

CS174a Project 3 Report – The Hospital

Task Division

Charles Weng

- Set up the JDBC and while loop through messages ResultSet
- UI for initial load database (database selection, user name)
- Updating patient information
- Debugged NULL table values and threw error messages for constraint violations
- Last two queries of Admin interface

Cameron McNair

- Prepared statements inside the Hospital class
- UI for initial load database (password field)
- UI for the three interfaces
- Viewing the patient information
- Viewing and editing Allergy information
- First two queries of Admin interface

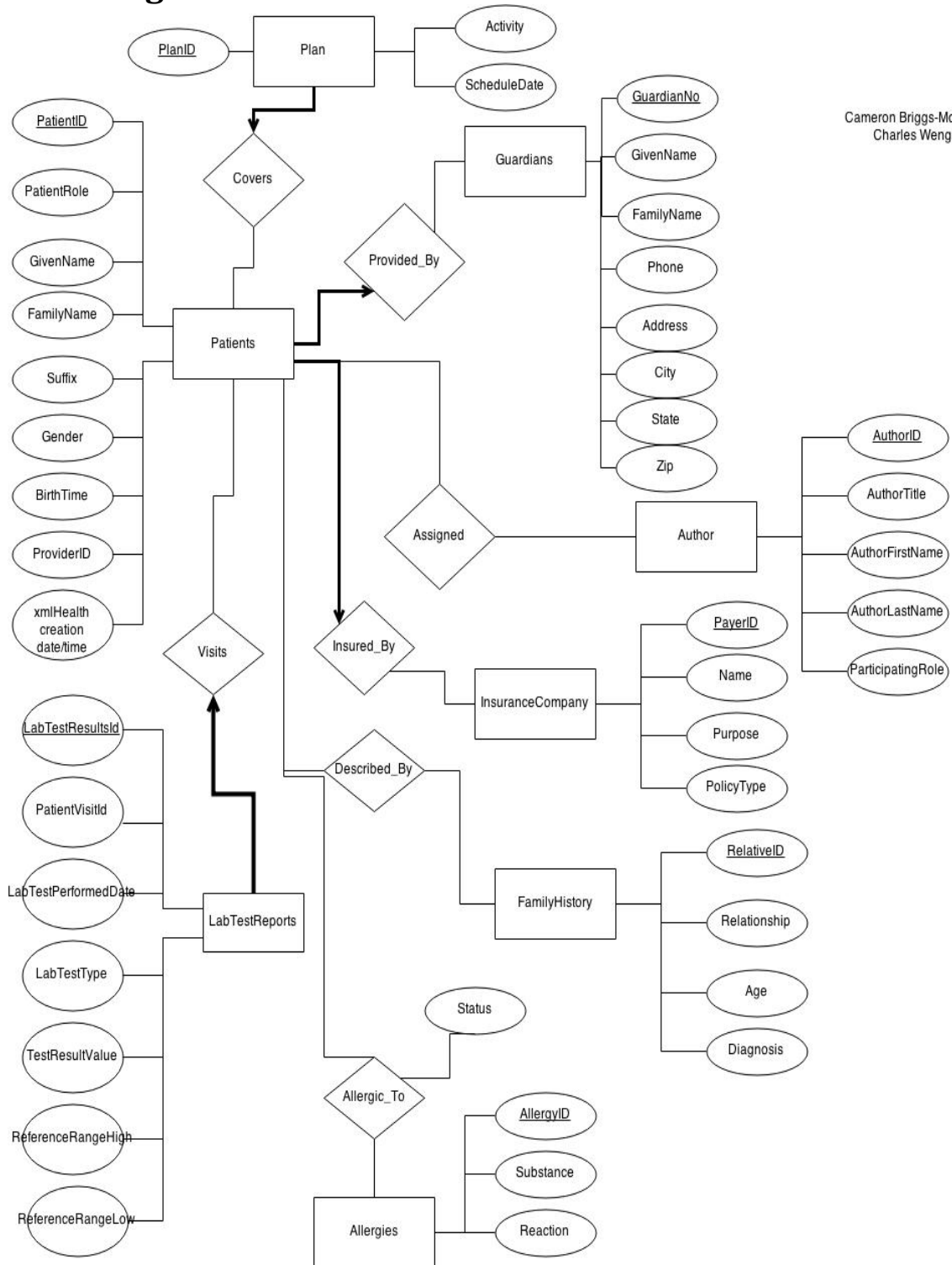
Parts of Schemas that Changed Since Project 2

