Ultra Value Pharmacy 4U™ Relational Database Design

Prepared by
Charles Zieres & Brandon Pimentel

Project Overview

Ultra Value Pharmacy 4U(™) (UVP) was founded in 1972 with the mission of providing prescription medication to the fiscally challenged. Ultra Value Pharmacy 4U(™) continues with this mission today by partnering with Dollar General and 99c Stores around the country to share floorspace and provide a convenient physical location for its customers to pick up their prescriptions.
The last update to Ultra Value Pharmacy 4U's prescription management system was in 1978 when a universal fax number and machine was introduced. Ultra Value Pharmacy 4U has decided to update their internal systems to a more modern platform and has engaged Pocket Sized Solutions to manage the database schema requirements.

Currently Ultra Value Pharmacy 4U manages its data at its central headquarters using folders and paper forms with phone calls to and from the stores as the primary way to communicate updates. Doctors provide prescriptions via the fax line and staff at headquarters are responsible for updating the files and papers and calling in orders to the individual stores. Reports are generated by Doug, the office administrator who has been with the company since its creation in 1972. Doug is familiar with the order and filing of all folders and papers and is currently overseeing the expansion of the paper platform to the 3rd floor of newly installed filing cabinets.

When a new contract is signed with a pharmaceutical company, Doug puts a copy of the contract in the appropriate file folder and calls the affected stores (and the contract supervisor) and updates them with the new pharmaceutical details.

Doctors submit subscriptions to Ultra Value Pharmacy 4U via the fax number. Doug receives the faxes at headquarters, calls the appropriate store and gives them the information on the fax. Doug then files the fax in the appropriate file folder. If there is a question about a specific prescription, the store calls Doug who pulls the information from the filing cabinets.

Pocket Sized Solutions goal is to create a web based, multi user, modern platform to manage Ultra Value Pharmacy 4U's data on customers, prescriptions, doctors, pharmacies, drugs, and drug providers before the current system is retired. The new system upgrade will not only speed up delivery of prescription fulfillment while costing less than the current system, it will also allow management to make data driven decisions in a timely manner.

Business Requirements:

BR1: The system must be able to track the name, age, address, and SSN of UVP's customers/patients.

BR2: The system must be able to track the name, specialty, years of experience and SSN of the doctors who write prescriptions

BR3: The system must be able to track the name and phone number of each pharmaceutical company.

BR4: The system must be able to track drugs by both their "generic" name and trade name.

BR5: The system must be able to track the name, address, and phone number of every pharmacy/store.

BR6: The system must be able to track the primary physician of every customer.

BR7: The system must be able to record any prescription for any customer written by any doctor. In other words, there must be no constraint on what doctor can write what prescription for which customer.

BR8: The system must be able to track the different prices that drugs are sold at different stores.

BR9: The system must be able to track the RX (prescription) number for each prescription. This number is unique per drug prescribed to one person per doctor.

BR10: The system must be able to track multiple RX per customer

BR11: The system must be able to track the difference between trade name drugs and generic drugs. If the prescription is for a trade name, then the system must track the specific pharmaceutical company that provided the drug, otherwise the system must track which pharmaceutical company ultimately provided the generic drug.

BR12: The system must track the date a prescription was filled, what store it was filled at, whether a trade name or generic drug was provided, and the pharmaceutical company associated with the drug

BR13: The system must track the various contracts between pharmaceutical companies and stores including the supervisor, start date, end date, and full text of the contract.

Technical Approach

Assumptions and Constraints

All people, customers, stores, pharmacies, and other entities are in the United States.
All supervisors of contracts are people with unique social security numbers.
Security will be handled elsewhere.
Doug will be readily available to answer questions or provide insight until the project is complete.
All currencies are in USD.

Implementation Notes

Stored Procedures will be provided to developers for all entity creations (e.g. addDoctor, addPrescription)

Views will be provided for developers to speed up development (e.g. patientPrescriptions, supervisorContracts)

Whether or not a prescription has been filled will be tracked by the prescriptions.filledOn date. If there is a value, then the prescription was filled on that date. If there is no value or a Null value, the prescription has not yet been filled.

