University of British Columbia, Department of Computer Science

CPSC 304

Cover Page for Project Part "Project Deliverables"

Date: November 18, 2018

Group Members:

Name	Student Number	CS Userid	Tutorial Section	Email Address
Sophia Shen	14747159	q8s0b	T1E	sophiashenziyi@gmail.com
Owen Tsai	26515155	f6c1b	T1G	tsaiyicheng3@gmail.com
Charlotte Zhu	53587151	r3s0b	T1A	coco99166@outlook.com
Brandon Djokic	26172056	t7s6	T1A	btdjokic@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 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

Zootopia

Zootopia is a zoo database management system. The project is composed of a sample database and a graphic user interface that allows the zoo manager and the employees to carry out maintenance and management of the zoo database.

The three groups of users for this application are the zoo managers, the animal keepers and the animal trainers. Each group of the users has its own class (ZooMananger, Keeper, and Trainer) and each class contains methods specific to the members of each group, so they can carry out their tasks. Moreover, each class has a JDBC Driver so they can interact with the Oracle database (which we implement in the Driver class).

To allow for access and connection to the database, we first set up the **Driver** class. Package java.sql is used. The constructor of the Driver loads the JDBC driver, connects to the database and populates the tables. executeAlter allows us to execute update, insert and delete statements and return a string "1" if it is successful. executeQuery allows us to execute query statements and returns a result set. Lastly, the disconnect method disconnects the user from the database.

Methods of ZooManager Class

1. **addEmployee** (Deliverable 2)
-when an employee is hired, the manager will assign him/her an employee ID and a walkeetalkee number, and his/her information will be added to the database (specifically to the Employee and Employeecommunicatoin tables)
2. **removeEmployee** (Deliverable 3)
-when an employee quits (or is let go), the manager will remove him/her from the database. Tables that contain this employee information will need to be modified as well. Here, in order to remove an employee, we have to check if he/she is a keeper or a trainer. If yes, we have to first delete his/her relationship with the animal that he/she cared for or trained (in the Caresfor and Trains tables). We can then delete his/her from the Keeper or Trainer table. Afterward, he/she will be deleted from the Employee table followed by removal from the Employeecommunication table.
3. **updateEmployee** (Deliverable 1 and 4)
-when an employee changes his/her name or gets a raise, the manager will have to update his/her information. The arguments are first name, last name, employee ID and pay. To carry out a name change update, the first name and last name were first selected from the Employee table based on the employee ID. Since first name and last name are foreign key referring to the Employeecommunication table, we have to first update the information in that table first. We do so by first saving the walkeetalkee number and insert a new tuple with the new first name, last name and the saved walktalkee number. The Employee table is then updated with the new information and the old information on the Employeecommunication can be deleted. Pay

change is a simple update query that takes place after the name change (or on its own if the name remains the same)

4. **searchEmployee** (Deliverable 8 - 10)

-this is a simple query that allows the manager to look up employee information based of the employees' first name and last name, which are the arguments for this method (note that there can be multiple employees with the same name)

5. **getVolunteer** (Deliverable 8 - 10)

-this method allows the manager to search up on all the volunteer information (volunteer has pay=0)

6. **animalIsInHabitat** (Deliverable 6)

-this method allows the manager to look up the animal in a particular habitat (enclosure ID is the argument). Only the animal ID, names, and species will be returned from this query.

7. **findHabitat** (Deliverable 8 - 10)

-this methods allows the manager to search a particular habitat based on the size (in terms of square and depth) and the biome.

8. **searchByTemperature** (Deliverable 8 - 10)

-this method allows the manager to search a particular habitat based on the temperature of that habitat. All the information regarding that habitat will be returned, which includes the site ID, location, biome, humidity, etc. This involves joining three tables: Site, Habitatbiome, and Habitat.

9. **siteNotUsed** (Deliverable 8 - 10)

-this method allows the manager to look up on a site that is currently not used for any purpose (not housing animal animal or storing food). The used_for will be null in the Site table if the site is not used. This query will return the site ID, the location and the zoo address.

10. **showAt** (Deliverable 11)

-this method creates a virtual table that displays information regarding the animal shows. The start time, duration of the show, show name, show type and show location will be displayed. This query joins the Site and Show tables.

11. **addAnimal** (Deliverable 2)

-when a new animal is introduced into the zoo or an animal is born, it has to be added to the Animal table. The new animal will be assigned an animal ID for identification.

12. **deleteAnimal** (Deliverable 3)

-when an animal passes, it has to be removed from the Animal table. To remove an animal, we must first remove it from all the relationships it is involved in (Caresfore, Trains, Performs, Eats, and Trades). Due to the constraint (every animal must at least have one keeper), the animal has be removed from the Caresfor table. If the animal removed is in the Trains, Performs, Eats and Trades table, then they must be removed too. After deleting the Animal from these tables, we can then delete animal from the Animal table.

13. **updateAnimal** (Deliverable 1 and 4)

-animal information must be updated constantly. For example, an animal ages, gains/loses weight, grows taller, etc. The manager are able to update any attribute of an animal. Therefore, this method takes in all the attributes as arguments and update the animal based on the animal ID (ID should not change).

14. **groupAnimalBySpecies** (Deliverable 7)
-this method allows the manager to display all animals in the order of their species. The query will return the animal ID, name, age, sex and species
15. **getSpeciesDetails** (Deliverable 8 - 10)
-this method allows the manager to look up on animals of a particular species. The query will return the animal ID, name, weight and height. Further analysis, such as the average weight of a particular, can then be done manual by the manager.
16. **getFoodDetails** (Deliverable 8 - 10)
-this method allows the manager to look up information on all the animal food in the zoo. The query will returned the food storage site ID and location along with all the attributes of Food.
17. **getFoodSoonExpires** (Deliverable 8 - 10)
-this method allows the manager to look up on all the soon expired food ordered by the expiry date. The query will return food ID, name and date of expiry.
18. **filterZooByCountry** (Deliverable 8 - 10)
-the zoo might be a franchise, so this method allows the manager to look up on all the information of a zoo in a particular country.

Methods of Keeper Class
19. **getKeeperView** (Deliverable 5)
-This method displays the duties of keepers and all the relevant information the keeper needs to complete his/her duties. A keeper has to know what animal that he/she is taking care of, the food the animal eats and the locations of the animal and food storage. This query joins 7 tables: Keeper, Caresfor, Animal, Habitat, Eats, Food, and Site.
20. **updateAnimal** (Deliverable 1 and 4)
-Part of the keeper job is to measure the animal weight and height when he/she cares for them. Animal might gain/lose weight and grow taller, so the keeper has to update this information on the Animal table. Note that keepers can only update these two attributes, whereas manager can update all the attributes of an animal.

Method of Trainer Class
21. **getTrainerView** (Deliverable 5)
-Similar to the getKeeperView method in the Keeper Class, this method displays the duties of trainers and all the relevant information he/she needs to complete his/her tasks. A trainer trains animals for show and he/she will also be performing with the animal, so a trainer needs to know what animal he/she is training and the information of the show he/she will be performing. This query joins 6 tables: Trainer, Trains, Animal, Performs, Show and Site.

GUI
LoginPage
-This is the first page that the user will see when opening the app. The GUI will ask the user to provide user ID(employee has to enter employee ID and manager has to enter the password). The GUI will determine if the user is a manager, a keeper or a trainer based on the user ID entered. Afterward, the user will be lead to his/her specific page.

MainPage_Manager
-This is the GUI for the manager. It consists of 18 buttons (on the left) representing the methods specific to the Manager class (All the methods are described above). To execute specific method, the user (manager) has to enter the parameter(s) and click the button on the left. This will lead the user to the ResultPage.
manager pw: “abcde”
keeper pw: “1234567890 ”
trainer pw: “1376542234”
more examples can be found in the driver.java file, in method insertData(), employees, 3rd attribute

MainPage_Keeper
-This is the GUI for the keeper and consists of 2 buttons representing the methods specific to the Keeper class. Similar to the manager page, the keeper has to enter parameter(s) for the specific he/she want to carry out and click the button on the left, which will lead the keeper to the ResultPage

MainPage_Trainer
-This is the GUI for the trainer which works as the same manner as the manager and keeper page. There is a button on the left representing the method specific to the Trainer class

ResultPage
-the user will be lead to this page when he/she execute a methods. It consists of a table containing the appropriate columns specified in the method if the method is successfully executed. Otherwise, it will return error or no results.

CHANGES TO ER DIAGRAM:

(Modification) relationship hires:
Explanation: Participation constraint from Employee because Employee needs to be hired in at least one zoo
No participation constraint from Zoo because a newly starting zoo might not have employees other than owner

(Added) relationship trades:
Explanation: Missing trade relationship in previous deliverable - this is so that two zoos can interact with each other and can trade animals

(Added) relationship LivesIn + (Deleted) relationship maintains:
Explanation: our previous relationship maintains was confusing as it directly linked Zoo to Animal, Habitat and Building in one relationship. We broke up the relationship so that animal is directed linked to a habitat through a livesIn relationship. Other parts of the maintains

relationship is explained below.

(Modification) entity Site (previously called Building) + (Added) relationships OccursAt/Locates:
Explanation: again, our previous relationship maintains was confusing as it directly linked Zoo to Animal, Habitat and Building in one relationship. Food had a storedIn relationship with a building. We modified our ER diagram so that Zoo has multiple Sites, and each Site can be in a relationship with Food (through StoredIn), Show (through OccursAt) and Habitat (through locates). Instead of Habitat and Show having a field 'location', they have a relationship with a Site, which has a location. We also added a participation constraint with all 3 relationships because Food, Show, and Habitat all have to have one and only one physical location (Site).

(Removed) entity Supplement
Explanation: based on feedback, since supplement entity does not participate in any relationships, it can just be stated in the name field of Food.

(Removed) relationship Keeps
Explanation: Since the relationship Keeps between animal and zoo has no fields, and animals can be linked to zoo through habitat and site, we removed the redundant relationship based on feedback.

