SE Proj 1c1

Team: Anmol Koul, Arnav Mejari, Om Kumar Singh, Shweta Patki

New Use Cases

1. Browse Restaurant Menu

Story: A customer wants to browse available restaurants and their menus. The customer should be able to filter and sort restaurants based on various criteria.

Scenario: Customer successfully browses the menu of a restaurant.

Preconditions:
- The customer is logged into the app.
- The app has an internet connection.

Main Flow:
- The customer views the list of restaurants [View Restaurants].
- The customer selects a restaurant [Select Restaurant].
- The customer views the restaurant's menu [View Menu].
- The customer browses menu categories [Browse Categories].
- The customer views menu items details [View Item Details].
- The customer adds items to the cart [Add to Cart].

Subflows:
- **[View Restaurants]**: The system displays restaurants based on location and other filters.
- **[Select Restaurant]**: The customer taps on a restaurant from the list.
- **[View Menu]**: The system displays the menu items, categorized and with images.
- **[Browse Categories]**: The customer navigates through different food categories (e.g., appetizers, desserts).
- **[View Item Details]**: The system shows detailed information for selected items (description, price, ingredients, dietary information).
- **[Add to Cart]**: The system adds the selected item to the customer's cart. If the item is out of stock, it notifies the user [Out of Stock].

Alternative Flows:
- **[No Internet Connection]**: The app displays a message indicating a lack of internet connectivity.
- **[Restaurant Unavailable]**: The selected restaurant is temporarily unavailable or closed. The app displays a message indicating this.
- **[Out of Stock]**: Selected item out of stock; the system prompts to select another item or removes it from cart.

2. Place Food Order

Story: A customer wants to order food through the app. The order is successful only if items are available, payment is successful, and the restaurant accepts the order.

Scenario: Customer places order successfully

Preconditions:
- The customer is logged into the app.
- The customer has selected at least one item.

Main Flow:
- The customer selects food items [Select Items].
- The customer reviews the cart and proceeds to checkout [Checkout].
- The customer enters their delivery address [Enter Address].
- The customer makes payment [Make Payment].
- The system validates payment [Validate Payment].
- The restaurant accepts the order [Accept Order].

- The order is confirmed [Order Confirmed].

Subflows:
- **[Select Items]** The customer browses the menu and adds items to the cart. Includes handling of quantity adjustments.
- **[Checkout]** The customer reviews the order summary, including price and delivery information.
- **[Enter Address]** Customer can use saved addresses or enter a new one. Includes address validation.
- **[Make Payment]** The customer selects a payment method (card, wallet, cash on delivery).
- **[Validate Payment]** If valid, proceed. If invalid, display error and allow retry or cancellation.
- **[Accept Order]** Restaurant confirms or rejects order within a time limit.
- **[Order Confirmed]** The system generates an order ID and sends notifications to customer and restaurant.

Alternative Flows:
- **[Out of stock]** Items unavailable. User is notified and can modify order or cancel.
- **[Invalid Address]** Customer enters an invalid address. Address correction prompted.
- **[Payment Failure]** Payment fails due to insufficient funds or card issues. Retry or give alternative payment.

3. Process Payment

Story: The system processes a customer's payment for an order. Payment succeeds if details are correct and funds are available.

Scenario: Payment is successful

Preconditions:
- The customer has items in the cart.
- The customer has proceeded to checkout.

Main Flow:
- The customer enters payment details [Enter Payment].
- The system submits details to the payment gateway [Send Request].
- The gateway verifies funds [Verify Funds].
- The payment succeeds [Confirm Payment].

Subflows:
- **[Enter Payment]** The customer selects a payment method and enters details. Includes input validation.
- **[Send Request]** Securely transmits payment information to the gateway. Includes handling of timeouts.
- **[Verify Funds]** The gateway verifies funds; success leads to [Confirm Payment]; failure leads to [Payment Failed].
- **[Confirm Payment]** Payment is confirmed; the order proceeds. Includes transaction ID generation.
- **[Payment Failed]** Payment rejected; user notified and can retry or choose another method.

