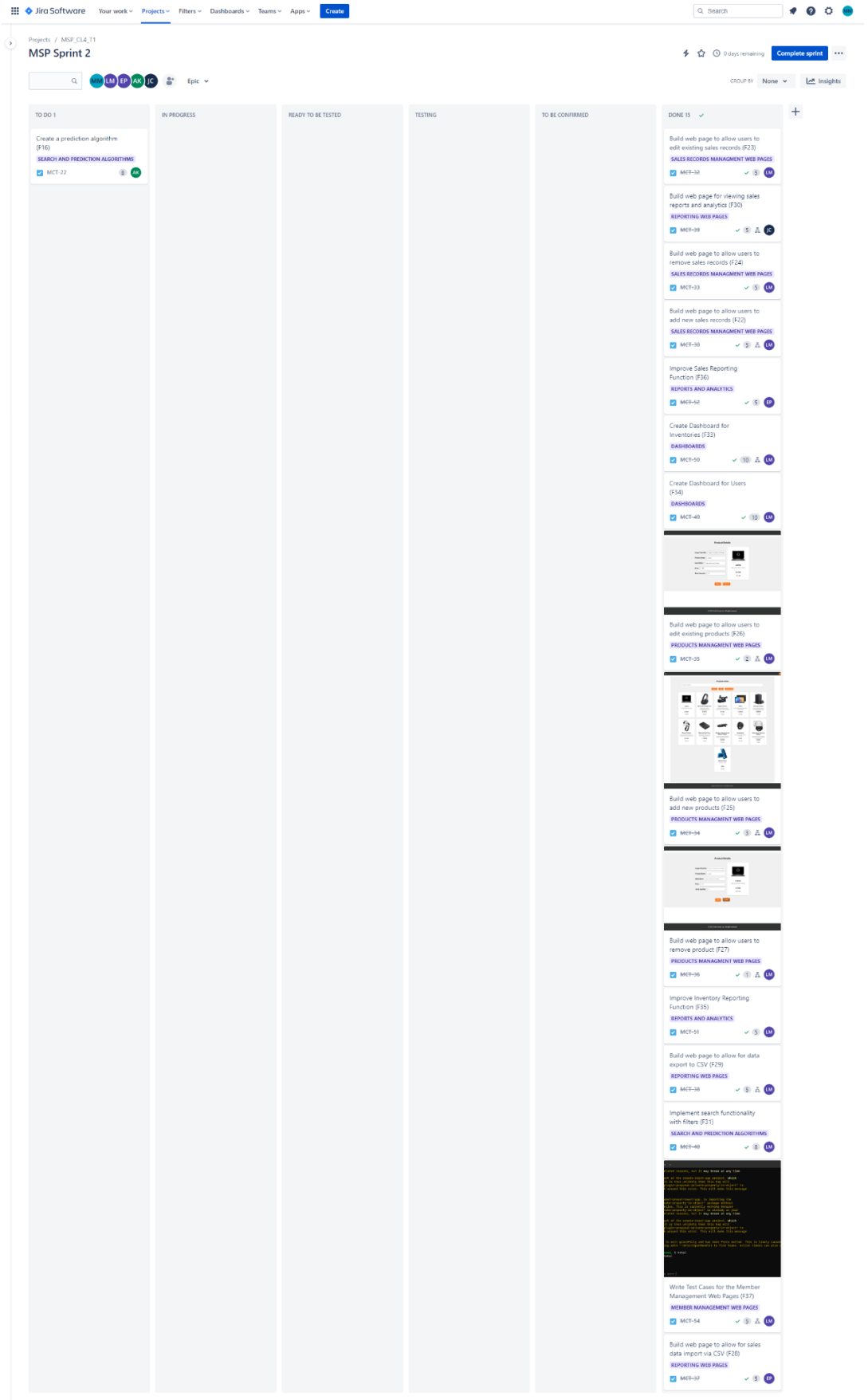
	Alexander Kyriacou	**Julian Codespoti**	**Nelchael Kenshi Turija**	**Enzo Peperkamp**	**Marella Morad**	**Lachlan Martin**
What are you working on?	Today I had a Job interview so was unable to work on this project today. Still working on putting together a usability evaluation of the inventory dashboards	Assisted the team with various programming-related tasks. Helped Ken with fixing the bugs associated with the 'sales records' they were working on.	Helping team with reports and other tasks that need to be done	Nearly completed F28.	Today I spent some time fixing bugs, but since they were not fully fixed, I did not merge them.	Completed F22 and F24. Still working on F23.
ETA	EOD Tomorrow	EOD	EOD	EOD	EDO	Likely End of tomorrow. Currently blocked for F23
What's up next?	Once this is done the sprint is over	Updating Github Actions CICD pipeline to run tests on every deployment.	Helping team with reports and other tasks that need to be done	Concluding F28 and reviewing all sprint items.	Go through all pages and make sure they are all consistent.	Likely F36
Any Blockers?	N/A	N/A	N/A	N/A		Edit functionality currently not working, waiting for fix to be developed.
The number of hours still required for each task you worked on	2 hours	N/A	N/a	F28: 1 hour	2 hours	N/A

Last Day of Sprint 2:

On the final day of our sprint, our focus was on moving all completed items to the "Done" category and obtaining approval from both the product owner and the scrum master. We continued to conduct internal testing and address any identified issues as part of our quality assurance process.

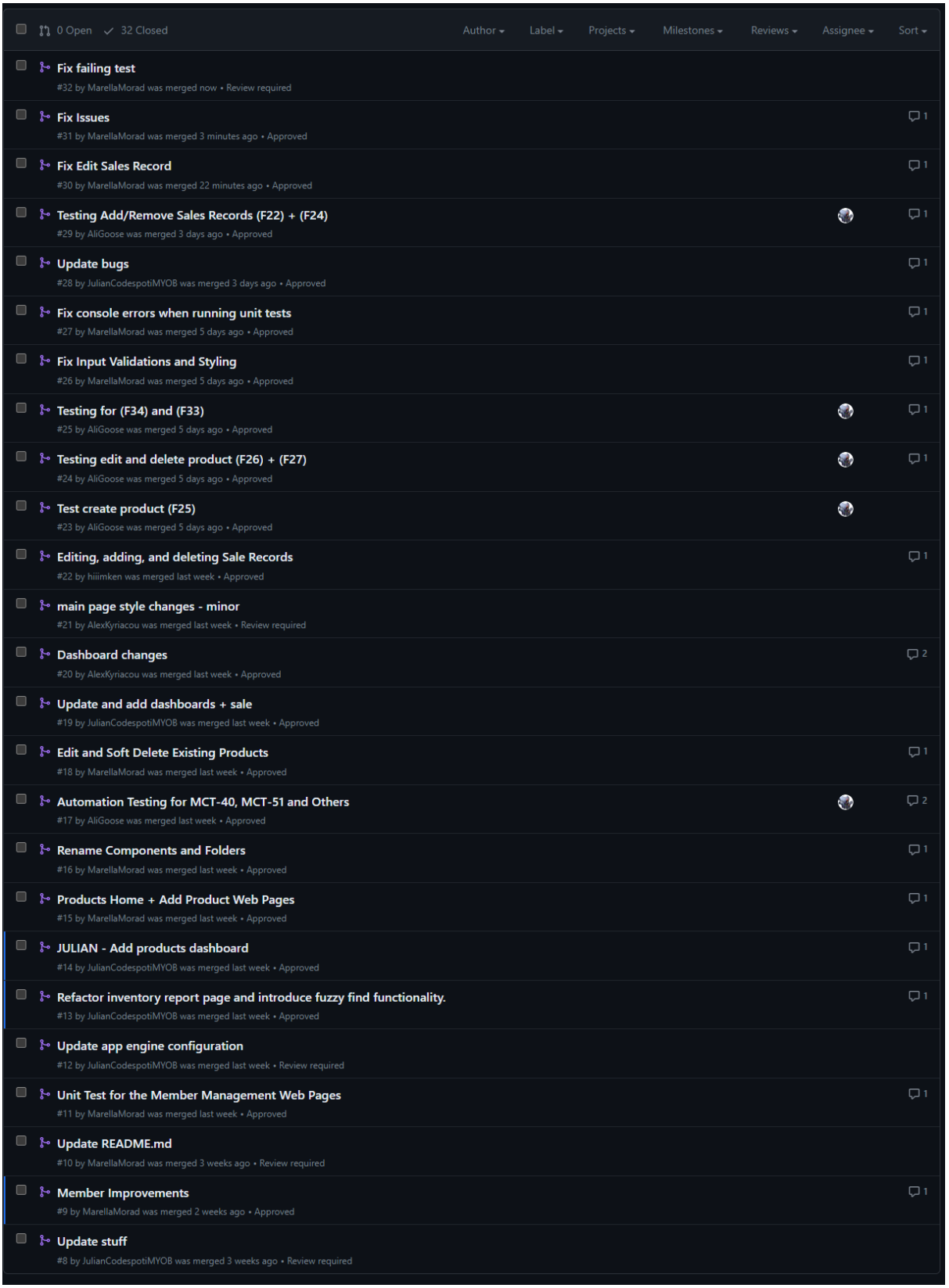
Despite some challenges, we are pleased to report that we've achieved all the goals we initially set for this sprint. The only exception was the "Prediction Algorithm" task, which was relatively ambiguous and more akin to an epic in terms of complexity. In our team standup today, we made the collective decision to postpone this task for future improvements. It's possible that we may have underestimated the effort required for this item. Nevertheless, overall, the team is content with the outcomes of this sprint.

Task Board:


Jira Software
Your work
Projects
Filters
Dashboards
Teams
Apps
Create
Search
Projects / MSP_CL4_T1
MSP Sprint 2
0 days remaining
Complete sprint
Epic
GROUP BY
None
Insights
TO DO 1
IN PROGRESS
READY TO BE TESTED
TESTING
TO BE CONFIRMED
DONE 15
Create a prediction algorithm (F16)
SEARCH AND PREDICTION ALGORITHMS
MCT-22
Build web page to allow users to edit existing sales records (F23)
SALES RECORDS MANAGMENT WEB PAGES
MCT-32
Build web page for viewing sales reports and analytics (F30)
REPORTING WEB PAGES
MCT-39
Build web page to allow users to remove sales records (F24)
SALES RECORDS MANAGMENT WEB PAGES
MCT-33
Build web page to allow users to add new sales records (F22)
SALES RECORDS MANAGMENT WEB PAGES
MCT-30
Improve Sales Reporting Function (F36)
REPORTS AND ANALYTICS
MCT-52
Create Dashboard for Inventories (F33)
DASHBOARDS
MCT-50
Create Dashboard for Users (F34)
DASHBOARDS
MCT-49
Build web page to allow users to edit existing products (F26)
PRODUCTS MANAGMENT WEB PAGES
MCT-35
Build web page to allow users to add new products (F25)
PRODUCTS MANAGMENT WEB PAGES
MCT-34
Build web page to allow users to remove product (F27)
PRODUCTS MANAGMENT WEB PAGES
MCT-36
Improve Inventory Reporting Function (F35)
REPORTS AND ANALYTICS
MCT-51
Build web page to allow for data export to CSV (F29)
REPORTING WEB PAGES
MCT-38
Implement search functionality with filters (F31)
SEARCH AND PREDICTION ALGORITHMS
MCT-40
Write Test Cases for the Member Management Web Pages (F37)
MEMBER MANAGEMENT WEB PAGES
MCT-54
Build web page to allow for sales data import via CSV (F28)
REPORTING WEB PAGES
MCT-37

Burndown Chart:

Project Repository Status:



Meeting Minutes:

	Alexander Kyriacou	**Julian Codespoti**	**Nelchael Kenshi Turija**	**Enzo Peperkamp**	**Marella Morad**	**Lachlan Martin**
What are you working on?	Finished the usability evaluation today. Marking the end of my sprint tasks	Updated the pipeline to run tests, added eslint - utilizing Airbnb's configuration - to the repository to ensure coding standards are consistent across said repository.	Helping team with reports and other tasks that need to be done	Completed F28. Reviewed and documented relevant code/features developed during this sprint.	Today I went through all the components and made them all consistent, from buttons to colours to transitions, etc.	Fixes have been implemented. Can finish F23 by end of day.
ETA	Done	Done	EOD	EOD	Done	EOD
What's up next?	Sprint is over	WOOHOO, sprint's done!	Sprint is done!	N/A	Sprint is over	Sprint is Done
Any Blockers?	N/A	N/A	N/A	N/A	N/A	N/A
The number of hours still required for each task you worked on	N/A	N/A	N/A	N/A	N/A	N/A

Sprint Review: Management System (GotoGro-MRM) for Goto Grocery Inc.

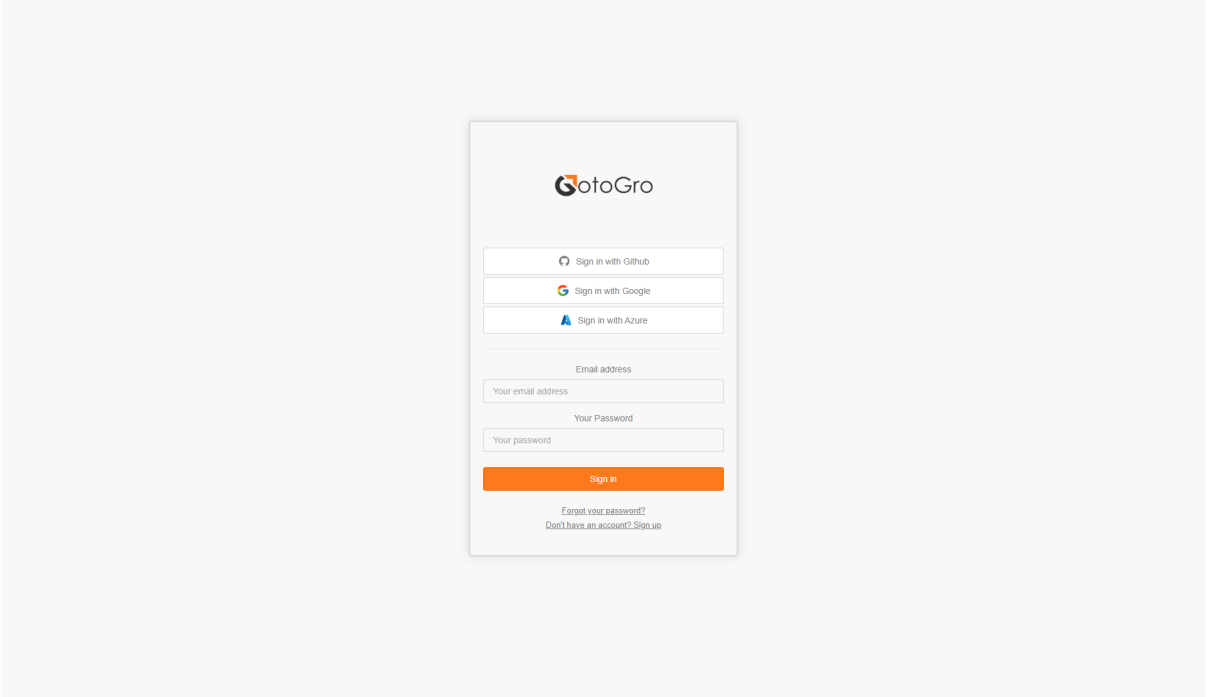
Team: MSP_CL4_T1	
Name	**ID**
Enzo Peperkamp	*102895415*
Nelchael Kenshi Turija	*103057559*
Julian Codespoti	*102997816*
Alex Kyriacou	*103059830*
Marella Morad	*103076428*
Lachlan Martin	*103067448*

Sprint Review Completion Evidence:

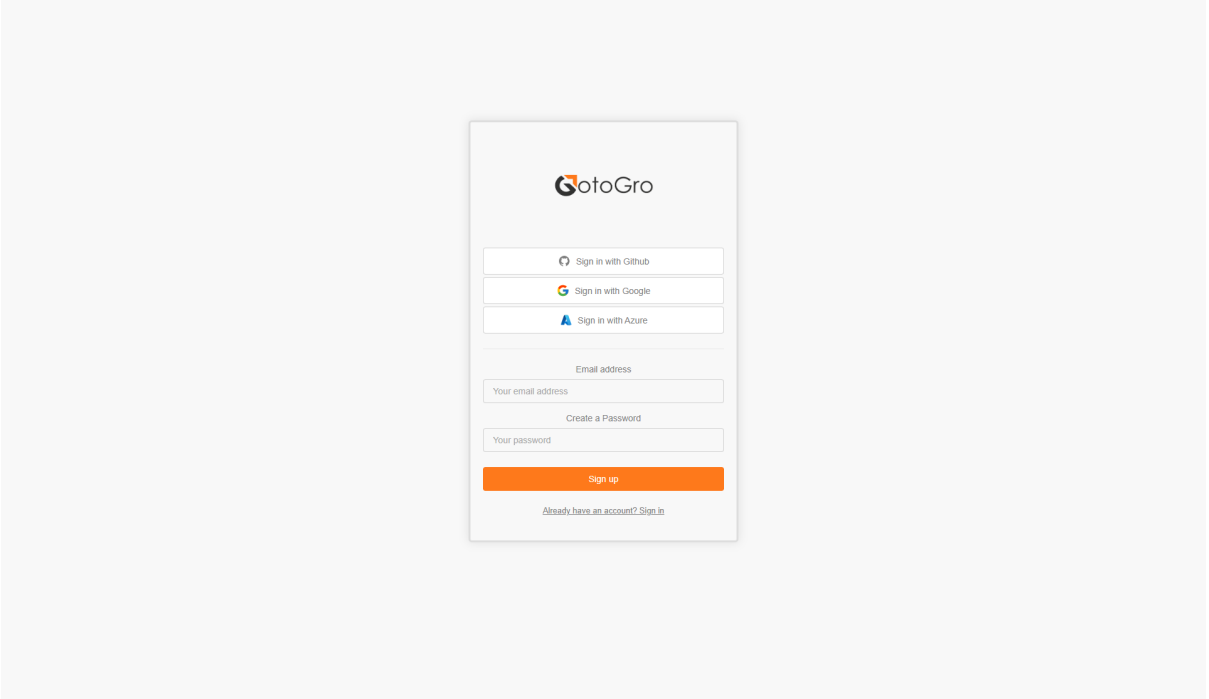
Link to GotoGro Live Website

Screenshots from various screens:

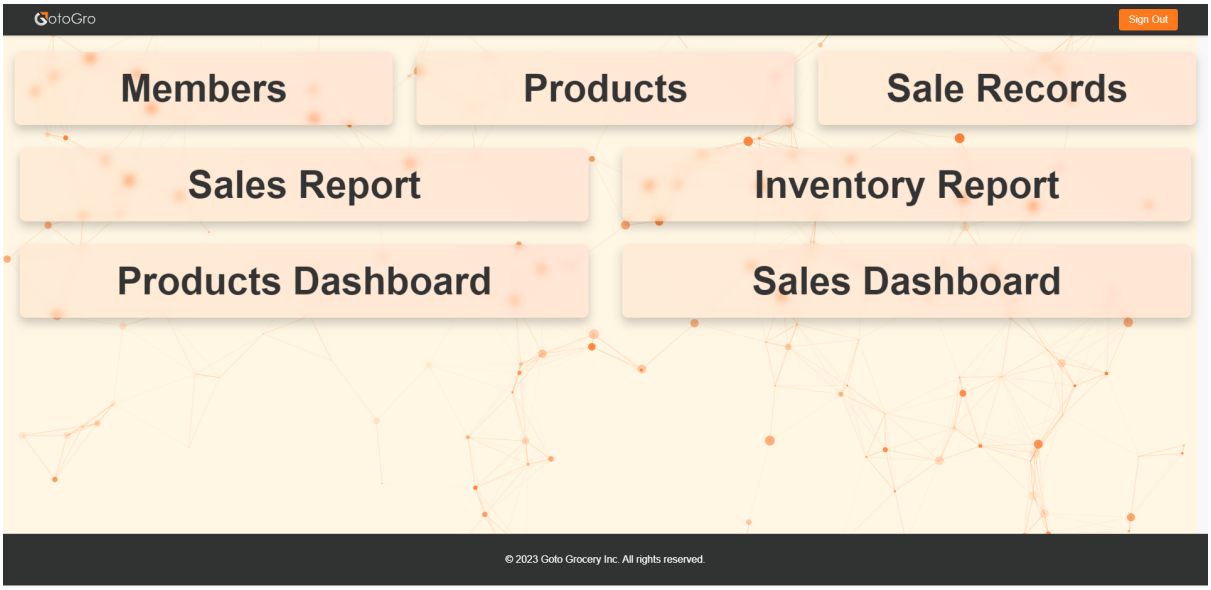
Sign In Page:



Sign Up Page:

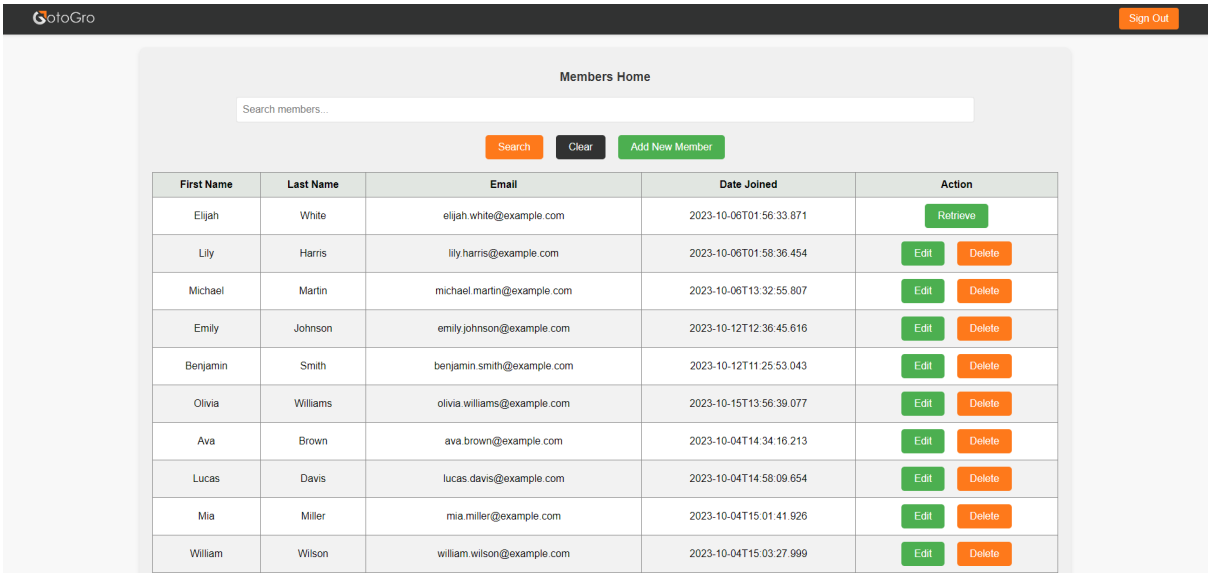


Home Page:

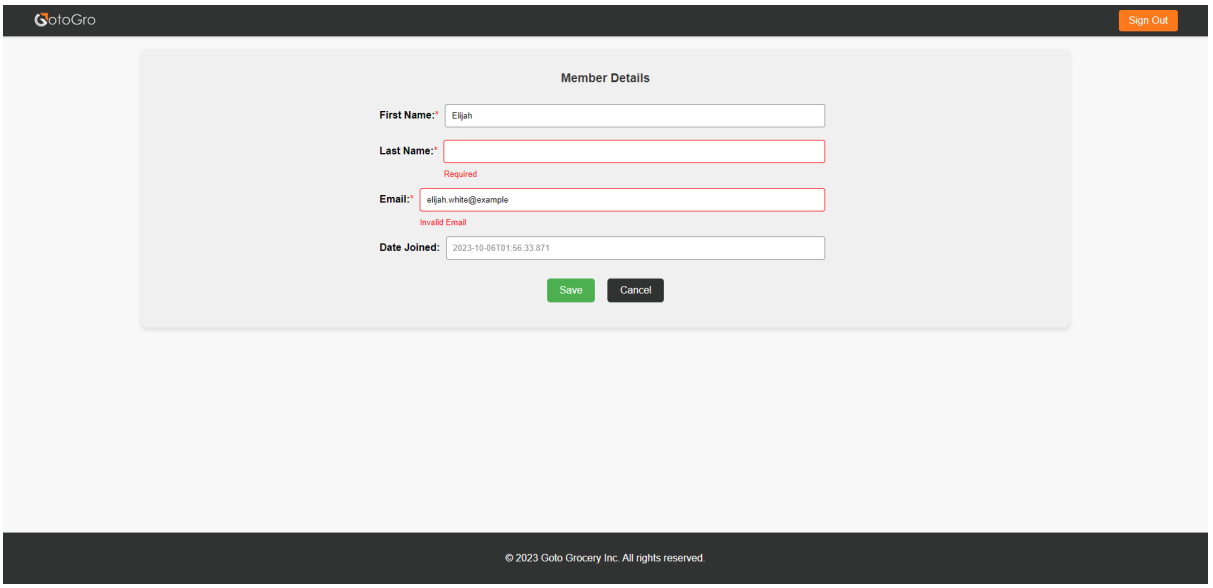


Members Home:

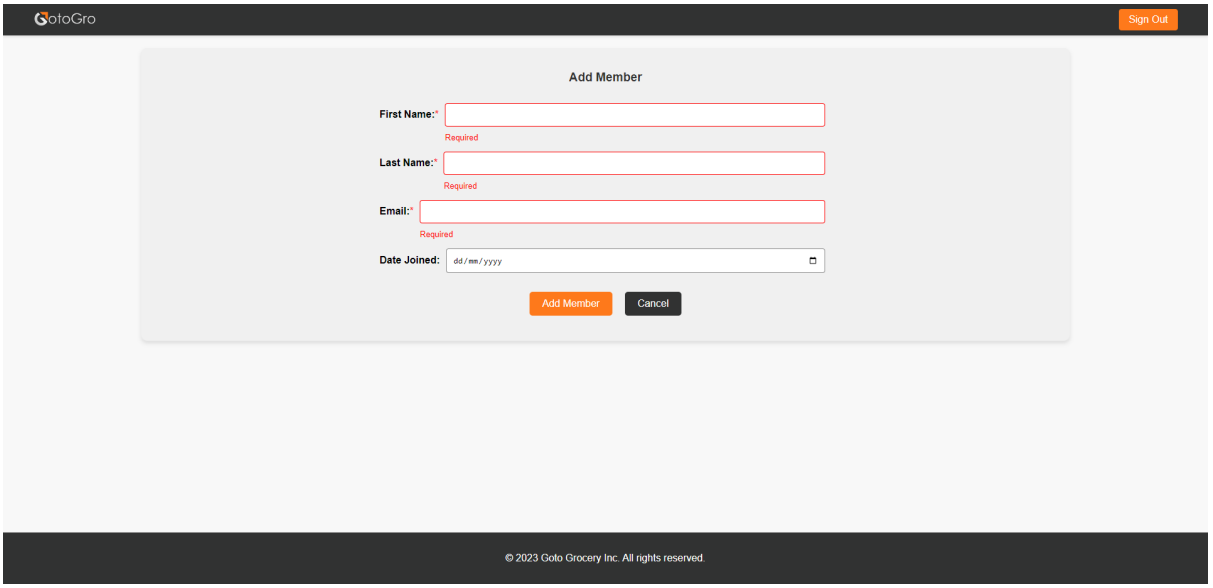
Includes Member Search, Adding a New Member, Editing, Deleting and Retrieving a Member.



