[**BUAN 6320.504 - Database Foundations for Business Analytics - F23**](https://elearning.utdallas.edu/webapps/blackboard/execute/courseMain?course_id=_330166_1)

Project – Technical Report BUAN 6320

**Group #11**

**Hospital Database Management System**

**Group Members:**

1. **Aashay Bhujbal -asb220014**
2. **Mani Saxena -mxs230066**
3. **Harsh Kumar Lal Das -hxl220024**
4. **Karthik Reddy Vuyyuru -kxv230006**

# Introduction

This Database Design Document describes the design and implementation of a database that will store user data for the online portion of the upcoming video game *Platformer Kings,* developed and maintained by Far Harbor Games, Inc. This document was developed and written concurrently with the development of the proposed database, and is intended strictly for internal use at Far Harbor Games and its business partners. The details contained within constitute trade secrets belonging to Far Harbor Games and should not be disseminated outside of official company business.

# Assumptions and Constraints

**Assumptions**

* Here we have only considered for inpatient.
* For appointments, only perfect hours can be selected.
* Each patient can only be assigned to one Bill at a time.
* Each patient can only be treated by one doctor at a time.

# Statement of Work

# Purpose

Overall, a hospital management system project is an important investment for healthcare providers looking to improve the quality and efficiency of their operations. With the right system in place, hospitals can better manage patient care and financial operations, leading to more sustainable business model for the hospital.

# Project Scope

The hospital management system is designed to manage and streamline the operations of a hospital or medical center. The system can help healthcare providers manage doctor, patient records, appointments, prescriptions, bills as well as other important aspects of hospital operations.

# Entity and Attribute Description:

1. **Doctor**- #Entity of the doctors working in the hospital.

Attributes as follows:

Doc\_ID(PK)(Int) #Primary Key

Apt\_ID(FK)(Int) #Foreign Key referencing Appointment Entity

Doc\_Name(varchar) #Name of Doctor

Doc\_Specialisation(varchar) #Specialisation of the Doctor

Doc\_Shift(Date) #Date of Doctors shift

1. **Patient**- #Entity of the patients visiting the hospital.

Attributes as follows:

P\_ID(PK)(Int) #Primary Key

Apt\_ID(FK)(Int) #Foreign Key referencing Appointment Entity

P\_Name(varchar) #Name of the patient

P\_Gender(varchar) #Gender of the patient

P\_Address(varchar) #Address of the patient

P\_Phone(Int) #Phone number of the patient

1. **Appointment**- #Entity of appointments given to patients.

Attributes as follows:

Apt\_ID(PK) (Int) #Primary Key

P\_ID(FK)(Int) #Foreign Key referencing to patient Entity

Apt\_Date(Date) #Date of the appointment

Apt\_Time(Time) #Time slot of the appointment

Apt\_Type(varchar) #General Consultation or specialised appointment

Apt\_Room(Int) #Designated room number

1. **Prescription**- #Entity of prescriptions written by doctor to patient.

Attributes as follows:

Pres\_ID(PK)(Int) #Primary Key

Doc\_ID(FK)(Int) #Foreign Key referencing Doctor Entity

P\_ID(FK)(Int) #Foreign Key referencing Patient Entity

Pres\_Date(Date) #Date of prescription given

Pres\_Validity(Date) #Date till validity of prescription

1. **Bill**- #Entity of bill generated by hospital to patient.

Attributes as follows:

Bill\_Id(PK)(Int) #Primary Key

P\_ID(FK)(Int) #Foreign Key referencing Patient Entity

Bill\_Date(Date) #Date of bill generated.

Bill\_Amt(Int) #Amount of bill

Bill\_Status(varchar) #Status whether Paid or Unpaid

# Relationship and Cardinality Description:

1. DOCTOR - APPOINTMENT 1 : M

1 Doctor can give out Multiple Appointments.

Multiple appointments can be given by 1 doctor.

1. APPOINTMENT - PATIENT 1 : 1

1 appointment can be given to 1 patient.

1 patient can be given 1 appointment.

