# Software Requirements Specification

# PRJ566 – Winter 2025

# PRJ566 – Team No:

Team 06

# Name of Project:   Restaurant Management Application – ChowHub

# Project Leader:

Mostafa Hasanalipourshahrabadi

**Last updated:**

**Team Members:**

Tingchen Tsao - 107253239

Lily Huang - 180923211

Furkan Bas - 121540215

Saad Ghori – 155972227

# TABLE OF CONTENTS

1. **Introduction/Overview - Document Information**
   1. **Document Authors**
   2. **Revision History**
   3. **Document Conventions**
   4. **Document Purpose**
   5. **Intended Audience**
   6. **Group Agreement**
2. **Project Overview**
   1. **Project Proposal**
   2. **Stakeholders and Users**
   3. **Project Scope**
   4. **System risks**
   5. **Operating Environment**
   6. **Functional Requirements**
   7. **Nonfunctional Requirements**
   8. **UI/UXD Interface Mockups**
3. **Process & Data Modeling**
   1. **UML Modeling: DFDs & Activity Diagrams**
   2. **Use Case Specification** 
      1. **Business Rules**
      2. **System Use Case Diagrams**
      3. **Use Case Description Tables**
4. **Domain Class Diagram**
5. **Database (Select either 5.1 or 5.2)**
   1. **RDBMS Artifacts**
      1. Scripts to create, populate, delete tables
      2. Data Dictionary
   2. **NoSQL Artifacts**
6. **Work breakdown Structure (WBS)**
7. **Milestones & Acceptance Criteria**
8. **Implementation Schedule (Agile/Waterfall)**
9. **Client / Faculty Sign-off**

# 1 - Introduction/Overview - Document Information

## 1.1 Document Authors

Mostafa Hasanalipourshahrabadi, Tingchen Tsao, Lily Huang, Furkan Bas, Saad Ghori

## 1.2 Revision History

|  |  |
| --- | --- |
| Week 03 | 1. Introduction/Overview  1.1 Document Authors  1.2 Revision History  1.3 Document conventions  1.4 Document purpose  1.5 Intended audience  1.6 Group agreement  2.1 Project proposal |
| Week 04 | 2.2 Stakeholders and Users  2.3 Project Scope  2.4 System Risks  2.5 Operating Environment |
| Week 05 | 2.6 Functional Requirements  2.7 Nonfunctional Requirements  3.1 Data Flow Diagram |
| Week 06 | 2.8 UI/UXD Interface Mockups  3.1 Activity Diagrams |
| Week 07 | 3.2 Use Case Specification  3.2.1 Business Rules  3.2.2 System Use Case Diagrams  3.2.3 Use Case Description Tables |
| Week 08 |  |
| Week 09 | 4. Domain Class Diagram |
| Week 10 | 5. Database |
| Week 11 |  |
| Final |  |

## 1.3 Document Conventions

Any text in red indicates an exception or error.

Any text in blue is in-progress.

Any text highlighted in yellow is an important point.

Any text in green was recently added.

Any text *italicized* represents definitions.

Any text with ~~strike-through~~ is deleted.

## 1.4 Document Purpose

The purpose of this Software Requirements Specification (SRS) document is to clearly outline the requirements for developing the ChowHub Restaurant Management Application. It provides a detailed description of the system’s features, goals, and limitations, ensuring everyone involved in the project understands what needs to be built and how it will work.

This document is meant to guide the development team as they design, build, and test the application, while helping the project manager plan resources and timelines. It also ensures that restaurant stakeholders know what to expect from the application and can refer to it for any future updates or improvements. By clearly listing the project’s requirements, this SRS helps avoid misunderstandings, reduces risks, and makes it easier for everyone to work together effectively.

## 1.5 Intended Audience

The intended audience for this Software Requirements Specification (SRS) document includes everyone involved in the project. The development team—software developers, testers, and designers—will use it to build and test the system according to the requirements. The project manager will refer to it to manage resources, track timelines, and ensure the project stays on schedule. Restaurant owners or managers, as clients, may review the document to ensure the system meets their needs. The quality assurance team will use the requirements to check that the system works correctly, is user-friendly, and reliable. System integrators, who connect ChowHub to tools like POS platforms, will rely on it to ensure smooth integration. Technical writers might use the SRS to create user manuals or training materials. Lastly, future teams or vendors working on updates or expansions can refer to the document to understand the original system requirements and keep everything consistent. This SRS helps all parties work together effectively and stay aligned.

## 1.6 Group Agreement

**TEAM AGREEMENT**

**Team #:** 06

**Project Title:** ChowHub

**Project Time Frame:** January 6, 2025 – August 15, 2025

**Team Members:**

Furkan Bas

Lily Huang

Saad Ghori

Tingchen Tsao

**Team Leadership:** Mostafa Hasanalipourshahrabadi

**Team Functions:**

* Attend all scheduled online meetings on the team's channel on MS Teams.
* Communicate and collaborate through MS Teams chat or in the issues/discussion section of GitHub.
* Complete all assigned issues for the deliverables of the project by the designated soft deadline.
* Communicate in advance on the MS Teams channel or chat if there are any challenges or difficulties with any assigned tasks.
* Assist and support other team members to foster a collaborative and stress-free environment within the team.
* Submit the deliverable for the week by the soft deadline of 12:00 PM every Sunday. The group will review and finalize the work by 6:00 PM every Sunday.
* The group leader will submit the deliverable and review the GitHub repository by 9:00 PM every Sunday.
* If any member wishes to make a change after the leader has submitted, they must inform the team and obtain the leader's approval before making the change.

**Team Meetings:**

* The team will meet three times a week - Tuesday at 7:00 PM, Wednesday at 1:25 PM, and Sunday or Saturday at 1:00 PM, depending on team availability.
* The meetings will be on MS Teams, Team 06 channel.
* On Wednesdays, the team will meet with the professor to discuss and review the progress of the project. The other two meetings will be held within the team to discuss upcoming tasks, address any issues or concerns, and plan for the next steps.

**Team Problems:**

* **Conflict Resolution:** If conflicts arise, the team will address them through open discussion and mediation among its members and the leader. Should the issue remain unresolved, the matter will be escalated to the course instructor for further guidance.
* **Workload Distribution:** Tasks will be distributed equitably among all team members. If a team member is having trouble with their workload, they are encouraged to communicate with the team on the Teams’ channel, allowing others to collaborate and help.
* **No-Show Policy:** If a team member fails to complete the tasks assigned to them and does not communicate or attend meetings, the remaining team members have the authority to recommend their exclusion from the team, after notifying the course instructor.

**Team Commitment**

**The undersigned members agree to work together on the project until the end of the PRJ666 next Semester. They recognize that as a team and individually they are responsible for the quality of all deliverables.**

**Name Date**

|  |  |
| --- | --- |
| Saad Ghori | 01-26-2025 |
| Mostafa Hasanalipourshahrabadi | 01-26- |
| Tingchen Tsao | 01-26-2025 |
| Lily Huang | 01-26-2025 |
| Furkan Bas | 01-26-2025 |

# ShapeShapeShapeShapeShape2 - Project Overview

## 2.1 Project Proposal

**Project Background**

The restaurant industry has long faced operational challenges stemming from inefficient inventory management, fragmented systems, and difficulty in adapting to dynamic customer demands. Traditional approaches to managing inventory, employee shifts, menu operations, and supplier coordination often result in waste, overstocking, ingredient shortages, order cancellations, and dissatisfied customers. Additionally, restaurants often lack real-time insights into sales performance and ingredient usage, making it harder to optimize operations or make data-driven decisions. This project seeks to address these inefficiencies through a streamlined, centralized solution—ChowHub. ChowHub integrates inventory management, automated menu updates, advanced sales analysis, supplier coordination, employee management, shift tracking, and secure POS integration, tailored specifically for the fast-paced, small-to-mid-sized restaurant environment.