Edit Member (Validations):

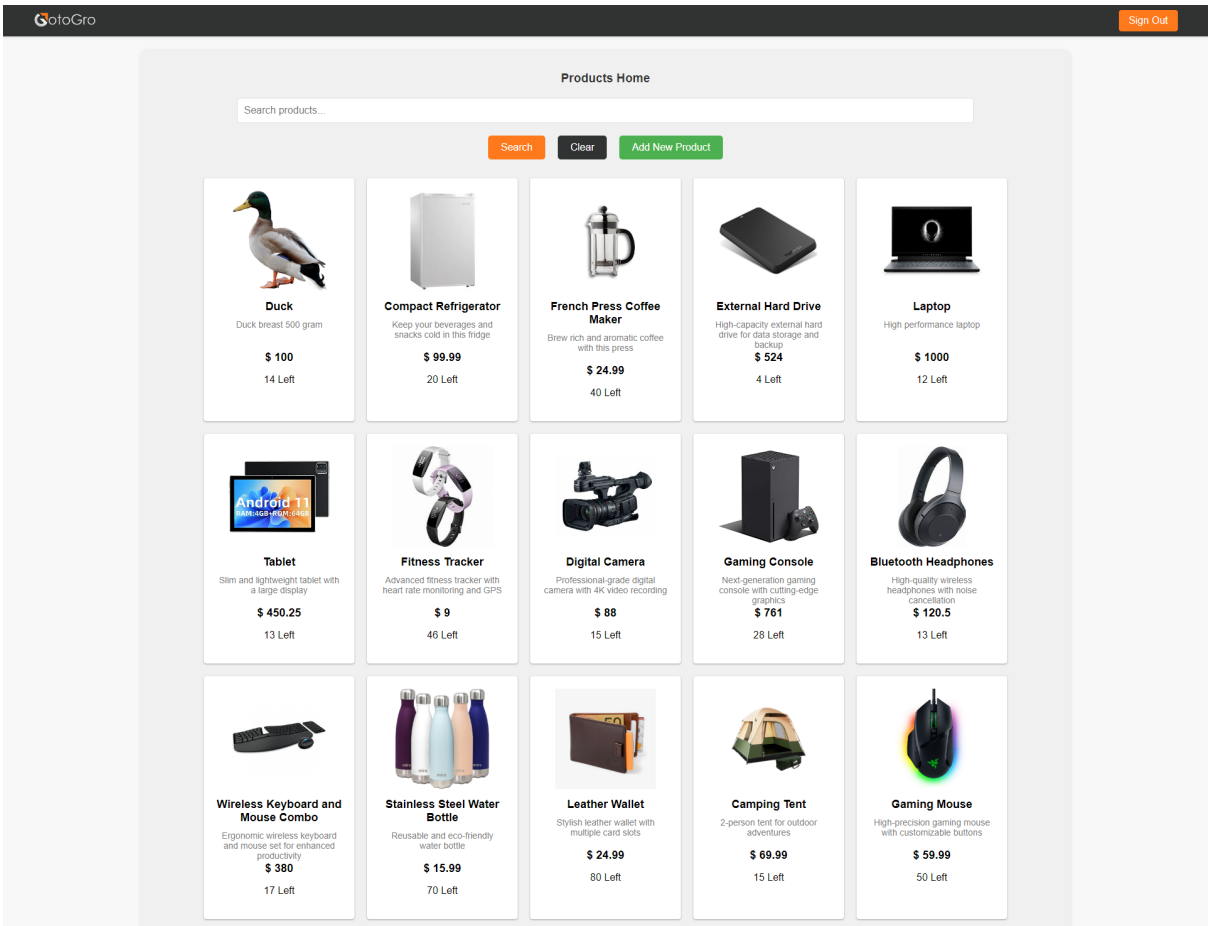


Add Member (Validations):

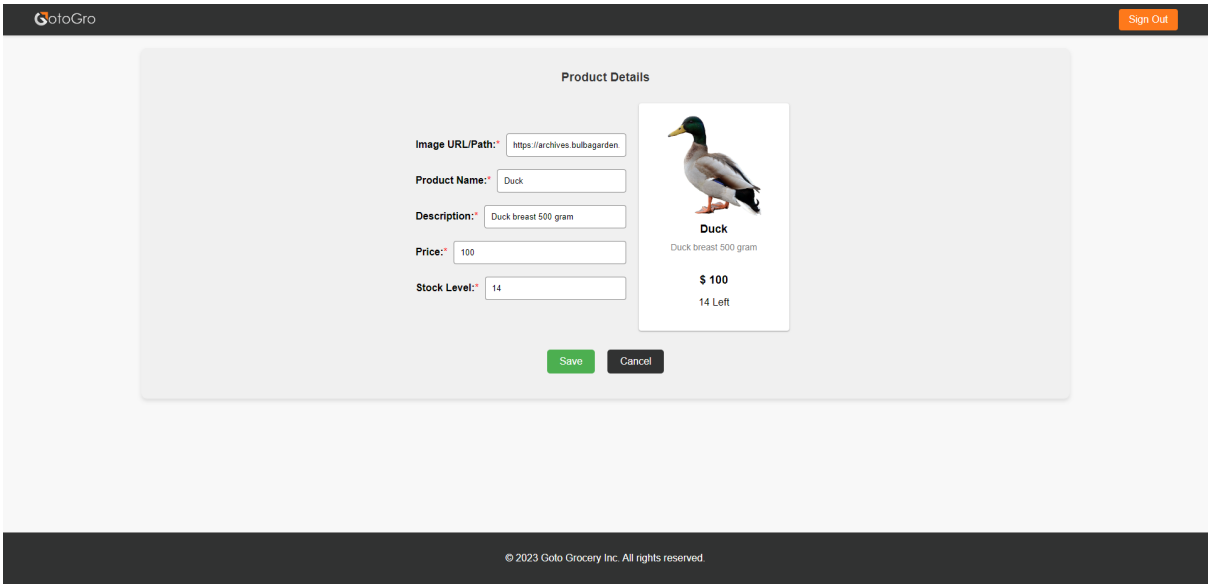


Products Home:

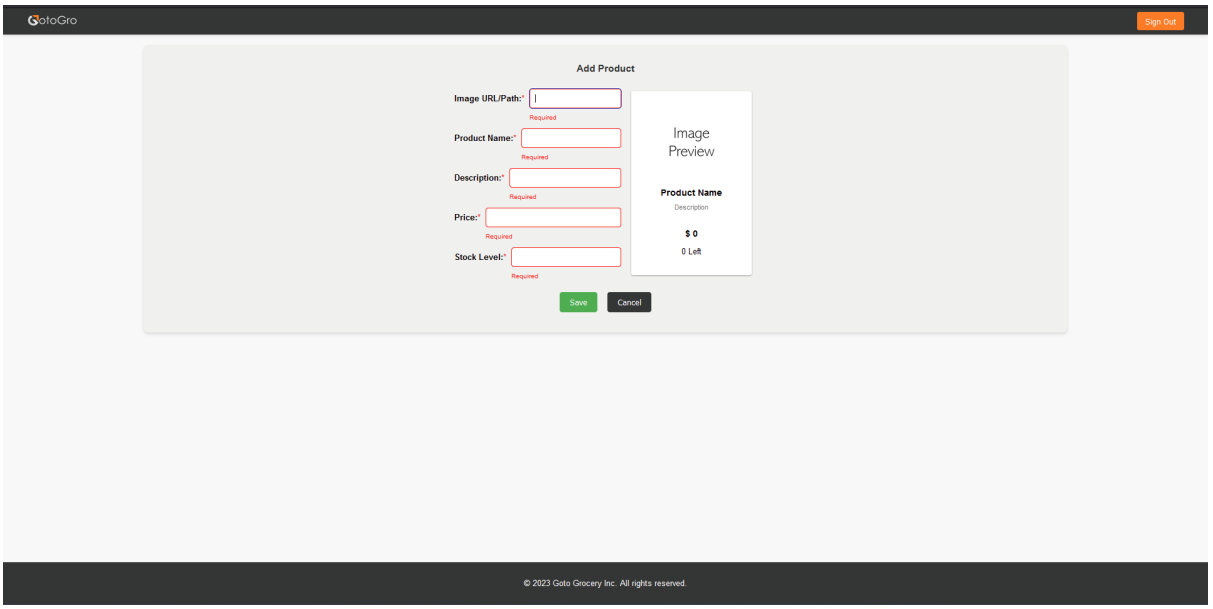
Includes Product Search, Adding a New Product and a Preview of the products. Clicking on a Product takes the user to edit the product details.



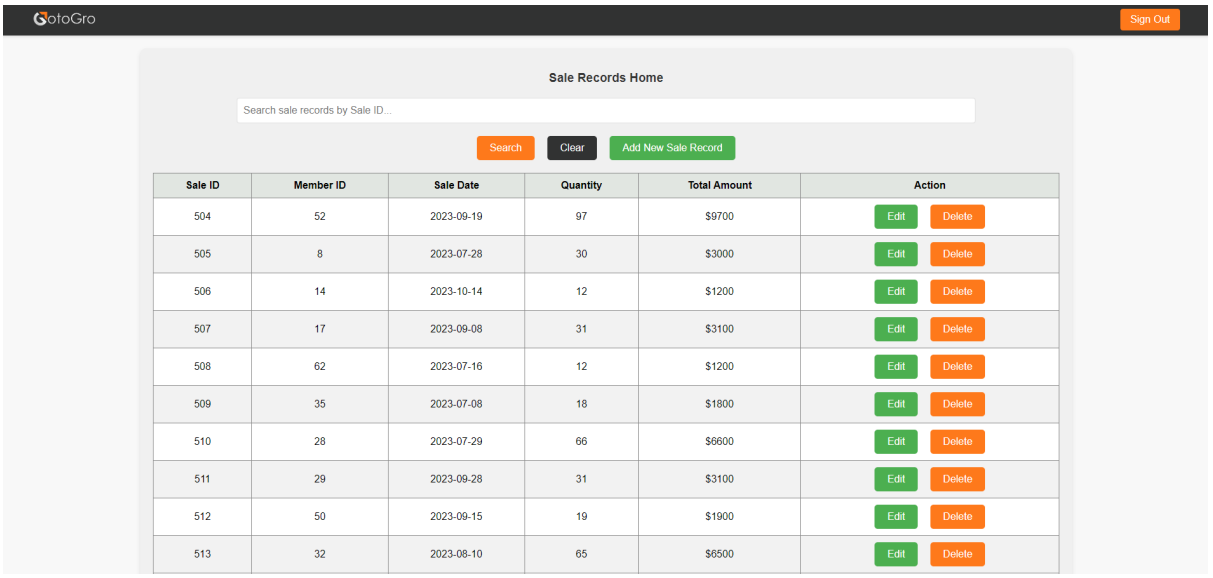
Edit a Product:



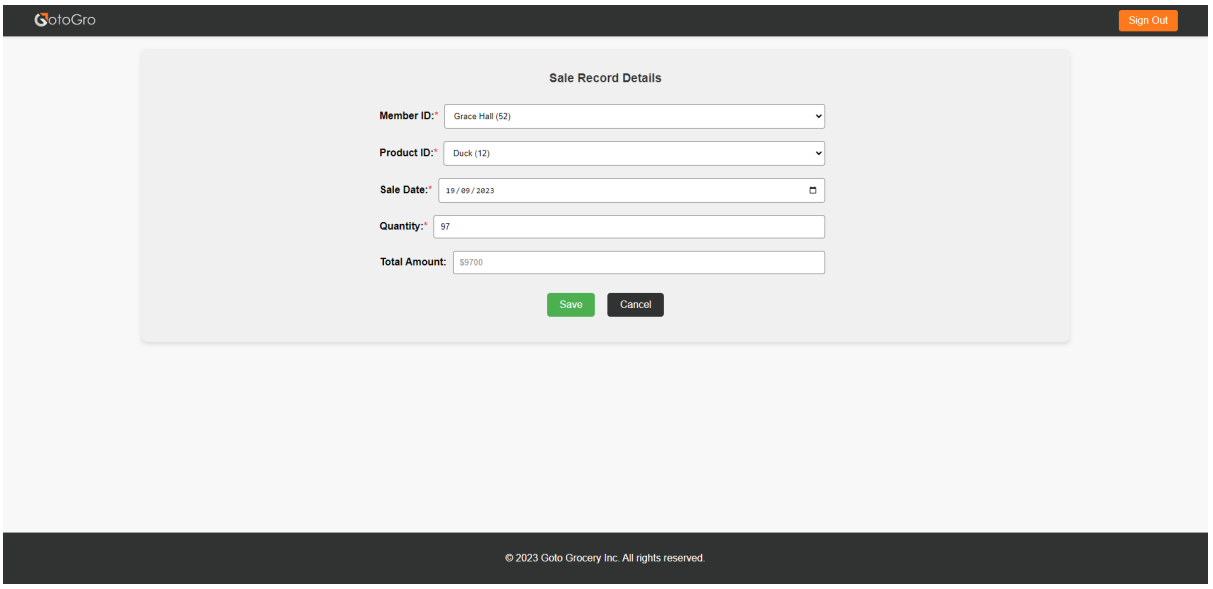
Add Product (Validations):



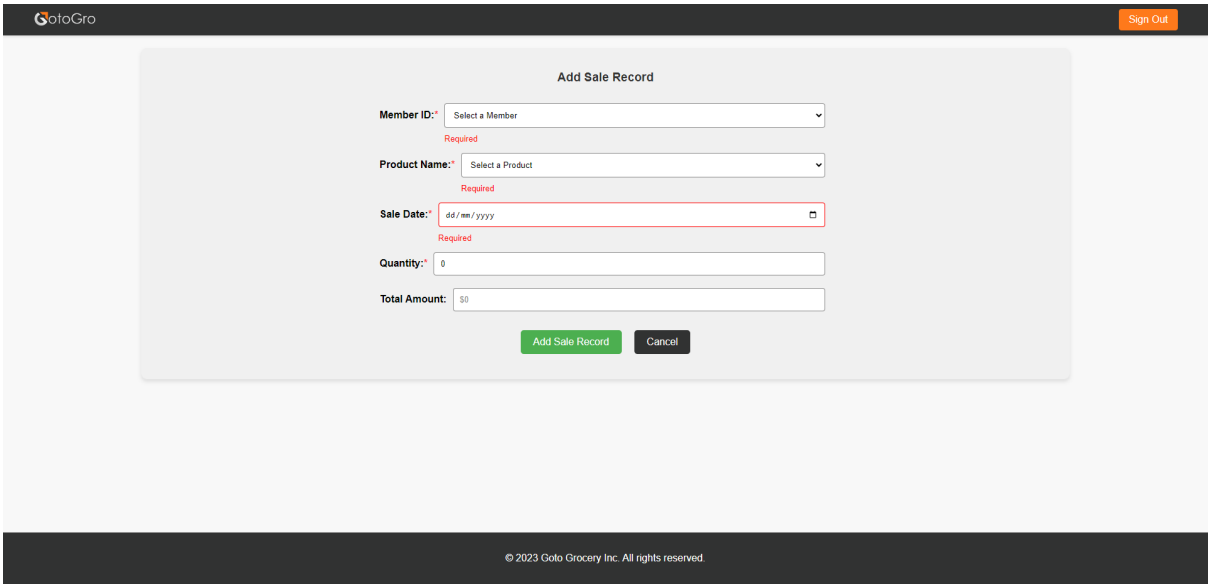
Sale Records Home:



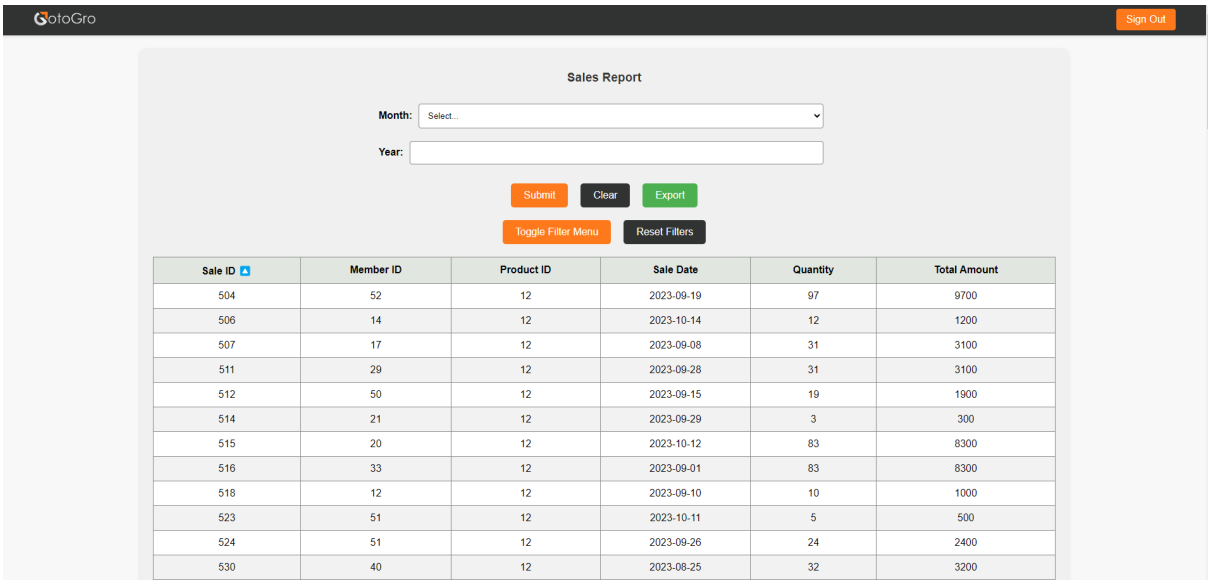
Edit a Sale Record:



Add a Sale Record:

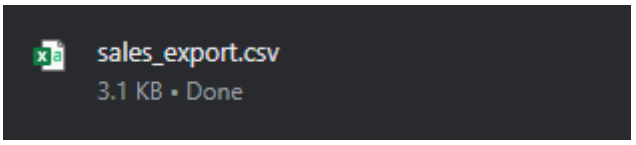


Sales Reports Home:



Sale ID	Member ID	Product ID	Sale Date	Quantity	Total Amount
504	52	12	2023-09-19	97	9700
506	14	12	2023-10-14	12	1200
507	17	12	2023-09-08	31	3100
511	29	12	2023-09-28	31	3100
512	50	12	2023-09-15	19	1900
514	21	12	2023-09-29	3	300
515	20	12	2023-10-12	83	8300
516	33	12	2023-09-01	83	8300
518	12	12	2023-09-10	10	1000
523	51	12	2023-10-11	5	500
524	51	12	2023-09-26	24	2400
530	40	12	2023-08-25	32	3200

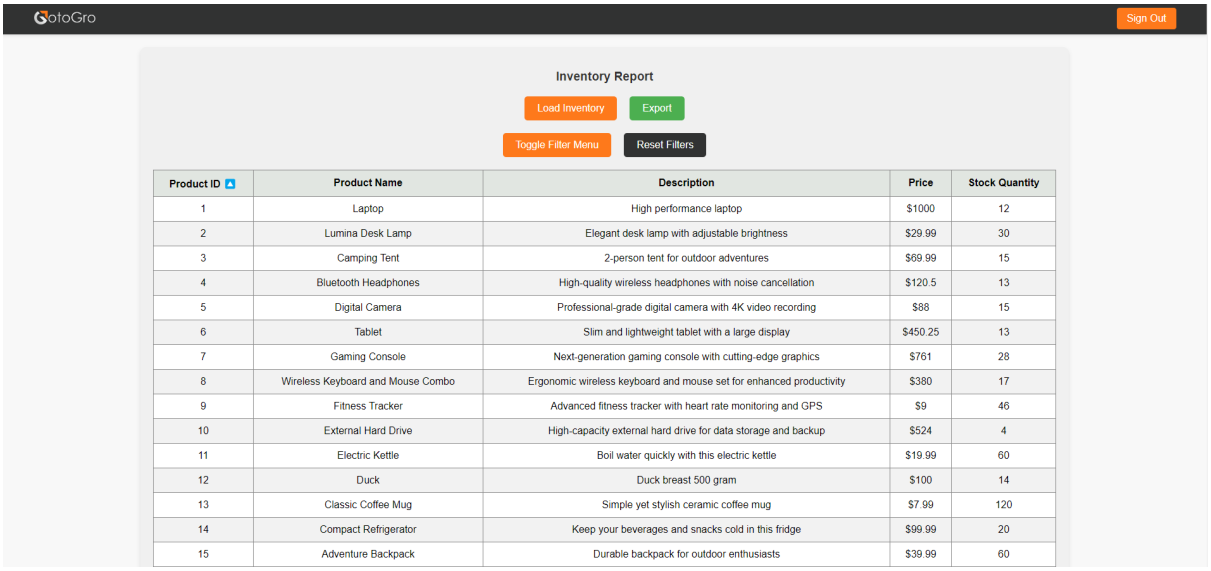
Export sales report:



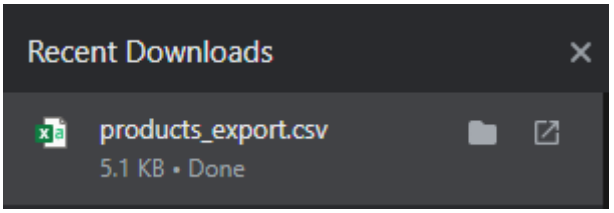
Sample of the Exported CSV Report:

sale_id	member_id	product_id	sale_date	quantity	total_amount
504	52	12	19/09/2023	97	9700
506	14	12	14/10/2023	12	1200
507	17	12	8/09/2023	31	3100
511	29	12	28/09/2023	31	3100
512	50	12	15/09/2023	19	1900
514	21	12	29/09/2023	3	300
515	20	12	12/10/2023	83	8300
516	33	12	1/09/2023	83	8300
518	12	12	10/09/2023	10	1000
523	51	12	11/10/2023	5	500
524	51	12	26/09/2023	24	2400
530	40	12	25/08/2023	32	3200
534	36	12	4/09/2023	64	6400
538	3	12	30/09/2023	74	7400
539	50	12	4/09/2023	92	9200

Inventory reports home:



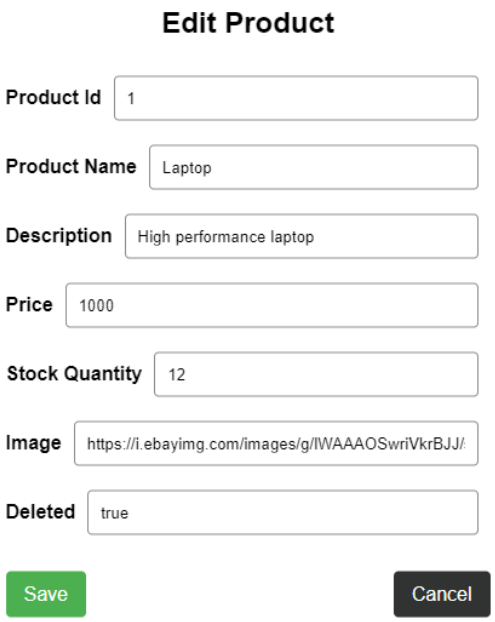
Export Inventory report:



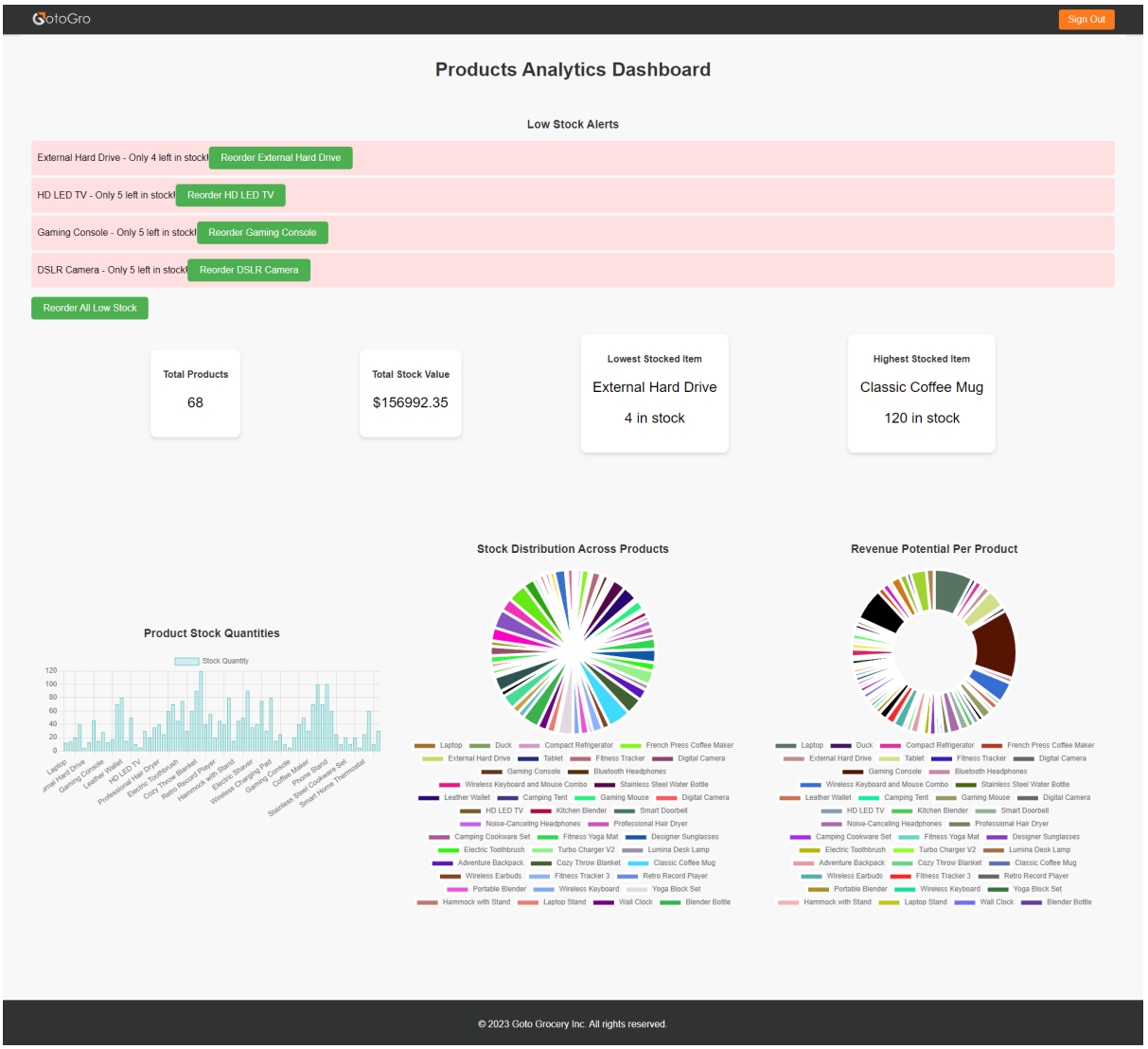
Sample of the Exported CSV Report:

product_id	product_name	description	price	stock_quantity
1	Laptop	High performance laptop	1000	12
2	Lumina Desk Lamp	Elegant desk lamp with adjustable brightness	29.99	30
3	Camping Tent	2-person tent for outdoor adventures	69.99	15
4	Bluetooth Headphones	High-quality wireless headphones with noise cancellation	120.5	13
5	Digital Camera	Professional-grade digital camera with 4K video recording	88	15
6	Tablet	Slim and lightweight tablet with a large display	450.25	13
7	Gaming Console	Next-generation gaming console with cutting-edge graphics	761	28
8	Wireless Keyboard and Mouse Combo	Ergonomic wireless keyboard and mouse set for enhanced productivity	380	17
9	Fitness Tracker	Advanced fitness tracker with heart rate monitoring and GPS	9	46
10	External Hard Drive	High-capacity external hard drive for data storage and backup	524	4
11	Electric Kettle	Boil water quickly with this electric kettle	19.99	60
12	Duck	Duck breast 500 gram	100	14
13	Classic Coffee Mug	Simple yet stylish ceramic coffee mug	7.99	120
14	Compact Refrigerator	Keep your beverages and snacks cold in this fridge	99.99	20
15	Adventure Backpack	Durable backpack for outdoor enthusiasts	39.99	60
16	Electric Toothbrush	Keep your teeth clean with this electric toothbrush	39.99	45
17	Portable Blender	Blend your favorite smoothies on the go	34.99	45
18	Smart Home Thermostat	Control your home temperature with your phone	89.99	25
19	Fitness Tracker 3	Track your health and fitness with this device	49.99	55

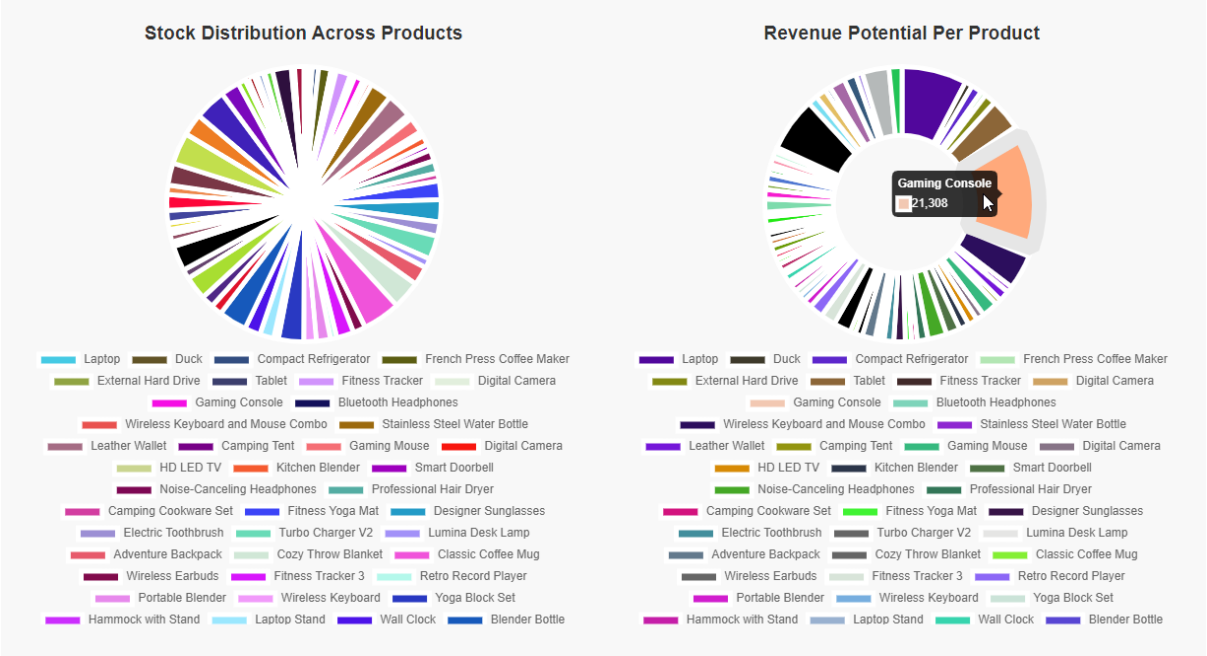
Edit Product (From Inventory Report):



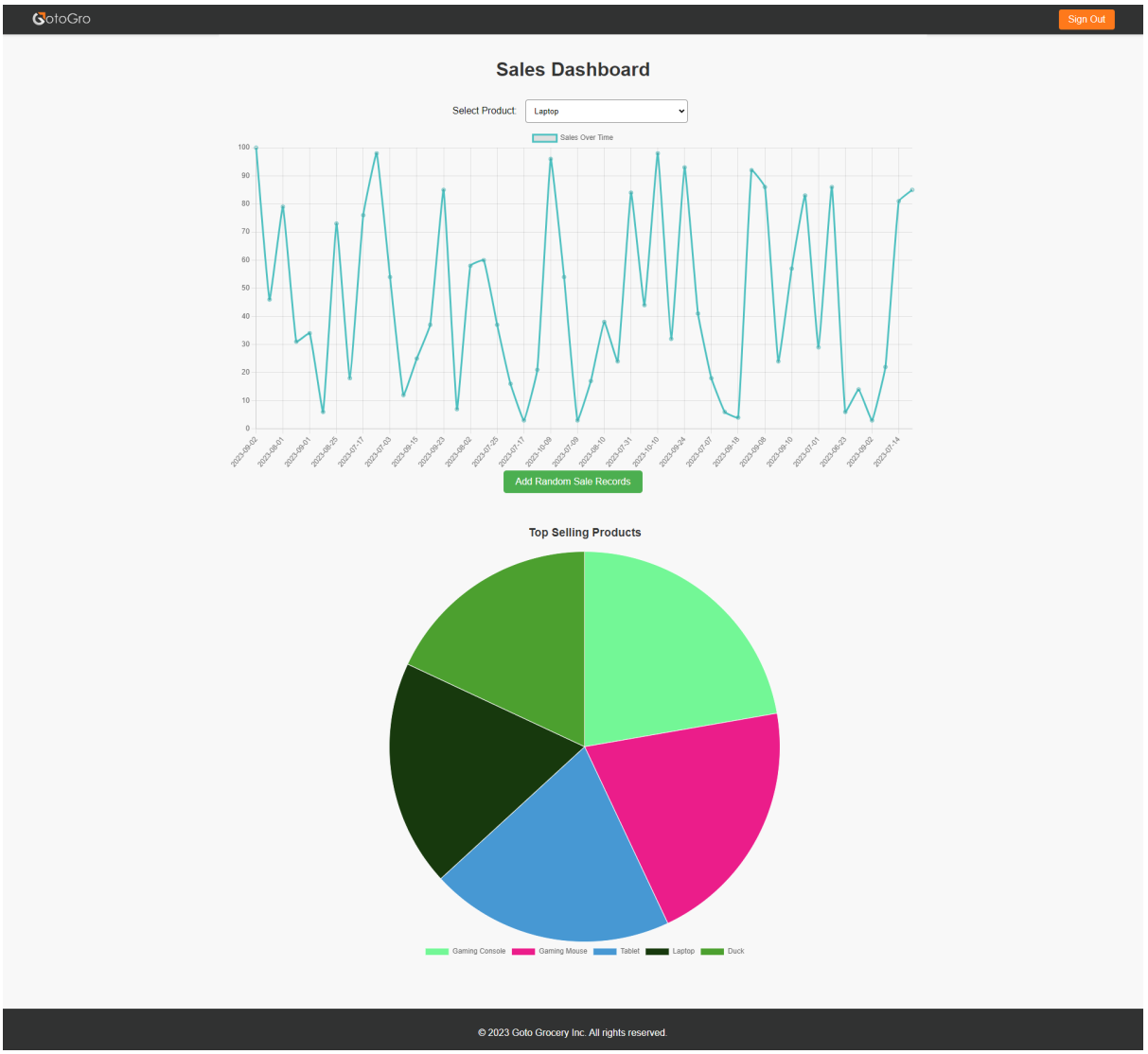
Product Analytics dashboard:



Responsive graphs:



Sales Analytics dashboard (with dynamic graph):



Sprint Review Meeting Minutes

Date:16/10/2023

The meeting commenced with Alex's discussion of the goals that were set out to complete during Sprint 2. The primary goals included making the user interface cleaner, more robust, addressing bug fixes, and enhancing core functionality for Goto Grocery's employees.

The jira board was then shown, and all except one backlog item was shown as being complete.

The backlog item that wasn't addressed as complete was to "Create a prediction algorithm (F16)". This was discussed by Alex as a non-specific scoped item, which leads to a discussion between him and the stakeholder on its broad nature. Marella has stated that this item feels more like an "epic" rather than a backlog item, which the stakeholder has agreen upon.

Suggestion by the stakeholder: Define a clear criteria and a definition of done for this backlog item and to put this backlog item back for Sprint 3.

