Phase 1 Report

Xining Li, Yi Zhao, Xiaoyan Liu, Yaguang Chen

GaTech

February 10, 2020

Abstract

This is abstract

1 Table of Contents

2 Data Types

Attribute	Data Type	is Nullable
User Name	String	Not Null
Password	String	Not Null
email address	String	Not Null
First Name	String	Nullable
Last Name	String	Nullable

Table 1: User

Attribute	Data Type	is Nullable
Number of Hours per day	Integer	Not Null
Password	String	Not Null

Table 2: Volunteer

Attribute	Data Type	is Nullable
Pet id	String	Not Null
Name	String	Not Null
Microchip Ids	String	Nullable
Surrender date	Integer	Not Null
Surrender Reason	String	Nullable
Species	String	Nullable
Breed	List[String]	Nullable
Sex	String	Not Null
Age	Integer	Nullable
Alteration Status	String	Not Null

Table 3: Animal

Attribute	Data Type	is Nullable
Application Number	Integer	Not Null
Phone Number	Long	is Nullable
Email Address	String	is Nullable
Address_Street	String	is Nullable
Address_City	String	is Nullable
Address_State	String	is Nullable
Address_ZipCode	String	is Nullable
Applicant Name	String	Not Null
Co-Applicant Name	String	is Nullable

Table 4: Contact Information

Attribute	Data Type	is Nullable
Associated Animal	String	Not Null
Associated Application	Integer	Not Nullullable
Adoption Date	Integer	Not Null
Adoption Fee	Integer	Not Null

Table 5: Adoption Information

Attribute	Data Type	is Nullable
Associated Species	String	Not Null
Vaccination Type	String	Not Null
Required	Boolean	Not Null

Table 6: Vaccination Requirement

Attribute	Data Type	is Nullable
Associated Animal	String	Not Null
Vaccination Type	String	Not Null
Expiration Date	Integer	Not Null
Vaccination Number	String	Not Null
Entrance person	String	Not Null

Table 7: Vaccination Administration Table

3 Business Logic Constraints

User

- Inge will choose each member of Users and decide the size of Users.
- New Users can only be added by the database adminstrator with Igne's approval.
- Users' information will be maintained by the database adminstrator.

Animals

- Can only store information of 30 cats and 15 dogs. Limit is adjustable as needed or as the shelter expands.

Monthly Adoption Report

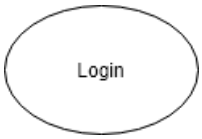
- A single adoption application can only be matched to a single animal
- animal with multiple breeds combined into a single value in alphabetical order
- A single adoption application can only be matched to a single animal

Vaccine Reminder Report

- The system should not allow you to enter a vaccine that is not yet due for an animal or that they have already been given.
- Animal breeds combined to one
- Vaccination could be updated by dba
- Sort the data in the ascending order of the vaccination due date (most recent date to latest date) and by pet ID ascending.

4 Task Decomposition with Abstract Code

4.1 Login



- Lock Type: Read-only on the **User** table
- Number of Locks: Semaphore permits depend on the server and database power. In our case, we should choose 10.
- Enabling Condition: None
- Frequency: Around 200 logins per day
- Consistency(ACID): Not critical, order is not critical
- Substacks: Mother Task is not needed. No decomposition is needed.

Abstract Code

- User enters **email**, **password** input fields.
- If data validation is successful for both username and password input fields, then:
- When *Enter* button is clicked
  - If User record is found and user entered password match the username's key associated password in the **User** table: Store login information as session variable '$UserID', and go to the **Animal Dashboard**
  - Else: Go back to the **Login**

4.2 Animal Dashboard

- Lock Type: Lookup **Animal** Name, Species, Breed, Sex, Alteration Status, Age, and Adoptability Status. Read-only on the **Animal** table
- Number of Locks: Semaphore permits depend on the server and database power. In our case, we should choose 10.
- Enabling Condition: Trigger by successful login
- Frequency: As same as the login
- Consistency(ACID): Not critical, order is not critical
- Substacks: Mother Task is not needed. No decomposition is needed.

