

# Team 21 Design Document

**Project Name:** STABLE

**Team Members:** Adam Baker, Pedro Del Moral Lopez, Shantanu Nair, Roy Ramstad

## Table of Contents

Title	Page
1.0 PURPOSE	2
2.0 DESIGN OUTLINE	2
2.1 High Level Overview of System	2
2.2 Design Pattern	2
3.0 DESIGN ISSUES	4
3.1 Functional Design issues	4
3.2 Non-Functional Design Issues	5
4.0 DESIGN DETAILS	6
4.1 Class Diagram	6
4.2 Description of Class and Models	7
4.3 Sequence of User Interactions Diagram	8
4.4 Activity Diagram	9
4.5 UI Mockups	10

## 1.0 Purpose

---

We will design and develop a green turf web application for use by veterinarians to track requisite data such as treatment records, bio-data, and training regiments for individual animals in their care, specifically race horses. This software will replace the current method of tracking such data which is manually on paper, and will provide the added functionality of generating paper reports and graphs of relevant data. The data that needs to be maintained for a race horse includes: gender, breed, weight, size, sire, mother, vaccination and medical history, training regimen, auction price, and requisite expenditures in order to rear the horse. This software will be able to keep all of this information neatly displayed on one page, instead of multiple stacks of papers on separate information. Furthermore this software will be able to generate paper reports and graphs to visually display the information kept in the software, which previously had to be done by hand. The closest piece of software to this is Microsoft Excel. This software is preferable to Microsoft Excel because it is specialized for horses, which removes a lot of unnecessary clutter and unused functionality from your basic “graphing” program, and this software will be able to communicate with a phone to provide reminders on important events such as vaccinations, training days and auctions, which Microsoft Excel cannot do.

## 2.0 Design Outline

---

### 2.1 High Level Overview of Our System

This project will have a single client connecting to a single personal server. The client will send requests to the server whenever they wish to update or retrieve information on a particular horse, vaccine or training program. The requests can come from the personal computer of the user or via phone connection. When a specific request is received, the server will update the information displayed on the software front end to account for the request.

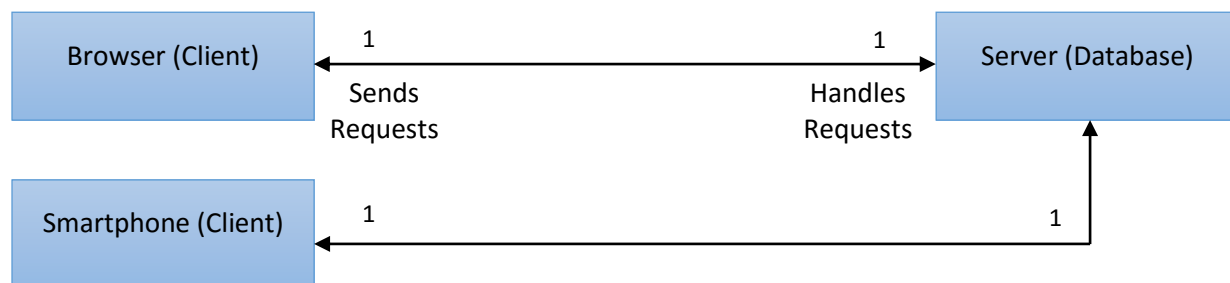


Figure 2.1 High Level Overview of System

### 2.2 Design Pattern

Our design pattern would be a client-server model where the client provides a visual framework for information that will be queried to the server to provide. The server will

solely be required to provide the information needed by the client side, and will not be concerned with providing the visual output to the user. The server will only connect to the local user, since the server will not be part of a larger network and no functionality is added by providing multiple user connections.

## Browser

The client's view will be within a browser and will be implemented using JavaScript and Angular. The user interface will provide an easy-to-use and intuitive method to track information, and will enable them to modify and display information queried from the database.

## Server

The server will control the client's view by controlling the flow of data from the client to the database in order to keep the information being displayed up to date. The server will handle all queries to the database, and query responses from the database. The server will be hosted on the same computer as the browser is open, since this will eliminate the need for an outside connection, and keep the data secure by keeping it within the intranet of the user.

