

Digital Queue Management for Canteen

UCS 503 Software Engineering Project Report Mid-Semester Evaluation

Submitted by:

Saksham Panwar 102303019

Subhendu Sharma 102303013

Manas Goyal 102303021

Mohammad Aaban 102303015

BE Third Year, COE

Submitted To:

Dr. Ashima Singh



THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

**Computer Science and Engineering Department
TIET, Patiala**

INDEX

S.No.	Assignment	Page No.
1.	Project Selection Phase	
1.1	Software Bid	3
1.2	Project Overview	4
2.	Analysis Phase	
2.1	Use Cases	6
2.1.1	Use-Case Diagrams	6
2.1.2	Use Case Templates	6
2.2	Activity Diagram and Swimlane Diagrams	7
2.3	Data Flow Diagrams (DFDs)	
2.3.1	DFD Level 0	8
2.3.2	DFD Level 1	9
2.3.3	DFD Level 2	9
2.4	Software Requirement Specification (IEEE Format)	10
2.5	User Stories and Story Cards	12

Software Bid

Project Title: Digital Queue Management System for Canteen

Objective:

To develop a web-based and mobile-accessible system that allows students to join and track virtual queues for canteen orders, thereby minimizing physical waiting time and improving service efficiency.

Scope of Work:

The system will include modules for:

- Online queue registration for canteen orders.
- Real-time queue status updates and notifications.
- Order and token management for canteen staff.
- Payment integration and ticket generation for students.
- Administrative dashboard for managing queues, users, and analytics.

Proposed Solution:

The solution will use a centralized database and a responsive web interface accessible to both students and canteen staff. Students can browse menu items, place orders online, and join a digital queue. The system will generate unique queue numbers and provide real-time updates to users as their order status progresses, ensuring an efficient and organized canteen experience.

Project Overview

Introduction

This document provides a comprehensive overview of our software product, outlining its parameters, goals, and the perspectives of our client, team, and target audience. It further details the project's intended user base, user interface specifications, and essential hardware and software requirements.

Scope of Development Project

Campus meals are meant to bring people together—not turn into a daily struggle. Yet, we've all seen the chaos: long queues, confused crowds, and everyone shouting to get their order first. Most existing mess apps only handle the basics like browsing the menu and making payments, but they don't solve the real problem—the disorder that happens when everyone shows up at once.

We believe there's a smarter way. Our goal is to build **MessMate**, a clean and reliable mess management app designed to bring order, efficiency, and peace of mind to the dining experience. It's not just another food ordering app—it's a complete system that streamlines everything from placing an order to picking it up.

Once a student places an order through the app, they'll automatically receive a **digital queue ticket** with a unique number. The kitchen staff can then update order statuses in real time, and students get notified the moment their food is ready. No more waiting in crowded lines or shouting across counters—just calm, organized service.

Here's what makes MessMate special:

1. Your Mess ID

Forget waiting in long lines or shouting your name at the counter. With our app, you'll create one simple digital profile—your **Mess ID**—containing your name, roll number, and preferences. From then on, placing an order is as easy as a single tap. It's your personalized entry to a smoother dining experience.

2. Order Made Simple

Right after you log in, you'll find a clean, easy-to-use interface to browse the day's menu and place your order instantly. Whether it's breakfast, lunch, or a late-night snack, everything you need is right at your fingertips. No confusion, no paperwork—just quick and effortless ordering.

3. Smart Queue System

Once you place an order, the app automatically generates a **digital queue ticket** with your

unique number. No more crowding or chaos. The kitchen staff can manage the live queue efficiently, and you'll receive **real-time updates** about your position and estimated waiting time.

4. Instant Notification

As soon as your food is ready, you'll get a notification on your phone. The moment your number comes up, simply walk to the counter, show your ticket, and collect your order. It's fast, fair, and frustration-free—everyone gets served in order, without the hassle.

