**Veterinary Clinic Management System**

**Final Report**

Stephanie Brinegar, Kenneth Harris, Neil Kohan, Jeremy Lantz

CMSC 495/6381

University of Maryland University College

**Table of Contents**

[1.0 Overview 1](#_Toc21856712)

[1.1 Summary 1](#_Toc21856713)

[1.2 Scope 1](#_Toc21856714)

[1.3 Individual Contributions 1](#_Toc21856715)

[1.3.1 Stephanie Brinegar 1](#_Toc21856716)

[1.3.2 Kenneth Harris 1](#_Toc21856717)

[1.3.3 Neil Kohan 1](#_Toc21856718)

[1.3.4 Jeremy Lantz 1](#_Toc21856719)

[2.0 Project Plan 2](#_Toc21856720)

[2.1 Overview 2](#_Toc21856721)

[2.2 Key Milestones 2](#_Toc21856722)

[2.2 Requirements Specification 2](#_Toc21856723)

[2.2.1 Functional Requirements 2](#_Toc21856724)

[2.2.2 System Specification 2](#_Toc21856725)

[3.0 User’s Guide 3](#_Toc21856726)

[3.1 Purpose 3](#_Toc21856727)

[3.2 Scope 3](#_Toc21856728)

[3.3 Users 3](#_Toc21856729)

[3.4 Procedures 3](#_Toc21856730)

[3.4.1 Main Screen Options 3](#_Toc21856731)

[3.4.2 Search 3](#_Toc21856732)

[3.4.3 Schedule an Appointment 4](#_Toc21856733)

[3.4.4 Client Card 5](#_Toc21856734)

[3.5 Test Matrix 6](#_Toc21856735)

[3.5.1 Test Animal Data Entry 6](#_Toc21856736)

[3.5.2 Test Animal Details Entry 7](#_Toc21856737)

[3.5.3 Test Appointment Scheduling Entry 7](#_Toc21856738)

[3.5.4 Test Search Tool 8](#_Toc21856739)

[4.0 Application Design 10](#_Toc21856740)

[4.1 Overview 10](#_Toc21856741)

[4.2 Index 10](#_Toc21856742)

[4.2.1 Methods 10](#_Toc21856743)

[4.3 Search 10](#_Toc21856744)

[4.3.1 Methods 10](#_Toc21856745)

[4.4 Client 11](#_Toc21856746)

[4.4.1 Methods 11](#_Toc21856747)

[4.5 Appointments 11](#_Toc21856748)

[4.5.1 Methods 11](#_Toc21856749)

[4.6 Database 12](#_Toc21856750)

[4.6.1 Methods 12](#_Toc21856751)

[5.0 Development History 13](#_Toc21856752)

[5.1 Overview 13](#_Toc21856753)

[6.0 Conclusions 14](#_Toc21856754)

[6.1 Lessons Learned 14](#_Toc21856755)

[6.2 Design Strengths 14](#_Toc21856756)

[6.3 Design Limitations 14](#_Toc21856757)

[6.4 Future Improvements 14](#_Toc21856758)

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason for Changes** | **Version** |
| Initial | 10/10/19 | Initial | 1.0 |
| Lantz | 10/13/19 | Added Individual Contributions | 1.1 |
|  | 10/13/19 | Added conclusions | 1.2 |

# 1.0 Overview

## 1.1 Summary

This is the final report is a summary for the project development for the Veterinary Clinic Management System (VCMS) by Delta Group. It is an all-inclusive document, chronicling all phases and progress of development.

## 1.2 Scope

The Veterinary Clinic Management System (VCMS) is a Java based tool for use in veterinary clinics to track owners and their pets, as well as the pets’ medications, shot records, scheduled appointments, and any other relevant information as needed.

## 1.3 Individual Contributions

### 1.3.1 Stephanie Brinegar

* GUI Development
* Front End Coding
* Testing

### 1.3.2 Kenneth Harris

### 1.3.3 Neil Kohan

* Front End Coding
* GUI Management
* Database logic
* Testing

### 1.3.4 Jeremy Lantz

* Project plan and specifications
* Project Design Document
* User Guide
* Test Plan
* Final Report Documentation
* Backend Coding

# 2.0 Project Plan

## 2.1 Overview

The purpose of this project will be to design a veterinary clinic management system. The specific functions will entail animal information management, scheduling appointments, and tracking treatment information. The animal information management function will serve to allow users to add or update information pertaining to specific animals such as name, age, breed, and owners contact information. The appointment scheduling function will allow users to schedule appointments for animals once they have been setup in the system and ensure there are no overlapping appointments or appointments outside of specific work hours. The last function which is tracking treatment information will serve as a place to store data pertaining to each animals’ appointments and notes for that appointment.

