**WEB BASED HOSPITAL MANAGEMENT SYSTEM USING AI**

**Final Year Project Report**



GSN: Fall 24-5

**Group Members**

|  |  |
| --- | --- |
| Muhammad Rafey Umar | 21L-5741 |
| Ali Khurram | 21L-7665 |
| Abdullah Ahmad | 21L-5748 |

|  |  |
| --- | --- |
| **Advisor** | Sir Ahmad Hamza |
| **Client** | Dr. Saima Zafar |

27th November 2024

Department of Electrical Engineering

National University of Computer and Emerging Sciences, Lahore

**CERTIFICATIONS**

This document has been prepared by all of us together and we take joint ownership of its contents. We have provided references to the material consulted in preparing this document and, to the best of our knowledge, have not plagiarized anything.

Muhammad Rafey Umar, 21L-5741 Date: 27th November 2024

Abdullah Ahmad, 21L-5748 Date: 27th November 2024

Ali Khurram, 21L-5741 Date: 27th November 2024

I am the client of the product proposed in this document and the product specifications and other details are according to my requirements.

**Client:**

Dr. Saima Zafar Date: 27th November 2024

The final year project proposal in this document is being submitted to the department of Electrical Engineering with my approval.

**Advisor:**

Sir. Ahmad Hamza Date: 27th November 2024

**Head of Department:**

Dr. Saima Zafar Date: 27th November 2024

**List of Contents:**

**1 Introduction 5**

**2 Problem Definition 6**

2.1 Problem Formulation 6

2.2 Mapping to Sustainable Development Goals (SDG) 8

2.3 Record of Meetings with Client 9

* 1. Preliminary Product Specification 10

2.5 Expected Functionality of Product 13

**3 Problem Analysis 16**

3.1 Engineering Problem Model 16

3.2 Recent Similar Projects 18

3.3 Distinguishing Features of Project 19

3.4 Societal and Environmental Implications of the Project 20

**4 Solution Design 21**

4.1 Requirements and Constraints 21

* 1. Preliminary Design 22

4.2.1 Hardware Block Diagram 24

4.2.2 Software Block Diagram 24

4.3 Detailed Hardware and Software Design 25

4.3.1 Calculations 27

4.3.2 Hardware Design 27

4.3.3 Software Design 29

**5 Deliverables and Cost 33**

5.1 Deliverables 33

5.2 Project Plan 34

5.3 Estimated Cost 36

5.4 Items to Purchase 37

**References 38**

**Glossary 39**

**List of Figures:**

[Figure 1: Sustainable Development Goals 8](#_Toc184814337)

[Figure 2: Target Sustainable Development Goal: SDG 3 9](#_Toc184814338)

[Figure 3: Hardware Block Diagram 24](#_Toc184814339)

[Figure 4: Software Block Diagram 24](#_Toc184814340)

[Figure 5: System Hardware Architecture -Terminal Connections 28](#_Toc184814341)

[Figure 6: Network Hardware Configuration 29](#_Toc184814342)

[Figure 7: System Architecture Overview 32](#_Toc184814343)

[Figure 8: Component Interaction Diagram 32](#_Toc184814344)

[Figure 9: Gantt Chart provides an overview of the complete project implementation 36](#_Toc184814345)

**List of Tables:**

[Table 1: Glossary 39](#_Toc184814525)

**Chapter 1: Introduction**

This document describes our Final Year Project report for the development of a *Hospital Management System (HMS) Using AI*. The proposed system is designed to address inefficiencies in current hospital operations by integrating functionalities such as patient appointment scheduling, real-time prioritization of admitted patients based on their vital data, and a centralized dashboard for administrative tasks. The HMS aims to provide a seamless, user-friendly experience for patients, doctors, and hospital administrators, ensuring timely care and operational efficiency.

The following sections of this report detail the problem definition, including the alignment of the project with Sustainable Development Goals (SDGs), particularly SDG 3: Good Health and Well-being. The document also includes a thorough analysis of the problem, the solution design comprising hardware and software aspects, and the deliverables of the project. Additionally, it discusses the expected functionality of the product, societal and environmental implications, and the estimated costs associated with the project.

**Chapter 2: Problem Definition (Client Requirements)**

**2.1 Problem Formulation**

The hospital faces challenges in **real-time patient prioritization** and **inefficient appointment scheduling**, which negatively affect patient care and hospital operations. Currently, critical patients are prioritized manually, and appointment booking requires direct communication with administrative staff, both of which are prone to delays and errors. The client seeks an **automated system** that not only prioritizes patients based on their health data but also integrates **chatbots for appointment booking** and **assistants for doctors** to streamline workflows.

Through discussions with the client, the following was clarified:

* **Data Integration**: The system should retrieve patient data from the existing database, without requiring additional sensors. Although data can be transferred to database sensors.
* **Machine Learning Models**: The client is open to using machine learning algorithms to predict patient deterioration but requests models that are **interpretable** and can be easily understood by medical staff.
* **Chatbots for Appointment Booking**: The system should incorporate **chatbots** that allow patients to book appointments independently, reducing the administrative burden.
* **Assistants for Doctors**: The system should also provide **virtual assistants for doctors** to prioritize patient care in real-time, based on predicted criticality.
* **Real-Time Alerts**: Critical patients need to be flagged, and alerts sent to healthcare providers promptly.

**Problem Definition**:

Hospitals face significant challenges in managing operations, particularly appointment scheduling and patient prioritization, which can lead to delays and compromised care. Studies reveal the following:

* **Manual Prioritization Delays:** A study found that in emergency departments (EDs), critically ill patients often experience delays of **6–12 hours** before being admitted to intensive care units (ICUs), with approximately **34.3% waiting beyond 12 hours**. These delays are linked to inefficiencies in communication, prioritization, and resource management​ [1].
* **Misprioritization Risks:** Mismanagement of triage procedures in EDs has been shown to result in increased mortality rates, particularly for patients requiring immediate care, highlighting the urgent need for automated prioritization tools​ [2].
* **Appointment Scheduling Challenges:** Overloaded administrative systems frequently cause delays in appointment bookings, extending wait times for non-critical patients by **2–3 days**, further straining hospital resources.

These inefficiencies underline the need for a **hospital management system** that integrates real-time prioritization, automated scheduling, and efficient data management to enhance patient care delivery.

**2.2 Mapping to Sustainable Development Goals (SDG)**



Figure 1: Sustainable Development Goals

Our project contributes to **SDG 3: Good Health and Well-being** by improving healthcare efficiency and accessibility. The **real-time patient prioritization system** uses machine learning to analyze vital signs, helping identify high-risk patients more quickly and potentially improving outcomes. The **chatbot for appointment scheduling** streamlines the process, making it easier for patients to access healthcare services and reducing administrative delays. Additionally, the **virtual assistants for doctors** assist in decision-making by providing real-time insights from patient data, improving care delivery. Overall, the system aims to enhance healthcare efficiency, optimize resource allocation, and support **universal health coverage**, which aligns with the goals of SDG 3.



