1. Stakeholders

1.1. Stakeholder List

Stakeholder	**Group**
Customers (students, faculty)	Users (direct day-to-day interaction)
WolfCafe Staff	Users
Administrators/Managers	Users, Domain Experts
Accessibility Users	Users, Regulators/Legal (indirect)
WolfCafe Business Owner(s)	Clients/Sponsors
University (dining services)	Clients/Sponsors, Indirect
CTOs/Product Owners (instructors)	Clients/Sponsors, Domain Experts
App Development Team	Developers/Engineers
Future Developers	Developers/Engineers, Indirect
IT Support/Operations	Developers/Engineers
Regulatory Bodies (food, IT)	Regulators/Legal
Accessibility Standards Bodies	Regulators/Legal
Competitors/Alternative Vendors	Indirect/Negative

1.2. Stakeholder Biases

- Users
  - Customers (students, faculty, staff ordering food) → Want fast service, affordable prices, convenience, minimal barriers to ordering.
  - WolfCafe Staff (baristas, prep, cashiers) → Want organized, accurate, manageable workflow, with a UI that is fast and simple under pressure.
  - Administrators/Managers (store/shift leads) → Want smooth operations, accurate tracking, maximum revenue, and oversight of staff and sales.
  - Accessibility Users (subset of customers/staff) → Want accessible, readable, non-flashy UIs, compliant with ADA/WCAG.
- Clients/Sponsors
  - WolfCafe Business Owner(s) → Want profitability, customer satisfaction, long-term adoption.
  - University (Dining Services) → Want a good reputation, alignment with campus systems, compliance.
  - CTOs/Product Owners (instructors) → Want feature completeness, adherence to good engineering practices, demonstrable learning outcomes.

- Domain Experts
    - (Same as above: Admins, Instructors → bring deep knowledge of operations/requirements.)
- Developers/Engineers
    - Dev Team → Want to experiment with features/tech, build impressive UI/UX, ship a working product that looks modern.
    - Future Developers → Want clear documentation, maintainable codebase, no excessive technical debt.
    - IT Support/Operations → Want reliability, ease of deployment, minimal maintenance overhead.
- Regulators/Legal
    - Regulatory Bodies (food safety, data security, IT policy) → Want compliance with health codes, security policies, data handling rules.
    - Accessibility Standards Bodies (ADA, WCAG) → Want accessible interfaces, no exclusion of disabled users.
- Indirect/Negative
    - Competitors (other food vendors) → Don't want WolfCafe to succeed → may highlight flaws.
    - Future Developers (inherit system) → High interest later, frustrated if poor design choices are made now.

1.3. Stakeholder Clashes

- **Customers vs. Staff** : Customers want fast service; staff want manageable workflow and fewer errors. *Example: Customers ordering highly customized drinks slows staff down and clutters UI.*
- **Customers vs. Administrators/Business Owners**: Customers want affordability and minimal logins; admins/owners want higher revenue, loyalty tracking, and strong security. *Example: Admins push loyalty programs requiring logins; customers prefer guest checkout.*
- **Staff vs. Dev Team / IT Ops**: Staff want a simple, reliable UI; dev team may build flashy features; IT Ops want stability. *Example: A "modern" animated order board looks cool but crashes under heavy load, frustrating staff.*
- **Accessibility Users vs. Dev Team**: Dev team favors trendy UI (animations, colors); accessibility users need readable, non-flashy design. *Example: High-motion effects impair navigation for visually impaired users.*
- **University/Regulators vs. Customers**: University and regulators want strict compliance (auth, food safety logs, data protection). Customers want speed and convenience. *Example: Multi-step login slows down ordering → customers frustrated.*

