

---

# *Medical Management System*

---

**Presented by**

Malika Patel

Noureen Tariq

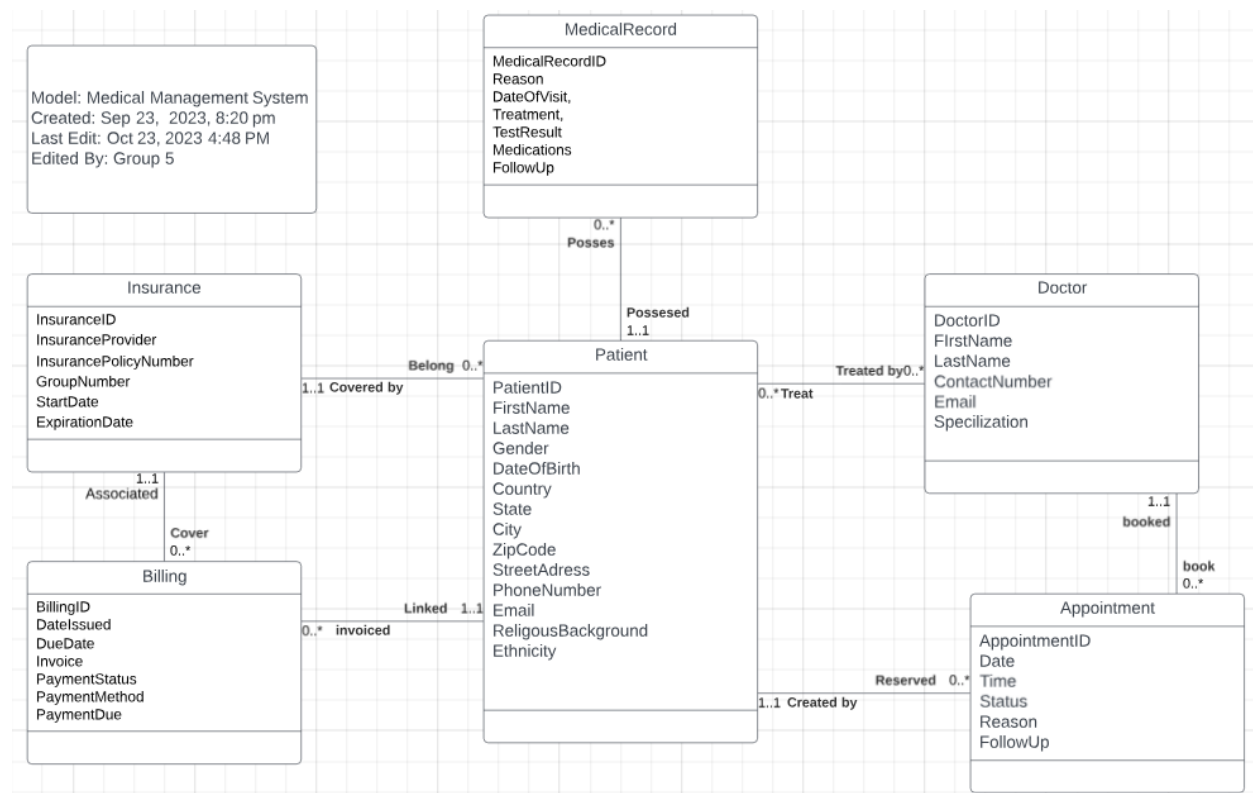
Andy Almanzar

Jaila Kala Gayadin

Mahdia Juber

In the ever-solving landscape of healthcare, the efficient management of medical data, patient information, and hospital operations is paramount. Our project, the Medical Management System, is designed to address the pressing challenges faced by healthcare facilities in managing their resources and patient care effectively.

The medical Management System is envisioned as an all-encompassing software solution tailored to the unique needs of healthcare providers, hospitals, and medical facilities. Its primary objective is to enhance the quality of patient care while optimizing the operational efficiency of healthcare organizations. It aims to do so by targeting recordkeeping, appointment scheduling, internal communication, and payment automation.



