-

DECLARATION

We, the Final Year Students having following Roll Nos and Names do hereby declare that the project "**Cloud Based Healthcare Management System**" has been done by us at Government Autonomous College, Rourkela under the guidance of **Mr. ABEG KUMAR JAISWAL** AND **Mr. ASTWIK BARIK** for the partial fulfillment of the requirement for the award of the Degree of Bachelor in Science from Sambalpur University, Jyotivihar, Burla.

Roll No	Name
22CSC048	PRANISH PRADHAN
22CSC022	ANSUMAN TRIPATHY
22CSC044	ASHUTOSH KHILAR
22CSC052 3	BARUN KUMAR BUDA

-

CERTIFICATE

This is to certify that the students with following Roll Nos and Names,

1. ROLL NO: 22CSC048 NAME: PRANISH PRADHAN
2. ROLL NO: 22CSC022 NAME: ANSUMAN TRIPATHY
3. ROLL NO: 22CSC044 NAME: ASHUTOSH KHILAR
4. ROLL NO: 22CSC052 NAME: BARUN KUMAR BUDA

have developed the project titled **CLOUD BASED HEALTHCARE MANAGEMENT SYSTEM** as a partial fulfillment for the award of the Degree of Bachelor in Science from Sambalpur University, Jyotivihar, Burla.

External Signature **Internal Signature**

-

ACKNOWLEDGEMENT

We Express Our deep sense of gratitude to our esteemed research advisor and guide **Dr Abeg Jaiswal**, **Mr Astwik Barik Lecture in Computer Science Department, Govt Autonomous college, Rourkela** further their day-to-day supervision invaluable suggestion, moral support and encouragement throughout the course of my research work. There can be no more asked of a research advisor thanks to our principal sir and our all teachers for their constant encouragement and timely cooperation during the entire course of our research.

Also thankful to our Computer Science department for providing necessary facilities to carry out our research work.

Last but not the least our regards and deep sense of gratitude are due to our loving parents whose silent blessing throughout our course of work enabled us to complete the task smoothly.

-

Index

Table of Contents:

-

Project Overview

A **Cloud-Based Healthcare Management System** is designed to streamline medical data storage, patient management, and doctor-patient interactions using cloud computing. It provides secure, real-time access to medical records, appointment scheduling, and telemedicine services.

Key Features

- **Patient Module:**
  - Register/login functionality
  - View/update personal details
  - Book/cancel appointments
  - Access medical history and prescriptions
- **Doctor Module:**
  - View assigned patients
  - Update patient medical records
  - Manage appointments and availability
  - Provide e-prescriptions
- **Admin Module:**
  - Manage users (patients, doctors, staff)
  - Monitor system usage and reports
  - Handle emergency cases
- **Telemedicine Support:**
  - Video/audio consultations
  - Chat-based consultation with doctors
- **EHR (Electronic Health Records) Management:**
  - Secure cloud storage of medical history
  - Easy retrieval of reports and test results
- **Pharmacy Integration:**
  - Online medicine ordering
  - Prescription validation
- **Billing & Insurance:**
  - Automated billing system
  - Insurance claim management

Technology Stack

- **Frontend:** React.js / Angular / Flutter (for mobile)
- **Backend:** Node.js / Django / Spring Boot
- **Database:** Firebase / MySQL / MongoDB
- **Cloud Services:** AWS / Google Cloud / Microsoft Azure
- **Security Measures:** JWT authentication, data encryption (AES), HIPAA compliance

Potential Enhancements

- AI-based symptom checker
- Blockchain for medical data security
- Wearable device integration (e.g., Fitbit, Apple Watch)

-

System Requirements

1. Product Perspective

A **Cloud-Based Healthcare Management System** designed for hospitals and clinics to manage patient records, appointments, and medical history securely.

2. Features:

- Secure cloud-based storage of medical records
- Role-based access for doctors, patients, and admins
- Integration with diagnostic centers and pharmacies
- Telemedicine support for remote consultations

3. Hardware Requirements:

- Minimum **Intel Core i5 processor** or equivalent
- **8GB RAM**, **1TB cloud storage**
- **Stable internet connection**

4. Software Requirements:

- **Operating System:** Windows/Linux/MacOS
- **Platform:** Cloud-based (AWS, Azure, Google Cloud)
- **Database:** SQL Server / Firebase / MongoDB
- **Language:** Java, Python, or Node.js

5. Security Considerations:

- **End-to-end encryption** for data security
- **Two-factor authentication** for login
- **HIPAA & GDPR compliance** for data privacy

Assumptions & Constraints

Assumptions:

- The hospital has **high-speed internet** for cloud connectivity.
- Healthcare professionals **are trained** in using digital systems.
- **Data synchronization** between cloud and local storage is seamless.

