Pharmacy Chain Database

Design by Aisclepius

CST 363-30, Spring 2023

Kyle Absten, Juan Gutierrez

Project Scope and Definitions

Our data management consulting firm, Aisclepius, has developed a database system for your pharmacies, named "pharmaBase". pharmaBase is designed to manage and track important information about your doctors, patients, drugs, pharmacies, and the pharmaceutical companies your organization does business with, including details about the contracts and supervisors hired to manage them.

Entities

Patients: One of the key benefits of using this system is the ability to easily track and manage patient information. The "patient" table contains information such as patient ID, SSN, name, date of birth, address, and primary doctor ID. This allows your staff to quickly access and update patient information as needed, which can improve the efficiency of your operations and the level of care provided to patients. Additionally, the linking of primary doctor ID to patients allows you to track the patients that are being treated by each doctor.

Drugs: Our system gives you the ability to effectively manage and track your inventory of medications and other pharmaceutical products. The "drug" table contains information such as the formula, pharmaceutical company, and trade name of each drug, while the "pharmacy_sells_drug" table tracks which drugs each pharmacy sells, including the price. This allows you to easily track the availability of drugs across all your pharmacies and ensure that you have the necessary inventory on hand. Additionally, the ability to track the price of the drugs across different pharmacies can help you in pricing your products competitively.

Doctors: The pharmaBase system helps you track your customer's physicians for easy verification of prescriptions. The "doctor" table contains information about each doctor such as their ID, SSN, name, specialty and when they started practicing. This allows you to easily track and manage information about your doctors, including their specialties and how long they have been practicing. This information can be used to identify areas where additional doctors are needed or to ensure that your patients are being treated by qualified and experienced doctors.

Pharmacies: The "pharmacy" table contains information about each pharmacy such as their ID, name, address and phone number. This allows you to easily track and manage information about your pharmacies, including their location and contact information. This can be useful for identifying areas where additional pharmacies are needed or for managing the operations of your existing pharmacies.

Prescriptions: Our system not only handles information about current prescriptions and their remaining refills, but also archives every purchase ever made in one of your pharmacies. The aggregated information in our "rx" and "filledRx" tables will provide valuable insight into the purchasing habits of your customers and the productivity of your pharmacies.

All tables are linked together through foreign keys to ensure data consistency and integrity. This prevents any data inconsistencies and ensures that the data is accurate, up-to-date and reliable.

Front End

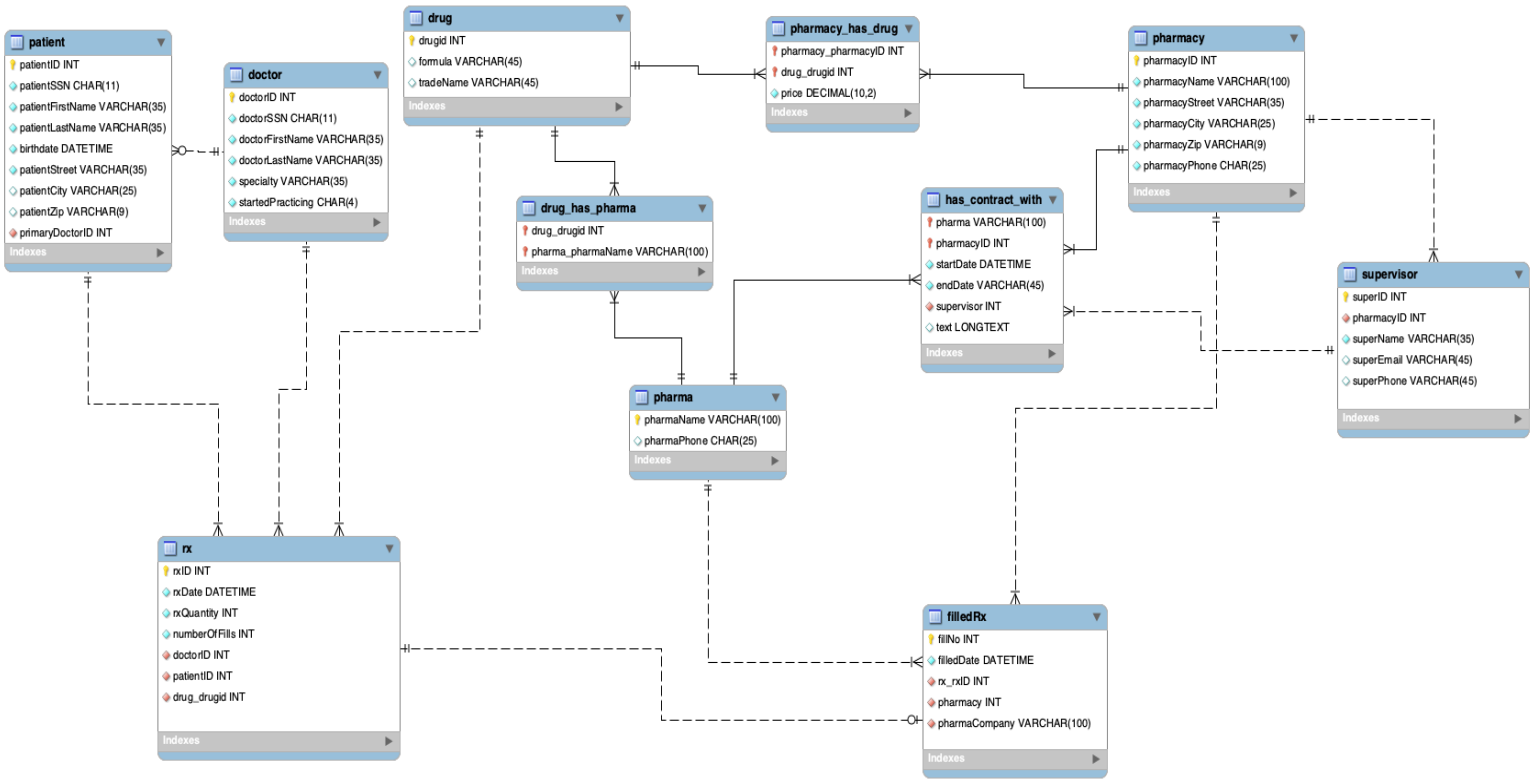
We at Aisclepius understand the importance of managing and tracking data for your pharmacy chain. That is why we propose to develop a front-end application that will enable your employees to easily access and manage the data stored in the "pharmaBase " database. This will enable them to carry out day-to-day operations more efficiently, while also providing regional managers and your administrative team with the ability to analyze the data for insights into your business.

