Group #6

CIS 9340 UWA

TERM PROJECT:

MEDIBLEND APP

Miguel Rivas (miguel.rivas@baruchmail.cuny.edu)

Balakumaran Kannan (balakumaran.kannan@baruchmail.cuny.edu)

Matthew Martinez (matthew.martinez4@baruchmail.cuny.edu)

Chhejom Sherpa (chhejom.sherpa@baruchmail.cuny.edu)

Table of Contents

Mediblend is a healthcare application that allows its users to access their medical records on a single platform. Users have access to their confidential medical history from all different doctors visited, insurance history and provider, medication, immunization, allergies, and primary care doctor information. The purpose of recollecting this information is to have a common platform for patients to access their health data wherever they are located and stop having to keep track of passwords depending on what information needs to be looked at. Providers would be able to connect and locate data on demand in a report instead of delaying the patient's care while their health records are located. Considerations are taken into what type of information is displayed based on who is accessing the patient's records to follow HIPAA standards.

Requirements

To build the Mediblend application, the requirements are as follows:

1. List of healthcare providers
2. List of health insurance providers
3. Invoice history
4. Medical history
5. Patient history
6. Pharmacies registered
7. Patient details
8. ICD-9 and ICD-10 codes built into database
9. Database Management: MYSQL Workbench
10. Login design and access control based on role (account)

Costs

1. Hardware Costs: Costs related to database servers, desktop computers, tablets/laptops, etc
2. DBMS software Costs: Costs related to the application and various interfacing modules in the DBMS
3. Tech Support and Implementation Assistance Costs: Costs related to IT Contractor, Attorney, Consultant, Hardware/Network Installation as well as teams dedicated maintenance for the lifecycle of the DB
4. Training Costs: Training the related physicians, nurses, and office staff to use the new software for inputting, reading, and modifying data
5. Additional Maintenance and support fee: Depends on the SLA agreement of the Software firm and Health care provider
6. Marketing is required for organizations and patients to adopt the application

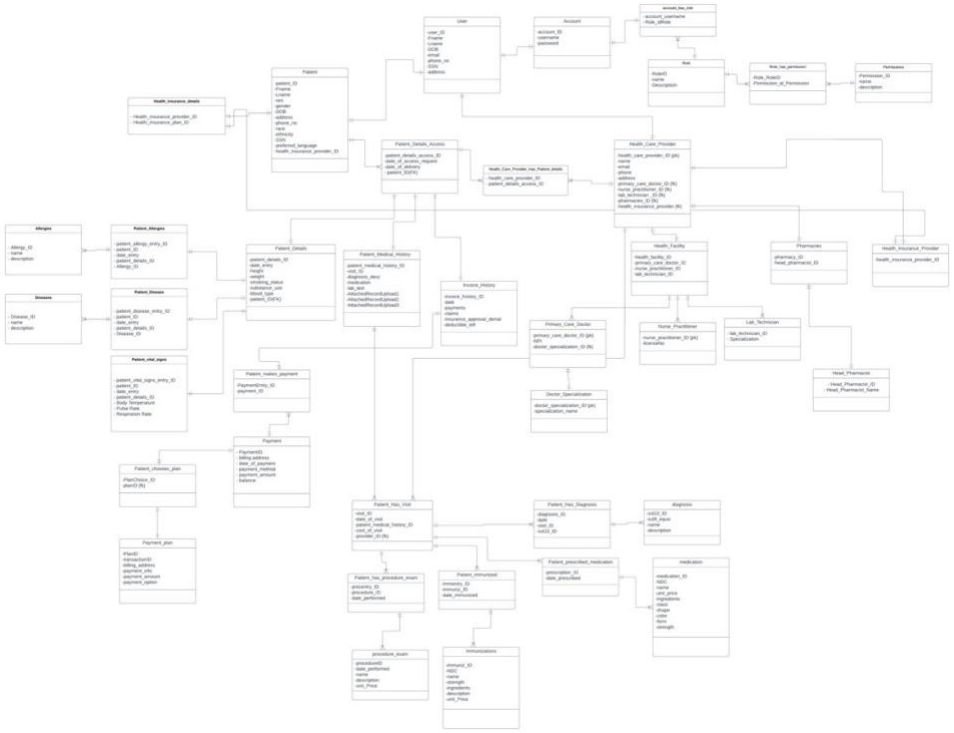
Benefits

1. Patients have quick and easy access to medical history when needing to supply new providers with their medical history
2. Can speed up admissions into medical offices such as emergency rooms as some patients struggle to remember their medical history
3. Payment management would be easier as oftentimes different medical offices use different apps to pay bills
4. Increased transparency patients now have more access to medication info, detailed surgical procedures, and other health related information in their chart
5. Patients can track their health history to avoid any incurable illness

