

Relational Database for Drug Store Chain

Kevin Daill, Nathan D. Huven, James Meaden, and Josiah Sanchez

Computer Science Online, California State University, Monterey Bay

Professor David Wisneski

May 25, 2021

Grid has been contracted by Drug Store Chain (DSC) to design and implement a database management system. The purpose of this system is to provide a relational database capable of storing and managing the details related to DSC's business of selling drugs and filling prescriptions. As part of this initiative, Grid has prepared the following report. Part one of the report details out the project's requirements, the proposed entity relationship (ER) diagram, normalized relational schema, and SQL queries answering questions relevant to DSC needs. Part two provides details regarding the functionality of the application.

Part One

Requirements

Grid understands there to be several major components to the database. It must be capable of maintaining records of patients, doctors, DSC pharmacies, and pharmaceutical companies. In addition, it must be able to store records of the drugs DSC stores sell or are requested to fill. The database must also maintain records of the prescriptions DSC stores are being requested to fill along with records of each fill. Finally, the database must be capable of creating records containing the details of contracts DSC stores have with various pharmaceutical companies as well as details of the supervisors DSC stores have appointed to maintain each contract. Each of these components have more detailed requirements that are summarized in the following sections.

Patient Records

Patient records represent the person who is receiving/taking a particular drug and who is being treated by a doctor. Each patient record must have a SSN that will be used to uniquely identify each patient. In addition, each record will allow DSC to capture the patient's first name,

last name, age, and address (street address, city, state, and zip code). Finally, each patient record will identify which doctor is the patient's primary doctor.

Doctor Records

Doctor records represent the person who is prescribing medication to one or more patients. Each doctor record must have a SSN that will be used to uniquely identify each doctor. The records will also store the doctor's first name, last name, specialty, and number of years of experience.

Pharmacy (DSC Store) Records

Pharmacy records represent a particular location of a DSC store. Each pharmacy record will have an ID number that will be automatically generated to uniquely identify each pharmacy. These records will also capture the name, address (street address, city, state, zip code), and phone number of the pharmacy.

Pharmaceutical Company Records

Pharmaceutical company records represent a company that manufactures and sells drugs to pharmacies. Pharmaceutical company records will be uniquely identified by the pharmaceutical company's name and will also capture its phone number.

Drug Records

Drug records represent a particular trade name and formulation of a drug sold by a pharmaceutical company. These records capture the trade name and formula of the drug along with the name of the pharmaceutical company that sells the drug. In combination, the trade name and formula uniquely identify each record.

Drug Pricing Records

Drug pricing records will allow each unique pharmacy to set and store the price that it sells a particular drug for. (Note: A drug is identified by its trade name, formula, and pharmaceutical company name.) These records will be uniquely identified by the combination of the drug and the pharmacy.

Prescription Records

Prescription records represent the drug and amount of drug being prescribed to a patient by a provider. Each prescription record will capture the patient, doctor, drug, quantity, and date prescribed. Prescription records are uniquely identified by an automatically generated id. A patient can be prescribed many prescriptions and a doctor can write many prescriptions.

Prescription Fill Records

A prescription fill record represents an instance of a particular pharmacy filling a particular prescription. These records are uniquely identified by the prescription and the pharmacy. In addition, these records capture the date the prescription was filled.

Contract Records

Contract records represent the agreements between pharmaceutical companies and pharmacies. These records capture the pharmaceutical company and pharmacy along with the text of the contract and the start and end date of the contract. These records will have their own automatically generated id that uniquely identifies each contract. Many pharmaceutical companies may contract with many different pharmacies and vice versa. Each contract record will also contain which supervisor is responsible for the contract.

Supervisor Records

Supervisor records represent the person from the pharmacy that is responsible for a contract. These records capture the first and last name of the supervisor. Each record will have an automatically generated id that uniquely identifies the supervisor.

Relational Schema

-- Relational Schema derived from the ER Model.

--

-- First normal form is satisfied, there are no fields that contain data mismatches and
 -- at this point there is no data in the table therefore no duplicate fields or
 -- multiple entries in a field. The database contains no composite keys that other
 -- tables are dependent on and therefore in second normal form. Non key columns are
 -- not dependent on other non key columns in the table and therefore database is in third
 -- normal form.

```
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_FO
R_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';
```

-- -----

-- Schema Project1

-- -----

```
CREATE SCHEMA IF NOT EXISTS `Project1` DEFAULT CHARACTER SET utf8 ;
USE `Project1` ;
```

-- -----

-- Table `Project1`.`Doctor`

-- -----

```
CREATE TABLE IF NOT EXISTS `Project1`.`Doctor` (
  `doctorssn` INT NOT NULL,
  `firstname` VARCHAR(45) NOT NULL,
  `lastname` VARCHAR(45) NOT NULL,
  `specialty` VARCHAR(45) NOT NULL,
  `yearsofexperience` INT NOT NULL,
  PRIMARY KEY (`doctorssn`),
  UNIQUE INDEX `doctorssn_UNIQUE` (`doctorssn` ASC) VISIBLE)
ENGINE = InnoDB;
```

