# SE Proj 1a1

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

## List of Stakeholders

| Primary | Supporting | Indirect |
|---|---|---|
| End Users / Customers | Restaurant Staff | Regulators / Government Bodies |
| Restaurant Owners / Managers | Payment Gateways / Financial Institutions | Suppliers (to restaurants) |
| Kitchen Staff | Technical / Development Team | Third-Party Service Providers |
| Delivery Drivers / Riders | Customer Support / Service Team | Investors / Shareholders |
| Business Owners / Platform Owners | Marketing & Sales Teams | Competitors |

## Biases

| | Profit | Quality | Compliance | Convenience | Cost Control | Risk Management |
|---|---|---|---|---|---|---|
| **Price** | Restaurant Owners / Customers | | | | | |
| **Speed** | | Kitchen Staff / Customers | | | | Development Team / Investors |
| **Growth** | | | Investors / Regulatory Officers | | | |
| **Security** | | | | Payment Gateways / Customers | | |
| **Compensation** | Delivery Drivers / Restaurant Owners | | | | | Business Owners / Delivery Drivers |
| **Cash Flow** | | | | | Restaurant Owners / Business Owners | |

- Price – Profit: Customers want low prices while restaurant owners need higher margins.
- Speed – Quality: Customers want faster delivery,kitchen staff need time for preparation.

- Growth – Compliance: Investors demand rapid expansion, regulators enforce compliance.
- Security – Convenience: Customers want seamless payments, but gateways enforce extra security steps.
- Compensation – Profit: Drivers seek fair pay, restaurant owners minimize labor costs.
- Speed – Risk Management: Investors want speed, development teams prioritize system stability.
- Cash Flow – Cost Control: Owners reinvest quickly, businesses manage expenses.
- Compensation – Risk Management: Business owners try to cut costs, but drivers see reduced pay as unfair risk.

## Comment on Prompting

We used prompting to develop the above biases table and list.
- Zero-shot prompting produced a list of 5 different conflicts between two stakeholders along with brief explanations in a few points.
- In contrast, a more careful approach using few-shot prompting - where examples, proper context, structured instructions, and output format were provided - resulted in a more comprehensive list and a tabular representation of the conflicts, making it easier to track each conflict between stakeholders, regardless of its scale. With further refinement of the prompts, we can develop a pipeline that first generates stakeholders, then simulates a conversation by taking on the roles of different stakeholders (capturing the flow of ideas), and finally outputs all potential biases in a tabular format.

Putting the outputs generated by Claude into ChatGPT and giving all the context to it results in it agreeing that the output given by few-shot prompting is better.

Overall better list: Second list (table + biases)

1. Comprehensive coverage: Includes more stakeholders, direct and indirect conflicts, and broader bias categories.
2. Multi-dimensional analysis: Shows conflicts along Profit, Quality, Compliance, Convenience, Cost, Protection, and Risk axes.
3. Future-proof: Captures nuanced operational, strategic, and technical trade-offs beyond immediate conflicts.
4. Structured for RE: Table format + one-line explanations make it actionable for requirements trade-off analysis, risk assessment, and system design decisions.

## Use cases

### 1. Place Food Order

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/Story: Customer places order successfully
Preconditions: The customer is logged into the app.

**Main Flow**

1. The customer selects food items [Select Items].
2. The customer reviews the cart and proceeds [Checkout].
3. The customer makes payment [Make Payment].
4. The system validates payment [Validate Payment].
5. The restaurant accepts the order [Accept Order].
6. 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 the payment method (card, wallet, cash).
- [Validate Payment] If valid, order continues; else, customer retries or cancels [Cancel Order].
- [Accept Order] Restaurant confirms order; else order is rejected [Cancel Order].
- [Order Confirmed] System notifies customer and restaurant.
- [Cancel Order] Order is canceled and customer notified.

**Alternate flows**

- [Backend Fails] App/server is down or slow. Customer retries again.
- [Out of stock] One or more items become unavailable at checkout. Customer replaces items or removes them and repeats [Checkout].
- [Price Changed] Menu price updated during checkout. System requests customer confirmation before proceeding.
- [Out of Delivery Zone] Address is outside coverage. System suggests pickup or alternate restaurants.

## 2. Process Payment

A customer wants to pay for an order. Payment can only be processed if details are correct and funds are available. Otherwise, the payment fails.

