**Term Project**

**Team # - Project Title**

**1. Project Description**

Describe the project overview in two to three paragraphs.

**1.1 Project Overview and Statement of Proposal**

**Our database project will focus on developing a comprehensive eye clinic management system. The goal is to create a solution that streamlines various administrative and clinical tasks within an eye care facility. This includes managing patient records, handling appointment scheduling, and overseeing billing and payments. By identifying these core functionalities, we aim to design a system that not only improves workflow efficiency but also enhances the overall patient experience. A clear understanding of how these components interact will guide the structure and logic of the database.**

**We propose to ensure the system is both efficient and secure, I am researching best practices in database design. This involves using normalization techniques to reduce redundancy, setting up proper relationships between tables, and implementing data validation rules. Additionally, I’m considering access controls and backup strategies to protect sensitive patient information and maintain data integrity. The structure will support scalability, making it easier to expand the system with new features if needed.**

**For implementation, I will be using MySQL as the database management system and PHP for server-side scripting. MySQL will handle all the data storage needs, including tables for patients, appointments, billing, doctors, and treatments. PHP will serve as the bridge between the database and the front-end user interface, allowing for data retrieval, insertion, and updates. Through PHP, users will be able to log in, manage appointments, view patient histories, and generate billing reports. Together, MySQL and PHP will form the backbone of a dynamic and responsive web-based application for the clinic.**

**Statement of Proposal:** *We propose to …*

**1.2 Project Scope and Objectives**

Define the scope of the project and the objectives to be satisfied by this project

**Together, PHP and MySQL provide a powerful backend for developing a reliable, interactive, and secure web-based clinic management system. Each business function is supported by its own set of scripts and database interactions—for example, the new patient registration form will insert user details into the appropriate patient table, while the login system will validate credentials against stored hashed passwords to ensure secure access. Appointment scheduling will check for available time slots in real-time, reducing the chances of double-booking, and update features will allow patients or clinic staff to modify contact details or medical history as needed.**

**On the clinical side, doctors will be able to securely log in and record examination results, which are saved directly into the system and linked to the patient's medical history. Prescriptions issued can be stored, accessed later, or sent digitally to the clinic’s eyewear department to order contact lenses or glasses. Billing records will be generated based on services provided, and patients will be able to process payments online using secure payment integration. Additionally, the system will include a feature to send appointment reminders automatically, helping reduce no-shows and keeping the clinic running smoothly.**

**In short, this project will demonstrate how web technologies like PHP and MySQL can be used to build a real-world application that addresses both administrative and clinical needs in a healthcare setting. By combining thoughtful database design with dynamic web functionality, the system aims to improve efficiency, reduce paperwork, and create a more seamless experience for both patients and staff at the eye clinic.**

**2. Requirement Collection and Analysis**

* + In this section, you should describe what data needs to be stored and what application will be built on top of it. Basically, it should cover functional requirement analysis and data requirement analysis.
  + In functional requirements, specify at least 10 business functions ( daily operations such as creating account, login, update profile, etc.) which you are expected to implement in the proposal application. These operations will be applied to the database, including both retrieval and updates.

**Some specific business functions that pertain to the eye clinic include a wide range of services that span clinical care, administrative operations, and customer engagement. For instance, patient consultation and diagnosis involves recording detailed patient history, documenting symptoms, and logging diagnostic results from eye exams or vision tests. The system will allow doctors to enter and update these records in real time, ensuring continuity of care and accurate medical documentation.**

**The system will also support optical retail services, enabling patients to browse and order prescription glasses or contact lenses based on their latest prescriptions. Inventory levels of frames, lenses, and accessories will be managed in real time to ensure stock availability. For surgical procedures, the database will track pre-operative assessments, schedule surgeries, store surgical notes, and follow-up records, providing a complete surgical workflow within the platform.**

**Preventative eye care and screenings will be managed through routine exam scheduling, alerts for due screenings (such as glaucoma or diabetic retinopathy checks), and documentation of results. In the realm of billing and insurance processing, the system will generate invoices based on services rendered, calculate co-pays, and manage insurance claims submission. This ensures both transparency for patients and streamlined reimbursements for the clinic.**

**To support growth and retention, marketing and patient outreach features will allow for sending newsletters, promotional offers, or educational content about eye health. Appointment reminders, follow-up messages, and personalized communications will enhance patient engagement. Inventory management will track clinical and retail supplies, automate reordering processes, and provide alerts for low stock.**