## Database

The database will hold all the persistent data for the software, which will include all the horse metadata (in this case metadata being defined as breed, age, gender, auction price, sire, etc.), all vaccine metadata (in this case metadata being defined as price, provider, medical purpose, medical schedule, etc.), and training regimens.

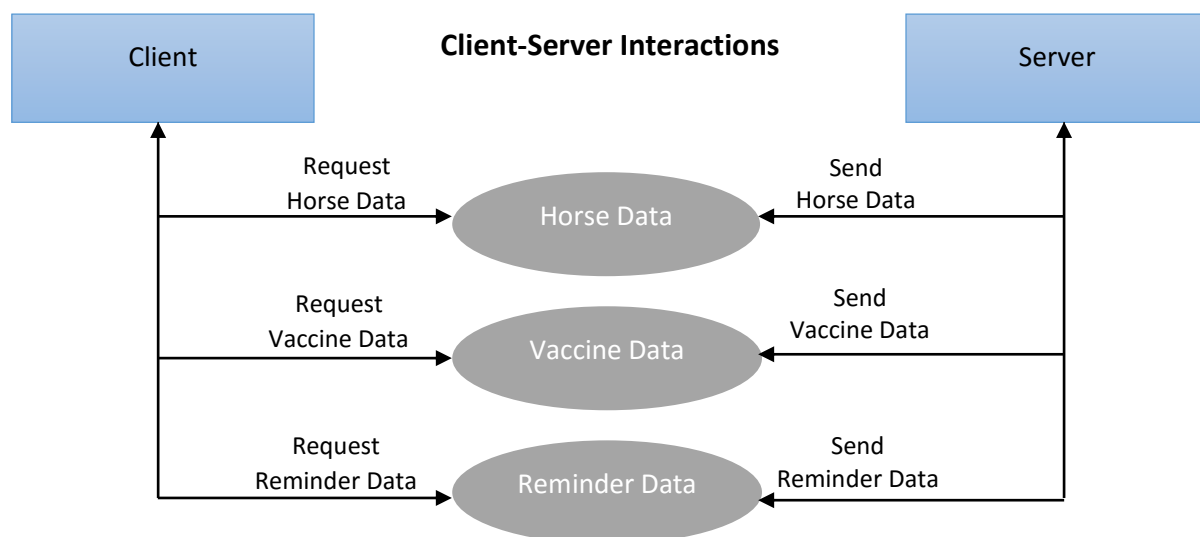


Figure 2.2 Client-Server Interactions

## 3.0 Design Issues

### 3.1 Functional Design Issues

---

1. How do we want to implement the calendar functionality?
  - a. **Option 1:** Traditional Calendar Style
  - b. **Option 2:** Radial Calendar Style
  - c. **Option 3:** Both
  - d. **Decision:** The traditional calendar would be easier to implement as there is an abundance of online resources. The radial calendar is what horse veterinarians are already used to and better represents the type of data. Our decision would be to implement both types of calendars, which would satisfy both types of users.
2. How do we want to implement user security?
  - a. **Option 1:** With an emphasis on multiple users logging into the same machine
  - b. **Option 2:** Singular user
  - c. **Decision:** We chose singular because we don't anticipate multiple veterinarians being in charge of the same horses. The software will be used individually and not as a part of a larger network so there would be no reason for multiple logins. The information the software would track is sensitive and has to be controlled via a login instead of simply launching the application.
3. How do we want to implement the exporting of information?
  - a. **Option 1:** Exporting individual data separately
  - b. **Option 2:** Adding individual data to a queue to be exported
  - c. **Decision:** Option 1 is quicker for the user and requires less navigation of the software to export the data you want to export. However, Option 1 is less convenient if you want to export larger amounts of metadata. Option 2 is slower when you want to export larger quantities because it requires more navigation of the software. However, it is more efficient when you want to export larger amounts of metadata, so we chose Option 2.
4. How do we want structure the database?
  - a. **Option 1:** Relational database
  - b. **Option 2:** Non-relational database
  - c. **Decision:** Option 1 is more popular and well-known, along with being easier to comprehend, implement and maintain. Furthermore, the data we want to keep lends itself better to a relational database as opposed to non-relational database.