5. Effortless Management

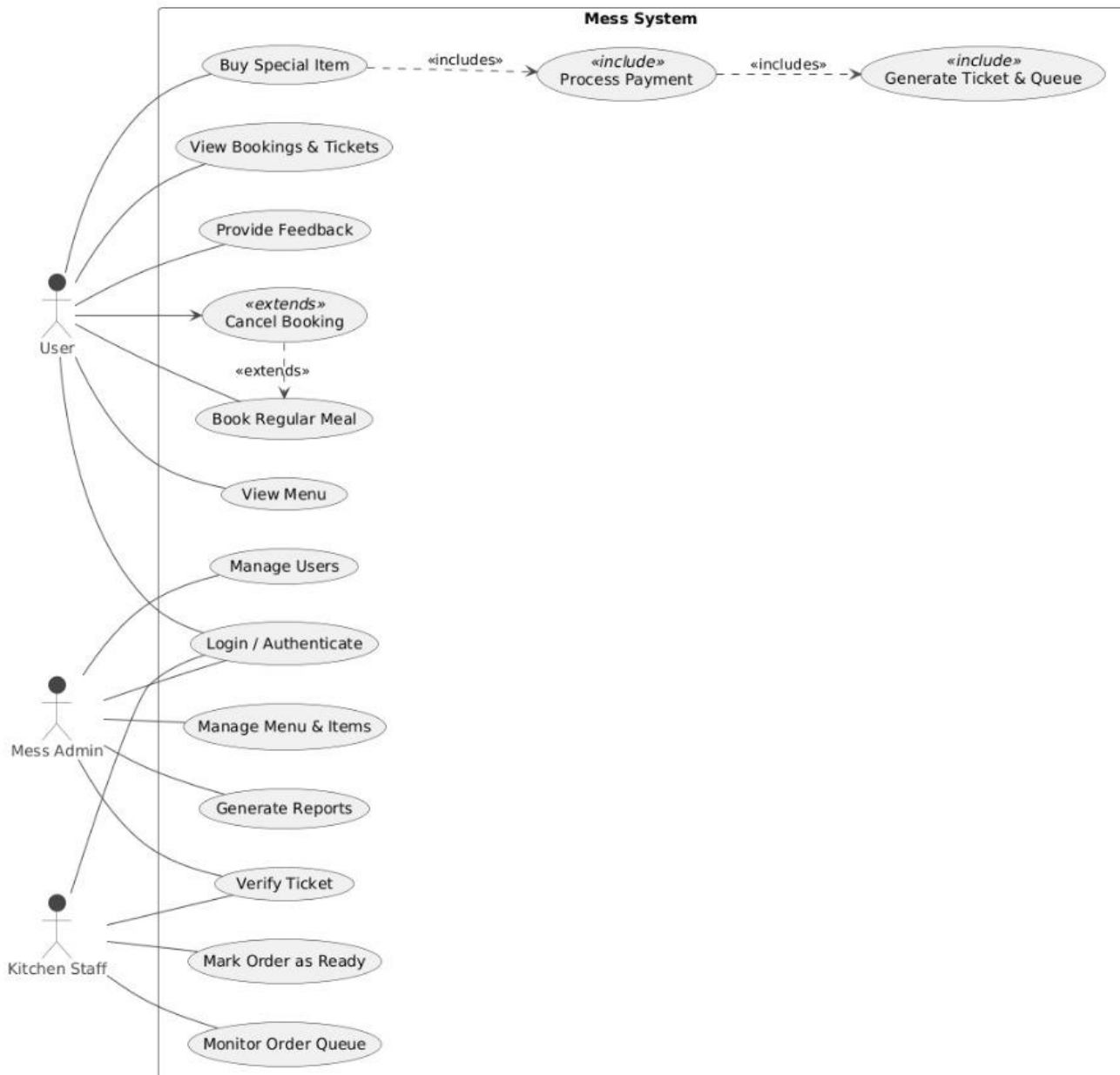
For mess staff, the app acts as a **command center**—track incoming orders, update statuses, and manage the queue all from one place. It simplifies operations, reduces confusion, and ensures every student gets their meal quickly and accurately.