## **Normalization**

### **Patient Relation:**

PatientID (Key)

FD1: PatientID → FirstName, LastName, Gender, DateOfBirth, StreetAddress, City, State, Zip, PhoneNumber, EmailAddress, ReligiousBackground, Ethnicity, EmergencyContactNumber

1NF: Meets the definition of a relation.

2NF: No partial Key dependencies

3NF: No Transitive dependencies

\* There is a transitive dependency, but we are de-normalizing zip code, city, and state back together.

### **Doctor Relation:**

DoctorID (Key)

FD1: DoctorID → FirstName, LastName, ContactNumber, Specialization

1NF: Meets the definition of a relation.

2NF: No partial Key dependencies

3NF: No Transitive dependencies

### **Appointment Relation:**

Appointment (Key)

FD1: AppointmentID → Date, Time, Status, Reason, FollowUp, PatientID, DoctorID

1NF: Meets the definition of a relation

2NF: No partial Key dependencies

3NF: No Transitive dependencies

### **MedicalRecord Relation:**

MedicalRecord (Key)

FD1: MedicalRecordID → Reason, DateOfVisit, Treatment, TestResult, Medications, FollowUp, PatientID

1NF: Meets the definition of a relation

2NF: No partial Key dependencies

3NF: No Transitive dependencies

### **Billing Relation:**

BillingID (Key)

FD1: BillingID → DateIssued, DueDate, Invoice, PaymentStatus, PaymentMethod, PatientID, InsuranceID

1NF: Meets the definition of a relation  
2NF: No partial Key dependencies  
3NF: No Transitive dependencies

### **Insurance Relation:**

InsuranceID (Key)

FD1: InsuranceID  $\rightarrow$  InsuranceProvider, InsurancePolicyNumber, GroupNumber, StartDate, ExpirationDate

1NF: Meets the definition of a relation  
2NF: No partial Key dependencies  
3NF: No Transitive dependencies

### **Patient Doctor Relation:**

(PatientID, DoctorID) Key

1NF: Meets the definition of a relation.  
2NF: No partial Key dependencies  
3NF: No Transitive dependencies

### **Normalization and Dependency Analysis:**

The normalization process is done to ensure that every relation is in its 3rd Normal Form (3FN), limiting redundancy and dependencies. The normalization process begins with distinguishing functional dependencies within each relation.

1NF: Ensured that each attribute met the six criteria for being a relation.

2NF: Eliminated partial key dependencies.

3NF: Eliminated transitive dependencies.

The normalization process includes certain steps to ensure that a relational database schema is structured effectively and efficiently:

- First, to achieve the first normal form, the team made sure that each attribute value is a single value only, all values for a given attribute must be of the same data type. Then each attribute should be unique, and the order of these attributes is significant. Lastly, no two tuples in a relation can be identical and the order of tuples is insignificant.

- To achieve the second normal form, the team had to ensure there were no partial dependencies and, in this case, we didn't have any PD.
- Lastly, to achieve the third normal form, the team ensures there are no transitive dependencies, and it must satisfy 1NF and 2NF and in this case, it does.

Denormalization in the patient relation is significant by combining the "Zip," "City," and "State" attributes back together for simplicity and to reduce complexity.

### Tables:

#### CREATE TABLE Patient

```
(
  PatientID VARCHAR(10) NOT NULL,
  FirstName VARCHAR(35),
  LastName VARCHAR(35),
  Gender VARCHAR(1),
  DateOfBirth VARCHAR(8),
  StreetAddress VARCHAR(50),
  City VARCHAR(25),
  State VARCHAR(2),
  PhoneNumber VARCHAR(15),
  EmailAddress VARCHAR(35),
  ZipCode VARCHAR(12),
  ReligiousBackground VARCHAR(20),
  Ethnicity VARCHAR(25),
  EmergencyContactNumber VARCHAR(15),
  PRIMARY KEY (PatientID)
);
```