Alternative Flows:
- **[Gateway Down]** The payment gateway is unavailable. Retry after a delay is suggested.
- **[Invalid Payment Details]** Incorrect payment information. Error message and opportunity to correct.

4. Restaurant Accepts Order

Story: A restaurant accepts an incoming order.

Scenario: Restaurant successfully accepts an order.

Preconditions:
- An order has been placed and is pending.

Main Flow:
- The restaurant receives an order notification [Receive Order].
- The restaurant views order details [View Order].
- The restaurant accepts the order [Accept Order].
- The system updates the order status [Update Status].
- The customer receives an order confirmation [Notify Customer].

Subflows:
- **[Receive Order]** The restaurant receives a notification (push notification, app alert).
- **[View Order]** The restaurant accesses the order details (items, address, customer information).
- **[Accept Order]** The restaurant confirms the order acceptance.
- **[Update Status]** The system changes the order status to "Accepted".
- **[Notify Customer]** The customer receives an app notification that the order is accepted.

Alternative Flows:
- **[Order Rejected]** The restaurant rejects the order due to unavailability of items or other reasons; notifies the customer and system.
- **[Order Timeout]** Order is automatically canceled after a specified time if the restaurant does not respond.

5. Assign Delivery Driver

Story: The system assigns a driver to an accepted order. A driver is assigned if one is available nearby.

Scenario: Driver assigned successfully

Preconditions:
- An order has been placed and accepted by the restaurant.

Main Flow:
- The system identifies nearby available drivers [Find Driver].
- The system sends a delivery request to the nearest driver [Send Request].
- The driver accepts the request [Driver Accepts].
- The driver is assigned to the order [Assign Success].

Subflows:
- **[Find Driver]** System queries driver location and availability using GPS data. Includes prioritization based on proximity and rating.
- **[Send Request]** Push notification sent to the driver with order details and estimated compensation.
- **[Driver Accepts]** Driver accepts or rejects the request. Rejection triggers [Find Driver] again.
- **[Assign Success]** Order is linked to the driver. Includes updating order status and notifying the customer.

Alternative Flows:
- **[No Drivers Available]** No drivers are available nearby. Order is delayed or canceled with notification to the customer and restaurant.
- **[Driver Unavailable]** Selected driver becomes unavailable. The system attempts to assign another driver.

6. Update Order Status

Story: The system updates the order's status as it progresses through the delivery process.

Scenario: Order status updated successfully

Preconditions:
- An order exists.

Main Flow:
- An event occurs (order received, preparing, ready, picked up, en route, delivered).
- The system updates the order status accordingly [Update Status].
- The system notifies relevant parties (customer, restaurant, driver) [Notify Parties].

Subflows:
- **[Update Status]** Database update of order status. Includes timestamps.
- **[Notify Parties]** Notifications sent via push notifications, SMS, or email. Includes customizable notification settings.
- **[Handle Exceptions]** Unusual situations (delayed, canceled) are handled with appropriate notifications.

Alternative Flows:

- **[Notification Failure]** Notification fails to be sent. System logs the error and attempts retry.
- **[Status Update Failure]** System fails to update status. Error is logged, and retry is attempted.

7. Driver Picks Up Order

Story: A delivery driver picks up an order from a restaurant.

Scenario: Driver successfully picks up an order.

Preconditions:
- An order is assigned to a driver and the order status is "Accepted".

Main Flow:
- The driver arrives at the restaurant [Arrive at Restaurant].
- The driver confirms order pickup [Confirm Pickup].
- The system updates the order status [Update Status].

Subflows:
- **[Arrive at Restaurant]** The driver uses the app's navigation to reach the restaurant.
- **[Confirm Pickup]** The driver confirms in the app that they have picked up the order.
- **[Update Status]** The system changes the order status to "Picked Up".

Alternative Flows:
- **[Order Not Ready]** The restaurant informs the driver that the order is not yet ready. The system updates the status accordingly and notifies the customer if necessary.
- **[Driver Cannot Locate Restaurant]** Driver unable to locate the restaurant; contacts support or cancel order.

8. Track Delivery in Real Time

Story: A customer can track the delivery driver's location in real time.

Scenario: Customer successfully tracks delivery.

