1. Create a list of stakeholders (e.g., Admin, Staff, Customer, etc.)

Primary Stakeholders: Direct Users

Customer: Places orders, pays for items, receives notifications.
Staff (baristas, kitchen, cashiers): Prepares items, updates order status, and interacts with inventory.
Admin: Manage user roles, permissions, inventory, menu items, and system settings, and monitor reports.

Secondary Stakeholders: Support and Business-oriented

Instructors (Professors as CTOs/POs): Define high-level requirements, set grading and constraints.
Teaching Assistants (as Managers): Manage teams, assign tasks, and evaluate progress.
Developers: Build, test, and deliver WolfCafe features.
IT / Security Team: Provide infrastructure, authentication, and enforce security.
Marketing Team: Create promotions, loyalty programs, and campaigns to attract customers.
Customer Support: Handle complaints, refunds, and order issues.
Investors: Set business rules, pricing models, promotions, and oversee growth metrics.

External Stakeholders:

Suppliers, Vendors(Community, Local): Provide ingredients and goods.
Regulatory Authorities: Ensure compliance with food safety, labor laws, payment, and other regulations.
Third-Party Service Providers: Notifications (SMS/email), mapping services, external APIs, and payment gateway providers.

Tertiary / Indirect Stakeholders:
Competitors: Influence market dynamics and feature expectations.
Local Community: Indirectly benefits from reliable service and job opportunities.

Power-Interest Grid:

Power / Interest	**High Interest**	**Low Interest**
High Power	Admins, Instructors (CTO), TAs (Managers), Investors, Customers	IT/Security, Regulatory Authorities, Marketing Team
Low Power	Staff, Developers, Customer Support	Third-Party Service Providers, Suppliers, Vendors(Community, Local), Local Community, Competitors

2. Identify stakeholder biases:

Stakeholder Biases:

- **Customers**: low cost, speed, convenience, reliability.
- **Staff (baristas, kitchen, cashiers)**: efficiency, low complexity, minimal errors.

- **Admins**: control, accuracy, reporting, strict permissions.
- **Instructors**: ambitious scope, educational value, innovation.
- **Teaching Assistants (Managers)**: fairness, workload balance, and timely progress.
- **Developers**: feasibility, code maintainability, and limited workload.
- **IT / Security Team**: stability, authentication, strong security, compliance.
- **Marketing Team**: engagement, campaigns, promotions, customer growth.
- **Customer Support**: customer satisfaction, complaint resolution, and refunds.
- **Investors**: profitability, growth metrics, ROI, pricing models.
- **Suppliers / Vendors**: predictable demand, timely payments, steady orders.
- **Regulatory Authorities**: compliance with food safety, labor, and payment laws.
- **Third-Party Providers**: API stability, service uptime, integration reliability.
- **Competitors**: market differentiation, feature benchmarking, and industry pressure.

Clash-Irrelevant Matrix:

	S1	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**	**S9**	**S10**	**S11**	**S12**
S1 Customer	—	Clash		Clash		Clash	Clash	Clash		Clash		Clash
S2 Staff	Clash	—	Clash						Clash		Clash	Clash
S3 Admin		Clash	—			Clash	Clash			Clash		Clash
S4 Instructor	Clash			—	Clash	Clash	Clash					
S5 TAs				Clash	—	Clash	Clash					
S6 Dev	Clash		Clash	Clash	Clash	—	Clash	Clash		Clash		Clash
S7 IT/Sec	Clash		Clash	Clash	Clash	Clash	—					Clash
S8 Marketing	Clash					Clash		—	Clash	Clash	Clash	Clash
S9 Cust. Support		Clash						Clash	—			Clash
S10 Investors	Clash		Clash			Clash		Clash		—		Clash
S11 Suppliers / Vendors		Clash						Clash			—	Clash
S12 Regulator	Clash	Clash	Clash			Clash	Clash	Clash	Clash	Clash	Clash	—

Explanation of Clash:
- **Customers - Staff:** Customers want high customization, staff prefer simple operations.
- **Customers - IT/Security:** Customers demand quick checkout, IT insists on strong authentication.

- **Customers - Investors:** Customers want discounts, investors want higher margins.
- **Staff - Admin:** Staff want flexible adjustments, admins enforce centralized control.
- **Instructors - Developers:** Instructors push ambitious scope, developers face resource constraints.
- **Marketing - Investors:** Marketing runs promotions, investors resist margin cuts.
- **Regulators - Developers:** Regulators demand compliance, developers want fast iteration.

Explanation of Irrelevance:

- **Customers - Admin**: Customers care about order status, not the admin's detailed revenue reports.
- **Staff - Developers**: Staff focus on day-to-day operations; developers care for code and deadlines.
- **Instructors - IT/Security**: Instructors set educational scope, while IT enforces technical scope.

3. Comment on prompt crafting:

1. Stakeholder Identification (Q1)
Zero-shot version
Strengths: Clear grouping (Primary, Management, Organizational, Indirect). Easy to skim. Good "final stakeholder list" at the end.
Weaknesses: More descriptive than analytical. It names stakeholders but doesn't explain why each one matters, or how you derived them. Reads like brainstorming notes rather than a polished assignment.

