SOFTWARE REQUIREMENTS SPECIFICATION

Inventory Management System

G2J

1.0 Introduction

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

1.1 Goals and objectives

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

1.2 Statement of 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.

An input file will be created to initially set up desired barcodes, units per case, and threshold for when to initiate an order to replace stock. Once the threshold is met, a re-order list will be generated. Bulk orders, by case, will be written to an output file, to restock product inventory. A report can be generated listing current inventory and order history.

1.3 Software context

The program will have a user interface, which allows products, associated barcodes, units per case and desired stocking requirements to be entered by stakeholders/software customers before implementation. This will be used as the program's input file. 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 as product stock is updated.

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.

1.4 Major 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 Usage scenario

Use case examples are listed below for store employees who will be responsible for using and updating the inventory management system.

2.1 User profiles

Inventory Control Specialist: Mid-level employee who will be updating the program when new inventory is received and responsible for overall inventory management. May have little programming experience.

Cashier: Entry-level employee who will be scanning barcodes to update product purchases. May have no programming or computer experience.

Store Manager: Higher level employee who will have program oversight and will be reviewing inventory updates and product orders. They will need to run reports for updates on store inventory. Managers will require training on the program’s functionality and have more experience with inventory management.

2.2 Use-cases

Inventory Management System: Inventory Processing Use Case	
Actors	Inventory Control Specialist (ICS), Store Manager, System Input File, Inventory Management System
Description	ICS may update the input file with desired product barcodes, units per case and threshold of when new product orders need to be initiated. ICS will transfer input file data to Inventory Management System and update as new Inventory is received. ICS can run reports listing current inventory and expected replacement inventory.
Data	Current Inventory count, Units per Case, new Inventory Count
Stimulus	User command issued by ICS
Response	Confirmation that Inventory has been updated.
Comments	The ICS must have appropriate security permissions to create the input file and update the Inventory Management System. The store manager should also have the same security permissions as backup to ICS and for proper oversight.

Inventory Management System: Process Barcodes Use Case	
Actors	Cashier, Cash Register, Inventory Management System
Description	Customer will arrive at checkout with their products for purchase. Cashiers will scan or enter barcodes in the cash register system, which will update the count in the Inventory Management System.

Data	Product barcodes
Stimulus	Cashier scans or enters barcode
Response	Confirmation that barcode is valid.
Comments	Cashiers will have access permissions for the system used to scan or enter barcode (cash register).

Inventory Management System: Report Generation Use Case	
Actors	Store Manager, Inventory Management System
Description	The store manager can generate reports on current inventory, product inventory trends and expected inventory for delivery.
Data	Current Inventory count, Units per Case, new Inventory Count, Report
Stimulus	User command issued by Store manager.
Response	Output file created with contents of report.
Comments	The Manager must have appropriate security permissions to generate reports.

2.3 Special usage considerations

Program needs to have simple user interface that does not require extensive computer programming knowledge to understand and use correctly. Product

inventory will be updated by grocery store employees who may have no experience with computers or programming.

3.0 Data Model and Description

The data model encompasses the structure, relationships, and storage mechanisms for all critical data, ensuring reliability, scalability, and persistence across multiple cash register sessions. It leverages an in-memory data structure for real-time performance and a relational database for long-term storage, supporting concurrent usage during peak hours. The cash register and inventory management system that is designed for grocery stores processes real-time customer purchases through barcode scans, updates the inventory levels, and automates reordering to maintain stock based on demand.

3.1 Data Description

The software will manage and manipulate the data related to the products, inventory, transactions, and orders to ensure accurate stock tracking and automated reordering. These data objects will be stored in an in-memory data structure for real-time processing and will be continually used across sessions in the database.

3.1.1 Data objects

Product's Attributes:

- ProductID (this is a unique identifier used for the product, ex: it can be a barcode/UPC)
- Name (this is the product name, ex: it can be represented as "Milk")
- UnitsPerCase (this is the number of individual units in a case, ex: it can be 12)
- DesiredStockLevel (this is the target stock level in units or cases)
- ReorderThreshold (this is the minimum stock level that triggers reorder)

Inventory Attributes:

- InventoryID (this is a unique identifier used for the inventory entry)
- ProductID (this foreign key links to Product)
- CurrentStockUnits (this is the current number of individual units in stock)
- CurrentStockCases (this is the current number of cases in stock)
- LastUpdated (this is the timestamp of the last inventory update)

Transaction Attributes:

- TransactionID (this is a unique identifier that is used for each sale)
- ProductID (this is a foreign key that links to Product)
- UnitsSold (this is the number of units sold in the transaction)
- Timestamp (this is the date and time of the transaction)
- RegisterID (this is the identifier of the cash register used)

Order Attributes:

- OrderID (this is a unique identifier that will be used for each order)
- ProductID (this is a foreign key that is linked to the Product)
- CasesOrdered (this is the number of cases that were ordered)
- OrderDate (this is the date the order was placed)
- Status (ex: it can be represented as, "Pending," "Delivered" and it will be used as a way for the customer to track the state of their product)
- ExpectedDeliveryDate (it tracks the anticipated delivery date for the product)

3.1.2 Relationships

Relationships among data objects are described using an Entity-Relationship Diagram (ERD) like form. No attempt is made to provide detail at this stage.

Product to Inventory: (1:1) One-to-One
Each product will have a single corresponding inventory entry that will track its stock levels.

Product to Transaction: (1:M) One-to-Many
A single product can have or be involved in multiple transactions as it is sold over time.