- Changed the primary key from a tuple of (LabTestId, PatientVisitId) to just PatientVisitId, because LabTestResults cannot exist without a patient visit
- We deleted some NOT NULL's due to the fact that some values can be NULL inside our table
- We added and moved some Foreign Key Constraints
- Moved Status inside our UML diagram to our Allergic_To
- Added several more constraints like Unique, Foreign Key, etc.

How We Dealt With Constraints

- Constraints such as Primary Key Constraints are outputted to the console
- Foreign Key Constraints are outputted onto the stack trace

UML Diagram



Cameron Briggs-McNair
Charles Weng

Final Schema Definitions

Formatted for easier readability!

```
CREATE TABLE IF NOT EXISTS guardians
(
    guardianno INT,
    firstname CHAR(100),
    lastname CHAR(50),
    phone CHAR(20),
    address CHAR(100),
    city CHAR(50),
    state CHAR(20),
    zip INT,
    PRIMARY KEY(guardianno)
);

CREATE TABLE IF NOT EXISTS insurancecompany
(
    payerid INT,
    name CHAR(50),
    purpose CHAR(50),
    policytype CHAR(50),
    PRIMARY KEY(payerid)
);

CREATE TABLE IF NOT EXISTS author
(
    authorid INT,
    authortitle CHAR(100),
    authorfirstname CHAR(50),
    authorlastname CHAR(50),
    participatingrole CHAR(50),
    PRIMARY KEY(authorid)
);

CREATE TABLE IF NOT EXISTS labtestreports
(
    labtestresultsid INT,
    patientvisitid INT,
    labtestperformeddate DATETIME,
    labtesttype CHAR(50),
    testresultvalue INT,
    referencerangehigh CHAR(50),
    referencerangelow CHAR(50),
    PRIMARY KEY(labtestresultsid),
    UNIQUE(labtestresultsid, patientvisitid)
);

CREATE TABLE IF NOT EXISTS patient
(
    patientid INT,
    patientrole INT,
    givenname CHAR(100),
    familyname CHAR(50),
    suffix CHAR(10),
    gender CHAR(8),
    birthtime DATETIME,
    providerid CHAR(50),
    xmlhealthcreationdatetime DATETIME,
    payerid INT,
    PRIMARY KEY(patientid),
```

```

        FOREIGN KEY(patientrole) REFERENCES guardians(guardianno),
        FOREIGN KEY(payerid) REFERENCES insurancecompany(payerid)
    );

CREATE TABLE IF NOT EXISTS assigned
(
    patientid INT,
    authorid INT,
    PRIMARY KEY(patientid, authorid),
    FOREIGN KEY(patientid) REFERENCES patient(patientid),
    FOREIGN KEY(authorid) REFERENCES author(authorid)
);

CREATE TABLE IF NOT EXISTS familyhistory
(
    relativeid INT,
    relationship CHAR(50),
    age INT,
    diagnosis CHAR(100),
    PRIMARY KEY(relativeid)
);

CREATE TABLE IF NOT EXISTS describedby
(
    patientid INT,
    relativeid INT,
    PRIMARY KEY(patientid, relativeid),
    FOREIGN KEY(patientid) REFERENCES patient(patientid),
    FOREIGN KEY(relativeid) REFERENCES familyhistory(relativeid)
);

CREATE TABLE IF NOT EXISTS allergies
(
    allergyid INT,
    patientid INT NOT NULL,
    substance CHAR(50),
    reaction CHAR(50),
    status CHAR(20),
    PRIMARY KEY(allergyid),
    FOREIGN KEY(patientid) REFERENCES patient(patientid)
);

CREATE TABLE IF NOT EXISTS plan
(
    planid INT,
    activity CHAR(50),
    scheduledate DATETIME,
    patientid INT NOT NULL,
    PRIMARY KEY(planid),
    FOREIGN KEY(patientid) REFERENCES patient(patientid)
);

CREATE TABLE IF NOT EXISTS visits
(
    patientvisitid INT,
    patientid INT,
    labtestresultsid INT,
    PRIMARY KEY(patientvisitid),
    FOREIGN KEY(patientid) REFERENCES patient(patientid),
    FOREIGN KEY(labtestresultsid) REFERENCES labtestreports(labtestresultsid)
);