Figure 2: Target Sustainable Development Goal: SDG 3

**2.3 Record of Meetings with Client**

**First Meeting (2nd September 2024):**

* **Purpose:**
  + To understand the client’s initial requirements and identify operational challenges in hospital management systems.
* **Discussion Points:**
  + The client highlighted issues with manual appointment scheduling and delays in prioritizing critical patients.
  + Recommended conducting field surveys at hospitals to observe workflows and gather insights into inefficiencies.
* **Actions Taken:**
  + Conducted surveys at local hospitals, which revealed delays in critical care due to administrative bottlenecks and errors in manual prioritization.
  + These insights directly informed the Problem Definition section.
* **Decisions Made:**
  + Agreed to develop a web-based Hospital Management System (HMS) focused on solving these inefficiencies.
  + Decided to use chatbot automation and machine learning to address the client’s key pain points.

**Second Meeting (5th September 2024):**

* **Purpose:**
  + To refine the project scope and confirm technical requirements.
* **Discussion Points:**
  + Presented hospital survey findings, emphasizing the impact of delays in prioritization and appointment scheduling.
  + Reviewed the proposed system architecture and technology stack.
* **Decisions Made:**
  + Finalized the Problem Definition and technical stack, selecting React.js, Node.js, MySQL, and Python.
  + Established milestones for the project, starting with the design phase in December 2024.

**Additional Guidance:**

Subsequent guidance and validation were provided by the project advisor, who reviewed progress at regular intervals. The advisor’s input shaped decisions on system design, technical choices, and implementation strategies.

**2.4 Preliminary Product Specification**

The **hospital management system** is designed to address challenges in healthcare efficiency by providing features such as **real-time patient prioritization**, **chatbot-based appointment scheduling**, and **doctor assistants for decision-making**. Based on the client’s requirements and development environment, the following technical specifications have been outlined:

#### 1. System Overview:

* **Platform**: The system will be developed as a **web-based application** with a **client-server architecture**. The **frontend** will use **React.js**, while the **backend** will be built with **Node.js**. The **database** will be managed with **MySQL**, storing and retrieving critical patient data.
* **Machine Learning**: The backend will integrate machine learning models to analyze patient data from the database, prioritizing cases based on criticality.

#### 2. Development Environment:

* **Operating Systems**: The development environment will primarily use **Windows** and **macOS** for coding, testing, and deployment.
* **Development Tools**: The team will utilize tools such as **Visual Studio Code** and **MySQL Workbench** for efficient development and testing.

#### 3. Power Requirements:

* As the system is cloud-hosted, it does not require physical power resources for deployment. Server requirements are as follows:
  + **CPU**: 4 vCPUs or more
  + **RAM**: Minimum of 8 GB
  + **Storage**: 100 GB SSD (scalable as needed)
  + **Bandwidth**: Minimum 1 Gbps to support real-time processing.

#### 4. Communication and Data Flow

* **Database Integration**: Patient data, including demographics, historical records, and vital signs, will be fetched from a **MySQL database**.
* **Data Security**: Communication between the frontend, backend, and database will use **HTTPS** to ensure secure data transmission.
* **Real-Time Alerts**: Alerts for critical patient prioritization will be delivered via **email, SMS, or in-app notifications**.

#### 5. Software Requirements:

* **Frontend**: Developed in **React.js** for responsiveness and cross-platform accessibility.
* **Backend**: Built using **Node.js**, with APIs developed using **Express.js**.
* **Machine Learning**: Implemented using **Python** libraries like **TensorFlow** or **Scikit-learn**.
* **Authentication**: **Firebase Authentication** will be used for secure user login, supporting multi-factor authentication and role-based access control.

#### 6. User Interaction and Interface:

* **Patient Chatbot**: Provides a conversational interface for patients to book appointments, integrated with the hospital's schedule.

#### 7. Scalability:

* The system will be designed to scale easily, allowing the addition of more features and supporting an increasing number of patients as the hospital grows.

#### 8. Security Measures:

* **Data Encryption**: All sensitive data will be encrypted using **AES-256 encryption**.
* **Role-Based Access**: Ensures that only authorized personnel can access sensitive patient data.

**2.5 Expected Functionality of Product**

The **hospital management system (HMS)** is designed to operate seamlessly in a real-life hospital environment, addressing critical challenges such as patient prioritization, appointment scheduling, and efficient decision-making for healthcare providers. The system’s functionality can be summarized as follows:

1. **Real-Time Patient Prioritization**
   * The system will analyze patients' vital data retrieved from the hospital's database.
   * Using machine learning algorithms, it will assess risk levels and generate a dynamic **priority list** of admitted patients.
   * Doctors will receive real-time notifications through a **dashboard interface**, enabling quick responses to critical cases and efficient resource allocation.
2. **Chatbot for Appointment Scheduling**
   * Patients will interact with a **chatbot interface** to schedule appointments based on doctor availability.
   * The chatbot will provide personalized recommendations and resolve scheduling conflicts in real time.
   * This feature will significantly reduce the workload on administrative staff and improve patient satisfaction.
3. **Doctor Assistant Features**
   * Doctors will access a **comprehensive dashboard** displaying prioritized patient data, historical medical records, and alerts for critical cases.
   * This assistant will serve as a decision-support tool, improving diagnostic efficiency and patient outcomes.
4. **System Workflow**
   * **Data Flow**: The system fetches patient information from the **MySQL database** and uses secure communication protocols (e.g., HTTPS).
   * **Alerts and Notifications**: The HMS will send critical updates to doctors via SMS, email, or push notifications, ensuring timely intervention.
   * **User Authentication**: Through **Firebase Authentication**, user access will be securely managed, allowing patients, doctors, and administrators to interact with the system based on their roles.
5. **Scalability and Adaptability**
   * The HMS is designed to operate efficiently in hospitals of various sizes, from small clinics to large healthcare facilities.
   * Its modular design allows for the addition of new features, such as integration with IoT devices for real-time monitoring, as the hospital's infrastructure evolves.
6. **Deployment**
   * The system will be hosted on cloud servers, ensuring accessibility from any web-enabled device.
   * Doctors and administrative staff will primarily access the system via desktops or laptops, while patients can use mobile devices for chatbot interaction and appointment scheduling.

**Chapter 3: Problem Analysis**

**3.1 Engineering Problem Model**

The engineering problem we are addressing involves the development of a **web-based hospital management system (HMS)** to bridge the gaps found in existing systems. These gaps include a lack of comprehensive functionality, role-specific access control, and insufficient integration of modern technologies to improve hospital operations. Our design focuses on creating an efficient and secure system that caters to the needs of administrators, doctors, and patients.

The **Role-Based Access Control (RBAC)** principle is central to the system's design. RBAC ensures that access to data and functionalities is strictly governed by the user’s role. This model is particularly appropriate for hospital management, where sensitive medical and administrative data must be protected. For example, an **administrator** can manage hospital records, while a **doctor** can access patient information relevant to their own appointments. Patients, on the other hand, can only interact with their own data and appointment information. By restricting access, RBAC ensures security and promotes operational efficiency. This principle is widely used in healthcare systems to maintain confidentiality and prevent unauthorized data access.

In addition to RBAC, the design adheres to **user interface (UI) design principles** that prioritize simplicity and intuitiveness. The goal is to ensure that the system is accessible to users with varying levels of technical expertise. The homepage of the system introduces users to the system’s features and directs them to the appropriate login page, depending on whether they are administrators or doctors.

After logging in, each user is presented with a tailored dashboard that aligns with their role, enhancing their experience and minimizing confusion [3].

