

PROJ1A1 - WolfCafe System

Akash Ramanarayanan

Adam Myers

Sailesh Sridhar

Swetha Manivasagam

Stakeholders of WolfCafe

Admin (Maintenance)

Customers

Staff (Kitchen)

Suppliers

CEO/ Product owner

IT support (application developers)

Audit/legal support staff

How to identify stakeholders of a system?

Start thinking about the entire flow involved in the process of a customer getting an order. Think of all the people making an impact on the functioning of the system (every person without whom the system would suffer). Club together redundant roles.

LLM (Gemini 2.5 Pro) responses for “how to find stakeholders”?

General Process to Identify Stakeholders: A stakeholder is anyone with an interest in or who is affected by the system. The general process involves three main steps: brainstorming, classifying, and prioritizing. This process ensures all relevant individuals and groups are considered and their needs are properly addressed.

Begin by brainstorming a broad list of potential stakeholders. Think about anyone who will use, manage, fund, or be impacted by the system. Once you have a list, classify each stakeholder. A common way to classify them is by their relationship to the system (e.g., users, developers, managers, regulators). Another useful method is to categorize them based on their level of power (influence) and interest in the system. Not all stakeholders are equally important. Prioritization helps you focus your efforts on those who can make or break the project.

LLM suggested stakeholders

Parents/Visitors

Maintenance and Cleaning Staff

University Administration

Student Government/Student Union

Health and Safety Regulators

Identifying stakeholder biases

We consider possible conflicts of interest that could arise between stakeholders.

- 1) Customers vs. Admin, IT support – Unsatisfactory user interfaces

The customers might want more intuitive, aesthetic and functional user interfaces for placing and managing orders. Admins might not be able to offer extensive and expensive software solutions. IT support might find it difficult to develop and test such features as well.

2) Admins vs. IT Support – Update Frequency

Admins might frequently demand newer functionalities for the software. The IT team/ developers will want lesser updates for the sake of maintainability.

The vice versa could also be possible. The development team might want to make changes to the existing application but the admins might be hesitant. This is because the admins are used to the current implementation and a new feature/ update might be an introduction of a new learning curve to the application.

3) Admin vs. Staff – The Menu

The list of items in the menu might bring conflicting views between the kitchen staff and the Admin. The Admin would want a wider variety of options on the menu. However, the Kitchen staff would want a simpler menu for easier cooking.

4) Customers vs. Admin – Product Pricing

The Customers would want affordable products. However the admin team would want higher prices for gaining higher levels of profit.

5) Suppliers vs. Admin – Inventory orders

Suppliers may get frustrated by fluctuating sizes of supply order. Admins may find suppliers unreliable if they don't prove to be flexible.

6) Admin vs Staff – Auditing/ Logging of activities

Admin team would like the staff to do a detailed logging of all the everyday activities (orders, supplies) for the purpose of auditing and management. However, the staff would rather finish work early and laze out instead of logging each and every (staff would want simpler workflows).

How to find stakeholder biases?

To find the stakeholder biases we first brainstormed the most pressing issues related to the system. These issues ranged from menu selection to software development. This led us to finding who would be most at stake with these issues. We were then also able to see that several stakeholders may have different perspectives on these issues. We then sought help from an LLM to discover more hidden biases. An LLM allowed us to get a perspective on biases that we did not initially consider.

LLM (Gemini 2.5 Pro) responses for “how to find stakeholder biases”?

A general process involves preparing by acknowledging your own biases, gathering information through open-ended questions and diverse perspectives and analyzing findings to validate assumptions. Common biases to look for include groupthink (the desire for harmony), hindsight bias (claiming to have predicted an outcome) and overconfidence bias (being overly optimistic about judgments or estimates).

LLM suggested biases

1. A manager who believes that the only way to increase sales is through offering more fried food might ignore data from the new system that shows vegetarian options are growing in popularity.
2. The IT team might be overly confident in their ability to integrate the new cafe system with other university systems (e.g., student ID cards for meal plans). This can lead to underestimating the time and resources needed for a smooth rollout.
3. A senior administrator might have had a very positive experience with a well-known, high-tech solution in a different context (e.g., a corporate cafeteria). This positive impression could lead them to believe that the same solution is the best fit.

