Homework 6

Mohammed Algadhib
algadhim@oregonstate.edu
Riley Kraft
kraftme@oregonstate.edu
Diego Saer
saerd@oregonstate.edu
Johnny Chan
chanjoh@oregonstate.edu

1. SOFTWARE

URL:
https://github.com/cs361-su18-group8/FinalProject

INSTRUCTIONS:

Pre-conditions: In order to run and test this web application the user should have Node and a web browser installed.

Download the application: To download the application there are two options, the first is by cloning the repository using git by executing this command `git clone <URL>. The second method is by downloading a .zip file of the software from GitHub, this can be done by navigating to the software's URL and then clicking on `Clone or download` green button, then clicking on `Download ZIP`.

Install dependencies and packages: In order to run the application successfully, dependencies and required packages should be installed first. This can easily be done using node package manager. To install the missing packages run the command `npm install` in the home directory of the project using any command line interface (PowerShell, CMD, Bash, etc.). After executing this command all required dependencies will be downloaded.

Run the app: After installing all the required dependencies, now it is possible to run the application. To run the application simply execute the command `node app.js` in the home directory of the project. If there are no error messages, the app is running.

Start using the app: The app can be used now by visiting http://localhost:3000

2. USER STORIES DUE

2.1 USER STORY A: DAILY HEALTH QUESTIONS

a. TEAM

Diego (pilot) and Mohammed (co-pilot)

b. TASKS

i. A1: Set up database to store surveys, questions, and answers.

1. Unit Tests
   - Tables reviewed to see if they make logical sense and can store required information correctly.
2. Problems
   - The team had never used the database software before
   - The design took multiple iterations many initial table designs would either duplicate much of the information or were inadequate to correctly hold the required information.
3. Time

   2 hours for the pair.

4. Status

   Completed

5. What's Left?
   - N/A

ii. A2: Come up with survey questions and add them to the database

1. Unit Tests
   - N/A
2. Problems
   - N/A
3. Time
   - N/A
4. Status
   - Not started.
5. What's Left?
   - Research possible questions.
   - Create a list of questions.
   - Add the questions to the database.

iii. A3: Create a homepage icon for the survey.

1. Unit Tests
   - N/A
2. Problems
   - N/A
3. Time

   N/A

4. Status

   Not started

5. What's Left?

- Research sources for icons.
- Integrate the icons to the homepage.

iv. A4: Create web page to display user instructions.

1. Unit Tests
   - From the homepage, click on “Take Survey” link.
   - The instruction page should be displayed.
   - Verify that the instructions are clear and easy to see and read.
   - Click on “Return to homepage” link.
   - Verify that the current page now is the home page.
   - The test pass if the the instructions on the instructions page are clear and the links are functional.
2. Problems
   - Some of the CSS styling elements were not being displayed correctly.
3. Time

   1 hour for the pair.

4. Status

   Not completed.

5. What’s Left?
   - Resolve styling problems, and make the homepage look more professional.
   - Awaiting client input on design.

v. A5: Create the web page user interface to answer survey questions.

1. Unit Tests
   - N/A
2. Problems
   - N/A
3. Time

   N/A

4. Status

   Not started

5. What’s Left?
   - Implement the survey interface that presents the questions to the user.

vi. A6: Write code to take user from homepage to survey page.

1. Unit Tests
   - Tested to see if the homepage link to the survey page functioned correctly.
   - Tested to see if one could return to the homepage from the survey page.
2. Problems
   - none
3. Time

   0.5 hours for the pair.

4. Status

   Completed

5. What’s Left?
   - N/A

vii. A7: Write code to retrieve questions from database.

1. Unit Tests
   - N/A
2. Problems
   - N/A
3. Time

   N/A

4. Status

   Not started

5. What’s Left?
   - Implement the code functionality that queries the questions from the database.

viii. A8: Write code to present questions one at a time.

1. Unit Tests
   - N/A
2. Problems
   - N/A
3. Time

   N/A

4. Status

   Not started

5. What’s Left?
   - Implement the functionality that presents the questions to the user.

ix. A9: Write code to record answers for database.

1. Unit Tests
   - N/A
2. Problems
   - N/A
3. Time

   N/A

4. Status

   Not started

5. What’s Left?
   - Implement the code that stores the answers entered by the user to the database.

x. A10: Create confirmation web page.