Another key aspect is **data management and security**. The system relies on **MySQL** for structured data storage and **Firebase Authentication** for user verification. These technologies ensure that all hospital-related data is stored securely and is easily accessible to authorized users. Firebase Authentication provides an additional layer of security, preventing unauthorized access while also enabling features like multi-device synchronization. The decision to use Firebase is also driven by its scalability, which is crucial as the system is designed to grow with the hospital’s needs.

The **functional breakdown** of the system focuses on role-specific portals. The **administrator portal** allows for the management of hospital records, including doctor schedules and patient appointments. It is the control center for hospital operations. The **doctor portal** displays relevant patient data, including medical history and upcoming appointments, and allows doctors to manage their time effectively. The **patient portal** enables patients to schedule appointments, view their health records, and check doctor availability [4].

The final challenge addressed by the system is **real-time data integration**. In-house patient data, including medical updates and admission records, is entered by attendants and made immediately accessible to doctors. This real-time data flow ensures that the healthcare team has the most up-to-date information when making decisions, ultimately improving patient care.

In summary, the engineering problem model integrates key principles like access control, user interface design, data security, and real-time information flow into the web-based HMS. These components collectively aim to enhance security, usability, and hospital efficiency, ultimately improving healthcare delivery.

**3.2 Recent Similar Projects**

We explored several projects related to our system. One such project is the Doctor Inn app, which allows patients to book appointments with doctors based on specialization and available time slots [4]. However, it is app-based, while our system is web-based, and we aim to add functionalities like real-time patient prioritization and administrative control.

Another relevant project is a medical billing app, which allows patients to view their bills and book appointments [5]. Unlike our system, it is limited to patients and lacks advanced features such as real-time doctor-patient interactions and comprehensive hospital management.

Additionally, some offline HMS solutions provide broad access to users, including doctors and administrators, but they lack security measures like role-based access control (RBAC) [6]. Our system will address this by implementing RBAC to ensure that users have restricted access according to their role, enhancing security and functionality.

These examples illustrate the gaps in existing solutions, emphasizing the need for a more comprehensive, secure, and web-based system like ours.

**3.3 Distinguishing Features of this Project**

Our hospital management system stands out from existing solutions by offering a more comprehensive, multi-functional approach. While many systems focus on basic patient interactions, our project integrates advanced features that address the needs of multiple user roles and incorporate cutting-edge technologies.

Key distinguishing features include:

1. **Multiple User Roles and Access Levels**: Our system supports distinct logins for the **Administrator**, **Doctor**, **Patient**, **Attendant** (for in-house patient data entry), and **Finance Department**. Each role has tailored access, ensuring that sensitive data is protected and only accessible to the appropriate users.
2. **Chatbots for Appointment Scheduling and Assistance**: We incorporate AI-powered chatbots to help **patients** schedule appointments efficiently and **doctors** to prioritize care based on real-time data. The chatbot simplifies interactions, making the system more user-friendly.
3. **Machine Learning for Critical Patient Prioritization**: Machine learning algorithms are integrated into the system to analyze **vital data** and prioritize patients based on their condition. This feature enhances real-time decision-making for doctors, helping them focus on the most critical cases.
4. **Real-time Information Flow**: The system ensures that all data flows seamlessly between departments, improving communication and response time.

By combining these advanced features with a secure, role-based access control system, our solution addresses the deficiencies of existing systems, offering a more robust, user-centric, and intelligent hospital management platform.

**3.4 Societal and Environmental Implications of the Project**

The web-based hospital management system improves healthcare delivery by enhancing access to medical records, streamlining appointment scheduling, and prioritizing critical patients using machine learning. This promotes **equitable access to healthcare** and supports **better decision-making** for doctors, improving patient outcomes.

In terms of **health and safety**, the system ensures data security, complying with relevant privacy laws like GDPR or HIPAA. The inclusion of chatbots aids **patient accessibility**, particularly for those who may struggle with traditional methods.

**Culturally**, the system can be adapted to different regions and languages, though we must address potential issues related to the **digital divide**. Environmentally, by reducing paper usage, the system contributes to **sustainability**, aligning with efforts to reduce hospital carbon footprints.

While the project promises significant benefits, ensuring **data security**, **digital inclusion**, and **environmental sustainability** is essential. Adjustments may be required for regions with limited technology access.

**Chapter 4: Solution Design**

**4.1 Design Requirements and Constraints**

The design of the web-based hospital management system (HMS) must meet specific functional and technical requirements while adhering to constraints.

#### Design Requirements:

* **User Authentication and Security**: Utilize Firebase Authentication for secure login with role-based access (administrator, doctor, patient, attendant).
* **User Interface**: A responsive frontend using React.js, accessible across devices (desktop, tablet, mobile).
* **Real-Time Data Integration**: Incorporate machine learning for patient prioritization and real-time decision-making.
* **Appointment Scheduling**: Implement chatbot functionality for patients to schedule appointments and for doctors to manage their schedules.
* **Scalability**: The system must scale efficiently with increased usage, supporting both small and large hospitals.

#### Design Constraints:

* **Time and Budget**: Project must be completed within the academic timeline and budget.
* **Legacy System Integration**: The system should integrate smoothly with existing hospital management tools.
* **Regulatory Compliance**: Ensure adherence to healthcare data protection regulations, like HIPAA and GDPR.

These requirements ensure that the system is functional, scalable, and compliant while addressing the key needs of the hospital environment.

* 1. **Preliminary Design**

**Baseline Solution:** The proposed system will be a **web-based hospital management system (HMS)** that supports real-time patient prioritization and appointment scheduling. It will incorporate **role-based access control** for administrators, doctors, patients, attendants, and the finance department. Machine learning algorithms will be used for critical patient prioritization.

### High-Level Design Description:

The system will consist of the following primary components:

1. **Frontend**: A web interface that allows various users to interact with the system. The frontend will be responsive, designed using **React.js**, and will have multiple pages for the login and management interfaces (admin, doctor, patient, attendant).
2. **Backend**: A server-side application built with **Node.js** to handle business logic, user management, authentication, data processing, and interactions with the database.
3. **Database**: A MySQL-based system that will store user data, appointment details, patient records, and historical information. This will be essential for real-time decision-making and generating reports.
4. **Machine Learning Model**: A part of the backend that uses machine learning to analyze patient vital data for critical patient prioritization in real-time.
5. **Firebase Authentication**: Ensures secure and reliable user authentication for different roles, including two-factor authentication for doctors.