CHANGES TO RELATION MODEL:

- Employee was broken down into two tables Employee and EmployeeCommunication (with first name, last name and walkeetalkee #) because the FD (f_name, l_name -> walkeetalkee #) violated 3NF and BCNF
- Habitat was broken down into two tables Habitat and HabitatBiome because the FD (Biome -> Temperature, humidity) violated 3NF and BCNF
- Keeper and Trainer were missing primary key (employee_id) so they were added
- The FD height, weight, species, sex -> eat_freq_week, eat_amount in Animal was removed because we realized that it is not very representative of zoo systems. Even if the animal's height and weight slightly increases, the animal's feeding frequency and amount usually stay constant, unless manually determined by the zoo management staff
- Due to changes in the ER diagram:
  - Supplement table was removed
  - Site table (originally Building) was modified to add FK that references Zoo
  - Habitat, Food, Show were modified to add FK that references Site
  - Trade table was added
- Some M:M relationships were missing and added:
  - Performs, Caresfor, Trains, Eats

CHANGES TO FORMAL SPECIFICATION:

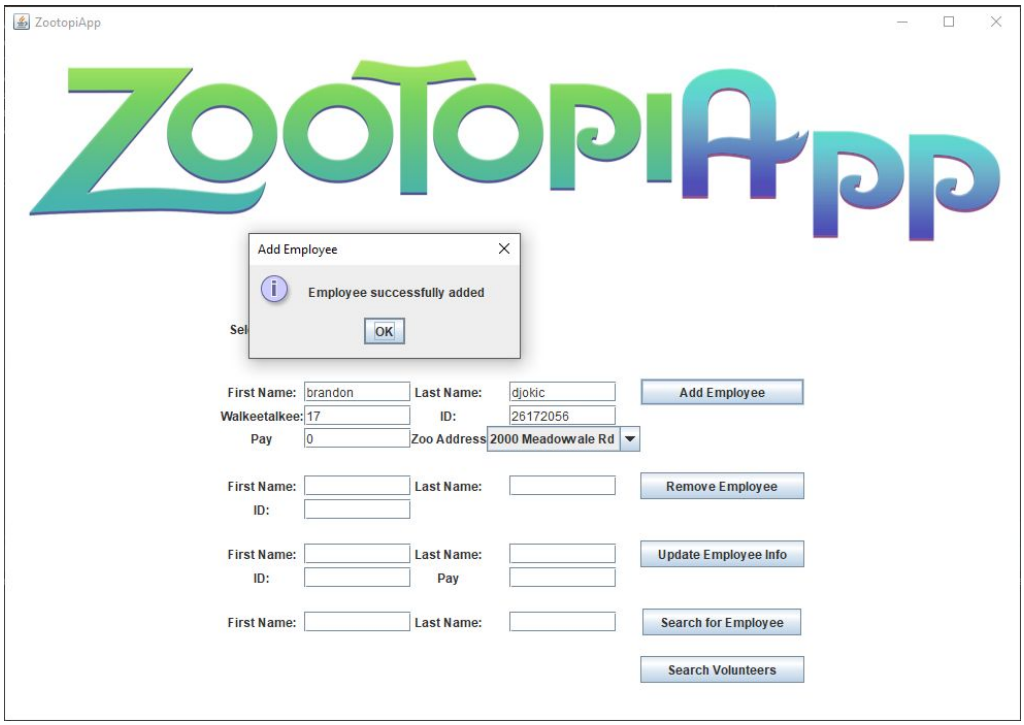
- In deliverable 1, we said that a trainer should be able to modify and update

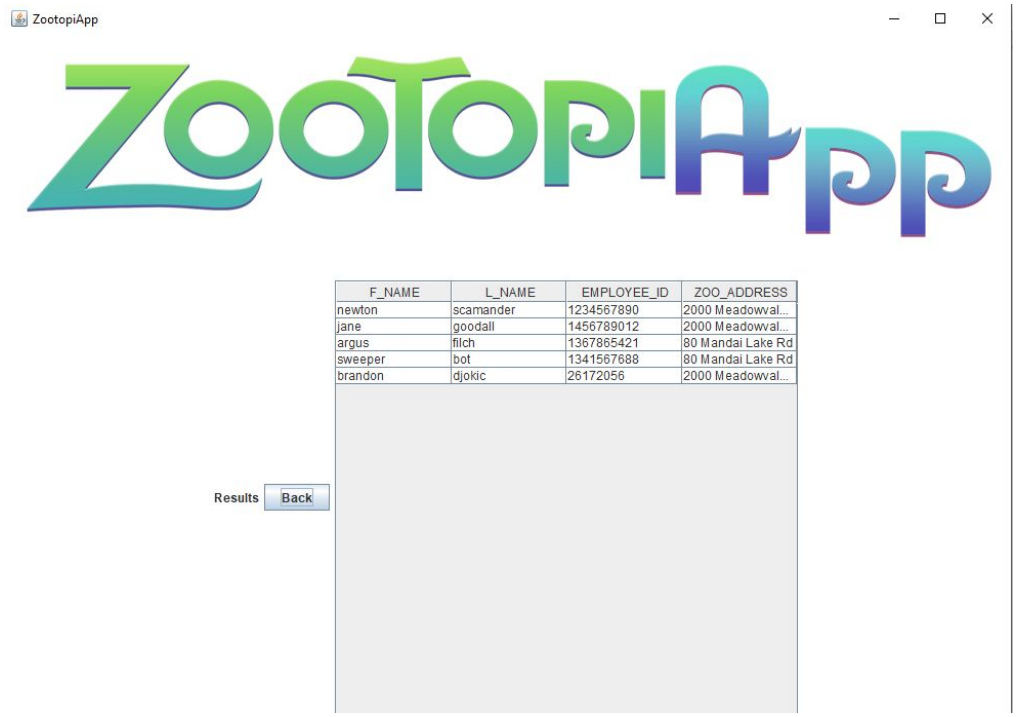
animals/shows/habitats. However, after further discussion, we felt like the trainer should only be able to view his schedule (shows, animals etc) and the described updates should be performed by the manager only
- For deliverable 4, instead of just setting weight, we changed it to updating both weight + height as these are typically measured together by the keeper.
- For Deliverable 10 #6, for a particular species (e.g. lion), instead of just returning the weight, the query also returns the animal's ID, height and name for a more complete description

Unexpected Challenges
We had a bug where one of our group members was having strange resultset behavior. After building tables and populating them, a query returned an "empty" result set. i.e. one where .next() returned false after the first call. We knew for sure this was incorrect behavior because the same code on a different computer would return the expected result. This bug was only present, for the one group member, and we were unable to figure out the cause.

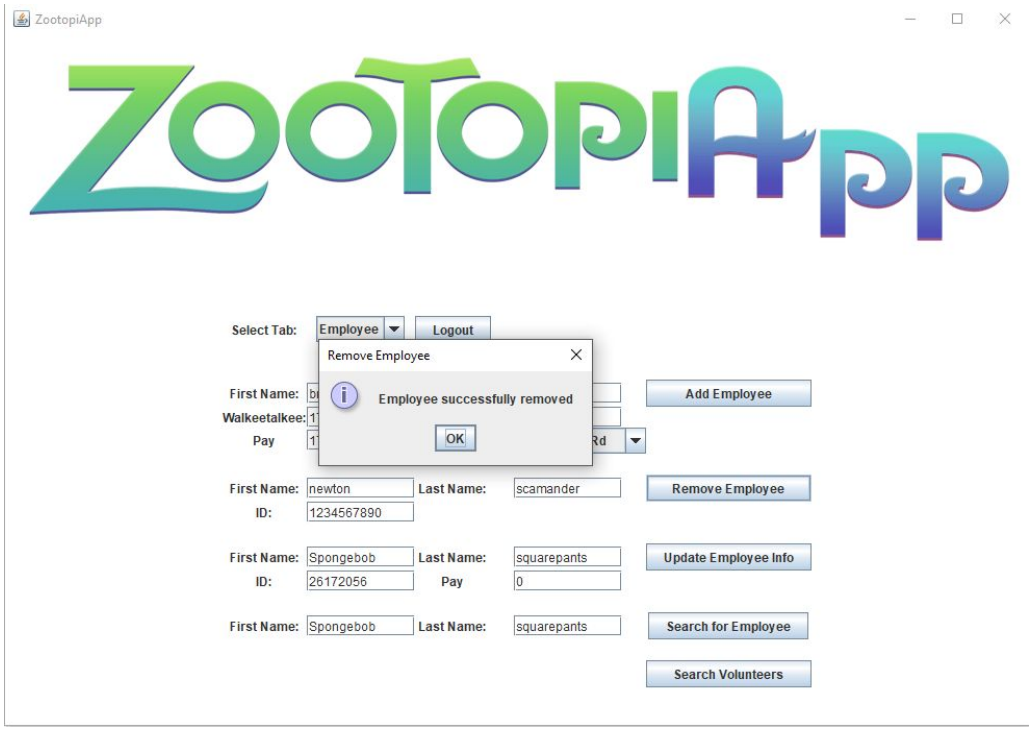
SAMPLE SCREENSHOTS

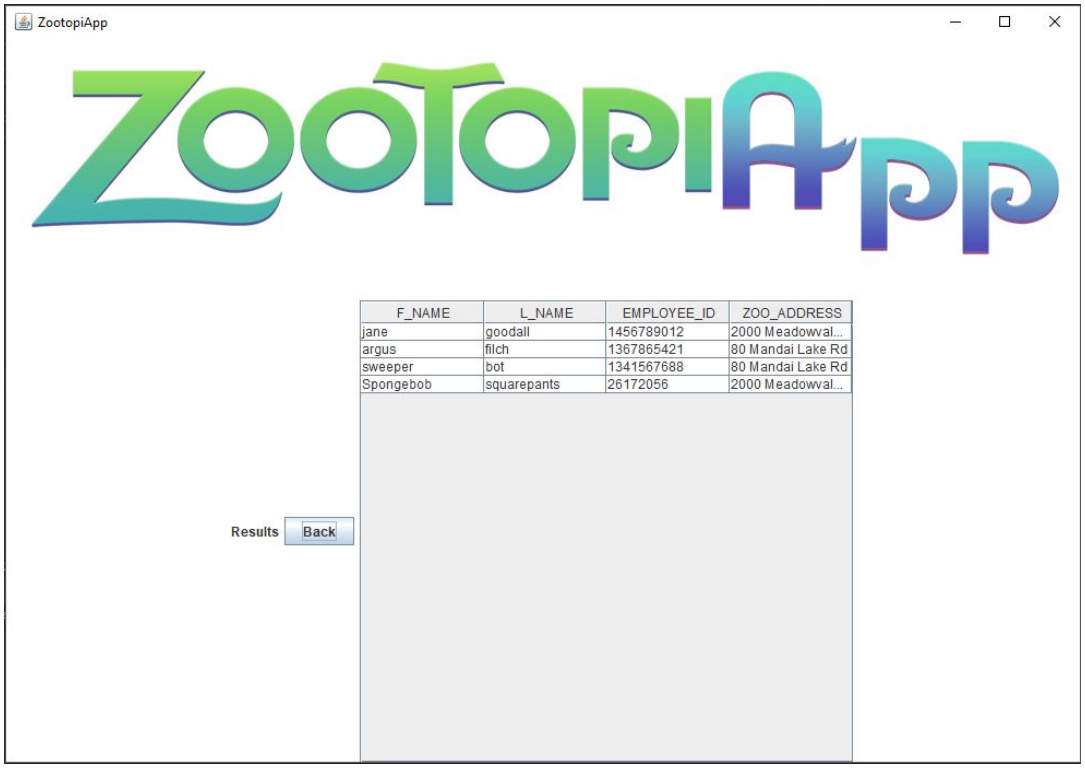
1. **addEmployee** (Deliverable 2)