6. Reliable and Secure

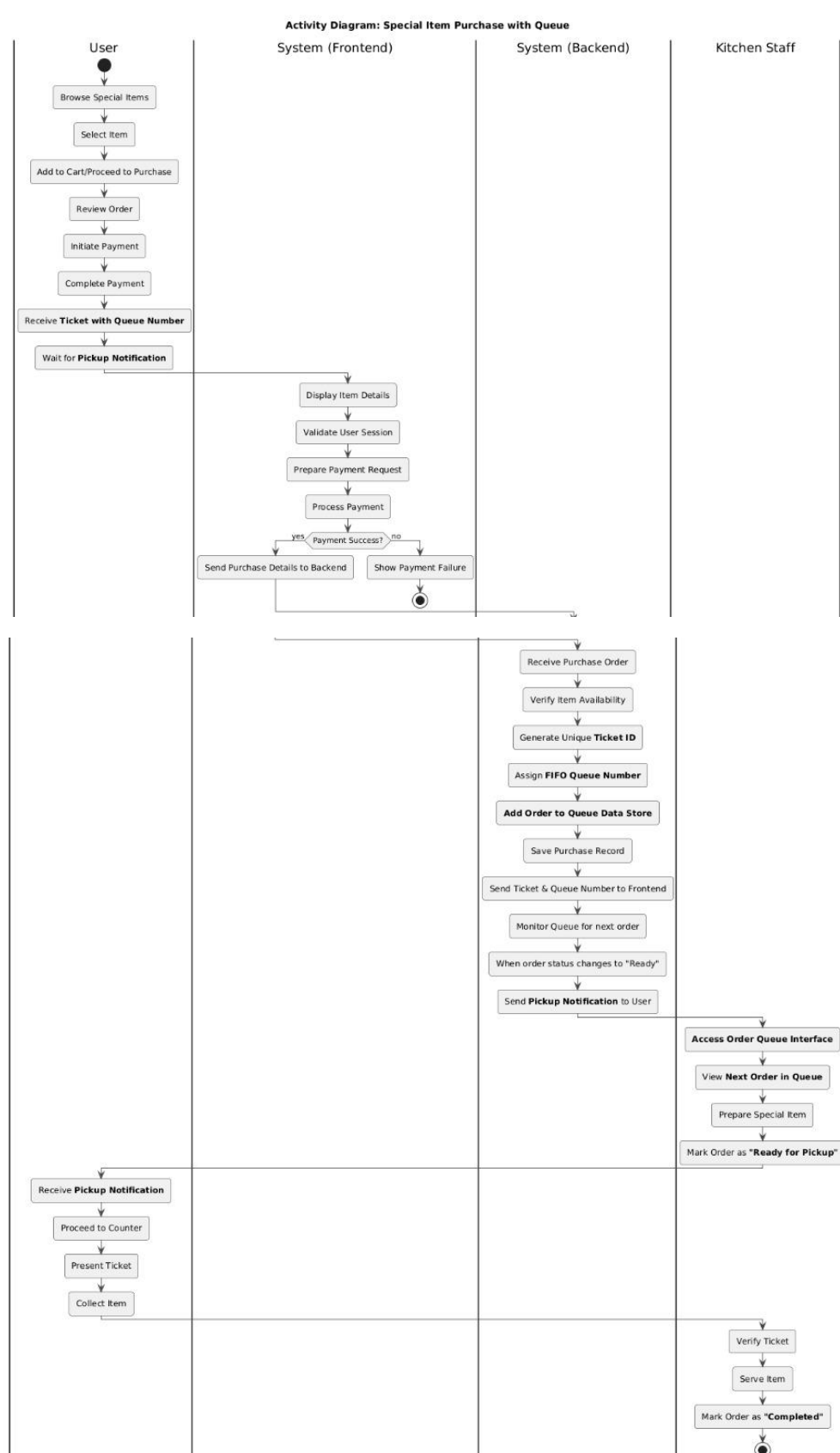
We're building the system on a **top-tier cloud platform** to ensure that every order, ticket, and update is stored safely with automatic backups. No lost orders, no missed updates—just smooth, secure service every day.