6. Insurance and billing information is transparent as oftentimes patients are left in the dark about how or why there are certain charges and must go on a scavenger hunt with the hospital, provider and insurance company to get an answer

Conceptual Diagram (Lucid Chart)

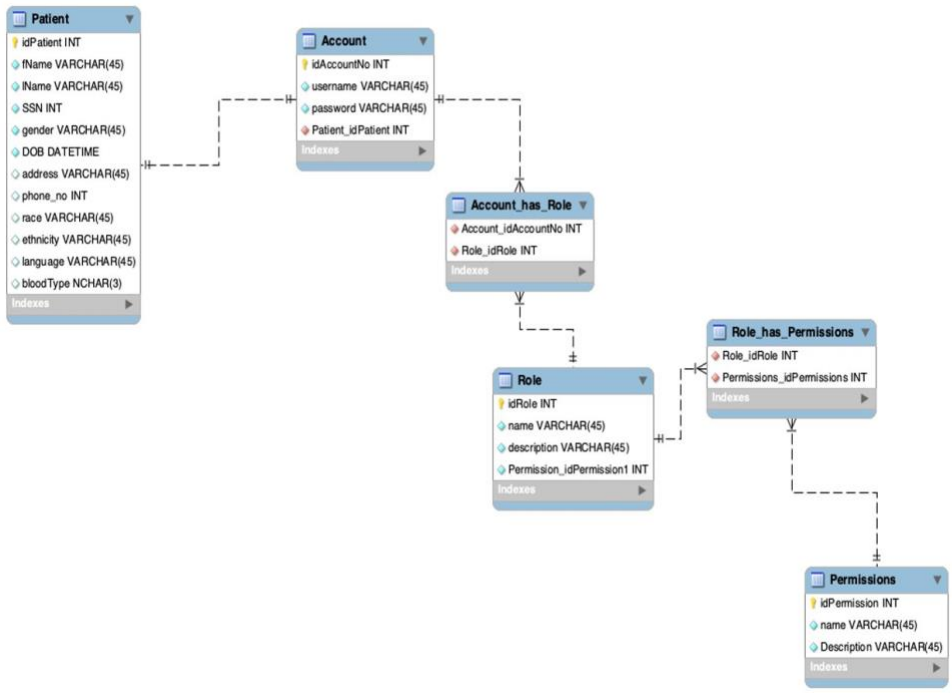
The rough draft consists of 40 business entities



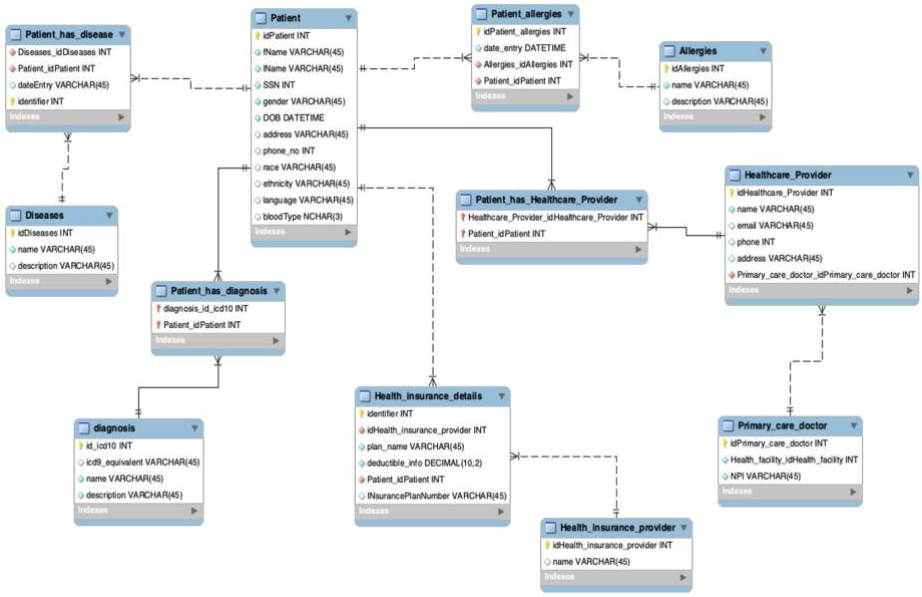
Logical Model

The following are the screenshots for Logical and Physical Model. As the entire diagram after appropriate normalizing and modifications was too huge to capture the diagram has been split into 4 sections for ease of understanding.

