University of British Columbia, Vancouver

Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 2

Date: Oct. 20, 2023

Group Number: 8

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Krisha Iquin	67576298	p0d0a	krishaiquin@gmail.com
Elizaveta Firsova	14255541	o9o9b	lu_evrata@mail.ru
Matthew Cai	54356670	b7x4z	matthewcaiuni@gmail.com

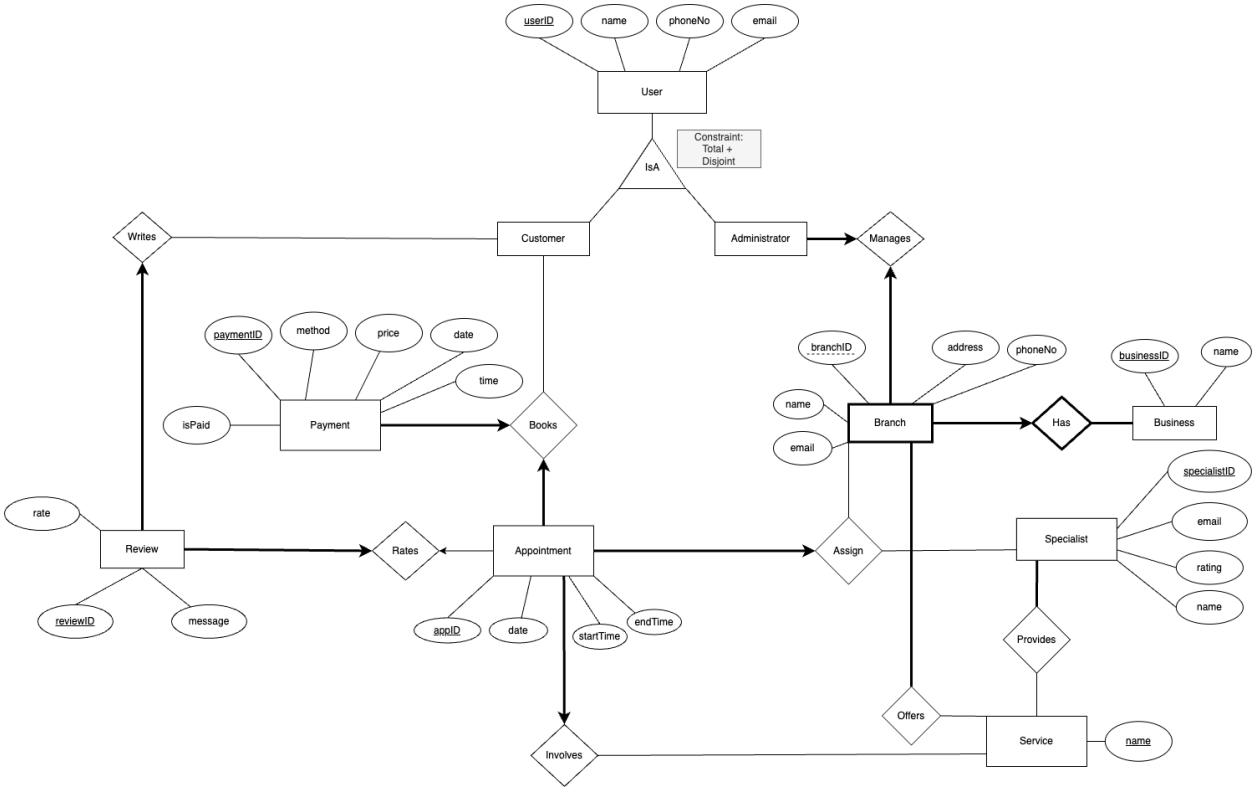
By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

University of British Columbia, Vancouver

Department of Computer Science

2) A Centralized Business Appointment Scheduling System. It is a platform with the ability to book, cancel and modify appointments; stores information about the customers, businesses, appointments and specialists, and all the listed entities above.
3) ER Diagram



Note: 1) appointment can have just one review (before several) because having multiple reviews from the same user for a single appointment could lead to redundancy and potential misuse of the review system 2) payment should be associated with customer and appointment (before: was no participation constraint) because it should not exist if appointment has not been scheduled by a customer.
Added time attribute for payment to create FD.

