R. V. COLLEGE OF ENGINEERING, BENGALURU-560059

(Autonomous Institution Affiliated to VTU, Belagavi)

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



##### Wildlife Sanctuary Management System

##### Mini - Project Report

###### *Submitted by*

**Aishwarya Seth** **1RV16CS187**

**Abhiram Natarajan 1RV16CS198**

***in partial fulfillment for the requirement of 5th Semester***

***DBMS Laboratory Mini Project (16CS54)***

**Under the Guidance of**

***Suma B.***

***Assistant Professor***

**Academic Year 2018 - 2019**

**R.V. COLLEGE OF ENGINEERING, BENGALURU - 560059**

**(Autonomous Institution Affiliated to VTU, Belagavi)**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

****

**CERTIFICATE**

Certified that the project work titled **‘Wildlife Sanctuary Management System’** is carried out by **Aishwarya Seth (1RV16CS187), Abhiram Natarajan (1RV16CS198),** who are bonafide students of R. V. College of Engineering, Bengaluru, in partial fulfillment of the curriculum requirement of 5th Semester Database Design Laboratory Mini Project during the academic year **2018-2019**. It is certified that all corrections/suggestions indicated for the internal Assessment have been incorporated in the report deposited in the departmental library. The report has been approved as it satisfies the academic requirements in all respect laboratory mini-project work prescribed by the institution.

**Signature of Faculty In-charge Head of the Department**

**Dept. of CSE, RVCE**

**External Examination**

**Name of Examiners Signature with date**

**1**

**2**

**Table of Contents**

[1. Introduction 1](#_Toc529991346)

[1.1 Objective 2](#_Toc529991347)

[1.2 Scope 2](#_Toc529991348)

[2. Software Requirement Specification 3](#_Toc529991349)

[2.1 Software Requirements 3](#_Toc529991350)

[2.2 Hardware Requirements 3](#_Toc529991351)

[2.3 Functional Requirements 4](#_Toc529991352)

[3. Entity-Relationship Diagram 6](#_Toc529991353)

[4. Detailed Design 8](#_Toc529991354)

[4.1 DFD Level 0 9](#_Toc529991355)

[4.2 DFD Level 1 10](#_Toc529991356)

[4.3 DFD Level 2 11](#_Toc529991357)

[5. Relational Schema and Normalization 13](#_Toc529991358)

[5.1 Normalization of Tables 14](#_Toc529991359)

[5.2 Relational Schema 15](#_Toc529991360)

[6. Conclusion 16](#_Toc529991361)

[7. References 17](#_Toc529991362)

[8. APPENDIX: Snapshots 18](#_Toc529991363)

[8.1 Tables – Original 18](#_Toc529991364)

[8.2 User Interface and Usage Demonstration 20](#_Toc529991365)

[Logging In & Usage as a Staff Member or General Public 20](#_Toc529991366)

[Animal Insertion 22](#_Toc529991367)

[Animal Search Operation 24](#_Toc529991368)

[Animal Queries 26](#_Toc529991369)

[Logging In & Usage as a Vet 27](#_Toc529991370)

[8.3 Usage with MongoDB 29](#_Toc529991371)

[8.4 Incorrect Login 30](#_Toc529991372)

[8.5 Webpage Developed for Visitors at the Wildlife Sanctuary 31](#_Toc529991373)

# Introduction

A wildlife sanctuary, is a naturally occurring sanctuary, such as an island, that provides protection for species from hunting, predation, competition or poaching; it is a protected area, a geographic territory within which wildlife is protected. Refuges can preserve animals that are endangered. These establishments maintain important organisms, and are frequently used to study animals in their natural habitats. They can be used for research.

It is important that these animals are regularly given supplies, such as their food, etc. At the same time, it is essential that their habitats are maintained. The wounds, health, and other requirements of all the creatures must be well looked after. Ensuring that these requirements are met would lead to a better understanding of all the local organisms found in the sanctuary. It also leads to preservation of the natural beauty of the selected areas.

In most existing systems, there is a vast variety of data collected regarding the growth, development, and maintenance of such regions. However, in most existing systems, such data is collected and maintained by hand. This makes it unreliable. A lot of important, useful data may just be lost. By converting this into a database, it is ensured that all requisite data is properly maintained. At the same time, no essential data is lost. When required, the data can be retrieved easily.

This would allow for easier and more effective management of the entire system. It also allows the caretakers to keep track of their findings, and provide the best possible care and facilities for the animals that they are looking after.

Thus, the development of a wildlife sanctuary system would greatly simplify the existing processes. The system is to be created in a manner that allows it to be reproducible. It can then be used to create zoo management systems or veterinary management systems as well, since all of these have a similar structure and purpose.

## Objective

The objective is to develop an efficient, reliable method for storing and usage of wildlife related information generated from a sanctuary. The project aims to preserve research, findings, and create related data such that all the organisms receive the required nutrition as well as healthcare using effective, advanced methods. It would also act as a storage mechanism for all known facts about any given animal, which acts as a useful resource in case of an adverse situation. Facilities provided by people to the sanctuary can then be quantised and controlled.

## Scope

India has a wide, diverse variety of flora and fauna. Without proper maintenance, care, and accounting for the preservation of the same, it is likely that a lot of unique organisms may be lost. In order to prevent such a phenomenon, it is essential to maintain proper records of all creatures, their development, and requirements. Additionally, understanding of these organisms is important to the understanding of human beings, and the world at large.

The system to be developed will be able to create and store such records. It allows the retrieval of related information, which would provide better and accurate healthcare for the organisms which have been chosen as a part of the sanctuary. The regions and habitats, along with all required details will be stored. This would allow for rehabilitation in case of natural calamities, and will lead to a better understanding of the system.

Staff members will be able to keep track of their records and findings. This should expedite research regarding the wildlife. Veterinary records can also be maintained.

The system is constrained by the specific varieties of organisms that have been considered in developing the basic application, as well as the available data. It can also be developed further to be used in zoo management systems, or in protection of endangered species.

# Software Requirement Specification

This section outlines the requirements for the project. It includes both the hardware and software requirements for the same. These define the basic software and platforms that were used for the development of the entire project. As a result, these are expected to be the best combination of tools to give an optimum output, and to ensure that the program runs efficiently.

## Software Requirements

The software required to build the Database Management System include the following:

* + - Microsoft Visio (2016) for the design and planning phase.
    - MySQL workbench and server (version 5.5+ for windows/8.5+ for linux) for hosting the database.
    - HeidiSQL (version 9.5+) to create the database.
    - Xampp (version 7.2.8+) to provide server platform and connectivity.
    - Python (version 3.6+) for front-end development.
    - Java (version JDK8+) for front-end development.
    - Electron JS (Javascript version 2.0.7+) for front-end development.
    - PHP (version 7.2.9+) for front-end development.
    - MongoDB (version 4.0) for implementation of the NoSQL sections of the project.

## Hardware Requirements

The minimum hardware requirements required to run this system are-

* + - Processor: 1.6 gigahertz (GHz) or faster
    - RAM: 2 GB (64-bit)
    - Free hard disk space: 1 GB
    - Graphics card: Microsoft DirectX 9 graphics device with WDDM driver

## Functional Requirements

The functional requirements of the system are:

* Authentication: Perform validation on the user to prevent unauthorized access. Three types of accounts are used- manager account, caretaker account and veterinary account.
* Easy to learn UI: The system will come with a beginner friendly UI such that no knowledge of the backend is required. This will allow the users to be able to input data easily and quickly.
* Unstructured records or reports: Since different researchers and different organisms store data in slightly different ways, the records are created using NoSQL. This keeps the data free of specific restrictions. For instance, the data points to be collected from a bat would be very different from those for a tiger. Keeping such data in NoSQL ensures that the data is specific to each organism only.
* Maintain logs: The system must be able to maintain an accurate, reliable log of all the measures taken to improve or look after the animals. This feature allows the project to be extended to other similar systems.
* Emergency information: The database will store emergency information which can then be used to treat the animals. This data can only be updated by veterinarians.
* New discoveries: To enhance understanding, and new observations or features of the animal must be added to the database. This would help their caretakers to look after the animals accordingly.
* Information about the habitat: By regularly capturing information about the current living conditions of the animal, the sanctuaries can ensure that they are getting the care that they require.
* Efficient Operation: There should be no noticeable delay between the request for a response and the response being displayed on the user’s screen. The system should be efficient and utilise minimal power.
* Robustness: The system should be robust against attacks to gain unprivileged access. All data should be backed up and accessible, so that it can easily be shared or used when required.