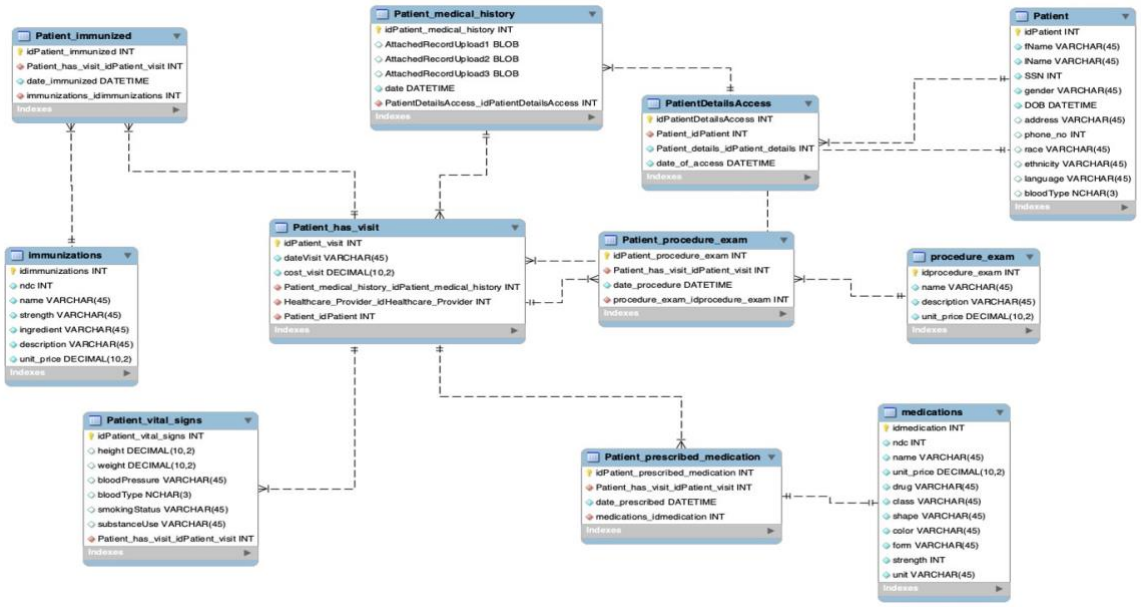
Patient and Account:



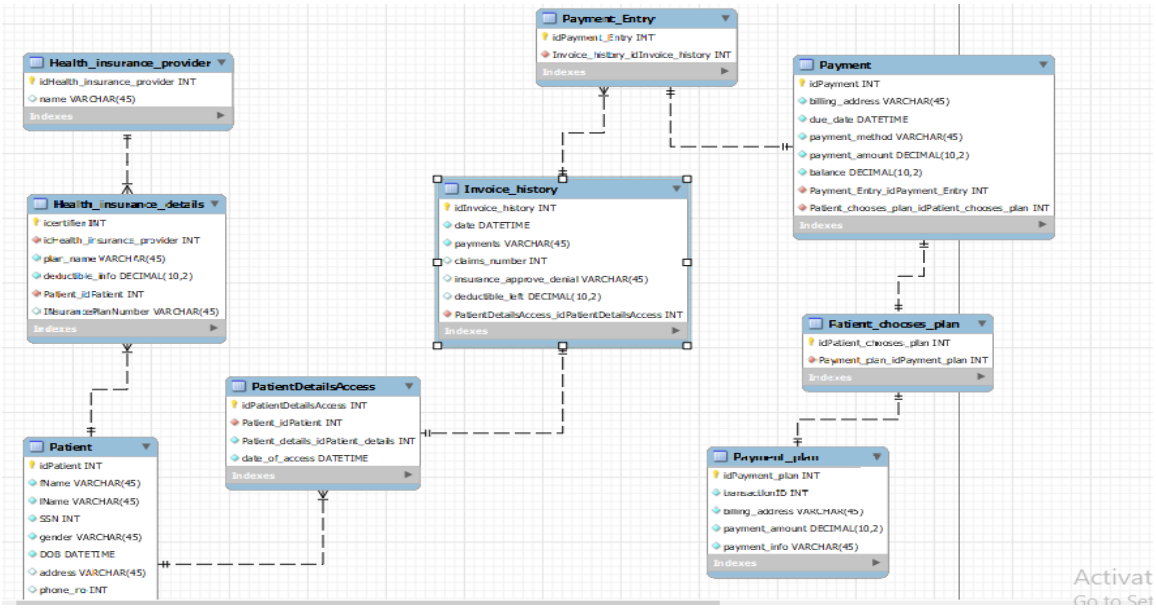
Patient and Disease/Allergies:



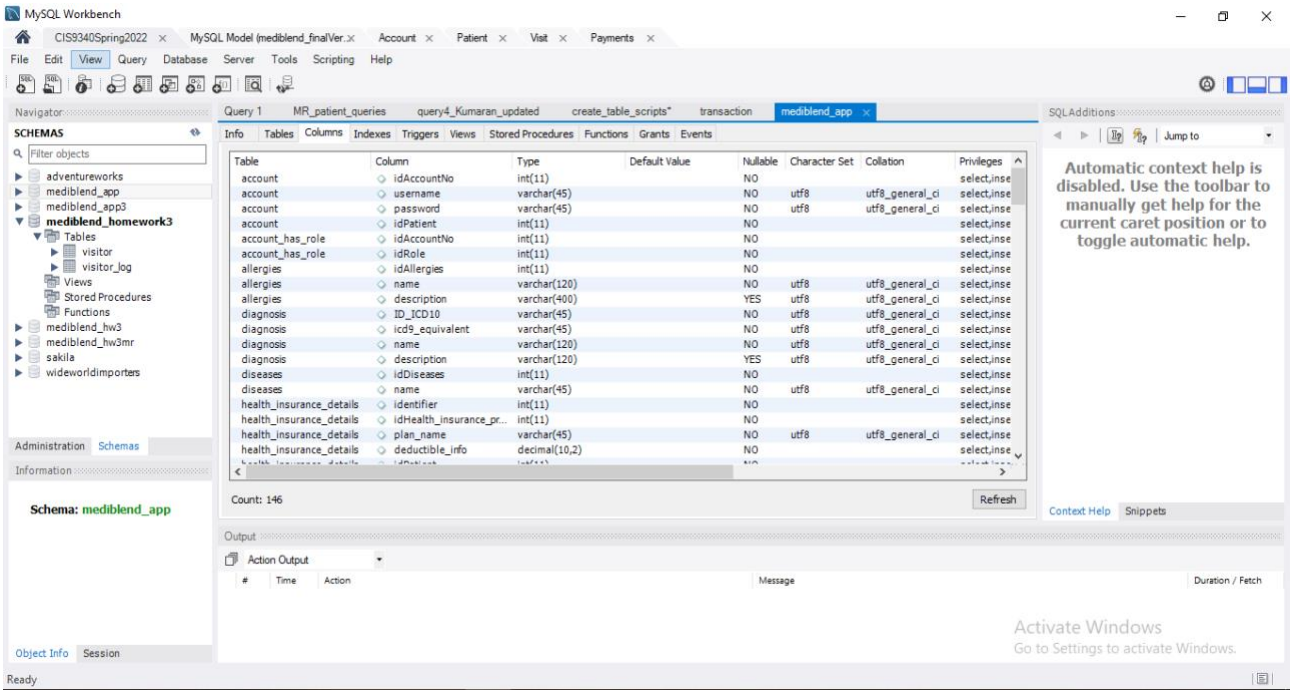
Patient and Visit:



Patient and Payments:



Physical Model



Model Relationships, Dependencies and Transformations

Most of the tables had to undergo the process of 1st Normalization and 2nd Normalization.

1st Normalization was applied to the following tables:

- 'Patient_has_visit' --> 'Patient_vital_signs'
- 'Patient_has_disease' --> 'Diseases' (created diseases to avoid duplicates in Patient_has_disease)
- 'Patient_allergies' --> 'Allergies'(created allergies to avoid duplicates in patient_allergies)
- 'Patient_immunization' --> 'immunization'
- 'Healthcare_Provider' --> 'Primary_care_doctor'.
- 'Patient_prescribed_medication' --> 'medications'

The arrow indicates how they were broken down further to achieve the degrees of normalization.

There were many intermediate relationships created to remove the many too many relationships as one can observe from the screenshots. For example, between patient and diseases there was a many too many relationships, so an intermediate table('Patient_has_disease') was created to achieve a one is too many on both ends. Similarly, 'Health_insurance_provider' and 'Patient' have an intermediate table named 'Health_insurance_details.

The purpose of implementing normalization was to identify the methods in which redundancy could be eliminated and reduce the inconsistent dependency.

A ‘user’ entity was created but later removed upon revision as it was the same as ‘Patient’ entity. Likewise, Lab technician, indicative rate notification, Onboarding, Nurse Practitioner, Fees, and Financial Statement. The entities were again grouped and depicted in 4 separate diagrams for better understanding and visualization. The 4 diagrams were Account, Patient, Visit, and Payments.

Example SQL DDL

Below is a snippet of DDL code used to create tables, along with primary key and foreign key constraints plus an insert statement for inserting data.

```
CREATE TABLE IF NOT EXISTS `mediblend_app`.`Patient` (
  `idPatient` INT NOT NULL,
  `fName` VARCHAR(45) NOT NULL,
  `lName` VARCHAR(45) NOT NULL,
  `SSN` VARCHAR(45) NOT NULL,
  `gender` VARCHAR(45) NOT NULL,
  `DOB` DATETIME NOT NULL,
  `address` VARCHAR(45) NULL,
  `phone_no` VARCHAR(45) NULL,
  `race` VARCHAR(45) NULL,
  `ethnicity` VARCHAR(45) NULL,
  `language` VARCHAR(45) NULL,
  `bloodType` VARCHAR(3) NULL,
  PRIMARY KEY (`idPatient`))
ENGINE = InnoDB;


-- Alter table to create FK constraint
ALTER TABLE `mediblend_app`.`patient_medical_history`
ADD CONSTRAINT `fk_idPatient`
  FOREIGN KEY (`idPatient`)
  REFERENCES `mediblend_app`.`patient` (`idPatient`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION;


INSERT INTO `mediblend_app`.`Patient` (idPatient, fName, lName, SSN, gender,
address, phone_no, race, ethnicity, language, bloodType) VALUES (1, “Dorette”,
“Hentze”, “730-26-7480”, “F”, “1990-01-09 00:00:00”, “0555 Declaration Court”,
“Philadelphia”, “PA”, 19184, “2151174755”. “Black”, “English”, “A-”);
```