### Design Features:

* **Reliability**: The system will include real-time updates, secure user authentication, and data backups to ensure that hospital operations can continue uninterrupted.
* **Maintenance**: It will be designed to be easy to maintain with modular components that can be updated or replaced independently, ensuring scalability and flexibility.
* **Testing**: Unit tests, integration tests, and user acceptance tests will be incorporated to ensure that all modules meet performance and security requirements.
  + 1. **Hardware Block Diagram**

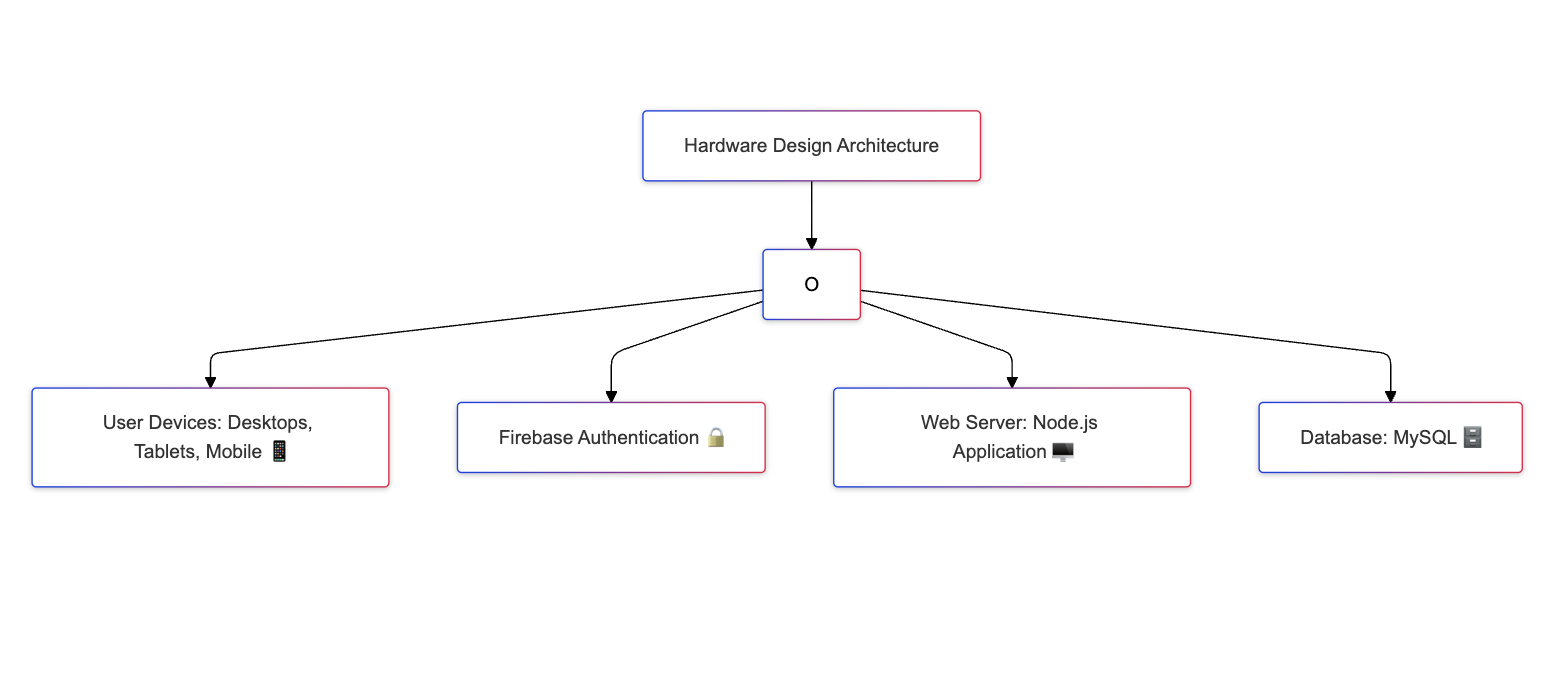


Figure 3: Hardware Block Diagram

* + 1. **Software Block Diagram**

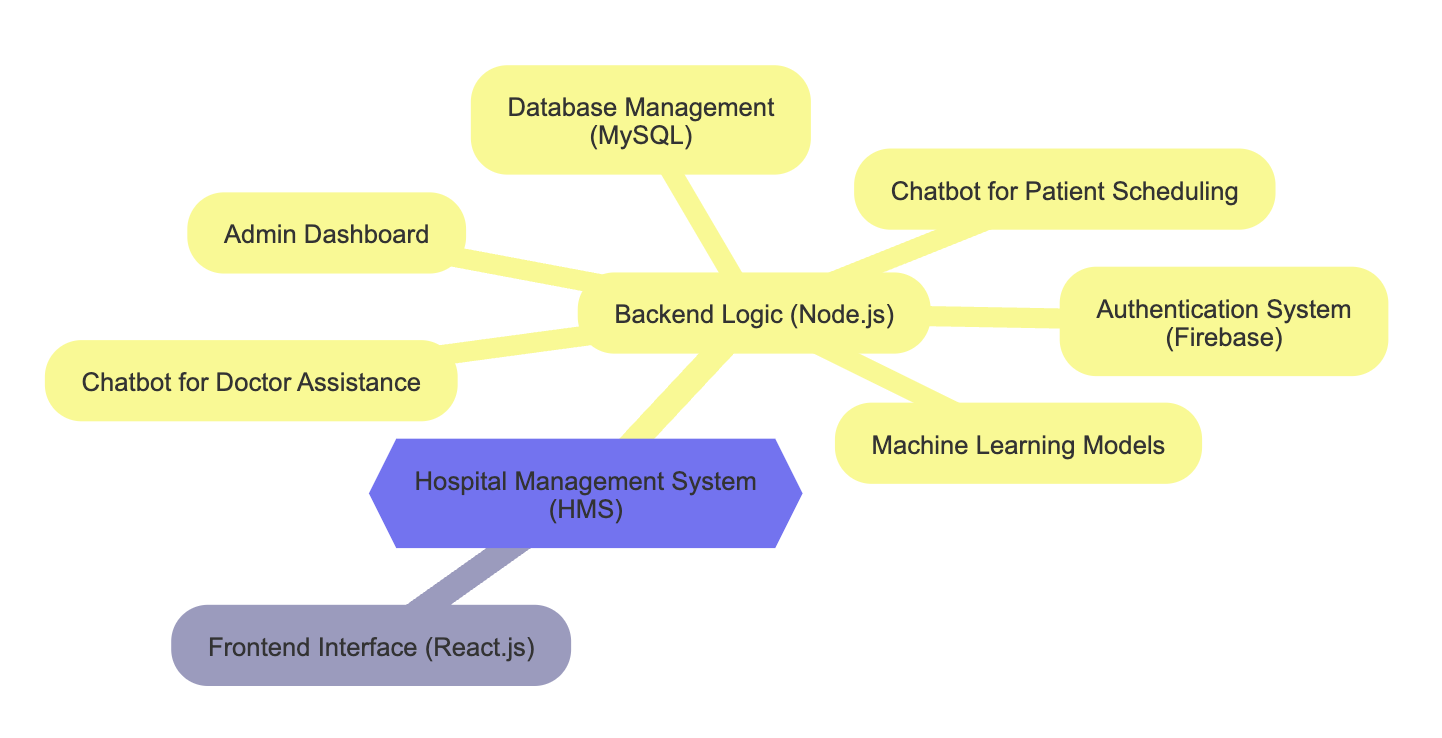
****

Figure 4: Software Block Diagram

* 1. **Detailed Hardware and Software Design**

This section presents the detailed design for the hardware and software components of the HMS, addressing the project's core functionality and performance requirements.

#### Hardware Design:

1. **Server Infrastructure**:
   * Cloud hosting (e.g., AWS) with scalable storage, 4 CPUs, and 8 GB RAM for primary and backup servers.
2. **Network Hardware**:
   * Router, firewall, and load balancer for secure, reliable data flow.
3. **Development Platforms**:
   * Windows and macOS for software development and testing.
4. **Power Backup**:
   * UPS system ensures uninterrupted service.