Preconditions:
- The order is assigned to a driver.
- The driver has accepted the order.

Main Flow:
- The system receives location updates from the driver [Receive Location].
- The system updates the customer's view of the delivery location [Update Map].

Subflows:
- **[Receive Location]** GPS data is received from the driver's app. Includes error handling for loss of signal.
- **[Update Map]** Real-time map update for the customer showing driver's location. Includes visual cues

Alternative Flows:
- **[GPS Signal Lost]** Driver's GPS signal is temporarily lost. Map displays last known location.
- **[Driver App Issue]** Driver's app malfunctions, preventing location updates. Customer is notified.

9. Driver Completes Delivery

Story: A delivery driver completes a food delivery to the customer.

Scenario: Driver successfully completes delivery.

Preconditions:
- The order status is "Picked Up".

Main Flow:

- The driver arrives at the delivery address [Arrive at Address].
- The driver confirms delivery [Confirm Delivery].
- The system updates the order status to "Delivered" [Update Status].
- The customer is notified of the delivery [Notify Customer].

Subflows:
- **[Arrive at Address]** The driver uses the app's navigation to reach the delivery location.
- **[Confirm Delivery]** The driver confirms in the app that the order has been delivered.
- **[Update Status]** The system changes the order status to "Delivered".
- **[Notify Customer]** The customer receives a notification that the order has arrived.

Alternative Flows:
- **[Customer Not Available]** The customer is not available at the delivery address. The driver contacts support or attempts redelivery (if applicable).
- **[Incorrect Address]** The delivery address is incorrect. The driver contacts support for clarification.

10. Manage Restaurant Menu

Story: Restaurant owners can manage their restaurant's menu on the platform.

Scenario: Restaurant owner successfully adds a menu item.

Preconditions:
- The restaurant owner is logged in.

Main Flow:
- The owner accesses the menu management section [Access Menu].
- The owner adds a new item [AddItem].
- The system validates the input [Validate Input].
- The menu is updated [Update Menu].

Subflows:
- **[Access Menu]** Secure access to menu management tools.
- **[AddItem]** Input fields for item name, description, price, image, and other relevant information.
- **[Validate Input]** Checks for correct data types and range of values (e.g., positive price).
- **[Update Menu]** Database update of menu information.

Alternative Flows:
- **[Invalid Input]** Incorrect input detected. Error messages and instructions for correction are provided.
- **[Duplicate Item]** Attempting to add an item that already exists. Appropriate notification given.

Reflection Document [How We Decided What NOT to Do]

Our primary strategy for defining the MVP was a relentless focus on the **"critical path"** of a single, successful order. We defined this path as the absolute minimum sequence of events required for the platform to fulfill its core promise: get food from a restaurant to a customer. We excluded features by grouping them into three main categories:

1. **Post-Transaction Features:** These are features that add value *after* the core delivery is complete. This was the largest category of exclusions.
   - **Examples:** **`Rate and Review Order`** (UC 8), **`View Order History`** (UC 25), and **`Handle Complaints/Refunds`** (UC 7).
   - **Reasoning:** While crucial for long-term trust, retention, and community building, the platform can successfully execute its primary function, delivering an order, without them. In an MVP stage, feedback can be gathered through direct user interviews, and refunds can be handled manually by the business owner via email, avoiding the need to build a complex support system.
2. **Growth and Optimization Features:** These are features designed to scale the business, increase efficiency, or improve marketing reach.
   - **Examples:** **`Promotional Offers Management`** (UC 20), **`Apply Coupon Code`** (UC 23), and **`Monitor Business Performance`** (UC 9).
   - **Reasoning:** The first goal of an MVP is to validate the core business model, not to optimize it. Before we provide restaurants with analytics or customers with discounts, we must first prove that they will use the platform at all. These features introduce complexity and are only valuable once a baseline of users and transactions exists.
3. **High-Effort "Self-Service" Onboarding:** These are features that automate the process of adding new partners to the platform.
   - **Examples:** **`Restaurant Onboarding`** (UC 17) and **`Driver Onboarding`** (UC 18).
   - **Reasoning:** This is a classic MVP tradeoff. Building a robust, secure, and automated onboarding system with document validation is a significant technical undertaking. For the initial launch, the business can follow the "do things that don't scale" principle. The platform owner can manually onboard the first cohort of restaurants and drivers, which is faster to launch and allows for direct relationship building with early partners.