Scenario/Story: Payment succeeds
    Preconditions: Customer has items in the cart and proceeds to checkout.

**Main Flow**

1. The customer enters payment details [Enter Payment].
2. The system submits details to the payment gateway [Send Request].
3. The gateway verifies funds [Verify Funds].
4. The payment succeeds [Confirm Payment].

**Subflows**

- [Enter Payment] Customer selects method (card, wallet, cash).
- [Send Request] System securely sends details to gateway
- [Verify Funds] Gateway checks balance and validity. If okay, proceed [Confirm Payment]; else [Payment Failed].
- [Confirm Payment] Payment success, order placed.
- [Payment Failed] Payment rejected, customer retries or cancels.

**Alternative Flows**

- [Invalid Card] Card number/expiry/CVV invalid. Prompt correction and retry [Enter Payment].
- [Insufficient Funds] Funds not available. The customer chooses another method or reduces order.
- [Gateway Timeout] No response from gateway. System retries; if still failing, show failure and preserve cart.

## 3. Assign Delivery Driver

The system needs to assign a driver for the order. A driver is assigned if one is available nearby. Otherwise, the order is delayed or canceled.

Scenario/Story: Driver assigned successfully
    Preconditions: The restaurant has accepted the order.

**Main Flow**

1. The system searches for available drivers [Find Driver].
2. The nearest driver is notified [Notify Driver].
3. The driver accepts an assignment [Driver Accept].
4. The driver is linked to the order [Assign Success].

**Subflows**

- [Find Driver] The system checks availability.
- [Notify Driver] Driver receives request notification.
- [Driver Accept] If accepted, proceed [Assign Success]; else retry [Find Driver].
- [Assign Success] Driver assigned to order.
- [No Driver] If no drivers available after retries, order canceled.

**Alternative Flows**

- [Driver Rejects] Notified driver declines. System retries [Find Driver].
- [No Driver Available] After N retries, System informs customers and offers to cancel or extend ETA.
- [Driver Timeout] Driver does not respond in time. Request is re-queued to the next driver.
- [Driver App Offline] Driver loses connectivity. System reassigns after grace period.

## <u>4. Update Order Status</u>

The restaurant needs to update the order progress. The order status is updated as *received, prepared, ready*.

Scenario/Story: Order prepared successfully
Preconditions: The restaurant has accepted the order.

**Main Flow**

1. Kitchen staff confirm receipt [Order Received].
2. Kitchen staff prepare food [Prepare Food].
3. Kitchen staff mark ready [Order Ready].
4. System notifies customer [Notify Customer].

**Subflows**

- [Order Received] System updates status.
- [Prepare Food] Staff begin cooking.
- [Order Ready] Staff confirm readiness.
- [Notify Customer] Customer is updated.

**Alternative Flows**

- [Auto-Cancel Timeout] Restaurant fails to acknowledge [Order Received] within SLA. Order auto-cancels and refunds issued.
- [Kitchen Delay] Prep time exceeds estimate. System updates ETA and notifies customer.

## <u>5. Track Delivery in Real Time</u>

A customer wants to track the delivery driver. The customer can only track if the driver has picked up the order and GPS is enabled.

Scenario/Story: Customer tracks delivery
Preconditions: Order is picked up.

**Main Flow**

1. The driver enables GPS [Enable GPS].
2. The driver's location is updated [Update Location].
3. The system shares location with customer [Share Location].
4. The customer views a live map [Track Map].

**Subflows**

- [Enable GPS] Driver app starts sending coordinates.
- [Update Location] Location sent at intervals.
- [Share Location] System relays updates.
- [Track Map] Customer views driver's route.

**Alternative Flows**

- [GPS Unavailable] GPS off or blocked. System shows the last known position and periodic status messages.
- [Driver Delayed] Traffic or incident slows progress. ETA recalculated and shared.

## 6. Manage Restaurant Menu

Restaurant owners update their menu. Menu updates succeed only if changes are valid (e.g., prices not negative).

Scenario/Story: Owner updates menu item
    Preconditions: Owner is logged into the restaurant dashboard.

**Main Flow**

1. The owner selects the menu option [Open Menu].
2. The owner edits item details [Edit Item].
3. The system validates input [Validate Input].
4. The system updates menu [Update Success].