4) Schema:

Review(<u>reviewID</u>: INT, message: VARCHAR(255), rate: INT NOT NULL, **appID**(CK): INT NOT NULL UNIQUE, **userID**: INT NOT NULL)

Appointment(<u>appID</u>: INT, date: DATE NOT NULL, startTime: TIME NOT NULL, endTime: TIME NOT NULL, **serviceName**: VARCHAR(100) NOT NULL**, paymentID**(CK): INT NOT NULL

University of British Columbia, Vancouver

Department of Computer Science

UNIQUE**, userID**: INT NOT NULL, **specialistID**: INT NOT NULL, **branchID**: INT NOT NULL, **businessID**: INT NOT NULL, **reviewID**(CK): INT UNIQUE)

Payment(paymentID: INT, isPaid: BOOLEAN NOT NULL, method: VARCHAR(50), price: DECIMAL(10, 2), date: DATE, time: TIME, **appID**(CK): INT NOT NULL UNIQUE, **userID**: INT NOT NULL)

Specialist(specialistId: INT, email(CK): VARCHAR(320) NOT NULL UNIQUE, rating: DECIMAL(2,1), name: VARCHAR(100) NOT NULL)

Provides(**specialistID:** INT, **serviceName:**VARCHAR(50)) *we will need assertions to cover this and will add them after we have been taught them*

Service(name: VARCHAR(100))

Business(businessID: INT, name: VARCHAR(100) NOT NULL)

Customer(userID: INT, name: VARCHAR (100) NOT NULL, phoneNo: VARCHAR(100), email(CK): VARCHAR(320) NOT NULL UNIQUE)

Administrator(userID: INT, name: VARCHAR (100) NOT NULL, phoneNo: VARCHAR (100) NOT NULL, email(CK): VARCHAR (320) NOT NULL UNIQUE, **branchID**: INT NOT NULL UNIQUE (in combo with business), **businessID**: INT NOT NULL UNIQUE (in combo with branch))

Branch(branchID: INT, **businessID**: INT NOT NULL, name: VARCHAR (100) NOT NULL, address: VARCHAR (100), phoneNo: VARCHAR (100) NUT NULL, email: VARCHAR(320), **userID**(CK): INT NOT NULL UNIQUE)

Offers(**branchID**: INT, **businessID**: INT, **serviceName**: (100)) *we will need assertions to cover this and will add them after we have been taught them*

5) Functional Dependencies

Review(reviewID: INT, message: VARCHAR(255), rate: INT NOT NULL, **appID**(CK): INT NOT NULL UNIQUE, **userID**: INT NOT NULL)

reviewID -> message, rate, appID, userID
appID -> userID, reviewID
userID, message -> appID

Appointment(appID: INT, date: DATE NOT NULL, startTime: TIME NOT NULL, endTime: TIME NOT NULL, **serviceName**: VARCHAR(100) NOT NULL**, paymentID**(CK): INT NOT NULL

University of British Columbia, Vancouver

Department of Computer Science

UNIQUE**, userID**: INT NOT NULL, **specialistID**: INT NOT NULL, **branchID**: INT NOT NULL, **businessID**: INT NOT NULL, **reviewID**(CK): INT UNIQUE)

appID -> date, startTime, endTime, serviceName, paymentID, userID, specialistID, branchID, businessID, reviewID
paymentID -> userID, appID
reviewID -> userID
specialistID, date, startTime -> branchID, businessID
businessID, specialistID, date, startTime -> userID

Payment(paymentID: INT, isPaid: BOOLEAN NOT NULL, method: VARCHAR(50), price: DECIMAL(10, 2) NOT NULL, date: DATE, time: TIME, **appID**(CK): INT NOT NULL UNIQUE, **userID**: INT NOT NULL)

paymentID, -> isPaid, method, price, date, time, appID, userID
appID -> paymentID, price
userID, date, time -> method