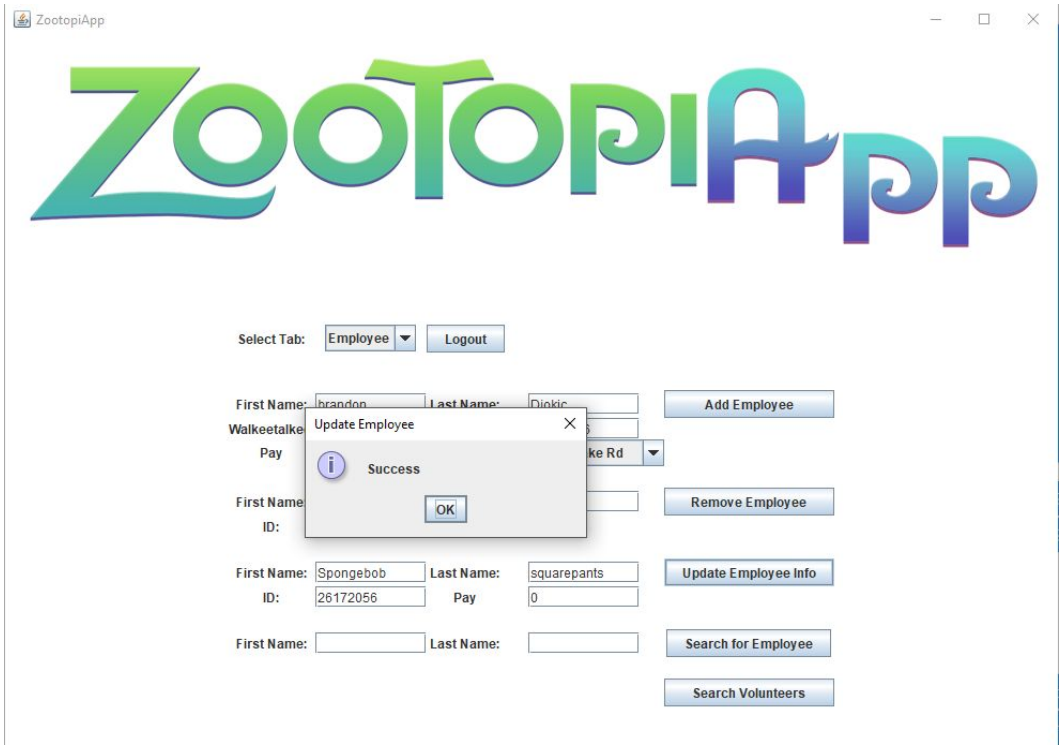


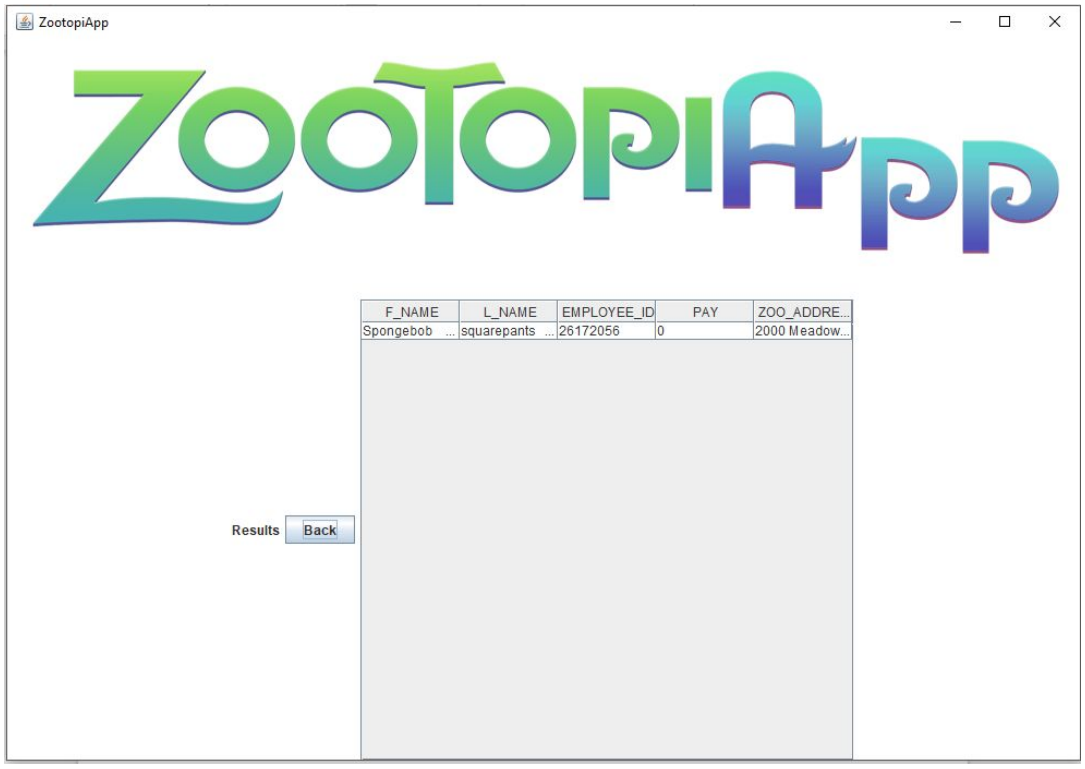
2. **removeEmployee** (Deliverable 3)
Example: remove employee newton scamander

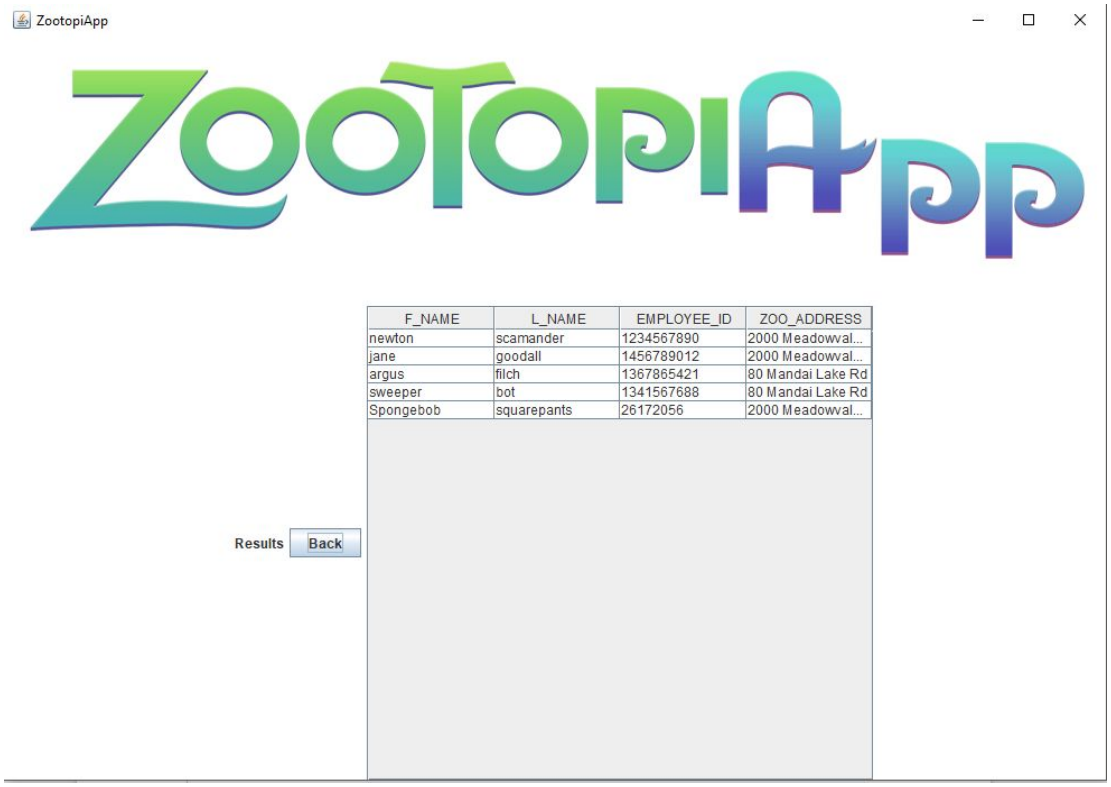




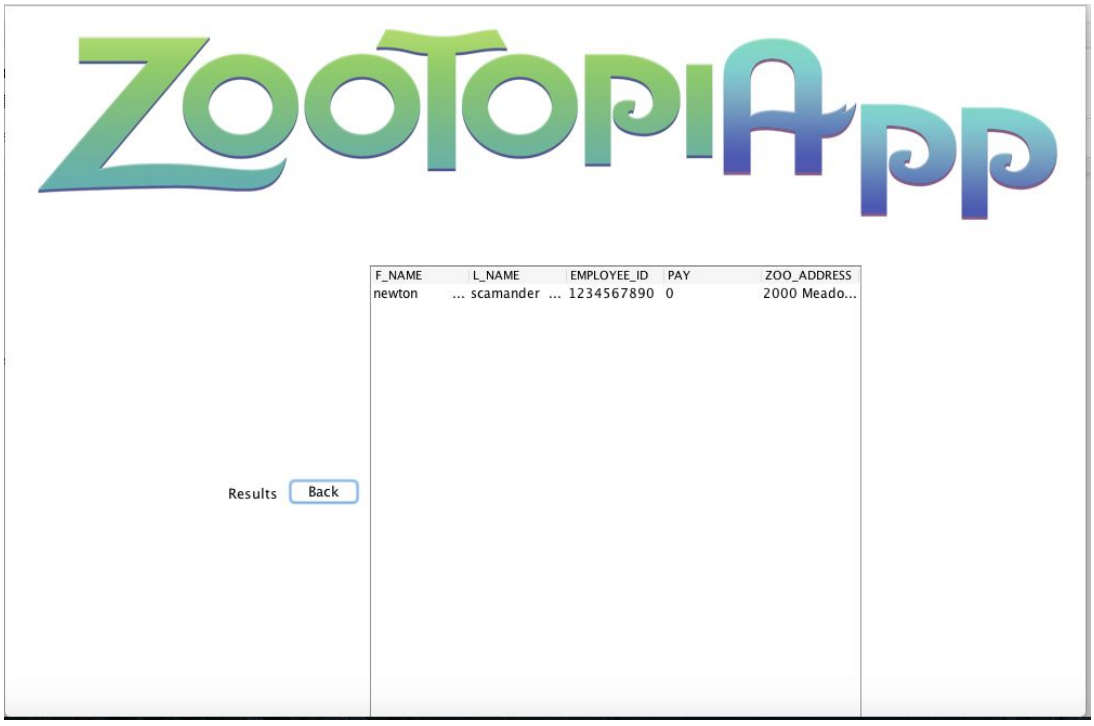
3. **updateEmployee** (Deliverable 1 and 4)
Example: change employee 26172056's name (first and last)


ZootopiApp
Select Tab:
Employee
Logout
First Name:
Last Name:
Add Employee
Update Employee
Success
OK
Pay
First Name
ID:
Remove Employee
First Name:
Spongebob
Last Name:
squarepants
Update Employee Info
ID:
26172056
Pay
0
First Name:
Last Name:
Search for Employee
Search Volunteers

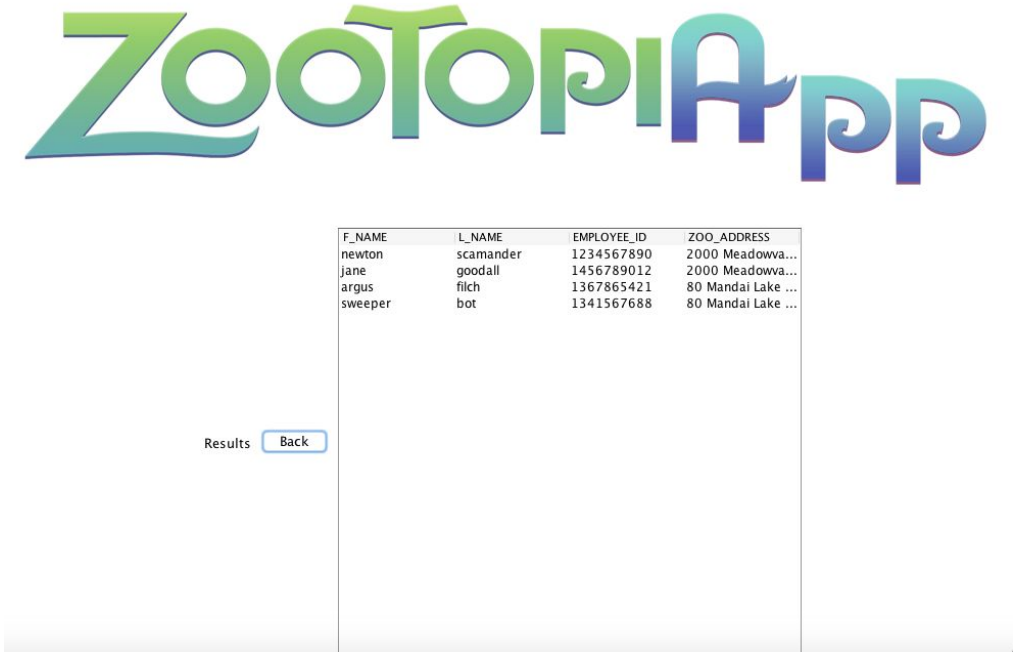

ZootopiApp
F_NAME
L_NAME
EMPLOYEE_ID
PAY
ZOO_ADDRE...
Spongebob
squarepants
26172056
0
2000 Meadow...
Results
Back