The application will be developed on the platform of your choice, be it web or mobile, and will ensure that the following constraints are met:

- Any physician can prescribe drugs for any patient, regardless of the physician's specialty.
- Each pharmacy stocks multiple drugs and has a different price for each drug. The same drug may be sold at different pharmacies with varying prices.
- Prescriptions are written by doctors for individual patients, and each prescription is assigned a unique RX number. A patient may have multiple prescriptions written by different doctors.
- Prescriptions include the date they were written, the quantity of the drug prescribed, and the drug's trade name or generic formula. If the prescription lists a trade name, the prescription is for a specific drug made by a specific pharmaceutical company. If the prescription lists a generic formula, any drug with that formula made by any company can be used.
- When a prescription is filled, the pharmacy that filled the prescription, the date of filling, and the name of the pharmaceutical company that supplied the drug used to fill the prescription will be tracked.
- Pharmaceutical companies and pharmacies often have long-term contracts that include start and end dates and the contract details. A single pharmaceutical company can have contracts with multiple pharmacies, and a single pharmacy can have contracts with multiple pharmaceutical companies.
- Each contract is supervised by a designated supervisor, who may change over time and may be responsible for multiple contracts.

By using our "pharmaBase" system, your pharmacy chain will benefit from improved efficiency, better patient care, accurate drug inventory management, and streamlined doctor and pharmacy management. Overall, it will allow your company to make better and more informed decisions, and help you in running your business more effectively.

Entity Relationship Diagram



ERD Design Process

In designing our ERD, we referred to this source to determine attribute lengths: https://www.geekslop.com/technology-articles/2016/here-are-the-recommended-maximum-data-length-limits-for-common-database-and-programming-fields.

To satisfy the requirements of the client, we settled on nine entities. Patients and Doctors were given auto-generated Integer ID's as their primary key. While SSN seemed like an obvious choice for the primary key for these entities, we felt as though using sensitive data as a primary key may not be the best choice, so we opted for ID numbers. Every patient has a foreign key that refers to their primary doctor. Patients must have exactly one primary, but doctors can have from zero to many patients.

To keep track of all the drugs in the database, we created a drug entity. Based on the requirements, some drugs may not have a trade name, but they will all have a formula name. Because of this, we chose to make the primary key for the drug table a composite primary key made up of two foreign keys, the formula name of the drug and the name of the pharmaceutical company that manufactures it. This will work if each pharmaceutical company only manufactures each formula under one trade name. This is an assumption we have made, and the ERD will need to be refactored if this is not a safe assumption.

For the branches of the pharmacy, we decided to use a unique, auto-generated Integer ID. The reasoning is that if this is a chain of pharmacies, they may all be named the same. This entity does not contain any foreign keys.

For the pharma companies, we used the name of the company as the primary key. This entity does not contain any foreign keys.

Our rx entity will keep track of all prescriptions written. This entity has an auto-generated Integer as the primary key. This seemed to be the most logical primary key, as doctors may write multiple prescriptions for the same patient for the same drug over time. The rx entity has four foreign keys: patientID, doctorID, drugFormula, and tradeName. The foreign key referencing the trade name may be null, as it was indicated that some prescriptions will be written just for the generic formula. We have also included a refills attribute which is an integer that is Non-Null and keeps track of the number of refills prescribed/remaining.

As prescriptions get filled, they will be recorded in the filledRx entity. The primary key for this entity will be a composite key of a foreign key referencing the rxID of the prescription being filled and the filledDate attribute. rxID is a one-to-one relationship where every filledRx must participate, but an rx may not if it hasn't been filled yet. We also included a foreign key pharmacy, which references the pharmacyID of the pharmacy that filled the order. This is a one-to-many relationship where every filledRx must participate and have only one pharmacy that filled the order. On the pharmacy side of the relationship, we left it as optional and up to many. This is because we may want to enter a pharmacy in the system before it is ready to fill prescriptions, and each pharmacy will likely fill many prescriptions over time. Lastly, the foreign key pharmaCompany indicates which pharma company manufactured the drug that was used to fill the order.

To keep track of the price of each drug entity at each pharmacy, we implemented the pharmacy_sells_drug table. The primary key for this entity is a composite of three foreign keys: pharmacyID, drug_formula, and pharmaCompany. There is also an attiribute of type DECIMAL called price. This should allow us to track all the different prices for all the different drugs at each pharmacy.

To record all the contracts between pharmacies and pharma companies, we used a has_contract_with entity. The primary key of this entity is a composite of two foreign keys: pharma company name and pharmacyID. This entity will also contain a foreign key referencing a superID which will indicate the assigned supervisor for this contract. We included an attribute of type LONGTEXT to record the text of the contract.

We decided to make the supervisor a separate entity, as we saw the chance for a supervisor to exist who does not currently have any contracts assigned to them. In this scenario, if the supervisor attribute only existed in the contract entity, then there would be no record of this supervisor, and the pharmacy may want to keep track of all supervisors on staff. This entity will have an autogenerated Integer ID, as two supervisors with the same name might work at the same pharmacy. We also included the foreign key pharmacyID to indicate which pharmacy the supervisor works for.

The following constraints could not be represented in the ERD:

- has_contract_with:
    - startDate must be before endDate
    - pharmacyID must match supervisor.pharmacyID

- pharmacy_sells_drug:
    - price >= 0.01

- rx:
    - quantity > 0
    - refills >= 0

Normalization

Developing a database starting from an ERD helps minimize normalization issues, which in our case meant that there was very little to do in terms of optimizing how we stored data. The only exception was that our rx and filledRx entities began as one, with nullable filledDate, pharmacy, and pharmaCompany fields. Separating these entities means gaining the ability to track each prescription as they are filled and refilled.

