G2J SOFTWARE PROJECT PLAN

1.0 Introduction

Grocery stores need to process many concurrent customer purchases while ensuring their inventory stays stocked with products in high demand. We will be developing a cash register program that will process the items as they are scanned and auto-order to assist with inventory management.

1.1 Project scope

Major inputs will be the products scanned corresponding to their barcode. The program will process those inputs, add to a counter and auto-order new products once the count meets pre-set conditions. It will also process when new inventory is received and added to the existing stock.

1.2 Major software functions

The program will consist of a user interface, which allows products, associated barcodes, units per case and desired stocking requirements to be entered by stakeholders/software customer prior to implementation. The software customer will also need to be able to update inventory when new stock is received and added to current stock.

Data storage will be needed using an in-memory data structure to track products, by case and individual units. The program will need to connect to a database to make the system persistent across sessions and take inputs from different register inputs.

The program will need to calculate when additional cases are needed based on the number of units sold and current stock inventory. The inventory will need to be updated and log future delivery orders.

Output will need to display updates on current inventory and pending orders. It will also need to generate reports, as needed of current inventory, weekly account of units sold (by case) and orders that are pending for delivery.

1.3 Performance/Behavior issues

Inventory needs to be updated in real time and requires a prominent level of reliability and efficiency. Needs to handle large volumes of usage by several different cash registers concurrently during peak hours.

1.4 Management and technical constraints

We will only be using a small sample of specific products and barcodes as initial inputs to ensure program functionality. There are 100,000+ possible products that the software customer will need, and we will be unable to include that large of a volume during this time limit.

Bulk lists of products and barcodes can be purchased online, but we are not provided with a budget to obtain this information.

2.0 Risk Management

2.1 Project Risks

Software tool underperformance: Product inventory will not be tracked accurately, leading to incorrect ordering of new stock, or not ordering enough stock to meet customer needs.

Storage underestimate: Memory is not properly allocated to meet the needs of the number of products and inventory needed by the customer.

Storage underperformance: Persistent storage of inputs to the system fail and inventory is not correctly tracked between sessions.

Size underestimate: The size of the system has been underestimated, slowing down processing of information and program execution.

2.2 Risk Table

Risk	Probability	Impact	RM3 Pointer
Software Tool underperformance	Moderate	Catastrophic	Implement rigorous testing (this includes unit, integration, and end-to-end), real-time monitoring of transactions, and continuous user feedback to refine the functionality.
Storage underestimate	Low	Serious	Optimize the memory allocation, conduct performance stress tests, and ensure efficient data handling to avoid storage limitations.
Storage underperformance	Moderate	Catastrophic	Implement automated backup systems, redundancy mechanisms, and database integrity checks to prevent data loss.

Size underestimate	Low	Moderate	Design a scalable system architecture, run the performance benchmarks, and allocate the resources dynamically based on usage demands.

2.3 Overview of Risk Mitigation, Monitoring, Management (RM3)

To effectively manage the project risks, we will implement a structured approach that focuses on mitigation, monitoring, and management (RM3). Our RM3 strategy includes identification of risks, assessment, and implementation of plans to minimize their impact.

Mitigation Strategies:

Software Tool Underperformance: Implement detailed testing procedures that includes unit and integration testing to validate the accuracy of inventory tracking and order generation. User feedback will be gathered to refine functionality regularly.

Storage Underestimate: Memory allocation and data structures will be optimized to ensure efficient storage use. Performance testing with increasing data loads to assess system scalability will be conducted.

Storage Underperformance: Automated backups and redundancy mechanisms will be implemented to ensure constant storage. The database system will be rigorously tested to prevent data loss across the sessions.

Size Underestimate: Scalable architecture will be utilized to accommodate potential system growth while the performance benchmarks will be set to evaluate the system's efficiency under varying loads.

Monitoring Approaches:

Performance assessments will be conducted regularly to track the system's efficiency. While the logging and error reporting mechanisms will be integrated to detect and diagnose potential failures.

There will be constant testing and validation that will be performed at each development stage to ensure the system's reliability.

Management Plans:

We will oversee the risk assessments and mitigation updates throughout the development cycle. In which, if the critical risks materialize, fallback solutions such as manual inventory adjustments or temporary overrides will be implemented as they resolve the system's failures.

The stakeholders will be updated on risk status and mitigation efforts regularly to ensure alignment with the project's objectives. This structured RM3 approach will help maintain the system's reliability, efficiency, and scalability, while minimizing disruptions in inventory management and transaction processing.

3.0 Project Schedule

3.1 Project task set

We will follow the **Incremental Process Model**, allowing us to develop the system in well-defined stages. This model provides flexibility, enabling us to test and validate each functional increment before moving forward.