2. Comments on Prompt Crafting

2.1. Zero-Shot Prompting

- **Definition**: This method gives the LLM a simple instruction without context, examples, or formatting rules. *Example: "Write 10 use cases for a food delivery system."*
- **Pros**: It's fast, straightforward, and requires minimal upfront effort.
- **Cons**: Generic, incomplete, and poorly structured output; Unlikely to meet WolfCafe-specific requirements (e.g., tip percentages, user roles, CoffeeMaker integration)

2.2. Careful Prompting

- **Definition**: This method provides the LLM with explicit instructions, strict formatting requirements, and one or more high-quality examples (e.g., CoT, Prompt Chaining). It is essentially a "script" that guides the model's reasoning and output. *Example*: *"Generate 10 use cases for the WolfCafe system. Each must include: Name, Preconditions, Main Flow, Subflows, and Alternative Flows. Example: [sample]."*
- **Pros**:
  - **High Precision:** Ensures outputs strictly adhere to requirements (e.g., differentiating staff vs. admin permissions).
  - **Structural Consistency**: Produces standardized outputs suitable for direct integration into requirement documents.
  - **Enhanced Quality:** Well-crafted examples improve the academic rigor, completeness, and professional tone of outputs.
- **Cons**:
  - **High Preparation Cost:** Requires significant upfront effort to analyze requirements and engineer detailed prompts.
  - **Cognitive Demand:** Effective prompt design depends on deep knowledge of system roles, features, and project context.
  - **Error Replication:** Flawed instructions or examples propagate directly into outputs ("garbage in, garbage out").
  - **Token Expense:** Long prompts and structured outputs increase API usage and cost.

3. Use Cases

To see which stakeholders would be affected by each use case, refer to the Appendix.

3.1. UC1 Customer Places an Order

This use case describes how a Customer browses the WolfCafe menu, builds a cart, and completes the checkout process, resulting in a confirmed order that is passed to Staff for fulfillment

3.1.1. Preconditions

- The Customer is authenticated in the WolfCafe system (via account or guest login, if allowed).
- At least one item has already been added to the shopping cart.

3.1.2. Main Flow

The Customer proceeds to checkout. On the summary screen, they may [Edit Cart], [Apply Coupon], add a tip, and select payment before submitting. The system validates the order: failed transactions trigger [Payment Declined], unavailable items trigger [Item Out of Stock]. Successful orders generate a confirmation and unique number. Inactivity causes [Session Timeout], clearing the cart.

3.1.3. Subflows

- **[Edit Cart]**: Add, remove, or update items. If unavailable, triggers [Item Out of Stock].
- **[Apply Coupon]**: Enter promo code; valid codes apply discount, invalid ones show inline error for retry.

3.1.4. Alternative Flows

- **[Payment Declined]:** Payment rejected; system notifies and returns to payment selection.
- **[Item Out of Stock]:** Removes unavailable items, updates totals, and prompts confirmation.
- **[Session Timeout]:** Inactivity logs out user, clears cart, and requires re-login.

3.2. UC2 Fulfill Customer Order

This use case describes how Staff process and complete a customer order once it has been placed.

3.2.1. Preconditions

- The staff member is logged into the WolfCafe system with Staff privileges.
- At least one order exists in "Pending" status.
- The required ingredients and equipment are available in the café.

3.2.2. Main Flow

A staff member [Claims Order] from the queue. They [View Order Details], prepare items, and the system [Adjusts Inventory]. If ingredients are low, a [Low Stock Alert] is issued. Staff may [Contact Customer] if issues arise; unresolved cases lead to [Order Cancellation]. When done, the order is marked "Ready for Pickup" (UC8).

3.2.3. Subflows

- **[View Order Details]:** Shows items, modifiers, and notes.
- **[Claim Order]:** Locks order to staff; prevents duplicates.
- **[Adjust Inventory]:** Deducts ingredients; triggers [Low Stock Alert] if insufficient.

3.2.4. Alternative Flows

- **[Low Stock Alert]:** Flags shortages; may allow substitution.
- **[Contact Customer]:** If instructions can't be fulfilled; unresolved → [Order Cancellation].
  [Order Cancellation]: Logs reason, updates status, notifies customer.