1. PATIENT - BILL 1 : M

1 patient can be issued multiple bill.

Multiple bill can be issued to 1 patient.

1. PATIENT - PRESCRIPTION 1 : M

1 patient can be prescribed multiple prescriptions.

Multiple prescriptions can be prescribed to 1 patient.

1. DOCTOR - PRESCRIPTION 1 : M

1 doctor can prescribe multiple prescriptions.

Multiple prescriptions can be prescribed by 1 doctor.

1. DOCTOR - PATIENT 1 : M

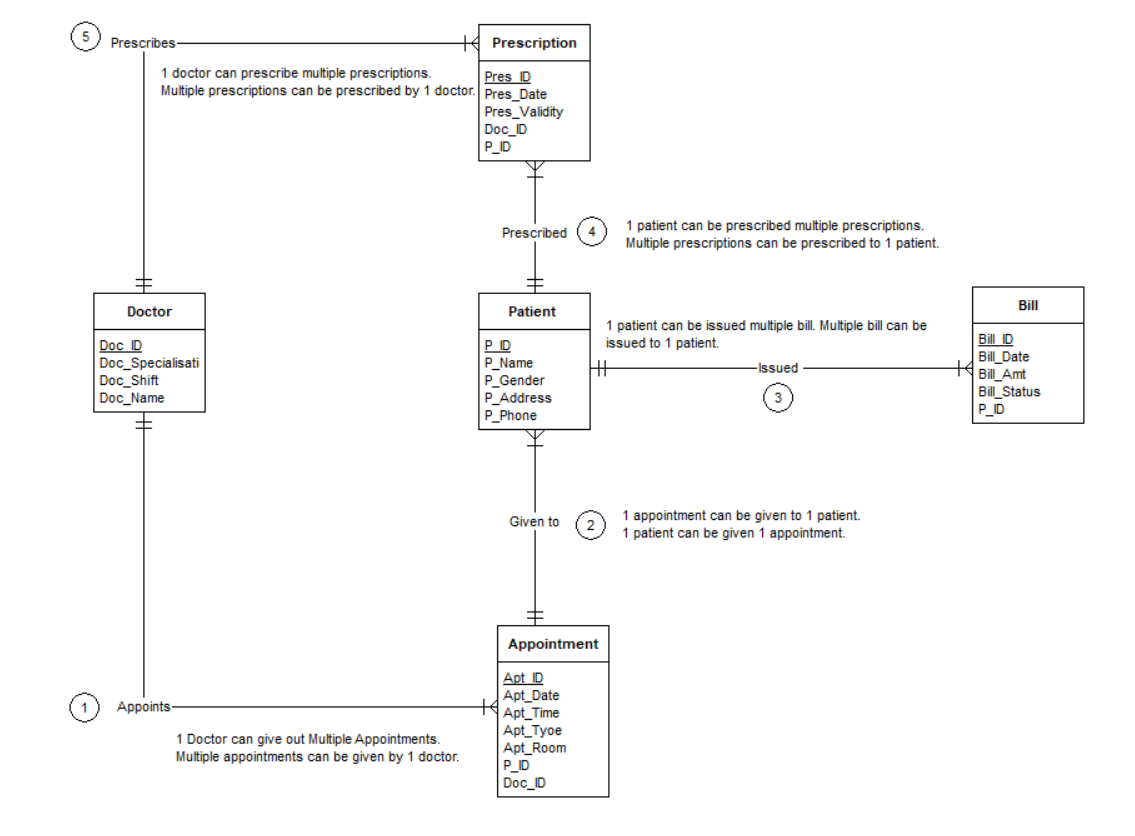
1 doctor can be appointed to multiple patients.

Multiple patients can be appointed to 1 doctor.