Potential Stakeholder Disappointments & Negative Impacts

This lean MVP, while functional, will inevitably lead to disappointments for every stakeholder group by failing to address their secondary needs.

- **For Customers:**
  - **Lack of Trust Signals:** Without ratings or reviews, customers cannot easily differentiate between restaurants, making their choice riskier.
  - **No Financial Incentives:** The inability to use coupons or see promotional offers may disappoint price-sensitive users.
  - **Clumsy Issue Resolution:** With no in-app support chat or formal complaint system, resolving a problem with an order will require emailing support, feeling slow and impersonal.
- **For Restaurant Owners:**
  - **Inability to Compete:** They cannot run promotions to attract new customers or reward loyal ones, limiting their marketing capabilities on the platform.
  - **Data Blindness:** The lack of an analytics dashboard means they have no insight into their sales trends, popular items, or customer demographics, hindering their ability to make informed business decisions.
- **For Delivery Drivers:**
  - **Opaque Earnings:** Without a dedicated earnings dashboard (`View Driver Earnings` was excluded), drivers may feel a lack of transparency and control over their income, potentially leading to mistrust.
  - **Manual Onboarding:** A manual sign-up process could be slower and more cumbersome than a competitor’s automated system.
- **For Business Owners:**
  - **High Operational Load:** The lack of self-service onboarding and in-app support tools means the business must handle these tasks manually, creating a significant and unscalable operational burden.
  - **Limited Growth Levers:** Without built-in promotional tools, driving initial user acquisition and growth will be more challenging.

Strategic Inclusions to Appease Stakeholders

While the MVP is lean, we made several deliberate inclusions that might seem secondary but are, in fact, critical for appeasing key stakeholders and ensuring the platform is viable even in its initial form.

1. **Why we included: `Manage Restaurant Menu`:** While we excluded automated restaurant *onboarding*, we specifically *included* a self-service menu management tool.
   - **Reasoning:** Onboarding is a one-time event that can be handled manually. However, menu items, prices, and availability change constantly. Forcing a restaurant owner to email a support address for every daily special or "86'd" item would create an immediate and unsustainable operational bottleneck. Providing this tool is a critical concession to **Restaurant Owners**, giving them the essential control they need and preventing the system from collapsing under the weight of manual update requests.
2. **Why we included : `Track Delivery in Real Time`:** This feature was prioritized as essential for the customer experience.
   - **Reasoning:** The single most common question in food delivery is "Where is my order?" By providing real-time tracking, we proactively answer this question, which has two benefits. First, it directly appeases **Customer** by managing their expectations and reducing anxiety, a key factor in satisfaction. Second, it drastically reduces the potential volume of inquiries to our (initially manual) customer support channel, thereby helping the **Business Owner** manage operational costs.
3. **Why we included : `Update Order Status`:** This is the communication backbone of the entire operation.
   - **Reasoning:** This use case appeases **all primary stakeholders** simultaneously with minimal complexity. It tells the **Customer** their order has been accepted. It informs the **Restaurant** that a driver has been assigned. It signals to the **Driver** when an order is ready for pickup. Removing this single feature would render the entire workflow opaque and dysfunctional for everyone involved. It is the minimum viable system for inter-stakeholder communication.

Prompt History:

Stakeholders and Biases Prompt: https://chatgpt.com/share/68b21303-5f5c-8007-ba51-c43f93cdcd1b
https://claude.ai/share/17b74850-0977-490b-a564-547498295637

Expansion Prompt :- (This prompt was given to Gemini PRO via API. We used a RAG model to get extra contextual information. We found that markdown was an efficient way to communicate with the LLM.)

