Software Requirements Specification

for

NITC Hostel Mess Management System

Version 1.0

Prepared by

Team Number: 19

Nishant Tanaji Bhandigare	M250923CS
Sikta Roy	M251045CS
Vengidesan K	M251103CS

Course: CS6103E - Software Systems lab

Date: 22 September 2025

1 Introduction

The **NITC Hostel Mess Management System** is a comprehensive digital platform designed to streamline the operations of the hostel mess at the National Institute of Technology Calicut (NITC). This system digitizes the management processes, including menu posting, student attendance tracking, meal feedback collection, inventory management, and automated billing. The goal is to enhance the efficiency, transparency, and overall quality of mess services provided to students. This introductory section offers an overview of the system's purpose, scope, and the kind of information presented in the Software Requirements Specification (SRS) document. Readers will find detailed descriptions of system functionalities, external interfaces, user roles, performance expectations, and quality attributes, intended to guide the development and deployment of the platform.

1.1 Document Purpose

This document specifies the software requirements for the **NITC Hostel Mess Management System**, version 1.0. It provides a detailed description of the functionalities, features, constraints, and interfaces necessary to design, implement, and deploy the system. The SRS serves as a formal agreement between stakeholders, including developers, mess managers, hostel administrators, and students, ensuring a shared understanding of the system's capabilities and requirements. The scope of this document covers the entire mess management system as a unified digital platform. It addresses core modules such as menu management, attendance tracking, feedback collection, inventory control, and automated billing. This SRS does not cover hardware installations or external payment gateways but allows for future integration with such systems.

1.2 Product Scope

The **NITC Hostel Mess Management System** is a web-based digital platform designed to modernize and automate mess operations at the National Institute of Technology Calicut. The software facilitates seamless interaction between three primary user groups: students, mess managers, and hostel administrators. Students can view weekly menus, provide meal feedback and ratings, and register non-attendance for vacations to pause billing. Mess managers can post weekly menus, analyse consolidated feedback to improve meal quality, and manage inventory efficiently. Hostel administrators can oversee the entire system, generate automated monthly bills based on student attendance, and track payment statuses. The primary benefits of this system include significant reduction in manual paperwork, enhanced transparency in mess operations, improved accountability through digital records, and better service quality through systematic feedback collection. The system objectives are to eliminate billing errors through automated calculations, provide real-time insights into student preferences and attendance patterns, streamline inventory management to reduce waste, and create a centralized platform for all mess-related activities. These goals collectively aim to improve the overall dining experience for students while reducing administrative burden on mess staff and ensuring efficient resource utilization across hostel operations.

1.3 Intended Audience and Document Overview

The NITC Hostel Mess Management System is intended for three primary user groups: students, mess managers, and hostel administrators. Students will use the platform to view weekly menus, mark their attendance or register vacations, provide feedback on meals, and access their billing details. Mess managers will rely on the system to post and update menus, manage inventory records, and review student feedback to maintain quality of service. Hostel administrators will oversee the overall operations, generate and manage monthly bills, track payments, and ensure smooth coordination between students and mess staff. This document outlines the requirements of the system with the aim of addressing the needs of these audiences and ensuring that each group can effectively perform their roles through the application.

The remainder of this SRS is organized into several key sections: Section 2 provides an overall system description including product overview, functionality, constraints, and dependencies. Section 3 details specific requirements covering external interfaces, functional requirements, and comprehensive use case models with UML diagrams. Section 4 addresses non-functional requirements including performance, security, and quality attributes essential for system reliability. Section 5 covers any additional requirements not addressed elsewhere, while the appendices contain activity logs documenting team contributions and project timeline. This structure ensures comprehensive coverage of all aspects necessary for successful system development and deployment.

1.4 Definitions, Acronyms and Abbreviations

API - Application Programming Interface; a set of protocols and tools for building software applications
CRUD - Create, Read, Update, Delete; basic database operations
CSS - Cascading Style Sheets; a stylesheet language used for describing the presentation of web documents
HTML - HyperText Markup Language; the standard markup language for creating web pages
HTTP - HyperText Transfer Protocol; the foundation of data communication for the World Wide Web
HTTPS - HyperText Transfer Protocol Secure; an extension of HTTP with encryption for secure communication
JWT - JSON Web Token; a compact, URL-safe means of representing claims between two parties
MERN - MongoDB, Express.js, React.js, Node.js; a full-stack development framework
NITC - National Institute of Technology Calicut; the educational institution for which this system is being developed
REST - Representational State Transfer; an architectural style for designing networked applications
SRS - Software Requirements Specification; a document that describes the intended purpose and environment for software under development
UI - User Interface; the space where interactions between humans and machines occur
UML - Unified Modeling Language; a standardized modeling language for software engineering
URL - Uniform Resource Locator; the address of a web resource

1.5 Document Conventions

This Software Requirements Specification follows IEEE formatting standards and specific conventions to ensure consistency and readability throughout the document. The document uses Arial font, size 12, for all body text to maintain professional appearance and readability. Single spacing is maintained throughout with 1-inch margins on all sides as specified in the template. Section and subsection titles follow the hierarchical numbering system provided in the template, with appropriate formatting for easy navigation.

Formatting Conventions:

Bold text is reserved for section headings and important terminology. Bulleted lists are used for enumeration of features and requirements, while numbered lists indicate sequential processes or priorities.

Naming Conventions:

Functional requirements are identified with the prefix "F" followed by a sequential number (e.g., F1, F2). Use cases are designated with "U" followed by a number (e.g., U1, U2). System components and modules use PascalCase naming (e.g., MessManager, StudentDashboard).

1.6 References and Acknowledgments

References:

1. **IEEE Software Requirements Specification Standard**: IEEE Std 830-1998, "IEEE Recommended Practice for Software Requirements Specifications."
2. **PlantUML Documentation**: This guide explains how to use PlantUML to create UML diagrams from plain text descriptions.
3. **Figma UI/UX Design Guidelines**: This document provides best practices for using Figma in creating user interfaces and prototypes

Acknowledgments:

This template is based on the SRS template available from the GMU site by Dr. Rob Pettit, with modifications specific to NITC for academic purposes. Special acknowledgment to course instructors and project stakeholders who provided guidance and feedback during the requirements gathering phase.

2 Overall Description

2.1 Product Overview

The NITC Hostel Mess Management System is a completely new, self-contained digital platform designed to replace the existing manual mess management processes at the National Institute of Technology Calicut. Currently, hostel mess operations rely heavily on paper-based systems for menu planning, attendance tracking, feedback collection, and billing calculations. This system addresses the inefficiencies, errors, and lack of transparency inherent in manual processes by providing a centralized, web-based solution that automates key operations while maintaining clear accountability and audit trails.