Constraints:

- **Strict regulatory compliance** (HIPAA, GDPR) must be followed.
- System **must handle high traffic** during peak hospital hours.
- **Only authorized personnel** can modify patient records.

-

Software Requirements Specification (SRS)

-

1. Introduction

1.1 Purpose

The **Cloud-Based Healthcare Management System (CBHMS)** is designed to **digitize and streamline** hospital operations, including **patient management, doctor consultations, appointments, medical records, and billing**. This system ensures secure, efficient, and real-time access to medical data via cloud technology.

1.2 Scope

- **User Roles:** Patients, Doctors, Admins
- **Core Features:**
  - Patient registration & profile management
  - Doctor scheduling & availability management
  - Appointment booking & reminders
  - Electronic Health Records (EHR)
  - Prescription & medication tracking
  - Online consultation via telemedicine
  - Secure billing & insurance management
  - Notifications & alerts
- **Cloud-Based Architecture**: Ensures accessibility, scalability, and data security
- **Compliance:** HIPAA (Health Insurance Portability and Accountability Act)

2. Functional Requirements

2.1 User Management

- Users can **register** and **log in** securely
- Role-based access for **patients, doctors, and admins**
- Patients can **update personal details & medical history**
- Doctors can **manage schedules & availability**

2.2 Appointment Scheduling

- Patients can **book, reschedule, or cancel appointments**
- Doctors can **approve or reject appointment requests**
- Automated **reminders via email/SMS**

2.3 Electronic Health Records (EHR)

- Doctors can **update patient records** (diagnosis, treatment, test reports)
- Patients can **view past medical records**
- Secure **cloud storage** with controlled access

-

2.4 Telemedicine & Consultations

- **Video/audio consultation** via integrated telehealth system
- **Text-based chat** with doctors
- Doctors can **upload prescriptions** after consultation

2.5 Prescription Management

- Doctors can **generate e-prescriptions**
- Patients can **download/view prescriptions**
- Option to **order medicines online**

2.6 Billing & Payments

- **Automated billing system** for consultations & treatments
- Patients can **pay online via credit/debit cards, UPI, insurance**
- **Invoice generation & payment history tracking**

2.7 Notifications & Alerts

- **Reminders for appointments & medication**
- **Emergency notifications** for doctors & admins
- **System updates & announcements**

2.8 Reports & Analytics

- Admins can **generate reports** (appointment trends, revenue, etc.)
- Patients can **track medical progress**

3. Non-Functional Requirements

3.1 Performance Requirements

- System must **support 1000+ concurrent users**
- Response time should be **<2 seconds** for most operations

3.2 Security Requirements

- **AES-256 encryption** for medical records
- **JWT authentication** for user login
- **Role-based access control**

3.3 Availability & Reliability

- **99.9% uptime** with cloud redundancy
- **Automatic backups** every 24 hours

-

3.4 Scalability & Maintainability

- **Modular architecture** to add new features
- **Auto-scaling servers** for handling traffic spikes

4. System Architecture

4.1 Technology Stack

- **Frontend:** React.js / Angular / Flutter (for mobile)
- **Backend:** Node.js / Django / Spring Boot
- **Database:** Firebase / MySQL / MongoDB
- **Cloud:** AWS / Google Cloud / Azure
- **Security:** OAuth 2.0, SSL/TLS encryption

4.2 System Flow

1. **User logs in → Dashboard**
2. **Patient books an appointment → Doctor accepts**
3. **Doctor consults patient → Updates EHR**
4. **Prescription generated → Patient downloads**
5. **Billing processed → Payment recorded**
6. **Reports generated for analytics**

5. Constraints

- Compliance with **healthcare regulations (HIPAA, GDPR)**
- Internet dependency for **cloud-based operations**
- **Multi-language support** for wider accessibility

