SOFTWARE ENGINEERING
PROJECT 1a1

Team Members

Names	Unity ID
1.Pradyumna Chacham	pchacha2
2. Sai Mahathi Suryadevara	ssuryad6
3. Sai Sumedh Kaveti	skaveti
4. Sadana Ragoor	sragoor

1. Stakeholder List

Primary Stakeholders (direct users):

- **Customer** – browses menu, places orders, pays, receives notifications.
- **Staff** – manages inventory, creates items,and fulfills orders.
- **Admin** – manages roles, permissions, overall system config.

Secondary Stakeholders (indirect users but affected):

- **Teaching Assistants (Managers)** – oversee progress, act as team managers.
- **Instructors (CTOs/Product Owners)** – define product vision, approve features.
- **Developers (Team Members)** – implement, test, and integrate features.
- **University IT Department** – may provide hosting, integration, and maintenance.
- **Accessibility Users (e.g., screen reader users)** – depend on compliance with accessibility standards.
- **External Payment Processor (System)** – handles customer transactions.
- **Notification System (Email/SMS/Push)** – informs customers about order status.
- **Third-party delivery services** - Similar to GrubHub, Uber Eats, and communication/ order status about orders to these vendors.

Stakeholder Identification Template

1. Primary Stakeholders (Direct Interaction)

👉 *Who directly uses or operates the system?*

- **End-users** – individuals interacting with the system (e.g., customers, students, patients).

- **Operational staff** – staff who use it as part of their daily tasks.
- **System administrators** – users with elevated privileges.

2. Secondary Stakeholders (Support & Maintenance)

👉 *Who supports, manages, or ensures the system runs smoothly?*

- **Developers / Engineers** – design, build, and maintain the system.
- **Testers / QA team** – validate correctness, performance, usability.
- **IT/Support staff** – troubleshoot and handle incidents.
- **Trainers** – help users learn to use the system.

---

3. High-Level / Tertiary Stakeholders (Oversight & Impact)

👉 *Who influences or is affected indirectly?*

- **Project sponsors / Product owners** – fund or prioritize requirements.
- **Managers / Decision-makers** – ensure system aligns with business/academic goals.
- **Regulators / Compliance bodies** – ensure legal, safety, or ethical compliance.
- **Vendors / Third-party providers** – e.g., payment systems, cloud services, hardware providers.
- **Auditors** – check logs, ensure accountability.

---

4. External Stakeholders (Contextual Influence)

👉 *Who doesn't interact directly, but still cares?*

- **Customers' families/communities** – impacted by accessibility, inclusivity, or safety.
- **Society at large** – if the system has environmental, ethical, or cultural implications.
- **Future maintainers/successors** – the next team that inherits the project.

✅ Quick Steps to Apply

1. Start with the **system context** → who touches it first?
2. Move outward layer by layer (users → support → management → external).
3. Ask the **impact questions** (who gains, who loses, who decides).

2. Stakeholder Biases & Conflicts

Here are **5 examples** of how needs may clash or diverge:

1. **Customer vs. Staff**
   - Customer wants fast service; staff may be understaffed and can't fulfill orders quickly. (speed vs quality)
2. **Admin vs. Staff**
   - Admin may enforce strict permissions (only staff can add inventory) → staff feels slowed down compared to a flexible system. (security vs flexibility/usability). The staff on the ground have more experience with how the system works.
3. **Customer vs. Accessibility User**
   - Typical customer just wants quick ordering → accessibility user needs extra features (tooltips, keyboard navigation) that may slow down development. (complexity increases to ensure usability and accessibility).
4. **Developers vs. Product Owners**
   - Developers want technically simple solutions → product owners want a feature-rich system, even if harder to implement. (maintainable code vs extra features)
5. **Payment System vs. Customer**
   - Customers prefer flexibility (cash, cards, PayPal, etc.) → external payment processor may limit options. (Security vs flexibility)

👉 To find biases:

- Think of each stakeholder's **goal**.
- Ask: Does this goal **conflict, overlap, or seem irrelevant** to another stakeholder?
- Each conflict or bias can be solved by considering its trade-offs and priorities and using models to better understand the system.

---

3. Prompt Crafting Reflection

- **Zero-Shot Prompting**:
  - You ask an LLM without much context.
  - Example: *"Write 10 use cases for WolfCafe."*
  - Outcome: generic, lacks details like roles, flows, and conflicts.

