Homeless Helping Hand Group – David Sobota, William McCumstie, Ian O'Leary, Huihao Huang, Markus Woltjer

CS 361

Project B Report- Homeless Helping Hands

Week 1 User Stories

- **Select Main Interface:** User indicates whether they are seeking help, or a provider looking to give help

USER (aka Homeless)

- **View Help Interface:** User can select different icons with a physical press
- **View nearby service:** A list of services will be shown, sorted by distance and available beds
- **Review Bed Locations:** User can choose a shelter location and view available beds
- **Call Shelter:** User can hit a phone icon and use the kiosk to call the shelter, and check up on bed status, etc. if it hasn't been updated in awhile
- **View Calendar:** Check location hours, see if there are any special dates/events planned, or closures for holidays, etc
- **Review Location:** User can leave a review of a particular shelter if they want, ranking it on cleanliness, response time, etc. to help other users make decisions.
- **Check Reliability:** User can see how often on average a location updates their information, to help users know how reliable the current information posted might be
- **Give Directions:** When user selects a location address , a map will be displayed
- **Give Directions (alternate):** When user selects a location address, a small piece of paper will print with step by step directions
- **Select shower:** Kiosk will show the nearest shower truck, and the closest stop to the user, along with estimated times of arrival for each stop on the route.
- **View route:** Kiosk will show the user where the truck is through GPS, so the user sees in real time how far away the truck is.

PROVIDER (Support Services e.g. Shower Truck)

- **Log in:** When selecting the give help option, user can register their location, or log in and update it
- **Provide shelter info:** User will fill out a form asking for various info, such as name, address, etc.
- **Update bed info:** User selects the update bed button, and inserts the current number of beds available. This is time-stamped, so homeless people know when it was updated.
- **Update Calendar:** User can update specific dates on a calendar, marking special events, or days that the shelter might close due to holidays, etc.
- **Input special notification:** User can add comments or notes, such as the soup kitchen ran out of food for the day, and that message will be displayed predominantly when a homeless person selects that location
- **Input custom directions:** If the clinic is in a hard to find location, the user can put in custom instructions, referencing landmarks, etc.
- **Provide additional resources:** Here, the user can input their phone number, hotlines to suicide services, etc., anything that they feel might be beneficial

User Story Task List

Note: Time Units are Hours spent coding in pairs

- **Select Main Interface:** Time Units: 3
  - Implement Welcome Web Page -1
  - Implement Home Page for Homeless Users -1
  - Implement Home Page for Providers - 1

USER (aka Homeless)

- **View Help Interface:** Time Units: 2
  - Implement Home Web Page displaying buttons and Icons - 1
  - Make the buttons take the user to different web pages - 1
- **View nearby service:** Time Units: 10+
  - Create Web Page Listing nearby services -3
  - Order the locations based on what's closest -3
  - Pull information from nearby services from database – 4+
- **Review Bed Locations:** Time Units: 6
  - Create webpage that shows availability of beds - 2
  - Connects page to BD with live bed availability - 4
- **Call Shelter:** Time Units: 8
  - Create button on the webpage that redirects to some free phone service - 3
  - Pull phone information from the provider database - 5
- **View Calendar:** Time Units: 5
  - Create webpage that displays a calendar - 2
  - Have indicators on the days that are specially marked (holidays, events, etc.) -1
  - When users click on a day, have a pop up that lists all relevant events/closures in depth -2
- **Review Location:** Time Units: 5
  - Create a 5 star rating that show's on a provider's page -3
  - Allow the user to write a review with a virtual keyboard - 1
  - Create a page where users can go through and browse all existing reviews -1
- **Check Reliability:** Time Units: 10+
  - Only count the time that the facility is open - 5
  - Pull average by calculating total number of updates given since signing up, over how many minutes the location has been open since signing up - 5
- **Give Directions:** Time Units: 3
  - Using address given, use google maps API to show location on a map - 1
  - Determine user's current location - 1
  - Get directions from current location to the address -1
- **Give Directions (alternate):** Time Units: 5
  - Determine user's current location -1
  - Get directions from current location to the address - 2
  - Print those directions -2
- **Select shower:** Time Units: 10+
  - Display schedule for shower truck -2
  - Select nearest location to user, and display when the truck will get there next -8+
- **View route:** Time Units: 10+
  - Pull GPS information from truck driver's phone (need to be logged in to a mobile app? Not sure how this would work) - 6
  - Display that location on a map on the kiosk, and display user location as well -4+