1. Unit Tests
   - N/A
2. Problems
   - N/A

3. Time

   N/A

4. Status

   Not started

5. What's Left?
   - Create the confirmation webpage.
   - Add link functionality to navigate to and from the confirmation webpage.

2.2 USER STORY K: MANUALLY ENTER MEDICATIONS

a. TEAM

Riley (pilot) & Johnny (co-pilot)

b. TASKS

i. K1: Set up databases to record medication object.

1. Unit Tests
   - Co-pilot reviewed tables against use case, user story, UML diagram, and sequence diagram.
2. Problems
   - Pilot realized we need two tables to cover this data; one for the parent class Medication, and another for the inherited Prescription class
   - Co-pilot found no reference in tables to account ID; not critical at this juncture; needs to be addressed after main functionalities are in place
3. Time

   pilot: 1 hour to create tables

   co-pilot: 30 min to test tables

4. Status

   Completed. For the purposes of this user story, the database requirements have been met.

5. What's Left?

   For the purposes of this user story, there is nothing left to complete for this tasks. However, in the future the team must take into consideration multiple users, after main functionalities are completed.

ii. K2: Create add prescription and add over the counter homepage icons.

1. Unit Tests
   - N/A as homepage style was not defined when this task was assigned to be done
2. Problems
   - Usability and overall style were not yet defined from user story L. This is a work in progress, thus the icons have remained text for now.
3. Time

   pilot: 5 min

4. Status

   Incomplete. Waiting on defined usability and style from user story L.

5. What's Left?

   Create icons and replace the linked text.

iii. K3: Create medication form webpage.

1. Unit Tests
   - Pilot ran node app.js to have a live iteration of the web pages while tweaking HTML and CSS features. This made testing easier in determining which organization and styling implementations worked best for the form
2. Problems
   - Pilot was VERY rusty in remembering web design skills. Spent a lot of time researching and testing functionality and style, i.e. reacquainting with web design coding.
   - Pilot used borrowed code to create expandable fieldsets in the medications forms. However, the borrowed code overrode the pilot's styling of the form. The pilot could not figure out how to circumvent the additional styling from the borrowed code, thus removed the attempted feature for now.
   - Pilot could not remember most CSS knowledge, thus spent a lot of time researching and testing while implementing, i.e. reacquainting with style coding.
   - Pilot used borrowed code to implement an edit icon for each text field in the prescription form. The functionality is to lock the fields for the prescription label API to have "first dibs" at entering the data, and prevent the user from accidentally changing their prescription data. However, the pilot could not figure out the error in the javascript function to enable the editing icon, thus removed feature for now.
3. Time

   pilot: 4+ hours creating the webpage, much of which was spent attempting and testing styling features

4. Status

   Incomplete. Waiting on final usability and style from the home page design to style the header and footer of the page. The forms

themselves need more styling work to be presentable as a final product.

5. What's Left?
   - Implement collapsible fieldset feature
   - Adjust styling (significant)
   - Integrate homepage template
   - Implement edit icons and subsequent features

iv. K4: Write code to take user to medication form from homepage.

1. Unit Tests
   - Pilot ran the web app on the OSU flip2 server to render the home page, then clicked on both the Add Prescription and Add Over-the-Counter links in order to render the appropriate forms
2. Problems
   - There were bugs in some of the initial environment setup parameters that were corrected, but which led to faulty connections from the home page to the medication forms. After some tinkering, the bugs were fixed and the links were live again.
3. Time

   pilot: 1 hour total for coding the links, resolving the link problem, and testing

4. Status

   Completed.

5. What's Left?
   - Pairing code with icons

v. K5: Write code for website to connect to database.

1. Unit Tests
   - The code from this task was borrowed by the pilot from a previous project in CS 340 successfully completed by the pilot. The success of the pilot's previous project has lended this team to address other possibilities for the database integration problems.
2. Problems
   - Currently the medication forms are not able to record data to the database. It is currently unclear if this task is responsible.
3. Time

   pilot: 5 min to adjust code for this project's credentials

4. Status

   Under review.

5. What's Left?

   Test the database connection with credentials provided in this task.

vi. K6: Write code to record data to database.

1. Unit Tests
   - Pilot ran the web app on the OSU flip2 server, filled out all fields in the form and clicked on the Submit button.
   - Pilot edited code in small increments, attempting to fix the connection, then re-ran the app and re-tested the field data and Submit button each time.
2. Problems
   - Pilot found that the program does not throw any errors during compilation or as a result of clicking on the Submit button. However, no records have yet been created in the database.
3. Time

   pilot: 4+ hours of writing and testing code

4. Status

   Incomplete.