# Entity-Relationship Diagram

- An entity-relationship diagram (ERD) is a data modeling technique that graphically illustrates an information system's entities and the relationships between those entities. Entities are represented by means of rectangles. Rectangles are named with the entity set they represent. Attributes are the properties of entities. Attributes are represented by means of ellipses. Every ellipse represents one attribute and is directly connected to its entity (rectangle).

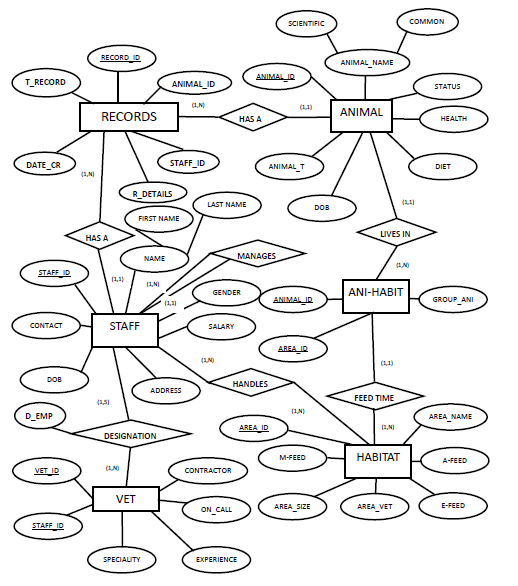
- If the attributes are composite, they are further divided in a tree like structure. Every node is then connected to its attribute. That is, composite attributes are represented by ellipses that are connected with an ellipse. Multivalued attributes are depicted by double ellipse. Derived attributes are depicted by dashed ellipse.

- Relationships are represented by diamond-shaped box. Name of the relationship is written inside the diamond-box. All the entities (rectangles) participating in a relationship, are connected to it by a line. A relationship where two entities are participating is called a binary relationship. Cardinality is the number of instance of an entity from a relation that can be associated with the relation. Total participation is represented by double lines, where each entity is involved in the relationship. Partial participation is represented by single lines.

- In the current system, the entities are records, animals, staff, habitat, vet, and ani-habitat. They are shown with their respective attributes in Fig. 1.

**Relationships between entities** –

* Records and Animals – Every animal “has a” record. And the record must have atleast 1 animal so as to be classified as a wildlife sanctuary.
* Animal\_Habitat and Animal – Every animal has a habitat it “lives in”. One animal can live in only one habitat. One habitat can have one or more animals.
* Habitat and Animal\_Habitat – The “feeding times” of each animal in a specific habitat is dependent on the habitat specifications. Every animal habitat has a habitat specific to its feeding time. Every habitat’s feeding time is relevant to one or more animal\_habitat.
* Staff and Habitat – Every staff “handles” a habitat and one or more staffs may handle a single habitat.
* Staff and Veterinary – Veterinary is a “designation” provided to staff that take care of the animals in the sanctuary. All staff need not be veterinaries.
* Staff and Records – Every staff “has a” record. The record must contain atleast one staff.
* Staff and Staff – Every staff either “manages” or is managed by another staff.



*Figure 3.1: Entity-Relationship Diagram*

# Detailed Design

The complete design of the proposed system is designed in terms of data flow diagrams.

A data flow diagram (DFD) is a graphical representation of the ‘flow’ of data through a system. This is used to model process aspects and gives a preliminary view for the data. Additionally, using this method makes it easier to bridge the gap between the user and the developer. It converts a descriptive design of the system to an implementable sequence of events.

Depending on the methodology (Gane and Sarson vs. Yourdon and Coad), DFD symbols vary slightly. However, the basic ideas remain the same. There are four basic elements of a data flow diagram: processes (transforms incoming data flow into outgoing data flow), data stores (repositories of data in the system), external entities (objects outside the system, with which the system communicates; sources and destinations of the system's inputs and outputs) and data flows (pipelines through which packets of information flow, indicating direction of the flow).

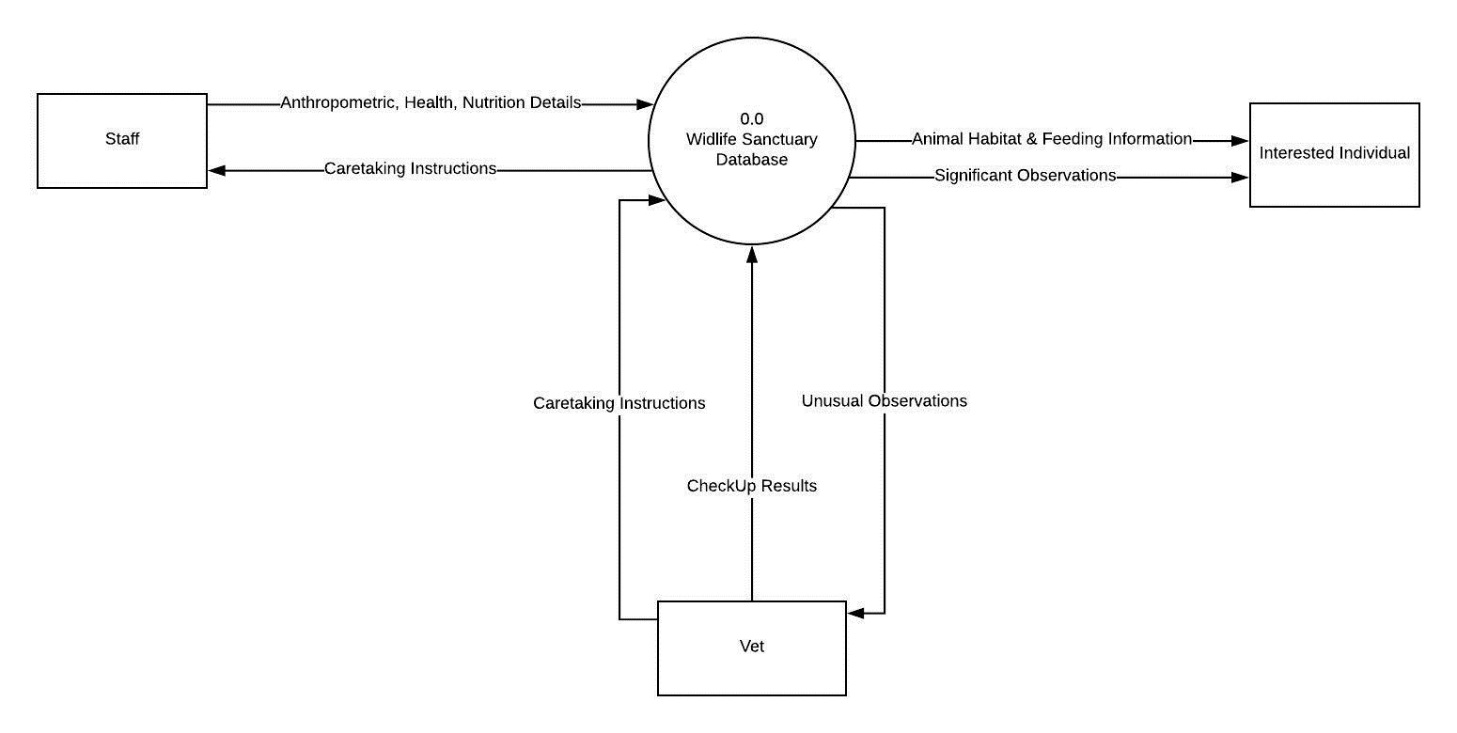
Visually, the biggest difference between the two ways of drawing data flow diagrams is how processes look. In the Yourdon and Coad way, processes are depicted as circles, while in the Gane and Sarson diagram the processes are squares with rounded corners.

Data flow diagrams are also categorized by level. Starting with the most basic, level 0, DFDs get increasingly complex as the level increases. As the data flow diagram is built, the developer will need to decide which level the diagram will be.

## DFD Level 0

Level 0 DFDs, also known as context diagrams, are the most basic data flow diagrams. They provide a broad view that is easily digestible but offers little detail. Level 0 data flow diagrams show a single process node and its connections to external entities.

In the diagram designed, external entities staff can add their observations or retrieve caretaking instructions from the database. An interested individual can access information and observations, while the vet may update health or instructions on the basis of retrieved observations. The central processing system is the wildlife sanctuary database.

