Concoctions

Written by

- Matt Greene greene.matthew@northeastern.edu
- Daniel Blum blum.da@northeastern.edu

Requirements

1. Java 11+ (we like sdkman to manage our java)
2. Docker ***OR*** MySql
3. Maven (if you're going to compile the backend and frontend)
4. A can do attitude

Technical Specifications

Database

The database is a MySQL database

Backend

The backend uses a java with libraries like Spring to simplify communication with the MySQL server and create a simplified API over http (using a *rest like* protocol. . . .though it's very *not* rest).

Frontend

The frontend uses java to create a CLI as a client for the backend api.

Running

tl;dr

open up a terminal `cd` into the project directory.

run the database server and backend server

```
docker-compose -f ./docker/docker-compose.yml up --build -d
java -jar concoctionsBackend.jar
```

open up another terminal and `cd` into the project directory

run the frontend client

```
java -jar concoctionsFrontend.jar
```

Database

--- Docker

If you want to dockerize the DB, from the main project directory run:

```
docker-compose -f ./docker/docker-compose.yml up --build -d
```

Docker is running the server on `3306`. If you want to change this, you'll need to

1. change `./docker/docker-compose.yml` under the ports line. It's in the format `[output port]:[internal port]`
2. change `concoctionsBackend/src/main/resources/application.properties` so the line read `spring.datasource.url=jdbc:mysql://localhost:[ENTER output port NUMBER HERE]/concoctionsDB`
3. follow the recompile instructions below for the backend.

If there's an issue, you can run

```
docker-compose -f ./docker/docker-compose.yml
```

--- Local MySQL

If you want to create and initialize the database yourself, run the sql files `01_init.sql` and `02_starterData.sql` in your MySQL instance.

Backend/Frontend

--- Jars

Both the backend and frontend are packaged in fatjars, so the only thing you need install is Java 11+. So, from the main project folder, go ahead and run:

```
java -jar concoctionsBackend.jar
```

and in a another terminal (unless you're running this `nohup`)

```
java -jar concoctionsFrontend.jar
```

--- Compile and run

Both projects are using maven so make sure that's installed. Then, `cd` into either sub-project's folder and run

```
mvn clean
mvn package
```

`mvn package` will create a jar file in the sub-projects `target` directory.

For the backend, there will only be one jar to run, so just `java -jar [new_jar_file].jar` that file.

For the front end, you'll see two jars, so you'll want to run the jar that's appended with `with-dependencies.jar`

Lessons Learned

Daniel Blum

I spent a lot of time learning a bit about Spring (and it's "inversion of control") and Spring-Boot to make this backend server.

Once I got a little handle on Spring, it took sometime to understand how to import "objects" from the database to persist as objects in java. Mainly, about how to get objects that had relations through table (like drinks to ingredients) into java objects that had fields (like a drinks field of a List).

After all that, I had to get a handle on how to get a json object from "a user" and convert that into a proper java object that *then* could be sent to be persisted in the database.

So, not as much as working in SQL, but more about how to get java to work with SQL.

Matt Greene

I spent a lot of time also learning about the spring architecture to understand how to best utilize and call everything to make the data base working.

I originally started trying to parson the Json's by hand but was then pointed to Gson which is much easier.

After discovering Gson and brushing up on generics, the passing of objects became very easy.

Using the native Http client was a bit of a pain but after some googling it became much easier. The biggest lesson learned between the Httpclient and Gson is that you have to pass the Json as a converted string not convert to string then send.

Lesson Relearned - even if you think its going to be a quick and easy controller, plan more for the code architecture, and you will deal with a lot less chaos.