#### CREATE TABLE doctor

```
(
  DoctorID VARCHAR(10) NOT NULL,
  FirstName VARCHAR(35),
  LastName VARCHAR(35),
  ContactNumber VARCHAR(15),
  DateOfBirth VARCHAR(8),
  Specialization VARCHAR(50),
  PRIMARY KEY (DoctorID)
);
```

#### CREATE TABLE Appointment

```
(
  AppointmentID VARCHAR(10) NOT NULL,
  AppointmentDate VARCHAR(20),
  AppointmentTime VARCHAR(10),
  AppointmentStatus VARCHAR(10),
  FollowUp VARCHAR(25),
  PatientID VARCHAR(10) NOT NULL,
  DoctorID VARCHAR(10) NOT NULL,
  PRIMARY KEY (AppointmentID)
```

```
);
```

### CREATE TABLE MedicalRecord

```
(
MedicalRecordID      VARCHAR(10) NOT NULL,
  Reason              VARCHAR(70),
  DateOfVisit         VARCHAR(10),
  Treatment           VARCHAR(50),
  TestResult          VARCHAR(15),
  Medication           VARCHAR (50),
  FollowUp            VARCHAR (30),
  PatientID           VARCHAR(10) NOT NULL,
  PRIMARY KEY (MedicalRecordID)
);
```

### CREATE TABLE billing

```
(
BillingID              VARCHAR (4) NOT NULL,
DataIssued             VARCHAR (8),
DueData               VARCHAR (8),
Invoice               VARCHAR (55),
PaymentStatus         VARCHAR (10),
PaymentMehod          VARCHAR (25),
PatientID             VARCHAR(10) NOT NULL,
InsuranceID           VARCHAR (20) NOT NULL,
  PRIMARY KEY (BillingID)
);
```

### CREATE TABLE Insurance

```
(
InsuranceID           VARCHAR(20) NOT NULL,
InsuranceProvider      VARCHAR (30),
InsurancePolicyNumber  INTEGER,
GroupNumber           INTEGER,
StartDate              VARCHAR (8),
ExpirationDate         VARCHAR (8),
  PRIMARY KEY (InsuranceID)
);
```

### CREATE TABLE patient\_doctor

```
(
  PatientID           VARCHAR(10) NOT NULL,
  DoctorID            VARCHAR(10) NOT NULL,
  PRIMARY KEY (PatientID, DoctorID)
);
```

### ALTER TABLE FOREIGN KEYS

```
ALTER TABLE appointment
ADD CONSTRAINT FK_patient
FOREIGN KEY (patientID)
REFERENCES patient(patientID);
```

---

### ALTER TABLE appointment

```
ADD CONSTRAINT FK_doctor
FOREIGN KEY (doctorID)
REFERENCES doctor(doctorID);
```

---

### ALTER TABLE MedicalRecord

```
ADD CONSTRAINT FK_MedicalRecord_patient
FOREIGN KEY (patientID)
REFERENCES patient(patientID);
```

---

### ALTER TABLE billing

```
ADD CONSTRAINT FK_patient
FOREIGN KEY (patientID)
REFERENCES patient(patientID);
```

---

### ALTER TABLE billing

```
ADD CONSTRAINT FK_insurance
FOREIGN KEY (insuranceID)
REFERENCES insurance(insuranceID);
```

---

### ALTER TABLE patient\_doctor ADD CONSTRAINT fk\_pd\_doctorid

```
FOREIGN KEY (doctorid) REFERENCES doctor (doctorid);
ALTER TABLE patient_doctor ADD CONSTRAINT fk_pd_patientid
FOREIGN KEY (patientid) REFERENCES patient (patientid)
```

