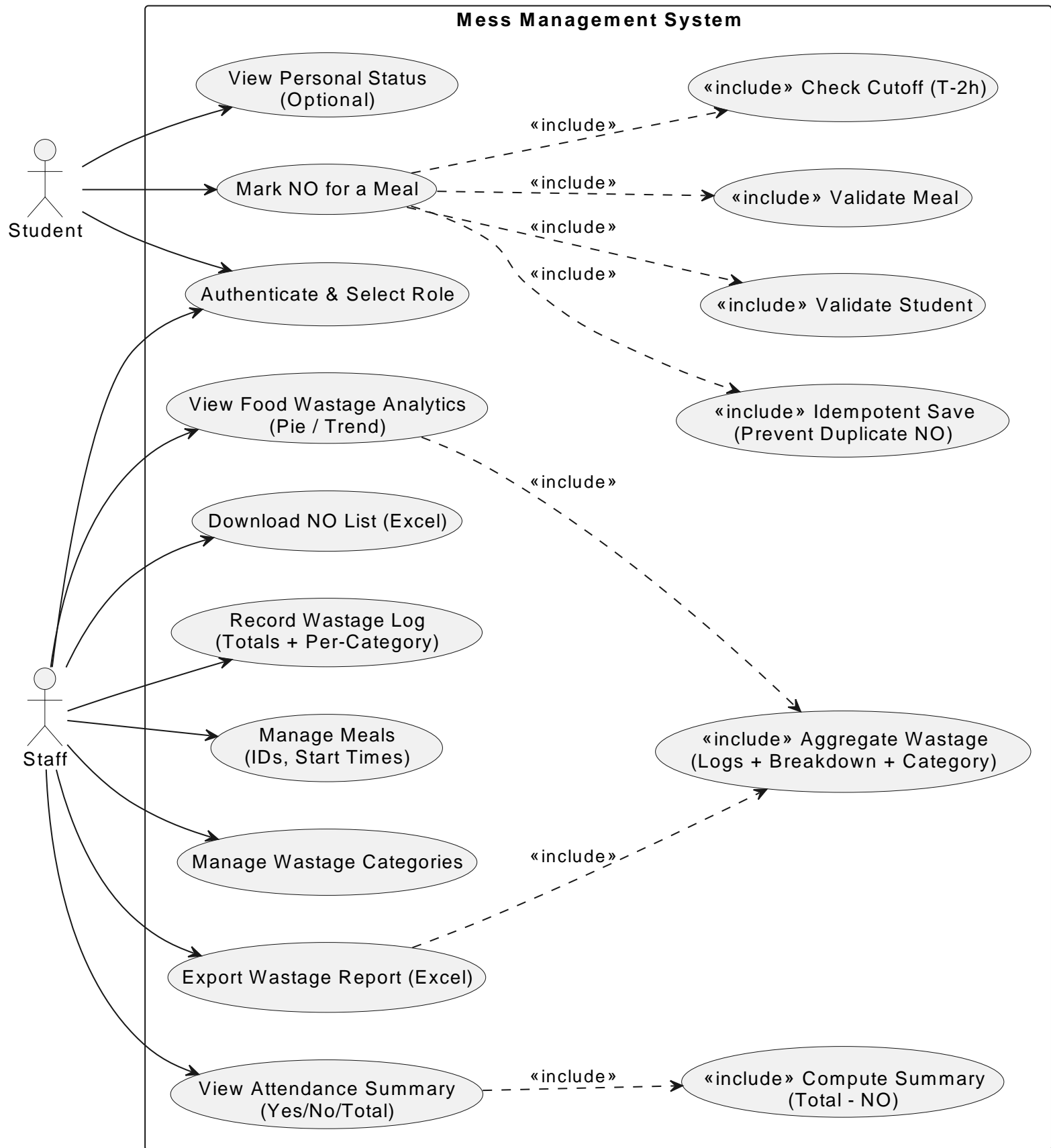


# Mess Management System - Use Case Diagram

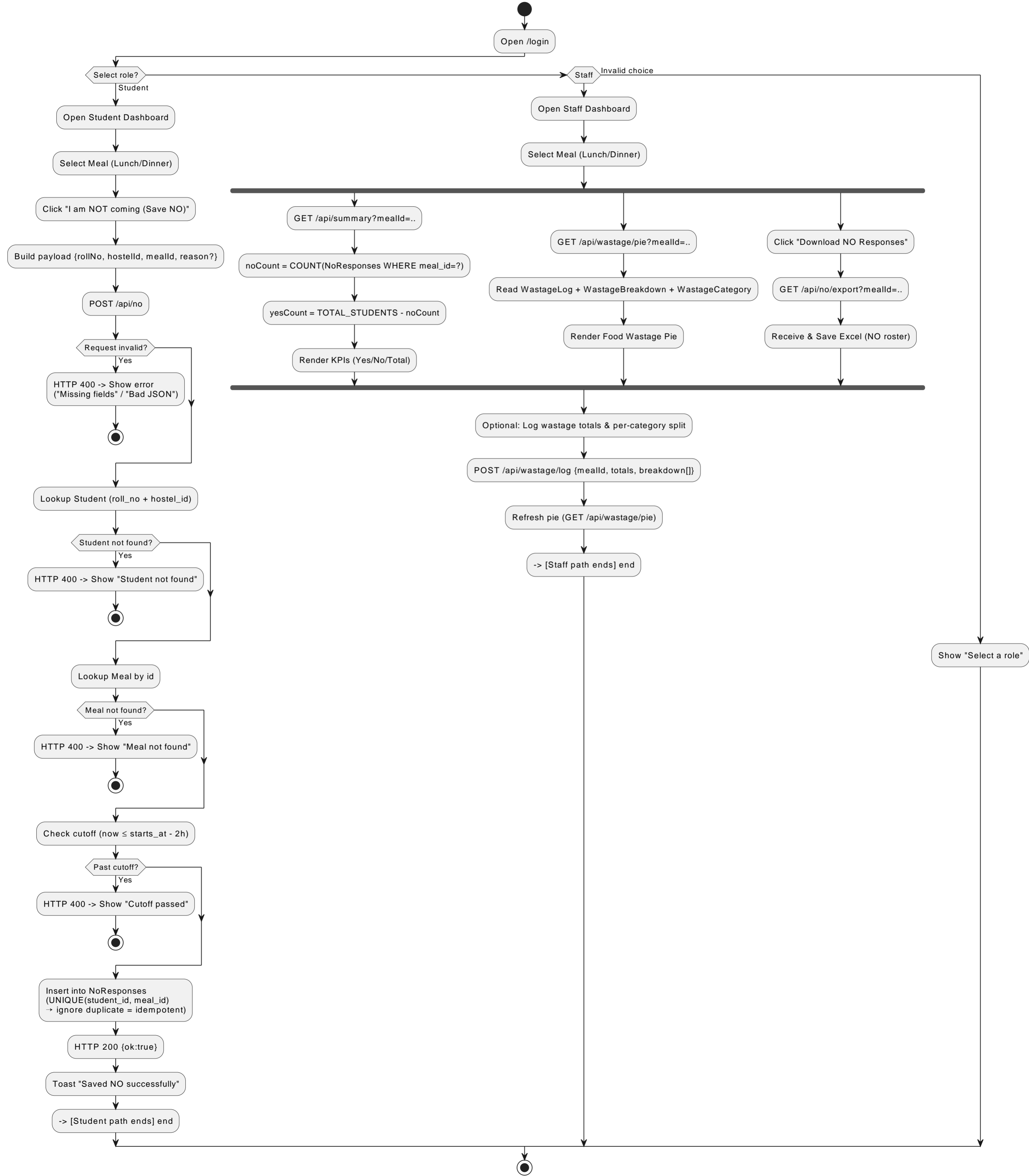


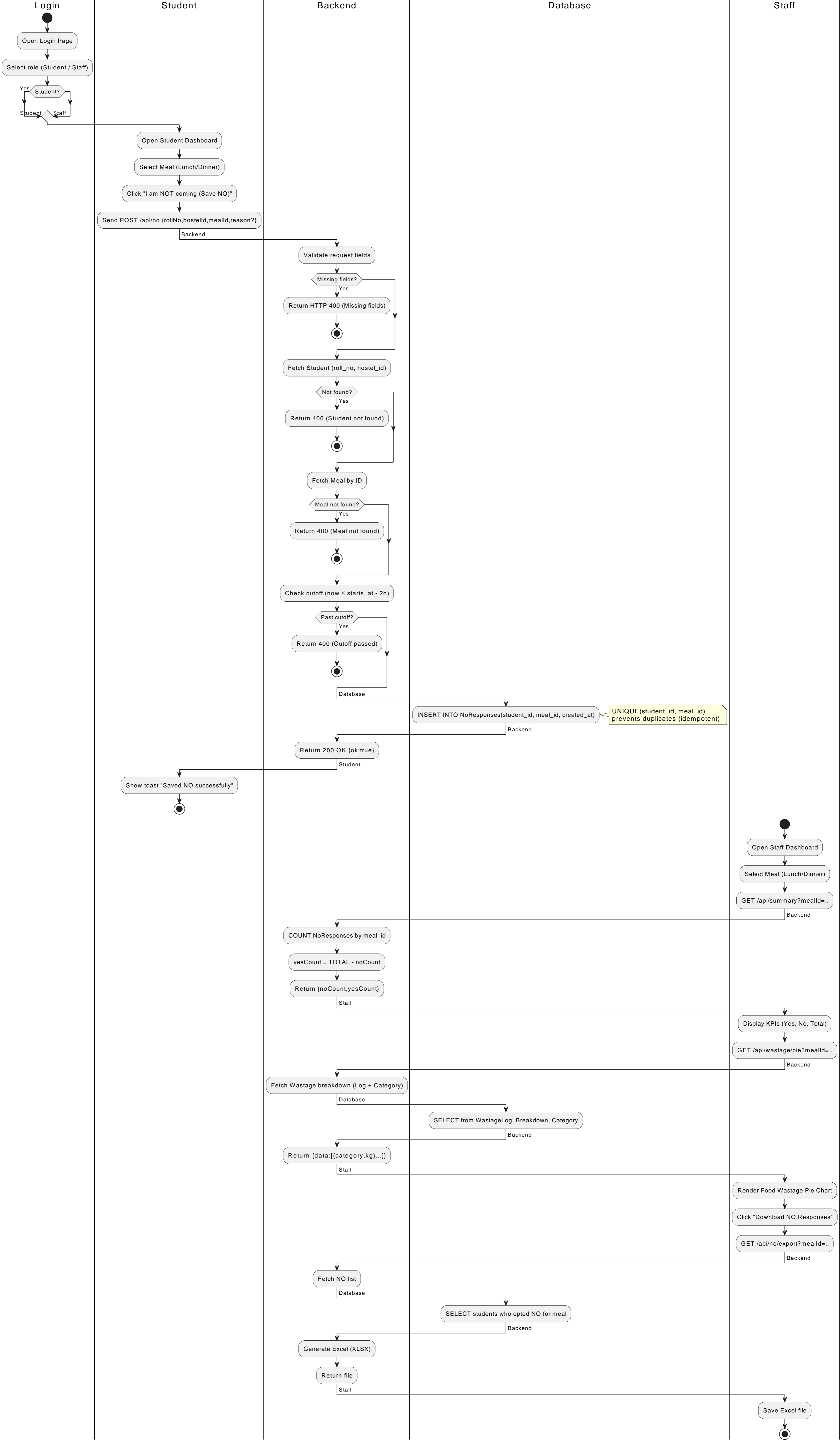
1. Use Case Title:	User Authentication and Role Access
2. Abbreviated Title:	Login & Role Selection
3. Use Case ID:	UC-01
4. Actors:	Student, Staff
5. Description: This use case allows users to log into the Mess Management System and access their respective dashboards based on role (Student or Staff). Authentication is performed using valid credentials.	
5.1 Pre-Conditions: User must have valid login credentials and be registered in the system.	
5.2 Task Sequence: 1. User opens the login page. 2. User enters username and password. 3. System validates credentials (include: Validate Student). 4. User selects role (Student or Staff). 5. System redirects to the appropriate dashboard.	
5.3 Post Conditions: 1. User successfully logged in and dashboard displayed. 2. Unauthorized users are denied access.	
6. Modification History: Date – 02-Nov-2025	

1. Use Case Title:	Meal Response Management
2. Abbreviated Title:	Meal Response
3. Use Case ID:	UC-02
4. Actors:	Student, Staff
5. Description: This use case allows students to mark their meal preference (YES/NO) and staff to view, summarize, and download meal response data in Excel format.	
5.1 Pre-Conditions: User must be logged in. The meal must be valid and cutoff time not passed	
5.2 Task Sequence: 1. Student selects meal and marks "NO." 2. System validates meal and cutoff (include: Validate Meal, Check Cutoff). 3. System saves response (include: Idempotent Save). 4. Staff views attendance summary 5. Staff downloads NO list (Excel).	
5.3 Post Conditions: 1. Meal response recorded successfully. 2. Updated summary available for staff.	
6. Modification History: Date – 02-Nov-2025	