The burndown charts are then shown and Enzo has reported that it's perfect in comparison to the estimated and actual effort it took to complete the backlog items for Sprint 2. The reason why it's not showing perfectly is due to the burndown chart not showing the transition from "in progress" to "testing" until a task is completed. It was also discussed that some backlog items have had their story points changed.

The team then discussed their estimation techniques, where Marella has discussed that analogy-based estimation techniques were particularly helpful, as the team has mentioned that leveraging the team's experience from Sprint 1 to Sprint 2 was really helpful, as backlog items found in Sprint 1 are similar to backlog items in Sprint 2. Marella then discussed the utilisation of WBS during sprint 1 which helped with the initial estimation of hours to be spent for tasks.

Lachlan then proceeded with testing demos and presented test cases. Lachlan then introduced a fully automated testing framework that worked perfectly when hosted locally, but faced difficulties when the framework was being utilised through the hosted website. The team observed that the execution of all tests and discussed how easy it was to use the automated testing features.

The stakeholder then posed a question in regards to testing: How difficult was it to use and implement?

Lachlan answered that the entire framework was easy to use. There are two different versions of Selenium, and Lachlan utilised the IDE version because it's easier to setup. It's a lot more limited compared to the other version, but with this software project, the IDE is more than sufficient. He also mentioned that the software is easier to test locally compared to on the website as it is faster.

Lachlan the proceeded to show the dashboards created during sprint 2. Alex explains that he's happy with how it shows analytical data for non-technical users who don't want to see raw data.

The stakeholder then asks if the graphs found within the dashboard are based on real data.

The team then showed that it does use real data, as the graphs show test data that was just made during the demonstration.

The team then demonstrated the fuzzy search functionality, which can search through the description or the product name instead.

The product owner then provided future sprint suggestions to be done:

1. Adding a prediction algorithm for future sprints.
2. Customization of dashboards such as custom colours for specific products, or the ability to change how the dashboard looks so that users can have their own custom dashboards so they know what to expect.

The stakeholder then provided future sprint suggestions:

1. Consistency in colour palettes within pages
2. Consistency in button placements within pages
3. Editing functions to be easily accessible within rows, instead of having a button that leads to another page
4. The idea of having a long reports to be separated into pages
5. The idea of having the search filters found at the top of the page instead of the bottom of the page.

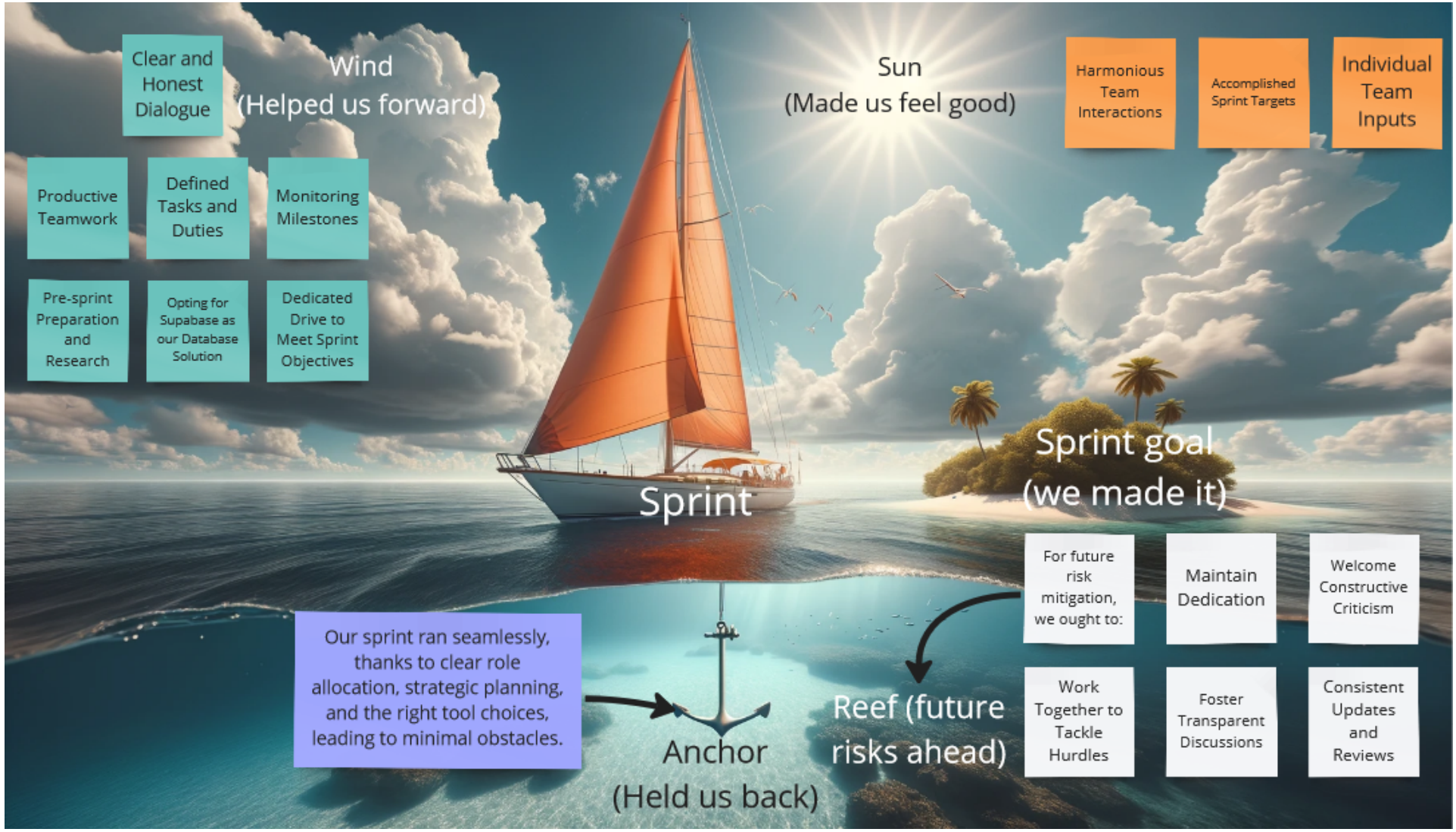
Action Items for the next sprint

1. Definition of done, or a clear criteria for the user prediction algorithm backlog item for the next sprint
2. Address issues related to consistency in design, button placement, and color schemes
3. Explore improvements in paging for long lists and filtering and sorting options.
4. Continue working on the automated testing framework's for future sprints

Sprint Retrospective: Management System (GotoGro-MRM) for Goto Grocery Inc.

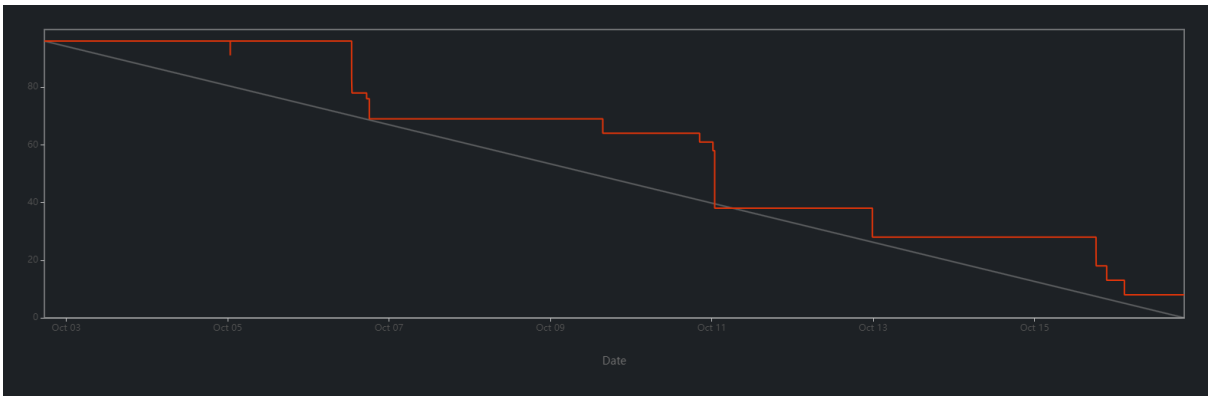
Team: MSP_CL4_T1	
Name	**ID**
Enzo Peperkamp	*102895415*
Nelchael Kenshi Turija	*103057559*
Julian Codespoti	*102997816*
Alex Kyriacou	*103059830*
Marella Morad	*103076428*
Lachlan Martin	*103067448*

Sailboat Retrospective



Meeting Minutes

Your team's velocity – ideal (from your ideal burn-down chart) vs actual (from your final burn-down chart)



Based on the analysis of our burn-down chart for Sprint 2, our initial estimation indicated that we would successfully accomplish all predetermined backlog items within the allocated sprint timeframe. It is worth noting that the burn-down chart, in its current form, may not account for the transition when tasks move from the "In-Progress" phase to the "Testing" Phase. Nevertheless, when we evaluate our ideal velocity against the actual progress achieved, we can assert that our team successfully met the projected ideal velocity.

This estimated vs actual effort underscores our team's capability to maintain a consistent and productive pace throughout Sprint 2. This demonstrates the team's ability to effectively manage and complete the work, even though there may be certain intricacies not fully represented in the chart.

Did your team overestimate your ability? Or did you under-estimate the effort required to complete the tasks?

Our team's estimation accuracy was generally on target for the majority of the backlog items in Sprint 2. It is worth noting that we encountered a setback in completing one specific task, Task F16, which involved the creation of a prediction algorithm. This particular setback can be attributed to the absence of a well-defined scope and a clear definition of "done" for this particular backlog item.

In hindsight, this deviation from our estimation highlights the importance of thoroughly defining and breaking down each backlog item into its most atomic components. By doing so, we can create a more precise understanding of the effort it takes to complete this item, and any potential challenges that may arise during its execution. It is evident that a comprehensive and unambiguous scope, along with a well-defined definition of "done" should be an integral; part of our backlog items to

mitigate the risk of any similar setbacks in the future. This approach will contribute to a smoother and more predictable sprint execution, which ultimately leads to improved accuracy in our estimations.

What can you do in order to get a better understanding of the "complexity" of the tasks required? Or What can you do in order to get better time estimates next time?

In order to gain a deeper understanding of the intricacies associated with the complexity of the tasks at hand, we have identified a couple of approaches. One such approach involves the regular organisation of backlog refinement sessions. These sessions serve as a platform where our team can engage in comprehensive discussions about the finer nuances of the upcoming tasks. Furthermore, they enable us to deconstruct larger tasks into smaller, more manageable sub-tasks. This breakdown provides us with a clearer view of the work required and the potential challenges the team might face.

Another valuable method that contributes to our understanding is the review of historical data. We have employed the estimation by analogy technique with a significant portion of our backlog items. This practice has proven to be most effective, and we plan to continue leveraging it in the future. By drawing insights from historical data and the outcomes of past endeavours, we can refine our estimation and bolster our capability to gauge the complexity of future backlog items with greater precision and confidence. These combined efforts empower us to make more reliable estimations, thereby enhancing our project management process.

Other questions, if any? (Please specify)

Your team's process

What is working? Why?

Currently, within our team, two key elements are working:

Firstly, our regular stand-up meetings have proven to be an effective process. These brief, daily gatherings have noticeably enhanced our internal communication and collaboration. The ability to address emerging issues and concerns promptly during these meetings has been pivotal in keeping our project on track. It ensures that everyone is aligned with the team's progress and any potential roadblocks can be quickly identified and resolved. As a result, our overall workflow and efficiency have been notably improved.

Secondly, the utilisation of Jira has emerged as a robust resource for our team. This tool provides us with invaluable visibility to the status of individual tasks and the overall project. It streamlines the tracking of work progress and fosters a collaborative environment among team members. The digital platform facilitates the

efficient sharing of information, documentation, and updates, further contributing to the team's overall effectiveness and productivity.

These two elements are currently fundamental in ensuring that our team operates well, stays well-informed, and efficiently manages the various aspects of this software management project.

What is not working? Why not? Any suggestions to improve the situation if this occurs in the future?

One challenge that our team encountered, over which we have limited control, relates to external disruptions. These disruptions can manifest in various forms, such as unexpected work scheduling conflicts or personal emergencies among team members, and they have the potential to disrupt planned work.

To address this challenge, we can implement a strategic approach during our sprint planning sessions. Specifically, we can allocate a designated buffer time within our sprint schedule. This buffer serves as a contingency to account for unforeseen external disruptions that may arise during the course of the sprint. By proactively setting aside buffer time, we can better accommodate unexpected events without severely impacting our sprint goals and timeline.

This prudent approach helps us maintain flexibility and adaptability in the face of unforeseen external factors, which enhances our ability to manage disruptions and maintain sprint efficiency. It is a measure that allows us to respond to the unexpected with minimal disruption to our planned work.

Software Design: Management System (GotoGro-MRM) for Goto Grocery Inc.

Team: MSP_CL4_T1	
Name	**ID**
Enzo Peperkamp	*102895415*
Nelchael Kenshi Turija	*103057559*
Julian Codespoti	*102997816*
Alex Kyriacou	*103059830*
Marella Morad	*103076428*
Lachlan Martin	*103067448*

Introduction

As we conclude Sprint 2, this document serves as a continuation and evolution of our software design journey, which began with the "14P Software Design" report from Sprint 1. Over the course of this sprint, our team has made significant strides in advancing the software's design, building upon the foundational principles and structures established in Sprint 1. Recognizing the importance of consistency and clarity, we have maintained several conventions from our initial sprint, while also introducing enhancements in various areas to accommodate the new features and functionalities.

Throughout this report, references to our Sprint 1 decisions and designs will be made to highlight the progression and rationale behind the changes and improvements implemented in Sprint 2. Our commitment to creating a robust and scalable software system remains unwavering, and this document aims to provide a comprehensive overview of our design choices, their justifications, and their alignment with established software design principles.

Design of the Software Components

Updated Software Diagram

In our journey from Sprint 1 to Sprint 2, the software architecture has seen a natural evolution. While the foundational MVC structure remains consistent, several new components and interactions have been integrated to accommodate the advanced features introduced in this sprint.

Components and Their Roles

The updated software diagram introduces new components that play specific roles within our system:

Model Layer:

The foundational setup of Supabase/GCP from Sprint 1 has been retained in Sprint 2. This decision was grounded in the realization that our existing database and cloud infrastructure were already robust and versatile, seamlessly accommodating the influx of new items introduced this sprint.

Controller/View Layer:

The enhancements in this layer are driven by a focus on user experience (UX). New visual components and user interfaces have been seamlessly integrated, providing users with a richer, more intuitive experience. The website's overall structure has been reimagined to improve navigability and coherence. A testament to these changes is the users' newfound ability to directly access the following pages from the home page:

- Dashboards for Products and Sales
- Reports on Inventory and Sales

- Dedicated home sections for Products, Sales, and Members

CI/CD pipeline:

With the recent addition of a team member specialising in Quality Assurance (QA), our CI/CD pipeline has undergone significant refinements. While our cornerstone tools – GitHub for version control and Jira for sprint task management – remain essential to our development cycle, Sprint 2 saw an evolution in our approach to synchronising Jira with GitHub. The initial goal of this synchronisation was to automate several transitions, such as moving a card to "done" upon merging a pull request or marking an issue as "in progress" upon branch creation.

However, our experiences during this sprint highlighted that while such synchronisation can automate certain transitions, it does not fully capture the intricate nuances of our development process. For example, the mere act of creating a branch or a pull request doesn't necessarily reflect the comprehensive status of a Jira card. Tasks often involve phases like research, analysis, and discussions before coding even begins. Consequently, instead of maintaining strict synchronisation, we've chosen a more adaptable approach. We primarily use the integration to link pull requests directly to their respective Jira cards, ensuring a truer representation of a card's status, which in turn facilitates improved project management and clarity.

A notable enhancement in our CI/CD pipeline is the "Testing/Quality Assurance" segment. Utilising the Selenium WebDriver IDE, comprehensive usability tests have become standard. These tests mimic user behaviours on our platform, ensuring the system offers an intuitive and error-free user experience. The automation capabilities of this tool allow for consistent testing across various user paths, thereby validating functionality in multiple scenarios.

Furthermore, the Selenium WebDriver IDE is also adept at functional testing. We've crafted a series of test cases that assess specific functionalities, ensuring they operate as intended. These evaluations include monitoring data transactions, user input processes, and interactions between system modules.

The integration of this testing tool into our CI/CD pipeline has established a continuous feedback loop. Any code modification by developers triggers automated tests, reinforcing the software's ongoing readiness.

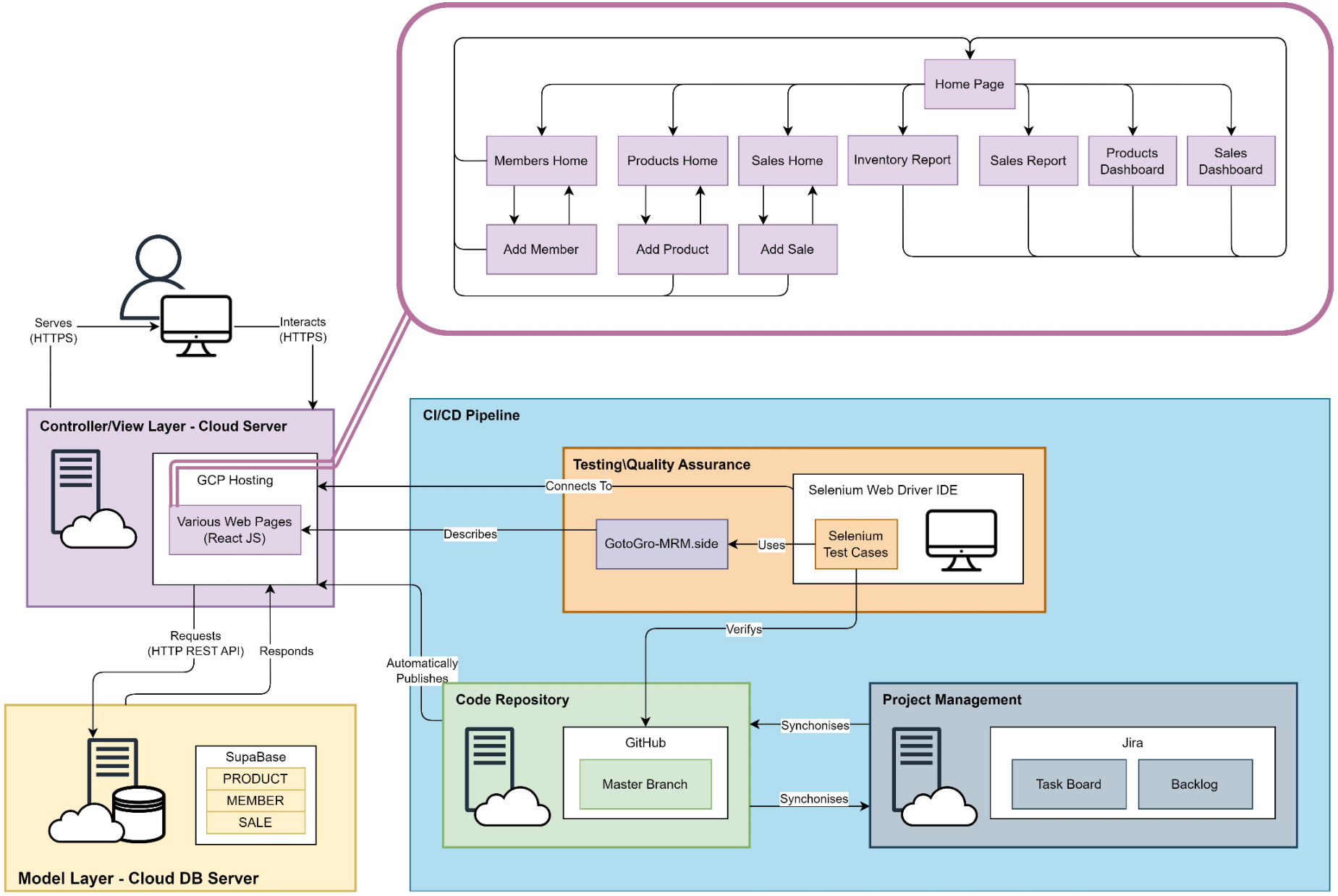
Interactions and Flow

The interactions in Sprint 2 showcase a system with broader capabilities:

- **User Interactions**: Users, depicted by monitor and person icons, connect with the system via HTTPS to access a range of web pages hosted on GCP Hosting.
- **UI Navigation**: From the "Home Page", users can explore the Members, Products, or Sales sections. Each of these has further detailed navigation options, such as adding a member or viewing the sales dashboard.
- **Server & Data Interactions**: Hosted web pages, crafted with React JS, reside on the cloud server. Any user data input or retrieval involves the cloud server

communicating with the Cloud DB Server using HTTP REST API, ensuring efficient data storage and retrieval.

- **Data Management**: Data entities like "PRODUCT", "MEMBER", and "SALE" are stored on the Cloud DB Server, which interacts with the Controller/View Layer based on user actions.
- **CI/CD Workflow**:
    - **Code Management**: Developers commit code to GitHub's "Master Branch", set up for automated updates.
    - **Task Synchronization**: GitHub syncs with Jira, a project management tool, which organizes tasks on a board and backlog, aligning code changes with tasks.
    - **Quality Assurance**: Upon code push, it's linked to the "Selenium WebDriver IDE" for testing. This tool uses test cases to ensure that all new additions function correctly before deployment.

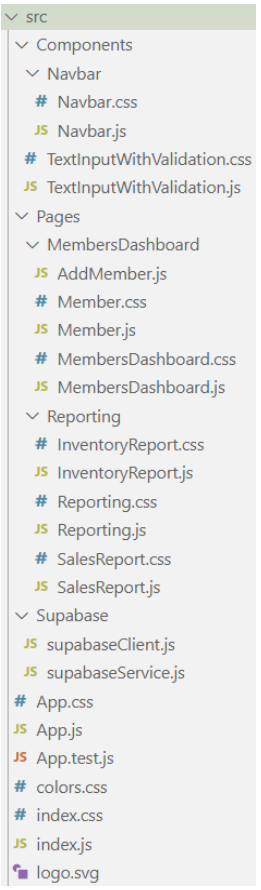

Home Page
Members Home
Products Home
Sales Home
Inventory Report
Sales Report
Products Dashboard
Sales Dashboard
Add Member
Add Product
Add Sale
Serves (HTTPS)
Interacts (HTTPS)
Controller/View Layer - Cloud Server
GCP Hosting
Various Web Pages (React JS)
CI/CD Pipeline
Testing\Quality Assurance
Connects To
Selenium Web Driver IDE
Describes
GotoGro-MRM.side
Uses
Selenium Test Cases
Verifys
Requests (HTTP REST API)
Responds
Automatically Publishes
Code Repository
GitHub
Master Branch
Synchonises
Project Management
Jira
Task Board
Backlog
Synchonises
SupaBase
PRODUCT
MEMBER
SALE
Model Layer - Cloud DB Server

Directory and Naming conventions

Building upon the conventions established in Sprint 1, our directory structure and naming conventions have remained consistent. The decision to maintain these conventions was influenced by our team's collective industry experience, which vouched for the clarity and efficiency of the chosen structure and naming protocols.

Here's a brief reminder of the conventions we've established:

- **Naming Conventions**:
  - For code readability and maintainability, consistent and descriptive names are used for variables, functions, and components.
  - React components are named in PascalCase, reflecting their function or content. For example, "Navbar" for a navigation bar.
  - Functions and variables use camelCase. Descriptive function names like `fetchSalesByDateRange` clearly indicate their purpose.
  - Function parameters are descriptive, e.g., `updatedMember` for a member update.
  - Asynchronous functions, particularly those interacting with the backend, are prefixed with their operations, like `signIn` or `searchMembersByName`.
- **Directory Structure**:
  - Components are systematically organized, with each stored in its own directory like `./Components/Navbar/Navbar`.
  - Pages or views are kept in a "Pages" directory, following the same naming consistency as components.
- **File Naming**:
  - Filenames, including component files, follow PascalCase, e.g., "MembersDashboard".
  - Styles associated with specific components or pages use the component name with a `.css` extension, like `App.css`.
- **Comments and Documentation**:
  - Functions include JSDoc-style comments detailing their purpose, parameters, and return values.
  - Inline comments clarify specific lines of code or operations that might be non-intuitive.
- **Database Operations**:
  - Database tables are referenced in PascalCase, like "SaleRecords" or "Members".
  - Queries are crafted to clearly indicate their purpose.



References to our Sprint 1 directory and naming decisions can be found throughout our codebase, emphasizing our dedication to a systematic approach. Our experienced team recognizes the value of clear and intuitive file structures, component naming, and the overall organization, ensuring that as our project grows, any developer—new or returning—can navigate and understand the system with ease.

Jira Workflow Enhancements

Our experiences in Sprint 1 prompted us to refine our task management approach for Sprint 2, leading to several improvements in our Jira workflow:

1. At the sprint's commencement, backlog items are transitioned into the "To do" column.
2. Developers, upon selecting a task, move it to the “In progress” column. They then create a branch from the main one, naming it after the User Story ID/Code.
3. Post-development and once changes are merged into the production server, the task is relocated to the “Ready to be tested” column.
4. Testers take on items from here and transition them to the “Testing” column.
5. Post-testing, they shift the items to the “To be confirmed” column, appending all collected test evidence.
6. The Product Owner or Scrum Master reviews the evidence. If satisfied, they transition the item to the "Done" column, signifying its successful completion.

This enhanced Jira workflow ensures clarity, transparency, and accountability at every stage, fostering a seamless progression from task inception to completion.

Design Justification using Design Principles

Object-Oriented Principles

In continuation of our commitment to object-oriented design from sprint 1, we have ensured that:

- Each React component still represents a cohesive unit of functionality. For instance:

```
import React from 'react';
import {Link} from 'react-router-dom';
import './Reporting.css';

function Reporting() {
    return (
        <div className='reporting-content'>
            ...
        </div>
    );
}
```

- We've maintained the encapsulation of behaviour and rendering logic in components like `ProductsDashboard` and `InventoryReport`:

```
const ProductsDashboard = () => {
    const [products, setProducts] = useState([]);
    ...
    return (
        ...
    );
}
```

```
function InventoryReport() {
    const [products, setProducts] = useState([]);
    ...
    return (
        ...
    );
}
```

- State variables and hooks continue to be the backbone of data flow and component interactions. This consistency ensures predictability and ease of understanding:

```
const [products, setProducts] = useState([]);
useEffect(() => {
    getProducts();
}, []);
```

Cohesion and Coupling

In our ongoing commitment to strong cohesion and weak coupling, the components discussed below are merely illustrative examples, chosen at random from the myriad of cohesive and loosely coupled components in our codebase. They demonstrate the consistent design philosophy we've adhered to throughout our development:

- **ProductsDashboard Component:**
  - **Cohesion:** This component is a testament to our commitment to strong cohesion, with a clear-cut responsibility of managing and displaying product information.
  - **Coupling:** Consistent with our previous components, its interactions with external services and libraries, like supabaseService and chart.js, are cleanly defined and minimal.
- **InventoryReport Component:**
  - **Cohesion:** It remains laser-focused on its core responsibility of managing and displaying inventory data.
  - **Coupling:** Its interactions with supabaseService, papaparse, fuse.js, and react-toastify are clear cut and minimal, ensuring weak coupling.

By consistently applying these principles across all components, not just the ones highlighted, we ensure a modular, understandable, and adaptable system.

Design Patterns and Architectural Styles

Building upon our foundational design principles, our codebase embraces various design patterns and architectural styles. These patterns and styles, coupled with our commitment to Object-Oriented Principles, fortify our system's maintainability, modularity, and extensibility.

1. **Module Pattern**
   We've encapsulated functionality within ES6 modules, ensuring each module serves a singular, cohesive purpose. For instance, the SalesDashboard, ProductsDashboard, and InventoryReport components are each contained within their respective modules.

```
import React, { useState, useEffect } from 'react';
...

const SalesDashboard = () => {
    ...
    return (
        ...
    );
}

export default SalesDashboard;
```

2. **Component-based Architecture**
   Embracing the power of React, our system thrives on a component-based architecture. Each component, like `ProductsDashboard`, represents a unit of functionality:

```
const ProductsDashboard = () => {
    const [products, setProducts] = useState([]);
    ...
    return (
        ...
    );
}
```

3. **Hooks**
   React hooks, especially `useState` and `useEffect`, have become the backbone of our components, driving state management and side effects.

```
const [products, setProducts] = useState([]);
useEffect(() => {
    fetchProducts().then(data => setProducts(data));
}, []);
```

4. **Service Layer Pattern**
   Our components remain clean and focused by offloading data-fetching logic to a service layer, as demonstrated with the `fetchProducts` function from the `supabaseService` module.

```
const fetchedProducts = await fetchProducts();
setProducts(fetchedProducts);
```

5. **Observer Pattern**
   Implicit in our usage of hooks, the observer pattern enables our components to respond to state and prop changes.

```
useEffect(() => {
    if (selectedProductId) {
        getSalesData();
    }
}, [selectedProductId]);
```

6. **Factory Pattern**
   Our system dynamically constructs data objects based on input data, much like how chart data is built from the products array.

```
const dataBar = {
    labels: products.map(p => p.product_name),
    ...
};
```

7. **Strategy Pattern**
   We provide flexibility in data visualization by adopting different charting strategies, such as Bar, Doughnut, and Pie charts.

```
<Bar data={dataBar} />
<Pie data={dataPie} />
```

8. **Facade Pattern**
   We've employed libraries like `papaparse` and `fuse.js` to offer simplified interfaces, masking more complex underlying operations.

```
Papa.parse(data, {...});
const fuse = new Fuse(products, options);
```

9. **Decorator Pattern**
   Enhancing user feedback, we've integrated toast notifications without altering the core functionality of our components.

```
toast.success('Product updated successfully!');
```

By integrating these patterns and styles, we've ensured a robust, scalable, and highly maintainable system. Each component, while part of a larger ensemble, is designed to be independently updated and changed, championing a modular approach.

Testing rationale

Tool Selection - Why Selenium WebDriver IDE?

Our choice of Selenium WebDriver IDE is rooted in its versatility and broad industry recognition. Its capabilities extend beyond just automating web application tests—it ensures that our software consistently meets expectations, and any potential regressions are pinpointed in real-time.

Immediate Feedback Mechanism

The power of Selenium isn't merely in automation but in the immediacy of its feedback. Each time developers introduce modifications to the code, automated tests spring into action, providing instantaneous insights. This real-time feedback mechanism ensures that our codebase remains perpetually deployable, bolstering our confidence in each release.

Fostering Collaborative Excellence

One of the standout benefits of our testing approach is the enhanced synergy between developers and testers. By clearly defining the testing phase, testers are unburdened from routine validations, directing their focus on more nuanced, scenario-specific evaluations. This not only ensures thorough testing coverage but also cultivates an environment where quality assurance is a shared responsibility.

Transparent Documentation & Comprehensive Evidence Collection

Beyond just test execution, the realm of quality assurance emphasizes the importance of transparency and traceability. Selenium WebDriver IDE champions this cause by meticulously logging test outcomes. This not only aids in immediate debugging but also serves as a repository of evidence, ensuring accountability and facilitating future audits.

Conclusion

As we wrap up Sprint 2, it is an opportune moment to reflect on our progress, understand our growth, and chart our future course. The advancements made during this sprint are a testament to our team's relentless dedication, innovative spirit, and unwavering commitment to excellence.

Comparing our journey from Sprint 1, it is evident that our software's design and functionality have matured considerably. While the foundational principles set in Sprint 1 provided the bedrock for our development, Sprint 2 saw the seamless integration of advanced features, enhanced user interfaces, and a fortified CI/CD pipeline. The adoption of Selenium WebDriver IDE has not only automated our testing processes but also fortified the quality of our deliverables, ensuring that every feature aligns with our high standards. The consistency in our directory and naming conventions, coupled with the refined Jira workflow, underscores our dedication to systematic and transparent development processes.