3.3. UC3 Add Menu Item

This use case describes how an Admin creates a new menu item in the WolfCafe system, including assigning metadata, recipe details, and pricing so that the item becomes available to customers.

3.3.1. Preconditions

- The Admin is authenticated in the system with Admin privileges.
- All required ingredients for the new recipe already exist in the inventory.

3.3.2. Main Flow

An Admin adds a new menu item by navigating to the management interface to [Enter Details] (name, price) and [Define Recipe] (ingredients, quantities). The system validates all data, preventing submission with [Invalid Input] or a [Duplicate Item Name]. The Admin may save the new item, making it available to customers, or [Cancel Creation] to discard all changes.

3.3.3. Subflows

- **[Enter Details]**: The Admin provides the item's name, description, and price. Invalid entries (e.g., empty name or negative price) trigger [Invalid Input].
- **[Define Recipe]**: The Admin selects ingredients and specifies their required quantities. If an invalid or unavailable ingredient is selected, the Admin must adjust before saving.

3.3.4. Alternative Flows

- **[Duplicate Item Name]:** Item name already exists; system rejects and returns to [Enter Details].
- **[Invalid Input]:** Invalid values (e.g., blank name, negative price); system flags errors and returns for correction.
- **[Cancel Creation]:** Admin cancels or exits; system discards data and no item is created.

3.4. UC4 Manage Customer Profile

This use case describes how a Customer updates their account information, including personal details and saved payment methods, to keep their profile current and secure.

3.4.1. Preconditions

- The Customer is authenticated in the WolfCafe system with an active session.

3.4.2. Main Flow

The Customer goes to account settings to [Edit Personal Info] or [Manage Payment Methods]. The system validates changes, blocking updates for [Password Mismatch] or [Invalid Input]. The Customer can save or [Cancel Edit].

3.4.3. Subflows

- **[Edit Personal Info]:** Edit name, email, or password. Password changes require current password; failure → [Password Mismatch].
- **[Manage Payment Methods]:** Add, update, or remove cards. Invalid formats trigger [Invalid Input].

3.4.4. Alternative Flows

- **[Invalid Input]**: Data is invalid (e.g., wrong email, expired card); system flags issue for correction.
- **[Password Mismatch]**: Current password incorrect during change; update blocked.
- **[Cancel Edit]**: Customer cancels or exits; system discards edits and keeps last saved data.

3.5. UC5 View Order History

This use case describes how a Customer reviews their past orders in the WolfCafe system and, if desired, re-orders items for convenience.

3.5.1. Preconditions

- The Customer is authenticated in the WolfCafe system.
- At least one completed order exists in the system (otherwise see [No Order History]).

3.5.2. Main Flow

The Customer accesses their order history, which displays a message if there is [No Order History][System Error]. They can select a past order to [View Order Details] and use the [Re-order] function to add all items to their current cart. The system automatically removes any unavailable items from the cart, triggering [Item Out of Stock] and notifying the customer of the change.

3.5.3. Subflows

- **[View Order Details]**: Shows items and details (e.g. quantities, modifiers, and prices).
- **[Re-order]**: Adds items from a past order to the cart; unavailable items trigger [Item Out of Stock].

3.5.4. Alternative Flows

- **[No Order History]**: Shown if the customer has no past orders.
- **[Item Out of Stock]**: Triggered during [Re-order] if one or more items from the historical order are unavailable. Removes unavailable items from re-order and updates the cart.
- **[System Error]**: Shown if order history retrieval fails; suggests retry later.

3.6. UC6 Staff Manages Inventory

This use case describes how a Staff member manages stock levels of ingredients and supplies within the WolfCafe system.

3.6.1. Preconditions

- The staff member is authenticated with Staff privileges.
- Ingredients and supply items (e.g., milk, espresso beans, cups) already exist in the inventory database.