Prompt Crafting

Prompts have to be accurate enough for the LLM to be able to give us sufficient and relevant information.

Zero-shot prompting gives diverged answers with less details. It is useful for exploring a variety of areas and getting a quick understanding of each of them. It is also relevant when the task is relatively simple and needs to be less time-consuming.

The prompt specifies lesser examples and more direct instructions. There are thus high chances of misinterpretation of the intent. The vagueness and inconsistency might be misleading. These risks of under-engineering the prompts can be reflected in the quality of the work output.

Careful prompting gives more specific answers based on the details shared in the prompt. Prompt crafting usually takes a longer time. The responses are tailored well enough though. It is well suited for complex tasks as well. Improved consistency, relevancy and accuracy in responses is shown. There are risks involved in over-engineering of prompts.

Use Cases

Use Case 1: Creating a Recipe (Admin)

Preconditions:

The user is an admin; The admin is logged into the system and is looking at the main menu.

Main Flow:

The admin selects the "Create a Recipe" button; Provides a name for the recipe; Adds ingredients to the recipe from a dropdown that includes current items in the inventory; Inputs the quantity for each ingredient in the recipe; Inputs the price of the recipe. Selects the add button

Subflow:

- **Add recipe based off of another recipe:** The admin selects a previously defined recipe.

Alternative Flow:

- **Adding a Recipe with a duplicate name:** The admin clicks add after giving a recipe a duplicate name. The admin is then prompted with an error message.
- **Missing a required field:** The admin clicks add after not inputting a price or name. The admin is then prompted to input the required fields

- **Improper Input Formatting:** The admin inputs a negative number into the price field. The admin clicks the add button. The admin is then prompted to input a valid price

Use Case 2: Adding Inventory (Admin)

Preconditions:

The Manager restocked the store with new ingredients; The User (admin/staff) is logged into the system; The user is on the main staff page

Main Flow:

The staff selects the “add inventory” option; A page is opened, allowing the user to input various information; The staff inputs the ingredient name; The staff fills in the required information (such as quantity, etc.); The staff reviews the Cart; The staff clicks “add” to finalize the inventory; System updates the inventory with the new stock levels; The staff is sent back to the main menu.

Subflow:

- **Adding Several Different Items:**
Staff returns to the item lists to add several items at once; The staff adds several ingredients to the inventory; The staff clicks add; All items are added to the inventory.
- **Editing/deleting quantities:**
The staff wants to remove ingredients from the inventory; The staff inputs an amount of an ingredient to delete and selects the “remove” button; The staff is sent back to the home screen; The specified ingredients are removed from the inventory.

Alternative Flow:

- **Insufficient Permissions:**
The Staff does not have the required permissions to add/edit the inventory; The system denies access and escalates to the admin; Admin either provides access, or the flow ends up without updating the inventory.
- **User inputs incorrect details:**
The staff types in incorrect information, such as numeric characters in the names or negative quantities; The system throws/displays an error asking the staff to rectify the mistake.

Use Case 3: Purchase Item (Customer)

Preconditions:

The user is logged in; The user is looking at the home page; The user has a customer status.

Main Flow:

The user selects the “Create Order” option; The user is redirected to a page with all currently available recipes and items; The user selects an item to add to their cart; The order total is shown at the bottom of the webpage including sales tax; The user selects “checkout”; The user is given the option to leave a 15%, 20%, 25%, or custom tip; The user is redirected to the home screen with a prompt that their order was created

Subflow:

- **Multiple Quantities of the Same Item:**
The user adds the same item to their cart multiple times; The system updates the cart accordingly

- **Adding Several Different Items:**

The user adds several different items to the cart; The system updates the cart accordingly

- **Editing Cart Before Checkout**

The user adds several items to the cart; Before checkout, the user decides to remove items or change the quantities of an item

Alternative Flow:

- **Item Unavailable:**

The user tries to add an item that is unavailable; The system blocks the request; The system notifies the user that the item is currently unavailable

- **Checkout Cancelled:**

The user leaves the “Create Order” page before purchasing; The user's cart is cleared, and no purchase is made; The user is returned to the main menu

- **Invalid Tip Entry:**

The user selects “custom tip” while checking out; The user inputs an invalid tip (negative, non-number, etc.); The system gives a prompt asking the user to retry

Use Case 4: View Orders (Admin)

Preconditions:

The user is an admin; The admin is logged into the system; The admin is looking at the main menu.