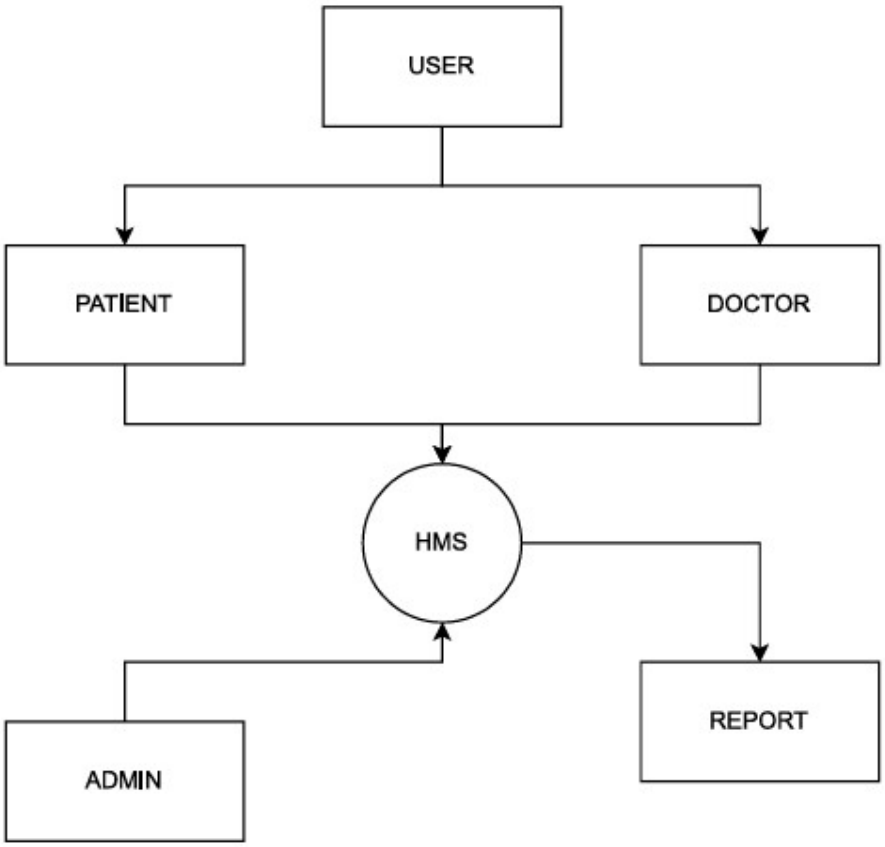
6. Future Enhancements

- **AI-powered symptom checker**
- **Blockchain for medical record security**
- **Wearable device integration (heart rate, BP tracking)**