**Problem Statement**

|  |  |
| --- | --- |
| The Problem of: | Inefficient inventory tracking, ingredient shortages, overstocking, waste, fragmented operational systems, and limited data-driven decision-making. |
| Affects: | Restaurant owners, managers, staff (including waitstaff and kitchen staff), and ultimately, customers. |
| The impact of which is: | Increased operational costs, loss of revenue due to waste or missed sales opportunities, reduced employee productivity, lower customer satisfaction, and difficulty in scaling operations effectively. |
| A successful solution would: | * Reduce waste through real-time inventory tracking and customizable low-stock alerts. * Automate menu updates based on ingredient availability to prevent unfulfilled orders. * Streamline employee and shift management, reducing manual administrative tasks. * Improve operational efficiency by integrating with existing POS systems to avoid costly replacements. * Enhance customer satisfaction by ensuring timely service, consistent food quality, and minimal operational disruptions. * Provide actionable analytics, such as identifying top-selling menu items, peak sales times, and cost-saving opportunities. * Prioritize security and privacy by implementing strong encryption standards and enabling on-premises data storage. |

**Product Vision**

|  |  |
| --- | --- |
| For | Small-to-mid-sized restaurant owners or managers who need a cost-effective and efficient way to handle inventory, supplier coordination, menu operations, and employee management. |
| Who | Require a streamlined, integrated system to track inventory, manage suppliers, assign shifts, and optimize backend operations while improving customer experience. |
| The Product Name | ChowHub |
| That | Provides real-time inventory tracking, automated menu management, advanced analytics, shift management, and secure user authentication—all seamlessly integrated with POS systems. |
| Unlike | Traditional systems that focus on singular aspects of restaurant management, require costly replacements, or lack customization, |
| Our product | Offers a centralized, scalable, and secure solution that enhances operational efficiency, reduces waste, improves profitability, and prioritizes data privacy—all without disrupting existing workflows. |

## 2.2 Stakeholders and Users

|  |  |  |
| --- | --- | --- |
| **Stakeholder Name/Identifier** | **Category** | **Role/Responsibilities** |
| **CEO (Chief Executive Officer)** | Administration, Sponsor | Oversees the overall project and ensures resources are allocated; provides executive support and vision. |
| **Construction Manager and Scheduler** | Administration, User | Needs accurate up-to-date information for costing, scheduling, and ensuring smooth execution of the project details. |
| **Administrative Assistant** | User | Assists with project coordination and document management, ensuring smooth communication and information flow. |
| **Schedulers** | User | Responsible for organizing and maintaining the project timeline and coordinating task schedules. |
| **Cost Accountant** | User | Tracks project costs, manages budgets, and ensures financial resources are allocated appropriately. |
| **Project Leader** | Developers | Oversees the development team, ensures project tasks are on track, and manages technical requirements and milestones. |
| **Developers** | Developers | Work on coding, system design, and implementation based on project requirements, collaborating with other team members. |
| **Restaurant Owners/Managers** | End Users, Client | Utilize the ChowHub platform for inventory, menu, and employee management. Ensure the system meets the operational needs of their restaurant. |
| **Quality Assurance (QA) Team** | Developers | Responsible for testing ChowHub to ensure it works as intended, identifying bugs, and verifying that features are functional before release. |
| **System Integrators** | Developers | Work on integrating ChowHub with existing systems such as POS platforms and supplier databases, ensuring compatibility and smooth operation. |
| **End Users (Restaurant Staff)** | End Users | Use ChowHub for tasks like placing orders, tracking ingredients, and managing shift schedules, enhancing the overall restaurant workflow. |
| **Technical Writers** | Documentation | Develop user manuals, training materials, and support documents to guide end-users in effectively using ChowHub. |

## 2.3 Project Scope

**Purpose:**

ChowHub aims to simplify restaurant operations and enhance customer experience by providing an easy-to-use, all-in-one management solution for small independent restaurants. With features such as advanced menu management, real-time ingredient inventory tracking, and intuitive ordering, the ChowHub platform ensures cost-effectiveness and efficiency in the restaurant's back-of-house operations while improving customer satisfaction in front of the house.

**Objective:**

To develop a comprehensive, all-in-one web-based restaurant management system that automates and integrates essential restaurant operations, including menu updates, real-time ingredient tracking, and order processing. The platform will support multi-level user access and provide management reports along with real-time insights.   
   
**In-Scope Features:**

* **Menu management system** that dynamically updates item availability based on inventory.
* **Supplier management module** for tracking ingredient sources and reordering supplies.
* **Real-time ingredient tracking** to monitor stock levels and prevent shortages.
* **Integration with POS systems** for seamless data synchronization.
* **User authentication with role-based access** for different levels of staff and management.
* **Sales reporting and analytics** to help restaurants track performance and trends.
* **Employee management features**, including shift scheduling and payroll tracking.

**Out-of-Scope Features:**

* **Full POS system development** (ChowHub is designed to function independently but can integrate for added convenience).
* **Mobile application** (initial version will be web-based only).
* **Advanced AI-driven forecasting** for ingredient usage and demand prediction.
* **Community Forum**
* **Loyalty and Marketing**

**High Level Constraints and Assumptions:**

**Regulatory Compliance:** ChowHub will comply with data privacy regulations outlined in PIPEDA for handling any customer and business data. Moreover, the system’s design will follow AODA (Accessibility for Ontarians with Disabilities Act) guidelines to ensure accessibility for all users of the platform.

**Assumptions:**

* Users will access the system exclusively through web browsers on desktops or tablets; a mobile app will not be included in the initial release.
* Restaurants using ChowHub already have an existing POS system, which will be integrated.
* Internet connectivity will be required at all times for real-time updates and synchronization.

**Project Deliverables:**

* Fully functional web-based restaurant management system with in-scope features.
* Secure integration APIs for connecting with third-party POS systems.
* User guides & training materials to guide restaurant staff on system usage.
* Successful testing reports relating to performance and usability.

**Success Criteria:**

* **User Satisfaction:** Positive feedback from users in the beta testing phase.
* **Regulatory Compliance:** Full adherence to **PIPEDA** and **AODA** requirements.
* **Seamless POS Integration:** Successful integration and synchronization of data with POS system of choice.
* **Project Completion:** All deliverables delivered within the allocated timeline (January–August 2025).
* **System Reliability:** Minimal downtime or disruptions.
* **Lighthouse Report Compliance:** Achieve industry-standard Lighthouse scores (performance ≥90, accessibility ≥90, best practices ≥90, SEO ≥90) to ensure the application meets benchmarks for speed, usability, and accessibility.

**Overview of System Interactions:**

ChowHub is designed as a standalone restaurant management system with the ability to integrate with third-party POS systems and supplier platforms. The system interactions will primarily focus on real-time data synchronization for inventory tracking, order management, and employee scheduling.

**Key Integrations**

1. **POS System Integration**
   * ChowHub will provide **API** to allow integration with existing **POS systems** used by restaurants.
   * Data exchanged will include **order details**, **ingredient consumption**, and **sales records**, ensuring seamless updates between ChowHub and the POS system.
   * The integration will allow for **automatic deduction of ingredients** when an order is placed in the POS system, preventing stock shortages.
   * If integration is not available, ChowHub will function independently, requiring manual inventory updates.
2. **Supplier and Inventory Management**
   * ChowHub will support **supplier tracking and reordering automation**, allowing restaurants to place new ingredient orders directly through the system.
   * Potential integration with supplier platforms could be explored in future versions for **automated restocking** based on ingredient levels.
3. **User Authentication & Access Control**
   * The system will include **role-based access control** to ensure secure access for restaurant owners, managers, and staff.
   * Secure login and **data encryption** mechanisms will be implemented to protect sensitive business information.
4. **Sales and Analytics Reporting**
   * Integration with **reporting and visualization tools** to provide real-time **business insights**, including sales trends, ingredient consumption, and revenue tracking.
   * Export options for **financial and operational reports** in formats such as CSV and PDF for easy record-keeping.
5. **System Testing & Mock POS Environments**
   * Since real POS system access may not be available during development, ChowHub will be tested using **mock POS environments** to simulate real-world integration scenarios.
   * Unit testing and performance testing will be conducted to ensure stability before deployment.