3.6.2. Main Flow

A staff member uses the inventory interface to [Search/Filter Inventory] and select an item. They [Update Stock] quantities, with the system validating the entry and preventing [Invalid Input]. Saving a change can trigger a [Low Stock Alert] if the quantity falls below a predefined threshold. The staff can also [Cancel Update] to discard any changes.

3.6.3. Subflows

- **[Search/Filter Inventory]**: Staff uses a search bar or filters to locate the desired item.
- **[Update Stock]**: Staff enters the new quantity, and the system validates it. If the value is non-numeric or negative, [Invalid Input] is invoked.

3.6.4. Alternative Flows

- **[Low Stock Alert]**: Triggered when the saved stock level is below a threshold; the system flags the item for reorder.
- **[Invalid Input]**: Triggered if entered values are invalid; the system rejects the update and prompts correction.
- **[Cancel Update]**: Triggered if the staff member cancels before saving; no changes are stored.

3.7. UC7 Generate Sales Report

This use case describes how an Admin generates a sales report to analyze revenue and operational performance.

3.7.1. Preconditions

- The Admin is authenticated with reporting privileges.
- At least one order with sales data exists in the system (otherwise [No Data] applies).

3.7.2. Main Flow

An Admin generates a sales report by specifying a time frame [Choose Time Period] and applying optional [Select Filters]. The system displays the results, or a [No Data] message if no transactions match the criteria. The Admin can then [Export Report] to a file; a failure in this process triggers an [Export Error].

3.7.3. Subflows

- **[Choose Time Period]:** Admin selects predefined ranges (daily, weekly, monthly) or a custom date range.
- **[Select Filters]:** Admin filters results by category, staff member, or payment type.
- **[Export Report]:** Admin exports the generated report to PDF or CSV format.

3.7.4. Alternative Flows

- **[No Data]:** Triggered when no transactions match the selected period or filters.
- **[Export Error]:** Triggered if the export fails due to technical issues. The Admin is prompted to retry or select another format.

3.8. UC8 Notify Customer of Order

This use case describes how the WolfCafe system notifies a Customer when their order is ready for pickup.

3.8.1. Preconditions

- A Customer has successfully placed an order (UC1).
- Staff has marked the order as Ready for Pickup (UC2).

3.8.2. Main Flow

When staff marks an order as ready for pickup, the system automatically triggers a [Send Notification] for the customer to [View Notification]. A [Notification Failure] prompts the system to retry or use a fallback communication channel.

3.8.3. Subflows

- **[Send Notification]**: System composes and queues notification messages.
- **[View Notification]**: Customer receives and views the notification details.

3.8.4. Alternative Flows

- **[Notification Failure]**: Triggered when a delivery channel fails. The system retries and/or uses fallback channels (e.g., switching from SMS to in-app).

3.9. UC9 Provide Customer Feedback

This use case describes how a Customer provides feedback on a completed order.

3.9.1. Preconditions

- The Customer is authenticated in the system.
- The order has been completed and marked as delivered/picked up.

3.9.2. Main Flow

After an order is completed, a customer can [View Order Details] in their history and leave feedback. They must [Enter Rating] (1-5 stars) and can optionally [Enter Comment]. The system rejects an [Empty Submission] if the rating is missing and stores the feedback for an Admin to [View Feedback Report]. The customer can [Cancel Feedback] at any point before submission.

3.9.3. Subflows

- **[Enter Rating]:** Customer selects a 1–5 star rating; missing input triggers [Empty Submission].
- **[Enter Comment]:** Customer adds optional written feedback.
- **[View Feedback Report]:** Admin later reviews aggregated results.

3.9.4. Alternative Flows

- **[Empty Submission]**: Triggered when a rating is missing; the system prompts correction.
- **[Cancel Feedback]**: Triggered if the Customer cancels or exits before submission.