**Subflows**

- [Open Menu] Owner accesses dashboard.
- [Edit Item] Owner modifies details (price, availability).
- [Validate Input] If valid, continue [Update Success]; else show error [Error Message].
- [Update Success] Menu updated for customers.
- [Error Message] Owner notified of invalid input.

**Alternative Flows**

- [Invalid Entry] Negative price/malformed field detected. System rejects and highlights fields.
- [Server Error] Save fails. System retries and offers to save draft.
- [Unauthorized Access] Insufficient permissions. Audit log created and access blocked.
- [Catalog Conflict] Concurrent edits detected. System requests merge/overwrite choice.

## 7. Handle Complaints/Refunds

A customer raises a complaint. A resolution is reached if the issue is valid. Otherwise, the complaint is closed without refund.

Scenario/Story: Customer complaint resolved with refund
Preconditions: Order is completed.

**Main Flow**

1. The customer submits a complaint [Submit Complaint].
2. Support reviews complaint [Review Complaint].
3. Support contacts restaurant/driver [Contact Party].
4. Support resolves issue [Resolve Issue].

**Subflows**

- [Submit Complaint] Customer describes issue.
- [Review Complaint] Support checks details.
- [Contact Party] Support asks restaurant/driver for info.
- [Resolve Issue] If valid, issue refund [Issue Refund]; else close [Close Complaint].
- [Issue Refund] Refund credited to customer.
- [Close Complaint] Complaint closed without refund.

**Alternative Flows**

- [Invalid Complaint] Evidence insufficient or outside policy. Close with explanation.
- [Duplicate Complaint] Merge with existing case; keep one resolution thread.
- [Partial Refund] Some items affected; issue partial credit.

## 8. Rate and Review Order

Customer rates and reviews order. Review submitted if order is completed.

Scenario/Story: Customer submits review
Preconditions: Order marked as delivered.

**Main Flow**

1. The customer opens the rating option [Open Rating].
2. The customer gives stars/comments [Submit Review].
3. The system saves reviews [Save Review].
4. The restaurant/driver views feedback [Share Feedback].

**Subflows**

- [Open Rating] Option appears after delivery.
- [Submit Review] Customer enters input.
- [Save Review] Review stored in database.
- [Share Feedback] Feedback sent to restaurant/driver.

**Alternative Flows**

- [Review Window Expired] Submission after allowed window. System blocks and informs customer.
- [Spam/Abuse Detected] Toxic or promotional content flagged; queued for moderation.
- [Edit Review] Customer edits within allowed window; system stores revision history.
- [Submission Error] Network/API error. Save draft locally and retry.

## 9. Monitor Business Performance

Business owners want to see reports. Reports are generated if data exists.

Scenario/Story: Owner checks sales dashboard
Preconditions: Owner has a registered business account.

**Main Flow**

1. The owner logs into dashboard [Login].
2. The owner requests analytics [Request Report].
3. The system gathers data [Fetch Data].
4. The system displays results [Show Report].

**Subflows**

- [Login] Owner authenticated.
- [Request Report] Owner chooses timeframe/metrics.
- [Fetch Data] Data retrieved from database.
- [Show Report] Report displayed visually.

**Alternative Flows**

- [Data Delay] Reports delayed due to processing lag. Data may not be up-to-date.
- [Access Denied] Unauthorized login attempt; blocks access.
- [Filter Error] Invalid date range/metric selection; prompt to correct.

## 10. Manage Driver Payments

Drivers want to receive earnings. Payments are processed if records exist and payout is scheduled.

Scenario/Story: Driver receives weekly payment
Preconditions: Driver completed deliveries within payment cycle.

**Main Flow**

1. The system calculates driver earnings [Calculate Earnings].
2. The system deducts platform fees [Deduct Fees].
3. The system initiates payout [Initiate Payout].
4. The driver receives funds [Receive Payment].

**Subflows**

- [Calculate Earnings] Earnings = delivery fees – commission.
- [Deduct Fees] Platform fees deducted.
- [Initiate Payout] Payment triggered via gateway.
- [Receive Payment] Driver notified of credit.

**Alternative Flows**

- [Bank Reject] Destination account invalid; hold payout and request new details.
- [Tax Hold] Regulatory requirement detected; partial payout withhold applied.
- [Chargeback Adjustment] Recent reversals reduce balance; notify driver and adjust amount.