-- -----

-- Table `Project1`.`Patient`

-- -----

```
CREATE TABLE IF NOT EXISTS `Project1`.`Patient` (
  `patientssn` INT NOT NULL,
  `firstname` VARCHAR(45) NOT NULL,
  `lastname` VARCHAR(45) NOT NULL,
  `age` INT NOT NULL,
  `streetaddress` VARCHAR(255) NOT NULL,
  `city` VARCHAR(100) NOT NULL,
  `state` VARCHAR(45) NOT NULL,
  `zipcode` VARCHAR(45) NOT NULL,
  `Doctor_doctorssn` INT NOT NULL,
```

```

PRIMARY KEY (`patientssn`),
UNIQUE INDEX `patientssn_UNIQUE` (`patientssn` ASC) VISIBLE,
INDEX `fk_Patient_Doctor1_idx` (`Doctor_doctorssn` ASC) VISIBLE,
CONSTRAINT `fk_Patient_Doctor1`
  FOREIGN KEY (`Doctor_doctorssn`)
    REFERENCES `Project1`.`Doctor` (`doctorssn`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

-----
-- Table `Project1`.`Pharmaceutical Company`
-----

```

```

CREATE TABLE IF NOT EXISTS `Project1`.`Pharmaceutical Company` (
  `companyid` INT NOT NULL AUTO_INCREMENT,
  `companyname` VARCHAR(45) NOT NULL,
  `phonenummer` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`companyid`))
ENGINE = InnoDB;

```

```

-----
-- Table `Project1`.`Drug`
-----

```

```

CREATE TABLE IF NOT EXISTS `Project1`.`Drug` (
  `drugid` INT NOT NULL AUTO_INCREMENT,
  `tradenname` VARCHAR(45) NOT NULL,
  `formula` VARCHAR(45) NOT NULL,
  `companyid` INT NOT NULL,
  PRIMARY KEY (`drugid`),
  INDEX `fk_Drug_Pharmaceutical Company1_idx` (`companyid` ASC) VISIBLE,
  CONSTRAINT `fk_Drug_Pharmaceutical Company1`
    FOREIGN KEY (`companyid`)
      REFERENCES `Project1`.`Pharmaceutical Company` (`companyid`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

-----
-- Table `Project1`.`Pharmacy`
-----

```

```

CREATE TABLE IF NOT EXISTS `Project1`.`Pharmacy` (
  `pharmacyid` INT NOT NULL AUTO_INCREMENT,
  `pharmacynname` VARCHAR(45) NOT NULL,
  `pharmacyphonenummer` VARCHAR(45) NOT NULL,
  `streetaddress` VARCHAR(255) NOT NULL,
  `city` VARCHAR(100) NOT NULL,
  `state` VARCHAR(45) NOT NULL,
  `zipcode` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`pharmacyid`))
ENGINE = InnoDB;

```

```

-----
-- Table `Project1`.`Prescription`
-----

```

```

CREATE TABLE IF NOT EXISTS `Project1`.`Prescription` (
  `prescriptionid` INT NOT NULL AUTO_INCREMENT,
  `date` DATE NOT NULL,
  `quantity` INT NOT NULL,

```

```

`Drug_drugid` INT NOT NULL,
`Patient_patientssn` INT NOT NULL,
`Doctor_doctorssn` INT NOT NULL,
PRIMARY KEY (`prescriptionid`),
INDEX `fk_Prescription_Drug1_idx` (`Drug_drugid` ASC) VISIBLE,
INDEX `fk_Prescription_Patient1_idx` (`Patient_patientssn` ASC) VISIBLE,
INDEX `fk_Prescription_Doctor1_idx` (`Doctor_doctorssn` ASC) VISIBLE,
CONSTRAINT `fk_Prescription_Drug1`
  FOREIGN KEY (`Drug_drugid`)
    REFERENCES `Project1`.`Drug` (`drugid`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
CONSTRAINT `fk_Prescription_Patient1`
  FOREIGN KEY (`Patient_patientssn`)
    REFERENCES `Project1`.`Patient` (`patientssn`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
CONSTRAINT `fk_Prescription_Doctor1`
  FOREIGN KEY (`Doctor_doctorssn`)
    REFERENCES `Project1`.`Doctor` (`doctorssn`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

-----
-- Table `Project1`.`Supervisor`
-----

```

```

CREATE TABLE IF NOT EXISTS `Project1`.`Supervisor` (
  `supervisorid` INT NOT NULL AUTO_INCREMENT,
  `supervisorfirstname` VARCHAR(45) NOT NULL,
  `supervisorlastname` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`supervisorid`))
ENGINE = InnoDB;