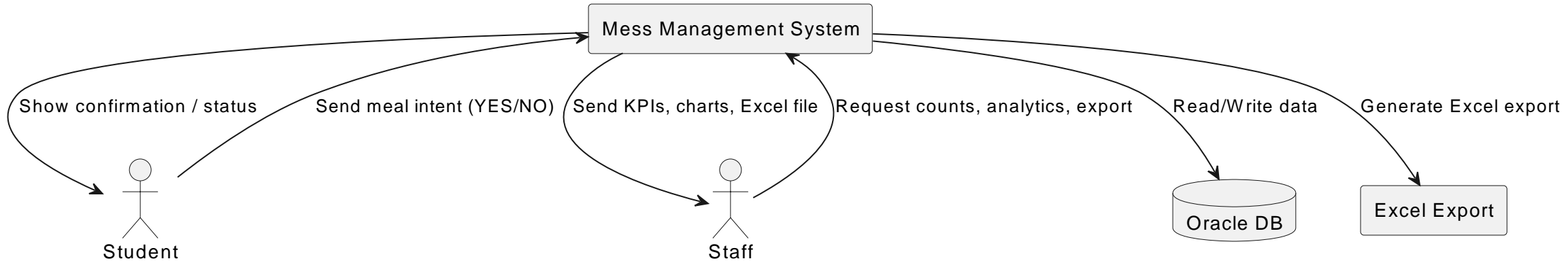
1. Use Case Title:	Wastage Monitoring and Reporting
2. Abbreviated Title:	Wastage Report
3. Use Case ID:	UC-03
4. Actors:	Staff
5. Description: This use case allows staff to record food wastage details, view analytics (pie/trend charts), and export detailed reports in Excel format for analysis.	
5.1 Pre-Conditions: Staff must be logged in. Previous meal data should exist.	
5.2 Task Sequence: 1. Staff records wastage quantities and per-category data. 2. System aggregates wastage (include: Aggregate Wastage). 3. Staff views graphical analytics (pie/trend). 4. Staff exports wastage report in Excel.	
5.3 Post Conditions: 1. Wastage logs stored in database. 2. Report successfully exported.	
6. Modification History:      Date – 02-Nov-2025	

Hostel Mess - Unified Activity (No Swimlanes)

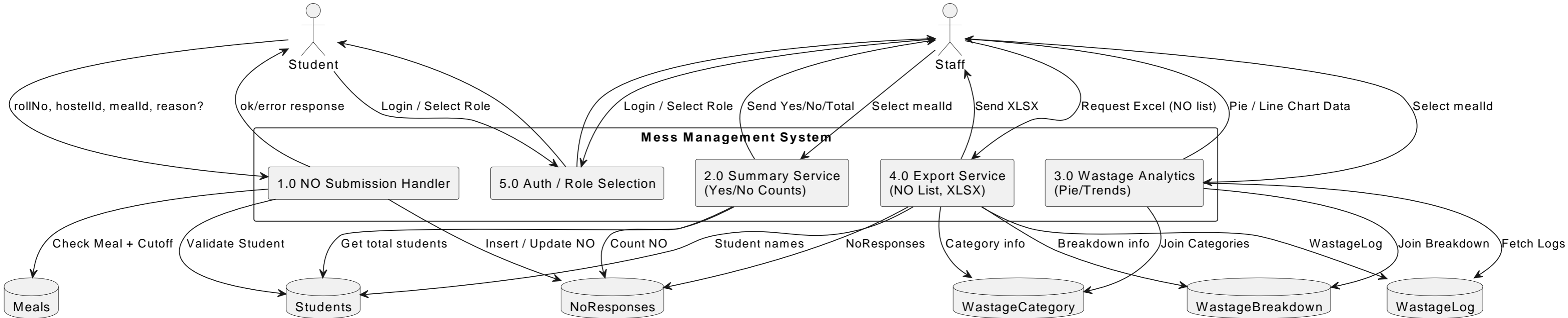




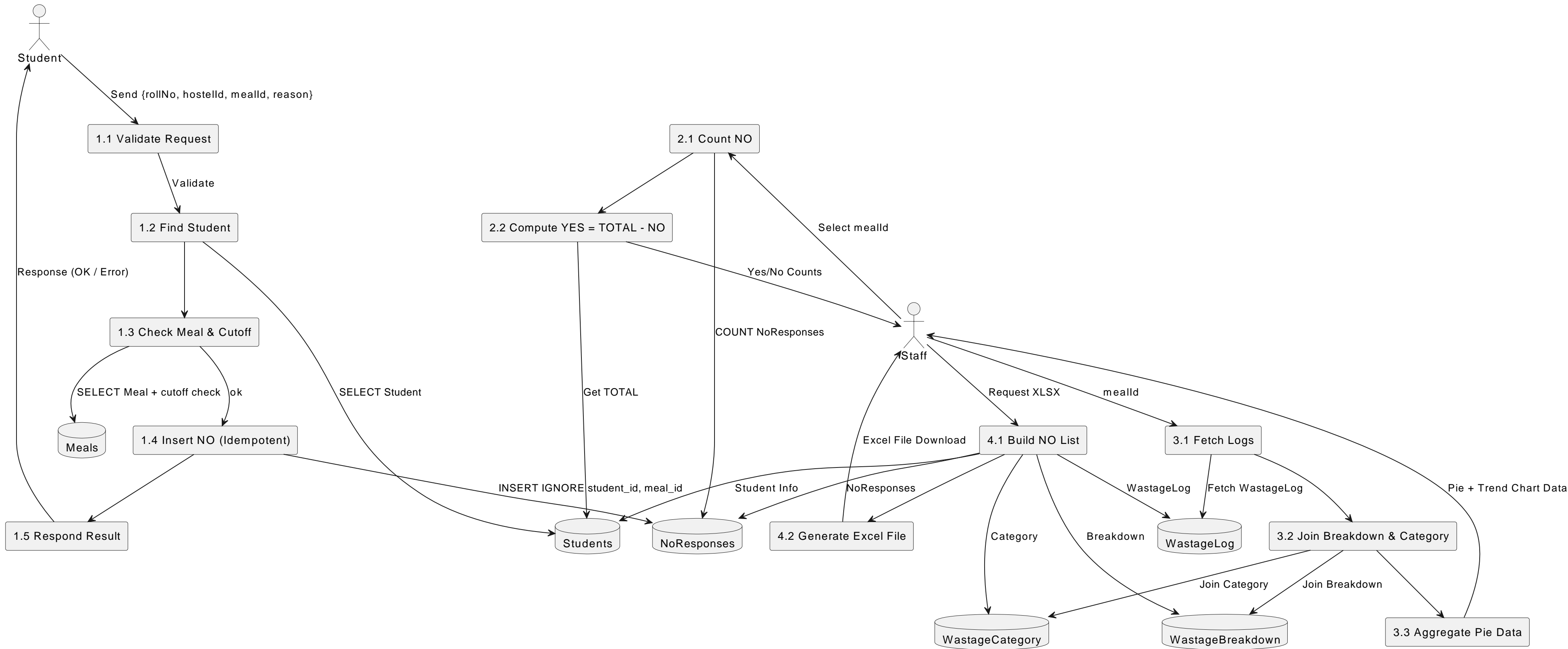
## DFD Level 0 - Mess Management System (Context)