```

Queries Used

Note: '?' represents things that can be replaced depending on other factors in a query

Patient

- Update Patient set GivenName = ?, FamilyName = ?, Suffix = ?, Gender = ?, Birthtime = ?, ProviderID = ?, PayerID = ? where PatientID = ?
- select * from (Patient join Guardians on Patient.PatientRole = Guardians.GuardianNo) where PatientId = ?
- select * from Patient where PatientID = ?

Doctor

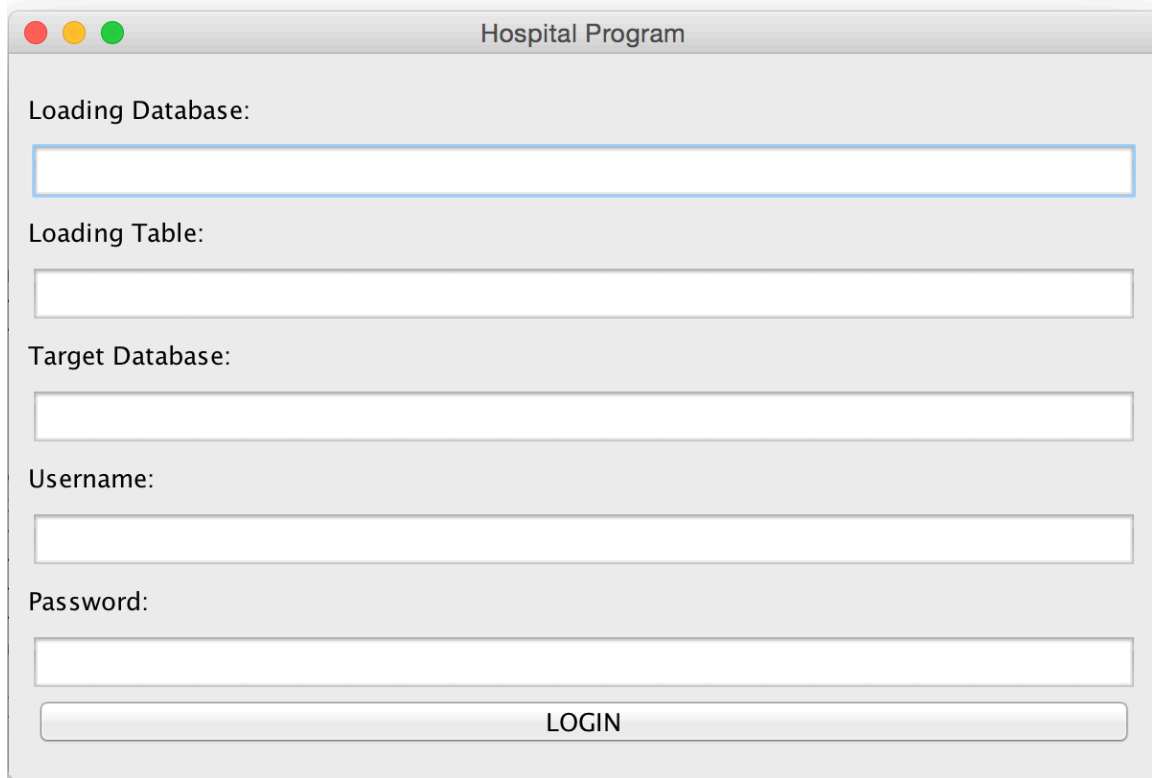
- Update Allergies set Substance = ?, Reaction = ?, Status = ? where PatientID = ?
- select * from Allergies where PatientID = ?
- select * from Plan where PatientID = ?