Framework Activities:

1. Communication: we will determine user requirements; John has worked in a grocery store for over 7 years with experience ordering products. He has the experience required to justly represent how the end user would expect the software to behave. He represents our user for this project.
2. Planning: We will develop a comprehensive plan for distributing our resources, as well as determine potential risks and errors and how we may mitigate them
3. Modeling: We will model the input/output handling for the backend including the way UPCs are read and processed, and the database schema.
4. Construction: Will be handled incrementally in the series of steps defined as follows.
    1. Develop and test the backend data processing
    2. Develop and test the data base
    3. Integrate the backend data processing and the database to complete the backend
    4. Develop and test the GUI
    5. Integrate the GUI and the backend

Task Set:

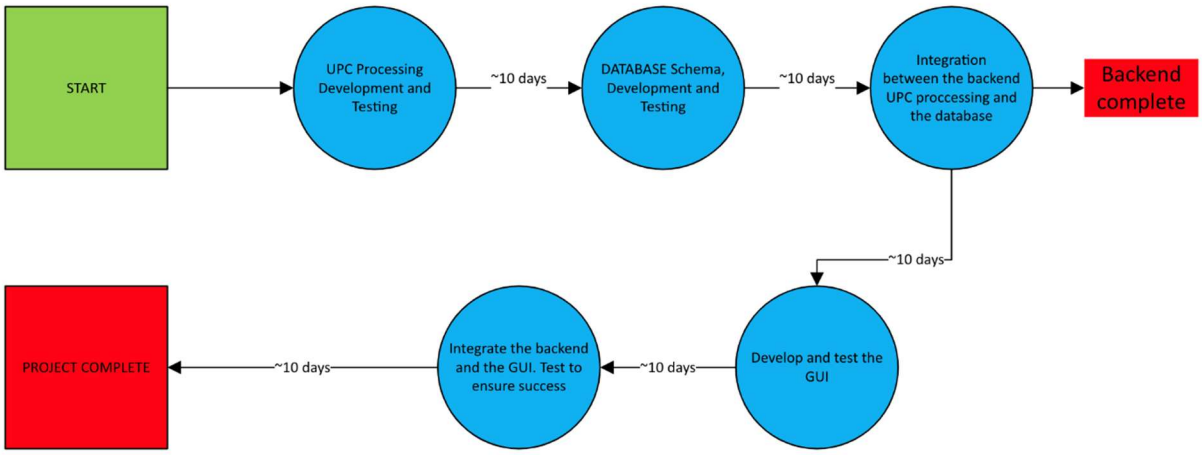
1. Requirement Analysis (Weeks 1-6)
2. UPC processing logic (Develop and Test) (Week 7-8)
3. Design Database schema and create the database (Week 8-9)
4. Integrate the UPC processing logic software with the database to form the complete backend (Week 9-10)

5. GUI (Development and Testing) (Week 10-11)
6. Integration of Backend and GUI (Develop and Test) (Week 11-12)

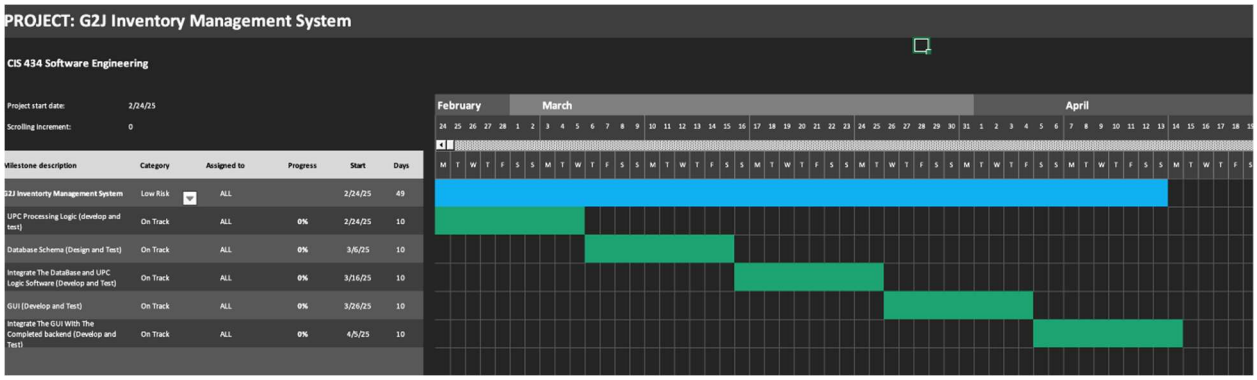
3.2 Functional decomposition

1. Input Handling
2. Read and process UPC codes from input files
3. Validate input data
4. Inventory Processing Logic
5. Track item counts against case sizes
6. Generate re-order lists based on thresholds
7. Output Handling
8. Write required orders to an output file
9. Graphical User Interface (GUI)
10. Build a simple and intuitive GUI for file selection, processing, and result display
11. System Integration
12. Connect backend logic with the GUI
13. Testing and Validation
14. Unit testing for backend
15. GUI testing
16. System testing (end-to-end)

3.3 Task network



3.4 Timeline chart



3.5 Schedule compliance

We will be using the class blackboard to ensure that each deliverable is available when it is due. Additionally, our team will communicate over both discord and email to ensure each portion of the project is developed and tested within the scope of proper requirements and expectations