## 2.2 Key Milestones

* Week 1 – Project Goal/Initial Planning
* Week 2 – Requirements Specifications
* Week 3 – User’s Guide, initial GUI, frontend, processing, and backend
* Week 4 – Project Design commit
* Week 5 – Phase 1 Coding
* Week 6 – Phase 2 Coding
* Week 7 – Phase 3 Coding
* Week 8 – Final commit and final report

## 2.2 Requirements Specification

### 2.2.1 Functional Requirements

* Connects to database
* Allows new animal creation
* Search for animal by name/owner
* Update animal information
* Create/Update appointment
* Create/Update treatment/medication records

### 2.2.2 System Specification

* Java: Version 8
* IDE: Netbeans 8.2
* Windows 10
* Database: MariaDB

# 3.0 User’s Guide

## 3.1 Purpose

The purpose of the user guide is to provide application users with instructions on how to use the Veterinary Clinic Management System (VCMS).

## 3.2 Scope

The user guide will provide detailed instruction related to basic use of the VCMS. It will explain the functions of the system and how to input/retrieve patient records. The VMCS is a standalone product.

## 3.3 Users

Veterinary staff

* Doctors
* Administration
* Doctor Aides

## 3.4 Procedures

### 3.4.1 Main Screen Options

The main screen has three options:

* Search
* Schedule an appointment
* Client Card

### 3.4.2 Search

The search screen allows users to search for records by Pet name and/or Owner last name (Figure 1). Once the user has input the data, select the “Search” button and matching records will be displayed on the client card (See section 3.4.4).

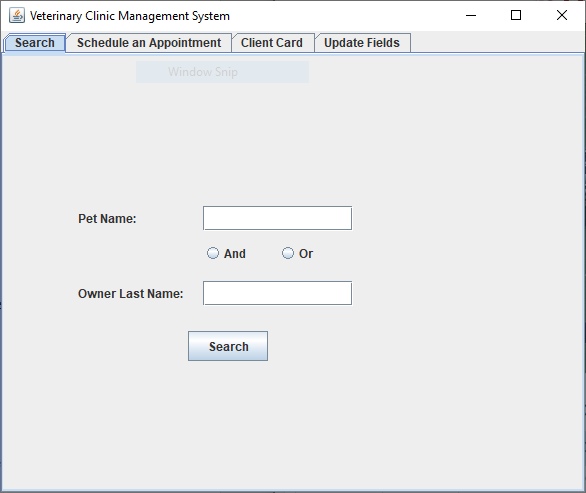


Figure 1. Search

### 3.4.3 Schedule an Appointment

This screen allows users to schedule appointments for customers. (Figure 2)

* Select Schedule an Appointment
* Enter required customer information
  + Pet name
  + Owner First Name
  + Owner Last Name
* Select Doctor Name
* Select Date
* Select Time
* Enter Duration
* Select “Schedule” button

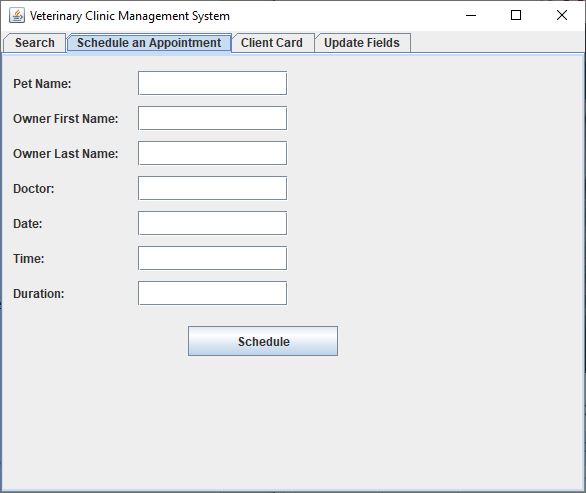


Figure 2. Schedule

### 3.4.4 Client Card

The client card will show following the search function. It will display all relevant information and allow for editing of data for a customer including:

* Pet Name
* Owner Last Name
* Owner First Name
* Gender
* Weight
* Medication List
* Shot List
* Schedule Appointment Info