## 3.2 Non-Functional Design Issues

---

5. How do we want to structure the software to be user-friendly and familiar?
  - a. **Option 1:** Try and replicate other non-veterinary organizational software.
  - b. **Option 2:** Try and replicate the look and feel of already existing organizational paperwork and calendars used by veterinarians.
  - c. **Decision:** Option 2 is more immediately preferable since it would allow the users to immediately recognize and use the software the same way they would immediately recognize the paperwork as opposed to requiring an acclimation to a different organization of information.
6. How do we gauge software performance in the real world?
  - a. **Option 1:** Using existing analytic solutions such as Google analytics.
  - b. **Option 2:** Providing a feedback button or way to report issues.
  - c. **Decision:** Option 2 is more immediately implementable and can be added without slowing down the implementation of the rest of the software. Option 1 would require that we first become familiar with using and integrating analytic systems into our software, but can be added silently at a later date without affecting the user experience.
7. Where should we keep our data?
  - a. **Option 1:** On a separate dedicated database computer.
  - b. **Option 2:** On the client computer.
  - c. **Decision:** Option 2. Since we chose to structure the software so that the client serves as his own web-server, there is no reason to store the data on another computer since that would require that the client have a second computer or that we set up a network between a computer of ours and the client which is also unnecessary and subject to internet outages amongst other issues. The client's data will stay on the client's computer, which will aid in security and prevent internet issues (which could be common given the location of most horse ranches in the countryside) from interfering with the client's ability to access the information.
8. What web framework or platform do we plan on using?
  - a. **Option 1:** Angular
  - b. **Option 2:** Meteor
  - c. **Decision:** Option 1, we chose angular because we have more immediate reference material and resources for learning to use it, along with Angular being more mainstream within the professional workplace, which would provide us with work-applicable experience.

