Oregon State University
College of Engineering

CS Capstone Winter Midterm Progress Report

February 16, 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 describes the progress made by Capstone Group 39 in the development of the ConnectBasket web application. A brief overview of the project will be provided as well as a summary of the work completed so far and work still left to complete. There will be sections describing in detail the work done by each individual group member.

CONTENTS

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.

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

3 Current State of Project

The project in its current state provides a basic login functionality, allowing a user to create an account and then login and see a home page with a welcome message. The website can be accessed from on-campus networks only at the url, http://vm-cs462-g39.eecs.oregonstate.edu. Users are stored in a MySQL database that is on the virtual machine which is running the website. There are templates set up for other pages that will be added to the website in the future, but they are not currently functioning. Code has been written to enable notifications to be sent to users but has not been implemented in the website yet.

4 Future Plans for Project

There are still several requirements that need to be met in order for the project to be complete. Key features that still need to be implemented include adding a new pet and owner, adding messages, adding notes to messages, viewing messages, routing messages, notifying users, and keeping a log of all website activity. The client needs to confirm some of the details of these pages before they can be fully implemented. By week eight, there will be a prototype that implements most or all of these features and by the end of the term, all of the features will be implemented with no major bugs that prohibit use of the website.

5 Problems Encountered

One issue that still exists for the development team is the office politics that exist at the hospital. The project for the Capstone class was created because the current IT team at the hospital was taking too long to implement a new communication system into their current system, and the IT team has not been informed of the Capstone project. Due to this, the development team has not been able to communicate with the IT team at all, which has made it more difficult to design a system that will work well with what the hospital is currently using. Ideally, the Capstone project will integrate easily with the VetHosp system that is currently used, but without any communication with the IT team or knowledge about the current system, the Capstone project will be a stand alone system that the hospital's IT team will be responsible for integrating.

Another large hurdle the development team has been facing is communication with the client, Dr. Eseonu. During the fall, we did not get very quick responses from our client, but we usually got a response sometime within one to two weeks of sending an email. However, near the end of fall term and all of winter term so far, we have not received any responses to any of the emails we have sent. We have been in contact with Kelly Warner, the process improvement manager at the hospital, and she has also had a hard time getting in contact with Dr. Eseonu. This has made scheduling meetings as well as getting our documents approved very difficult. To solve this problem, we are treating Kelly as if she were our client, and having her approve the documents instead of Dr. Eseonu until we hear back from him.

6 Development

6.1 Henry

6.1.1 Database Design

While waiting for our group to get a machine to use as a web server, I started by working on the design of the database that we will use for our project. Using the information gathered from meetings last term and at the beginning of this term, I started to design the tables that will be needed for the database for our project as well as some of the queries that will be used by our website.

I started with a users table, which will store information about the veterinary hospital employees that will use our website. This table will store the first and last name of the employee as well as a username, password, and email address for that employee. The username and password fields will be used to allow users to login to the website and the first and last name will be used to give personalized messages after logging in. Passwords will be hashed before being stored in the database, so the password field will not hold the actual text the user types in for their password. An email address will be necessary in order to send notifications to the users of the website when they have a message waiting for them to take some action with.

The next tables that were needed for the project were a pets table and an owners table to store information about the pets treated by the hospital and the owners of those pets. The exact information to be stored in these tables has not been finalized, but it includes information about pets species, size, breed, and age, and owners contact information and names. There will be a third table that connects potentially one or more owners to one or more pets.

The final tables that I included in the initial design were for the storage of messages and any edits that are made to messages. The information stored will include the message entered initially, who entered it, when they entered it, and who they sent it to. Another table will hold all of the additional notes added, who added them, what message they are attached to, and who if anyone the message was additionally routed to.

After a meeting with our contact at the veterinary hospital, I added a new table to the design that would hold log messages of every important action taken by a user. This table will store the username of the person who took the action, what they did, and when they did it. Actions could include creating a new user, pet, or owner, adding a new message, adding a note to a message, or viewing a message. This will allow managers to see if employees are using the system correctly and if there is a mistake, they will be able to go back and see where things went wrong.

6.1.2 Web Server Setup

The first task in completing our project was to set up a web server to host the website. After receiving access to our virtual machine on the Oregon State Engineering servers, I installed a Linux Ubuntu Server operating system on the virtual machine. To do this, I downloaded an ISO image of the operating system from the Ubuntu website and then installed the necessary tools to launch the virtual machine from the vSphere web client with the ISO mounted in order to install the operating system.