SQL Schema

```
-- MySQL Script generated by MySQL Workbench
-- Sun Feb  5 11:43:56 2023
-- Model: New Model     Version: 1.0
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-- -----------------------------------------------------
-- Schema 363project2
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema 363project2
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `363project2` DEFAULT CHARACTER SET utf8 ;
USE `363project2` ;

-- -----------------------------------------------------
-- Table `drug`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `drug` (
  `drugid` INT NOT NULL AUTO_INCREMENT,
  `formula` VARCHAR(45) NULL,
  `tradeName` VARCHAR(45) NULL,
  PRIMARY KEY (`drugid`),
  UNIQUE INDEX `tradeName_UNIQUE` (`tradeName` ASC) VISIBLE)
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `doctor`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `doctor` (
  `doctorID` INT NOT NULL AUTO_INCREMENT,
  `doctorSSN` CHAR(11) NOT NULL,
  `doctorFirstName` VARCHAR(35) NOT NULL,
  `doctorLastName` VARCHAR(35) NOT NULL,
  `specialty` VARCHAR(35) NOT NULL,
  `startedPracticing` CHAR(4) NOT NULL,
  PRIMARY KEY (`doctorID`))
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `patient`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `patient` (
  `patientID` INT NOT NULL AUTO_INCREMENT,
  `patientSSN` CHAR(11) NOT NULL,
  `patientFirstName` VARCHAR(35) NOT NULL,
  `patientLastName` VARCHAR(35) NOT NULL,
  `birthdate` DATETIME NOT NULL,
  `patientStreet` VARCHAR(35) NOT NULL,
  `patientCity` VARCHAR(25) NULL,
  `patientZip` VARCHAR(9) NULL,
  `primaryDoctorID` INT NOT NULL,
  PRIMARY KEY (`patientID`),
  INDEX `fk_patient_doctor1_idx` (`primaryDoctorID` ASC) VISIBLE,
  CONSTRAINT `fk_patient_doctor1`
    FOREIGN KEY (`primaryDoctorID`)
    REFERENCES `doctor` (`doctorID`)
    ON DELETE RESTRICT
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `pharma`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `pharma` (
`pharmaName` VARCHAR(100) NOT NULL,
`pharmaPhone` CHAR(25) NULL,
PRIMARY KEY (`pharmaName`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `pharmacy`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `pharmacy` (
`pharmacyID` INT NOT NULL AUTO_INCREMENT,
`pharmacyName` VARCHAR(100) NOT NULL,
`pharmacyStreet` VARCHAR(35) NOT NULL,
`pharmacyCity` VARCHAR(25) NOT NULL,
`pharmacyZip` VARCHAR(9) NOT NULL,
`pharmacyPhone` CHAR(25) NOT NULL,
PRIMARY KEY (`pharmacyID`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `rx`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `rx` (
`rxID` INT NOT NULL AUTO_INCREMENT,
`rxDate` DATETIME NOT NULL,
`rxQuantity` INT NOT NULL,
`numberOfFills` INT NOT NULL,
`doctorID` INT NOT NULL,
`patientID` INT NOT NULL,
`drug_drugid` INT NOT NULL,
PRIMARY KEY (`rxID`),
INDEX `fk_rx_patient1_idx` (`patientID` ASC)
VISIBLE,
INDEX `fk_rx_doctor1_idx` (`doctorID` ASC) VISIBLE,
INDEX `fk_rx_drug1_idx` (`drug_drugid` ASC)
VISIBLE,
CONSTRAINT `fk_rx_patient1`
FOREIGN KEY (`patientID`)
REFERENCES `patient` (`patientID`)
ON DELETE RESTRICT
ON UPDATE RESTRICT,
CONSTRAINT `fk_rx_doctor1`
FOREIGN KEY (`doctorID`)
REFERENCES `doctor` (`doctorID`)
ON DELETE RESTRICT
ON UPDATE RESTRICT,
CONSTRAINT `fk_rx_drug1`
FOREIGN KEY (`drug_drugid`)
REFERENCES `drug` (`drugid`)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `filledRx`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `filledRx` (
`fillNo` INT NOT NULL AUTO_INCREMENT,
`filledDate` DATETIME NOT NULL,
`rx_rxID` INT NOT NULL,
`pharmacy` INT NOT NULL,
`pharmaCompany` VARCHAR(100) NOT NULL,
INDEX `fk_filledRx_pharmacy1_idx` (`pharmacy`
ASC) VISIBLE,
INDEX `fk_filledRx_pharma1_idx` (`pharmaCompany`
ASC) VISIBLE,
PRIMARY KEY (`fillNo`),
CONSTRAINT `fk_filledRx_rx1`
FOREIGN KEY (`rx_rxID`)
REFERENCES `rx` (`rxID`)
ON DELETE RESTRICT
ON UPDATE RESTRICT,
CONSTRAINT `fk_filledRx_pharmacy1`
FOREIGN KEY (`pharmacy`)
REFERENCES `pharmacy` (`pharmacyID`)
ON DELETE RESTRICT
ON UPDATE RESTRICT,
CONSTRAINT `fk_filledRx_pharma1`
FOREIGN KEY (`pharmaCompany`)
REFERENCES `pharma` (`pharmaName`)
ON DELETE RESTRICT
ON UPDATE RESTRICT)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `supervisor`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `supervisor` (
`superID` INT NOT NULL AUTO_INCREMENT,
`pharmacyID` INT NOT NULL,
`superName` VARCHAR(35) NOT NULL,
`superEmail` VARCHAR(45) NULL,
`superPhone` VARCHAR(45) NULL,
PRIMARY KEY (`superID`),
INDEX `fk_supervisor_pharmacy1_idx` (`pharmacyID`
ASC) VISIBLE,
CONSTRAINT `fk_supervisor_pharmacy1`
FOREIGN KEY (`pharmacyID`)
REFERENCES `pharmacy` (`pharmacyID`)
ON DELETE CASCADE
ON UPDATE CASCADE)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `has_contract_with`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `has_contract_with` (
  `pharma` VARCHAR(100) NOT NULL,
  `pharmacyID` INT NOT NULL,
  `startDate` DATETIME NOT NULL,
  `endDate` VARCHAR(45) NOT NULL,
  `supervisor` INT NOT NULL,
  `text` LONGTEXT NULL,
  PRIMARY KEY (`pharma`, `pharmacyID`),
  INDEX `fk_pharma_has_pharmacy_pharmacy1_idx`
    (`pharmacyID` ASC) VISIBLE,
  INDEX `fk_pharma_has_pharmacy_pharma1_idx`
    (`pharma` ASC) VISIBLE,
  INDEX `fk_has_contract_with_supervisor1_idx`
    (`supervisor` ASC) VISIBLE,
  CONSTRAINT `fk_pharma_has_pharmacy_pharma1`
    FOREIGN KEY (`pharma`)
    REFERENCES `pharma` (`pharmaName`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT
  `fk_pharma_has_pharmacy_pharmacy1`
    FOREIGN KEY (`pharmacyID`)
    REFERENCES `pharmacy` (`pharmacyID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `fk_has_contract_with_supervisor1`
    FOREIGN KEY (`supervisor`)
    REFERENCES `supervisor` (`superID`)
    ON DELETE RESTRICT
    ON UPDATE RESTRICT)
  ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `drug_has_pharma`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `drug_has_pharma` (
  `drug_drugid` INT NOT NULL,
  `pharma_pharmaName` VARCHAR(100) NOT NULL,
  PRIMARY KEY (`drug_drugid`,
    `pharma_pharmaName`),
  INDEX `fk_drug_has_pharma_pharma1_idx`
    (`pharma_pharmaName` ASC) VISIBLE,
  INDEX `fk_drug_has_pharma_drug1_idx`
    (`drug_drugid` ASC) VISIBLE,
  CONSTRAINT `fk_drug_has_pharma_drug1`
    FOREIGN KEY (`drug_drugid`)
    REFERENCES `drug` (`drugid`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_drug_has_pharma_pharma1`
    FOREIGN KEY (`pharma_pharmaName`)
    REFERENCES `pharma` (`pharmaName`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
  ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `pharmacy_has_drug`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS
  `pharmacy_has_drug` (
  `pharmacy_pharmacyID` INT NOT NULL,
  `drug_drugid` INT NOT NULL,
  `price` DECIMAL(10,2) NOT NULL,
  PRIMARY KEY (`pharmacy_pharmacyID`,
    `drug_drugid`),
  INDEX `fk_pharmacy_has_drug_drug1_idx`
    (`drug_drugid` ASC) VISIBLE,
  INDEX `fk_pharmacy_has_drug_pharmacy2_idx`
    (`pharmacy_pharmacyID` ASC) VISIBLE,
  CONSTRAINT `fk_pharmacy_has_drug_pharmacy2`
    FOREIGN KEY (`pharmacy_pharmacyID`)
    REFERENCES `pharmacy` (`pharmacyID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_pharmacy_has_drug_drug1`
    FOREIGN KEY (`drug_drugid`)
    REFERENCES `drug` (`drugid`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
  ENGINE = InnoDB;


SET SQL_MODE=@OLD_SQL_MODE;
SET
FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_C
HECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Our database can be used to query many useful selections of your data. Some examples include:

-Show a list of the top 5 patients with the most prescriptions