## 2.4 System Risks

|  |  |
| --- | --- |
| **Risk** | **Response** |
| Integrating with the POS system may pose a risk if the integration is not successful. | ChowHub’s features are standalone and do not require integration into the POS system. However, integration would enhance ease of use. |
| There may be compliance risks related to storing employee data. | The system will not handle sensitive data such as employee social insurance numbers. It will only manage performance data associated with employee IDs. |
| There may be technical risks if access to certain technologies, such as a POS system or cashier terminal, is unavailable. | We can create mock POS systems to simulate how we will interface with the actual POS system or cashier terminal. |
| There may be risks associated with system downtime. If the POS system is unavailable, the restaurant will not be able to process orders and could lose revenue. | We can implement unit testing and other tests to ensure compatibility. Integration can be done during off-peak hours to minimize disruptions. |
| Resource and time risks. We may not have enough time to dedicate to ChowHub, which could result in incomplete development. | We can adjust the project scope to focus on completing the core features of ChowHub, with additional features implemented in later phases. |
| There may be a risk that restaurant staff and management might be resistant to adopting the new system due to unfamiliarity or concerns about usability. | We can provide training and user-friendly documentation to ensure smooth adoption. A gradual implementation phase may also help staff adapt to the new system. |
| There could be difficulties when migrating data from existing systems to ChowHub, especially if the data is incomplete or inconsistent. | We can conduct thorough data validation and ensure a well-planned data migration strategy, including backup procedures and testing before the full implementation. |
| If ChowHub’s features grow over time, there may be risks related to system performance as the restaurant business expands or data volume increases. | We can design the system to be scalable from the start, using cloud services or modular architecture to handle increased demand as the system grows. |

## 2.5 Operating Environment

ChowHub is designed to operate in a restaurant management environment, ensuring seamless integration with existing workflows. The application will function across multiple platforms and environments, providing flexibility for restaurant owners and staff.

**Hardware Requirements**

* Client Devices: Computers, tablets, or smartphones used by restaurant owners, managers, and staff.
* POS System Compatibility: ChowHub must integrate with existing POS terminals without requiring hardware replacement.
* Server Requirements: Cloud-based or on-premise servers capable of handling database storage, analytics processing, and real-time updates.

**Software Requirements**

* Operating Systems: ChowHub will be compatible with Windows, macOS, iOS, and Android.
* Web Browser Support: The platform will run on modern browsers such as Google Chrome, Mozilla Firefox, Microsoft Edge, and Safari.
* Database Management: ChowHub will use **PostgreSQL** or **MongoDB** for efficient data storage and retrieval.

**Network Requirements**

* Internet Connection: Required for real-time updates, cloud synchronization, and remote access.
* Local Network Support: Restaurants with on-premise servers can operate within a local network in case of internet disruptions.

**Security Considerations**

* Data Encryption: Strong encryption will protect sensitive information such as payroll and sales data.
* Access Control: Role-based access ensures that only authorized users can modify critical data.
* Data Storage: Sensitive data can be stored locally within the restaurant to enhance privacy and security.

## 2.6 Functional Requirements

## **Registration & Authentication of User Accounts:**

## User registers for a new account.

## User logs in securely using a username and encrypted password.

## User resets password if password is forgotten.

## Administrator assigns and updates user roles and permissions.

## Administrator deletes users restricting access.

## **Track & Manage Ingredient Inventory:**

## Staff views current ingredient inventory levels.

## System updates ingredient quantities in real time when orders are processed.

## System sends low-stock alerts to designated users.

## Staff manually update ingredient quantities when purchases are made from suppliers.

## Staff update multiple ingredient quantities (batches) at once, streamlining large restocks or supplier deliveries.

## Staff records ingredient wastage with reasons, for example, spoilage, over-preparation, expired stock.

## The system tracks inventory based on purchases, sales, and wastage.

## **Manage Menu Items:**

## The manager adds new items on the menu with images, descriptions, pricing, ingredients and their quantities.

## Manager assigns menu items to categories for ease in navigation

## Manager hides or makes menu items visible to customers.

## Manager updates existing menu items. For e.g., price changes, ingredient updates etc.

## System automatically updates menu item availability based on current ingredient inventory.

## Manager applies promotion/discount to menu items

## **Supplier Coordination:**

## Manager views supplier contact details and order history.

## Manager sets low-inventory thresholds to trigger automated reorder alerts.

## Manager assigns suppliers to specific ingredients.

## Manager manually creates and tracks purchase orders to suppliers.

## Manager manually sets and updates order status.

## **Create and Manage Employee Schedules:**

## Manager creates and manages employee schedules.

## The system only allows managers to assign shifts based on Employee availability, position and business demand (e.g, peak hours or holidays).

## Staff view their assigned shifts via the system.

## Staff request shift change, swap or break (needs manager approval).

## System enforces labor laws and restaurant policies to prevent overtime violations.

## **Manage Staff Timesheets and Payrolls:**

## Staff will clock in and clock out using the system.

## The system records work hours, including break times.

## Manager reviews and approves timesheets before payroll processes.

## Overtime will automatically be calculated according to the restaurant policy.

## The manager sets hourly rates and overtime rates for staff.

## The system generates payroll reports with employee hours, pay rate, overtime compensation.

## **Place and Process Orders:**

## Staff will place customers’ orders through an interface

## Orders are sent to the kitchen upon submission.

## System updates and track inventory based on ingredient usage from orders (Track & Manage Ingredient Inventory).

## Order modifications or cancellation are allowed before kitchen preparation starts.

## The system notifies staff when orders are ready to deliver.

## Staff change order status through the order lifecycle (e.g. Order Placed, Preparing, Fulfilled, Cancelled)

## **Managing POS Integrations:**

## The system integrates with existing POS systems (sale transaction, order details, payment processing).

## System deducts inventory based on ingredient usage from orders.

## Ensures real-time synchronization of menu items and availability.

## Generate detailed sale reports from POS data.

## If POS integration is unavailable, staff can manually process orders, and the inventory will be adjusted by the system

## **Generate Reports & Analytics:**

## Managers use the system to generate inventory reports, showing stock levels, usage trends, and wastage details.

## System tracks wastage reasons (e.g., spoilage, over-preparation, expired stock) and compiles reports for cost analysis.

## The system provides manager sales reports, including total revenue, top-selling items, and daily/weekly/monthly sales trends.

## Managers use the system to generate employee performance reports, tracking work hours, sales contributions, and attendance records.

## Managers use the system to generate payroll reports, including employee hours, pay rates, overtime compensation, and approved timesheets.

## Manager exports reports in multiple formats (CSV, PDF, Excel) for further analysis.

## The system visualizes reports with charts and graphs for better insights.

## Managers apply filters and date ranges to get custom reports.

## **Administer System Settings:**

## Administrator/manager configures user roles and permissions for different levels of access.

## Administrator/manager sets low-inventory warning thresholds for ingredient tracking.

## The administrator/manager manages employee settings, such as overtime policies and shift scheduling rules.

## The administrator/manager defines menu categories and updates system-wide pricing settings.

## Administrator/manager sets business hours and holiday closures for scheduling automation.

## Administrator/manager manages notification preferences, such as alerts for low inventory, order status updates, and payroll processing reminders.

## Administrator/manager updates security settings, including password policies and session timeouts.

## **Support Multi-Platform Access:**

## Users can access ChowHub via web browsers on desktops, laptops, and tablets.

## The system is optimized for responsive design, ensuring usability across different screen sizes.

## All features are accessible without requiring installation of additional software.

## Secure login and authentication ensure safe access across multiple devices.

## The system maintains consistent data synchronization across devices to prevent conflicts.

## 2.7 Nonfunctional Requirements

**1. Compatibility and Portability**

* Operating Systems:
  + Requirement: The ChowHub system will be compatible with most operating systems.
  + Detail: Ensure the platform runs on common operating systems such as Windows, macOS, iOS, and Android.
* Web Browsers:
  + Requirement: The ChowHub system will be compatible with most web browsers.
  + Detail: Support modern browsers like Google Chrome, Mozilla Firefox, Microsoft Edge, and Safari for a consistent user experience.

**2. Interoperability**

* POS System Integration:
  + Requirement: The ChowHub system will be compatible with existing POS systems.
  + Detail: Provide standard APIs and integration protocols to ensure seamless communication between ChowHub and various POS platforms.

**3. Scalability**

* System Scalability:
  + Requirement: The ChowHub system will be scalable.
  + Detail: Design the system architecture to support increases in users, transactions, and data volume without performance degradation.

