ABSTRACT

The cornerstone of the Hospital Management System (HMS) lies in its database management capabilities, powered by MySQL—a choice reflective of its reliability, scalability, and widespread industry support. The database was meticulously architected in the implementation phase to handle complex and sensitive healthcare data, including patient records, appointment schedules, treatment histories, and administrative operations.

A pivotal aspect of this phase was the design and normalization of the database schema. It was tailored to eliminate redundancy, enhance data integrity, and facilitate quick access. Tables were carefully structured to represent entities like patients, doctors, appointments, treatments, and prescriptions, with relationships modeled to reflect the intricacies of hospital workflows.

The creation of indexes, triggers, and stored procedures within the database optimized query performance and automated routine tasks, such as updating patient statuses or calculating billing amounts. This automation ensured a reduction in human error and improved the efficiency of data manipulation.

To secure sensitive data, rigorous security measures were integrated, including data encryption, role-based access controls, and audit trails to monitor data access and modifications. These security protocols ensure compliance with healthcare regulations like HIPAA, safeguarding patient confidentiality and trust.

The implementation phase also included the rigorous testing of database transactions to ensure ACID (Atomicity, Consistency, Isolation, Durability) properties, guaranteeing reliable and safe data operations even under concurrent access scenarios. This testing is crucial in a hospital setting where data integrity is paramount.

The integration of the MySQL database with PHP scripts facilitated a dynamic data exchange between the server and the front end. The scripts were crafted to perform complex queries, aggregate data for reports, and handle transactional operations, all while maintaining a seamless user experience on the front end.

In conclusion, the database implementation of the HMS was executed with an acute focus on creating a robust, secure, and efficient system that serves as the digital backbone of hospital operations, ready to handle the evolving needs of healthcare data management.

Table of Contents:

0. Pre-Illumination

This report delineates the implementation phase of our hospital management system application, chronicling the journey from conception to creating a fully functional database. Throughout the phases of this project, we have employed MySQL as our chosen database management system due to its robustness and widespread industry adoption.

The updated relational structure is listed in Part 1. Tables, all other structures, constraints, data types, and formats are created in Part 2 of the database-building process. Part 3 consists of designing and implementing query scenarios. Finally, our team has integrated the backend with PHP scripts bridging user interactions on an intuitive HTML, CSS, and JavaScript frontend, streamlining healthcare operations effectively.

1. Modified Relational Schema

In the updated version of the hospital management system, substantial modifications have been made to the relational schema to streamline the database and enhance data normalization. Key changes include:

1. Table Consolidation: Some tables have been merged to reduce redundancy. For example, patient contact details may now be included in the patient table rather than having a separate Patient_Contact table.

2. Attribute Refinement: Attributes across various tables have been refined to ensure they adhere to the third normal form (3NF). For instance, instead of having a Doctor_record_id in the Patient_medical_record table, this information might be integrated into the treatment_records table to eliminate transitive dependencies.

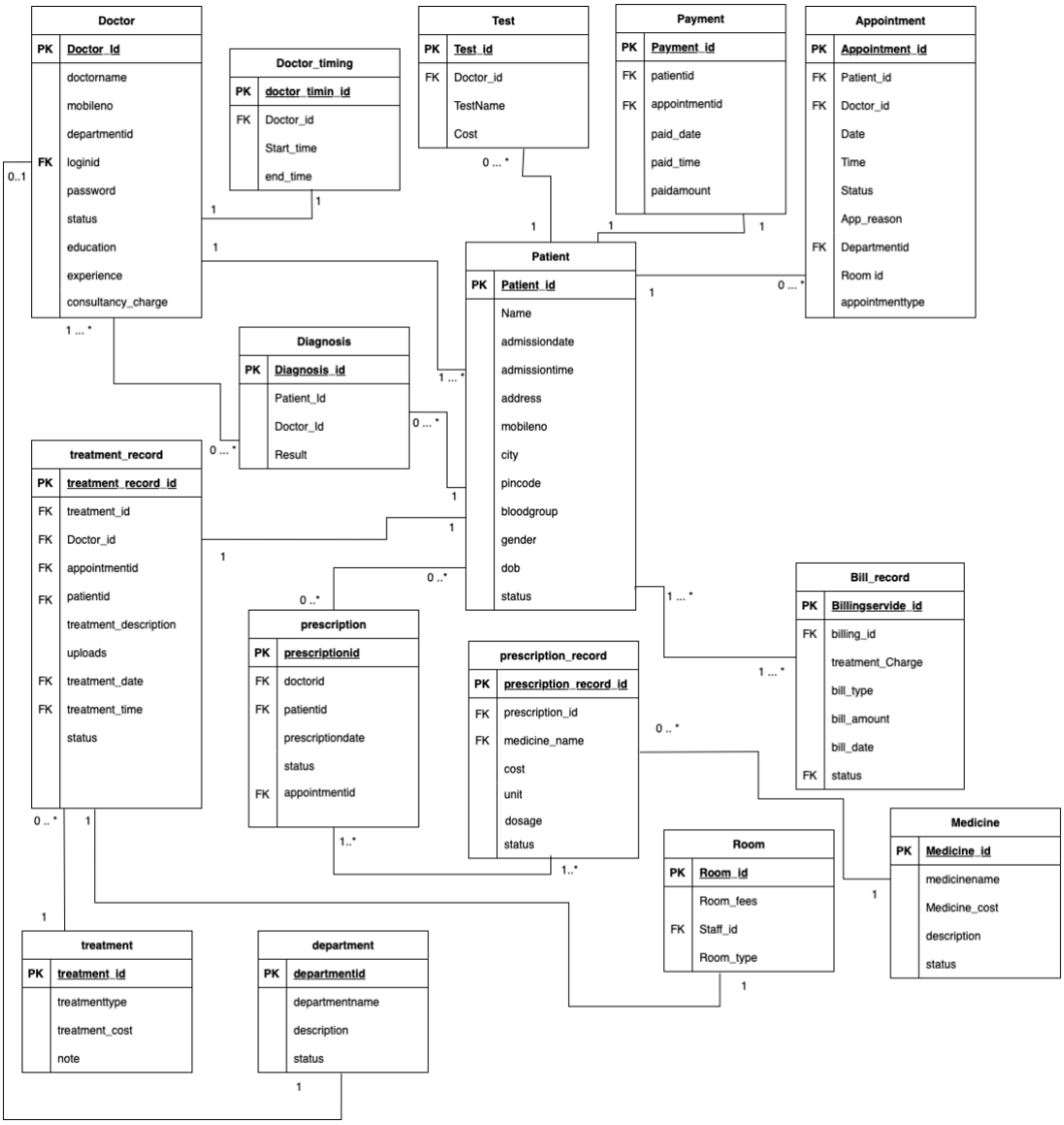
3. Data Type Standardization: Data types have been standardized across the database for consistency. This ensures that attributes representing similar data across tables maintain data integrity and enable more efficient queries.

4. New Tables for Specialized Data: Additional tables such as billing_records and prescription_records have been introduced to handle complex many-to-many relationships, ensuring that each table contains data related only to a single concept.

5. Inclusion of Status Fields: Tables like admin, appointment, treatment, and user now include a status field to track the active or inactive state of records, thus allowing for soft deletes and better management of data lifecycle.

These adjustments provide a more robust, efficient, and scalable database design, laying a solid foundation for the subsequent development phases. The updated relational schema is systematically arranged to reflect these changes, ensuring that the hospital management system is better equipped to handle complex queries and provide reliable and accurate data management.

