CS Capstone Final Report

June 9, 2018

ConnectBasket

Prepared for

OSU MIME

Dr. Chinweike Eseonu

Prepared by

Group 39

ConnectBasket Development Team

Henry Fowler

Kailyn Hellwege

Taylor Kirkpatrick

Abstract

This document gives a detailed description of the ConnectBasket project. All requirements and the design of the project will be given as well as any changes that ended up being made during development of the project. The decisions for what technologies to use for the project will also be given along with important documents related to the project such as the poster and instructional documentation.

Contents

1 Introduction to Project

1.1 Project Purpose

This project will create a web application called ConnectBasket that will serve as a new tool for the Oregon State University Veterinary Hospital to improve communication within the hospital and externally with patient owners and other hospitals. The current workflow in the hospital involves receptionists taking calls, typing a message into the system, printing out that message, and walking it across the hospital to a veterinary technician or doctor to look at and add a handwritten note to. After that, the doctor or technician will carry the message to another doctor or technician, or back to reception or scheduling, who will call the initial caller and answer the question they had or schedule an appointment for them. The main purpose of this project is to eliminate the paper aspects of this process and make an electronic system that allows messages to be created and sent to other employees of the hospital, and notes to be added to those messages. It will then be able to display messages back to users with all of their associated notes. This project will help increase the efficiency of communication at the hospital as well as provide a less error-prone system that allows users to see a complete history of all messages.

1.2 Project Goals

- Allow users to create a message which will be associated with a patient and an owner of that patient
- Allow messages to have a status to tell whether they are open or closed
- Allow messages to be routed to other users or groups of users
- Allow notes to be added to messages
- Allow users to select a group or groups to be a part of
- Provide an audit log of all messages created over a given time period
- Increase the overall efficiency of messages being transported throughout the hospital
- Reduce the amount of paper being printed
- Reduce errors in the communication process of the hospital

1.3 Clients

The official client was Dr. Chinweike Eseonu, a professor in the College of Engineering at Oregon State University. The other client was Kelly Warner, the Process Improvement Manager at the Oregon State University Veterinary Hospital.

1.4 Development Team

All parts of the project were discussed and planned by all members of the team, but each team member had specific parts of the project they focused on. Most features of the website were a result of contribution from multiple team members.

1.4.1 Henry Fowler

Henry focused on the database and back end of the project. Designing and creating the tables and stored procedures stored in the MySQL database and writing the PHP code to interact with the database from the website.

1.4.2 Kailyn Hellwege

Kailyn was responsible for most of the front end development of the website. Writing the HTML and CSS that style all of the pages of the site and the Javascript code that makes the pages dynamic.

1.4.3 Taylor Kirkpatrick

Taylor designed and created the notification system for the website, including the shell script responsible for sending the notifications. He also worked on setup and configuration of the web server that the website is running on.

1.5 Role of the Clients

Dr. Eseonu provided some guidance and suggestions about overall design and features of the project. Kelly provided most of the input into what the website should look like and what features were needed.

2 Requirements Document

2.1 Introduction

2.1.1 Purpose

The purpose of this document is to outline the requirements for an improved communication system for the Oregon State University (OSU) Veterinary Hospital, called ConnectBasket. This document's audience is the client, as well as certain staff members at the hospital. This document will be used as a way to measure the success of the Capstone development team working on this project.

2.1.2 Scope

ConnectBasket will be a web based portal for the OSU Veterinary Hospital that will streamline the way messages are moved around the hospital and provide a better way to track the route that messages have taken. When an owner calls the hospital, receptionists will be able to type a message into the computer, assign it a category, and route it to the necessary staff members, which may be service specific vet techs, house officers, or faculty. Those staff members will be alerted, either by email or text, that they have a message waiting for them in ConnectBasket, and once viewed, notes can be added and the message can be re-routed to other staff members. After a message has been addressed by the necessary staff, the message can be closed, meaning no more notes can be added. The audit trail of when the message was created and closed will be visible, along with all the notes, timestamps of notes, and staff who added the notes.

2.1.3 Definitions, Acronyms, and Abbreviations

- Capstone - CS 461 class at Oregon State that the developers of ConnectBasket are in
- OSU - Oregon State University
- Hospital - referring to the OSU Veterinary Hospital
- Owner - a person who owns a pet that is a client of the hospital
- Patient - an animal who has/is going to be treated at the hospital
- VetHosp - the hospital management system used by the OSU Veterinary Hospital for keeping patient and owner records
- Mobile Devices - smartphones or tablets

2.1.4 References

IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications

2.1.5 Overview

The second section of this document will contain a further description of the ConnectBasket project, including product functions, user characteristics, and constraints. The third section will define the specific requirements of the project.

2.2 Overall Description

2.2.1 Product Perspective

The ConnectBasket web based portal will be a stand alone system that will have additional functionality to connect to the current VetHosp system used by the hospital. ConnectBasket will use patient and owner information stored in the VetHosp system to connect information collected with the correct patients and owners, given that access to the data in VetHosp is provided to the Capstone development team.

2.2.1.1 User Interfaces

There will need to be some level of security requiring users to login to access the information in the system. The system will provide users with the ability to search for a patient or owner and to enter new messages for them as well as view older messages and the audit trail associated with the message.

2.2.1.2 Hardware Interfaces

Staff will be able to access the system from their desktop or laptop computers as well as mobile devices.

2.2.1.3 Software Interfaces

ConnectBasket might be able to interface with the VetHosp program in order to connect information collected with the information about patients and owners stored in VetHosp. Access to the patient and owner information stored in VetHosp will be required for a system that could be integrated with VetHosp.

2.2.2 Product Functions

2.2.2.1 User Stories

The receptionists at the hospital will need to be able to take calls from owners, enter a message into ConnectBasket, and forward it on to necessary staff members, such as vet techs or doctors. After that, the staff receiving the message can perform the action requested of them, and add any notes to the message. Then, it can be rerouted to other staff members. At the end of the chain, the message will be routed back to scheduling or reception, where they can contact the owner with information or to set up an appointment. They will then mark the chain as closed, and it will be able to be viewed later by staff members with permission to view audit logs.

2.2.2.2 Function Dependencies

See Original Gantt Chart.

2.2.3 User Characteristics

Users of ConnectBasket will be staff members of the hospital. The users will have a variety of different education levels and experience. Therefore, the system will need to be designed so that technical expertise is not required in order to properly use it.

2.2.4 Constraints

The performance of the ConnectBasket application will be limited by a staff member's internet connection. Staff should not use the service on unsecure networks, as messages sent with it should have some degree of confidentiality due to the nature of the work, the hospital's policies, and government regulations. This limits where the application can be accessed from, as non-trusted networks may expose these messages. Staff may also only access this application with a verified user account.

2.2.5 Assumptions and Dependencies

It will be assumed that staff will not be attempting to use non-English languages for messaging. This system might depend on the hospital's existing VetHosp software, and integration with it is a goal for this project. Integration will depend on the VetHosp software being available to the developers and the level of access the Capstone development team will have to it. VetHosp integration will also depend on the hospital's IT team arranging an integration of ConnectBasket with VetHosp. Any changes that need to be made to the VetHosp system for integration will need to be done by the team in charge of developing VetHosp, not the Capstone development team. However, ConnectBasket will be capable of operating independently without any integration.

2.3 Specific Requirements

2.3.1 External interface requirements

2.3.1.1 VetHosp

ConnectBasket might be integrated with VetHosp as it nears completion, so that it can pull information from VetHosp as needed to populate fields and reduce separate instances of data lookup. Detailed information about VetHosp is not currently available to the development team, and successful integration will depend on the level of access provided to the Capstone development team.

2.3.2 System features

2.3.2.1 Creation of Messages

ConnectBasket will allow receptionists to create messages and route them to other parts of the hospital. Receptionists and other staff should be able to route messages with patient information to the relevant locations in the hospital. These messages should be secure and not exposed to public channels, and may need to be encrypted as well.

2.3.2.2 Message Auto Population

ConnectBasket will allow message composers to easily write messages by auto populating some fields of the message, such as the current date and patient and owner names. By auto filling and not allowing free text for certain fields, message creation will be more efficient and less error-prone. These fields will be prefilled for convenience and to send off messages quicker and in a standard format. This feature will also include utilizing drop-down menus instead of plain text to reduce error for other fields, such as recipients of messages and category of messages.

This feature will be useful for both convenience and safety, as constraining messages to at least some pre-defined standard will reduce the likelihood of human error in medical situations.

2.3.2.3 Message Notification

ConnectBasket will send message recipients notifications when a message has been routed to them or a category their profile belongs to. Notifications will be delivered as a text message, email, or both, depending on the preferences of the staff member. Staff members will have the option to receive alerts simply after a message is received or after a message is unread for a portion of time.

2.3.2.4 Message Review and Forwarding

ConnectBasket will allow message recipients to review their messages online via any internet-accessible device, should the connection or other factors not restrict them from doing so. Staff who have viewed a message will be capable of rerouting this same message to other necessary staff, along with any comments or addendums added by the previous recipient.

2.3.2.5 Message Resolution

ConnectBasket will allow message recipients to resolve messages as closed, should the task or job mentioned within it be completed. This closed state should remove the message from all staff member's queues, allowing them to focus on other tasks that have not been resolved.

Ideally, this feature will include more than just a closed state; it will include additional states that show other reasons for a task to no longer be active, such as cancelled, duplicate, or stopped, to provide more information about the status of a task.

2.3.2.6 Message Audit Record

ConnectBasket will allow staff with sufficient permissions to view a message's history for audit or review. Authorized staff will be able to see when and where the message was created, when and where it was routed, and when and where it was closed. The name of any staff member who took any action should also be recorded, along with other activity regarding a message.

Providing an audit trail for all messages will provide the hospital with valuable information and allow them to track metrics such as the average amount of time the hospital takes to respond to calls and common reasons for owners calling the hospital.

2.3.2.7 Message Recipient Roles and Profile Roles

ConnectBasket will allow staff to select and be assigned to one or more roles that reflect their duties and responsibilities at the hospital. Staff will have profiles to manage their roles, with the ability to change their roles at will. When messages are routed to a specific department, any staff member that has selected the associated role to that department will receive the message in a queue accessible by anyone assigned to the role. These staff members will be able to take ownership of the message to interact with it, or interact with it to a lesser degree without taking ownership of the message. All of the regular message features that are accessible to individual staff members will also be accessible to the all the staff members that receive a message in a role.