5. What's Left?
   - Find and correct bugs in the code preventing the creation of new records in the database

vii. K7: Design a window for the confirmation from the user.

1. Unit Tests
   - No testing took place as there was no time left to attempt creating this interface
2. Problems
   - The pilot experienced too many bugs which arose during creating the form interfaces and code that there was no remaining time to attempt creating the confirmation page.
3. Time

   pilot: 0 hours

4. Status

   Incomplete.

5. What's Left?
   - Create webpage
   - Link medication form Submit button to render confirmation page, only upon successful creation of new medication record
   - Write code for confirmation page Continue button to take user back to home page

2.3USER STORY L: USER FRIENDLINESS

a. TEAM

Diego (pilot) and Mohammed (co-pilot)

Riley (pilot) & Johnny (co-pilot) - L3 only

b. TASKS

i. L1: Find optimal color scheme and font size.

1. Unit Tests
   - Reached out to customer for specific color requests for her product.
2. Problems
   - Customer did not have a particular preference, but did have two colors in mind that she thought looked good together. The two colors are not highly contrasting to each other or to the main background color of the web pages, white.
3. Time
   - 2 hours
4. Status
   - Completed
5. What's Left?
   - Integrate the requested colors into the web design, along with a high-contrast color to the main color theme, white.

ii. L2: Create homepage as a template for remaining web pages.

1. Unit Tests
   - Rendered other web pages created for the app to test homepage wrap around. The intent is to ensure continuity in the header of the web app with a title and navigation bar for the user.
2. Problems
   - Technical difficulties arose from inexperience with GitHub. Files could not be downloaded and then modified.
   - Technical difficulties arose from inexperience with Express-Handlebars. The web page was not processing the CSS file.
   - Create the homepage using express and handlebars frameworks, this is beneficial as it will help in rescuing the files as templates for other pages on the application.
3. Time
   - 6 hours
4. Status
   - Incompete. Problems were resolved, but the homepage is not completed.
5. What's Left?
   - Finish styling header
   - Integrate colors specified by customer
   - Implement logo
   - Create icons for footer links
   - Style footer

iii. L3: Integrate homepage design into design of other pages of web site.

1. Unit Tests
   - N/A
2. Problems
   - Technical difficulties led to the usability homepage template being incomplete by it's scheduled date, thus pushing some design tasks to a later date
   - Despite the environment setup for the homepage template to wrap around the other web pages, which server as three bodies of the web app, the header and footer of the main.hbs file are not rendering with the other web pages in the program.
3. Time

   Riley: 30 min adjusting files, folders, and code to get the medications forms to render the homepage template

4. Status

   Work in progress.

5. What's Left?
   - Integrate usability features into all other user interfaces
   - Enable homepage template to wrap around all other web pages created for the app

3. SPIKES & SEQUENCE DIAGRAMS

3.1 USER STORY A, TASKS 1-10

Was the diagram useful? Why or why not?

At this point in time the user story diagrams were not very useful. The team only implemented the survey related database tables and the survey instruction web page so a need to reference the table never really occurred. The sequence diagram does provide a clue to the database design. Since the web page pulls questions from the database, this means we needed a table with pre-written questions. Also, since the web page stores every answer in the database, the answers table will probably be very large and it may be a good idea to make it as lean as possible.

Were there any diagrams that you wish that you had? Why or why not?

No. A diagram would not have been useful. The main problem that the team ran into was about how to best design the survey and answers tables when we are supposed to have only one record in the survey table per survey taken and many answers recorded in the answers table per survey taken. For example, what fields should go in these and what should the primary keys be. The

solution to this required a thought experiment and the drawing of sample tables on paper.

3.2 USER STORY K, TASKS 1-7

Was the diagram useful? Why or why not?

The sequence diagram for user story K was useful in organizing the steps we needed to take to build the user story. We had hoped to accomplish all the events in the diagram, but we got stuck on writing data to the database, thus we made it only halfway through. The diagram should be useful in the same fashion for this upcoming week of work since we are planning to complete the remaining events outlined in the diagram.

One thing to note about the diagram is that, when the pilot got to creating the tables and coding the database connection script, it was found that a couple of the diagrams order of events would not be feasible. Instead of writing the prescription object first, then getting the prescription ID and assigning it to the medication object, we need to write the medication object first, then get the medication ID and assign it to the prescription object.