PROVIDER (Support Services e.g. Shower Truck)

- **Log in:** Time Units: 6
  - Integrate a log in/sign up service - 4
  - Implement home page for providers -2
- **Provide shelter info:** Time Units: 8
  - Create web form to insert data -3
  - Create database to store information -3
  - Allow user to view database (only their location should display) -1
  - Allow user to edit/delete specific elements of the database (for only their location) -1
- **Update bed info:** Time Units: 8

- Create a webpage for bed info -3
  - Allow user to either just input a value, or maybe have up and down arrows to modify availability -5
- **Update Calendar:** Time Units 8
  - Create webpage of Calendar -3
  - Allow user to click/select any day on the calendar -2
  - Allow user to add multiple entries for that day on the calendar -2
  - Store the info and give a visual indicator on the map for that day -1
- **Input special notification:** Time Units: 10+
  - Create Webpage for notification section -2
  - Create form for the user to enter in the alert -2
  - Put the alert in the database(?) so the homeless user can see it when they select the provider -4
  - Make it show up predominantely on the page, or have a pop-up alert on the kiosk -2+
- **Input custom directions:** Time Units: 8
  - Create webpage for custom directions -4
  - Create form for the user to input in the directions -4
- **Provide additional resources:** Time Units: 8
  - Create webpage for additional info - 2
  - Create multiple forms, for the user to put in what they want - 4
  - Create a “get additional resources” button on the homeless client, where they can view all of the info in a list -2

Week 1 User Story Priority List

For the first week, we didn’t hear back from the customer on how to prioritize the user stories. So, as a group we determined that we could do 12 units of work this week, and prioritized for the first iteration only.

Main Priorities:

- **Select Main Interface:** User indicates whether they are seeking help, or a provider looking to give help
    - Implement Welcome Web Page
    - Implement Home Page for Homeless Users
    - Implement Home Page for Providers
    - Time Units: 3

- **Update bed info:** User selects the update bed button, and inserts the current number of beds available. This is time-stamped, so homeless people know when it was updated.
    - Create a webpage for bed info
    - Allow user to either just input a value, or maybe have up and down arrows to modify availability
    - Time Units: 8

Secondary Priorities:

- **Review Bed Locations:** User can choose a shelter location and view available beds
    - Create webpage that shows availability of beds – 2
    - Connects page to BD with live bed availability – 4
    - Time Units: 6

- **View Help Interface:** User can select different icons with a physical press
    - Implement Home Web Page displaying buttons and Icons
    - Make the buttons take the user to different web pages
    - Time Units: 2

Summary of First Iteration Process

The first iteration was a bit rough for our group, but we ended up making some good progress in the end. In terms of pairing up, we divided the week's work into two distinct categories: client side work (which dealt with the web pages for the user, provider, homepage, etc.) and server side work (getting a database set up, connecting to the database, sending information back and forth). Ian, David and Huihao teamed up on the client side work initially, and William and Markus teamed up on database/server stuff. Our group used Google Hangouts to have full-team meetings or hold discussions within the sub-groups. Our group used Slack for general communication throughout. William set up a GitHub repository to host all files for the project, and agreed to push changes to his master branch. In order to adhere to the pair programming method, we used Google Hangouts, using the built in screen sharing option in order for the other person to see what the pilot was coding. People would take turns, and the person who was watching would now code for a while, and vice versa. The role of the copilot saves group time in catching simple errors as they occur, suggesting alternative methods, and bringing a second understanding of the code back to the group. Our group compositions were fluid, and pairs shuffled around during the week, depending on the strengths and weaknesses of team members. For example, later in the week William and Ian teamed up to get the database connected and speaking to the server which the website is being hosted on, while the rest worked on finishing up the client side pages and code. Our biggest problems in our group happened during the coding/co-pilot process. For starters, people were hosting code in their own repositories, or repositories not under William's master branch, leading to scattered code, multiple versions, etc. Once this problem was recognized, we re-emphasized only pushing changes to William's master branch. Next, people were coding without a copilot to aid, leading to instances of code that was confusing to others. We clamped down on this too, and reinforced that no one can tackle code without a co-pilot. Finally, communication was an issue as well, with people not getting in touch for multiple days, not being able to work on the project for extended periods of time, etc., leading some to believe that they were pulling an unfair portion of the work. However, as time went on, and our group got into a groove, these problems gradually improved, and we started working more efficiently as one unit.