However, it's essential to recognize that while we have achieved significant milestones, the world of software is ever-evolving. As we look ahead, our focus remains on scaling our software system further, refining our user experiences, and continuously adapting to the dynamic needs of our stakeholders. We foresee a hypothetical Sprint 3 bringing forth newer challenges, and we remain poised to tackle them head-on. Our journey from Sprint 1 to Sprint 2 has been transformative, and we are confident that with each subsequent sprint, we will continue to break boundaries, set higher benchmarks, and redefine excellence.

SWE20001 Managing Software Projects
Self and Peer Review Assessment Form [Sprint #1/2]

Date: _22/10/2023_____

Your Team: _MCP-CL4-T1_____ Your Name: __Marella Morad____

Use the instructions (see below) to fill in scores for each category A to J.

Team Members (Name)	A	B	C	D	E	F	G	H	I	J	Total
Self	5	5	5	5	5	5	4	5	5	5	49
Enzo Peperkamp	5	4	5	5	5	5	4	5	5	5	48
Nelchael Kenshi Turija	5	4	5	5	5	5	4	5	5	4	47
Julian Codespoti	5	5	5	5	5	5	4	5	5	4	48
Alexander Kyriacou	5	5	5	5	5	5	4	5	5	4	48
Lachlan Martin	5	5	5	4	5	5	4	5	5	4	47

Your Reasoning / Justification (You must write a paragraph about each team member below. Incomplete reviews will not be accepted.)

Name, student number	**Comments (complete sentences required)**
Self	I consistently strive for excellence in my work, setting and exceeding high standards. My contributions in terms of quantity, quality, and efficiency are well-established, and I'm known for my positive attitude and enthusiasm. I am a dependable team member, always willing to go the extra mile. However, I recognize the need for improvement in terms of group meeting attendance, which is an area I am committed to working on.
Enzo Peperkamp	Enzo consistently exceeds expectations and is an inspiring team member. His unwavering work ethic, exceptional communication skills, and impressive efficiency make him a standout contributor. He consistently sets a positive tone within the team, setting a high bar for excellence and motivating others to perform at their best.
Nelchael Kenshi Turija	Nelchael is a consistent and reliable team member who consistently delivers high-quality work and communicates effectively. His presence adds stability to the team, and his efficiency ensures tasks are completed in a timely manner. He can always be depended upon to meet the team's needs and maintain a positive working atmosphere.
Julian Codespoti	Julian is a reliable and efficient team member who consistently sets and achieves high goals, making significant contributions to the team. His strong initiative and effective communication skills facilitate seamless collaboration. His personal relations and positive attitude further enhance team cohesion and stability.
Alexander Kyriacou	Alexander excels in various areas, including the quality of his work and his ability to build positive relationships. His consistently positive attitude and dedicated effort significantly contribute to the team's overall success. His presence adds a sense of enthusiasm and drive that benefits the entire team.

Lachlan Martin	Lachlan is a dedicated and dependable team member who consistently delivers high-quality work. His contributions to the team are marked by a strong work ethic and unwavering commitment. Additionally, Lachlan brings a positive dynamic to the team, fostering a collaborative and supportive environment that enhances overall productivity.

Self and Peer Assessment Form

The main purpose of this form (on Sheet 2) is for all Group members, including yourself, to reflect on its interactions, but it may be also be helpful in resolving disputes over the relative contributions of Group members.

Using the spreadsheet **Self and Peer Assessment Form**

1. List the members of your Project Group
2. Enter a score between 0 and 5, for categories A to J for all members of the group including yourself.
3. You will be asked to take a newly completed form to Group meetings with your supervisor: your supervisor will tell you which meetings.

S. Winger-Haunty (1990). University of Wisconsin-Stout Modified by Pheroza Daruwalla and Ian Knowd, 1994

A. Quantity of Work

0 - Did nothing - uninvolved
1 - Does enough to get by
2 - Occasionally exceeds standards- needs improvement
3- Satisfactory. Does more than what is required
4 - Very industrious. High Quality. Consistent
5. Always exceeds productivity standards. Outstanding

B. Quality of Work

0 - Careless. Makes frequent mistakes. Assignment suffers.
1 - Mistakes frequent enough to question results.
2 - Work is basically correct.
3 - Accurate when and where it really counts. Satisfactory.
4 - Almost always accurate in all areas of contribution
5 - Outstanding. Perfect quality. No mistakes.

C. Communication Skills

0 - Blunt, discourteous, does not listen, antagonistic, distant, aloof.
1 - Sometimes tactless. Approachable and friendly once known by others.
2 - Agreeable and pleasant. Warm, friendly , sociable, listens.
3 - Always very polite and willing to help. Very sociable and outgoing. Listens and understands.
4 - Courteous and very pleasant. Excellent at establishing good will.
5 - Inspiring to others. Artful listener. Really understanding.

D. Initiative

0 - Displays no self starting characteristics. Acts without purpose.
1 - Puts forth little effort. Requires prodding - sets no speed records.
2 - Puts in minimal effort to get task completed.
3- Strives hard. Desire to achieve.
4 - High desire to achieve. Always puts in a solid days work.
5 - Sets high goals. Self starter with high motivation. Constantly goes beyond call of duty.

E. Efficiency

0 - Work is invariably late.
1 - Work occasionally completed on schedule.
2 - Work usually complete on schedule. Some contribution to minor problem solving.
3 - Work always complete on schedule.
4 - Work complete. Consistent in defining and resolving major problems.
5 - Work invariably done ahead of schedule. Imaginative.
Can be counted on to make major contributions.

F. Personal Relations
0 - A very disruptive influence
1 - Is source of some friction
2 - Causes no problems
3 - Satisfactory, harmonious
4 - Is a positive factor
5 - Respected by others. Presence adds to environmental stability

G. Group Meeting Attendance
0 - Never attended any meetings. Showed no interest.
1 - Occasionally attended. Would commit and then not show.
2 - Sometimes uncooperative in planning schedule. Hard to get in touch with.

3 - Would attend. Usually late
4 - Could be counted on to attend.
5 - Never missed a meeting. Always on time

H. Attitude and Enthusiasm
0 - Poor disposition, uninvolved, indifferent
1 - Unenthusiastic, blase
2 - Half hearted
3 - Positive demeanour
4 - Positive attitude and spirited.
5 - Exuberant and eager. Positive influence. Inspiring to others. Team builder.

I. Effort
0 - Puts forth no effort. Expects others to carry the load.
1 - Puts forth some effort.
2 - Displays enough effort to get by.
3 - Solid contributions
4 - Strives very hard. Energetic.
5 - Self starter. Consistently goes beyond call of duty.

J. Dependability
0 - Uninvolved. Unreliable
1 - Unsteady, but tries somewhat.
2 - Occasionally would come through. Inconsistent.
3 - Needs some improvement. Suitable.
4 - Very trustworthy. Could be counted on to take responsibility.
5. Always responsible. Kept the group together and in the right direction. Steady influence

Sprint 1 Stage

Estimation Effort 1 – Task 61C

Marella Morad

Table of Contents

Introduction:

Based on previous experience with web development, gained through several units at university and my role as a Frontend Developer at work, I have selected the task of developing "Member Management Web Pages" for Sprint 1. To ensure an accurate effort estimate, it is necessary to decompose this backlog item into smaller, more manageable components using the Activity based WBS. Since there are multiple stages involved in achieving the item's Definition of Done as outlined in our Project Proposal, I will consider these stages, including designing, consulting, developing functionalities, and testing, as integral parts of completing this item.

Effort Estimations and Justification:

Design Phase:

Task	Estimation	Justification
Design the **UI** for the **Add New Member** Page	90 minutes	Designing the user interface could involve creating wireframes, mockups, and consulting with the product owner and the team to ensure it meets the requirements. This is a substantial task that may take up to 1.5 hours (the first page may take longer, but the second page will be based on the first one, so combined, they might take around 90 minutes).
Design the **UI** for the **Edit/Delete Existing Member** Page		

Development Phase:

Task	Estimation	Justification
Develop the **Add New Member** page functionality	90 minutes	Implementing the basic functionality of adding a new member, including form validation and error handling.
Develop the **Edit/Delete Existing Member** page functionality	90 minutes	Implementing the functionality to edit existing member details, including deleting existing member records and form validation.

Integration Phase:

Task	Estimation	Justification
Connect the **Add New Member** and **Edit/Delete Existing Member** pages to the backend and database.	30 minutes	Integrating the frontend with the backend and database for adding, editing, and deleting member records. This task does not require a lot of time as the API will already be set up.

Testing and QA Phase:

Task	Estimation	Justification
Test the **Add New Member** page and fix any issues	90 minutes	Performing manual testing to ensure the pages function correctly, i.e., members can be added successfully, then edited or deleted, and all the changes are reflected in the relevant database table(s).
Test the **Edit Existing Member** page and fix any issues		

Documentation and Review Phase:

Task	Estimation	Justification
Document the code and functionality	30 minutes	Documenting the code and functionality for future reference and maintenance.
Review with the product owner and team (through a team Showcase)	30 minutes	Reviewing the completed pages with the product owner and team to ensure they meet requirements and expectations.

Deployment Phase:

Task	Estimation	Justification
Deploy the pages to the production environment	30 minutes	Deploying the pages is a relatively quick process once development and testing are complete.

Total Estimation = 90+90+90+30+90+30+30+30 = 480 minutes = 8 hours

WBS for Member Management Web Pages

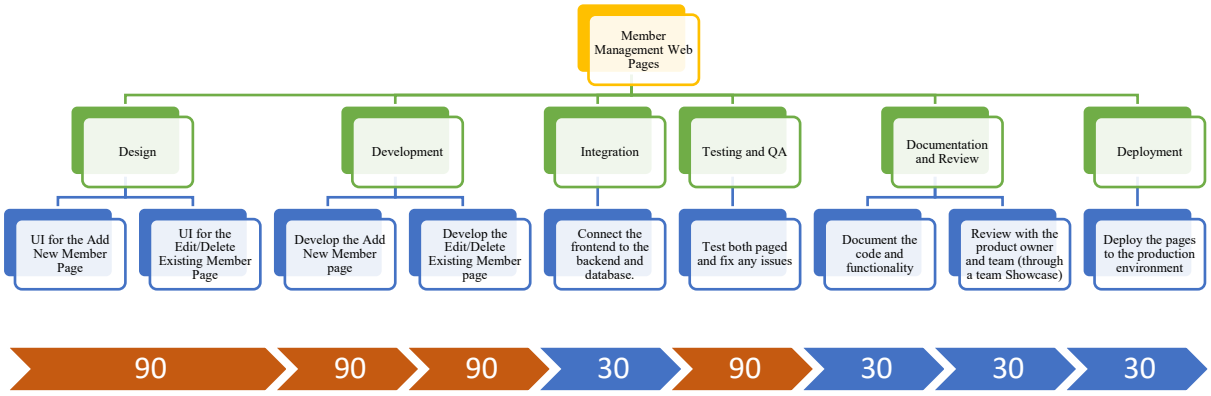


Figure 1 WBS for Member Management Web Pages including the Time Estimations in Minutes

Sprint 1 Stage

Estimation Reflection 1 – 62C

Marella Morad

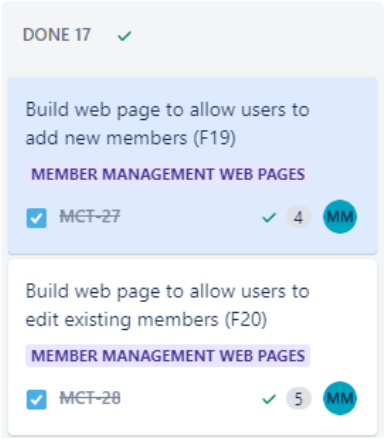
Table of Contents

Introduction

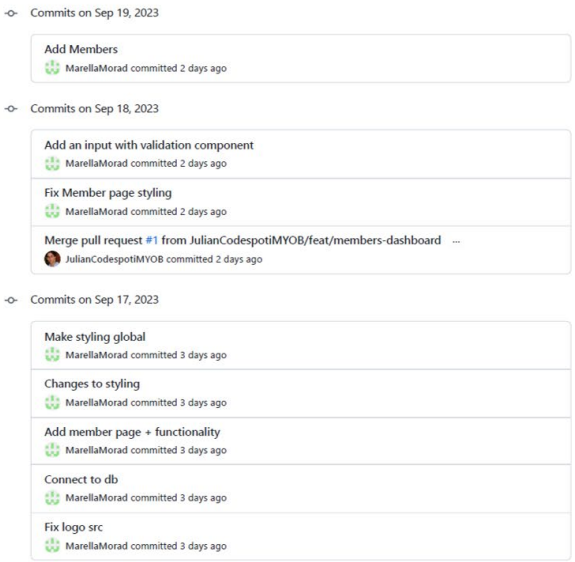
In this reflection report, I will assess the accuracy of my initial effort estimation for the development of the "Member Management Web Pages" during Sprint 1. The original estimation was 480 minutes (**8 hours**), and the actual time spent on the tasks was **9 hours**. I will discuss the reasons for the variance and provide insights into improving future estimations.

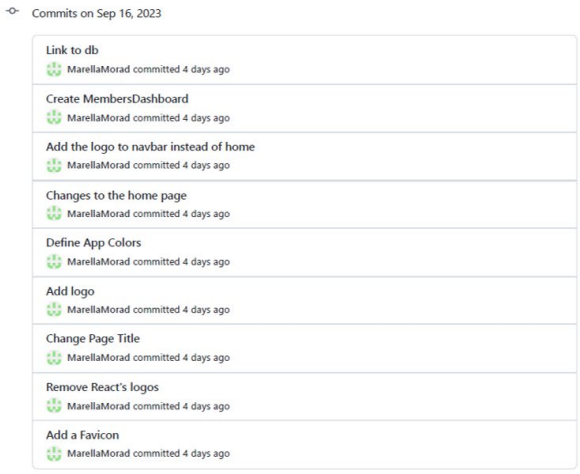
Evidence

JIRA Borad Tasks



GitHub Commit History





Accuracy Analysis

The original estimation for this project was reasonably close to the actual effort, with only a 12.5% variance. While this falls slightly outside the 10% range, it is still a relatively accurate estimation considering the complexity of the tasks involved.

Reasons for Accuracy

Experience

My previous experience with web development, both academically and professionally, played a crucial role in making accurate estimations. Having a solid understanding of the development process, including design, development, integration, testing, and deployment, allowed me to gauge the effort required for each task accurately.

Detailed Task Breakdown

Breaking down the project into smaller tasks, each with its own estimation, helped in identifying potential bottlenecks and areas that might require more time.

Justification

Providing detailed justifications for each task's estimation allowed for a clear understanding of why a particular amount of time was allocated to it. This transparency helped in assessing the estimations more critically.

Improvement Strategies

Although my estimations were reasonably accurate in this sprint, continuous improvement is vital. To enhance my estimation accuracy, I'll implement two key strategies. Firstly, I'll conduct regular retrospectives at the end of each sprint to scrutinize estimation accuracy, pinpoint discrepancies, and dissect the underlying reasons. These insights will serve as a foundation for refining future estimations. Secondly, recognizing the significance of contingency planning became evident during this task, particularly when developing the Edit/Delete Members web page. I encountered unforeseen delays while attempting to establish a connection to the members' table to retrieve, view, edit, or delete records. These unexpected issues consumed an additional hour as I worked diligently to identify the root cause. Fortunately, with the assistance of a team member, we were able to resolve the issue promptly, enabling me to get back on schedule. This firsthand experience underscores the importance of allocating buffers within estimations to accommodate such unforeseen challenges, reinforcing the need for a more robust and adaptable project management approach in future sprints.

Sprint 2 Stage

Estimation Effort 2 – Task 63C

Marella Morad

Table of Contents

Introduction:

In Sprint 2, I have undertaken the task of developing the “Product Management Web Pages”. This endeavour demands an accurate estimation of effort to ensure efficient project planning and successful delivery. To achieve this, I have decomposed the backlog item into its constituent components:

- Creating a Web Page for Searching and Viewing the List of Products.
- Creating a Web Page for Adding New Products.
- Creating a Web Page for Editing and Deleting Existing Products.

My choice of estimation technique for this task is “Estimating by Analogy”. This method has been selected based on the previous task I estimated and completed in Sprint 1, the “Member Management Web Pages” backlog item. The two tasks are very similar which inclined me to select the “Estimating by Analogy” method as it bases estimations by drawing comparisons with similar past projects or tasks. In this report, I will provide detailed information about the estimation process, its rationale, and the insights gained from experience, which together will facilitate a more accurate estimation of effort for the "Product Management Web Pages" in Sprint 2.

Estimations by Analogy:

Step 1: Finding a Similar Task:

As I mentioned earlier, I chose the estimations by analogy technique as I could base the efforts estimation of this task based on the “Member Management Web Pages” task that I completed in Sprint 1. Therefore, I will be using the “Member Management Web Pages” task for this comparison.

Step 2: Analyse Similarity and Differences:

In this step, I will compare the “Product Management Web Pages” task with the past task (Member Management Web Pages) in terms of functionality, user interface complexity, integration requirements and any unique challenges.

Similarities:

Both tasks involve designing and developing web pages with forms for managing data. i.e., search, view, add, edit, and delete data from the database. They share similarities in terms of functionality, development phases, and integration with the backend.

Differences:

One of the main differences in the specific data being managed (products vs. members). This could result in some variations in data validation and processing requirements. For example, products will have a product image, price, stock level and some other product specific properties that were not relevant to members.

Step 3: Estimate Effort for Similar Parts

In Sprint 1, I estimated 8 hours for the completion of the “Member Management Web Pages” task. In reality, it took me 9 hours to complete the task. Given the big similarity these two tasks have in common, I will estimate 6 hours for the completion of the “Product Management Web Pages” mainly because I can rely on the existing architecture when building the product specific pages, which will largely reduce my efforts.

Step 4: Estimate Effort for Differences

The primary distinction between members and products lies in our desire to incorporate an image for each product, not only to enhance visualization but also to align with industry standards. In order to

accommodate image display and retrieval on the new web pages, I would estimate an additional hour of work. I am confident that allocating one hour for this modification is adequate, drawing from my previous experience with integrating images into web pages.

Step 5: Calculate the Total Effort:

Summing up the efforts of similarities and differences adds up to a total of 7 hours (420 minutes).

Justification and Conclusion:

The "Estimating by Analogy" technique was employed to estimate the effort required for the current task, which involves the development of Products Management Web Pages, based on its similarity to a previous task involving Member Management Web Pages. Upon close analysis, it became evident that these two tasks share numerous similarities, especially in terms of development phases and functionality.

One significant advantage that influenced the estimation was the fact that the previous task was executed within the same project. This provided a valuable head start for the current task by offering existing templates and architectural foundations for the new web pages. As a result, the estimation for the common elements was reduced to 6 hours.

In regard to the different properties, the primary change involves the incorporation of image support, a feature not present in the previous task. This additional functionality is estimated to require an extra hour of effort. Consequently, the total estimated time for the completion of the "Products Management Web Pages" is 7 hours.

I have confidence in this estimation, given the thorough analysis of similarities and differences between the two tasks. I believe that the allocated 7 hours represent a realistic timeframe for the successful completion of this task.

Sprint 2 Stage

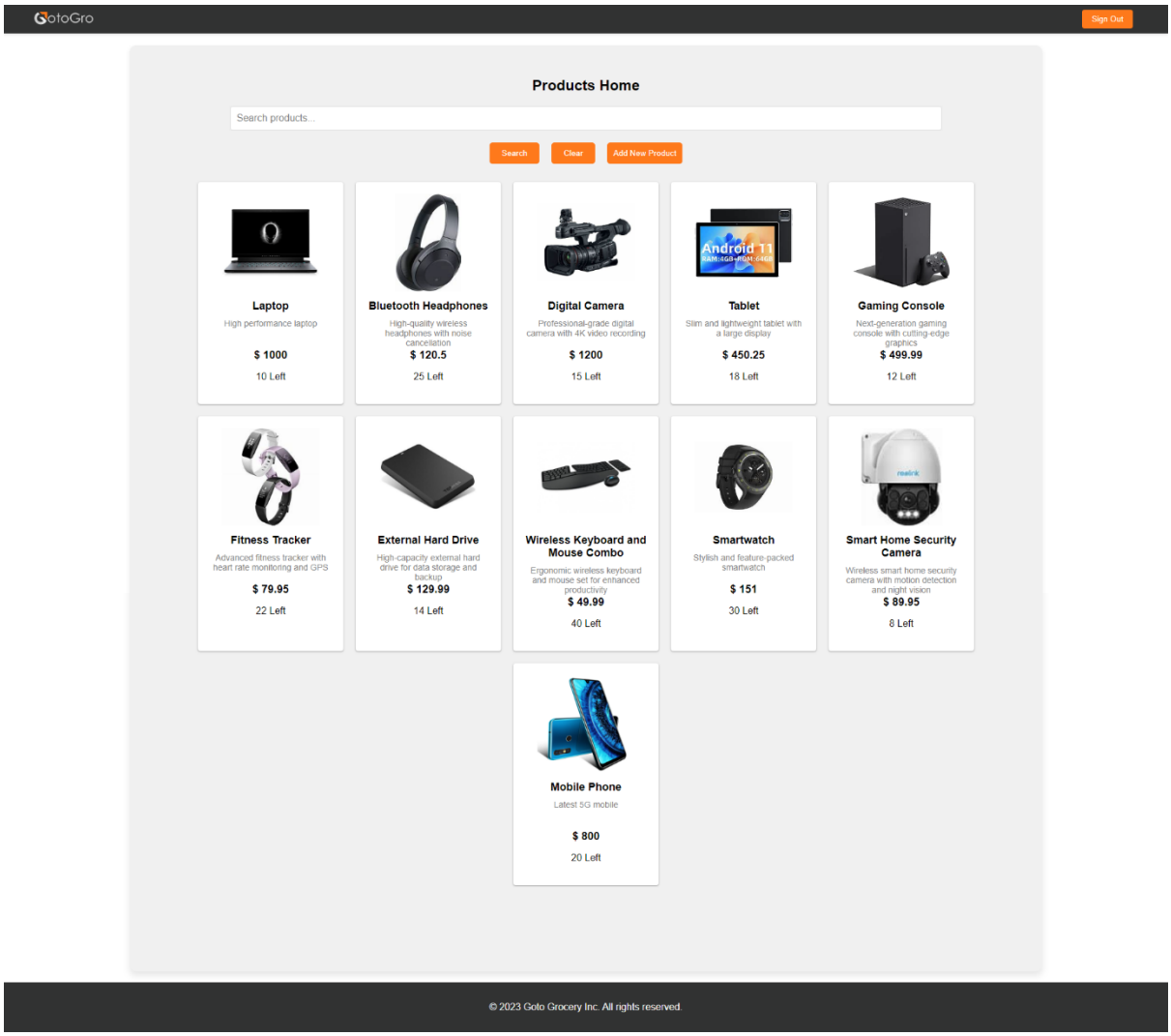
Estimation Reflection 2 – 64C

Marella Morad

Table of Contents

Introduction:

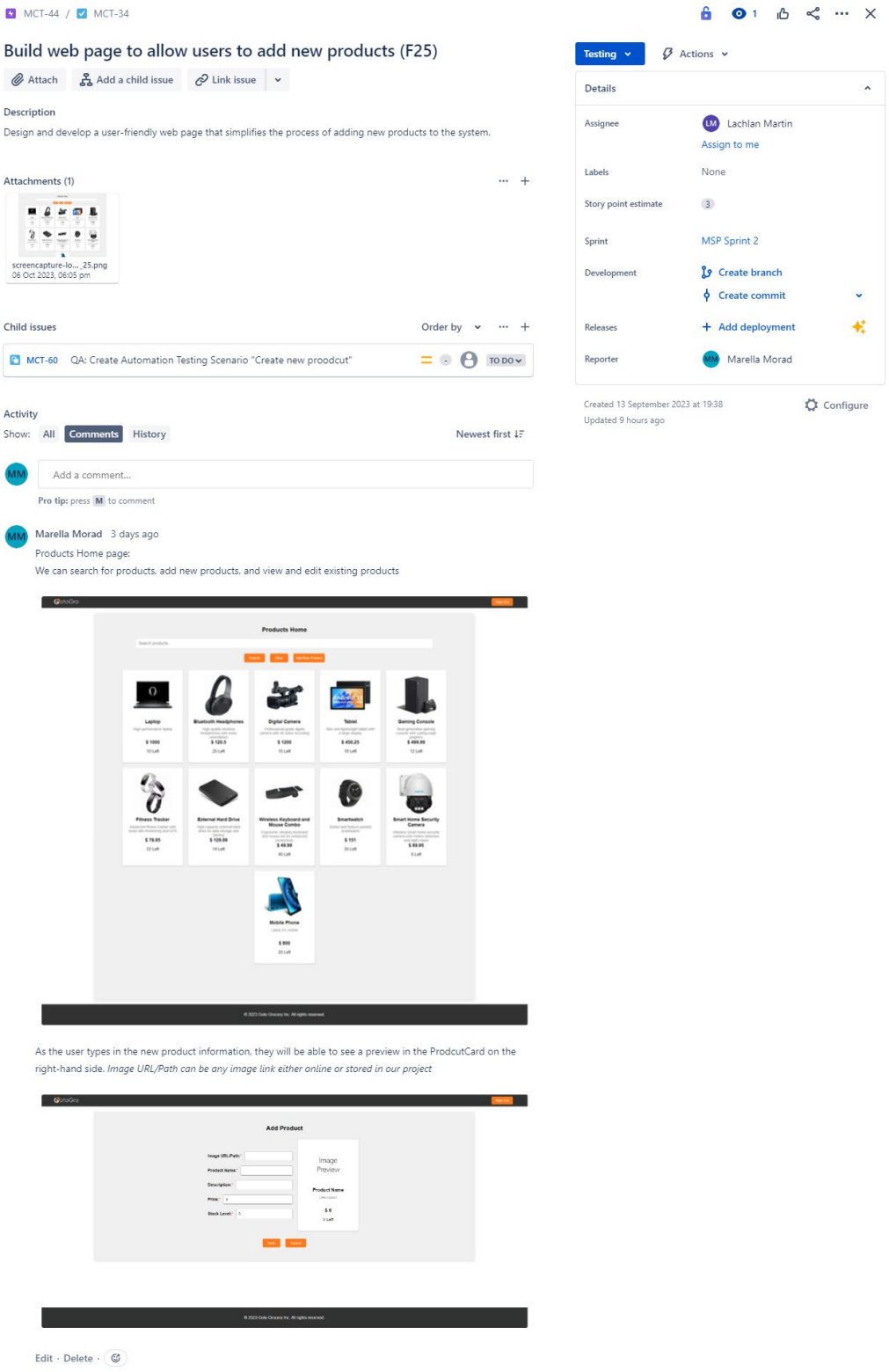
In Sprint 2, I undertook the task of developing the “Product Management Web Pages” epic and estimated the effort required using the “Estimating by Analogy” technique. The initial estimation was 7 hours, and I completed the task in 6.5 hours. In this reflection report, I will analyse the accuracy of my estimation and discuss the reasons for this accuracy, as well as suggest ways to improve future estimations.



