CS432 – Module 3 Task Report

Medical Dispensary Management System for IIT Gandhinagar

Team Members

- **Harshith Venigalla** (22110283)
- **Sai Charan Miriyala** (22110153)
- **Sriram Srinivasan** (22110258)
- **Pavan Deekshith** (22110190)

---

1. Introduction

We developed a secure, database-integrated **Medical Dispensary Management System** for IIT Gandhinagar as part of CS432 Module 3. This system supports:

- Appointments,
- Prescriptions,
- Staff management,
- Inventory control, and
- Emergency handling.

It integrates seamlessly with the **centralized Campus Information Management System (CIMS)** and enforces **session-based validation**, **role-based access control**, and **logging mechanisms**.

2. System Architecture

2.1 Database Design

We use **two databases** in compliance with project guidelines:

Centralized CIMS Database (cs432cims)

Used without duplication:

- `members`: Stores user identity information.
- `login`: Manages credentials and session tokens.
- `members_group_mapping`: Maintains user-role mapping.

Project-Specific Database (cs432g12)

Custom tables for dispensary operations:

- `G12_Doctors`: Doctor profiles.
- `Staff`: Non-medical staff.
- `Appointments`: Doctor-patient bookings.
- `Prescriptions`: Doctor-issued prescriptions.
- `Medications`: Medicine inventory.
- `MedicinesGiven`: Medicine dispensation records.
- `Students`: Details of student users.
- `Vehicles`: Ambulance tracking.
- `HospitalReferrals`: Partner hospitals for referrals.
- `EmergencyCases`: Emergency patient records.
- Supporting tables like `DoctorAvailability`, `Referrals`, etc.

No central tables like `members` or `login` were duplicated.

2.2 System Components

Component	Description
Authentication	Session validation via JWT tokens
Role-Based Access	Roles: Admin, Doctor, Staff, Student
Frontend	Responsive UI with separate views per role
API Layer	RESTful APIs with session & role validation

Logging	Logs all access and data changes locally and to server

3. Implementation Details

3.1 Member Creation (Task 1)

When new members are created:

- We insert into `members` (CIMS).
- Corresponding `login` entry is added with default hashed credentials.
- Role is linked via `members_group_mapping`.

3.2 Role-Based Access Control (Task 2)

Implemented via backend decorators:

- Admin: Full access to all modules.
- Doctor: View appointments, add prescriptions.
- Staff: Manage inventory and ambulance requests.
- Student: Book appointments and view history. Unauthorized access attempts are blocked and logged.

3.3 Member Deletion (Task 3)

Handled carefully to maintain integrity:

- If a member has other group links: only `members_group_mapping` is updated.
- Else: remove from both `members` and `login`.

3.4 Table Creation (Task 4)

All dispensary-specific tables were created only in `cs432g12`, with appropriate foreign key constraints referencing CIMS tables.

3.5 API Development (Task 5)

All APIs:

- Use middleware to verify session tokens via `/isValidSession`.
- Restrict admin-level functions via role-check decorators.
- Log both successful and failed operations locally and to a server log file.
- Unauthorized direct DB access is flagged and logged.

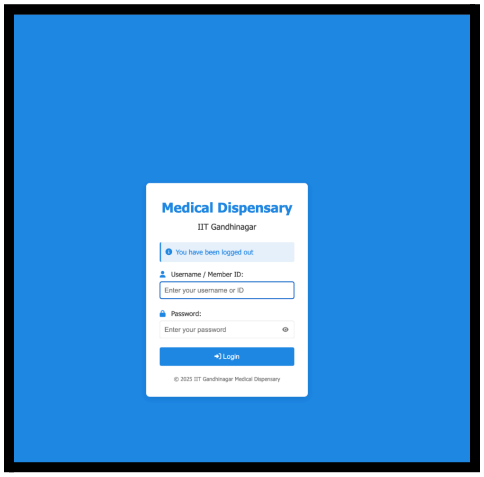
3.6 Logging Changes (Task 6)

- Every API call that modifies CIMS data logs both:
    - **Local log** (`logs/changes.log`)
    - **Server log** (via logging microservice)
- Transactions without valid sessions are not logged on server, exposing any bypass attempts.

