1. Introduction

- Project Name: Clinic Management System
- Version: 1
- Author:[PRAGYA MAKADARIYA]
- Date: [19/04/2024]

2. System Overview

The Clinic Management System is designed to streamline clinic operations, including patient registration, appointment scheduling, billing, and medical record management. The system comprises the following modules:

- **Frontend**: React-based user interfaces for doctors, receptionists, and administrators.
- **Backend**: Node.js server handling business logic .
- **Database**: Firebase ,Firestore for data storage.
- **Authentication**: Firebase Authentication for secure user management.

3. Functional Design

3.1User Authentication

- **Login/Signup**: Users authenticate via Firebase Authentication using email and password.
- **Role-Based Access**: (Doctor, Receptionist, Admin) have specific permissions.

3.2Patient Management

- **Registration**: Receptionists can register new patients with personal details.
- **Medical History**: Doctors can update and view patient medical records.
- **Appointments**: Patients can book, reschedule, or cancel appointments through admin.

3.3 Billing System

- Generate by Receptionist
- Bills are download in Pdfs.

4.Technical Design

4.1 Frontend Architecture

- **Components**: Modular React components for each UI element.
- **State Management**: Context API for global state management.
- **Routing**: React Router for navigation between pages.

4.2 Backend Architecture

- **API Endpoints**: RESTful APIs for CRUD operations.
- **Controllers**: Separate controllers for handling business logic.
- **Services**: Utility services for token generation and validation.

4.3 DataBase Schema

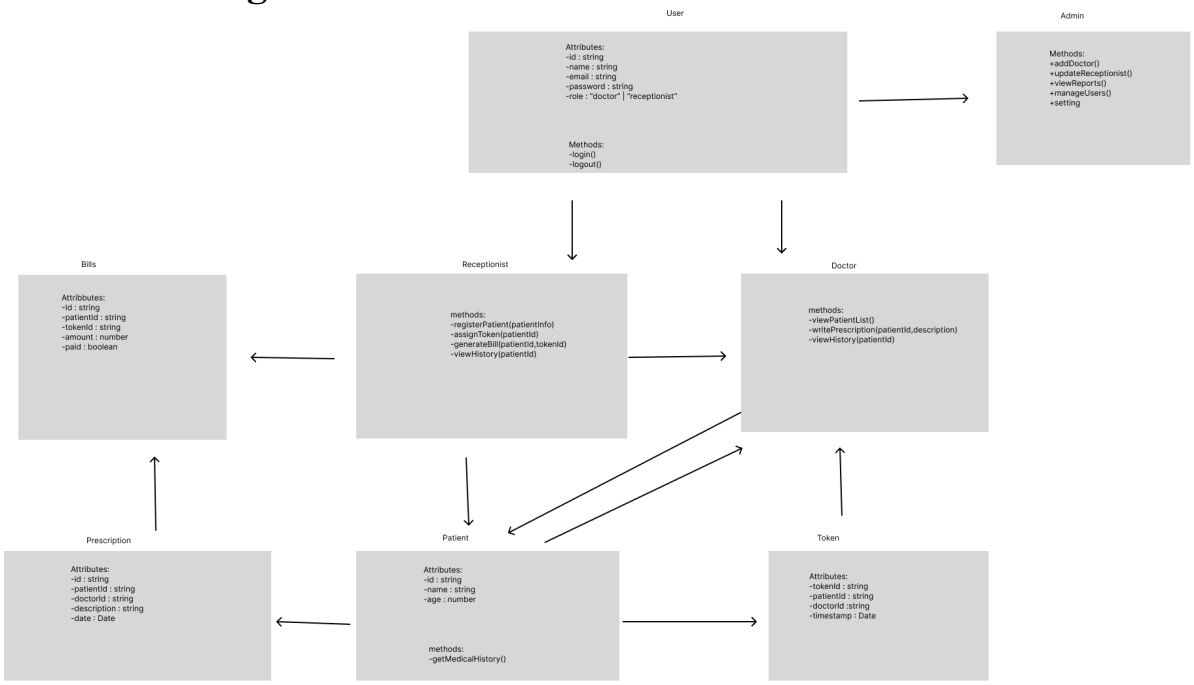
- **Users Collection**: Stores user credentials and roles.
- **Patients Collection**: Stores patient personal and medical information.
- **Appointments Collection**: Stores appointment details.
- **Bills Collection**: Stores billing information and payment status

5. Third-Party Integrations

- **Firebase Authentication**: Manages user authentication and authorization.
- **Firebase Firestore**: NoSQL database for storing application data.
- **Firebase Storage**: Stores generated PDF bills.
- **Firebase Functions**: Handles backend logic and triggers