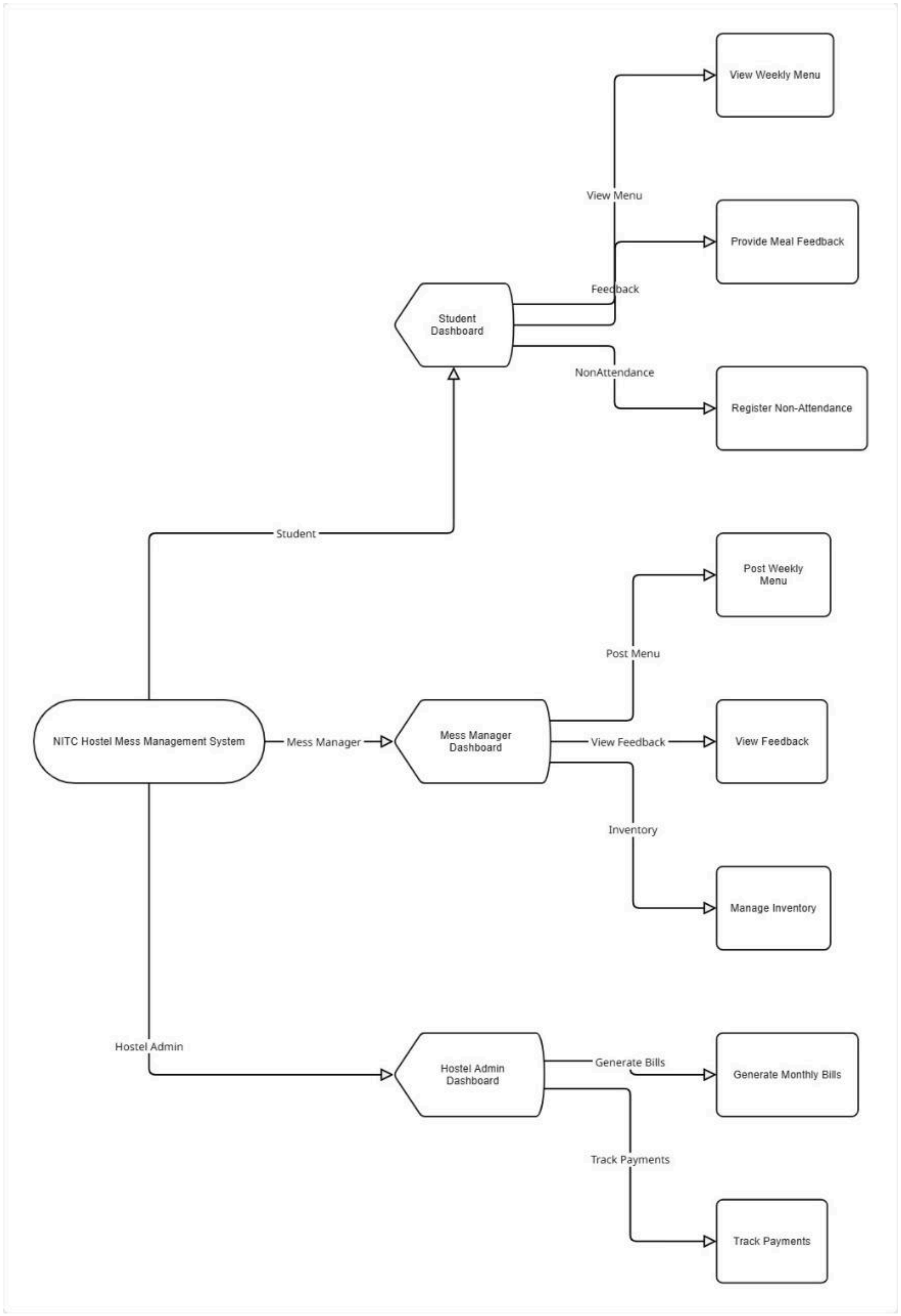


Figure 1 : Workflow Diagram

The product operates within the broader ecosystem of NITC's hostel infrastructure, serving as the primary interface between students and mess administration. While the system is standalone, it is designed with future integration capabilities for institutional databases, payment gateways, and biometric attendance systems. The platform creates a digital bridge connecting three distinct user communities: students who consume mess services, mess managers who operate daily services, and hostel administrators who oversee policy and billing. The system maintains data integrity through role-based access controls while ensuring seamless information flow between these stakeholder groups.

2.2 Product Functionality

Student Functions:

- View weekly mess menu and meal schedules
- Submit daily meal ratings and feedback
- Register non-attendance for vacations or leave periods
- View personal mess bills and payment history
- Track personal attendance records

Mess Manager Functions:

- Post and update weekly mess menus
- View consolidated feedback analytics and ratings
- Manage mess inventory and stock levels
- Monitor student attendance patterns
- Generate feedback reports for quality improvement

Hostel Administrator Functions:

- Oversee complete system operations and user management
- Generate automated monthly mess bills for all students
- Track payment status and manage billing records
- Access comprehensive system reports and analytics
- Manage user roles and system permissions

System-Wide Functions:

- User authentication and role-based access control
- Notification system for important updates

2.3 Design and Implementation Constraints

Technology Stack Requirements:

The system must be developed using the MERN stack architecture, specifically MongoDB for database operations, Express.js for backend API development, React.js for frontend user interfaces, and Node.js as the runtime environment. This constraint ensures consistency with modern web development practices and provides a unified JavaScript-based development environment.

Security and Compliance Constraints:

All data transmission must be encrypted using HTTPS protocols, and user authentication must be implemented using JSON Web Tokens (JWT) with role-based access control.

Platform and Deployment Constraints:

The application must be web-based and responsive, supporting modern browsers.

Performance and Resource Constraints:

The system must support up to 20 concurrent users during peak hours, with response times not exceeding 3 seconds for any operation. Database queries must be optimized for efficiency.

Integration and Interface Constraints:

While the system is standalone, it must be designed with RESTful API architecture. All API endpoints must follow REST conventions and provide proper documentation for future maintenance and integration efforts.

2.4 Assumptions and Dependencies

User and Infrastructure Assumptions:

- All students, mess managers, and administrators have regular access to internet connectivity and computing devices (smartphones, tablets, or computers) capable of running modern web browsers
- Users possess basic computer literacy and can navigate web-based applications without extensive training
- The hostel mess operates on a standardized weekly schedule that remains relatively consistent throughout academic terms

Third-Party Service Dependencies:

- NPM package registry will continue to provide access to required MERN stack libraries and dependencies including React, Express, Mongoose, and JWT authentication packages

Operational Environment Assumptions:

- The current manual mess billing system can be accurately digitized without loss of institutional requirements or compliance issues
- Staff training will be provided to mess managers and administrators to ensure smooth transition from manual to digital processes

External Integration Dependencies:

- Future integration with institutional payment systems or student databases will follow standard API protocols

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The NITC Hostel Mess Management System will provide three distinct user interfaces tailored to the specific roles and requirements of each user group. All interfaces will be responsive web-based applications built using React.js with Material-UI components to ensure consistency and modern design standards.