Main Flow:

The admin selects the “View Orders” button; The admin is shown a page of all currently unfulfilled orders; The page filters the orders based upon recency that the orders were placed; Each order has details on it (items, quantities, customer, and time placed); Each order has the option to fulfill or cancel

Subflow:

- **The admin immediately exits the page:**

The admin enters the “View Orders” menu; The admin immediately selects the “Exit” button; The admin is returned to the main menu

Alternative Flow:

- **No Unfulfilled Orders Exist:**

The admin opens the orders page; The admin is displayed with the text “No unfulfilled orders”

- **Order already fulfilled by other staff**

The currently fulfilled orders is changed by another admin; The admin is warned that the current orders have changed

- **API retrieval failure**

API fails to retrieve orders; System notifies admin that “the page failed to load, try again later”

Use Case 5: Fulfilling Orders (Admin)

Preconditions: The user is an admin; The admin is logged into the system and is looking at the main menu

Main Flow: The admin selects the “View Orders” button; The admin is shown a page of all currently unfulfilled orders; The customer picks up their order; Next to the order that the admin would like to fulfill, the admin selects “fulfill order”; The order is removed from the orders page and the customer is notified of their order being fulfilled; The admin forwards the request to kitchen staff for item preparation; The admin is kept on the orders page with the new list of unfulfilled orders

Subflow:

- **Fulfilling Multiple Orders:** The admin opens the orders page; The admin selects the “fulfill order” button next to several orders; All of the fulfilled orders are removed from the page
- **Fulfilling All Orders:** The admin opens the orders page; The admin selects the “fulfill order” button next to all current orders; The page goes blank and shows the text “No unfulfilled orders”

Alternative Flow:

- **Fulfilling An Invalid Order:** The admin attempts to fulfill an order that has already been fulfilled or cancelled by another admin; The admin is prompted with an error message stating that the order could not be fulfilled.
- **No Unfulfilled Orders Exist:** The admin opens the orders page; The admin is displayed with the text “No unfulfilled orders”
- **No kitchen staff to fulfill orders**

Use Case 6: Paying for order (Customer)

Preconditions: Customer has an order ready in cart and is at the checkout screen; customer has selected payment.

Main Flow: Customer selects “Checkout -> Pay”; System displays payment options (saved card, new card, campus meal plan, cash on pickup); Customer selects a payment method and supplies required info (card details or meal-plan credentials; Customer confirms the payment amount (order + tax + tip); System sends the payment to the payment gateway and waits for confirmation; Payment gateway returns success; system marks order as paid and issues receipt (on-screen + email/SMS); System updates order status to **Paid** and notifies staff.

Subflows:

- **Use saved card:** Customer chooses a saved card and only needs the CVV.
- **Use meal plan integration:** System validates meal-plan balance; if OK, charges meal plan.
- **Cash on pickup:** Customer selects “cash on pickup” – no immediate charge; order status is “Pending - Cash”.

Alternative Flows:

- **Payment failure:** Gateway returns failure -> system prompts retry or choose another method.
- **Insufficient funds:** System informs customer and offers to use another method.
- **Network/ gateway down:** System shows “Payment unavailable” and offers cash-on-pickup or retry later.
- **Partial charge (split payments):** If allowed, the customer chooses split; if one part fails, prompt to fix.

Use Case 7: Manage Users (Admin: Create/ Edit/ Delete Staff & Customers)

Preconditions: Admin logged in with user-management privileges.

Main Flow: Admin navigates to “User Management.”; Admin selects “Create User” and inputs details (name, email, role, permissions); Admin assigns role (Staff/ Customer/ Other) and submits; System creates user, sends activation email, and records action in audit log; For Edit/Delete, admin selects user, edits fields or selects delete, confirms action, system updates and logs.