#### Software Design:

1. **Frontend**:
   * React.js provides responsive user interfaces for admin, doctor, patient, and attendant portals.
2. **Backend**:
   * Node.js enables real-time data processing and efficient API handling.
3. **Database**:
   * MySQL stores structured hospital records with secure role-based access.
4. **Authentication**:
   * Firebase Authentication supports multi-role secure logins.
5. **AI Features**:
   * Chatbots for patient appointment booking and doctor assistance.
   * Machine learning models prioritize patients based on real-time vital data analysis.
6. **Testing**:
   * Tools like Postman and MySQL Workbench validate system performance.

#### Design Alternatives:

* React.js over Angular.js for better performance.
* Node.js over Django for real-time processing.
* MySQL over MongoDB for structured data requirements.

This concise design balances efficiency, scalability, and security, aligning with the project's goals.

**4.3.1 Calculations**

For the web-based hospital management system, the key calculations focus on server load and query performance. These include:

1. **Database Load:** Estimate the storage requirements based on expected patient and doctor data. For example, with 1000 patients, the system may require 1MB for patient data.
2. **Authentication Load:** Ensure Firebase Authentication can handle the anticipated number of concurrent logins, especially during peak hours.
3. **Query Optimization:** Use MySQL query performance tools (e.g., EXPLAIN) to optimize queries for fast data retrieval.
4. **ML Algorithm Response Time:** Calculate the time needed for machine learning algorithms to process and prioritize patient data, ensuring real-time responsiveness.

These calculations help ensure the system's efficiency and scalability.

**4.3.2 Hardware Design**

Here’s a brief breakdown:

1. Server and Networking:
   * Cloud Hosting: Use cloud services like AWS or Azure to host the backend and database, ensuring sufficient server resources and scalability.
   * Networking: Include necessary hardware for smooth data flow, such as routers, switches, and load balancers.
2. Security Components:
   * Authentication Devices: Secure login mechanisms might require devices like fingerprint scanners or RFID for physical access control.
   * Firewall and Encryption: To protect sensitive hospital data, include hardware solutions like firewalls and encryption hardware for secure communication.
3. Backup and Redundancy:
   * Data Backup Systems: Implement RAID or other redundant storage solutions to ensure data availability and prevent loss.