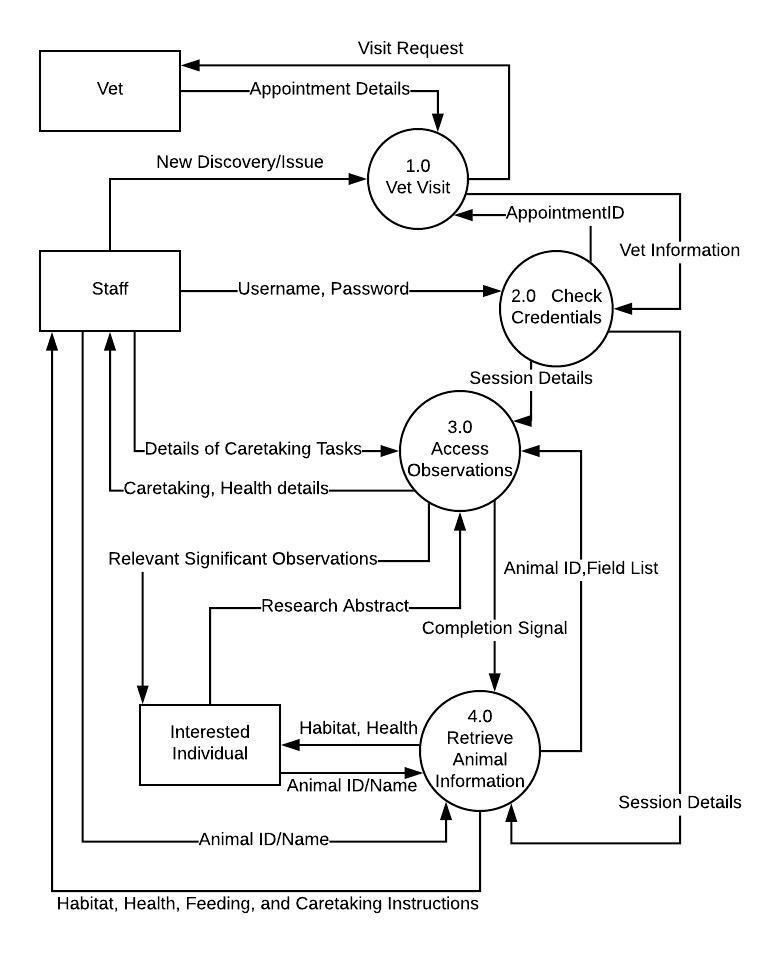


*Figure 4.1: Data Flow Diagram Level 0*

## DFD Level 1

Level 1 DFDs are still a general overview, but they go into more detail than a context diagram. In a level 1 data flow diagram, the single process node from the context diagram is broken down into subprocesses. As these processes are added, the diagram will need additional data flows and data stores to link them together.

As per the design, a staff member can request for a vet visit in case of a discovery or issue, and receives a confirmation when the request is completed. To look after the animals, the staff can retrieve information, and update the records based on the caretaking steps performed. The vet receives requests and generates acceptance signals. An interested individual can request for data, and receive the corresponding information. All updates to the database require checking of the staff credentials.



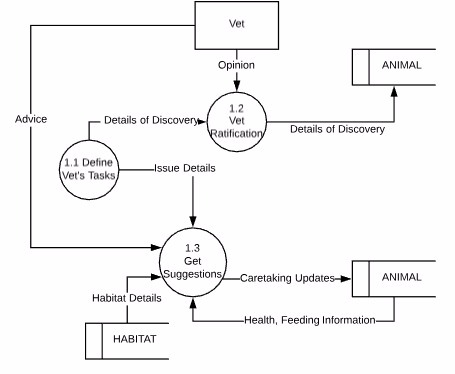
*Figure 4.2: Data Flow Diagram Level 1*

## DFD Level 2

Level 2+ DFDs simply break processes down into more detailed subprocesses. In theory, DFDs could go beyond level 3, but they rarely do. Level 3 data flow diagrams are detailed enough that it doesn’t usually make sense to break them down further.

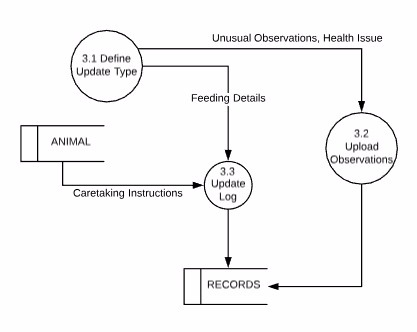
For the designed system, a level 2 DFD provided sufficient detail. Completing a vet’s visit involves definition of the vet’s tasks. If there is a new discovery, it is ratified by the vet before updating the database. If an issue is observed, the system gets suggestions by retrieving animal health and habitat information, and updating records based on the advice received from the vet.

This is shown in Figure 4.3.



*Figure 4.3: Data Flow Diagram Level 2*

To process an update, determine its type. If the data is daily observations, directly update the database. To look after the animal, retrieve caretaking instructions, generate a log and update the store in the database. These steps are shown in Figure 4.4.



*Figure 4.4: Data Flow Diagram Level 2*

# Relational Schema and Normalization

Relational or Database schema is a collection of meta-data. Database schema describes the structure and constraints of data representing in a particular domain. A relational schema can be described a blueprint of a database that outlines the way data is organized into tables. This blueprint will not contain any type of data. In a relational schema, each tuple is divided into fields called domains. A relational database schema helps you to organize and understand the structure of a database. This is particularly useful when designing a new database, modifying an existing database to support more functionality, or building integration between databases.

From E-R to Relational Schemas: Summary

(1) Suppressing generalizations from ER schemas

- Keeping the superentity

- Keeping the subentity

- Modeling by ordinary relationships

(2) ER-to-relational mapping

- Entities, weak entities

- Relationships

- Multi-valued attributes

(3) Direct mapping of generalizations to relations

Before creating the final relational schema, the normalized form of the tables must be created. This will help reduce redundancy. The diagram depicting the designed relational schema is shown in Figure 5.1

Normalization is a database design technique which organizes tables in a manner that reduces redundancy and dependency of data. It divides larger tables to smaller tables and links them using relationships. It has several sub-steps:

1. First Normal Form –

If a relation contain composite or multi-valued attribute, it violates first normal form or a relation is in first normal form if it does not contain any composite or multi-valued attribute. A relation is in first normal form if every attribute in that relation is singled valued attribute.

2. Second Normal Form –

To be in second normal form, a relation must be in first normal form and relation must not contain any partial dependency. (If proper subset of candidate key determines non-prime attribute, it is called partial dependency.)

3. Third Normal Form –

A relation is in third normal form, if there is no transitive dependency for non-prime attributes is it is in second normal form.

4. Boyce-Codd Normal Form (BCNF) –

A relation R is in BCNF if R is in Third Normal Form and for every FD, LHS is super key. A relation is in BCNF iff in every non-trivial functional dependency X –> Y, X is a super key.

## Normalization of Tables

**1NF**

None of the attributes are multi-valued, or nested. Thus, no updation to the FDs.

**2NF**

No partial dependencies on the primary keys exist in the respective tables. Thus, the FDs remain the same.

**3NF**

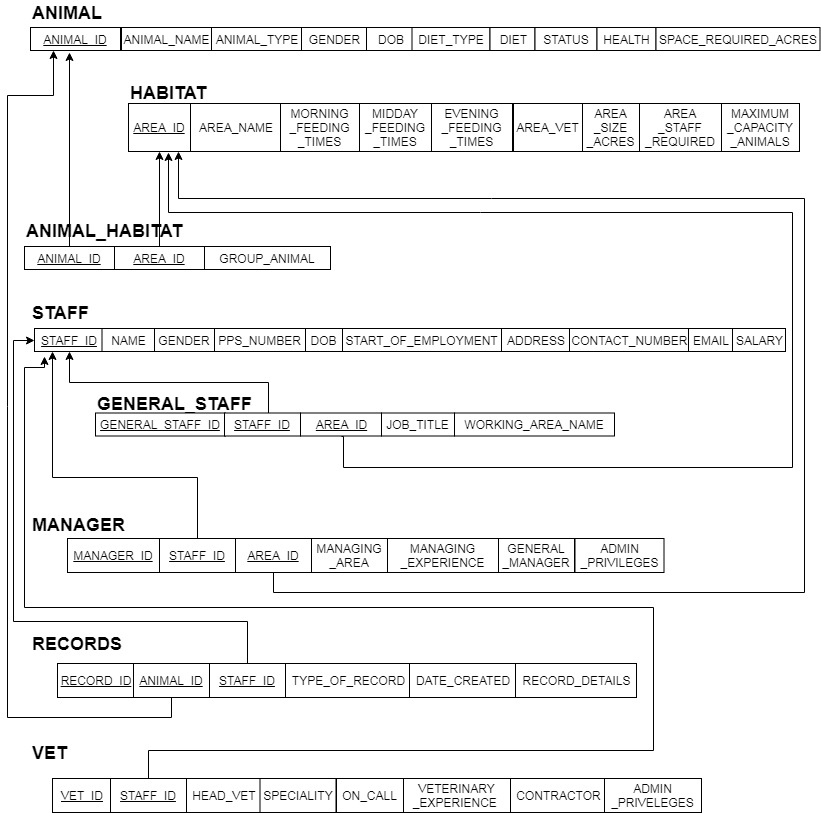
No non key attribute functionally determines another non key attribute. Therefore, no change is required.