GUI of some of the pages made using Figma is given below:

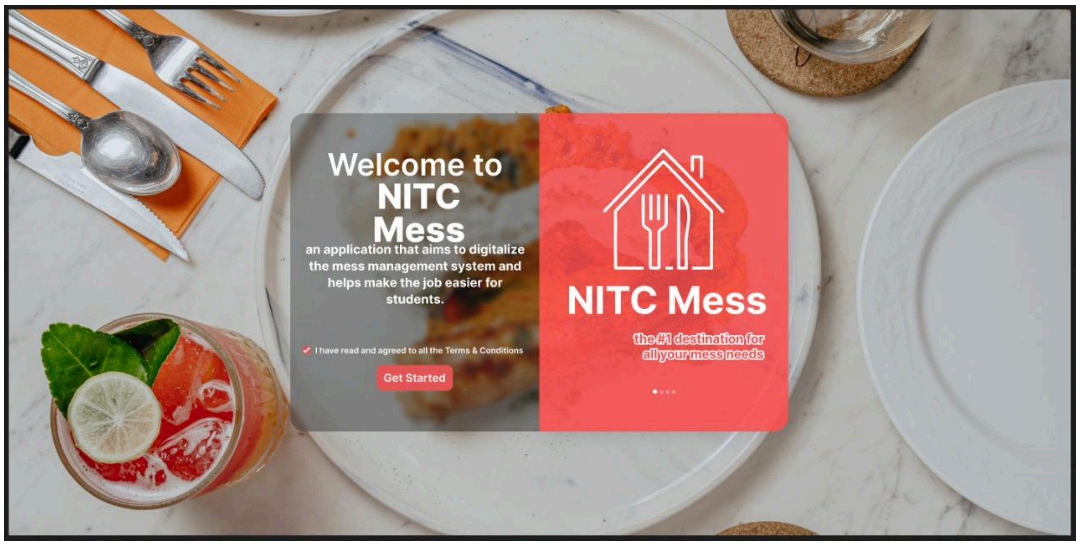

Welcome to NITC Mess
an application that aims to digitalize the mess management system and helps make the job easier for students.
I have read and agreed to all the Terms & Conditions
Get Started
NITC Mess
the #1 destination for all your mess needs