6. LLd Diagram:

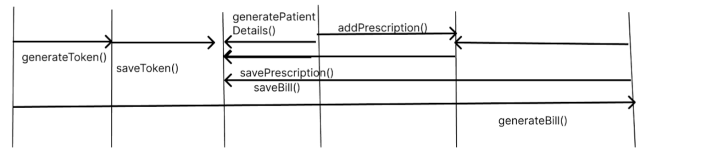
6.1 Class Diagram:



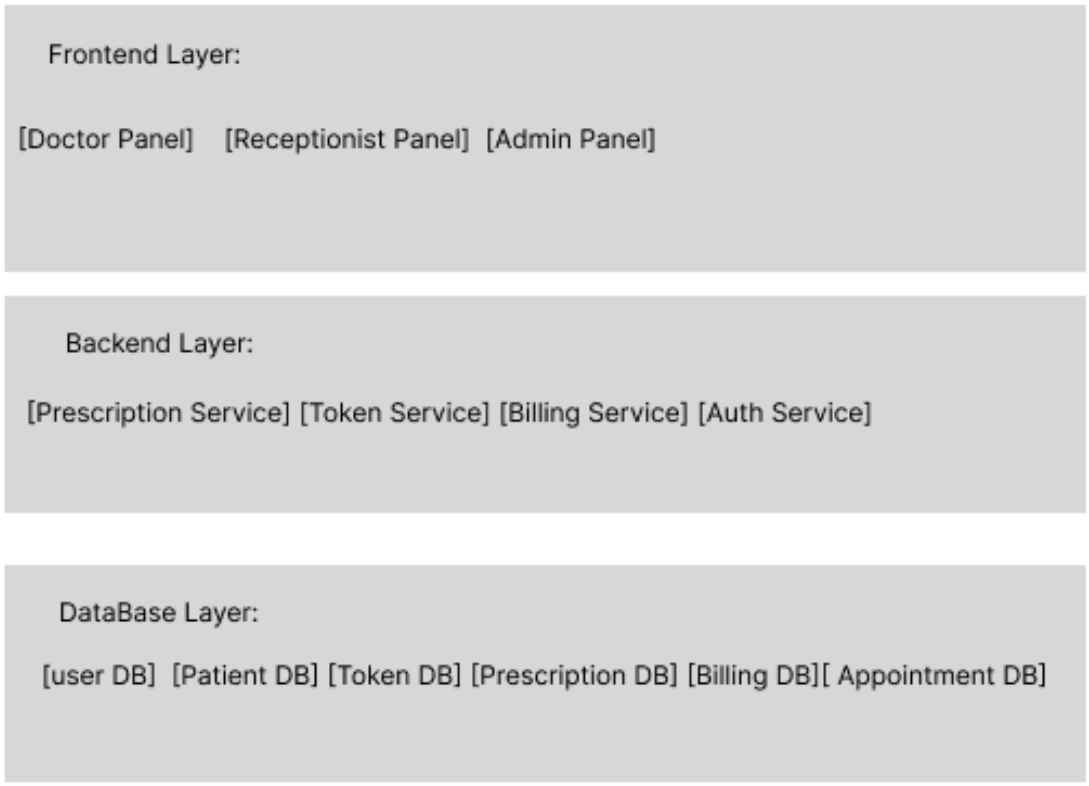
6.2 Sequence Diagram:

Sequence Diagram – Patient Visit Flow

Scenario: A patient arrives at the clinic → Receptionist registers the patient and generates a token → Doctor views patient info and adds prescription → Receptionist views prescription and generates the bill



6.3 Component Diagram:



7. Security Considerations

- **Data Encryption**: All sensitive data is encrypted using Firebase's built-in encryption.
- **Access Control**: Role-based access control ensures users can only access authorized resources.
- **Secure Communication**: All communications are conducted over HTTPS to prevent data interception.

8. Performance Optimization

- **Caching**: Frequently accessed data is cached to reduce database load.
- **Lazy Loading**: Components are loaded only when needed to improve load times.
- **Database Indexing**: Firestore indexes are created on frequently queried fields.

9. Error Handling

- **Frontend**: User-friendly error messages are displayed for common issues.
- **Backend**: API responses include appropriate HTTP status codes and error messages.
- **Logging**: Errors are logged for debugging and monitoring purposes

10. Deployment Strategy

- **Frontend**: Deployed on Firebase Hosting for fast and secure delivery.
- **Backend**: Deployed using Firebase Functions for serverless execution.
- **Database**: Firebase Firestore and Storage are used for data storage and file management