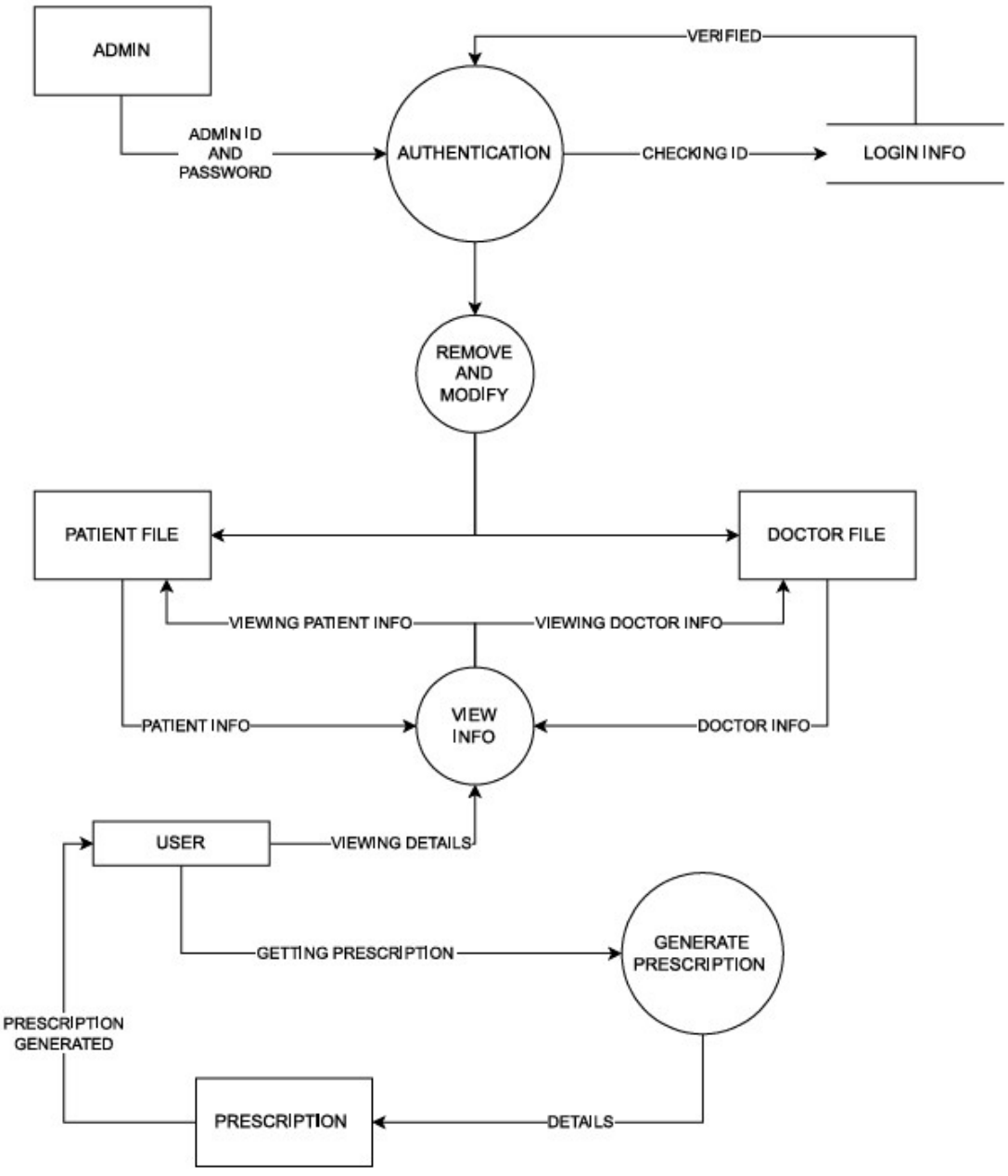
-

Data Flow Diagram (DFD)