## 4.0 Design Details

### 4.1 Class Diagram

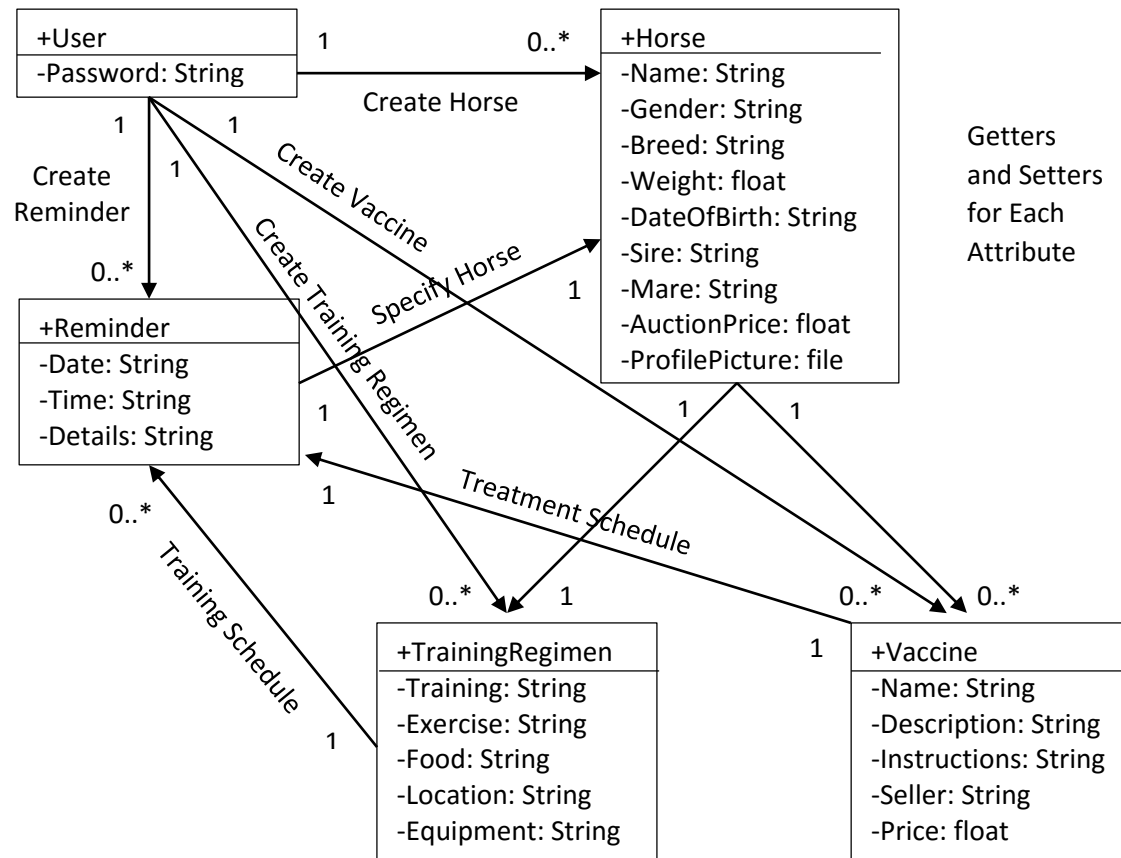


Figure 4.1 Class diagram.

## 4.2 Description of classes and models

### Horse

- This contains the biological and medical data of a horse, along with some logistical data such as pictures of the horse, auction price and name.
- There can be many instances of this class instantiated.

### User

- This contains the credentials of the user and encompasses the veterinarian of the horses, who should be the only one managing this information.
- There should only ever be one user for every installed copy of the software in order to act as authentication to protect the data that is tracked inside.

### Reminder

- This contains a simple plain text reminder of an event scheduled for that day such as applying Vaccine Y to Horse X, along with containing the date and time for the reminder.
- This may be sent to a phone if it is registered to the software.
- There can be multiple instances of the reminder class instantiated.

### Vaccine

- This contains all the medical and logistical data for an individual vaccine such as price, provider, purpose, and schedule for administering the vaccine. This is used by the horse class in keeping track of the horse medical data.
- There can be multiple instances of the vaccine class instantiated.

### Training Regimen

- This contains a list of exercise routines, necessary equipment, related costs and other logistical data such as time and place for training a horse.
- There can be multiple instances of the training regimen class instantiated, for different training regimens.
- Each horse will be associated with a training regimen.

