**Hospital Management System**

Prepared by:

Justin Lee

Ian Pallares

Olive Snider

Akhila Vuppalapati

December 6, 2020

# **1. Application Background**

Hospitals can be tremendous organizations with many moving parts. On a daily basis the systems at large hospitals such as University Hospitals are expected to register thousands of new patients each day, assign varying staff to these patients, and contact insurance companies to verify payment for procedures. With the recent influx in patients and increased stress added to staff life by the coronavirus (COVID-19) pandemic, sorting these patients has become paramount even though faster patient turnover is more important than ever.

The large number of patients means that it is unrealistic to expect these accounts to be processed by hand and separately. Updating these charts separately causes many issues when data from different sources are conflicting. Sometimes even providers for a patient in different departments have conflicting information which makes diagnoses so much more difficult and possibly inaccurate. It would be much easier for hospital management if all of the information could be accessed and updated all in one place. To this purpose, we have created a database that allows for automated creation and sorting of both patients and staff within a hospital in order to relieve stress and increase efficiency throughout the organization. The system also tracks which staff have been assigned to a patient, along with that doctor’s department in the hospital. This would allow faster patient turnover as less time would be spent on frivolous concerns such as ensuring everything was up to date and accurate.

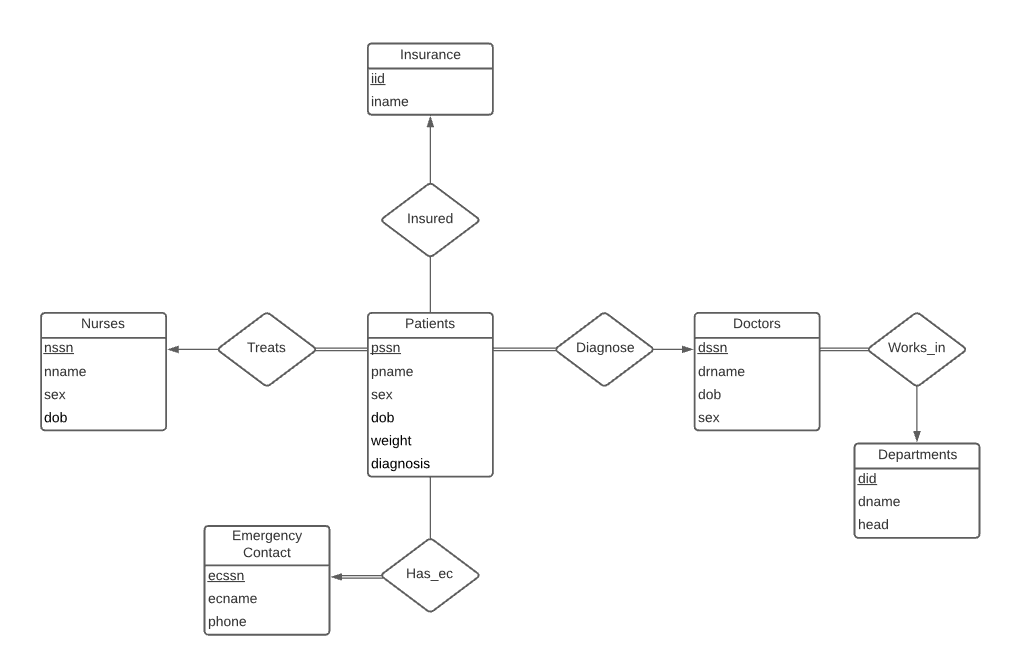
Our software, Hospital Management Database System, will be used to make all patient care and hospital management personnel’s lives so much easier by consolidating all of the necessary information. The hospital management system can be used for administrative duties such as billing insurance and contacting patients’ emergency contacts when need be. Our system can also be used to figure out which departments have more patients and would need more funding and resources in the future. Doctors and nurses, on the other hand, can use this system to ensure better patient care and diagnoses. While our system is named Hospital Management Database System, it can actually be used in any setting with patient care such as hospitals, clinics, and dispensaries.

# **2. Data Description**

Our database has six entities and five relations. Patients holds all the personal information for the various patients in the hospital, including attributes such as their date of birth, sex, and their diagnosis. When a patient is entered into the system, they also must be assigned a doctor and nurse from their respective tables. The patient may also select their emergency contact and insurance from those entities. Each emergency contact belongs to a patient. And nurses and doctors can have multiple patients, with every doctor belonging to a department. Doctors are assigned to different departments, each of which have a head doctor listed.