Figure 2 : The starting page

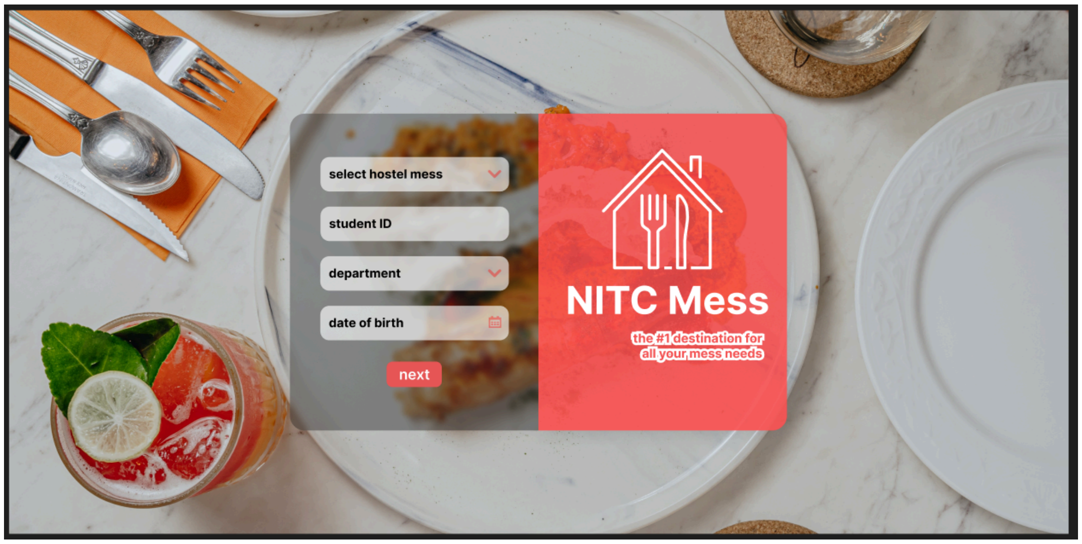


Figure 3 : The register page

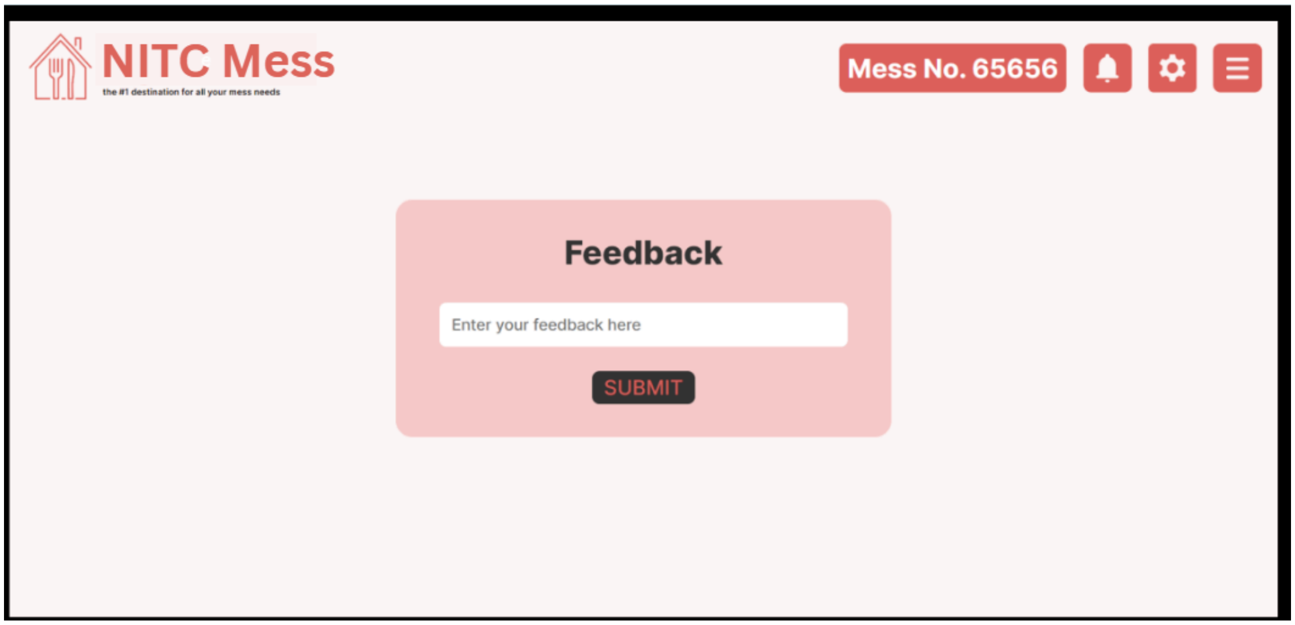


Figure 4 : The Feedback page

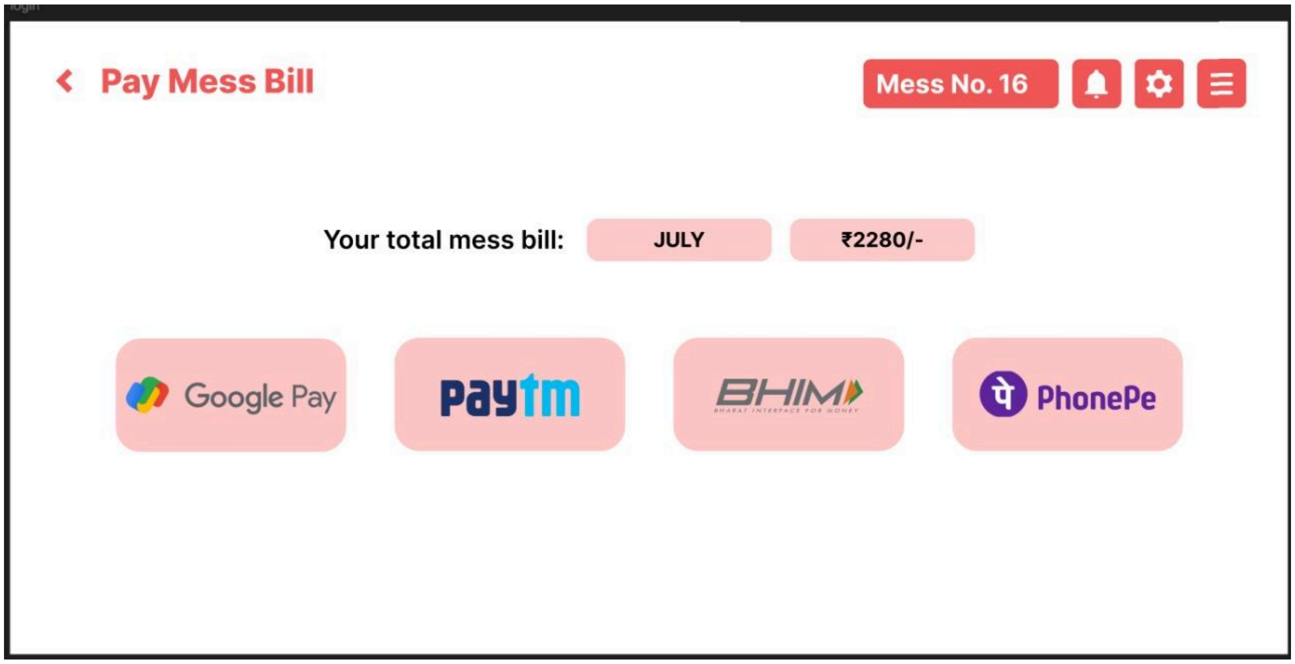
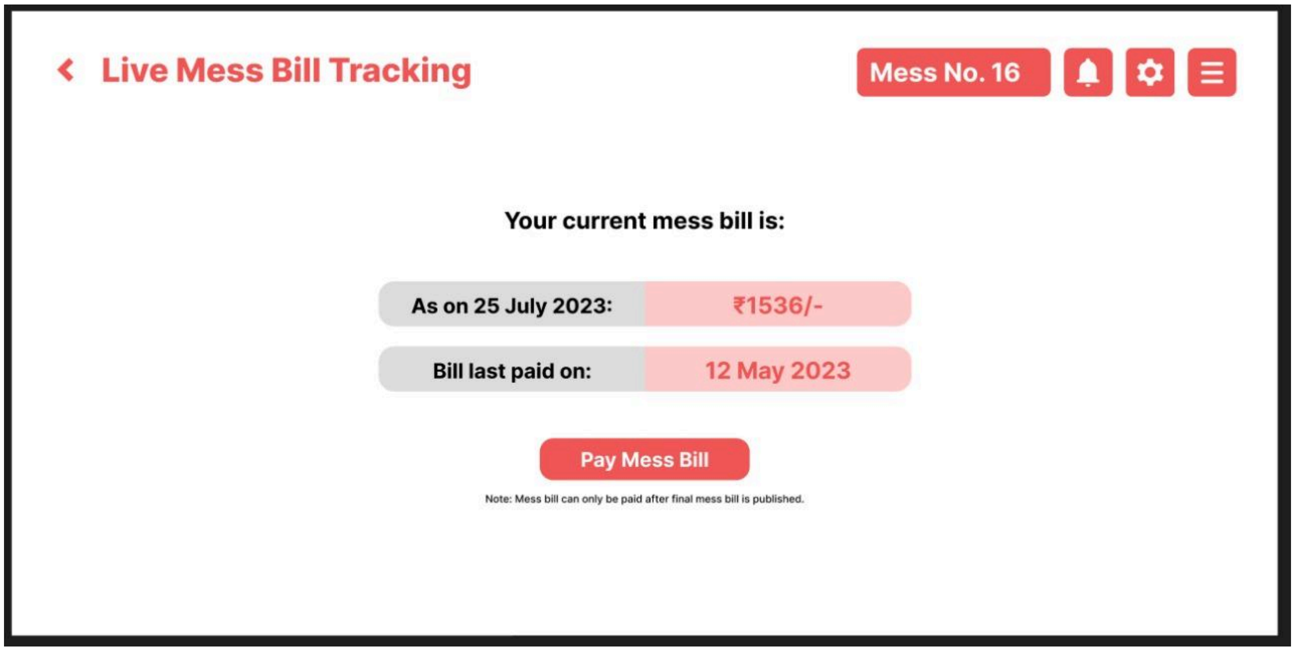