# Detailed Database Design

**Entity Relationship Diagram**

*(Note: Due to limitations in ER-Assistant, the software used to create this entity-relationship diagram, it is not possible to resize the entity boxes to avoid truncating entity and attribute names.)*

******

# DDL Source Code

/\*

Project - DDL

\*/

CREATE TABLE Doctor (

Doc\_id integer PRIMARY KEY,

Doc\_Name varchar,

Doc\_Specialisation varchar,

Doc\_Shift Date

);

CREATE TABLE Patient (

P\_ID integer PRIMARY KEY,

P\_Name varchar,

P\_Gender varchar,

P\_Address varchar,

P\_Phone integer

);

CREATE TABLE Appointment (

Apt\_ID integer PRIMARY KEY ,

P\_ID integer ,

Doc\_id integer ,

Apt\_Date Date ,

Apt\_Time Time ,

Apt\_Type varchar,

Apt\_Room integer ,

FOREIGN KEY (P\_ID) REFERENCES Patient(P\_ID),

FOREIGN KEY (Doc\_id) REFERENCES Doctor(Doc\_id)

);

CREATE TABLE Prescription (

Pres\_ID integer PRIMARY KEY,

Doc\_ID integer,

P\_ID integer,

Pres\_Date Date,

Pres\_Validity Date,

FOREIGN KEY (P\_ID) REFERENCES Patient(P\_ID),

FOREIGN KEY (Doc\_ID) REFERENCES Doctor(Doc\_ID)

);

CREATE TABLE Bill (

Bill\_Id integer PRIMARY KEY,

P\_ID integer ,

Bill\_Date Date,

Bill\_Amt integer,

Bill\_Status varchar,

FOREIGN KEY (P\_ID) REFERENCES Patient(P\_ID));

# DML and Query Source Code

/\* Populate all tables \*/

INSERT INTO Doctor (Doc\_id, Doc\_Name, Doc\_Specialisation, Doc\_Shift, Doc\_Professional\_Experience) VALUES

(1, 'Dr. Smith', 'Cardiologist', '2023-11-09', 11),

(2, 'Dr. Johnson', 'Paediatrician', '2023-11-10', 12),

(3, 'Dr. Davis', 'Dermatologist', '2023-11-11', 5),

(4, 'Dr. Wilson', 'Neurologist', '2023-11-12', 6),

(5, 'Dr. Miller', 'Gastroenterologist', '2023-11-13', 3),

(6, 'Dr. Harris', 'Ophthalmologist', '2023-11-14', 7),

(7, 'Dr. Robinson', 'ENT Specialist', '2023-11-15', 8),

(8, 'Dr. Lee', 'Psychiatrist', '2023-11-16',10),

(9, 'Dr. Turner', 'Obstetrician', '2023-11-17', 4),

(10, 'Dr. Martinez', 'Orthopaedic Surgeon', '2023-11-18', 3),

(11, 'Dr. Baker', 'Urologist', '2023-11-19', 5),

(12, 'Dr. Carter', 'Rheumatologist', '2023-11-20', 7);

INSERT INTO Patient (P\_ID, P\_Name, P\_Gender, P\_Address, P\_Phone) VALUES

(1, 'John Doe', 'Male', '123 Main St', 555-1234),

(2, 'Jane Smith', 'Female', '456 Oak St', 555-5678),

(3, 'David Johnson', 'Male', '789 Pine St', 555-9012),

(4, 'Emily Davis', 'Female', '101 Elm St', 555-3456),

(5, 'Daniel Wilson', 'Male', '202 Maple St', 555-7890),

(6, 'Olivia Miller', 'Female', '303 Birch St', 555-1234),

(7, 'William Harris', 'Male', '404 Cedar St', 555-5678),

(8, 'Sophia Robinson', 'Female', '505 Spruce St', 555-9012),

(9, 'Ethan Lee', 'Male', '606 Pine St', 555-3456),

(10, 'Ava Turner', 'Female', '707 Oak St', 555-7890),

(11, 'Mia Martinez', 'Female', '808 Maple St', 555-1234),

(12, 'Noah Baker', 'Male', '909 Elm St', 555-5678);

INSERT INTO Appointment (Apt\_ID, Doc\_id, P\_ID, Apt\_Date, Apt\_Time, Apt\_Type, Apt\_Room) VALUES