Abstract Code

- Show the animal's name, species, breed, sex, alteration status, age, and adoptability status by the result of the corresponding SQL query.
- Populate species and adoptability status dropdowns, if no buttons are pushed, do nothing. If clicking on sepcies and/or adopatability status, query corresponding animals and display them on dashboard only.
- Upon:
  - Clicking on each column would pop out 2 choices: sort in increasing order, sort in decreasing order.
  - Clicking on the animal's name will go to the **Animal Detail**'s Servlet (implemented using RestFul API GET method).
- Show the number of available spaces.
- if the user has appropriate permission (fetched by the corresponding SQL query), an *Add Animal* button will show directing to the **Add Animal** screen.
- if the user has appropriate permission (fetch by the corresponding SQL query), an *Add Adoption Application* button will show directing to the **Add Adoption Application** screen.

4.3 Add Animal

- Lock Type: Lock on on the process of the successfully converted data structure from the post request sending the *Add Animal* process in the database.
- Number of Locks: Single Lock
- Enabling condition: Trigger by the previous **Animal Dashboard**'s *Add Animal* Button.
- Frequency: Low, about 5 per day.
- Consistency (ACID): Super critical.
- Substacks: Mother task is not needed.

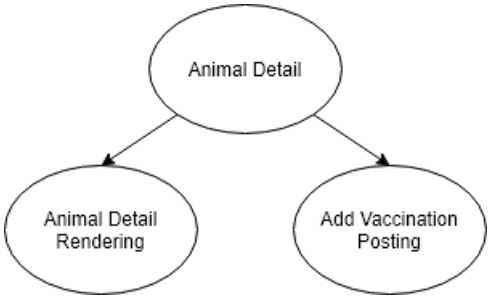
Abstract Code

- Process user's post request and convert to a data structure (for example Json or POJO)
- Validate user input, if valid continue, else return user the error message.
- Acquire permit from the semaphore
- TRY:
  - Parse the animal's species from the user's post request
  - If the animal's species from the user's post request is exist in the database and the number of availability associated with the animal's species is greater than 0:
    - Generate the unique petId by accessing the **Animal** table and pass to the setter of the petId field of the upper-mentioned data structure.
    - Submit the data structure to the database.
    - Take the user to the **Animal Detail** Screen.
  - Elae:

    * Return the user an error message with the corresponding error code.

- Catch Exception:
  - Log the error message and return the error message to the user.
- Finally:
  - Rlease the permit to the semaphore.

4.4 Animal Detail



- Lock Type: Synchronized lock on the animal's hashcode. Lookups of **Animal** information from and **Vaccinations** information from **Animal** table.
- Number of Locks: Lock on the AddVaccinationServlet's get and post methods
- Enabling condition: Trigger by the previous **Animal Dashboard**'s *Animal's name* Button.
- Frequency: Medium, about 50 per day.
- Consistency (ACID): Not critical.
- Substacks: Mother task is not needed. Two substacks:
- - Animal Detail rendering
  - Add Vaccination Posting
- Show animal's all details (attributes) by the result of the corresponding SQL query.
- Show a "Vaccinations" section which shows any Vaccination history.
- Show *Add Vaccination* button if the corresponding animal has not been adopted.
- Show *Adopt pet* button if the animal is eligible for adoption and if the session user has the permission.

Abstract Code

AnimalDetailServlet

Background: Animal Dashboard's restful API takes the user to the corresponding Animal Detail's front end, and the Animal Detail's front end call the AnimalDetailServlet using the GET method.

And here is the implementation of the AnimalDetailServlet's get method.

- the AnimalDetailServlet query the database with the restful api's parameter (**PetId**) and convert to a POJO
- the AnimalDetailServlet return the Json converted from the POJO to the front end.
- the AnimalDetailServlet call the AddVaccinationServlet using the GET method and return the user the vaccination that can be implemented.
- (frontend code) the user can optionally call the AddVaccinationServlet using the POST method.