3.10. UC10 Deploy System Update

This use case describes how the Dev Team deploys a new version of the WolfCafe system during a scheduled maintenance window.

3.10.1. Preconditions

- The Dev Team has packaged and tested the release.
- An Admin has approved a maintenance window.

3.10.2. Main Flow

During a maintenance window, the Dev Team deploys a system update. The process automatically applies database changes [Run Migration] and then executes a series of automated checks [Run Tests]. A [Migration Failure] or [Test Failure] triggers an immediate rollback to the previous stable version. An Admin is notified via [Downtime Exceeded] if the deployment surpasses its allotted time.

3.10.3. Subflows

- **[Run Migration]:** Applies database schema/data changes. Errors trigger [Migration Failure].
- **[Run Tests]:** Executes automated regression and smoke tests. Failures trigger [Test Failure].

3.10.4. Alternative / Error Flows

- **[Migration Failure]:** Triggered if migrations fail; rollback initiated, deployment halted.
- **[Test Failure]:** Triggered if automated tests fail; previous version restored.
- **[Downtime Exceeded]:** Triggered if the deployment surpasses the approved maintenance window; Admin notified.

Appendix: Use Case Traceability Matrix

Stakeholder Traceability Matrix 1: Customer & Business Operations

Use Case	Requirement	Customers	Staff	Admins/Managers	University (Deans)
UC1 - Customer Places Order	R1: Customer Ordering & Checkout	✓	✓		
UC2 - Fulfill Customer Order	R2: Order Fulfillment & Inventory Adjustment	✓	✓	✓	
UC3 - Add Menu Item	R3: Menu Management	✓	✓	✓	
UC4 - Manage Customer Profile	R4: Customer Account & Profile	✓			
UC5 - View Order History	R5: Order History and Ordering	✓			
UC6 - Staff Manages Inventory	R6: Inventory Management		✓	✓	
UC7 - Generate Sales Report	R7: Sales Reporting			✓	✓
UC8 - Notify Customer of Order	R8: Customer Notifications	✓	✓		
UC9 - Provide Customer Feedback	R9: Feedback Collection	✓		✓	
UC10 - Deploy System Update	R10: System Deployment & Updates				

This matrix is the first part of the Use Case Traceability Matrix, focusing on stakeholder groups who either directly interact with the system or have a direct impact on business operations. It maps each Use Case (UC) and its corresponding Requirement (R) to key stakeholders like Customers, Staff, Admins/Managers, and the University Dining Services. This approach ensures every system function can be traced to a specific stakeholder, validating that the system's design meets core business needs and helping to manage closely and keep satisfied stakeholders during development.

Stakeholder Traceability Matrix 2: Technology & Support

Use Case	Requirement	Business Owners	Dev Team	IT Support Ops	QA/Testing Team
UC1 - Customer Places Order	R1: Customer Ordering & Checkout				✓
UC2 - Fulfill Customer Order	R2: Order Fulfillment & Inventory Adjustment				✓
UC3 - Add Menu Item	R3: Menu Management				✓
UC4 - Manage Customer Profile	R4: Customer Account & Profile				✓
UC5 - View Order History	R5: Order History and Ordering				✓
UC6 - Staff Manages Inventory	R6: Inventory Management	✓			✓
UC7 - Generate Sales Report	R7: Sales Reporting	✓			✓
UC8 - Notify Customer of Order	R8: Customer Notifications				✓
UC9 - Provide Customer Feedback	R9: Feedback Collection				✓
UC10 - Deploy System Update	R10: System Deployment & Updates		✓	✓	✓

This table, the second part of the Use Case Traceability Matrix, maps each use case and requirement to Business Owners, Developers, IT Support, and QA. It shows who is responsible for development, deployment, maintenance, and testing, ensuring technical work aligns with business goals. It also clarifies stakeholder roles, reduces communication gaps, and supports delivery of a stable, reliable product.