**CLASS: BCS 3106 SOFWARE ENGINEERING**

**TITTLE: HEALTH CENTER MANAGEMENT SYSTEM**

**GROUP F**

Jamal Yakub –BSCNRB186024

Joy Winslet – BSCNRB 101924

Lidian Kitiyo – BSCNRB191424

Lebu Ayiga – BSCNRB734421

**SUBMITTED TO: CECILIA ANGELA NAFUKA**

**DATE: 25th OCTOBER 2024**

**INTRODUCTION**

Healthcare facilities, particularly local health centers, play an important role in delivering medical treatment to communities. The growing number of registered patients, together with the variety of healthcare services provided, necessitates efficient and streamlined management systems. This project focuses on establishing a comprehensive Health Center Management System to serve approximately 3000 registered patients at a local health center.

The proposed system will make it easier to register patients, schedule appointments, cancel them, and handle prescriptions. It will also have specialized features for emergency situations, as well as services in High Dependency Units (HDU) and maternity wards. The technology intends to improve the overall workflow of the health facility by providing seamless interaction among patients, receptionists, and doctors.

Using Python, this project will provide an intuitive and scalable solution that can be utilized by both health center personnel and patients, improving efficiency and service delivery at the center.

**PROJECT PROPOSAL**

In a healthcare setting, a management system that efficiently handles a wide range of medical conditions, from routine consultations to critical care, is required. This proposal describes how the Health Center Management System will handle two critical use cases: emergency surgery and specialized treatments including high dependency units (HDUs) and maternity services.

**Use Cases**

1. **Emergency Surgery**

* In an emergency medical issue, the standard procedure of scheduling an appointment and waiting for availability is not viable. The technology will have a fast-track option for emergency circumstances, allowing healthcare staff to bypass regular protocols.

. **Process** - When a patient presents with an emergency, the receptionist or attending personnel will record the case as urgent in the system. This will bypass the normal booking system and automatically assign the next available doctor or specialist to the patient without delay.

. **Notification** - Doctors and other necessary workers will receive timely notifications concerning emergency cases, ensuring that the patient receives prompt care.

. **Outcome** - The system's emergency module guarantees that vital surgeries or urgent treatments are completed quickly, decreasing time spent on administrative procedures.

1. **Specialized Treatments (HDU, Maternity Services)**

* Patients who require specialist treatments, such as those requiring critical care in a High Dependency Unit (HDU) or those undergoing maternity services, will benefit from personalized features that will allow them to be registered and managed easily. These therapies require collaboration among multiple healthcare personnel and resources, and the system will guarantee that patients are sent to the proper departments.

. **HDU** - When a patient requires care in the HDU, the system flags their record and assigns them to a specific ward staffed by HDU-trained doctors and nurses. This guarantees that the patient is constantly watched and given specialized care. . **Maternity Services** - Pregnant patients who require maternity services will be registered in the system, and their records will be flagged for specialist treatment. The technology will automatically allocate maternity doctors and nurses, ensuring that all required equipment and resources are available throughout labor and delivery.

**REQUIREMENT ANALYSIS**

The data flow diagrams (DFDs) and use case diagrams given serve as the foundation for the requirements analysis that follows. The system under development is a Healthcare administration System that will simplify patient contacts, registration, appointment scheduling, and medical record administration.

**Functional Requirements**

1. **Patient Registration**

* Patients must be able to register through the healthcare system, either manually by the receptionist or via a self-service portal.
* The system should store the patient's details (name, ID, contact information) in the **Patient Records**.
* The system should allow **de-registration** when patients leave or no longer require the center’s services.

1. **Appointment Booking**

* Patients can request appointments, which the system will check for availability.
* The system allows patients to **cancel** appointments if needed.
* Appointments will be handled by the receptionist, who manages the scheduling process based on doctor availability.

1. **Appointment Management**

* Doctors can check the appointments allocated to them through the system.
* The system should notify doctors of their schedules and allow them to update their availability.
* In the case of **emergency** appointments, the receptionist can fast-track the process to immediately allocate an available doctor.

1. **Medical Outcome and Records**

* After an appointment, the doctor will issue an **outcome** (diagnosis, prescription, or referral).
* This outcome will be recorded in the **Patient Records** module for future reference.
* The patient can view the outcome through the system or request it in person.