Use Case Diagram

Enhanced Mess Management System - Use Case

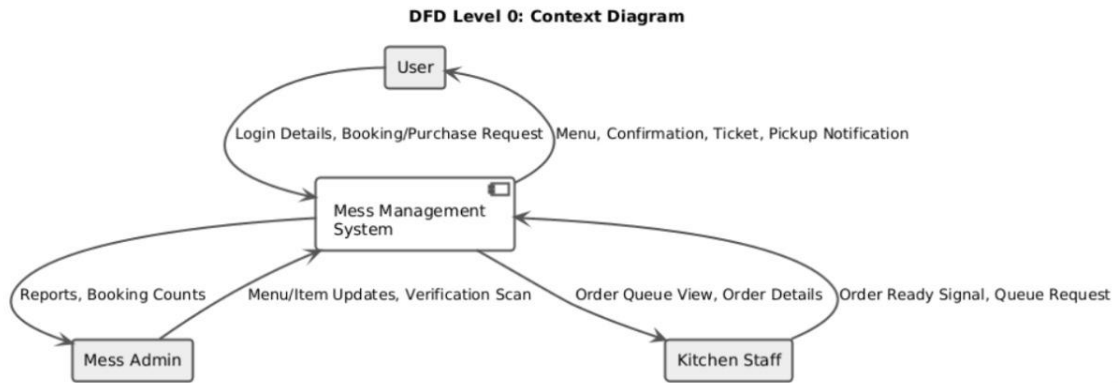


Activity & Swimlane Diagram

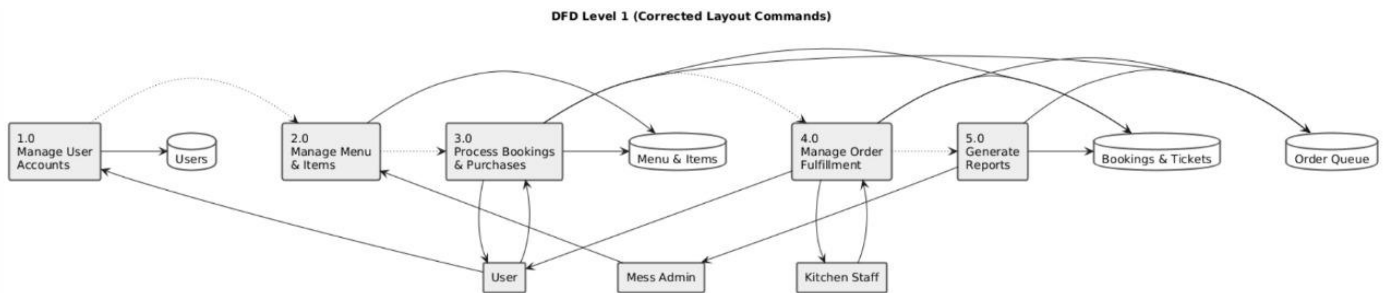


DFD Diagram

DFD Level 0:

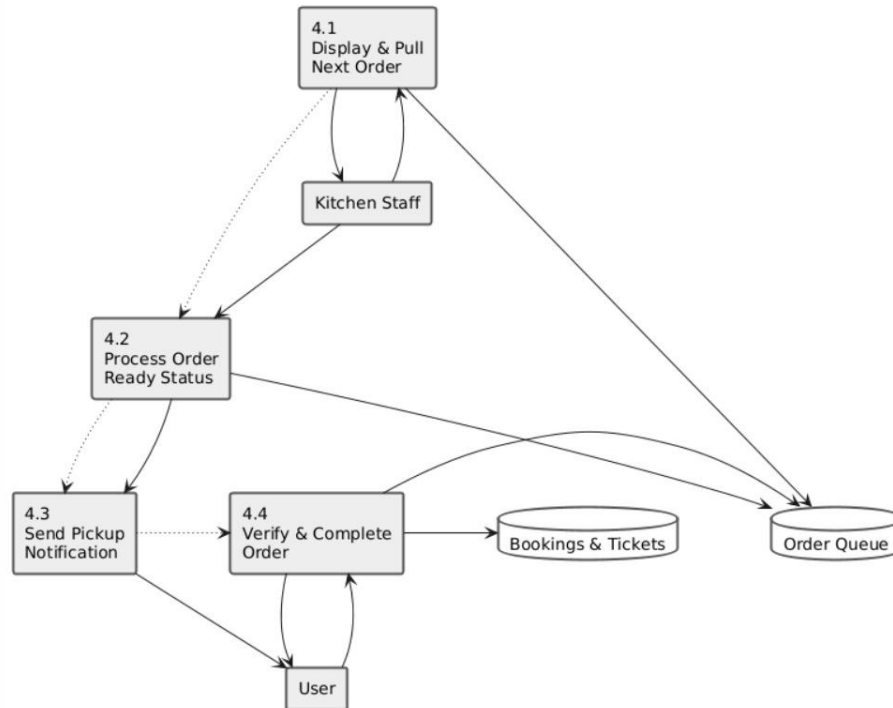


DFD Level 1:



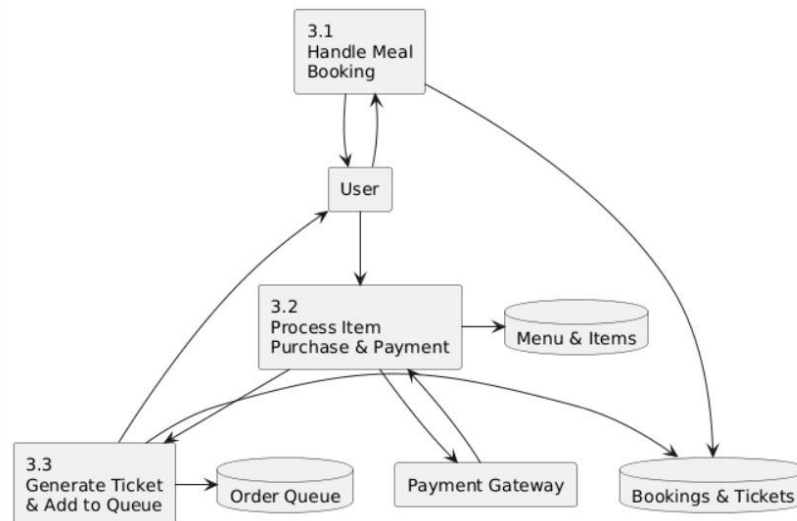
DFD Level 2 (1):

DFD Level 2: Exploding "4.0 Manage Order Fulfillment"



DFD Level 2 (2):

DFD Level 2: Exploding "3.0 Process Bookings & Purchases" (Barebones)



Software requirement Specification

(IEEE Format)

The Software Requirements Specification (SRS) document provides a comprehensive overview of the requirements for the **MessMate – Enhanced Mess Management System**. It serves as a key reference for all project stakeholders — including developers, testers, administrators, and end-users — to ensure a shared understanding of the system's:

- **Functionalities:** What the system is designed to accomplish in managing daily mess operations.
- **Performance expectations:** How efficiently and reliably it should operate to minimize waiting times and improve service flow.
- **Design and operational constraints:** Any system limitations, dependencies, or integration requirements.

This document adheres to the **IEEE Std 830-1998** standard for SRS and guides the development, testing, and validation of the MessMate system throughout its lifecycle.

The **MessMate** platform is a **web and mobile-based digital queue and meal management solution** that simplifies the process of meal booking, order tracking, and queue management in campus mess environments. The system aims to enhance efficiency and user satisfaction by providing real-time updates, digital token generation, and role-based management for users, mess administrators, and kitchen staff.

It supports the following major functions:

- Online meal booking and cancellation
- Special item purchase with secure digital payment
- Automated ticket and queue generation
- Real-time order tracking and status notifications
- Menu and user management by mess administrators

- Queue monitoring and order verification by kitchen staff

This SRS defines:

- All **functional and non-functional requirements** for the initial release of MessMate.
- Key **terms and acronyms** (e.g., Queue ID, Token, Admin Panel, API, UI) used in the document.
- Reference **technical resources and APIs** (e.g., payment gateways, notification APIs, and database integration tools) that support the implementation and operation of the system.

User Stories & User Cards

User Story 1: Browse and Purchase Special Items

As a student/user

I want to browse special menu items and purchase them online

So that I can skip the physical queue and receive a ticket with my queue number

Acceptance Criteria:

1. User can view all available special items with details.
2. User can select items and add them to the cart.
3. User can complete payment through the system.
4. User receives a unique ticket ID and FIFO queue number after successful payment.
5. User gets a pickup notification when the order is ready.

FRONT

User Story Example: Front of Card

#0001 **BROWSE AND PURCHASE SPECIAL ITEMS**

As a [student/user], I want to [browse special menu items and purchase them online], so I can [skip the physical queue and receive a ticket with my queue number].

For new features, annotated wireframe. For bugs, steps to reproduce with screenshot. For non-functional stories, region scope/standards.

Special Menu

Paneer Tikka
Grilled cottage cheese with spices
₹120

Add to Cart

Veg Biryani
Aromatic rice with vegetables
₹80

Add to Cart

Display confirmation after adding to cart. Show total and checkout button.

- User can view all available special items with details
- User can select items and add to cart
- User can complete payment through the system
- User receives unique ticket ID and queue number
- User gets pickup notification when ready

Further information is attached to this story on VSTSS Product Backlog

BACK

User Story Example: Back of Card

#0001 **BROWSE AND PURCHASE SPECIAL ITEMS**

Confirmation Scenarios

- Payment successful: Display ticket ID (e.g., TKT-1234) and queue number
- Payment failed: Show error message and retry option
- Item unavailable: Remove from cart and notify user
- Order ready: Send push notification with pickup instructions

Technical Notes

- Integrate payment gateway (Razorpay/PayU)
- Implement FIFO queue system for order processing
- Generate unique ticket IDs using timestamp + random string
- Enable push notifications for order status updates
- Store order history in user profile

Test Cases

TC-1: Successful Purchase
Browse menu → Add items → Complete payment → Verify ticket received

TC-2: Empty Cart Checkout
Attempt checkout with empty cart → Verify error message displayed

TC-3: Pickup Notification
Order marked ready → Verify user receives notification

User Story 2: Manage Menu Items

As a mess admin

I want to add, edit, or remove menu items

So that I can keep the menu updated with available offerings

Acceptance Criteria:

1. Admin can create new menu items with details.
2. Admin can edit existing items (price, description, availability).
3. Admin can mark items as unavailable or remove them.
4. Changes reflect immediately on the user-facing menu.