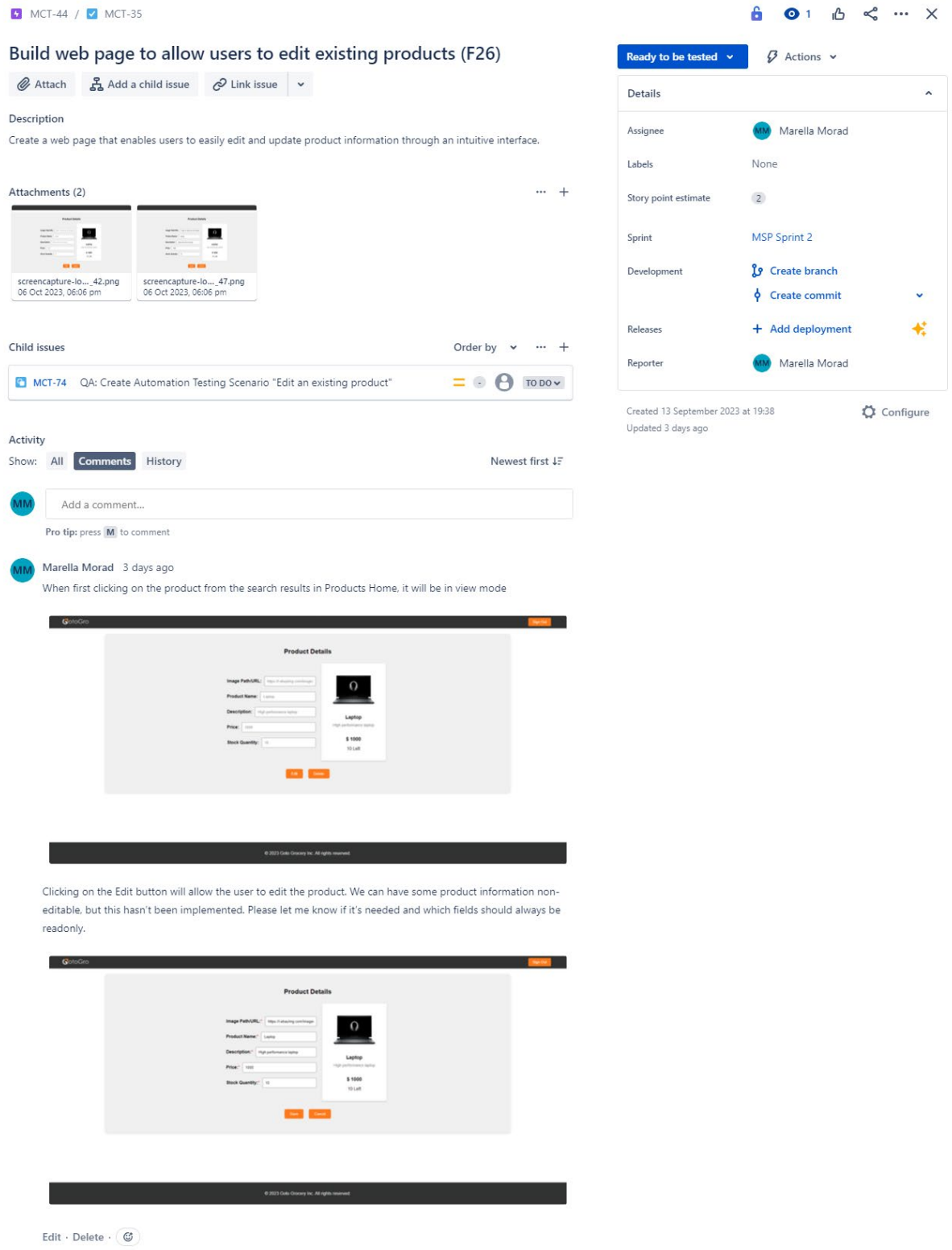
Evidence

JIRA Borad Tasks – Link to Epic and Tasks

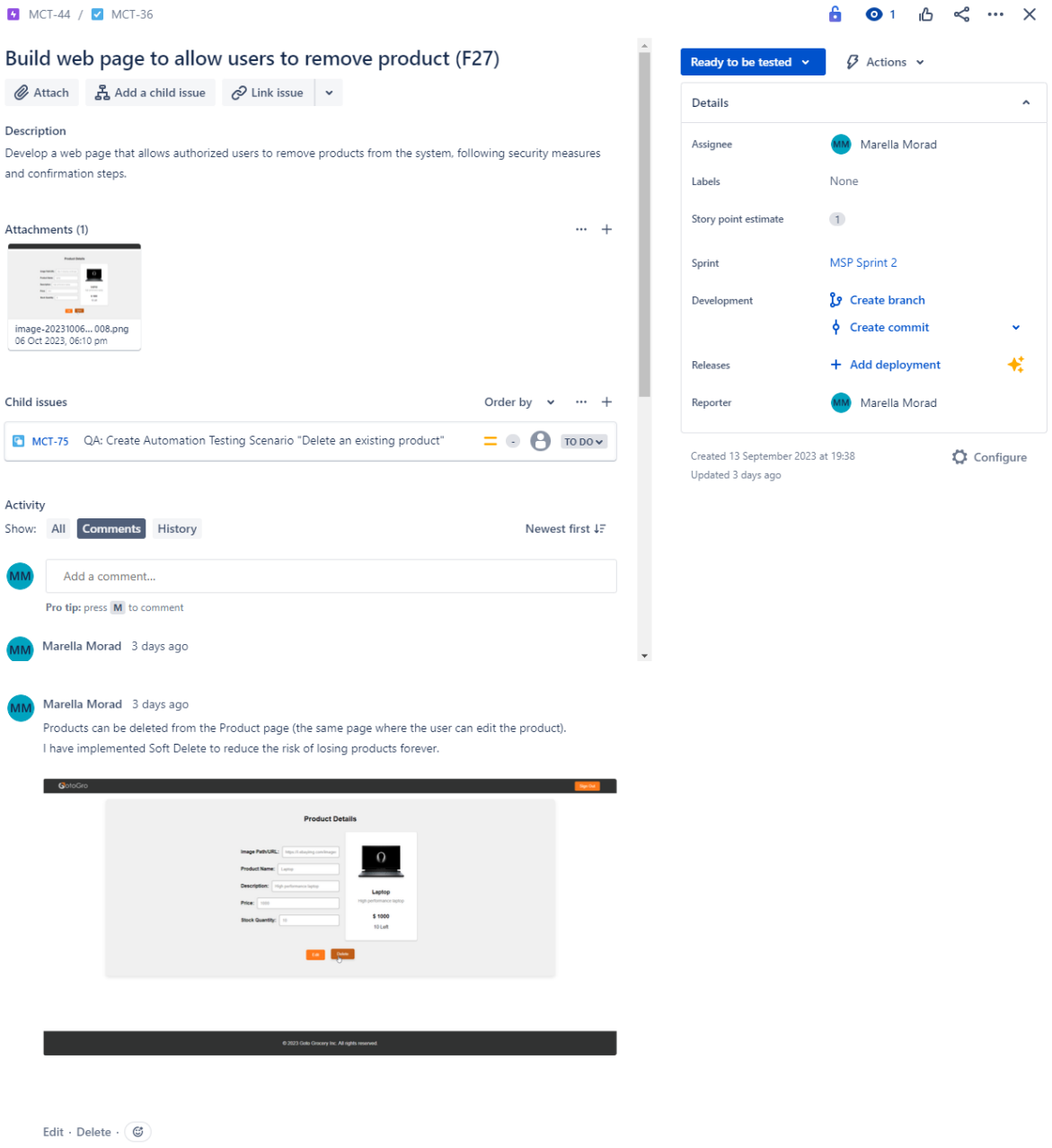
Build Web Page to allow Users to Add New Products



Build Web Page to allow Users to Edit Existing Products

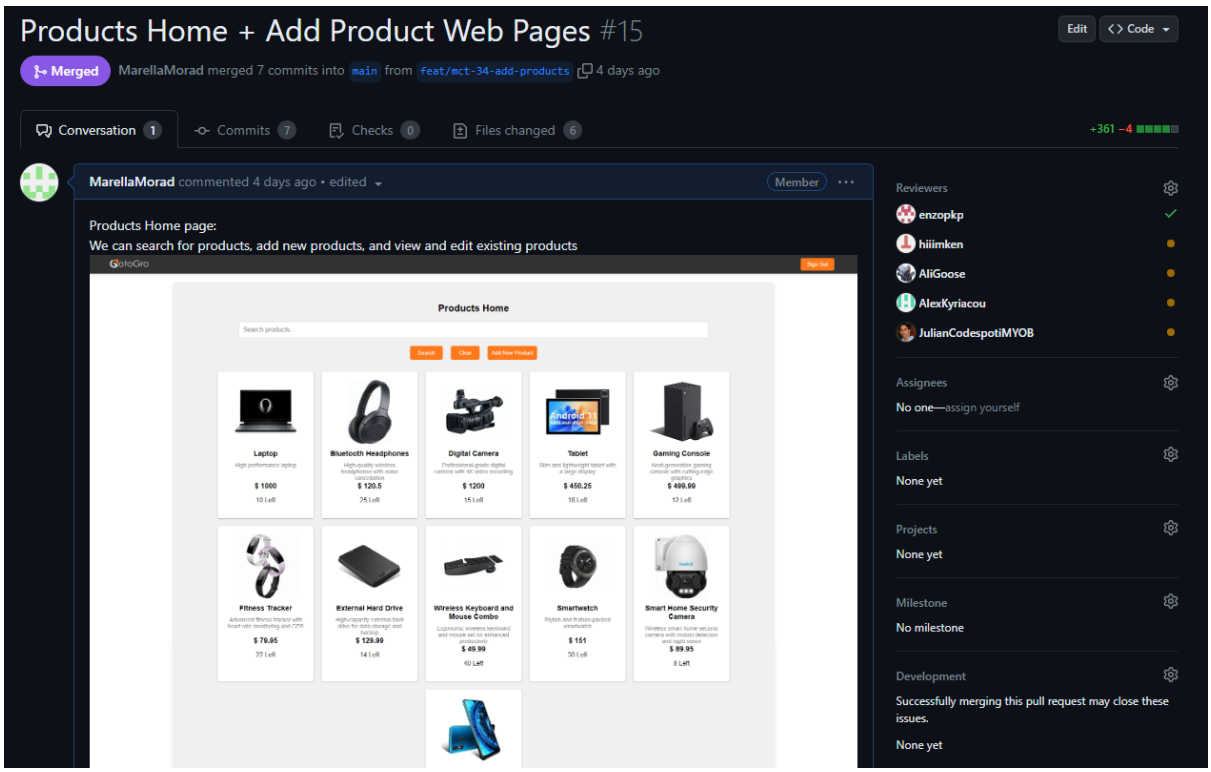


Build Web Page to allow Users to Remove Existing Products

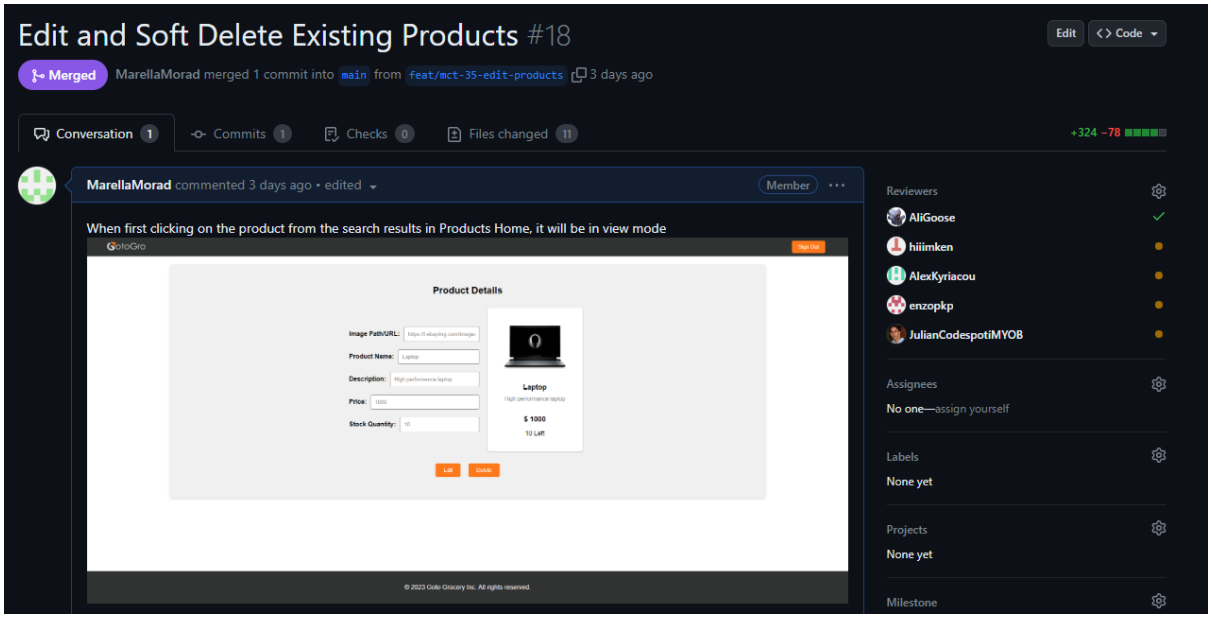


GitHub Pull Requests:

Products Home + Add Product Web Pages - Link to PR



Edit and Soft Delete Existing Products – Link to PR



Accuracy of Estimation:

The actual effort of 6.5 hours falls within 10% of my original estimate of 7 hours. Therefore, I can conclude that my estimation was reasonably accurate for this sprint backlog item which is an improvement from the previous estimation in task 61C for Sprint 1 backlog item. This level of accuracy can be attributed to several factors:

Reasons for Accuracy:

Having completed a Similar Task in Sprint 1:

One of the key reasons for the accuracy of my estimation was the experience gained from a similar task in Sprint 1, the “Member Management Web Pages”. Since both tasks involved designing and developing web pages with similar functionality and integration requirements, my experience from the first task allowed me to estimate the second task more accurately.

Use of “Estimating by Analogy” Technique:

The "Estimating by Analogy" technique proved effective in this case. By drawing parallels between the two tasks, I could identify commonalities and differences, which guided my estimation process. This technique leveraged my past experience and reduced estimation errors.

Thorough Analysis of Differences:

I accurately identified the differences between the two tasks, mainly the incorporation of product images. By estimating an additional hour for this specific modification, I accounted for the extra effort required for this unique feature accurately.

Realistic Contingency:

In my initial estimate, I did not allocate a significant amount of contingency time because I had confidence in the existing project foundations. However, I did acknowledge the potential for variations, which allowed for a slightly flexible estimate.

Improvements for Future Estimations:

While my estimation for this sprint was accurate, there is always room for improvement to ensure consistently precise estimates in future sprints.

Detailed Analysis:

Continue to conduct a thorough analysis of the similarities and differences between the current task and past tasks especially if they have similarities. This analysis should include not only functionality but also any potential challenges or complexities that may arise during development.

Historical Data:

Maintain a record of past estimation accuracy to better inform future estimations. This historical data can help identify patterns and trends in estimation accuracy and allow for adjustments as needed.

Collaboration:

Seek input from team members who have experience with similar tasks, for example, if another team member had completed the Member Management Web Pages task, it would've been wiser to seek their advice about this estimation rather than estimating it myself without previous experience. Collaborative estimation can provide different perspectives and insights, leading to more accurate estimates.

Adjust Contingency:

Depending on the level of uncertainty, consider adjusting the contingency time in the estimation. While it's essential to maintain a realistic estimate, a slightly wider contingency can account for unexpected challenges.

Conclusion:

In summary, my effort estimation for the "Product Management Web Pages" task during Sprint 2 proved to be reasonably accurate, falling within 10% of the actual effort required. This level of precision was the result of a combination of factors, including my experience, a thorough analysis of the task, and a pragmatic approach to contingency planning.

Looking ahead, I am committed to further enhancing my estimation skills for future sprints. This involves a commitment to detailed task analysis, leveraging historical data to inform estimates, fostering collaboration within the team to gain diverse insights, and remaining flexible in adjusting contingency time when necessary. By continuously learning and refining my estimation capabilities, I aspire to consistently deliver accurate estimations, thereby contributing significantly to the success of our ongoing projects.

71D

Quality Definition

Marella Morad

Table of Contents

Chosen Backlog Item:

For this analysis, I have selected the "Member Management Web Pages" task, which was the same task I chose for 61C and 62C. I chose this task because I was the main developer working on it during Sprint 1, and I am familiar with all its functionalities. Additionally, I selected it because it enables me to test the following characteristic and sub-characteristic from the ISO25010 Software Quality Model:

- Characteristic: Useability
- Sub-characteristic: Operability

Characteristic – Usability:

Usability refers to the extent to which a software product can be used by specified users to achieve specific goals with **effectiveness**, **efficiency**, and **satisfaction** in a specified context of use.

Justification:

Usability is a fundamental characteristic because it directly impacts how users interact with and experience our software. In the context of GotoGro MRM, the usability of the "Member Management Web Pages" is critical because it determines how easily users can perform tasks related to member management. For example, how easy and straightforward is it for the user to add a new member? Are there enough pieces of information on the page to guide the user for a smooth experience? Are there warnings for invalid inputs? If the system is not usable, users may struggle with basic functions, become frustrated, and potentially abandon the software. Therefore, ensuring usability is essential for **user satisfaction**, **productivity**, and the **overall success** of the project.

Sub-characteristic – Operability:

Operability is a sub-characteristic of usability that focuses on the ease with which users can operate and control the software to achieve their specific tasks efficiently. It also measures the software's tolerance for user errors and its alignment with user expectations.

Justification:

In the context of GotoGro Member Management Web Pages, operability plays a pivotal role in ensuring that users can perform member management tasks with minimal effort and without encountering obstacles. It addresses the need for an intuitive user interface that aligns with users' mental models and expectations. Furthermore, by assessing the software's tolerance for user errors, operability contributes to error prevention and recovery, enhancing the overall user experience. Operability is essential for the project's success as it directly impacts user satisfaction, reduces the learning curve for new users, and fosters efficient usage of the application.

Metric 1 - Efficiency of Adding a Member:

Threshold Value:

The average time taken by users to add a new member should be less than 2 minutes.

Justification:

Efficiency in adding a member is a crucial aspect of operability. It directly affects how quickly users can perform a core task within the "Member Management Web Pages". Setting a threshold of less than 2 minutes ensures that adding a new member is a quick and straightforward process, aligning with user expectations for efficiency. Longer processing times could frustrate users and reduce their productivity. Therefore, by keeping the average time within this threshold, the project aims to provide an operable system that allows users to efficiently complete this essential task.

Metric 2 - System Recovery in Case of Member Deletion by Mistake:

Threshold Value:

The system should provide a "soft delete" feature, allowing users to recover deleted members within 24 hours of the deletion without needing to contact IT Support.

Justification:

The ability to recover from user errors, such as accidental member deletions, is a critical aspect of operability. By setting a threshold of 24 hours for recovery, the project aims to align with user expectations for a reasonable window of time to correct mistakes. Allowing users to retrieve deleted members within this timeframe enhances the software's tolerance for user errors and contributes to a positive user experience. This metric emphasizes the importance of system robustness and error recovery, both of which are essential for operability.

References:

Standards Australia/New Zealand, 2013, Systems and software engineering - Systems and software quality requirements and evaluation (SQuaRE) - System and software quality models, AS/NZS ISO/IEC 25010:2013, Sydney/Wellington: Standards Australia/New Zealand, viewed 20 August 2023.

72D

Quality Planning

Marella Morad

Table of Contents

Introduction:

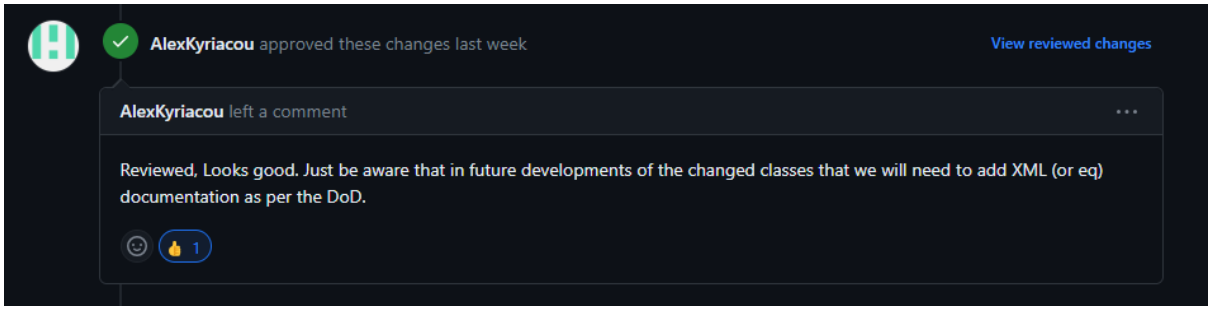
In this report, I will document my comprehensive plan to ensure the quality of the two metrics defined in 71D. The two metrics are as follows:

Metric	Threshold
Efficiency of Adding a Member.	The average time taken by users to add a new member should be less than 2 minutes.
System Recovery in Case of Member Deletion by Mistake.	The system should provide a "soft delete" feature, allowing users to recover deleted members within 48 hours by contacting IT Support.

Quality Planning Actions:

Code Review:

Since this feature was developed in Sprint 1, I already have evidence for conducting a code review. After I finished implementing this feature, I raised a Pull Request and requested review from my team members. I received a review from the Product Owner – Alexander Kyriacou as shown below:



I had missed XML documentation which was one of the standards that we agreed on before commencing development. I added the relevant documentation before merging the changes into our main branch.

Unit Testing:

In Sprint 2, I am planning to develop unit tests to cover critical functions and components of the Member Management Web Pages task. In Sprint 1, I focused on getting the functionality working and relied on manual testing during development, without running any automated tests. Personally, I prefer using a test-driven development (TDD) approach whenever I'm developing a new feature. However, I believe it is important to also have unit tests, including cases related to adding members and retrieving soft-deleted members. These tests will enable me to check if the above-mentioned metrics have been met and will indicate if a change is required. For example, a performance boost may be needed, or there may be a mistake in the logic of retrieving records.

Usability Testing:

As part of Sprint 1, the tester in our team, Nelchael, performed a simple usability testing on these web pages and raised some concerns about appearance and functionality, and I tried to address most of them in Sprint 1. In Sprint 2, I'll work on improvements to these pages, and I will write some test scenarios for Nelchael to execute that reflect real-world use cases, such as adding a new member and recovering a deleted member.

Detailed Test Plan:

Participants:

I will try to get all the team involved in this testing phase as these pages are very critical for the performance of our web application. I will seek help from the other developer in the team for better coding practices and performance boosting approaches. I will also ask the product owner, scrum master and tester for their feedback regarding different functionalities.

Test Cases:

Unit Test Cases: I will write some internal component (unit) test cases to validate code correctness and component behaviour.

- Test case examples: Verify member creation, test soft delete functionality, and recovery process.

Usability Test Cases: Developed in collaboration with product owner for a better insights and representation of the end user group.

- Test case examples: Add a new member, attempt to delete a member by mistake and recover within 24 hours, and navigate through the member management pages for efficiency.

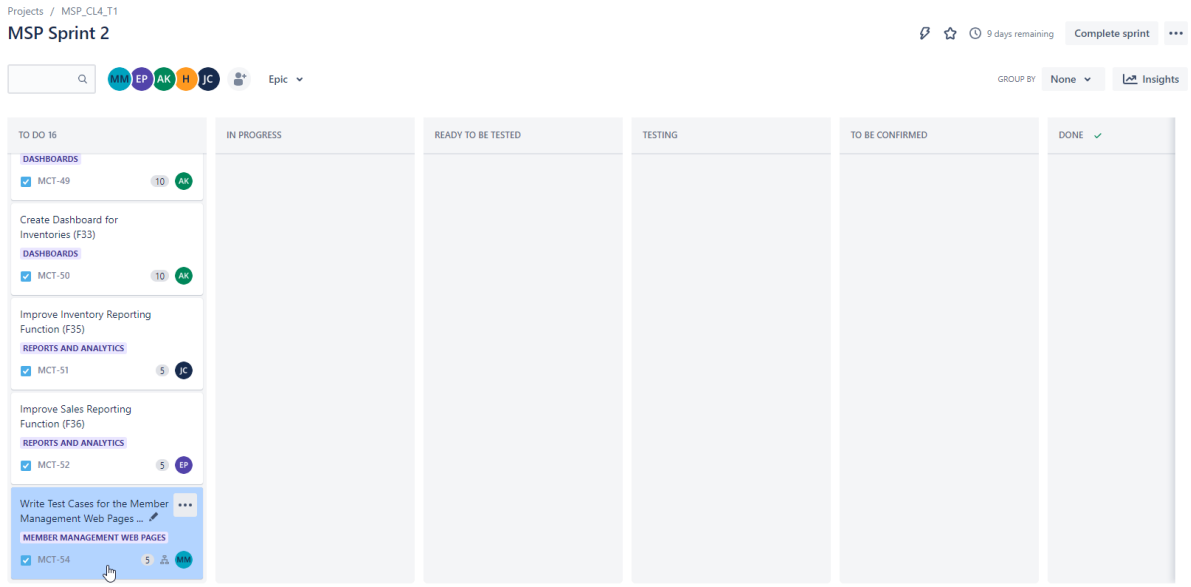
Tasks:

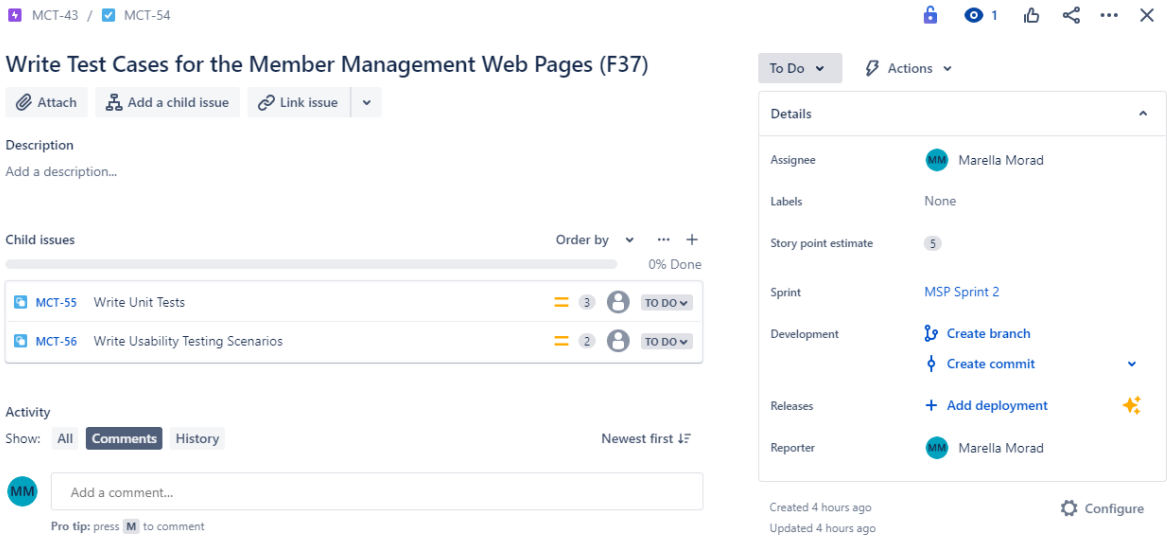
I will write unit tests for the Member Management Web Pages task which includes the MembersDashboard.js, Member.js and AddMember.js pages. Then once these tests pass, I will move the task to Testing and assign it to our tester who will get a chance to execute the usability testing scenarios, document any issues and provide feedback or raise bugs as applicable.

Metrics:

- Unit Test Metrics: Pass/fail status for each unit test case.
- Usability Test Metrics: Measure task completion time, and user satisfaction, and identify usability issues.

Test plan evidence in JIRA Board:




MCT-43 / MCT-54
1
Write Test Cases for the Member Management Web Pages (F37)
Attach
Add a child issue
Link issue
Description
Add a description...
Child issues
Order by
0% Done
MCT-55 Write Unit Tests
3
TO DO
MCT-56 Write Usability Testing Scenarios
2
TO DO
Activity
Show:
All
Comments
History
Newest first
Add a comment...
Pro tip: press M to comment
To Do
Actions
Details
Assignee
Marella Morad
Labels
None
Story point estimate
5
Sprint
MSP Sprint 2
Development
Create branch
Create commit
Releases
Add deployment
Reporter
Marella Morad
Created 4 hours ago
Updated 4 hours ago
Configure

73D

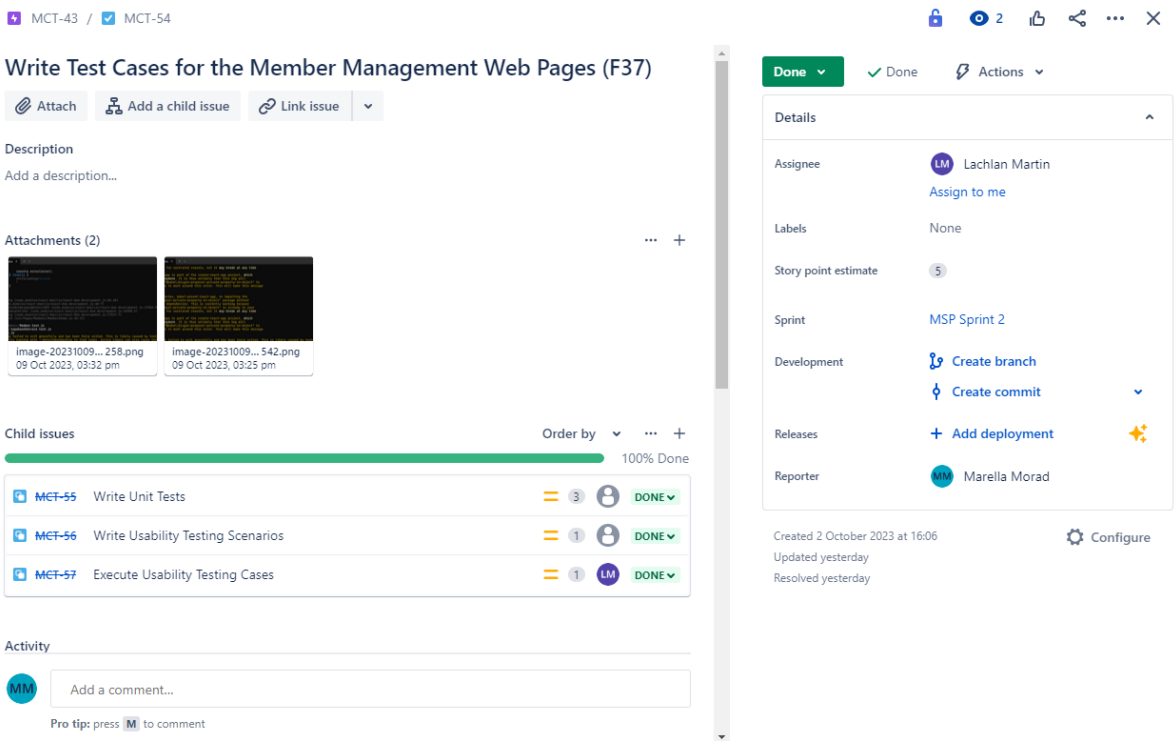
Quality Review and Reflection

Marella Morad

Table of Contents

Introduction:

After conducting a thorough review of the quality planning and review activities in accordance with Distinction Tasks 71D and 72D, I have gathered substantial evidence from both our JIRA board and GitHub Repository. The primary focus of this review was to assess two key metrics: Unit Tests, which serve as a means to verify the technical functionality of our system, and Usability Testing Scenarios, which provide valuable insights into the user-friendliness of our system, particularly with regard to our Member Management Web Pages. In this report, I will present the results of this quality review, shedding light on the strengths and areas for improvement in our software development process.



Unit Tests Execution:

As outlined in this Pull Request, I wrote test cases to internally test the member management web pages. The test cases include the following:

AddMember component:

- `renders AddMember component`: Checks if the AddMember component renders correctly with all the required input fields and buttons.
- `adds a member`: Simulates user input and tests if the addMember function is called with the expected data when the "Add" button is clicked.
- `fails to add a member if required field is missing a value`: Tests if the addMember function is not called when a required input field is left empty.

Member component:

- `renders Member component`: Ensures that the Member component renders correctly with member details, edit button, and delete button.
- `edit member details`: Simulates clicking the edit button, updating a member's details, and checks if the updateMember function is called with the expected data.
- `soft delete member`: Simulates clicking the delete button and checks if the softDeleteMember function is called with the correct member data.

MembersHome component:

- `renders MembersHome component`: Checks if the MembersHome component renders correctly with various UI elements like search bar, buttons, and title.
- `searches for members`: Tests the search functionality by simulating a search query and verifying that the searchMembersByName function is called with the correct search query.
- `isLoading is true after handleSearch`: Verifies that the "Loading..." message is displayed when the search button is clicked.
- `saves selected member to local storage and navigates to the member page`: Tests if selecting a member from search results saves their details to local storage and allows navigation to the Member page.
- `retrieve recently deleted member (before 24 hours have passed)`: Tests the retrieval of a recently deleted member, including calling retrieveDeletedMember and checking if the selected member is saved to local storage.

```
 PASS  src/Pages/Members/AddMember.test.js
 PASS  src/Pages/Members/MembersHome.test.js
 PASS  src/Pages/Members/Member.test.js
 PASS  src/App.test.js
 PASS  src/Supabase/supabaseService.test.js

Test Suites: 5 passed, 5 total
Tests:       15 passed, 15 total
Snapshots:   0 total
Time:        2.477 s, estimated 4 s
```

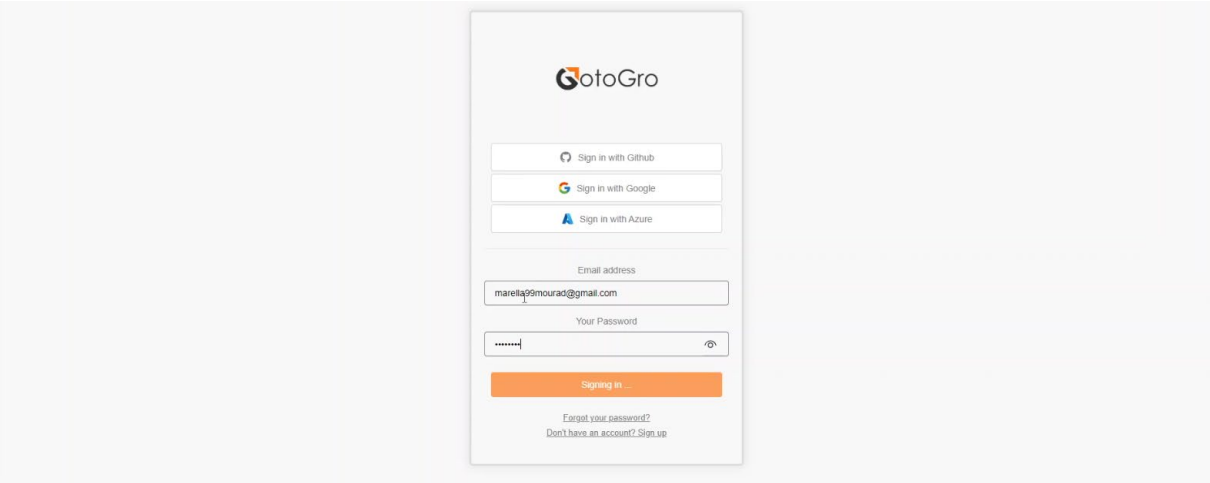
As can be seen above, all of the above tests passed.