Level 0 DFD

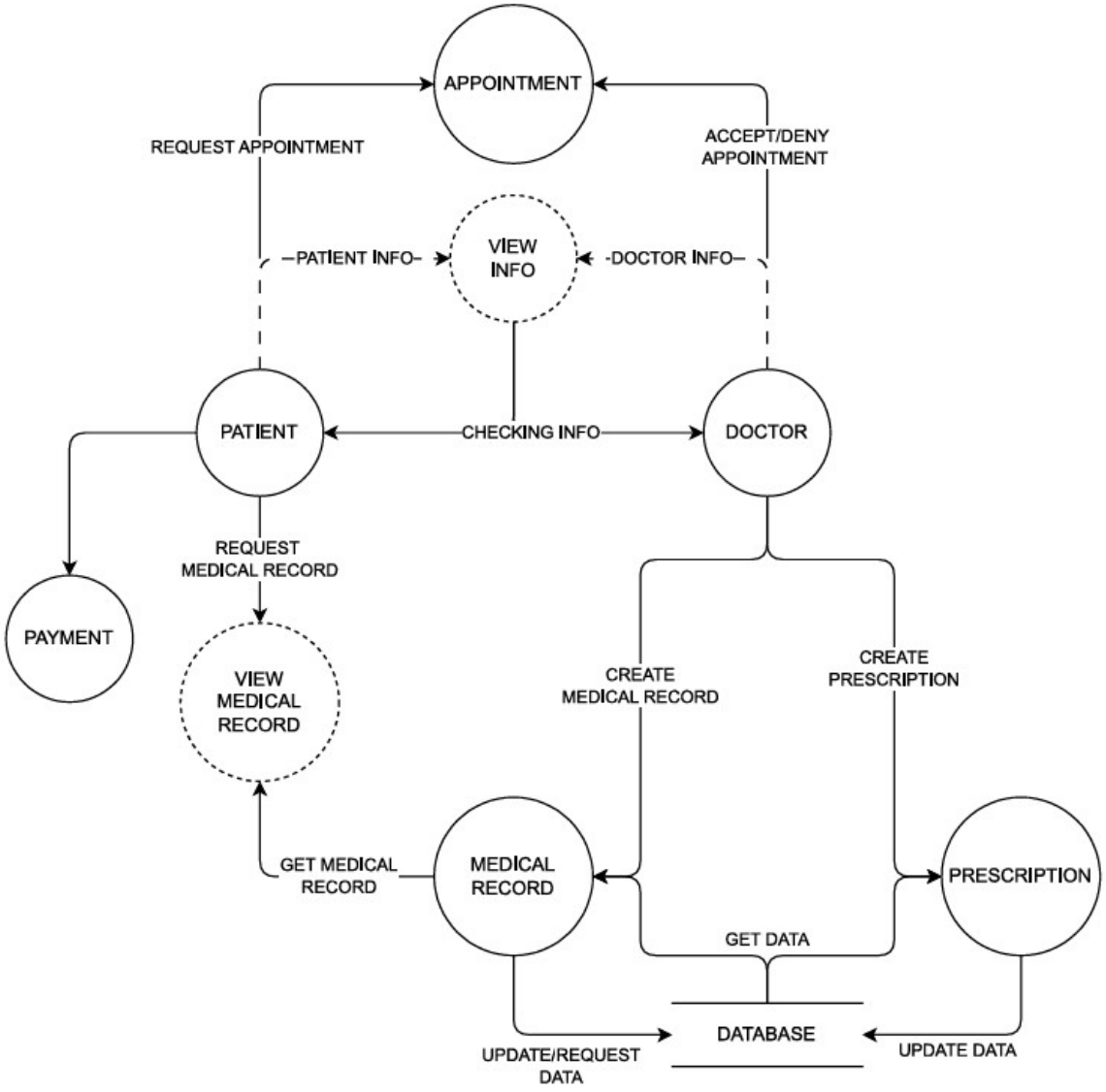


Level 1 DFD


ADMIN
VERIFIED
ADMIN ID AND PASSWORD
AUTHENTICATION
CHECKING ID
LOGIN INFO
REMOVE AND MODIFY
PATIENT FILE
DOCTOR FILE
VIEWING PATIENT INFO
VIEWING DOCTOR INFO
PATIENT INFO
VIEW INFO
DOCTOR INFO
USER
VIEWING DETAILS
GETTING PRESCRIPTION
GENERATE PRESCRIPTION
PRESCRIPTION GENERATED
PRESCRIPTION
DETAILS

-

Level 2 DFD


APPOINTMENT
REQUEST APPOINTMENT
ACCEPT/DENY APPOINTMENT
VIEW INFO
PATIENT INFO
DOCTOR INFO
PATIENT
CHECKING INFO
DOCTOR
REQUEST MEDICAL RECORD
PAYMENT
VIEW MEDICAL RECORD
CREATE MEDICAL RECORD
CREATE PRESCRIPTION
GET MEDICAL RECORD
MEDICAL RECORD
PRESCRIPTION
GET DATA
DATABASE
UPDATE/REQUEST DATA
UPDATE DATA

-

Data Dictionary

1. **Entities & Attributes**:
    - **User** → (User_ID, Name, Email, Password, Role)
    - **Patient** → (Patient_ID, User_ID, DOB, Gender, Address, Medical_History)
    - **Doctor** → (Doctor_ID, User_ID, Specialization, License, Experience)
    - **Appointment** → (Appointment_ID, Patient_ID, Doctor_ID, Date, Status, Notes)
    - **Medical_Record** → (Record_ID, Patient_ID, Doctor_ID, Diagnosis, Prescription, Test_Results)
    - **Prescription** → (Prescription_ID, Patient_ID, Doctor_ID, Medicines, Issued_Date)
    - **Payment** → (Payment_ID, Patient_ID, Amount, Status, Method, Transaction_ID)
    - **Telemedicine** → (Session_ID, Patient_ID, Doctor_ID, Session_Link, Date, Status)
    - **Notification** → (Notification_ID, User_ID, Message, Status)
    - **Admin_Log** → (Log_ID, Admin_ID, Action, Timestamp)
2. **Relationships**:
    - **One User → One Patient (1:1)**
    - **One User → One Doctor (1:1)**
    - **One Patient → Many Appointments (1:M)**
    - **One Doctor → Many Appointments (1:M)**
    - **One Appointment → One Medical Record (1:1)**
    - **One Doctor → Many Prescriptions (1:M)**
    - **One Patient → Many Prescriptions (1:M)**
    - **One Patient → Many Payments (1:M)**
    - **One Doctor ↔ One Patient → Telemedicine Session (M:M)**

-

Use Case

In software engineering, a use case is a list of steps, typically defining interaction between a role (known in Unified modeling language (UML) as an "actor") and a system to achieve a goal.