Whether or not to treat a drug as “generic” or not is handled with the prescriptions.generic field. A 1 (or TRUE) value indicates that the generic version will be used. A 0 or NULL (FALSE) value indicates that the trade name version must be used.

Management Insights

At the conclusion of the project, management will be able to answer the following questions without waiting for Doug to provide the reports (a process that currently takes weeks, or even months). See Appendix C for SQL.

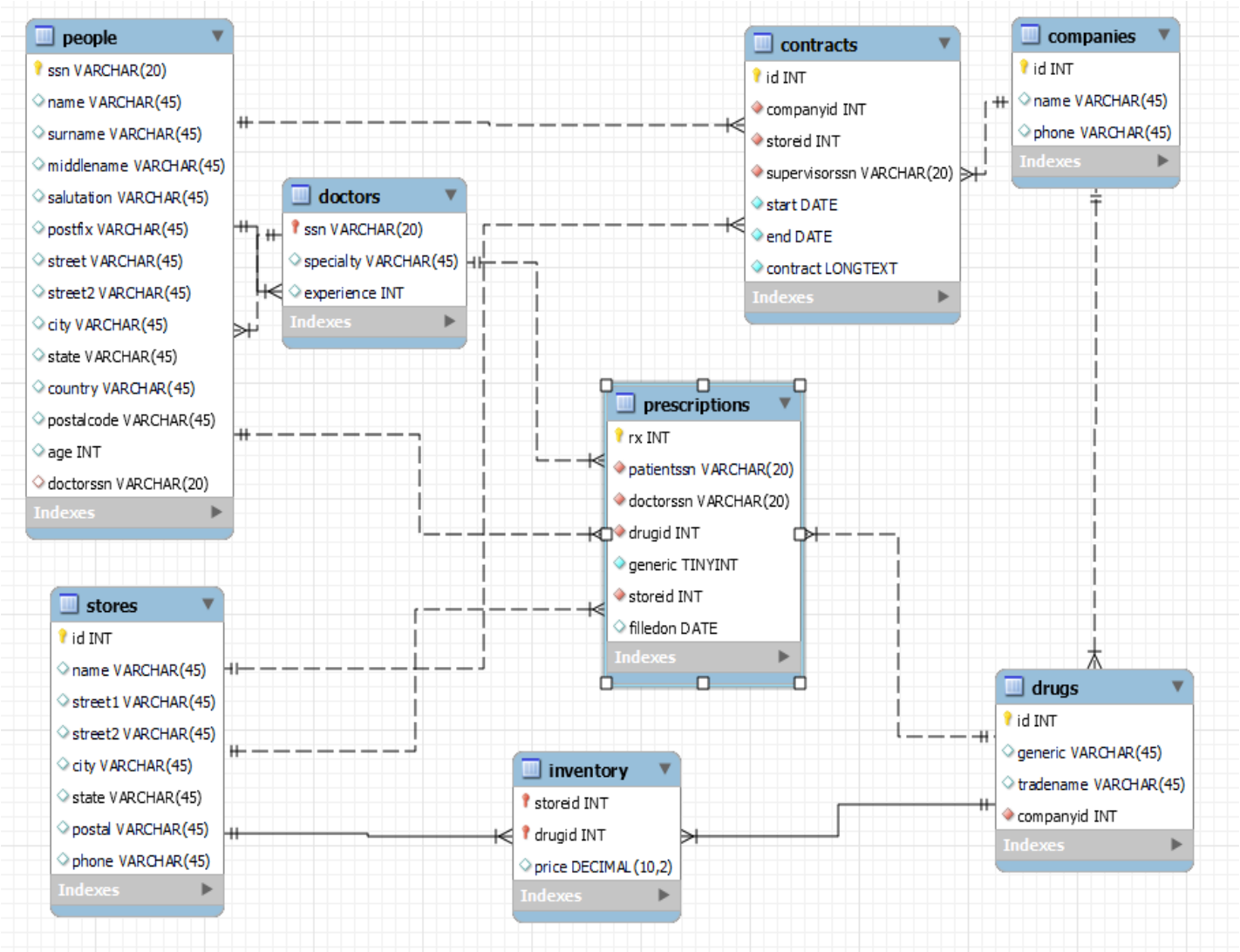
1. What store is the top producer?
2. What company provides the most drugs overall?
3. What is the average number of contracts per supervisor?
4. Do most customers travel out of state to fill a prescription?
5. What is the busiest day of the year?