Usability Testing Scenarios:

Test Case 1: Add a New Member in Less Than 2 Minutes

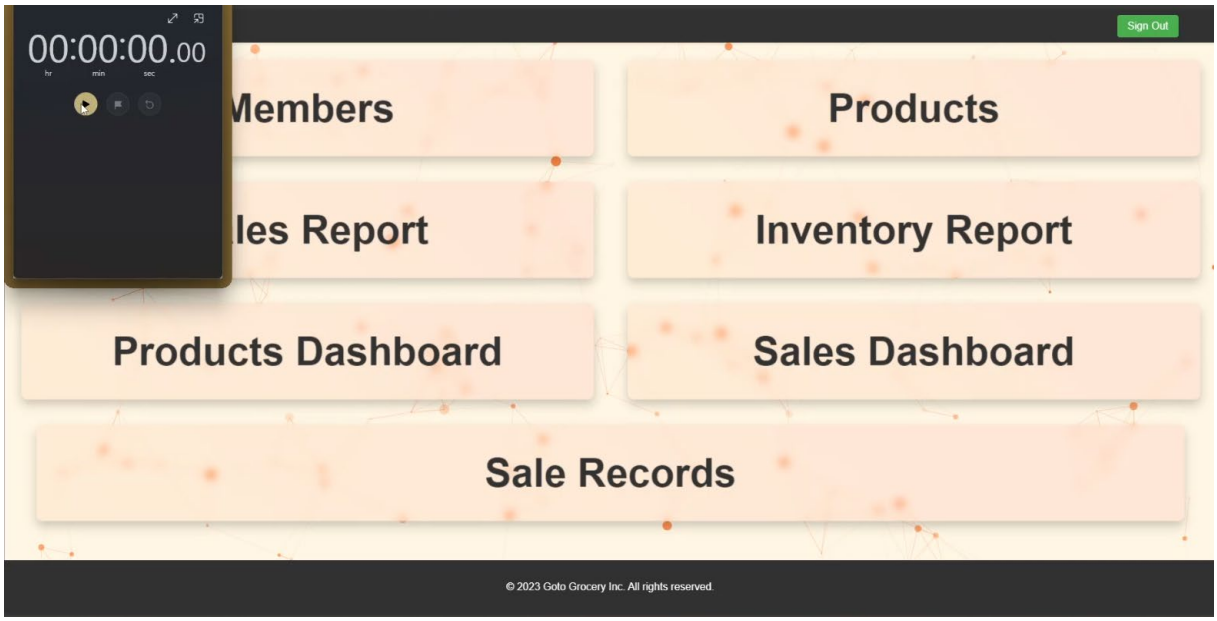
I executed the **Outlined Steps as Per MCT-56** and found the following results**:**

1. Navigate to GotoGro MRM.
2. Log in with valid credentials.



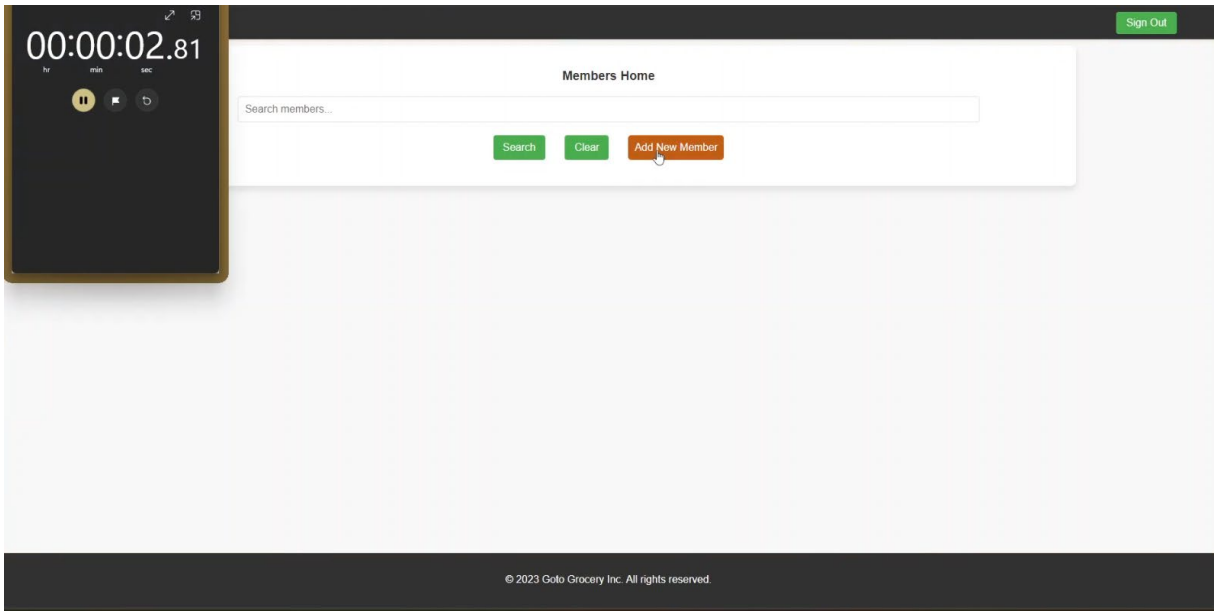
I was able to successfully login to GotoGro using credentials I signed up with a few days ago ✓.

3. Start a timer to measure the time from starting the process to completing the submission.

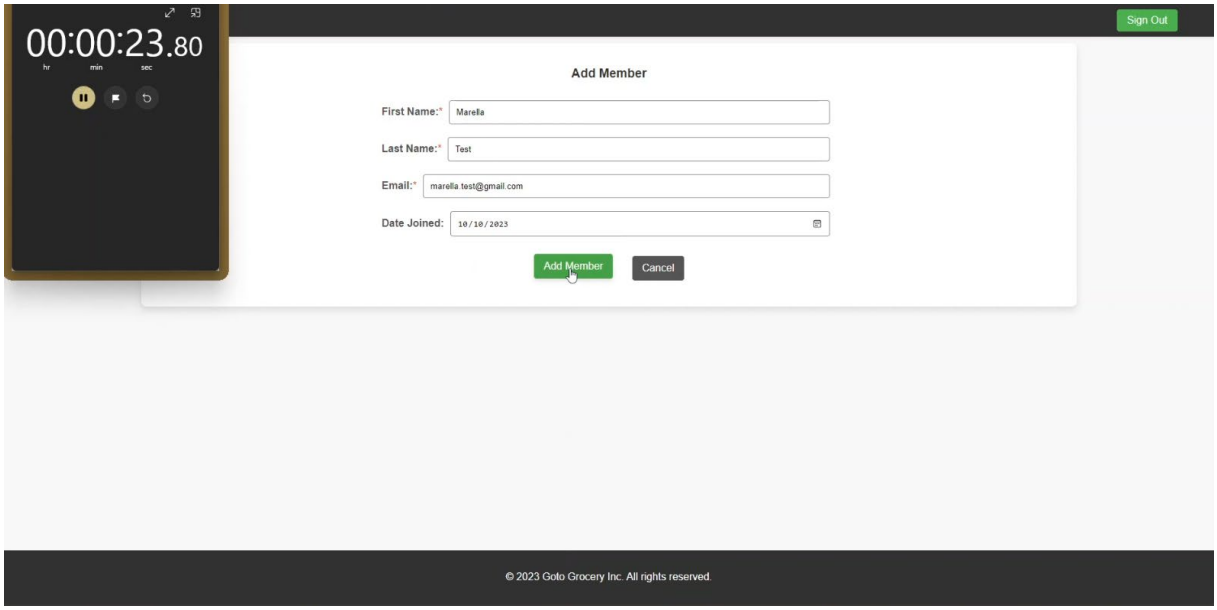


I started a timer as soon as the home screen of GotoGro loaded ✓

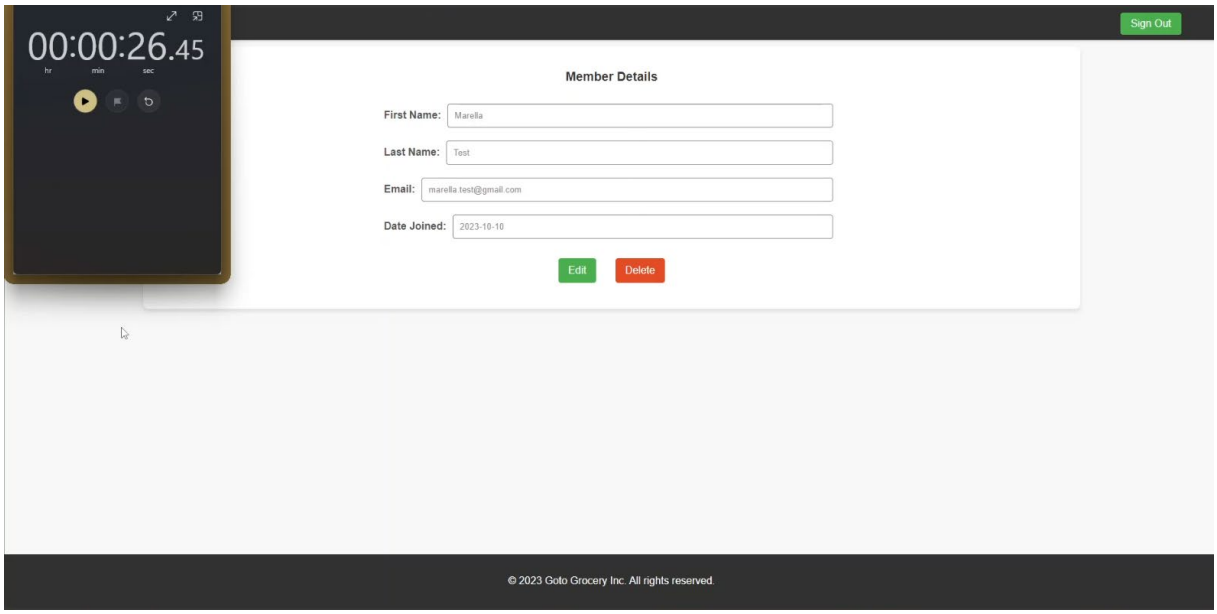
4. Navigate to **Members**.
5. Click on **Add New Member**.



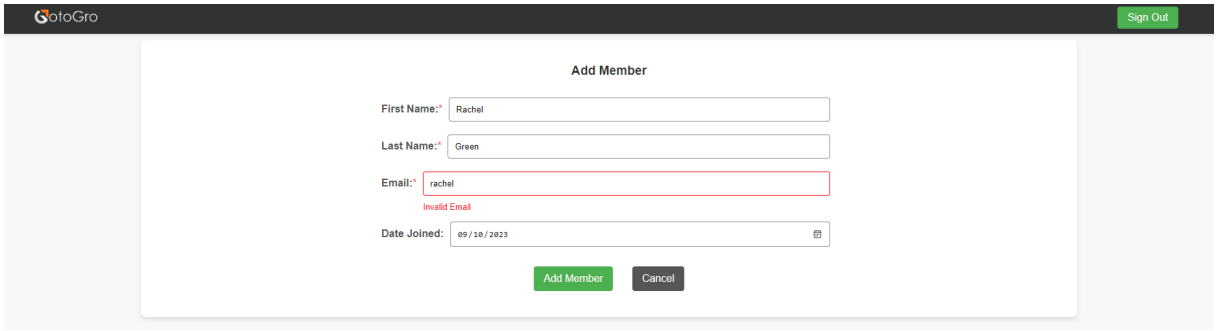
6. Fill out the necessary member details (e.g., first name, last name and email).
7. Verify that all required fields are clearly labelled and easy to find.
8. Submit the form to add the new member.

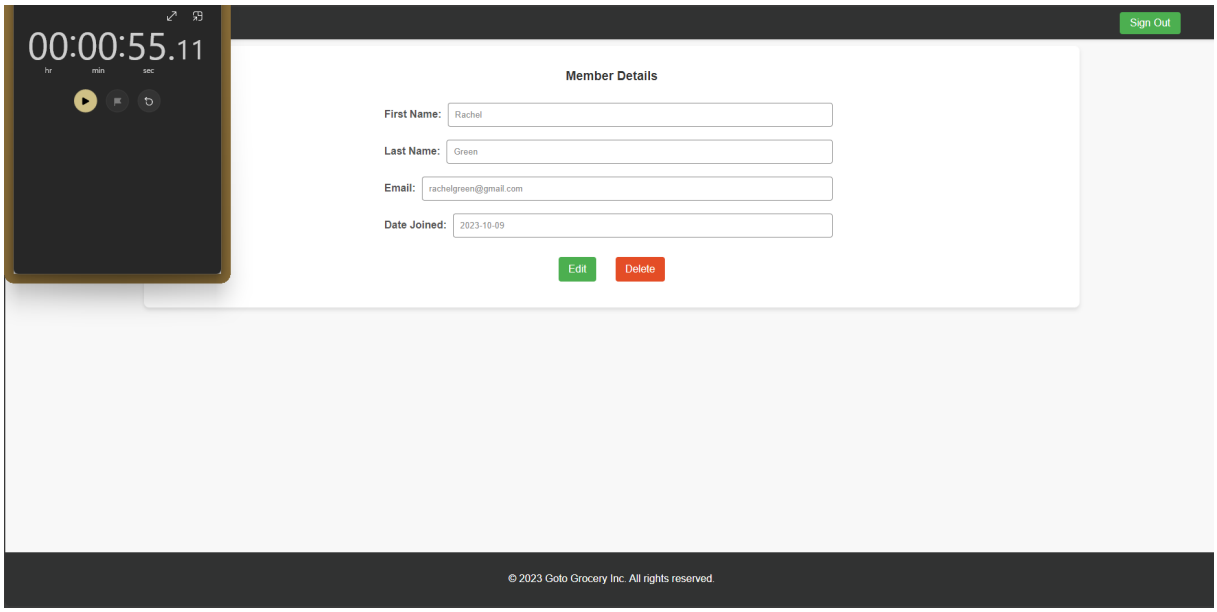


9. Ensure that the entire process takes less than 2 minutes.



As shown above, adding a new member indeed takes less than 2 minutes; in fact, it takes less than 30 seconds. However, I conducted this test personally, and I consider myself an experienced user. Additionally, I developed the page, making it very familiar to me. Therefore, I have decided to ask someone else with no prior experience with GotoGro to perform the test.





The other user was still able to add a new member within a minute, even after entering an invalid email address. The form prompted the user about the invalid email, and the user was able to save after entering a valid email. This outcome still meets the defined quality measure.

Expected Results:

- The system should allow users to add a new member in less than 2 minutes.
- The process should be straightforward and intuitive, with clear instructions and error handling.

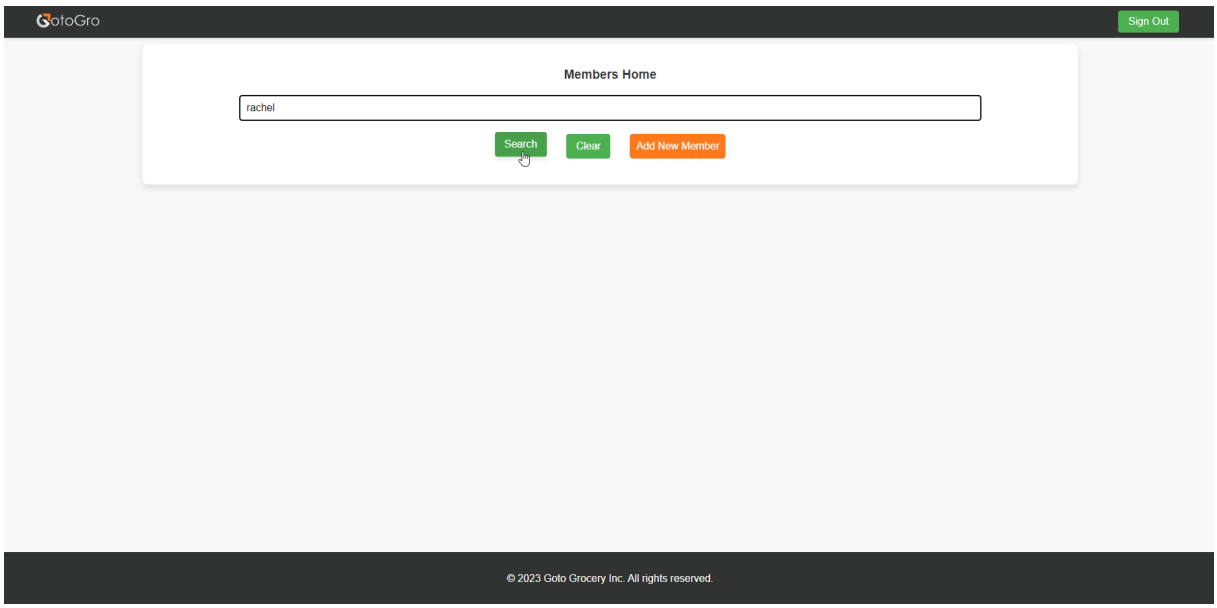
Actual Results:

- The system allows users to add a new member in less than 2 minutes.
- The process is straightforward and intuitive, with clear instructions and error handling.

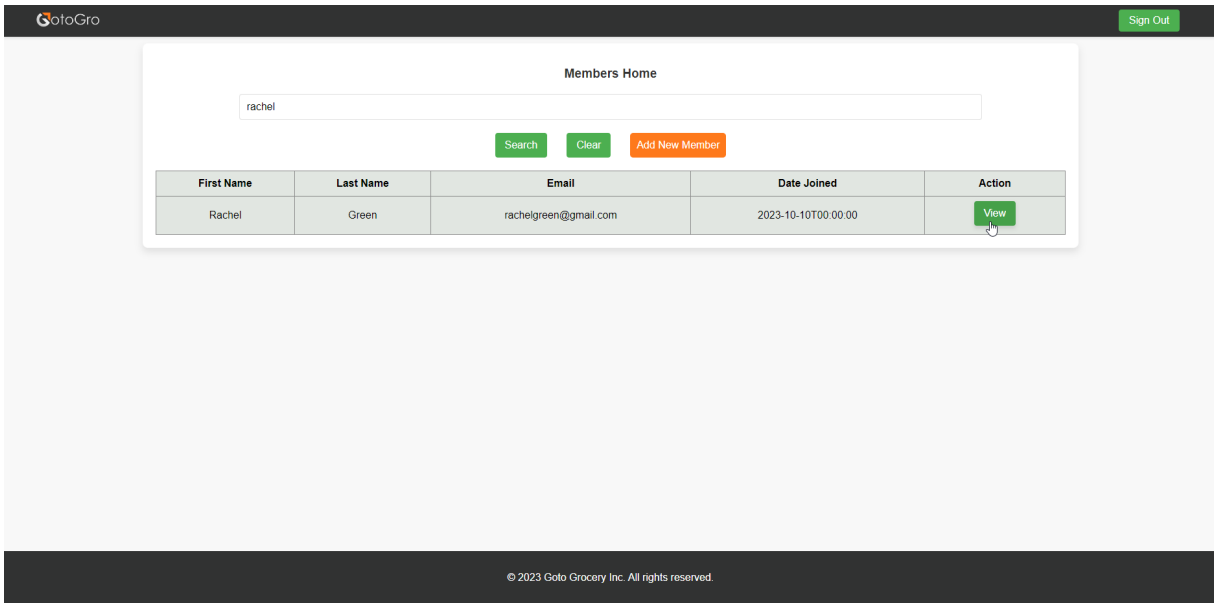
Test Case 2: Attempt to Delete a Member by Mistake and Recover Within 24 Hours

I executed the **Outlined Steps as Per** [MCT-56] and found the following results**:**

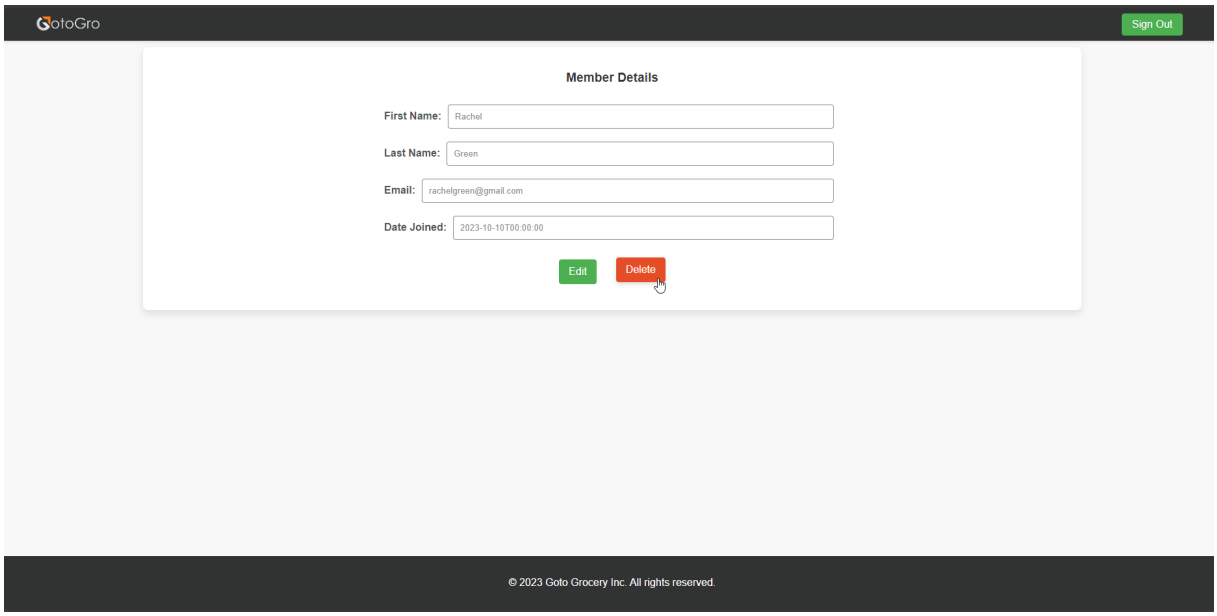
1. Navigate to GotoGro MRM.
2. Log in with valid credentials.
3. Navigate to **Members Home**.
4. Enter a search term (or leave it blank) and click on **Search**.



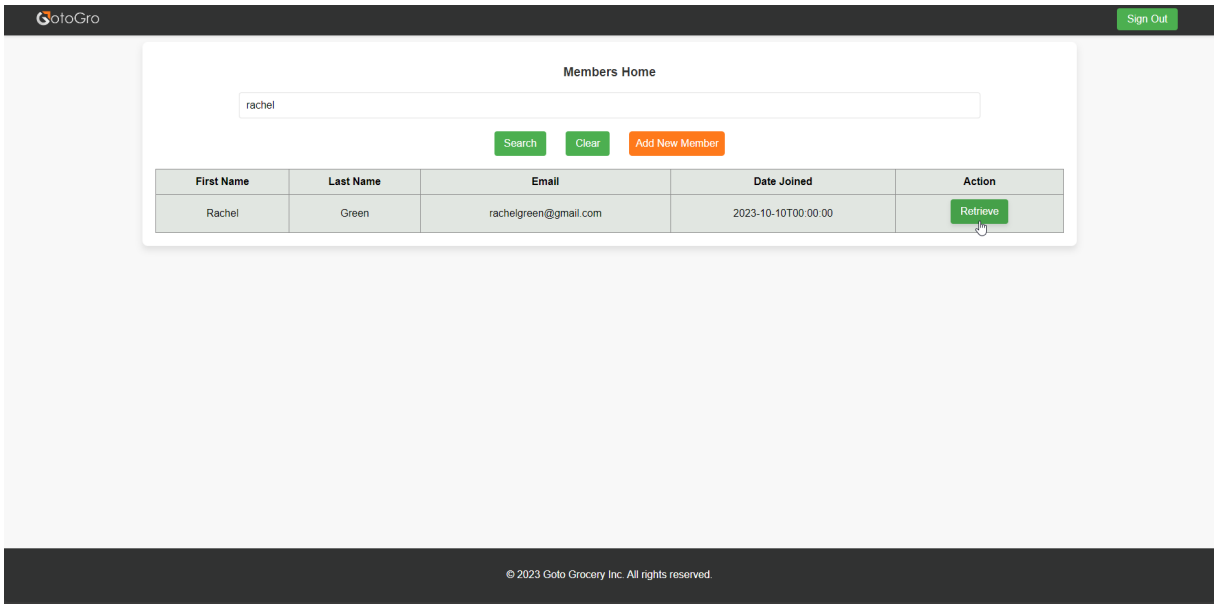
5. Click on the **View** button of one of the found members.



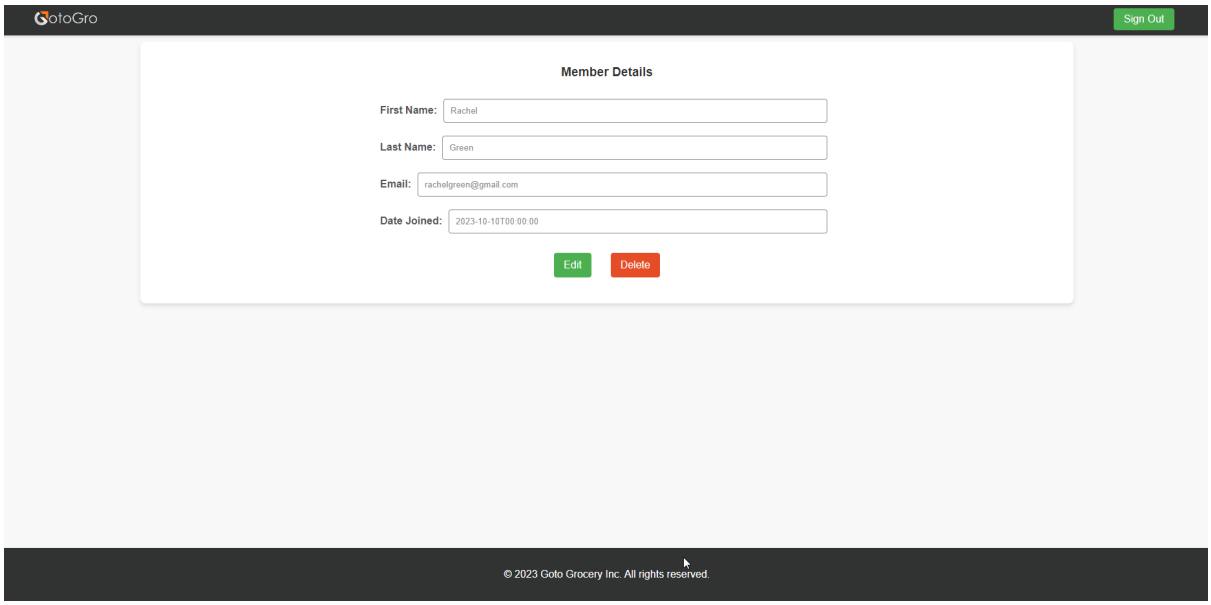
6. Click on the **Delete** button from the **Member Details** screen.



7. Search for the deleted member from **Members Home**.
8. Click on the **Retrieve** button to retrieve the deleted member record.



9. Verify that the system allows you to undo the deletion and restore the member.



Expected Results:

- The system should offer an option to undo the deletion of a member within 24 hours.
- The process of recovering a deleted member should be straightforward and effective.
- Users should be able to recover a deleted member without significant delay or obstacles.
- Users should be navigated to the Member Details screen after Retrieving a soft deleted member.

Actual Results:

- The system offers an option to undo the deletion of a member within 24 hours.
- The process of recovering a deleted member is straightforward and effective.
- Users can recover a deleted member without significant delay or obstacles.
- Users are navigated to the Member Details screen after Retrieving a soft deleted member.

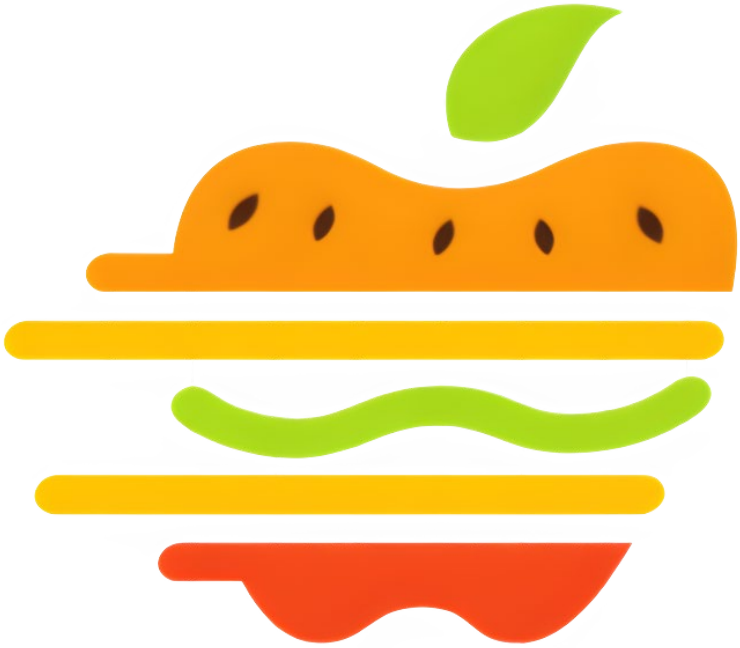
Conclusion:

The quality review for the defined metrics has yielded valuable insights into the efficiency of our Member Management Web Pages and our system's responsiveness to user actions.

For **Metric 1**, assessing the efficiency of adding a member, the threshold value was set at an average time of less than 2 minutes. Our actual value, as determined after quality assurance, demonstrates that experienced users can add a new member in a remarkably swift 30 seconds, while new users require approximately 1 minute. This result aligns well with our target, emphasizing the user-friendliness and efficiency of our system for this critical task.

Metric 2, which focuses on system recovery in case of member deletion by mistake, sets a threshold value for a "soft delete" feature, allowing user retrieval of deleted members within 24 hours without IT support intervention. Our actual value confirms that our system effectively provides this functionality, permitting users to retrieve deleted members within the specified 24-hour timeframe. This responsive and user-centric approach ensures that users can rectify accidental deletions without undue delay or inconvenience.

In summary, our software's performance in these two key metrics highlights its effectiveness and user-centric design. The efficient member addition process and responsive member recovery feature demonstrate our commitment to providing a streamlined and user-friendly experience for our users, ultimately contributing to the overall quality of our product.



NUTRISWAP

Software Project Planning, Design, and Quality Management

Prepared By: Marella Morad

Student ID: 103076428

Tutor Name: Harsharan Kaur

Tutorial Class: Monday 2:30 Ba708

Table of Contents

Table of Figures

1 Introduction:

In this report, we embark on a journey into the realms of software project management, design, and quality assurance, all while exploring the exciting potential of a project idea that interests me, I have called it NutriSwap. The NutriSwap project is aimed at addressing a pressing concern in today's world – the need for healthier dietary choices and improved access to nutritional information. This report will delve into the project's business needs, usage scenarios, product backlog items, schedule planning, architectural considerations, and quality management strategies. It is through this comprehensive analysis and planning that we will lay the foundation for a successful software project.

2 Background:

In today's fast-paced world, the importance of making healthier dietary choices is more evident than ever. NutriSwap comes as a response to the growing need for a solution that simplifies the process of finding healthier food alternatives and gaining insights into one's nutritional choices. Many health apps lack accurate information about home-cooked food, making this an opportunity to provide a valuable solution. This project is designed to offer a convenient and user-friendly solution to empower individuals in their journey toward healthier eating.

3 Scope:

3.1 Objective:

The objective of the NutriSwap project is to develop a user-friendly, cross-platform web application that provides valuable resources and suggestions for making healthier food choices. The project aims to enable users to access accurate nutritional information for various food items and discover substitutes with lower calories and healthier ingredients. By doing so, NutriSwap intends to foster a healthier lifestyle and create a more informed food consumption culture.

3.2 In Scope:

3.2.1 Nutritional Information:

The application will offer users access to comprehensive and accurate nutritional data for a wide range of food items, including macronutrients, micronutrients, and caloric content.