## ALTER TABLE PRIMARY KEYS

### ALTER TABLE Patient

```
ADD CONSTRAINT pk_patient PRIMARY KEY (patientID);
```

---

### ALTER TABLE doctor

```
CONSTRAINT pk_doctor PRIMARY KEY (doctorID)
```

---

### ALTER TABLE appointment

```
CONSTRAINT pk_appointment PRIMARY KEY (appointmentID)
```

---

### ALTER TABLE MedicalRecord

```
CONSTRAINT pk_MedicalRecord PRIMARY KEY (MedicalRecordID)
```

---

### ALTER TABLE billing

```
CONSTRAINT pk_billing PRIMARY KEY (insuranceID)
```

---

## ALTER TABLE insurance

```
CONSTRAINT pk_insurnace PRIMARY KEY (insuranceID)
```

```
INSERT INTO Patient VALUES ('P101', 'Lia', 'Jones', 'F', '02/09/2002', '8989 Smith Rd',  
'Brooklyn', 'NY', '22335', '201-222-3333', 'lia.jones@gmail.com', 'Christian', 'Hispanic', '222-  
345-7689');
```

```
INSERT INTO Patient VALUES ('P102', 'Tony', 'Stark', 'M', '05/29/1979', '10880 Malibu Point',  
'Malibu', 'CA', '90265', '103-456-9870', 'tony.stark@gmail.com', 'Catholic', 'White', '103-865-  
4567');
```

```
INSERT INTO Patient VALUES ('P103', 'Natalie', 'Portman', 'F', '06/09/1981', '25 Dumont Ave',  
'Manhattan', 'NY', '11001', '347-698-8897', 'natalie.portman@gmail.com', 'Jewish', 'White', '917-  
356-6676');
```

```
INSERT INTO Patient VALUES ('P104', 'Mohammed', 'Salah', 'M', '06/15/1992', '725 Main St',  
'Queens', 'NY', '11426', '718-954-9823', 'mo.salah@gmail.com', 'Muslim', 'Arab', '222-345-  
7689');
```

```
INSERT INTO Patient VALUES ('P105', 'Deepika', 'Padukone', 'F', '01/05/1986', '4567 Hart Ln',  
'Bronx', 'NY', '12532', '222-345-9876', 'deepika.padukone@gmail.com', 'Hindu', 'Asian', '543-674-  
9221');
```

```
INSERT INTO Patient VALUES ('P106', 'Olivia', 'Pope', 'F', '11/13/1977', '37 Pine Rd', 'Staten  
Island', 'NY', '28974', '201-345-6721', 'olivia.pope@gmail.com', 'Christian', 'Black', '123-456-  
7890');
```

---

```
INSERT INTO Doctor VALUES ('D101', 'Meredith', 'Grey', '567-785-5575', '04/23/1978', 'General');
```

```
INSERT INTO Doctor VALUES ('D102', 'Derek', 'Shepherd', '345-567-9876', '08/13/1966',  
'Neurology');
```

```
INSERT INTO Doctor VALUES ('D103', 'Cristina', 'Yang', '456-987-6464', '07/20/1977',  
'Cardiothoracic');
```

```
INSERT INTO Doctor VALUES ('D104', 'Alex', 'Karev', '121-343-5454', '05/01/1980', 'Pediatric');
```

```
INSERT INTO Doctor VALUES ('D105', 'Izzie', 'Stevens', '456-987-6464', '09/10/1976', 'OB/GYN');
```

```
INSERT INTO Doctor VALUES ('D106', 'George', 'OMalley', '669-987-7767', '12/24/1978',  
'Cardiothoracic');
```