Figure 5 : Bill payment page

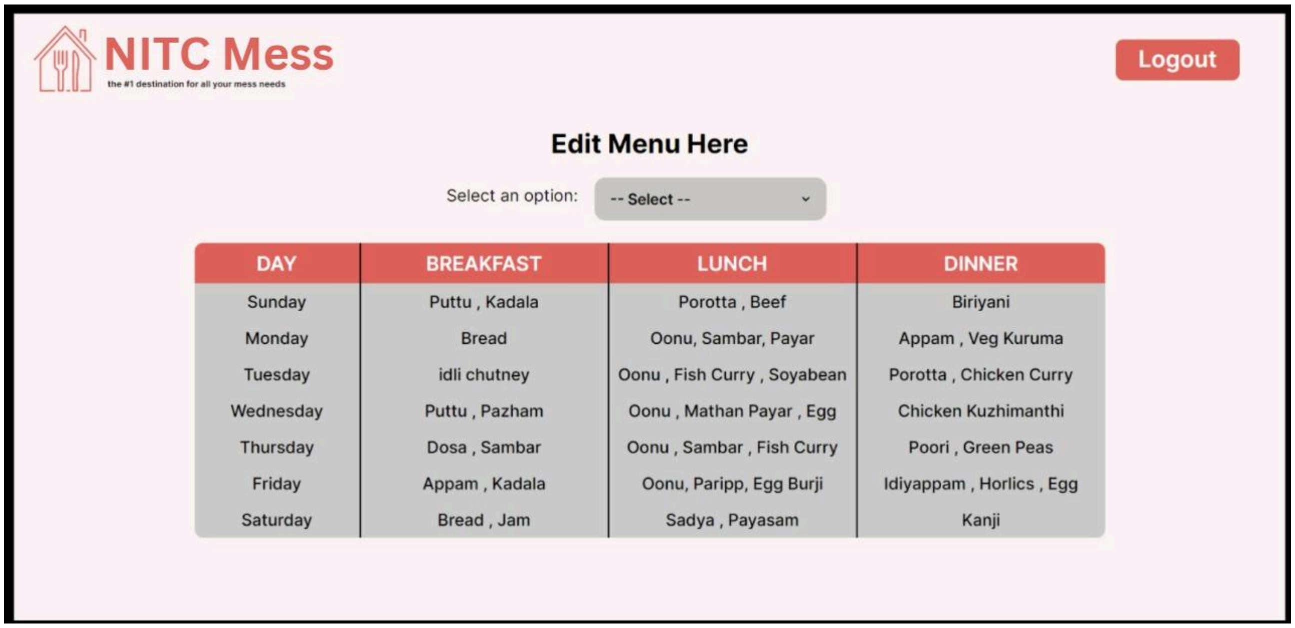


Your current mess bill is:

As on 25 July 2023:	₹1536/-
Bill last paid on:	12 May 2023

Note: Mess bill can only be paid after final mess bill is published.

Figure 6 : Bill Tracking Page



Edit Menu Here

Select an option: -- Select --

DAY	BREAKFAST	LUNCH	DINNER
Sunday	Puttu , Kadala	Porotta , Beef	Biriyani
Monday	Bread	Oonu, Sambar, Payar	Appam , Veg Kuruma
Tuesday	idli chutney	Oonu , Fish Curry , Soyabean	Porotta , Chicken Curry
Wednesday	Puttu , Pazham	Oonu , Mathan Payar , Egg	Chicken Kuzhimanthi
Thursday	Dosa , Sambar	Oonu , Sambar , Fish Curry	Poori , Green Peas
Friday	Appam , Kadala	Oonu, Paripp, Egg Burji	Idiyappam , Horlics , Egg
Saturday	Bread , Jam	Sadya , Payasam	Kanji

Figure 7 : Menu Editing Page

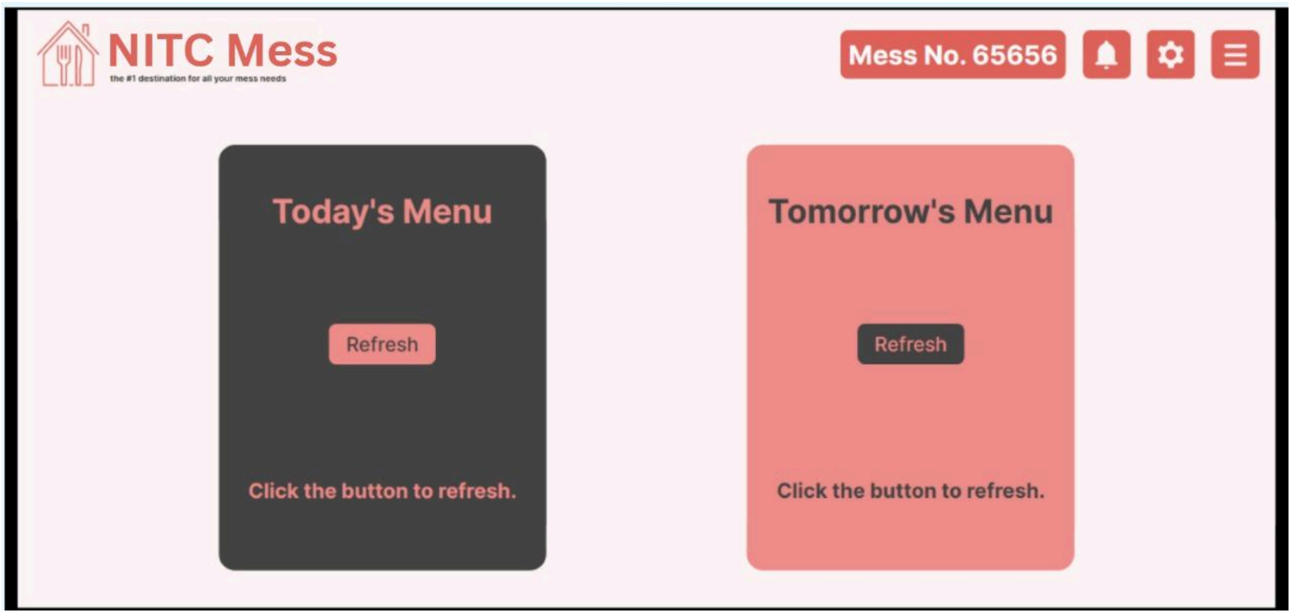

NITC Mess
the #1 destination for all your mess needs
Mess No. 65656
Today's Menu
Refresh
Click the button to refresh.
Tomorrow's Menu
Refresh
Click the button to refresh.

Figure 8 : Viewing Menu Page

Student Interface:

The student dashboard will feature a clean, intuitive layout with four primary sections: Menu View (displaying weekly meals in card format with images and nutritional information), Feedback Center (rating system with 1-5 stars and text comment fields), Attendance Manager (calendar interface for marking vacation/absence dates), and Bill Summary (displaying current month charges and payment history). Navigation will be through a bottom tab bar on mobile devices and side navigation on desktop, optimized for quick access to frequently used features.

Mess Manager Interface:

The mess manager portal will include a comprehensive dashboard with Menu Editor (drag-and-drop interface for weekly meal planning with ingredient lists), Feedback Analytics (charts and graphs showing meal ratings and common feedback themes using Chart.js), Inventory Management (table-based CRUD interface for stock tracking with low-stock alerts), and Attendance Reports (filtered views of student attendance patterns). The interface will support bulk operations and provide export functionality for reports.

Administrator Interface:

The admin panel will offer system-wide oversight capabilities including User Management (role-based user administration with search and filter options), Billing System (automated bill generation interface with batch processing capabilities), Payment Tracking (comprehensive payment status dashboard with overdue alerts), and System Analytics (comprehensive reporting dashboard showing usage statistics, popular meals, and financial summaries). The interface will feature advanced filtering, sorting, and export capabilities.

3.1.2 Software Interfaces

The NITC Hostel Mess Management System is designed as a standalone web application but includes provisions for integration with external software systems to enhance functionality and institutional compatibility.

Database Interface:

The system interfaces with MongoDB Atlas cloud database service through the Mongoose ODM (Object Document Mapper) library. This interface handles all data persistence operations including user authentication, menu management, feedback storage, and billing records. The connection utilizes MongoDB's native drivers with connection pooling and automatic failover capabilities.

Authentication Services Interface:

The system implements its own JWT-based authentication.

Future Integration Interfaces:

The system architecture includes provisions for future integration with payment gateways (Razorpay, PayU), and student information systems through standardized REST API endpoints. These interfaces will follow institutional data exchange protocols and security requirements.