Queries – 14 total queries

```
USE mediblend_app;

-- Average payment for year 2021
SELECT AVG(payment_amount)
FROM payment
WHERE YEAR (due_date) = 2021;
```

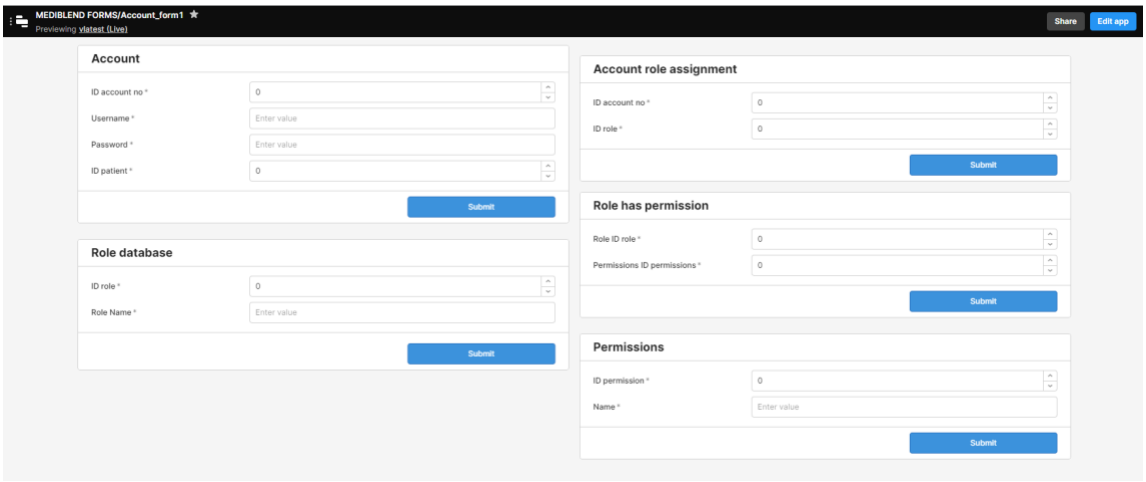
```
-- Insurance decisions for each patient
SELECT insurance_approve_denial, p.fname, p.lname
FROM patient p
INNER JOIN patientdetailsaccess pd ON pd.idPatient = p.idPatient
INNER JOIN invoice_history i ON i.idPatientDetailsAccess=
pd.idPatientDetailsAccess;

-- Balance and payment due date for patients
SELECT  p.fname, p.lname, pa.balance, pa.due_date
FROM patient p
INNER JOIN patientdetailsaccess pd ON pd.idPatient = p.idPatient
INNER JOIN invoice_history i ON i.idPatientDetailsAccess=
pd.idPatientDetailsAccess
INNER JOIN payment_entry pe ON pe.idInvoice_history= i.idInvoice_history
INNER JOIN payment pa ON pa.idPayment_Entry= pe.idPayment_Entry ;

-- Count of diseases of each patient
select p.idPatient,p.fname,p.lname,COUNT(d.name) as disease_count
from Patient p
INNER JOIN Patient_allergies pa ON p.idPatient = pa.idPatient
INNER JOIN Allergies a ON a.idAllergies = pa.idAllergies
INNER JOIN Patient_has_disease phd ON phd.Patient_idPatient = p.idPatient
INNER JOIN Diseases d ON d.idDiseases = phd.Diseases_idDiseases
GROUP BY p.idPatient,p.fname,p.lname

-- Details about a patient and related plan, health insurance details
select p.fname,p.lname, hid.plan_name,hip.Name as
company_name,hid.deductible_info
from patient p
INNER JOIN Health_insurance_details hid ON p.idPatient =  hid.idPatient
INNER JOIN Health_insurance_provider hip ON hip.idHealth_insurance_provider =
hid.idHealth_insurance_provider

-- Details of the patient's latest deductible info
select p.idPatient,p.fname,p.lname,ih.deductible_left,ih.date
from Patient p
INNER JOIN PatientDetailsAccess pda ON p.idPatient = pda.idPatient
INNER JOIN invoice_history ih  ON ih.idPatientDetailsAccess =
pda.idPatientDetailsAccess
WHERE (p.idPatient,ih.date) IN (select p.idPatient,MAX(ih.date)
from Patient p
INNER JOIN PatientDetailsAccess pda ON p.idPatient = pda.idPatient
INNER JOIN invoice_history ih  ON ih.idPatientDetailsAccess =
pda.idPatientDetailsAccess
GROUP BY  p.idPatient)

-- Description of the Patient's diagnosis info
