Project Design Phase

Technology Stack

Date	**01 NOV 2025**
Team ID	**NM2025TMID02942**
Title	**Medical Inventory System**
Maximum Marks	**4 Marks**

1. Introduction

The Medical Inventory Management System (MIMS) will be developed using a modern, secure, and scalable technology stack. The chosen technologies support:

- Real-time data processing,
- Cross-platform accessibility,
- High availability, and
- Ease of maintenance.

This stack is designed to support a web-based system accessible by hospital staff, pharmacists, procurement officers, and administrators across different devices.

---

2. Technology Stack Overview

Layer	Technology / Tool	Purpose / Description
Frontend (Client-Side)	HTML5, CSS3, JavaScript, ReactJS / Angular	To build an interactive and responsive user interface that can run on web browsers across devices.

Layer	Technology / Tool	Purpose / Description
Backend (Server-Side)	Python (Django / Flask) or Node.js (Express)	To handle business logic, process requests, and manage data flow between the frontend and the database.
Database	MySQL / PostgreSQL	To store structured data such as inventory details, user accounts, transaction logs, and audit trails.
Authentication & Security	JWT (JSON Web Tokens), OAuth 2.0	To provide secure user authentication and role-based access control.
Web Server	Apache / Nginx	To host and serve the web application to clients.
Application Server	Gunicorn / uWSGI (for Python-based apps)	To run backend applications efficiently in production.
Cloud Platform / Hosting	AWS / Microsoft Azure / Google Cloud	For cloud deployment, storage, and scalability.
Notifications	Twilio / SMTP (Email API)	To send automated alerts and notifications for low stock or expiry warnings.
Version Control	Git, GitHub / GitLab	For source code management and collaborative development.
Testing Tools	PyTest / Jest / Selenium	For unit testing, integration testing, and UI testing.
Continuous Integration / Deployment (CI/CD)	GitHub Actions / Jenkins	To automate testing and deployment pipelines.
Containerization (Optional)	Docker	To package and deploy applications consistently across environments.

Layer	Technology / Tool	Purpose / Description
Operating System	Linux (Ubuntu / CentOS)	For server deployment due to its stability and security.
Backup & Recovery	AWS S3 / Azure Backup	To store regular backups of the database and application data securely.

---

3. System Architecture Overview

The system follows a 3-tier architecture:

a. Presentation Layer (Frontend)

- Handles user interface and interaction.
- Provides dashboards for different roles (Admin, Pharmacist, Nurse, Procurement Officer).
- Built using ReactJS or Angular for responsive, single-page applications.

b. Application Layer (Backend)

- Contains business logic for inventory tracking, alert generation, and purchase order processing.
- Communicates between frontend and database through RESTful APIs.
- Built using Python Django/Flask (Model-View-Controller architecture) or Node.js with Express.

c. Data Layer (Database)

- Stores persistent data such as stock records, supplier info, transactions, and audit logs.
- Implements data integrity, indexing, and relationships using MySQL/PostgreSQL.
- Supports backup, restore, and recovery operations.

---

4. Integration Components

Integration Type	Technology	Purpose
API Layer	RESTful APIs	To integrate with hospital management systems or supplier portals.
Email/SMS Gateway	SMTP / Twilio API	For sending alerts and reorder notifications.
Authentication Service	JWT / OAuth2	For user authentication and secure session management.
Reporting Tools	Chart.js / Power BI / Google Charts	For visual analytics and dashboard reports.

---

5. Development Environment Setup

Environment	Tools Used
Code Editor / IDE	Visual Studio Code / PyCharm / IntelliJ
Local Server	XAMPP / Docker / Django built-in server
Package Manager	npm / pip
Version Control	Git
Project Tracking	Jira / Trello / Asana
Testing Environment	Postman (API testing), PyTest, Selenium

---

6. Security Considerations

- HTTPS and SSL encryption for all data in transit.
- Passwords stored using bcrypt / Argon2 hashing.
- Role-based access and authentication using JWT tokens.
- Regular backups and server-side logging for audit trails.
- Protection against SQL Injection, XSS, and CSRF attacks.

7. Deployment Plan

Stage	Description
Development Environment	Local setup for coding and testing individual modules.
Testing Environment	Separate staging server for integration and user acceptance testing.
Production Environment	Final deployment on cloud server (AWS EC2 / Azure VM) with continuous monitoring and automated backups.