You are a software developer who is performing requirements engineering on a food delivery platform.
Here is what you know about software engineering.
I will provide a use case format that I would like you to follow for future requests. Please analyze this structure and be ready to apply it.
The format consists of:
- A definition of a use case
- The first principles that should be considered when creating a use case
- An example of a use case divided into Preconditions, Main Flow, Subflows, and Alternative Flows.
- Make sure that each use case has about (2-6 points only) main flows,(4-8 points only) subflows and (2-6 points only)alternative flows.
- Keep the format of the output the same as this one.
- GIVE ENTIRE USE CASES DO NOT REDUCE.

Use Case: A use case is a method to organize system requirements by describing a sequence of interactions between a system and its users (actors) to achieve a specific goal. A Use case divides into Predictions, Main Flow, Subflows, Alternative Flows. For example:

Precondition=
- The traffic light has been initialized

Main Flow =
- The driver approaches the intersection.
- The driver checks the status [Check Status].
- The driver clears the intersection [Go].

Subflows =
- [Check Status] The driver checks the light [Check Light] and the queue [Check Queue]. If the light is green and the queue is empty the driver clears the intersection [Go]. Otherwise, the driver joins a queue [Join Queue].
- [Check Light] Check if the light is red, yellow, or green.
- [Check Queue] Check if the queue is empty or not.
- [Go] The driver clears the intersection and the use case ends.
- [Join Queue] The driver joins the end of the queue and checks status every 15 seconds [Check Status].

Alternative Flows =
- [Light Out] The light is not turned on. Wait for a clear intersection and gun it.
- [Accident] An accident is blocking the intersection. Rubber neck and slowly drive around it.
- [Ice Storm] There is an ice storm preventing safe driving. Abandon your car and walk.

Your task is to take the provided content and improve it.
Here is the PDF content you should improve on:

1. Place Food Order

Story: A customer wants to order food through the app. The customer can only place an order if items are available, payment is successful, and the restaurant accepts the order. Otherwise, the order is canceled.

Scenario: Customer places order successfully

Preconditions:
The customer is logged into the app.

Main Flow:
The customer selects food items [Select Items].
The customer reviews the cart and proceeds [Checkout].
The customer makes payment [Make Payment].
The system validates payment [Validate Payment].
The restaurant accepts the order [Accept Order].
The order is confirmed [Order Confirmed].

Subflows:
[Select Items] The customer browses the menu and adds items to the cart.
[Checkout] The customer confirms items and provides a delivery address.
[Make Payment] The customer selects a payment method (card, wallet, cash).
[Validate Payment] If the payment is valid, the order continues. Otherwise, the customer retries or cancels the order [Cancel Order].
[Accept Order] The restaurant confirms the order. Otherwise, the order is rejected [Cancel Order].
[Order Confirmed] The system notifies the customer and restaurant.
[Cancel Order] The order is canceled and the customer is notified.

Alternative Flows:

[Backend fails] The app experiences downtime or slowness. The user tries again later.
[Out of stock] Items have run out between adding them to the cart and payment. The user chooses other items.
2. Process Payment
Story: A customer wants to pay for an order. Payments are processed if details are correct and funds are available. Else payment fails.
Scenario: Payment succeeds
Preconditions:
Customer has items in the cart and proceeds to checkout.
[Receive Payment] The driver is notified of the credit.
[Given the List of 10 Use Cases generated before hand]
Question or Task: Can you give me 10 use cases for food delivery app in the same form?
Here is additional retrieved information for context, use this to improve the PDF content:
[8 chunks queried through RAG] DO NOT include any of these use cases or any variation in the output, GIVE ENTIRELY DIFFERENT ONES
1) Place Food Order
2) Process Payment
3) Assign Delivery Driver
4) Update Order Status
5) Track Delivery in Real Time
6) Manage Restaurant Menu
7) Handle Complaints/Refunds
8) Rate and Review Order
9) Monitor Business Performance (Restaurant)
10) Manage Driver Payments
11) Manage Customer Accounts
12) Manage Restaurant Profiles
13) Customer Support Chat
14) Restaurant Search and Filtering
15) Push Notifications
16) Driver App Login
17) Restaurant Onboarding
18) Driver Onboarding
19) Customer Profile Management (Photo Upload)
20) Promotional Offers Management (Restaurant)
Problem Condensation:- https://g.co/gemini/share/fac10df59f14