select idPatient,fname,lname,id_icd10 ,d.description as diagnosis_info
from patient p
INNER JOIN Patient_has_diagnosis phd ON p.idPatient = phd.Patient_idPatient
```

```
INNER JOIN diagnosis d on d.id_icd10 = phd.ID_ICD10new

-- Show healthcare provider for each patient with their specializations
SELECT p.fName AS PatientFirstName,
       p.lName AS PaatientLastName,
       hp.name AS HealthCareProviderName,
       hps.specialization_name AS Specialization
FROM patient p
INNER JOIN patient_has_healthcare_provider php
ON php.idPatient = p.idPatient
INNER JOIN healthcare_provider hp
ON hp.idHealthcareProvider = php.idHealthcareProvider
INNER JOIN heathcare_provider_specialization hps
ON hps.idHealthcareProviderSpecialization = hp.idHealthcareProviderSpecialization
WHERE YEAR(DOB) > 1990
ORDER BY PatientFirstName;

-- Show healthcare provider name and specialization
SELECT hp.idHealthcareProvider,
       hp.name AS Name,
       hps.specialization_name
FROM healthcare_provider hp
INNER JOIN heathcare_provider_specialization hps
ON hps.idHealthcareProviderSpecialization = hp.idHealthcareProviderSpecialization
INNER JOIN healthcare_provider_subspecialization hpsub
ON hpsub.idHealthcareProviderSpecialization =
hps.idHealthcareProviderSpecialization
GROUP BY hp.idHealthcareProvider

-- View patient vital signs: needed to view the patient vital signs at each visit
for history
Select idPatient, fName FirstName, lName LastName, gender Gender, date(DOB) DOB,
date(dateVisit) dateVisit, height, weight, bloodPressure, bloodType,
smokingStatus, substanceUse
From patient
Inner join patient_has_visit using (idPatient)
Inner join patient_vital_signs using (idPatient_visit)
Order by fName;

-- View patient total cost from visits in 2020 to calculate historical visit cost
cumulatively
Select idPatient, fName FirstName, lName LastName, truncate(sum(cost_visit),2)
CostOfVisits
From patient
Inner join patient_has_visit using (idPatient)
Where Year(dateVisit)=2021
Group by idPatient
Order by fName;

-- View patient's medications for drug history
Select idPatient, fName FirstName, lName LastName, date(date_prescribed)
DatePrescribed, medications.name Drug
```

