Grocery Delivery Service SRS

Paul Swenson, Jheremi Villarreal, Joshua Villarreal,
Matt Marroquin, Oscar San Juan

Table of Contents

1 Introduction

This document will identify and break down each intended process for the grocery delivery service application. This is intended for software engineers, software developers, and any stakeholders in the development process.

1.1 Motivation

Grocery shopping can take a lot of time out of a person's day when they may not have much free time to begin with. The Grocery Delivery Service application will allow customers to shop online via a web app to place orders to be delivered to their home from a local grocery warehouse.

1.2 Scope

This application will keep track of the current inventory available, its pricing, and any coupons available to the customer. The warehouse will be capable of being monitored by any employees to restock and set pricing as necessary.

In store shoppers will be able to use the app to view orders they need to assemble for delivery. Drivers will also be able to access the delivery service application to view orders and take them to their final destination.

All of these functions will be carried out via a web app capable of running on the most popular web browsers.

1.3 Definitions

Terms	**Definitions**
User	Any human who interacts with the application
Customer	A type of user who will interact with the application to shop and place an order
GDS	The Grocery Delivery Service application
Purchase Order	An order placed by a customer
Driver	A user who delivers the purchase orders to the customer
ROP	A threshold that indicates a product must be re-stocked in inventory
Product	An item that is available for purchase
Shopper	A user who will gather items in the warehouse to prepare purchase orders for shipment
Employee	A user who is paid by the company
In-store Employee	A type of employee including: Managers, Shoppers, and Warehouse Managers

2 Description

This section will outline different tables to be used for the implementation of the GDS. Included are use cases, use case diagrams, functional requirements, non-functional requirements and the architecture of the intended system.

2.1 Use Cases

Use Case Table

Actor	**Use Case**	**Description**
Customer	UC1: Browse and Search	The customer must be able to browse and search for products
Customer	UC2: Shopping Cart Editing	The customer must be able to add, remove, and save items in their shopping cart
Customer	UC3: Account Creation	The customer must be able to create an account once providing personal information.
Customer	UC4: Purchase Order Creation	The customer must be able to create and pay for a purchase order from their shopping cart for fast or normal delivery.
Customer and Manager	UC5: Purchase Order History	The customer and Manager must be able to view all previous purchase orders and receipts.
Driver	UC6: Delivery of Purchase Order	The driver must be able to access a purchase order and customer address to deliver an order.
Driver	UC7: Delivery Confirmation	The driver must be able to confirm the successful delivery of a purchase order.
Manager and Warehouse Workers	UC8: Warehouse Orders	The managers and warehouse workers should have access to a list of all products that need to be reordered.
IT Staff	UC9: Warehouse Maintenance	The IT staff should be able to access the warehouse database to edit any discrepancies or fix any issues pertaining to the warehouse.
Customer	UC10: Coupon Access	The customer must be able to access a list of weekly coupons.
Shopper	UC11: Purchase Order Fulfillment	The shopper must be able to access a list of purchase orders to fill for the drivers.
Managers	UC12: Access Employee Information	The manager must be able to access relevant employee information for normal business purposes.

2.2 Functional Requirements

Functional Requirements Table

Identifier	Description	Priority
REQ1	The system shall allow the customer to browse products.	5
REQ2	The system shall allow the customer to place and remove products from a shopping cart	5
REQ3	The system shall allow the customer to create a user account with their personal information.	4
REQ4	The system shall allow the customer to purchase all items in a shopping cart to be delivered to their address.	4
REQ5	The system shall create a shopping list for every order so shoppers can gather the items.	2
REQ6	The system shall keep a history of previous orders.	2
REQ7	The system shall allow managers and shoppers to edit customer orders if products are not in stock.	2
REQ8	The system shall allow in-store employees to scan items when they are being picked from the shelves	1
REQ9	The system shall allow the customer and manager to cancel an order	2
REQ10	The system shall allow for refunds to be made in case of damaged or incorrect orders.	1
REQ11	The system shall provide drivers with a GPS navigation tool.	1
REQ12	The system shall allow warehouse managers to edit the amount/weight of products in inventory	4
REQ13	The system shall allow managers to change product pricing	3
REQ14	The system shall allow managers to create coupons for products	2
REQ15	The system shall allow managers to see sales numbers for different time periods	2
REQ16	The system shall allow managers to create new employee profiles	1
REQ17	The system shall allow managers to update employee information	1
REQ18	The system shall give recommendations for products to existing customers	1
REQ19	The system shall allow customers to set delivery times	4