The updated ER diagram is as follows:



2. Creation of Database with SQL Statements

2.1 Table Creation

Admin

```
  CREATE TABLE `admin` (
`adminid` int(10) NOT NULL,
`adminname` varchar(25) NOT NULL,
`loginid` varchar(25) NOT NULL,
`password` varchar(25) NOT NULL,
`status` varchar(10) NOT NULL,
`usertype` varchar(30) NOT NULL
     ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Appointment

```
CREATE TABLE `appointment` (
```

```
  `appointmentid` int(10) NOT NULL,
  `appointmenttype` varchar(25) DEFAULT 'Admin',
  `patientid` int(10) NOT NULL,
  `roomid` int(10) DEFAULT 0,
  `departmentid` int(10) NOT NULL,
  `appointmentdate` date NOT NULL,
  `appointmenttime` time NOT NULL,
  `doctorid` int(10) NOT NULL,
  `status` varchar(10) NOT NULL,
  `app_reason` text NOT NULL
        ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Billing

```
CREATE TABLE `billing` (
  `billingid` int(10) NOT NULL,
  `patientid` int(10) NOT NULL,
  `appointmentid` int(10) NOT NULL,
  `billingdate` date NOT NULL,
  `billingtime` time NOT NULL,
  `discount` float(10,2) NOT NULL,
  `taxamount` float(10,2) NOT NULL,
  `discountreason` text NOT NULL,
  `discharge_time` time NOT NULL,
  `discharge_date` date NOT NULL
            ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Billing_Records

```
CREATE TABLE `billing_records` (
  `billingservice_id` int(10) NOT NULL,
  `billingid` int(10) NOT NULL,
  `bill_type_id` int(10) NOT NULL COMMENT 'id of service charge or treatment
charge',
  `bill_type` varchar(250) NOT NULL,
  `bill_amount` float(10,2) NOT NULL,
  `bill_date` date NOT NULL,
  `status` varchar(10) NOT NULL
       ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Department

```
CREATE TABLE `department` (
  `departmentid` int(10) NOT NULL,
  `departmentname` varchar(100) NOT NULL,
  `description` text NOT NULL,
  `status` varchar(10) NOT NULL
        ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Doctor

```
CREATE TABLE `doctor` (
  `doctorid` int(10) NOT NULL,
  `doctorname` varchar(50) NOT NULL,
  `mobileno` varchar(15) NOT NULL,
  `departmentid` int(10) NOT NULL,
  `loginid` varchar(25) NOT NULL,
  `password` varchar(25) NOT NULL,
  `status` varchar(10) NOT NULL,
  `education` varchar(25) NOT NULL,
  `experience` float(11,1) NOT NULL,
  `consultancy_charge` float(10,2) NOT NULL
        ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Doctor_Timings

```
CREATE TABLE `doctor_timings` (
  `doctor_timings_id` int(10) NOT NULL,
  `doctorid` int(10) NOT NULL,
  `start_time` time NOT NULL,
  `end_time` time NOT NULL,
  `status` varchar(10) NOT NULL
        ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Medicine

```
CREATE TABLE `medicine` (
  `medicineid` int(10) NOT NULL,
  `medicinename` varchar(25) NOT NULL,
  `medicinecost` float(10,2) NOT NULL,
  `description` text NOT NULL,
  `status` varchar(10) NOT NULL
        ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Orders

```
CREATE TABLE `orders` (
  `orderid` int(10) NOT NULL,
  `patientid` int(10) NOT NULL,
  `doctorid` int(10) NOT NULL,
  `prescriptionid` int(10) NOT NULL,
  `orderdate` date NOT NULL,
  `deliverydate` date NOT NULL,
  `address` text NOT NULL,
  `mobileno` varchar(15) NOT NULL,
  `note` text NOT NULL,
  `status` varchar(10) NOT NULL,
  `payment_type` varchar(20) NOT NULL,
  `card_no` varchar(20) NOT NULL,
  `cvv_no` varchar(5) NOT NULL,
  `expdate` date NOT NULL,
  `card_holder` varchar(50) NOT NULL
        ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Patient

```
CREATE TABLE `patient` (
  `patientid` int(10) NOT NULL,
  `patientname` varchar(50) NOT NULL,
  `admissiondate` date NOT NULL,
  `admissiontime` time NOT NULL,
  `address` varchar(250) NOT NULL,
  `mobileno` varchar(15) NOT NULL,
  `city` varchar(25) NOT NULL,
  `pincode` varchar(20) NOT NULL DEFAULT '000000',
  `loginid` varchar(50) NOT NULL,
  `password` varchar(25) NOT NULL,
  `bloodgroup` varchar(20) NOT NULL,
  `gender` varchar(10) NOT NULL,
  `dob` date NOT NULL,
  `status` varchar(10) NOT NULL
        ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Payment

```
CREATE TABLE `payment` (
  `paymentid` int(10) NOT NULL,
```

```
  `patientid` int(10) NOT NULL,
  `appointmentid` int(10) NOT NULL,
  `paiddate` date NOT NULL,
  `paidtime` time NOT NULL,
  `paidamount` float(10,2) NOT NULL,
  `status` varchar(10) NOT NULL
        ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Prescription

```
CREATE TABLE `prescription` (
  `prescriptionid` int(10) NOT NULL,
  `doctorid` int(10) NOT NULL,
  `patientid` int(10) NOT NULL,
  `delivery_type` varchar(10) NOT NULL DEFAULT 'delivery' COMMENT 'Delivered
through appointment or online order',
  `delivery_id` int(10) NOT NULL DEFAULT 0 COMMENT 'appointmentid or
orderid',
  `prescriptiondate` date NOT NULL,
  `status` varchar(10) NOT NULL,
  `appointmentid` int(10) NOT NULL
        ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Prescription_Records

```
CREATE TABLE `prescription_records` (
  `prescription_record_id` int(10) NOT NULL,
  `prescription_id` int(10) NOT NULL,
  `medicine_name` varchar(25) NOT NULL,
  `cost` float(10,2) NOT NULL,
  `unit` int(10) NOT NULL,
  `dosage` varchar(25) NOT NULL,
  `status` varchar(10) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Room

```
CREATE TABLE `room` (
  `roomid` int(10) NOT NULL,
  `roomtype` varchar(25) NOT NULL,
  `roomno` int(10) NOT NULL,
  `noofbeds` int(10) NOT NULL,
  `room_tariff` float(10,2) NOT NULL,
  `status` varchar(10) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Treatment

```
CREATE TABLE `treatment` (
  `treatmentid` int(10) NOT NULL,
  `treatmenttype` varchar(25) NOT NULL,
  `treatment_cost` decimal(10,2) NOT NULL,
  `note` text NOT NULL,
  `status` varchar(10) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Treatment_Records

```
CREATE TABLE `treatment_records` (
  `treatment_records_id` int(10) NOT NULL,
  `treatmentid` int(10) NOT NULL,
  `appointmentid` int(10) NOT NULL,
  `patientid` int(10) NOT NULL,
```

```
  `doctorid` int(10) NOT NULL,
  `treatment_description` text NOT NULL,
  `uploads` varchar(100) NOT NULL,
  `treatment_date` date NOT NULL,
  `treatment_time` time NOT NULL,
  `status` varchar(10) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

User

```
CREATE TABLE `user` (
  `userid` int(11) NOT NULL,
  `loginname` varchar(50) NOT NULL,
  `password` varchar(10) NOT NULL,
  `patientname` varchar(50) NOT NULL,
  `mobileno` varchar(15) NOT NULL,
  `email` varchar(50) NOT NULL,
  `createddateandtime` datetime NOT NULL DEFAULT current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

2.2 A Database State

To ensure the database is populated for testing and development purposes, sample data was inserted into each table. The following records were added to each table, maintaining data consistency and validity.

INSERTION OF TABLE ADMIN