### 4.3 Sequence of User Interactions Diagram

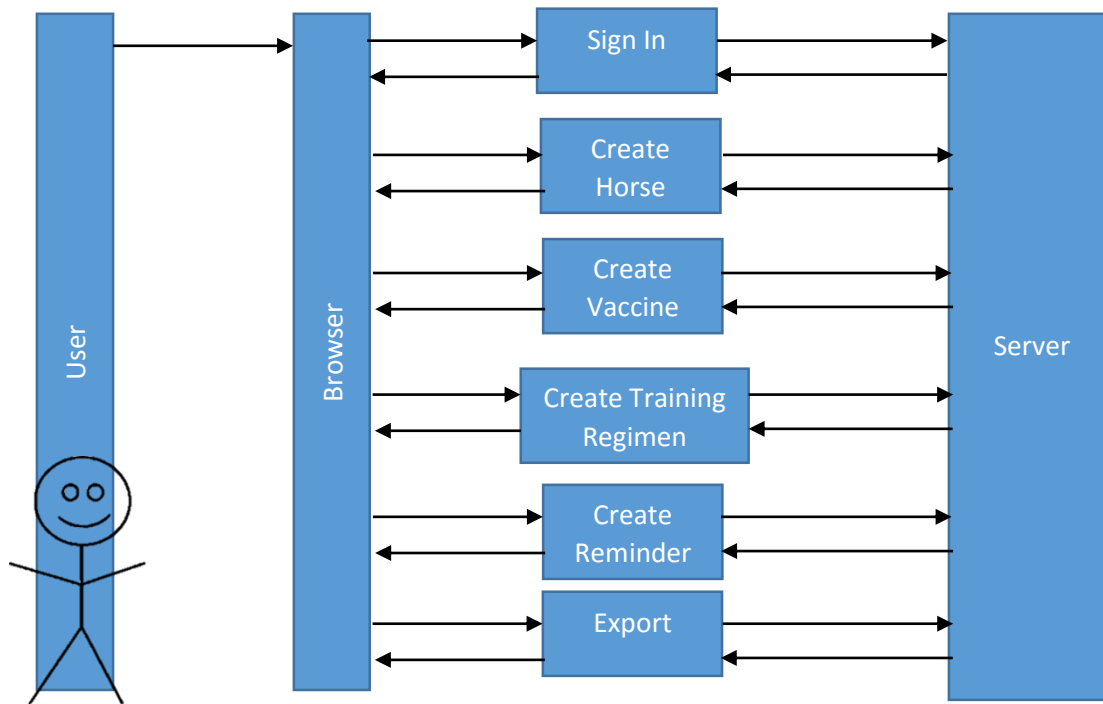


Figure 4.3 Sequence diagram.



#### 4.4 Activity Diagram

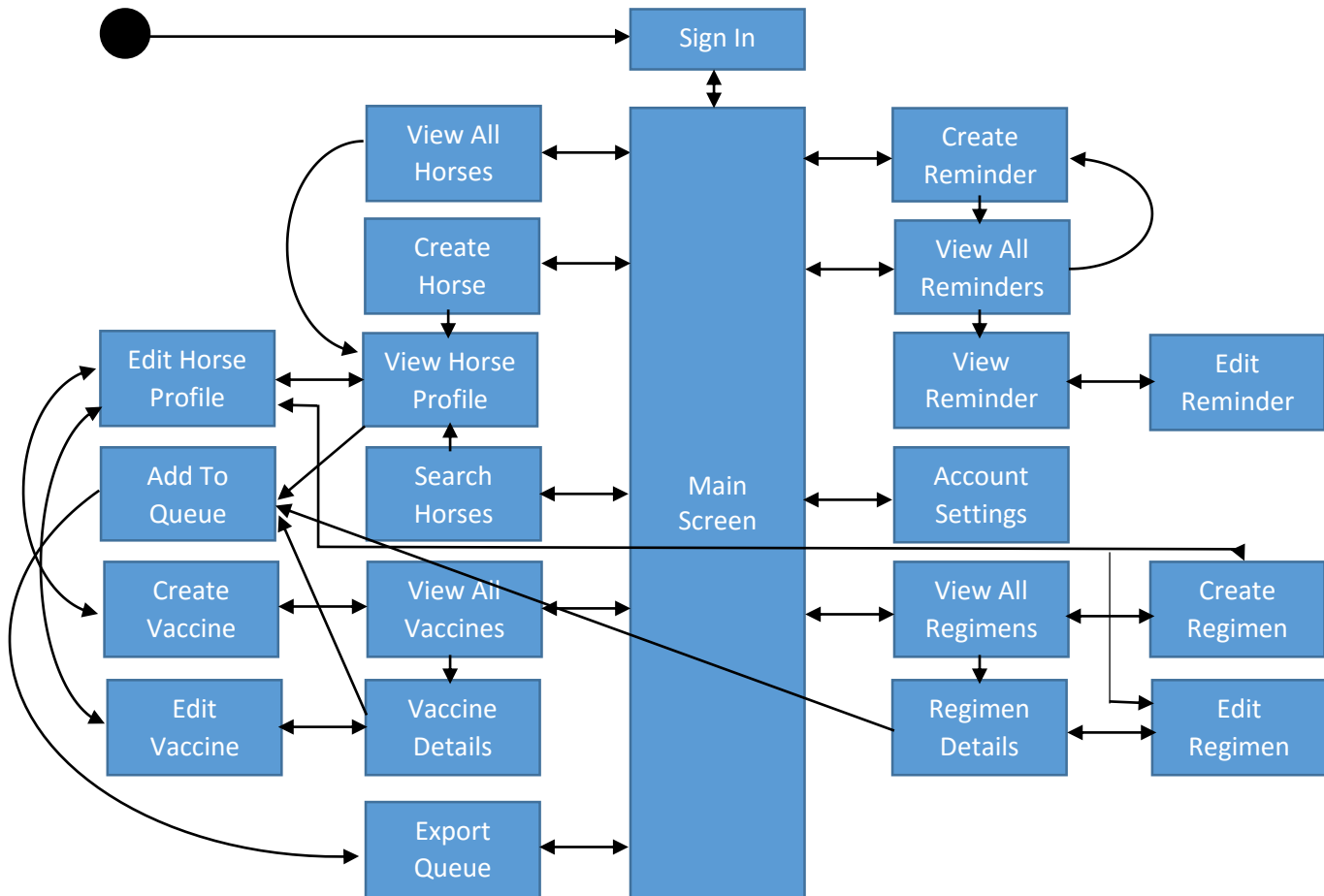


Figure 4.4 Activity diagram.