**4. Security**

* Data Encryption:
  + Requirement: The ChowHub system will use 256-bit AES encryption.
  + Detail: Protect sensitive data both in transit and at rest using industry-standard encryption protocols.
* Data Compliance and Storage:
  + Requirement: The ChowHub system will use local storage for data compliance.
  + Detail: Ensure data storage complies with regional data protection regulations by allowing local on-premises storage options.

**5. Reliability**

* System Reliability:
  + Requirement: The ChowHub system will be reliable.
  + Detail: Ensure high availability with minimal downtime through robust error handling, failover strategies, and regular system health monitoring.

**6. Adjustability (Configurability)**

* System Adjustability:
  + Requirement: The ChowHub system will be adjustable.
  + Detail: Provide configurable options for administrators to tailor settings, features, and user permissions based on evolving operational needs.

**7. Usability**

* User-Friendliness:
  + Requirement: The ChowHub system will be usable.
  + Detail: Offer an intuitive and accessible user interface that minimizes the learning curve for restaurant staff and management, ensuring ease of navigation and operation.

## 2.8 UI/UXD Interface Mock-ups

**2.8.1.1 Ordering Screen**

**A screenshot of a menu

AI-generated content may be incorrect.**

**2.8.1.2 Adding item to order**

**A screenshot of a computer

AI-generated content may be incorrect.**

**2.8.2.1 Creating Menu**

**A screenshot of a menu

AI-generated content may be incorrect.**

**2.8.2.1 Adding Product to Menu**

**A screenshot of a menu

AI-generated content may be incorrect.**

**2.8.2.2 Adding Category to Menu**

**A screenshot of a computer

AI-generated content may be incorrect.**

**2.8.3 Employee Management**

**2.8.3A Personal Details**

**A screenshot of a computer

AI-generated content may be incorrect.**

**2.8.3A Shift**

**A screenshot of a calendar

AI-generated content may be incorrect.**

**2.8.3A Pay Stub generation**

**A screenshot of a computer

AI-generated content may be incorrect.**

**2.8.4 Sale reports and analysis**

A screenshot of a chart

AI-generated content may be incorrect.A screenshot of a data analysis

AI-generated content may be incorrect.

**2.8.5 Ingredient tracking**

A screenshot of a computer

AI-generated content may be incorrect.

**2.8.6 Supplier Management**

**A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.**

**2.8.7 User Authentication**

**A screenshot of a computer

AI-generated content may be incorrect.**

**2.8.7A Registration**

**A screenshot of a computer

AI-generated content may be incorrect.**

**2.8.7A Sign In**

A screenshot of a login form

AI-generated content may be incorrect.

# Process and Data Modeling

## **3.1 UML/DFD Modeling and Data Modeling**

**Data Flow Diagram (DFD)**

A diagram of a company

AI-generated content may be incorrect.

The Data Flow Diagram (DFD) illustrates the flow of information between different stakeholders and system components in ChowHub, ensuring efficient restaurant management. This diagram represents how ChowHub processes and manages data related to inventory, orders, employee management, and reporting.

**Key Components:**

1. External Entities:

* Customer: Places an order through restaurant waitstaff.
* Restaurant WaitStaff: Inputs customer orders into the system.
* Restaurant Manager: Manages the restaurant menu, updates employee information and shifts, orders restocking upon notification and receives insights for decision-making.
* Supplier: Handles restocking requests from ChowHub.

2. Processes & Data Flow:

* Orders & Menu Management:
  + The restaurant waitstaff places orders in ChowHub, which processes them.
  + The menu database is updated when items become unavailable due to ingredient shortages.
* Inventory Management:
  + ChowHub tracks ingredient levels through the inventory database.
  + If stock is low, a notification is sent to the restaurant manager.
  + Restocking requests are sent to suppliers when necessary.
* Employee & Payroll Management:
  + ChowHub stores and processes employee information.
  + It generates payroll data for the employee management system.
* Sales & Reporting:
  + ChowHub generates insights and reports for restaurant managers to track performance and make business decisions.

**Purpose of the DFD:**

This DFD highlights how different stakeholders, databases, and processes interact with ChowHub, ensuring seamless restaurant operations. It also emphasizes key functionalities like order processing, inventory tracking, payroll generation, and reporting, making it a vital reference for system implementation.

**Activity Diagram**

**A diagram of a restaurant staff

AI-generated content may be incorrect.**

This activity diagram represents the workflow of ChowHub when processing restaurant orders, managing inventory, and triggering restocking requests.

**1. Order Processing**

* Restaurant staff receives an order from a customer.
* The staff inputs the order into the system, which updates the inventory.

**2. Inventory Management**

* The system checks whether any ingredient has fallen below the low threshold.
* If no ingredient is below the threshold, the system simply updates sales records and ends the process.
* If an ingredient is below the threshold, the system will:
  + Disable menu items that require the missing ingredient.
  + Notify management about the low inventory.

**3. Restocking Decision**

* The system asks whether restocking is needed.
* If restocking is not needed, the process ends.
* If restocking is required, the system will:
  + Retrieve supplier information.
  + Displays supplier lists and prompts to select and place a restocking request.
    - Updates purchase history.

## **3.2 Use Case Specifications**

## **3.2.1. Business Rules**

|  |  |  |
| --- | --- | --- |
| Business Rule Number | Business Rule Description | Related UC |
| BR01 | |  | | --- | | Sales data must be recorded and analyzed for reporting. |  |  | | --- | |  | | UC01-UC02 |
| BR02 | The user must be logged in to access Sales Reports and analysis. | UC01-UC02 |
| BR03 | Only managers can access sales reports and analysis. | UC01-UC02 |
| BR04 | Reports can be filtered by dates | UC01-UC02 |
| BR05 | Users must be logged in to access Ingredient tracking reports. | UC03-UC04-UC05 |
| BR06 | Managers and waitstaff can assess ingredients tracking reports. | UC03-UC04-UC05 |
| BR07 | Only Managers can add, remove, or update ingredients levels | UC03-UC04-UC05 |
| BR08 | Restaurant staff must input customer orders into the system for processing. | UC06 |
| BR09 | Inventory levels must be updated automatically when an order is placed. | UC06 |
| BR10 | Only authorized staff (e.g., waitstaff or order takers) can initiate an order and they must be logged in. | UC06 |
| BR11 | |  | | --- | | If an ingredient falls below a threshold, related menu items should be disabled. |  |  | | --- | |  | | UC06 |
| BR12 | |  | | --- | | The system must notify management when stock is low. |  |  | | --- | |  | | UC06 |
| BR13 | |  | | --- | | Only managers can create and modify menu items. |  |  | | --- | |  | | UC07 |
| BR14 | In Menu creation, all required fields (name, description, price, ingredients) must be provided. The system validates inputs for completeness. | UC07 |
| BR15 | Only Manager can add categories in Menu and they need to be logged in to do so. | UC08 |
| BR16 | The system must allow managers to view and modify employee details upon login. | UC09-UC10 |
| BR17 | Employees can view their assigned shifts, and managers can update shift schedules. | UC11 |
| BR118 | |  | | --- | | The system must generate payroll data based on employee work hours. |  |  | | --- | |  | | UC12 |
| BR19 | The system must allow managers to add supplier details and update supplier information. | UC13 |
| BR20 | |  | | --- | | Restocking requests must be sent to suppliers when stock reaches a critical level. |  |  | | --- | |  | | UC13 |
| BR21 | Only restaurant managers can create, modify, or delete user accounts and assign roles. | UC14 |
| BR22 | To add new user, manager must provide first name, last name, username, role, phone number, employee id (filled automatically), address, postal code, email address of the new user. | UC14 |
| BR23 | Users must provide a valid email, and password (min. 8 characters, alphanumeric) to register. | UC14 |
| BR24 | |  | | --- | | Users must enter valid credentials (employee id, username, password) to sign in. |  |  | | --- | |  | | UC15 |

## **3.2.2. System Use Case Diagrams.**

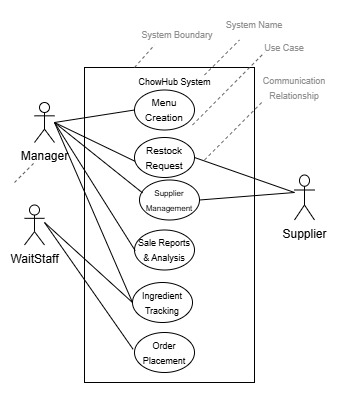


Figure 1. Partial Use Case Context Diagram – Menu, Order, Sales & Restocking

A diagram of a user authentication

AI-generated content may be incorrect.