There are several constraints set on the entities in this database. The first is that each entity must have a primary key which is Unique and Not Null. All other attributes in entities also have the Not Null attribute assigned to them, regardless of data type. This is to ensure that no patient, staff, insurance plan, emergency contact, or department is missing information that could be critical to the patient. Each table also has Not Null foreign key constraints to other tables in order to preserve relationships within the diagram. The data populating these tables has been falsely generated using Python’s Faker and Panda libraries in order to preserve privacy when displaying and confirming data, as opposed to using information regarding real medical patients.

# **3. E/R Diagram**



# **4. Functional Dependencies**

**Entities**

**Patients**

The patients attributes are personal information about the patient including their diagnosis, they are independent of each other so the only functional dependency (FD) is the pssn implying all other attributes:

pssn→{pssn, pname, sex, dob, weight, diagnosis}

BCNF: The pssn is a superkey of Patients.

**Nurses**

Nurse’s attributes are a list of personal information about the nurse, given independence the only FD here is the primary key nssn implying all other attributes:

nssn→{nssn, nname, sex, dob}

BCNF: The nssn is a superkey of Nurses.

**Doctors**

Doctor’s attributes are a list of personal information about the doctor, the only FD here is the primary key dssn implying all the other attributes:

dssn→{dssn, drname, dob, sex}

BCNF: The dssn is a superkey of Doctors.

**Emergency Contact**

Emergency Contact has multiple attributes like their ssn, name, and phone number. Since a phone number is unique to each person, it could theoretically be used to identify the emergency contact. This is due to phone having the attribute ecssn which contains the rest of the emergency contact’s information.

ecssn→{ecssn, ecname, phone}

phone→{ecssn}

phone→{ecssn, ecname, phone}

BCNF: The ecssn is a superkey of Emergency\_contact, and phone is superkey because it’s attribute closure is {ecssn, ecname, phone} using the transitivity axiom.

**Insurance**

Insurance contains the insurance ID and the name of the insurance company, the only FD is the primary key iid implying the other attributes:

iid→{iid, iname}

BCNF: The superkey is iid and it is the only superkey.

**Departments**

Departments contains the department ID, the name of the department and the head of the department. The functional dependencies are as follows and because they are superkeys of Departments it implies BCNF:

did→{did, dname, head}

dname→{did}

dname→{did, dname, head}

head→{did}

head→{did, dname, head}

BCNF: The did, dname, and head are superkeys of Departments. This is because every department has a unique did and a unique department name. Also a department can only have one unique department head.

**Relationships**

**treats, insured, diagnose, has\_ec**

“diagnose” describes the relationship between the doctor and the patient, “treats” describes the relationship between the nurse and the patient, “insured” describes the relationship between the insurance company and the patient, and “has\_ec” describes the relationship between the emergency contact and the patient.

pssn→{all attributes}

BCNF: The pssn is a superkey for treats, insured, diagnose, and has\_ec. This is because all of these relationships are one to many with patients on the many side.

**works\_in**

“works\_in” describes the relationship between a doctor and a department.

dssn→{dssn, drname, dob, sex, did, dname, head}

head→{dssn}

head⊆dssn

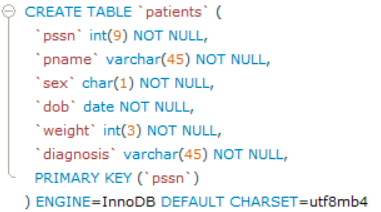
pssn→{dssn, drname, dob, sex}

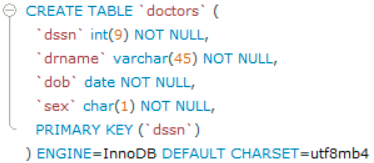
pssn→{dssn, drname, dob, sex, did, dname, head}

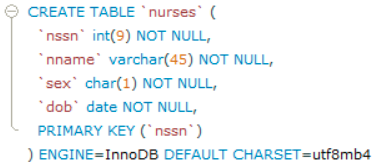
BCNF: The pssn again is a superkey of works\_in because a unique doctor treats a patient, and many doctors are in one department. Head is a subset of dssn so the functional dependency is trivial. Also, dssn is a superkey of works\_in.