In system engineering use cases are used at a higher level that within software engineering, often representing missions or stakeholder's goals.

Use Case 1: Book an Appointment

Primary Actor: Patient

Precondition: Patient has logged into the system.

Main Success Scenario:

1. Patient navigates to the appointment booking section.
2. Patient selects a doctor based on specialization and availability.
3. Patient chooses a preferred date and time slot.
4. System confirms the appointment and sends a notification to both the patient and doctor.

Exception Scenario:

- **a)** No available doctors for the selected specialization.
    - System shows an error message and suggests alternative doctors.
- **b)** Selected time slot is no longer available.
    - System prompts the patient to choose another slot.

Use Case 2: View Medical Records

Primary Actor: Patient

Precondition: Patient has logged in and has past medical records stored.

Main Success Scenario:

1. Patient navigates to the medical records section.
2. System retrieves the patient's stored medical history.
3. Patient selects a specific record (prescriptions, lab reports, past diagnoses).
4. System displays the selected medical record.

Exception Scenario:

- **a)** No medical records are found.
    - System shows an error message: "No records available."

-

Use Case 3: Update Patient Diagnosis

Primary Actor: Doctor

Precondition: Doctor has logged into the system.

Main Success Scenario:

1. Doctor navigates to the patient management section.
2. Doctor searches for a specific patient using name or ID.
3. System retrieves and displays the patient's medical history.
4. Doctor updates the patient's diagnosis and prescriptions.
5. System saves the changes and notifies the patient.

Exception Scenario:

- **a)** No patient data found.
    - System shows an error message.
- **b)** Doctor enters incomplete or incorrect data.
    - System prompts the doctor to review and correct the input.

Use Case 4: Process Payment for Consultation

Primary Actor: Patient

Precondition: Patient has booked an appointment that requires payment.

Main Success Scenario:

1. Patient navigates to the billing section.
2. System displays the pending payment details.
3. Patient selects a payment method (Credit Card, Insurance, UPI, etc.).
4. System processes the payment.
5. System confirms the transaction and updates the billing status.

Exception Scenario:

- **a)** No pending payments found.
    - System displays a message: "No payments due."
- **b)** Payment fails due to technical issues.
    - System displays an error message and suggests retrying.

-

Use Case 5: Conduct Telemedicine Consultation

Primary Actor: Doctor & Patient

Precondition: Both doctor and patient have logged in and scheduled a telemedicine appointment.

Main Success Scenario:

1. At the scheduled time, patient and doctor join the virtual consultation.
2. System verifies connectivity and video/audio functionality.
3. Doctor discusses symptoms and updates medical records in real time.
4. Doctor prescribes medication and updates the system.
5. System sends an automatic notification to the patient with prescription details.

Exception Scenario:

- **a)** Patient or doctor fails to join the session.
    - System sends a reminder notification.
- **b)** Technical issues disrupt video consultation.
    - System provides an option to switch to chat-based consultation.

-

Function Point (FP)

Function point (FP) analysis is a structural technique of problem solving. FP is a method to break system into small components so that they can better understand and analysed.

Function point analysis is a unique measure for software.

Function point can used to

Estimate the cost or effort required to design, code and test the software.

Predict the numbers of the errors that will be encountered during the testing.

Forecast the number of component and/or the number of projected source code lines in the implemented system.

FP are derived using an empirical relationship based on

Countable (direct) measures of the software's information domain.

Assessments of the software's complexity.

Information Domain Values Explanation

1. External Inputs (EIs)

- Patient registration involves entering demographic details.
- Appointment booking requires selecting doctor and time slots.
- Updating patient medical history and symptoms.
- Payment information entry for billing.

2. External Outputs (EOs)

- System generates appointment confirmations.
- Prescription reports are shared with patients and pharmacies.
- Billing invoices and payment receipts are provided to patients.