```

```

-----
-- Table `Project1`.`Company Contract`
-----

```

```

CREATE TABLE IF NOT EXISTS `Project1`.`Company Contract` (
  `contractid` INT NOT NULL AUTO_INCREMENT,
  `startdate` DATE NOT NULL,
  `enddate` DATE NOT NULL,
  `contracttext` TEXT(100000) NOT NULL,
  `Pharmacy_pharmacyid` INT NOT NULL,
  `Pharmaceutical Company_companyid` INT NOT NULL,
  `Supervisor_supervisorid` INT NOT NULL,
  PRIMARY KEY (`contractid`),
INDEX `fk_Company Contract_Pharmacy1_idx` (`Pharmacy_pharmacyid` ASC) VISIBLE,
INDEX `fk_Company Contract_Pharmaceutical Company1_idx` (`Pharmaceutical Company_companyid` ASC) VISIBLE,
INDEX `fk_Company Contract_Supervisor1_idx` (`Supervisor_supervisorid` ASC) VISIBLE,
CONSTRAINT `fk_Company Contract_Pharmacy1`
  FOREIGN KEY (`Pharmacy_pharmacyid`)
    REFERENCES `Project1`.`Pharmacy` (`pharmacyid`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
CONSTRAINT `fk_Company Contract_Pharmaceutical Company1`
  FOREIGN KEY (`Pharmaceutical Company_companyid`)
    REFERENCES `Project1`.`Pharmaceutical Company` (`companyid`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
CONSTRAINT `fk_Company Contract_Supervisor1`

```

```

FOREIGN KEY (`Supervisor_supervisorid`)
REFERENCES `Project1`.`Supervisor` (`supervisorid`)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

-----
-- Table `Project1`.`Pharmacy_filled_Prescription`
-----

```

```

CREATE TABLE IF NOT EXISTS `Project1`.`Pharmacy_filled_Prescription` (
  `Pharmacy_pharmacyid` INT NOT NULL,
  `Prescription_prescriptionid` INT NOT NULL,
  `datefilled` DATE NULL,
  PRIMARY KEY (`Pharmacy_pharmacyid`, `Prescription_prescriptionid`),
  INDEX `fk_Pharmacy_has_Prescription_Pharmacy1_idx` (`Pharmacy_pharmacyid` ASC) VISIBLE,
  INDEX `fk_Pharmacy_filled_Prescription_Prescription1_idx` (`Prescription_prescriptionid` ASC) VISIBLE,
  CONSTRAINT `fk_Pharmacy_has_Prescription_Pharmacy1`
    FOREIGN KEY (`Pharmacy_pharmacyid`)
      REFERENCES `Project1`.`Pharmacy` (`pharmacyid`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  CONSTRAINT `fk_Pharmacy_filled_Prescription_Prescription1`
    FOREIGN KEY (`Prescription_prescriptionid`)
      REFERENCES `Project1`.`Prescription` (`prescriptionid`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

-----
-- Table `Project1`.`Pharmacy_has_Drug`
-----

```

```

CREATE TABLE IF NOT EXISTS `Project1`.`Pharmacy_has_Drug` (
  `Pharmacy_pharmacyid` INT NOT NULL,
  `Drug_drugid` INT NOT NULL,
  `price` DECIMAL(6,2) NOT NULL,
  PRIMARY KEY (`Pharmacy_pharmacyid`, `Drug_drugid`),
  INDEX `fk_Pharmacy_has_Drug_Pharmacy1_idx` (`Pharmacy_pharmacyid` ASC) VISIBLE,
  INDEX `fk_Pharmacy_has_Drug_Drug1_idx` (`Drug_drugid` ASC) VISIBLE,
  CONSTRAINT `fk_Pharmacy_has_Drug_Pharmacy1`
    FOREIGN KEY (`Pharmacy_pharmacyid`)
      REFERENCES `Project1`.`Pharmacy` (`pharmacyid`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  CONSTRAINT `fk_Pharmacy_has_Drug_Drug1`
    FOREIGN KEY (`Drug_drugid`)
      REFERENCES `Project1`.`Drug` (`drugid`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

USE `Project1`;

```

```

DELIMITER $$
USE `Project1` $$
$$

```

```

DELIMITER ;

```

```

SET SQL_MODE=@OLD_SQL_MODE;

```



```
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

SQL Queries

When it comes to questions that are of interest to management, we realized that there are a multitude of tables and relations to be seen. For the sake of this report we will limit our focus to these six (6) questions:

1. What is the most prescribed drug?
2. How many contracts between pharmacy and pharmaceutical companies are there?
3. How many patients do each primary care doctor have?
4. Which doctor writes the most prescriptions?
5. What are the number of prescriptions at a pharmacy grouped by drug?
6. How many patients share the same doctor?

Question 1 (one), would give us a better understanding of what prescription drug is in the most demand and which drugs fail to meet criteria or are not working well amongst most patients. A great example of this would be two very similar drugs, but one may cause more side effects than the other causing doctors to be less inclined to prescribe them to patients.