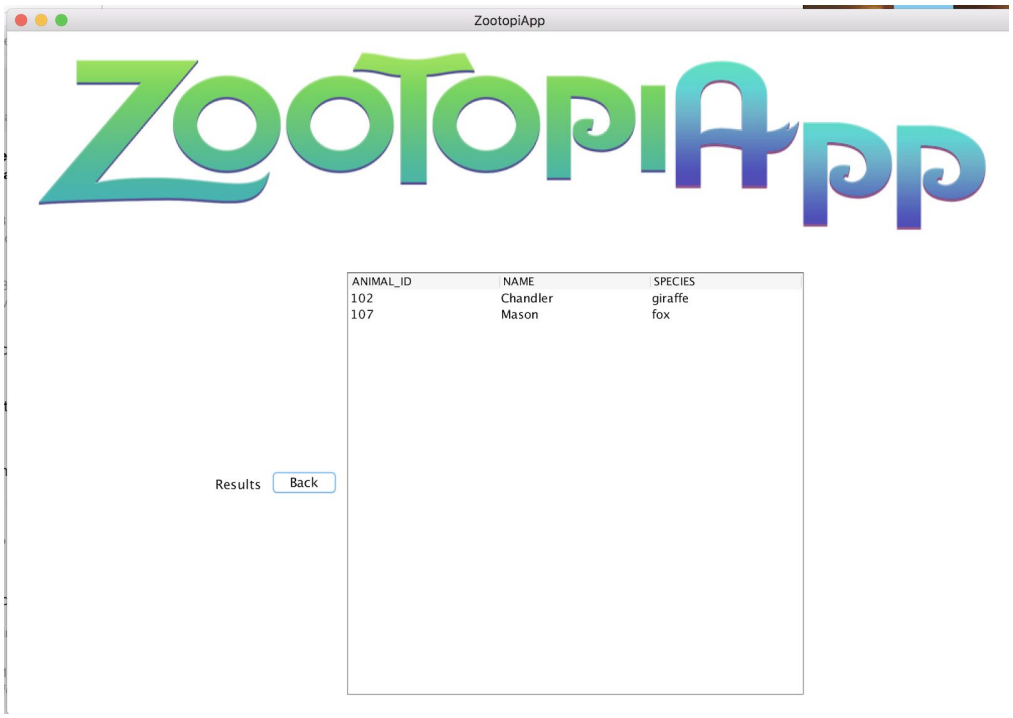
4. **searchEmployee** (Deliverable 8 - 10)
Example: search employ with first name newton and last name scamander



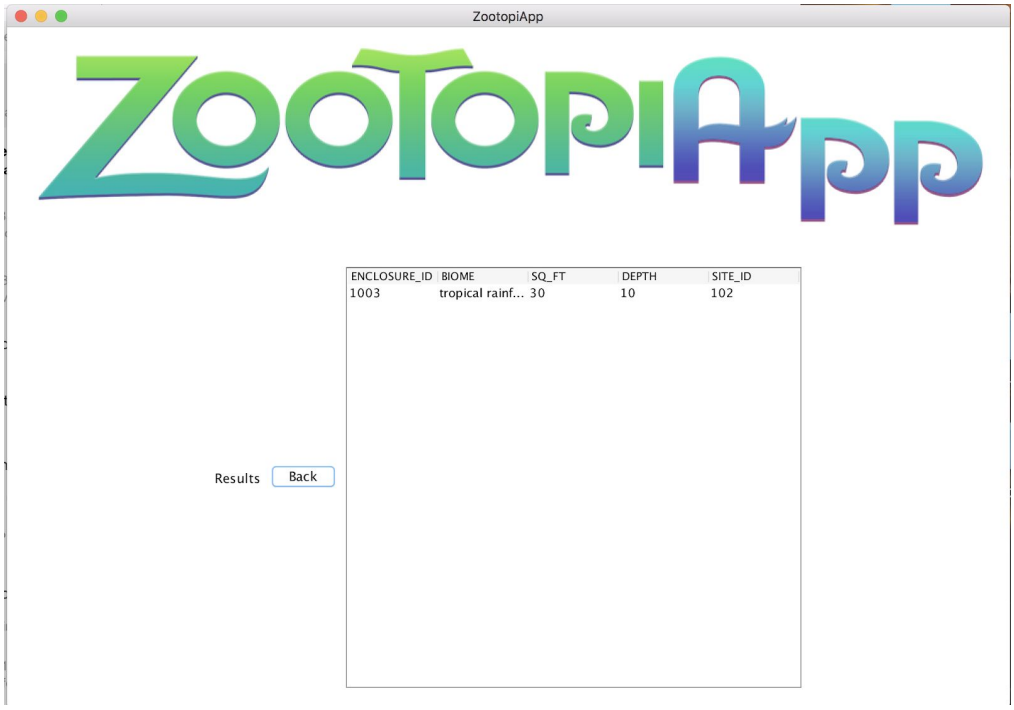
5. **getVolunteer** (Deliverable 8 - 10)



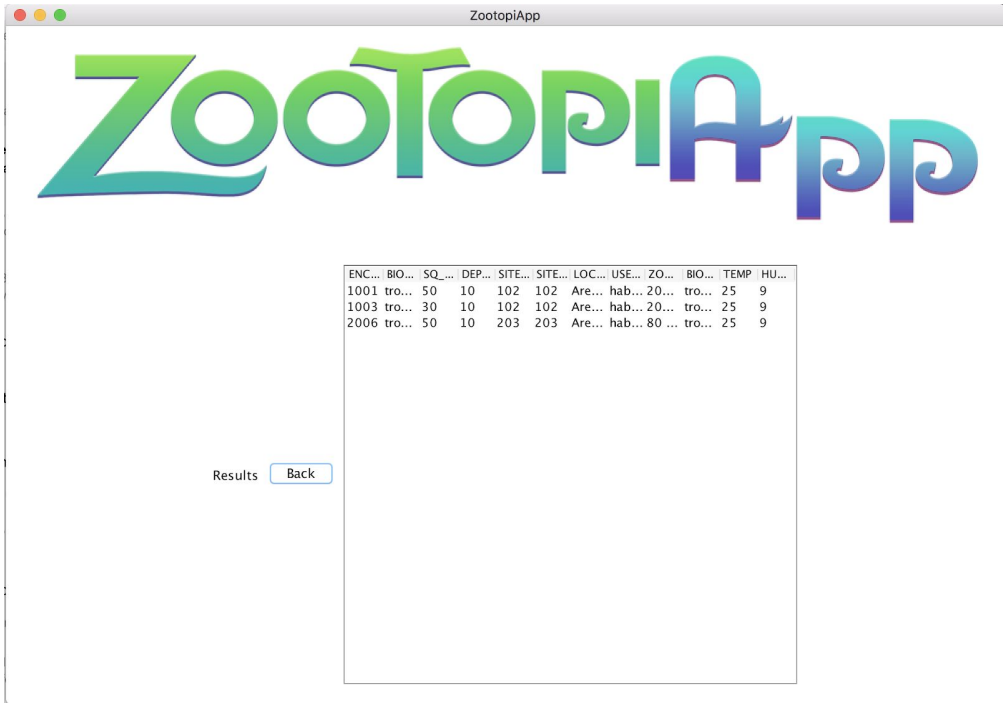
6. **animalsInHabitat** (Deliverable 6)
Example: animals in habitat with ID 1002:



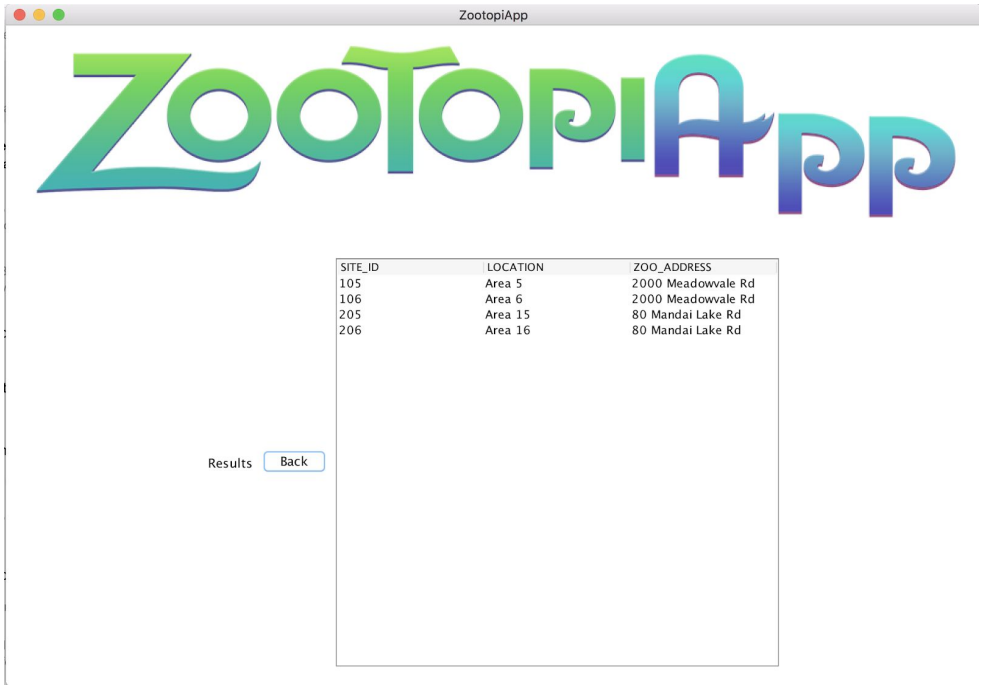
7. **findHabitat** (Deliverable 8 - 10)
Search for habitat with 50 sq feet, depth = 10m and biome = tropical rainforest:



8. **searchByTemperature** (Deliverable 8 - 10)
Example: searched for temperature = 25 degree Celsius:

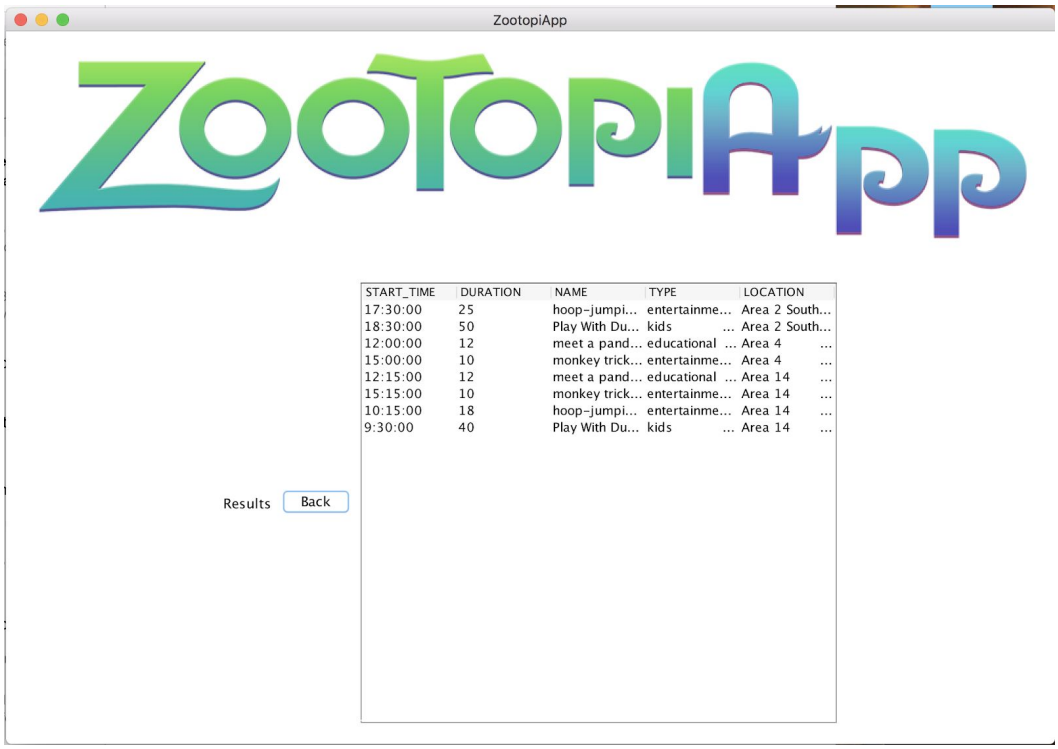


9. **siteNotUsed** (Deliverable 8 - 10)