FRONT

User Story Example: Front of Card

#0002 **MANAGE MENU ITEMS**

As a [mess admin], I want to [add, edit, or remove menu items], so I can [keep the menu updated with available offerings].

For new features, annotated wireframe. For bugs, steps to reproduce with screenshot. For non-functional stories, region scope/standards.

Add/Edit Menu Item

Item Name:

Description:

Price (₹):

Category:

☐ **Available**

Save

Cancel

Validate required fields. Show success/error message after save.

- Admin can create new menu items with details
- Admin can edit existing items (price, description, availability)
- Admin can mark items as unavailable or remove them
- Changes reflect immediately on user-facing menu

BACK

User Story Example: Back of Card

#0002 **MANAGE MENU ITEMS**

Confirmation Scenarios

- Item added successfully: Display success message and redirect to menu list
- Item updated: Show confirmation and reflect changes in user menu
- Item deletion: Prompt confirmation before removing item
- Validation errors: Highlight fields and show error messages
- Unavailable item: Grey out in user menu, show "Currently Unavailable"

Technical Notes

- Implement role-based access control (admin only)
- Real-time menu updates using WebSocket or polling
- Image upload support for menu items (optional)
- Maintain audit log of all menu changes
- Validate price as positive number, required fields not empty

Test Cases

TC-1: Add New Item

Login as admin → Add item with details → Verify saved and visible to users

TC-2: Edit Existing Item

Select item → Modify price → Save → Verify updated price on menu

TC-3: Mark Unavailable

Toggle availability → Verify item shows as unavailable to users

TC-4: Validation

User Story 3: Monitor and Process Order Queue

As a kitchen staff member

I want to view the order queue and mark orders as ready

So that I can prepare orders in sequence and notify users for pickup

Acceptance Criteria:

1. Staff can see all pending orders in queue order.
2. Orders display ticket ID, items, and queue number.
3. Staff can mark orders as ready for pickup.
4. System sends a pickup notification to the user automatically.

FRONT

User Story Example: Front of Card

#0003 **MONITOR AND PROCESS ORDER QUEUE**

As a [kitchen staff member], I want to [view the order queue and mark orders as ready], so I can [prepare orders in sequence and notify users for pickup].

For new features, annotated wireframe. For bugs, steps to reproduce with screenshot. For non-functional stories, region scope/standards.

Order Queue Dashboard

TKT-1234

#1 2x Paneer Tikka, 1x Veg
Biryani
Ordered: 10:30 AM

Mark Ready

TKT-1235

#2 1x Masala Dosa, 1x Filter
Coffee
Ordered: 10:35 AM

Mark Ready

Auto-refresh queue. Show confirmation dialog before marking ready. Send notification automatically.

Further information is attached to this story on VSTSS Product Backlog

- Staff can see all pending orders in queue order
- Orders display ticket ID, items, and queue number
- Staff can mark orders as ready for pickup
- System sends pickup notification to user automatically

BACK

User Story Example: Back of Card

#0003 **MONITOR AND PROCESS ORDER QUEUE**

Confirmation Scenarios

- Order marked ready: Display success message and remove from pending queue
- Notification sent: Show confirmation that user was notified
- Notification failed: Log error and show retry option to staff
- Empty queue: Display "No pending orders" message
- Order picked up: Move to completed orders list with timestamp

Technical Notes

- Implement real-time queue updates (WebSocket or polling every 10s)
- Role-based access for kitchen staff dashboard
- Push notification integration for order ready alerts
- Track order states: Pending → Preparing → Ready → Completed
- Display estimated preparation time for each order
- Maintain order history with timestamps for analytics

Test Cases

TC-1: View Queue

Login as kitchen staff → Verify all pending orders visible in FIFO order

TC-2: Mark Order Ready

Click Mark Ready → Confirm action → Verify notification sent to user

TC-3: Real-time Updates

New order placed → Verify appears in queue within 10 seconds

TC-4: Empty Queue