```
select distinct prescription Drug_tradename
from prescription
join (select prescriptionid, sum(quantity) as totalquantity
      from prescription group by prescriptionid) as highdrug on
prescription.prescriptionid=highdrug.prescriptionid
where highdrug.totalquantity= (select distinct max(totalquantity)
                               from (select prescriptionid, sum(quantity) as
totalquantity
                                     from prescription group by prescriptionid) as
highdrug);
```

Question 2 (two), can provide insight to which pharmaceutical companies have the best relationship with fellow pharmacies. One can assume having more contracts would mean better business, better quality medications, and all around better satisfaction not only for management but also for the consumer.

```
select pc.companyname, ph.pharmacyname, count(cc.contractid)
from `Pharmaceutical Company` as pc
join `company contract` as cc on pc.companyid=cc.`Pharmaceutical
Company_companyid`
join pharmacy ph on cc.Pharmacy_pharmacyid=ph.pharmacyid
group by pc.companyid;
```

Questions 3 (three) and 4 (four) are somewhat related. Question 3 allows us to see the number of patients that each primary care doctor has and question 4 allows us to see whether or not they are writing prescriptions and to what ratio they are writing prescriptions to patients. However, the prescriptions they write do not have to be from those who label them as primary care doctors since the doctor could write prescriptions for those that he feels need the prescribed medication.

```
-- Question 3
select doc.firstname, doc.lastname, count(p.Doctor_doctorssn) as
totalpatients
from Doctor as doc
join Patient as p on doc.doctorssn=p.Doctor_doctorssn
group by doc.doctorssn
order by totalpatients;

-- Question 4
select doc.firstname, doc.lastname, count(pres.Doctor_doctorssn) as
totalprescriptions
from Doctor as doc
join Prescription as pres on doc.doctorssn=pres.Doctor_doctorssn
group by doc.doctorssn;
```

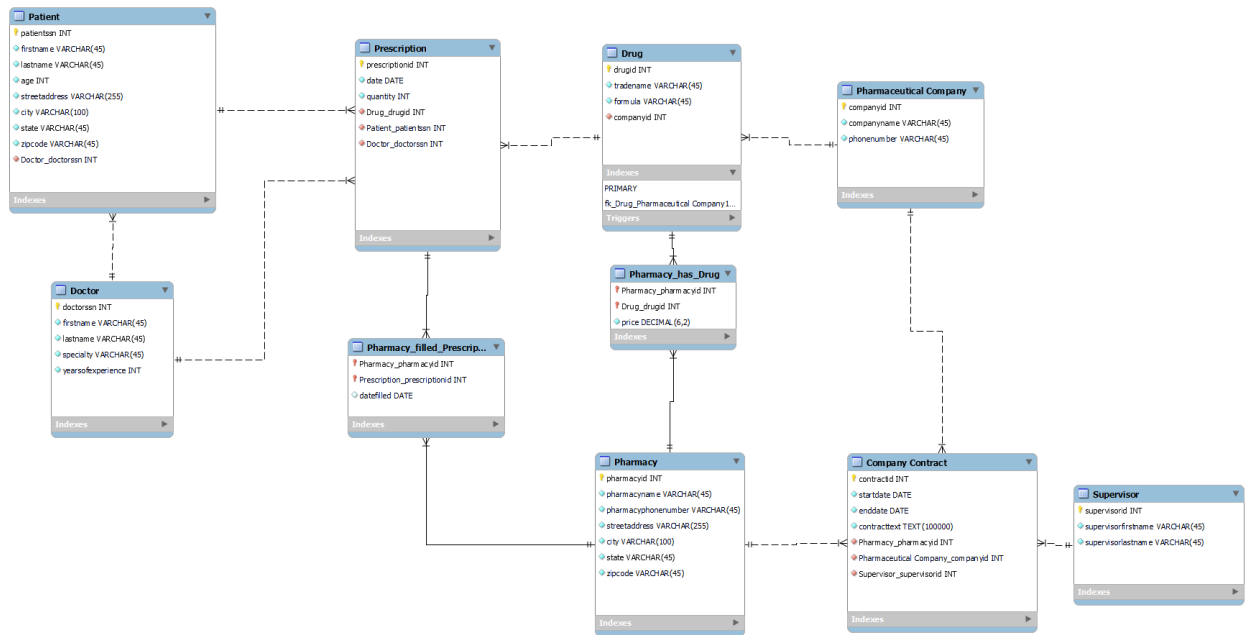
Question 5 (five), allows us to better understand which pharmacies are filling out the most prescriptions and which are not quite meeting the quota. Along with that, we will also better understand which drugs are bringing in the most revenue when each prescription is filled.