Conclusion

The key to any successful IT Project is fully understanding the needs of the client and then translating them into technical requirements. As an example, in this project, multiple points of data are often repeated (addresses, names, etc). This does not mean that every table should have copies of the same data fields, but that normalization must be put in place. A pertinent example of this would be the requirements around prescriptions, drugs, and pharmaceuticals. The requirements state that the customer, doctor, company providing the drug, and so on must be provided. Rather than creating one large table that replicates the fields, we created separate tables and will provide a view to the developer to use to display prescriptions. .

Appendix A

ER Diagram



Appendix B

Create Table Statements

```
-- MySQL Script generated by MySQL Workbench
-- Tue Jan 24 19:45:57 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_
ENGINE_SUBSTITUTION';
-- -----------------------------------------------------
-- Schema uvp_db
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `uvp_db` DEFAULT CHARACTER SET utf8 ;
USE `uvp_db` ;
-- -----------------------------------------------------
-- Table `uvp_db`.`doctor`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `uvp_db`.`doctor` (
`doctorId` INT NOT NULL AUTO_INCREMENT,
`last_name` VARCHAR(45) NULL,
`first_name` VARCHAR(45) NULL,
`specialty` VARCHAR(45) NULL,
`experience` INT NULL,
`ssn` VARCHAR(20) UNIQUE NOT NULL,
`patientId` INT NULL,
PRIMARY KEY (`doctorId`),
UNIQUE INDEX `doctorId_UNIQUE` (`doctorId` ASC) VISIBLE,
UNIQUE INDEX `doctorSSN_UNIQUE` (`ssn` ASC) VISIBLE,
CONSTRAINT `patientId`
FOREIGN KEY (`patientId`)
REFERENCES `uvp_db`.`patient` (`patientId`)
ON DELETE CASCADE
ON UPDATE CASCADE)
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `uvp_db`.`patient`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `uvp_db`.`patient` (
`patientId` INT NOT NULL AUTO_INCREMENT,
`last_name` VARCHAR(45) NULL,
`first_name` VARCHAR(45) NULL,
`birthdate` VARCHAR(10) NULL,
`ssn` VARCHAR(20) UNIQUE NOT NULL,
```

```
`street` VARCHAR(45) NULL,
`city` VARCHAR(45) NULL,
`state` VARCHAR(45) NULL,
`zipcode` VARCHAR(45) NULL,
`doctorId` INT NULL,
PRIMARY KEY (`patientId`),
UNIQUE INDEX `patientId_UNIQUE` (`patientId` ASC) VISIBLE,
UNIQUE INDEX `patientSSN_UNIQUE` (`ssn` ASC) VISIBLE,
INDEX `doctorId_idx` (`doctorId` ASC) VISIBLE,
CONSTRAINT `doctorId`
FOREIGN KEY (`doctorId`)
REFERENCES `uvp_db`.`doctor` (`doctorId`)
ON DELETE CASCADE
ON UPDATE CASCADE)
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `uvp_db`.`company`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `uvp_db`.`company` (
`id` INT NOT NULL AUTO_INCREMENT,
`name` VARCHAR(45) NULL,
`phone` VARCHAR(45) NULL,
PRIMARY KEY (`id`),
UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE)
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `uvp_db`.`drug`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `uvp_db`.`drug` (
  `drug_id` int(11) NOT NULL,
  `trade_name` varchar(100) DEFAULT NULL,
  `formula` varchar(200) DEFAULT NULL,
  PRIMARY KEY (`drug_id`),
  UNIQUE INDEX `id_UNIQUE` (`drug_id` ASC) VISIBLE,
  UNIQUE INDEX `tradename_UNIQUE` (`trade_name` ASC) VISIBLE)