3.2.2 Suggest Healthier Alternatives:

NutriSwap will provide users with personalized, health-conscious substitutes for the foods they search for in the app. These recommendations consider individual preferences and dietary restrictions outlined in the user's profile. These results also align with the user's specific objectives for using the app, such as cutting calories, reducing sugar, or adhering to specific diets like gluten-free or vegan, etc…

3.2.3 Locates Healthier Alternatives:

NutriSwap will also provide users with links to get the suggested alternatives from local stores or an e-commerce platform. The local store search will be based on the address information provided by the user (stored in the user profile).

3.2.4 Recipe Transformation:

Users will be able to input their favourite recipes, either by typing it in, or by entering the URL to an online recipe (in any language), and the application will provide alternatives to make the same dish with healthier ingredients.

3.2.5 Personalised Recommendations:

Leveraging AI and machine learning algorithms, NutriSwap will learn from user preferences and offer personalised dietary recommendations over time**.** This feature will only be enabled if the user agrees to it as it will be using the user data for this analytic purpose.

3.2.6 User-Friendly Interface:

One of NutriSwap's main goals is to simplify the process of finding healthy food for everyone. Therefore, it needs an intuitive GUI for seamless user interaction and data entry.

3.2.7 User Registration and Profile Creation:

Users need to create accounts, allowing the app to tailor recommendations based on their information, like address, dietary needs, and health goals.

3.3 Out of Scope:

3.3.1 Healthy Food Education:

NutriSwap won't have a dedicated education section about healthy eating.

3.3.2 Detailed Mapping and Navigation Integration:

The app won't provide directions to stores, just store names or online store links.

3.3.3 Payment Processing:

The application will not handle payment processing, as its focus is on providing nutritional information and food alternatives not purchasing these alternatives.

3.3.4 Third-party Integrations:

The app won't connect to external systems beyond nutrition and food suggestions.

4 Deliverables and Schedule:

4.1 Deliverables:

- **Source Code:** A complete source code of the application for potential future modifications and enhancements.
- **Running Application:** A fully functional software web application, compatible with all platforms.
- **User Manual:** A comprehensive user manual that guides users on how to effectively use the application.
- **Training Materials:** Basic training videos to help users become familiar with the application's features and functions.

4.2 Product Backlog Items & Business Values Justification:

The following is a list of product backlog items thoughtfully allocated across two consecutive sprints: each spanning two weeks or 10 working days. Our team is composed of six dedicated professionals: a Product Owner, a Scrum Master, Three Developers, and a Tester. Adhering to the given constraint of dedicating 8 hours per team member per week, we have meticulously designed these sprints to accommodate a total of 96 hours. The subsequent backlog items, each with their respective importance explained, have been planned in consideration of the available resources and the expected timeframe.

Note: In square brackets, you will find a reference to the listed In Scope features that each task corresponds to.

4.2.1 Sprint 1: Core Functionality and User Interaction

No.	Item Name	BV	Justification	Reference
1	Create User Database Table	8	To store user details and preferences for personalized recommendations	3.2.1
2	Develop User Registration Forms and Validation	8	Develop sign-up forms for new users to join securely	3.2.7
3	Create User Profile Pages	7	Enable users to input dietary needs and health goals	3.2.3
4	Implement User Authentication and Login	8	Ensure secure logins for a personalized experience	3.2.4
5	Develop User Settings	8	Let users control their recommendations	3.2.5
6	Develop a User Dashboard	8	Create a user-friendly interface to manage profiles	3.2.6
7	Plan User Registration and Profile Management Test Cases	7	Plan test cases to test the functionality of user registration and profile management	3.2.7
8	Execute User Registration and Profile Management Unit Testing	8	Verify the correctness of user profile features by internally testing the functionalities	3.2.7

9	Develop Location-Based Search	8	Improve user experience with location-based searches	3.2.2
10	Display Search Results with Links to Stores	7	Show healthier food options with store links	3.2.3
11	Create Recipe Database Table	8	Important to save the imported recipes so that the user can retrieve them from their account whenever they need them	3.2.4
12	Develop Recipe Import Page	7	Simplify recipe entry by importing the recipe from an online source (Website)	3.2.4
13	Develop the Manual Recipe Entry Page	7	Allow users to input recipes directly (if it's a personal recipe for example)	3.2.4
14	Implement Recipe Transformation Algorithm	9	Implement an algorithm to analyse recipes, identify ingredients, and propose healthier alternatives	3.2.4
15	Display Transformed Recipes	9	Display transformed recipes with detailed nutritional information for each ingredient	3.2.4

4.2.2 Sprint 2: Advanced Functionality and Quality Assurance

No.	Item Name	BV	Justification	Reference
1	Integrate a translation tool for recipes in different languages	7	Adds multilingual support to translate recipes into different languages to expand the app's reach and enable the user to have healthier authentic food	3.2.4
2	Plan Location-Based Search and Recipe Transformation Test Cases	7	Create tests for location-based search and recipe transformation	3.2.3, 3.2.4
3	Execute Location-Based Search and Recipe Transformation Unit Testing	7	Check the accuracy and effectiveness of these core features	3.2.3, 3.2.4
4	Design AI and Machine Learning Architecture for Personalized Recommendations	8	Design architecture for personalized dietary recommendations (turned off by default – for privacy reasons, the user can choose to turn it on)	3.2.5
5	Develop a Personalized Recommendation Engine	8	Build personalized dietary suggestions for user engagement	3.2.5
6	Develop a Page to display the privacy statement	7	Provide transparency on data usage and address privacy concerns	3.2.6
7	Plan Personalized Recommendations and User Privacy Settings Test Cases	7	Validate personalized recommendations and privacy settings	3.2.5, 3.2.6
8	Execute Personalized Recommendations and	8	Ensure the security and effectiveness of advanced features	3.2.5, 3.2.6

	User Privacy Settings Unit Testing			
9	Finalize Internal Application Testing and Bug Fixing	9	Thoroughly test for critical bugs to ensure reliability	3.2.1 - 3.2.7
10	Run Usability Testing	8	Enhance user experience through user-friendly design and functionality	3.2.1 - 3.2.7
11	Run User Acceptance Testing	8	Confirm the app meets user expectations and requirements	3.2.1 - 3.2.7
12	Another round of Bug Fixes	8	Address issues raised by the Usability and User Acceptance testing	3.2.1 - 3.2.7
13	Create a User Guide and Documentation	7	Provide resources for effective app usage and issue troubleshooting	3.2.1 - 3.2.7
14	Develop Video Tutorials for using NutriSwap	7	Offer visual guidance to enhance app usability	3.2.1 - 3.2.7
15	Prepare the application for release and deployment	8	Get the app ready for user access	3.2.1 - 3.2.7

5 Solution Direction and High-Level Design:

5.1 Web-Based Application

The decision to choose a web-based application is so that NutriSwap is accessible through web browsers on different devices, offering a more versatile and platform-independent solution. The web application still provides users with a user-friendly interface to access nutritional data, discover healthier food options, and manage their profiles. The server-side components handle data processing, recommendation algorithms, and data synchronization for a seamless user experience, while external APIs and databases provide accurate, up-to-date information. The AI and machine learning engine continues to deliver personalized food recommendations in alignment with the project's goals.

5.1.1 High-Level Architecture Diagram:

In this solution direction, NutriSwap will be developed as a web-based application, accessible through web browsers on various platforms. Below is the high-level architecture diagram for the web application:

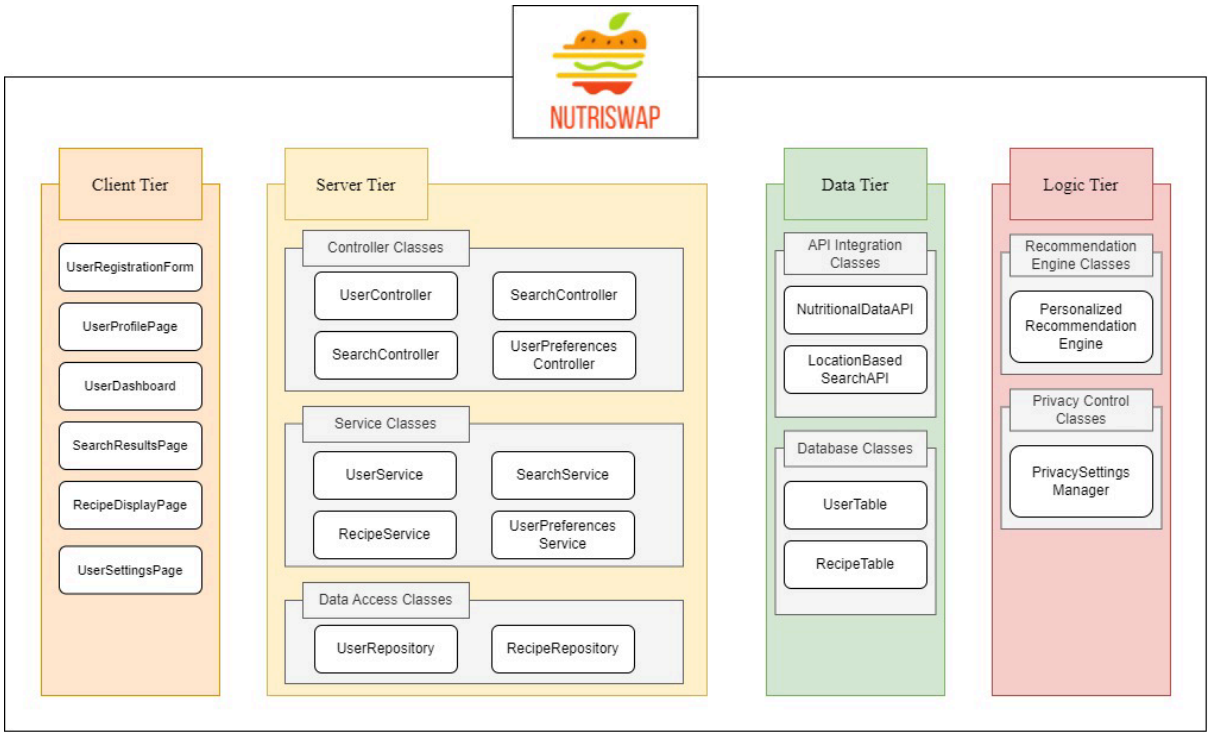


Figure 5.1 High-Level Design of NutriSwap

5.1.2 Architecture Components:

5.1.2.1 Web Application (Client Tier):

Role: The web application is the client-side interface that users access via web browsers. It's responsible for rendering the user interface, collecting user inputs, and displaying search results, recipes, and recommendations.

Responsibilities: The web app communicates with the server-side components to request nutritional data, search for healthier alternatives, and manage user profiles. It integrates features such as location-based search, recipe transformation, and multilingual support.

5.1.2.2 Web Server (Server Tier):

Role: The web server serves as the server side of the web application, handling data processing, storage, and business logic. It connects to external APIs and databases to provide accurate nutritional data and alternatives to the web app.

Responsibilities: The web server manages user data, including profiles, preferences, and health goals. It integrates with external sources to fetch nutritional information and provides real-time suggestions based on user inputs. It supports location-based search and recipe transformation algorithms.

5.1.2.3 External APIs and Databases (Data Tier):

Role: External APIs and databases are responsible for supplying nutritional data, recipes, and alternative food choices to the web server. These external sources offer real-time, accurate data for NutriSwap.

Responsibilities: External APIs provide access to comprehensive nutritional data for various food items, while databases store user profiles and their saved recipes for easy retrieval. The web server accesses this data to provide users with accurate information.

5.1.2.4 AI and Machine Learning Engine (Logic Tier):

Role: The AI and machine learning engine, part of the web server, is responsible for personalized dietary recommendations. It analyses user preferences and dietary restrictions to offer tailored food suggestions over time.

Responsibilities: The engine continuously learns from user behaviour and preferences to improve the quality of recommendations. It respects user privacy settings and only operates when explicitly enabled by the user.

5.2 Version Control and Issue Tracking

NutriSwap's development process will be managed through GitHub and Jira to facilitate efficient and organized software development.

5.2.1 GitHub:

GitHub will be used for version control, allowing the development team to collaboratively work on code, track changes, and manage different versions of the application. GitHub simplifies code merging and provides a repository for the entire project.

5.2.2 Jira Integration:

To streamline development efforts and enhance project management, NutriSwap's GitHub repository will be integrated with Jira. This integration will provide several advantages:

- **Issue Progress Synchronization:** Jira issues will be synchronized with the progress of corresponding code-related activities in GitHub. For instance, when a pull request is merged, the related issue in Jira will be moved to the "Done" status.
- **Issue Assignment:** When a task is moved to "In Progress" in Jira, a branch will be automatically created in GitHub, helping developers stay organized.
- **Automated Workflows:** The integration between Jira and GitHub will establish automated workflows, reducing manual overhead and providing a more accurate reflection of the project's status.
- **Efficient Collaboration:** Developers, testers, and other team members can easily collaborate within Jira, ensuring that everyone is aligned with the project's objectives and progress.

5.3 Design Justification using Design Principles

5.3.1 Object-Oriented Principles:

At NutriSwap, we place a strong emphasis on object-oriented principles to ensure that our codebase is not only well-structured but also easily maintainable and extensible. We've chosen object-oriented languages and frameworks such as React, and C# to help us better comply with these principles. Our adherence to these principles is reflected in the following aspects:

5.3.1.1 Strong Cohesion and Weak Coupling

Our software design is built upon the foundation of strong cohesion and weak coupling. We have meticulously organized our code into separate modules, each dedicated to specific functionalities. This division enhances code clarity, reduces complexity, and fosters maintainability. For instance, rather than creating a monolithic component for the entire application, we've wisely divided it into smaller, cohesive components, each responsible for distinct tasks. This approach enables individual components to interact with one another with minimal dependencies, achieving weak coupling. The benefit is that these components can function effectively without requiring an intricate understanding of the internal workings of other components. This modularity empowers us to make changes or introduce new features without causing ripple effects throughout the codebase.

5.3.1.2 Separation of Concerns (SoC)

We've adhered to the separation of concerns principle as well, ensuring that different aspects of the application are treated as separate concerns. This helps us maintain code clarity and makes it easier to manage the different layers and functionalities within NutriSwap. For example, user registration and profile management are distinct concerns separated from core features like search functionality and recipe transformation. This separation of concerns keeps our codebase organized and promotes maintainability.

5.3.1.3 Encapsulation

Encapsulation is another fundamental aspect of our design. Each component in NutriSwap encapsulates its behaviour and rendering logic, exposing only necessary interfaces to the outside world. This approach safeguards data integrity and ensures that changes to one part of the system do not unintentionally impact other parts.

5.3.1.4 Information Hiding

NutriSwap is designed with information hiding in mind, concealing internal details and data within modules. This minimizes complexity, enhances security, and prevents unintended data modifications.

5.3.1.5 Scalability and Performance Considerations

The architecture of NutriSwap has been carefully crafted to accommodate scalability and optimize performance. We employ techniques such as load balancing, caching, and asynchronous processing to ensure that the application responds to user interactions within an acceptable response time and can handle anticipated user loads without significant performance degradation.

5.3.2 Designing the Logo and UI

At NutriSwap, we've given careful thought to the visual identity of our web application. We recognize the significance of creating a distinct and memorable user experience, and the design of our logo and user interface (UI) plays a pivotal role in achieving this objective.

5.3.2.1 Logo Design

We with crafting a logo that embodies the essence of warmth and professionalism, promoting healthier choices and symbolising NutriSwap's name. After undergoing several design iterations as you can see below:

Figure 5.2 Initial Logo Design Ideas

However, there was still something missing from the logo. That is, the logo didn't depict the idea of "Swapping" junk food to healthy food. Which we managed to achieve in the chosen logo that shows a Hamburger being transformed into an apple.

Figure 5.3 Chosen Logo

5.3.2.2 Colour Palette

In addition to the logo, we spent considerable time choosing a colour palette for our web application. Ensuring that our colours reflect bright colours (symbolising vegetables, fruits and healthy food in general) as well as ensuring we have a range of contrasting colours to comply with accessibility standards.

Figure 5.4 NutriSwap's Colour Palette

5.3.3 Naming Conventions:

5.3.3.1 Component and Class Names:

When it comes to naming components and classes in NutriSwap, we adhere to a structured approach that emphasizes clarity and consistency. We have adopted the PascalCase naming convention, which means that names are written with the first letter of each word capitalized. For example, "UserRegistrationForm," "SearchResultsPage," and "UserService." This naming convention helps developers quickly identify and understand the purpose and functionality of each component, ensuring our codebase is well-organized and comprehensible.

5.3.3.2 Variable and Function Names:

In the context of variable and function names, we follow the camelCase naming convention, which is an established industry standard. This convention is known for its readability and

ease of use. With camelCase, names are written with an initial lowercase letter, and subsequent words within the name are capitalised.

5.4 Use Case Diagram

Figure 5.5 NutriSwap's Use Case Diagram illustrates the core functionality and interactions within the NutriSwap web application, with the "User" as the primary actor. Users initiate various use cases, reflecting the application's objectives to promote healthier eating and informed food choices. The diagram begins with the "Register" use case, where users create accounts to personalize their experience. The "Login" use case, included in the "Register" process, grants access to the application. Subsequently, the "Search for Healthier Alternatives" use case is activated, allowing users to discover and view alternative food choices with improved nutritional profiles after they have signed in to the application. This process may extend to "View Detailed Nutritional Information," offering in-depth insights.

In parallel, the "Import Recipe" use case emerges, enabling users to input their favourite recipes either by importing from a URL or manually typing in the recipe. Following this, users may opt for "Receive Recipe Transformation Recommendations," further aligning with their health goals. Additionally, users can "Translate Recipe" if they wish to make their favourite dishes healthier. Furthermore, there is an "Include" relationship between "Search for Healthier Alternatives" and "Locate Healthier Alternatives at Local Store" to facilitate finding local stores for recommended alternatives.

This Use Case Diagram encapsulates the core functions of NutriSwap, emphasizing user account management, food alternative discovery, and recipe transformation, all contributing to the project's mission of promoting healthier eating and informed food choices.

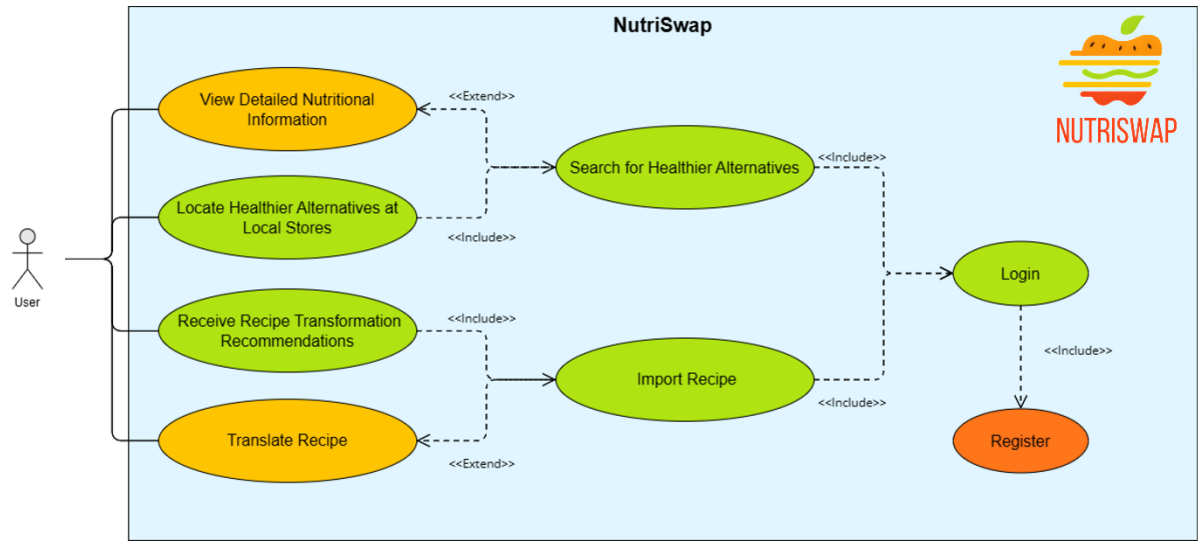


Figure 5.5 NutriSwap's Use Case Diagram

6 Definition of Done and Quality:

6.1 Definition of Done:

The “Definition of Done” (DoD) for NutriSwap, in alignment with the ISO 25010 Software Quality Model and project-specific goals, comprises a comprehensive set of criteria. It serves as a vital framework to determine the completion of features or user stories, safeguarding the software's overall quality and its ability to fulfil its intended purposes. This DoD encompasses both functional and non-functional (quality) requirements, guaranteeing the NutriSwap web application's readiness for successful delivery and release (Standards Australia/New Zealand, 2013).

6.1.1 Functional Suitability (ISO 25010 – 4.2.1)

- All specified user stories have been implemented and tested successfully.
- All core features, including user registration, profile management, search functionality, recipe transformation, and personalized recommendations, are fully functional and meet the defined acceptance criteria.

6.1.2 Performance Efficiency (ISO 25010 – 4.2.2):

- The application responds to user interactions within an acceptable response time.
- The application can handle the expected user load without significant performance degradation.

6.1.3 Compatibility (ISO 25010 – 4.2.3):

- The web application is compatible with commonly used web browsers (Chrome, Firefox, Safari, Edge).
- It provides a responsive design that adapts to various screen sizes and resolutions, including mobile devices and desktops.

6.1.4 Usability (ISO 25010 – 4.2.4):

- The user interface is intuitive and user-friendly, allowing users to easily navigate, search for food items, and view recommendations.
- User feedback is positive, indicating high usability and user satisfaction.

6.1.5 Reliability (ISO 25010 – 4.2.5):

- The application is stable and does not crash or produce errors during normal usage.
- Data integrity is maintained, ensuring that user profiles, preferences, and dietary information are accurately stored and retrieved.

6.1.6 Security (ISO 25010 – 4.2.6):

- User data is protected through secure authentication and authorization mechanisms.
- Personalized recommendations respect user privacy settings and only operate when explicitly enabled by the user.

6.1.7 Maintainability (ISO 25010 – 4.2.7):

- The codebase is well-documented, organized, and follows coding standards.

- Updates and changes can be made with relative ease to accommodate future feature enhancements and bug fixes.

6.1.8 Portability (ISO 25010 – 4.2.8):

- The web application can be deployed on various web hosting platforms.
- It is cross-browser compatible and works on different devices and operating systems (Windows, Android, IOS, etc…).

6.2 SMART Goals:

6.2.1 S: Specific

- Improve the user interface for mobile devices to enhance accessibility.

6.2.2 M: Measurable

- Achieve a user satisfaction rating of at least 4 out of 5 based on user feedback and reviews.
- Reduce the average response time for user interactions to less than 1 second.

6.2.3 A: Achievable

- Develop and implement a personalized recommendation system based on machine learning to improve user engagement.
- Ensure that the web application is compatible with the latest versions of Chrome, Firefox, Safari, and Microsoft Edge.

6.2.4 R: Relevant

- Align the application's dietary recommendations with the Australian Dietary Guidelines to promote healthier food choices.
- Enhance security by implementing two-factor authentication to protect user data.

6.2.5 T: Time-bound

- Complete the integration of a translation tool for recipes in various languages within the next three months.
- Roll out regular updates and feature enhancements every two months to ensure the application remains competitive and user-friendly.

6.3 Quality Model:

6.3.1 Functional Suitability:

6.3.1.1 Functional Completeness:

- Percentage of Completed Features
  - ≥ 95% of planned features are implemented and operational.

6.3.1.2 Functional Correctness:

- Number of Critical Bugs
  - ≤ 1 Critical Bug in the application.

6.3.1.3 Functional Appropriateness:

- Compliance with Dietary Guidelines
  - All dietary recommendations align with the Australian Dietary Guidelines (National Health and Medical Research Council 2013)

6.3.2 Performance Efficiency:

6.3.2.1 Time Behaviour:

- Average Response Time for User Interactions
  - ≤ 1 second

6.3.2.2 Resource Utilization:

- The percentage of Application Resource Usage
  - < 80% of system resources

6.3.2.3 Capacity:

- The number of concurrent users the application supports without a significant drop in performance:
  - 100 Concurrent Users

6.3.3 Compatibility:

6.3.3.1 Co-existence:

- Browsers on which NutriSwap functions without critical errors
  - The latest versions of Chrome, Firefox, Safari and Microsoft Edge

6.3.4 Usability:

6.3.4.1 Appropriateness Recognizability:

- The percentage of recognizable key features by users without guidance
  - ≥ 85%

6.3.4.2 Learnability:

- Amount of training required for non-technical and technical users to be able to learn basic functions respectively:
  - < 1 hour and < 30 minutes respectively

6.3.4.3 Operability:

- The success rate of users completing key tasks (such as searching for a healthier alternative)
    - ≥ 90%

6.3.4.4 User Error Protection:

- Error Handling
    - Users receive clear error messages for incorrect inputs, reducing user errors by ≥ 90%.

6.3.4.5 Accessibility:

- Compliance with WCAG (Web Content Accessibility Guidelines) Standards
    - NutriSwap complies with WCAG 2.1 (latest) accessibility standards for users with disabilities.

6.3.5 Reliability:

6.3.5.1 Maturity:

- The continuous period the application operates without a critical failure.
    - ≥ 30 days

6.3.5.2 Availability:

- Percentage of uptime the system is required to have.
    - ≥ 99% (excluding planned maintenance windows)

6.3.5.3 Fault Tolerance:

- Error Recovery
    - The application can recover from non-critical errors without data loss.

6.3.5.4 Recoverability:

- Data Recovery of Accidental Deletion
    - 24 Hours from deletion (for example, if the user accidentally deletes one of their imported recipes, the application should offer an option to retrieve it within 24 hours.

6.3.6 Security:

6.3.6.1 Confidentiality:

- All Personally Identifiable Information (PII) will be inaccessible until an authorized user has been authenticated.
- Sensitive User data is encrypted using AES-256 an industry-standard encryption algorithm.

6.3.6.2 Integrity:

- The number of vulnerabilities found in the software through security assessment.
    - 0

6.3.6.3 Non-repudiation:

- User Actions Tracking
  - The system tracks user actions to prevent the denial of user actions.

6.3.6.4 Accountability:

- User Activity Logging
  - User activities are logged for accountability and audit purposes.

6.3.6.5 Authenticity:

- User Authentication
  - All users will require authentication before being allowed access to the system using a 2-factor authentication method (e.g., Google Authenticator or Text Message Authentication).

6.3.7 Maintainability:

6.3.7.1 Reusability:

- Code Reuse
  - Code components are designed for reuse in future features or modules.

6.3.7.2 Analysability:

- Code Documentation
  - The codebase is well-documented, allowing for easy analysis and understanding.

6.3.7.3 Modifiability:

- Code Change Complexity
  - The average complexity of code changes should be below a value of 10 in terms of the Cyclomatic Complexity metric.

6.3.7.4 Testability:

- The percentage of code paths tested effectively.
  - $\geq 90\%$

6.3.8 Portability:

6.3.8.1 Adaptability:

- Cross-platform Compatibility
  - The web application functions on different platforms (e.g., Windows, macOS, Linux).

6.3.8.2 Replaceability:

- Third-party Integration
  - The application can integrate with third-party services for future enhancements or extensions.

7 Sprint 1 Detailed Plan:

In the development of our Sprint 1 Detailed Plan, we employed a variety of estimation techniques to ensure a well-rounded and accurate assessment of the tasks at hand. These techniques included Estimation by Analogy, Estimation by Size Comparison, and the "Experts" or "Delphi" Techniques. By utilizing different methods tailored to the specific nature of each task, we aimed to provide a comprehensive and reliable plan for Sprint 1. This approach allowed us to leverage the collective expertise of our team members, make informed comparisons to past efforts, and ensure that all aspects of task complexity were considered.

7.1.1 Estimation by Analogy

Working on the GotoGro project provided me with valuable experience, especially in the realm of estimating story points. For Task 2, I decided to allocate 8 story points based on a similar task we worked on for GotoGro. This estimation also takes into account the task's complexity, considering that it involves setting up the first web page of the entire application and configuring frameworks like React, which can be time-consuming.

Likewise, Task 3 is allocated 4 story points, primarily because the initialization of the web application was completed as part of Task 2. Using a similar technique, I assigned 4 story points to tasks 5, 10, 13, and 15, as they share commonalities. Task 6, on the other hand, received a higher estimate due to its broader scope, encompassing the viewing and editing of profile information.

No.	Item	Estimation
2	Develop User Registration Forms and Validation	8
3	Create User Profile Page, including input fields for dietary needs and health goals	4
5	Develop User Settings Page	4
6	Develop a Basic User Dashboard to view and edit profile information	5
10	Display Search Results with Links to Stores	4
13	Develop a Manual Recipe Entry Page	4
15	Display Transformed Recipes with Nutritional Information	4

In addition to my development experience, I've also invested time in testing GotoGro and gained professional testing experience at work. This exposure has given me insights into the time required for test planning, which is why I estimated 3 story points for Task 7. Likewise, I've allocated 3 story points for executing unit tests and usability tests, leveraging my understanding of the testing process and its associated effort.

No.	Item	Estimation
7	Plan and Design Test Cases for User Registration and Profile Management	3

8	Execute Unit Testing and Validate User Profile Functionality	3

7.1.2 Estimation by Size Comparison

Task 1, which involves creating the user database, is relatively small in size and complexity compared to other tasks. However, it includes setting up the database as an internal step. Drawing from my experience working on GotoGro, where we used Supabase for our backend, I found that setting up and creating tables was straightforward. Nonetheless, considering the expected user base of NutriSwap, it's clear that Supabase might not be the most suitable solution. Therefore, we've chosen to adopt a more traditional approach by creating a backend with APIs that communicate with both the frontend and the database.

Once the initial database schema is completed, Task 11, which involves creating the recipe database table, is a relatively straightforward task, and I've allocated it 1 story point to reflect its simplicity and low complexity.

No.	Item	Estimation
1	Create User Database Table	5
11	Create Recipe Database Table	1

Task 9 is estimated to be of higher complexity compared to Task 7 primarily because it involves all three layers of the application working in concert to achieve its objectives. In Task 9, the frontend initiates a search request, the backend retrieves the user's address details from the database, and then conducts the search based on the user's address. This interplay between the frontend, backend, and database adds to the complexity, and I've allocated 6 story points to reflect this.

Similarly, Task 12 is estimated to be of similar complexity, receiving 6 story points. This task involves the installation and integration of third-party libraries to enable the import functionality. This process can be intricate and, therefore, justifies a higher point estimate.

Finally, Task 14 is also given 6 story points. While it partially relies on the search algorithm from Task 9, the difference lies in the search being conducted on all recipe ingredients. This expanded scope increases the complexity, resulting in a higher point estimate.

No.	Item	Estimation
9	Develop Location-Based Search Functionality	8
12	Develop Recipe Import Page	6
14	Implement Recipe Transformation Algorithm	4

7.1.3 Estimation by "Experts" / "Delphi" Techniques:

I decided to use the "Experts" / "Delphi" technique for tasks that I have limited exposure to or tasks that are complex and need expert advice. Task 4 is a prime example of such a task, which is relatively new to me. However, upon research, it became evident that Task 4 is too broad (Ranwa, 2020). As a result, it was recommended to break this task down into smaller,

more manageable sub-tasks, each with its respective estimation. This approach allows for a more accurate estimation and better planning of the work involved.