Staff in one location can not assume, and should not have to assume, whether a certain employee is available or whether a task falls under their responsibilities. Assigning messages to a role and putting them in queues removes the necessity for staff to understand structure in other departments, and allows tasks and messages to be passed faster and with less room for error.

2.3.3 Performance Requirements

2.3.3.1 Web Service

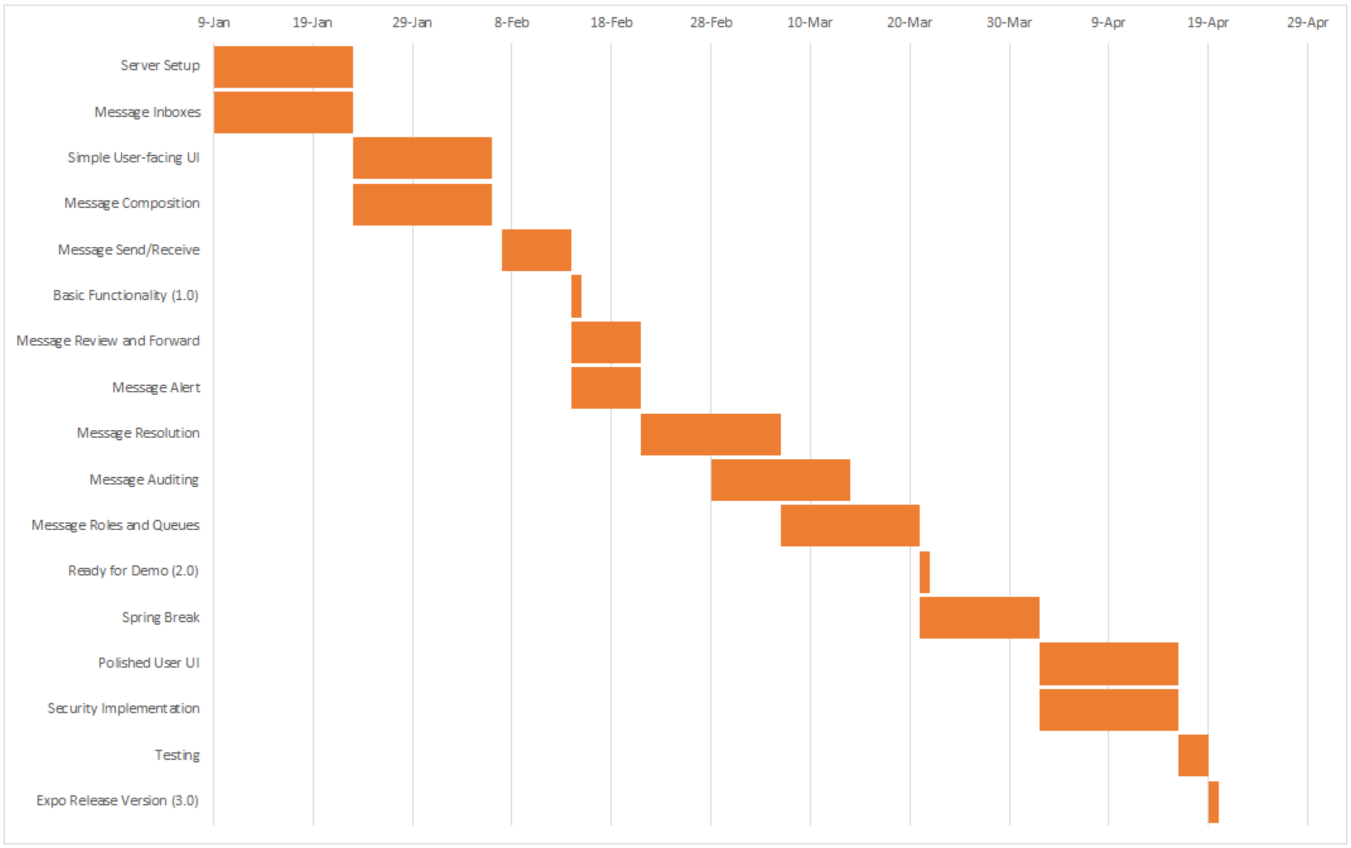
This web based portal will need to be accessible from any internet accessible device. Staff will be doing vital work through this application, and so it must be accessible to them, no matter what device they have. However, there are constraints on how accessible this is. Internal messaging must remain confidential and secure, and staff should have secure profiles to restrict access to malicious persons. ConnectBasket must be reliable and have minimal downtime, as this application going down could cripple the work of the hospital staff once it is deployed. Downtime should be kept to a maximum of two hours a week once in a fully completed state. Two hours per week allows for security patching and maintenance should it be required, though this number may change depending on the specific processes as they are defined.

2.3.4 *Design Constraints*

ConnectBasket will be constrained only by any government or hospital regulations on message security or reliability. Regulations may require audit trails or set a maximum downtime for the application, which will restrict the design as these come to light.

In addition, while not a mandatory constraint, eventual integration with VetHosp is the long term goal for this project. If the project is built in such a way as to ease the connection between the two, this design choice may save a significant amount of time. However, with a lack of knowledge about VetHosp, designing in such a way is unlikely and thus, not a mandatory element of the design.

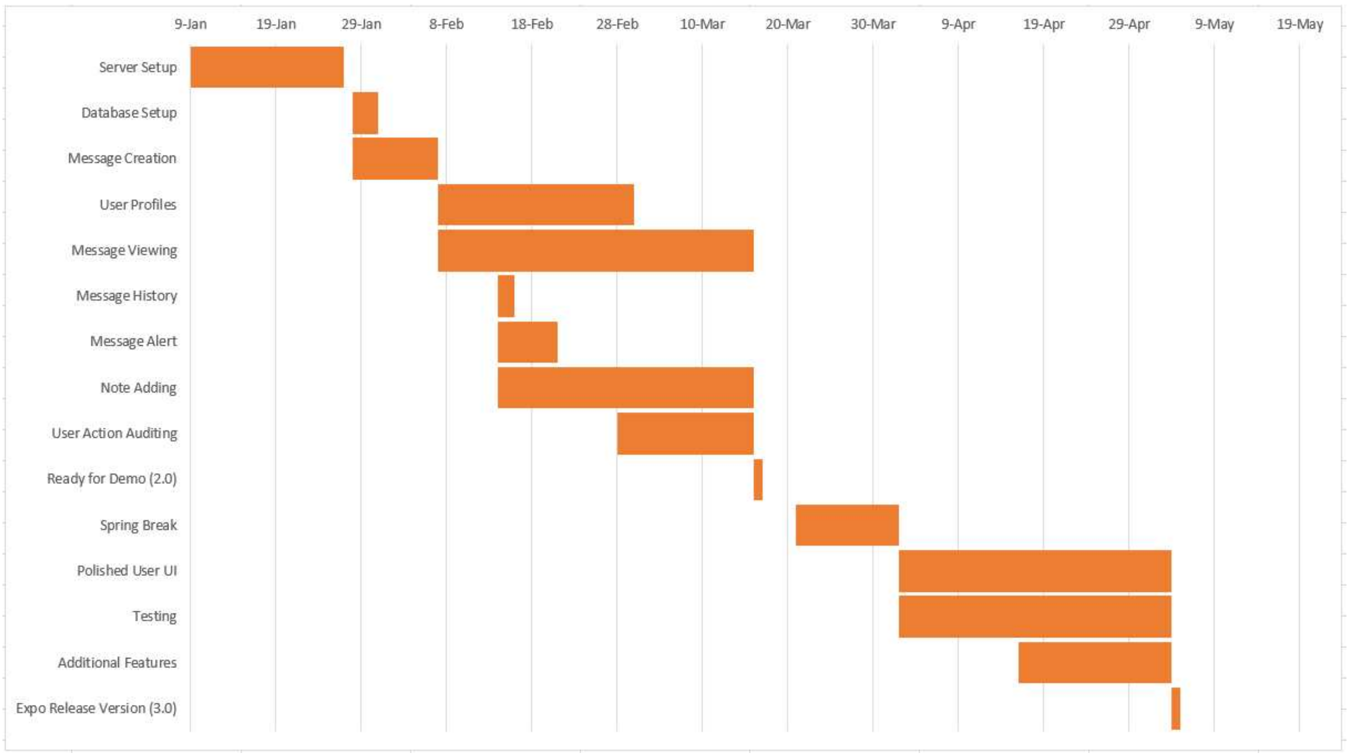
2.4 Original Gantt Chart



2.5 Changes

As no more information about the VetHosp system was given to the development team, no integration with VetHosp was completed. In place of this the develpment team added an additional feature to allow messages to be exported to a PDF file that can then be uploaded into the VetHosp system. It was also decided that email notifications would be sufficient and sms messaging was not necessary.

2.6 Final Gantt Chart



3 Design Document

3.1 Introduction

3.1.1 Purpose

The purpose of this document is to layout a design plan for the ConnectBasket project. It will provide design viewpoints as well as identify stakeholders in the project. Each component's design will be explained in detail, and a timeline and testing information will also be provided for each component.

3.1.2 Scope

The ConnectBasket project is intended to be a stand-alone solution to help the Oregon State Veterinary Hospital increase the efficiency of communication between employees within the hospital and to patients. The hospital's current system is heavily paper based and error-prone, and this project will allow for a more efficient workflow.

Employees of the hospital will need to be able to enter messages about or connected to patients, route the message to other employees to take some action or add a note, and close the message as being completed. A notification system will be implemented to let employees know when they have a message to look at. There will also be a way for administrators to view a history of messages and their corresponding notes. This project will help the hospital quickly respond to patient inquiries and either answer their questions or schedule an appointment.

3.1.3 Intended Audience

The intended audience for this document is the Capstone development team working on the project as well as the client for the project. It is a plan for how the Capstone development team will implement the project. In addition to that, it will provide the client with information about the design and timeline for the development of the project.

3.1.4 Stakeholders

The stakeholders in this project include the client and the employees of the hospital. The client is working with both the Capstone development team and the hospital to complete the project. He will not be directly involved in using the product once it is developed. The employees of the hospital will be the end users of the project and interact with the finished product as a part of their daily jobs.