## 4.5 UI Mockups

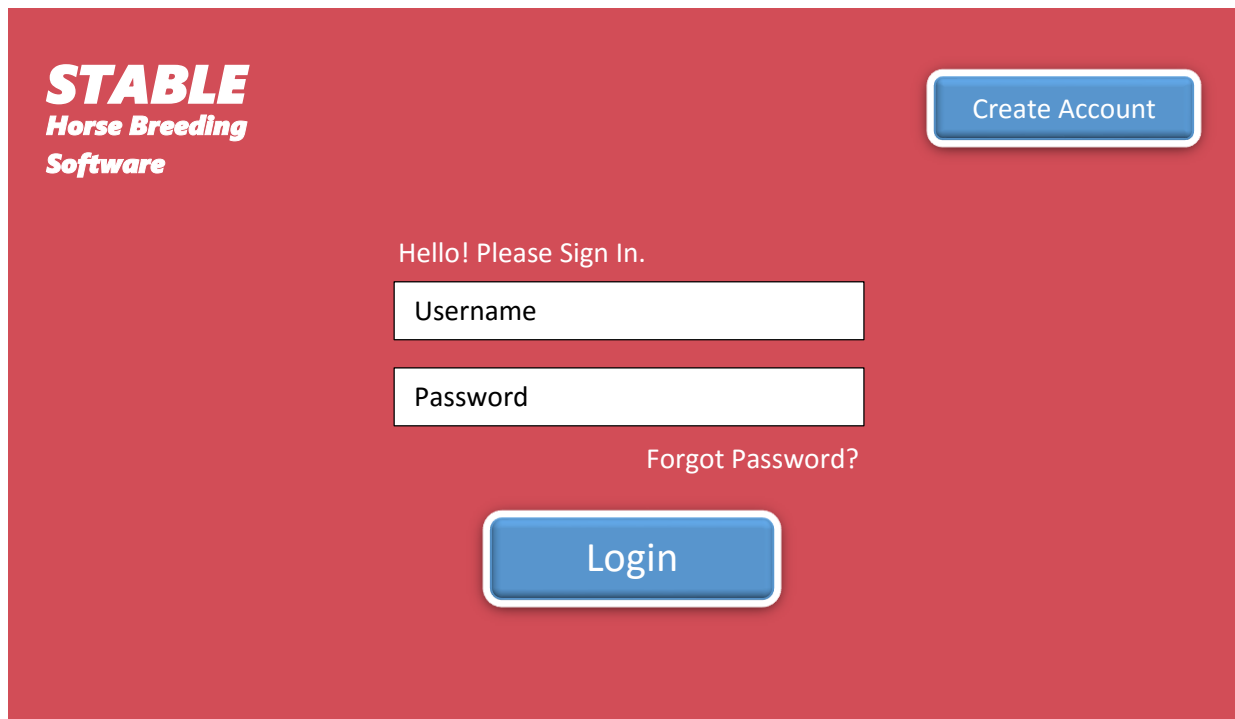


Figure 4.5.1 Sign-in screen mockup.