**Beyond clinical functions, the system will also support human resource management, such as storing staff credentials, managing work schedules, tracking attendance, and handling payroll records. Compliance and legal management ensures that the system operates within healthcare regulations such as HIPAA or local privacy laws, including audit trails, access control, and consent management for sensitive patient data.**

**Lastly, the platform will include capabilities for telemedicine and online consultations. Patients will be able to schedule and attend virtual appointments with doctors via secure video conferencing tools integrated with the system. Doctors can share test results, update prescriptions, and provide care remotely, making services more accessible, especially for those in rural or underserved areas.**

1. **New Patient Registration**
   * **Input data:** ID, name, date of birth, phone, email, address, gender, SSN, password, medical history, contactMethod
   * **Possible output:** If the provided email address is unique and the password is valid, the new patient will be recorded. Otherwise, an error message will be provided.
   * **Anticipated frequency:** 10 per day
   * **Anticipated performance goal:** 0.003 seconds
2. **Patient Login**
   * **Input data:** Email and password
   * **Possible output:** If the provided email address and password are valid, the patient will be logged in successfully. Otherwise, an error message will be provided.
   * **Anticipated frequency:** 100 per day
   * **Anticipated performance goal:** 0.002 seconds
3. **Make Appointment**
   * **Input data:** appID, date, time, patient ID,
   * **Possible output:** If the date and time are valid, the patient will be able to book an appointment. Otherwise, an error message will be provided.
   * **Anticipated frequency:** 500 per day
   * **Anticipated performance goal:** 2 seconds
4. **Update Patient Information**
   * **Input data:** ID, name, date of birth, phone, email, address, gender, SSN, password, medical history, contactMethod
   * **Possible output:** If the patient ID exists, the provided details will be updated successfully. SSN and patient’s ID won’t be updated. Otherwise, an error message will be provided.
   * **Anticipated frequency:** 20 per day
   * **Anticipated performance goal:** 0.004 seconds
5. **Doctor Login**
   * **Input data:** Email, password
   * **Possible output:** If the provided email and password are valid, the doctor will be logged in successfully. Otherwise, an error message will be provided.
   * **Anticipated frequency:** 50 per day
   * **Anticipated performance goal:** 0.002 seconds
6. **Record Eye Examination Results**
   * **Input data:** testResults, date, notes, prescription, docID, diagnosis, patient\_ID, exam\_ID
   * **Possible output:** If the patient and doctor IDs are valid, the examination results will be saved successfully. Otherwise, an error message will be provided.
   * **Anticipated frequency:** 100 per day
   * **Anticipated performance goal:** 1 second
7. **Update Patient Medical History**
   * **Input data:** Patient ID, new medical history details (e.g., diagnosis, treatment, medications)
   * **Possible output:** If the patient ID is valid, the medical history will be updated successfully with the new details. If the patient ID is invalid, an error message will be provided.
   * **Anticipated frequency:** 100 per day
   * **Anticipated performance goal:** 0.003 seconds
8. **Bill Payment Processing**
   * **Input data:** invoiceNumber, patient\_ID, transactionDate, amount, paymentMethod
   * **Possible output:** If the payment details are valid, the transaction will be processed, and a receipt will be generated. Otherwise, an error message will be provided.
   * **Anticipated frequency:** 200 per day
   * **Anticipated performance goal:** 1 second
9. **Order Contact Lenses/Glasses**
   * **Input data:**
     1. Orders table: patient\_ID, totalAmount, orderDate, order\_ID
     2. Items table: productType, qtyInStock, itemName, item\_ID, salePrice, order\_ID
   * **Possible output:** If the item ordered is in stock and order\_ID is valid, the order will be placed successfully. Otherwise, an error message will be provided.
   * **Anticipated frequency:** 300 per day
   * **Anticipated performance goal:** 2 seconds
10. **Update Appointment Time**
    * **Input data:** Appointment ID, new appointment date, new appointment time
    * **Possible output:** If the appointment ID is valid, the appointment date and time will be updated successfully. If the appointment ID is invalid or the new time conflicts with existing appointments, an error message will be provided.
    * **Anticipated frequency:** 50 per day
    * **Anticipated performance goal:** 0.005 seconds

**3. Conceptual Database Design**

* + **E-R Diagram**

Construct the E-R diagram according to the system specification above. The proposed database application should have 8 -12 entity sets and 6-10 relationship sets. All entity sets should be somehow directly or indirectly related.

A diagram of a flowchart

AI-generated content may be incorrect.