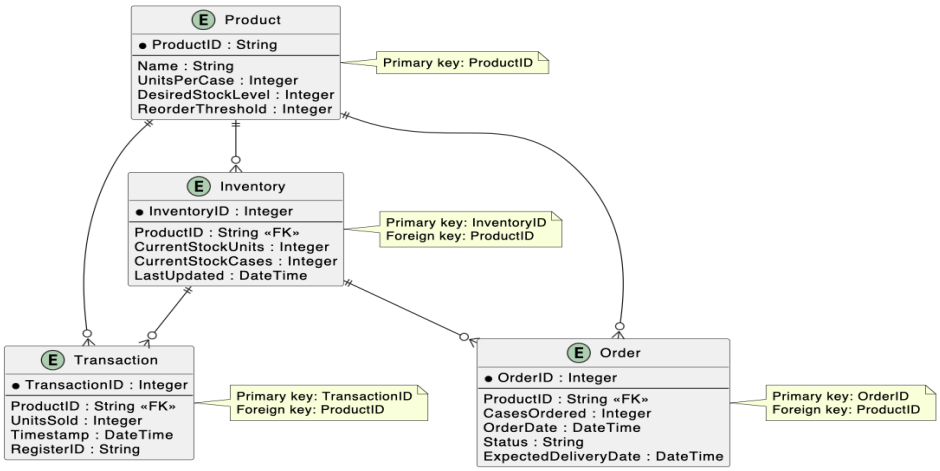
Product to Order: (1:M) One-to-Many
A single product can have multiple orders placed when the stock falls below the reorder threshold.

Inventory to Transaction: (1:M) One-to-Many
Each inventory entry is updated by multiple transactions as the products are sold.

Inventory to Order: (1:M) One-to-Many
Each inventory entry can trigger multiple orders when the stock levels require a refill.

3.1.3 Complete data model

Below is the detailed Entity-Relationship Diagram (ERD) that represents the data objects and their relationships with each other.



3.1.4 Data dictionary

A reference to the data dictionary is provided. The dictionary is maintained in electronic form.

<u>This dictionary includes these features:</u>

Field Name: ex: ProductID, UnitsSold

Type/Datatype: ex: Integer, String, DateTime

Description: ex: "a Unique identifier for each product based on its barcode"

Constraints: ex: "Non-null," "Unique"

Related Objects: ex: "Links to Inventory through ProductID"

An example of what the dictionary will look like with the features listed above:

Field Name	Type	Description	Constraints	Related Objects
ProductID	String	Unique barcode/UPC for a product	Non null and it is unique	Relates to inventory, transaction, order
UnitsSold	Integer	The number of units sold in a transaction	Non-null Or (it can greater than or equal to 0)	Relates to transaction

4.0 Functional Model and Description

A description of each major software function and software interface is presented.

4.1 Description of Major Functions

Each requirement is uniquely identified based on the major software functions outlined in the project plan

4.1.1 Input Handling

- The software shall read and process Universal Product Codes (UPCs) from a text file generated by scanned inputs at cash registers. It shall validate input data to ensure accurate product identification and prevent processing errors.
- Inputs are product UPCs scanned via barcode during customer transactions. The system processes these inputs to update inventory counts.

4.1.2 Inventory Processing Logic

- The software shall track individual item counts against predefined case sizes and generate re-order lists when inventory levels fall below specified thresholds.
- The system calculates when additional cases are needed based on units sold and current stock, ensuring automatic re-ordering to maintain stock levels. This will be accomplished via the modulus function

4.1.3 Output Handling

- The software shall generate a txt file to be sent out as an order. It will be capable of displaying updates on current inventory, pending orders, and reports including weekly units sold (by case) and pending delivery orders.

4.1.4 Data Storage and Persistence

- The software shall maintain an in-memory data structure to track products (by case and individual units) and connect to a persistent database to store inventory data across sessions.

- Ensures data consistency and availability across multiple cash register inputs and sessions.

4.1.5 User Interface for Configuration

- The software shall provide a graphical user interface (GUI) allowing stakeholders to enter products, associated barcodes, units per case, and desired stocking requirements prior to implementation, as well as update inventory counts and desired levels when new circumstances arise.

4.2 Software Interface Description

The software interface to the outside world is described based on the project plan's scope and functionality.

4.2.1 External Machine Interfaces

- The software interfaces with cash register hardware to receive barcode scan inputs from multiple registers concurrently.
- The system must handle large volumes of real-time inputs during peak hours, ensuring compatibility with standard cash register barcode scanners.

4.2.2 External System Interfaces

- The software interfaces with a persistent database system to store and retrieve inventory data across sessions.
- The database ensures data persistence and supports integration with backend logic for inventory tracking and re-ordering.

4.2.3 Human Interface

- The software provides a graphical user interface (GUI) for stakeholders to configure product details (barcodes, units per case, stocking requirements), update inventory, and view reports.
- The GUI will be simple, designed for file selection, data entry, transaction processing, and result display (e.g., current inventory, pending orders). It supports real-time monitoring and manual inventory updates.

5.0 Restrictions, Limitations, and Constraints

Special issues which impact the specification, design, or implementation of the software are noted here, in Sections 1.4 and 2.0 of the project plan.

- **Limited Product Scope**: The initial implementation will only include a small sample of specific products and barcodes to ensure functionality, rather than the

full 100,000+ products required by a potential software customer. This constraint is due to time and resource limitations.

- **Performance Requirements**: The system must update inventory in real time and handle large volumes of concurrent usage from multiple cash registers during peak hours, requiring high reliability and efficiency.
- **Risk of Underperformance**: Potential risks include inaccurate inventory tracking (software tool underperformance), insufficient memory allocation (storage underestimate), database failures (storage underperformance), and slow processing due to underestimated system size. Mitigation strategies (e.g., rigorous testing, scalable architecture) are planned but impose additional design and implementation demands.
- **Scalability Limitation**: While a scalable architecture is planned, the initial system size may not fully accommodate future growth without further resource allocation and performance benchmarking.