Figure 2 Partial Use Case Context Diagram - Employee Management

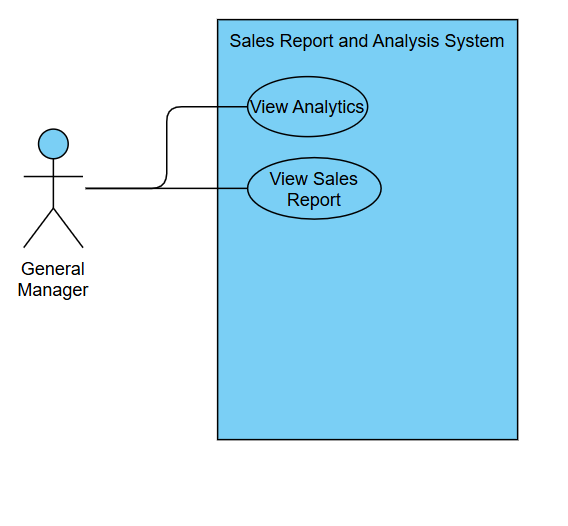
## **3.2.3 Use Case Descriptions.**

**3.2.3.1. Sales Reports and analysis**

1- Business Rules.

* Sales data must be recorded and analyzed for reporting.
* The user must be logged in to access Sales Reports and analysis.
* Only managers can access sales reports and analysis
* The user can filter by date

2- System Use Case Diagrams.



3- Use Case Descriptions.

|  |  |
| --- | --- |
| Use Case Name: | View Analytics |
| ID | UC01 |
| Importance | Low |
| Primary Actor | General Manager |
| Use Case Type | Managing. Analytics. |
| Stakeholders and interest | General Manager – Wants to see data on how the store could improve |
| Brief Description: | This use case describes ChowHub could display analytics to the general manager |
| Trigger:  Type: | General Manager wants to view the analytics  Internal |
| Relationships: | |  |  | | --- | --- | | Association | General Manager | | Include |  | | Extend |  | | Generalization |  | |
| Normal Flow of events: | |  |  | | --- | --- | | 1. | Capture current view settings | | 2. | Capture transactions based on view settings | | 3. | Create analysis based on captured transactions | | 4. |  | | 5. |  | |
| Sub Flows: |  |
| Alternate Flows: |  |

|  |  |
| --- | --- |
| Use Case Name: | View Sales Report |
| ID | UC02 |
| Importance | Low |
| Primary Actor | General Manager |
| Use Case Type | Managing. Analysis. |
| Stakeholders and interest | General Manager – Wants an overview on transactions |
| Brief Description: | This use case describes ChowHub could display a sales report to the general manager |
| Trigger:  Type: | General Manager wants to view the sales report  Internal |
| Relationships: | |  |  | | --- | --- | | Association | General Manager | | Include |  | | Extend |  | | Generalization |  | |
| Normal Flow of events: | |  |  | | --- | --- | | 1. | Capture current view settings | | 2. | Capture transactions based on view settings | | 3. | Create display for transactions | | 4. |  | | 5. |  | |
| Sub Flows: |  |
| Alternate Flows: |  |

Post Condition:  
4- Corresponding Mockups

A screenshot of a chart

AI-generated content may be incorrect.A screenshot of a data analysis

AI-generated content may be incorrect.

**3.2.3.2 Ingredient tracking**

1- Business Rules.

* Users must be logged in to access Ingredient tracking reports.
* Managers and waitstaff can access ingredientst tracking reports.
* Only Managers can add, remove, or update ingredients levels

2- System Use Case Diagrams.

A diagram of a recipe

AI-generated content may be incorrect.

3- Use Case Descriptions.

|  |  |
| --- | --- |
| Use Case Name: | Add Ingredient |
| ID | UC03 |
| Importance | High |
| Primary Actor | General Manager |
| Use Case Type | Managing. Ingredients. |
| Stakeholders and interest | General Manager – Wants to add a new ingredient |
| Brief Description: | This use case describes how ChowHub would save a new ingredient |
| Trigger:  Type: | General Manager wants to add a new ingredient  Internal |
| Relationships: | |  |  | | --- | --- | | Association | General Manager, Ingredient, Supplier, Dish | | Include |  | | Extend |  | | Generalization |  | |
| Normal Flow of events: | |  |  | | --- | --- | | 1. | Capture Ingredient name | | 2. | Add association with supplier | | 3. | Add association with dish | | 4. | Add ingredient to list of ingredients | | 5. |  | |
| Sub Flows: |  |
| Alternate Flows: |  |

|  |  |
| --- | --- |
| Use Case Name: | Remove Ingredient |
| ID | UC04 |
| Importance | High |
| Primary Actor | General Manager |
| Use Case Type | Managing. Ingredients. |
| Stakeholders and interest | General Manager – Wants to delete an ingredient |
| Brief Description: | This use case describes how ChowHub would delete an ingredient |
| Trigger:  Type: | General Manager wants to remove an ingredient  Internal |
| Relationships: | |  |  | | --- | --- | | Association | General Manager, Ingredient | | Include |  | | Extend |  | | Generalization |  | |
| Normal Flow of events: | |  |  | | --- | --- | | 1. | Capture Ingredient ID | | 2. | Remove Ingredient from list based on ID | | 3. |  | | 4. |  | | 5. |  | |
| Sub Flows: |  |
| Alternate Flows: |  |

|  |  |
| --- | --- |
| Use Case Name: | Update Ingredient |
| ID | UC05 |
| Importance | High |
| Primary Actor | General Manager |
| Use Case Type | Managing. Ingredients. |
| Stakeholders and interest | General Manager – Wants to update an ingredient |
| Brief Description: | This use case describes how ChowHub would update an ingredient |
| Trigger:  Type: | General Manager wants to update an ingredient  Internal |
| Relationships: | |  |  | | --- | --- | | Association | General Manager, Ingredient | | Include |  | | Extend |  | | Generalization |  | |
| Normal Flow of events: | |  |  | | --- | --- | | 1. | Capture Ingredient ID | | 2. | Return data on ingredient | | 3. | Return ingredient to ingredient list with updated information | | 4. |  | | 5. |  | |
| Sub Flows: |  |
| Alternate Flows: |  |

4- Corresponding Mockups

A screenshot of a computer

AI-generated content may be incorrect.

**3.2.3.3 Order Placement**

1- Business Rules:

* Users must be logged in to place an order.
* Only authorized staff (e.g., waitstaff or order takers) can initiate an order.
* The system must validate ingredient availability before confirming the order.
* Inventory must be updated when order is placed.

2- System Use Case Diagrams.

A diagram of a payment process

AI-generated content may be incorrect.

3- Use Case Descriptions.

|  |  |
| --- | --- |
| Use Case Name: | Place an Order |
| ID | UC06 |
| Importance | High |
| Primary Actor | Waitstaff |
| Use Case Type | Order Transaction. Order Processing. |
| Stakeholders and interest | Waitstaff: Wants to process orders seamlessly.  Manager: Requires real-time inventory updates and order records.  Customer: Expects prompt service and order accuracy. |
| Brief Description: | This use case details how a staff member places a customer order, updating inventory, and sends order details for preparation and billing. |
| Trigger:  Type: | Waitstaff initiates an order when a customer is ready to order  Internal |
| Relationships: | |  |  | | --- | --- | | Association | Waitstaff | | Include | Inventory update, Order confirmation, Payment processing | | Extend |  | | Generalization |  | |
| Normal Flow of events: | |  |  | | --- | --- | | 1. | The waitstaff selects one or more items and specifies quantities, modifiers, and variations. | | 2. | The system calculates the subtotal, Tax and total payment amount and displays the order summary for confirmation. | | 3. | The waitstaff confirms the order. | | 4. | The system updates inventory and creates a record of the order. | | 5. | The system generates an order confirmation. | |
| Sub Flows: | Payment Processing:Integrated POS system processes the payment securely. |
| Alternate Flows: | Order Cancellation: If the order is canceled before confirmation, the system discards the order data without updating inventory. |

4- Corresponding Mockups:

A screenshot of a menu

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**3.2.3.4 Menu Creation**

1- Business Rules:

* Only roles with authorization (admin/manager) can add new menu products or categories.
* All required fields (name, description, price, ingredients) must be provided.
* The system validates inputs for completeness.