* + **E-R diagram Narrative**

Interpret the diagram by specifying the properties of each entity type and relationship types.

**This ER diagram represents a Healthcare Management System, outlining the relationships between various entities involved in patient care, appointments, billing, prescriptions, and medical history.**

**Entities and Their Attributes(put bullet points in full sentences)**

1. **Patient**
   * **A patient has several attributes, including their SSN (Social Security Number), name, password, gender, date of birth, phone number, email, address, medical history, contactMethod and patient ID. Each patient is uniquely identified by their SSN, which is linked to their medical history.**
2. **Doctor**
   * **A doctor has several attributes, including their unique docID, name, email, and password. Each doctor is identified by their unique docID and has the ability to write prescriptions for patients.**
3. **Appointment**
   * **An appointment has attributes such as appID, patient ID, date, and time. Patients schedule appointments, and each appointment is attended to by a doctor. Each appointment is specified by it’s unique appID.**
4. **Results**
   * **Results include attributes like date, test results, notes, diagnosis, exam\_ID, doctor ID, patient ID, and prescription. These results document the findings of a patient's medical examination, which are recorded by the doctor. Each exam result is specified by it’s unique exam\_ID.**
5. **Orders**
   * **An order has attributes such as patient ID, order ID, total amount, and order date. Patients place orders for prescribed medicines based on their doctor’s recommendations. Each order is specified by it’s unique order\_ID.**
6. **BillPayment**
   * **Bill payment has attributes such as invoice number, patient ID, transaction date, amount, and payment method. Patients are billed for medical services, and payments for these services are recorded. Each bill payment is specified by it’s unique invoice number.**
7. **Prescription**
   * **The Prescription table contains attributes such as prescription ID, prescription details, expiration date, date issued, exam ID, and patient ID. Prescriptions are issued to patients based on medical examinations, and each prescription records details about the prescribed treatment and its validity period. Every prescription is uniquely identified by its prescription ID. The exam ID links the prescription to the corresponding examination results, and the patient ID identifies the patient receiving the prescription.**
8. **Items**
   * **The Items table includes attributes such as product type, quantity in stock, item name, item ID, sale price, and order ID. It stores information about individual products available for sale, including their classification, availability, and pricing. Each item is uniquely identified by its item ID. The order ID associates the item with a specific customer order, indicating which items were included in which transactions.**

**Relationships**

1. **WorkedBy is a relationship between the Doctor and Appointment entities. Each appointment is handled by exactly one doctor, but a doctor can work on many appointments, making this a one-to-many (1:N) relationship. Participation is total on the Appointment side, meaning every appointment must be handled by a doctor, and partial on the Doctor side, as not every doctor is necessarily assigned to an appointment.**
2. **SchedulesAn links the Patient and Appointment entities. A patient can schedule multiple appointments, while each appointment is scheduled by one patient, forming a one-to-many (1:N) relationship. Participation is total on the Appointment side, indicating that every appointment must be scheduled by a patient, and partial on the Patient side, as not all patients must schedule appointments.**
3. **purchasedBy links Patient and Orders, where each order is placed by one patient, but a patient can place many orders, forming a one-to-many (1:N) relationship. Participation is total on the Orders side and partial on the Patient side.**
4. **placedBy is a one-to-many (1:N) relationship between Patient and billPayment. Each bill payment is made by one patient, while a patient can make many bill payments. The relationship shows total participation from billPayment, indicating every payment must be linked to a patient, and partial participation from Patient.**
5. **The Includes relationship connects the Orders and Items entities. An order can include multiple items, and each item can be part of multiple orders, making this a many-to-many (M:N) relationship. This relationship also has an attribute, quantity, which specifies the number of units of each item included in an order. Participation is partial on both sides, as not every item must be included in an order and not every order must include every item.**
6. **Issues is a many-to-many (M:N) relationship between Patient and Prescription. A patient can be issued multiple prescriptions, and a single prescription can be associated with multiple patients (likely due to data normalization or reuse scenarios). Participation is total on the Prescription side, meaning every prescription must be linked to at least one patient, and partial on the Patient side, since not every patient needs to have prescriptions issued.**
7. **The Has relationship links Exam and Results in a one-to-one (1:1) relationship. Each exam generates exactly one result, and each result is associated with exactly one exam. Participation is total on both sides, indicating that every exam must have a result and every result must belong to a specific exam.**
8. **Exam–Patient:  
   A Patient can undergo multiple exams, but each exam is associated with exactly one patient. This forms a one-to-many (1:N) relationship.**
   1. **: Total on the Exam side—every exam must be linked to a patient. Partial on the Patient side, as not every patient needs to have an exam.**
9. **Exam–Doctor:  
   A Doctor can perform multiple exams, but each exam is associated with exactly one doctor. This forms a many-to-one (N:1) relationship.**
   1. **Participation: Total on the Exam side—every exam must have a doctor. Partial on the Doctor side, as not every doctor must conduct an exam.**
10. **Exam–Results:  
    Each Result is linked to exactly one exam, and an exam can produce only one result. This forms a one-to-one (1:1) relationship.**
    1. **Participation: Total on the Results side—every result must be linked to an exam. Partial on the Exam side, as not every exam produces a result.**