```
--------------------------------------------------------------------------------------------------------

INSERT INTO `admin` (`adminid`, `adminname`, `loginid`, `password`, `status`,
`usertype`) VALUES
(1, 'Vrushant', 'admin', 'Password@123', 'Active', '');
--------------------------------------------------------------------------------------------------------
```

INSERTION OF TABLE APPOINTMENT

```
--------------------------------------------------------------------------------------------------------

INSERT INTO `appointment` (`appointmentid`, `appointmenttype`, `patientid`, `roomid`,
`departmentid`, `appointmentdate`, `appointmenttime`, `doctorid`, `status`,
`app_reason`) VALUES
(28, '', 11, 0, 1, '2023-12-05', '10:30:00', 11, 'Approved', 'Routine check-up'),
(29, '', 12, 0, 2, '2023-12-06', '14:00:00', 12, 'Approved', 'Experiencing chest pain'),
(30, '', 13, 0, 3, '2023-12-07', '11:45:00', 13, 'Approved', 'Post-surgery evaluation'),
(31, '', 14, 0, 4, '2023-12-08', '15:30:00', 14, 'Approved', 'Severe injury'),
(32, 'Admin', 15, 0, 5, '2023-12-09', '13:15:00', 15, 'Active', 'Annual health check-
up'),
(33, '', 16, 0, 6, '2023-12-10', '09:00:00', 16, 'Approved', 'Abdominal pain'),
(34, '', 17, 0, 7, '2023-12-11', '16:45:00', 17, 'Approved', 'Accident trauma'),
(35, '', 18, 0, 8, '2023-12-12', '23:33:00', 18, 'Approved', 'Follow-up on medication'),
(36, 'Admin', 19, 0, 9, '2023-12-13', '10:15:00', 19, 'Active', 'Discussing test
results'),
(37, 'Admin', 20, 0, 2, '2023-12-14', '17:00:00', 20, 'Active', 'Breathing difficulties');
--------------------------------------------------------------------------------------------------------
```

INSERTION OF TABLE BILLING_RECORDS

```
--------------------------------------------------------------------------------------------------------
```

```
INSERT INTO `billing_records` (`billingservice_id`, `billingid`, `bill_type_id`,
`bill_type`, `bill_amount`, `bill_date`, `status`) VALUES
(61, 28, 11, 'Consultancy Charge', 150.00, '2023-12-03', 'Active'),
(62, 28, 20, 'Treatment', 179.00, '2023-12-03', 'Active'),
(63, 28, 20, 'Treatment', 179.00, '2023-12-03', 'Active'),
(64, 28, 26, 'Treatment', 97560.00, '2023-12-03', 'Active'),
(65, 30, 13, 'Consultancy Charge', 180.00, '2023-12-03', 'Active'),
(66, 30, 23, 'Treatment', 530.00, '2023-12-03', 'Active'),
(67, 30, 20, 'Treatment', 179.00, '2023-12-03', 'Active'),
(68, 29, 12, 'Consultancy Charge', 200.00, '2023-12-03', 'Active'),
(69, 29, 23, 'Treatment', 530.00, '2023-12-03', 'Active'),
(70, 35, 18, 'Consultancy Charge', 170.00, '2023-12-03', 'Active'),
(71, 35, 20, 'Treatment', 179.00, '2023-12-03', 'Active');
```

INSERTION OF TABLE DEPARTMENT

```
INSERT INTO `department` (`departmentid`, `departmentname`, `description`, `status`)
VALUES
(1, 'Medicine', 'Medicine', 'Active'),
(2, 'Cardiology', 'Provides medical care to patients who have problems with their heart
or circulation.', 'Active'),
(3, 'Gynecology', 'Investigates and treats problems relating to the female urinary tract
and reproductive organs, such as Endometriosis, infertility and incontinence',
'Active'),
(4, 'Haematology', 'These hospital services work with the laboratory. In addition
doctors treat blood diseases and malignancies related to the blood', 'Active'),
(5, 'Maternity', 'Maternity wards provide antenatal care, delivery of babies and care
during childbirth, and postnatal support', 'Active'),
(6, 'Nephrology', 'Monitors and assesses patients with various kidney (renal) problems
and conditions', 'Active'),
(7, 'Oncology', 'A branch of medicine that deals with cancer and tumors. A medical
professional who practices oncology is an oncologist. The Oncology department provides
treatments, including radiotherapy and chemotherapy, for cancerous tumors and blood
disorders', 'Active'),
(8, 'Orthopaedics', 'Treats conditions related to the musculoskeletal system, including
joints, ligaments, bones, muscles, tendons and nerves', 'Active'),
(9, 'Radiology', 'Deals with the study and application of imaging technology like XRay',
'Active');
```

INSERTION OF TABLE DOCTOR

```
INSERT INTO `doctor` (`doctorid`, `doctorname`, `mobileno`, `departmentid`, `loginid`,
`password`, `status`, `education`, `experience`, `consultancy_charge`) VALUES
(11, 'Dr John Smith', '1234567890', 1, 'johnsmith', 'password123', 'Active', 'MD', 10.0,
150.00),
(12, 'Dr Emily Johnson', '9876543210', 2, 'emilyjohnson', 'pass456', 'Active', 'MS',
8.0, 200.00),
(13, 'Dr Michael Davis', '2345678901', 3, 'michaeldavis', 'pass789', 'Active', 'MD',
12.0, 180.00),
(14, 'Dr Jennifer Lee', '8765432109', 4, 'jenniferlee', 'pass123', 'Active', 'MS', 6.0,
120.00),
(15, 'Dr Robert Miller', '3456789012', 5, 'robertmiller', 'pass234', 'Active', 'MBBS',
15.0, 160.00),
(16, 'Dr Sarah White', '6543210987', 6, 'sarahwhite', 'pass567', 'Active', 'MS', 9.0,
220.00),
(17, 'Dr Christopher Brown', '4321098765', 7, 'christopherbrown', 'pass890', 'Active',
'MD', 11.0, 190.00),
```

```
(18, 'Dr Amanda Taylor', '2109876543', 8, 'amandataylor', 'passabc', 'Active', 'MD',
14.0, 170.00),
(19, 'Dr Kevin Wilson', '7890123456', 9, 'kevinwilson', 'passdef', 'Active', 'MD', 7.0,
130.00),
(20, 'Dr Rachel Moore', '5432109876', 2, 'rachelmoore', 'passtuv', 'Active', 'MS', 13.0,
240.00);
```

INSERTION OF TABLE DOCTOR_TIMINGS

```
INSERT INTO `doctor_timings` (`doctor_timings_id`, `doctorid`, `start_time`, `end_time`,
`status`) VALUES
(36, 11, '10:30:00', '17:00:00', 'Active'),
(37, 12, '09:30:00', '13:00:00', 'Active'),
(38, 13, '13:30:00', '17:00:00', 'Active'),
(39, 14, '14:00:00', '18:00:00', 'Active'),
(40, 15, '17:00:00', '21:00:00', 'Active'),
(41, 16, '13:00:00', '19:00:00', 'Active'),
(42, 17, '07:00:00', '11:00:00', 'Active'),
(43, 18, '13:30:00', '16:30:00', 'Active'),
(44, 19, '11:30:00', '14:30:00', 'Active'),
(45, 20, '12:30:00', '16:30:00', 'Active');
```

INSERTION OF TABLE MEDICINE