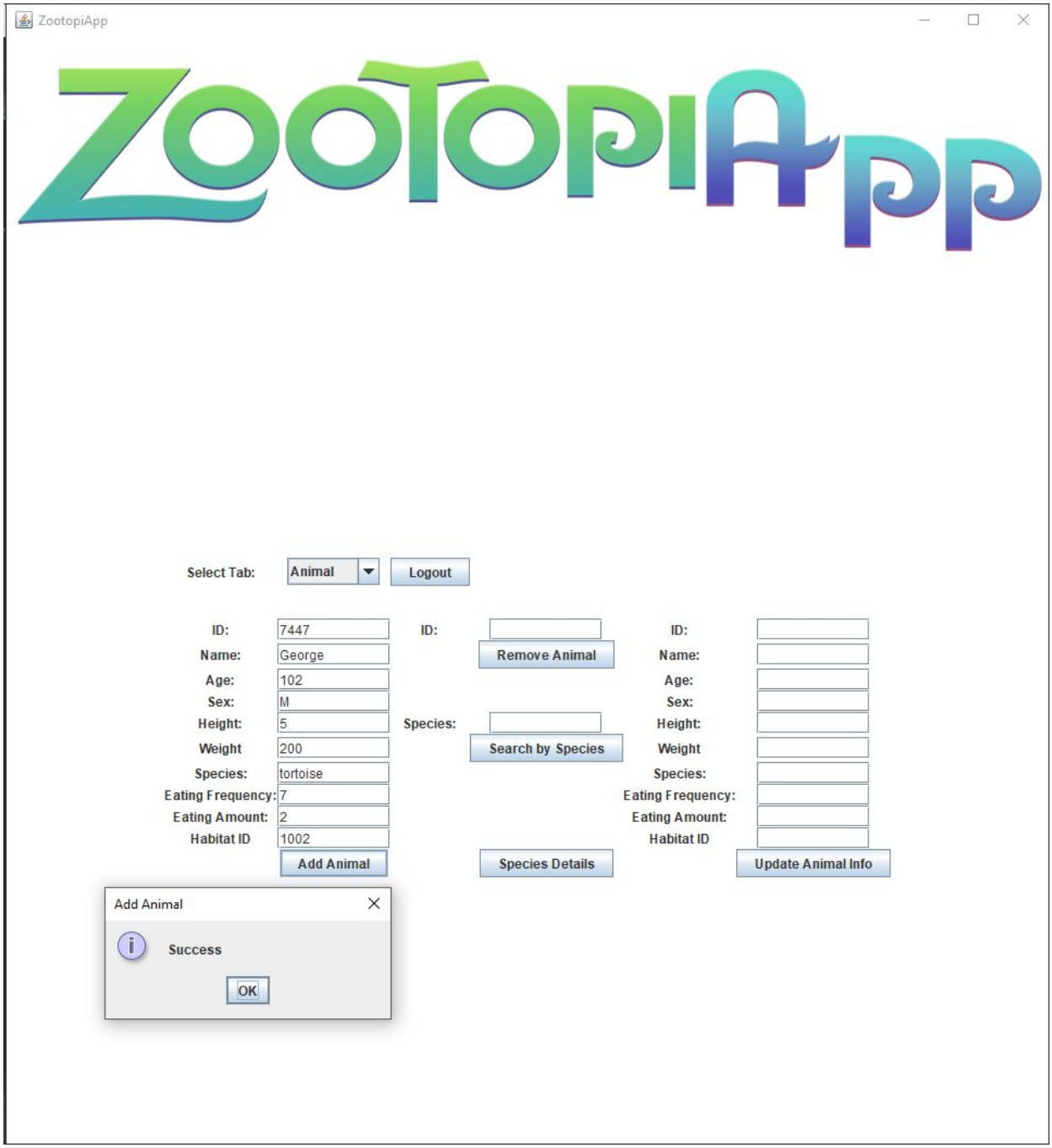
10. **showAt** (Deliverable 11)

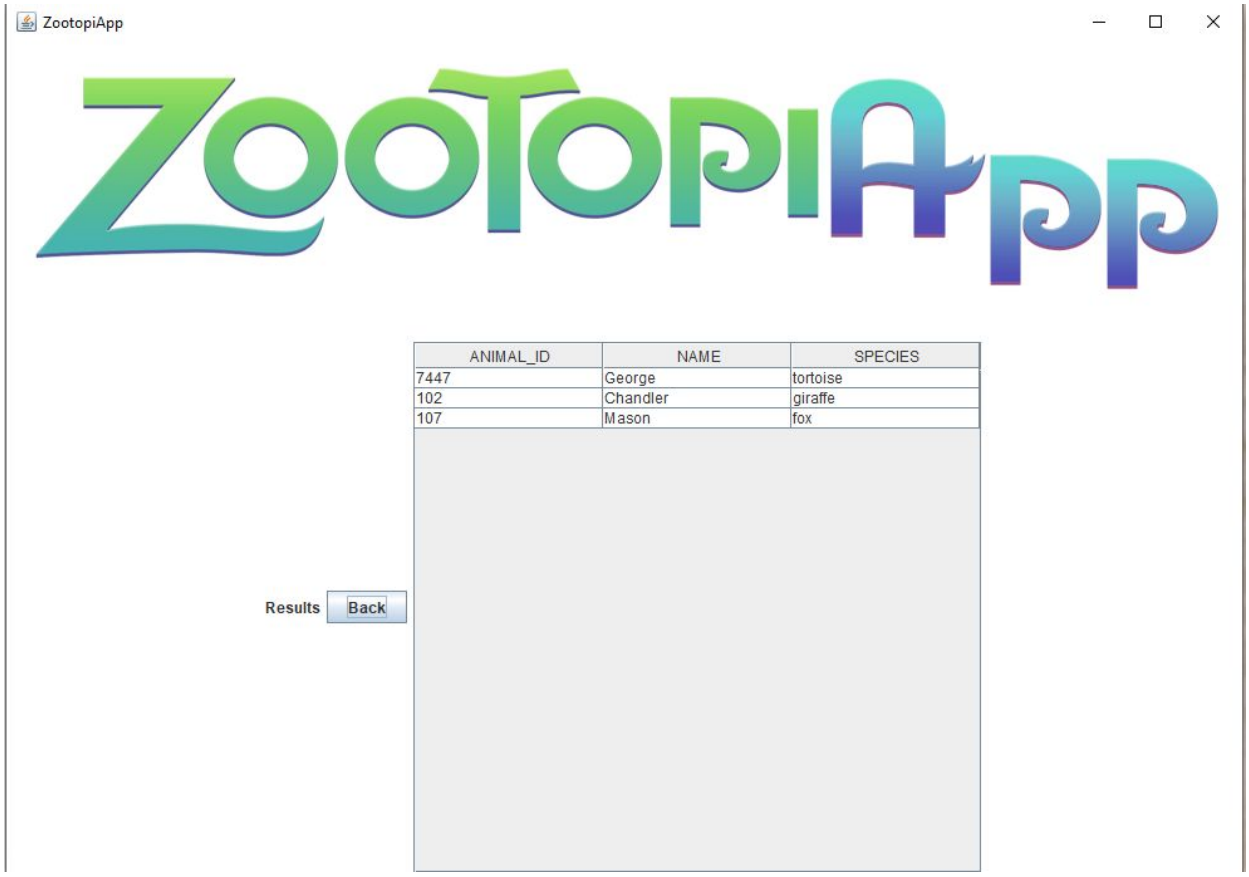
This is the view after pressing button “Showtimes” in the Show page



11. **addAnimal** (Deliverable 2)

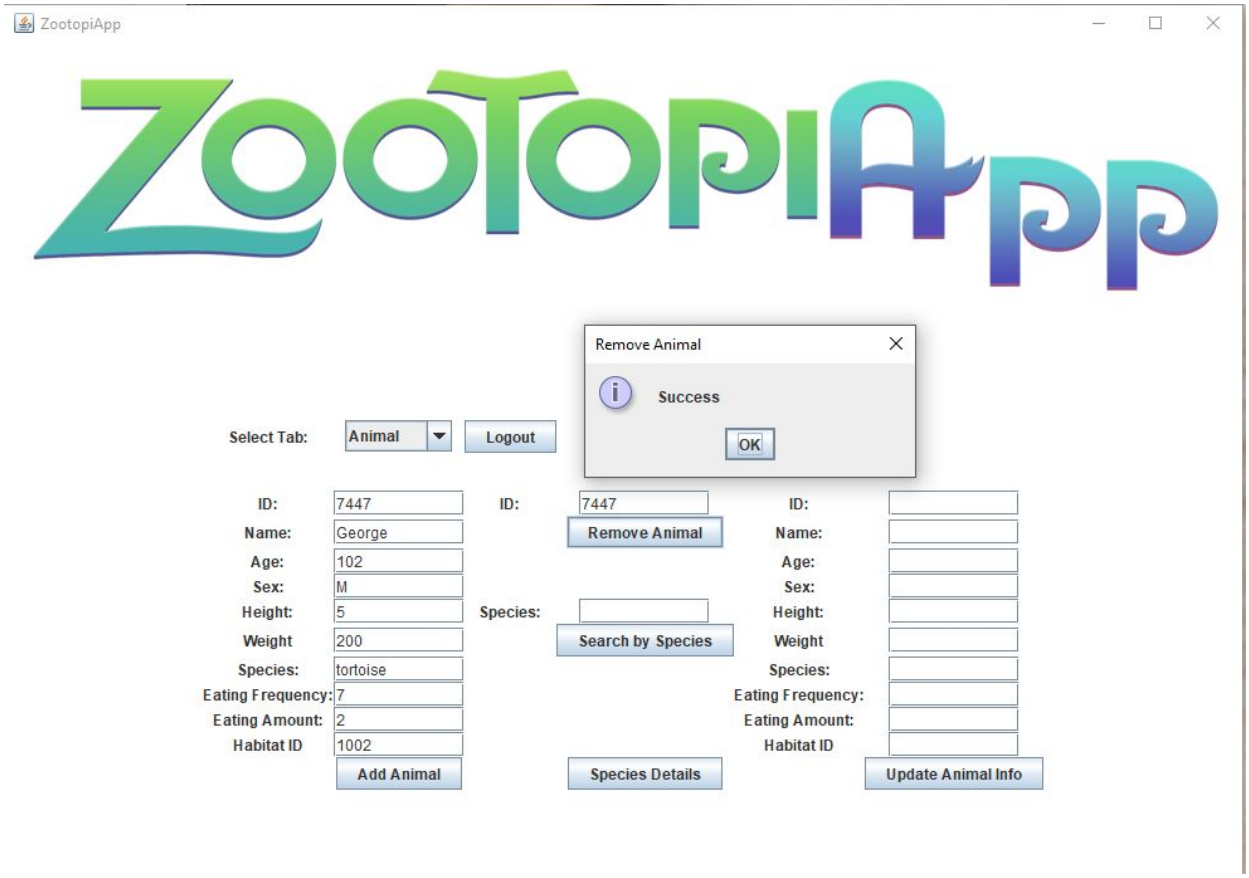
Entering info for tortoise “george”, viewing all animals that he shares a habitat with