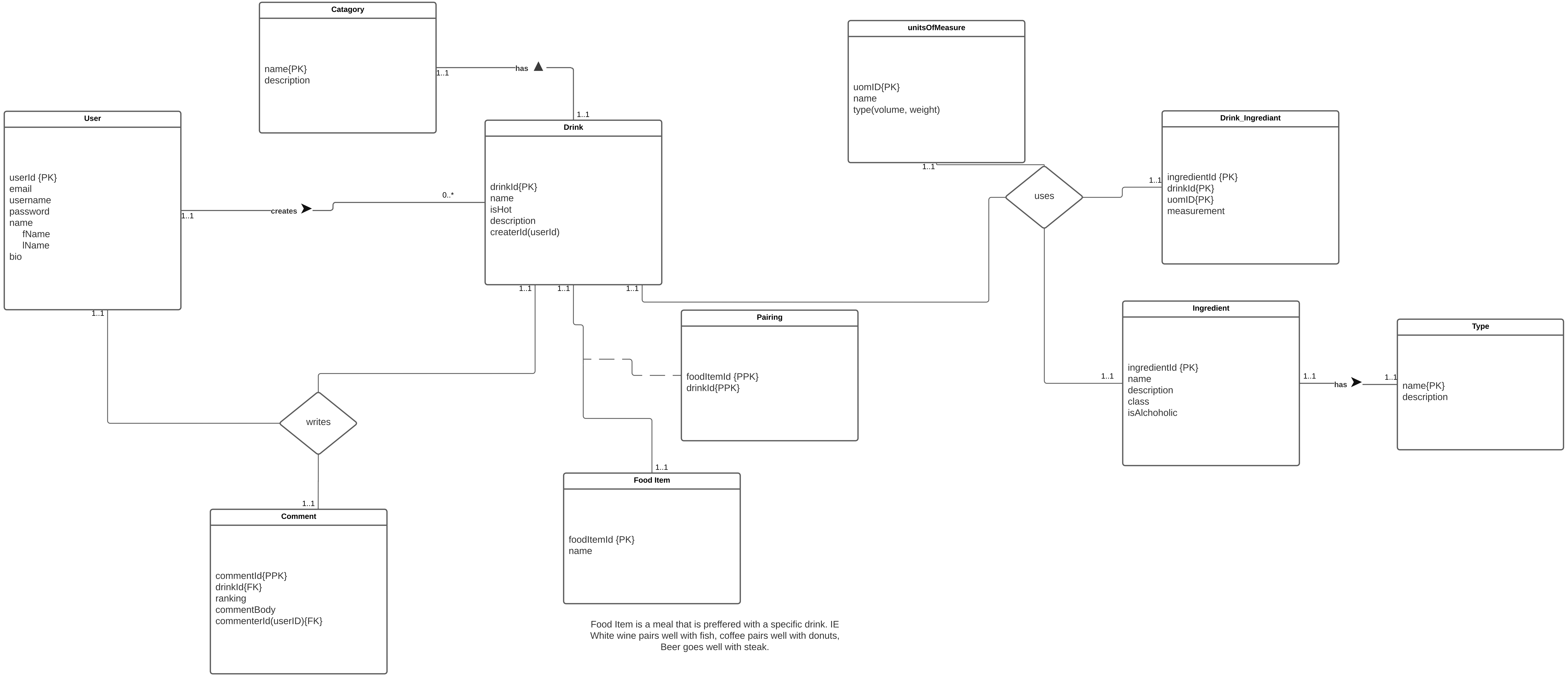
Future Work

Backend

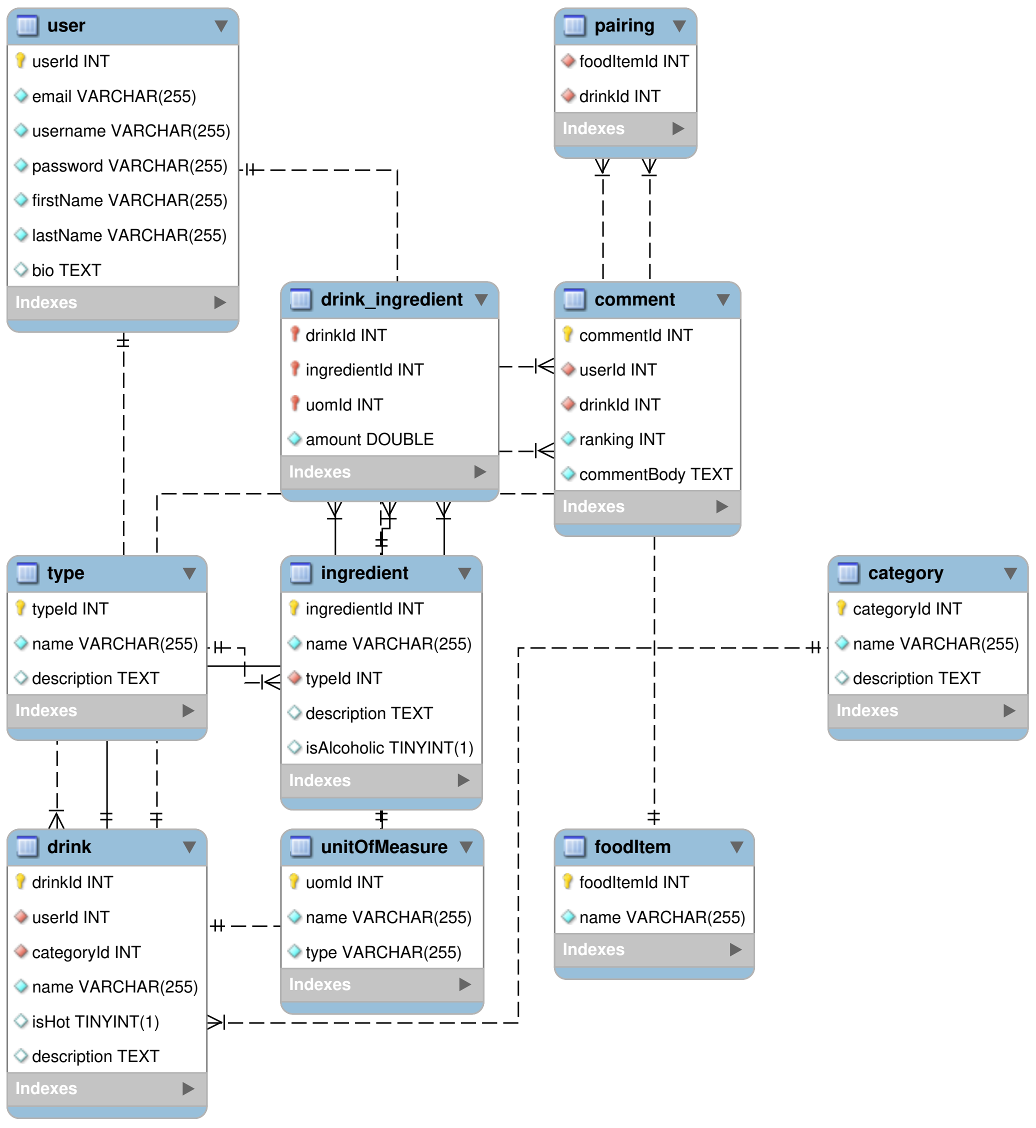
- easier option changes for ports for the backend server and mysql port selection
- better transaction management: right now, I'm not implementing much custom transaction management and letter spring use its default setting. This is definitely and issue as some transactions make additional calls to the database (like saving a drink) and I want to make sure all these calls to the db
- better api calls: the api calls aren't super consistent. I'm learning, so I'm making some rookie mistakes and need to understand better, more consistent ways to make the calls
- better object representation: this relates to the previous point, as the backend currently has DTO and regular objects. But, this can be refactored to use one object.
- validation: There's currently ***NO*** backend validation 😃
- error handing: I mean, the server sends the right status code, but I'm not sure the user wants a full stack dump of the error in the body of the response when one occurs 😃