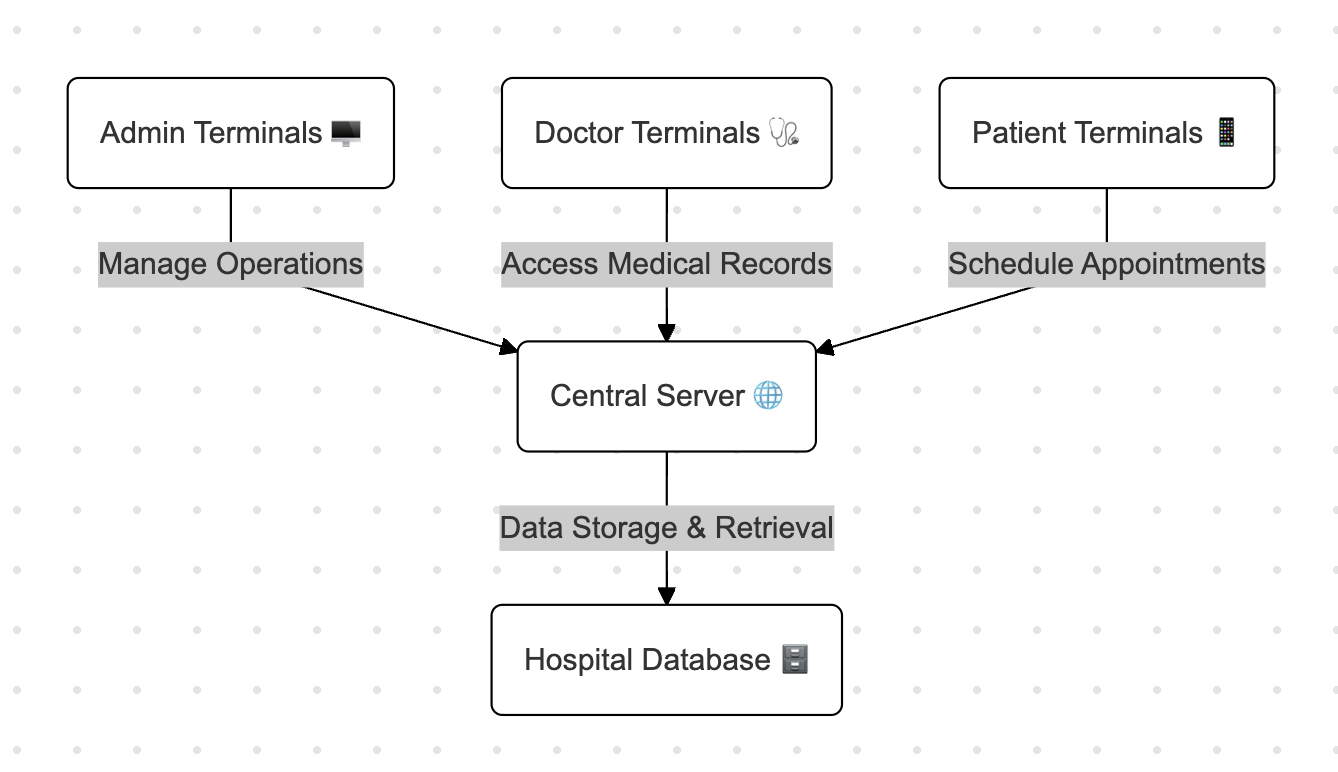
****

Figure 5: System Hardware Architecture -Terminal Connections

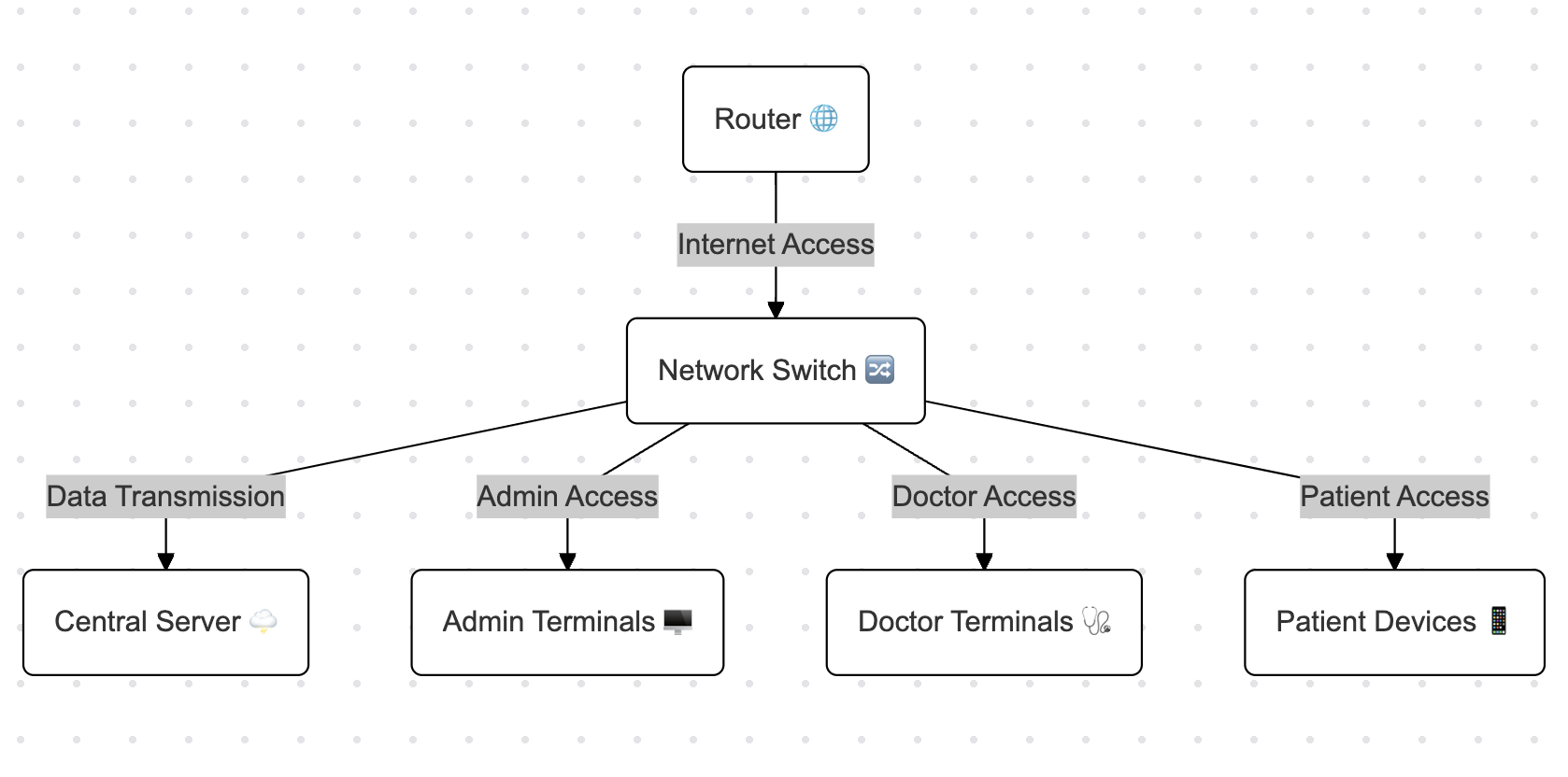


Figure 6: Network Hardware Configuration

**4.3.3 Software Design**

The Hospital Management System (HMS) leverages a streamlined software design, integrating modern technologies for optimal performance, scalability, and security. Each component is carefully chosen to address specific project requirements.

**Key Software Components**

* **Backend (Node.js):**
  + Manages user authentication through **Firebase Authentication**, ensuring secure access for admins, doctors, and patients.
  + Implements **RESTful APIs** for smooth communication between the client and server.
  + Integrates **machine learning models** to prioritize patients based on real-time data and predict critical conditions.
  + **Justification:** Node.js’s non-blocking I/O and asynchronous capabilities make it ideal for real-time healthcare systems requiring rapid data updates. Compared to Django, Node.js offers better performance for event-driven applications.
* **Frontend (React.js):**
  + Provides a responsive and user-friendly interface for all user roles (admin, doctor, patient).
  + Features dynamic forms for appointment scheduling and doctor availability updates.
  + Includes chatbot functionality:
    - **Patient-facing chatbot** for appointment booking.
    - **Doctor-facing chatbot** to assist with patient prioritization.
  + **Justification:** React.js’s component-based architecture ensures scalability and reusability, making it suitable for complex interfaces. Compared to Angular.js, React.js offers lighter, faster-loading components, improving user experience.
* **Database (MySQL):**
  + Stores hospital records, including patient data, appointment schedules, and billing details, ensuring secure and structured data management.
  + Supports **role-based access control (RBAC)** to restrict data access based on user roles.
  + **Real-time syncing** ensures up-to-date information for users.
  + **Cost Advantage:** MySQL’s Community Edition is free and open-source, making it a cost-effective choice for development and deployment.
  + **Justification:** MySQL’s structured query capabilities and ACID compliance make it ideal for relational data management, outperforming MongoDB for projects requiring consistent schemas.
* **Machine Learning (Python):**
  + Processes patient data using predictive algorithms to prioritize critical cases.
  + Utilizes libraries like **Scikit-learn** and **Pandas** for efficient data analysis.
  + **Justification:** Python’s extensive library ecosystem and versatility make it the preferred choice for developing machine learning models.
* **Security and Data Integrity:**
  + Ensures data encryption and secure storage to maintain privacy.
  + Implements **role-based access control (RBAC)** to restrict system functionalities based on user roles.
* **Real-time Features:**
  + Enables real-time patient prioritization, alerting doctors immediately about critical cases.
  + Automates appointment scheduling via chatbots, reducing staff workload and enhancing user experience.

**Design Philosophy:**

The HMS is designed to be scalable and extensible, allowing for future additions such as telemedicine and patient portals. Comprehensive unit and integration tests ensure smooth interaction between all components.