(101, 1,1, '2023-11-09', '10:00:00', ' Cardiologist Consultation', 101),

(102, 2, 2,'2023-11-10', '11:30:00', 'Paediatrician Consultation', 102),

(103, 3, 3, '2023-11-11', '14:00:00', 'Dermatology Appointment', 103),

(104, 4, 4,'2023-11-12', '15:30:00', 'Neurology Consultation', 104),

(105, 5, 5, '2023-11-13', '17:00:00', 'Gastroenterology Appointment', 105),

(106, 6, 6,'2023-11-14', '09:30:00', 'Ophthalmology Consultation', 106),

(107, 7, 7, '2023-11-15', '13:00:00', 'ENT Appointment', 107),

(108, 8, 8, '2023-11-16', '16:30:00', 'Psychiatry Consultation', 108),

(109, 9, 9, '2023-11-17', '18:00:00', 'Obstetrics Appointment', 109),

(110, 10, 10, '2023-11-18', '10:30:00', 'Orthopaedic Consultation', 110),

(111, 11, 11, '2023-11-19', '12:00:00', 'Urology Appointment', 111),

(112, 12, 12, '2023-11-20', '14:30:00', 'Rheumatology Consultation', 112);

INSERT INTO Prescription (Pres\_ID, Doc\_ID, P\_ID, Pres\_Date, Pres\_Validity) VALUES

(1, 1, 1, '2023-11-09', '2023-12-09'),

(2, 2, 2, '2023-11-10', '2023-12-10'),

(3, 3, 3, '2023-11-11', '2023-12-11'),

(4, 4, 4, '2023-11-12', '2023-12-12'),

(5, 5, 5, '2023-11-13', '2023-12-13'),

(6, 6, 6, '2023-11-14', '2023-12-14'),

(7, 7, 7, '2023-11-15', '2023-12-15'),

(8, 8, 8, '2023-11-16', '2023-12-16'),

(9, 9, 9, '2023-11-17', '2023-12-17'),

(10, 10, 10, '2023-11-18', '2023-12-18'),

(11, 11, 11, '2023-11-19', '2023-12-19'),

(12, 12, 12, '2023-11-20', '2023-12-20');

INSERT INTO Bill (Bill\_Id, P\_ID, Bill\_Date, Bill\_Amt, Bill\_Status) VALUES

(1, 1, '2023-11-09', 100, 'Paid'),

(2, 2, '2023-11-10', 150, 'Unpaid'),

(3, 3, '2023-11-11', 120, 'Paid'),

(4, 4, '2023-11-12', 80, 'Unpaid'),

(5, 5, '2023-11-13', 200, 'Paid'),

(6, 6, '2023-11-14', 90, 'Unpaid'),

(7, 7, '2023-11-15', 110, 'Paid'),

(8, 8, '2023-11-16', 130, 'Unpaid'),

(9, 9, '2023-11-17', 180, 'Paid'),

(10, 10, '2023-11-18', 160, 'Unpaid'),

(11, 11, '2023-11-19', 140, 'Paid'),

(12, 12, '2023-11-20', 170, 'Unpaid');

---------Sequence for Doctor id

CREATE SEQUENCE doc\_id\_seq

INCREMENT BY 1

START WITH 1

MINVALUE 1

NO MAXVALUE

CACHE 1;

---------Sequence for Appointment Id

CREATE SEQUENCE Apt\_ID\_seq

INCREMENT BY 1

START WITH 1

MINVALUE 1

NO MAXVALUE

CACHE 1;

------Sequence for Patient Id

CREATE SEQUENCE P\_ID\_seq

INCREMENT BY 1

START WITH 1

MINVALUE 1

NO MAXVALUE

CACHE 1;

--------Sequence for Bill Id

CREATE SEQUENCE Bill\_Id\_seq

INCREMENT BY 1

START WITH 1

MINVALUE 1

NO MAXVALUE

CACHE 1;

--------Sequence for Prescription Id

CREATE SEQUENCE Pres\_Id\_seq

INCREMENT BY 1

START WITH 1

MINVALUE 1

NO MAXVALUE

CACHE 1;