After going through the steps to install Ubuntu Server, I created users for each of the members of our group and enabled ssh connection for each of us, so that we can ssh to our machine when on a campus network or using a vpn. Once the user accounts were set up, I installed Apache web server software, MySQL server, and PHP on the server. This process involved setting up a root user for MySQL in order to access the database and then giving permissions to other users to create tables in the database. In order to help manage our database, I installed PHPMyAdmin on our server to provide a graphical interface for managing the tables and data stored in them. Using this tool, I started to create some of the tables that were a part of my database design I made earlier in the term.

After the database was set up, I installed git and cloned our group's repository onto the server. I then changed the default path for Apache to look for code to run to be the directory in our git repository where our website code is, so that when someone accesses our server through a web browser, they will see our website.

6.1.3 Website Database Interaction

After the web server was set up, I tested to make sure that we could reach a basic hello world page on our server. Then I tested it with the pages that had already been made for our project website. Once I was able to reach the login and create user pages that were finished, I started to work on writing the PHP code that we will use to have our website interact with the database.

I started with the code for creating a new user for our system. This works by taking a first and last name, username, password, and email address that were validated client-side and using an insert statement to add the new user into the users table of the database. Before inserting the new user's information the PHP password hashing function is called on the password that is entered so that the actual password is not stored in our database. After the new user is inserted, I run a query that checks to see if a user exists with the username entered, just to make sure it was successfully created. If the creation was successful, a value is returned to the website telling it the user was successfully created and to reroute to the Login page.

The next code to write was for the Login page. This simply took a username and password from the login form and ran a query looking for a user that matched the username entered. It then checks to see if any rows were returned and if there were checks the password using the PHP password verify function to compare with the hashed password that was stored in the database. If the verification is successful, then a valid authentication as well as some information about the user is returned to the page, which can then redirect to the home page. An example of the function to handle logging in is shown in the code below.

```
$stmt = $conn->prepare('SELECT Username, FirstName, LastName, EmailAddress, Password FROM Users WHERE Us
$stmt->bind_param('s', $username);

$username = $data->username;
$password = $data->password;
$hashpass = password_hash($password, PASSWORD_DEFAULT)
```

6.2 Kailyn

6.2.1 HTML Pages and Navigation Organization

Once the server was set up, the first step for creating ConnectBasket was to actually create the HTML pages for the website where the content will go. I created a Home page, Login page, Create User page, Create Pet page, Create Owner page, Add Message page, View Messages page, and Edit Profile page.

The next step was to create a navigation bar and organize the pages in a way that would easily make sense to users. To start with, there needed to be a way for users to login to the website so they would be using their own account. When someone is not logged in, the only tab they will be able to see is the Login tab, because if they are not logged in, there should not be any messages for them to see and they should not be able to send any messages. The Login page is very simple and just has entry boxes for the user to type their username and password, and then they can click the Login button.

Once a user is logged in, more tabs will be visible for them. Instead of a Login tab, they would see a Logout tab. In addition, there will be a Messages tab, Edit Profile tab, and Admin tab, if they have permission to see it. If the Messages tab is selected, a dropdown will pop up with two more pages: Add Message and View Messages. The main purpose of ConnectBasket is creating, viewing, and sending messages, and so the Messages tab will be visible to anyone who is logged in. From the View Messages page, users will be able to see all of their new messages as well as old messages, and view any details, such as who created it, when it was created, and any notes added, about the messages by clicking on one.

As I was sketching out a layout for the View Messages and Add Message pages, I realized that we are still missing some important information that is necessary to plan out these two pages. While I tried to sketch a plan for these pages, I compiled a list of questions and missing information that we need to complete these pages. We have contacted Kelly Warner at the hospital to get answers to these questions as soon as possible.

Certain tabs will only be visible to certain users. For instance, creating a user, owner, or pet is not something that every employee should be allowed to create. Therefore, there is an Admin tab that will be limited to those with permission. The Admin tab has a dropdown that contains the Create User, Create Pet, and Create Owner pages. The Create User page is fairly simple, as it is just a form to enter a user's name, email, username, and password. At the bottom of the form, there is a submit button, which will send the information entered to the database to be saved as a user. The Create Owner and Create Pet pages will have a similar setup.

The last page created was the Edit Profile page. The intent of this page is for users to be able to change which groups they are members of, in case an employee changes departments or is helping with a project somewhere else in the hospital. Currently, we do not have all the information from the hospital about what exactly they want on this page, but it will have some kind of checklist where users can simply check or uncheck groups they want to or do not want to be a part of. The groups that they are a part of will determine which messages are sent to them.

6.2.2 Logo and Icon Design

Since the ConnectBasket website is a stand-alone application that is not directly going to be a part of the hospital's current system, a logo seemed necessary to give ConnectBasket a recognizable brand. Using Microsoft Publisher, I created a logo using an image of a basket followed by the name of the website, ConnectBasket, which is shown in Fig. 1.