2- System Use Case Diagrams.

A diagram of a person with a stick figure

AI-generated content may be incorrect.

3- Use Case Descriptions.

|  |  |
| --- | --- |
| Use Case Name: | Add a Menu Product |
| ID | UC07 |
| Importance | Medium |
| Primary Actor | Manager |
| Use Case Type | Configuration. Management. |
| Stakeholders and interest | Manager: Wants to update menu to reflect current offerings  Waitstaff: Wants an up-to-date menu |
| Brief Description: | This use case describes how a manager adds a new product to the menu, including the product’s image, description, pricing, modifiers, and ingredient details. |
| Trigger:  Type: | The manager selects the "Add Product" option from the menu management interface.  Internal |
| Relationships: | |  |  | | --- | --- | | Association | Manager | | Include | Data Validation | | Extend |  | | Generalization |  | |
| Normal Flow of events: | |  |  | | --- | --- | | 1. | The manager selects "Add Product." | | 2. | The system displays fields for entering product details (name, description, price, variants, ingredients, modifiers). | | 3. | The manager enters the required details and clicks save. | | 4. | The system validates and adds the new product to the menu | |
| Sub Flows: |  |
| Alternate Flows: | If required fields are missing, the system displays an error focusing on the problematic field. |

|  |  |
| --- | --- |
| Use Case Name: | Add a Menu Category |
| ID | UC08 |
| Importance | Medium |
| Primary Actor | Manager |
| Use Case Type | Configuration. Management. |
| Stakeholders and interest | Manager: Wants to organize menu products into logical, easily navigable groups.  Waitstaff: Wants a well-organized and easy to navigate menu. |
| Brief Description: | This use case outlines how a manager adds a new category to the menu to help structure and organize products effectively. |
| Trigger:  Type: | The manager selects the "Add a Category" option from the menu management interface.  Internal |
| Relationships: | |  |  | | --- | --- | | Association | Manager | | Include | Data Validation | | Extend |  | | Generalization |  | |
| Normal Flow of events: | |  |  | | --- | --- | | 1. | The manager selects "Add a Category." | | 2. | The system displays fields for entering category details (name, description, parent category). | | 3. | The manager enters the required details and clicks save. | | 4. | The system validates and adds the new category to the menu | |
| Sub Flows: |  |
| Alternate Flows: | If required fields are missing, the system displays an error focusing on the problematic field. |

4- Corresponding Mockups:   
A screenshot of a menu

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**3.2.3.5. Employee Details**

1- Business Rules.

* User must be logged in to access employee details
* Only managers and administrators can add, update, or remove employee details.

2- System Use Case Diagrams.

A diagram of a person with text

AI-generated content may be incorrect.

3- Use Case Descriptions.

|  |  |
| --- | --- |
| Use Case Name: | Manage Employee Details |
| ID | UC09 |
| Importance | High |
| Primary Actor | Manager / Administrator |
| Use Case Type | Managing. Employees. |
| Stakeholders and interest | Manager – Needs accurate employee details for scheduling. |
| Brief Description: | This use case describes how a manager or administrator can view, update, and manage employee records. |
| Trigger:  Type: | Manager wants to update an employee’s information  Internal |
| Relationships: | |  |  | | --- | --- | | Association | Manager, Administrator | | Include |  | | Extend |  | | Generalization |  | |
| Normal Flow of events: | |  |  | | --- | --- | | 1. | Manager selects "Employee Management" from the dashboard. | | 2. | System displays the employee list with details. | | 3. | Manager selects an employee and clicks "Edit" | | 4. | Manager updates name, role or contact details. | | 5. | System validates input and saves the updated details. | |
| Sub Flows: |  |
| Alternate Flows: |  |
| Post Condition: | Employee details are updated and stored successfully. |

|  |  |
| --- | --- |
| Use Case Name: | Delete Employee Record |
| ID | UC10 |
| Importance | Medium |
| Primary Actor | Manager / Administrator |
| Use Case Type | Managing. Employees. |
| Stakeholders and interest | Manager – Ensures the active employee list remains accurate. |
| Brief Description: | This use case describes how a manager or administrator can delete an employee record. |
| Trigger:  Type: | Manager wants to remove an employee from the system.  Internal |
| Relationships: | |  |  | | --- | --- | | Association | Manager, Administrator | | Include |  | | Extend |  | | Generalization |  | |
| Normal Flow of events: | |  |  | | --- | --- | | 1. | Manager selects "Employee Management" from the dashboard. | | 2. | System displays the employee list with details. | | 3. | Manager selects an employee and clicks "Delete". | | 4. | System asks for confirmation before deletion. | | 5. | System updates the employee list and logs the deletion action. | |
| Sub Flows: | If canceled, system returns to the employee list without changes. |
| Alternate Flows: | If an inactive employee is rehired, the manager can restore their record. |
| Post Condition: | Employee is removed from the active list but remains in the archive for record-keeping. |

4- Corresponding Mockups

A screenshot of a computer

AI-generated content may be incorrect.

**3.2.3.6 Employee Shifts**

1- Business Rules.

* User must be logged in to access employee shifts
* Only managers and administrators can assign and approve shifts.

2- System Use Case Diagrams.

A diagram of a person with a stick figure

AI-generated content may be incorrect.

3- Use Case Descriptions.

|  |  |
| --- | --- |
| Use Case Name: | Manage Employee Shifts |
| ID | UC11 |
| Importance | High |
| Primary Actor | General Manager |
| Use Case Type | Managing. Scheduling. |
| Stakeholders and interest | Manager – Ensures efficient shift planning without conflicts. |
| Brief Description: | This use case describes how a manager assigns shiftsand request shift changes. |
| Trigger:  Type: | Manager creates or updates a shift schedule  Internal |
| Relationships: | |  |  | | --- | --- | | Association | Manager, Employee, Shift Schedule | | Include |  | | Extend |  | | Generalization |  | |
| Normal Flow of events: | |  |  | | --- | --- | | 1. | Manager selects "Shift Management" from the dashboard. | | 2. | System displays shift calendar with current assignments. | | 3. | Manager selects an employee and assigns a shift. | | 4. | System validates the shift, ensuring no conflicts. | | 5. |  | |
| Sub Flows: |  |
| Alternate Flows: | System alerts if the maximum shift hours are exceeded. |
| Post Condition: | Employee details are updated and stored successfully. |

4- Corresponding Mockups

A screenshot of a calendar

AI-generated content may be incorrect.

**3.2.3.7 Generate Pay Stubs**

1- Business Rules.

* User must be logged in to access pay stab generation
* only Managers and Administrators can generate pay stubs.

2- System Use Case Diagrams.

A diagram of a person's structure

AI-generated content may be incorrect.

 3- Use Case Descriptions

|  |  |
| --- | --- |
| Use Case Name: | Generate Pay Stubs |
| ID | UC12 |
| Importance | High |
| Primary Actor | General Manager |
| Use Case Type | Managing. Payroll. |
| Stakeholders and interest | Manager – Needs accurate payroll processing for staff.  Employee – Wants to receive correct pay for hours worked. |
| Brief Description: | This use case describes how a manager selects an employee, chooses a pay period, adds optional earnings/deductions, and generates a pay stub with a live preview before finalizing. |
| Trigger:  Type: | Manager processes payroll.  Internal |
| Relationships: | |  |  | | --- | --- | | Association | Manager, Payroll System | | Include |  | | Extend |  | | Generalization |  | |
| Normal Flow of events: | |  |  | | --- | --- | | 1. | Manager navigates to the "Generate Paystub" section. | | 2. | System displays employee and pay period selection options. | | 3. | Manager selects an employee and a pay period. | | 4. | System retrieves recorded work hours and earnings. | | 5. | Manager enters additional earnings and deductions if needed. | | 6. | System updates the paystub preview in real-time. | | 7. | Manager reviews and clicks "Generate" to finalize. | | 8. | System saves the pay stub with download/print options. | |
| Sub Flows: |  |
| Alternate Flows: | If no recorded work hours exist for the selected pay period, system displays an error message.  If a deduction exceeds earnings, system warns the manager before proceeding. |
| Post Condition: | Pay stub is successfully generated and stored for payroll records. |

4- Corresponding Mockups

A screenshot of a computer

AI-generated content may be incorrect.