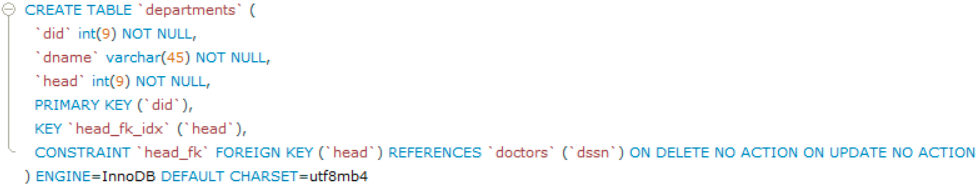
# **5. Relation Schemas**

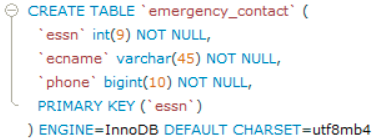
### Entities Schemas

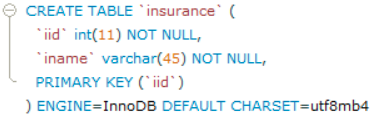




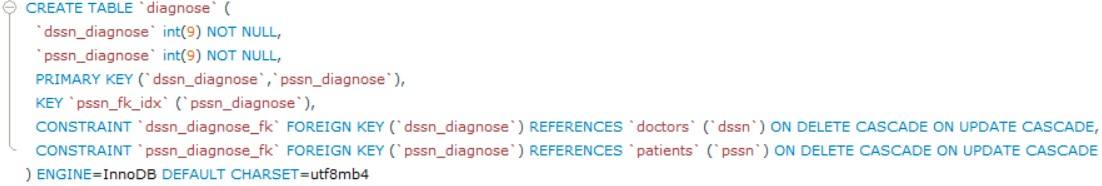


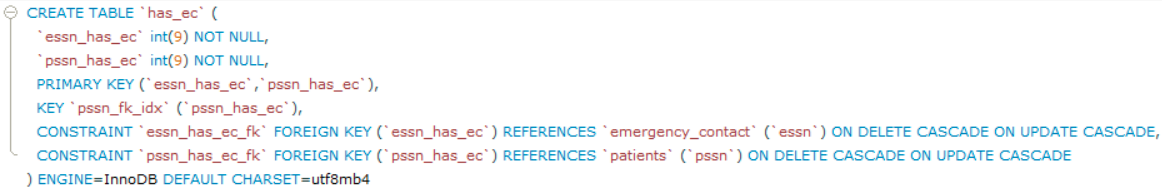


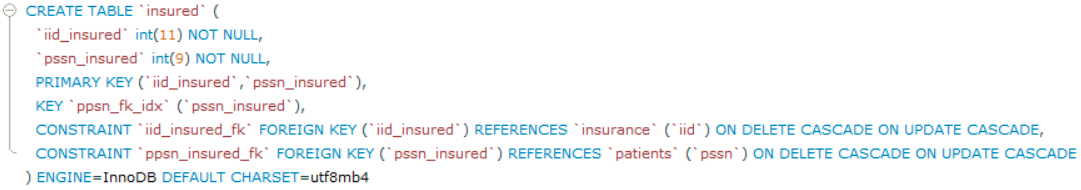


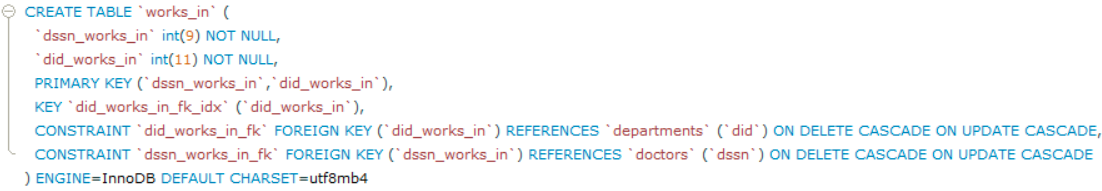
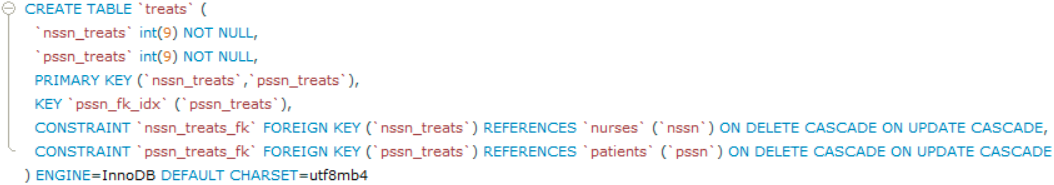


### Relationship Schemas





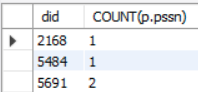




# **6. Example Queries**

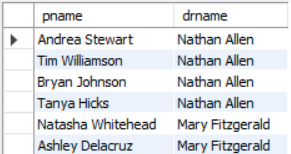
Query finds department id and the number of patients under 100lbs per that department.