**4. ER Model Mapping to Relational Database**

* 1. **SQL statements for Entities and Relationships Creation**

Make sure the data type and data size are appropriate for each attribute. For example, data type TINYINT is better than INT for the attribute age. Do not forget to include a suitable Primary Key and/or Foreign Key in your definition of each table.

CREATE TABLE Patient(

patient\_ID CHAR(9),

ssn CHAR(9),

password VARCHAR(40),

gender CHAR(1),

dateOfBirth DATE,

medHistory VARCHAR(100),

address VARCHAR(100),

phoneNumber CHAR(11),

email VARCHAR(50),

name VARCHAR(50),

contactMethod VARCHAR(20),

PRIMARY KEY (patient\_ID),

UNIQUE(ssn)

);

CREATE TABLE AppointmentSchedule(

appID CHAR(9),

patient\_ID CHAR(9),

date DATE,

time TIMESTAMP,

PRIMARY KEY(appID),

FOREIGN KEY(patient\_ID) REFERENCES Patient(patient\_ID)

);

CREATE TABLE DoctorWork(

docID CHAR(9),

name VARCHAR(50),

email VARCHAR(50),

password VARCHAR(40),

appID CHAR(9),

PRIMARY KEY(docID),

FOREIGN KEY(appID) REFERENCES AppointmentSchedule(appID)

);

CREATE TABLE Results(

testResults VARCHAR(100),

date DATE,

notes VARCHAR(300),

prescription VARCHAR(100),

docID CHAR(9),

diagnosis VARCHAR(50),

patient\_ID CHAR(9),

exam\_ID CHAR(9),

PRIMARY KEY(exam\_ID),

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

FOREIGN KEY(docID) REFERENCES DoctorWork(docID)

);

CREATE TABLE Exam(

patient\_ID CHAR(9),

exam\_ID CHAR(9),

docID CHAR(9),

PRIMARY KEY(patient\_ID, exam\_ID, docID),

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

FOREIGN KEY(exam\_ID) REFERENCES Results(exam\_ID),

FOREIGN KEY(docID) REFERENCES DoctorWork(docID)

);

CREATE TABLE Prescription(

prescription\_ID CHAR(9),

prescriptionDetails VARCHAR(100),

expirationDate DATE,

dateIssued DATE,

exam\_ID CHAR(9),

patient\_ID CHAR(9),

PRIMARY KEY(prescription\_ID),

FOREIGN KEY(exam\_ID) REFERENCES Results(exam\_ID),

FOREIGN KEY(patient\_ID) REFERENCES Patient(patient\_ID)

);

CREATE TABLE Issues(

exam\_ID CHAR(9),

prescription\_ID CHAR(9),

PRIMARY KEY(exam\_ID, prescription\_ID),

FOREIGN KEY(exam\_ID) REFERENCES Results(exam\_ID),

FOREIGN KEY(prescription\_ID) REFERENCES Prescription(prescription\_ID)

);

CREATE TABLE Orders(

patient\_ID CHAR(9),

totalAmount INT,

orderDate DATE,

order\_ID CHAR(9),

PRIMARY KEY(order\_ID),

FOREIGN KEY(patient\_ID) REFERENCES Patient(patient\_ID)

);

CREATE TABLE Items(

productType VARCHAR(50),

qtyInStock INT,

itemName VARCHAR(50),

item\_ID CHAR(9),

salePrice DECIMAL(6,2),

order\_ID CHAR(9),

PRIMARY KEY(item\_ID),

FOREIGN KEY(order\_ID) REFERENCES Orders(order\_ID)

);

CREATE TABLE Includes(

quantity INT,

order\_ID CHAR(9),

item\_ID CHAR(9),

PRIMARY KEY(order\_ID, item\_ID),

FOREIGN KEY(order\_ID) REFERENCES Orders(order\_ID),

FOREIGN KEY(item\_ID) REFERENCES Items(item\_ID)

);

CREATE TABLE billPaymentPlaced(

invoiceNumber VARCHAR(10),

patient\_ID CHAR(9),

transactionDate DATE,

amount DECIMAL(6,2),

paymentMethod VARCHAR(30),

PRIMARY Key(invoiceNumber),

FOREIGN KEY(patient\_ID) REFERENCES Patient(patient\_ID)

);

* 1. **SQL for loading data for each relation**

Make sure to include one sample record from each table here.

