***Grill Fresh Online***

Contents

[1. Introduction 3](#_Toc56625183)

[2. Context 3](#_Toc56625184)

[2.1 Grill Fresh Online (GFO) 3](#_Toc56625185)

[3. Functional Overview 4](#_Toc56625186)

[4. Quality Attributes 5](#_Toc56625187)

[5. Constraints 6](#_Toc56625188)

[6. Principles 6](#_Toc56625189)

[6.1 Microservices Architecture 6](#_Toc56625190)

[6.2 API Gateway / BFFs 6](#_Toc56625191)

# Introduction

This document offers a high-level overview and explains the proposed architecture for “Grill Fresh Online”, a pizza online ordering system for Grill Fresh Inc.

# Context

Grill Fresh Inc wants to grow its business across the country in a gradual manner, in terms of customer base, volume of business and number of stores. It wants to achieve this goal through the following accelerators of business:

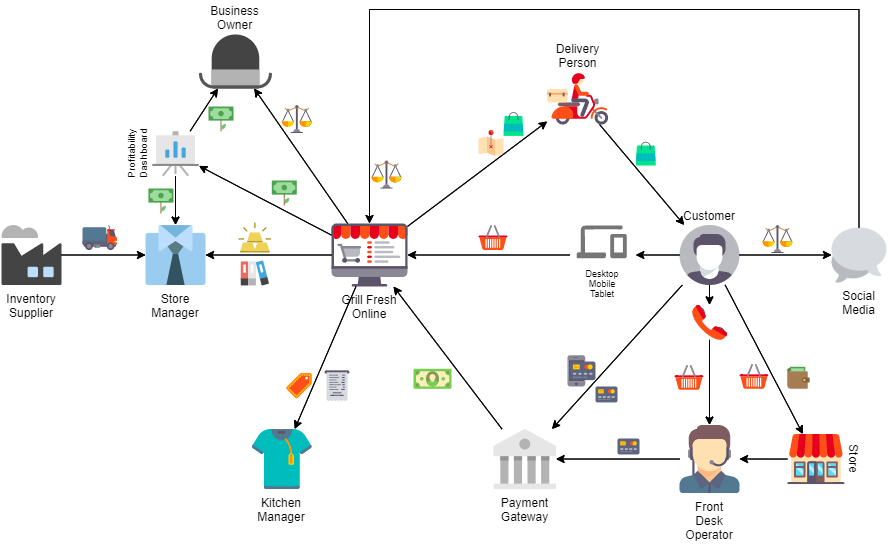
* A customer places orders for pizzas through online with any nearby store – instant and easy
* A customer receives delivery through additional modes like Take away and Home delivery, apart from dine-in option
* The business owner gathers a holistic view of profitability at business level or individual store level – to locate customer interests and expansion opportunities
* A store owner gathers a holistic view of profitability at store level – to understand business health and to act
* The business owner and store owners prefer to understand the sentiments of customers about business – to collate and correlate feedbacks (in near real-time, if possible)

Grill Fresh Inc is convinced that the potential enabler for this goal is to embrace the online platform opportunities.

## Grill Fresh Online (GFO)

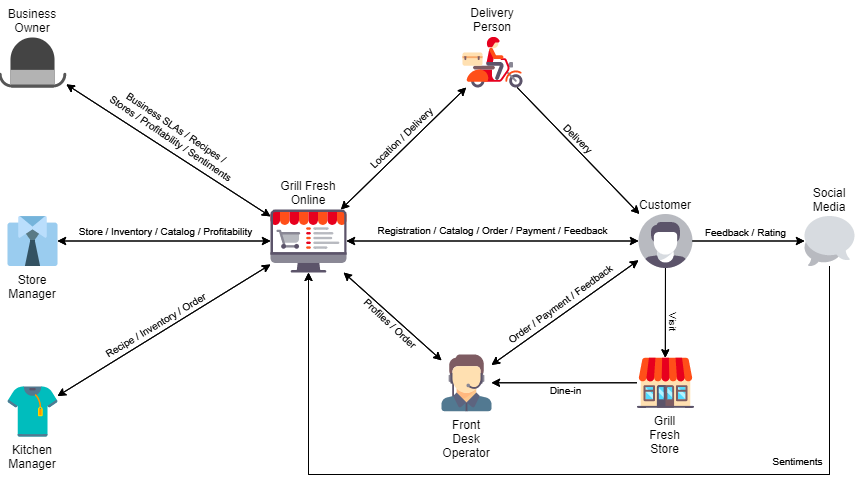
GFO system is a unified platform for all stakeholders of business – customers, business owner, store manager, front desk operator, kitchen manager and delivery person – catering to their varied interests. It is accessible through various devices – desktops, laptops, tablets, smart phones and Point-of-Sale.