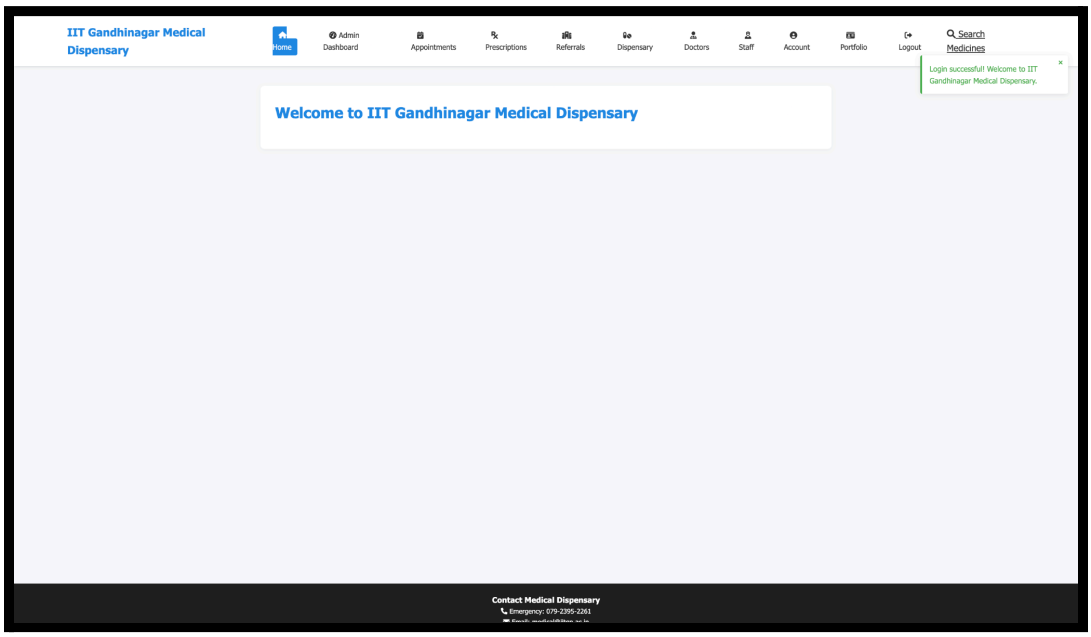
4. User Interface Showcase

4.1 Login System

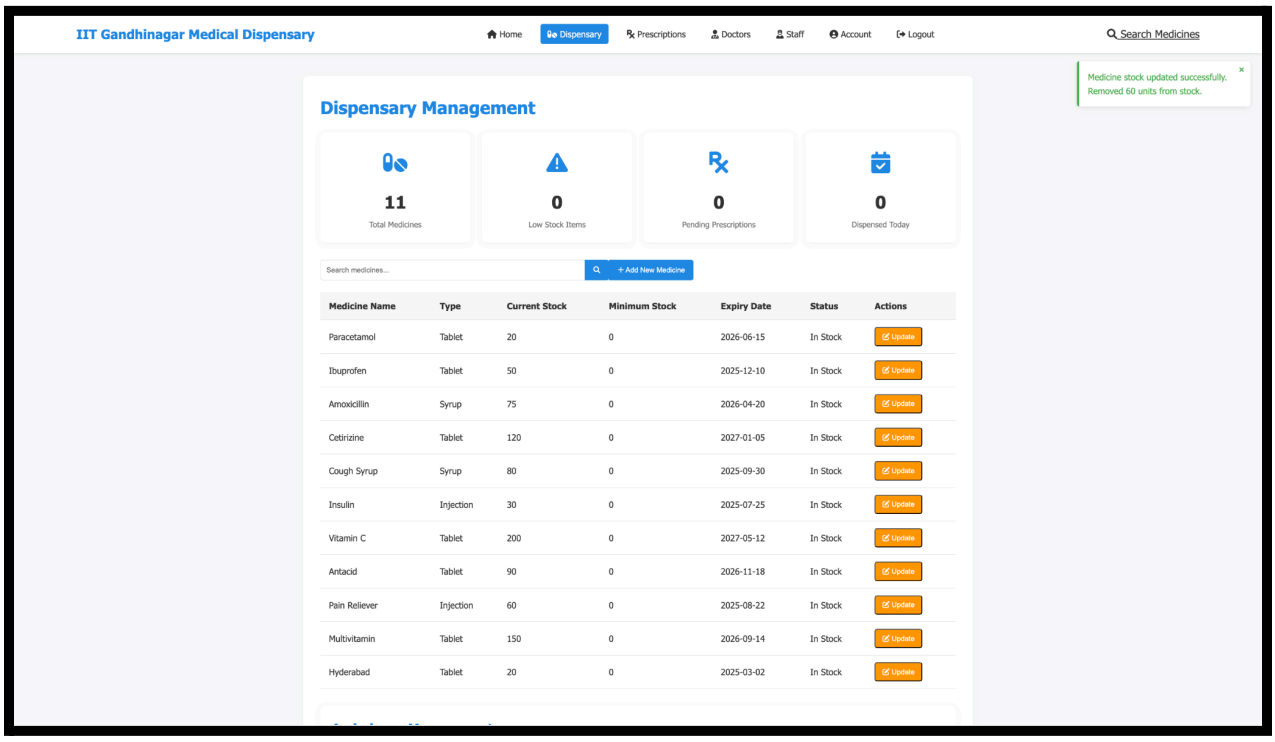
- Secure password hashing (bcrypt)
- JWT token issuance
- Role-based dashboard redirection



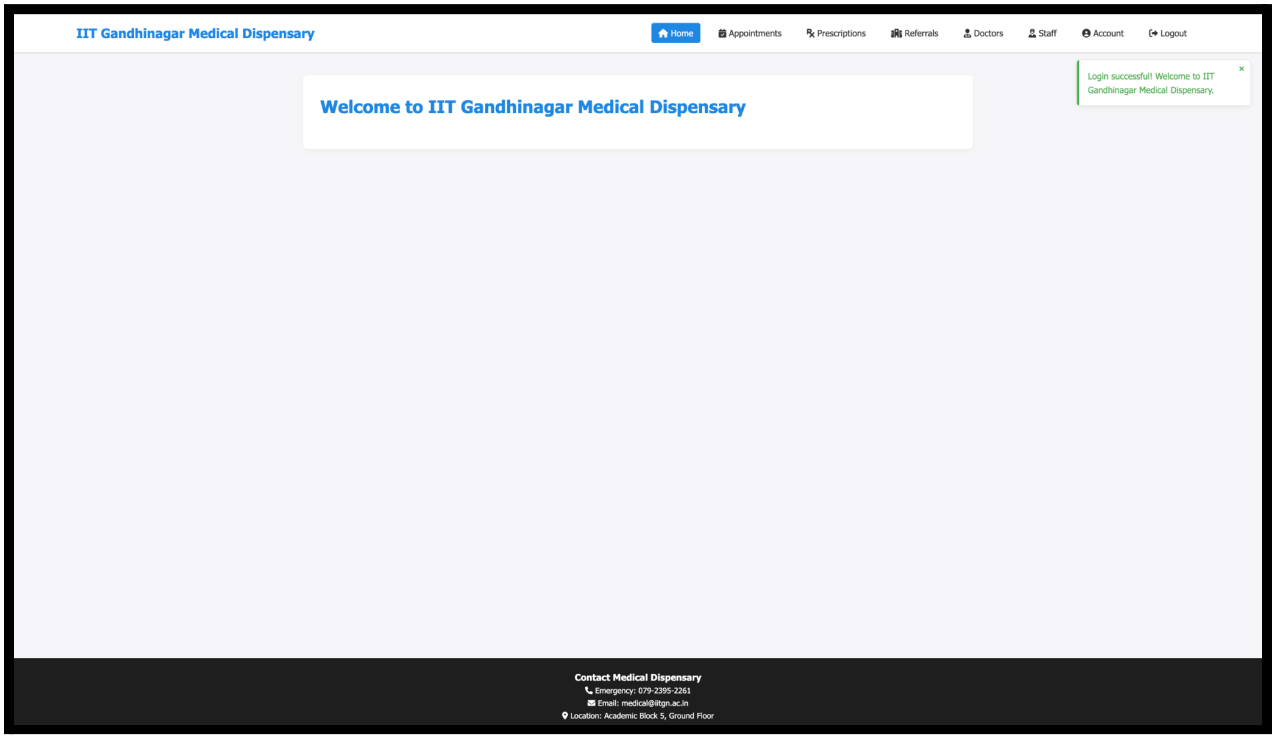
For Admin:



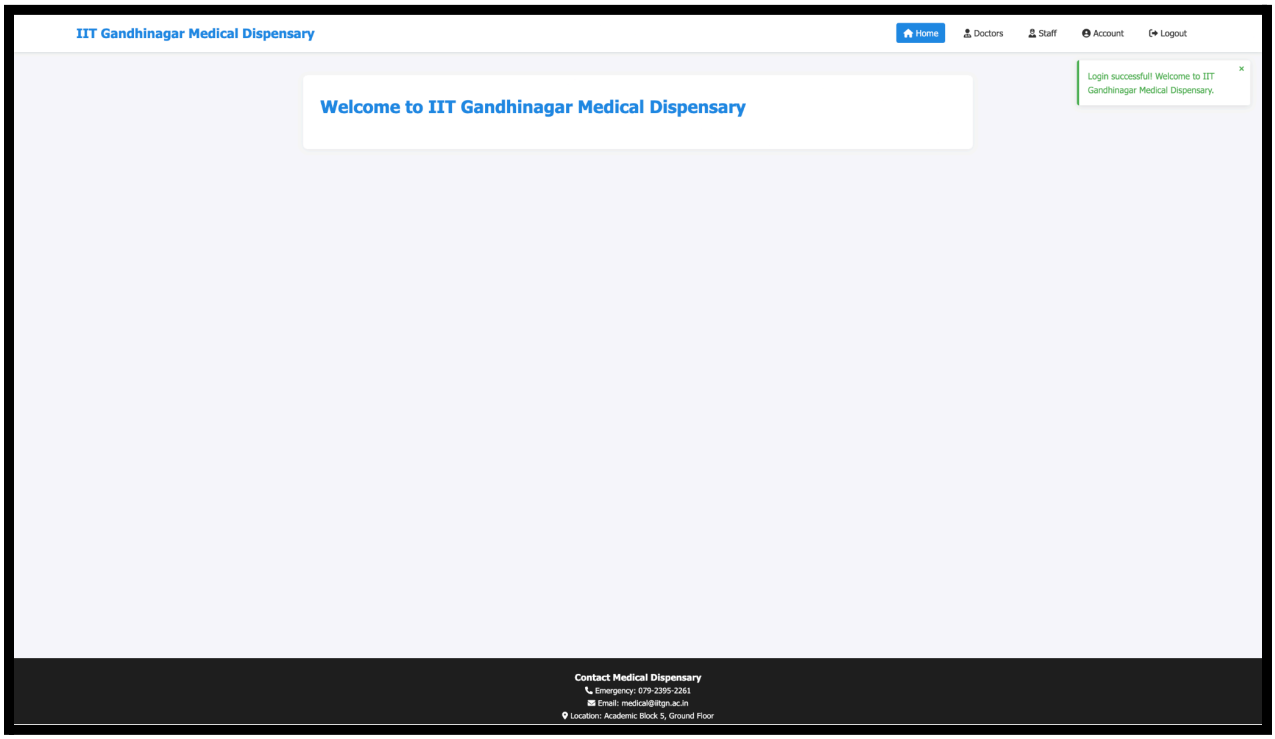
For Staff:



For Doctors:

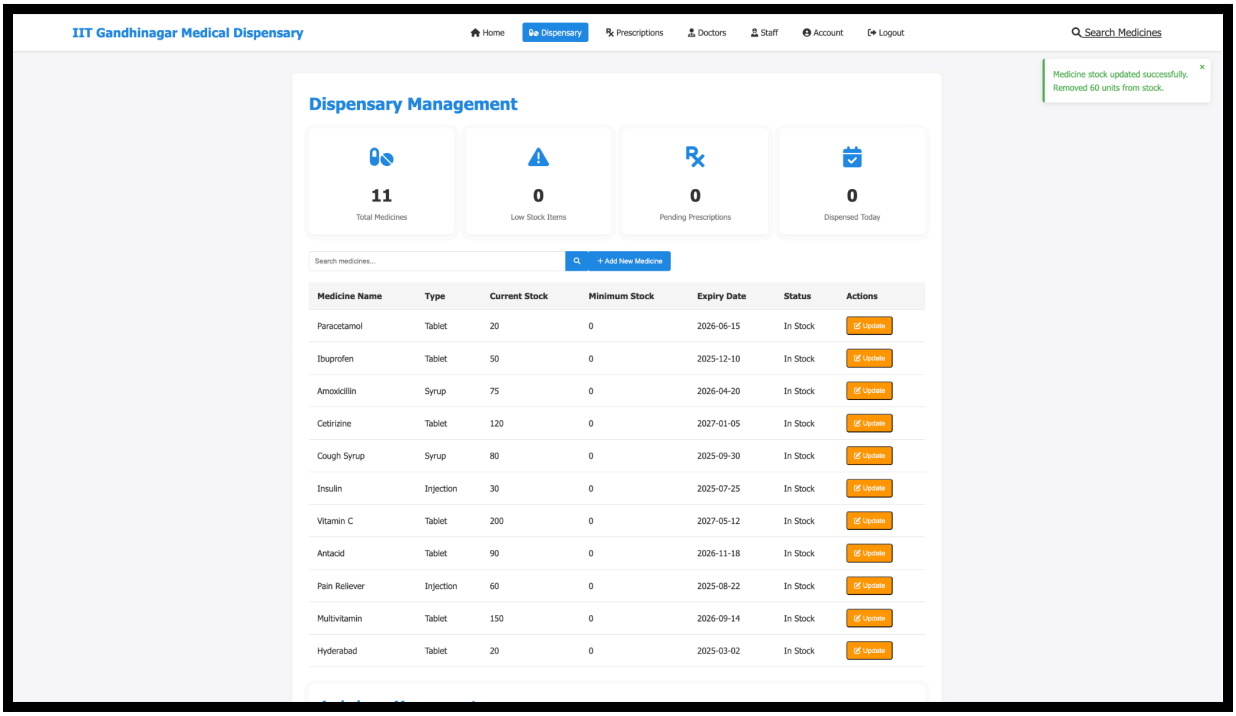


For students:

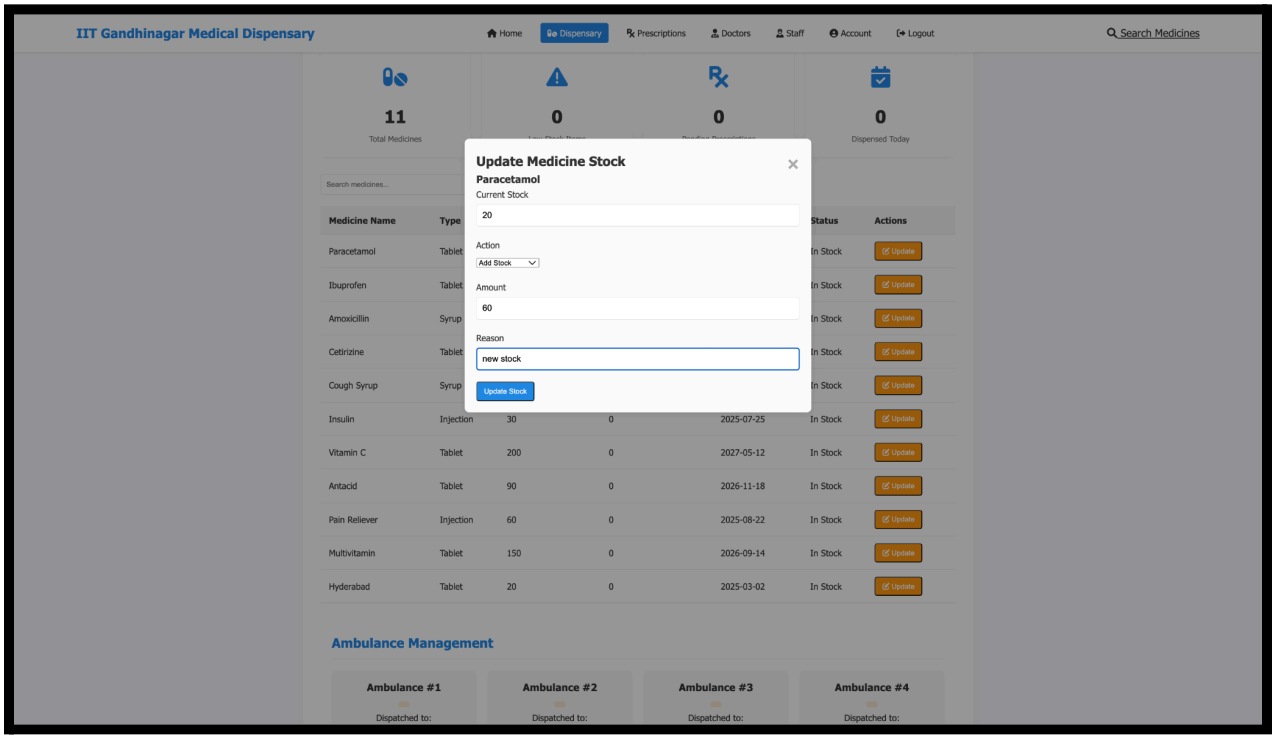


4.2 Dispensary Dashboard (Staff)

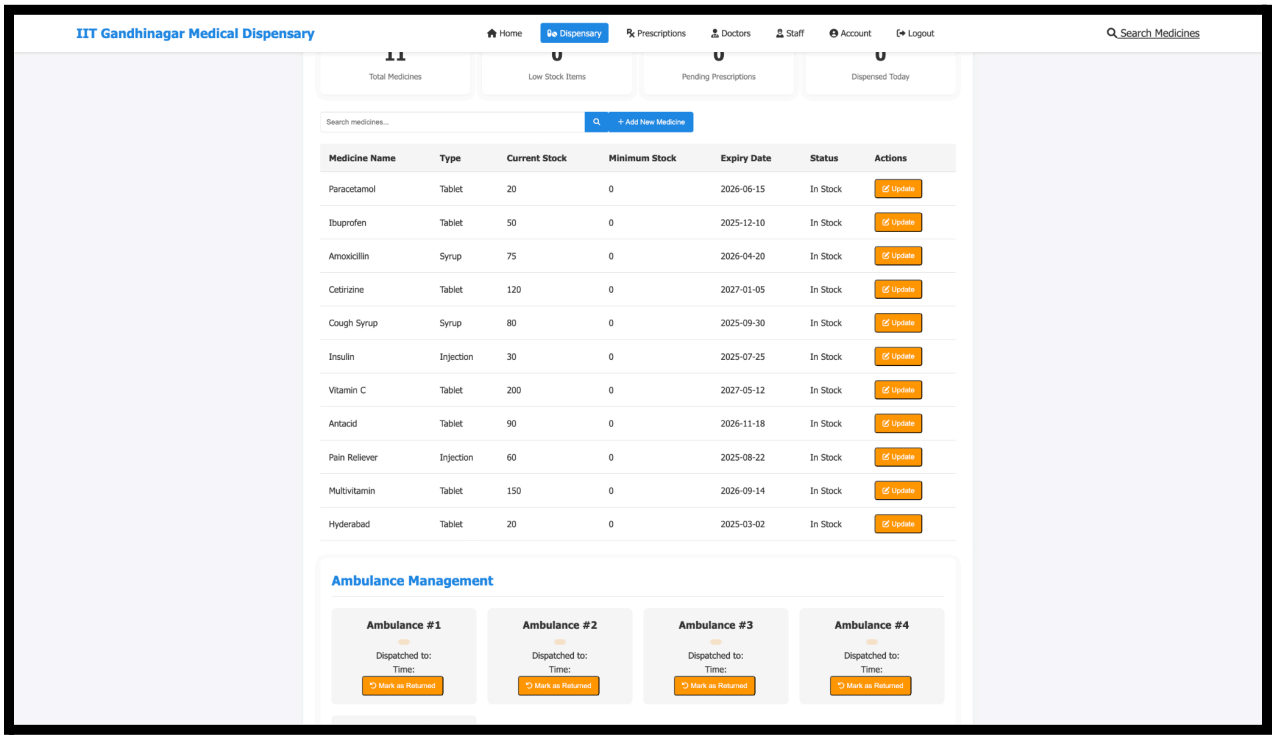
- Track stock and low inventory



- Dispense and log medicines