```
SELECT patient.patientFirstName, patient.patientLastName, COUNT(rx.rxID) AS total_prescriptions
FROM patient
JOIN rx ON patient.patientID = rx.patientID
GROUP BY patient.patientID
ORDER BY total_prescriptions DESC
LIMIT 5;
```

-Show the 5 most profitable pharmacies in the database

```
SELECT pharmacy.pharmacyName,
   SUM(pharmacy_has_drug.price * rx.rxQuantity) as 'Total Revenue'
FROM rx
   JOIN filledRx ON rx.rxID = filledRx.rx_rxID
   JOIN drug ON rx.drug_drugid = drug.drugid
   JOIN pharmacy_has_drug ON drug.drugid = pharmacy_has_drug.drug_drugid
   JOIN pharmacy ON filledRx.pharmacy = pharmacy.pharmacyID
GROUP BY pharmacy.pharmacyName
ORDER BY 'Total Revenue' DESC LIMIT 5;
```

-Show the top 5 doctors who prescribe the most to patients.

```
SELECT doctor.doctorFirstName, doctor.doctorLastName,
  COUNT(rx.rxID) as 'Total Prescriptions'
FROM rx
  JOIN doctor ON rx.doctorID = doctor.doctorID
GROUP BY doctor.doctorID
ORDER BY 'Total Prescriptions' DESC LIMIT 5;
```

-Show the average age of the patients with a prescription for each drug.

```
SELECT drug.drugid, drug.formula
  AVG(DATEDIFF(NOW(), patient.birthdate) / 365) as 'Average Age'
FROM rx
  JOIN patient ON rx.patientID = patient.patientID
  JOIN drug ON rx.drug_drugid = drug.drugid AND
GROUP BY drug.drugid;
```

- Show the pharmacy with the most sales in each city