At a high level, this system along with its neighborhood systems:



# Functional Overview

Various major workflows fulfilled by the GFO system are as described below:



* Customer:
  + Register and manage profile – contact details and preferences
  + Browse through catalog of available items – pizzas, toppings, sides, desserts, beverages
  + Place orders for take away and home delivery
  + Pay for orders using varied secure payment options
  + Track status updates for orders placed
  + Provide sentiments about a store, item or overall business by direct feedback or over social media
* Store Front Desk Operator:
  + Browse through catalog and place orders on behalf of a customer – for dine-in, take away and home delivery
  + Approve and move orders into queue for Kitchen Manager
  + Track status updates for all orders placed in store
* Kitchen Manager:
  + Browse through recipes and ingredients for preparing items
  + Check inventory of ingredients and notify purchase needs
  + View and receive notification about incoming orders in queue
  + Process and notify status updates for orders
* Delivery Person:
  + Pick-up out-for-delivery orders from delivery queue
  + View order and delivery information for customer
  + Confirm after delivering orders
* Store Manager:
  + Manage store – address, serving radius, front desk operators, kitchen managers and delivery persons
  + Manage catalog of items serving in store
  + Manage inventory and purchases for ingredients
  + Find profitability of store
  + Find sentiments from customers about store
* Business Owner:
  + Set business SLAs – order fulfillment time, max inventory levels, delivery charges, etc.
  + Manage stores – stores and store managers
  + Manage catalog of items serving in business and prices
  + Manage recipes and ingredients for preparing items
  + Find profitability of stores
  + Find sentiments from customers about business and stores

# Quality Attributes

GFO needs to fulfill following quality expectations of stakeholders to offer a flawless, convenient and delightful experience:

* Availability
  + System should be available 24x7 to fulfill important workflows:
    - Receive orders from customers
    - Fulfill business activities inside stores – recipe, inventory, order and payment management
  + A downtime of “less than 3 minutes per day” can be tolerated.
  + System shall continue being available even during deployment, if possible.
* Security
  + System should be available only to authorized users reaching through internet cloud.
  + System should support role-based authorization.
  + System should support different authentication mechanisms for easy and intuitive onboarding experience:
    - Internal identity provider based on stored credentials
    - External authentication providers: Google, Facebook and Twitter
  + System should be deployed within the country for statutory data compliance requirements.
* Scalability
  + System should initially support workflow traffic happening across 2 existing outlets.
  + The workflow traffic is expected to increase gradually with an estimated growth plan as described below:

|  |  |
| --- | --- |
| **Year** | **# of cities (3 outlets per city)** |
| 1st | 5 |
| 2nd | 10 |
| 3rd | 20 |
| 4th | 40 |
| 5th | 80 |

* + The number of customers is expected to grow to 80 lakhs in 5 years.
  + There should not be any impact in system response time due to such planned increase in workflow traffic.
* Reliability
  + System should 100% reliably capture shopping cart details even on client-session expirations.
  + System should 100% reliably capture orders once the cart is checked-out and payment is made.
  + System should 100% reliably track and notify order status.
  + System should 100% reliably alert on lower inventory levels to raise purchase orders on time.
* Accuracy
  + System should 100% accurately calculate prices, taxes and other charges for every order.
  + System should 100% accurately pass approved orders to kitchen queue in same sequence as received.
  + System should 100% accurately calculate inventory levels based on orders and wastages.
  + System should 100% accurately calculate profitability at store-level and at business-level.
* Latency
  + System should notify about order status updates within 2 seconds.
  + System should pass approved orders to kitchen queue with zero-time delay.
  + System should fetch customer profile, wish-list items and respective availability-in-store details within 1 second for front desk operator.
  + System should fetch catalog details and respective availability-in-store details within 2 seconds for customer.
* Observability
  + System should provide continuous updates about and ability to monitor system health.
  + System should automatically restart on any failures.
  + System should provide system-wide activity logs.
* Usability
  + Learning curve of transition to cloud-based online platform for the users should be intuitive

Any other attributes can be evaluated periodically and decided based on more understanding of requirements.