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `uvp_db`.`store`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `uvp_db`.`store` (
`id` INT NOT NULL AUTO_INCREMENT,
`name` VARCHAR(45) NULL,
`street1` VARCHAR(45) NULL,
`street2` VARCHAR(45) NULL,
`city` VARCHAR(45) NULL,
`state` VARCHAR(45) NULL,
`postal` VARCHAR(45) NULL,
```

```
`phone` VARCHAR(45) NULL,
PRIMARY KEY (`id`),
UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE)
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `uvp_db`.`prescription`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `uvp_db`.`prescription` (
`rx` INT NOT NULL AUTO_INCREMENT,
`quantity` INT NOT NULL,
`drug_name` varchar(100) DEFAULT NULL,
`patientSSN` VARCHAR(20) NOT NULL,
`patient_first_name` VARCHAR(45) NULL,
`patient_last_name` VARCHAR(45) NULL,
`doctorSSN` VARCHAR(20) NOT NULL,
`doctor_first_name` VARCHAR(45) NULL,
`doctor_last_name` VARCHAR(45) NULL,
`pharmacyId` INT NULL,
`pharmacyAddress` VARCHAR(45) NULL,
`pharmacyPhone` VARCHAR(45) NULL,
`pharmacyName` VARCHAR(45) NULL,
`dateFilled` DATE NULL,
`cost` INT NULL,
UNIQUE INDEX `rx_UNIQUE` (`rx` ASC) VISIBLE,
PRIMARY KEY (`rx`),
INDEX `patientSSN_idx` (`patientSSN` ASC) VISIBLE,
INDEX `doctorSSN_idx` (`doctorSSN` ASC) VISIBLE,
INDEX `pharmacyId_idx` (`pharmacyId` ASC) VISIBLE,
CONSTRAINT `pre_patientSSN`
FOREIGN KEY (`patientSSN`)
REFERENCES `uvp_db`.`patient` (`ssn`)
ON DELETE CASCADE
ON UPDATE CASCADE,
CONSTRAINT `pre_doctorSSN`
FOREIGN KEY (`doctorSSN`)
REFERENCES `uvp_db`.`doctor` (`ssn`)
ON DELETE CASCADE
ON UPDATE CASCADE,
CONSTRAINT `drug_name`
FOREIGN KEY (`drug_name`)
REFERENCES `uvp_db`.`drug` (`trade_name`)
ON DELETE CASCADE
ON UPDATE CASCADE,
CONSTRAINT `pharmacyId`
FOREIGN KEY (`pharmacyId`)
REFERENCES `uvp_db`.`store` (`id`)
ON DELETE CASCADE
ON UPDATE CASCADE)
```

```
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `uvp_db`.`contract`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `uvp_db`.`contract` (
`id` INT NOT NULL AUTO_INCREMENT,
`companyid` INT NOT NULL,
`storeid` INT NOT NULL,
`start` DATE NOT NULL,
`end` DATE NOT NULL,
`contract` LONGTEXT NOT NULL,
PRIMARY KEY (`id`),
UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE,
INDEX `companyid_idx` (`companyid` ASC) VISIBLE,
INDEX `storeid_idx` (`storeid` ASC) VISIBLE,
CONSTRAINT `con_companyid`
FOREIGN KEY (`companyid`)
REFERENCES `uvp_db`.`company` (`id`)
ON DELETE CASCADE
ON UPDATE CASCADE,
CONSTRAINT `con_storeid`
FOREIGN KEY (`storeid`)
REFERENCES `uvp_db`.`store` (`id`)
ON DELETE CASCADE
ON UPDATE CASCADE)
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `uvp_db`.`inventory`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `uvp_db`.`inventory` (
`storeid` INT NOT NULL,
`drugid` INT NOT NULL,
`price` DECIMAL(10,2) NULL,
PRIMARY KEY (`storeid`, `drugid`),
INDEX `drugid_idx` (`drugid` ASC) VISIBLE,
CONSTRAINT `inv_storeid`
FOREIGN KEY (`storeid`)
REFERENCES `uvp_db`.`store` (`id`)
ON DELETE NO ACTION
ON UPDATE NO ACTION,
CONSTRAINT `inv_drugid`
FOREIGN KEY (`drugid`)
REFERENCES `uvp_db`.`drug` (`drug_id`)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Appendix C

Top 5 Question Queries

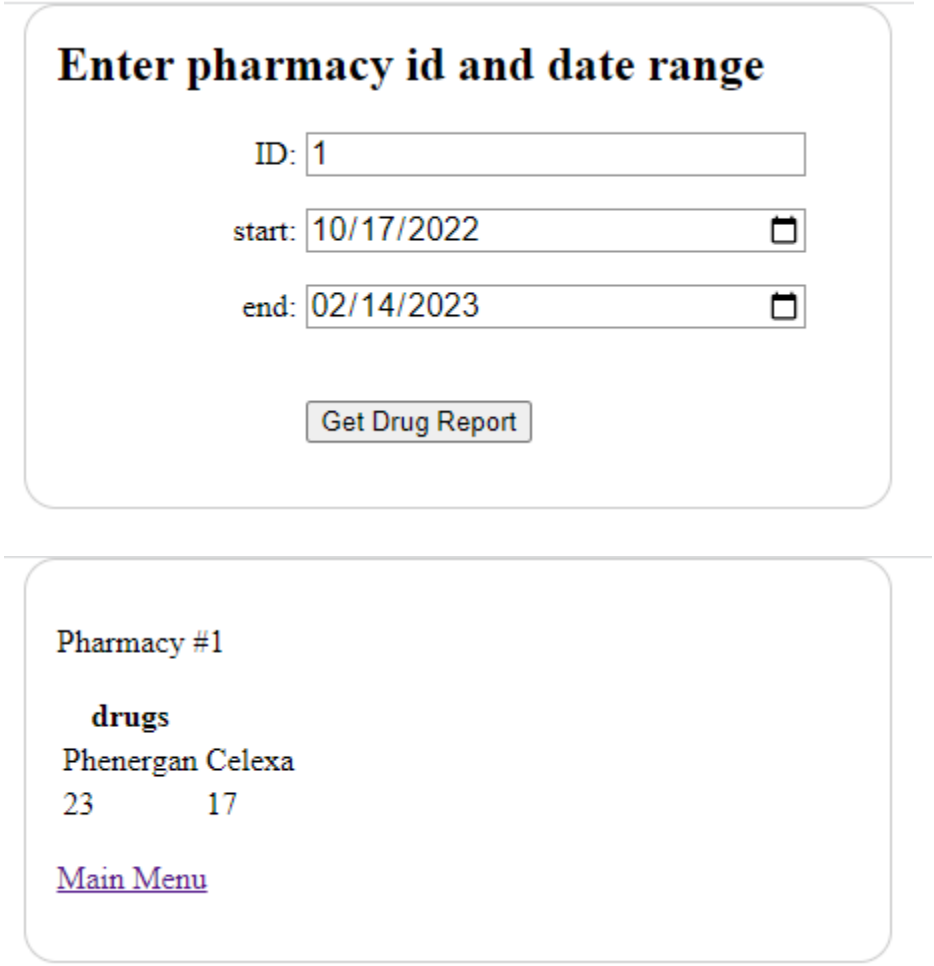
```
-- 5 QUESTION QUERIES