Identifier	Description	Priority
REQ20	The system shall allow customers to view the status of their order	3
REQ21	The system shall allow customers to save shopping carts for future orders.	3
REQ22	The system shall allow customers to apply coupons to orders	3
REQ23	The system shall allow customers to search for products	3
REQ24	The system shall provide warehouse managers a list of products that need to be restocked	2
REQ25	The system shall allow drivers to confirm the delivery of an order	2

2.3 Non-functional Requirements

Non-functional Requirements Table

Identifier	Description
NF1	The database must respond and update quickly to any changes that may be made to it
NF2	The GDS must be easy to use for both customers and employees
NF3	The GDS must be able to recover data if it fails at any time
NF4	The GDS must be able to run on any web browser
NF5	The GDS must store passwords and user information in a secure manner
NF6	Employee and customer information should only be accessible to those who need to see it
NF7	The GDS should be able to easily adapt to new changes
NF8	The GDS must be able to be easily maintained
NF9	The GDS must be able to relay information via a remote server
NF10	The GDS should be capable of interacting with scanners that the shoppers carry
NF11	The GDS should log out users and employees after a set period of inactivity
NF12	The GDS should have a sense of interoperability, being able to be used across different platforms

2.4 Architecture

The Grocery Delivery System (GDS) that has been designed is a robust system meant to be available always for customers to interact with. There are backups for both the database and server units and the main portion of the system has been secured by firewall as well as a user access control node at the gateway portion of the main system device network. The payment method for the GDS has been isolated to a payment gateway service. In addition to firewall protection, a user access control (UAC) gateway both checks and routes all incoming traffic to and from the system by access right classification.

A hybrid of layered and microservice architecture has been selected to be the style of choice for this system. As the company who will be utilizing the GDS will likely have a distributed network of stores across a vast area, implementing the core functions of information technology networks as a layered architecture will play well to the strengths of a large company's organizational and communication structure of employees. The term "layered" is being broadened here to encompass subsets of client/server, object-oriented software, and database within the same system, but all are ultimately organized in a layered architecture fashion. At the application end, as well as a few other key functions, the microservice architecture style is used to maintain a degree of agility and flexibleness to improve deployment, scalability, and development of applications to better serve the customer.

A high-level overview of the layered architecture:

Layer 1	Presentation	UIs, applications, browser communication logic
Layer 2	Business logic layer	Servers, UAC, admin subsystem
Layer 3	Database	database

As this company is going to focus primarily on customer service over internet-based applications, it would make sense to have a very powerful network able to provide the customer with service. For this reason, layered architecture has been chosen to most strongly build a IT infrastructure that is able to sustain such a critical responsibility for the company. Layered architecture has been chosen over a pure client/server or central repository model for the simple fact that there are too many components which need to work together in an orchestrated fashion to get the product to the customer on a web-based application, this is especially true for a large company such as this one. To maintain an organized system structure as well as more technical things such as synchronicity of datastores and a-synchronicity of processes serving customers, layered is superior to others such as peer-to-peer, which is very poor at maintaining synchronicity, or pipe and filter, which requires a specific set of ordered computations performed on ordered data. On a final note, layered architecture makes testing systems very easy.

Seeing as applications are the main way the company interacts with customers it also makes sense to design the GDS in a way that facilitates the fast development and deployment of applications. Layered architecture unfortunately is not suited to this. However, if the edge system components are designed in a way such that they are largely independent components with isolated functions, as is the case with microservices, the components can be developed and deployed independently. As a side note, the asynchronous facilitation of processes will work well with the unpredictable nature of the delivery driver availability and input.

As the GDS matures, it will be able to perform and be developed well by company employees due to its layered architecture foundation, whatever needs must be met in terms of scale and agility will either be met or aided by the microservice edge-of-system architecture. If a high degree of understanding and customer knowledge is required at the application end, the microservice architecture will isolate these parts of the system and enable team to work entirely within the bounds of the application without needing to worry about other components of the larger system.

In terms of satisfying functional requirements, the GDS is separated into components which are specialized to perform their functions. All user information, transaction data, accounts, history, etc. is stored in the database which passes data to retrievals and updates. The GDS uses a centralized database to prevent things such as inter-service communication which might lead to asynchronous data. The servers handle HTTP requests, domain specific logic, retrieval and updating the data in the database, and generate the views to be sent to the browser. The end applications meet all customer needs.

As this is a large, scalable system which will no doubt have many moving parts and additions/detractions throughout its life cycle, each component is going to need to be independent from the others to facilitate: ease of exchangeability, minimal interdependency, clarity of responsibility, separation of concerns, maintainability, isolation of failure events, security, etc. which are met nicely by the layered/microservice hybrid architecture. Data integrity is more than covered by the database, server, application subsystems. The development environment as was explained earlier is more than sufficient. Applications that are integrated into the system are meant to be as easy to use and agreeable to whatever device the customer may use, which encompasses many aspects of the transaction including: browsers, payment methods, etc.