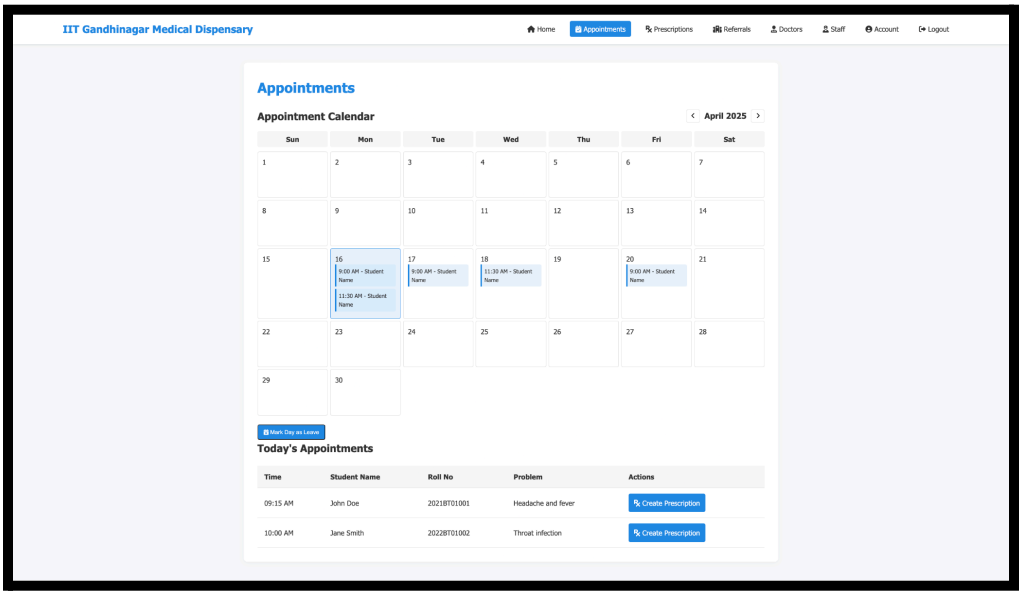


- **View ambulance status**

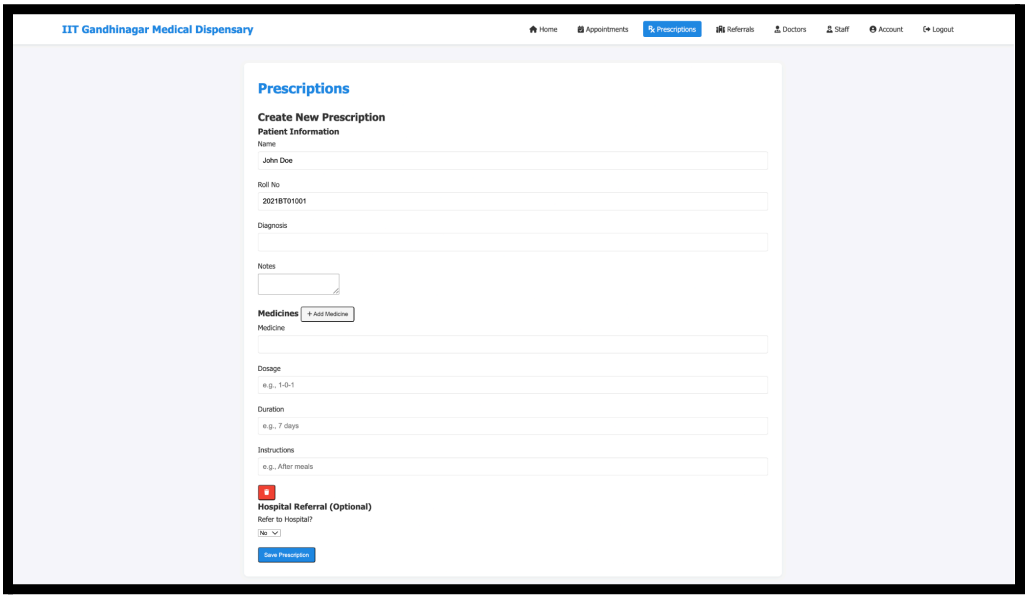


4.3 Doctor Dashboard

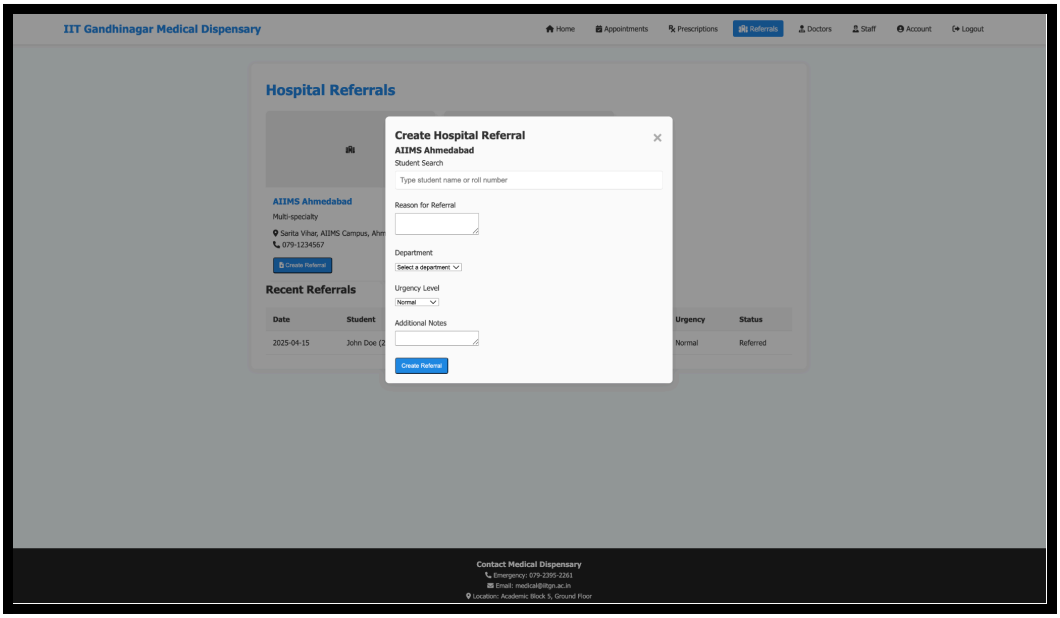
- View/manage appointments