---

```
INSERT INTO Appointment VALUES ('A101', '01/01/23', '01:30 PM', 'Completed', 'Yes', 'P101',  
'D105');
```

```
INSERT INTO Appointment VALUES ('A102', '01/23/23', '09:30 AM', 'Completed', 'Yes', 'P104',  
'D101');
```

```
INSERT INTO Appointment VALUES ('A103', '02/14/23', '02:15 PM', 'Completed', 'Yes', 'P102',  
'D103');
```

```
INSERT INTO Appointment VALUES ('A104', '03/19/23', '4:45 PM', 'Completed', 'Yes', 'P105',  
'D104');
```



```

INSERT INTO Appointment VALUES ('A105', '04/30/23', '8:00 AM', 'Completed', 'Yes', 'P106',
'D102');

INSERT INTO Appointment VALUES ('A106', '07/22/23', '12:30 PM', 'Completed', 'Yes', 'P103',
'D106');

-----
INSERT INTO MedicalRecord VALUES ('M101', 'Second Trimester Check Up', '01/01/23', 'None',
'Good', 'None', 'No', 'P101');

INSERT INTO MedicalRecord VALUES ('M102', 'Heart Failure', '02/14/23', 'Heart Valve Surgery',
'Good', 'Anticoagulant', 'Yes', 'P102');

INSERT INTO MedicalRecord VALUES ('M103', 'Collapsed Lung', '07/22/23', 'Lung Surgery', 'Good',
'Pain Killer', 'Yes', 'P103');

INSERT INTO MedicalRecord VALUES ('M104', 'Heart Failure', '02/14/23', 'Heart Valve Surgery',
'Good', 'Anticoagulant', 'Yes', 'P104');

INSERT INTO MedicalRecord VALUES ('M105', 'Heart Failure', '02/14/23', 'Heart Valve Surgery',
'Good', 'Anticoagulant', 'Yes', 'P105');

INSERT INTO MedicalRecord VALUES ('M106', 'Heart Failure', '02/14/23', 'Heart Valve Surgery',
'Good', 'Anticoagulant', 'Yes', 'P106');

-----
INSERT INTO Billing VALUES ('B101', '01/02/23', '02/02/23', '$100', 'Unpaid', 'Card', 'P101',
'I101');
INSERT INTO Billing VALUES ('B102', '02/15/23', '03/15/23', '$625', 'Paid', 'Check', 'P102',
'I102');

INSERT INTO Billing VALUES ('B103', '07/24/23', '08/24/23', '$800', 'Paid', 'Check', 'P103',
'I103');

INSERT INTO Billing VALUES ('B104', '01/24/23', '02/24/23', '$625', 'Paid', 'Check', 'P104',
'I104');

INSERT INTO Billing VALUES ('B105', '03/20/23', '04/20/23', '$625', 'Unpaid', 'Check', 'P105',
'I105');

INSERT INTO Billing VALUES ('B162', '05/01/23', '06/01/23', '$625', 'Paid', 'Check', 'P106',
'I106');

-----
INSERT INTO Insurance VALUES ('I101', 'United Health', '12345', '23', '01/01/23', '12/31/25');

INSERT INTO Insurance VALUES ('I102', 'Kaiser Foundation', '54321', '32', '01/01/23',
'12/31/25');

INSERT INTO Insurance VALUES ('I103', 'Anthem INC', '67890', '45', '01/01/23', '12/31/25');

INSERT INTO Insurance VALUES ('I104', 'Centene Corp', '09876', '54', '01/01/23', '12/31/25');

INSERT INTO Insurance VALUES ('I105', 'Humana', '13579', '68', '01/01/23', '12/31/25');

INSERT INTO Insurance VALUES ('I106', 'CVS Health', '97531', '86', '01/01/23', '12/31/25');

```

## **SQL Queries and business questions:**

### **1. What patients still need to pay their invoices? And when does it need to be paid?**

#### **Paitent\_Unpaid\_Info**

```
SELECT p.patientid, p.firstname, p.lastname, a.appointmentid,
a.appointmentdate, b.paymentstatus, b.invoice, b.duedata as DueDate
FROM (patient AS p
      INNER JOIN appointment AS a ON p.patientid = a.patientid)
INNER JOIN billing AS b ON p.patientid = b.patientid
WHERE b.paymentstatus = 'Unpaid';
```

#### **Report:**

Patient_Unpaid_Info							
patientid	firstname	lastname	appointme	appointmentd	paymentst	invoice	Due
P101	Lia	Jones	A101	01/01/23	Unpaid	\$100	02/0
P105	Deepika	Padukone	A104	03/19/23	Unpaid	\$625	04/2

**Description:** This SQL query identifies patients who haven't paid their invoices. It retrieves information like patient ID, first name, last name, appointment ID, appointment date, payment status, invoice details, and the due date for the payment. This query includes an inner join between the "Patient", "Appointment", and "Billing" tables, connecting all of them based on the patient ID. The results are then filtered to only include the records where the payment status is shown as "Unpaid", giving it a list of patients who have good invoices and its associated information, like the due date for the payment.