SELECT de.did, COUNT(p.pssn) FROM Patients p, Departments de, Diagnose di, Doctors dr, Works\_In w WHERE p.weight < 100 AND p.pssn = di.pssn\_diagnose AND di.dssn\_diagnose = dr.dssn AND dr.dssn = w.dssn\_works\_in AND w.did\_works\_in = de.did GROUP BY de.did



The head of the “Traditional Chinese Medicine” department has been arrested for running a drug smuggling operation for the notorious gang leader Ian Pallares, and all the doctors in their department, are suspected of malpractice. List all the patients and their respective doctors in that department.

SELECT p.pname, dr.drname FROM Patients p, Doctors dr, Diagnose di, Works\_In w, Departments d WHERE p.pssn = di.pssn\_diagnose AND di.dssn\_diagnose = dr.dssn AND dr.dssn = w.dssn\_works\_in AND w.did\_works\_in = d.did AND d.dname = "Traditional Chinese Medicine"



# **7. Technology Used**

This project was created using MySQL as the primary Database Management System. We utilized the free MySQL Workbench program to create schemas and entities, test queries, and verify data entry. Using several python scripts with faker we were able to generate large amounts of data which were then imported as CSV files into MySQL Workbench. Using PHP plugins for Eclipse we created the front end of the website, and used local XAMPP Apache and MySQL servers to communicate across the platforms. Git was used for version control.

# **8. Code Repository**

The GitHub repository for this project can be found at: <https://github.com/avuppalapati/HospitalManagement>.

Screenshots of the web application will be included in section 10.

# **9. Member Roles**

Justin Lee

* Worked on html and css
* Provided additional features for forms
* Created ER diagram with relational schema and functional dependencies
* Worked on php scripts and MySQL