**Diagrams:**

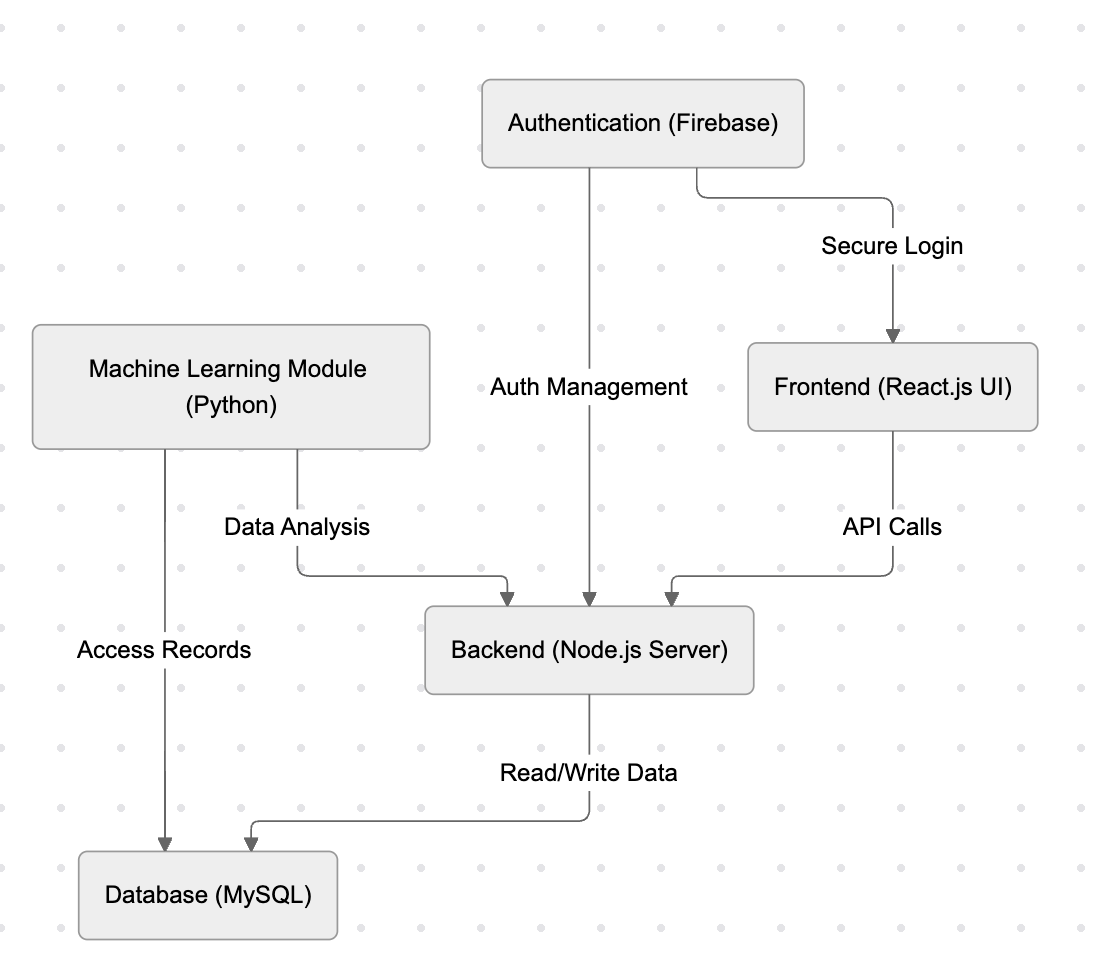


Figure 7: System Architecture Overview

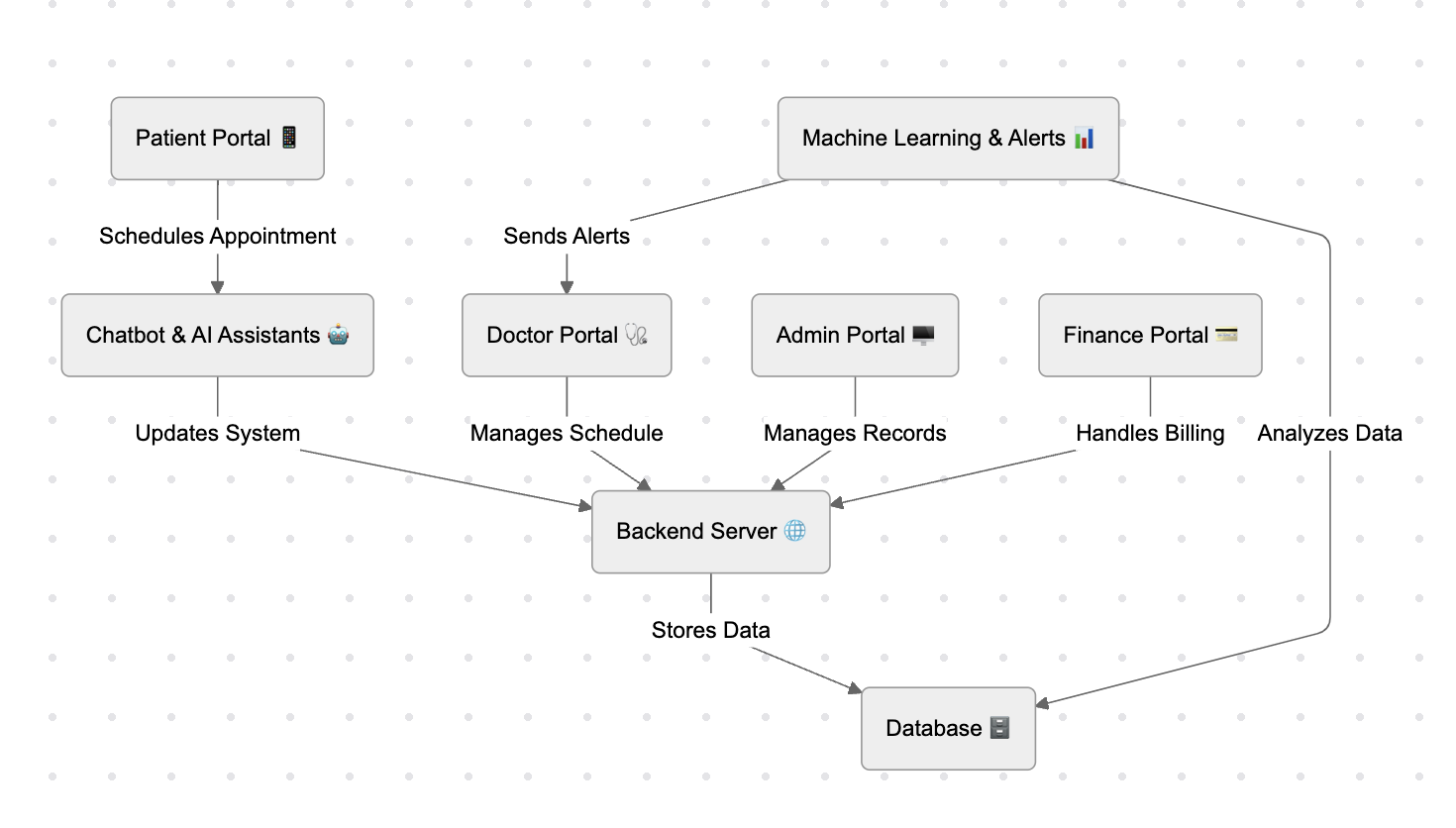


Figure 8: Component Interaction Diagram

**Chapter 5: Deliverables and Cost**

**5.1 Deliverables**

The project will deliver the following key components:

* **Software Deliverables:**
  + A fully functional web-based Hospital Management System (HMS).
  + Frontend (React.js) for user interfaces (admin, doctor, patient).
  + Backend (Node.js) with secure user authentication and real-time functionalities.
  + Machine learning integration for real-time patient prioritization.
  + Database (MySQL) for secure record storage and access control.
* **Documentation Deliverables:**
  + Technical documentation detailing the system architecture, design, and functionality.
  + User manuals for the different roles (admin, doctor, patient).

**5.2 Project Plan**

Planning our project implementation we have categorized it into these 4 main categories.

**1. Work Breakdown Structure (WBS):**

**Main Tasks:**

* **Project initiation:** Defining scope and objectives.
* **System design:** Planning frontend, backend, database, and AI integration.
* **Development:** Implement features like authentication, scheduling, prioritization, and chatbot.
* **Testing:** Perform unit, integration, and user acceptance tests.
* **Deployment:** Set up the live system and conduct final testing.

### 2. Time Management Plan:

* **System Design:** 2 weeks.
* **Frontend Development:** 3 weeks.
* **Backend Development:** 4 weeks.
* **Machine Learning Integration:** 2 weeks.
* **Testing & Deployment:** 2 weeks.
* **Buffer Time:** 1 week for revisions.

### 3. Human Resource Management Plan:

**Roles and Responsibilities:**

* **Project Manager:** Oversee execution and team coordination.
* **Frontend Developer:** Work on React.js interface and responsiveness.
* **Backend Developer:** Manage Node.js and database integration.
* **ML Specialist (if applicable):** Develop ML models for patient prioritization.
* **Tester:** Ensure functionality through comprehensive testing.

**Team Distribution:**As this is a group of three students, responsibilities will overlap to leverage individual strengths. For instance:

* **Muhammad Rafey Umar** could act as both Frontend and Backend Developer.
* **Abdullah Ahmad** could focus on backend Development and Testing.
* **Ali Khurram** could contribute to ML integration, Testing, and assist in other areas as needed.

This ensures all roles are covered, while promoting collaboration and shared ownership of the project.