Specialist(specialistID: INT, email (CK): VARCHAR(320) NOT NULL UNIQUE, rating: DECIMAL(2,1), name: VARCHAR(100) NOT NULL)

specialistID -> email, rating, name
email -> specialistID

Provides(**specialistID:** INT, **serviceName:**VARCHAR(50)) *we will need assertions to cover this and will add them after we have been taught them*

Service(name: VARCHAR(100))

No FDs

Business(businessID: INT, name: VARCHAR(100) NOT NULL)

businessID -> name

Customer(userID: INT, name: VARCHAR (100) NOT NULL, phoneNo: VARCHAR(100), email(CK): VARCHAR(320) NOT NULL UNIQUE)

userID -> name, phoneNo, email
email -> userID

University of British Columbia, Vancouver

Department of Computer Science

Administrator(<u>userID</u>: INT, name: VARCHAR (100) NOT NULL, phoneNo: VARCHAR (100) NOT NULL, email(CK): VARCHAR (320) NOT NULL UNIQUE, **branchID**: INT NOT NULL UNIQUE (in combo with business), **businessID**: INT NOT NULL UNIQUE (in combo with branch))

userID -> Name, PhoneNo, email, branchID, businessID
email -> userID, branchID, businessID
branchID, businessID -> userID

Branch(<u>branchID</u>: INT, **<u>businessID</u>**: INT NOT NULL, name: VARCHAR (100) NOT NULL, address: VARCHAR (100), phoneNo: VARCHAR (100) NUT NULL, email: VARCHAR(320), **userID**(CK): INT NOT NULL UNIQUE) *do not use userID as primary key since from ER diagram userID is not direct attribute of the Branch entity*

branchID, businessID -> name, address, phoneNo, userID, email
userID -> branchID, businessID

Offers(**<u>branchID</u>**: INT NOT NULL, **<u>businessID</u>**: INT NOT NULL, **<u>serviceName</u>**: (100)) *we will need assertions to cover this and will add them after we have been taught them*

6) Normalization

Appointment(<u>appID</u>: INT, date: DATE NOT NULL, startTime: TIME NOT NULL, endTime: TIME NOT NULL, **serviceName**: VARCHAR(100) NOT NULL**, paymentID**(CK): INT NOT NULL UNIQUE**, userID**: INT NOT NULL, **specialistID**: INT NOT NULL, **branchID**: INT NOT NULL, **businessID**: INT NOT NULL, **reviewID**(CK): INT UNIQUE)

appID -> date, startTime, endTime, serviceName, paymentID, userID, specialistID, branchID, businessID, reviewID
paymentID -> userID, appID
reviewID -> userID, appID
specialistID, date, startTime -> branchID, businessID
businessID, specialistID, date, startTime -> userID

Find minimal cover:

Step 1	Step 2	Step 3
appID -> date	appID -> date	appID -> date
appID -> startTime	appID -> startTime	appID -> startTime

appID -> endTime	appID -> endTime	appID -> endTime
appID -> serviceName	appID -> serviceName	appID -> serviceName
appID -> paymentID	appID -> paymentID	appID -> paymentID
appID -> userID	appID -> userID	X
appID -> specialistID	appID -> specialistID	appID -> specialistID
appID -> branchID, businessID	appID -> branchID, businessID	X
appID -> reviewID	appID -> reviewID	appID -> reviewID
paymentID -> userID	paymentID -> userID	X
paymentID -> appID	paymentID -> appID	paymentID -> appID
reviewID -> userID	reviewID -> userID	X
reviewID -> appID	reviewID -> appID	reviewID -> appID
specialistID, date, startTime -> branchID	specialistID, date, startTime ->branchID	specialistID, date, startTime ->branchID
specialistID, date, startTime -> businessID	specialistID, date, startTime -> businessID	specialistID, date, startTime -> businessID
businessID, specialistID, date, startTime -> userID	specialistID, date, startTime -> userID	specialistID, date, startTime -> userID

specialistID, date, startTime -> businessID is not in 3NF, decompose:
R_1(specialistID, date, startTime, **businessID**), R_2(appID, **date, startTime, specialistID,** endTime, **serviceName, paymentID**(CK)**, reviewID, branchID**)

specialistID, date, startTime -> branchID is not in 3NF, decompose:
R_3(specialistID, date, startTime, **branchID**),R_4(appID, **date, startTime, specialistID,** endTime, **serviceName, paymentID**(CK)**, reviewID**)