Ian Pallares

* Created php scripts
* Worked on front end design

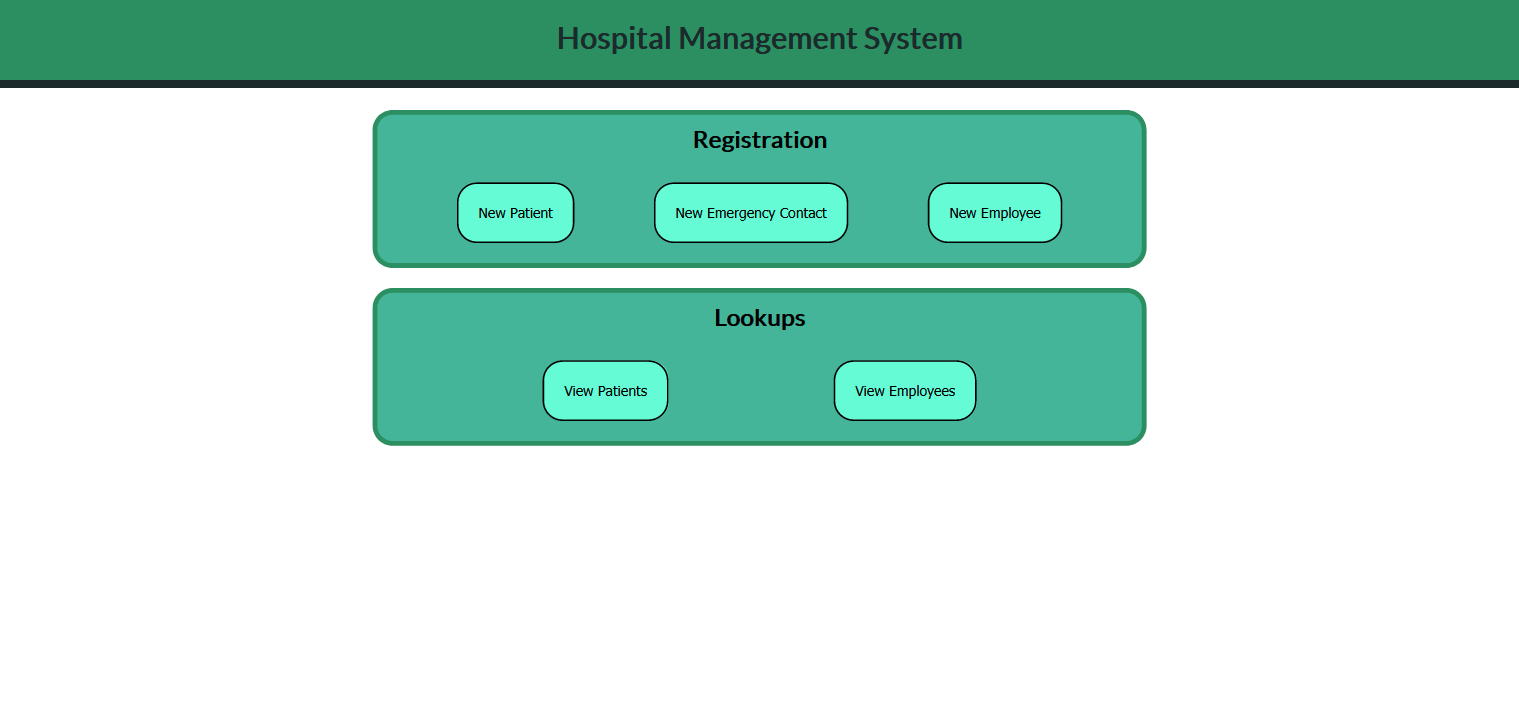
Olive Snider

* Functional dependencies/3NF satisfaction
* Python data generation
* Assisted ER diagram updates