### 4. Gantt Chart:

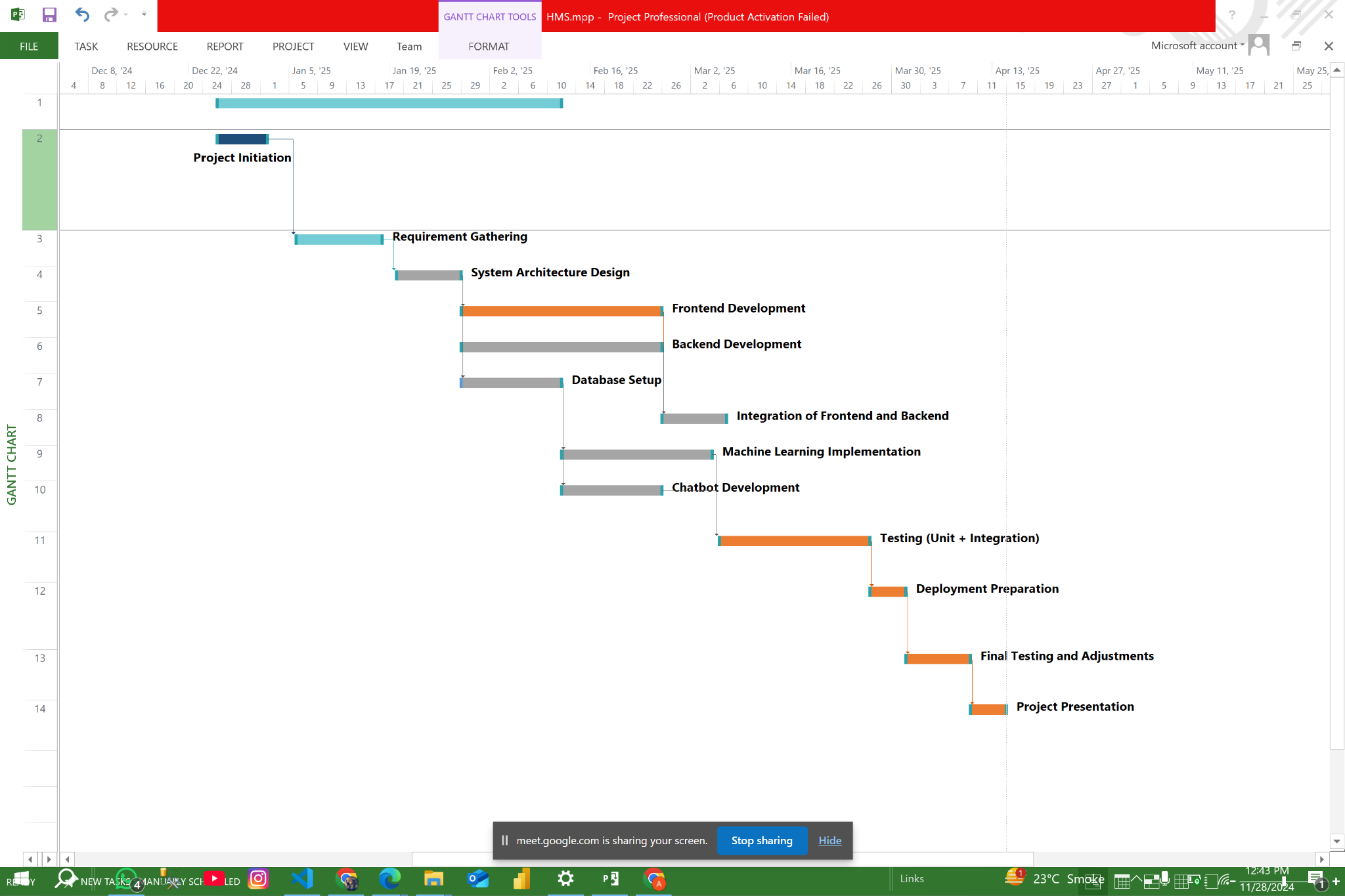
****

Figure 9: Gantt Chart provides an overview of the complete project implementation

**5.3 Estimated Cost**

This Hospital Management System requires a computer or laptop with the following minimum specifications:

* Processor: Core i5, 2nd generation
* RAM: 8 GB
* A reliable internet connection

The cost for these items is approximately 40k [7]. If the domain and website hosting are included, an additional $30 per month will be required.

**5.4 Items to Purchase**

To run the website efficiently, the following items will be required:

1. **Computer Hardware**:
   * **Intel Core i5 (2nd Gen)**, **8GB RAM**, **500GB SSD** for optimal performance.
2. **Web Hosting**:
   * **Cost**: $15 (annual/monthly) [8] for reliable server space and uptime.
3. **Domain Name**:
   * **Cost**: $30 (annual) for a unique web address.
4. **SSL Certificate**:
   * **Cost**: $15 for secure data encryption and trust.
5. **CMS/Framework**:
   * **Cost**: Free to $50 depending on platform choice.

**References**

|  |  |
| --- | --- |
| [1] | BMC Emergency Medicine, "Factors contributing to delay intensive care unit admission of critically ill patients," 2020. [Online]. Available: bmcemergmed.biomedcentral.com. |
| [2] | JAMA Network, "Reducing avoidable delays in emergency care," 2022. [Online]. Available: jamanetwork.com. |
| [3] | K. Selvi, "A Study of Advanced Hospital Management System.," IOSR Journal of Dental and Medical Sciences, 2 2017. [Online]. Available: doi.org/10.9790/0853-160203127134. |
| [4] | S. S. A. A. S. R. A. R. Mina Parveen, "HOSPITAL MANAGEMENT SYSTEM," April 2023 . [Online]. Available: https://www.irjmets.com/uploadedfiles/paper//issue\_4\_april\_2023/37037/final/fin\_irjmets1682505166.pdf. |
| [5] | I. A. J. .. I. S.F, "Designing A Web Based Hospital Management System For MOUAU Clinic," 2015. [Online]. Available: https://www.ijtrd.com/papers/IJTRD1361.pdf.. |
| [6] | S. G. &. V. Kumar, "THE HOSPITAL MANAGEMENT INFORMATION SYSTEMS (HMIS): A JOURNEY TO PAPERLESS HOSPITALS," 4 4 2020. [Online]. |
| [7] | "select.pk," laptops-computers, 6 8 2024. [Online]. Available: https://select.pk/apple-mac-mini-a1347-core-i5-4th-generation-late-2014.html.. |
| [8] | A. Stuopelis, "Hostinger," 2004. [Online]. Available: A. Stuopelis, "Hostinger," 2004. [Online]. Available: https://www.hostinger.pk/?utm\_medium=ppc&utm\_campaign=Brand-Exact|NT:Se|LO:PK&gad\_source=1&gclid=Cj0KCQiAo5u6BhDJARIsAAVoDWuUfRK9RQOz9lJR5ERFQPKJ8e0aLDIEdHW8wFutSY7N4rGp64js5HgaApy-EALw\_wcB. . |

**Glossary:**

|  |  |
| --- | --- |
| HMS | Hospital Management System |
| SDGs | Sustainable Development Goals |
| HTTPS | Hyper-Text Transfer Protocol Secure |
| APIs | Application Programming Interfaces |
| AES-256 | Advanced Encryption Standard - 256-bit |
| RBAC | Role-Based Access Control |
| GDPR | General Data Protection Regulation |
| HIPAA | Health Insurance Portability and Accountability Act |
| AWS | Amazon Web Services |
| WBS | Work Breakdown Structure |

Table 1: Glossary