```
INSERT INTO `medicine` (`medicineid`, `medicinename`, `medicinecost`, `description`,
`status`) VALUES
(1, 'Paracetamol', 3.00, 'For fever per day 1 pc', 'Active'),
(2, 'Clotrimazole', 14.00, 'Clotrimazole is an antifungal, prescribed for local fungal
infections', 'Active'),
(3, 'Miconazole', 26.00, 'Prescribed for various skin infections such as jockitch and
also for vaginal yeast infections', 'Active'),
(4, 'Nystatin', 6.00, 'Antifungal drug, prescribed for fungal infections of the skin
mouth vagina and intestinal tract', 'Active'),
(5, 'Lotensin', 3.00, 'prevent your body from forming angiotensin', 'Active'),
(6, 'Cozaan', 5.00, 'ARBs block the effects of angiotensin on your heart.', 'Active'),
(7, 'Lovenox', 59.00, 'may prescribe an anticoagulant to prevent heart attack, stroke,
or other serious health problems', 'Active'),
(8, 'Abemaciclib', 278.00, 'drug for the treatment of advanced or metastatic breast
cancers.', 'Active'),
(9, 'Cyclophosphamide', 231.00, ' to treat lymphoma, multiple myeloma, leukemia, ovarian
cancer, breast cancer, small cell lung cancer', 'Active'),
(10, 'Captopril', 92.00, 'used alone or in combination with other medications to treat
high blood pressure and heart failure.', 'Active'),
(11, 'Enalapril', 18.00, 'to treat high blood pressure, diabetic kidney disease, and
heart failure', 'Active'),
(12, 'Ramipril', 31.00, 'to treat high blood pressure, diabetic kidney disease',
'Active'),
(13, 'Hydroxyurea', 55.00, 'used in sickle-cell disease, essential thrombocythemia,
chronic myelogenous leukemia and cervical cancer', 'Active'),
(14, 'Phenprocoumon', 258.00, 'Used for prevention of thrombosis', 'Active');
```

INSERTION OF TABLE PATIENT

```
INSERT INTO `patient` (`patientid`, `patientname`, `admissiondate`, `admissiontime`,
`address`, `mobileno`, `city`, `pincode`, `loginid`, `password`, `bloodgroup`, `gender`,
`dob`, `status`) VALUES
(11, 'John Doe', '2023-12-03', '06:47:08', '123 Main St', '9809875676', 'Anytown',
'12345', 'john_doe', 'password123', 'O+', 'MALE', '1990-12-05', 'Active'),
(12, 'Jane Smith', '2023-02-02', '14:45:00', '456 Oak St', '5555678', 'Othercity',
'67890', 'jane_smith', 'pass456', 'A-', 'Female', '1985-11-12', 'Active'),
(13, 'Bob Johnson', '2023-03-10', '09:15:00', '789 Pine St', '5559876', 'Somewhere',
'54321', 'bob_johnson', 'secure789', 'B+', 'Male', '1972-08-30', 'Active'),
(14, 'Alice Davis', '2023-04-05', '12:00:00', '101 Cedar St', '5554321', 'Anytown',
'12345', 'alice_davis', 'safe123', 'AB-', 'Female', '1988-04-18', 'Active'),
(15, 'Sam Wilson', '2023-05-20', '15:30:00', '202 Birch St', '5558765', 'Othercity',
'67890', 'sam_wilson', 'key567', 'A+', 'Male', '1995-09-25', 'Active'),
(16, 'Eva Brown', '2023-06-08', '11:45:00', '303 Elm St', '5552345', 'Somewhere',
'54321', 'eva_brown', 'safe789', 'O-', 'Female', '1978-12-03', 'Active'),
(17, 'Mike Taylor', '2023-07-15', '08:00:00', '404 Maple St', '5556543', 'Anytown',
'12345', 'mike_taylor', 'key123', 'B-', 'Male', '1980-02-15', 'Active'),
(18, 'Grace Miller', '2023-08-22', '13:20:00', '505 Pine St', '5558765', 'Othercity',
'67890', 'grace_miller', 'secure456', 'AB+', 'Female', '1992-07-08', 'Active'),
(19, 'Tom Clark', '2023-09-30', '10:10:00', '606 Cedar St', '5559876', 'Anytown',
'12345', 'tom_clark', 'pass789', 'O+', 'Male', '1983-03-28', 'Active'),
(20, 'Linda Hall', '2023-10-12', '14:15:00', '707 Oak St', '5553456', 'Othercity',
'67890', 'linda_hall', 'key789', 'A-', 'Female', '1975-06-14', 'Active');
```

INSERTION OF TABLE PAYMENT

```
INSERT INTO `payment` (`paymentid`, `patientid`, `appointmentid`, `paiddate`,
`paidtime`, `paidamount`, `status`) VALUES
(7, 13, 30, '2023-12-03', '08:20:49', 933.45, 'Active'),
(8, 18, 35, '2023-12-03', '08:23:17', 366.45, 'Active'),
(9, 12, 29, '2023-12-03', '08:23:59', 766.50, 'Active'),
(10, 11, 28, '2023-12-03', '08:24:17', 102971.40, 'Active');
```

INSERTION OF TABLE PRESCRIPTION

```
INSERT INTO `prescription` (`prescriptionid`, `doctorid`, `patientid`, `delivery_type`,
`delivery_id`, `prescriptiondate`, `status`, `appointmentid`) VALUES
(6, 11, 11, 'delivery', 0, '2023-12-04', 'Active', 28),
(7, 11, 11, 'delivery', 0, '2023-12-13', 'Active', 28),
(8, 13, 13, 'delivery', 0, '2023-12-04', 'Active', 30),
(9, 12, 12, 'delivery', 0, '2023-12-04', 'Active', 29),
(10, 18, 18, 'delivery', 0, '2023-12-04', 'Active', 35);
```

INSERTION OF TABLE PRESCRIPTION_RECORD

```
INSERT INTO `prescription_records` (`prescription_record_id`, `prescription_id`,
`medicine_name`, `cost`, `unit`, `dosage`, `status`) VALUES
(6, 6, '3', 26.00, 1, '1-1-1', 'Active'),
(7, 6, '1', 3.00, 1, '0-0-1', 'Active'),
(8, 7, '5', 3.00, 3, '0-1-1', 'Active'),
(9, 8, '4', 6.00, 1, '0-1-1', 'Active'),
(10, 8, '3', 26.00, 1, '1-0-1', 'Active'),
(11, 9, '4', 6.00, 1, '0-1-1', 'Active'),
(12, 9, '3', 26.00, 1, '0-1-1', 'Active'),
```

```
(13, 10, '4', 6.00, 1, '1-0-1', 'Active'),
(14, 10, '1', 3.00, 1, '1-1-0', 'Active');
```

INSERTION OF TABLE ROOM

```
INSERT INTO `room` (`roomid`, `roomtype`, `roomno`, `noofbeds`, `room_tariff`, `status`)
VALUES
(15, 'GENERAL WARD', 1, 20, 500.00, 'Active'),
(16, 'SPECIAL WARD', 2, 10, 100.00, 'Active'),
(17, 'GENERAL WARD', 2, 10, 500.00, 'Active'),
(18, 'GENERAL WARD', 121, 13, 150.00, 'Active'),
(19, 'GENERAL WARD', 850, 11, 500.00, 'Active');
```

INSERTION OF TABLE TREATMENT