INSERT INTO Patient (patient\_ID, ssn, password, gender, dateOfBirth, medHistory, address, phoneNumber, email, name, contactMethod)

VALUES ('P12345678', '123-45-6789', 'password123', 'M', '1985-07-12', 'Asthma, High blood pressure', '123 Elm Street, Springfield', '555-123-4567', 'john.doe@example.com', 'John Doe', 'Phone');

INSERT INTO DoctorWork (docID, name, email, password, appID)

VALUES ('D12345678', 'Dr. Sarah Smith', 'sarah.smith@clinic.com', 'doctorpass123', 'A12345678');

INSERT INTO AppointmentSchedule (appID, patient\_ID, date, time)

VALUES ('A12345678', 'P12345678', '2025-04-20', '2025-04-20 10:30:00');

INSERT INTO Results (testResults, date, notes, prescription, docID, diagnosis, patient\_ID, exam\_ID)

VALUES ('Blood Pressure: 140/90, Heart rate: 75', '2025-04-20', 'Patient showed signs of hypertension.', 'Beta-blocker prescribed', 'D12345678', 'Hypertension', 'P12345678', 'E12345678');

INSERT INTO Exam (patient\_ID, exam\_ID, docID)

VALUES ('P12345678', 'E12345678', 'D12345678');

INSERT INTO Prescription (prescription\_ID, prescriptionDetails, expirationDate, dateIssued, exam\_ID, patient\_ID)

VALUES ('R12345678', 'Beta-blocker 50mg', '2026-04-20', '2025-04-20', 'E12345678', 'P12345678');

INSERT INTO Issues (exam\_ID, prescription\_ID)

VALUES ('E12345678', 'R12345678');

INSERT INTO Orders (patient\_ID, totalAmount, orderDate, order\_ID)

VALUES ('P12345678', 150, '2025-04-21', 'O12345678');

INSERT INTO Items (productType, qtyInStock, itemName, item\_ID, salePrice, order\_ID)

VALUES ('Medication', 100, 'Beta-blocker 50mg', 'I12345678', 50.00, 'O12345678');

INSERT INTO Includes (quantity, order\_ID, item\_ID)

VALUES (2, 'O12345678', 'I12345678');

INSERT INTO billPaymentPlaced (invoiceNumber, patient\_ID, transactionDate, amount, paymentMethod)

VALUES ('INV1234567', 'P12345678', '2025-04-22', 100.00, 'Credit Card');

* 1. **SQL statements for 10 business functions**

**--1: New Patient Registration**

**INSERT INTO Patient (patient\_ID, ssn, password, gender, dateOfBirth, medHistory, address, phoneNumber, email, name)**

**VALUES ('P12345674', '123-45-6789', 'securepassword123', 'Male', '1985-06-15', 'No previous medical history', '123 Main St, Springfield, IL', '555-1234', 'johndoe@email.com', 'John Doe');**

**--2: Patient Login**

**SELECT patient\_ID**

**FROM Patient**

**WHERE email = 'johndoe@email.com' AND password = 'securepassword123';**

**--3: Make Appointment**

**INSERT INTO AppointmentSchedule (appID, patient\_ID, date, time)**

**VALUES ('A1234567', 'P12345678', '2025-04-10', '10:00 AM');**

**--4: Update Patient Information**

**UPDATE Patient**

**SET password = 'newpassword123', gender = 'Male', dateOfBirth = '1985-06-15',**

**medHistory = 'No previous medical history', address = '123 Main St, Springfield, IL', phoneNumber = '555-5678',**

**email = 'newemail@email.com', name = 'John Doe', contactMethod = 'Phone'**

**WHERE ssn = '123-45-6789';**

**--5: Doctor Login**

**SELECT docID, name, email, password**

**FROM DoctorWork**

**WHERE email = 'asmith1@example.com' AND password = 'pass1234';**

**--6: Record Eye Examination Results**

**INSERT INTO Results (testResults, date, notes, prescription, docID, diagnosis, patient\_ID, exam\_ID)**

**VALUES ('Eye pressure within normal range, no significant findings', '2025-04-10', 'Patient reported no discomfort during the exam. Eye health stable.', 'Eye drops prescribed for dryness', 'DOC000014', 'Normal eye health', 'P12345697', 'EX000041');**