Final relationships: R_{13}(**specialistID**: INT, date: DATE NOT NULL, startTime: TIME NOT NULL, **branchID**: INT NOT NULL**, businessID**: INT NOT NULL)(if branchID and businessID are separate we will not be able to reference Branch entity anymore because it has compound foreign key), R_4(appID: INT, **date**: DATE NOT NULL**, startTime**: TIME NOT NULL**, specialistID**:

INT NOT NULL**,** endTime: TIME NOT NULL, **serviceName**: VARCHAR(100) NOT NULL**,** **paymentID**(CK): INT NOT NULL UNIQUE**, reviewID**(CK): INT UNIQUE), R_5(**specialistID**: INT, date: INT NOT NULL, startTime:TIME NOT NULL, **userID**: INT NOT NULL) (lost dependency)

Payment(paymentID: INT, isPaid: BOOLEAN NOT NULL, method: VARCHAR(50), price: DECIMAL(10, 2), date: DATE, time: TIME, **appID**(CK): INT NOT NULL UNIQUE, **userID**: INT NOT NULL)

paymentID, -> isPaid, method, price, date, time, appID, userID
appID -> paymentID
userID, date, time -> method

Find minimal cover

Step 1	Step 2	Step 3
paymentID -> isPaid	paymentID -> isPaid	paymentID -> isPaid
paymentID -> method	paymentID -> method	X
paymentID -> price	paymentID -> price	paymentID -> price
paymentID -> date	paymentID -> date	paymentID -> date
paymentID -> time	paymentID -> time	paymentID -> time
paymentID -> appID	paymentID -> appID	paymentID -> appID
paymentID -> userID	paymentID -> userID	paymentID -> userID
appID -> paymentID	appID -> paymentID	appID -> paymentID
userID, date, time -> method	userID, date, time -> method	userID, date, time -> method

userID, date, time -> method is not in BCNF, decompose:
R_1(**userID**, date, time, method); R_2(paymentID, isPaid, price, **date, time, userID, appID**)

Final result: R_1(**userID**: INT NOT NULL, date: DATE, time: TIME, method: VARCHAR(50)), R_2(paymentID: INT, isPaid: BOOLEAN NOT NULL, price: DECIMAL(10, 2), **date**: DATE**, time**: TIME**, userID**: INT NOT NULL**, appID**(CK): INT NOT NULL UNIQUE)

7) SQL DDL Statements

University of British Columbia, Vancouver

Department of Computer Science

Review(reviewID: INT, message: VARCHAR(255), rate: INT NOT NULL, **appID**(CK): INT NOT NULL UNIQUE, **userID**: INT NOT NULL)

```
CREATE TABLE Review(
        reviewID INT PRIMARY KEY,
        message VARCHAR(255),
        rate INT NOT NULL,
        userID INT NOT NULL,
        appID INT UNIQUE NOT NULL,
        FOREIGN KEY (userID) REFERENCES Customer(userID),
        FOREIGN KEY (appID) REFERENCES Appointment(appID)
)
```

SpecialistTimeslot_Location(**specialistID**: INT, date: DATE NOT NULL, startTime: TIME NOT NULL, **branchID**: INT NOT NULL, **businessID**: INT NOT NULL),

Appointment(appID: INT, **date**: DATE NOT NULL**, specialistID**: INT NOT NULL**, startTime**: TIME NOT NULL**,** endTime: TIME NOT NULL, **serviceName**: VARCHAR(100) NOT NULL**, paymentID**(CK): INT NOT NULL UNIQUE**, reviewID**(CK): INT UNIQUE),