--TOP PRODUCING STORE
SELECT
	stores.name
FROM
	stores
JOIN
	prescriptions ON stores.id = prescriptions.storeid
GROUP BY
	storeid
ORDER BY
	count(*) DESC
LIMIT 1;


-- TOP PROVIDING DRUG COMPANY
select
	companies.name
FROM
	companies
JOIN
	drugs ON companies.id = drugs.companyid
JOIN
	prescriptions ON drugs.id = prescriptions.drugid
GROUP BY
	companyid
ORDER BY
	count(*) DESC
LIMIT 1;

--Average number of contracts per supervisor
SELECT AVG(sup.cont) FROM (SELECT count(*) as cont FROM contracts GROUP BY
supervisorssn) as sup;

-- in state customers
SELECT
	COUNT(*) as inState
FROM
	people
JOIN
	prescriptions ON prescriptions.patientssn = people.ssn
JOIN
```

```
        stores ON prescriptions.storeid = stores.id AND stores.state = people.state
GROUP BY
        people.state;

-- out of state customers
SELECT
        COUNT(*) as outOfState
FROM
        people
JOIN
        prescriptions ON prescriptions.patientssn = people.ssn
JOIN
        stores ON prescriptions.storeid = stores.id AND stores.state <> people.state
GROUP BY
        people.state;