Were there any diagrams that you wish that you had? Why or why not?

Considering how we organized our tasks for this week, a Model View Controller (MVC) diagram may have been useful. An MVC diagram depicts the data model, presentation information, and control information separated into different objects.

The first day of implementation for this user story consisted of the data model object. We created the necessary database, tables, and attributes required to complete the user case.

The second day of implementation for this user story consisted of the presentation information. We created the web pages for the user interfaces.

Finally, the third day of implementation for this user story consisted of the control information. We created the Javascript code to enable the user interface features, as well as manipulate the data entered by the user between the user interface and the database server.

When creating the task outline for this past week, we simply tried to group similar related actions to make the process more efficient, exactly what an MVC attempts to accomplish. However, we did not organize simultaneous work very well. The MVC design potentially could have helped us organize and decompose our process to be more friendly to simultaneous work between teams, as well as between teammates.

3.3 USER STORY L, TASKS 1-3

Was the spike useful? Why or why not?

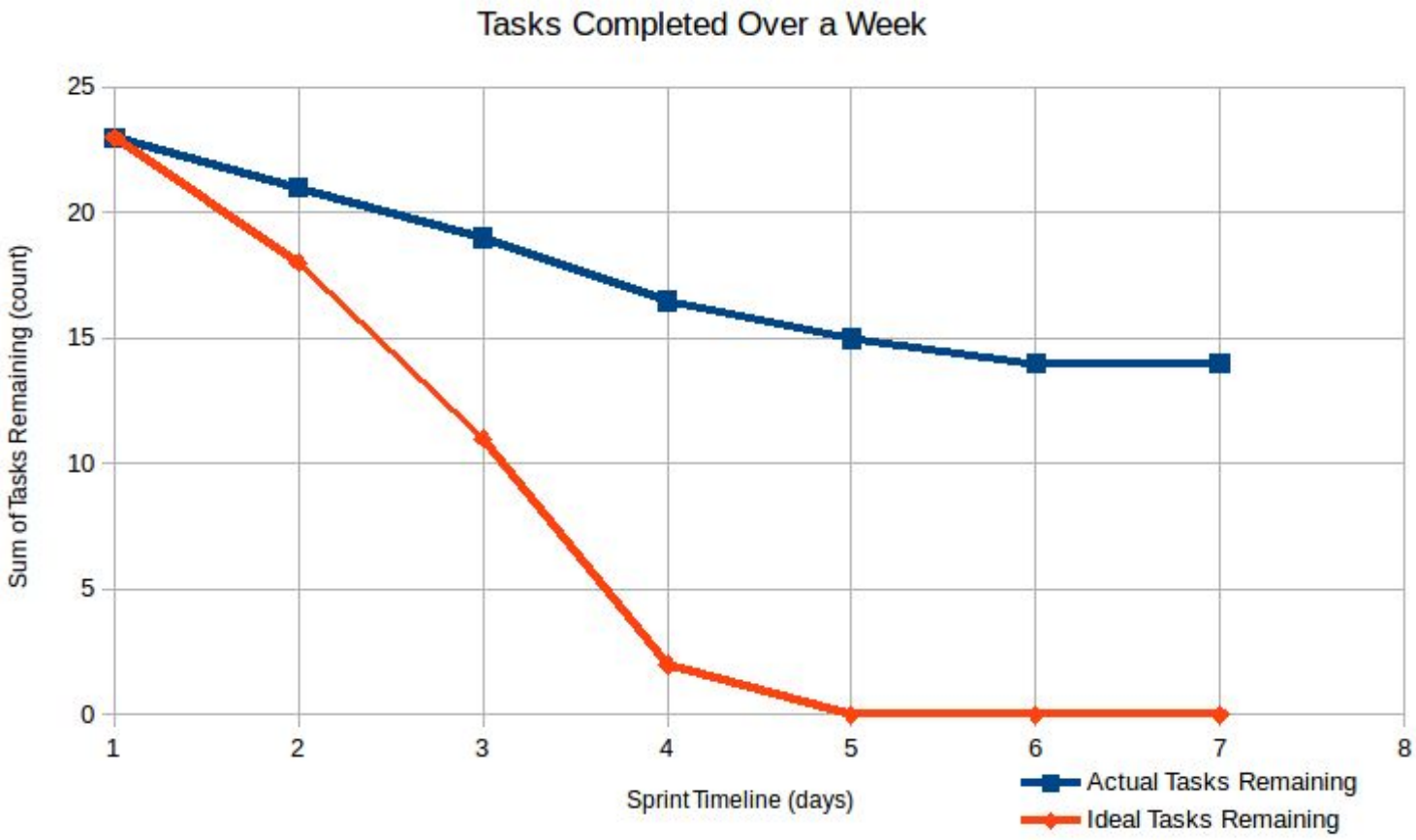
The spike for use story L was essentially a research project in finding out how to design senior friendly websites. Once the data was collected and analyzed it was very useful in helping the teams responsible for user stories A and K design their features. The information provided in the spike gave us direction in what design elements we needed to include in the medications forms, including colors and event functions. Among these were the need for high contrasting colours and the need to be able to resize text when designing websites for senior citizens. These details made it less daunting to come up with design features for user stories A and K that could have ended up different from other features in the web app. Thus, this spike served as an excellent source to ensure a cohesive design.