SpecialistTimeslot_Customer(**specialistID**: INT, date: INT NOT NULL, startTime: TIME NOT NULL, **userID**: INT NOT NULL)

```
CREATE TABLE SpecialistTimeslot_Location(
        specialistID INT,
        date DATE,
        startTime TIME,
        branchID INT NOT NULL,
        businessID INT NOT NULL,
        PRIMARY KEY (specialistID, date, startTime),
        FOREIGN KEY (specialistID) REFERENCES Specialist(specialistID)
                ON DELETE CASCADE,
        FOREIGN KEY (branchID, branchID) REFERENCES Branch(branchID, branchID)
                ON DELETE CASCADE
)
```

```
CREATE TABLE Appointment(
        appID INT PRIMARY KEY,
        date DATE NOT NULL,
        specialistID INT,
        startTime TIME NOT NULL,
        endTime TIME NOT NULL,
```

```
        serviceName VARCHAR(100) NOT NULL,
        paymentID UNIQUE INT NOT NULL,
        reviewID INT UNIQUE,
        UNIQUE(specialitID, date, startTime),
        FOREIGN KEY (specialistID) REFERENCES Specialist(userID)
                ON DELETE CASCADE,
        FOREIGN KEY (specialistID, date, startTime) REFERENCES
                SpecialistTimeslot_Location(specialistID, date, startTime)
                        ON DELETE SET DEFAULT,
        FOREIGN KEY (serviceName) REFERENCES Service(name)
                        ON DELETE SET DEFAULT,
        FOREIGN KEY (paymentID) REFERENCES Payment(paymentID)
                        ON DELETE SET DEFAULT,
        FOREIGN KEY (reviewID) REFERENCES Review(reviewID)
)

CREATE TABLE SpecialistTimeslot_Customer(
        specialistID INT,
        date DATE,
        startTime TIME,
        userID INT NOT NULL,
        PRIMARY KEY (specialistID, date, startTime),
        FOREIGN KEY (specialistID) REFERENCES Specialist(specialistID)
                ON DELETE CASCADE,
        FOREIGN KEY (userID) REFERENCES Customer(userID)
                ON DELETE CASCADE,
)
```

UserPayment_Method(**userID**: INT NOT NULL, date: DATE, time: TIME, method: VARCHAR(50)),

Payment(paymentID: INT, isPaid: BOOLEAN NOT NULL, price: DECIMAL(10, 2), **date**: DATE**,** **time**: TIME**, userID**: INT NOT NULL**, appID**(CK): INT NOT NULL UNIQUE)

```
CREATE TABLE UserPayment_Method(
        userID INT,
        date DATE,
        time TIME,
        method VARCHAR(50),
        PRIMARY KEY (userID, date, time),
        FOREIGN KEY (userID) REFERENCES Customer(userID)
                ON DELETE SET DEFAULT
```

)

```
CREATE TABLE Payment(
        paymentID INT PRIMARY KEY,
        isPaid BOOLEAN NOT NULL,
        price DECIMAL(10, 2),
        date DATE,
        time TIME,
        userID INT NOT NULL,
        appID INT NOT NULL,
        FOREIGN KEY (userID, date, time)
                REFERENCES UserPayment_Method(userID, date, time)
                        ON DELETE SET DEFAULT,
        FOREIGN KEY (userID) REFERENCES Customer(userID)
                ON DELETE SET DEFAULT,
        FOREIGN KEY (appID) REFERENCES Appointment(appID)
                ON DELETE SET DEFAULT
)

CREATE TABLE Specialist(
        specialistID INT PRIMARY KEY,
        email VARCHAR(320) NOT NULL UNIQUE,
        rating DECIMAL(2,1),
        name VARCHAR(100) NOT NULL
)
```

Provides - *we will need assertions to cover this and will add them after we have been taught them*

```
CREATE TABLE Service(
        name VARCHAR(100) PRIMARY KEY
)

CREATE TABLE Business(
        businessID INT PRIMARY KEY,
        name VARCHAR(100) NOT NULL
)
```

Customer(<u>userID</u>: INT, name: VARCHAR (100) NOT NULL, phoneNo: VARCHAR(100), email(CK): VARCHAR(320) NOT NULL UNIQUE)