**3.2.3.8 Maintain Supplier**

1 - Business Rules:

* The system must allow managers to add supplier details and update supplier information.
* Restocking requests must be sent to suppliers when stock reaches a critical level.

2 - System Use Case Diagrams:

A diagram of a person with a person figure and text

AI-generated content may be incorrect.

3 - Use Case Descriptions:

|  |  |
| --- | --- |
| Use Case Name: | Maintain Supplier |
| ID | UC13 |
| Importance | high |
| Primary Actor | General Manager |
| Use Case Type | Managing. |
| Stakeholders and interest | General Manager – Wants to add a supplier to system database |
| Brief Description: | This use case describesthe process of adding a supplier to chowHub application |
| Trigger:  Type: | General Manager wants to add a supplier  Internal |
| Relationships: | |  |  | | --- | --- | | Association | General Manager | | Include |  | | Extend |  | | Generalization |  | |
| Normal Flow of events: | |  |  | | --- | --- | | 1. | User navigates to the Supplier page and press seeting icon to “Add supplier” | | 2. | User enters all required information (e.g., name, product category, phone number, email, address, etc.) | | 3. | System validates the provided information | | 4. | If validation is successful, the system saves the supplier details in the database. | | 5. | System confirms the supplier has been successfully added | | 6. | User is redirected to the supplier list page. | |
| Sub Flows: | 3.1 if validation is unsuccessful, the system prompts suer to renter the information. |
| Alternate Flows: |  |

4 - Corresponding Mockups:

A screenshot of a computer

AI-generated content may be incorrect.

**3.2.3.9 Register Users**

1 - Business Rules:

* Only restaurant managers can create, modify, or delete user accounts and assign roles.
* To add new user, manager must provide first name, last name, username, role, phone number, employee id (filled automatically), address, postal code, email address of the new user.
* Users must provide a valid email, and password (min. 8 characters, alphanumeric) to register.

2 - System Use Case Diagrams:

A diagram of a person

AI-generated content may be incorrect.

3 - Use Case Descriptions:

|  |  |
| --- | --- |
| Use Case Name: | Create account |
| ID | UC14 |
| Importance | High |
| Primary Actor | employee |
| Use Case Type | registration |
| Stakeholders and interest | Employee-create account to use application |
| Brief Description: | This use case describes registration of chowHub application |
| Trigger:  Type: | Employee wants to create account to use chowHub  internal |
| Relationships: | |  |  | | --- | --- | | Association | Employee | | Include |  | | Extend |  | | Generalization | Registration | |
| Normal Flow of events: | |  |  | | --- | --- | | 1. | User navigates to the registration page | | 2. | User enters all required information’s | | 3. | System validates provided information | | 4. | If validation successful, the system creates new account | | 5. | System sends a confirmation email to the user’s email | | 6. | User verifies their email by following email instructions | | 7. | System confirms successful account creation | | 8. | User is redirected to the main page and automatically logged in | |
| Sub Flows: |  |
| Alternate Flows: | 4.1 If validation unsuccessful, system prompts user to enter correct information again.  5.1 if user didn’t receive the confirmation email, user can request top resend. |

4 - Corresponding Mockups:

A screenshot of a computer

AI-generated content may be incorrect. 

**3.2.3.10 User Sign in**

1 - Business Rules:

* Users must enter valid credentials (employee id, username, password) to sign in.

2 - System Use Case Diagrams:

A diagram of a person with a sign in

AI-generated content may be incorrect.

3 - Use Case Descriptions:

|  |  |
| --- | --- |
| Use Case Name: | Sign in |
| ID | UC15 |
| Importance | High |
| Primary Actor | employee |
| Use Case Type | Sign in |
| Stakeholders and interest | Employee-wants to sign in to use the application |
| Brief Description: | This use case describes sign in of chowHub application |
| Trigger:  Type: | Employee wants to use chowHub  internal |
| Relationships: | |  |  | | --- | --- | | Association | Employee | | Include |  | | Extend |  | | Generalization | Sign in | |
| Normal Flow of events: | |  |  | | --- | --- | | 1. | User navigates to the sign-in page | | 2. | User enters all required information’s | | 3. | System validates if provided information is in the system database | | 4. | If validation successful, the system logs user in | | 5. | User is redirected to the main page and sign-in | |
| Sub Flows: |  |
| Alternate Flows: | 4.1 If validation unsuccessful, system prompts user to enter correct information again. |

4 - Corresponding Mockups:

A screenshot of a login form

AI-generated content may be incorrect. 

# Domain Class Diagram

A diagram of a computer

AI-generated content may be incorrect.

Figure . Domain Class Diagram for Chowhub System

# Database

The system is built using a NoSQL database, ensuring flexibility, scalability, and efficient data retrieval. This document details the system’s requirements, including data models and data samples.

**Data Model**

**Ingredient** {

id: int

name: String

quantity: double

unit: String

threshold: double

Suppliers: [Suppliers]

}

**Supplier** {

id: int

name: String

phoneNumber: String

email: String

}

**MenuItemIngredient** {

id: int

ingredient: Ingredient

quantity: double

unit: String

}

**MenuItem** {

id: int

ingredients: [menuItemIngredient]

name: String

description: String

isAvailable: Bool

category: Category

price: Double

image: String

}

**Category** {

id: int

name: String

description: String

}

**Customer** {

id: int

name: String

}

**Order** {

id: int

customer: Customer

orderLineItems: [OrderLineItem]

status: Enum(pending, prepared, served)

placedBy: RestaurantStaff

createdAt: Date

}

**OrderLineItem** {

id: int

menuItem: MenuItem

quantity: int

price: double

}

**Inventory** {

id: int

ingredient: Ingredient

currentStock: Double

lastUpdated: DateTime

}

**SaleReport** {

id: int

reportStartDate: Date

reportEndDate: Date

totalRevenue: Double

totalCost: Double

profit: Double

generatedDateTime: DateTime

details: [SaleReportDetail]

}

**SaleReportDetail** {

id: int

menuItem: MenuItem

quantitySold: int

revenueGenerated: Double

}

**Roles** {

"id": "UUID",

"roleName": "String" // e.g., "admin", "chef", "cashier", "customer"

}

**User**  {

"id": int,

"firstName": String,

"lastName": String,

"email": String,

"password": String, // Hashed password

"address": String,

"postalCode": String,

"phone": String,

"emergencyContact": String,

"employeeID": String, // Nullable (if user is not an employee)

"roleID": String // Defines user role

}

**ShiftSchedule** {

"id": int,

"employeeID": String, // Reference to Users Collection

"shiftDate": Date,

"startTime": DateTime, // Format: "HH:mm"

"endTime": DateTime // Format: "HH:mm"

}

**Attendance** {

"id": int, // Unique Attendance ID

"employeeID": String, // Reference to Users Collection

"checkInTime": DateTime, // Timestamp for check-in

"checkOutTime": DateTime, // Timestamp for check-out

"status": String // "Present", "Absent", "Late", "On Leave"

}

**Payroll** {

"id": inr, // Unique Payroll ID

"employeeID": String, // Reference to Users Collection

"payRegularHours": Number,

"bonusPayHours": Number,

"bonus": Number,

"deductions": Number,

"netPay": Number

}

**Example/Data Sample:**

**Ingredient {**

“Id  
: “ingredient\_id\_1”,

“name”: “meat patties”,

“Quantity”: 26.0,

“Unit”:”Whole”,

“Threshold”: 12.0,

Suppliers: [{ “id”:”supplier\_id\_2”, “name”: “Meat Inc”, “PhoneNumber”: “6471234567”, “Email”: [ordering@meatinc.ca](mailto:ordering@meatinc.ca) } ,

{“id”:”supplier\_id\_3”, “name”: “Premium Inc”, “PhoneNumber”: “6461234567”, “Email”: “ordering@premiuminc.ca”}]