**BCNF**

All attributes are related only to the primary keys in their respective tables, i.e. , the LHS is always a super key. Hence, all tables are in the BCNF form, or all FDs are redundant.

## Relational Schema

Combining the outputs for all of the tables into the final relational schema, Figure 5 is obtained, which shows the final implementation of the database with controlled redundancy.



*Figure 5.1: Relational Schema after Normalization Process*

# Conclusion

The application developed works with both an SQL and NOSQL backend that supports a multi user system. This allows one to create an entry when an animal is introduced to the system, this also allows for querying of the database and displaying the contents input. Common data such as the animal species, date of birth, habitat etc. are stored on an SQL backend. For those key value pairs that are specific to an entry in the record such as prescriptions, health status etc., a NOSQL backend is used. This allows for variable data content to be stored and queried directly. Additional facilities for other users include a veterinary doctor service where in the current health conditions of the animal can be monitored. Also there is a user option named customer which is for the public to see. This provides with all common data of the animals. A multi user system allows for a more robust and fault tolerant program.

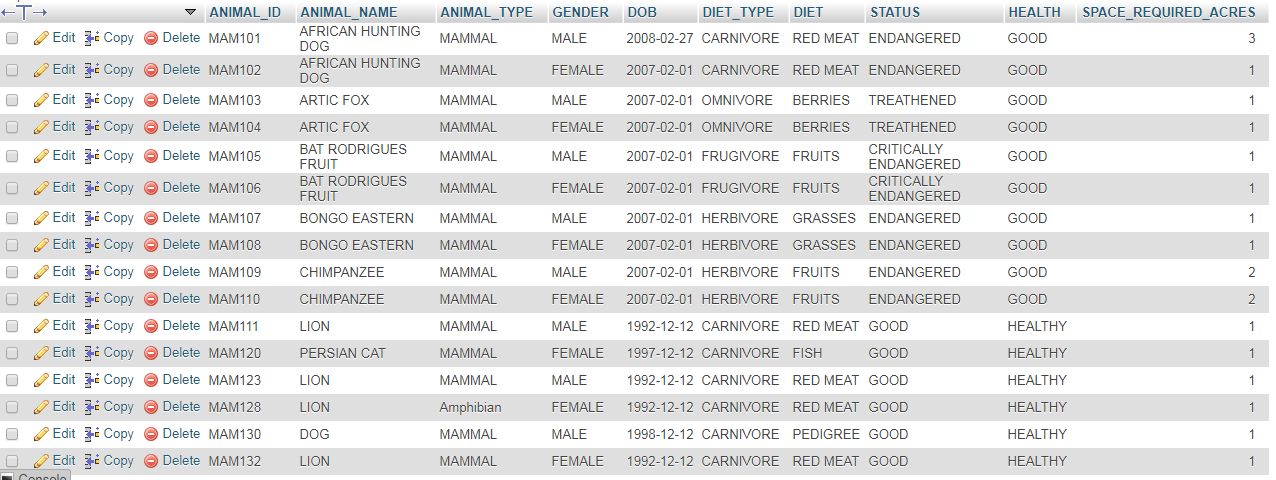
The structured backend is implemented on MySQL and the non-structured backend with MongoDB. The development of the front end has been performed on Java using the SWING toolkit. While the current system has been made as user friendly as possible by minimizing key strokes and replacing them with on screen buttons for choices, there are still some possible optimizations that can be made in the design. The structured backend has been normalised to the BCNF form to prevent any situations that could arise from redundancy or anomalies. Thus, the application developed satisfies all requirements to be considered a Wildlife Sanctuary Management system.

# References

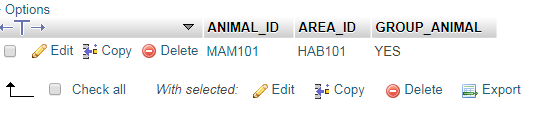
* Introduction to mongo db - <https://www.tutorialspoint.com/mongodb/>
* Interfacing mongodb with java -https://www.mongodb.com/blog/post/getting-started-with-mongodb-and-java-part-i
* MySQL references - <https://www.tutorialspoint.com/mysql/>
* MySQL to Java interfacing - <http://www.vogella.com/tutorials/MySQLJava/article.html>
* Normalization tutorial - <https://www.geeksforgeeks.org/database-normalization-normal-forms/>
* Java Swing tutorial - <https://www.javatpoint.com/java-swing>
* Zoo and Wildlife database entries - <http://wildwelfare.org/zoo-animal-databases-associations/>
* Zoo database - <http://www.wzd.cz/>