Third-Party Library Interfaces:

The system utilizes various npm packages including Express.js for server framework, React.js for frontend components. All third-party dependencies are managed through npm package manager with version locking to ensure stability.

3.2 Functional Requirements

User Authentication and Management:

F1: The system shall provide secure user registration and login functionality with role-based access control for students, mess managers, and administrators.
F2: The system shall maintain user profiles with personal information, hostel details, and role-specific permissions.

Menu Management:

F3: The system shall allow mess managers to create, edit, and publish weekly meal menus with detailed descriptions and nutritional information.
F4: The system shall enable students to view current and upcoming weekly menus in an organized, user-friendly format.

Feedback and Rating System:

F5: The system shall allow students to submit meal ratings on a 1-5 scale for each meal they consume.
F6: The system shall enable students to provide written feedback and suggestions for meal quality improvement.
F7: The system shall provide mess managers with feedback analytics including average ratings.

Attendance Management:

F8: The system shall allow students to register non-attendance for specific dates during vacations or leave periods.
F9: The system shall maintain accurate attendance records for all registered students with date-wise tracking.

Billing and Payment Management:

F10: The system shall automatically generate monthly mess bills based on student attendance records and applicable rates.

F11: The system shall allow administrators to view, modify, and finalize monthly billing for all students.
F12: The system shall track payment status and provide payment history for each student account.

Administrative Functions:
F13: The system shall provide administrators with comprehensive reporting capabilities including attendance summaries, payment reports, and system usage analytics.
F14: The system shall enable inventory management functionality for mess managers to track food stock levels and consumption patterns.