3. External Inquiries (EQs)

- Patients check their medical history and prescriptions.
- Doctors retrieve previous consultation details.
- Users query the system for available appointment slots.

4. Internal Logical Files (ILFs)

- Patient medical records are securely stored.
- Doctor schedules and availability are maintained.
- Billing and payment transaction logs are recorded.

-

5. External Interface Files (EIFs)

- Insurance company integration for verifying claims.
- Third-party pharmacies for e-prescription fulfilment.
- Laboratory integration for automatic test report retrieval.

How to Calculate Function Points

1. Identify and collect the **information domain values**.
2. Complete the **count table** to get the total.
3. Associate **weighting factors** (complexity values).
4. Evaluate and sum up the **adjustment factors**.
5. Compute the function point using the formula:

FP = count total × [0.65 + (0.01 × ∑(Fi))]

Value Adjustment Factors for CBHMS

Factors	**Score (0-5)**
Does the system require reliable backup and recovery?	5
Are specialized data communications required?	4
Are there distributed processing functions?	3
Is performance critical?	3
Will the system be heavily utilized?	4
Does the system require online data entry?	5
Is the data entry transaction built over multiple screens?	4
Are the ILFs updated online?	4
Are the inputs, outputs, files, or inquiries complex?	3
Is the internal processing complex?	4
Is the code designed to be reusable?	3
Are conversion and installation included in the design?	2
Is the system designed for multiple installations?	3
Is the application designed for ease of use?	5
Total:	**45**

-

Information Domain Values for CBHMS

Information Domain Values	Count	Weighting Factor
External Inputs (EIs) - Patient registration, appointment booking, medical data entry	5	3
External Outputs (EOs) - Prescription generation, appointment confirmation, billing details	3	4
External Inquiries (EQs) - Viewing medical records, checking appointment status, retrieving prescriptions	4	3
Internal Logical Files (ILFs) - Patient records, doctor records, billing data	3	7
External Interface Files (EIFs) - Insurance database, pharmacy system, third-party lab reports	2	5
Count Total:		**72**

Since complexity is simple,
∑ (Fi) = 45,
And project FP is **79.2**

-

Effort Estimation

Work effort is the labour required to complete an activity. Work effort is typically the amount of focused an uninterrupted labour time required to compute an activity.

FP-Based Estimation

The **Function Points (FP)** calculated for CBHMS is **79.2**.
To derive an estimate of effort, we use the organizational **productivity rate** for similar systems, assumed to be **6.5 FP/person-month**.

Estimated Effort Calculation

Formula:

Estimated Effort =FP/Productivity Rate

=79.2/6.5

=12.2 person-months

-

Risk Table

Risks	Category	Probability	Impact
Underestimation of project size	PS (Project Size)	50%	3
Higher user traffic than expected	PS	40%	3
Limited reuse of existing components	PS	60%	2
Resistance to new system by hospital staff	BU (Business)	50%	3
Missed deadlines due to requirement changes	BU	60%	2
Funding issues	CU (Customer)	40%	3
Frequent changes in healthcare regulations	PS	70%	2
Integration issues with third-party systems	TE (Technology)	50%	3
Lack of training for hospital staff	DE (Development)	80%	3
Inexperienced development team	ST (Staff)	40%	2
High staff turnover	ST	50%	2

-

CONCLUSION

The Cloud-Based Healthcare Management System (CBHMS) is designed to enhance healthcare services by digitizing patient records, streamlining doctor-patient interactions, and improving hospital management. By leveraging cloud technology, this system ensures secure, scalable, and remote access to healthcare data, benefiting both medical professionals and patients.

Key takeaways from this project include:

- Efficiency: Reduces paperwork and automates appointment scheduling, medical records, and reports.
- Security: Implements role-based access and encryption to protect sensitive health data.
- Scalability: Cloud storage ensures seamless expansion for large healthcare institutions.
- User-Friendliness: An intuitive interface for doctors, patients, and administrators simplifies adoption.

This project serves as a foundation for future improvements, such as AI-driven diagnostics, telemedicine integration, and predictive healthcare analytics, making it a valuable step towards modernized healthcare systems.

-

BIBLIOGRAPHY

- ChatGPT
- Google
- Geeksforgeeks
- Diagram:- https://www.drawio.com/