Careful prompt version
Strengths: Structured stakeholder register (role, why stakeholder, how identified). This is exactly what professors usually expect - it shows reasoning, not just listing. Includes lifecycle stakeholders (future devs), regulators, and negative/indirect ones.
Weaknesses: Longer, denser, might be harder to quickly skim in presentation slides. But for a homework-style answer, it's much stronger.
Verdict: Zero-shot works if you need a brainstorming list. Careful prompting produces a more complete, graded-assignment-quality answer.

2. Stakeholder Biases & Clashes (Q2)
Zero-shot version
Strengths: Reads naturally, gives 5 well-explained conflicts (customer vs staff, customer vs admin, etc.). Easy to understand.
Weaknesses: Narrative style only; no structured matrix/table. Doesn't align with the typical "requirements engineering" homework format.

Careful prompt version
Strengths: Provides matrix + explanatory notes, directly mirroring the Functional Requirements Interactions example you referenced. Explicitly marks positive/negative interactions. Feels more "engineered" and systematic.
Weaknesses: Heavier cognitive load to parse. Less conversational, more rigid.
Verdict: Zero-shot gives a nice explanatory narrative. Careful version matches assignment expectations (matrix first, then notes).

3. Overall Comparison
Zero-shot = Idea generation/brainstorming (good for warm-up or early drafts).
Careful prompt = Deliverable / homework-ready (structured, systematic, more persuasive for grading).
If this were an actual course submission:
Q1 → The careful prompt version is stronger (explicit roles, rationale, lifecycle view).

Q2 → The careful prompt version is also stronger because of the structured matrix, though you could combine it with the narrative clarity of the zero-shot.

4. Write at least 10 Use Cases(UC) (≈5 pages total):

UC-01: Customer Places a Pickup Order

Preconditions

- Customer is authenticated (or opts for guest checkout if allowed).
- Menu Items exist, and at least one is in stock.

Main Flow

1. Customer opens Order screen.
2. System shows Items (and Recipes if enabled), prices, stock/lead times.
3. Customer adds one or more Items to the cart, selects options (size, milk, etc.).
4. The system calculates the subtotal, tax, and estimated ready time.
5. The customer chooses the payment method and confirms the pickup name.
6. The system authorizes payment with the Payment Processor.
7. On success, the System creates an Order with status **PLACED** and a queue position.
8. System reserves inventory for the Order.
9. The Notification Service sends order confirmation & ETA.

Subflows

- **SF-A (Options & modifiers):** Customer customizes Item (e.g., extra shot). Price updates dynamically.
- **SF-B (Promo):** Customer applies valid promo; System recalculates totals.
- **SF-C (Loyalty):** If loyalty is enabled, the System accrues points post-payment.

Alternative Flows

- **AF-1 (Out of stock at add):** System flags item; suggests substitutes.
- **AF-2 (Payment failed):** Show error, let Customer retry/change method.
- **AF-3 (Auth hold but order create fails):** System voids auth; shows error.

UC-02: Staff Accepts & Fulfills an Order

Preconditions

- At least one **PLACED** order exists.
- Staff is authenticated with the **staff** role.

Main Flow

1. Staff opens Order Queue.
2. System lists orders by SLA priority and time.
3. Staff selects an order and taps **Start**.
4. System sets status **IN_PROGRESS** and logs start time.
5. Staff prepares all line items.
6. Staff taps **Ready**.
7. System sets status **READY_FOR_PICKUP** and records finish time.
8. The Notification Service sends "Ready for pickup".

Subflows

- **SF-A (Batch prep):** Staff starts multiple orders; System tracks parallel in-progress orders.
- **SF-B (Station routing):** Items auto-routed to espresso, cold bar, etc. **Alternative Flows**

- **AF-1 (Missing ingredient mid-prep):** Staff marks item unavailable → System offers substitution flow (see UC-06) or partial refund.
- **AF-2 (Device offline):** System caches status updates and syncs when online.

UC-03: Customer Picks Up Order

Preconditions

- Order is **READY_FOR_PICKUP**.
- Pickup code/QR available.

Main Flow

1. Customer arrives and shows pickup name/QR.
2. Staff scans the QR or searches the order.
3. System validates and shows order contents.
4. Staff hands over items; taps **Picked Up**.
5. System sets status **COMPLETED** and closes the ticket. **Subflows**

- **SF-A (Locker/Smart shelf):** Customer scans QR at locker; System unlocks bin, sets **COMPLETED**.

Alternative Flows

- **AF-1 (Wrong customer):** Name/QR mismatch → deny; log attempt.
- **AF-2 (No-show timeout):** After the grace period, the System flags **ABANDONED** and triggers refund/hold release per policy.

UC-04: Customer Cancels or Edits Order (Pre-Prep)

Preconditions

- Order status is **PLACED** (not yet **IN_PROGRESS**).

