Parkview Pet Hospital: Veterinary Practice Management System Development

Ian T. Wiedenman
Loyola Marymount University, ian.wiedenman@gmail.com

Follow this and additional works at: https://digitalcommons.lmu.edu/honors-thesis

Part of the Business Administration, Management, and Operations Commons, and the Management Information Systems Commons

Recommended Citation
https://digitalcommons.lmu.edu/honors-thesis/169

This Honors Thesis is brought to you for free and open access by the Honors Program at Digital Commons @ Loyola Marymount University and Loyola Law School. It has been accepted for inclusion in Honors Thesis by an authorized administrator of Digital Commons@Loyola Marymount University and Loyola Law School. For more information, please contact digitalcommons@lmu.edu.
Parkview Pet Hospital: Veterinary Practice Management System Development

A thesis submitted in partial satisfaction
of the requirements of the University Honors Program
of Loyola Marymount University

by

Ian Wiedenman

May 3rd, 2017
Introduction

Background
We are Ian Wiedenmann and Taylor Kozakar, AIMS Seniors. Prior to the beginning of this project, we both had an interest in healthcare IT systems, as well as database and server development and deployment in a professional environment. As such, our project for Parkview Pet Hospital was an intriguing and stimulating project.

Parkview Pet Hospital is a small but full-service animal hospital that deals with routine checkup for dogs and cats as well as emergency cases. Parkview Pet hospital is a comfortable, kid-friendly, and calm environment. They were established in 2009 and currently have one veterinarian, Dr. Gamble, and 3-4 technicians on full-time staff fulfilling a number of roles, including administration, technical expertise, nursing, and various other defined roles.

The Primary Problem Statement
How does Parkview Pet Hospital maintain an operational, enterprise RDBMS with accompanying front-end, while keeping costs at a minimum and maintaining ease of use?

Exploration of the Problem Statement and Enterprise Requirements
Parkview Pet Hospital is continuously searching for new deployments in their information technology architecture. As such, the organization needs a centralized system for recording and tracking key business processes and the data/metadata they create, as well as storing data regarding the company's inventory, employees, clients, client's pets, and critical organizational actions. The problem, as it stands, is that the hospital lacks a robust system for storage of data, as well as templates for data entry, and reporting services that reliably produce the properly formatted data that serves both employees and clients alike. Parkview Pet Hospital has used many practice management softwares in the past but the cons often outweigh the pros.

Therefore, there is a need for an easy-to-use, low-cost/free, open-source database system for operational enterprise data storage, as well as a dependable website-hosting system and accompanying database. Finally, there is a need for a well-designed, modern website that allows for easier customer interaction, as well as a portal for staff to quickly execute queries on patient and client records.
The Development Process

Initial Steps

In the early process, we utilized a combination of Kimball data discovery techniques and traditional approaches to data discovery/requirements gathering. We interviewed users of the
data and systems to determine needs and gaps, as well as questions regarding the general nature of the enterprise, as well as their future plans and what they could need in the future. As such, this information was ascertained through: phone/in-person interviews with primary physician and hospital administrator, in-person discussion with IT staff member, brief discussions with clients regarding the website interface, and analysis of current, high-cost system (FileMaker).

**Early Database Development Phase**

Using the information we gathered through the interviews, we first constructed an ER model that best fit the needs of the organization. See the attached ER diagram.

After settling on a general schema structure, we began developing the necessary fields and data types required for accurately storing organizational and client data. See the attached field names.

Initially, we used these table structures and field name delineations to build out a an instance of MySQL Server (on an InnoDB database engine) hosted locally on our MacBook PRO OSX laptop. MySQL Workbench was the RDBMS software utilized to build out and manage this database, as well as input sample data. Accessing the localhost instance of MySQL Server at 127.0.0.1 (localhost) with UNIX socket port of 3306 was the primary means of linking the two pieces of software. This was implemented with the SQL language, on top of the MySQL Workbench platform.

**Early Web Development Phase**

Since this project requires a front-end for end-users, we designed a sitemap that delineated all of the pages of our site in a hierarchical formation. See attached sitemap.

Next, we used the sitemap as a guide to create a storyboard for the website which shows a rough outline of what the pages will look like as well as their functionality. See attached storyboard slides.

After developing the blueprint materials for the web development phase, we used a bootstrap template by the name of AdminLTE to kickstart out front-end development. Having a nicely formatted wireframe with built-in grid systems allowed us to streamline our backend development and fit our code into the front-end site, and helped us concentrate more of our time and effort into database development, PHP programming, SQL script deployment, and architectural configuration.

This phase was implemented with HTML/CSS, Javascript, jQuery, PHP, and the AdminLTE bootstrap templates.