- Issue or create prescriptions

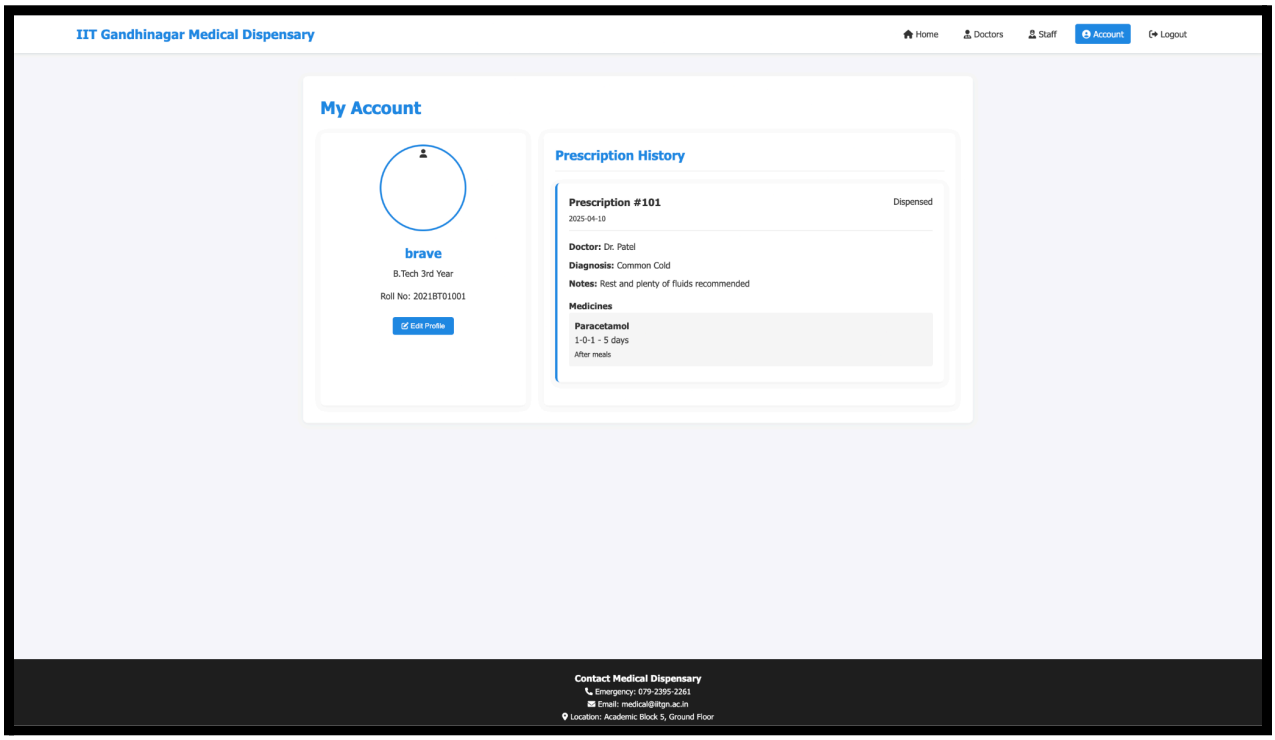


- Create referrals

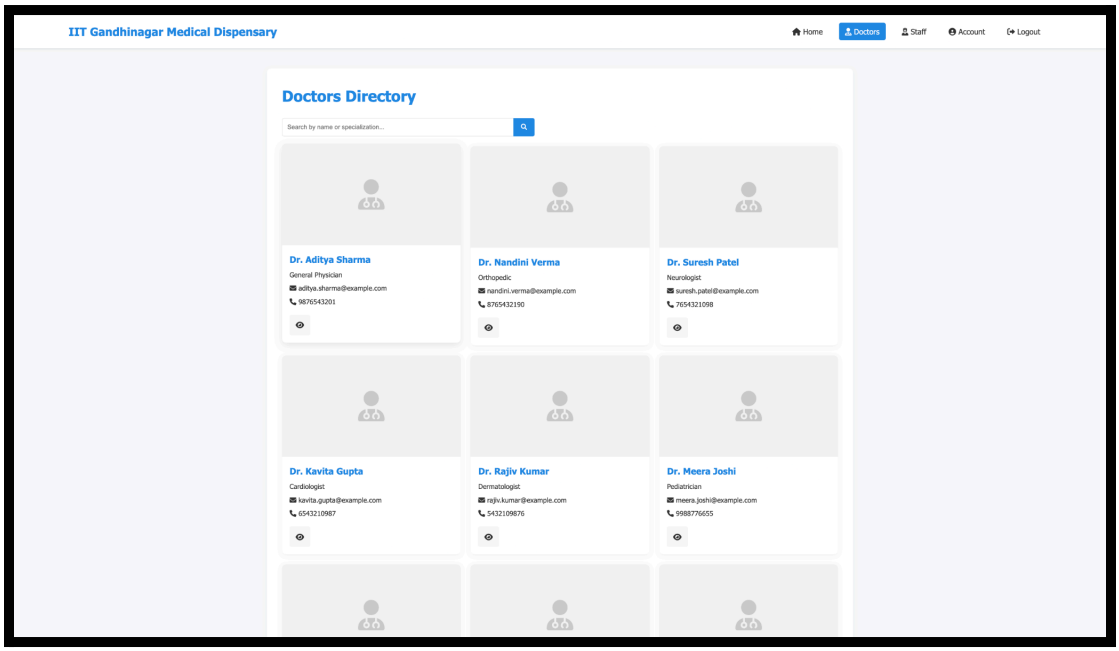


4.4 Student Dashboard

- View prescriptions