3.1.5 Glossary of Terms

Capstone - CS 461 class at Oregon State that the developers of ConnectBasket are in

Capstone development team - The developers of ConnectBasket

ConnectBasket - A web application being created for the OSU Veterinary Hospital

Hospital - Referring to the OSU Veterinary Hospital

Message - Initial information taken down by a receptionist

Mobile Devices - Smartphones or tablets

Note - Information added to a message at some point before the message is closed by a user the message was routed to

OSU - Oregon State University

Owner - A person who owns a pet that is a client of the hospital

Patient - An animal who has/is going to be treated at the hospital

Route - Send or transfer, usually referring to the forwarding of a message to one or more staff members

VetHosp - The hospital management system used by the OSU Veterinary Hospital for keeping patient and owner records
Vet techs - Veterinary technicians

3.2 Design Viewpoints

3.2.1 Functional Viewpoint

3.2.1.1 Functional Elements

There are many functional elements that will make up the ConnectBasket web application. The four main pages of the web application are message entry, adding notes to messages, message viewing, and an audit trail of messages. The other functional element of the system is a database that stores the information collected through the application.

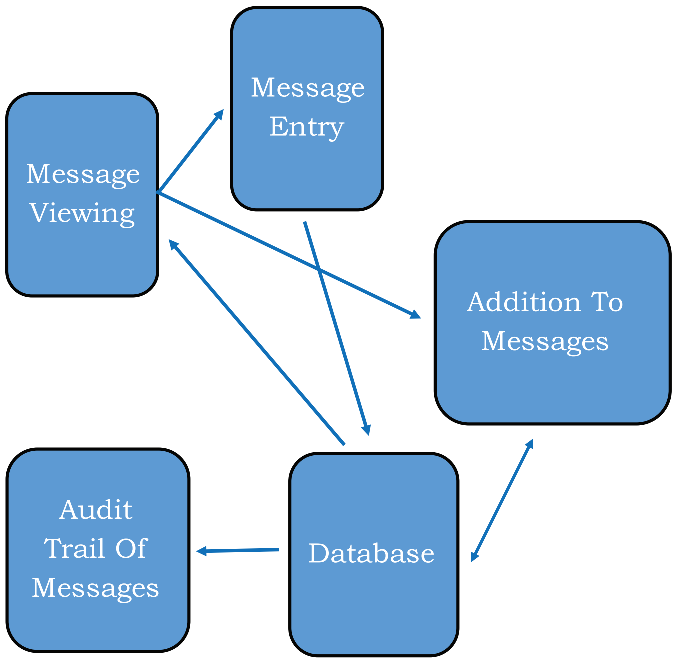


Fig. 1. Functional elements and their relationships

3.2.1.2 Responsibilities

Each element has its own responsibilities. The message entry element will allow users to add a message to the database and route that message to other users. Adding notes to messages will allow users to add more information to a message that has already been created and either route to other users or close the message. Message viewing will allow users to see all messages that have been routed to them and see all of the previous notes that are associated with each message. The audit trail of messages will provide administrators with the ability to look at a history of messages and their notes that were created during a given time period. The database will have tables to store information about messages and their associated notes, and will have queries that return groups of messages that meet certain criteria.

3.2.1.3 Interfaces

Each element will have its own page in the application. Message entry and adding notes to messages will be done through a web form, and the message viewing and audit trail of messages will be displayed through a table or grid. The database will be accessed through each page of the application.

3.2.1.4 Primary Interactions

Several of the pages will interact with each other. From the message viewing page, a user will be able to go to the message entry page or they will select a message and go to the adding notes to messages page for that message. Creating messages and adding notes to messages will insert new information into the database, where message viewing and audit logs of messages will pull information out of the database.

3.2.2 Context Viewpoint

3.2.2.1 Relationships and Interactions

The ConnectBasket system has many relationships with its users. Users that work in reception and scheduling at the hospital will use the system to add new messages as well as view messages and their associated notes to call owners and give them answers to their questions or schedule appointments. Vet techs and doctors want to be able to view messages that have been routed to them and add notes to them. Administrators want to be able to view the audit logs of messages.

There will also be several external systems that ConnectBasket will interact with. A user profile framework will be used for managing users of the application and keeping login information secure. ConnectBasket will also interact with an email client in order to send notifications to users that a message is waiting for them. If proper access is provided to the development team, ConnectBasket will integrate with the VetHosp database to access patient and owner information.

3.2.2.2 Dependencies

The notification feature of ConnectBasket will depend on users having access to their email through a mobile device or desktop computer that they check frequently. The user profile system will depend on users keeping their own information up to date in order to receive notifications and see their messages.

3.2.3 Information Context

3.2.3.1 Information Storage

The information for ConnectBasket will be stored in a relational database. There will be tables that store information about messages, notes, owners, patients, and possibly users. The different pages of the web application will allow for the information to be managed and displayed.

3.3 Components

3.3.1 Server Setup

3.3.1.1 Details

This project will require having a server configured to host the web application ConnectBasket. The application will not be supporting heavy traffic and does not have any specific speed requirements so the server requirements are not very strict.

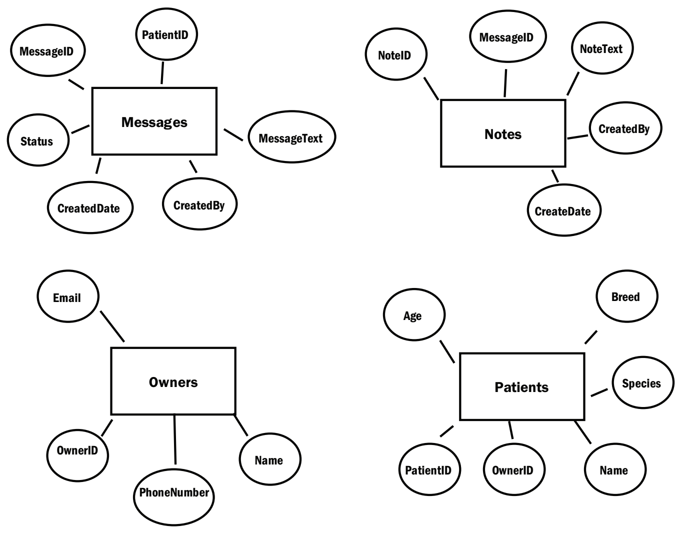


Fig. 2. Design of ConnectBasket database

The server will need to work with the web server software and database to make the application accessible to the users. Users will interact with the server through the API created as a part of the ConnectBasket application.

3.3.1.2 Design Decisions

The server operating system that will be used for this project is Linux Ubuntu Server. It was chosen for its low cost, short time to set up, and well written documentation. Most of the components of the project rely on the server being up and running so the decision was primarily based on the amount of time it will take to get set up [1].

3.3.1.3 Timeline

The setup of the server will be the first priority in the development of this project. For this reason, the completion of this component will be done by the third week of winter term.

3.3.1.4 Testing

There will be relatively little testing done for this particular component. If the web application is able to successfully be accessed, then this component will be considered successfully implemented.

3.3.2 Web Server

3.3.2.1 Details

In order for ConnectBasket to work, the server must have web server software running on it. This software will allow users to access the web application running on the server from any mobile device or desktop computer that has internet access.

3.3.2.2 Design Decisions

For this project, Apache HTTP will be the web server software that is used. It was chosen because of its cost, documentation, and flexibility. It is free software that can be used by the ConnectBasket development team for no cost. There is a large amount of well written documentation that will help during the setup period. Due to the short development window for the project, speed of implementation was a key factor in deciding which web server software to use. It is also a very flexible web server that can run on almost any hardware and with almost any server operating system[2].

3.3.2.3 Timeline

The web server software properly working is critical to the rest of the components being able to work and be tested. This component will be implemented as soon as the server component is completed. The web server software will be completely implemented by the end of week four of winter term.

3.3.2.4 Testing

The testing of the web server software will be done by attempting to access the ConnectBasket application after it has been partially or fully created. This component will be considered complete when the web application is able to be used from all required devices.

3.3.3 Database

3.3.3.1 Details

ConnectBasket will have a database that stores the information collected by the application. Messages that are created as well as all notes that are added to those messages will be stored in tables. There will also be tables that store information about the patients of the hospital and their owners. If the information is provided to the Capstone development team, the patient tables will be filled with information from the current VetHosp system. Otherwise, the tables will be filled with fake data for testing purposes, and patient information will have to be manually added by a hospital employee through ConnectBasket the first time an owner calls after the system is being used.

The data gathered through the application will be used for many different purposes. First, the messages entered and notes added will need to be immediately available through the web application for other users to view. Second, users will need to be able to see a chain of all notes that are associated with a certain message. Finally, administrators will be able to view an audit trail of all messages created and notes added over a given time period.

3.3.3.2 Design Decisions

The database used for this project will be a MySQL database. It is important to have a relational database to store the data needed for the project. MySQL is also very flexible and can work well in almost any system. It is not the fastest database that could be used, but this application will not require any large amount of data manipulation, so the simple and easy to use interface provided by MySQL is the best choice[3].

3.3.3.3 Timeline

The database will be critical for testing many other parts of the application, but will also be a large part of the project. Initial setup of the database and creation of tables will start around week two or three of winter term and finish by week five. The finished version week five is still subject to change, if the need for different or more tables or queries is found during the development of the rest of the project.

3.3.3.4 Testing

The database will be tested in several ways. Queries will be run on their own to make sure that the correct data is being returned, in addition to using the web application to test and make sure data can be added and updated.

3.3.4 Mobile Notification System

3.3.4.1 Details

ConnectBasket will have the ability to notify users. Users will have unique profiles that can contain information about them, including their email. This email will be used, according to user specifications, to alert them on their mobile device when a message is received or some other event designated as important by the user occurs. ConnectBasket will rely on the user's email application on their mobile device to deliver alerts and notifications. The body of this email will contain a link to the relevant page appropriate to view the specific event. The email body will also contain identifiers for any other users involved in the activity, along with a short description of the activity.

3.3.4.2 Design Decisions

Rather than email, using SMS was an alternative option. However, due to the small scale of this project, paying for a VOIP service to handle SMS to most providers is unreasonable, especially with such few differences between them [4]. While it is possible to use email to send a SMS message, it requires that one know the user's cell phone provider. In addition, messages sent in this method can be ugly and cluttered, possibly resulting in the user ignoring the message, or even suspecting it as spam. On top of this, SMS messages are generally for personal communication, and an automated message relayer may confuse the users. Because of all this, the Capstone development team decided against using SMS here. The ideal choice would be to use native notifications for mobile devices, but this is also an unreasonable option, as ConnectBasket will be a web application and will lack native mobile versions. Notifications through a mobile browser are technically possible, but will only take effect while the page is open on the users phone, which is likely to be very rare and is not a good solution. Sending emails from a service email or a noreply account is easier than both, free when done directly like this, can reach users even when they do not have ConnectBasket open, and easy to avoid confusion with custom body content and a more appropriate medium than texting for automated messages.