```
INSERT INTO `treatment` (`treatmentid`, `treatmenttype`, `treatment_cost`, `note`,
`status`) VALUES
(20, 'Blood Test', '179.00', 'test checks for levels of 10 different components of every
major cell in your blood', 'Active'),
(21, 'Electrocardiogram', '70.00', 'Records the electrical activity of the heart',
'Active'),
(22, 'Echocardiogram', '1750.00', 'Provides an ultrasound picture that shows the
structure of the heart chambers and surrounding areas, and it can show how well the
heart is working.', 'Active'),
(23, 'Nuclear cardiology', '530.00', 'Nuclear imaging techniques use radioactive
materials to study cardiovascular disorders and diseases in a noninvasive way.',
'Active'),
(24, 'Colposcopy', '318.00', 'procedure to visually examine the cervix as well as the
vagina and vulva using a colposcope.', 'Active'),
(25, 'Colporrhaphy', '5518.00', 'surgical procedure in humans that repairs a defect in
the wall of the vagina.', 'Active'),
(26, 'Spine Surgery', '97560.00', 'This entails opening the operative site with a long
incision so the surgeon can view and access the spinal anatomy', 'Active'),
(27, 'Trauma surgery', '25448.00', 'surgical specialty that utilizes both operative and
non-operative management to treat traumatic injuries, typically in an acute setting',
'Active'),
(28, 'Diagnostic Tests', '989.00', 'may include MRI, CT, Bone Scan, Ultra sound, blood
tests', 'Active'),
(29, 'Chest XRay', '258.00', ' projection radiograph of the chest used to diagnose
conditions affecting the chest, its contents, and nearby structures', 'Active'),
(30, 'Ultrasound of the Abdomen', '560.00', 'noninvasive procedure used to assess the
organs and structures within the abdomen', 'Active'),
(31, 'Exercise Stress Test', '198.00', 'This test is good for evaluating chest pain to
see if your heart is the cause.', 'Active'),
(32, 'Ultrasound of the Pelvis', '520.00', 'noninvasive diagnostic exam that produces
images that are used to assess organs and structures within the female pelvis',
'Active'),
(33, 'Chemotherapy and Radiatio', '4850.00', 'Most common types of cancer treatment.
They work by destroying these fast-growing cells.', 'Active');
```

INSERTION OF TABLE TREATMENT_RECORDS

```
INSERT INTO `treatment_records` (`treatment_records_id`, `treatmentid`, `appointmentid`,
`patientid`, `doctorid`, `treatment_description`, `uploads`, `treatment_date`,
`treatment_time`, `status`) VALUES
(37, 20, 28, 11, 11, 'Blood Test for viral fever', '1911564826', '2023-12-03',
'12:22:00', 'Active'),
(38, 20, 28, 11, 11, 'Blood Test', '191156434', '2023-12-05', '11:00:00', 'Active'),
(52, 26, 28, 11, 11, 'spine surgery', '1292876765', '2023-12-03', '14:02:00', 'Active'),
(53, 23, 30, 13, 13, 'nucelear cardiology', '392578993', '2023-12-03', '23:34:00',
'Active'),
(54, 20, 30, 13, 13, 'blood test', '797619254', '2023-12-03', '11:33:00', 'Active'),
(55, 23, 29, 12, 12, 'cardiology', '999764995', '2023-12-03', '23:03:00', 'Active'),
(56, 20, 35, 18, 18, 'blood test', '1512785646', '2023-12-03', '23:02:00', 'Active');
```

INSERTION OF TABLE USER

```
INSERT INTO `user` (`userid`, `loginname`, `password`, `patientname`, `mobileno`,
`email`, `createddateandtime`) VALUES
(1, 'admin', 'admin', 'admin', '', '', '2017-12-14 11:21:45');
```

---

3. Query Scenario Design

We developed a series of SQL queries tailored to the operational needs of a hospital management system. These queries aim to extract critical data from our MySQL database, offering insights into patient appointments, doctor charges, and departmental finances. By categorizing appointments and analyzing financial metrics, we provide a foundation for effective decision-making within the healthcare facility. The queries are designed to be robust and scalable, reflecting the complexity and precision required for medical data management. The final output is presented in an organized format to facilitate easy access and analysis for system users.

1. Description:

Write a SQL query to list each patient's name along with their appointment dates. Label each appointment as 'Upcoming' if it is after December 3, 2023, and 'Completed' otherwise.

SQL Code:

```
SELECT p.patientname, a.appointmentdate,
CASE
    WHEN a.appointmentdate > '2023-12-03' THEN 'Upcoming'
    ELSE 'Completed'
END AS appointment_status
FROM patient p
JOIN appointment a ON p.patientid = a.patientid;
```

Result Table:

patientname	appointmentdate	appointment_status
John Doe	2023-12-05	Upcoming
Jane Smith	2023-12-06	Upcoming
Bob Johnson	2023-12-07	Upcoming
Alice Davis	2023-12-08	Upcoming
Sam Wilson	2023-12-09	Upcoming
Eva Brown	2023-12-10	Upcoming
Mike Taylor	2023-12-11	Upcoming
Grace Miller	2023-12-12	Upcoming
Tom Clark	2023-12-13	Upcoming
Linda Hall	2023-12-14	Upcoming

2. **Description:**

Write a SQL query to display the name, department, and consultancy charge of each doctor whose consultancy charge is higher than the average consultancy charge of all doctors.

SQL Code:

```
SELECT d.doctorname, dep.departmentname, d.consultancy_charge
FROM doctor d
JOIN department dep ON d.departmentid = dep.departmentid
WHERE d.consultancy_charge > (SELECT AVG(consultancy_charge) FROM doctor);
```

Result Table:

doctorname	departmentname	consultancy_charge
Dr Emily Johnson	Cardiology	200.00
Dr Michael Davis	Gynecology	180.00
Dr Sarah White	Nephrology	220.00
Dr Christopher Brown	Oncology	190.00
Dr Rachel Moore	Cardiology	240.00

3. **Description:**

Write a SQL query to find the average consultancy charge for each department, but only include departments where the average consultancy charge is greater than 150.

SQL Code:

```
SELECT departmentid, AVG(consultancy_charge) AS average_charge
FROM doctor
GROUP BY departmentid
HAVING AVG(consultancy_charge) > 150;
```

Result Table:

departmentid	average_charge
2	220.000000
3	180.000000
5	160.000000
6	220.000000
7	190.000000
8	170.000000

4. **Description:**

Write a SQL query to determine the total revenue and count of doctors for each department. Display the department name, the number of doctors, and the total revenue.

SQL Code:

```
SELECT dep.departmentname AS department_name,
COUNT(d.doctorid) AS doctor_count,
SUM(d.consultancy_charge) AS total_revenue
FROM department dep
JOIN doctor d ON dep.departmentid = d.departmentid
GROUP BY dep.departmentname;
```

Result Table:

department_name	doctor_count	total_revenue
Medicine	1	150.00
Cardiology	2	440.00
Gynecology	1	180.00
Haematology	1	120.00
Maternity	1	160.00

Nephrology	1	220.00
Oncology	1	190.00
Orthopaedics	1	170.00
Radiology	1	130.00

5. **Description:**

Calculate the average wait time for each doctor's patients from the time of appointment booking to the actual appointment date.

SQL Code:

```
SELECT
    d.doctorname,
    AVG(DATEDIFF(a.appointmentdate, p.admissiondate)) AS avg_wait_time_days
FROM
    doctor d
JOIN
    appointment a ON d.doctorid = a.doctorid
JOIN
    patient p ON a.patientid = p.patientid
WHERE
    a.status = 'Approved'
GROUP BY
    d.doctorid
LIMIT 0, 1000;
```

Result Table:

doctorname	avg_wait_time_days
Dr John Smith	2.0000
Dr Emily Johnson	307.0000
Dr Michael Davis	272.0000
Dr Jennifer Lee	247.0000
Dr Sarah White	185.0000
Dr Christopher Brown	149.0000
Dr Amanda Taylor	112.0000

6. **Description:**

Write a SQL query to display each doctor's ID, name, and the total number of appointments they have. Order the results by the number of appointments in descending order.

SQL Code:

```
SELECT d.doctorid, d.doctorname,
(SELECT COUNT(*) FROM appointment a WHERE a.doctorid = d.doctorid) AS
appointments_count
FROM doctor d
ORDER BY appointments_count DESC;
```

doctorid	doctorname	appointments_count
11	Dr John Smith	1
12	Dr Emily Johnson	1
13	Dr Michael Davis	1
14	Dr Jennifer Lee	1
15	Dr Robert Miller	1
16	Dr Sarah White	1
17	Dr Christopher Brown	1
18	Dr Amanda Taylor	1
19	Dr Kevin Wilson	1
20	Dr Rachel Moore	1

7. **Description:**

Write a SQL query to calculate the total cost of each medicine. List the medicine name and the total cost and sort the results by the total cost in descending order, then by medicine name in ascending order.

SQL Code:

```
SELECT medicinename, SUM(medicinecost) AS total_cost
FROM medicine
GROUP BY medicinename
ORDER BY total_cost DESC, medicinename ASC;
```

Result Table:

medicinename	total_cost
Abemaciclib	278.00
Phenprocoumon	258.00
Cyclophosphamide	231.00
Captopril	92.00
Lovenox	59.00
Hydroxyurea	55.00
Ramipril	31.00
Miconazole	26.00
Enalapril	18.00
Clotrimazole	14.00
Nystatin	6.00
Cozaan	5.00
Lotensin	3.00
Paracetamol	3.00

8. **Description:**

Write a SQL query to count the number of active and inactive patients in the patient table.

SQL Code:

```
SELECT
   SUM(CASE WHEN status = 'Active' THEN 1 ELSE 0 END) AS active_patients,
   SUM(CASE WHEN status = 'Inactive' THEN 1 ELSE 0 END) AS inactive_patients
FROM patient;
```

Result Table:

active_patients	inactive_patients
10	0

9. **Description:**

Analyze if there's a correlation between the experience level of doctors and the average treatment costs of the patients they see.

SQL Code:

```
SELECT
    d.doctorname,
    d.experience,
    AVG(t.treatment_cost) AS avg_treatment_cost
FROM
    doctor d
JOIN
    appointment a ON d.doctorid = a.doctorid
JOIN
    treatment_records tr ON a.appointmentid = tr.appointmentid
JOIN
    treatment t ON tr.treatmentid = t.treatmentid
GROUP BY
    d.doctorid
ORDER BY
    d.experience DESC
LIMIT 0, 1000;
```

Result Table:

doctorname	experience	avg_treatment_cost
Dr Amanda Taylor	14.0	179.000000
Dr Michael Davis	12.0	354.500000
Dr John Smith	10.0	32639.333333
Dr Emily Johnson	8.0	530.000000

10. Description:

Write a SQL query to find the total earnings from consultancy charges for each department, including only doctors with more than 5 years of experience and who have seen more than one patient.

SQL Code:

```
SELECT dep.departmentname, SUM(d.consultancy_charge) AS total_earnings
```

```
FROM department dep
JOIN doctor d ON dep.departmentid = d.departmentid
JOIN appointment a ON d.doctorid = a.doctorid
WHERE d.experience > 5
GROUP BY dep.departmentname
HAVING COUNT(DISTINCT a.patientid) > 1;
```

Result Table:

departmentname	total_earnings
Cardiology	440.00

11. Description:

Write a SQL query to identify patients who have a sequence of appointments where each subsequent appointment is exactly one week after the previous one. Display the patient ID, appointment ID, and appointment date for these sequences.

SQL Code:

```
WITH RECURSIVE SequentialAppointments AS (
   SELECT patientid, appointmentid, appointmentdate
   FROM appointment
   WHERE NOT EXISTS (
      SELECT 1
      FROM appointment a2
      WHERE a2.patientid = appointment.patientid
      AND a2.appointmentdate = DATE_SUB(appointment.appointmentdate,
INTERVAL 7 DAY)
   )
   UNION ALL
   SELECT a.patientid, a.appointmentid, a.appointmentdate
   FROM appointment a
   JOIN SequentialAppointments sa ON a.patientid = sa.patientid
   WHERE a.appointmentdate = DATE_ADD(sa.appointmentdate, INTERVAL 7 DAY)
)
SELECT patientid, appointmentid, appointmentdate
FROM SequentialAppointments
ORDER BY patientid, appointmentdate;
```

Result Table:

patientid	appointmentid	appointmentdate
11	28	2023-12-05
12	29	2023-12-06
13	30	2023-12-07

14	31	2023-12-08
15	32	2023-12-09
16	33	2023-12-10
17	34	2023-12-11
18	35	2023-12-12
19	36	2023-12-13
20	37	2023-12-14

12. Description:

Write a SQL query to identify patients who have not had any appointments for more than 6 months following their last appointment. Display the names of these patients.

SQL Code:

```
SELECT
    DISTINCT p.patientname
FROM
    patient p
WHERE
    NOT EXISTS (
        SELECT 1
        FROM appointment a
        WHERE
            a.patientid = p.patientid AND
            a.appointmentdate > (SELECT MAX(a2.appointmentdate) FROM
appointment a2 WHERE a2.patientid = p.patientid) + INTERVAL 6 MONTH
    );
```

Result Table:

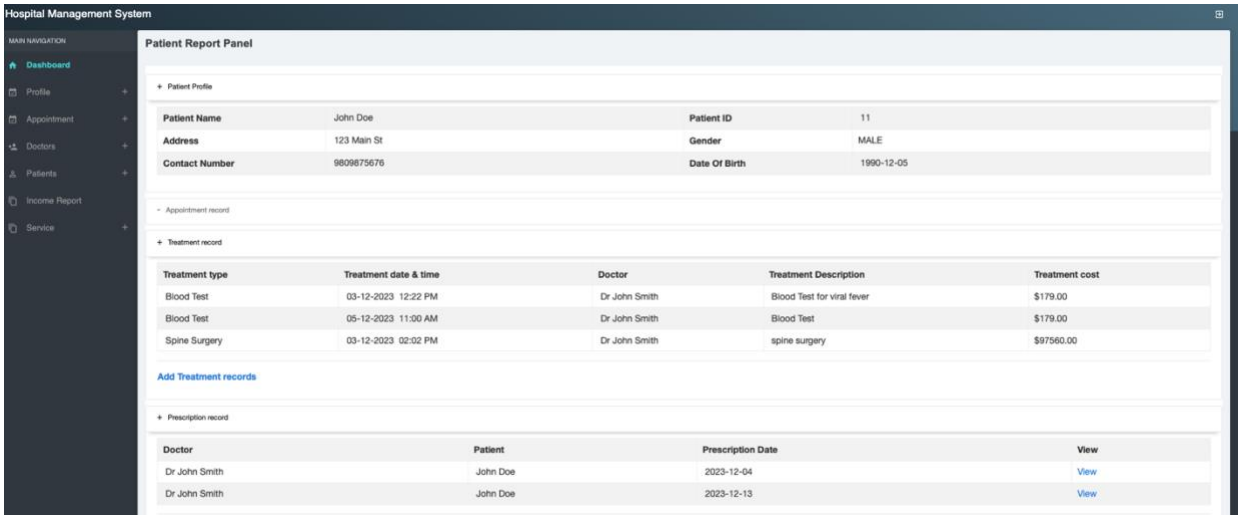
patientname
John Doe
Jane Smith
Bob Johnson
Alice Davis
Sam Wilson
Eva Brown
Mike Taylor
Grace Miller
Tom Clark

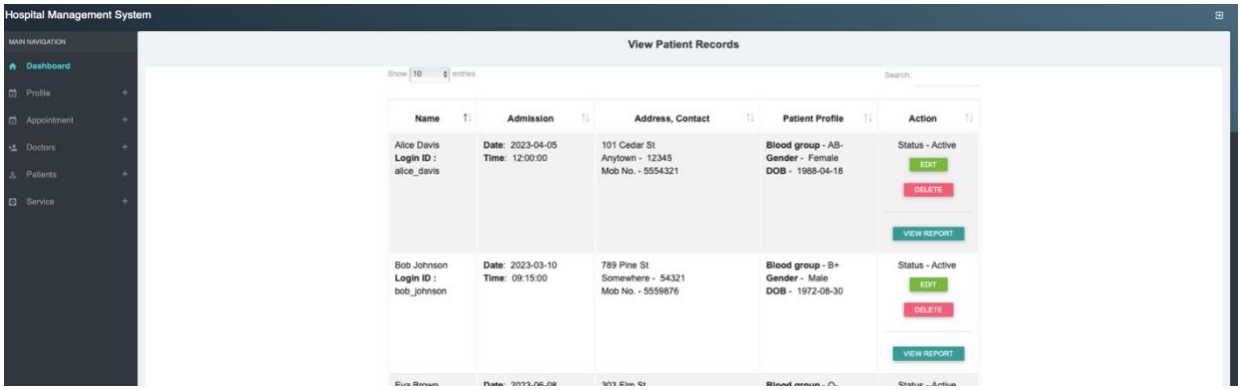
Linda Hall

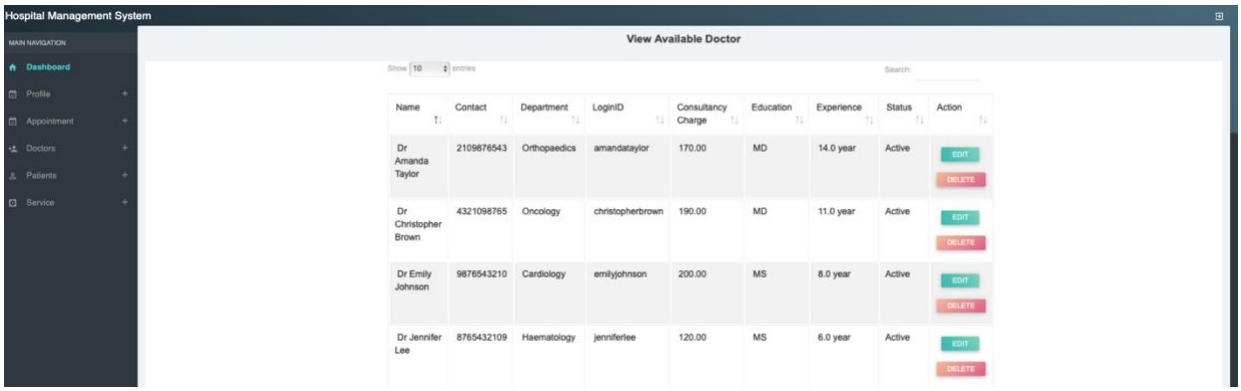
4. Software Integration

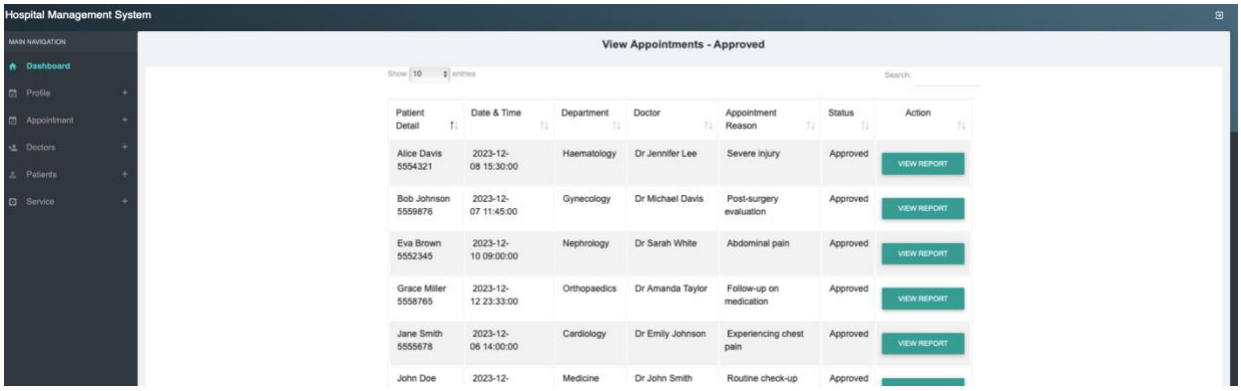
Collaboratively, we've successfully crafted a resilient hospital database system utilizing MySQL, adeptly managing backend data encompassing patients, appointments, treatments, and administrative records. Vital PHP scripts, such as doctorprofile.php, adminaccount.php, and treatment.php, act as crucial connectors between the frontend and backend, adeptly processing user requests and updating the MySQL database. For instance, the appointment.php script oversees the validation and secure storage of appointment details initiated by patients through the frontend. Similarly, prescriptiondetail.php effectively manages prescription data inputted by doctors, ensuring its secure storage in the database. MySQL serves as the steadfast database management system on the backend, organizing information related to patients, appointments, treatments, and administrative tasks. The frontend, meticulously designed with HTML and CSS, prioritizes user-friendly navigation, while JavaScript validates input data before transmitting it to the server. The incorporation of JavaScript enables AJAX calls, improving efficiency by asynchronously retrieving data from the server. Our team takes pride in developing a highly efficient system that significantly streamlines healthcare operations and elevates the overall user experience.

Snippets of the Application:







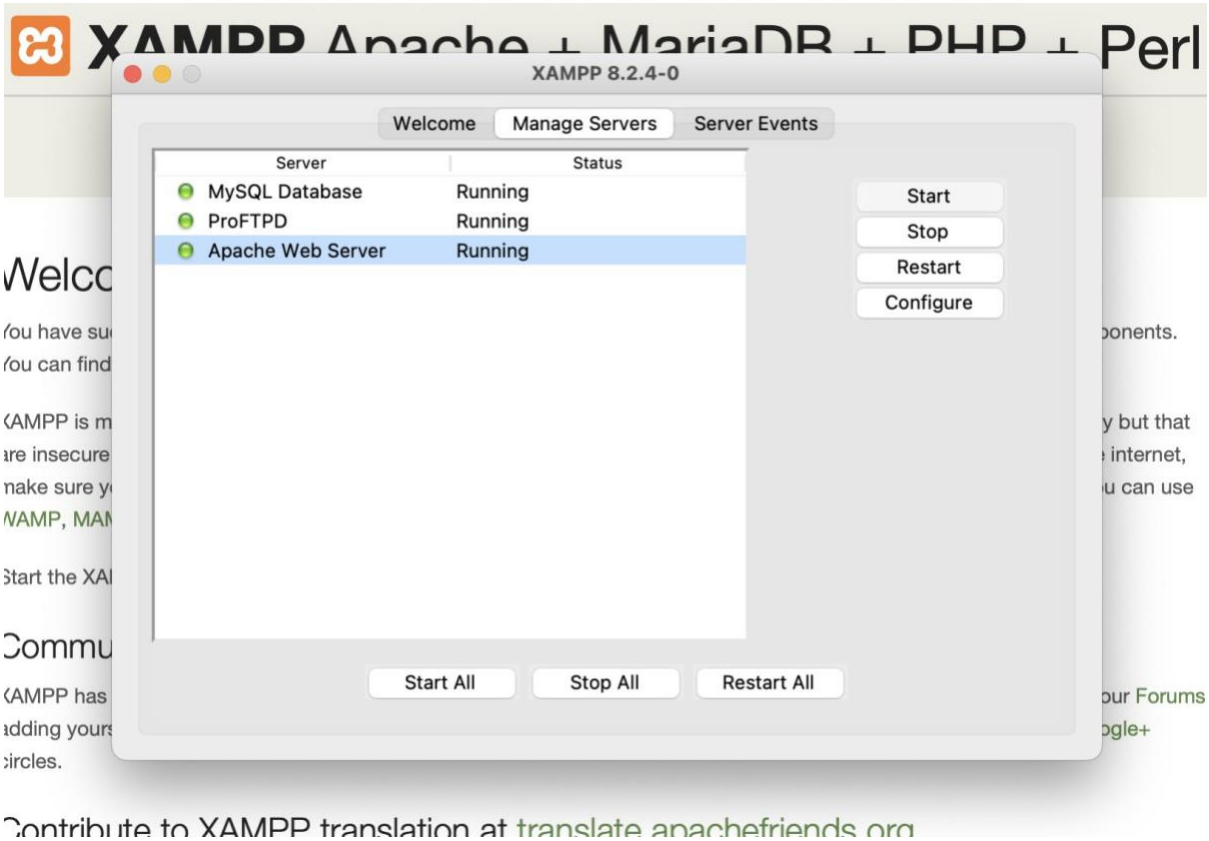


Instructions to run and test the Software Component:

Step 1: Download and install XAMPP (for Macintosh), and WAMP (for Windows)