/\* 14 SQL Queries \*/

-- Q1. Select all columns and all rows from one table

SELECT \* FROM Doctor;

SELECT \* FROM Doctor;

SELECT \* FROM Patient;

SELECT \* FROM Appointment;

SELECT \* FROM Prescription;

SELECT \* FROM Bill;

-- Q2. Select five columns and all rows from one table

SELECT

Doc\_id,

Doc\_Name,

Doc\_Specialisation,

Doc\_Shift,

Doc\_Professional\_Experience FROM  Doctor;

SELECT

P\_ID,

P\_Name,

P\_Gender,

P\_Address,

P\_Phone

FROM

Patient;

SELECT

Apt\_ID,

Doc\_id,

P\_ID,

Apt\_Date,

Apt\_Time,

Apt\_Type

FROM

Appointment;

SELECT

Pres\_ID,

Doc\_ID,

P\_ID,

Pres\_Date,

Pres\_Validity

FROM

Prescription;

SELECT

Bill\_Id,

P\_ID,

Bill\_Date,

Bill\_Amt,

Bill\_Status

FROM

Bill;

-- Q3. Select all columns from all rows from one view

CREATE VIEW

HealthcareView AS

SELECT

A.Apt\_ID,

A.Doc\_id AS app\_Doc\_ID,

A.P\_ID AS app\_P\_ID,

A.Apt\_Date,

A.Apt\_Time,

A.Apt\_Type,

A.Apt\_Room ,

D.Doc\_id,

D.Doc\_Name,

D.Doc\_Specialisation,

D.Doc\_Shift,

D.Doc\_Professional\_Experience,

P.P\_ID AS patient\_ID,

P.P\_Name,

PP.\_Gender,

P.P\_Address,

P.P\_Phone ,

PR.Pres\_ID,

PR.Doc\_ID AS

Prescription\_DOC\_ID,

PR.P\_ID AS prescription\_patient\_id,

PR.Pres\_Date,

PR.Pres\_Validity,

B.Bill\_Id,

B.P\_ID AS bill\_Patient\_id,

B.Bill\_Date,

B.Bill\_Amt,

B.Bill\_Status

FROM

Appointment A

JOIN

Doctor d on d.doc\_id = A.doc\_id

JOIN

Patient P on P.P\_ID = A.P\_ID and

JOIN

Prescription PR on

(PR.doc\_id = A.doc\_id

AND

PR.p\_id = A.doc\_id

AND

PR. Pres\_Date = A. Apt\_Date)

JOIN

bill B

ON

(B. P\_ID = A.P\_ID AND B.Bill\_Date = A. Apt\_Date);

SELECT \* FROM HealthcareView;

-- Q4. Using a join on 2 tables, select all columns and all rows

SELECT \*

FROM Bill B

Join Patient P ON P.P\_ID = B.P\_ID;

-- Q5. Select and order data retrieved from one table

SELECT

Doc\_id,

Doc\_Name,

Doc\_Specialisation,

Doc\_Shift,

Doc\_Professional\_Experience

FROM

Doctor

ORDER BY Doc\_Name ASC;

-- Q6. Using a join on 3 tables, select 5 columns from the 3 tables. Use syntax that would limit the output to 10 rows

SELECT

A.Apt\_ID,

D.Doc\_id,

P.P\_ID,

A.Apt\_Date,

A.Apt\_Time

FROM

Appointment A

JOIN

doctor D

ON

D.DOC\_id = A.doc\_id

JOIN

Patient P

ON

P.P\_ID = A.P\_ID

LIMIT 10;

-- Q7. Select distinct rows using joins on 3 tables

SELECT DISTINCT \*

FROM Prescription P

JOIN Doctor D ON D.Doc\_ID = P.Doc\_ID

JOIN Patient PA on PA.P\_ID = P.P\_ID

-- Q8. Use GROUP BY and HAVING in a select statement using one or more tables

SELECT

D.Doc\_ID,

D.Doc\_Name,

Count(D.Doc\_ID)