Administrator

- select count(*) from (SELECT distinct(substance) from allergies where substance is not null group by substance) a
- select distinct(substance), count(*) from allergies where substance is not null group by substance
- select PatientID, count(*) from allergies group by PatientID having count(*) > 1
- select count(*) from (select a.PatientID, count(*) from allergies a group by a.PatientID having count(*) > 1) AS MultiAllergy
- select * from Plan pl, Patient pt where pl.PatientID = pt.PatientID and DATE(pl.ScheduleDate) = DATE(CURDATE())
- select a.AuthorID, a.AuthorTitle, a.AuthorFirstName, a.AuthorLastName, a.ParticipatingRole from Author a where a.AuthorID in (select AuthorID from Assigned group by AuthorID having count(*) > 1)

UI Description

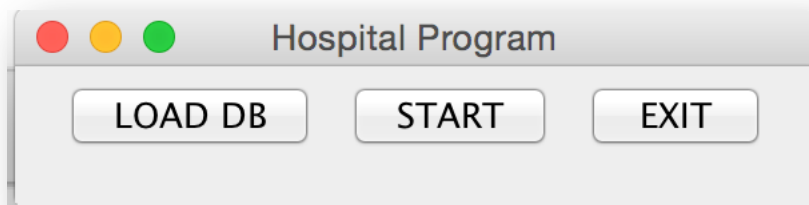
Login Screen - Initial login screen that preloads your database into your code



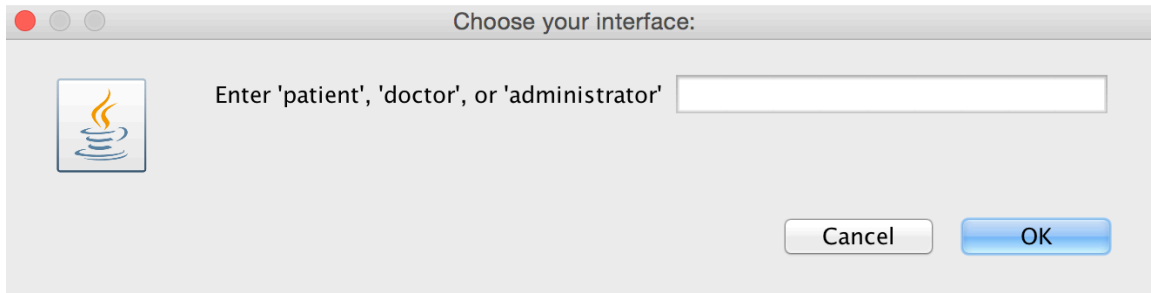
The screenshot shows a window titled "Hospital Program" with a light gray background. It contains several input fields and a button. The fields are labeled "Loading Database:", "Loading Table:", "Target Database:", "Username:", and "Password:". Each label is followed by a white rectangular input box. At the bottom, there is a "LOGIN" button. The window has standard macOS-style window controls (red, yellow, green buttons) in the top-left corner.

Main Menu

- Loads the DB given and puts it into our DB
- Start our Java Program that allows to edit and change info about Patients
- Exit exits our Main Menu



Interface Selection Menu – Choose Patient, Doctor, or Admin UI



Patient and Doctor Interfaces – Figure 1 => Enter Patient ID

Figure 2 => View, Edit, Exit a Patient's Info

Figure 3 => View

Figure 4 => Edit

Figure 1.

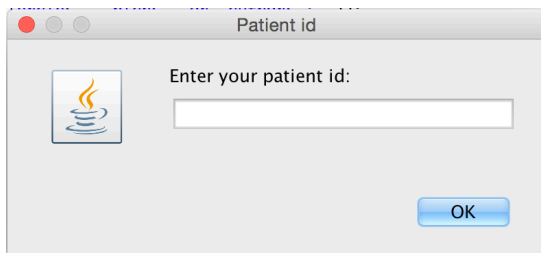


Figure 2.