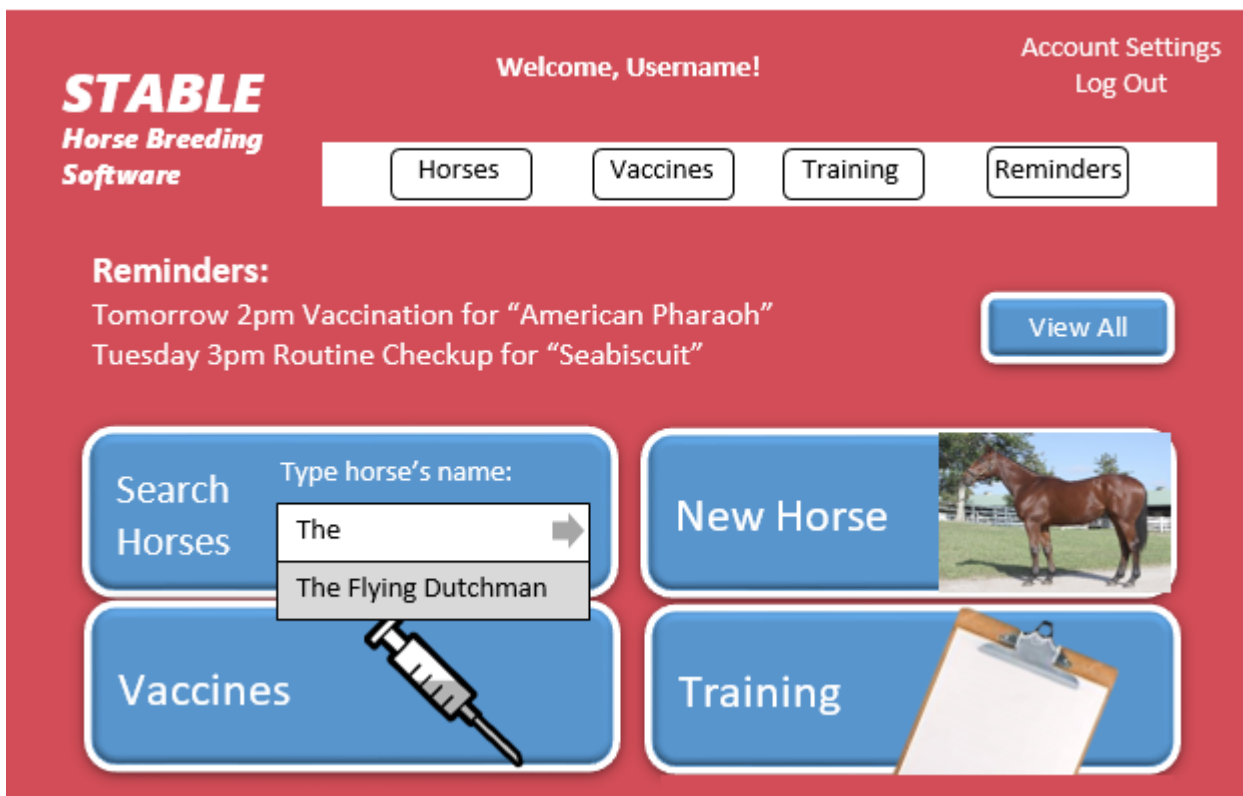


Figure 4.5.2 Main dashboard.

**STABLE**  
Horse Breeding  
Software

Enter Horse Information:

Name
Gender
Breed
Weight
Date of Birth
Sire
Mare
Auction Price

+ Add Vaccination  
+ Add Training Regimen  
+ Attach Picture

Back Submit

Figure 4.5.3 Horse information form.

**STABLE**  
Horse Breeding  
Software


☐ Deactivate Profile

**"The Flying Dutchman"**

Gender: Male  
Breed: Thoroughbred  
Weight: 120 lb  
Date of Birth: Feb 27, 2012  
Sire: Bay Middleton  
Mare: Barbelle  
Auction Price: \$6,000  
Vaccinations: Fluvac | Encevac

Back Edit Info


Remove Horse

Training Regimen


Figure 4.5.4 Horse profile.