1. **Emergency Handling**

* The system must support a feature for emergency cases where appointments can be bypassed, and a patient is directly assigned to a doctor.
* This ensures a streamlined flow for urgent cases where immediate medical attention is required.

1. **Specialized Treatment (HDU, Maternity)**

* For patients requiring specialized care, such as in the **High Dependency Unit (HDU)** or maternity services, the system should automatically assign them to the relevant departments.
* The system should track the care provided in these specialized wards and allow doctors and nurses to access patient records specific to these treatments.

#### **Non-Functional Requirements**

1. **System Performance**

* The system should handle multiple users simultaneously (receptionists, doctors, patients).
* Appointment booking and checking should be completed within a few seconds to ensure efficiency.

1. **Security**

* Patient records should be securely stored and only accessible by authorized personnel (doctors, receptionists, and the patients themselves).
* Data encryption must be used to protect sensitive information such as medical records.

1. **Reliability and Availability**

* The system should be available 24/7, ensuring that patients can check appointments or outcomes even after working hours.
* Regular backups must be performed to prevent data loss.

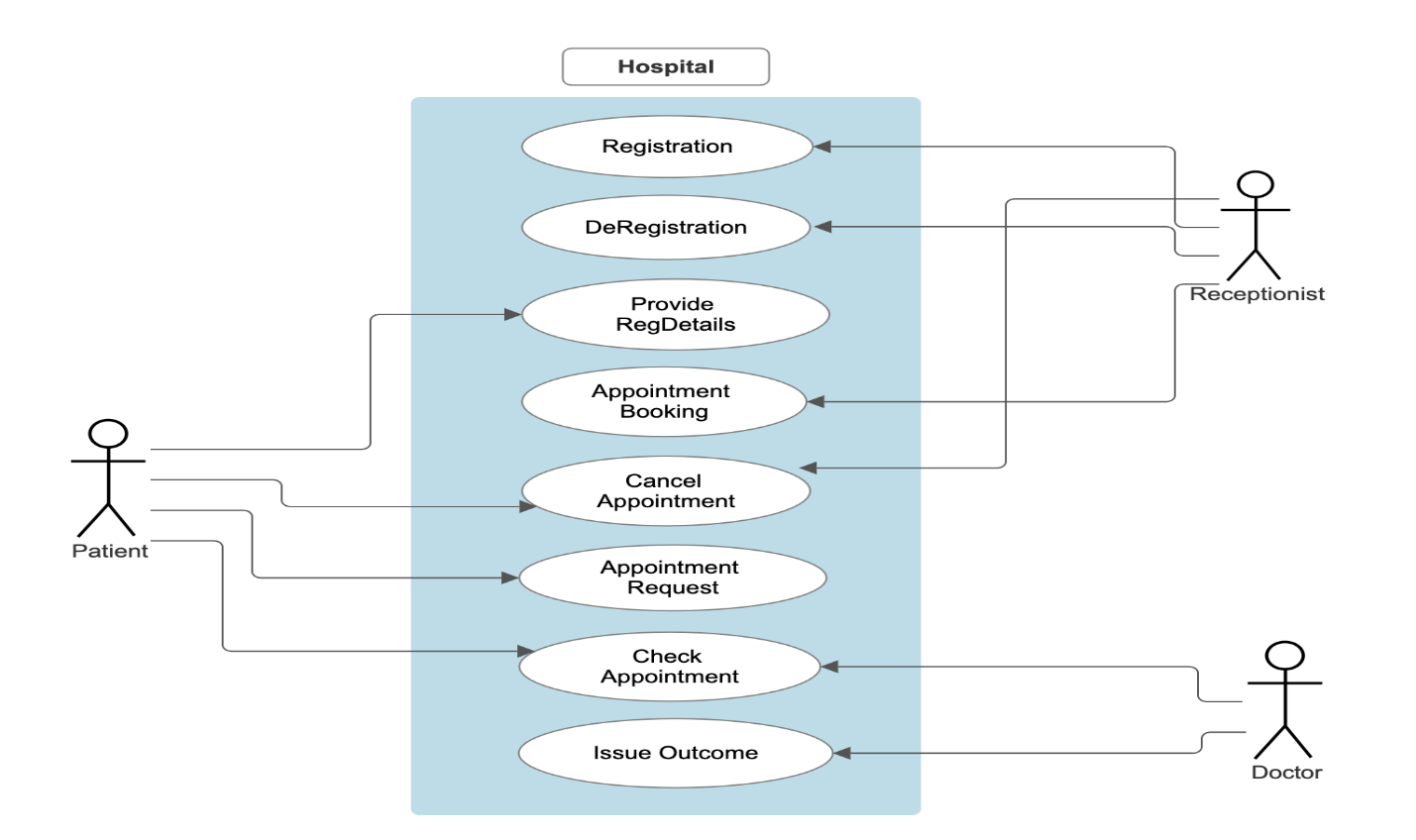
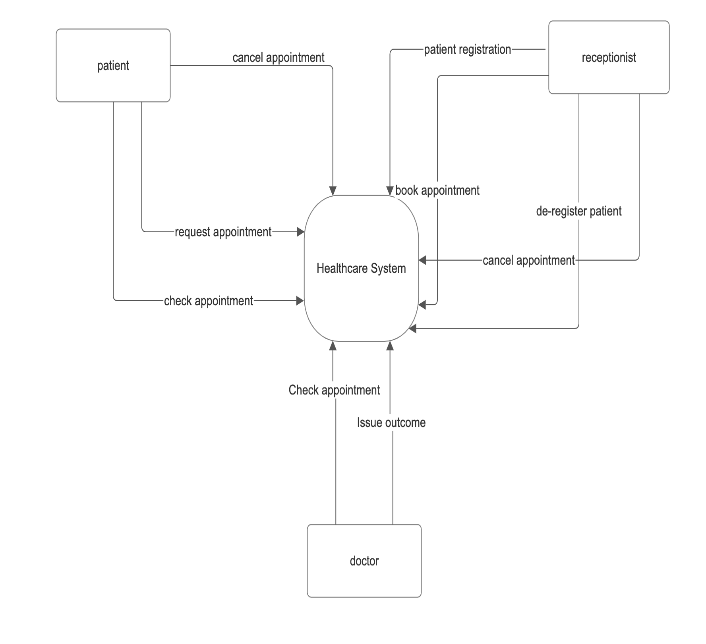
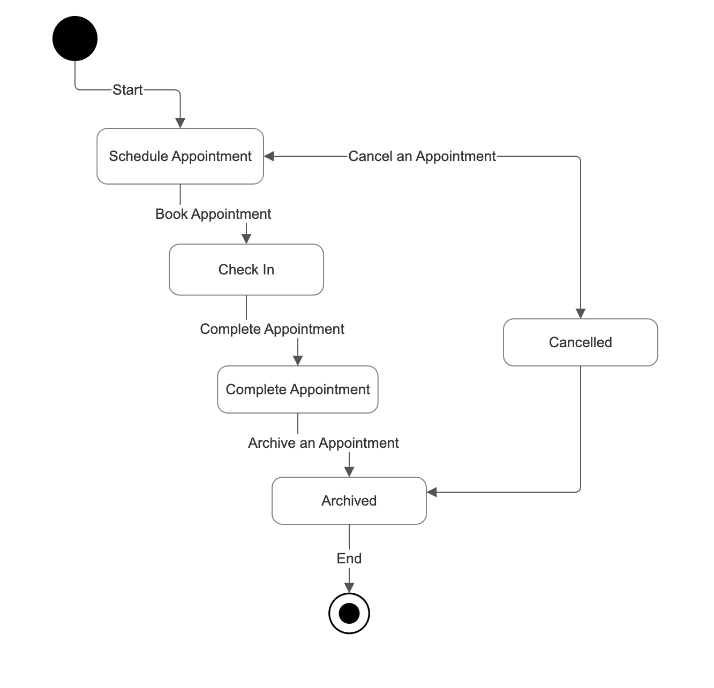
1. **Scalability**

* The system should be scalable to accommodate more patients, doctors, and services as the health center expands.
* It should allow easy integration with other medical systems or databases if needed.