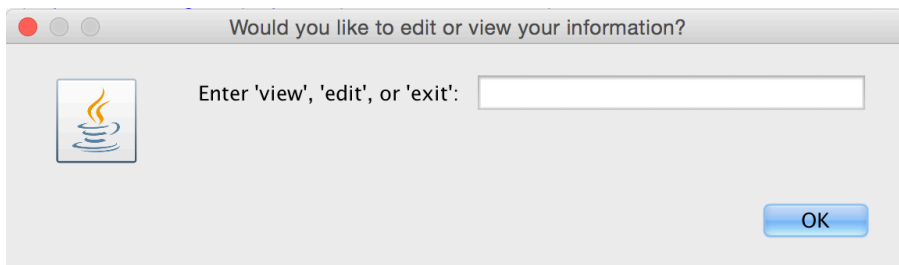
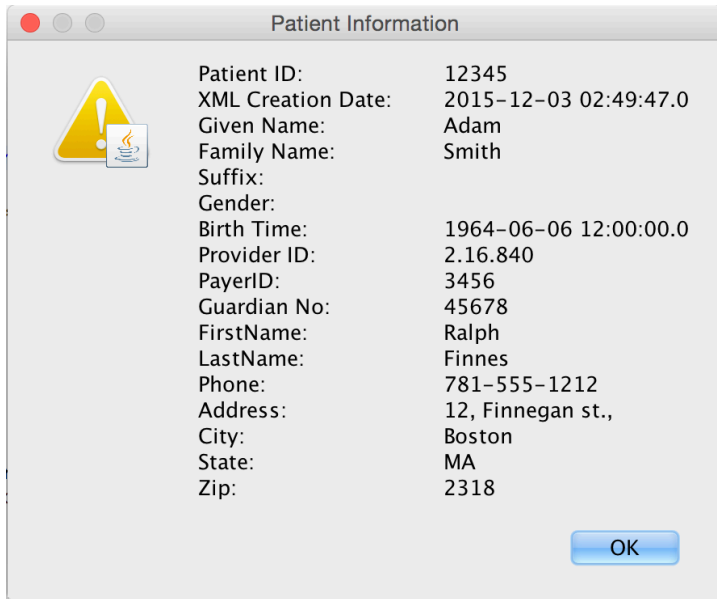


Figure 3.