Were there any diagrams that you wish that you had? Why or why not?

A diagram no, but an expert opinion on how to make our web page look more professional would have been useful.

4. BURNDOWN DIAGRAM

The following diagram depicts the expected burdown of slated tasks for this week based on the task outline we formulated in HW 5. The expected burndown is represented by the red line in the sum of remaining tasks per day, over the course of the 7 days of the week. The blue line depicts the actual burndown of the tasks for this week. Note that the lack of steepness to the blue line reflects the reports we presented regarding our technical difficulties this week. We started several of the user stories but were not able to complete all that were started. As well, due to a significant underestimation of the estimated time required for these tasks there were several user stories slated for the week that were not even able to be attempted. The team has readjusted our estimates for the tasks we've scheduled for the upcoming week, and believe we have created a much more realistic outline that fits to our abilities and time constraints as a team. Hopefully, this will render a more positive outlook on our planning and estimations for next week's burndown diagram compared to this week's diagram.



5. REFACTORING

5.1 USER STORY A, TASKS 1-10

During the implementation of user story A, a couple of the application pieces were reused and refactored to make them achieve new functionality. Moreover, in some cases, multiple iterations of a design were implemented until the desired functionality was achieved. During this week only tasks A1, 4, 6 were implemented due to lack of recent practice and/or more advanced experience.

In A1, the initial work was based on the UML diagram to indicate which tables and columns needed to be implemented. After implementing the necessary parts of the tables, it was found that it wasn't possible to use the UML diagram to design the tables, as a result the team started brainstorming the possible configurations that could be used. The main challenge was making a column in the survey table that associate one survey with a list of questions and answers. After that, by researching how to create such column, quickly the team found that this is not possible. During brainstorming, the team found a way to implement this task to be able to associate the one survey with multiple answers and questions. The approach taken was not the ideal solution to this problem, but it does work. To solve this problem an "Answer" table was designed to have a column for the survey id, question id, and the answer it self. With this approach, a filter could be utilized to filter the rows by the survey id and then the ids of the questions and their answers can be accessed.

Moving to task A4 and A6, there were not many refactoring opportunities due to the simplicity of the tasks. To implement A4, the template of the homepage was used and changed to display the instructions instead of the homepage content. Then, the CSS style file was modified to format the page properly. And for the task A6, the code snippet that renders the homepage was copied and modified to render the instructions page instead of the homepage.

5.2 USER STORY K, TASKS 1-7

During the coding process for user story K, there was much refactoring to clean up the code. Originally, before the environment setup was updated, this user story was designed as HTML pages, with paired Javascript and CSS files in their own folders in the directory. After the setup environment was updated, the format of this user story's .html pages needed to be changed. Rather than structuring the .html files with their own DOCTYPEs, headers, bodies, and footers, we could now remove those superfluous sections as they were accounted for in our main.hbs layout file. This eliminated at least a third of the code in each of this user story's view files. The view files are now .hbs and are organized by div's in what will only be the body of the user interface for this user story.

To enhance the styling of the user interfaces, and to consolidate the code in the files, all input features were paired on the same lines with their label features. Originally, the label features would precede the input features on the line above.

To clean up the code and make is easier to follow, the code follows the standard indentation style of an HTML file. The pilot also added an extra space between each div and fieldset, as well as commented at the beginning of each fieldset which section of the user interface the code belonged to. This should make is easier for the co-pilot and/or other programmers to find the section of code they want to alter faster.