Fig. 1. This image shows the logo designed for the ConnectBasket website.

Fig. 1 was chosen as the main logo for ConnectBasket because the overall shape was long and thin, which is an ideal shape for placing in the navigation bar of the website. Additionally, keeping the logo in the navigation bar and keeping the navigation bar thin leaves a majority of all of the pages' space open for the content of each page to prevent as much scrolling as possible for the user. In addition to the first logo, I created two more variations that have slightly different layouts and shapes, but the same colors, fonts, and images (Fig. 2).

Fig. 2. This image shows two alternative logos designed for the ConnectBasket website.

In addition to creating ConnectBasket logos, I also created some icons that can be used on the main page. The main functions that will be used on the ConnectBasket website are creating a new message and viewing all messages. Editing one's profile is another function that should be easily accessible as well, so an employee can easily change which groups they are members of. Because these will be the three most used pages, we decided to create quick links to those pages from the home page. Each icon is colored differently, so users will be able to associate a page with a specific color. I considered making the icons the same color for consistency, but similar icons can easily be confused for each other, which led to the decision of different colors for each. The three icons are shown below in Fig. 3.

6.2.3 AngularJS

The ConnectBasket website will be a Single Page Application (SPA), meaning that the entire page will never be reloaded; only specific sections of it will. There is a directory of template files that hold the HTML for a header and footer that are included using the AngularJS ng-include directive on every page of the site. The content in between the header and footer is updated using the AngularJS ui-router feature, which allows for states to be changed on certain actions such as clicking a link or button. These states correspond to different HTML files which allow the content on the screen to be

Fig. 3. The three icons are shortcuts to the most frequently visited pages.

dynamically updated without reloading the entire page. An example of the code that handles state changes is shown below.

```
.state('home', {
        url : '/home',
        templateUrl : 'modules/Home.html',
        controller : 'HomeController'
      })
```

Each of the states and accompanying HTML have a corresponding controller that is defined in a sitewide JavaScript file. The only controllers that have been completed so far are for the Login and Create User pages. They are responsible for validating the input in the forms and displaying error messages if the content is not valid. In addition to validation, the controllers call the PHP functions that interact with the database and take some action based on the value returned from those functions.

6.3 Taylor

6.3.1 Notifications

Aside from assisting the group work and documentation, my first task was the notification feature. As decided in the technology review, the notifications will be in the form of emails as opposed to SMS or smart phone notifications due to a number of reasons including freedom of message design, user friendliness, and complexity. This was the easiest feature to work on while the group did not have a server, and so I started with it.

My original design and basic implementation of the notification system involved a PHP script sending off a pre-made message to an email address. The message body contained the body of the ConnectBasket message with a wrapper, and the recipient was the user's registered email address, both to be pulled from the database once it was operational.

While this feature was mostly completed in this way and tested locally on my desktop, I soon discovered that I would have issues triggering the PHP script to run. While looking into solutions for this, I quickly realized that it is reasonably foolish to attempt to include this feature in the front-end of the site when the backend Unix server has a comparable solution already, requiring fewer interactions between scripts and systems.

The new notification system is very similar, but uses a cron job to repeatedly check the database for new entries. If a new entry should be found, the message is assembled with the wrapped and a unix mailx command is executed like so:
mailx -s New ConnectBasket Messages [user email] ¡ message.txt
Where [user email] is the user's email as gathered from the database. Further work on this feature will include a better method of checking for differences in the database, and refining how often the cron job should run.

6.3.2 Messaging System Design

Design for the messaging system is complete and will enter development soon. Initial assumptions about this feature likened it to a boardless Kanban board or issue tracker. After initial designs and some interaction with the client, however, it was found that this was an overestimation of the feature. The core elements of this feature are the messages themselves, the user queues, and the auditing.

Messages will be stored plaintext in the database in a table that also contains the author's information, the timestamp information, the recipient'ss information, and any notes that have been added to the message. Forwarding messages to other will log the former recipient/sender pair, and simply move the message to the next user.

Auditing information, as currently designed, will not require significant changes to how regular messages work. Read times will be logged and stored in the database per message, and all information on notes and send times are already recorded anyway. Changing the status of a message, making notes, and forwarding the message will all already be timestamped, and can be made available to a user with higher permissions. To further refine how a user should view this and which users should have access, the client will be contacted and asked to assist.

User queues are the most complex element of the messaging system, and the design will probably require a review once the basic messaging system is implemented. As the design is for now, a user will be able to click a button viewable from their regular homepage where they view messages sent directly to them. This will show them any queues they are a member of, and any unclaimed messages sent to the queues. The user will be able to claim a message which will be removed from the queue visible to all and will instead be sent to the user who claimed it as a regular message.