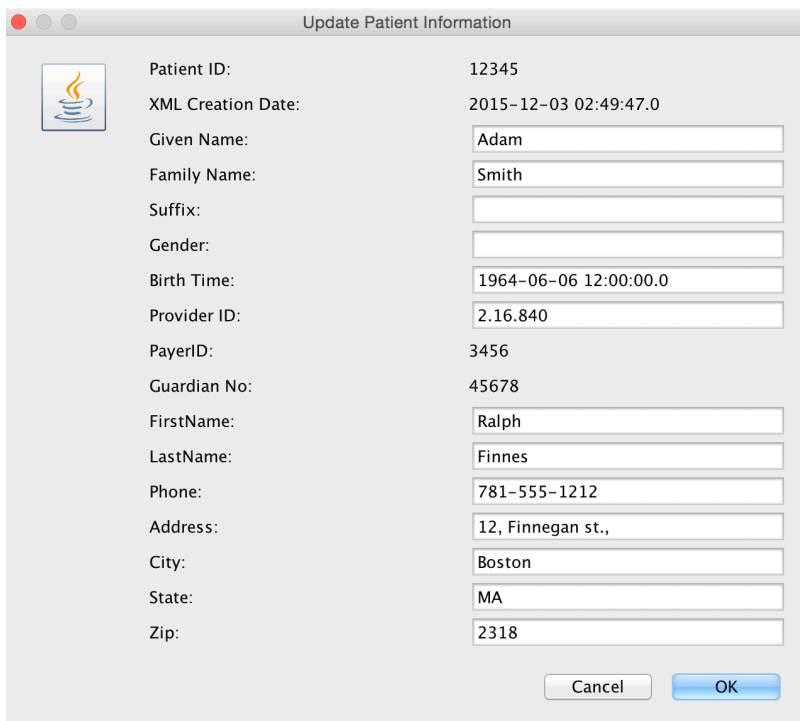


A dialog box titled "Patient Information" with a yellow warning icon. It displays a list of patient details in a read-only format. The details include Patient ID, XML Creation Date, Given Name, Family Name, Suffix, Gender, Birth Time, Provider ID, PayerID, Guardian No, FirstName, LastName, Phone, Address, City, State, and Zip. An "OK" button is located at the bottom right.

Patient ID:	12345
XML Creation Date:	2015-12-03 02:49:47.0
Given Name:	Adam
Family Name:	Smith
Suffix:	
Gender:	
Birth Time:	1964-06-06 12:00:00.0
Provider ID:	2.16.840
PayerID:	3456
Guardian No:	45678
FirstName:	Ralph
LastName:	Finnes
Phone:	781-555-1212
Address:	12, Finnegan st.,
City:	Boston
State:	MA
Zip:	2318

OK

Figure 4.



A dialog box titled "Update Patient Information" with a blue icon. It displays a list of patient details in a form where most fields are read-only, but several are editable text boxes. The details include Patient ID, XML Creation Date, Given Name, Family Name, Suffix, Gender, Birth Time, Provider ID, PayerID, Guardian No, FirstName, LastName, Phone, Address, City, State, and Zip. "Cancel" and "OK" buttons are located at the bottom right.

Patient ID:	12345
XML Creation Date:	2015-12-03 02:49:47.0
Given Name:	<input type="text" value="Adam"/>
Family Name:	<input type="text" value="Smith"/>
Suffix:	<input type="text"/>
Gender:	<input type="text"/>
Birth Time:	<input type="text" value="1964-06-06 12:00:00.0"/>
Provider ID:	<input type="text" value="2.16.840"/>
PayerID:	3456
Guardian No:	45678
FirstName:	<input type="text" value="Ralph"/>
LastName:	<input type="text" value="Finnes"/>
Phone:	<input type="text" value="781-555-1212"/>
Address:	<input type="text" value="12, Finnegan st.,"/>
City:	<input type="text" value="Boston"/>
State:	<input type="text" value="MA"/>
Zip:	<input type="text" value="2318"/>

Cancel OK

Admin Interface – Figure 1 => Four different types of views we can choose from
Figure 2 => Querying for counts of allergies of each
Figure 3 => Patients with multiple Allergies
Figure 4 => Plans for today
Figure 5 => Authors with multiple patients assigned

Figure 1.

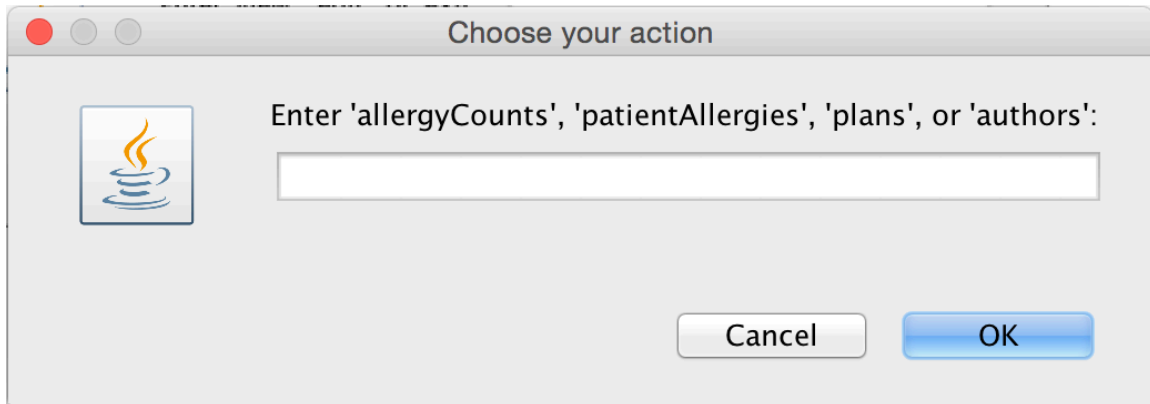


Figure 2.