# APPENDIX: Snapshots

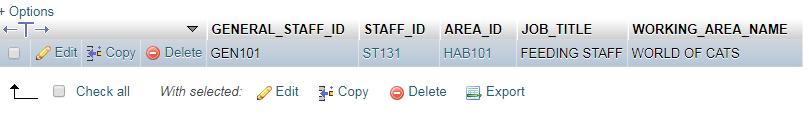
## Tables – Original



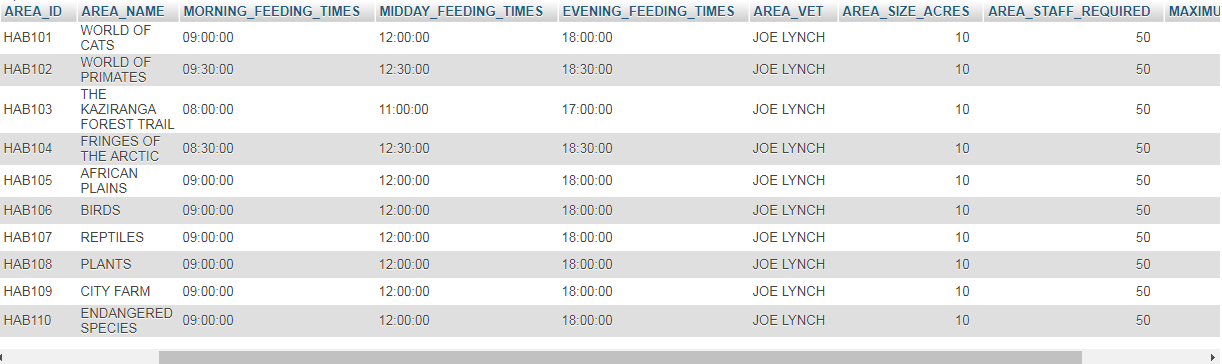
*Fig. 8.1: Animal Table*



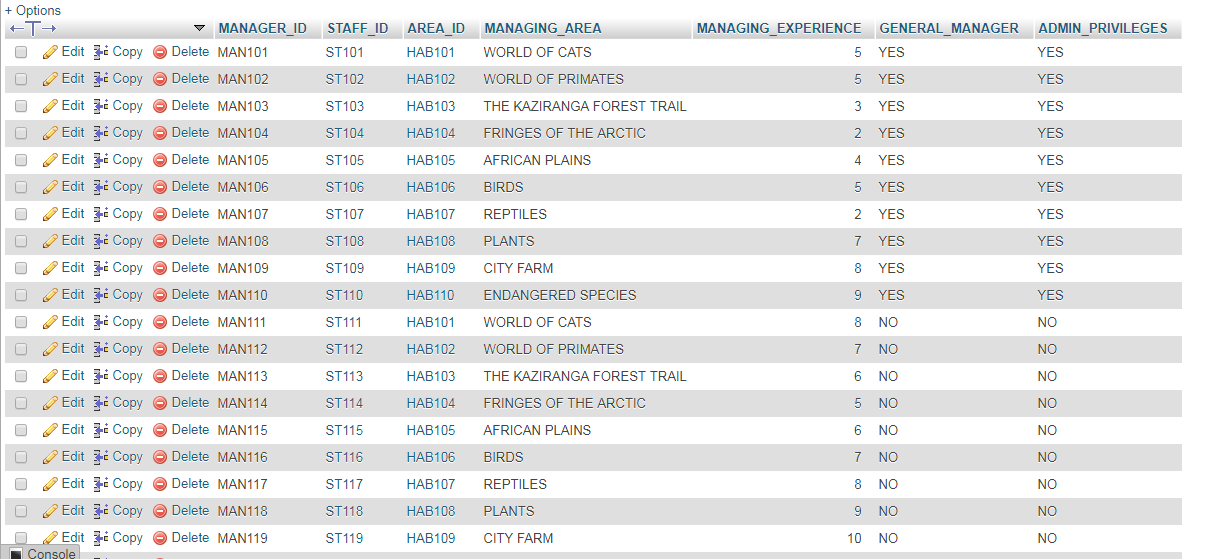
*Fig. 8.2: Animal Habitat Table*



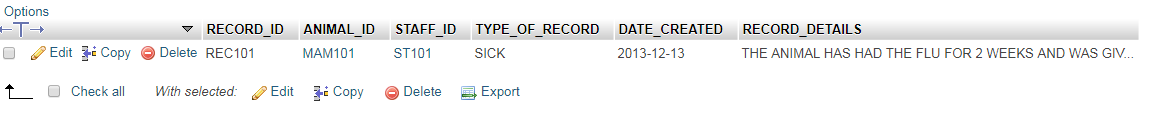
*Fig. 8.3: General Staff Table*



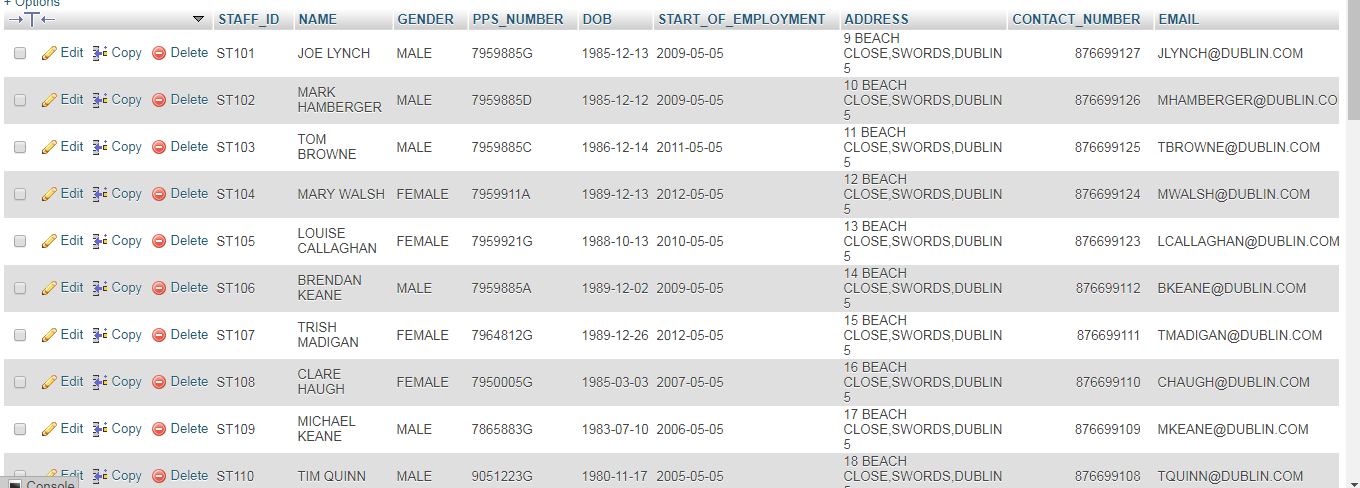
*Fig. 8.4: Habitat Table*



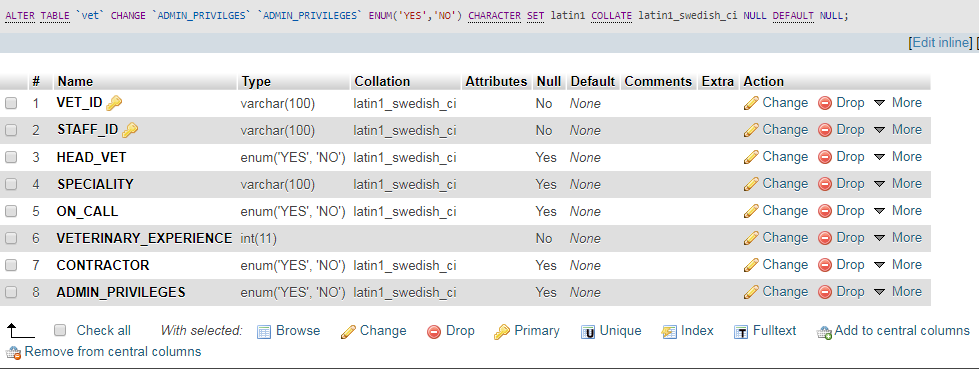
*Fig. 8.5: Manager Table*



*Fig. 8.6: Records Table*



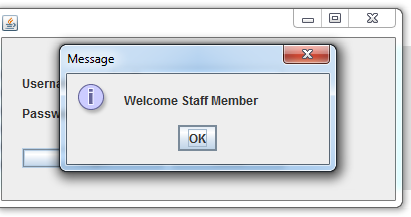
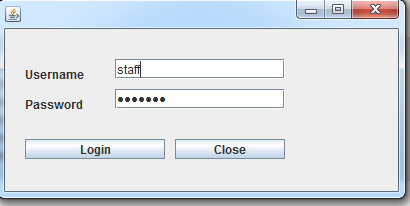
*Fig. 8.7: Staff Table*

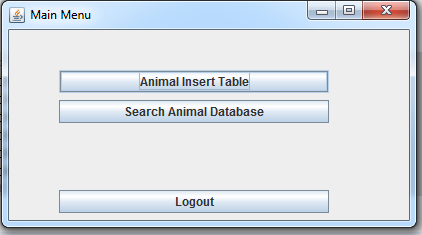


*Fig. 8.8: Vet Table*

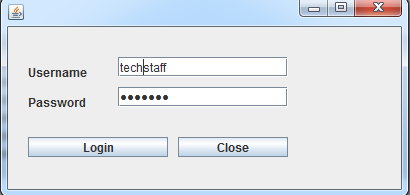
## User Interface and Usage Demonstration

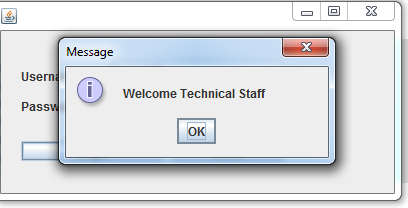
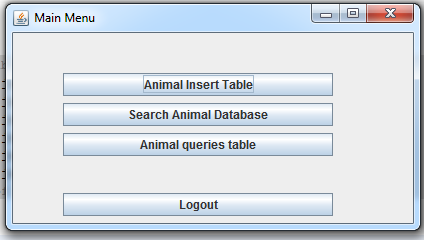
### Logging In & Usage as a Staff Member or General Public



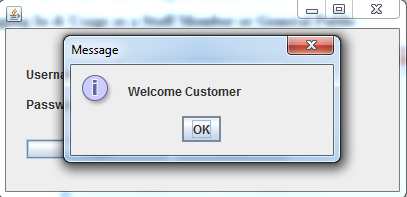
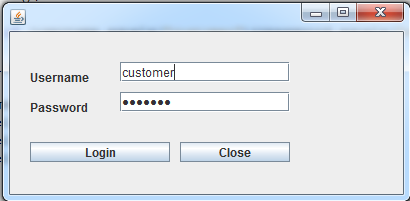
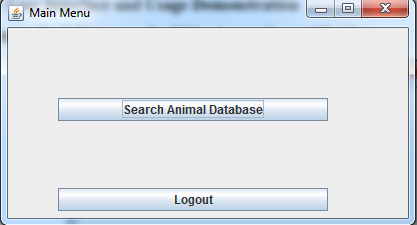


*Fig. 8.9: Login Screens for Staff*





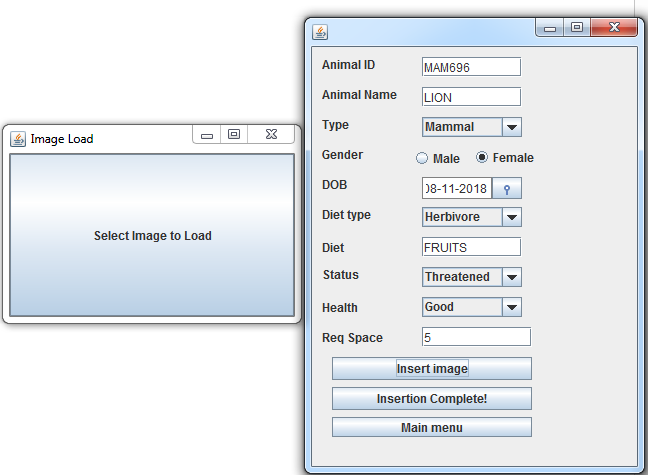
*Fig. 8.10 Login for Technical Staff*



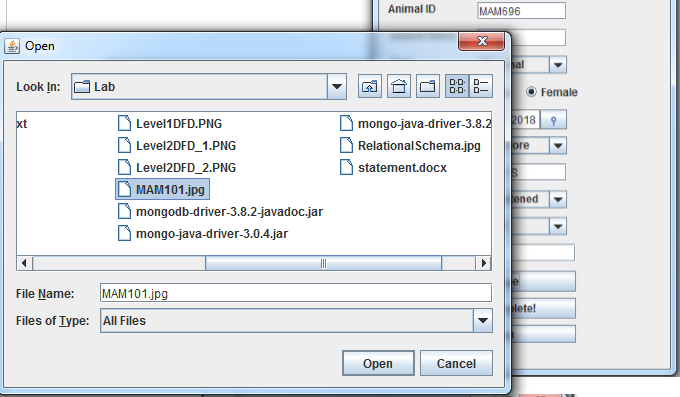
*Fig. 8.11: Login for Customer*

The three fields, i.e. Animal Insertion, Searching for an Animal and the Input Query have shared operations for these three logins. Hence, the three fields have been discussed for all logins in the section.