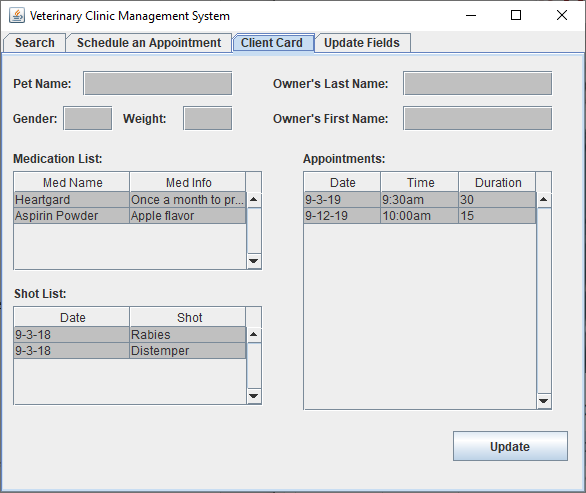


Figure 3. Client Card

## 3.5 Test Matrix

### 3.5.1 Test Animal Data Entry

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test Objective |  | Input | | | | Expected Result | Actual Result |
| Pet Name | Owners First Name | Owners Last Name | Gender | Weight |
| Test successful entry into Animal Data Database File | Sammy | John | Smith | Male | 30 | Record for new entry is created with new unique Animal ID key field |  |
| Test failure to enter all fields for database entry | Jake |  |  | Male | 5 | Owners Name field is highlighted, and error is displayed informing user it’s a required field |  |
| Test entry of already existing animal in the system | Sammy | John | Smith | Male | 30 | Error message panel is displayed to the user informing them this animal already exists in the system |  |

### 3.5.2 Test Animal Details Entry

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Objective | Input | | | Expected Result | Actual Result |
| Animal ID | Record Type | Record Detail |
| Test successful entry into Animal Details Database File for Medication type record | 1 | Medication | Benadryl | Record is added to the Animal Detail Database file tied to the Animal ID and a Record type of Medication |  |
| Test successful entry into Animal Details Database File for Shot type record | 1 | Shot | Rabies Vaccination | Record is added to the Animal Detail Database file tied to the Animal ID and a Record type of Shot |  |

### 3.5.3 Test Appointment Scheduling Entry

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test Objective | Input | | | | | | | Expected Result | Actual Result |
| Pet Name | Owner First Name | Owner Last Name | Doctor | Date | Time | Duration |
| Test successful entry into Appointment Scheduling Database File | Sammy | John | Smith | Dr. Joe | 09/07/2019 | 13:30 | 30 | Record is added to the Appointment Scheduling Database File |  |
| Test creating appointment for non-existing animal | Bruiser | Jessica | Alba | Dr. Joe | 09/07/2019 | 14:00 | 15 | Error panel will popup informing the user that the animal doesn’t currently exist in the database |  |
| Test creating appointment for duplicate date and time | Jake | Jane | Doe | Dr. Joe | 09/07/2019 | 13:30 | 30 | Error panel will popup informing the user that the specified date and time is already being booked for another appointment |  |

### 3.5.4 Test Search Tool

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Objective | Input |  | Expected Result | Actual Result |
| Pet Name | Owner Last Name |
| Test searching for a known animal in the Animal Data Database | Sammy | Smith | Client Card screen is displayed for Sammy |  |
| Test searching for an animal not known to the Animal Data Database | Bruiser | Alba | Error Message is displayed to user informing them there is no record for that animal/owner last name combination |  |
| Test searching for without an animal name |  | Smith | Error message panel is displayed to the user informing them all search fields are required entry fields |  |

# 4.0 Application Design

## 4.1 Overview

In the Veterinary Clinic Management System (VCMS) application there will be a centralized design which centers around the Index class. This class will be responsible for calling the correct GUI based on parameters received from the previous class that was called. During the initial run this class will by default call the Search class then will continue to call the appropriate class based on the return data from the last class that was called. From the Index class one of three GUI classes can be called which consists of the Search, Client, and Appointment classes. Each of the classes will implement its own GUI and handle displaying/processing the data accordingly based on what is handled by the specific class. Behind all the individual GUI classes will be the Database class which will handle retrieving and posting of data in the appropriate databases based on what method is being implemented at a given point.

## 4.2 Index

This is the main hub of the application that determines which class to call.

### 4.2.1 Methods

Index – This method will be an infinite looping method. The loop will be set to a condition that will always be false until a specific state is achieved which will be when someone closes the window which will trigger the loop to cease. This method will call out to the appropriate class based on the returns from each previous class call.

## 4.3 Search

This is the initial search GUI.

### 4.3.1 Methods

* **Search** – This method will handle building of the initial search panel GUI with all its input fields which contains the “Pet Name” and “Owner Last Name” input fields. It will also contain the Search, Schedule an Appointment, and Client Card buttons at the top along with the Search button at the bottom.
* **actionPerformed** - This method will handle the actions from the button clicks. Based on if one of the buttons at the top of the screen is clicked it will return to the index class to tell it which class to call next. If the Search button is used the searchInput method will then be utilized.
* **searchInput** – When the search button is clicked this method will retrieve the input values from the screen. It will then pass those values to the Database class to search for a valid record. If a valid record isn’t found, then an error panel will be displayed. If a valid record is found, then those values will be returned to the index class for the Client class to be called.

