Black Dog

Analysis and Design Document

Student: Victor Pădurean

**Group: 30434**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <06/03/20> | <1.0> | First look at analysis and design | Victor Pădurean |
| <30/03/20> | <1.5> | Adding class diagram | Victor Padurean |
|  |  |  |  |
|  |  |  |  |

Table of Contents

I. Project Specification 4

II. Elaboration – Iteration 1.1 4

1. Domain Model 4

2. Architectural Design 4

2.1 Conceptual Architecture 4

2.2 Package Design 4

2.3 Component and Deployment Diagrams 4

III. Elaboration – Iteration 1.2 4

1. Design Model 4

1.1 Dynamic Behavior 4

1.2 Class Design 4

2. Data Model 4

3. Unit Testing 4

IV. Elaboration – Iteration 2 4

1. Architectural Design Refinement 4

2. Design Model Refinement 4

V. Construction and Transition 5

1. System Testing 5

2. Future improvements 5

VI. Bibliography 5

# Project Specification

Design and implement a desktop application for Veterinary Lab. The application should have two types of users (a regular user - the doctor/employee - and an administrator user) which have to provide a username and a password in order to use the application.

The regular user can perform the following operations:

* CRUD on consultations
* CRUD on animals involved in consultations
* Search the list of consultations/animals

The administrator can perform the following operations:

* CRUD on animals
* CRUD on regular users' information.
* Generate two types of report files, one in pdf format and one in txt or html format, with the consultation’s details. The reports \*\*need\*\* to be saved in a \*\*user-selected location\*\* (not predefined by the application), similar on how one would save a file from Notepad.

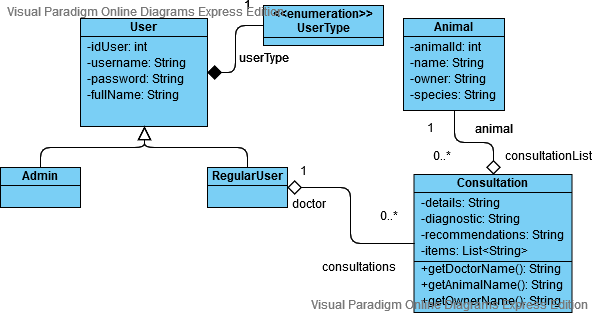
Other features:

* A consultation can have the state: in progress.
* For a consultation to be executed, a set of items needed can be allocated (e.g., needles, knife, food, bandages). A consultation that doesn't have all the items cannot be moved to in progress (and thus can't be closed).
* The administrator can update the stock of items in the veterinary lab.

# Elaboration – Iteration 1.1

# Domain Model

The domain model is shallow, as there are not many classes needed for describing what happens at a Veterinary Laboratory. Firstly, there will be animals, which will be consulted by vets. The consultations details will be persisted. Besides the vets, there will be another type of person interacting with the system, responsible for organizing it: the administrator. The relationships between the classes is presented in the next diagram.



# Architectural Design

## Conceptual Architecture

The system will adopt a Client-Server architectural style, this being the most appropriate approach for such applications. The data will reside on the Server side, whereas the Client side will need to request it. Most of the services will be handled by the Server, yet some may need to be handled by the Client.

The adopted architectural pattern will be represented by a Three Tier Architecture. Even though a Model-View-Controller architecture is closely related to this, Three Tier Architecture brings a clear separation between the data layer and the presentation layer. The MVC architecture pattern will be used to implement the Presentation Layer.

## Package Design

*A close up of a map

Description automatically generated*

## Component and Deployment Diagrams

A close up of a sign

Description automatically generated

A screenshot of a cell phone

Description automatically generated

# Elaboration – Iteration 1.2

# Design Model

## Dynamic Behavior

Communication and Sequence diagrams for Consultation Creation

A screenshot of a map

Description automatically generated

## 

A screenshot of a cell phone

Description automatically generated

Communication and Sequence diagrams for Report Creation

A picture containing screenshot

Description automatically generated

A screenshot of a cell phone

Description automatically generated

## Class Design

A screenshot of a cell phone

Description automatically generated

# Data Model

A screenshot of a cell phone

Description automatically generated

# Unit Testing

*[Present the used testing methods and the associated test case scenarios.]*

# Elaboration – Iteration 2

# Architectural Design Refinement

*[Refine the architectural design: conceptual architecture, package design (consider package design principles), component and deployment diagrams. Motivate the changes that have been made.]*

# Design Model Refinement

## *[Refine the UML class diagram by applying class design principles and GRASP; motivate your choices. Deliver the updated class diagrams.]*

# Construction and Transition

# System Testing

After integrating the units, the system should work as expected, given the use-case diagram. Test cases can be designed based on use-cases. The system can be tested using the built-in user interface.

Test cases examples:

* Login – the user enters the login details accordingly; if erroneous data is introduced or the username and password mismatch, they shall be notified, else they continue to the next page, according to the user type
* Create a user – only an admin user can do this, if logged in; the corresponding data is introduced; if erroneous data is present, the user shall be notified; otherwise, the new user will be added to the database
* Update a user – only an admin user can do this, if logged in; the corresponding (which needs to be changed) data is introduced; if erroneous data is present, the user shall be notified; otherwise, the user’s data will be changed in the database
* Delete a user - only an admin user can do this, if logged in; the corresponding user is selected and the “delete” action is initiated; the deleted entry is no longer present in the database (or is still in the database with the STATUS changed to ‘deleted’)
* Inspect a user - only an admin user can do this, if logged in; the corresponding user is selected and the “inspect” action is initiated; the corresponding user details are shown
* Create an animal – similar to “Create a user” test case
* Update an animal – similar to “Update a user” test case; can be done by a regular user as well
* Delete an animal – similar to “Delete a user” test case
* Inspect an animal – similar to “Inspect a user” test case; can be done by a regular user as well
* Create a consultation – can be done by both types of users; the corresponding animal and regular user are selected from a list of existing ones; the needed details are completed; a new consultation is inserted in the database
* Update a consultation – can be done by the regular user; the desired consultation is selected, and the corresponding data is modified; the database should be updated
* Generate a report – can be done by an admin; the desired consultation is selected, the “Generate Report” (as .pdf or .txt) is pressed, and a popup menu for selecting the path for the new file will become available; the file will be created with the corresponding consultation details

# Future improvements

There is room for many improvements. One major improvement would be a deeper separation of responsibilities, a new type of user being needed: the assistant, which will be responsible for scheduling consultations and generating reports.

Other improvements include the integration of a medicine class, adding more details for each doctor and animal, even by using photos.

# Bibliography