```
select pharm.pharmacyname, drug.tradename, count(pres.prescriptionid)
as totalprescriptions
from pharmacy as pharm
join pharmacy_filled_prescription as filled on
pharm.pharmacyid=filled.Pharmacy_pharmacyid
join prescription as pres on
filled.Prescription_prescriptionid=pres.prescriptionid
join drug on pres.Drug_tradename=drug.tradename
group by pharm.pharmacyname, drug.tradename
order by pharm.pharmacyname;
```

Lastly, question 6 (six) understanding how many patients share the same doctor could lead us to find trends in regards to prescription habits and generally what is being prescribed the most as seen by tendencies. Patients sharing the same doctor could end up being prescribed more of the same drug which would cause us to refill our inventory more on a regular basis.

```
select doc.firstname, doc.lastname, count(pat.Doctor_doctorssn) as
ShareDoctor
from Doctor as doc
join Patient as pat on doc.doctorssn=pat.Doctor_doctorssn
where pat.Doctor_doctorssn=doc.doctorssn
group by doc.doctorssn;
```

ER Model



The above ER diagram contains multiple relationships about a growing drug store chain.

The first is a non-identifying one to many relationship between a doctor and one or more of its patients. A patient contains a primary key of the patient's SSN along with its first name, last name, age, street address, city, state and zipcode. In addition the patient has its own doctor ssn to which links the patient to its primary physician through its foreign key. The doctor has its primary key being its doctor's SSN along with its first name, last name, specialty and years of experience. A patient has a one to many non-identifying relationship with obtaining a prescription. The prescription's primary key includes the prescription id along with its date and quantity. Also the prescription table has the foreign keys of drug id, patient ssn and doctor ssn. A doctor can prescribe multiple prescriptions for any given patient. A drug has the primary key of the drug id along with its trade name formula, in addition has the foreign key company id. The Drug table has a 1 to many non-identifying relationship with the prescription table. In addition a drug has a many to 1 non-identifying relationship with a pharmaceutical company to which it is

administered. The pharmaceutical company to which the drug is produced has the primary key of its company id along with its company name and phone number. To fill a prescription it goes through a many to many identifying relationship with a pharmacy. A pharmacy's primary key is the pharmacy id along with its pharmacy name, pharmacy phone number, street address, city, state and zip code. The prescription and pharmacy tables are linked through the pharmacy id and the prescription id along with the date filled. The pharmacies have a many to many identifying relationship with drugs. The many to many relationship between the drug and pharmacy tables determine whether or not the desired pharmacy contains a drug is by the pharmacies id, drugs id and the price of the drug. In order for a pharmacy to get distribution for a particular drug, they need a company contract with a pharmaceutical company. Between the pharmacy and company contract tables, there is a non-identifying one to many relationship. In addition between the pharmaceutical company and the company contract table, there is also a non-identifying one to many relationship. The contract between the pharmacy and the pharmaceutical company needs a supervisor as well to conduct the business end of the relationship. A supervisor's primary key is its supervisor id along with its supervisor first name and supervisor last name. The supervisor and the company contract tables have a non-identifying one to many relationship. The company contract tables have the primary key of contract id along with a start date, end date and contract text. Also the company contract contains the pharmacy id, company id and supervisor id as foreign keys.

Part One Conclusion

In summary, Grid has detailed out the requirements as Grid understands them, created an initial ER diagram that conforms to and addresses the requirements, produced relational schema in alignment with the ER diagram, and provided a list of suggested queries to be used as

potential reports. During the course of compiling this information, Grid has obtained a more thorough understanding of the database design process and gained experience in utilizing tools to create database diagrams and schema. Moving into the next phase of the project, Grid has additional questions related to the types of reports DSC would like to include in its application as well as questions regarding the layouts for the different types of user interfaces the application should have.

Part 2

In part two of the project we develop the appropriate java program necessary to interact with the web browser and MySQL database using Spring Boot and JDBC. Through the use of mapping annotations the spring server will call the method associated with the URL.