# DFD Level 1 - Mess Management System



DFD Level 2 - Detailed Flow (NO Submission + Summary + Wastage + Export)



# **SOFTWARE REQUIREMENTS SPECIFICATION (SRS)**

---

## **Hostel Mess Management System**

**Version 1.0**

Prepared by:

Code and Conquer

Date:

31/10/2025

# Table of Contents

1. Introduction
  - 1.1 Purpose of this Document
  - 1.2 Scope of the Development Project
  - 1.3 Definitions, Abbreviations, and Acronyms
  - 1.4 References
  - 1.5 Overview
2. Overall Description
  - 2.1 Product Perspective
  - 2.2 Product Functions
  - 2.3 User Characteristics
  - 2.4 General Constraints, Assumptions, and Dependencies
  - 2.5 Apportioning of Requirements
3. Specific Requirements
  - 3.1 External Interface Requirements
  - 3.2 Detailed Description of Functional Requirements
  - 3.3 Performance Requirements
  - 3.4 Logical Database Requirements
  - 3.5 Quality Attributes
  - 3.6 Other Requirements
4. Change History
5. Document Approvers

# 1. Introduction

## 1.1 Purpose of this Document

The purpose of this SRS is to describe the functional and non-functional requirements of the Hostel Mess Management System (HMMS). This document defines the product's objectives, users, interfaces, performance criteria, and constraints. It is intended for developers, testers, and stakeholders involved in the design and deployment of the system.

## 1.2 Scope of the Development Project

The Hostel Mess Management System automates the process of tracking meal attendance and reducing food wastage in a hostel environment.

The system enables:

- Students to indicate whether they will attend a particular meal (Lunch/Dinner).
- Staff to monitor student meal responses, analyze attendance statistics, and record wastage data.

Primary functions include:

- **Student Meal Opt-Out:** Students can log in, select a meal, and mark "I am NOT coming".
- **Cut-off Enforcement:** The backend checks time cut-offs to prevent late submissions.
- **Attendance Summary:** Staff can view counts of YES/NO responses.
- **Food Wastage Analytics:** Staff can view wastage by category in a pie chart.
- **Export Reports:** Staff can download NO-response lists as Excel (XLSX) files.

The application is web-based, responsive, and accessible from any modern browser. Data is stored securely in a centralized MySQL (Oracle) database through RESTful APIs. The Hostel Mess Management System automates the process of tracking meal attendance and reducing food wastage in a hostel environment. This system streamlines operations and contributes to more efficient resource management within the hostel.

The system enables:

- Students to indicate whether they will attend a particular meal (Lunch/Dinner).

This allows for accurate meal forecasting and reduces last-minute changes.

- Staff to monitor student meal responses, analyze attendance statistics, and record wastage data. These insights help in optimizing food procurement and identifying areas for improvement.

Primary functions include:

- **Student Meal Opt-Out:** Students can log in, select a meal, and mark “I am NOT coming”. This simple feature empowers students to communicate their meal intentions proactively.
- **Cut-off Enforcement:** The backend checks time cut-offs to prevent late submissions. This ensures that meal counts are finalized in a timely manner for kitchen planning.
- **Attendance Summary:** Staff can view counts of YES/NO responses. This provides an immediate overview of expected meal attendance for each serving.
- **Food Wastage Analytics:** Staff can view wastage by category in a pie chart. This visual representation helps in understanding the types and amounts of food being wasted, facilitating targeted reduction strategies.
- **Export Reports:** Staff can download NO-response lists as Excel (XLSX) files. These reports are useful for record-keeping and further analysis of meal preferences.

The application is web-based, responsive, and accessible from any modern browser. This ensures broad accessibility and ease of use for both students and staff. Data is stored securely in a centralized MySQL (Oracle) database through RESTful APIs. This robust architecture guarantees data integrity and efficient communication between the application and the database.

### 1.3 Definitions, Abbreviations, and Acronyms

Term	Definition
HMMS	Hostel Mess Management System
API	Application Programming Interface
DB	Database
JSON	JavaScript Object Notation

<b>KPI</b>	Key Performance Indicator
<b>CRUD</b>	Create, Read, Update, Delete
<b>UI</b>	User Interface
<b>UX</b>	User Experience

#### 1.4 References

- IEEE 830: IEEE Recommended Practice for Software Requirements Specifications.
- Project Swimlane and Activity Diagrams (2025).
- ReactJS Official Documentation (<https://react.dev>)
- Node.js Express Documentation (<https://expressjs.com>)
- MySQL Reference Manual (Oracle, 2025).

#### 1.5 Overview

The remainder of this document outlines the system's functional, non-functional, and data requirements. Section 2 provides a high-level product overview, while Section 3 details the specific functional modules and database design.

---

## 2. Overall Description

### 2.1 Product Perspective

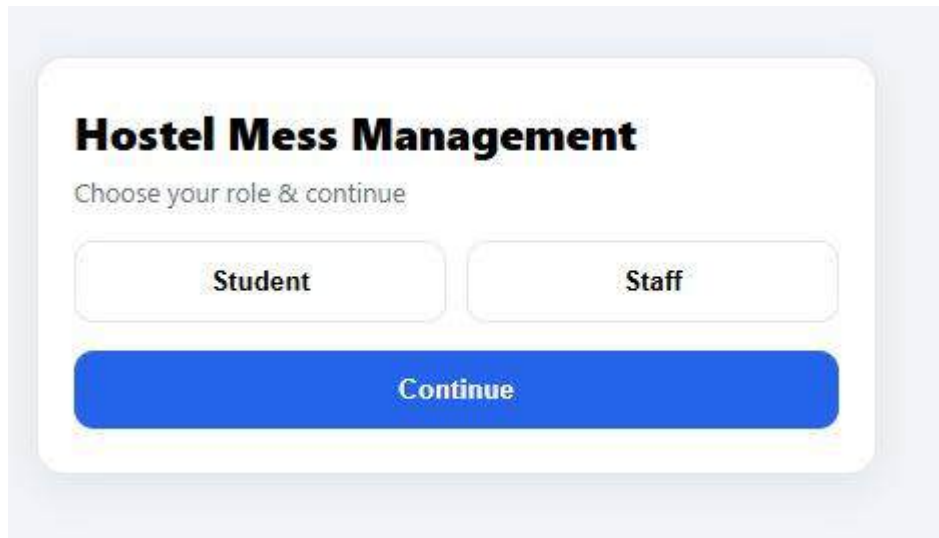


Figure: Login page

HMMS is an independent web application, meaning it operates as a self-contained system. Its architecture comprises a React-based frontend, responsible for the user interface and interactions; an Express.js REST API backend, which handles server-side logic and data requests; and a MySQL database, serving as the persistent storage for all system data.

The system integrates multiple distinct modules, each communicating with the others using HTTPS JSON requests. This standardized communication protocol ensures secure and efficient data exchange between different parts of the application.