```
SELECT pharmacy.pharmacyCity, pharmacy.pharmacyName, COUNT(filledRx.fillNo) as 'Total Rx
Filled’
FROM filledRx
   JOIN pharmacy ON filledRx.pharmacyID = pharmacy.pharmacyID
GROUP BY pharmacy.pharmacyCity, pharmacy.pharmacyName
ORDER BY pharmacy.pharmacyCity, 'Total Rx Filled' DESC;
```

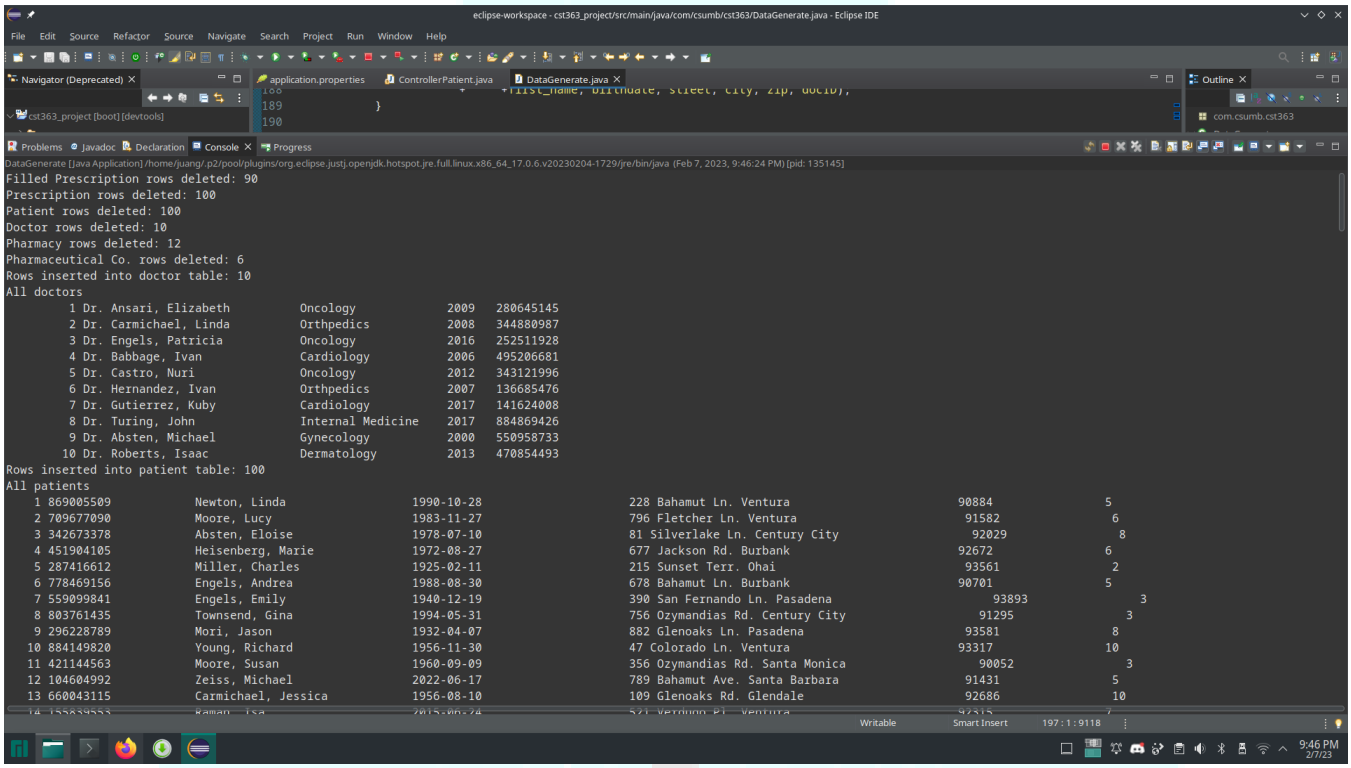
Part 2

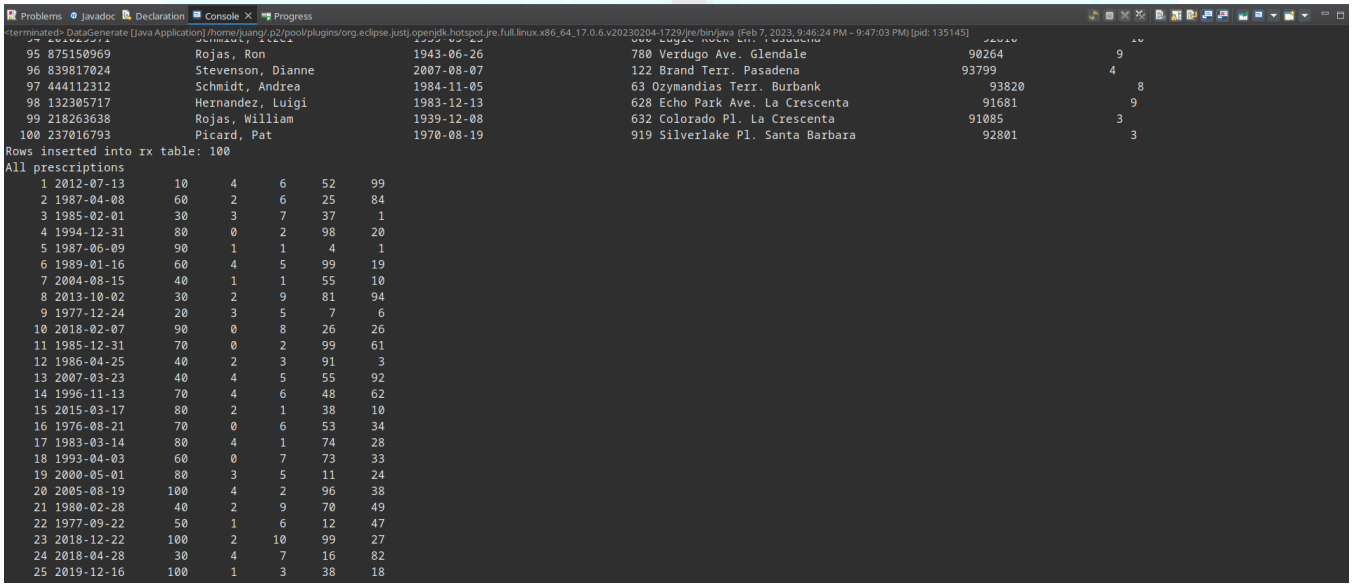
Java/Web Applications

In order to alpha test your new database, we created several Java applications and web applications:

DataGenerate

This Java app fills in random data into the database to play with, including 10 doctors, 100 patients, 100 prescriptions, 90 filled prescriptions, 12 pharmacies, and randomized prices for each drug at each pharmacy. It confirms doing so in the console:





```
Problems  Javadoc  Declaration  Console ×  Progress
<terminated> DataGenerate [Java Application] /home/juang/.p2/pool/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_17.0.6.v20230204-1729/jre/bin/java (Feb 7, 2023, 9:46:24 PM – 9:47:03 PM) [pid: 135145]
    95 2018-02-21       40      3      6     62     42
    96 2006-09-20       60      4     10     82     47
    97 1986-05-04       10      1     10     50     70
    98 2021-04-18      100      2      9     55     25
    99 2019-01-23       80      1      5      5      1
   100 1981-12-19       90      4      9     58      3
Rows inserted into filledRx table: 90
All filled prescriptions
     1 2022-07-16     84      2 Bayer
     2 2012-01-12     91      5 Bayer
     3 2009-01-12     17      7 Johnson & Johnson
     4 2000-07-22     44     12 Bayer
     5 1998-06-08     28      8 Eli Lilly
     6 2008-09-22     27     12 Johnson & Johnson
     7 2022-10-13     75      4 Pfizer
     8 2021-11-08      2      1 Pfizer
     9 1984-04-25      9      1 Johnson & Johnson
    10 1995-09-16     67      5 Eli Lilly
    11 2006-07-28     40     12 Johnson & Johnson
    12 2014-07-24     36      2 AstraZeneca
    13 1998-11-03     54      5 Pfizer
    14 2006-03-20     82     10 AstraZeneca
    15 1993-03-31     41      3 Bayer
    16 2004-02-15     73     10 Bayer
    17 2014-06-18     57      7 Bayer
    18 2022-02-10     58      7 Bayer
    19 1991-06-17     17      5 Bayer
    20 2018-11-07     85      9 Bayer
    21 2008-10-24     68      8 Merck
    22 1998-11-14     28     11 AstraZeneca
    23 1993-07-19     72      2 AstraZeneca
    24 2012-08-07     32     10 AstraZeneca
    25 2011-01-26     37     10 Eli Lilly
    26 2021-12-03     58      2 AstraZeneca
```