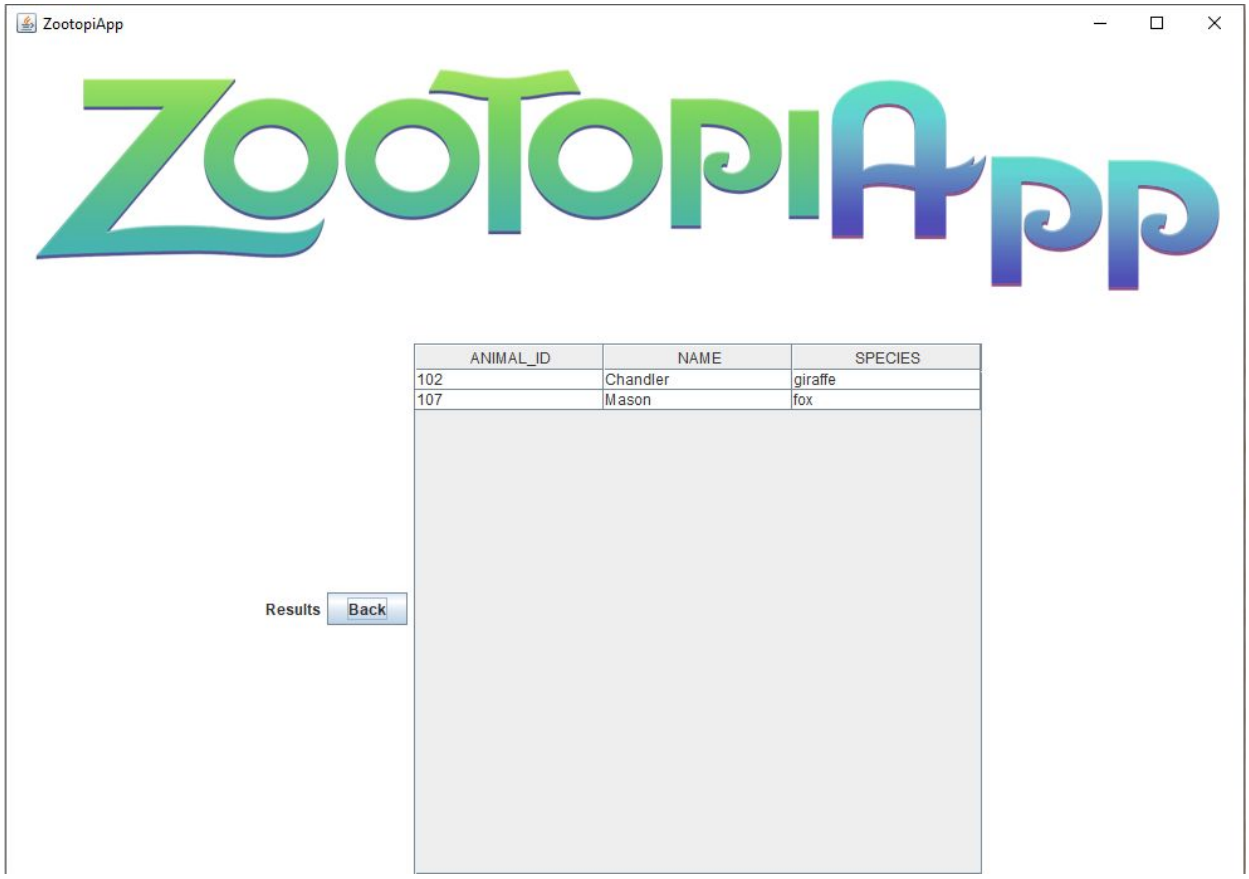

ZootopiApp
ZOOTOPIAPP
Select Tab:
Animal
Logout
ID:
7447
Name:
George
Age:
102
Sex:
M
Height:
5
Weight
200
Species:
tortoise
Eating Frequency:
7
Eating Amount:
2
Habitat ID
1002
Add Animal
ID:
Remove Animal
Species:
Search by Species
Species Details
ID:
Name:
Age:
Sex:
Height:
Weight
Species:
Eating Frequency:
Eating Amount:
Habitat ID
Update Animal Info
Add Animal
Success
OK



12. **deleteAnimal** (Deliverable 3)

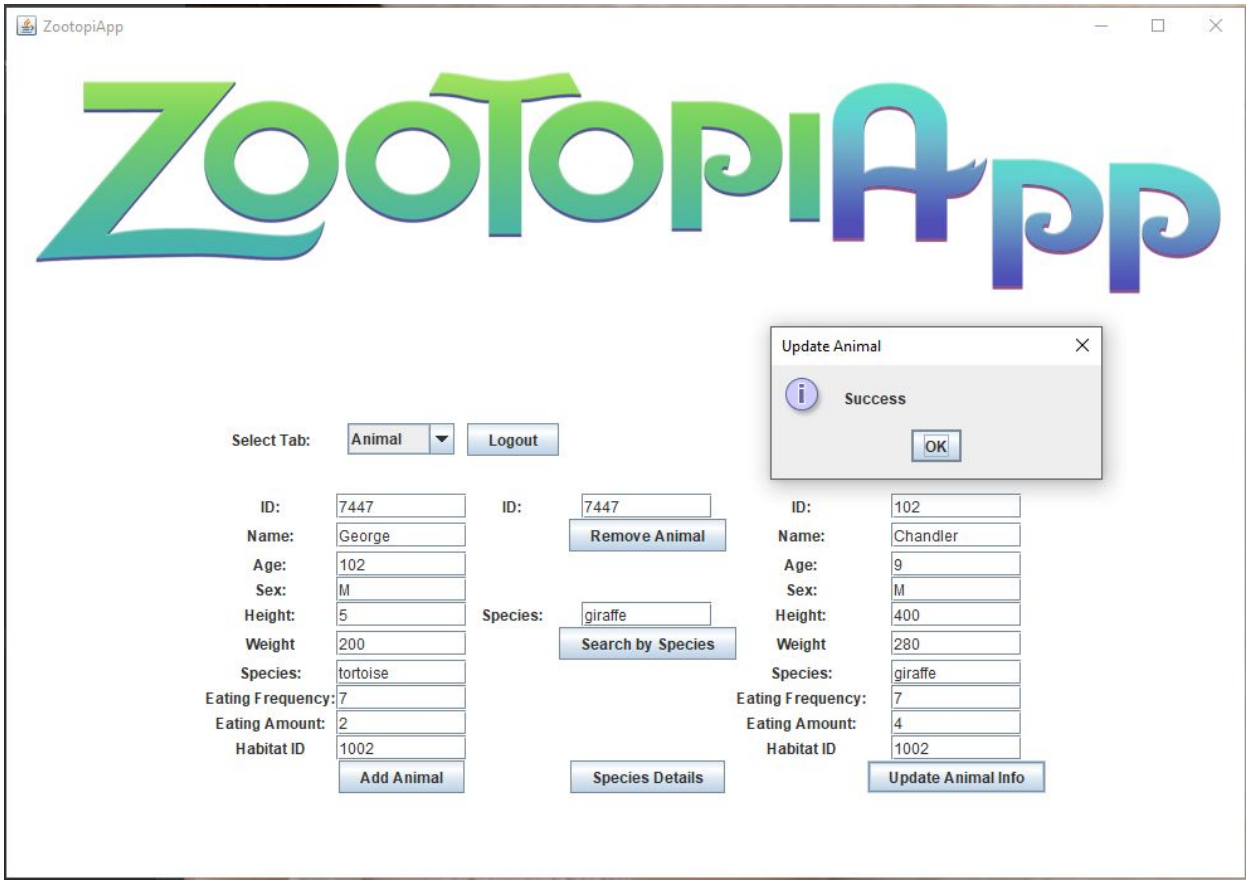
Removing george from the db, and viewing old habitat

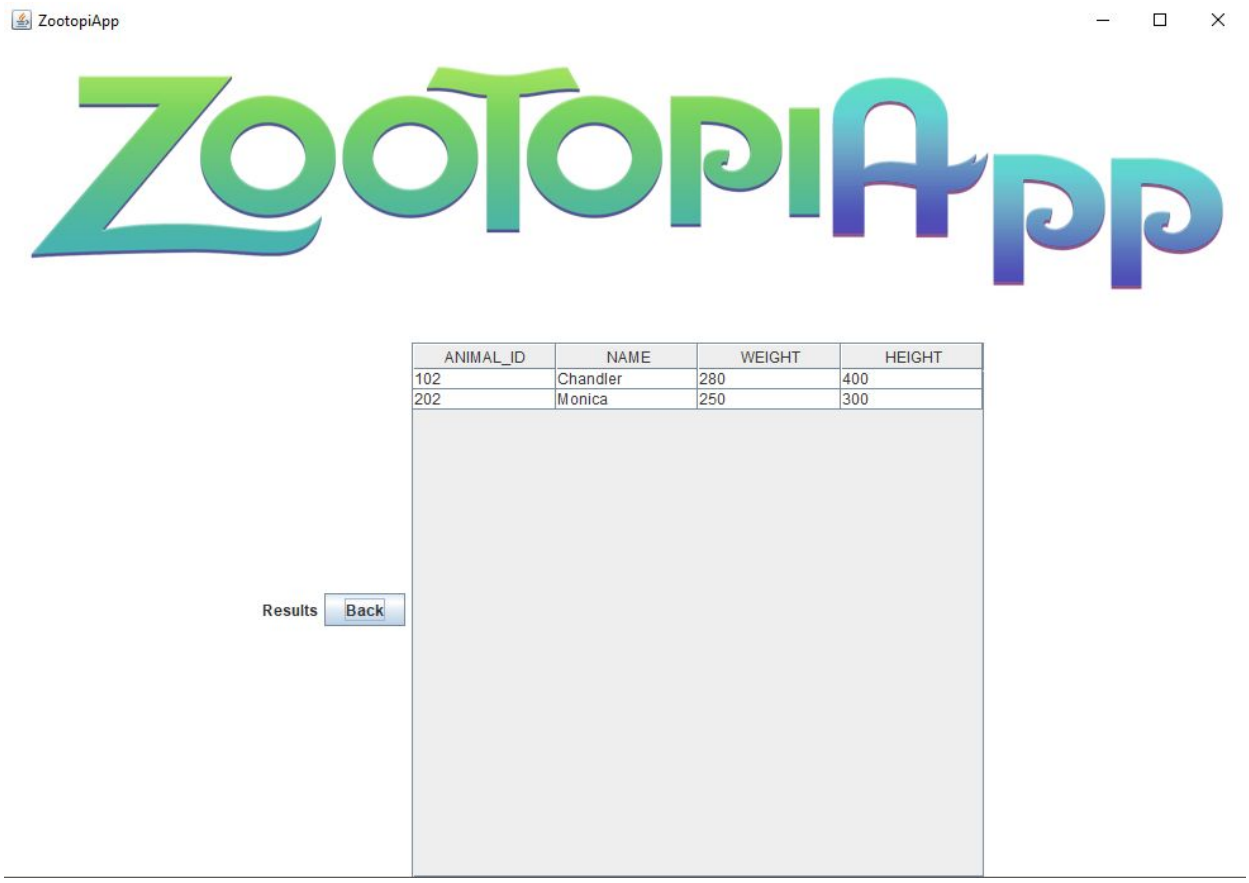

ZootopiApp
ZOOTOPIAPP
Remove Animal
Success
OK
Select Tab:
Animal
Logout
ID:
7447
Name:
George
Age:
102
Sex:
M
Height:
5
Weight
200
Species:
tortoise
Eating Frequency:
7
Eating Amount:
2
Habitat ID
1002
Add Animal
ID:
7447
Remove Animal
Species:
Search by Species
Species Details
ID:
Name:
Age:
Sex:
Height:
Weight
Species:
Eating Frequency:
Eating Amount:
Habitat ID
Update Animal Info