3.3.4.3 Timeline

Notifications should be a more simple element to the project, and should not take longer than a week to develop and test completely. While a primitive version can be created ahead of time if needed, actual use-case testing will not be possible until user profiles and user messages exist to a certain degree. Notifications will need to make use of user information, at minimum a user's contact information and their identifying username. Notifications will also need to make use of a message sent to a user, requiring the message system to be implemented and to the point where it can send at least basic messages to specific users.

3.3.4.4 Testing

Testing notifications should be very simple. Basic functionality can be tested immediately, by sending a dummy message to a specific email. If the basic function works, it will alert the email's owner on their mobile device, should the settings be enabled. Otherwise, testing will occur after both profiles and messages are implemented, with success criteria being: notification sent automatically, notification sent upon user receiving a message, notification sent using contact information from the user's profile, and user being able to navigate to the message via the notification. If all of these things are possible, this feature will be considered finished.

3.3.5 User Profile Framework

3.3.5.1 Details

Users of the ConnectBasket software will be able to securely log in to accounts containing potentially confidential information. These accounts will be unique and linked to a single employee of the hospital, though not every employee of the hospital must have an account. These accounts will have identifying information, for both the ConnectBasket backend to use, and for other employees to use. The user accounts will have profiles that contain their information. User accounts will have the ability to be part of one or multiple groups, which can also contain other users. These groups will be used by the issue tracker system.

3.3.5.2 Design Decisions

The focus for this feature is, rather than the functionality of the users, the security of the accounts and the ability to log in from various devices. Users will have profiles, but these profiles will not need to contain much more information than a username, an email, some settings, and a list of groups the user is a member of. Thus, since there is not a lot of user information, the PHP user login framework HUGE will be used, or at least referenced, for the user logins for ConnectBasket [5]. HUGE is a simple framework without many fancy features, perfect for use in ConnectBasket. Other options researched by the team, like Userfrosting, were also simple and easy to use, but generally had more functionality than needed, which could lead to confusion or compatibility issues [6].

3.3.5.3 Timeline

Users should be one of the first things implemented in the system. While the message tracking functionality of ConnectBasket does not necessarily depend on users to exist, it is not functional and cannot be tested without it. Thus, user profile functionality is likely one of the first parts of the project software that should be done, once the web server is ready. It will need to be mostly implemented to allow other features to be started or tested.

3.3.5.4 Testing

Testing a basic user login function is as simple as seeing if a login is possible. This can be tested early in the project, possibly even offline, before the web server is prepared on a developers personal computer. Once the webserver and database, a basic user-facing front-end is prepared, and user logins are implemented, they can be tested with success criteria being: a user can, without cookies in place, or in an incognito mode, or a new device, navigate to the ConnectBasket webpage and log in to their unique account using a username and password, a user can view information that the issue tracking system provides for them, a user cannot see messages unrelated to them, a user can change contact information, a user can log out, a different unique user can log in and do all of the above, and a user can close the browser or navigate to another page and be able to return to the ConnectBasket site without needing to log in again. All of these tests should also be verified on a mobile device as well.

3.3.6 *Issue Tracker Framework*

3.3.6.1 Details

The core functionality of ConnectBasket is similar to many issue trackers or Kanban boards, without the graphical interface or board layout. ConnectBasket will include features one could expect from any issue tracker, but personalized for the hospital. The core of the issue tracking is the messages sent between users, which should be secure, accurate, and relatively fast. These messages will be directed to users and to groups. Users, as mentioned in the user profile framework section, can be part of groups, and can view and add notes to messages sent to these groups. These messages can be forwarded to other users or groups, or closed. These messages will have a history, which will contain and report the activity of a message, with all potentially relevant details. Messages will have multiple states such as closed, open, and duplicate that can be changed on demand.

3.3.6.2 Design Decisions

Much of the desired functionality for this project is visible in the message system. Since much of what the client is asking for is similar to a ticketing system or kanban board, the development team will be looking for examples in similar technologies. The team considered simply wrapping an existing solution with a different user interaction system, but found that such a solution would contain too many unneeded features, and would not be much effort for a senior Capstone project. There are a few open source options that do similar or almost exactly what is requested by the client, but these are unlikely to fulfill the needs of this project for the reasons listed above and because the client has requested specific features that may not exist in these solutions. The team will be implementing an issue tracking message system from scratch, using the Best Practical Issue Tracker as a guide for what the final product may look like, and to use as reference should any roadblocks or issues be encountered [7].

3.3.6.3 Timeline

This feature depends on the physical hardware and the webserver existing before implementation. While it cannot be formally tested beforehand, development can be done offline on a development computer. This work may not function in the same way as it will when on a webserver, so development will also need to take place on the server regardless of how much is done offline. This feature depends on and cannot be completed until the webserver has been implemented. In addition, to view any meaningful output and for testing, the user login feature will need to be implemented to some degree.

3.3.6.4 Testing

A successful test of this feature denotes a working version of ConnectBasket. Early tests before full completion will likely be on whether or not the feature is capable of sending a message to a dummy user, and allowing users to enter and view messages.

3.3.7 *Content Delivery Network*

3.3.7.1 Details

A content delivery network, or CDN, is a group of servers that are distributed geographically that work together to quickly deliver Internet content, such as images, videos, and HTML pages. Currently, most web traffic is served through

CDNs, and CDN's can also help protect against some malicious attacks. Reasons to use a CDN include improving website load times, increasing content availability, and improving website security [8].

3.3.7.2 Design Decisions

The CDN chosen for the ConnectBasket project is Cloudflare. The CDN that Cloudflare provides is designed to optimize security and performance, has multiple price levels, and is easy to set up. Cloudflare's CDN provides a flat price per month that does not depend on bandwidth, which makes it a good choice because there will never be any unexpected spikes in cost[9].

3.3.7.3 Timeline

As this feature is not a required one for the project, its development will not be a priority. The addition of a CDN will be done after all other parts of the project are complete. This would likely be done in the final week of winter term or during spring term.

3.3.7.4 Testing

The testing of the CDN would be minimal. The only testing that would need to be done is to make sure all features of the web application were still working after the CDN had been set up. As long as it does not cause problems with the functionality of ConnectBasket, it will be considered working.

3.3.8 Web Development Framework

3.3.8.1 Details

A web development framework is considered to be tools and resources used by software developers to create and manage websites [10]. Web development frameworks extend the capabilities of a language and provide libraries so developers do not have to start from scratch and hand-code everything [11]. ConnectBasket will have many basic features that most web development frameworks will provide libraries to easily implement. Using a web development framework to complete this project will help to speed up the development process and make the finish product aesthetically pleasing and user friendly.

3.3.8.2 Design Decisions

The web development framework that will be used for the ConnectBasket web application is AngularJS. AngularJS was created and is maintained by Google and provides a powerful framework for building quick and easy to use web applications. It is a Javascript framework that extends basic HTML, adding many useful features that will work nicely to provide the features needed for ConnectBasket [12]. AngularJS was chosen for this project because it provides simple, easy to use tools that can build everything the project needs to have.

3.3.8.3 Timeline

The web development framework will be used to create the ConnectBasket application. Development will begin at the start of winter term and will continue until the end of winter term and into spring term until the project is complete. Development can be done on the development teams computers, but for full testing, the setup of the web server and database will need to be completed first.

3.3.8.4 Testing

The testing of the ConnectBasket application created by the web development framework will need to be very thorough. There will need to be tests done of all of the individual features that make up the web application. In addition to testing by the developers, testing will need to be done with the users of the application to receive feedback and make final changes to the project.

3.3.9 Software Design Pattern

3.3.9.1 Details

A design pattern is a solution to a general problem that can be reused in multiple situations. There are many different design patterns that can be used for a variety of situations, but often, there is one pattern that best fits a scenario.

3.3.9.2 Design Decisions

The design pattern that will be used for the ConnectBasket project is MVC, or Model-View-Controller. There are three components of MVC: Model, View, and Controller. The Model represents the data, the structure of the data, and how the data is represented in the application. The View is the component that makes the data displayable to the user in a useful way. The controller supplies an interface between the model and view components, taking the data from the model and converting it into something that can be used by the view component to display to the user [13]. Using the MVC design pattern for the ConnectBasket project makes sense, because it can be split into these three components. The Model component will be the patient and owner data in the database. The View will be the interface of the application. The Controller will be the way that the Data and View components communicate and are connected. Additionally, there are three developers on the development team, which makes it easy to assign roles.

3.3.9.3 Timeline

The software design pattern is used throughout implementation of the project and will be finished when the project is complete during spring term. It is closely associated with the web development framework and they will be completed at the same time.

3.3.9.4 Testing

The chosen design pattern will allow for easier testing of the finished ConnectBasket application through unit tests that can test each function that the Controller provides.

3.4 Conclusion

3.4.1 Summary

ConnectBasket is a web application that will be hosted on a server running a Linux Ubuntu Server operating system and an Apache HTTP web server. A custom database will be created for the project using MySQL. User notifications will be done through email and user profiles will be managed through a user profile framework. The web application will be developed using the AngularJS web development framework and create a customized system for managing hospital communications. The architecture of the application will follow an MVC software design pattern.

3.5 Changes

There were two main changes made to the initial design of the project. The first was not using an existing user profile framework. The development team decided to manage user profiles as a part of the ConnectBasket website in order to better provide the exact needs of the client. The second change was that no integration with VetHosp was done because the development team never received any additional information that would allow for that integration to be done.

4 Tech Review (Henry)

4.1 Introduction

4.1.1 Reason for Project

The current software system used by the Oregon State University Veterinary Hospital for communication is out of date and inefficient. Their current process is very slow and manual, with lots of movement around a large hospital and unnecessary printing of papers. For instance, if an owner calls the hospital with a medical question, the receptionist would log the message into the computer, print it out, and physically walk the paper across the hospital to put in a spinning queue. When a doctor or other staff member looks at it, they would write an answer or necessary action on the paper, and physically carry the paper to the next location. Eventually, after it reaches all the necessary staff, it would be returned to the receptionist, who would then call the owner back to provide them with the information they requested, whether that is an answer to a question or to schedule an appointment.

4.1.2 Overview of Project

This project aims to create a new stand-alone solution, ConnectBasket, that will streamline the communication process in the hospital. In this document, three main components of the project will be examined, looking at three different technologies for each that could be used to implement that component. ConnectBasket will be a web-based application, so it will be important to set up a server with the best operating system for this project. There will also need to be web server software that is compatable with the server operating system and meets all of the requirements of the project. Lastly, the information collected by the application will need to be stored in a database, so it is important for the best option to be used.