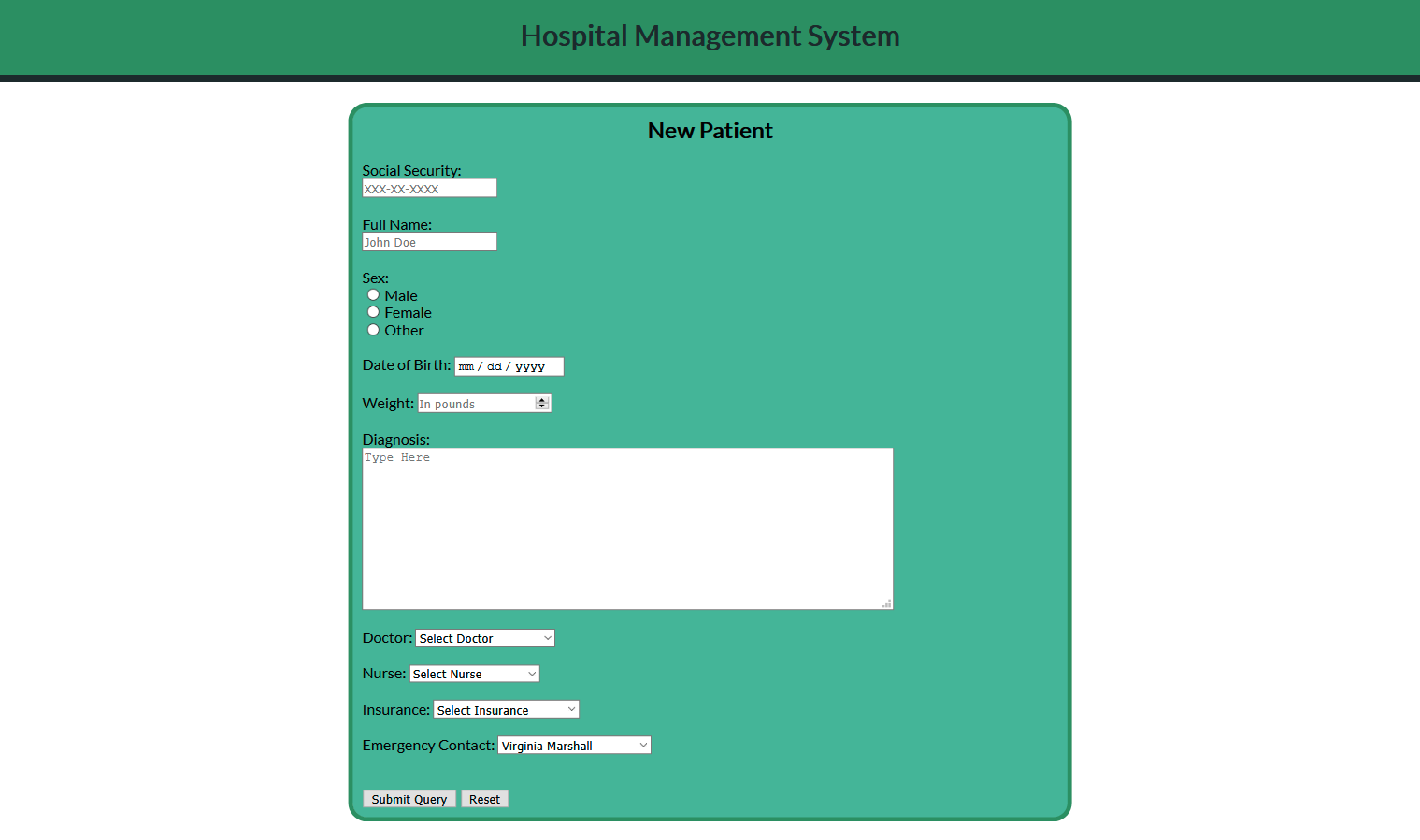
Akhila Vuppalapati

* Wrote the application background
* Created the example queries
* Assisted Python data generation