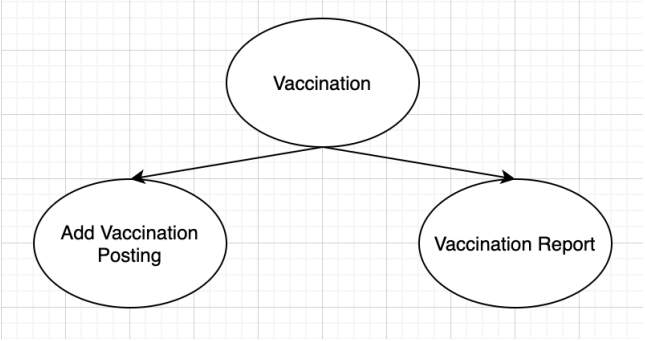
AddVaccinationServlet

Background: Animal Dashboard's restful API takes the user to the corresponding Animal Detail's front end, and the Animal Detail's front end call the AddVaccinationServlet using the GET method.

```
public class Vaccination {
        @NotNull Enum VaccinationType;
        @NotNull Date date;
        @NotNull Date nextDoseDate;
        Long vaccineTagNumber;
}


public class AddVaccinationServlet extends HttpServlet {
        public final Semaphore vaccinationSemaphore= new Semaphore(1);
        Set<VaccinationType> Get(Animal animal){
                synchronized (animal.getSingleton()){
                        if (!animal.eligibleForAdoption()){
                        return null;
                        } else {
                                return animal
                                        .getSpecies()
                                        .getRequiredVaccinations()
                                        .setSubtraction(animal
                                                .implementedVaccinations
                                        );
                        }
                }
        }
        void Post(Set<Vaccination> userImplementations) {
                synchronized (animal.getSingleton()){
                        vaccinationSemaphore.acquire();
                        for (Vaccination v: userImplementations) {
                                /*
                                implementVaccination talks to
                                the database and put the corresponding
                                vaccination into the animal's
                                corresponding vaccination table.
                                */
                                DataBaseSingleton.get(animal)
                                .implementVaccination(v);
                        }
                        vaccinationSemaphore.release();
                }
        }
}
```

4.5 Vaccinations



- Lock Type: Look up the **Vaccinations** table.
- Number of Locks: Single Lock
- Enabling condition: Trigger by the previous **Animal Detail**'s *Add Vaccinations* Button.
- Frequency: Different Frequencies
- Consistency (ACID): Not critical
- Substasks: Mother task is the **Animal Detail**.

Abstract Code

- Populate vaccinations dropdown lists
- When submit button is not clicked, nothing happened
- When submit button is clicked:
  - If vaccine is not chosen: catch Exception, log error message and return the error message to the user;
  - If vaccine is chosen, not both vaccination date or next does date are entered: catch Exception, log error message to the user;
  - Else: update Animal table,
- If submit successfully, Display **Animal Detail**

4.6 Adoption

- Lock Type: Look up Adopter Contact Information table, and is read-only. Lookup Adoption Information table, can read, update, insert
- Number of Locks: Single Lock
- Enabling condition: Trigger by the previous **Animal Detail**'s *Add Adoption* Button.
- Frequency: Different Frequencies
- Consistency (ACID): Not critical
- Substasks: Mother Task is not needed. No decomposition is needed.

Abstract Code

- Show search dialog where user can search both applicant's last name and co-applicant's last name
- Query and display Applicant's contact information
- If select Adopter, show pop up for adoption date and adoption fee
- If enter adoption date and adoption fee, update **Adoption Information** table, display **Animal Detail**
- If cancellation, display **Animal Detail**

4.7 Add Adoption Application

- Lock Type: Look up Adopter **Contact Information** table
- Number of Locks: Single Lock
- Enabling condition: Trigger by the previous **Animal Dashboard**'s *Add Adoption Application* Button.
- Frequency: Different Frequencies
- Consistency (ACID): Not critical
- Substasks: Mother Task is not needed. No decomposition is needed.