FROM Doctor D

INNER JOIN

Patient P

ON

P.P\_ID = D.Doc\_ID

GROUP BY D.Doc\_ID,

D.Doc\_Name

HAVING D.Doc\_ID>0

ORDER by D.Doc\_ID;

-- Q9. Use IN clause to select data from one or more tables

SELECT \*

FROM Doctor

Where Doc\_ID IN (1,3,5,7,9,11);

-- Q10. Select length of one column from one table (use LENGTH function)

SELECT LENGTH(Doc\_ID)

FROM Doctor;

-- Q11. Delete one record from one table. Use select statements to demonstrate the table contents before and after the DELETE statement. Make sure you use ROLLBACK afterwards so that the data will not be physically removed

SELECT \* FROM bill ;

-- Start a transaction

BEGIN TRANSACTION;

DELETE FROM bill

WHERE bill\_ID = 1;

Rollback;

SELECT \*

FROM

bill;

-- Q12. Update one record from one table. Use select statements to demonstrate the table contents before and after the UPDATE statement. Make sure you use ROLLBACK afterwards so that the data will not be physically removed

SELECT \* FROM doctor;

-- Start a transaction

BEGIN TRANSACTION;

-- Update the row with Doc\_id = 1

UPDATE doctor

SET

Doc\_Name = 'Dr. Avnish',

Doc\_Specialisation = 'Ophthalmologist',

Doc\_Shift = '2023-11-03',

Doc\_Professional\_Experience = 3

WHERE Doc\_id = 1;

-- Display the state of the "doctor" table after the update

SELECT \* FROM doctor;

-- Rollback the changes made during the transaction

ROLLBACK;

**Additional Advance queries**

Query 1 : Details of patients with total unpaid and paid bills who have an appointment with a doctor's name starting with M.

---List of Patients who has an appointment with a doctor's name starting with M.

WITH Patient\_list as

( SELECT

DISTINCT P.P\_ID,

P.P\_Name,

P.P\_Gender,

P.P\_Address,

P.P\_Phone

FROM

appointment A

JOIN

patient P

ON

P.P\_ID = A.P\_ID

JOIN

doctor D

ON

D.Doc\_ID = A.Doc\_ID

WHERE Doc\_name LIKE 'Dr. M%')

SELECT

DISTINCT

b.P\_ID,

PL.P\_Name,

PL.P\_Gender,

PL.P\_Address,

PL.P\_Phone,

SUM( CASE WHEN b.bill\_status = 'Paid'

THEN

b.Bill\_Amt else 0 end ) AS total\_paid\_amount,

SUM(CASE WHEN b.bill\_status = 'Unpaid' then b.Bill\_Amt else 0 end)

AS total\_unpaid\_amount

FROM bill b

JOIN Patient\_list PL ON PL.P\_ID = B.P\_ID

GROUP BY

b.P\_ID,

PL.P\_Name,

PL.P\_Gender,

PL.P\_Address,

PL.P\_Phone;

Query 2 : Details of doctors who have appointments with patients with maximum and minimum bills.

WITH patient\_total\_bill as

(SELECT

DISTINCT P\_ID,

SUM(bill\_amount) as total\_bill ,

FROM bill

Group by P\_ID ),

max\_min\_bill as

(SELECT

MAX(total\_bill) as max\_bill,

MIN(total\_bill) as min\_bill

FROM patient\_total\_bill),

Patient\_list AS

(SELECT

DISTINCT P\_ID

FROM patient\_total\_\_bill

WHERE (total\_bill = (select max\_bill from\_max\_min\_bill)

OR

(total\_bill = (select min\_bill from\_max\_min\_bill)))

SELECT

DISTINCT D.Doc\_id,

D.Doc\_Name,

D.Doc\_Specialisation,

D.Doc\_Shift,

D.Doc\_Professional\_Experience

FROM Appointment A

JOIN Doctor D on D.doc\_id = A.doc\_id

WHERE A.P\_ID in (select distinct P\_ID from patient\_list);