**--7: Update Patient Medical History**

**UPDATE Patient**

**SET medHistory = 'Patient has a history of high blood pressure and seasonal allergies.'**

**WHERE patient\_ID = 'P12345678';**

**--8: Bill Payment Processing**

**INSERT INTO billPaymentPlaced (invoiceNumber, patient\_ID, transactionDate, amount, paymentMethod)**

**VALUES ('INV000041', 'P12345678', '2025-04-10', 250.00, 'Credit Card');**

**--9: Order Contact Lenses/Glasses**

**--Patient places order**

**INSERT INTO Orders (patient\_ID, totalAmount, orderDate, order\_ID)**

**VALUES ('P12345717', 180, '2025-04-30', 'ORD000041');**

**--Orders is received from items**

**INSERT INTO Items (productType, qtyInStock, itemName, item\_ID, salePrice, order\_ID)**

**VALUES ('Eyewear', 50, 'Polarized Sunglasses', 'ITM000041', 45.99, 'ORD000041');**

**--10: Update Appointment Time**

**UPDATE AppointmentSchedule**

**SET date = '2025-04-30', time = '14:00:00'**

**WHERE appID = 'A123456';**

1. **Normalization**

**Table: Patient**

**Attributes:** Patient(patient\_ID, ssn, password, gender, dateOfBirth, medHistory, address, phoneNumber, email, name, contactMethod)

* **Candidate keys**: {(patient\_ID), (ssn)}
* **FDs**:
  + patient\_ID → ssn, password, gender, dateOfBirth, medHistory, address, phoneNumber, email, name, contactMethod
  + ssn → patient\_ID, password, gender, dateOfBirth, medHistory, address, phoneNumber, email, name, contactMethod
* **Prime attributes**: patient\_ID, ssn
* **Non-prime attributes**: password, gender, dateOfBirth, medHistory, address, phoneNumber, email, name, contactMethod
* **Normal Form**: This table is in **BCNF** because all non-trivial FDs have superkeys on the left-hand side.

**Table: AppointmentSchedule**

**Attributes:** AppointmentSchedule(appID, patient\_ID, date, time)

* **Candidate key**: {(appID)}
* **FDs**:
  + appID → patient\_ID, date, time
* **Prime attribute**: appID
* **Non-prime attributes**: patient\_ID, date, time
* **Normal Form**: **BCNF**, as appID is a superkey and determines all other attributes.

**Table: DoctorWork**

**Attributes:** DoctorWork(docID, name, email, password, appID)

* **Candidate key**: {(docID)}
* **FDs**:
  + docID → name, email, password, appID
* **Prime attribute**: docID
* **Non-prime attributes**: name, email, password, appID
* **Normal Form**: **BCNF**

**Table: Results**

**Attributes:** Results(testResults, date, notes, prescription, docID, diagnosis, patient\_ID, exam\_ID)

* **Candidate key**: {(exam\_ID)}
* **FDs**:
  + exam\_ID → testResults, date, notes, prescription, docID, diagnosis, patient\_ID
* **Prime attribute**: exam\_ID
* **Non-prime attributes**: testResults, date, notes, prescription, docID, diagnosis, patient\_ID
* **Normal Form**: **BCNF**, because exam\_ID (the primary key) determines all other attributes.

**Table: Exam**

**Attributes:** Exam(patient\_ID, exam\_ID, docID)

* **Candidate key**: {(patient\_ID, exam\_ID, docID)}
* **FDs**:
  + (patient\_ID, exam\_ID, docID) → {} (no non-trivial dependencies)
* **Prime attributes**: patient\_ID, exam\_ID, docID
* **Non-prime attributes**: *none*
* **Normal Form**: **BCNF**, as no non-trivial dependencies exist.