### **2. How many patients are there in each city or state?**

```
SELECT City, State,
COUNT(PatientID) AS Total_Patients
FROM Patients
GROUP BY State, City DESC;
```

#### **Report:**

## Total\_Patient\_By\_State

City	State	Total_Patients
Malibu	CA	1
Bronx	NY	1
Brooklyn	NY	1
Manhattan	NY	1
Queens	NY	1
Staten Island	NY	1

**Description:** This SQL query provides several patients grouped by city and state. It selects the state, city, and count of patient IDs for each of the combinations of the city and state from the “Patients” table. The result shows the total number of patients in each of the cities and states. The query uses the GROUP BY clause to organize the data which is also in descending order by city.

### 3. What are the demographics of the patients?

```
SELECT Gender, Ethnicity,
COUNT(*) AS NumberOfPatients
FROM Patient
GROUP BY Gender, Ethnicity;
```

Report:

## Patient\_demographic

Gender	Ethnicity	NumberOfPatients
F	Asian	1
F	Black	1
F	Hispanic	1
F	White	1
M	Arab	1
M	White	1

**Description:** This SQL query gives an overview of patient demographics by counting the number of patients for every unique combination of gender and ethnicity. It selects the “Gender” and “Ethnicity” columns from the Patient table and uses the GROUP BY to organize it. The result then includes the count of patients for each gender and ethnicity, giving it a summary of the patient population based on the factors.

#### 4. What patient is linked to a specific insurance provider?

```
SELECT TOP 1
P.PatientID,
P.FirstName,
P.LastName,
P.DateOfBirth,
P.State,
I.InsuranceProvider,
I.InsurancePolicyNumber
FROM
Patient AS P
INNER JOIN Insurance AS I
ON P.PatientID = I.PatientID;
```

#### Report:

Patient_insurance_info						
PatientID	FirstName	LastName	DateOf	State	InsuranceProvider	Number
P101	Lia	Jones	02/09/2	NY	United Health	12345
P102	Tony	Stark	05/29/1	CA	Kaiser Foundation	54321
P103	Natalie	Portman	06/09/1	NY	Anthem INC	67890
P104	Mohammed	Salah	06/15/1	NY	Centene Corp	9876
P105	Deepika	Padukone	01/05/1	NY	Humana	13579
P106	Olivia	Pope	11/13/1	NY	CVS Health	97531

**Description:** This SQL query would retrieve information about a patient that is linked to a specific insurance provider. The “TOP 1” would limit the result to only 1 record, providing details like the patient’s ID, date of birth, name, state of residence, and the insurance provider

that's associated with it. The query uses an inner join between the Patient and Insurance tables based on the familiar "PatientID" to make sure that only patients associated with the insurance details are included in the result.

##### 5. Navigating the Patient Information based on the patient ID; gives all the entities on a single patient ID

Report:

Patient_Navigation									
Patient	FirstName	LastName	Date	State	City	Phone	EmailAddress	Religious	Emerg
P101	Lia	Jones	02/09	<input type="text" value="NY"/>	Brooklyn	22335	201-222-3333	Christian	222-34

```
SELECT P.PatientID, P.FirstName, P.LastName, P.DateOfBirth, P.State, P.City,
P.PhoneNumber, P.EmailAddress, P.ReligiousBackground, P.EmergencyContactNumber
FROM Patient AS P INNER JOIN Insurance AS I ON P.PatientID = I.PatientID
GROUP BY P.PatientID, P.FirstName, P.LastName, P.DateOfBirth, P.City, P.State,
P.PhoneNumber, P.EmailAddress, P.ReligiousBackground, P.EmergencyContactNumber;
```

**Description:** This SQL query combines patient information based on their distinct patient ID. It then combines details such as the name, contact information, religious background, emergency contact, and date of birth. The query uses an inner join between the "Patient" and "Insurance" tables, which grouped the results by their patient ID and give a summary of patient data.