When the initial Drug System web page is displayed the user is presented with the 4 choices. After selecting a choice the user is brought to the appropriate web page and given the opportunity to supply the input necessary to generate a report.

When a Doctor selects the first link they will be presented with a form to fill out the prescription for the patient. The form contains entries for the Doctor SSN, Doctor Name, Patient SSN, Patient Name, Drug Name, and Quantity. When the Doctor is finished they can click the “Create Prescription” button. In the event the Doctor enters the wrong information for example an incorrect SSN, the form will display a message alerting them of the mistake. After the doctor has successfully created the prescription the request is sent to the Spring Boot Server for processing. The `@GetMapping` will map the url to the form and the `@PostMapping` will begin the process of validating the entries. At anypoint an error is encountered for example, a quantity of zero for the drug, the error message will display alerting the Doctor of the mistake. If the information is valid, the MySQL statement is created using the `PreparedStatement` object. Also used is the `ResultSet` object to execute a query with the user provided information. Upon successful database queries, the insertion of the prescription is made into the database and the user is brought to a new screen showing the prescription entered.

The following screenshots show the initial screen, an error message for unknown drug name, and the screenshot of a correctly filled form, and a successfully created prescription:

Drug System	New Prescription Form
<p>Click on a choice below.</p> <p>Write a new prescription (for Doctors only)</p> <p>Request a prescription be filled (for Patients only)</p> <p>Report on drug usage (Pharmacists only)</p> <p>FDA reporting (FDA only)</p>	<p>Drug Does Not Exist</p> <p>Doctor SSN: <input type="text" value="111111111"/></p> <p>Doctor Name: <input type="text" value="John"/></p> <p>Patient SSN: <input type="text" value="444444444"/></p> <p>Patient Name: <input type="text" value="Jane"/></p> <p>Drug Name: <input type="text" value="Candy"/></p> <p>Quantity: <input type="text" value="7"/></p> <p><input type="button" value="Create Prescription"/></p>

New Prescription Form	
<p>Doctor SSN: <input type="text" value="111111111"/></p> <p>Doctor Name: <input type="text" value="John"/></p> <p>Patient SSN: <input type="text" value="444444444"/></p> <p>Patient Name: <input type="text" value="Jane"/></p> <p>Drug Name: <input type="text" value="Motrin"/></p> <p>Quantity: <input type="text" value="7"/></p> <p><input type="button" value="Create Prescription"/></p>	<p>Rx: 15</p> <p>Doctor: 111111111</p> <p>Name: John</p> <p>Patient: 444444444</p> <p>Name: Jane</p> <p>Drug: Motrin</p> <p>Quantity: 7</p> <p>Pharmacy:</p> <p>Name:</p> <p>Address:</p> <p>Phone:</p> <p>Date Filled:</p> <p>Cost: \$ 0.0</p>

Prescription Filled

- A patient requests that a pharmacy fill a prescription. The patient enters a prescription number and the name and address of a pharmacy. The prescription is updated with the pharmacy data, cost and current date, and redisplayed to the user.

<p>Request Prescription be filled.</p> <p>Enter pharmacy name and address and prescription Rx number.</p> <p>Rx: <input type="text" value="5"/></p> <p>Patient Name: <input type="text" value="Joey Dirt"/></p> <p>Pharmacy Name: <input type="text" value="Walgreens"/></p> <p>Pharmacy Address: <input type="text" value="111 Pharm Dr"/></p> <p><input type="button" value="Request Fill for Prescription"/></p>	<p>Rx: 5</p> <p>Doctor: 111111111</p> <p>Name: John Doc</p> <p>Patient: 777777777</p> <p>Name: Joey Dirt</p> <p>Drug: Tylenol</p> <p>Quantity: 40</p> <p>Pharmacy: 1</p> <p>Name: Walgreens</p> <p>Address: 111 Pharm Dr</p> <p>Phone: (666)666-6666</p> <p>Date Filled: 2021-05-25</p> <p>Cost: \$ 79.6</p>
--	--

Here is a photo from the results of the `@PostMapping("/prescription/fill")`. The patient Joey Dirt enters a prescription number and the name and address of a pharmacy. The prescription

updates with the updated cost of the total quantity of drugs dispensed along with the current date filled. The results are then displayed to the user.

<h3>Request Prescription be filled.</h3> <p>Enter pharmacy name and address and prescription Rx number.</p> <p>Rxid already filled</p> <p>Rx: <input type="text" value="1"/></p> <p>Patient Name: <input type="text" value="John Doe"/></p> <p>Pharmacy Name: <input type="text" value="Walgreens"/></p> <p>Pharmacy Address: <input type="text" value="111 Pharm Dr"/></p> <p><input type="button" value="Request Fill for Prescription"/></p>	<h3>Request Prescription be filled.</h3> <p>Enter pharmacy name and address and prescription Rx number.</p> <p>Invalid Rxid</p> <p>Rx: <input type="text" value="6"/></p> <p>Patient Name: <input type="text" value="Michael Smith"/></p> <p>Pharmacy Name: <input type="text" value="CVS"/></p> <p>Pharmacy Address: <input type="text" value="333 Pharm Dr"/></p> <p><input type="button" value="Request Fill for Prescription"/></p>