Frontend

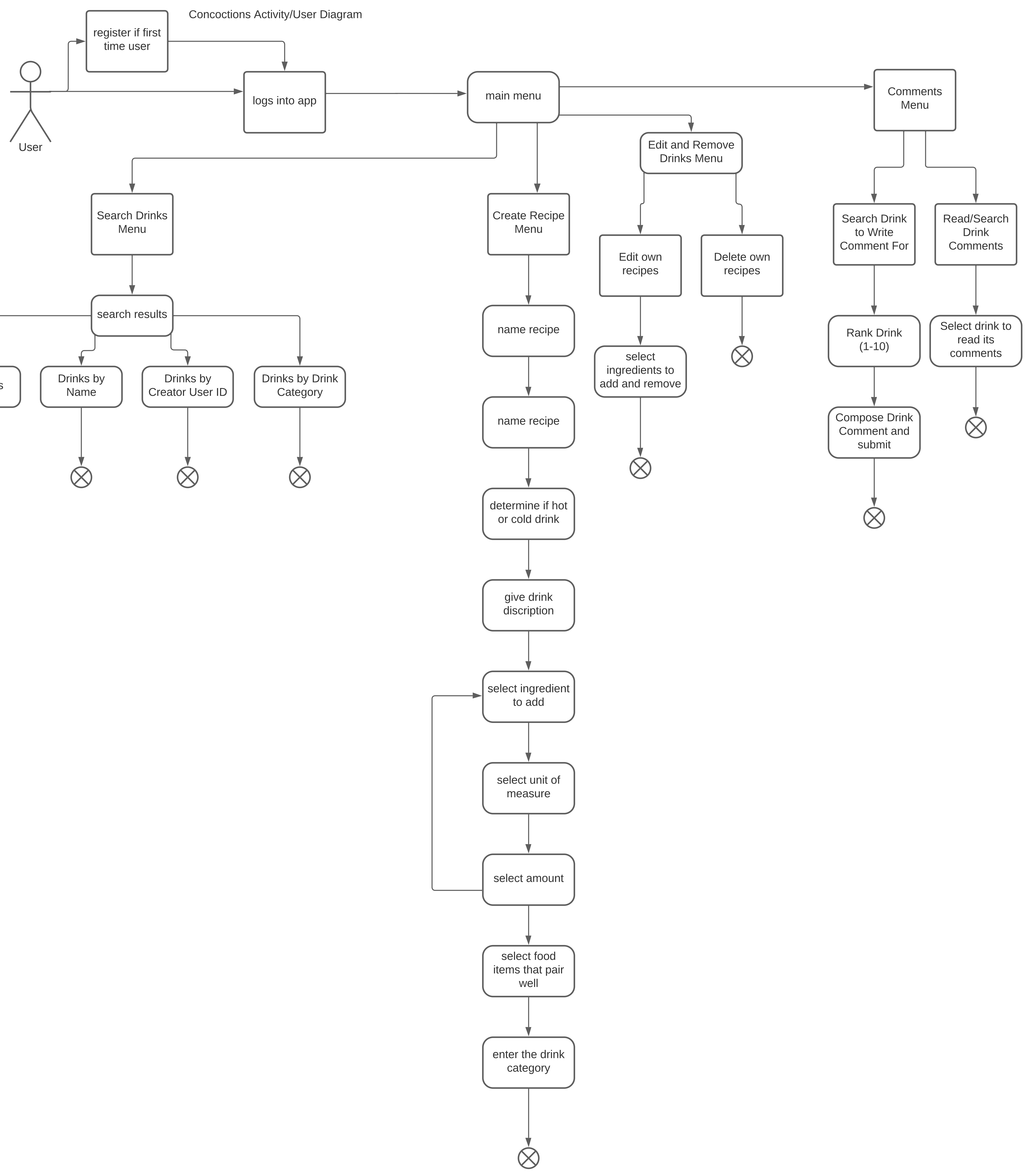
- better code management, especially the menus: I had a lot of issue with passing things back and forth at first in the controller, this caused me to write a lot of the code in the controller itself which should really be in other classes.
- need to expand the ability to search by all parameters not just the ones specified in the project spec
- did not get to the functionality of being able to copy another user's recipe, edit it and submit it as a new one



Food Item is a meal that is preffered with a specific drink. IE White wine pairs well with fish, coffee pairs well with donuts, Beer goes well with steak.


user
userId INT
email VARCHAR(255)
username VARCHAR(255)
password VARCHAR(255)
firstName VARCHAR(255)
lastName VARCHAR(255)
bio TEXT
Indexes
pairing
foodItemId INT
drinkId INT
Indexes
drink_ingredient
drinkId INT
ingredientId INT
uomId INT
amount DOUBLE
Indexes
comment
commentId INT
userId INT
drinkId INT
ranking INT
commentBody TEXT
Indexes
type
typeId INT
name VARCHAR(255)
description TEXT
Indexes
ingredient
ingredientId INT
name VARCHAR(255)
typeId INT
description TEXT
isAlcoholic TINYINT(1)
Indexes
category
categoryId INT
name VARCHAR(255)
description TEXT
Indexes
drink
drinkId INT
userId INT
categoryId INT
name VARCHAR(255)
isHot TINYINT(1)
description TEXT
Indexes
unitOfMeasure
uomId INT
name VARCHAR(255)
type VARCHAR(255)
Indexes
foodItem
foodItemId INT
name VARCHAR(255)
Indexes

Concoctions Activity/User Diagram


User
register if first time user
logs into app
main menu
Comments Menu
Edit and Remove Drinks Menu
Search Drinks Menu
Create Recipe Menu
Edit own recipes
Delete own recipes
Search Drink to Write Comment For
Read/Search Drink Comments
search results
name recipe
select ingredients to add and remove
Rank Drink (1-10)
Select drink to read its comments
All Drinks
Drinks by Name
Drinks by Creator User ID
Drinks by Drink Category
name recipe
Compose Drink Comment and submit
determine if hot or cold drink
give drink discription
select ingredient to add
select unit of measure
select amount
select food items that pair well
enter the drink category