##### 6. What patients have a heart failure-related medical issue and what are the doctors that are addressing it?

```
SELECT p.patientid, p.firstname, p.lastname, m.medicalrecordid, m.reason, d.doctorid,
d.firstname, d.lastname, d.specialization
FROM ((MedicalRecord AS m
INNER JOIN patient AS p ON m.patientid = p.patientid)
INNER JOIN patient_doctor AS pd ON p.patientid = pd.patientid)
INNER JOIN doctor AS d ON pd.doctorid = d.doctorid
WHERE m.reason = 'Heart Failure';
```

## Report:

### Patient\_Heart\_doctor

patie	p.firstname	p.lastname	medic	reason	docto	d.firstname	d.lastname	specializ
P102	Tony	Stark	M102	Heart Failure	D103	Cristina	Yang	Cardioth
P104	Mohammed	Salah	M104	Heart Failure	D106	George	OMalley	Cardioth
P105	Deepika	Padukone	M105	Heart Failure	D101	Meredith	Grey	General
P106	Olivia	Pope	M106	Heart Failure	D106	George	OMalley	Cardioth

**Description:** This SQL query selects patient, medical record, and doctor information and cross-references them to find patients at risk for heart failure and doctors who are treating them. It utilizes INNER JOIN to combine patients with their medical records as well as combine doctors with their patients. By filtering specifically for m.reason heart failure, the query can effectively return all patients and doctors who are associated with that medical reason.

## Data Entry Forms:

# Appointment

Please enter appointment information regarding the patient.

AppointmentID

A101

AppointmentDate

01/01/23

AppointmentTime

01:30 PM

AppointmentStatus

Completed

FollowUp

Yes

PatientID

P101


DoctorID

D105

Add appointment

Save Appointment

Close Form





**Description:** The Appointment Form accepts the AppointmentID, AppointmentDate, AppointmentTime, AppointmentStatus, FollowUp, PatientID, and DoctorID. The basis of this form is to keep track of appointment information. This will be useful when being referred back to

during a follow-up appointment. This form will be most useful to those creating appointments as they can easily add new appointments and save appointments.

# Billing

Please add customer billing information.

BillingID	<input type="text" value="B101"/>	<input type="button" value="Add Billing"/>	
DataIssued	<input type="text" value="01/02/23"/>	<input type="button" value="Save Billing Information"/>	
Due Date	<input type="text" value="02/02/23"/>		
Invoice	<input type="text" value="\$100"/>		
PaymentStatus	<input type="text" value="Unpaid"/>		
PaymentMethod	<input type="text" value="Card"/>		
PatientID	<input type="text" value="P101"/>		
InsuranceID	<input type="text" value="I101"/>	<input type="button" value="Previous Billing"/>	

**Description:** The Billing form streamlines the billing management information. At the core is the BillingID which is also attached to the PatientID. Since most patients have insurance, we have also included the InsuranceID. This will make it easier for those working on the bills to contact the insurance provider. The billing form also included the options to Add Billing, Save Billing Information, and go back to the Previous Billing.

## Doctor

Please enter information regarding the doctor here.

DoctorID

FirstName

LastName

ContactNumber

DateOfBirth

Specialization



First Record



**Description:** The Doctor information form serves as an important tool to maintain a centralized system of information on all the medical professionals within the hospital. The information utilized is the DoctorID, which is a distinct ID given to each doctor. It also includes personal information such as FirstName, LastName, ContactNumber, DateofBirth, and specialization.

## Insurance

Please enter all information regarding insurance.

InsuranceID

InsuranceProvider

InsurancePolicyNum

GroupNumber

StartDate

ExpirationDate



**Description:** The Insurance form is designed to capture important information related to an individual's insurance information. This form requires the unique identifier, which is assigned to





all insurance providers, the policy number, group number, and the start and expiration date of the insurance.