---	---

For the case where a prescription is already filled, we add an attribute message “Rxid already filled” to the prescription/fill page to inform the user. In addition, we handled the case of user input being a wrong Rx. We added an attribute message to indicate to the user that they typed in wrong information with the error message “Invalid Rxid”.



Pharmacy Drug Report

The Pharmacy Drug Report allows any of DSG's pharmacies to obtain details about the quantity of drugs they have dispensed while filling prescriptions. The report is intended to provide the details about which drugs were filled and the total quantity of each drug filled over a period of time by a specific pharmacy.

Report users will provide the ID of the pharmacy they are interested in and then provide the period of time they would like to report over. When the report user leaves the drug field blank, the report will provide details for every drug filled.

<div style="border: 1px solid #ccc; border-radius: 10px; padding: 10px; margin-bottom: 10px;"> <h3 style="text-align: center; margin: 0;">Pharmacy Drug Report</h3> <p style="margin: 5px 0;">Enter a drug name or partial name and date range below.</p> <p style="margin: 5px 0;">Pharmacy ID: <input style="width: 150px;" type="text" value="1"/></p> <p style="margin: 5px 0;">Drug: <input style="width: 150px;" type="text"/></p> <p style="margin: 5px 0;">Start date: <input style="width: 100px;" type="text" value="12/27/2020"/> </p> <p style="margin: 5px 0;">End date: <input style="width: 100px;" type="text" value="06/30/2021"/> </p> <p style="text-align: center; margin-top: 10px;"><input type="button" value="Search"/></p> </div>	<h3 style="margin: 0;">Pharmacy usage of drugs by drugname.</h3> <p style="margin: 5px 0;">Pharmacy: 1</p> <p style="margin: 5px 0;">Start Date: 2020-12-27</p> <p style="margin: 5px 0;">End Date: 2021-06-30</p> <table style="margin-top: 10px; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 2px 10px;">Drug</th> <th style="text-align: left; padding: 2px 10px;">Quantity Used</th> </tr> </thead> <tbody> <tr> <td style="padding: 2px 10px;">Motrin</td> <td style="padding: 2px 10px;">5</td> </tr> <tr> <td style="padding: 2px 10px;">Tylenol</td> <td style="padding: 2px 10px;">37</td> </tr> <tr> <td style="padding: 2px 10px;">Ibuprofen</td> <td style="padding: 2px 10px;">27</td> </tr> </tbody> </table>	Drug	Quantity Used	Motrin	5	Tylenol	37	Ibuprofen	27
Drug	Quantity Used								
Motrin	5								
Tylenol	37								
Ibuprofen	27								

If the report user would like to limit the results to a specific drug they can provide the name of the drug they are interested in. They may choose to only input the first few letters of a drug name and the report will provide results for any drugs that match the partial name. In addition, if the user does not provide a start date, the report will default the start date to '01/01/1900'. Likewise, if the user does not provide an end date, the report will default the end date to the current date.

<h3>Pharmacy Drug Report</h3> <p>Enter a drug name or partial name and date range below.</p> <p>Pharmacy ID: <input type="text" value="1"/></p> <p>Drug: <input type="text" value="tyl"/></p> <p>Start date: <input type="text" value="mm/dd/yyyy"/> </p> <p>End date: <input type="text" value="mm/dd/yyyy"/> </p> <p><input type="button" value="Search"/></p>	<h3>Pharmacy usage of drugs by drugname.</h3> <p>Pharmacy: 1</p> <p>Start Date: 1900-1-1</p> <p>End Date: 2021-05-25</p> <table border="1"> <thead> <tr> <th>Drug</th> <th>Quantity Used</th> </tr> </thead> <tbody> <tr> <td>Tylenol</td> <td>37</td> </tr> </tbody> </table>	Drug	Quantity Used	Tylenol	37
Drug	Quantity Used				
Tylenol	37				

If the report user enters a Pharmacy ID that does not exist, the results will inform the user that the pharmacy does not exist. The report will also provide the user notice if an invalid date range has been provided, such as in instances where the start date comes after the end date.

<h3>Pharmacy Drug Report</h3> <p>Enter a drug name or partial name and date range below.</p> <p>Pharmacy ID: <input type="text" value="15"/></p> <p>Drug: <input type="text" value="tyl"/></p> <p>Start date: <input type="text" value="05/25/2021"/> </p> <p>End date: <input type="text" value="05/03/2021"/> </p> <p><input type="button" value="Search"/></p>	<h3>Pharmacy usage of drugs by drugname.</h3> <p>Pharmacy: 15 - Pharmacy not found</p> <p>Start Date: 2021-05-25 - Start Date must be <= End Date</p> <p>End Date: 2021-05-03</p> <table border="1"> <thead> <tr> <th>Drug</th> <th>Quantity Used</th> </tr> </thead> <tbody> </tbody> </table>	Drug	Quantity Used
Drug	Quantity Used		

FDA REPORT