Summary of Unit Tests

We had unit tests comprising three different categories: server-side unit tests, server-client integration unit tests, and webpage unit tests.

Server Side: For server side stuff, we did unit tests regarding to the setup of the server, installing node modules, and setting up express. We encountered an error by accidentally trying to host the site on the wrong port, but we fixed that. All modules installed correctly, and a blank website hosted on the port worked. We created and tested a MYSQL database, and there were no errors. All fields showed up correctly. When creating another table, for the shelters, there were no errors and all fields worked there as well. Finally, we created dummy pages for the user site and the provider site, and the MYSQL databases correctly connected to the server. Overall, there were almost no errors here through unit testing, which was great.

Client Side: For the hosting of the client side webpages, unit tests were conducted to setup bootstrap correctly, design the webpages, and correctly display the webpages on the server, with proper formatting and style. Many bugs showed up at this point during unit testing, mainly through setting up bootstrap. When hosting bootstrap on the OSU servers, all the formatting worked great. However, when we tried incorporating it onto our own server, and set up everything through handlebars, the formatting and style repeatedly broke, leading to many hours trying to research and fix this problem. Finally, we were able to put the necessary files in the public folder on our server, and the formatting worked out correctly.

Client-Server Interaction: The unit tests here were similar to the client side, making sure everything was running on the server correctly, although this time we checked to make sure that our post and get requests were working to add or pull data from the database, and populating the webpage correctly. We had to make changes in order to allow the provider to select between updating an entry and creating a new one. Our method for this was creating a separate JavaScript file. There were also issues submitting the provider form to the server, pertaining to making JSON requests, but we fixed that as well. After all the unit tests, we had all sections working, and the client could successfully communicate and exchange information with the server and database.

Acceptance Test Summary

For this project, our group did internal acceptance testing. We did a good job going through and doing extensive unit testing, so thankfully we did not run into any problems. For the acceptance testing, we viewed our project through the eyes of a customer. So, primarily, we wanted to make sure the website loads, and the page is easy to understand and minimalist in its design. Next, we made sure that clicking either button would take you to the right webpage, and display the necessary info. For the provider site, we wanted to make it easy to select to either insert a new provider, or update an existing one. This was easy to differentiate, because you have to select either option before a form shows up. The customer will not be that knowledgeable/interested in the back end and database stuff, so we focused our tests on the client side. If you submit a form for a new provider, does it show up on the user page? As long as all the information passed in to the form also shows up on the user page, then that is a success. Next, we wanted to make sure that there are not any providers missing from the table shown on the user page. We double checked this against the database, and kept track of all the information we were inputting into the forms, and everything checked out well. Next, we checked to make sure that if we update the number of beds on the provider site, the user page table will only show the most recent value, and we passed this test. Finally, we wanted to see if you try to put in a provider that already exists, it should not overwrite a provider that already exists in the table, and this passed. The only way you can update an entry is by selecting the update option and filling out the appropriate form. Therefore, since we made numerous changes already during unit testing, acceptance testing went off without a hitch. Still, it was good to have a secondary check and make sure everything was working well together on a higher level as well.

Summary of Week 2 User Story Changes

We were able to hit all of our goals for the first iteration, but it definitely took us longer than we expected it to. We originally aimed to complete 12 time units for the first week, but due to trouble shooting, figuring out issues, and implementing all features, we ended up putting in 19 time units into the first week, and ate up a lot more of our free time than we wanted. As such, we went back through all our user stories, and revised most of them, increasing estimated time for most cases. Since we were able to complete 4 user stories in the first week, we were able to remove them, as well as tweaked existing stories, and adding a new one, relating to the completion of client/database interactions. After revising and tweaking expectations, we again contacted the customer, with another goal of around 12 time units to be completed for the second iteration process. Below are the revised user stories that the customer prioritized for the second iteration:

Main Priority:

- **Finish up Client/Database Interactions(New Story)**
  - Get user page to pull data from the database
  - Get the provider web page to post data to the database
  - Time Units: 4
- **Give Directions:** When user selects a location address , a map will be displayed
  - Using address given, use google maps API to show location on a map
  - Determine user's current location
  - Get directions from current location to the address
  - Time Units: 7

Secondary Priority:

- **Provide additional resources:** Here, the user can input their phone number, hotlines to suicide services, etc., anything that they feel might be beneficial
  - Create webpage for additional info
  - Create a "get additional resources" button on the homeless client, where they can view all of the info in a list
  - Time Units: 5
- **Log in:** When selecting the give help option, user can register their location, or log in and update it
  - Integrate a log in/sign up service
  - Implement home page for providers
  - Time Units: 6