4.2 Server Operating System

4.2.1 Overview

For this project, there will need to be a server to host the web based services involved in the project. Because servers are mainly used to run applications and do not need to be interacted with frequently by non-technical users, server operating systems focus on the ability to manage resources with little attention being paid to a user interface. There are many different choices of operating systems that can run on a server and this section will analyze three different options and show that one will be the best fit for the project.

4.2.2 Criteria

The criteria that will be used to choose the best server operating system for this project will be the cost, difficulty of setup, available support, necessary downtime, and amount of maintenance needed. For this project using low cost or free technologies will contribute to keeping the overall budget low. There is a relatively small time window for development of the project, so a fast and easy installation is very important. It will also be important to have to have a low maintenance system that will continue to work after the development team is gone.

4.2.3 Potential Choices

4.2.3.1 Linux Ubuntu Server

Ubuntu is a distribution of Linux that is very popular on desktop computers, but also has a server version that is growing in popularity. Unlike the very popular Red Hat Linux distribution, Ubuntu is free. Updates and maintenance of the code and security of Ubuntu are made by the global community as well as a paid development team. There is twenty four seven support available that can be very important if a quick response is needed to critical errors. Ubuntu is an easy system to setup with an installation that can be completed in as little as fifteen minutes. Like other versions of Linux, Ubuntu Server is very easy to maintain and rarely requires reboots, making it ideal for an application that requires very little downtime [1].

4.2.3.2 Windows Server

The Windows Server operating systems have been very popular with businesses since they first came out. Many tools are provided that will work very well with Windows Server, including Microsoft SQL Server and Internet Information Services. A large number of successful businesses use Windows Server, which means that there are many examples of how to use it as a part of a business. Windows Server is considered by many to be the best server operating system for mission-critical systems, meaning that there is very little downtime required when using Windows Server. Windows Server has built in tools to help improve power efficiency of the server that can help lower power consumption for a company. A virtualization platform is also included with Windows server which can allow for easily migrating the system from one physical machine to another or allowing users to remotely access applications and run them like they were locally installed [14].

4.2.3.3 Solaris

Solaris is a version of Unix that has been developed as a server operating system. Solaris was created by Sun and they are also responsible for supporting it. This makes Solaris a good operating system for a user who will need a lot of help in setting up their system. Solaris is based off of Unix, which is an operating system that has been around a long time and is trusted by many businesses. A problem with Solaris is that it will not run on many different hardware systems including most HP and IBM systems. Solaris is not free, although there is a version of Solaris that is more similar to Linux and can be optained for free with optional paid support [15].

4.2.4 Discussion

Unix, which Solaris is based off of, has been around a long time compared to Windows Server and Linux Ubuntu Server, but is not currently as popular as either of the other two. Linux Ubuntu Server is completely free with optional support packages that can be purchased, in contrast to Windows Server and Solaris which both cost money. All of the operating systems provide key features that make any server operating system useful such as security, support, easy setup. It might be slightly harder to get support when using a Linux system than the other two because Linux isn't owned by anyone.

4.2.5 Conclusion

The best server operating system to use for this project is Ubuntu Server. Solaris is not free and does not provide as many features as Windows Server or Ubuntu, so it is definitely not the right choice for this project. Windows Server might provide slightly more features than Ubuntu, but all the server will be used for is to host a simple web application, so things like allowing users to run desktop applications remotely are not important for this project. For all of the important features to this project like setup, support, downtime, and maintenance, Windows Server and Ubuntu are not that different and both would be good choices. The main difference is that Ubuntu is free, and for that reason, it is a better choice than Windows Server for this project.

Criteria	Ubuntu Server	Windows Server	Solaris
Cost	Free	$100+	Free version available
Difficulty of setup	Very easy	Slightly difficult	Easy
Available support	24 x 7 support available	Some support available	Sun provides support
Necessary downtime	Very little	Very little	Very little
Maintenance needed	Almost none	Some maintenance needed	Some maintenance needed

4.3 Web Server Software

4.3.1 Overview

Web server software is essential to running a website, and allows the content created by developers to be displayed to the end users through a web browser. There are many different options to use and it will be important for this project to have one that meets all of the needs of the project.

4.3.2 Criteria

The most important factors for this project are a web server software that is well documented and fast to setup, as well as cheap and able to run at fast speeds with relatively little downtime. With the short time period for development, it will be critical to have a web server that is up and running quickly. In addition, a low cost is another factor to consider for this project. The web application will be very important to daily operations of the hospital making small amounts of downtime another factor in evaluation.

4.3.3 Potential Choices

4.3.3.1 Apache HTTP

The Apache Web Server is very powerful and provides many useful features to its users. It is an open source software, making it free to use and also gives users of this software the ability to modify its source code to meet their needs. Features include a control panel, customizable error messages, authentication schemes, Domain Name Service, Simple Mail Transfer Protocol, and File Transfer Protocol. Apache can run on almost any operating system including Windows, MacOS, Linux, and Unix. There is also support for many different programming languages, including PHP and Python, and the ability to have SSL and TSL encryption for websites [2].

4.3.3.2 Cherokee

Cherokee is a high performing web server that runs quickly and is easy to set up. It supports many other technologies, such as PHP and SSL encryption, and can be run of many different operating systems including Linux, Unix, and Windows. Cherokee also provides updates that do not require any downtime. Another important feature of Cherokee is that it provides an easy to use configuration interface called cherokee-admin. Cherokee is also free and has a an open source code base that can be modified by anyone who wants to make changes to adjust the system to meet their needs [16].

4.3.3.3 NGINX

NGINX is a system that has been optimized to handle a large number of simultaneous connections. It provides support for PHP and Python as well as email protocols like SMTP, POP3, and IMAP. It is a very fast system that maximizes hardware efficiency. NGINX mainly only supports Linux and Unix based systems, and has very little support for

Windows and other operating systems. There is plenty of documentation explaining how to use NGINX and training to learn more about the system. This software is open source and free to use with an optional purchase that includes support [17].

4.3.4 Discussion

Apache HTTP and NGINX are both widely used web servers compared to Cherokee, which is a much less used product. All three are open source, available for free, and provide many of the same features and support. NGINX doesn't support very many operating systems compared to both Apache and Cherokee, which support almost any operating system.

4.3.5 Conclusion

For this project, the best choice of web server software is Apache. It provides all of the features that are necessary for the project, and can run on almost any operating system, where NGINX can not. Cherokee is also a good choice, but is not as widely used and does not have as much support or documentation as Apache.

Criteria	**Apache HTTP**	**Cherokee**	**NGINX**
Cost	Free	Free	Free
Documentation	Very well documented	Some documentation	Well documented
Setup time	Fast	Very fast	Fast
Necessary downtime	Very little	Very little	Very little

4.4 Database

4.4.1 Overview

A database will be a very important part of this project. User information as well as information about patients will need to be stored in the database. The web application will rely heavily on interaction with the database to provide the features users will need.

4.4.2 Criteria

Important factors to consider when selecting a database for this project are the cost, speed, compatability with chosen operating systems and web servers, and ability to provide easy ways to report on the data it holds. As always it will be important to keep the cost of the project low. Users will need to enter information while on phone calls with owners of the patients, so speed will be important as they don't want to waste the time of the person on the phone. Reporting on the data collected will be very important for administrators to evaluate the success of the hospital and the new system.

4.4.3 Potential Choices

4.4.3.1 MySQL

MySQL is an open source database management tool that provides many important features. MySQL is known for its excellent data security and is used by many popular websites, including Facebook and Twitter. It provides high performance and excellent scalability that gives it great speeds for any size system and allows it to only take up as much space as needed. MySQL comes with a guarantee of no downtime and can work with almost any operating system and web server [3].

4.4.3.2 SQLite

SQLite is a relational database management system that is imbedded into the system that is using it. It is a file-based database that interacts directly with the system making it very fast and efficient. The system is easily portable because it is contained in a single file on the system. SQLite uses a slightly modified SQL that has only a few features removed. It does not provide any ability to manage different users access to the database like some other management systems might [18].

4.4.3.3 Microsoft SQL Server

Microsoft SQL Server is a very easy to use and powerful database management system. It provides integration with tools like SQL Server Management Studio and SQL Server Profiler that help users manage their database. There is a lot of support and documentation provided for SQL Server and it is much easier to learn and get help with than many other SQL based products. SQL server is a paid product and purchasing comes with support from Microsoft [19].

4.4.4 Discussion

SQLite is a slightly less feature rich product compared to MySQL and Microsoft SQL Server. All of these systems can work with many different operating systems and web servers. Microsoft SQL Server is a paid product, in contrast to MySQL and SQLite, which can both be used for free.

4.4.5 Conclusion

The best database management system for this project is MySQL. The speed of SQLite is better than the other two, but is not worth the tradeoff of lost functionality for this application. Microsoft SQL Server provides slightly more features, documentation, and support than MySQL, but they are not important enough to the project to offset the large cost difference between the two.

Criteria	**MySQL**	**SQLite**	**Microsoft SQL Server**
Cost	Free	Free	Around $1000
Speed	Fast	Extremely fast	Fast
Reporting capability	Good	Average	Excellent
Compatability	Works with any system	Designed for smaller systems	Works best with other Microsoft products

5 Tech Review (Kailyn)

5.1 Introduction

5.1.1 Reason for Project

The current software system used by the Oregon State University Veterinary Hospital for communication is outdated and inefficient. In their current system of receiving messages, if an owner calls the hospital with a medical question, the receptionist would log the message into the computer, print it out, and physically walk the paper across the hospital to put in a spinning queue. When a doctor or other staff member looks at it, they would write an answer or necessary action on the paper, and physically carry the paper to the next location. Eventually, after it reaches all the necessary staff, it would be returned to the receptionist, who would then call the owner back to provide them with the information they requested. Currently, their process is very slow and manual, with lots of movement around a large hospital and unnecessary printing of papers.

5.1.2 Overview of Project

The ConnectBasket project will be a stand-alone solution that will streamline the communication process in the hospital. A receptionist will be able to receive a call and create a new message. They can then assign a category to the message and route it to the necessary staff, who will be able to add notes and reroute it to more staff members. Staff will receive notifications, by email or text, when they have a message waiting to be viewed. Staff will have profiles where they can select their notification preferences, as well as which categories of messages they want to receive. When the message returns to the receptionist, they can call back the owner and inform them of the decisions or recommendations by the staff. The whole chain of where the message has traveled to and who has seen the message or added notes will be visible. This will help the hospital track specific metrics, such as response time to calls.