The CSS file for the .hbs files of this user story were originally housed in a "style" file. However, this proved to be problematic for the updated setup environment, which only specifies a "public" folder. Thus, the CSS files were moved to the "public" folder to be consolidated with the rest of the program's public files. The code within the CSS files is organized by type, class, and then id. This allows more specific style requirements to be defined last, thus overriding more generalized styling code. Each section of style code is identifiable by name.

The Javascript files for this user case each contained varying types of code for their respective view files. Thus, the pilot split the types of Javascript code between two files. One file contains the script for event functions executed from the user interface that are specific to elements defined in the user interface view files. The second file contains the script for manipulating data to and from the user interface to and from the database server. The division of the types of Javascript code needed is now more organized and easier for other coders to find exactly which functionalities their looking for, as well as makes it easier to find the source of errors in the code. Both types of files are organized with standard coding practice indentation and use of whitespace, as well as comments and documentation.

The Javascript files specific to event features within the user interfaces are held in the "public" files along with the CSS files to consolidate all code associated with user interface design and functionality.

The Javascript files specific to database features are held in the main directory with the database connection and main app files to consolidate all code files that enable the system's background functionalities.

5.3 USER STORY L, TASKS 1-3

In the implementation of user story L, the webpages were designed to be reusable. This should result in more consistent page design across all the application pages and create a reusable template that other developers can use when implementing their own assigned user stories.

Starting with L1, during the development of this task the team responsible for the implementation iterated through a couple versions with different font sizes, based on the spike research, to determine the proper default font size. In addition, the team also experimented with the page design and colors, both suggested by the client and found through research. As a need for an application easily usable by seniors is one of the top priorities according to the client, the team decided to choose the color pattern found from the research as the colours had more contrast.

During the development of task L2, the developers utilized the express-handlebars framework to create a template to be used across all the pages of the application to deliver a consistent reusable design across the app. In the process of implementing these templates the team started by naming the pages with ".handlebars" extension. However, taking this approach introduced some bugs in rendering the pages. As a result the app.js file and all other page files were edited to reflect the change of the file extension from ".handlebars" to ".hbs".

Moving to L3, due to the delayed and the incompleteness of the dependencies of this task, specifically the homepage, the developers of this task had to modify the homepage template and it's styling file to be able to render the page properly.

6. CUSTOMER QUESTIONS

The customer did not indicate the need for any requirement changes, thus there were no surprises in response to the work we completed this week.

Do you have a name for your system?

The customer responded that she did not have a name in mind for the product. Therefor, a team member came up with a prototype name, "My Complete Healthcare". For now, this is the name we will proceed with unless the customer requests another name.

Are there any colors in particular, i.e. a signature look, you want for the user interfaces?

The customer indicated that she did not have any visual preferences for the system. However, she did offer us two colors, #FDBB4D (similar to mustard yellow) and #56CAC1 (similar to turquoise blue/green), that she mentioned she thought looked good together.

The team is readjusting our estimates and priorities in order to provide at least one working user story, with applied usability features, by the end of next week. One of our teams is considering the merits of continuing with building the Survey Questions feature vs. creating the Medications repository to complement the already created Medications forms. Do you have a preference on what you would like to see created first? The customer did not respond within 24 hours, thus the programming pair responsible for user story A or user story P for the upcoming week has decided to continue working on user story A.

The group provided our document to the customer with reports on the tasks we accomplished this week, problems we're having, the task outline for next week, and a live rendering of the current version of the web application.

The customer had no feedback to offer and responded that everything was looking great thus far.

7. INTEGRATION & USABILITY TESTS

The following table is an outline of the integration and usability tests performed on the current version of our web application. The integration tests detail the specifics of how well the different components are working together, and how to successfully run and navigate the system. The chart also includes the positive and/or negative findings of performing the tests, and any changes that need to be made to the system in order to run the same test flawlessly again in the future. The usability tests detail the functionalities and feel of the features that were implemented this past week, any positive and/or negative findings from the tests, and any changes recommended by the tester to improve the user stories and system as a whole.