University of British Columbia, Vancouver

Department of Computer Science

```
CREATE TABLE Customer(
        userID INT,
        name VARCHAR(100) NOT NULL,
        phoneNo VARCHAR(100),
        email VARCHAR(320) UNIQUE NOT NULL,
        PRIMARY KEY(userID)
)
```

Administrator(<u>userID</u>: INT, name: VARCHAR (100) NOT NULL, phoneNo: VARCHAR (100) NOT NULL, email(CK): VARCHAR (320) NOT NULL UNIQUE, **branchID**: INT NOT NULL UNIQUE (in combo with business), **businessID**: INT NOT NULL UNIQUE (in combo with branch))

```
CREATE TABLE Administrator(
        userID INT,
        name VARCHAR(100) NOT NULL,
        phoneNo VARCHAR(100) NOT NULL,
        email VARCHAR(320) UNIQUE NOT NULL,
        branchID INT NOT NULL,
        businessID INT NOT NULL,
        UNIQUE (branchID, businessID),
        PRIMARY KEY (userID),
        FOREIGN KEY (branchID) REFERENCES
                Branch(branchID)
                        ON DELETE CASCADE,
        FOREIGN KEY (businessID) REFERENCES
                Business(businessID)
                        ON DELETE CASCADE
)
```

Branch(<u>branchID</u>: INT, **<u>businessID</u>**: INT NOT NULL, name: VARCHAR (100) NOT NULL, address: VARCHAR (100), phoneNo: VARCHAR (100) NUT NULL, email: VARCHAR(320), **userID**(CK): INT NOT NULL UNIQUE)

```
CREATE TABLE Branch(
        branchID INT,
        businessID INT,
        name VARCHAR(100) NOT NULL,
        address VARCHAR(100),
        phoneNo VARCHAR(100) NOT NULL,
        email: VARCHAR(320),
        userID INT NOT NULL UNIQUE,
        PRIMARY KEY (branchID, businessID),
```

```
        FOREIGN KEY (businessID) REFERENCES
                Business(businessID)
                        ON DELETE CASCADE,
        FOREIGN KEY (userID) REFERENCES
                Administrator(userID)
                        ON DELETE SET DEFAULT
)
```

Offers - *we will need assertions to cover this and will add them after we have been taught them*

8. SQL Insert Statements

```
INSERT INTO Review (reviewID, message, rate, appID, userID)
VALUES
        (1, “Awesome experience”, 5, 1, 1),
        (2, “Not awesome experience”, 3, 2, 2),
        (3, “Liked this salon”, 4, 3, 1),
        (4, “Will not come back”, 2, 4, 4),
        (5, “Not satisfied”, 2, 5, 5)

INSERT INTO SpecialistTimeslot_Location(date, specialistID, startTime, branchID, businessID)
VALUES
        (‘17/12/2023’, 100, ‘13:30’, 1823, 8888),
        (‘29/09/2023’, 101, ‘09:00’, 9101, 1057),
        (‘01/08/2023’, 102, ‘11:30’, 1234, 1234),
        (‘01/08/2023’, 103, ‘10:30’, 5678, 5678),
        (‘09/07/2023’, 103, ‘11:30’, 5678, 5678)

INSERT INTO Appointment (appID, date, specialistID, startTime, endTime, serviceName,
        paymentID, reviewID)
VALUES
        (1, ‘17/12/2023’, 100, ‘13:30’, ‘15:30’, ‘Make-up’, 1, 1),
        (1, ‘29/09/2023’, 101, ‘09:00’, ‘10:00’, ‘Haircut’, 2, 2),
        (1, ‘01/08/2023’, 102, ‘11:30’, ‘13:00’, ‘Manicure’, 3, 3),
        (1, ‘01/08/2023’, 103, ‘10:30’, ‘11:15’, ‘Tattoo’, 4, 4),
        (1, ‘09/07/2023’, 103, ‘11:30’, ‘12:30’, ‘Tattoo’, 5, 5)

INSERT INTO SpecialistTimeslot_Customer(date, specialistID, startTime, userID)
VALUES
        (‘17/12/2023’, 100, ‘13:30’, 1),
        (‘29/09/2023’, 101, ‘09:00’, 2),
```