1. **Usability**

* The interface should be intuitive and user-friendly for non-technical users, such as patients and receptionists.
* The system should also provide tutorials or help features to assist users in navigating the system.

#### **Process Flow (Based on the Diagrams)**



**Patient Interaction**:

* Patients can **request** appointments, **cancel** them, and view their medical records or outcomes.
* They interact primarily with the receptionist and doctors through the system.

**Receptionist Role**:

* The receptionist handles patient **registration**, **appointment booking**, and **cancellations**.

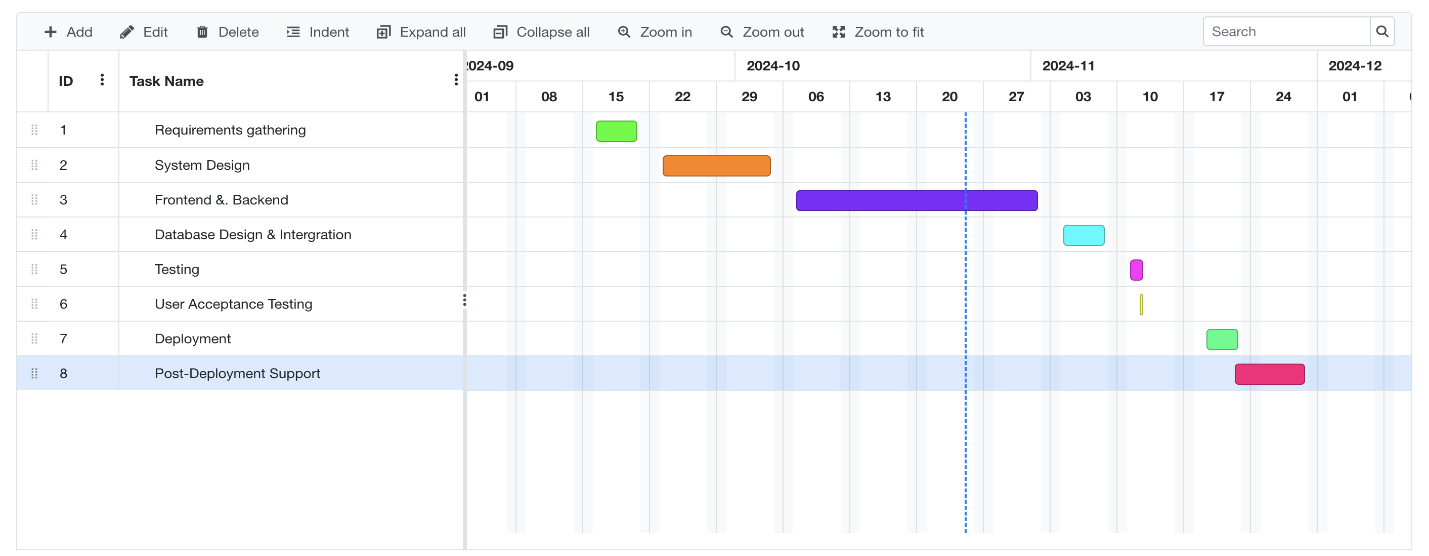
They play a central role in connecting patients with doctors through the system.