TEST	RESULTS	CHANGES
Test installing the dependencies: To perform this test, start with a new workplace by cloning the repository locally, then run the command `npm install` after running this command the script will run and download the required packages to run the app.	To verify the passing condition of the test, there should be a new directory called packages and when executing the command `node app.js` the app starts without any errors about missing packages.	Improve the script to be able to run the app by executing `npm start` and removing the temporary files using the command `npm clean`
Running the app: In the home directory of the project run the command `node app.js`, then open the page http://localhost:3000	Verify that there are no errors after running the command. After that verify that the link provided in the test is functional and leads to the homepage of the app.	Change the running command to `npm start`, this would be helpful if the app back end is composed by multiple files.
Verify the Home Page: In the home directory of the project run the command `node app.js`, then open the page http://localhost:3000	Verify that this list of links/information are present on the homepage: ● User name. ● The date ● Add prescription link. ● Add over the counter medication link ● Take survey link ● view or create calendar reminder link ● View my medication link ● view my alerts link ● view/create report link. Verify that the text and colors of the page are easy to read.	Improve the design of the webpages to be more modern and navigable.
Verify the Add Prescription Page: In the home directory of the project run the command `node app.js`, then open the page http://localhost:3000 then click on “Add Prescription” link	Verify that the link from the home page is functional and leads to “add prescription page”.	Implement memorable icon in addition to linked text.
Verify the Survey Page: In the home directory of the project run the command `node app.js`, then open the page	Verify that the link from the home page is functional and leads to the survey instruction page.	The survey webpage needs to be implemented. Implement memorable icon in addition

http://localhost:3000 then click on "Take Survey" link		to linked text. Add a Continue button to the instructions page to continue to the actual survey.
Verify the Add Over-the-counter Page: In the home directory of the project run the command `node app.js`, then open the page http://localhost:3000 then click on "add over-the-counter" link	Verify that the link from the home page is functional and leads to "add over the counter" page.	Implement memorable icon in addition to linked text.
Verify the Calendar Page: In the home directory of the project run the command `node app.js`, then open the page http://localhost:3000 then click on "View Or Create Calendar Reminder" link	Verify that the link from the home page is functional and leads to "View Or Create Calendar Reminder" page.	Currently this page is not implemented. Implement memorable icon in addition to linked text.
Use of Name, Instructions, and Rx Number text fields: From the Prescription or Over-the-Counter form page, click on field, type in a variety of different text, delete text, tab out of field, click out of field.	The fields take all text options possible. Able the tab in and out of the fields. Fields accept what appears to be unlimited characters.	Recommend limiting number of characters for these fields to comply with the database memory standards for text fields.
Use of Dosage, Duration limit, Frequency day, Refills doses quantity, and Refills refill reminder count fields: From the Prescription or Over-the-Counter form page, tab to click on field, enter number by keyboard, use up and down arrows, try to enter alphabet characters.	Can enter numbers by keyboard. Can use up and down arrows to increase or decrease current number in field. Default value is 0. Cannot enter alphabet characters.	Performs as expected.
Use of Dosage, Duration, and Refills doses unit fields: From the Prescription or Over-the-Counter form page, click on the down arrow and select a unit, move up and down the list with the keyboard arrows, type a unit into the field.	Can tab to field and use up and down keys to scroll through the dropdown lists. Can click on down arrow, expand list, and select item. Cannot enter freeform text.	Put dropdown menu items in alphabetical order. Allow for free-form text, up to a limited size.
Use of Instructions, and Frequency custom days checkboxes: From the Prescription or Over-the-Counter form page, click on all boxes, leave all boxes unchecked, check only some boxes.	Can check and and all checkboxes at the same time. Can uncheck any and all checkboxes at the same time.	Performs as expected.
Use of Schedule Start Date, Expiration Date, Rx Date, and Rx Expiration Date fields: From the Prescription or Over-the-Counter form page, enter a date by the keyboard, use up and down arrows, use calendar GUI, click on X.	All fields perform the same functions in the same fashion. Tab automatically moves to next section of the date when finished typing in month, then day, then year. Up and down arrows only work on individual fields of the date, not the entire date. Calendar GUI is very user friendly. X icon clears the date field.	Provides many options and is standard date format. No need to change.