# **10. Web Application Screenshots**



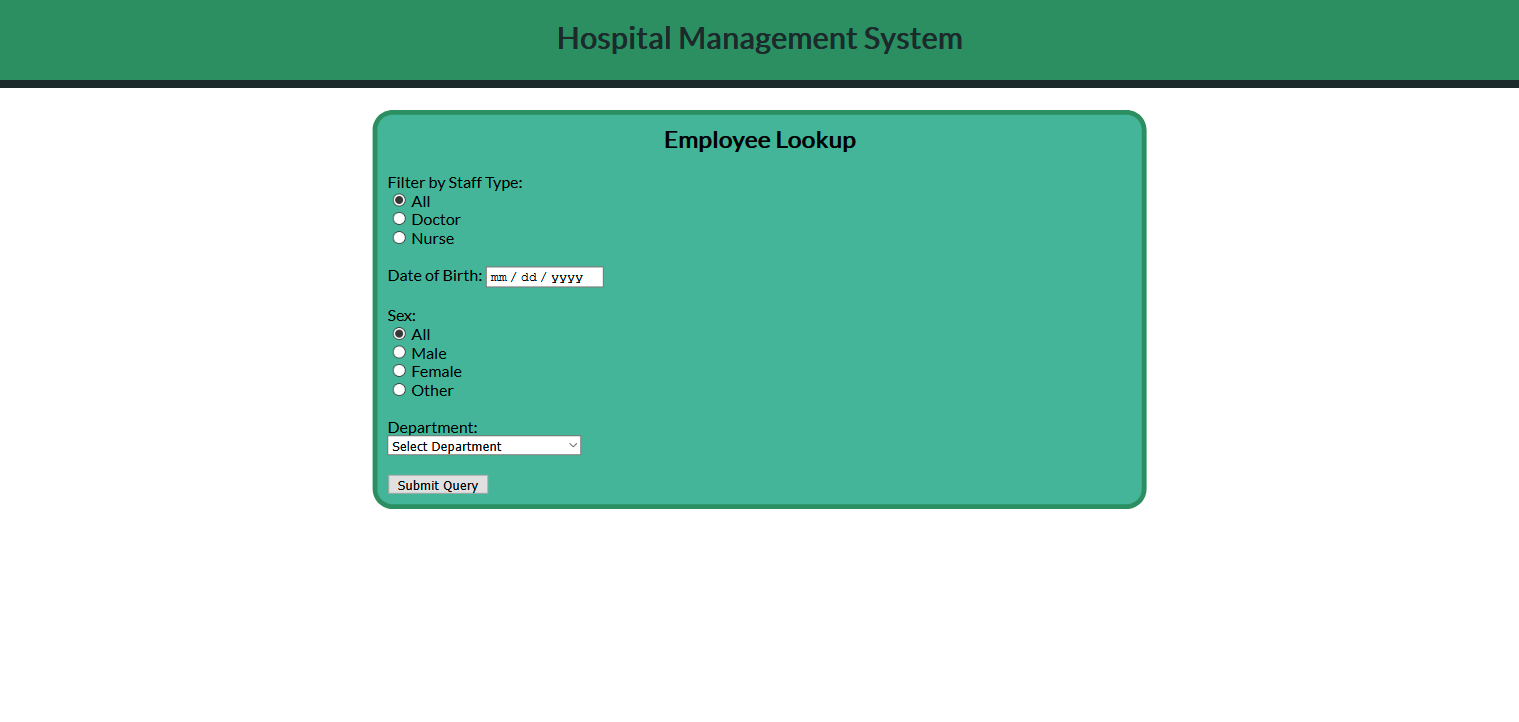
1. Home Page



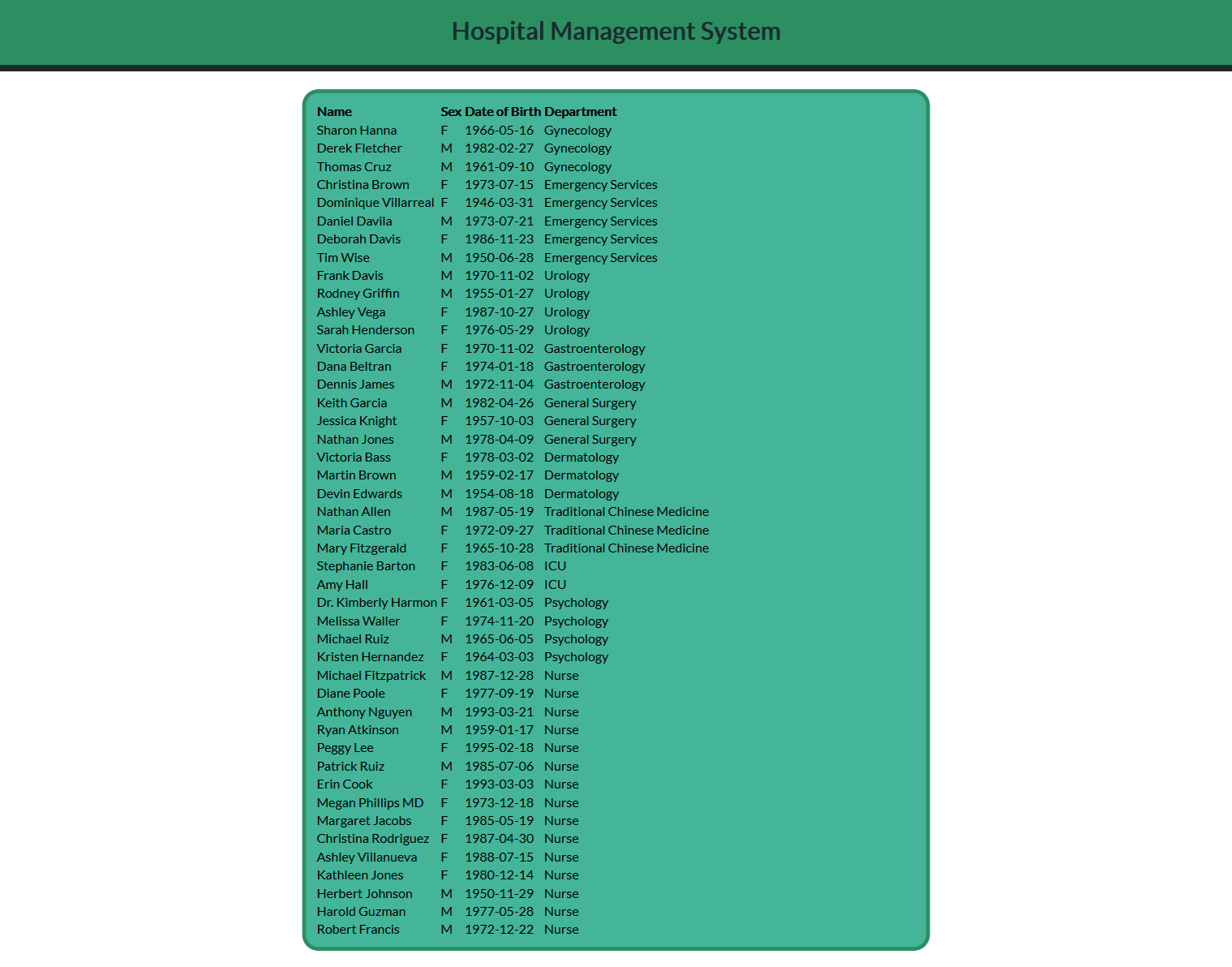
2. New Patient Form



3. View Patient Results



4. View Employee Form



5. View Employee Results