**The MAMP Phase**
At this point, we had a working instance of MySQL Server and a functioning website, but no adequate way of having the two platforms interact or exchange data. As such, we realized we needed to implement some form of webserver software (most likely Apache), as well as extensive PHP code to facilitate interaction. Instead of writing a large amount of middleware in C and cranking out Unix BASH scripts, we decided to install and use MAMP Pro, a webserver virtualization platform that integrates Apache Webserver, MySQL Server, PHP/Python/Perl, and the Mac OSX operating system environment. See below for configuration screenshots that demonstrate the necessary settings for running a virtualized server environment.

Prototype Implementation

Hardware/OS configurations:
MySQL Workbench: Connection
MySQL Workbench: Configuration of Hospital Table

MySQL Workbench: Configuration of Employee Foreign Keys
MySQL Workbench: Editing Records in the Veterinarian-Pet Table

MySQL Workbench: Basic query of LabTest table:
MySQL Workbench: Advanced query of PetOwner and Pet tables:

MAMP Pro: Port Configurations
MAMP Pro: General Settings

MAMP Pro: Apache configurations
MAMP Pro: MySQL Server configurations

Full Stack Demonstration
A regular user would use the application primarily as a source of information. These users would be clients of Parkview Pet Hospital. After navigating to the website, public users have access to 6 pages: Home, Overview & Services, Mission, Gallery, Staff, as well as News & Events. Below are examples of some of these pages. Home, for example, displays the most basic information about the clinic. There are a number of images provided in a carousel, and the hospital contact information is listed to the right. Below these items is a quick description of Parkview Pet Hospital.

The tabs titled Overview, Mission, and Gallery are very similar in that they are purely static information. The Staff tab gives information about the staff members at Parkview to the clients, but this tab pulls directly from the database. Shown below is a screenshot of this page.
This is accomplished by running a query that selects all records from the Employee table and then loops through each record to echo each relevant field into HTML bootstrap cards. The
A staff member would use the application primarily to interact with and update information about clients. Staff members can visit all of the public pages as a client would, but
they can access an additional page titled “Staff Portal” after logging in. Before accessing this page, staff members must first log in. Below is the process for reaching this page.

First, a staff member clicks on the tab titled “Staff Portal”. They are brought to a confirmation screen where they are required to sign in. This page is listed below.
After a successful sign-in, staff members are brought to the staff portal. The prototype of this application provides two functions: patient lookup and client lookup. Clicking patient lookup brings users to an interface where they can search the database using a form to look-up patients. This page gives employees access to relevant patient information. Entering a '*' into the search field and pressing “Go!” will pull all records from the database. This is shown below:

![Staff Portal Image](image-url)

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Weight</th>
<th>Length</th>
<th>Height</th>
<th>Eye Color</th>
<th>Color</th>
<th>Species</th>
<th>Sub-Species</th>
<th>Primary Allergy</th>
<th>Owner Name</th>
<th>Insurance Provider</th>
<th>Pet Owner Dell</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Abba</td>
<td>90</td>
<td>105</td>
<td>38</td>
<td>Green</td>
<td>Yellow</td>
<td>Dog</td>
<td>Poodle</td>
<td>Penicillin</td>
<td>Peter Griffin</td>
<td>HealthyPaws</td>
<td>2012-02-02</td>
</tr>
<tr>
<td>2</td>
<td>Fido</td>
<td>50</td>
<td>15</td>
<td>18</td>
<td>Black</td>
<td>Blue</td>
<td>Muskot</td>
<td>Cylpimuskot</td>
<td>N/A</td>
<td>Lisa Griffin</td>
<td>HealthyPaws</td>
<td>2013-02-01</td>
</tr>
<tr>
<td>3</td>
<td>Eddy</td>
<td>180</td>
<td>19</td>
<td>280</td>
<td>Green</td>
<td>Black</td>
<td>Dog</td>
<td>Husky</td>
<td>Penicillin</td>
<td>Lisa Griffin</td>
<td>HealthyPaws</td>
<td>2013-02-01</td>
</tr>
</tbody>
</table>
Entering a specific name of a patient will yield all records matching the query. For example, entering “Abba” and pressing “Go!” will yield one record of a patient with that name. This is shown below:

To accomplish this, we wrote an if statement that determines whether to run the query that selects all fields or selects only the fields where the Pet_Name is equal to $q, which is set to the value of the text box near the Go button.

```php
28  mysqli_select_db($con,"ajax_demo");
29  if ($q == "*"):
30        $sql="SELECT * FROM Pet";
31  else:
32        $sql="SELECT * FROM Pet WHERE Pet_Name = ".$.q."";
33  endif;
34  $result = mysqli_query($con,$sql);
```

The results of the query are echoed to a table that is displayed as HTML.
By navigating to the second tab titled “Client Lookup”, staff members will be able to perform the same search functions on clients as opposed to patients. Entering a ‘*’ and pressing “Go!” will yield all records in the client table as shown below:

<table>
<thead>
<tr>
<th>Name</th>
<th>DOB</th>
<th>Address</th>
<th>City</th>
<th>State</th>
<th>Zip Code</th>
<th>Preferred Language</th>
<th>Email Address</th>
<th>Phone Number</th>
<th>Active Client</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gregor Spencer</td>
<td>1957-09-31</td>
<td>4632 W. 5th St.</td>
<td>Moline</td>
<td>Illinois</td>
<td>61265</td>
<td>English</td>
<td><a href="mailto:gspence@gmail.com">gspence@gmail.com</a></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lawrence Pekka</td>
<td>1966-12-03</td>
<td>36 Walnut Way</td>
<td>Moline</td>
<td>Illinois</td>
<td>61265</td>
<td>Polish</td>
<td><a href="mailto:lpkee@facebook.com">lpkee@facebook.com</a></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lois Griffin</td>
<td>1982-12-32</td>
<td>24 Third St.</td>
<td>Moline</td>
<td>Illinois</td>
<td>61265</td>
<td>English</td>
<td><a href="mailto:lgriffin@gmail.com">lgriffin@gmail.com</a></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Matt Reimert</td>
<td>1987-11-01</td>
<td>208 Rocking Lane</td>
<td>Moline</td>
<td>Illinois</td>
<td>61265</td>
<td>English</td>
<td><a href="mailto:mreimert@yahoo.com">mreimert@yahoo.com</a></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Peter Griffin</td>
<td>2012-03-23</td>
<td>4541 65th St.</td>
<td>Moline</td>
<td>Illinois</td>
<td>61265</td>
<td>English</td>
<td><a href="mailto:prgriffin@gmail.com">prgriffin@gmail.com</a></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Entering a specific name such as “Gregor Spencer” will yield one record as shown below.

To accomplish this, we wrote an if statement that determines whether to run the query that selects all fields or selects only the fields where the Pet_Name is equal to $q, which is set to the value of the text box near the Go button.

```php
28   mysqli_select_db($con,"ajax_demo");
29   if ($q == "*"):
30       $sql="SELECT * FROM PetOwner";
31   else:
32       $sql="SELECT * FROM PetOwner WHERE PetOwner_Name = \\
33                  \'".$q."\'";
34   endif;
35   $result = mysqli_query($con,$sql);
```

The results of the query are echoed to a table.
Finally, staff members will have the ability to use the built-in form function in MySQL Workbench to enter new information into the database and update existing information. For instance, a staff member can add a new employee to the database, as shown below. Filling out the form and clicking submit will create a new record in the Employee table.

**Conclusion**
What we accomplished

As a team, we successfully developed a functioning instance of the MySQL Server with a working database, using the InnoDB engine, along with several foreign keys, indexes, and queries utilized to show the interaction of data within the system. This database fulfilled many of the initial requests made by the organizational administration, as well as general characteristics that would allow it to be used generally for businesses that are not necessarily Parkview.

Next, we successfully designed and implemented a local deployment of our website, in a proof-of-concept form, with the Admin LTE platform. We also successfully secured the staff portal login that allows for direct interaction with enterprise data.

After developing both platforms independent of one another, we managed to install and run an instance of the MAMP Pro software, regenerating the previous MySQL Server instance into the integrated MAMP stack version, and integrating our HTML/CSS webpages with the database using the built application sockets and self-written PHP code to facilitate data transfer.

Finally, we successfully integrated out MAMP Pro with our organizational GitHub repository, allowing for remote changes and management of pages, and SQL/PHP scripts without needing direct access to the localhost machine.

Issues Faced

We both suffered from a lack of prior, deep experience with the general (X)AMP-stack architectures for web-hosting. Configuration of our MAMP (Mac OSX, Apache Server, MySQL, PHP) stack was incredibly complex; it presented us with a lot of trial-and-error and database table corruption. In this same vein, we both had low knowledge of programming PHP scripts. MySQL, specifically the InnoDB engine, is error-prone and small mistakes in the creation of constraints, indexes, etc., can require a complete re-deployment of the database. DML is also less easy to set up than Microsoft products (Access, SQL Server). There were limitations of the local machine (Taylor’s 2011 MacBook Pro), and before integration of programming efforts through Github Desktop, difficulty in version control and dynamically testing deployments of new code and configurations.

Further Work To Be Done

Firstly, a transition of (X)AMP-stack architecture to a preferably locally-based server, with dedicated software and hardware. We both have vague experience with this, although we simply ran out of time to implement and test this system. Next, development of multiple databases for varying purposes. An operational database should remain independent of a database utilized for much of the daily functioning of the website, such as logins, passwords, web-data storage, etc. After that, integration of Wordpress Engine for better site/page administration. We experienced some strange issues in PHP deployment of this software, and had to move on to more critical requirements. Finally, Analytics/BI services would be ideal, once
there was an adequate amount of data stored in the system to warrant it. Ideally we could deploy this on a dedicated 'application server', preferably with Tableau, PyCharm/Tensorflow, Excel (with integrated Solver software) and other analytics software.