Abstract Code

- Show screen to let users enter applicant information including Application first name and last name, Address (street, city, state, zip code), phone number, email address, Date of Application, (optional) co-applicant first name and last name
- When click submit button:
  - If at least one contact information is not entered: catch Exception, log error message and return the error message to the user;
  - Else: Add **Contact Information**, display application ID generated by system
- When submit button is not clicked, nothing happen

4.8 Adoption Application Review

- Lock Type: Look up Contact Information table
- Number of Locks: Single Lock
- Enabling condition: Trigger by the login by Inge and her **Animal Dashboard**
- Frequency: Low Frequency
- Consistency (ACID): Not critical
- Substasks: Mother Task is not needed. No decomposition is needed.

Abstract Code

- Find applicationID with pending approval, display applicationID and status
- Update status with dropdown, approved or rejected, Save application status to Contact Information table

4.9 Animal Control Report

- Lock Type: Read-only lookup of the numbers of animals surrendered by Animal Control in a month and each animal's information from Animal table. Another read-only lookup of count and information of current-month adopted animals which were in the rescue for 60 or more days from Animal table and Adoption Informationl.
- Number of Locks: Several different schema constructs are needed.
- Enabling condition: Trigger by the previous **Animal Dashboard**'s *Animal Control Report* Button.
- Frequency: Low - Both have the same frequency
- Consistency (ACID): Not critical
- Substasks: Mohter task is not needed.

4.9.1 Abstract Code

- SuperUser clicked on *Animal Control Report* button from **Animal Dashboard**:
- Run the **Animal Control Report** task: query for animals' information.
  - Select month;
  - Find the animals that was surrendered by Animal Control in selected month;
  - Count these animals;
  - Display the volume of this count;
  - For each animal in this count:
    - Sort by Pet ID ascending;
    - Display animal's information;
  - Find the animals that was adopted in selected month;
  - Find these adopted animal's adopted date and surrendered date;
  - For each animal that was adopted in selected month:
    - Calculate sheltered dates
    - If sheltered dates are greater or equal to 60 days:
      - Add this animal to count
  - Display the volume of this count;
  - For each animal in this count:
    - Display animal's information.

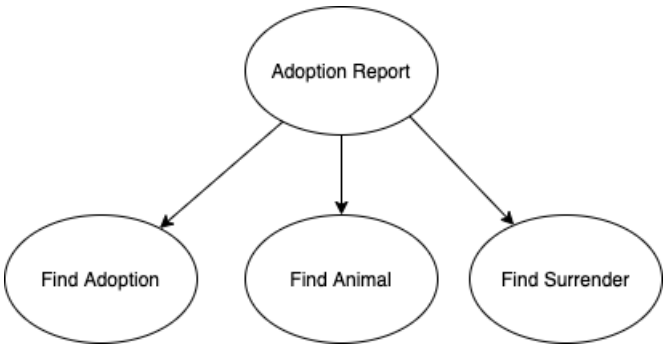
4.10 Volunteer of the Month

- Lock Type: Read-only lookup of Personal information and total hours volunteered in a month for a VolunteerUser.
- Number of Locks: Single
- Enabling condition: Trigger by the previous **Animal Dashboard**'s *Volunteer of the Month* Button.
- Frequency: Low
- Consistency (ACID): Not critical
- Substasks: Mother Task is not needed. No decomposition is needed.

4.10.1 Abstract Code

- SuperUser clicked on *Volunteer of the Month* button from **Animal Dashboard**:
- Run the **Volunteer of the Month** task: query for information and volunteered hours about the volunteers.
  - Select month and year;
  - Find all volunteers that has volunteered for selected month and year;
  - Calculate the total volunteered hours for each of these volunteers;
  - Sort the list of volunteers by descending total hours volunteered for selected month and year;
  - Display first 5 volunteers' first name, last name, email address and total volunteered hours in that month