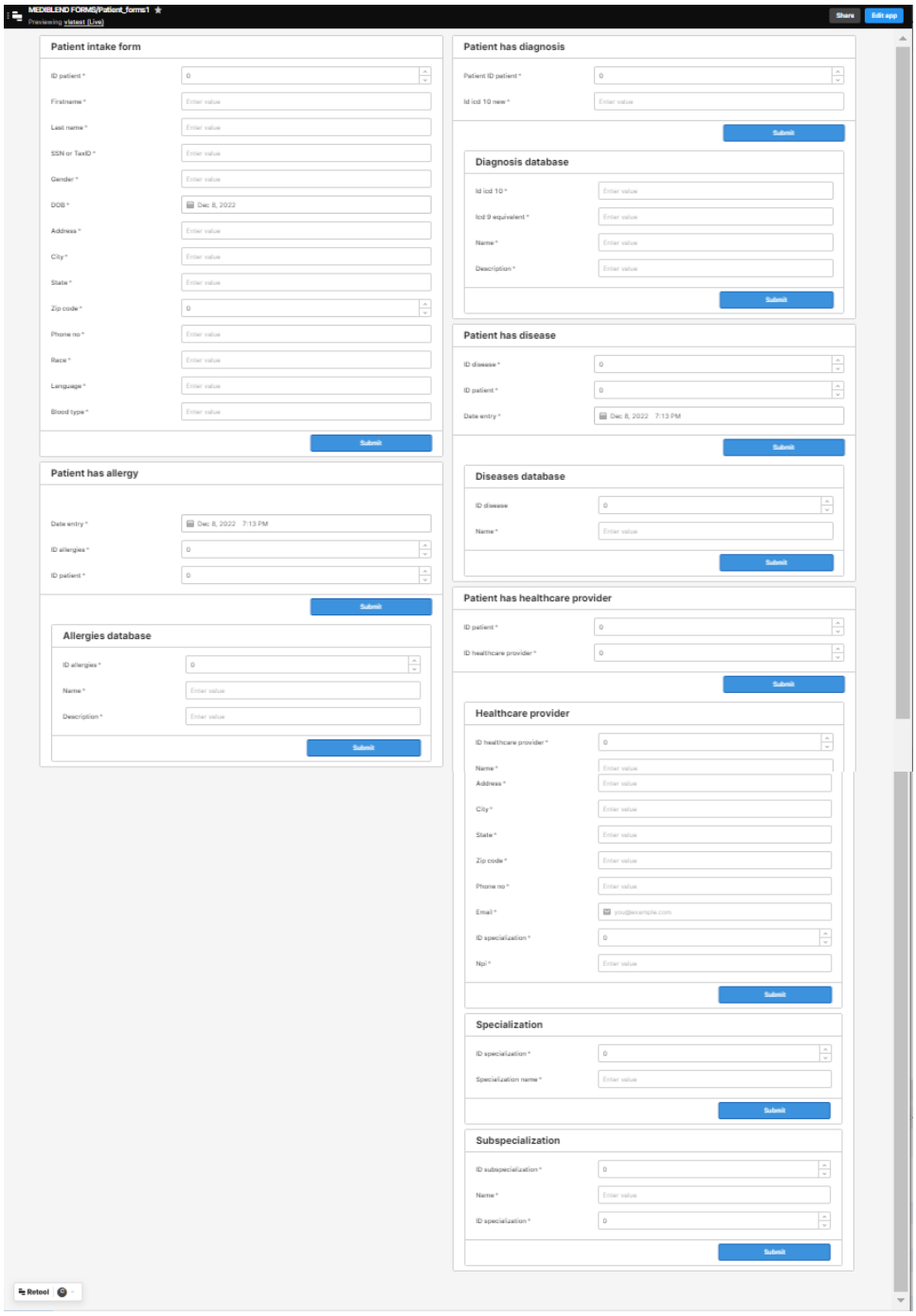
```
From patient
Inner join patient_has_visit using (idPatient)
Inner join Patient_prescribed_medication using (idPatient_visit)
Inner join medications using (idmedication)
Order by FirstName;

-- View patient's procedure exam for history
Select idPatient, fName FirstName, lName LastName, date(date_procedure)
DateProcedure, procedure_exam.name Name
From patient
Inner join patient_has_visit using (idPatient)
Inner join Patient_procedure_exam using (idPatient_visit)
Inner join procedure_exam using (idprocedure_exam)
Order by FirstName;

-- View patients immunizations for history on vaccines and recommend new
Select idPatient, fName, lName, date(DOB) DOB, date(date_immunized)
DateImmunized, immunizations.name Name
From patient
Inner join patient_has_visit using (idPatient)
Inner join Patient_immunized using (idPatient_visit)
Inner join immunizations using (idimmunizations)
Order by fName;
```