Summary of Iteration 2:

After the growing pains of week 1, the second iteration moved along much smoother. For the second iteration, the group split into two groups. William and Nick were responsible for updating the database information to include the relevant address fields for giving user directions, as well as implementing the additional resources webpage, where shelters have an outlet to provide additional information to users as they see fit. Meanwhile, David, Ian and Markus worked on researching and implementing the Google Maps API, in order to generate a map and directions to a given provider on the user page. Since the second group was a group of three, David started writing the report, as well as co-piloting with the other two whenever necessary. Since we already created the webpages for the site, the framework was already there, making it easier to add and build upon it rather than starting on something from scratch. We knew what to expect, and managed our expectations a bit more this time around.

However, as these things always go, coding the google maps api ended up taking a lot more time than we anticipated. It took a lot of effort to create the additional fields in the database, pass them back in correctly, add buttons to dynamically pass in the address and get directions, etc. William and Nick helped too, and we were finally able to get the google maps up and running, and everything else working smoothly. However, this meant that we ended up with less features than we originally hoped for this week, but it's better to flesh out one task fully, instead of being left with two half baked features.

In terms of teamwork, everyone was working much more cohesively the second time around. Again, we organized copiloting on Google Hangouts, the pilot sharing screens with the copilot. Our group members all did good portions of the work, communicating with each other and helping each other out with the code at all times. Communication was good; we had frequent meetings, and helped each other out on different parts of the project in times of need, so everyone worked well with each other. Being thrown into a new development process with a brand new team proved challenging at the beginning, but after a week to work out the kinks, we were able to get into a groove and work productively together. Since we didn't get too deep in over our heads this time, we were able to portion out the work appropriately, and it led to a less stressful iteration.

<u>**Reflection:**</u>

Working together as a project team with a brand new development process was a great learning experience for all of us involved. One of the major benefits of working with an XP process was the direct interaction with the customer. It really feels like the customer is involved every step of the way, and you can work directly with them in order to realize their own goals and visions. The XP process is well suited for time crunches, which was definitely the case for this project. Not only were we restricted to two weeks, but also each of us had large loads of professional and academic work outside of this project, limiting our weekly available hours in quantity and range. The combination of asynchronous and synchronous communication becomes critical here, since the video calling was necessary to sort out complex issues that required input from multiple group members, and the consensus of these meetings was relevant to the work of each individual (in which group video calling is preferable to text chat). The asynchronous group and individual chats on Slack were good for leaving reminders and notes for individuals and organizing the times for video calling. By creating user stories, we were able to plan all the smaller sections, and work on how to tackle them with input from the customer. However, at the beginning, the learning process can be rather slow. When going into a brand new project fresh, you have to make time estimates for all the different sections, and it might end up being way off (especially because you haven't dabbled with them yet), causing you to work overtime to get those features polished up and finished. One strength of using the Waterfall process is that you're constantly building on what you made before, step by step. You are able to polish something up before you move on, and the linearity allows everyone to be fully on-board and contribute ideas and suggestions at each step. However, the downside is that the linearity of the process significantly increases the minimum completion time, since smaller parts are not parallelized. XP can risk the group waiting on a subgroup or individual for the completion of an iteration, but Waterfall ensures that the group will always be bottlenecked, just at a larger bottleneck, since more people and the most experience in that specific task are always working on it. With the XP method, you're able to work on multiple sections at the same time, and get a working prototype up and running much sooner. XP is also advantageous when the group is not working on the same schedule, as was the case. Waterfall seems designed more for workplaces where all workers are available at the same hours at the same time, and can communicate easily as an entire group. A little hybridization of methods is best, such as struggling members or subgroups notifying the others to bring additional help on a tough task. Overall, we preferred the XP method to the Waterfall method, due to the amount of interaction with the customer. We were able to show her our webpages after only one week of work, and were able to get feedback and change things on the fly. If we were stuck with the Waterfall method, we would have been much further along the process before the customer even got to see what we were working on. At that point, trying to go back and make fundamental changes, or focus on entirely new and different features, would be too much of a hassle. Using the XP method, we felt sufficiently challenged, hitting small goals along the way, working as one cohesive unit to tackle multiple features at once. Overall, both Waterfall and XP are good development processes, and choosing one over the other really depends on the nature of the project you have.