**Table: Prescription**

**Attributes:** Prescription(prescription\_ID, prescriptionDetails, expirationDate, dateIssued, exam\_ID, patient\_ID)

* **Candidate key**: {(prescription\_ID)}
* **FDs**:
  + prescription\_ID → prescriptionDetails, expirationDate, dateIssued, exam\_ID, patient\_ID
* **Prime attribute**: prescription\_ID
* **Non-prime attributes**: prescriptionDetails, expirationDate, dateIssued, exam\_ID, patient\_ID
* **Normal Form**: **BCNF**, as prescription\_ID is a superkey.

**Table: Issues**

**Attributes:** Issues(exam\_ID, prescription\_ID)

* **Candidate key**: {(exam\_ID, prescription\_ID)}
* **FDs**:
  + (exam\_ID, prescription\_ID) → {} (no non-trivial FDs)
* **Prime attributes**: exam\_ID, prescription\_ID
* **Non-prime attributes**: *none*
* **Normal Form**: **BCNF**

**Table: Orders**

**Attributes:** Orders(patient\_ID, totalAmount, orderDate, order\_ID)

* **Candidate key**: {(order\_ID)}
* **FDs**:
  + order\_ID → patient\_ID, totalAmount, orderDate
* **Prime attribute**: order\_ID
* **Non-prime attributes**: patient\_ID, totalAmount, orderDate
* **Normal Form**: **BCNF**, as order\_ID is a superkey.

**Table: Items**

**Attributes:** Items(productType, qtyInStock, itemName, item\_ID, salePrice, order\_ID)

* **Candidate key**: {(item\_ID)}
* **FDs**:
  + item\_ID → productType, qtyInStock, itemName, salePrice, order\_ID
* **Prime attribute**: item\_ID
* **Non-prime attributes**: productType, qtyInStock, itemName, salePrice, order\_ID
* **Normal Form**: **BCNF**, because the primary key item\_ID determines all other attributes.

**Table: Includes**

**Attributes:** Includes(quantity, order\_ID, item\_ID)

* **Candidate key**: {(order\_ID, item\_ID)}
* **FDs**:
  + (order\_ID, item\_ID) → quantity
* **Prime attributes**: order\_ID, item\_ID
* **Non-prime attribute**: quantity
* **Normal Form**: **BCNF**, as the composite key determines the non-prime attribute.

**Table: billPaymentPlaced**

**Attributes:** billPaymentPlaced(invoiceNumber, patient\_ID, transactionDate, amount, paymentMethod)

* **Candidate key**: {(invoiceNumber)}
* **FDs**:
  + invoiceNumber → patient\_ID, transactionDate, amount, paymentMethod
* **Prime attribute**: invoiceNumber
* **Non-prime attributes**: patient\_ID, transactionDate, amount, paymentMethod
* **Normal Form**: **BCNF**, since the FD is based on a superkey.

1. **GUI Design**

The scorpe of our application covers the following use cases(Patient Perspective):

1. The patient registers as a new user.
2. The patient logs in and views his/her profile.
3. The patient logs in and updates his/her profile.
4. The patient logs in and views his/her appointment.
5. The patient logs in and schedules his/her appointment.
6. The patient logs in and enters his/her bill payment information.
7. The patient logs in and places an order.
8. The patient logs in and views his/her orders they have placed.
9. The patient logs in and view their exam results.

The scorpe of our application covers the following use cases(Doctor Perspective):

1. The doctor registers as a new user.
2. The doctor logs in and views his/her profile.
3. The doctor logs in and view the appointments he/she has to work.
4. The doctor logs in and enters exam results.

To cover this basic functionality, we implement the following PHP files:

# For Patient:

1. The “home” page index.php for the patient to login and register.
2. The patientLogin.php page is for signing into the website.
3. The patientRegistration.php page is for registering a new patient.
4. The patientViewProfile.php page is for patients to view their profile.
5. The patientUpdateProfile.php page is for patients to update their profile.
6. The scheduleAppointment.php page is for patients to schedule appointments.
7. The billPayment.php page is for patients to enter bill payment information.
8. The makeOrder.php page is for the patients to place orders.
9. The viewOrders.php page is for the patients to be able view the orders they placed.
10. The viewExamResults.php is for the patients to be able view their exam results.
11. The patientLogout.php page is for the patients to log out.

A diagram of a patient login

AI-generated content may be incorrect.

# For Doctor:

1. The “home” page index.php for the doctor to login and register.
2. The doctorLogin.php page is for signing into the website.
3. The doctorRegistration.php page is for registering a new doctor.
4. The doctorViewProfile.php page is for doctors to view their profile.
5. The doctorViewAppointments.php page is for doctors to view the appointment they’ve been assigned to.
6. The writeExamResults.php page is for the doctors to write their exam results for their appointments.
7. The doctorLogout.php page is for the doctors to log out.