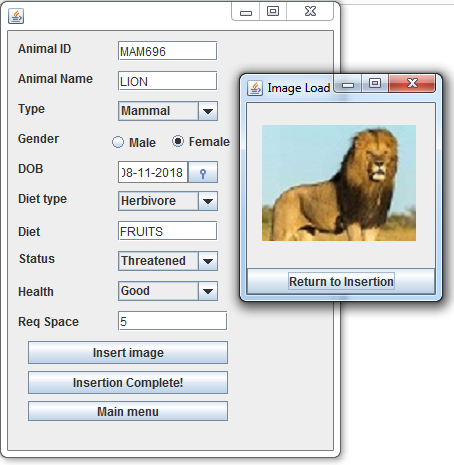
### Animal Insertion



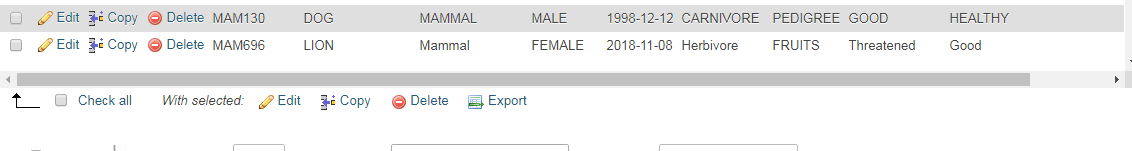
*Fig. 8.12: Input the Animal Details*



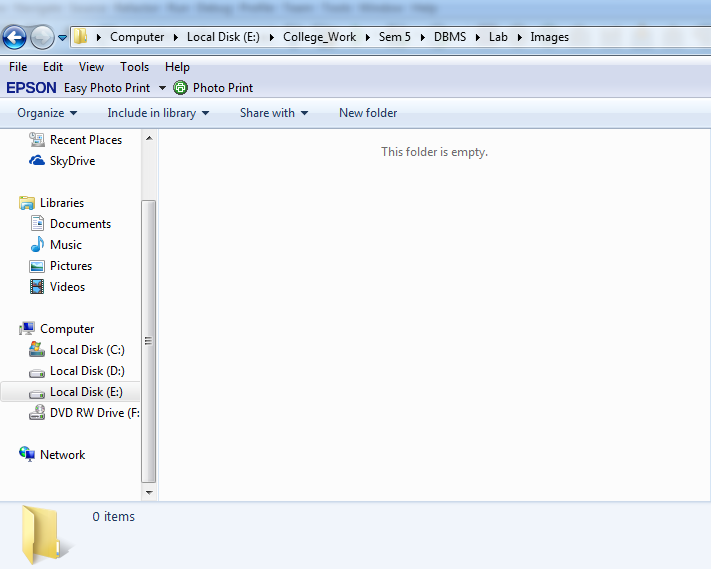
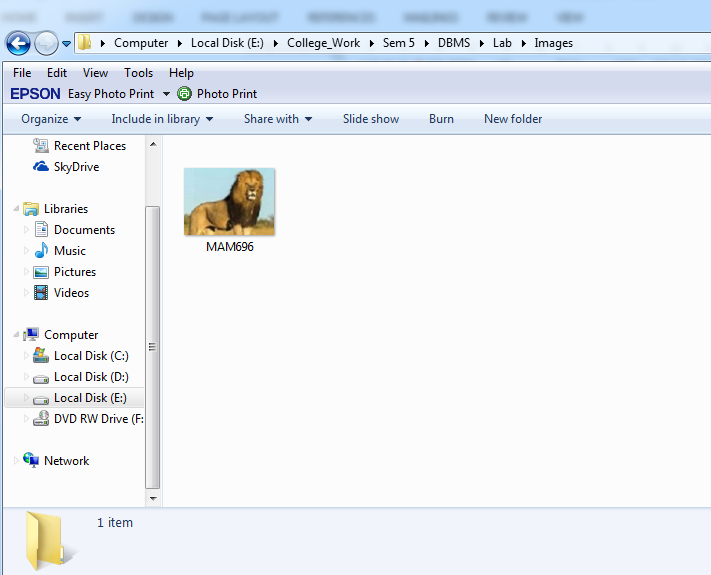
*Fig. 8.13: Select the Image to Load*



*Fig. 8.14: Select ‘Return to Insertion’ and then ‘Insertion Complete!’ to finish the Insertion process*