('01/08/2023', 102, '11:30', 1),
('01/08/2023', 103, '10:30', 4),
('09/07/2023', 103, '11:30', 5)

INSERT INTO UserPayment_Method(userID, date, time, method)
VALUES
('01/08/2023', '14:38', "Cash"),
('29/09/2023', '10:02', "Debit"),
('01/08/2023', '12:32', "Credit"),
('03/08/2023', '10:30', "E-Transfer"),
('11/07/2023', '14:11', "Credit")

INSERT INTO Payment(paymentID, isPaid, price, date, time, userID, appID)
VALUES
(1, 1, 70.25, '17/12/2023', '13:30', 1, 1),
(2, 1, 30.00, '29/09/2023', '09:00', 2, 2),
(3, 1, 180.50, '01/08/2023', '11:30', 1, 3),
(4, 0, 200.09, '03/08/2023', '10:30', 4, 4),
(5, 1, 401.02, '11/07/2023', '11:30', 5, 5)

INSERT
INTO Specialist (specialistID, email, rating, name)
VALUES
(100, 'zariabruce@yahoo.com', 5.0, 'Zaria Bruce'),
(101, 'ashlyn_fry@gmail.com', 3.5, 'Ashlyn Fry'),
(102, 'kaydensanchez@hotmail.com', 4.6, 'Kayden Sanchez'),
(103, 'cobyliu@outlook.com', 2.7, 'Coby Liu'),
(104, 'david_wang1@business.com', 1.5, 'David Wang')

Provides - *we will need assertions to cover this and will add them after we have been taught them*

INSERT
INTO Service (name)
VALUES
('Manicure'),
('Tattoo'),
('Physiotherapy'),

('Haircut'),
('Make-up')

INSERT
INTO Business (businessID, name)
VALUES
(1234, 'ACME Salon'),
(5678, 'Inkwell Tattoo Shop'),
(1057, 'The Barber Shop'),
(9384, 'Smile Thai Wellness'),
(8888, 'Sephora Mac Salon')

INSERT
INTO Customer (userID, name, phoneNo, email)
VALUES
(1, phillip chariot, 123-435-1239, 'philipchariot@gmail.com')
(2, jared chinatown, 132-421-8123, 'jc@gmail.com')
(3, alejandro mexico, 213-489-1293, 'am@gmail.com')
(4, lipa dua, 213-423-4823, 'dualipa@gmail.com')
(5, chess crackers, 123-483-9123, 'cc@gmail.com')

INSERT
INTO Administrator (userID, name, phoneNo, email, branchID, businessID)
VALUES
(1, james Doe, 111-111-1118, 'jamesdoe@hotmail.com', 1234, 1234)
(2, james cameron, 111-111-5222, 'jamescameron@outlook.com,' 5678, 5678)
(3, james cordon, 111-222-3353, 'james_cordon@gmail.com', 9101, 1057)
(4, james pepperoni , 222-333-6444, 'jamespepps@yahoo.com', 1, 9384)
(5, reginold leopold the fifth, 555-666-2777, 'regleothe5@regleo.com', 1823, 8888)

INSERT
INTO Branch(branchID, businessID, name, address, phoneNo, email, adminID)
VALUES
(1234, 1234, 'East Vancouver ACME Salon', '215 Tolmie St', 123-456-7899, 'Eastvanacme@gmail.com', 1)
(5678, 5678, 'Inkwell oakville', '210 North Service Rd W', 101-112-1314, 'oakville@inkwell.com', 2)
(9101, 1057, 'The Barber Shop LLoydmaster', '5612 44 St', 123-543-5673, 'thebarbershop@gmail.com' , 3)
(1, 9384, 'Smile Thai Kelowna', '1567 Pandosy St', 341-435-1235, 'Smilethaikelowna@yahoo.com', 4)

University of British Columbia, Vancouver

Department of Computer Science

(1823, 8888, ‘Sephora Bromont’ ‘229 de Bromont Boul’, 214-543-5673, ‘sephora@bromont.com’, 5)

Offers - *we will need assertions to cover this and will add them after we have been taught them*