4.11 Monthly Adoption Report



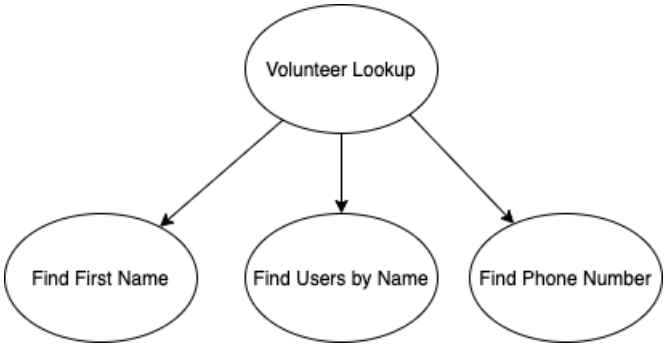
Tasks decomposition

- Lock Type: All three read-only action
- Number of Locks: Three schemas: adoption, surrender and animal
- Enabling condition: All 3 enabled by link clicked in Inge's Animal Dashboard
- Frequency: Low-all 3 have similar frequency and table need monthly frequency
- Consistency (ACID): consistency is not critical
- Subtasks: all 3 must be done, mother task is needed. Should be decomposed to 3 different sub-tasks
  - animal, surrender and adoptions

Abstract Code

- **find the adoption** info for this month
- **find the surrender** info for this month
- **find the animal** info according to adoption and surrender
- group by species, breed and count total
  - order by time and alphabetical order
  - combine breed for mixed animal

4.12 Volunteer Lookup



Constrains

- order by last name ascending and first name ascending.

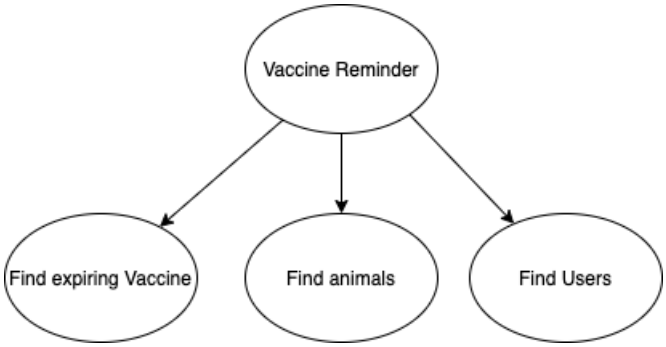
Tasks decomposition

- Lock types: Only two lookups to users and volunteer table is needed. read only action
- Number of Locks: two schema, users and volunteer
- Enabling Conditions: all 2 enabled by link clicked in Inge's Animal Dashboard
- Frequency: volunteer frequency might be lower than user table, depends on how many volunteer vs employees
- Consistency (ACID): consistency is not critical
- Subtasks: could be decomposed to 2 sub-tasks
  - find users by first and last name and email
  - find phone number from volunteer table

Abstract Code

- **find people** by their first name matching the search item (jon means jonson, jonstone etc.)
- **find users** by first and last name and email
- **find phone number** from volunteer table

4.13 Vaccine Reminder Report



Tasks decomposition

- Lock Types: All 3 read-only lookups options.
- 3 tables: user, vaccination, animal
- Enabling Conditions: all 3 enabled by link clicked in Inge's Animal Dashboard
- Consistencyt (ACID): consistency is important, when user is updating animal, this report should be unavailable
- Frequency: frequency are similar, consistency is not important
- Subtasks: all 3 must be done, mother task is needed. Should be decomposed to 3 different sub-tasks
  - animal, surrender and vaccination

Abstract Code

- User expiration date to **find all vaccine** that are expiring
- **Find animals** by vaccine activity id
- **Find person** who did the vaccine

  Here is the detailed code

  VaccineReminderReportServlet

  Background: Animal Dashboard's restful API takes the user to the corresponding Animal Detail's front end, and the Animal Detail's front end call the AddVaccinationServlet using the GET method.

```
public class VaccineReminderReportServlet extends HttpServlet {

        public JsonObject getVaccineReminderReport(Time t){
                AddVaccinationServlet.vaccinationSemaphore.acquire();

                Result result = VaccineReminderReport
                        .getInstnace()
                        .getSqlSession()
                        .query(t);

                AddVaccinationServlet.vaccinationSemaphore.release();
                return result.toJson();

        }

}
```