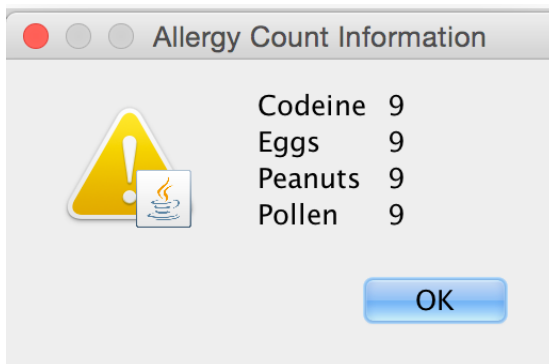


Figure 3.

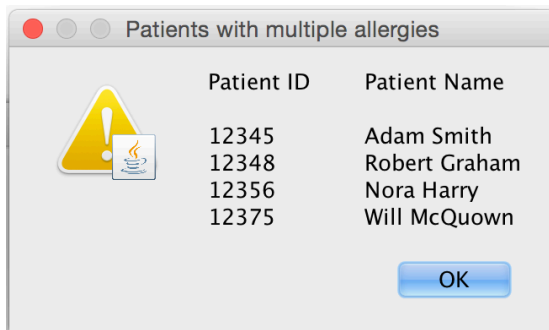
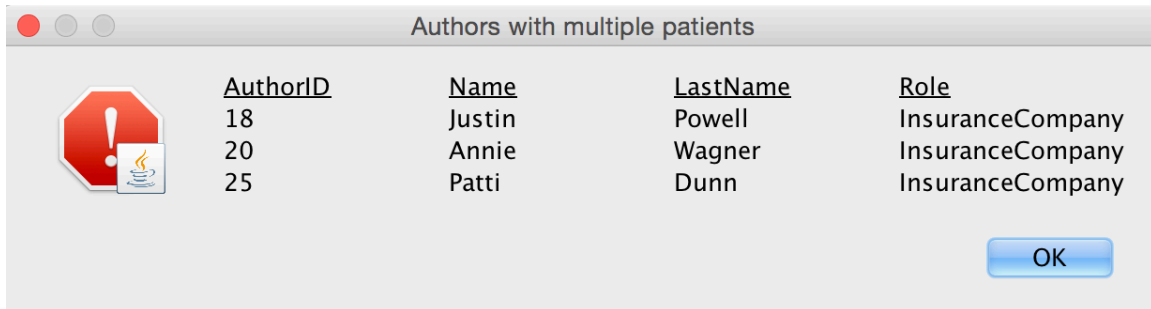


Figure 4.



PatientID	PlanID	Name	LastName	Activity	ScheduleDate
12345	456	Adam	Smith	Colonoscopy	2015-03-12 12:00:00.0

Figure 5.



AuthorID	Name	LastName	Role
18	Justin	Powell	InsuranceCompany
20	Annie	Wagner	InsuranceCompany
25	Patti	Dunn	InsuranceCompany

Classes and Methods Implemented

Classes	Methods
MainMenu	MainMenu constructor – set up our UI resetFrame – logic for loading db vs ui doInterface – choosing between admin, patient, doctor main – main thread where our ui is running
Hospital	Hospital constructor – loading db stuff into our own database notNull – if null for primary keys we don't insert into certain ones of our table
PatientInterface	PatientInterface – Choosing between view and edit patient info
DoctorInterface	DoctorInterface – choosing between viewing allergy counts and allergy info for each patient
AdminInterface	AdminInterface – combined querying and selections on admin
Patient	Patient – place where all the queries are made
Doctor	Doctor – place where all allergy queries are made