Main Flow

1. Customer opens Order details.
2. Customer selects **Edit** or **Cancel**.
3. The system shows the change/cancel policy.
4. For **Edit**: Customer updates items/options → System re-prices.
5. Customer confirms; Payment Processor adjusts charge (capture/void/partial).
6. System updates order and sends confirmation.

Subflows

- **SF-A (Price decrease):** System partially refunds the difference.
- **SF-B (Price increase):** System performs incremental authorization.

Alternative Flows

- **AF-1 (Prep already started):** System disallows edit/cancel; offers support request.

UC-05: Admin Manages Roles & Access

Preconditions

- Admin authenticated with the **admin** role.

Main Flow

1. Admin opens **User Management**.
2. Searches for a user by email/ID.
3. Assigns role (**customer**, **staff**, **admin**) and site(s).
4. System persists RBAC settings and logs audit entries.
5. Changes take effect immediately.

Subflows

- **SF-A (Bulk upload):** Admin uploads CSV to assign many roles; System validates and applies.

Alternative Flows

- **AF-1 (Conflict):** Attempt to demote the last remaining admin → System blocks and warns.

<u>UC-06: Handle Out-of-Stock During Prep (Substitution/Refund)</u>

Preconditions

- Order status **IN_PROGRESS**.
- A required ingredient or item becomes unavailable.

Main Flow

1. Staff taps **Item Issue** on the order line.
2. System proposes substitutions based on rules (e.g., “2% milk → whole”).
3. Staff selects option; System calculates price delta.
4. Customer is prompted via app/SMS to approve substitution or refund line.
5. On approval, System updates order and totals; continues prep.

Subflows

- **SF-A (Auto-approve rules):** For minor changes (e.g., milk type), the System can auto-approve if user settings allow.

Alternative Flows

- **AF-1 (No response):** After timeout, default to refund or cancel line per policy.
- **AF-2 (Customer rejects):** System refunds the line and continues the remaining items.

<u>UC-07: Inventory Management & Auto-Depletion</u>

Preconditions

- Ingredients/Items exist in the catalog with stock units and recipes (if used).

Main Flow

1. Admin/Staff views **Inventory**.
2. The system shows quantities on hand, reserved, and reorder thresholds.
3. On order placement, the System **reserves** the required quantities.
4. On order completion, the System **depletes** actual usage.
5. When the threshold is reached, the System flags **Low Stock** and can create a replenishment task.

Subflows

- **SF-A (Receive stock):** Staff records delivery; System increases on-hand, clears low stock.
- **SF-B (Recipe to Item mapping):** Depletion uses recipe BOM for composite items; simple items deplete directly.

Alternative Flows

- **AF-1 (Variance):** Staff records waste/spillage; System adjusts and logs variance.

<u>UC-08: Catalog Management (Items & Recipes)</u>

Preconditions

- Admin authenticated.

Main Flow

1. Admin opens the **Catalog**.
2. Creates/edits an **Item** (e.g., “Bottled Water”) or a **Recipe-backed Item** (e.g., “Latte”).
3. Sets price, tax class, availability schedule, options/modifiers.
4. Links recipe/ingredient usage where applicable.
5. Publishes changes; System versions and activates new catalog.

Subflows
- **SF-A (A/B price test):** Admin defines test cohort; System serves variant pricing to selected users.

Alternative Flows
- **AF-1 (Conflicting SKUs):** Duplicate SKU detected → System prevents publish.

<u>UC-09: Refunds, Voids, and Disputes</u>

Preconditions
- An order exists in **PLACED/IN_PROGRESS/READY/COMPLETED**.

Main Flow
1. Staff/Admin opens the order.
2. Chooses **Refund** (full/partial) or **Void** (if uncaptured).
3. The system sends a request to the Payment Processor.
4. On success, System updates order financials and status notes.
5. The Notification Service informs the customer.

Subflows
- **SF-A (Item-level refund):** Select specific line(s); System recalculates tax and loyalty.

Alternative Flows
- **AF-1 (Processor error):** System records failure; prompts retry or escalation.

<u>UC-10: Order Throttling & ETA Calculation</u>

Preconditions
- The System has capacity rules (max concurrent drinks/food, per-station SLA).

Main Flow
1. Customer opens menu; System fetches current load.
2. System computes dynamic ETA and available pickup slots.
3. Customer selects a slot, adds items, and checks out.
4. System books the slot and updates station workloads.

Subflows
- **SF-A (Prep time model):** ETA based on historical durations per item and current queue length.

Alternative Flows
- **AF-1 (Capacity reached):** System hides near-term slots; offers later times.

<u>UC-11: Operational Reports & Audits</u>

Preconditions
- Transactions/orders exist; Admin or Manager role.

Main Flow
1. Admin opens **Reports**.

2. Selects report (sales by hour, item performance, waste, SLA adherence, staff productivity).
3. System aggregates data with role-based access.
4. Admin exports CSV/PDF or schedules email delivery.

Subflows

- **SF-A (Drill-down):** Click KPI → view underlying orders, inventory movements, refunds.

Alternative Flows

- **AF-1 (PII restricted):** If the user lacks privilege, System masks customer identifiers.