```
Problems  Javadoc  Declaration  Console ×  Progress
<terminated> DataGenerate [Java Application] /home/juang/.p2/pool/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_17.0.6.v20230204-1729/jre/bin/java (Feb 7, 2023, 9:46:24 PM – 9:47:03 PM) [pid: 135145]
    58 2005-05-04     83      8 Bayer
    59 2020-09-05     25      3 Eli Lilly
    60 2012-02-04     51      1 AstraZeneca
    61 1998-11-12     67      4 Johnson & Johnson
    62 2020-04-26     24     10 Bayer
    63 2008-04-11     66     10 Eli Lilly
    64 2012-09-13     19      2 Johnson & Johnson
    65 2016-12-07      3     11 Johnson & Johnson
    66 2017-04-20     48      4 Johnson & Johnson
    67 1997-02-13     34      2 AstraZeneca
    68 1984-09-10     76      6 Pfizer
    69 2005-04-10     97      2 Johnson & Johnson
    70 2021-05-08     62      3 Eli Lilly
    71 2002-05-30      2      5 Pfizer
    72 2021-08-25     91      2 Merck
    73 2020-11-10     29      6 Merck
    74 1984-09-24     46      7 AstraZeneca
    75 1978-04-11     37      1 Johnson & Johnson
    76 2016-03-08     38      8 Eli Lilly
    77 2011-06-05     21     11 AstraZeneca
    78 2020-06-24     75      4 Bayer
    79 1978-06-24     82      6 Eli Lilly
    80 2020-07-13     45     12 Bayer
    81 1994-09-19     51     10 Pfizer
    82 2005-02-05     21      3 Eli Lilly
    83 2020-12-06     42      4 AstraZeneca
    84 2005-12-08     79      4 AstraZeneca
    85 1986-01-27     11      7 Bayer
    86 2021-01-19     48     10 Pfizer
    87 2023-01-16     99      4 Eli Lilly
    88 2017-03-20     78      8 AstraZeneca
    89 2009-04-11     47      1 Pfizer
    90 2018-06-11     91      8 Merck
```

DataGenerate does not display the pharmacies, the pharmaceutical companies, or the prices inserted into the database. The prices alone would mean displaying 12 * 99 = 1188 rows!

Drugs_used_report

This Java application allows the manager of one of your pharmacies to retrieve information about how many of each medication has been dispensed at their pharmacy from a range of dates input by the user. First it validates the information entered and prompts the user to correct any missing/erroneous information, then displays the name and amounts for each medication processed at that particular pharmacy.

```
Please enter the pharmacy ID:
1
Please enter the starting date of the report(Format= MM/DD/YYYY):
10/28/1982
Please enter the ending date of the report(Format= MM/DD/YYYY):
10/29/1982
Sorry, there are no matching records for that time period
```

No matching records

```
Please enter the pharmacy ID:
28
That is not a valid pharmacy id. Please enter a valid pharmacy id:
```

When pharmacy id doesn't exist

```
Please enter the pharmacy ID:
1
Please enter the starting date of the report(Format= MM/DD/YYYY):
10/10/1899
Sorry that date is too long ago. We don't keep records that old.
That is not a valid date. Please enter a valid date in the form MM/DD/YYYY
10/28/1955
Please enter the ending date of the report(Format= MM/DD/YYYY):
10/28/1992
```

Date entered is before records were kept

```
Please enter the pharmacy ID:
2
Please enter the starting date of the report(Format= MM/DD/YYYY):
10/28/1982
Please enter the ending date of the report(Format= MM/DD/YYYY):
10/28/2022
Drugs used to fill prescriptions between 10/28/1982 and 10/28/2022:

Drug Name                                        Amount Used
-----------------------------------------------------------