5.1.3 Overview of Document

In this document, three components of the project will be examined, looking at three different technologies for each that could be used to implement that component. ConnectBasket will be a web-based application, and a content delivery network will be useful for delivering internet content, as well as improving load times and security. ConnectBasket will also require a web development framework. Finally, a software design pattern will be helpful for developing a simple, effective user interface.

5.2 Content Delivery Networks

5.2.1 Overview

A content delivery network, or CDN, is a group of servers that are distributed geographically that work together to quickly deliver internet content, such as images, videos, and HTML pages. Currently, most web traffic is served through CDNs, and CDN's can also help protect against some malicious attacks. There are four main reasons to use a CDN: to improve website load times, to reduce bandwidth costs, to increase content availability, and to improve website security [8].

5.2.2 Criteria

For this project, security is an important factor to consider. Since the ConnectBasket website will contain patient and owner information, it is important that the website is secure. Another essential factor to consider for the ConnectBasket project is price. It is critical for the cost to stay as low as possible, so choosing a free or low cost CDN is important to the project. In addition, fast and easy setup for the CDN would be beneficial, since there is a relatively small time frame for the development of ConnectBasket.

5.2.3 Potential Choices

5.2.3.1 Cloudflare

The CDN that Cloudflare provides is designed to optimize security and performance, has multiple price levels, and is easy to set up. Cloudflare's CDN provides a flat price per month that does not depend on bandwidth. There are four pricing levels. The free version is for personal websites. The Pro version is twenty dollars per month for professional websites. The Business version is two hundred dollars per month for websites and businesses requiring advanced security and performance. Finally, the Enterprise version is for companies requiring enterprise-grade security and performance, and Cloudflare muse be contacted to determine a monthly price. In addition, setting up Cloudflare is easy, taking less than five minutes to set up a domain [9].

5.2.3.2 Akamai

Akamai serves thirty percent of internet traffic, and it optimizes networks for content delivery [20]. Akamai's CDN has three components [20]. The Aura Edge Exchange allows operators to deliver video content to customers [20]. The Aura Control System is a set of tools that allows operators to manage components such as performance and CDN security [20]. Lastly, Akamai Federeation allows operators to work with the Akamai Intelligent Platform, which is a cloud-computing platform. Akamai does not provide pricing information on their website; they must be directly contacted to get a quote [20].

5.2.3.3 Amazon CloudFront

The Amazon CloudFront CDN is built on the Amazon Web Service (AWS) infrastructure, with forty-four Availability zones in sixteen regions, and there are plans for further expansion. This large network provides high performance for customers all over the world. CloudFront is a highly secure CDN, and for no additional cost, all customers benefit from the protections of AWS Shield Standard. Amazon CloudFront is easily integrated and works best with other AWS services, such as Amazon Simple Storage Service and Amazon API Gateway. For pricing, there are no fixed fees or long term contracts; payment is only for the data transfer and requests used to deliver content to customers [21].

5.2.4 Discussion

Cloudflare and Amazon seem to have opposite pricing systems. Cloudflare has a fixed pricing system, where the amount of traffic per month does not affect how much is paid. In contrast, for Amazon CloudFront has a variable pricing system, where payment is only for the amount of data transfer and requests. Akamai does not provide pricing information, and so a price comparison is not possible. All three have a high level of security, and so they would all be good choices for that factor.

5.2.5 Conclusion

The best CDN for this project is Cloudflare because it has a fixed cost, which means there will not be any spikes in cost in a given month. In addition, the amount of traffic to the ConnectBasket website is unknown, so a fixed cost is more reliable. Amazon CloudFront would be a good CDN, especially since it integrates well with other AWS services, but it seems to be the best choice only if other AWS services are being used, and we are not using other AWS services.

Criteria	**Cloudflare**	**Akamai**	**Amazon CloudFront**
Cost	Fixed cost per month	Unknown	Variable cost per month
Difficulty of setup	Very easy, taking less than five minutes	Easy, with 24/7 support from Akamai	Easy when used with other AWS services
Security	Secure	Secure	Secure

5.3 Web Development Frameworks

5.3.1 Overview

A web development framework is considered to be tools and resources used by software developers to create and manage websites [10]. Web development frameworks extend the capabilities of a language and provide libraries so developers do not have to start from scratch and hand-code everything. A framework may include libraries, APIs, security, and compilers, among other things [11].

5.3.2 Criteria

One important criteria for this project for web development frameworks is that it needs to be easily scalable to many different devices and sizes of devices. This is important because the ConnectBasket website will need to be accessible from staff's desktop or laptop computers as well as a variety of smartphone devices. Another key factor is it needs to be simple as opposed to cluttered with a large number of features that will likely go unused. This project is relatively simple, and so tons of features are not necessary. Additionally, a framework that is flexible would be best.

5.3.3 Potential Choices

5.3.3.1 ASP.NET

ASP.NET was created by Microsoft, and it is used to create interactive and data-driven web applications[22]. ASP.NET works with HTML to create dynamic web pages. One advantage of ASP.NET is that it provides built-in Windows authentication, so it is secure [23]. In addition, an ASP.NET application can be written in a variety of languages, including C# and Visual Basic [22].

5.3.3.2 AngularJS

AngularJS is a JavaScript based web application framework that is open source and currently maintained by Google [12]. It is fast for development and creates simple, dynamic websites [24]. AngularJS uses HTML as a template and extends HTML syntax to describe a website or application's components easily. AngularJS provides data binding capability to HTML, provides reusable components, and is unit testable, among many other features. In addition, AngularJS is available on all major browsers and smartphones. One disadvantage of AngularJS is that it is not secure, and so server side authentication is needed to keep an application secure [12].

5.3.3.3 ReactJS

ReactJS is a JavaScript library that makes it easy to build user interfaces by creating reusable UI components[25]. ReactJS involves thinking about everything as a component [25]. Advantages of ReactJS include that it can be used with other frameworks, and can be used on the client side or the server side [25]. One major disadvantage of ReactJS is that other tools will need to be combined with React to get a complete set of tools that is required for development [25].

5.3.4 Discussion

AngularJS and ReactJS are both JavaScript based, while ASP.NET is very flexible because it can be written in more than one language. AngularJS is useful because it is available on all major browsers and smartphones, but a downside is that it is not secure and would still require server side authentication. However, another component of the project is user authenication, so it is not required that the chosen web development framework contains secure authentication. In contrast, ASP.NET is secure and provides built-in Windows authentication. ReactJS would be reasonable for creating reusable components, but it would possibly require additional tools for successful development. AngularJS also can easily create reusable components.

5.3.5 Conclusion

AngularJS is the best web development framework for the ConnectBasket project. The main reason for choosing AngularJS is because it is designed to quickly produce dynamic and elegant websites. It is well-documented and maintained by Google, which makes it a sensible choice. Even though it does not have user authenication, that is something that can easily be implemented using another technology.

Criteria	**ASP.NET**	**AngularJS**	**ReactJS**
Flexibilty	Can be written in multiple languages	Works with all major browsers and smartphones	Works with all major browsers and smartphones
Complexity	Moderately complex	Simple	Simple
Integration with other tools	Works best with other Microsoft products	Easily integrates with other tools	Easily integrates with other tools

5.4 Software Design Patterns

5.4.1 Overview

Software design patterns are a important part of any software application. One very important feature of this project is to provide a user interface that will be accepted by users who are reluctant to change. This means how users are able to interact with the data is very important and modeling a system that closely reflects their current system in a more efficient way is critical.

5.4.2 Criteria

One major factor to consider when choosing a software design pattern for this project is how well the system will be able to model data in a way that our system needs it to. Users will be inputting data into the system, which must to be able to display that information to other users. Some data will be filtered for specific users, while some data will need to be displayed as an overall summary for other users.

5.4.3 *Potential Choices*

5.4.3.1 MVC

MVC is a software design pattern for implementing a user interface. It separates an application into three parts known as the model, view, and controller. The model is the structure of the data in the application and how the data is represented in the application. View is the component that makes the data displayable to the user in a useful way. The controller supplies an interface between the model and view components, taking the data from the model and converting it into something that can be used by the view component to display to the user. MVC supports the development of multiple different views for one model so the same information can be displayed in multiple ways if that is important for the application. It is also a very fast development process because one person or team can develop the view component while another works on the controller and another works on the model. This design pattern requires multiple developers for it to work properly [13].

5.4.3.2 MVP

MVP is a software design pattern that is based on similar concepts to the MVC design pattern. It separates an application into four different components that are responsible for determining how a user can interact with the system. Those components are the view, view interface, presenter, and model. The view is responsible for determining how the user will see the information displayed while the view interface connects the view to the presenter. The presenter connects the view to the model, which is responsible for the data that is displayed by the view. A view will usually only be connected to one presenter. Testing can be done easily with this design pattern because all interactions are done through an interface. Development is sped up by the fact that multiple programmers can split up the components and work on them separately [26].

5.4.3.3 MVVM

MVVM is a software design pattern that is an extension of the MVC design pattern. MVVM has four components, which are model, view, controller, and view model. In the MVVM pattern, view models convert data from the model layer into something that is usable by the view layer, which keeps the code for controllers from growing too large. In MVVM, the controller no longer depends on the model, so it is much easier to test. Views are only used to present the data they are given. The view controller interacts with both the view model and view layers and the view model also interacts with the model [27].

5.4.4 *Discussion*

MVVM and MVP are both similar to MVC, and provide some additional features compared to MVC. In contrast to the four components of MVVM and MVP, MVC only has three components which can make it a simpler and easier to use pattern. MVVM and MVP have an additional components which further modularizes the functionality of each component and makes it easier to test.

5.4.5 *Conclusion*

The best design pattern to use for this project is MVC. The user interface for this project will not be very complicated, so, while having additional components might improve testability, it will not be worth it for the increased complexity.

Having three components also provides the option for dividing the components between the three person development team, with one team member working on each component.

Criteria	**MVC**	**MVP**	**MVVM**
Components	Model, view, controller	View, view interface, presenter, model	Model, view, controller, view model
Number of components	3	4	4

6 Tech Review (Taylor)

6.1 Introduction

6.1.1 Purpose

The purpose of this document is to give an overview of what parts of the ConnectBasket project Taylor Kirkpatrick will have responsibility over, and analyze the technologies that may be used for each part. Each part will choose a specific technology that may be used in the final project, along with justification for why.