    - Example for this project: “Here is a WolfCafe system, give me a report to be submitted for the homework focused on Requirements engineering”.
- **Careful Prompting (Few-shot + Structured)**:
    - You provide format, examples, and context.
    - Example: *“Use the following template (Preconditions, Main Flow, Subflows, Alternative Flows). Here’s a sample from lecture slides (Clear Intersection). Now, generate 10 WolfCafe use cases.”*
    - Outcome: targeted, consistent, useful for assignment.
    - Example: “How to find the list of stakeholders for any system/project?” “ The RE report should be technical but still be understandable to undergrads (who will be learning this course next year). – Adds the right amount of technical detail for it to be easily followed

👉 Reflection:
Zero-shot is fast but unfocused; careful prompting yields structured, higher-quality results. For software engineering tasks like use cases, **careful prompting is far superior**.

4. Use Cases (10 examples)

UC-1: Create New Item (Staff)

Preconditions: Staff logged in.
Main Flow:

1. Staff selects “Create Item.”
2. Staff enters item details (name, price, description).
3. System saves the item and confirms availability.
   Subflows:
- Validate item details (name not empty, price > 0).
  Alternative Flows:
- Invalid input → system displays error message.

UC-2: Add Inventory (Staff)

Preconditions: Item exists.
Main Flow:

1. Staff selects an item.
2. Staff enters the quantity to add.
3. System updates inventory.
   Alternative Flows:
- Item not found → error.
- Quantity invalid (negative/zero).

UC-3: Register New Customer (Customer)

Preconditions: Customer not logged in.
Main Flow:

1. Customer selects “Register.”
2. Provides details (username, password, email).
3. System confirms account creation.
   Alternative Flows:

- Username already exists → error.

UC-4: Browse Menu (Customer)

Preconditions: Items exist in the system.
Main Flow:

1. The customer opens the menu page.
2. The system displays items grouped by category.
3. Customer views details.
   Alternative Flows:

- No items available → system displays “Menu unavailable.”

UC-5: Place Order (Customer)

Preconditions: Customer logged in, items in menu.
Main Flow:

1. Customer selects items.
2. Adds to cart.
3. Proceeds to checkout.
4. Submits payment.
5. The system creates an order and sends a confirmation.
   Alternative Flows:

- Payment fails → system cancels order.
- Item out of stock → system notifies customer.

UC-6: Fulfill Order (Staff)

Preconditions: Order exists and is pending.
Main Flow:

1. Staff views pending orders.
2. Selects an order.
3. Prepares items.
4. Marks order is complete.
5. System notifies the customer.
   Alternative Flows:

- Ingredient shortage → order canceled.

UC-7: Manage User Roles (Admin)

Preconditions: Admin logged in.
Main Flow:

1. Admin selects “Manage Users.”
2. Assigns role (customer, staff).
3. System saves changes.
   Alternative Flows:

- User not found → error.

UC-8: Track Order Status (Customer)

Preconditions: Order placed.
Main Flow:

1. Customer logs in.
2. Navigates to “My Orders.”
3. System shows order status (Pending, In Progress, Ready).
   Alternative Flows:

- Order not found → error.

UC-9: Notification of Order Completion (System → Customer)

Preconditions: Order fulfilled by staff.
Main Flow:

1. Staff marks order complete.
2. System sends notification (email/app).
   Alternative Flows:

- Notification service unavailable → system retries after delay.

UC-10: Generate Sales Report (Admin/Staff)

Preconditions: Orders exist in the system.
Main Flow:

1. Admin selects "Generate Report."
2. System compiles sales by day/week/month.
3. Report displayed.
   Alternative Flows:

- No sales data → report shows empty.

Best Prompts to Ask an LLM for Discovering Use Cases

When brainstorming use cases for a system like WolfCafe, you can use prompts like:

1. **Role-based prompt**
   "List what actions a [customer/staff/admin/system] might want to do in WolfCafe."

2. **Scenario-based prompt**
   "Imagine a customer ordering during peak hours—what could go wrong or what actions are needed?"

3. **Goal-based prompt**
   "What are the end goals of each stakeholder (e.g., customer wants food fast, admin wants security) and how do those translate into system actions?"

4. **Error/exception prompt**
   "What unusual or failure situations might occur in WolfCafe (e.g., payment fails, item runs out)? Turn these into use cases."

5. **System integration prompt**
   "What other systems (payment gateway, notifications, inventory tracking) does WolfCafe connect to, and what use cases arise from that?"

6. **Lifecycle prompt**
   "Follow the lifecycle of an order—from menu browsing to fulfillment—what use cases appear at each step?"