**STABLE**  
Horse Breeding  
Software

## Important Events

**MAY 2016**


SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
1	2 2pm Vaccination "Big Brown"	3	4	5	6	7
8	9	10	<div> <b>May 2, 2016</b>            2pm – Vaccination for            "Big Brown"         </div> <div> <input type="button" value="Remove"/> <input type="button" value="Edit"/> </div>		13	14
15	16	17			20	21
22	23	24			27	28
29	30	31				

Figure 4.5.5 Reminder calendar. Note: This is one of two calendar formats that will be implemented. The other format can be found in Figure 4.5.8

Enter Vaccine Information:

Name

Description

Instructions

Seller

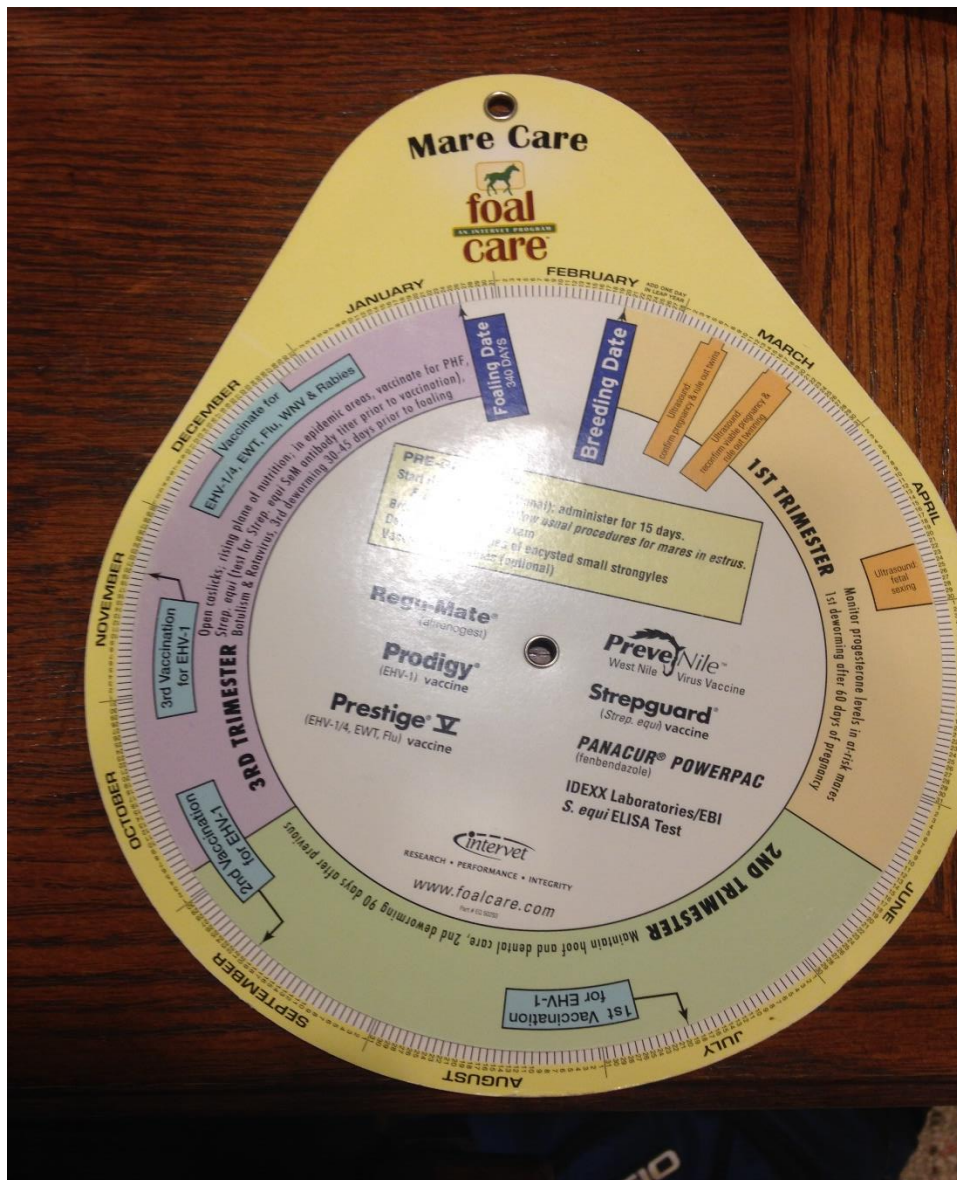
Price

Enter Training Information:

Training
Exercise
Food
Location
Equipment

Figure 4.5.6 Vaccination form.

Figure 4.5.7 Training regimen form.



*Figure 4.5.8 Radial calendar format*