#### System Components:

- **Frontend (React):** This component provides the interactive user dashboards for both students and staff members. It is built using the React JavaScript library, offering a dynamic and responsive user experience.
- **Backend (Node + Express):** The backend, powered by Node.js and the Express.js framework, exposes various API routes. These routes facilitate core functionalities such as recording meal attendance, generating summaries of meal consumption, and providing analytics related to food wastage.
- **Database (MySQL/Oracle):** The system utilizes either a MySQL or Oracle

database to store all critical information. This includes tables for managing student data, meal schedules and details, instances of "NoResponses" (likely for meal preferences or attendance), and comprehensive logs and breakdowns of food wastage.

### Deployment Context:

The typical deployment flow for HMMS follows a standard web application architecture:

- **Web client (browser) → REST API → Database server:** Users interact with the web client through their browsers, which then sends requests to the REST API. The API processes these requests, often interacting with the database server to retrieve or store information, and then sends responses back to the client.
- **Authentication managed via login credentials or institutional IDs:** Access to the system is secured through an authentication mechanism. Users can log in using their specific credentials, or, where applicable, their institutional IDs, ensuring that only authorized individuals can access the system's features and data.

## 2.2 Product Functions

Function	Description
<b>Student Login</b>	Students can securely authenticate their credentials to access the system and manage their meal-related activities.
<b>Mark Meal Absence</b>	Students can submit a POST request to <code>/api/no</code> with their <code>rollNo</code> , <code>hostelId</code> , <code>mealId</code> , and an optional <code>reason</code> to mark themselves absent from a particular meal.
<b>Validate Fields</b>	The backend system rigorously checks for any missing mandatory fields, verifies the existence of the specified meal, and ensures that the request is made before the designated cut-off time for meal absence submissions.
<b>Generate Summary</b>	An administrator or authorized user can retrieve a summary of meal attendance by sending a GET request to <code>/api/summary?mealId</code> . This will return counts of students

	marked "yes" (present) and "no" (absent) for the specified meal.
<b>Display Wastage Data</b>	To visualize food wastage, a GET request to <code>/api/wastage/pie?mealId</code> will return data indicating the weights of different food categories contributing to wastage for a given meal, which can then be used to generate a pie chart.
<b>Export Excel</b>	For detailed analysis, a GET request to <code>/api/no/export?mealId</code> will generate and return an <code>.xlsx</code> file containing a comprehensive list of all students who have marked themselves absent for the specified meal.
<b>Data Integrity</b>	To maintain the accuracy and reliability of the data, the system prevents duplicate "NO" responses by enforcing a unique constraint on the combination of <code>student_id</code> and <code>meal_id</code> , ensuring that each student can only mark themselves absent once for a specific meal.

## 2.3 User Characteristics

User	Description	Technical Skill
<b>Student</b>	Marks attendance (YES/NO) for meals	Basic computer/mobile user
<b>Staff</b>	Views summary dashboards, wastage charts, exports data	Intermediate computer skills

Both user types are expected to have institutional login credentials and access to internet-connected devices.

## 2.4 General Constraints, Assumptions, and Dependencies

- Application must operate within hostel network or internet access.
- Database should support concurrent transactions without conflict.
- The time cutoff for marking “NO” is 2 hours before meal start time.
- Duplicate entries are ignored (idempotent API).
- Frontend should load within 3 seconds under normal conditions.
- MySQL server must be synchronized with Oracle schema compatibility.
- APIs must follow REST architecture and return JSON responses.

## 2.5 Apportioning of Requirements

- **Phase I:** Pilot deployment in one hostel block (Lunch/Dinner only).
  - **Phase II:** Extend to all hostels; introduce wastage logging and analytics.
  - **Phase III:** Optional dashboard for administrators with trend insights.
- 

# 3. Specific Requirements

## 3.1 External Interface Requirements

### User Interface:

- **Student Dashboard:** This interactive dashboard provides students with a comprehensive overview of their meal plan, including a detailed meal list. It also features a convenient NO-marking form for students to indicate when they will not be having a meal, and a confirmation toast to ensure their selections are registered.
- **Staff Dashboard:** Designed for administrative staff, this dashboard presents key performance indicators (KPIs) in a summarized format for quick analysis. It includes a pie chart visualization to represent data distribution effectively and an export button for easy downloading of reports.

### API Endpoints:

- **POST /api/no** → Save absence

- `GET /api/summary?mealId` → Attendance summary
- `GET /api/wastage/pie?mealId` → Wastage data
- `GET /api/no/export?mealId` → Download Excel report

#### Hardware:

- Works on desktop/laptop/mobile browsers.

#### Software:

- React 18+, Node 18+, MySQL 8+.

## 3.2 Detailed Description of Functional Requirements

### 3.2.1 Functional Requirements for Student Module

Attribute	Description
<b>Purpose</b>	This feature allows students to proactively opt out of specific meals in advance, preventing unnecessary food preparation and reducing waste.
<b>Inputs</b>	The system requires the student's <code>rollNo</code> , their <code>hostelId</code> , the specific <code>mealId</code> they wish to opt out of, and optionally a <code>reason</code> for opting out.
<b>Processing</b>	The system will first validate all provided fields to ensure their correctness. It will then check for the existence of the student's record and the specified meal. Finally, it will verify that the opt-out request is made before the designated cutoff time.
<b>Outputs</b>	Upon successful processing, the system will return a <code>200 OK</code> status with <code>{ok:true}</code> . If there's an error, it will return a <code>400</code> status along with a descriptive error message.

<b>Error Handling</b>	Potential errors include invalid JSON formatting, missing required fields, the student not being found in the system, or the opt-out request being made after the cutoff time has expired.
-----------------------	--

### 3.2.2 Functional Requirements for Staff Module

Attribute	Description
<b>Purpose</b>	This module enables staff members to gain valuable insights into student attendance at meals and monitor food wastage, facilitating better resource management and planning.
<b>Inputs</b>	Staff users will need to provide a <b>selected meal ID</b> to retrieve the relevant data.
<b>Processing</b>	The system will query the <b>NoResponses</b> database to compute attendance counts, and then read the <b>WastageLog</b> to gather information on food wastage.
<b>Outputs</b>	The system will display key performance indicators (KPIs), a visual pie chart illustrating attendance or wastage, and provide a downloadable Excel file for further analysis.
<b>Error Handling</b>	Possible errors include an invalid meal ID being provided, a failure in the database connection, or the absence of data for the selected meal.

### 3.3 Performance Requirements

- 95% of requests processed in < 2 seconds.
- Supports up to 200 concurrent student submissions per minute.
- API uptime ≥ 99%.
- Excel exports should complete within 5 seconds for ≤ 1000 records.