No.	Item	Estimation
4	Implement User Authentication and Login Functionality • Develop server-side API for registration, login, and logout. • Write C# server-side code for database interaction and authentication. • Create React-based client-side login form for API communication. • Securely store user credentials, including sensitive data encryption.	10 3 3 2 2

7.2 Overall Sprint Tasks and Estimations:

No.	Item	Dep	Estimation
1	Create User Database Table		5
2	Develop User Registration Forms and Validation	1	8
3	Create User Profile Page, including input fields for dietary needs and health goals	2	4
4	Implement User Authentication and Login Functionality • Develop server-side API for registration, login, and logout. • Write C# server-side code for database interaction and authentication. • Create React-based client-side login form for API communication. • Securely store user credentials, including sensitive data encryption.	2	10 3 3 2 2
5	Develop User Settings Page	3	4
6	Develop a Basic User Dashboard to view and edit profile information	4, 5	5
7	Plan and Design Test Cases for User Registration and Profile Management		3
8	Execute Unit Testing and Validate User Profile Functionality	7	3
9	Develop Location-Based Search Functionality		8
10	Display Search Results with Links to Stores	9	4
11	Create Recipe Database Table		1
12	Develop Recipe Import Page	11	6
13	Develop a Manual Recipe Entry Page	11	4
14	Implement Recipe Transformation Algorithm	12, 13	4
15	Display Transformed Recipes with Nutritional Information	14	4

8 References:

National Health and Medical Research Council 2013, *Australian Dietary Guidelines*, viewed <https://www.eatforhealth.gov.au/sites/default/files/2022-09/n55_australian_dietary_guidelines.pdf>.

Sanwar Ranwa 2020, 'Create User Registration and Login Using Web API and ReactJS', *dzone.com*, DZone, viewed 17 October 2023, <https://dzone.com/articles/create-user-registration-and-login-using-web-api-a>.

Standards Australia/New Zealand, 2013, Systems and software engineering - Systems and software quality requirements and evaluation (SQuaRE) - System and software quality models, AS/NZS ISO/IEC 25010:2013, Sydney/Wellington: Standards Australia/New Zealand, viewed 18 August 2023.

SWE20001 – Managing Software Projects

Learning Summary Report

Marella Morad (103076428)

Portfolio Submission Due

All Grades: Week 14 Mon (6th Nov 2023), 9:00 am

Portfolio Interview Dates

Distinction / High Distinction: Week 14 Tue – Fri (7th – 10th Nov 2023), (15 minutes per student, book your interview time on Canvas)

Self-Assessment Details

The following checklists provide an overview of my self-assessment for this unit.

	Pass (P)	Credit (C)	Distinction (D)	High Distinction (HD)
Self-Assessment (please tick)				✓

Self-assessment Statement

	Included (please tick)
Learning Summary Report	✓
All Pass Tasks are Completed on Canvas	✓

Minimum Pass Checklist

	Included (please tick)
All Credit Tasks are Completed on Canvas	✓

Minimum Credit Checklist, in addition to the Pass Checklist

	Included (please tick)
Interview booked	✓
All Distinction Tasks are Completed on Canvas	✓
Other pieces (please specify)	

Minimum Distinction Checklist, in addition to Credit Checklist

	Included (please tick)
81 HD Software Project Document [Plan, Design, QA] meet HD criteria and standards	✓
82 HD Research Article / Essay meets HD criteria and standards	
Other pieces (please specify)	

Minimum High Distinction Checklist, in addition to Distinction Checklist

Declaration

I declare that this portfolio is my individual work. I have not copied from any other student's work or from any other source except where due acknowledgment is made explicitly in the text, nor has any part of this submission been written for me by another person.

Signature: Marella Morad

Portfolio Overview

This portfolio includes work that demonstrates that I have achieved all Unit Learning Outcomes for SWE20001 Managing Software Projects to an **HD** level.

In this portfolio, I will demonstrate my achievement of all Unit Learning Outcomes for SWE20001 - Managing Software Projects, exceeding expectations and achieving a High Distinction level. Throughout this journey, I have committed myself to mastering the principles and practices of software project management, emphasising precision, understanding, and excellence in each task and assignment.

ULO1: Select an appropriate development methodology with justification to develop a software project and plan the project for implementation.

Throughout the unit, I have gained the knowledge and skills necessary to select an appropriate development methodology and plan a software project. The following are examples to support my claim:

Tasks 01P - Project Proposal and 02P - Scope and Product Backlog

- This was the first opportunity we had to plan the chosen project, GotoGro. I successfully created a project proposal outlining the background, scope, and initial schedule of the product backlog items (Task 01P). After doing so, I worked with my teammates to develop a more detailed plan, incorporating ideas from team members with diverse backgrounds and various skill sets and experiences (02P). Although this concept was new to me, it was a great learning experience to understand different development methodologies and how to select the most suitable one for the given situation. Additionally, the diverse experiences of my teammates helped me gain more insights into these various methodologies.

Task 81HD - Software Project Planning, Design, and Quality Management

- After gaining experience with tasks 01P through 09P, I gained more confidence in creating a comprehensive project plan, which I successfully applied to the NutriSwap application. The plan includes the selection of a web-based development methodology, which is justified by its versatility and accessibility. It also incorporates an architectural diagram for the project. These choices demonstrate my ability to select the appropriate methodology and create a well-thought-out plan for a software project. I took into account factors such as accessibility, user-friendliness, and data synchronization, aligning them with the project's goals.

ULO2: Apply tools and techniques to define scope, break down tasks, estimate effort, manage risks and schedule resources in the planning of a software development project.

Task 08P – Sprint Planning

- In this task, I've defined the project scope by identifying and detailing factors for prioritising backlog items, such as feature dependency, development effort, and business value, ensuring a clear understanding of what needs to be accomplished. Additionally, I've utilised the Work Breakdown Structure (WBS) method to systematically break down complex tasks into smaller, manageable components, which has enhanced my task understanding, resource allocation, task dependency identification, estimation, and progress tracking abilities. My awareness of the importance of accurate development effort estimation and risk management demonstrates a well-rounded approach to project planning, and my prioritisation of backlog items based on the project's unique needs showcases my ability to schedule resources effectively, ultimately contributing to the successful execution of my software development project.

Task 09P – Project Initiation

- I've effectively applied tools such as GitHub and JIRA boards to manage schedule resources in my software development project. These platforms have allowed me to create a structured backlog, decompose tasks, and assign them to team members, ensuring a clear understanding of project goals and resource allocation. While they primarily assist in task management, I've integrated them into my broader project management approach.

Task 16P – Sprint Planning

- In addition to the techniques, we used in Task 08P, for sprint 2, we employed different techniques, all grounded in the same prioritisation principles, including feature dependency, development effort, and business value. However, this time, we incorporated more comprehensive estimation techniques. One of these was Estimation by Analogy, which proved to be particularly helpful since we had many tasks in sprint 2 that were similar to those, we completed in sprint 1. Additionally, we used Estimating by Size Comparison to estimate relatively tasks based on their complexity and the effort required. Lastly, for tasks that were new to our team, we utilised the Estimating by "experts" / "Delphi" techniques, where we based the estimation on research and extensive team discussions.

Task 61C – Estimation Effort 1

- In this task, drawing upon my experience as a Frontend Developer and my academic background in web development, I have effectively applied tools and techniques to plan the creation of "Member Management Web Pages" for Sprint 1 of a web development project. Employing the Activity-based Work Breakdown Structure (WBS) method, I meticulously decomposed the project into distinct phases, such as Design, Development, Integration, Testing and QA, Documentation, Review, and Deployment. Through careful effort estimations and justifications for each task, I have quantified the time required for each activity, ensuring a realistic and well-structured schedule for resource allocation.

Task 63C – Estimation Effort 2

- In this task, I have effectively applied the "Estimating by Analogy" technique to estimate the effort required for developing the "Product Management Web Pages" in Sprint 2 of a web development project. I began by finding a similar task, which was the "Member Management Web Pages" task from Sprint 1, and then carefully analysed the similarities and differences between the two tasks in terms of functionality, user interface complexity, integration requirements, and unique challenges. By considering the similarities and differences. I justified this estimation by considering the benefits of leveraging existing architecture and templates from the previous task, as well as my experience in integrating images into web pages. This estimation process reflects a well-informed and reasoned effort estimation, ensuring efficient project planning and delivery.

Task 81HD - Software Project Planning, Design, and Quality Management

- For my HD task, I possessed all the required skills and knowledge that I had gained from previous tasks, as well as individual efforts for improvement, which allowed me to successfully achieve this unit's learning outcome. I applied various tools and techniques for project planning and management. In my Sprint 1 Detailed Plan, I employed estimation techniques like Estimation by Analogy, Estimation by Size Comparison, and the "Experts" or "Delphi" Techniques. I also meticulously broke down tasks, assigned story point estimations, and considered dependencies. This evidence demonstrates my ability to define project scope, divide tasks into manageable components, estimate efforts accurately, and consider potential risks.

ULO3: Select appropriate architecture styles, design patterns, algorithms, and data structures with justification; and apply tools and techniques to design, develop, and test the software solution.

Task 03P – Software Design

- In this report, I have successfully met ULO3 by carefully selecting a web-based, multi-layer architecture for Goto Grocery's member management system, supported by a well-justified rationale. I have performed a KoST analysis to identify the necessary knowledge, skills, and technology components, ensuring that I am well-prepared for the software development process. Moreover, I have provided a high-level design of the chosen architecture, clearly defining the roles and responsibilities of each layer, which demonstrates my ability to design and structure software solutions effectively. This comprehensive approach aligns with ULO3's requirements, showcasing my proficiency in selecting the right architectural styles, justifying decisions, and applying relevant tools and techniques in software development.

Task 04P – Solution Direction and Design

- In this report, I worked with my teammates to strategically select a cloud-hosted Model-View-Controller (MVC) architecture for Goto Grocery's system, a choice rooted in its compatibility with the organization's size and operational needs. Our well-substantiated rationale and high-level design for the MVC architecture illustrate my adeptness at making informed software design decisions and structuring solutions effectively, thus aligning with ULO3. Furthermore, the comprehensive KoST analysis reveals my understanding of the essential knowledge, skills, and technology components necessary for the software development process, ensuring that the solution is both practical and aligned with business requirements.

Task 14P – Software Design

- In this report, I have effectively addressed ULO3 by thoroughly designing the software components for the Goto Grocery system. The chosen architectural components, such as the Model Layer using Supabase for data management and the dual-purpose View/Controller Layer powered by React JS on the Google Cloud platform, align well with the project's needs and constraints, showcasing my ability to select and design appropriate software architecture. Moreover, the report's emphasis on design principles, including naming conventions, object-oriented principles, strong cohesion, and weak coupling, highlights my dedication to creating maintainable and scalable software solutions. The thoughtful design of the logo and colour palette also reflects my attention to user interface and visual consistency, demonstrating the application of design principles, all of which contribute to achieving ULO3's objectives effectively.

Task 72D – Quality Planning

- This report effectively meets ULO3 (Understand and apply software design principles and best practices) by providing a clear plan for ensuring the quality of two key metrics in the project. The report outlines a strategy for improving efficiency and system recovery. The inclusion of code review, unit testing, and usability testing demonstrates a well-rounded approach to quality assurance. The defined test cases and metrics enabled me to evaluate the software's performance, quality, and usability. The report also highlights the importance of adhering to coding standards and following best practices to maintain code quality and reliability.

Task 81HD - Software Project Planning, Design, and Quality Management

- I have decided to develop a web-based application for NutriSwap, as it ensures accessibility and versatility across various devices. The high-level architecture diagram I've presented provides a clear overview of how the application will function, giving me confidence in the project's direction. To maintain a structured and maintainable codebase, I've employed essential object-oriented principles, like strong cohesion and weak coupling, and made sure to separate different concerns within the application. This approach empowers me to handle future changes and feature

additions with ease. By integrating GitHub for version control and Jira for issue tracking, I've established an efficient workflow that enhances collaboration among my team members. The thoughtful design of the logo and colour palette reflects my commitment to providing users with an appealing and distinct experience. I've also created a Use Case Diagram that effectively outlines the application's core functionalities and interactions, emphasizing the user-centric approach I value. In summary, this report demonstrates my well-structured design for NutriSwap, aligning perfectly with the project's goals and the best practices in software development and design that I value.

ULO4: Utilise contemporary tools and techniques for software development projects including version control, testing and issue tracking, software artifacts and documentation, and track and report project progress.

Task 09P – Project Initiation

- To meet ULO4 for project initiation, our team utilised contemporary tools and techniques. We established a GitHub repository for version control, which facilitated collaboration and code management throughout the project. Additionally, we created essential project management artifacts, including a Sprint Burndown Chart and a Task Board through JIRA, which helped us track and report project progress, manage tasks, and ensure efficient issue tracking. These tools and practices enabled us to maintain organised and efficient software development throughout the project's lifecycle.

Tasks 10P – MidWeek Check

- Despite initial challenges in adopting Jira during Sprint 1, we took proactive steps to ensure accurate progress tracking. We manually recreated the missing data on our task board and generated a burndown chart, supplementing the auto-generated one. This meticulous approach allowed us to reflect a true picture of our work. Additionally, our proactive setup of the project repository and integration with Google Services streamlined our development process. Leveraging Supabase as our database solution further optimized our development efforts. We maintained a collaborative spirit, even though not all contributions were immediately visible in the commit history.

Task 14P – Software Design

- In our project's software component design, we effectively met ULO4 by leveraging contemporary tools and techniques. Our choice of Supabase for the Model Layer streamlined data management and integrity enforcement. The React-powered View/Controller Layer allowed us to efficiently handle user interface and state management, eliminating the need for traditional, separate controllers. Additionally, we established a robust CI/CD pipeline using GitHub Actions and Jira, ensuring seamless development and accurate project tracking. We followed essential design principles, such as naming conventions and the Model-View-Controller pattern, which enhanced code readability and maintainability.

Task 73D – Quality Review and Reflection

- In this report, I've conducted a comprehensive quality review of our software, focusing on two critical metrics: the efficiency of adding a member and the system's recovery in case of member deletion by mistake. The report outlines the results of unit tests, and usability testing scenarios, and provides a thorough analysis of our software's performance in meeting these metrics. I found that our system allows users to add a new member efficiently, even surpassing the target with experienced users taking only 30 seconds and new users requiring about 1 minute. Additionally, the system's responsiveness to recover deleted members within 24 hours aligns with our quality measure, ensuring a user-friendly and prompt recovery process. Overall, this review confirms the effectiveness and user-centric design of our software, contributing to its quality and reliability.

Task 81HD - Software Project Planning, Design, and Quality Management

- In my HD project, I successfully addressed ULO4 by employing contemporary tools and practices from the very inception. My choice of version control through GitHub and integration with Jira streamlined project management and code collaboration, enhancing my ability to track progress efficiently. I diligently adhered to object-oriented principles, fostering code modularity and maintainability.

ULO5: Apply and use contemporary tools and techniques to work effectively as a member of a software development team, and to reflect upon group work experiences.

Tasks 10P & 18P – MidWeek Check and 11P & 19P - Final Check

- In retrospect, I successfully met ULO5 in the above-mentioned reports through my active engagement with the team. As a member of the software development team, I effectively collaborated with my peers, ensuring tasks were distributed and executed cohesively. Additionally, I participated in rigorous quality assurance and testing, demonstrating our commitment to maintaining software reliability. We encountered challenges along the way, but collectively, we addressed them and made informed decisions, such as postponing tasks when necessary. In hindsight, our dedication to documentation also contributed to our collaborative knowledge-sharing efforts. These experiences have undoubtedly enhanced my ability to work effectively in software development teams while reflecting on our group work dynamics. Having daily standups, even when not in person, was helpful as it was our meeting point to check on project progress, discuss any blockers or personal issues that would stop a teammate from doing their work, and provide handovers in case of the inability to complete tasks due to either inexperience or personal reasons.

Tasks 12P and 20P – Sprint Review

- Conducting these Sprint reviews was undeniably essential. I actively engaged with fellow team members, participated in goal discussions, and demonstrated proficiency in utilising contemporary tools and technologies, including Supabase and Selenium. Moreover, effectively handling stakeholder feedback and planning for future improvements highlighted my collaborative skills and my capacity to reflect on our group work experiences. In summary, these interactions and contributions have significantly enhanced my ability to work effectively in a software development team, all while adapting to evolving tools and techniques.

Task 13P and 21P – Sprint Retrospective

- Reflecting on our project and the associated meeting minutes, I can affirm that ULO5 was met effectively. Our comparison of the ideal and actual velocity demonstrated our ability to align our efforts with the project's planned tasks and timelines. While we encountered a setback towards the project's end, the thorough reflection showcased our commitment to navigating challenges and making continuous strides. The examination of estimation accuracy revealed a learning experience, emphasising the importance of well-defined scopes and "definitions of done." To gain a better understanding of task complexity and improve time estimates, we adopted proactive measures like backlog refinement and the utilisation of historical data, reflecting our commitment to continuous learning. Our team process was enhanced through regular stand-up meetings and Jira utilisation, underscoring effective communication and collaborative tools. Addressing external disruptions with buffer time in future sprints demonstrated our adaptability and problem-solving skills. In retrospect, these experiences enriched our capacity to work effectively in software development teams while adapting to evolving tools and techniques.

ULO6: Design, plan, and evaluate the quality of a software product based on a chosen quality model/framework with justifications.

Tasks 05P and 06P – Definition of Done and Quality

- I successfully met ULO6 by designing, planning, and evaluating the quality of a software product (GotoGro) based on a chosen quality model/framework. In my report, I defined the "Definition of Done" for the project, outlining specific conditions aligned with the product quality model. I justified my choices by explaining how each quality metric relates to the project's functional and non-functional requirements. For each metric, I provided clear examples and thresholds for meeting the quality standards. This comprehensive approach demonstrates my thorough understanding of quality management and adherence to the chosen quality model, enabling effective planning and evaluation of the software product's quality.

Task 81HD - Software Project Planning, Design, and Quality Management

- I successfully met ULO6 by designing, planning, and evaluating the quality of the NutriSwap software product based on the ISO 25010 Software Quality Model and specific project goals. In my report, I established a comprehensive "Definition of Done" (DoD) that covered both functional and non-functional (quality) requirements. This DoD ensured that NutriSwap would meet the standards for successful delivery and release. I also set SMART goals to make sure the software's quality could be measured and evaluated effectively. Each SMART goal was related to a specific quality aspect, such as user satisfaction, response time, and compatibility.
- I employed the ISO 25010 Quality Model to break down the quality evaluation into various dimensions, such as Functional Suitability, Performance Efficiency, Compatibility, Usability, Reliability, Security, Maintainability, and Portability. Within each dimension, I defined specific quality metrics, and for each metric, I set clear criteria and thresholds that the NutriSwap software needed to meet. This included criteria related to functional completeness, correctness, appropriateness, and other quality aspects.
- The report also highlighted the importance of each quality metric and its relevance to the project's goals, such as aligning dietary recommendations with the Australian Dietary Guidelines for improving food choices or enhancing security with two-factor authentication to protect user data. Additionally, I incorporated a focus on maintainability and portability by setting criteria related to code reuse, documentation, adaptability, and third-party integration.

Based on the evidence provided, I believe I deserve a High Distinction grade for this unit. I have consistently demonstrated a commitment to academic excellence and a clear ability to go beyond the material presented in the course. Throughout the semester, I excelled in assigned tasks, as evidenced by my submissions on Canvas. I not only met the requirements but also consistently sought to deepen my understanding of the subject matter through additional research, innovative perspectives, and creative problem-solving. I have shown a strong capacity for critical thinking and the application of concepts to real-world situations. My dedication to academic growth and the ability to consistently exceed the expected scope of the unit's content makes me a deserving candidate for a High Distinction grade.

Reflection

The most important things I learnt:

Software Project Management Stages:

One of the most important things I've learned is the sequence of required steps when moving from a project synopsis to a fully functional project. I truly appreciate all the tasks we've undertaken, as they've been incredibly informative. This was my second time writing a project proposal, with the first being in year 12. Initially, it was quite challenging to determine what backlog items we would have, particularly in Task 01P. However, I found the challenge rewarding as it pushed me to dive deeper into the project brief and identify the necessary tasks and sub-tasks. Now, when I compare Task 01P to my high-distinction (HD) report and the tasks I devised for my HD project, I can see a significant difference. The tasks in my HD project are more specific and broken down into smaller, more manageable components compared to the somewhat vague tasks in 01P.

Definition of Done and Quality:

I found this topic particularly intriguing because comprehending it required me to gain a full grasp of the Australian/New Zealand Standards ISO 25010. Delving deeply into these standards helped me grasp the rationale behind certain choices made during the software design and development process. While I had prior knowledge of some standards, such as functional and usability standards, there were also some new ones I wasn't previously aware of. I believe it's highly valuable that I now have a better understanding of these standards.

Task breakdown and Estimations:

Another important skill I've gained from this unit is the ability to accurately estimate the effort required for tasks, whether through the use of the Work Breakdown Structure (WBS) or one of the estimation techniques discussed in this module (such as analogy, size comparison, or seeking advice from experts).

I've been with my company for almost two years, and I used to rely on estimations from my senior developers or BAs. These tasks were sometimes over-estimated and sometimes under-estimated. But, overall, they were usually close to the actual effort required.

Now, after completing this unit, I better understand how to estimate tasks. Instead of guessing, I've learned to base my estimates on past tasks, their complexity, and expert advice.

I used to wonder why we have many small tasks instead of one big one. Now I see the benefit of breaking things down. Smaller tasks are easier to estimate and complete, compared to vague, complex ones.

How to run and effectively participate in a Sprint Planning

During my two years at work, I attended one or two sprint planning sessions. This was primarily because I work part-time and hold a junior developer position. I observed that during these sessions, the team systematically reviews all existing stories and any new stories or customer requests. They prioritise these based on factors like urgency, development efforts, and business value. I now have a clear understanding of these processes, thanks to our coverage of these topics in both tasks 08P and 16P.

The things that helped me most were:

Working with this team:

To be honest, throughout my four years at university, I've always believed that I work best as an individual. However, this perspective changed when I took this unit. First, I want to express my gratitude for accommodating my request to work with specific teammates for this unit; it made a significant difference for me as it enhanced my ability to grasp the material and learn from my team members.

Collaborating with this team has been a fantastic experience. Each team member brings a diverse set of skills, backgrounds, and experiences to the table, which has been beneficial for all of us while also aiming to produce work of the highest standards. I genuinely appreciate the unit's structure and how it exposed us to all stages of Software Management as a group before requiring us to do the same individually, particularly for the High-Distinction Project.

I speak from personal experience, having faced the challenge of working in a team where some members didn't show much dedication to the unit or their grades. In contrast, I remained committed to my studies. Indeed, I can attest that choosing the right teammates is of utmost importance, as it can be the deciding factor for the success or failure of a project.

We maintained regular check-ins and conducted daily standup meetings. Additionally, we stayed in close contact with one another, readily addressing any comments or questions related to our work. What truly stood out was everyone's willingness to step up and initiate a task when others were occupied. The absence of anyone declining tasks due to other assignments was a testament to the strength of our team, demonstrating a collective understanding that we were all in the same boat, facing similar challenges.

Previous experience with Software Design and Development:

One of the most important factors that helped me was my prior experience with software design and development, acquired through previous university units as well as my work experience. These skills enabled me to speed through certain tasks and provided me with more time to focus on new assignments and unfamiliar concepts.

Regular feedback from Tutor:

Having a supportive and understanding tutor was a crucial factor in my success in this unit. I was absent for the first two weeks of the semester due to being overseas, and I greatly appreciate Harsharan's understanding and willingness to accommodate my situation by allowing me to join a specific team. Throughout the semester, I consistently received valuable feedback from her and her positivity as a tutor truly made a significant difference in my academic journey.

I found the following topics particularly challenging:

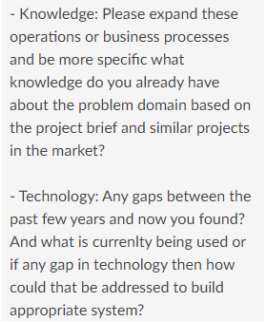
SDLC – Scrum

Initially, I found the Software Development Life Cycle (SDLC) models a bit challenging, especially since I missed the first two weeks of lectures where these concepts were covered. I did, however, have a strong grasp of the problem-solving methodology from my year 12 Software Development course, which encompassed all the SDLC steps, including Analysis, Design, Development, and Evaluation. What initially proved challenging was the Scrum concept, but as we advanced through the unit, I learned it through hands-on experience.

I realised that Scrum is essentially based on an iterative software development process, as evidenced by our work in Sprints where we prioritised tasks and moved less critical ones to the next Sprint. The daily standup meetings were also instrumental in my understanding of Scrum. Interestingly, I've been following a similar approach at my workplace, although I hadn't previously recognised it as Scrum. In my role, the regular check-ins primarily occur with the team manager rather than the product owner, making it more of an agile approach in my experience.

KoST Analysis:

The topic of KoST Analysis was indeed challenging for me, especially since it was covered in week 2, and I was still trying to catch up. Fortunately, my teammates were incredibly helpful and took the time to provide a brief explanation of KoST Analysis, which allowed me to complete task 03P. However, I still find this concept somewhat challenging, and it's evident from the feedback on my 03P submission that I need further practice and improvement in this area.

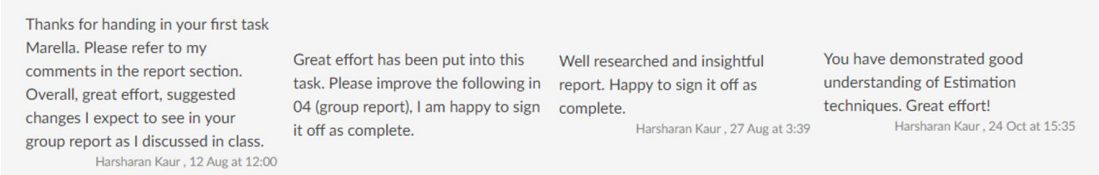


Sprint Retrospective:

This was one of the topics that I had no prior experience with. I initially found the sailboat retrospective a bit confusing, especially due to the term "retrospective." However, it started making sense when I had the chance to try it out as part of task 13P. I'm grateful for the opportunity to learn about this technique in this unit, especially considering its widespread use in the industry.

Except for the above, I was content with all the other topics covered in this unit. I was particularly pleased with the approach taken to instruct us on the fundamentals of Software Management. It's my only regret that I didn't have the opportunity to take this unit before my placement, as it would have had a substantial impact on my role and involvement in task planning, estimating, and reflection.

This can be seen from my various pass and credit tasks that were ticked off from the first submission with comments about having a good understanding and high-quality reports.



I found the following topics particularly interesting:

I found the exposure to new tools and frameworks quite interesting throughout this unit. Over the course, I gained proficiency in using JIRA, which was a new experience for me as we primarily use Azure DevOps at my workplace. Additionally, I familiarised myself with Supabase, the database framework employed in our GotoGro project. Additionally, my research on member management systems greatly contributed to a better understanding of the requirements and functionalities necessary for GotoGro.

The High-Distinction (HD) project was a particularly engaging aspect of the course. It challenged me to think creatively and step outside the box while planning an app idea that personally interested me. I thoroughly enjoyed the process, from designing the project's logo to outlining what's within scope and what's not. I also appreciated brainstorming and incorporating cutting-edge technologies such as artificial intelligence and machine learning into the project, all while ensuring that the system met accessibility requirements and satisfying the definition of done.

I feel I learnt these topics, concepts, and/or tools really well:

Scope and Product Backlog:

I have demonstrated a solid understanding of defining project scope and creating a comprehensive product backlog which is evident in my HD Report. I've been able to effectively identify and prioritize project requirements, user stories, and features based on their importance, feasibility, and business value, ensuring a clear direction for the project.

Architecture and Software Design

My understanding of software architecture and design has greatly advanced as I had the opportunity to learn and further practice it. I've developed the capability to effectively conceptualise and craft

software architectures that are in harmony with the goals of a project. This includes the appropriate selection of design patterns and the ability to make well-informed choices regarding the technologies and frameworks to employ, ultimately leading to the creation of more scalable and maintainable solutions. This improvement is clearly reflected in my work on report 14P, particularly in the section dedicated to Design Justification using Design Principles, as well as in the feedback received from the tutor, which acknowledges the quality of my architectural and UML diagrams.

Definition of Done and Quality:

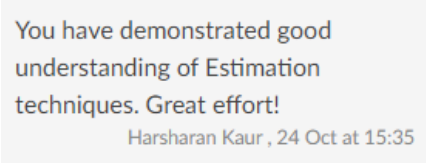
I've learned the importance of setting clear definitions of "done" for each project deliverable and have integrated quality assurance practices into the development process. I now understand that maintaining high-quality standards is crucial for producing reliable and effective software. This is shown in both my 05P, 06P and the HD reports.

Sprint Planning:

I've acquired a strong understanding of the Sprint planning process. I can efficiently conduct Sprint planning meetings, define Sprint goals, and break down user stories into manageable tasks. My ability to estimate the amount of work a team can complete within a Sprint has improved, ensuring more realistic and achievable goals. All this is shown in tasks 08P and 16P.

Estimating using WBS, Analogy, Size Comparison and Expert techniques

I've become proficient in various estimation techniques, such as Work Breakdown Structure (WBS), Analogy-based estimation, Size Comparison, and leveraging Expert judgment. This has enabled me to make more accurate estimates for project tasks and user stories, leading to better project planning and resource allocation. The feedback I received from my tutor on my credit task 61C – 63C is proof of this.



I still need to work on the following areas:

KoST Analysis:

While I have made progress in understanding KoST Analysis, I acknowledge that there's room for improvement. It's a complex concept, and I recognize that I need further practice and study to fully understand it. I plan to invest more time in researching and applying KoST Analysis principles to enhance my ability to assess and manage the knowledge and software aspects of a project effectively.

JIRA (Task Management):

Although I have gained valuable experience with JIRA during the course, there is still room for growth in terms of maximising its potential for task management. I intend to dedicate more time to exploring the advanced features of JIRA to become more proficient in utilising it for efficient task planning, tracking, and collaboration. As JIRA is widely used in the industry, having this skill will undoubtedly be beneficial.

Risk Assessment and Management:

Unfortunately, since this topic was covered later in the year, I did not get much time to review it and learn it properly. This is a critical aspect of project management, and I aim to work on identifying potential risks, analysing their impact, and developing effective risk mitigation strategies. Enhancing my capabilities in this area will contribute to a more comprehensive approach to project management and ensure smoother project execution.

This unit will help me in the future:

The knowledge and skills I've gained from this unit will undoubtedly have a significant impact on my future studies and career. Understanding the Software Project Management stages, such as project scoping, product backlog creation, and Sprint planning, will enable me to excel in project management roles, both in academic settings and the professional world. The ability to estimate task effort accurately using various techniques and my improved understanding of software architecture and design principles will be invaluable as I tackle complex projects and work as part of a development team. The exposure to tools like JIRA and database frameworks like Supabase will enhance my ability to collaborate, provide valuable insights and deliver high-quality software. Moreover, learning the importance of quality standards and adhering to Australian/New Zealand Standards ISO 25010 will allow me to produce more reliable and user-friendly software.

If I did this unit again I would do the following things differently:

If I were to repeat this unit, I would make an effort to address my areas of improvement more diligently. I would dedicate additional time and practice to thoroughly grasp KoST Analysis, work on my task management, and strengthen my skills in risk assessment and management. Additionally, I would actively seek out more opportunities to work in a team, as I've discovered the immense value of collaborative work during this unit. It has broadened my perspective and enriched my learning experience. Furthermore, I would ensure a consistent and disciplined approach to regular check-ins and daily standups with my team, as real-time communication and collaboration proved to be highly effective. Lastly, I would continue to be proactive in seeking guidance from my tutor, as their feedback and support greatly contributed to my progress in this unit.

Other:

Thank you for your time, and I sincerely hope that my dedication and hard work are evident in both my submitted work and this report. I hope you find me deserving of the grade I am requesting.