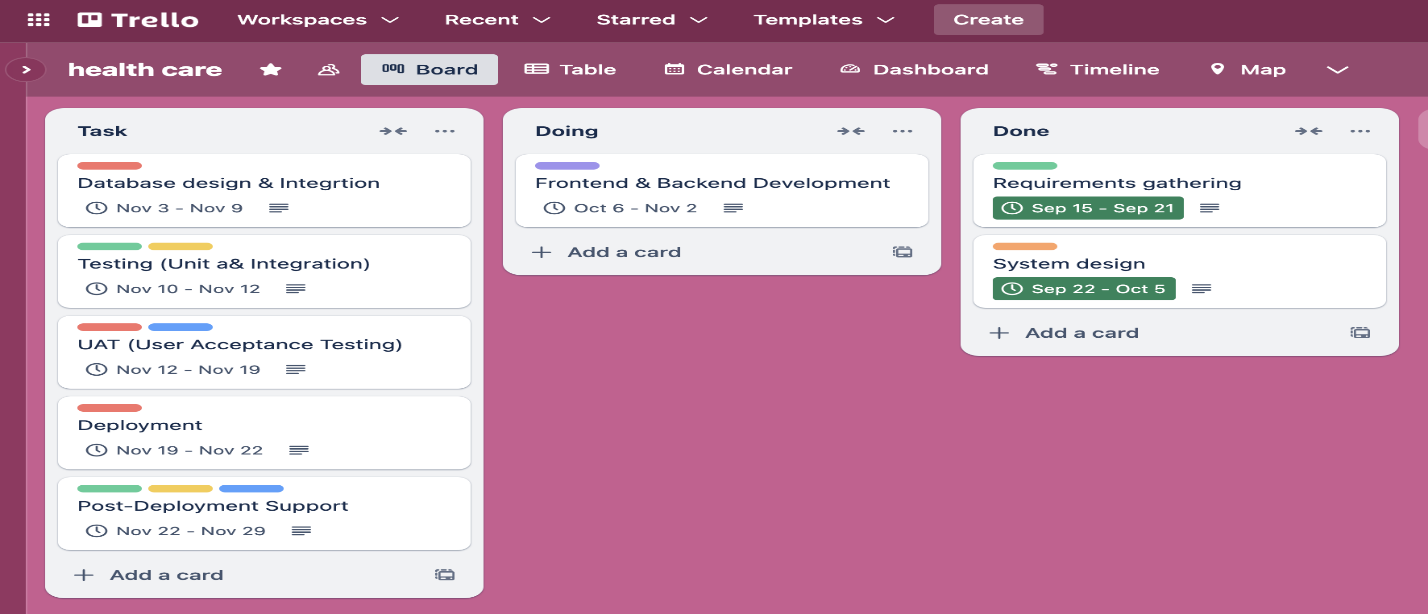
**Doctor Interaction**:

* Doctors use the system to **check appointments** and **issue medical outcomes**.
* Their availability and interaction with patients are managed through the system, ensuring smooth coordination with the receptionist and patients.

**PROJECT PLAN**

#### **Gantt Chart Design -** The Gantt chart will outline the timeline and tasks for each phase, ensuring the project stays on track.

**Activity Board**



#### **1. Objectives**

* **Streamline patient registration** and management.
* **Facilitate appointment scheduling, cancellations, and emergency handling**.
* Manage **medical outcomes and prescriptions** securely.
* Provide specialized modules for **HDU and maternity services**.
* Ensure system scalability, security, and usability.

#### **2. Scope**

* **Patient Registration**: For new and existing patients, both manual and self-service.
* **Appointment Management**: Scheduling, cancellation, emergency bypass.
* **Medical Record Management**: Patient outcomes, prescriptions, and history.
* **Emergency Care**: Fast-tracked appointments for urgent cases.
* **Specialized Services**: HDU and maternity wards management.
* **User Interfaces**: For receptionists, doctors, and patients.
* **Security and Compliance**: Data encryption, secure access, and backups.

#### **3. Key Deliverables**

* **Patient Registration Module**.
* **Appointment Scheduling System**.
* **Doctor and Receptionist Dashboards**.
* **Emergency Handling Module**.
* **Medical Records Management**.
* **HDU and Maternity Management**.
* **Testing and Deployment**.

#### **4. Project Phases & Timeline**

1. **Requirements Gathering & Analysis** (1 weeks)
   1. Stakeholder meetings
   2. Identify functional and non-functional requirements
2. **System Design** (2 weeks)
   1. Create Data Flow Diagrams (DFDs), Entity-Relationship Diagrams (ERD)
   2. Develop the system architecture (microservices, database, frontend design)
   3. Define technical stack
3. **Frontend and Backend Development** (4 weeks)
   1. **Phase 1**: Develop **Patient Registration**, **Appointment**, and **Doctor Management** modules.
   2. **Phase 2**: Develop **Emergency Handling** and **Medical Record** modules.
   3. **Phase 3**: Integrate **HDU**, **Maternity**, and **Notification** modules.
4. **Database Design & Integration** (1 weeks)
   1. Design relational database (MySQL)
   2. Set up security protocols (encryption, access control)
5. **Testing (Unit & Integration)** (3 days
   1. Unit testing for all services
   2. Integration testing between microservices and database
   3. Security and performance testing
6. **User Acceptance Testing (UAT)** (1 week)
   1. UAT with hospital staff (doctors, receptionists)
   2. Make revisions based on feedback
7. **Deployment** (4 days)
   1. Deploy the system to production
   2. Set up backups and recovery processes
8. **Post-Deployment Support & Maintenance** (1 weeks)
   1. Monitor system performance
   2. Fix bugs or issues reported by users