### **3.4 Logical Database Requirements**

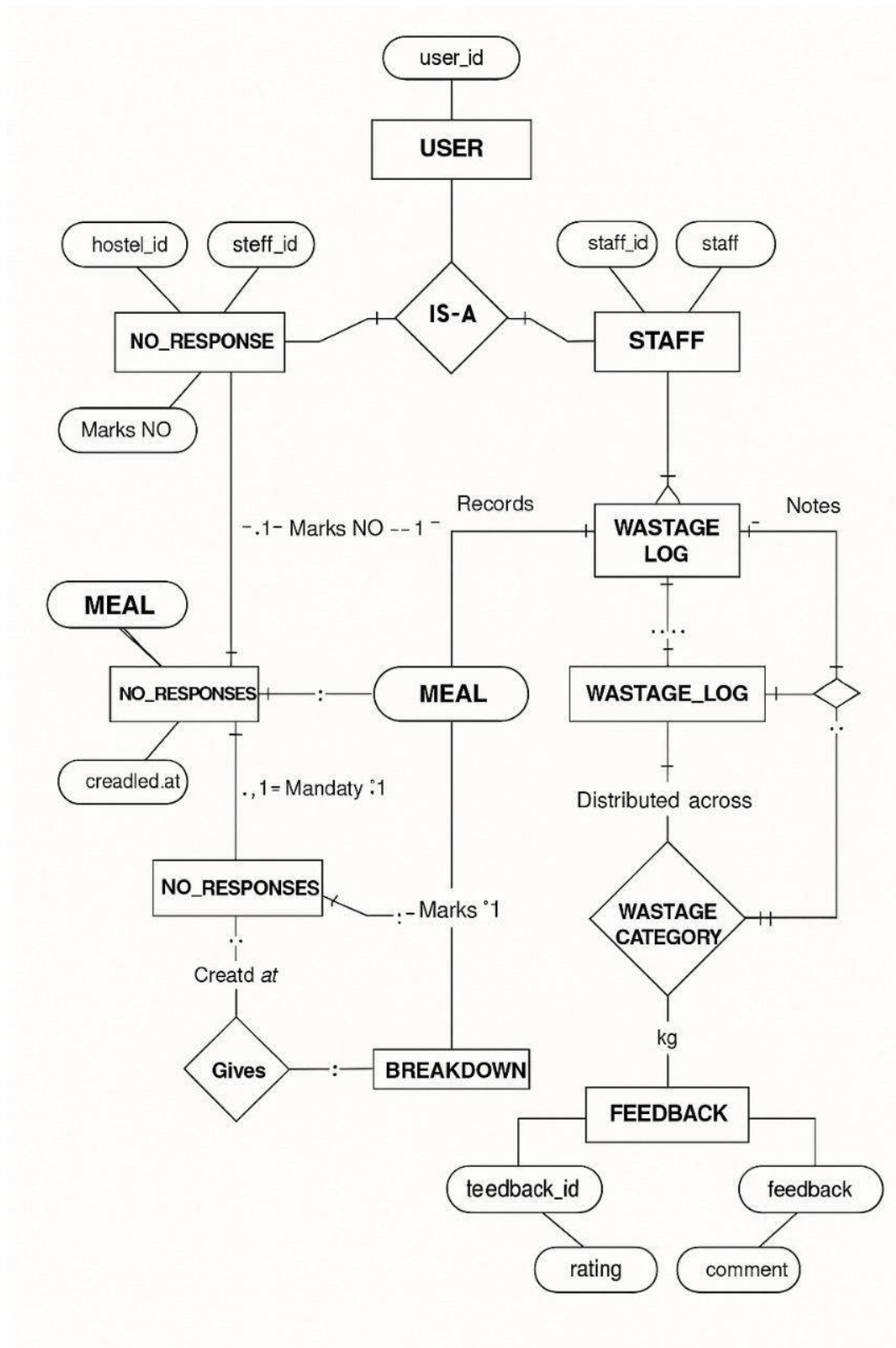


Figure: E R diagram

### Key Tables:

- Students(student\_id, roll\_no, hostel\_id, name, password\_hash)
- Meals(meal\_id, meal\_type, starts\_at)
- NoResponses(student\_id, meal\_id, created\_at)
- WastageLog(meal\_id, total\_kg, recorded\_at)
- WastageBreakdown(wastage\_id, category, kg)

### Relationships:

- Students 1-N NoResponses
- Meals 1-N NoResponses
- Meals 1-1 WastageLog
- WastageLog 1-N WastageBreakdown

## 3.5 Quality Attributes

- **Usability:** Simple, mobile-friendly UI for quick meal marking.
- **Reliability:** Prevents duplicate entries and ensures data consistency.
- **Security:** Secure login and parameter validation in all API calls.
- **Maintainability:** Modular architecture with REST API separation.
- **Scalability:** Ready for multi-hostel expansion.

## 3.6 Other Requirements

- System shall log all actions for audit trail.
  - Error logs maintained on backend for debugging.
  - All timestamps stored in UTC format.
-

## 4. Change History

Version	Date	Description	Author
1.0	31/10/25	Initial Release	Code and Conquer

---

## 5. Document Approvers

Name	Designation	Signature	Date
	Project Guide		
	Developer / Team Lead		

**As a [user role], I want to [goal], so I can [reason].**

## **1. Role Management**

1. As a user, I want to choose between **Student** or **Staff**, so I can see the appropriate dashboard.
  2. As a user, I want incorrect credentials to show a clear error message, so I know when my login fails.
  3. As a user, I want my selected role to persist for the session, so refreshing the page doesn't log me out of my role.
- 

## **2. Student Meal Intent (Default = YES)**

4. As a student, I want to select a meal (Lunch/Dinner) and mark **NO**, so staff can plan meals accordingly.
  5. As a student, I want to be prevented from marking **NO** after the cutoff time (2 hours before meal start), so I can't change my intent too late.
  6. As a student, I want duplicate **NO** submissions to be ignored, so I don't accidentally submit multiple entries.
  7. As a student, I want my **roll number** and **hostel ID** to be case-insensitive (e.g., "s004" same as "S004"), so I don't face errors due to casing.
  8. As a student, I want to optionally add a reason when marking **NO**, so staff can understand why I'm skipping the meal.
- 

## **3. Staff Attendance Summary**

9. As a staff member, I want to view **Yes**, **No**, and **Total** counts for a selected meal, so I can plan meal quantities and staffing.
  10. As a staff member, I want to **refresh** the data manually or automatically, so the attendance summary stays updated.
  11. As a staff member, I want to **download** the list of students who marked **NO** as an Excel file, so I can use or share it offline.
- 

## **4. Wastage Logging & Analytics**

12. As a staff member, I want to record the **total cooked, used, and leftover** food for each meal, so I can maintain wastage records.