3.3 Use Case Model

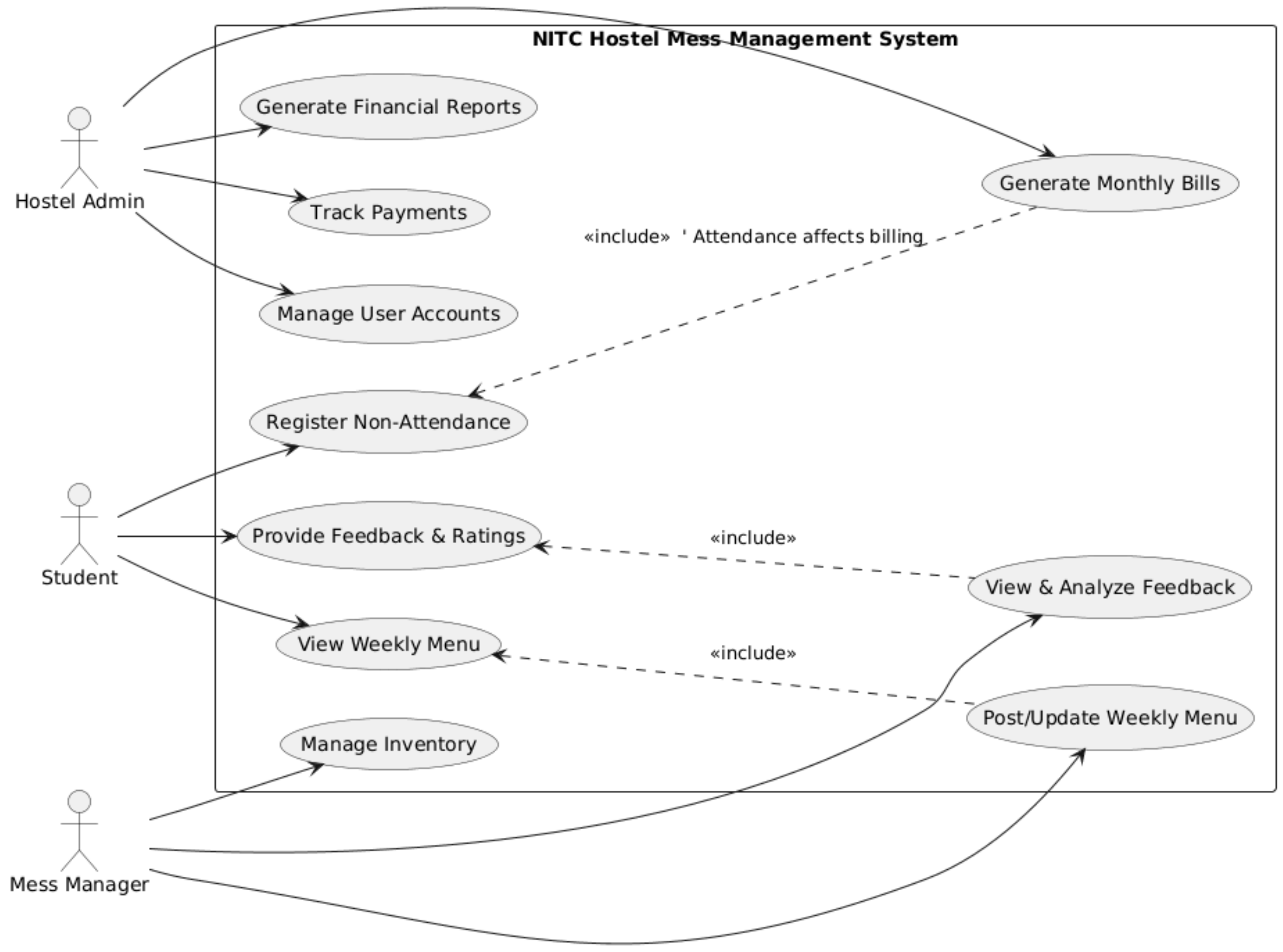


Figure 9 : Use Case diagram that encapsulates the entire system and all actors

3.3.1 Use Case #1 (U1): Hostel Admin & Billing Dashboard

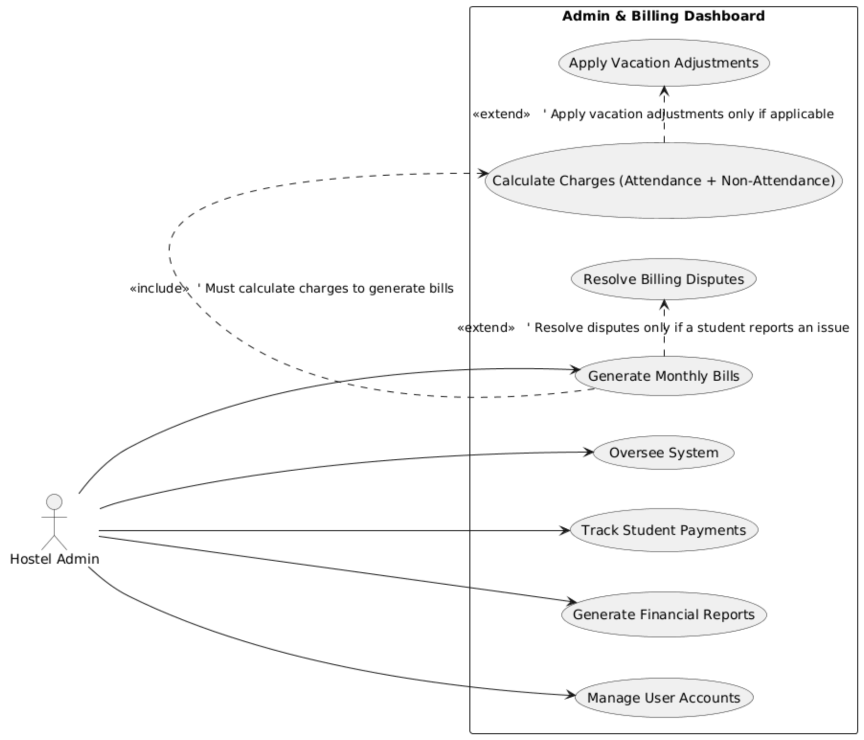

Admin & Billing Dashboard
Apply Vacation Adjustments
«extend» ' Apply vacation adjustments only if applicable
Calculate Charges (Attendance + Non-Attendance)
«include» ' Must calculate charges to generate bills
Resolve Billing Disputes
«extend» ' Resolve disputes only if a student reports an issue
Generate Monthly Bills
Oversee System
Track Student Payments
Generate Financial Reports
Manage User Accounts
Hostel Admin

Figure 10: Use Case diagram for Admin and Billing

Author – Sikta

Purpose:
This use case delineates the administrative and financial management functionalities accessible to the Hostel Admin. The objective is to ensure accurate student charge calculation, monthly bill generation, payment tracking, and financial report provision.

Requirements Traceability:

- Fn-Bill-001: Automated Bill Generation
- Fn-FinRep-002: Financial Reporting
- Fn-Acc-003: User Account Management
- Fn-Sys-004: System Oversight

Priority: High

Preconditions:

- Hostel Admin must be authenticated and authorized.
- Student attendance, non-attendance, and vacation approval data must be present in the database.

Postconditions:

- Monthly bills generated and recorded.
- Financial reports created and stored.
- Payment records updated.
- User accounts maintained.

Actors: Hostel Admin

Extends:

- Apply Vacation Adjustments → Calculate Charges
- Resolve Billing Disputes → Generate Monthly Bills

Includes:

- Calculate Charges → Generate Monthly Bills

Flow of Events:

- **Basic Flow:**
  - Admin initiates Generate Monthly Bills.
  - The system calculates charges (attendance + non-attendance).
  - Bills generated and accessible to students.
  - Admin tracks payments.
  - Admin generates financial reports.
  - Admin manages accounts and oversees the system.
- **Alternative Flow:**

- AF1: Apply Vacation Adjustments during charge calculation.
- AF2: Resolve Billing Disputes if reported.

- **Exceptions:**
  - E1: Data Incompleteness halts bill generation.
  - E2: Processing Error terminates calculation, logs error.

Notes/Issues:
Dispute resolution requires robust tracking to ensure timely handling.

3.3.2 Use Case #2 (U2): Mess Manager Console

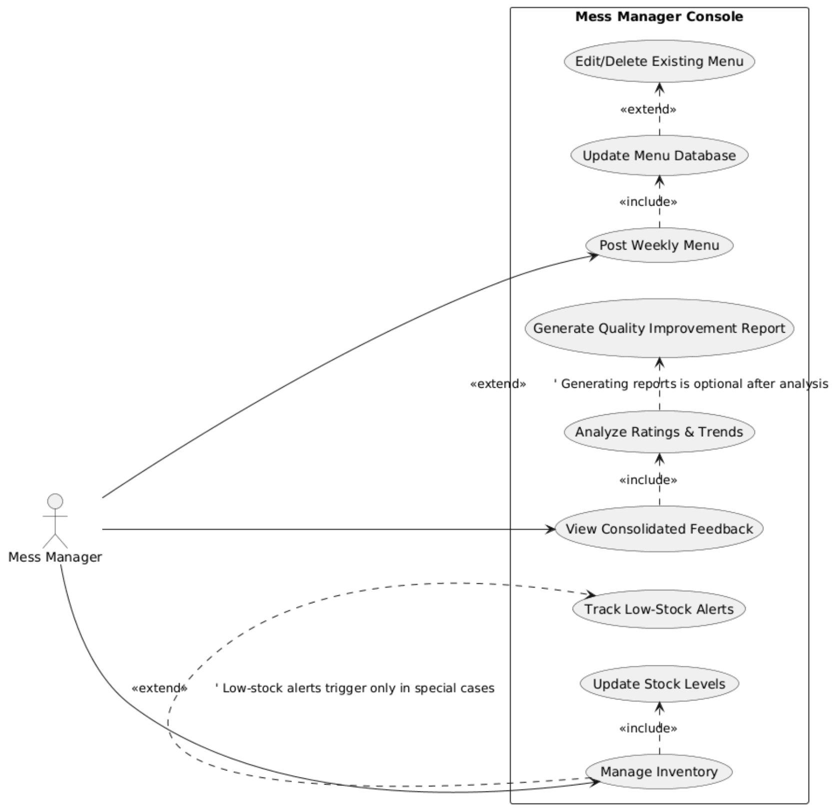


Figure 11: Use Case diagram for Mess Manager Console

Author: Vengidesan

Purpose:
This use case specifies menu management, inventory control, and feedback-based reporting by the Mess Manager.

Requirements Traceability:

- Fn-Menu-001: Menu Database Management
- Fn-Inv-002: Inventory Control
- Fn-Feedback-003: Feedback Analysis
- Fn-Report-004: Quality Reporting

Priority: High

Preconditions:

- Mess Manager logged in with valid credentials.
- Menu, inventory, and feedback data available in the database.

Postconditions:

- Weekly menu updated and accessible.
- Inventory levels updated.
- Reports generated if required.

Actors: Mess Manager

Extends:

- Edit/Delete Existing Menu → Update Menu Database
- Generate Quality Improvement Report → Analyze Ratings & Trends
- Track Low-Stock Alerts → Manage Inventory

Includes:

- Update Menu Database → Post Weekly Menu
- Analyze Ratings & Trends → View Consolidated Feedback
- Update Stock Levels → Manage Inventory

Flow of Events:

- **Basic Flow:**
    - Manager posts weekly menu (includes update).
    - Manager manages inventory (includes stock updates).
    - Manager views feedback (includes analysis).
- **Alternative Flow:**
    - AF1: Edit/Delete menu entry.
    - AF2: Generate quality report.
    - AF3: Low-stock alerts triggered.
- **Exceptions:**
    - E1: Database write failure prevents updates.
    - E2: Invalid data input rejected.

Notes/Issues: Low-stock thresholds must be clearly defined for effectiveness.