-- busiest date
SELECT
        filledOn
FROM
        prescriptions
WHERE
        fileldOn IS NOT NULL
GROUP BY
        filledOn
ORDER BY
        COUNT(*) DESC
LIMIT 1;
```

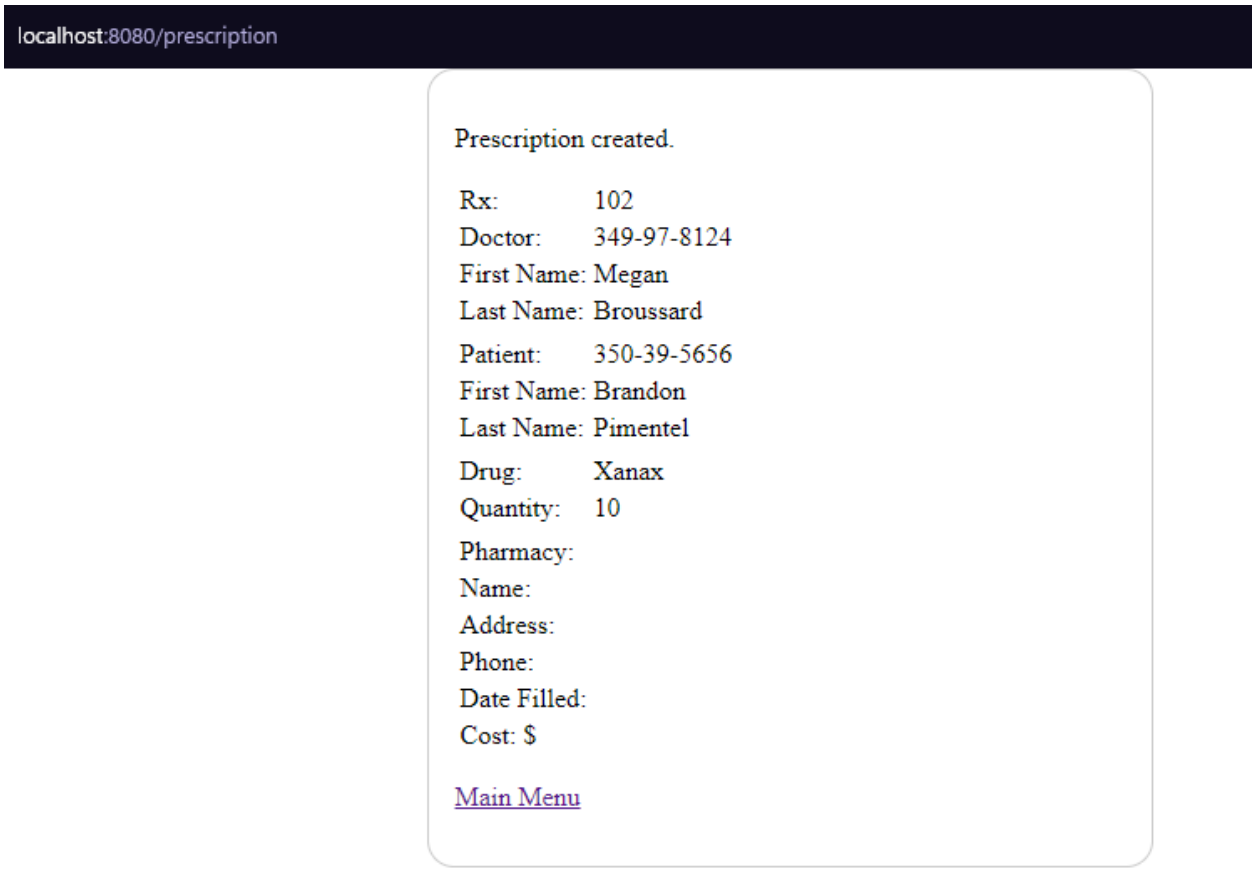
Appendix D

Screenshots

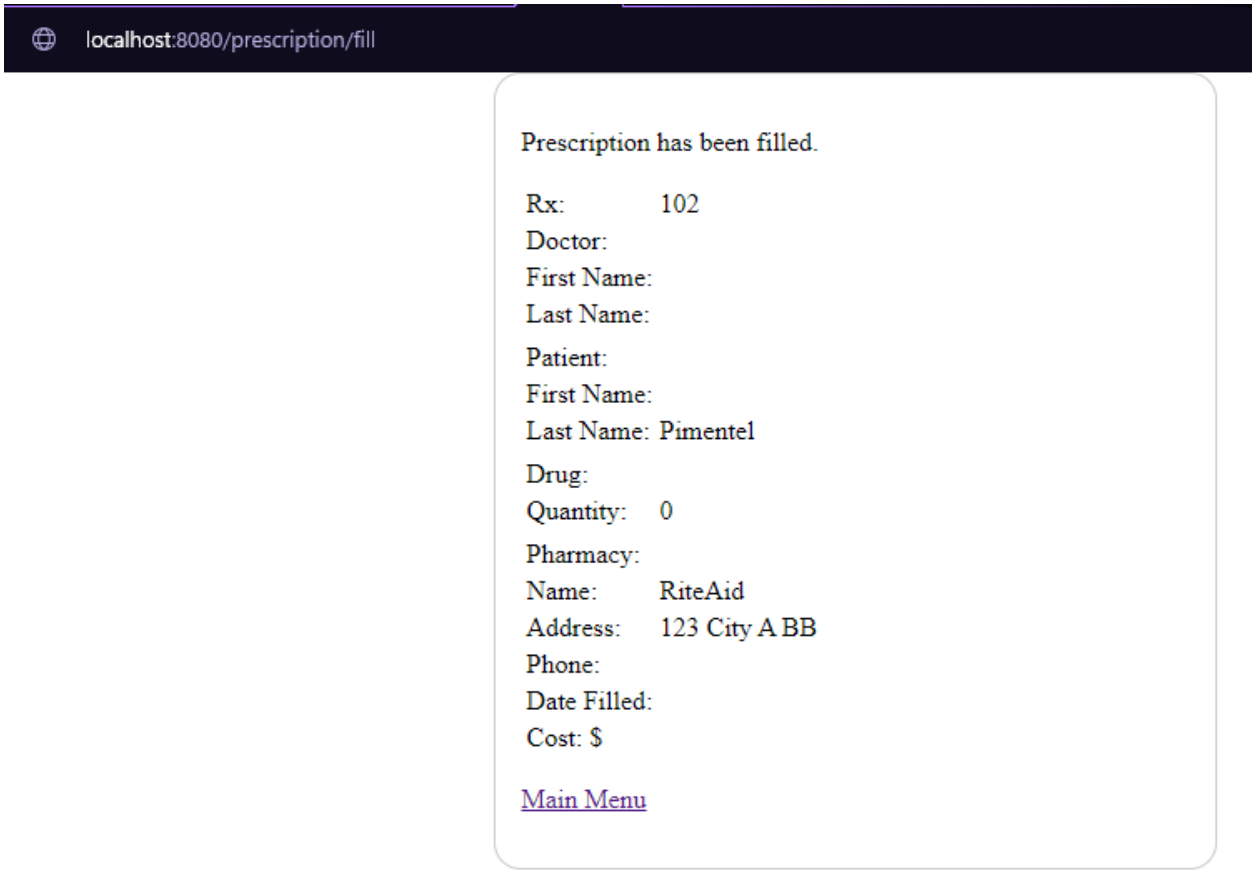
Pharmacy Report:



New Prescription



Patient Fills



New Patient

Register as new user

Your SSN:	•••••••••
Your First Name:	Charles
Your Last Name:	Zieres
Birth Date:	08/12/1976
Street:	412 N Earlham St
City:	Orange
State:	California
Zipcode:	92869
Primary Physician Name:	John Brown

Register

Registration successful.

Patient ID:	104
First Name:	Charles
Last Name:	Zieres
Birthdate:	1976-08-12
Street:	412 N Earlham St
City:	Orange
State:	California
Zipcode:	92869
Primary Physican:	John Brown

Edit | Main Menu

Appendix E

JDBC Console Logs

DataGenerate.Java Console Log:

Connecting to database...

Generating all the things

Creating patient statement...

All patients generated!

Closing patient statement...

Patients checked in

Creating doctor statement...

Generating doctors...

All doctors generated!

Closing doctor statement

Doctors registered

Creating prescription statement...

All prescriptions written!

Closing prescription statement

Drugs distributed