## 4.4 Client

This is the client data GUI.

### 4.4.1 Methods

* **Client** – This method will build the Client Card GUI which will contain the Pet Name, Owners Last Name, Gender, Weight, Owners First Name, Medication List, Shot List, and Appointment information fields. If it is passed the Pet Name and Owners Last Name from the Search class then it will utilize the getPet method to prepopulate all the fields.
* **actionPerformed** – This will handle clicking of the Cancel and Submit buttons at the bottom of the page or clicking of any of the Search, Schedule an Appointment, or Client Card buttons at the top of the screen. When the Submit button at the bottom of the screen is used the postAnimal method will be called.
* **postAnimal** – This method will first retrieve all the values of the screen. It will then check if it wasn’t passed any incoming parameters to see if the data entered was for an animal that already exists in the system and if so then a warning message will be displayed. I parameters were passed in then the information that was retrieved from the screen will be passed to the Database class for the appropriate record to be updated in the correct database.

## 4.5 Appointments

This is the schedule appointment GUI.

### 4.5.1 Methods

* **appointment** – This method will build the Schedule Appointment GUI which will contain the Pet Name, Owners First Name, Owners Last Name, Doctor, Date, Time, Duration input fields. It will also contain the Schedule button at the bottom along with the Search, Schedule an Appointment, and Client Card buttons at the top.
* **actionPerformed** – Here the clicks of any of the individual buttons on the screen will be handled. If the Schedule button is used, then the schedule method will be implemented. If any of the buttons at the top of the panel are clicked, then they will return to the Index class to call the next appropriate class GUI.
* **Schedule** – This method will retrieve the input values from the field. Once it has the input values it will verify that the animal exists in the Animal Data database and that there isn’t an overlapping appointment. If either one of these is found in error a error panel will be displayed stating the appropriate error to the user. If an error isn’t found, then the data will be passed to the Database class to add the new appointment to the appropriate database file.

## 4.6 Database

This class will handle both retrieving and posting of data to and from the database files.

### 4.6.1 Methods

* **searchPetName** – This method will search the Animal Data database for a matching record based on the parameters passes to it. It will return the animal ID if a matching record is found or 0 for a non-match.
* **searchMedicaton** – This method will be passed the animal ID to search the Animal Details database file and return an array of values for any records that have a type of “Medication”.
* **searchShot** – This method will be passed the animal ID to search the Animal Details database file and return an array of values for any records that have a type of “Shot”.
* **postAnimal** – This method will be passed all the data for an animal and either add or update a record based on if it exists or not in the Animal Data database file.
* **postMediation** – This method will be passed an animal ID and an array of values to add to the Animal Details database file with a record type of “Medication”.
* **postShot** – This method will be passed an animal ID and an array of values to add to the Animal Details database file with a record type of “Shot”.
* **checkAppointment** – This method will be passed a date, time, and duration for an appointment and check to see if it will overlap an existing appointment.
* **postAppointment** – This method will be passed a all the parameters required to create a record in the Appointments database file.
* **getAppointment** – This method will be passed an animal ID which will be used to return an array of all appointments that are found to be linked to that animal ID.

# 5.0 Development History

## 5.1 Overview

* Week 1 – Project Goal
  + Established group
  + Formed initial project plan
* Week 2 – Requirements Specifications
  + Organized team roles
  + Setup communication channels (Discord)
  + Outlined Milestones
  + Submitted Project Plan
* Week 3 – User’s Guide, initial GUI, frontend, processing, and backend
  + Developed testing plan
  + Refined design ideas
  + Submitted User Guide
* Week 4 – Project Design commit
  + Developed design structure
  + Submitted Project Design Document
* Week 5 – Phase 1 Coding
  + Created GUI
  + Developed front end
  + Submitted Phase 1 source
* Week 6 – Phase 2 Coding
  + Created Database
  + Revise Phase 1 Code
  + Submitted Phase 2 source
* Week 7 – Phase 3 Coding
  + Revised current code
  + Conducted initial testing
  + Submitted Phase 3 source
* Week 8 – Final commit and final report
  + Finalized project
  + Conducted final testing
  + Submitted Final Report

# 6.0 Conclusions

## 6.1 Lessons Learned

## 6.2 Design Strengths

* The use of a GUI makes the program easy to use and relatively self-explanatory
* Simple and easy to understand code

## 6.3 Design Limitations

* When the program needs edited, database knowledge is mandatory
* Limit of 50 records on client panel
* Application is not portable

## 6.4 Future Improvements

* More interactive GUI
* In-depth calendar tracking all upcoming appointments
* Add portability and lose database reliance
* Enable deletion of pets/owners
* Convert to standard Javadoc notation