# Constraints

* IT infrastructure – stakeholder wants to invest in IT infrastructure incrementally in concurrence to business expansion.
* Point of entry – customers shall be able to place orders from variety of devices – laptops, tablets and smart phones
* Point of service – front desk operators shall be able to place orders from point-of-sale devices and smart phones, kitchen managers shall be able to update orders from point-of-service devices and business owner and store managers shall be able to access system from laptops.
* Technology stack – modern trending development stack to be used for system to portray as technology innovator or pioneer in domain and to attract new tech savvy generation.

# Architecture Principles

## Microservices Architecture

The architecture proposes a microservices architecture implementation with multiple autonomous microservices in order to ensure:

* Services aligned to business capabilities – ownership of business capabilities can be with small focused teams
* Loosely coupled – clear microservices boundaries and interaction across boundaries through well-defined APIs
* Highly cohesive and isolated – each microservice owns its data and database
* Independently scalable – scale-out only certain required business capabilities on high demand
* Highly resilient and non-disruptive deployment

[Polly](https://github.com/App-vNext/Polly) is used to support resiliency measures using patterns like retries and circuit breakers.

Each microservice runs inside a separate container.

## API Gateway / BFFs

**API gateway**

There are few concerns in exposing APIs from individual microservices directly to client:

* Typically, business capabilities to support a single functionality are spread across multiple focused microservices. For e.g. a functionality for placing an order involves the services namely “Catalog”, “Cart”, “Payment” and “Order”. APIs are very fine-grained in nature, while the client needs are much larger. Consequently, the client code that offers this functionality needs to fetch information by calling APIs from multiple services. This results in multiple roundtrips to server, thus increasing latency.
* When API endpoints change, client dependencies get disrupted and consequently need to be synchronized.

In order to address these concerns, the architecture places the APIs behind an API Gateway that takes care of fanning out a client request to multiple required microservices appropriately.

Since the API gateway acts on behalf of server by consolidating all APIs, it encapsulates all boilerplate functionalities like load balancing, authentication, authorization and more in a single place.

**Backends for Frontends (BFFs)**

There are few significant differences in supporting disparate devices like laptops and mobile devices:

* Data types – due to smaller size of screens, mobile devices tend to show a subset of data than laptops.
* Data formats – since only a subset of data needed, mobile devices can use efficient techniques like GraphQL than REST APIs.
* Network speeds – due to slower mobile network speeds, mobile devices prefer lesser requests and roundtrips.

In order to serve for these differences efficiently, the architecture defines a separate API gateway for each kind of client, in other words, a separate backend for each frontend – laptops and mobile devices. Each BFF provides a unique endpoint for its clients and then forwards the call to specific microservice or custom aggregator.

There are 4 BFFs – one for customer-facing shopping functionalities and another for internal business operation functionalities (like business and sentiments insights), one variant each for different clients.

The architecture proposes an implementation using [Envoy](https://www.envoyproxy.io/) to employ its advanced capabilities towards load balancing, health-check observability, dynamic routing, JWT and external authentications, role-based access controls.

Each BFF runs inside a separate container.

## Communication

**Event-driven workflows**

This system exchanges messages (or events) between microservices to establish an orderly, reliable and efficient communication mechanism. It uses publish/subscribe and topic-based communication through messaging channels of RabbitMQ to exchange these messages.

RabbitMQ runs inside a separate container.

**Data interchange format**

The architecture proposes JSON as the preferred data interchange format for major communications between microservices and between client/server.

## Diagnostics and Observability

**Logging**

The architecture proposes a system-wide, centralized and structured logging using [Serilog](https://serilog.net/) such that it becomes easy and robust to diagnose failures faster.