glyburide                                            40
amitriptyline                                        60
benazepril and amlodipine                            90
lorazepam                                            90
```

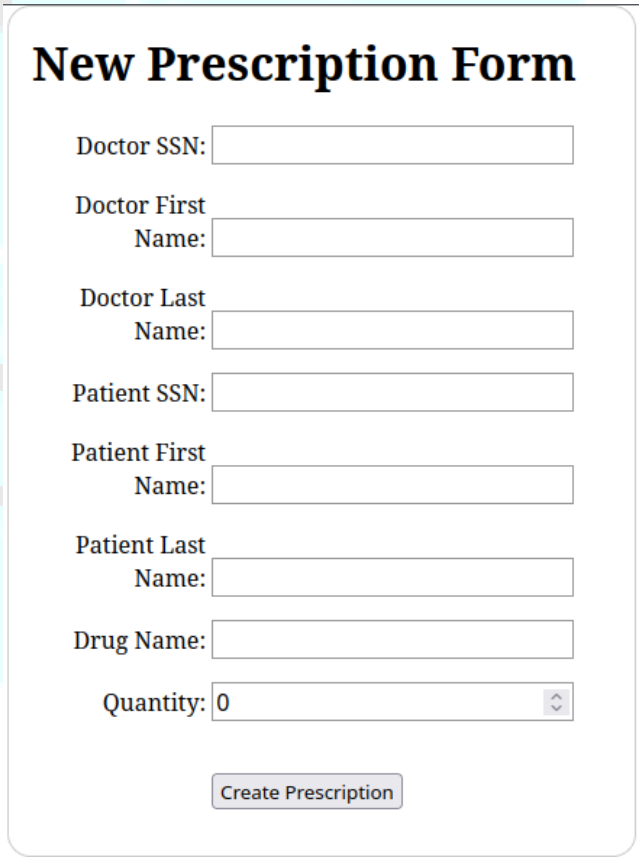
Successful retrieval of records

Web Application

As part of testing, we created a web portal to demonstrate the functionality of our database and how interactions with it would appear for different users, including doctors, patients, and pharmacy employees.

Starting with the portal:

← A doctor can create a new prescription.



Here, a doctor can input his identifying information, as well as the patient's information and what medication is being prescribed and in what amount. The web app validates all of the info before allowing the new prescription to be stored in the database.

In the next few pages, the different error messages can be seen which prompt the doctor as to what they need to correct.

New Prescription Form

Doctor SSN: 28064514

Doctor First Name: Elizabet

Doctor Last Name: Ansar

Patient SSN: 4262164061

Patient First Name: Rub

Patient Last Name: Watanab

Drug Name: lorazepam

Quantity: 20

Create Prescription

New Prescription Form

Invalid Doctor

Doctor SSN: 280645145

Doctor First Name: Elizabeth

Doctor Last Name: Ansari

Patient SSN: 4262164061

Patient First Name: Rub

Patient Last Name: Watanab

Drug Name: lorazepam

Quantity: 20

Create Prescription

New Prescription Form

Invalid Patient

Doctor SSN: 280645145

Doctor First Name: Elizabeth

Doctor Last Name: Ansari

Patient SSN: 426216406

Patient First Name: Ruby

Patient Last Name: Watanabe

Drug Name: lorazepam

Quantity: 20

Create Prescription

Prescription created.

Rx: 101
Doctor: 280645145
First Name: Elizabeth
Last Name: Ansari
Patient: 426216406
First Name: Ruby
Last Name: Watanabe
Drug: lorazepam
Quantity: 20
Pharmacy:
Name:
Address:
Phone:
Date Filled:
Cost: $

Main Menu

After being prompted to correct the information entered for the doctor and the patient, finally the prescription is created and stored in the database.

DrugStore Data System

Click on a choice below.

Write a new prescription (for Doctors only)

Request a prescription be filled (for Patients only)

Register as a new patient. (for Patients only)

Display patient data.

Register as new doctor. (for Doctors only)

Display doctor profile.

← A patient can request to fill a prescription.

Bringing up this request form. The patient enters his prescription number, their last name, and the information for the pharmacy where they want to fill the prescription.

Request Prescription be filled.

Enter pharmacy name, address and prescription Rx number.

Rx:

Patient Last Name:

Pharmacy Name:

Pharmacy Address:

Request Fill for Prescription

In the following pages, you will see the different error messages that appear, and the corrections made along the way until the prescription is successfully filled.

Request Prescription be filled.

Enter pharmacy name, address and prescription Rx number.

Rx: 1001

Patient Last Name: Watanab

Pharmacy Name: Turing Pharmacy

Pharmacy Address: 896 Deku Ave. Ohai 9514

Request Fill for Prescription

Request Prescription be filled.

Enter pharmacy name, address and prescription Rx number.

Prescription not found.

Rx: 101

Patient Last Name: Watanab

Pharmacy Name: Turing Pharmacy

Pharmacy Address: 896 Deku Ave. Ohai 9514

Request Fill for Prescription

Request Prescription be filled.

Enter pharmacy name, address and prescription Rx number.

Last name does not match prescription.

Rx: 101

Patient Last Name: Watanabe

Pharmacy Name: Turing Pharmacy

Pharmacy Address: 896 Deku Ave. Ohai 9514

Request Fill for Prescription

Request Prescription be filled.

Enter pharmacy name, address and prescription Rx number.

Pharmacy not found.

Rx: 101

Patient Last Name: Watanabe

Pharmacy Name: Turing's Pharmacy

Pharmacy Address: 889 Central Rd. Ventura 95627

Request Fill for Prescription

The error messages displayed in these images show the result of entering the previous image's information, and the corrected information prompted by the message.

Prescription filled.

Rx:	101
Doctor:	