## MedicalRecord

MedicalRecordID	<input type="text" value="M101"/>	<input type="button" value="Add New Medical Record"/>
Reason	<input type="text" value="Second Trimester Check Up"/>	
DateOfVisit	<input type="text" value="01/01/23"/>	
Treatment	<input type="text" value="None"/>	
TestResult	<input type="text" value="Good"/>	
Medication	<input type="text" value="None"/>	
FollowUp	<input type="text" value="No"/>	
PatientID	<input type="text" value="P101"/>	<input type="button" value="X"/>

The MedicalRecord form organizes the details related to a patient's medical history and their healthcare treatment. It will be able to track and manage the patient's health, providing valuable information for future consultations, treatments, and follow-up care and be able to connect it to a patient using their PatientID.

# Patient

PatientID	<input type="text" value="P101"/>	
FirstName	<input type="text" value="Lia"/>	
LastName	<input type="text" value="Jones"/>	
Gender	<input type="text" value="F"/>	
DateOfBirth	<input type="text" value="02/09/20"/>	
StreetAddress	<input type="text" value="8989 Smith Rd"/>	
City	<input type="text" value="Brooklyn"/>	
State	<input type="text" value="NY"/>	
PhoneNumber	<input type="text" value="22335"/>	
EmailAddress	<input type="text" value="201-222-3333"/>	
ZipCode	<input type="text" value="lia.jones@gr"/>	
ReligiousBackground	<input type="text" value="Christian"/>	
Ethnicity	<input type="text" value="Hispanic"/>	
EmergencyContactN	<input type="text" value="222-345-7689"/>	

**Description:** The Patient form allows the user to enter basic and essential patient information including name, gender, date of birth, address, number, email, religious background, ethnicity, and emergency contact.

## Navigation Form

The screenshot displays a web application interface for a medical management system. At the top, a purple header bar contains a menu icon and the text "Navigation Form". Below this, a horizontal navigation bar features several tabs: "Appointment" (highlighted with an orange border), "Billing", "Doctor", "Insurance", "MedicalRecord", "Patient", "patient\_doctor\_form", and "[Add New]". The main content area has a light purple background. On the left, the word "Appointment" is written in large, bold, dark grey font. To its right, a smaller instruction reads: "Please enter appointment information regarding the patient." Below this, a form is displayed with the following fields and values: "AppointmentID" (A101), "AppointmentDate" (01/01/23), "AppointmentTime" (01:30 PM), "AppointmentStatus" (Completed), "FollowUp" (Yes), "PatientID" (P101), and "DoctorID" (D105). To the right of the form fields are three buttons: "Add appointment", "Save Appointment", and "Close Form". Below the "Close Form" button is a calendar icon with a checkmark. At the bottom of the interface, a dark grey footer bar contains the text "Record: 1 of 6", a "No Filter" button, and a "Search" button.

**Description:** The Navigation Form provides a centralized platform for efficient healthcare administration. It ensures ease of use and facilitates seamless navigation between different forms for comprehensive patient care and record-keeping.

### **Conclusion:**

Throughout the development of our Medical Management System, our team implemented a comprehensive communication strategy utilizing platforms such as WhatsApp and Zoom to communicate and track the progress the team made at each milestone. This approach allowed the team to address questions and concerns, seek help from Professor Holowczak, and troubleshoot any issues as a team. The team also utilized Microsoft Word and Google Docs which allowed us

to collaborate on the actual write-up of the project. Over the course of our project, we gained valuable insight into medical management and healthcare systems, including the significance of accurate record-keeping, streamlined communication, and the integration of automation in billing processes. We also learned more about the function of databases by executing the lessons from the course curriculum in our formation of the database starting from the formation of our Entity Relationship Model (ERD) to our final product in Microsoft Access. The part that we found to be the most challenging was developing the ERD, specifically trying to understand the relationships between entities. The easiest part was creating the INSERT statements and ALTER tables because they mainly were procedural steps, offering a clear and efficient approach based on our entities and attributes. Overall, the team learned how to utilize MS Access, write SQL, and how useful it is in developing a database application. Reflecting on our project this semester, something we would have done differently would be to not utilize as many entities to make it less complex. Additionally, we could have incorporated each step in the document more thoroughly as well as spent more time planning at the start to help with our organization overall.