3.3.3 Use Case #3 (U3): Student Mess Services

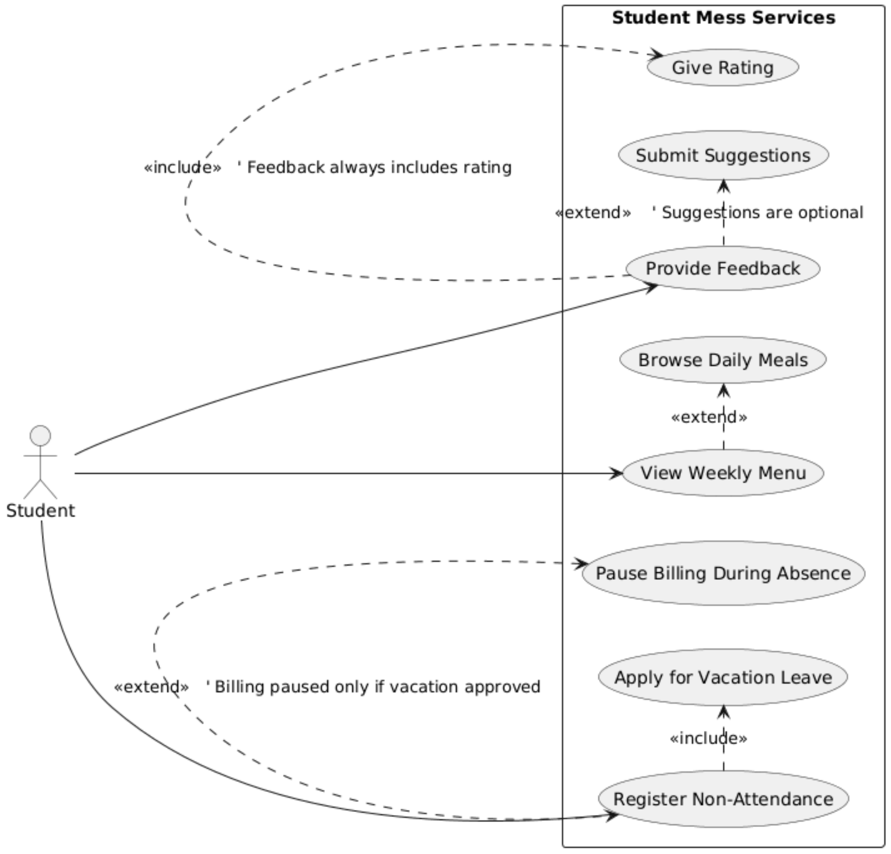


Figure 12 : Use Case diagram for Student Mess Services

Author: Nishant

Purpose:
Defines student interactions, including viewing menus, giving feedback, and managing billing.

Requirements Traceability:

- Fn-Menu-001: Menu Viewing
- Fn-Feedback-002: Feedback Submission
- Fn-Billing-003: Billing Management

Priority: High

Preconditions:

- Student authenticated.

- Menu and billing policies present in system.

Postconditions:

- Feedback recorded.
- Billing updated on leave approval.
- Menu successfully viewed.

Actors: Student

Extends:

- Submit Suggestions → Provide Feedback
- View Weekly Menu → Browse Daily Meals
- Pause Billing During Absence → Apply for Vacation Leave

Includes:

- Give Rating → Provide Feedback
- Register Non-Attendance → Apply for Vacation Leave

Flow of Events:

- **Basic Flow:**
  - Student browses daily meals.
  - Student provides feedback (includes rating).
  - Student registers non-attendance.
  - Student applies for vacation leave (includes non-attendance).
- **Alternative Flow:**
  - AF1: Student submits suggestions with feedback.
  - AF2: Student views weekly menu.
  - AF3: Vacation approval triggers billing pause.

- **Exceptions:**
    - E1: Leave request denial prevents billing pause.
    - E2: Database error prevents saving feedback/leave.

Notes/Issues:
Billing pause must be tightly integrated with vacation approval.

4 Other Non-functional Requirements

4.1 Performance Requirements

Response Time Requirements

- The system should provide response times of less than 5 seconds for standard operations such as menu viewing, feedback submission, and attendance marking under normal load.
- Database queries for retrieving user-specific information (bills, attendance history) should complete within 3 seconds to keep the application responsive.
- Complex operations such as generating a monthly bill or creating reports should complete within 15–20 seconds.

Concurrent User Support

- The system should support at least 20–30 concurrent users, representing students and one or two mess managers accessing the system at the same time.
- The architecture only needs to handle basic simultaneous access without noticeable slowdowns (e.g., students viewing menus or giving feedback at the same time).

4.2 Safety and Security Requirements

Authentication and Access Control:
The system will keep user accounts secure by using strong password encryption, and only authorized users will have access to sensitive features. Administrators will have extra security measures when logging in. Students cannot access admin features, and managers cannot change billing records.

Data Protection and Privacy:

Student data, feedback, and bills will be encrypted when stored or shared. Personal information will be kept private and separate from other system data. Feedback will stay anonymous. The system will follow institutional privacy policies.

Financial Data Security:

Every billing change will be recorded with who made the change and when. No one can change past bills without proper logs for review.

4.3 Software Quality Attributes

4.3.1 Reliability

Database operations will be reliable, preventing data corruption through safe transaction handling.

4.3.2 Maintainability

The system will use a modular structure, making it easy to update individual parts of the frontend, backend, and database.

4.3.3 Usability

The system will be easy to use, with a simple interface that allows users to complete main tasks (like viewing menus or submitting feedback) in a few clicks.

4.3.4 Portability

The application will run reliably across different computers and major cloud platforms, and will work with modern browsers.

5 Other Requirements

Legal and Compliance Requirements:

All user data collection and processing shall be transparent with clear privacy policies communicated to users during registration. The system shall provide data export functionality allowing students to retrieve their personal data upon request, supporting data portability rights.

Appendix A - Activity Log

Team Lead: Nishant
Team Members: Sikta, Vengidesan

This log records the group’s meetings and contributions during the preparation of use case diagrams and specifications for the System Engineering Specification (SES) document on **September 20–21, 2025**.

Date	Session	Time	Activity	Contributions
Saturday, Sep 20, 2025	Morning Session	10:00 AM - 1:00 PM	**Meeting**	**Nishant**: Led discussion on Hostel Admin & Student use cases (U1, U4), finalized U1 diagram, began drafting specification. **Sikta**: Worked on Mess Manager use case (U2), clarified inventory details, initiated U2 diagram & documentation. **Vengidesan**: Defined actor roles, completed U3 diagram, began its specification.
	Afternoon Session	2:30 PM - 5:00 PM	**Work Session**	**Nishant**: Completed detailed specification for U3. **Sikta**: Completed detailed specification for U1. **Vengidesan**: Drafted detailed specification for U2.
Sunday, Sep 21, 2025	Morning Session	11:00 AM - 12:30 PM	**Meeting**	**Nishant**: Reviewed & approved all specifications, responsible for document compilation. **Sikta**: Verified terminology consistency across U1 & U2. **Vengidesan**: Presented & finalized consolidated high-level diagram (U2).
	Afternoon Session	2:00 PM - 4:00 PM	**Finalization**	**Nishant**: Compiled final activity log, proofread SES document. **Sikta**: Backed up all completed files. **Vengidesan**: Prepared all diagrams, verified requirement traceability numbers.