For FDA reports, we have made the process simpler and easy to use to assist in getting the correct information quickly and reliably. Below, you will find examples of how the form works and also some of the checks we have implemented in the course of designing the application. Here, you will see screenshots of a successful input and output coming from the user.

<div style="border: 1px solid #ccc; border-radius: 10px; padding: 10px; margin-bottom: 10px;"> <h3 style="text-align: center; margin: 0;">FDA Drug Report</h3> <p style="margin: 5px 0;">Enter a drug name or partial name and date range below.</p> <div style="margin-bottom: 10px;"> Drug: <input style="width: 100%;" type="text" value="ibu"/> </div> <div style="margin-bottom: 10px;"> Start date: <input style="width: 100%;" type="text" value="02/01/2020"/> <div style="float: right; font-size: 0.8em;">📅</div> </div> <div style="margin-bottom: 10px;"> End date: <input style="width: 100%;" type="text" value="03/31/2021"/> <div style="float: right; font-size: 0.8em;">📅</div> </div> <div style="text-align: center; margin-top: 10px;"> <input style="border: 1px solid #ccc; padding: 2px 10px;" type="button" value="Search"/> </div> </div>	<h3 style="margin: 0;">FDA report drug usage by doctor.</h3> <p style="margin: 5px 0;">Start Date: 2020-02-01</p> <p style="margin: 5px 0;">End Date: 2021-03-31</p> <p style="margin: 5px 0;">Drug: Ibuprofen</p> <p style="margin: 5px 0;">Doctor Quantity Prescribed</p> <p style="margin: 5px 0;">Jack 27</p>
---	---

- Successful Input/Output



Notice in this example, how the user has input a partially complete drug name. We have designed our application to find the full drug name and output in our FDA report for easy understanding and easy access. We also include the Doctor's first name and also the quantity of drug he has prescribed within the specified date range.

What about the cases when we have incorrect output such as a drug not being found within our system or improper dates being inserted (i.e. start date that comes after the end date or no date at all)? We did not forget to account for these errors also and within the next several screenshots.

<h3>FDA Drug Report</h3> <p>Enter a drug name or partial name and date range below.</p> <p>Drug: <input type="text" value="ben"/></p> <p>Start date <input type="text" value="02/01/2020"/> </p> <p>End date <input type="text" value="03/31/2021"/> </p> <p><input type="button" value="Search"/></p>	<h3>FDA report drug usage by doctor.</h3> <p>Start Date: 2020-02-01</p> <p>End Date: 2021-03-31</p> <p>Drug: DRUG NOT FOUND</p> <p>Doctor Quantity Prescribed</p>
--	--



- Drug Not Found in Database

In the case where a drug is not found within our database, notice how we handle the situation. We still show the report to the user, with the dates that they have input, however, we display a message in the “drug” attribute to show the drug does not exist or the drug was not found in the system.

<h3>FDA Drug Report</h3> <p>Enter a drug name or partial name and date range below.</p> <p>Drug: <input type="text" value="mot"/></p> <p>Start date <input type="text" value="05/25/2021"/> </p> <p>End date <input type="text" value="05/24/2021"/> </p> <p><input type="button" value="Search"/></p>	<h3>FDA report drug usage by doctor.</h3> <p>Start Date: 2021-05-25: START DATE SHOULD COME PRIOR TO END DATE</p> <p>End Date: 2021-05-24</p> <p>Drug: mot</p> <p>Doctor Quantity Prescribed</p>
--	---

- Start Date Comes After End Date

Here, we can see that the user has input a start date that comes after our end date. As done with our failed drug input, we still display the report with an improper date. However, we concatenate a message to the improper date, notifying the user that the start date should come before the end date. Which is why we do not show anymore information.

<h3 style="text-align: center;">FDA Drug Report</h3> <p>Enter a drug name or partial name and date range below.</p> <p>Drug: <input type="text" value="tyl"/></p> <p>Start date <input type="text" value="mm/dd/yyyy"/> </p> <p>End date <input type="text" value="mm/dd/yyyy"/> </p> <p style="text-align: center;"><input type="button" value="Search"/></p>	<h3 style="text-align: center;">FDA report drug usage by doctor.</h3> <p>Start Date: 1900-01-01</p> <p>End Date: 2021-05-25</p> <p>Drug: Tylenol</p> <p>Doctor Quantity Prescribed</p> <p>Jack 7</p> <p>John 70</p>
--	---

- No Date Input

Lastly, there is the case of no start date and no end date entered into the form. To handle this, we as a team decided to fill in those dates in the case of no input. As our default start date, we have set it to be “1900-01-01” and for our end date, we made our default to be that of the current date. One thing to note is that we still get proper output for both “Doctor” and “Quantity Prescribed” columns. This is because the drug is still a proper entry and we have prescriptions that fall within our default date range.