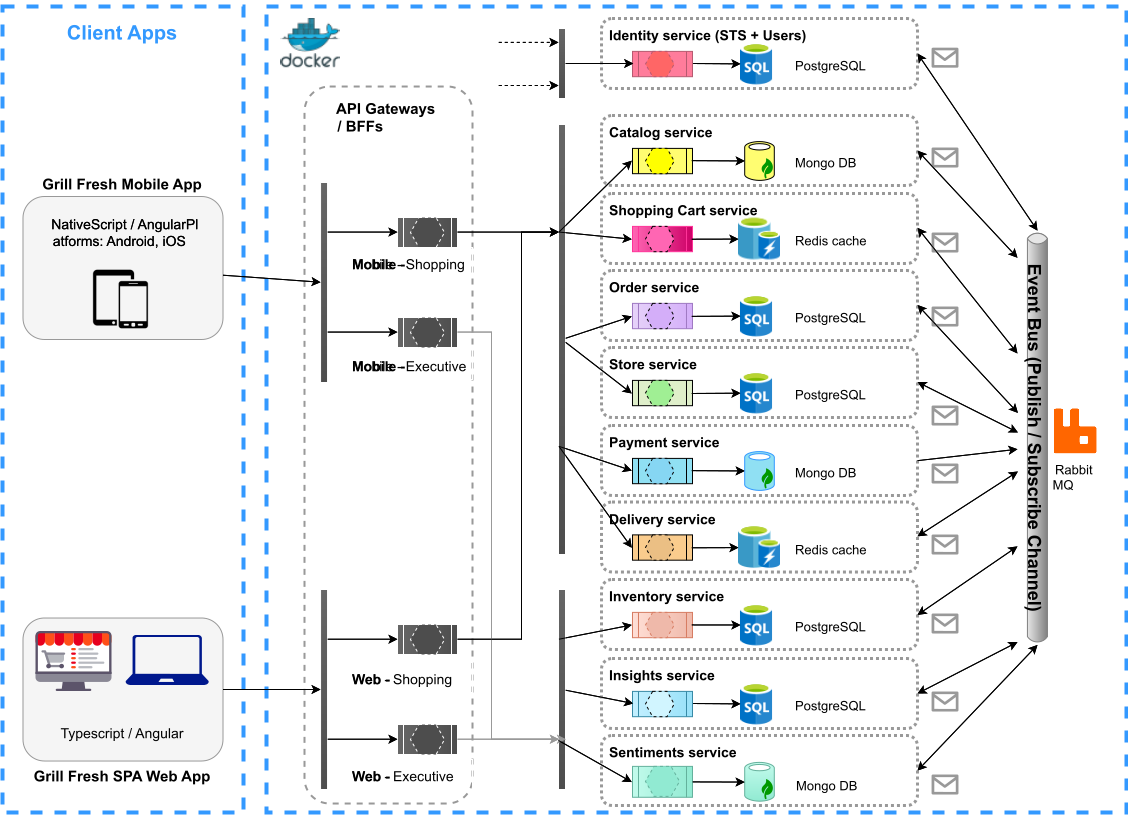
Serilog and logging service run inside a separate container.

**Observability**

All applications and microservices expose endpoints that check the application and all its dependencies.

Health check APIs run on a separate container.

# Architecture



## Application

The architecture proposes two variants of the application that can be respectively accessed from mobile devices and laptops.

**SPA Web App**

The architecture proposes to develop the laptop client as a SPA Web App.

Technology Stack

Identity Service

* Store Front Desk Operator:
  + Handle payments using varied secure payment options
  + To view serviceability to a customer location from this store for home delivery
  + To locate a store able to serve a customer at a given location for dine-in, take away or home delivery
  + To view recent “n” orders of a customer from this and any other store
  + To receive alerts about all orders or orders placed by self in the store – order-going-into-delay, order-delayed
  + To view rich details about all orders placed by self during the day, week, month or year
* Kitchen Manager:
  + To start processing an order – to notify the status of order as in-preparation
  + To fulfill processing an order – to notify the status of order as ready
  + To view list of items possible to serve in the store for the day and next “n” days
  + To view predicted maximum order quantities for individual items
  + To view recipes for preparing various items including ingredients
  + To check availability of inventory levels for ingredients in comparison to max inventory levels required to fulfill the predicted maximum order quantities
  + To trigger purchase orders for required ingredients for approval by Store Manager
* Delivery Person:
  + To view the order number, customer name, customer address
  + To continuously report current location
  + To pick-up orders in ready state and to confirm about orders that are delivered
* For Store Manager:
  + To set list of items possible to serve in the store for the day and next “n” days
  + To set predicted maximum order quantities for individual items (for next “n” days)
  + To set alert quantity levels for individual items
  + To check availability of inventory levels for ingredients in comparison to max inventory levels required to fulfill the predicted maximum order quantities
  + To approve and raise purchase orders for required ingredients
  + To assign an order that is ready to pick-up to a delivery person available
  + To manage store – update address, update serving radius, create/read/update/delete store front desk operators and kitchen managers
* For Business Owner:
  + To set order fulfillment SLA time
  + To set list of items possible to serve through the business and prices
  + To set the recipes for preparing various items including ingredients
  + To manage stores – create/read/delete store, create/read/update/delete store manager
* Order Placing Module
  + To not allow store front desk operator and direct customers to place order for “Sold Out” items in the store
* Order Fulfillment Module
  + To raise alerts about orders-going-into-delay, order-delayed based on order fulfillment SLA time
  + To raise alerts to order placing module about “Sold Out” items in a store
* Catalog Management Module

– order-placed, order-confirmed, in-preparation, ready, out-for-delivery, delivered