A diagram of a medical procedure

AI-generated content may be incorrect.

1. **User manual**

In the search bar, type in “<http://localhost/Project/>”, a page displaying the options patient registration, doctor registration, patient login, doctor log in will pop up.

A screenshot of a computer

AI-generated content may be incorrect.

If you’re a new patien you will register to create an account.

Patient Registration:

A screenshot of a computer

AI-generated content may be incorrect.

After a successful registration this will page will appear for a patient.

A screenshot of a computer

AI-generated content may be incorrect.

Then you will be able to log-in to your account after you register.

Patient Login:

A screenshot of a computer

AI-generated content may be incorrect.

As a patient once you enter a valid Patient ID and password you will be redirected to a portal where you will have options to view your profile, update your profile, view appointments, schedule appointments, enter bill payment information, place orders, view the orders that you have placed, and view exam results.

A screenshot of a computer

AI-generated content may be incorrect.

View Profile Page:

A screenshot of a computer

AI-generated content may be incorrect.

Update Profile Page:

A screenshot of a computer

AI-generated content may be incorrect.

After Successful Profile Update:

A screenshot of a computer

AI-generated content may be incorrect.

Doctor Registration:

A screenshot of a computer

AI-generated content may be incorrect.

View Appointments Page:

A screenshot of a computer

AI-generated content may be incorrect.

Schedule an Appointment Page:

A screenshot of a computer

AI-generated content may be incorrect.

After You Schedule an appointment you’ll be redirected to view appointments page with the new appointment:

A screenshot of a computer

AI-generated content may be incorrect.

Bill Payment Page:

A screenshot of a computer

AI-generated content may be incorrect.

Bill Payment Page after you successfully enter information:

A screenshot of a computer

AI-generated content may be incorrect.

Place Order Page:

A screenshot of a computer

AI-generated content may be incorrect.

After an order is placed this screen is displayed:

A screenshot of a computer screen

AI-generated content may be incorrect.

View Orders Page:

A screenshot of a computer

AI-generated content may be incorrect.

View Exam Results Page:

A screenshot of a computer

AI-generated content may be incorrect.

Once you sign out you’ll be redirected to the patient log in page:

A screenshot of a computer

AI-generated content may be incorrect.

If you’re a new doctor you will register to create an account.

Doctor Registration:

A screenshot of a computer

AI-generated content may be incorrect.

After a successful registration for a doctor this page will pop up:

A screenshot of a computer

AI-generated content may be incorrect.

Doctor Login:

A screenshot of a computer

AI-generated content may be incorrect.

As a doctor once you’ve entered a valid Doctor ID and password you will be redirected to a portal where you will have options to view your profile, view your appointments that you have been assigned to, and enter exam results for those appointments.

AsA screenshot of a computer

AI-generated content may be incorrect.

View Profile Page:

A screenshot of a computer

AI-generated content may be incorrect.

View Appointments(they’ve been assigned to) page:

A screenshot of a computer

AI-generated content may be incorrect.

Enter Exam Results Page:

A screenshot of a computer

AI-generated content may be incorrect.

After the doctor selects an appointment they’d like to submit exam results for they then can put in the notes/results:

A screenshot of a computer

AI-generated content may be incorrect.

After you submit exam results this page will show:

A screenshot of a computer

AI-generated content may be incorrect.

Once a doctor logs out they will be redirected to the doctor login page:

A screenshot of a computer

AI-generated content may be incorrect.

1. **Conclusion and Future Work**

All in all, the online eye clinic management system centralizes around giving patients a seamless way to manage their eye care. Our users can register a new account, securely log in, view and update their profile, schedule appointments, submit bill payments, place orders for eyewear or medication, and view their exam results—all through an intuitive web interface. Each of these business functions has been implemented using PHP and MySQL, allowing for dynamic data interaction and secure storage. The E-R model we designed illustrates all key relationships within the system, such as the linkage between patients, appointments, doctors, exam results, and orders.

Moving forward, we aim to further enhance the patient experience by expanding our features. One goal is to integrate telemedicine capabilities, allowing patients to attend virtual eye consultations with doctors directly through the platform. This would be especially useful for those in remote or underserved areas. Additionally, we plan to continue improving our GUI design to make it even more user-friendly, responsive across devices, and visually engaging. These advancements will ensure that our platform remains both functional and accessible, meeting the growing needs of modern healthcare delivery.