Subflows:

- **Bulk import:** Admin uploads CSV to create multiple users — system validates and reports errors.
- **Password reset:** Admin triggers reset which emails user a reset link.

Alternative Flows:

- **Duplicate email or invalid data:** System rejects and shows validation errors.
- **Unauthorized admin (insufficient rights):** System denies access and logs attempt.
- **Failed activation email:** Admin can resend activation or manually activate account.

Use Case 8: Configure Tax Rate & System Settings (Admin)

Preconditions: Admin has system-settings privileges and is on the settings page.

Main Flow: Admin navigates to “Settings -> Financial.”; Admin edits sales tax rate, default tip suggestions, and store hours; Admin clicks “Preview” to see sample order price changes; Admin saves changes. System validates inputs and stores settings in config; System logs change (who/ when/ old -> new) and notifies affected services (pricing module).

Subflows:

- **Preview impact:** Admin selects sample items/date to preview recalculated totals.
- **Rollback:** Admin views history and reverts to a prior config version.

Alternative Flows:

- **Invalid value (negative rate or >100%):** System rejects save and shows error.
- **Concurrent change:** Another admin changed settings; system shows conflict and asks to reconcile.

Use Case 9: View Order History & Generate Sales Report (Staff/Admin)

Actors: Staff, Admin, System, Reporting Module

Preconditions: User has reporting permissions; system has historical order data.

Main Flow: User navigates to “Reports -> Sales History.”; User selects date range, filters (by -item, staff, location) and clicks “Generate.”; System queries DB and renders results (table + charts); User optionally exports results (CSV/ PDF) or schedules repeat reports via email.

Subflows:

- **Drill-down:** User clicks an item to see order-level details for that item.
- **Scheduled email report:** User sets up automated weekly email to stakeholders.

Alternative Flows:

- **Insufficient data for range:** System indicates “no data” and suggests changing filters.
- **Export error:** System logs error and suggests retry or smaller filters.
- **Permission denied:** System prevents access and logs the attempt.

Use Case 10: Refund/ Cancel Order (Admin/Staff)

Preconditions: Order exists in system; a refund/cancel request has been initiated within policy limits.

Main Flow: Staff/admin locates the order in the system; Staff selects “Refund” or “Cancel.”; Staff chooses full or partial refund and reason; System processes refund via payment gateway (if paid) or marks it for manual cash refund; System restocks inventory for canceled items automatically (if applicable) and updates financial ledger; System notifies customer and logs the action for audit.

Subflows:

- **Partial refund for missing/ damaged item:** Staff specifies item-level refund amount.
- **Issue store credit instead of cash:** Staff selects “Store Credit” — system issues credit code.

Alternative Flows:

- **Already picked up/ beyond refund window:** System denies automatic refund and routes to manager approval.
- **Payment gateway refuses refund:** System creates a support ticket and flags finance team for manual resolution.
- **Refund attempted on guest order with no payment details:** Staff must issue store credit or manual cash refund.

How to find use cases?

List all the roles (stakeholders) and their responsibilities. Actions and responsibilities can be summarised into use cases. Each of the use cases can address a group of activities. Redundant activities of the same stakeholder can be combined.

LLM (Gemini 2.5 Pro) response for “how to find use cases?”

A use case describes a specific goal a user wants to achieve with a system. The general procedure for finding use cases involves an iterative process of identifying key elements and then detailing their interactions (Identify the actors, identify the goal, define the scope).

Prompt given to LLM:

Describing the WolfCafe system; Giving the template requirements for use-cases; Giving examples (clear intersection); Asking for sample use cases for the WolfCafe system;

LLM sample response:

Check Order Status: This use case allows a customer to check the real-time status of their placed order.

Preconditions: Customer has already placed an order; User is logged into their account.

Main Flow: Customer navigates to the "My Orders" or "Order History" section of the system; The system retrieves and displays a list of the customer's recent orders; The customer selects a specific order to view its details; The system retrieves and displays the current status of the selected order (e.g., "In Progress", "Ready for Pickup").

Sub-flows: Get Order Status: The system queries the order database for the specific order and retrieves its current status.

Alternative Flows: The customer attempts to view an order that does not exist or has been deleted from the system. The system displays a "Order Not Found" message.