Maintaining the system on the business logic and database portion of the system is a function of how well the developers organize their efforts. On the application end the independent components are also very well able to be maintained as they operate as independent components, which means that developers who are tasked to maintain a specific component can become well acquainted to the well-defined boundaries of that component. Along the same lines, separation of responsibility of the system components mirror that of the developer organization structure. Each component is responsible for well-defined tasks, in some cases only these tasks, which make for a very easy definition of responsibility. As the system is organized modularly and in a layered architecture, various components can be swapped out, scaled up or down, or maintained as an isolated event, which covers many nonfunctional requirements the system may face throughout its lifetime.

2.5 Acceptance Test Cases

Test Case: Browse (Customer)

Input: Open inventory view

Expected Outcome: Inventory should be listed and user should be able to browse the inventory

Test Case: Search (Customer)

Input: In the search bar enter “Cereal”

Expected Outcome: All available Cereal should display

Test Case: Shopping Cart Adding (Customer)

Input: While browsing the inventory click on “+” next to an item

Expected Outcome: The item should be added to Shopping Cart

Test Case: Shopping Cart Remove (Customer)

Input: While browsing the inventory click on “-” next to an item

Expected Outcome: The item should be removed from Shopping Cart

Test Case: Create new Account (Customer)

Input:
1: Open the Account View
2: Click on “Create Account”
3: Fill in customer information
4: Click “Create”

Expected Outcome: The New Account View should be displayed.

Test Case: Create new Purchase Order (Customer)

Input:
1: Open Purchase Order View
2: Click on “Create New Purchase Order”
3: Fill in required information
4: Click “Create”

Expected Outcome: A new Purchase Order should be saved to the database and set to “Not Paid”

Test Case: Pay for Purchase Order (Customer)

Input:
1: Open Purchase Order View
2: Click on “Pay Purchase Order”
3: Fill in required information
4: Click “Pay”

Expected Outcome: Purchase Order should be set to “Paid”

Test Case: View Purchase Orders (Customer / Manager)

Input:
1: Open Purchase Order View
2: Click on “View Purchase Orders”

Expected Outcome: Current and previous purchase orders should be displayed

Test Case: View Purchase Orders (Driver)

Input:
1: Open Purchase Order View
2: Click on “View Purchase Orders”

Expected Outcome: Purchase orders and customer’s address to deliver an order are displayed.

Test Case: Delivery Confirmation (Driver)

Input:
1: Open Purchase Order View
2: Click on “View Purchase Orders”
3: Click on “Delivered” next to a Purchase Order

Expected Outcome: Purchase Order should be set to “Delivered”

Test Case: Warehouse Orders (Managers & Warehouse Worker)

Input:
1: Open “Restock Needed” View

Expected Outcome: A list of products that need to be ordered should be displayed

Test Case: Database Management (IT)

Input:
1: Open “Database” View
2: Attempt to edit Database

Expected Outcome: Database should be displayed and be modified by IT Personnel

Test Case: Coupons (Customer)

Input:
1: Open “Coupons” View

Expected Outcome: A list of current coupons should be displayed

Test Case: Shopper Fulfillment (Shopper)

Input:
1: Open “Purchase Orders” View

Expected Outcome: Several Purchase Orders should be displayed with a list of items to be purchased.

Test Case: Employee Profiles (Manager)

Input:
1: Open “Employee Profiles” View

Expected Outcome: A list database of Employee Profiles should be displayed

Appendix

I List of Diagrams

All tables are included in the folder for submission.

Sequence Diagrams:
Warehouse Workers.png
Shopper.png
Customer.png
Manager.png
Driver.png

Use Case Diagram:
UseCase.png

Architecture:
GDS_ArchDiagram.png

II List of Tables

III Sources

https://medium.com/microservices-in-practice/microservices-layered-architecture-88a7fc38d3f1
https://www.cl.cam.ac.uk/teaching/0910/ConcDistS/14-accessCtrl.pdf
https://www.susanjfowler.com/blog/2016/12/18/the-four-layers-of-microservice-architecture
https://www.safaribooksonline.com/library/view/software-architecture-patterns/9781491971437/ch02.html
http://microservices.io/patterns/data/event-driven-architecture.html
https://dzone.com/articles/why-we-need-a-new-breed-of-hybrid-microservices-pl
https://www.safaribooksonline.com/library/view/software-architecture-patterns/9781491971437/ch01.html