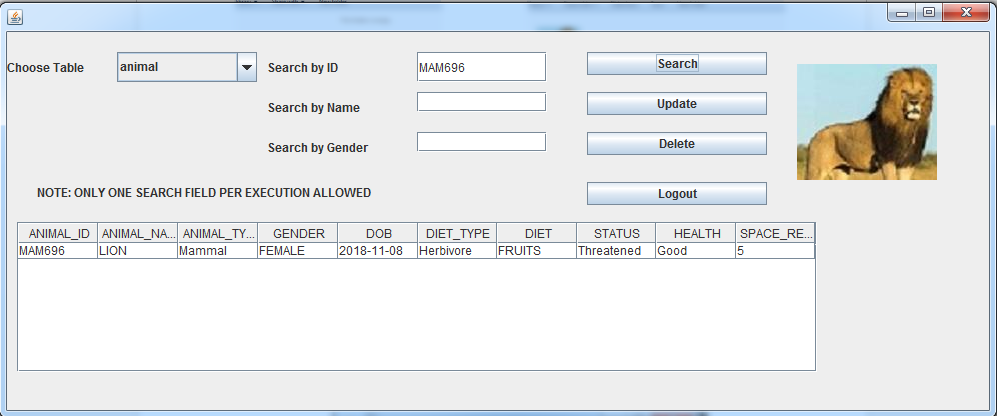


*Fig. 8.15: Updated Database*

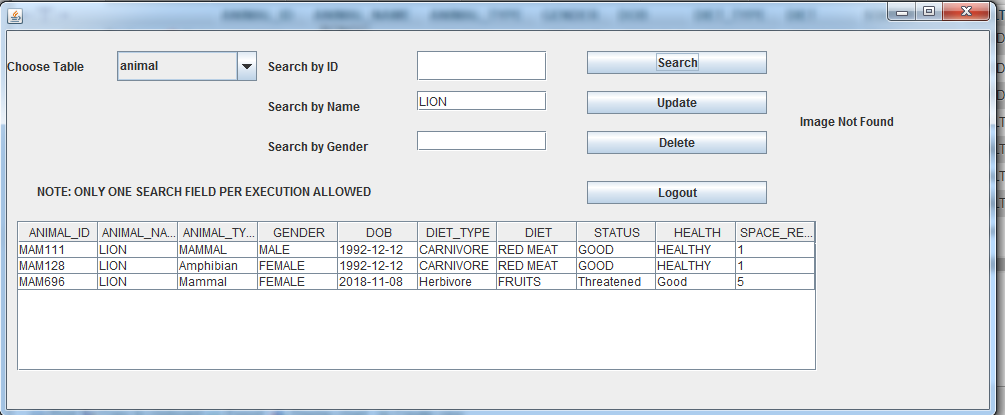
 

*Fig. 8.16: Inserted Corresponding Image*

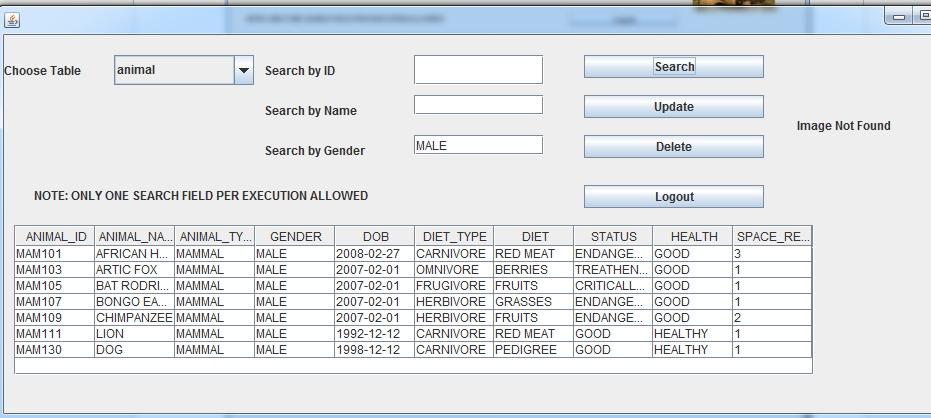
### Animal Search Operation



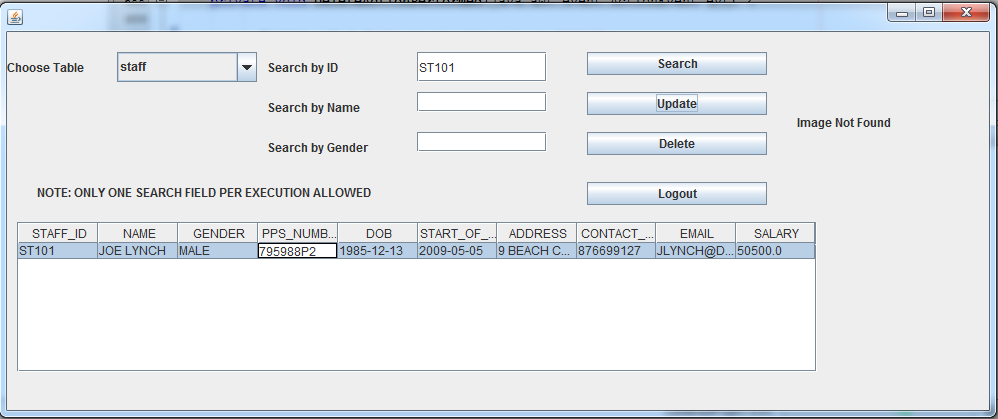
*Fig. 8.17: Basic Search Operation*



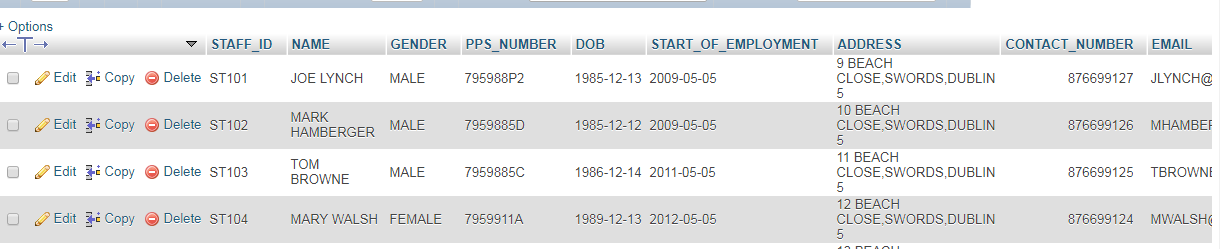
*Fig. 8.18: Search by Name*



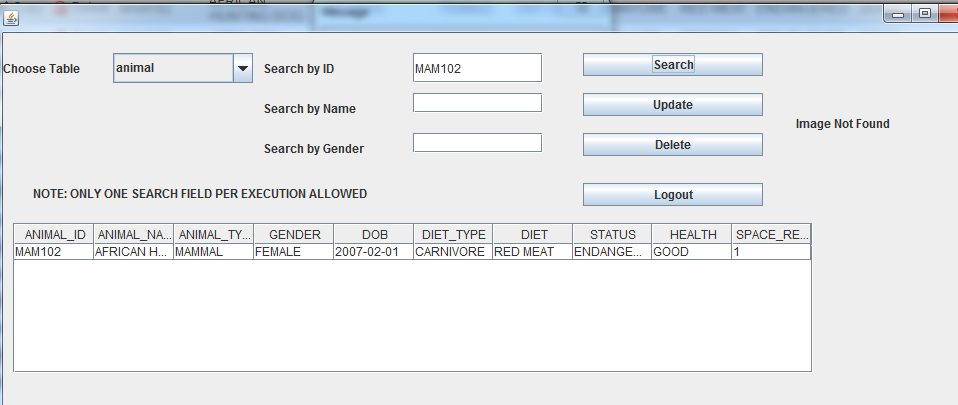
*Fig. 8.19: Search by Gender*



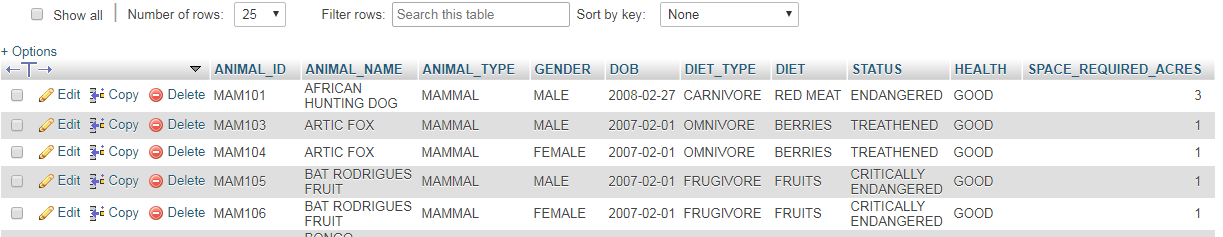
*Fig. 8.20: Search and Update in Another Table (Habitat)*



*Fig. 8.21: Updated Value*

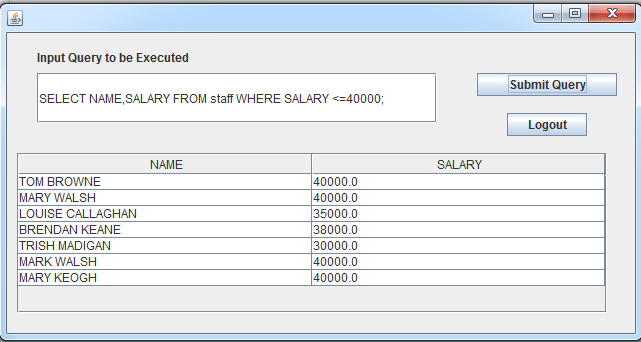


*Fig. 8.22: Deleting a Record*



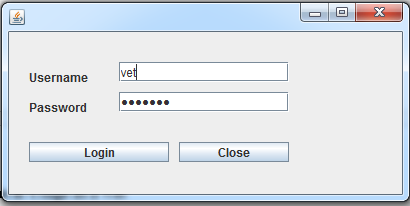
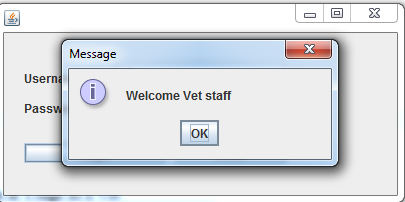
*Fig. 8.23: Record has been deleted*