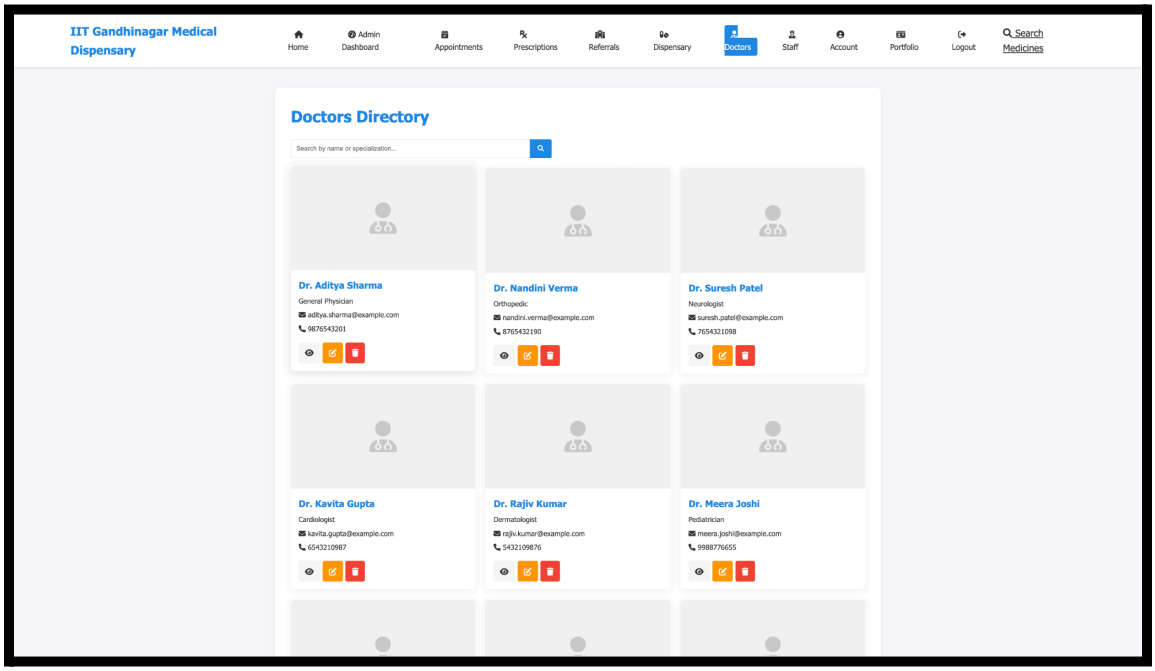
- Browse doctors and staff



5. Admin Features

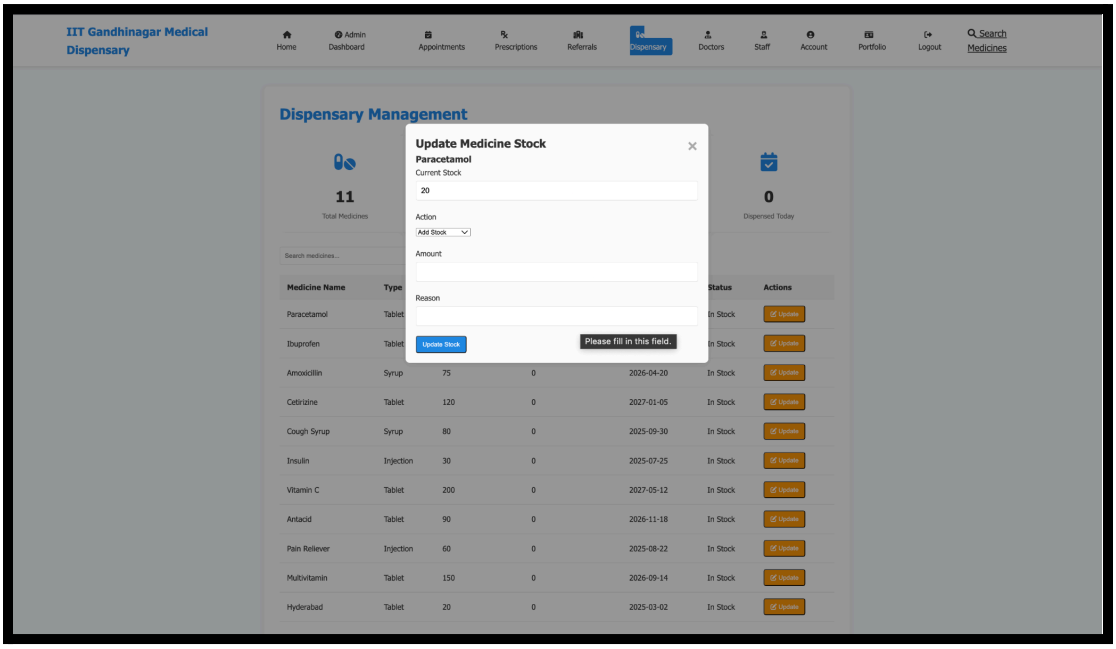
Admins can:

- Add/edit/delete users (via API)