13. As a staff member, I want to enter **category-wise wastage** (e.g., rice, gravy, scraps), so analytics can show detailed insights.
  14. As a staff member, I want to see a **pie chart** showing category-wise wastage for a meal, so I can visualize wastage distribution.
  15. As a staff member, I want to view a **7-day trend or forecast** of wastage and attendance, so I can anticipate requirements in advance.
  16. As a staff member, I want to **export wastage logs** and breakdowns as Excel files, so I can share data for audits.
- 

## 5. Admin / Master Data Management

17. As an admin, I want to **add or update student details** (name, roll number, hostel), so onboarding is simple.
  18. As an admin, I want to **manage meal details** (IDs, types, start times), so cutoff rules work correctly.
  19. As an admin, I want to **maintain wastage category names**, so category breakdowns remain consistent.
- 

## 6. Non-Functional & Cross-Cutting Features

20. As a user, I want the system's APIs to respond in **under 300 ms**, so dashboards stay responsive.
21. As a developer, I want proper **error logs** and **request timing** recorded, so I can debug and monitor performance easily.
22. As a system, I want **input validation** and **CORS security**, so the data remains safe and endpoints are protected.
23. As an operator, I want a **.env configuration file** for values like TOTAL\_STUDENTS, CUTOFF\_MINUTES, DB\_CREDENTIALS, and CORS, so I can deploy without changing code.

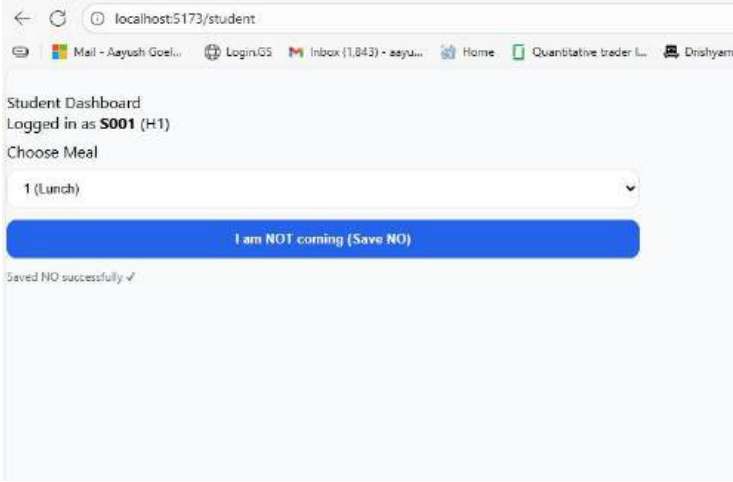
A	Role Selection & Session
<ul style="list-style-type: none"><li>As a user, I want to choose <b>Student</b> or <b>Staff</b> (and optionally authenticate), so that I can access the appropriate dashboard.</li><li>As a user, I want my selected role to persist for the session so that refreshing the page doesn't log me out.</li></ul>	
<div><div>Hostel Mess Management</div><div>Choose your role &amp; continue</div><div><div>Student</div><div>Staff</div></div><div>Roll No</div><div>e.g. S001</div><div>Hostel ID</div><div>e.g. H1</div><div>Continue</div></div>	

Success:

- Selecting Student → routes to Student Dashboard.
- Selecting Staff → routes to Staff Dashboard.
- If credentials are implemented, valid login → dashboard loads correctly
- Selected role stored in sessionStorage or localStorage.
- Refreshing retains current dashboard.

Failure Scenarios:

- If role not selected → show “Please select a role.”
- If credentials invalid → show “Incorrect username or password.”
- If server error → show “Unable to load dashboard, please try again later.”
- Storage blocked (e.g., browser privacy mode) → fallback to role selection page.
- Tampered or expired session → redirect to login/role selection.

B	Student Meal Intent
<ul style="list-style-type: none"><li>As a student, I want to select a meal (Lunch/Dinner) and save <b>NO</b>, so that staff can plan food quantity accordingly.</li><li>As a student, I want cutoff validation so that I cannot mark <b>NO</b> after the cutoff time.</li><li>As a student, I want duplicate <b>NO</b> submissions to be ignored so that no duplicate records are created.</li><li>As a student, I want my <b>roll number</b> and <b>hostel ID</b> to match case-insensitively so that “s004/h2” and “S004/H2” are treated the same.</li><li>As a student, I want to optionally add a <b>reason</b> when marking <b>NO</b>, so that staff can identify patterns.</li></ul>	
	

Success:

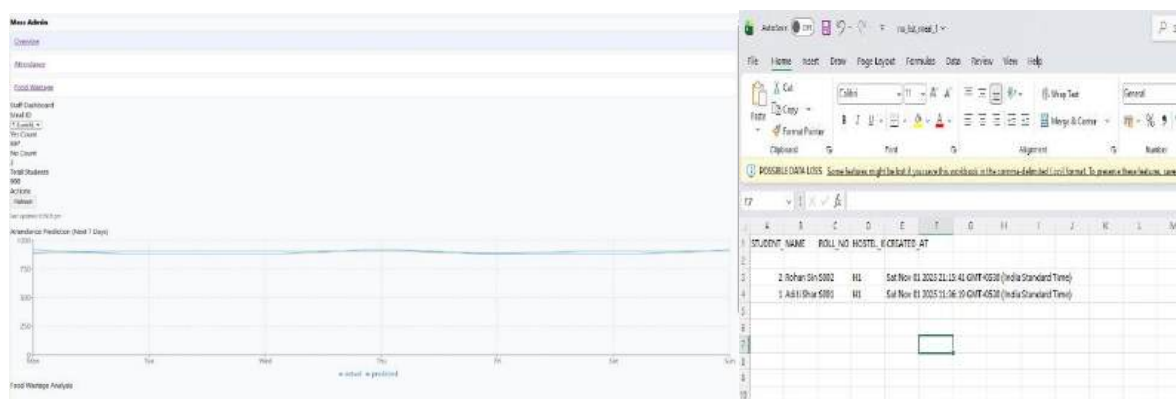
- Request: {rollNo, hostelId, mealId}
- “Saved NO successfully” message displayed.
- If current time < starts\_at – 2h → save allowed.
- If already marked NO → same success message, no duplicate row created.
- “s004” and “S004” treated the same.
- reason saved if provided; ignored if blank.

Failure Scenarios:

- Missing required field → “Incomplete details, please fill all fields.”
- Invalid mealId → “Meal not found.”
- Database/network failure → “Unable to save, try again later.”
- If current time ≥ starts\_at – 2h → show “Cutoff passed, cannot save now.”
- Backend time mismatch or invalid meal time → show “Invalid meal timing configuration.”
- Race condition (multiple clicks) → backend must ensure only one record is inserted.
- DB constraint failure → “Already submitted.”

## C Staff Attendance Summary

- As a staff member, I want to select a meal and see **Yes**, **No**, and **Total** counts so that I can plan meal preparation.
- As a staff member, I want to refresh the attendance data manually or automatically so that I always see current counts.
- As a staff member, I want to download the **NO** list for a selected meal so that I can share or use it offline.