### Animal Queries

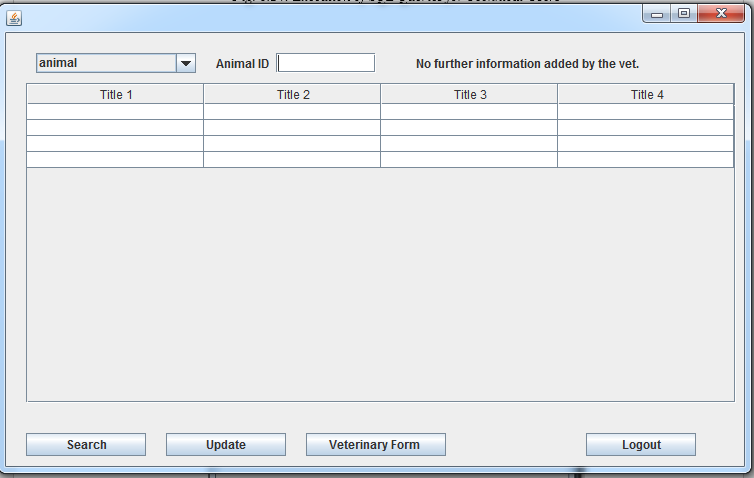


*Fig. 8.24: Execution of SQL Queries for Technical Users*

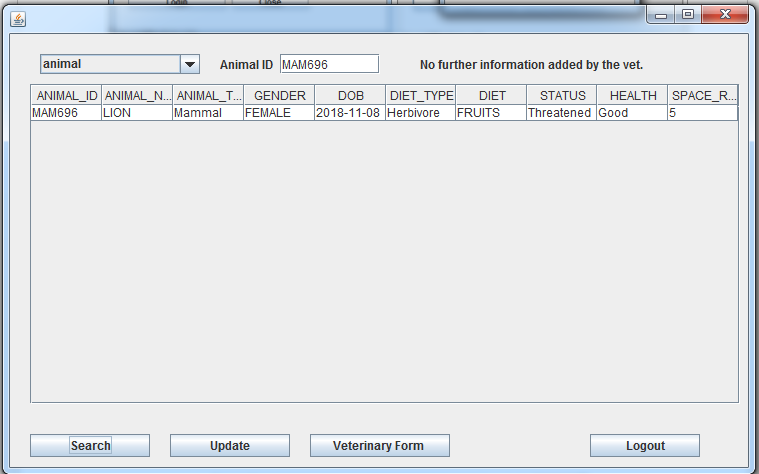
### Logging In & Usage as a Vet

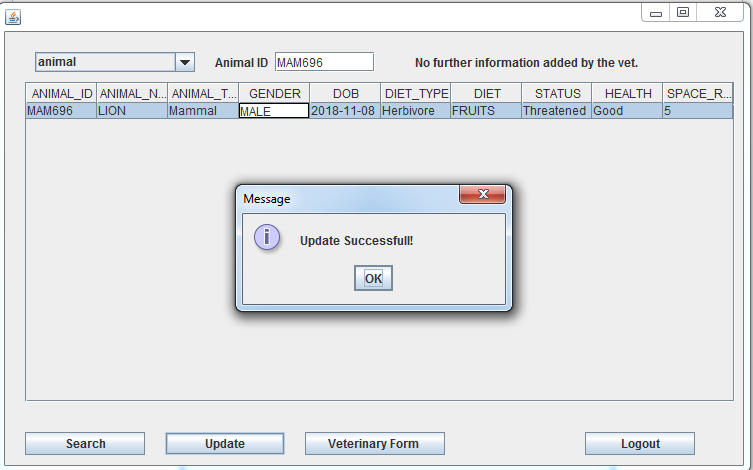
*Fig. 8.25: Logging in as a Vet*



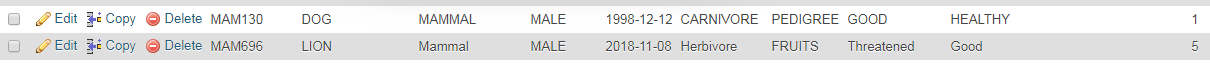
*Fig. 8.26: Basic Vet Form*



*Fig. 8.27: Identifying an Individual*

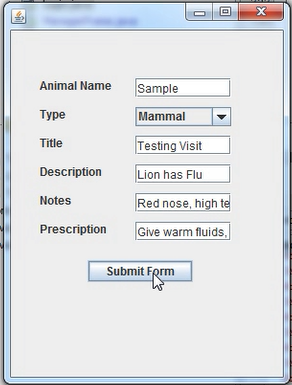


*Fig. 8.28: Update Done by Vet*

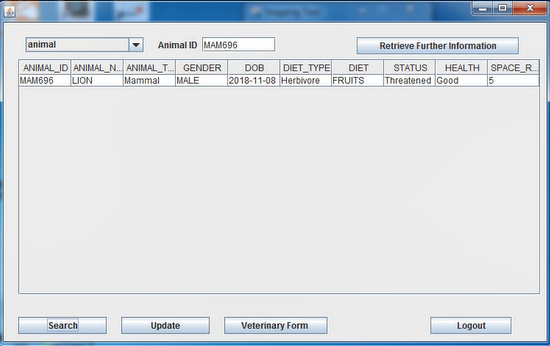


*Fig. 8.29: Update in Database*

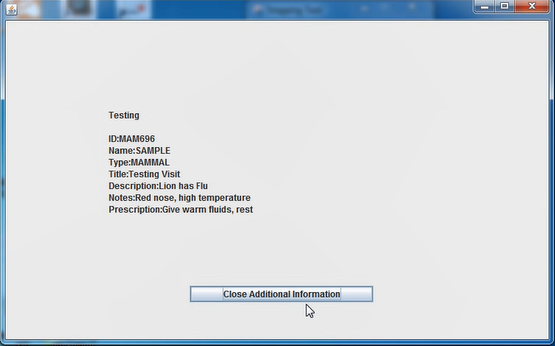
## Usage with MongoDB



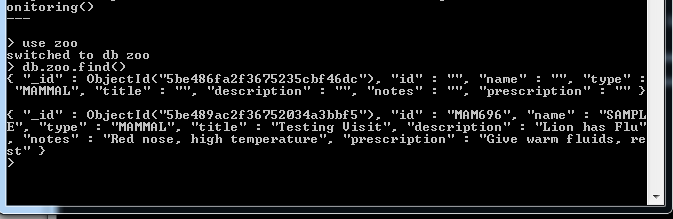
*Fig. 8.30: Sample Veterinary Form (Stored via MongoDB)*



*Fig.8.31: On searching again, can access information*

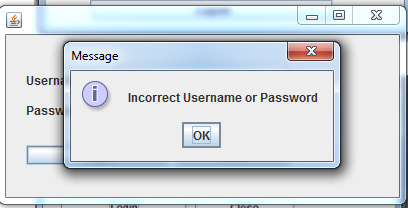
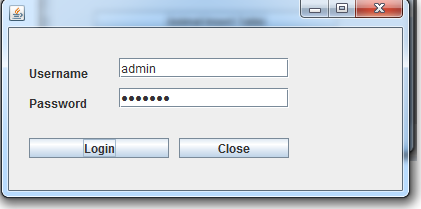


*Fig. 8.32: Information is retrieved from system*



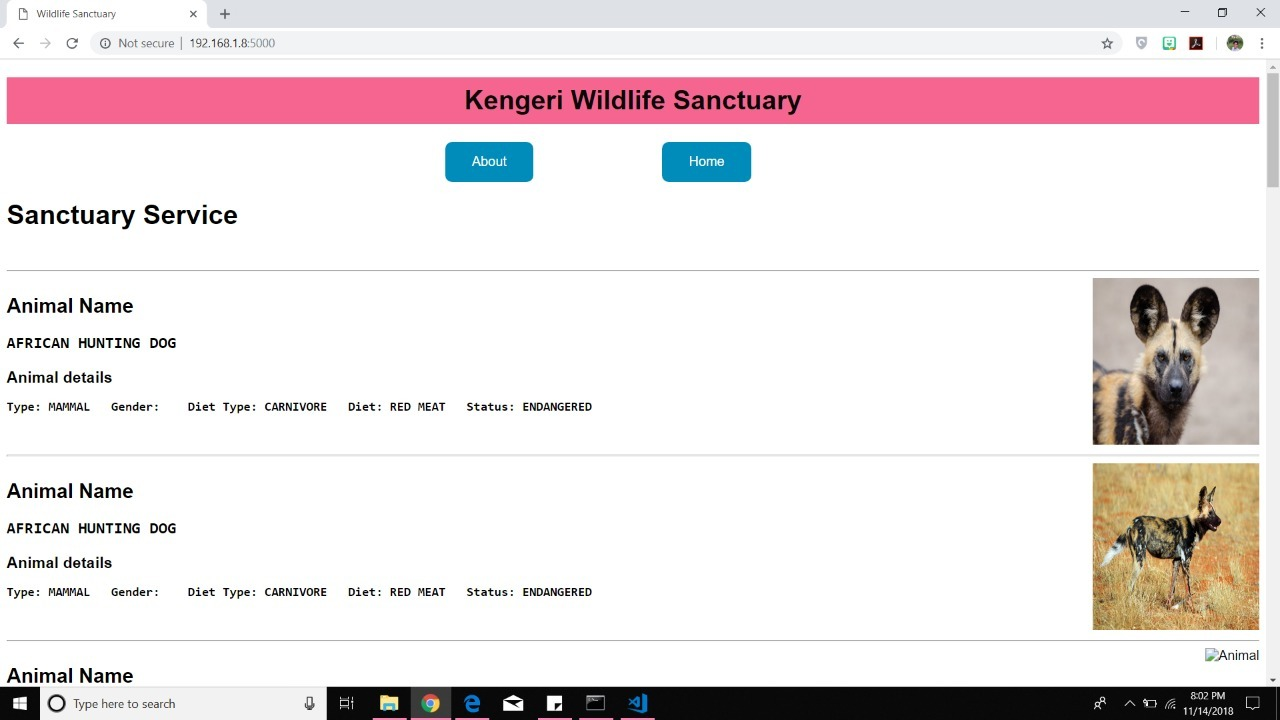
*Fig. 8.33: Reflection of the Update in Mongo Records*

## Incorrect Login



*Fig. 8.34: Incorrect Login*

## Webpage Developed for Visitors at the Wildlife Sanctuary



*Fig. 8.35 Webpage of Wildlife Sanctuary for Visitors*