Navigation Forms

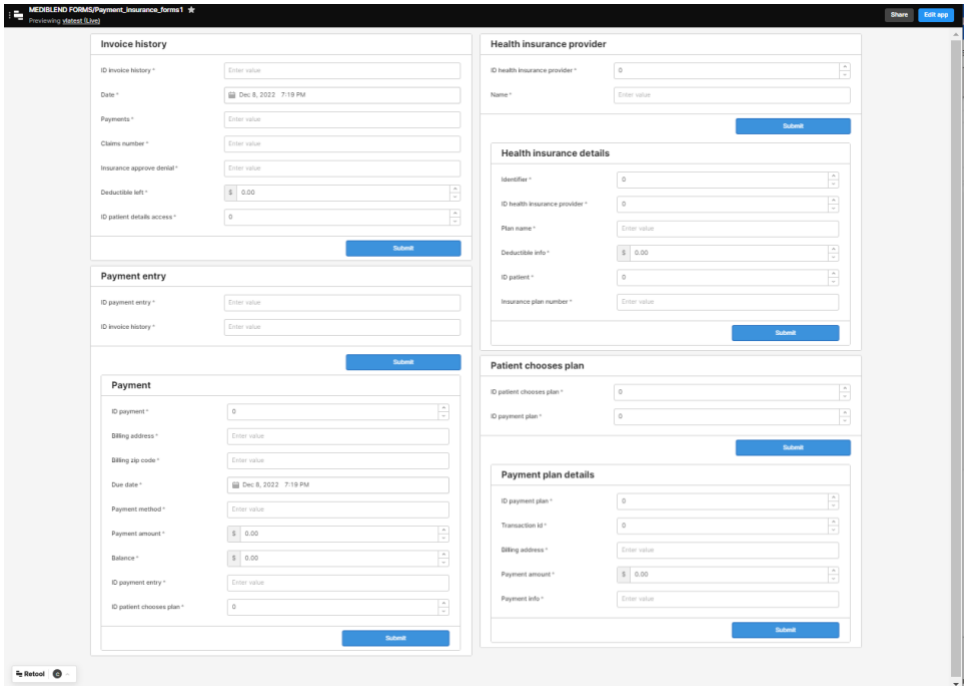
Account



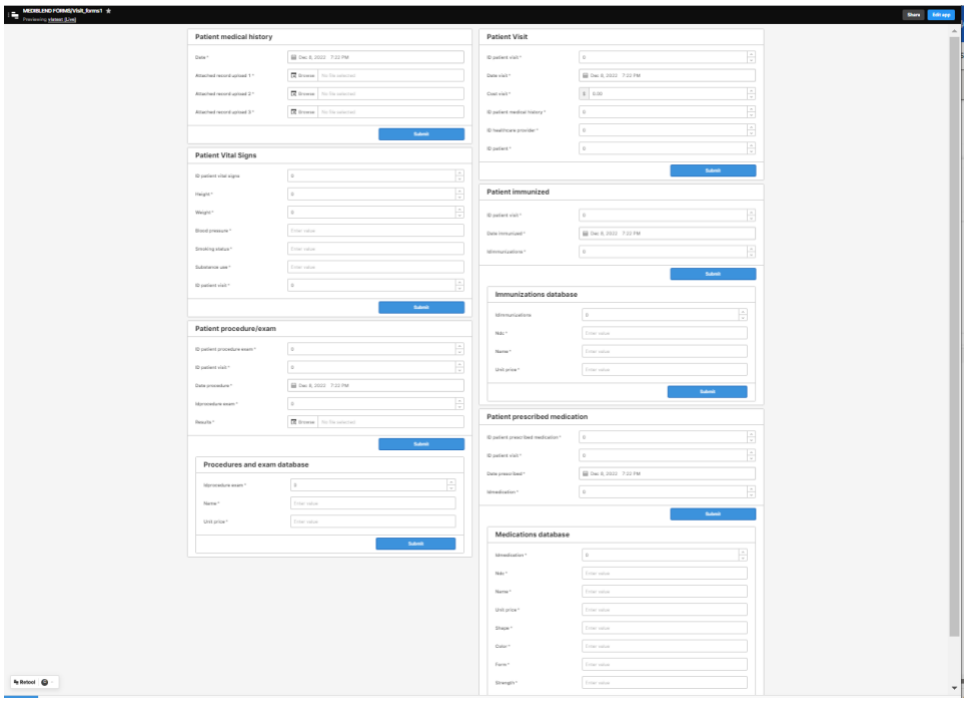
Patient


MEDBLEND FORMS/Patient_forms1
Share
Edit app
Patient intake form
ID patient *
Firstname *
Last name *
SSN or TaxID *
Gender *
DOB *
Address *
City *
State *
Zip code *
Phone no *
Race *
Language *
Blood type *
Submit
Patient has allergy
Date entry *
ID allergies *
ID patient *
Submit
Allergies database
ID allergies *
Name *
Description *
Submit
Patient has diagnosis
Patient ID patient *
Id icd 10 new *
Submit
Diagnosis database
Id icd 10 *
Icd 9 equivalent *
Name *
Description *
Submit
Patient has disease
ID disease *
ID patient *
Date entry *
Submit
Diseases database
ID disease
Name *
Submit
Patient has healthcare provider
ID patient *
ID healthcare provider *
Submit
Healthcare provider
ID healthcare provider *
Name *
Address *
City *
State *
Zip code *
Phone no *
Email *
ID specialization *
Npi *
Submit
Specialization
ID specialization *
Specialization name *
Submit
Subspecialization
ID subspecialization *
Name *
ID specialization *
Submit
Retool

Patient Insurance Forms



Visit



Conclusion

The consensus of the team regarding the hardest part of the project is normalization. Normalization required a lot of our time due to the many stages and processes we went through in order to reach 3NF. This required breaking down the many to many relationships and forming intermediate tables as well as deciding which tables should be kept or removed. Other issues that spurred from implementing the physical model in MySQL were due to the many errors coming from creating a foreign key in many tables and editing them once created. Lastly, inserting data also proved to be troublesome as we had to go through many iterations and solve the errors that were preventing us from importing data to MySQL. Furthermore, the easiest part of the project was the conceptual as we were able to simultaneously work on the model together using Lucid Chart.

We learned how a database is created from concept to implementation and what to consider before releasing for production. It must be carefully examined and proofed to avoid failures and security risks in order to maintain data integrity. In addition, we also constructed useful queries that would aid our business activities.

As shown in the queries above we do believe are proposed benefits can be realized in our new application. For example, one of the queries allows you to view the patients' total cost for a designated year and calculate their cumulative visit cost. This allows the patient to have a wider scope of all their medical visits and the charges they receive. We also provided a query that looks at all the medication a patient has received which will save the patient time when filling out a form or providing doctors with such information.