},

{“Id  
: “ingredient\_id\_2”,

“name”: “burger buns”,

“Quantity”: 13.0,

“Unit”:”Pairs”,

“Threshold”: 12.0,

Suppliers:[ {“id”:”supplier\_id\_3”, “name”: “Premium Inc”, “PhoneNumber”: “6461234567”, “Email”: “ordering@premiuminc.ca”}]

}

**Supplier** {

“id”:”supplier\_id\_2”,

“name”: “Meat Inc”,

“PhoneNumber”: “6471234567”,

“Email”: “ordering@meatinc.ca”

} {“id”:”supplier\_id\_3”,

“name”: “Premium Inc”,

“PhoneNumber”: “6461234567”,

“Email”: “ordering@premiuminc.ca”

}

**MenuItemIngredient** {

“Id”: “menuItemIngredient\_1”,

Ingredient: {“Id: “ingredient\_id\_1”, “name”: “meat patties”, “Quantity”: 26.0, “Unit”:”Whole”, “Threshold”: 12.0, Suppliers: [{ “id”:”supplier\_id\_2”, “name”: “Meat Inc”, “PhoneNumber”: “6471234567”, “Email”: [ordering@meatinc.ca](mailto:ordering@meatinc.ca) } , {“id”:”supplier\_id\_3”, “name”: “Premium Inc”, “PhoneNumber”: “6461234567”, “Email”: “ordering@premiuminc.ca”}] },

Quantity: 1.0,

Unit:”Whole”

},{

“Id”: “menuItemIngredient\_id\_2”,

Ingredient: {“Id : “ingredient\_id\_2”, “name”: “burger buns”, “Quantity”: 13.0, “Unit”:”Pairs”, “Threshold”: 12.0, Suppliers:[ {“id”:”supplier\_id\_3”, “name”: “Premium Inc”, “PhoneNumber”: “6461234567”, “Email”: “ordering@premiuminc.ca”}] },

Quantity: 1.0,

Unit:”Pair”

}

**MenuItem** {

“Id”: “menu\_item\_id\_1”,

“Name”: “burger” ,

“Description”: “Hamburger bun with hamburger meat”,

“Available”: true,

“Category”: { “Id” : “category\_id\_1”, “Name”: “Main Dish”, “Description”: “The main dish for you to eat” },

“Price”: 17.99 ,

Image: image,

“Ingredients”: [{ “Id”: “menuItemIngredient\_1”, Ingredient: {“Id: “ingredient\_id\_1”, “name”: “meat patties”, “Quantity”: 26.0, “Unit”:”Whole”, “Threshold”: 12.0, Suppliers: [{ “id”:”supplier\_id\_2”, “name”: “Meat Inc”, “PhoneNumber”: “6471234567”, “Email”: [ordering@meatinc.ca](mailto:ordering@meatinc.ca) } , {“id”:”supplier\_id\_3”, “name”: “Premium Inc”, “PhoneNumber”: “6461234567”, “Email”: “ordering@premiuminc.ca”}] }, Quantity: 1.0, Unit:”Whole” },

{“Id”: “menuItemIngredient\_id\_2”, Ingredient: {“Id : “ingredient\_id\_2”, “name”: “burger buns”, “Quantity”: 13.0, “Unit”:”Pairs”, “Threshold”: 12.0, Suppliers:[ {“id”:”supplier\_id\_3”, “name”: “Premium Inc”, “PhoneNumber”: “6461234567”, “Email”: “ordering@premiuminc.ca”}] }, Quantity: 1.0, Unit:”Pair”],

}

**Category** {

“Id” : “category\_id\_1”,

“Name”: “Main Dish”,

“Description”: “The main dish for you to eat”

}

**Order** { "Id": 5001, "Customer": { "Id": 1, "Name": "John Smith" }, "OrderLineItems": [ { "Id": 1001, "MenuItem": { "Id": "menu\_item\_id\_1", "Name": "Classic burger", "Description": "Hamburger bun with hamburger meat", "Available": true, "Category": { "Id": "category\_id\_1", "Name": "Main Dish", "Description": "Our selection of delicious main items" }, "Price": 17.99, "Image": "image", "Ingredients": [ { "Id": "menuItemIngredient\_1", "Ingredient": { "Id": "ingredient\_id\_1", "Name": "meat patties", "Quantity": 26.0, "Unit": "Whole", "Threshold": 12.0, "Suppliers": [ { "Id": "supplier\_id\_2", "Name": "Meat Inc", "PhoneNumber": "6471234567", "Email": "ordering@meatinc.ca" } ] }, "Quantity": 1.0, "Unit": "Whole" }, { "Id": "menuItemIngredient\_2", "Ingredient": { "Id": "ingredient\_id\_2", "Name": "burger buns", "Quantity": 13.0, "Unit": "Pairs", "Threshold": 12.0, "Suppliers": [ { "Id": "supplier\_id\_3", "Name": "Premium Inc", "PhoneNumber": "6461234567", "Email": "ordering@premiuminc.ca" } ] }, "Quantity": 1.0, "Unit": "Pair" } ] }, "Quantity": 2, "Price": 35.98 } ], "Status": "Pending", "PlacedBy": { "UserID": 101, "FirstName": "Alice", "LastName": "Johnson", "Email": "alice@example.com", "Password": "hashedPassword123", "Role": "Staff", "Address": "123 Main St", "PostalCode": "A1B2C3", "Phone": "1234567890" }, "CreatedAt": "2025-03-22T14:00:00Z" }

**Inventory** {

"id": 1,

"ingredient": {

"id": 101,

"name": "Burger Patties",

"quantity": 120.0,

"unit": "Pieces",

"threshold": 50.0,

"suppliers": [

{

"id": 201,

"name": "Fresh Meat Co",

"phoneNumber": "6475550101",

"email": "orders@freshmeatco.ca"

}

]

},

"currentStock": 120.0,

"lastUpdated": "2025-03-22T10:30:00Z"

}

**SaleReport** {

"id": 5001,

"reportStartDate": "2025-03-01",

"reportEndDate": "2025-03-31",

"totalRevenue": 8500.50,

"totalCost": 5400.25,

"profit": 3100.25,

"generatedDateTime": "2025-04-01T00:00:00Z",

"details": [

{

"id": 6001,

"menuItem": {

"id": 301,

"name": "Classic Cheeseburger",

"description": "Cheeseburger with cheddar, lettuce, tomato",

"available": true,

"category": {

"id": 401,

"name": "Main Dish",

"description": "Delicious main courses"

},

"price": 14.99,

"image": "burger.png"

},

"quantitySold": 150,

"revenueGenerated": 2248.50

}

]

}

**User** {

"id": "132",

"firstName": "John",

"lastName": "Doe",

"email": "johndoe@example.com",

"password": "hashed\_password",

"address": "123 Main St, Toronto, ON",

"postalCode": "M5H 2N2",

"phone": "+1 416-123-4567",

"emergencyContact": "+1 416-987-6543",

"employeeID": "132",

"roleID": "2"

}

**ShiftSchedule** {

"id": "123",

"employeeID": "132",

"shiftDate": "2025-03-25",

"startTime": "09:00",

"endTime": "17:00"

}

**Attendance** {

"id": "111",

"employeeID": "123",

"checkInTime": "2025-03-25T09:00:00Z",

"checkOutTime": "2025-03-25T17:00:00Z",

"status": "Present"

}

**Payroll** {

"id": " 501",

"employeeID": "132",

"payRegularHours": 40,

"bonusPayHours": 5,

"bonus": 200.00,

"deductions": 50.00,

"netPay": 1450.00

}

# Work Breakdown Structure (WBS)

## 

## Work Breakdown Structure

Sample WBS:

Diagram

Description automatically generated

# Milestones and Acceptance Criteria

* 1. Milestone one

Definition

Acceptance Criteria

* …
* ….
* ….
  1. Milestone Two
  2. Milestone Three
  3. ..
  4. …
  5. …
  6. ..
  7. ..
  8. ...etc.

# Implementation Schedule

Implementation Schedule using MS Project (Waterfall)

OR

Product Backlog (Agile-Scrum)

# Client / Faculty Sign-off

**Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

X .

Name of Client/Rep/Professor