6.1.2 Scope

ConnectBasket will be a web based portal for the OSU Veterinary Hospital that will streamline the way messages are moved around the hospital and provide a better way to track the route that messages have taken. When an owner calls the hospital, receptionists will be able to type a message into the computer, assign it a category, and route it to the necessary staff members, which may be service specific vet techs, house officers, or faculty. Those staff members will be alerted, either by email or text, that they have a message waiting for them in ConnectBasket, and once viewed, notes can be added and the message can be re-routed to other staff members. After a message has been addressed by the necessary staff, the message can be closed preventing further notes or comments. The audit trail of when the message was created and closed will be visible, along with all the notes, timestamps of notes, and staff who added the notes.

6.1.3 Definitions, Acronyms, and Abbreviations

- Capstone - CS 461 class at Oregon State that the developers of ConnectBasket are in
- OSU - Oregon State University
- Hospital - referring to the OSU Veterinary Hospital
- Owner - a person who owns a pet that is a client of the hospital
- Patient - an animal who has/is going to be treated at the hospital
- VetHosp - the hospital management system used by the OSU Veterinary Hospital for keeping patient and owner records
- Mobile Devices - smartphones or tablets
- CPaaS - Communications Platform as a Service

6.1.4 References

References

[1] ”Userfrosting — Modern user management framework for PHP”, *Userfrosting.com*, 2017. [Online]. Available: https://www.userfrosting.com. [Accessed: 14- Nov- 2017].

[2] panique and others, HUGE, (2015), GitHub repository, https://github.com/panique/huge

[3] Microsoft, ”Introduction to ASP.NET Identity”, *Docs.microsoft.com*, 2017. [Online]. Available: https://docs.microsoft.com/en-us/aspnet/identity/overview/getting-started/introduction-to-aspnet-identity. [Accessed: 14- Nov- 2017].

[4] Trello, ”Trello”, *Trello.com*, 2017. [Online]. Available: https://trello.com/. [Accessed: 14- Nov- 2017].

[5] Best Practical, ”Request Tracker”, *Best Practical Solutions*, 2017. [Online]. Available: https://bestpractical.com/request-tracker. [Accessed: 14- Nov- 2017].

[6] Twilio, ”Communication APIs for SMS, Voice, Video and Authentication”, *Twilio.com*, 2017. [Online]. Available: https://www.twilio.com. [Accessed: 14- Nov- 2017].

[7] Nexmo, ”Nexmo - APIs for SMS, Voice and Phone Verifications”, *Nexmo.com*, 2017. [Online]. Available: https://www.nexmo.com/. [Accessed: 14- Nov- 2017].

[8] M. Grech, ”Twilio vs. Nexmo: The Head to Head Showdown for Developers — GetVoIP”, *Getvoip*, 2017. [Online]. Available: https://getvoip.com/blog/2017/01/05/twilio-vs-nexmo/. [Accessed: 14- Nov- 2017].

6.1.5 Overview

The rest of the document will contain three sections with an overview of the parts of the ConnectBasket project Taylor Kirkpatrick is responsible for. Each section will contain five subsections, three to analyze specific technologies, one to compare them, and one to state the chosen technology with project-specific reasoning.

6.2 User Logins/Profiles

A feature highly anticipated by the client, users must have persistent, personalized states to distinguish them from other users when tasks are to be passed on. These user profiles must be private and secure, one to each user. Messages may be user-specific and carry information that is confidential or otherwise not to be made accessible to everyone, even within the company. This part specifies what framework or project will be built off of to create these users and their logins and profiles. Of note is the possible overlap this system may have with the issue tracking technology, as those are usually distributed as whole projects, including user logins. While we do expect to be using one of these frameworks, it may clash with what the issue tracker framework brings to the table.

6.2.1 Userfrosting

Userfrosting is a complete and easy to use framework for PHP. It depends on the PHP tool Composer and also makes use of Node.js and Bower for client-side packages. The framework fully automates and handles persistent user sessions without requiring the dev team to handle sessions manually between many potentially dynamically updating pages. Userfrosting also adds easily customizable and rather nice looking user profile pages, allowing a dev to simply present some information to users under the conditions that they desire. In addition, this framework provides simple methods for a dev to create user roles for role based control, and assign users to them. Lastly, the framework allows the creation of a user administration page for user operations. This interface can be easily built for non-technical users, and ease the process of user administration for them.[1]

6.2.2 HUGE

HUGE is a PHP framework for doing simple user authentication. HUGE came about in a time when PHP was lacking many quality login solutions, and was built primarily off of the password_compat library once it was released. HUGE is no longer in active development, but has remained an excellent choice when facing up against many other PHP frameworks. HUGE makes use of the official PHP password hashing functions, with many other nice features that one would expect in a login framework like email account verification, form encryption, a ”remember me” function, etc. HUGE also offers many other nice features like URL rewriting (masking possibly sensitive info), and supports user portraits already. HUGE requires PHP 5.5 or above, a MySQL database, a handful of PHP extentions, a few developer tools, and software for sending mail if desired.[2]

6.2.3 ASP.NET Identity

ASP.NET includes a relatively robust user login API for use in ASP.NET projects. As this project may be done in ASP.NET, this is something worth checking. As the web became wider and demanded even better user management support, ASP.NET created the framework ASP.NET Identity to keep up with alternatives, offering full support with other ASP.NET packages along with the expected features in a login framework. Playing well with other ASP.NET packages is not to be underestimated, however, as a project built in ASP.NET is sure to have numerous other packages. The framework does, of course, support simple login features, along with more advanced things like social media integration and a persistent ”remember me” cookie login, and role based control.[3]

6.2.4 Comparison

At a first glance, it appears that ASP.NET Identity lacks features compared to the other two options, and this observation is correct. However, it does boast the reliability of Microsoft’s products, and development with it in Visual Studio would be easier than the other two. Between the other two, Userfrosting is much more hands-off than HUGE, and while doing less complex and error prone work is nice, it also restricts the freedom of the developers. HUGE and Userfrosting have very similar features available, but HUGE may even boast more features. However, HUGE was never really formally released as anything other than a simple open source framework. Identity and Userfrosting are both more formal projects, and may not be as bug prone.

6.2.5 Selection

The HUGE framework is the tentative winner in this category. It is a happy medium between the hands-off GUIs of Userfrosting and the mostly manual configuration of ASP.NET Identity. HUGE has roughly the same number of features as the other two combined, is much better documented than both, and has many years of fellow users for support that Identity may not have, and Userfrosting is unlikely to have. This victory comes with a caveat, however, as it is dependent on what language the software is in. Should the dev team be unable to use PHP, or otherwise choose not to in favor of ASP.NET or something else, HUGE would be useless for the project. Of the three present here, however, it is the better of them, while the others provide options should other development decisions invalidate HUGE as an option.

6.3 Issue Tracker

The primary goal of this project is, in essence, to create a create a custom issue tracker, personalized for the veterinary hospital. The bulk of the requested features for this project are in tune with making a non-IT bug or feature tracker. The features requested are both standard issue tracker features and more specialized items as well. Thus, finding a framework or at least a simple, freely licensed issue tracker to build off of would simplify the development of the bulk of the features and reduce room for error.

6.3.1 Trello

Trello is a popular kanban style scrum team management application. It is currently owned and maintained by Atlassian, who also own the similar software JIRA, which is more technical and is not free. Trello allows easy ”whiteboard style” organization for tasks, and can easily be applied to non-technical uses. Trello also has a robust API available to the public, and is what the developers use for further work on the product, proving the power of the API. Trello’s interface is easy to read and navigate, and interaction is simple and intuitive. Trello also works with mobile devices, supporting most modern mobile browsers as well as desktop.[4]

6.3.2 Request Tracker

Request Tracker, made by Best Practical, is an issue tracker designed for general use, instead of technical users. Request tracker has a very clean interface, with simple messaging and includes user login support by default. Request Tracker is an entirely open source project, and the community and developers seem to be quick to fix discovered bugs or issues. Request Tracker boasts email integration and allows users to interact fully with the system through email only, not even needing to open the application to resolve or interact with tickets.[5]

6.3.3 Manual Issue Tracking

This option involves creating a bare-bones, incredibly simple request tracker in-house. This option would have exactly the features desired and would integrate the easiest with the rest of the project, with the caveat of taking a significant amount of time. However, when parts of the project are still unknown, this is guaranteed to be the most fluid option, as the development team can change or customize anything they want at any time.

6.3.4 Comparison

Creating the request tracker manually is a very valid option, as a major problem some of the other issue trackers may have is the extent of the features they have that are mostly unneeded. Trello, while probably the most reliable and one of the easier options is especially guilty of this, as users would hardly benefit from the visual style of a board, and tickets being open and unsecured may prove to be problematic. Request Tracker is a good middle ground, as it is not bloated with unnecessary features, but does not require a custom built system.

6.3.5 Selection

Request Tracker would be the ideal here. It contains almost exactly the features the client is looking for, and has bells and whistles that the users might appreciate without being overwhelmed by them. Trello is much too bulky, and the kanban style is not very applicable in this case. If Request Tracker can be integrated with what the client wishes, or built off of into a new application, this would give the users the most functionality while reducing the risk of human error and more bugs that would be inherent in manually creating the issue tracking.

6.4 Notifications

The client has specifically requested smartphone notifications when notable activity takes place related to a specific user. Users should be able to be alerted to any activity related to their user profile or role on their smartphone, should they so wish. The client has not specified how this notification is to take place, and so could be up to which technology is chosen. The initial idea for notifications, though alternatives were considered, was using SMS, as it does not depend on wifi connection, and anyone with a functioning phone would have the ability to receive such a notification, even without a smart phone. Push notifications, while pretty and useful, could be impractical or impossible as the ConnectBasket software will be a web application rather than a native mobile application.

6.4.1 Twilio

The Twillio tool allows an application to send SMS messages regardless of provider to a user, among other phone related features such as programmable calls and call redundancy allowing for more reliable calls and messaging. Twilio has a nice looking interface and easy to understand setup, and offers unique features like voice call support and robust message quality of life improvements like sticky sender and MMS support. Pricing for Twilio varies, and while they offer free API keys for limited use, many features even for small uses are not free.[6]

6.4.2 Nexmo

Nexmo, recently acquired by Vonage, boasts all of the features one could expect from a CPaaS tool. It allows a user to send messages through most providers and includes useful functions in the API for number screening, allowing a service to catch spam or malicious numbers before devoting resources to them. Nexmo also offers a real-time chat function to interact with users should such a situation be desired. Nexmo benefits from the resources of their parent company, able to operate in 80 countries, supporting many popular chat systems, and offers free inbound texts. Pricing is on a per-text basis, but does not contain a free trial for most features.[7]

6.4.3 Manual Email

This option forgoes the external tools and the use of SMS entirely. Most email providers allow for drafting and sending off a simple email to a target address. While this option would not directly solve the issue of mobile notifications, it could be used to take advantage of an existing feature most smartphones have, email notifications. Notifications could be delivered by the user's email application, giving the user more control over when or what notifications they would like to see, and also should be free.