Step 2: For Mac, XAMPP needs to give permission from the System setting to install.

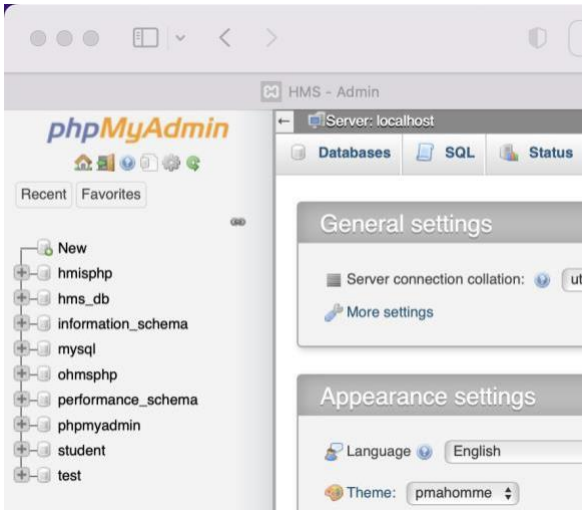
Step 3: Click on ‘Start All’.



Step 4: Type localhost in the browser

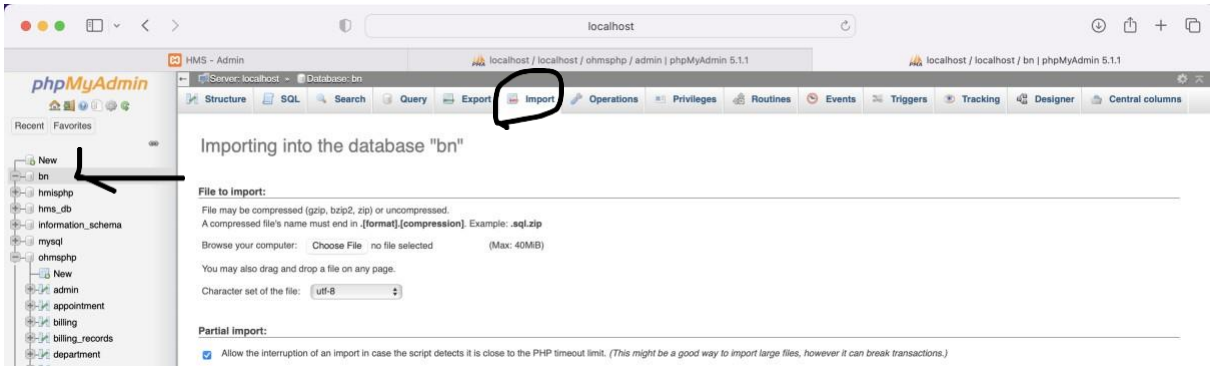
You will see this webpage -> now click on phpMyAdmin

Step 5: Click new -> give Database Name = ohmsphp -> then click on ‘Create’



Step 6: Now extract the zip file I have provided in the group and -> go inside hospital-management/ DATABASE FILE -> ohmsphp.sql (look for this file)

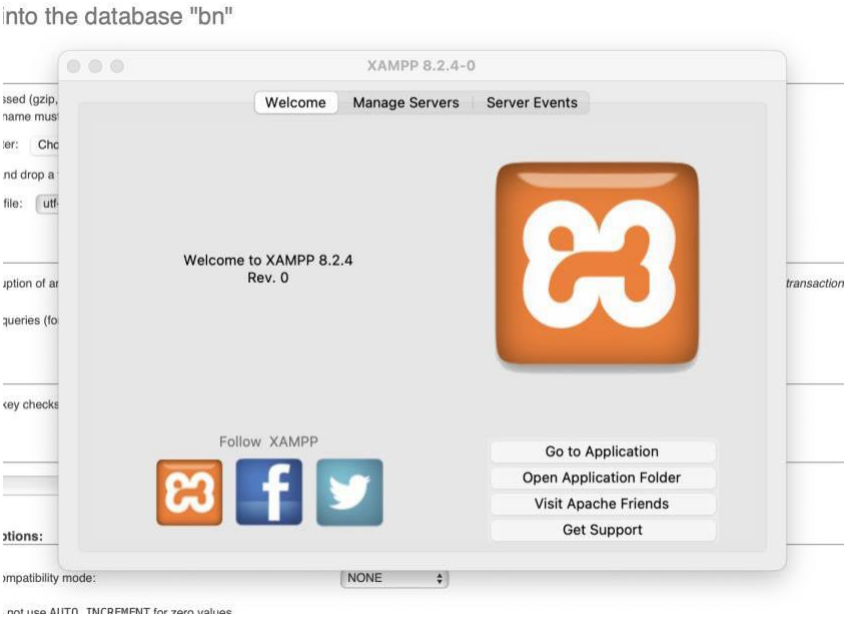
Step 7: Go again in the PHPmyadmin -> click on the database we have created -> Click Import (you will find in the top bar) -> choose file (ohmsphp.sql from the extracted folder)



Here, we have created a database bn just for the example, you have to create **ohmsphp**

Note: After doing this, you should have tables and a couple of records in your database.

(Hospital-management) and paste here in htdocs



Step 9: now go to the browser and type: http://localhost/hospital-management/

admin: admin, password@123
Doctor: vrushant, vrushant1234

5. Conclusion

The implementation phase of the HMS project has been successfully completed, marking a significant milestone in our endeavor to modernize hospital management. Through meticulous planning and execution, we have developed a system that not only adheres to the initial design specifications but also exhibits flexibility for future enhancements. Creating a MySQL database, complete with a modified relational schema, and integrating a user-friendly web application interface, has laid down a robust infrastructure for managing hospital operations. The database is populated with consistent and valid data, and the query scenarios are designed to provide insightful analytics, facilitating informed decision-making. The optional software component has been crafted to interact seamlessly with the database, ensuring a smooth user experience. This report has documented the challenges encountered, the solutions applied, and the successful outcomes of the implementation phase. As we move forward, the HMS is poised to deliver improved operational efficiency, better patient care, and enhanced data management within the hospital environment.