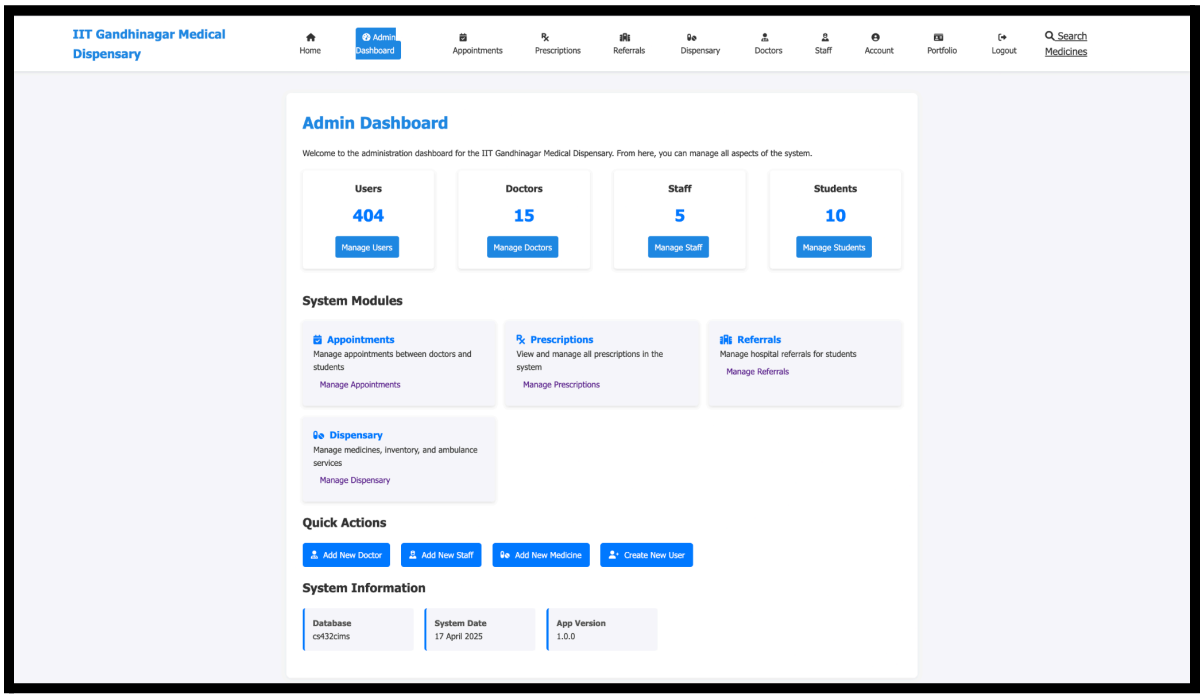


Admin can edit or delete a doctor

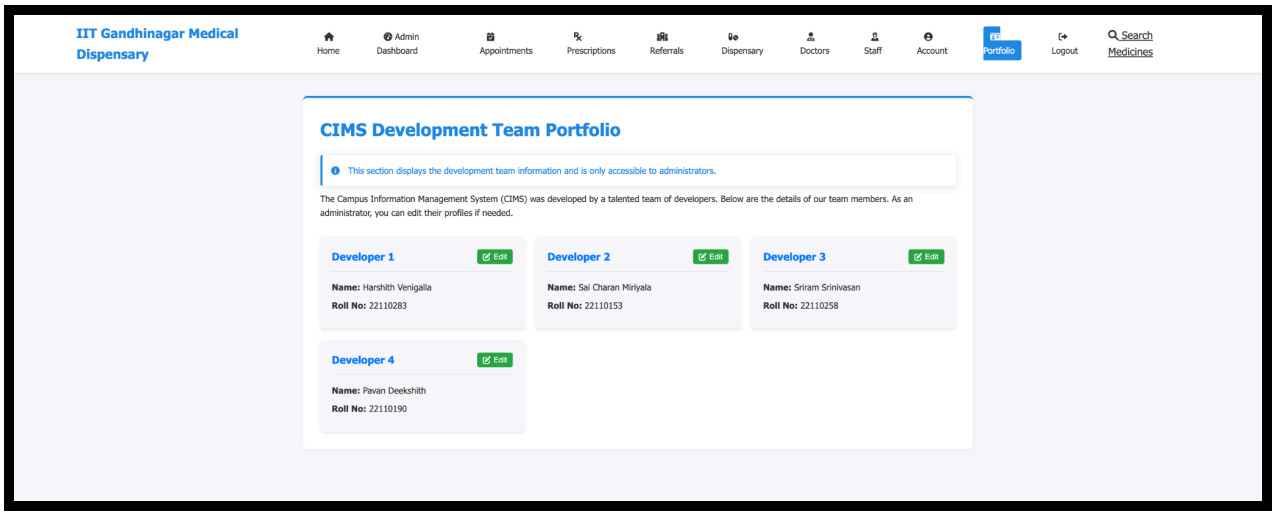
- Edit dispensary details



- Access complete system analytics



- View developer portfolios



6. Security Highlights

Security and data integrity are central to our system design. We have implemented multiple layers of protection to ensure safe and authorized access:

Security Feature	Implementation Details
Session Management	JWT tokens with expiry are issued at login and validated on every API request.
Cookie Handling	Tokens are stored as secure, HTTP-only cookies to prevent XSS and client-side tampering.
Password Security	All passwords are hashed using bcrypt before storage in the `login` table.
Role-Based Access Control (RBAC)	Access to APIs and frontend routes is strictly controlled using user roles stored in the `members_group_mapping` table.
Input Validation	All API inputs are validated using server-side checks to prevent SQL injection and malformed requests.
SQL Query Safety	Parameterized queries are used throughout the application to avoid injection vulnerabilities.

These measures collectively ensure that only authorized users can perform sensitive operations and data remains protected from misuse or attacks.

7. Member Portfolio Management (Task 7)

As per the task requirements, we implemented a **Member Portfolio Management Module** that allows the Admin to view and manage basic information about each registered team member.

🔐 Access Control

- **Admin-Only**: The portfolio is strictly accessible only to users with the Admin role.
- **Hidden from Other Groups**: Students, doctors, and staff cannot view or modify portfolio information.

📄 Features

- Displays each member's **Name**, **Roll Number**, and **Role**.
- Admins can **edit details** of the members via a secure UI.
- Provision for **image upload** is structurally in place (though not enabled), allowing easy extension in future versions.

This module fulfills **Task 7** requirements by demonstrating secure admin-level data access and group-based content visibility.

8. Logging and Audit Trail

We implemented a robust logging mechanism to monitor and audit all critical operations, especially those involving CIMS data.

🗂️ Local Logging

- All changes to the `members`, `login`, and `members_group_mapping` tables are logged to a **local file** (`logs/changes.log`).
- Includes timestamps, operation type, affected user ID, and API endpoint.

🌐 Server Logging

- Valid session-based operations are also logged to a **central server log**, ensuring consistency and traceability across deployments.

🚨 Unauthorized Activity Logging

- Attempts to access APIs without valid sessions are:
    - Denied access,
    - Logged locally,

    - **Not logged to the server**, which highlights potential bypass attempts.

These audit mechanisms ensure transparency, simplify debugging, and make the system resilient to misuse.

9. Conclusion

This project successfully meets all the objectives outlined in **CS432 Module 3 – Secure API and Database Integration**. We designed a functional, secure, and scalable **Medical Dispensary Management System** that:

- Integrates with the existing **CIMS** without duplicating central tables,
- Supports **role-based access control** and **session-based validation**,
- Offers a fully functional **admin dashboard** and **member portfolio module**,
- Logs all operations for **accountability and auditability**, and
- Separates sensitive modules with strict **group-level visibility** rules.

The system is modular, extensible, and deployable, making it a solid foundation for real-world institutional use.

10. GitHub Repository

Github: https://github.com/pavandeekshith/Medical_Center_g12
Youtube:

11. Team Member Roles

Name	Contribution
Harshith Venigalla	Backend APIs, Database schema, Logging mechanism
Sai Charan Miriyala	Role-based decorators, Authentication system, Portfolio module
Sriram Srinivasan	Frontend design, Admin dashboard, Session management
Pavan Deekshith	Prescriptions module, Appointments, UI testing