Project and Database Outline

a) Overview:

A hospital consisting of 50+ doctors and staff members, 1000+ patients, and 5000+ orders and results, allows for quite a bit of complexity. There are five entities in the database: 1) Doctors, 2) Patients, 3) Staff, 4) Orders, and 5) Results. Among these 5 entities, there can be different types, multiple relationships, and numerous ways for information to get lost/mixed up.

A database with around 5-10 tables (for entities and relationships) in combination with a neat front-end display with about 3-5 pages, allows for a much easier and efficient way of viewing and managing this data.

Doctors would be able to put in Orders (Prescription, Blood Tests, X-Ray) for Patients. An Order need not be assigned to a Staff when first initiated. So, Staff can check the Orders that need to be fulfilled, assign themselves Orders that are within their expertise and time frame, fulfill those Orders accordingly and load up their Results.
Eventually, Doctors, in their own time, or in Patient's next visit would be able to check on the Results of their Order.

Our Database will allow a small hospital to initiate, track, and fulfill orders for all Patients that decide to get treatment there.

b) Database outline, in words:

A database for a small clinic/hospital.

1. **Doctors**: Records the details of Doctors that work in Hospital.
   - doctorID: int, auto_increment, unique, not NULL, PK
   - lastName: varchar(255), not NULL
   - firstName: varchar(255), not NULL
   - department: varchar(255), not NULL
   - Relationship (Doctors-Orders)(11:0M): An Order can come from one and only one Doctor, while a Doctor can send zero to multiple Orders, doctorID is FK in Order.
   - Relationship (Doctor-Patient)(11:0M): All Patients must have one and only one primary Doctor, while a Doctor can be a primary Doctor for zero to multiple Patients, primaryDoctorID is a FK in Patient
   - Implemented by: Sanjay Ramanathan
2. **Patients**: All the Patients who are being/have been treated by the hospital
   - patientID: int, auto_increment, unique, not NULL, PK
   - lastName: varchar(255), not NULL
   - firstName: varchar(255), not NULL

- primaryDoctorID: assigns Doctor to a Patient, not NULL FK
- Relationship (Patient-Order)(11:0M): an Order can be connected to just one Patient, but a Patient can have zero to multiple Orders. patientID is FK in Order.
- Relationship (Patient-Staff)(0M:0M): A Patient can be handled by zero or multiple Staff and vice versa. Defined in Staff_Patients table.
- Relationship (Doctors-Patients)(11:0M): Doctors can have zero or many Patients and a Patient can have only one Doctor.
- Implemented by: Sanjay Ramanathan

3. **Staff**: All types of Staff members currently active in the hospital
   - staffID: int, auto_increment, unique, not NULL, PK
   - staffType: (Nurse, Pharmacy, Lab Technician, Radiology, Neurology etc.), varchar(255), not NULL
   - lastName: varchar(255), not NULL
   - firstName: varchar(255), not NULL
   - Relationship (Staff-Order)(01:0M): A Staff can have zero or multiple Orders to handle. An Order can only be handled by one and only one Staff. staffID is FK in Order.
   - Relationship (Patient-Staff)(0M:0M): A Patient can be handled by zero or multiple Staff and vice versa. Defined in Staff_Patients table.
   - Implemented by: Abhash Sharma
4. **Orders**: All the Orders made to Patients
   - orderID: int, auto_increment, unique, not NULL, PK
   - date: date, not NULL, date the Order was generated
   - time: time, not NULL, time the Order was generated
   - orderType: (prescription, x-ray, blood test, specialist appointment, etc.), varchar(255), not NULL
   - patientID: used to connect Patient to Order, not NULL, FK
   - doctorID: used to connect an Order to Doctor, not NULL, FK
   - staffID: NULL (until assignment), used to connect an Order to Staff, FK
   - Relationship (Order-Result)(11:11): An Order can have one and only one Result. And a Result can have one and only one Order. orderID is FK in Results
   - Implemented by: Abhash Sharma
5. **Results**: The Results of all the Orders made to Patients
   - resultID: int, auto_increment, unique, not NULL, PK
   - status (Received/in progress/Completed/Sent): varchar(255), not NULL
   - orderID: used to connect an Order to a Result, not NULL, FK

- date: NULL, date on which the Result was generated (on which the Order was fulfilled)
- accessedByDoctor: bool, FALSE, changed to TRUE only after doctorID in Order or patientID's primaryDoctorID accesses the Results.
- Implemented by: Abhash Sharma

6. **Staff_Patients**: Assigns a Staff to a Patient (M:M relationship table)
   - staffID: the Staff being assigned to a Patient, not NULL, FK
   - patientID: the Patient being assigned to a Doctor, not NULL, FK
   - Implemented by: Sanjay Ramanathan

Identify entities and M:M relationship to implement and assign to team members.

- Sanjay Ramanathan and Abhash Sharma will implement Tables: Doctors, Patients, Staff, Orders, Results, and Staff_Patients.
- The one many-to-many relationships from Step 1 is between Staff and Patients.
- Sanjay Ramanathan will implement the Staff_Patients 0M:0M relationship.
- Other site features such as the home page and error pages will be implemented by Sanjay Ramanathan and Abhash Sharma.