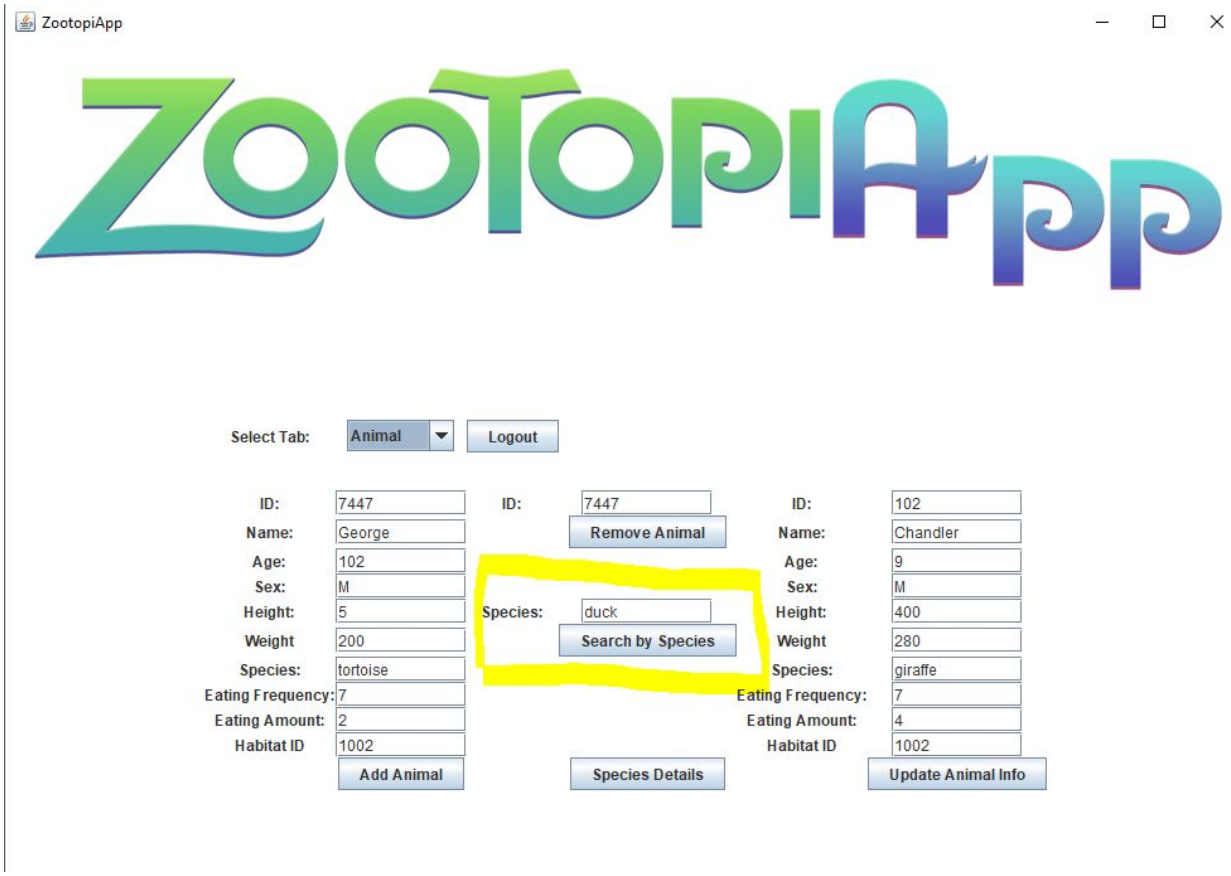
13. **updateAnimal** (Deliverable 1 and 4)
Updating chandler the giraffes weight 300 -> 280, and other attributes, and viewing result


ZootopiApp
ZOOTOPIAPP
ANIMAL_ID
NAME
WEIGHT
HEIGHT
102
Chandler
300
400
202
Monica
250
300
Results
Back

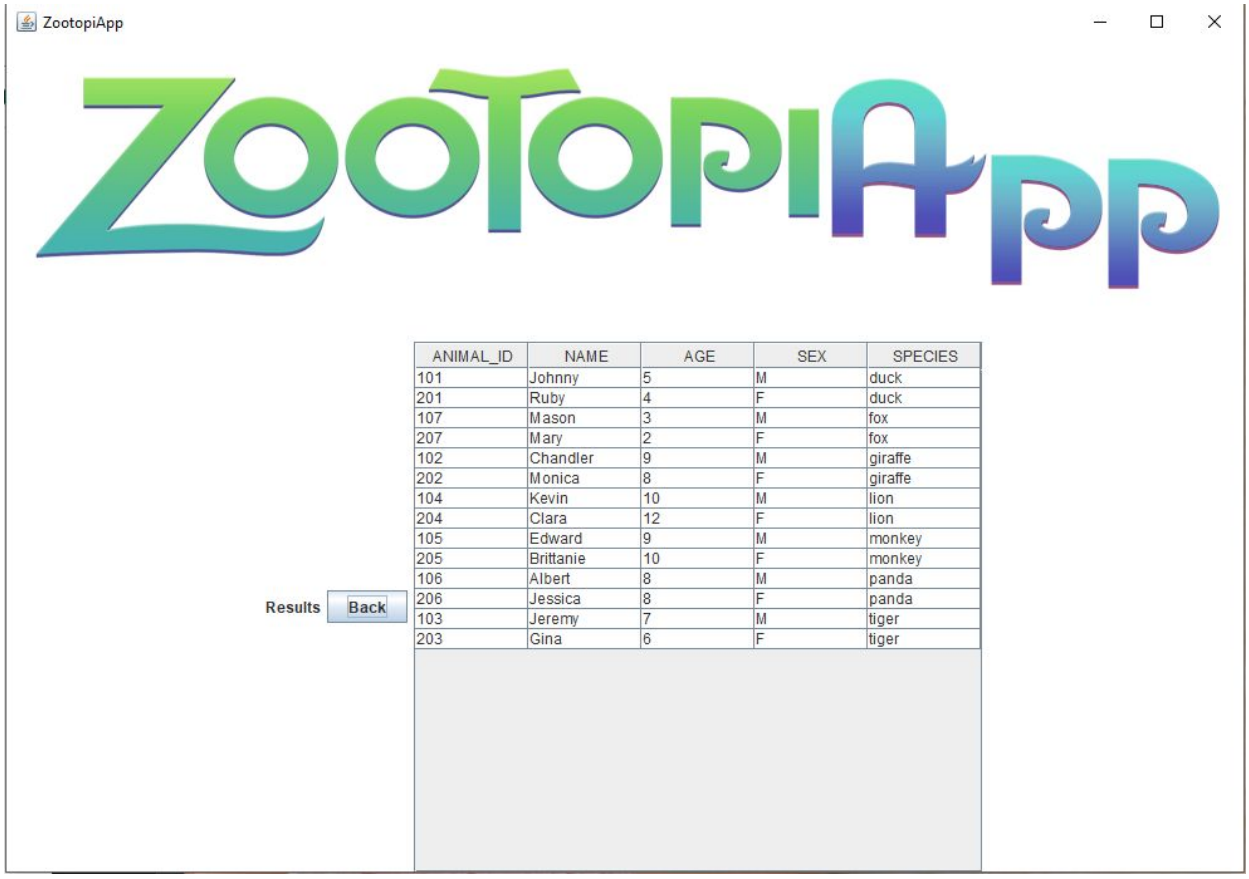

ZootopiApp
ZOOTOPIAPP
Update Animal
Success
OK
Select Tab:
Animal
Logout
ID:
7447
Name:
George
Age:
102
Sex:
M
Height:
5
Weight
200
Species:
tortoise
Eating Frequency:
7
Eating Amount:
2
Habitat ID
1002
Add Animal
ID:
7447
Remove Animal
Species:
giraffe
Search by Species
Species Details
ID:
102
Name:
Chandler
Age:
9
Sex:
M
Height:
400
Weight
280
Species:
giraffe
Eating Frequency:
7
Eating Amount:
4
Habitat ID
1002
Update Animal Info



14. **groupAnimalBySpecies** (Deliverable 7)


ZootopiApp
ZOOTOPIAPP
Select Tab:
Animal
Logout
ID:
7447
Name:
George
Age:
102
Sex:
M
Height:
5
Weight
200
Species:
tortoise
Eating Frequency:
7
Eating Amount:
2
Habitat ID
1002
Add Animal
ID:
7447
Remove Animal
Species:
duck
Search by Species
Species Details
ID:
102
Name:
Chandler
Age:
9
Sex:
M
Height:
400
Weight
280
Species:
giraffe
Eating Frequency:
7
Eating Amount:
4
Habitat ID
1002
Update Animal Info

15. **getSpeciesDetails** (Deliverable 8 - 10)



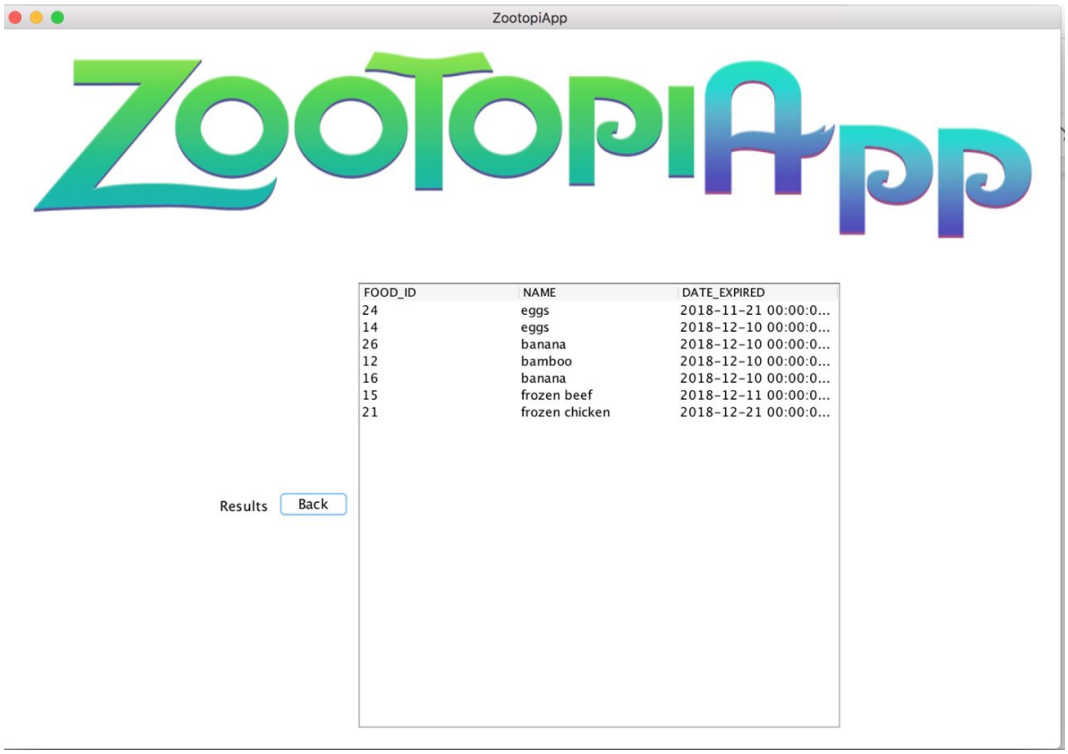
ANIMAL_ID	NAME	AGE	SEX	SPECIES
101	Johnny	5	M	duck
201	Ruby	4	F	duck
107	Mason	3	M	fox
207	Mary	2	F	fox
102	Chandler	9	M	giraffe
202	Monica	8	F	giraffe
104	Kevin	10	M	lion
204	Clara	12	F	lion
105	Edward	9	M	monkey
205	Brittanie	10	F	monkey
106	Albert	8	M	panda
206	Jessica	8	F	panda
103	Jeremy	7	M	tiger
203	Gina	6	F	tiger