6.4.4 Comparison

Without the option of push notifications, which would be ideal, pricing and immediacy are the most important things. Twilio and Nexmo are rivals in the CPaaS industry, and have similar pricing plans that compete with each other. However, neither of them are free for the purposes of this project, as both lock messaging tools behind a ”pay per text” plan. Emailing is free and may likely even be easier than the other two to use, but lacks the immediacy that a SMS provides and also may not be received if the user lacks a wifi connection.[8]

6.4.5 Selection

Manual Emailing is probably the best choice here. While it is possible that the client or a member of the college would be willing to pay for either Twillio or Nexmo, the features they offer are overkill for this project, and these notifications may not even be used by many of the users. Those that do, however, would likely not appreciate texts every time there is any activity on a ticket.

7 Weekly Blog Posts

7.1 Henry

7.2 Kailyn

7.3 Taylor

8 Final Poster

COLLEGE OF ENGINEERING

Electrical Engineering and Computer Science

CS39

INTRODUCTION AND BACKGROUND

Having an efficient and easy-to-use software system for managing operations is integral for any company, and the Oregon State University Veterinary Hospital is no exception. The current software used by the hospital to manage their daily operations is decades old and doesn't allow efficient communication within the hospital. This software causes many unnecessary errors and patient and owner suffering that could be avoided with a better system. The hospital needs a new software package that will not only provide a system that doesn't hinder workflow, but will improve it and allow for increased efficiency.

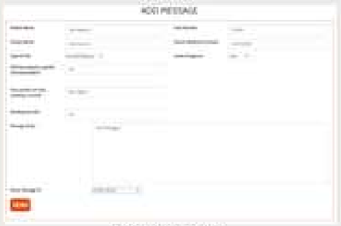
PROJECT GOALS

- Allow users to create messages containing information such as case number, message category, level of urgency, patient name, and owner name and contact information
- Allow messages to have a status to tell whether they are new, in progress, or complete
- Allow messages to be routed to a group of users
- Allow notes to be added to messages
- Allow users to have profiles where they can select or change their group membership
- Provide an audit log that shows user actions such as creating, viewing, or claiming messages
- Increase the overall efficiency of messages being transported throughout the hospital
- Reduce the amount of paper being printed
- Reduce errors in the communication process of the hospital

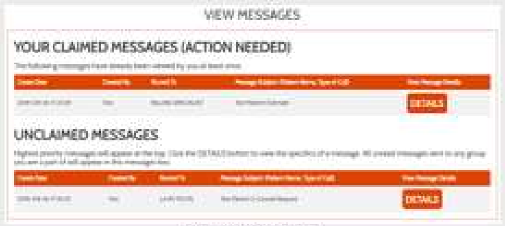
Oregon State University

CONNECTBASKET

Software to help improve pet care at the OSU Veterinary Hospital by streamlining internal hospital communication

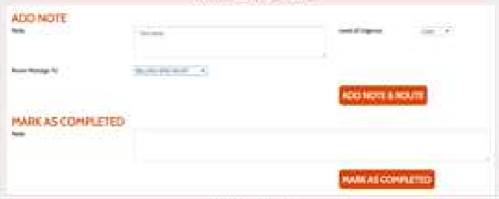


Add Message Page



View Messages Page

Audit Log Page



Add Note Page

PROJECT DESCRIPTION

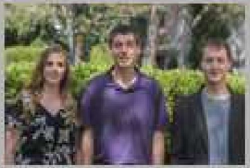
In order to accomplish the goals of this project, the ConnectBasket website was created. The website provides a platform for hospital employees to log in and access several features such as creating, editing, and viewing messages.

The website is hosted using an Apache web server running on a Linux Ubuntu server operating system. The data for our website is stored in a MySQL database and PHP was used for interaction between the website and the database. The front end of our website was made using the AngularJS framework along with HTML and CSS.

We chose to create a website for our project because the users wanted to be able to access the project from desktop computers and mobile devices, and a website was the best way provide that in the time available for the project.

PROJECT OUTCOMES

- Using a digital solution will provide the hospital with a decreased use of paper and ink, as well as providing a better system for communicating quickly back and forth across the hospital.
- The new system will provide a way for administrators to make sure the proper workflow is being followed by employees. They will be able to see when messages are being viewed or claimed with no action being taken and correct any mistakes that are being made for the future.
- Having access from mobile devices allows employees to be quicker in responding to messages providing faster and better care for pets.



Pictured from left to right:
Kailyn Hellwege, Henry Fowler, Taylor Kirkpatrick

Project Client:

Dr. Chinweike Eseonu

Affiliation:

Oregon State University Veterinary Hospital

Contact Information:

fowlerh@oregonstate.edu

hellwegk@oregonstate.edu

kirkpatt@oregonstate.edu

RESULTS

ConnectBasket users can view a history of all messages that are marked as new, in progress, or completed, and view details of the messages. Completed messages are read-only and no further notes can be added.

After a message has been completed, it can be exported as a pdf file that contains all of the original message information as well as all of the notes that have been added to the message.

9 Project Documentation

10 Recommended Technical Resources for Learning More

10.1 Helpful Websites

- https://docs.angularjs.org/guide/introduction
- https://www.w3schools.com/
- http://php.net/
- https://dev.mysql.com/doc/

11 Conclusions and Reflections

11.1 Henry

11.1.1 What technical information did you learn?

11.1.2 What non-technical information did you learn?

11.1.3 What have you learned about project work?

11.1.4 What have you learned about project management?

11.1.5 What have you learned about working in teams?

11.1.6 If you could do it all over, what would you do differently?

11.2 Kailyn

11.2.1 What technical information did you learn?

11.2.2 What non-technical information did you learn?

11.2.3 What have you learned about project work?

11.2.4 What have you learned about project management?

11.2.5 What have you learned about working in teams?

11.2.6 If you could do it all over, what would you do differently?

11.3 Taylor

11.3.1 What technical information did you learn?

11.3.2 What non-technical information did you learn?

11.3.3 What have you learned about project work?

11.3.4 What have you learned about project management?

11.3.5 What have you learned about working in teams?

11.3.6 If you could do it all over, what would you do differently?

12 Appendix 1: Essential Code Listings

13 APPENDIX 2: PHOTOS OF PROJECT

References

[1] K. Hess, "Ubuntu server: The linux server operating systems dark horse." https://www.serverwatch.com/trends/article.php/3870141/Ubuntu-Server-T
3 2010.

[2] D. Sullivan, "Advantages of apache web server." https://www.techwalla.com/articles/advantages-of-apache-web-server.

[3] T. Branson, "8 major advantages of using mysql." https://www.datamation.com/storage/8-major-advantages-of-using-mysql.html, 11 2016.

[4] M. Grech, "Twilio vs. nexmo: The head to head showdown for developers." https://getvoip.com/blog/2017/01/05/twilio-vs-nexmo/, 1 2017.

[5] panique *et al.*, "Huge." https://github.com/panique/huge/, 4 2017.

[6] "Modern user management framework for php." https://www.userfrosting.com/.

[7] B. Practical, "Request tracker." https://bestpractical.com/request-tracker/.

[8] "What is a cdn?." https://www.cloudflare.com/learning/cdn/what-is-a-cdn/.

[9] "Fast, global content delivery network." https://www.cloudflare.com/cdn/.

[10] M. Rouse, "web development framework (wdf)." http://searchcontentmanagement.techtarget.com/definition/web-development-framework-WDF.

[11] C. Wodehouse, "What is a framework?." https://www.upwork.com/hiring/development/understanding-software-frameworks/.

[12] "Angularjs - overview." https://www.tutorialspoint.com/angularjs/angularjs_overview.htm.

[13] Jithin, "What is mvc? advantages and disadvantages of mv." https://www.interserver.net/tips/kb/mvc-advantages-disadvantages-mvc/, 10 2016.

[14] O. Rist, "Windows server top 5 benefits for todays workforce." https://blogs.technet.microsoft.com/windowsserverexperts/2010/02/08/windows-serve
2 2010.

[15] K. Milberg, "When to use solaris vs. linux: Operating system comparison." http://searchdatacenter.techtarget.com/tip/When-to-use-Solaris-vs-Linux-O
12 2008.

[16] A. L. Ortega, "Cherokee basics." http://cherokee-project.com/doc/basics_why_cherokee.html.

[17] A. Leslie, "Nginx vs. apache (pro/con review, uses, & hosting for each)." http://www.hostingadvice.com/how-to/nginx-vs-apache/, 7 2017.

[18] O. Tezer, "Sqlite vs mysql vs postgresql: A comparison of relational database management systems." https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems, 2 2014.

[19] G. Farrar, "Microsoft sql server: Advantages and challenges from the virtualization admins view." https://turbonomic.com/blog/microsoft-sql-server-advantages-challenges-virtualization-admins-view/, 8 2016.

[20] "Cdn services." https://www.akamai.com/us/en/resources/cdn-services.jsp.

[21] "Amazon cloudfront." https://aws.amazon.com/cloudfront/.

[22] "Asp.net - introduction." https://www.tutorialspoint.com/asp.net/asp.net_introduction.htm.

[23] A. Hasan, "Top 12 main advantages of asp.net framework." http://www.arpatech.com/blog/top-advantages-of-asp-net-framework/, 3 2017.

[24] "Angularjs." https://angularjs.org/.

[25] "React." https://www.tutorialspoint.com/reactjs/reactjs_overview.htm.

[26] T. Snyder, "Mvc or mvp pattern whats the difference?." https://www.infragistics.com/community/blogs/todd_snyder/archive/2007/10/17/mvc-or-m
10 2007.

[27] B. Jacobs, "What are the benefits of model-view-viewmodel." https://cocoacasts.com/what-are-the-benefits-of-model-view-viewmodel/, 11 2016.

[28] N. Rozanki and E. Woods, "Viewpoints." https://www.viewpoints-and-perspectives.info/home/viewpoints/.