Use of Duration and Frequency radio buttons: From the Prescription or Over-the-Counter form page, try to click on all radio buttons, tab between fields and hit Enter.	Can only click on one radio button (in each respective section of the form). Can tab to radio buttons. Cannot select radio button by using Enter when highlighted on field.	Performs as expected.
Use of form Submit button: From the Prescription or Over-the-Counter form page, click on the button to end the entry of the form.	Refreshed current web page and clears all data. No field requirements prevented from re-rendering the page.	Fix link for Submit button to go to confirmation page. Put restrictive requirements on significant fields so user can't record a new Medication without the pertinent data.
Navigation of the Medications form: From the Prescription or Over-the-Counter form page, navigate through the sections of the form from top to bottom and left to right.	Though unattractive, the fieldset borders lend some organization to an otherwise mostly unorganized form. Fields stacked on top of each other are too close, radio buttons and checkboxes are arranged haphazardly, it's difficult to discern which labels go with which fields. The Submit button is nice and large but the color clashes with the theme of the website.	Implement better style to the entire form, particularly the spacing and organization of labels and fields. Change color of submit button.

8. PLAN OUTLINE

Due to the lack of web design practice for most of the group since taking CS 290 and/or CS 340, we ran into many bugs during this first week while getting reacquainted with the necessary coding knowledge and skills. We did not consider the issue of dormant knowledge, or how much time it would cost the team when we made our original task lists and time estimates. Therefor, we've re-evaluated the time it will take us to complete the tasks we outlined for this past week, and added several subtasks to ensure we don't miss steps we had not considered before.

The new estimated times mostly reflect the time it took our group to accomplish this past week's tasks. Though we have considered that we are all now a little more comfortable working with this type of code again, since this past week, and expect to work a little bit faster.

Our aim is to complete the main styling of the web app to give the customer a good idea of what the final product will look like for their users. As well, we are also aiming to complete at least one additional user story, and in this case the majority of one of the 3 user cases the group worked on this entire term. Since the rest of the system is dependent on the user creating records of medications, it is our goal for this upcoming week to fix the database connection code and upgrade the style for at least one of the medication forms, in order to demonstrate to the customer the feel and function of one of the most important features of the system.

Having these user stories to present to the customer next week should give us a good opportunity to address any major changes the customer wants to implement before proceeding with other user stories for the software system.

WHEN	WHAT	WHO
Monday	L2 - style homepage: title, logo, nav bar, coloring, etc. (~3.5 hours)	Johnny & Riley
	K2 - make icons to replace linked text in homepage body: make images folder, create/find image, link image, style image (~1.5 hours) A3 - create Take Survey icon on homepage (~45 min) L2 - make footer icons for homepage template: make images folder, create/find images, link images, style images (~1.5 hours)	Mohammed & Diego

Tuesday	K5 - test database connection: design, write, and perform tests (~1 hour) K6 - fix bugs in GET code to successfully write a medication to the database: implement in over-the-counter form first, then integrate into prescription form (~3 hours)	Johnny & Riley
	A2 - write default questions and add as records to Question table (~1 hour) A5 - create webpage to answer survey questions (~3 hours) L3 - integrate homepage template (~1 hour)	Mohammed & Diego
Wednesday	K3 - adjust form pages styling (~3 hours) L3 - integrate homepage template (~1 hour)	Johnny & Riley
	A6 - write code for icon to take user to survey page (~30 min) A7 - write code to retrieve and display questions from database (~4 hours)	Mohammed & Diego
Thursday	L - final tests	Mohammed & Diego Johnny & Riley
	K - final tests	Johnny & Riley
	A - final tests	Mohammed & Diego
	Present stories to customer for evaluation and revisions	Team and Customer

9. REFERENCES

[1] Campbell. Feb 2015. *Designing For The Elderly: Ways Older People Use Digital Technology Differently*. [Online] https://www.smashingmagazine.com/2015/02/designing-digital-technology-for-the-elderly/

[2] Dailey. *How Senior Friendly is your website*. [Online] https://www

.illuminage.com/files/2014/05/webinar-050913.pdf

[3] GeeksforGeeks: A computer science portal for geeks. *MVC Design Pattern*. [Online] https://www.geeksforgeeks.org/mvc-design-pattern/

[4] How to store a list in a column of a database table. Stack Overflow. [Online]. https://stackoverflow.com/questions/3070384/how-to-store-a-list-in-a-column-of-a-database-table

[5] Moth. May 2013. *Six design tips for making your website senior friendly*. [Online] https://econsultancy.com/blog/62815-six-design-tips-for-making-your-website-senior-friendly

[6] Mountain Goat Software. *User Stories.* [Online] https://www.mountaingoatsoftware.com/agile/user-stories

[7] NaturalReader. [Online] https://www.naturalreaders.com/index.html