16. **getFoodDetails** (Deliverable 8 - 10)

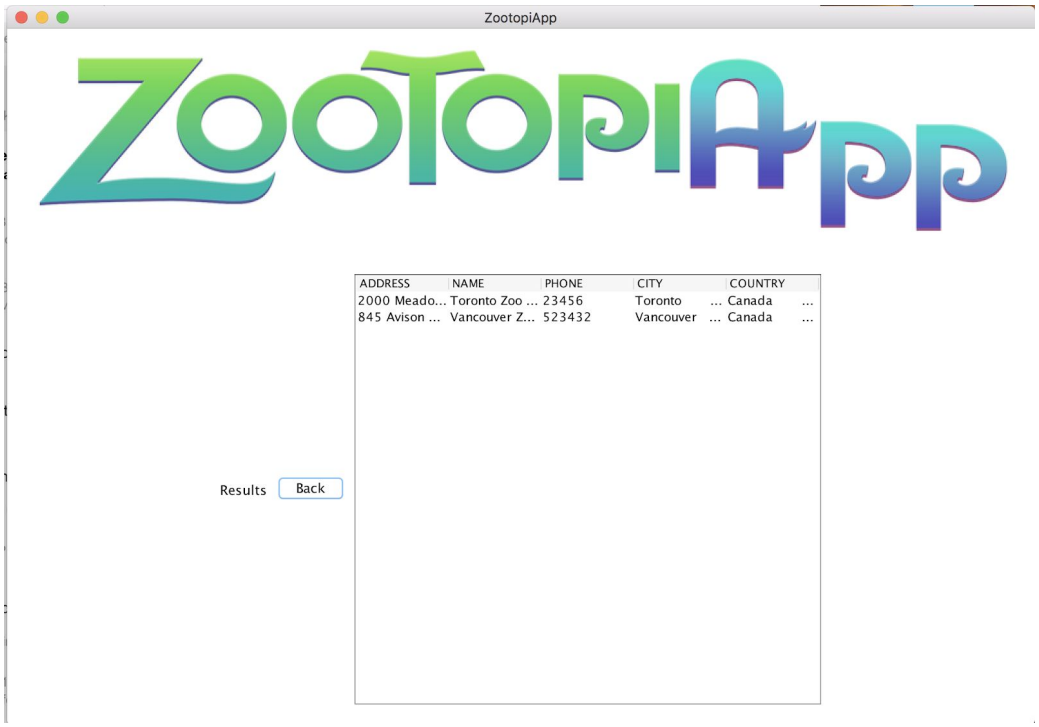
Results Back

FOOD_ID	NAME	DATE_EXPIRED
24	eggs	2018-11-21 00:00:0...
14	eggs	2018-12-10 00:00:0...
26	banana	2018-12-10 00:00:0...
12	bamboo	2018-12-10 00:00:0...
16	banana	2018-12-10 00:00:0...
15	frozen beef	2018-12-11 00:00:0...
21	frozen chicken	2018-12-21 00:00:0...

17. **getFoodSoonExpires** (Deliverable 8 - 10)

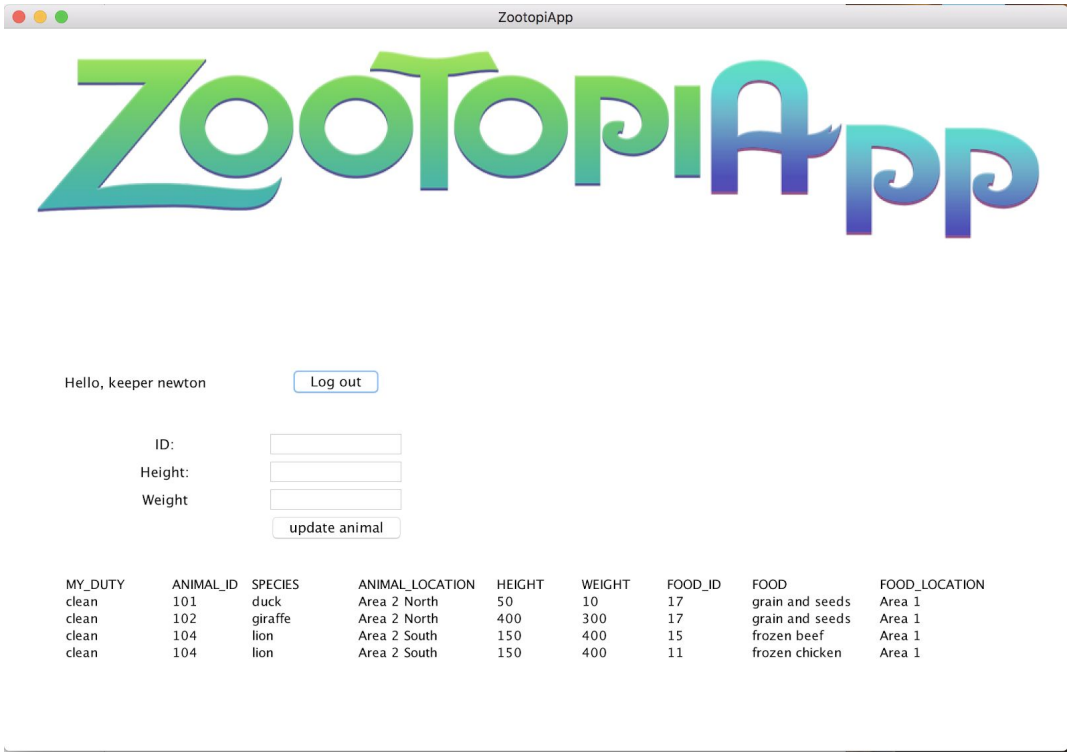


18. **filterZooByCountry** (Deliverable 8 - 10)
Example: find all zoos in Canada



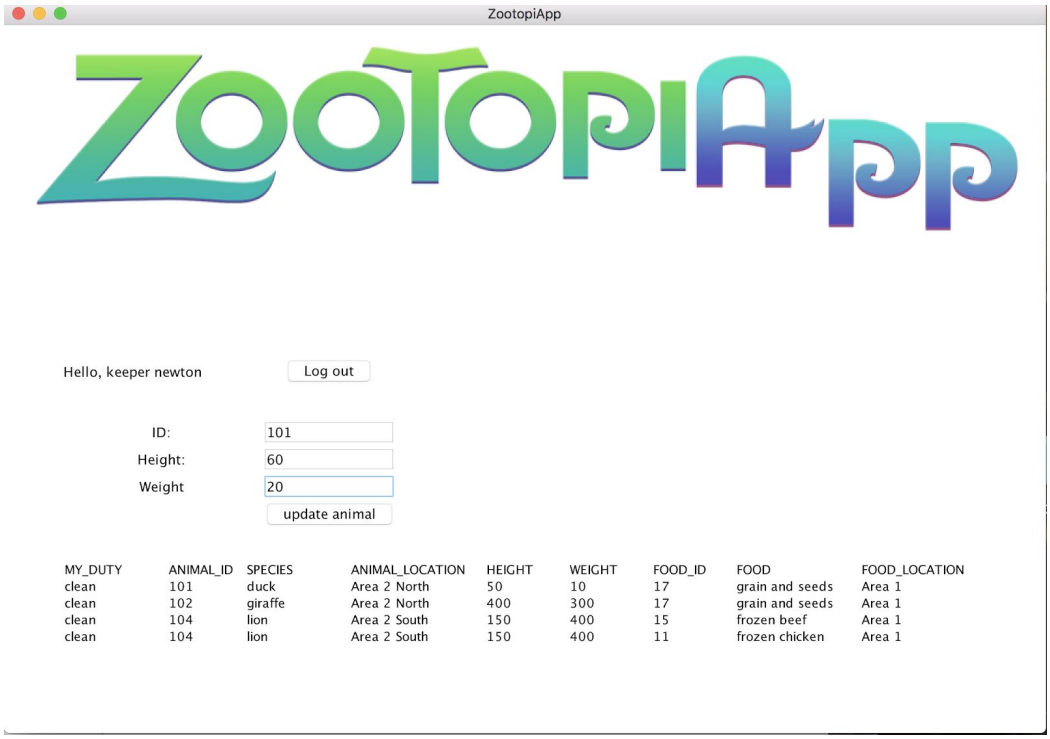
19. **getKeeperView** (Deliverable 5)

Example: this is the view when keeper with employee_id = 1234567890 logs in

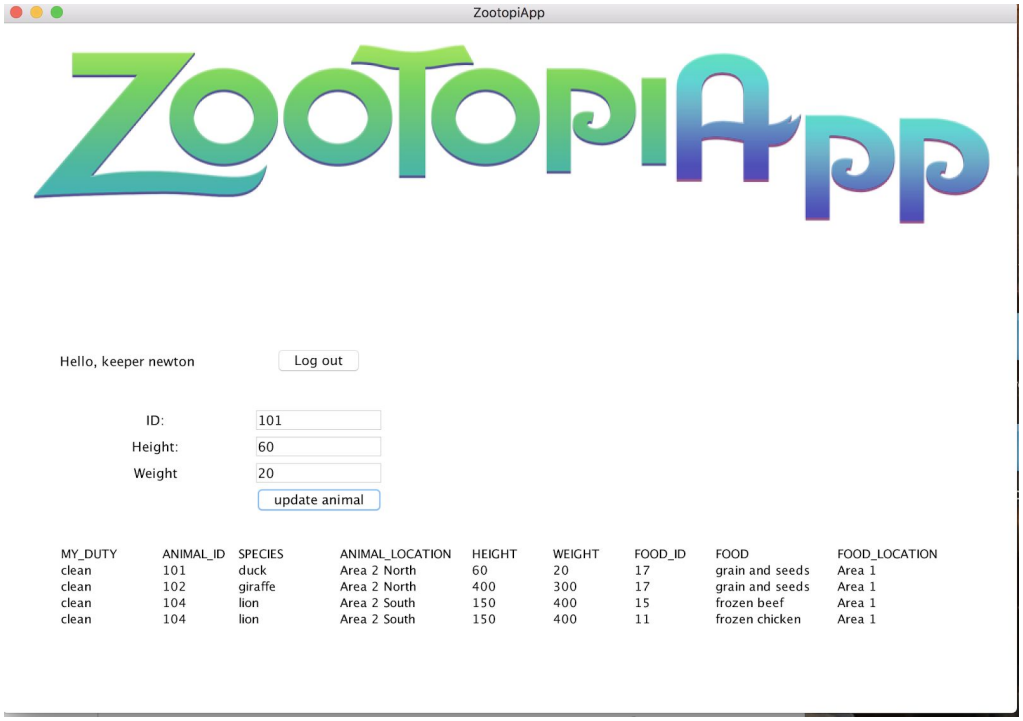


20. **updateAnimal** (Deliverable 1 and 4)

Before update:



After update:



21. **getTrainerView** (Deliverable 5)

Example: this is the trainer view for trainer with employee_id = 1376542234