First Name:	Elizabeth
Last Name:	Ansari
Patient:	
First Name:	Ruby
Last Name:	Watanabe
Drug:	Ativan
Quantity:	20
Pharmacy:	7
Name:	Turing's Pharmacy
Address:	889 Central Rd. Ventura 95627
Phone:	(213) 972-7400
Date Filled:	2023-02-07
Cost: $	$20.00

Main Menu

Finally, the prescription is filled. This is the same display page as when this prescription was created, with updated information about where it was filled and the price.

DrugStore Data System

Click on a choice below.

Write a new prescription (for Doctors only)

Request a prescription be filled (for Patients only)

Register as a new patient. (for Patients only)

Display patient data.

Register as new doctor. (for Doctors only)

Display doctor profile.

← A patient can register into the system.

Register as new user

SSN format is incorrect. Please enter as 123456789.

Your SSN:

Your First Name: Kyle

Your Last Name: Absten

Birth Date: 10/28/1982

Street: 123 Easy Street

City: Moorpark

State: California

Zipcode: 93021

Primary Physician Name: Charles Gupta

Register

This form prompts the new patient to input their information in order to register, and in this example the SSN was entered incorrectly, prompting this message.

Register as new user

First name is not formatted correctly.

Your SSN:

Your First Name: Kev7

Your Last Name: 7Abssten

Birth Date: mm/dd/yyyy

Street: 123

City: Moorpar7

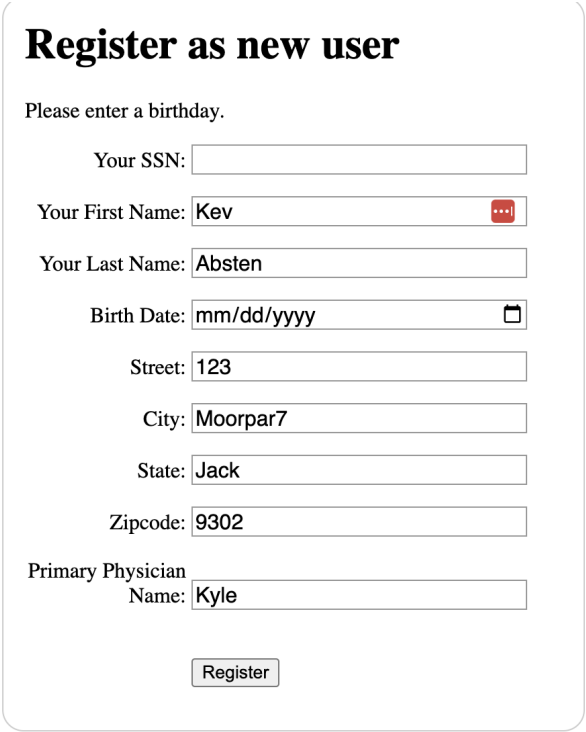
State: Jack

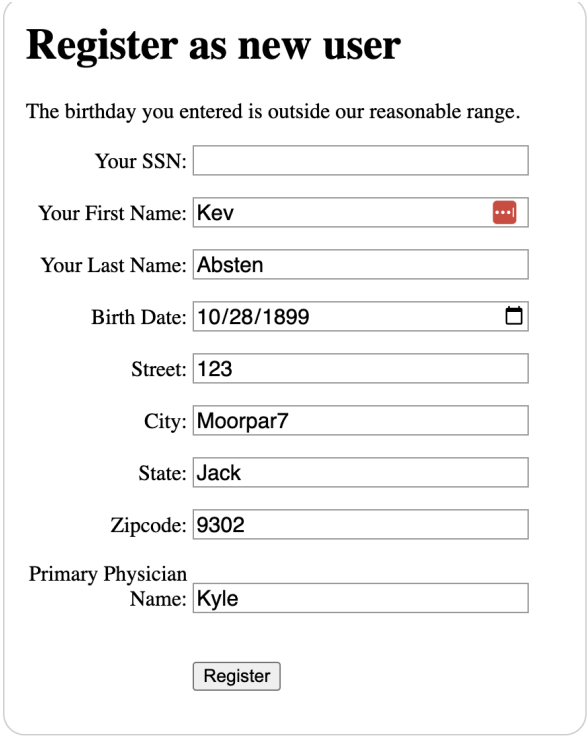
Zipcode: 9302

Primary Physician Name: Kyle

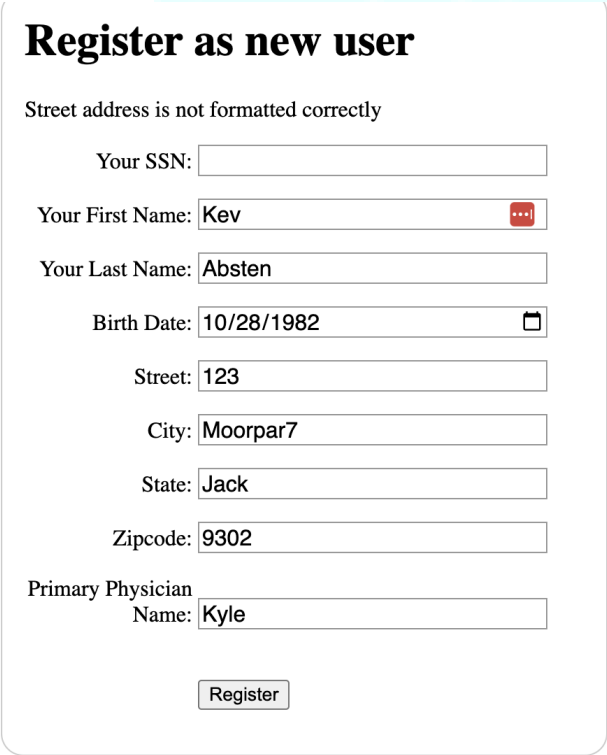
Register

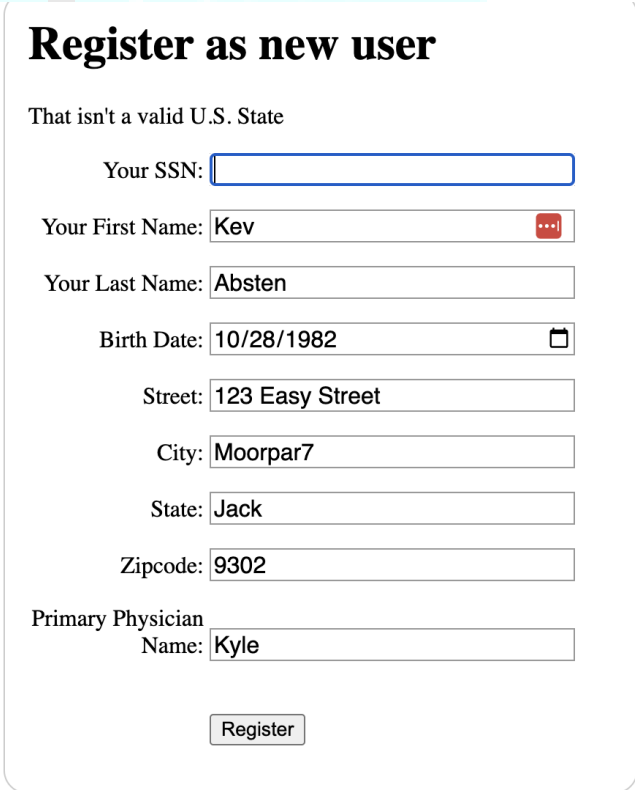
The application checks that the input names have only alphabetical characters with a regex.





The application ensures that a birthday is entered, and that it is a reasonable date.





We also confirm that the address entered is done so in the correct format, and that the information entered is valid.

Register as new user

Zip code is not formatted correctly

Your SSN:

Your First Name: Kev

Your Last Name: Absten

Birth Date: 10/28/1982

Street: 123 Easy Street

City: Moorpark

State: California

Zipcode: 9302

Primary Physician Name: Kyle

Register

This includes the Zip Code!

Register as new user

Doctor name is not formatted correctly. Please enter as Firstname Lastname

Your SSN:

Your First Name: Kev

Your Last Name: Absten

Birth Date: 10/28/1982

Street: 123 Easy Street

City: Moorpark

State: California

Zipcode: 93021

Primary Physician Name: Kyle

Register

It ensures the doctor's name is valid,

Register as new user

That doctor doesn't qualify as a primary doctor. Please choose another doctor

Your SSN:

Your First Name: Kev

Your Last Name: Absten

Birth Date: 10/28/1982

Street: 123 Easy Street

City: Moorpark

State: California

Zipcode: 93021

Primary Physician Name:

Register

and that the doctor's specialty is adequate for a primary care physician.

Registration successful.

Patient ID:	102
First Name:	Kev
Last Name:	Absten
Birthdate:	1982-10-28
Street:	123 Easy Street
City:	Moorpark
State:	California
Zipcode:	93021
Primary Physican:	Mitchell Young

Edit | Main Menu

Once all of the information is correct and validated, a new patient is registered into the database.

Conclusions

Developing this database was a rewarding challenge to take on. It was interesting to figure out which entities are needed and the relationships between them to get the appropriate level of granularity with the data for the scope of the project. Some considerations for the next version; as it stands we can query information about revenue from prescriptions filled at the different pharmacies, but we have no way to manipulate data on drugs that are over the counter and do not require a prescription. Further refactoring would be necessary to add this functionality, including adding a purchase table, a drug.needsRx boolean field, and a way to handle checking the purchases for a valid prescription for those drugs that require them, which is probably best solved as an app-based constraint. Creating a front end, both as a Java application and web application was edifying, we learned a great deal about how databases are linked to end-user applications.