### Success:

- noCount fetched from DB.
- yesCount = TOTAL – noCount.
- Manual refresh button reloads data.
- Optional auto-refresh every 30–60s.
- Excel includes: Roll No, Name, Hostel, Timestamp (sorted).

### Failure Scenarios:

- Meal not found → “Invalid meal selected.”
- DB query fails → “Unable to fetch attendance summary.”
- TOTAL\_STUDENTS not configured → show “Configuration error.”
- Network issue → “Failed to refresh data.”
- Auto-refresh stops due to token expiry → “Session expired.”
- No NO entries → Excel with header only + message “No NO records.”
- Export service error → “Download failed, please retry.”

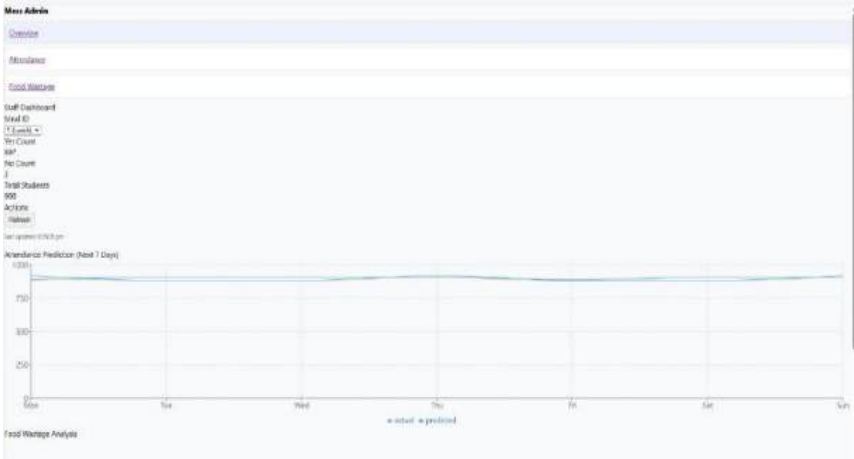
D	Wastage Logging & Analytics
<ul style="list-style-type: none"> <li>As a staff member, I want to record the total food cooked, used, and leftover for each meal so that wastage is tracked daily.</li> <li>As a staff member, I want to enter category-wise wastage (e.g., rice, curry, scraps) so that analytics show meaningful distribution.</li> <li>As a staff member, I want a pie chart of category-wise wastage for a meal so that I can visualize the distribution.</li> <li>As a staff member, I want a 7-day line chart comparing <b>actual vs projected</b> wastage or attendance so that I can anticipate needs.</li> <li>As a staff member, I want to export wastage logs (with breakdowns) to Excel so that I can share them for audits.</li> </ul>	
	

Success:

- Record added to WastageLog(meal\_id, totalCookedKg, usedKg, leftoverKg, notedByUser).
- Each entry inserted into WastageBreakdown(wastage\_id, category\_id, kg).
- GET /api/wastage/pie?mealId= returns { name, kg }.
- “No wastage yet” displayed if zero data.
- Shows 7 data points for actual and projected.
- Includes totals, categories, date, and meal filters.

Failure Scenarios:

- Missing field → “All quantities are required.”
- Negative or invalid kg value → “Invalid input.”
- Category not found → “Invalid category.”
- Non-numeric kg → “Please enter a valid number.”
- Invalid mealId → “Invalid meal.”
- Missing data → “Not enough data to show trend.”

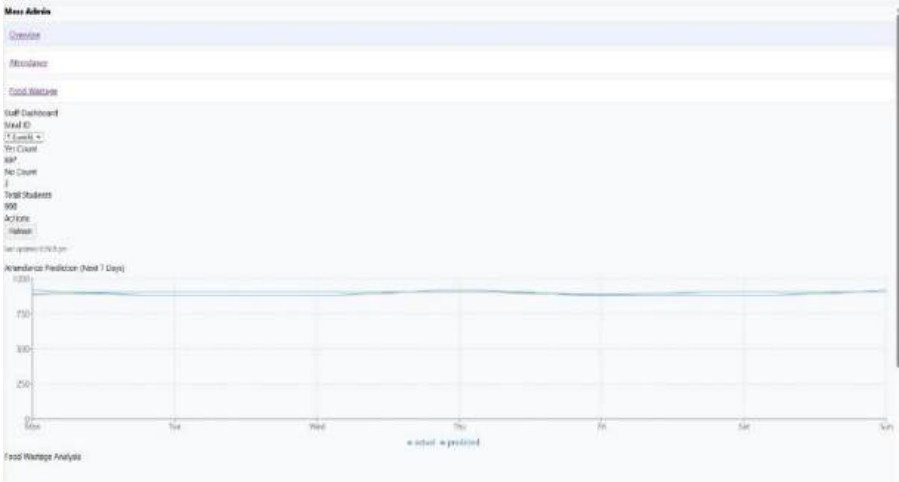
E	Admin / Master Data
<ul style="list-style-type: none"><li>As an admin, I want to add or update student details (name, roll_no, hostel) so that onboarding is simple.</li><li>As an admin, I want to manage meal IDs and start times so that cutoff rules work correctly.</li><li>As an admin, I want to manage category names so that wastage breakdowns remain consistent.</li></ul>	
	

Success:

- Uniqueness on roll\_no.
- Bulk import supported (CSV or SQL Developer).
- CRUD on Meals(id, meal\_type, starts\_at).
- CRUD on WastageCategory(name); unique names.

Failure Scenarios:

- Duplicate roll\_no → “Roll number already exists.”
- CSV format invalid → “Invalid import file.”
- Duplicate meal\_type → “Meal already exists.”
- Invalid start time → “Invalid meal time format.”
- Duplicate name → “Category already exists.”
- Empty name → “Name cannot be blank.”

F	<b>Non-Functional &amp; Cross-Cutting</b>
<ul style="list-style-type: none"><li>As a user, I want dashboard APIs to respond in under 300 ms (local) so that the UI remains smooth and responsive</li><li>As a developer, I want detailed logs for errors and request timings so that debugging is efficient.</li><li>As a system, I want strong validation and CORS configuration so that data integrity and safety are ensured.</li><li>As an operator, I want environment-based configuration so that deployments don't require code changes.</li></ul>	
	

Success:

- Cached lookups; indexed DB queries.
- Logs contain method, path, status, and latency.
- ☐ Trimmed, uppercased inputs; valid fields only.
- ☐ CORS allows only configured frontend origin.

Failure Scenarios:

- ☐ Slow query (>300ms) → log performance warning.
- ☐ Cache miss → fallback still returns correct data.
- ☐ Logging service fails → fallback to file logging.
- ☐ Stack traces must never include sensitive info.
- ☐ Missing or invalid data → “Invalid request.”
- ☐ CORS misconfiguration → frontend blocked with error in console.