#### 5. Team Roles

* **Project Manager**: Oversee development and ensure timelines are met.
* **Business Analyst**: Requirements gathering, liaising with stakeholders.
* **Frontend Developer**: Develop the user interface using React.
* **Backend Developer**: Develop REST APIs using Python (Flask/Django).
* **Database Administrator**: Set up and manage MySQL database.
* **QA/Test Engineer**: Conduct testing (unit, integration, security).
* **System Administrator**: Handle deployment, monitoring, and maintenance.

#### 6. Risk Management

* **Potential Risks**:
  + Delays in development due to technical issues.
  + Security vulnerabilities in patient data.
  + Resistance from hospital staff to adopt the new system.
* **Mitigation**:
  + Use agile development to track progress and address issues early.
  + Implement strong encryption and access control for data security.
  + Conduct training sessions for hospital staff before deployment.

**Software Architecture & Pattern**

The proposed system will use a microservices architecture, with each module developed as a separate service. This will enable scalability, flexibility, and maintainability.

### **Microservices Architecture**

**Overview:** The microservices architecture involves building the system as a suite of small, independently deployable services. Each service is responsible for a specific functionality and can communicate with other services through well-defined APIs.

**Why Microservices?**

* **Scalability**: Each component, like patient registration, appointment scheduling, or doctor management, can be scaled independently based on demand.
* **Modularity**: Since different parts of the healthcare system (like patient registration, emergency handling, and specialized treatments) can operate independently, each service can be updated, modified, or replaced without affecting the entire system.
* **Resilience**: A failure in one service (e.g., appointment booking) does not take down the entire system, ensuring high availability for critical services like emergency case handling.
* **Technology Flexibility**: Different services can use different technology stacks. For example, the appointment system might use a lightweight queueing system for fast responses, while the patient record system can use a robust database for secure data storage.
* **Faster Development Cycles**: Teams can work on different microservices simultaneously without conflicting with each other. This speeds up development, testing, and deployment.

### **System Breakdown Using Microservices:**

1. **Patient Management Service**
   1. Handles patient registration, updates, and de-registration.
   2. Provides secure storage and retrieval of patient information.
2. **Appointment Service**
   1. Manages the booking, cancellation, and modification of appointments.
   2. Includes emergency handling, where certain appointments bypass the regular process.
3. **Doctor Management Service**
   1. Manages doctor schedules, checking availability, and appointment outcomes.
4. **Medical Record Service**
   1. Stores all patient medical records, prescriptions, and outcomes.
   2. Ensures secure, encrypted access to sensitive patient data.
5. **Notification Service**
   1. Sends appointment confirmations, cancellations, and reminders to patients and doctors.
   2. Can include email or SMS notifications for emergencies.
6. **Specialized Treatment Service**
   1. Manages the routing and handling of patients requiring specialized care such as maternity or HDU.
   2. Automatically allocates resources and staff to specific wards.

**Chosen Architecture:**

* **Frontend:**  **React (JavaScript/TypeScript)**: Component-based framework, ideal for building user interfaces for patients, doctors, and staff.
* **Backend:** The backend will be developed using python, its great for rapid development and has a large ecosystem of healthcare-focused libraries (e.g., Flask/Django for APIs).
* **Database:** The database will be designed using a relational database management system such as MySQL ensuring data security and confidentiality.
* **Flask/Django (Python)**: Lightweight frameworks for developing RESTful APIs.

**Using Python:**

Python will be used as the primary programming language for the system, due to its simplicity, flexibility, and scalability. The following Python frameworks and libraries will be used:

* **React:** For developing the frontend
* **Microservices framework:** For developing the backend
* **MYSQL:** For interacting with the database

By using Python and the proposed architecture, the system will be able to handle a large number of patients and appointments, provide a user-friendly interface, and ensure data security and confidentiality.