Project B Report

Small Repair Volunteer Coordination

# 

# Instructions for Project

### Project URL: <http://52.34.186.228:3000/>

# Report Index

|  |  |
| --- | --- |
| All user stories collected | *p. 1* |
| Estimate of tasks and effort required for user stories | *p. 2* |
| Priority list- priority assigned by the customer to each user story | *p. 3* |
| Summary of pair work- organization, cowork, problems and solutions (including screen sharing, etc.) | *p. 4* |
| Summary of unit tests and description of what kinds of bugs, if any, those unit tests enabled us to find | *p. 5-6* |
| Summary of acceptance testing and description of what kinds of bugs, if any, acceptance testing enabled us to find | *p. 7-8* |
| 2nd XP cycle: Summary of changes to the list of user stories, effort estimates, and priority list | *p. 9* |
| Summary of how division into pairs for the second XP cycle and comparison to first cycle | *p. 10* |
| Reflection: Summary of ways XP was preferable to waterfall, ways waterfall was preferable to XP, and strengths and weaknesses of each process | *p. 11* |

# All user stories collected

1. **Create Account**

A user provides their personal information and sets up an account with a password.

1. **Request repair**

A requester (user) can submit a request for a home repair to be fulfilled by a volunteer.

1. **Accept request**

A volunteer (user) can accept a repair request, which connects the volunteer with the requester for purposes of scheduling, etc.

1. **View requests**

A volunteer (user) can view a list of open repair requests, sorted/filtered by location or repair type.

1. **Submit feedback/rating of volunteer**

A requester (user) can fill out a feedback/rating form regarding a volunteer who serviced their request, which may or may not be anonymous, and which may or may not be shared with the volunteer.

1. **Submit feedback/rating of requester**

A volunteer (user) can fill out a feedback/rating form regarding a requester whose request they serviced, which may or may not be anonymous, and which may or may not be shared with the volunteer.

1. **Verify eligibility of requester**

A requester (user) can submit an eligibility verification form which is reviewed by Helping Hands staff to determine whether the requester meets the eligibility criteria.

# Estimate of tasks and effort required for user stories

*Unit estimates are notated in (parenthesis)*

1. **Account Creation**

A user provides their personal information and sets up an account with a password.

* 1. create requester/volunteer user account general area, general area, email, phone (2)
  2. profile page (2)

1. **Request repair**

requester (user) can submit a request for a home repair to be fulfilled by a volunteer.

* 1. name, give brief description, type of repair (dropdown), give schedule constraints/ general availability (maybe add calendar later), status of repair (3)
  2. update status of request (2)
  3. put on list/database (3)

1. **Accept request**

A volunteer (user) can accept a repair request, which connects the volunteer with the requester for purposes of scheduling, etc.

* 1. view list (3)
  2. requesting user is notified, conversation is initiated (through app) (30)

1. **View requests**

A volunteer (user) can view a list of open repair requests, sorted/filtered by location or repair type.

* 1. sees list or map of requests (3)
  2. based on capability/availability can accept request (2)
  3. starts conversation with requester to schedule repair (30)

1. **Submit feedback/rating of volunteer**

A requester (user) can fill out a feedback/rating form regarding a volunteer who serviced their request, which may or may not be anonymous, and which may or may not be shared with the volunteer.

* 1. requester gives feedback (description, rating 1-5 star) about volunteer, important for safety (were you comfortable, etc?) (2)
  2. display volunteer profile with feedback (2)

1. **Submit feedback/rating of requester**

A volunteer (user) can fill out a feedback/rating form regarding a requester whose request they serviced, which may or may not be anonymous, and which may or may not be shared with the volunteer.

* 1. volunteer give feedback (description, rating 1-5 star) (2)
  2. opportunity to alert staff about requesters who don’t meet eligibly requirements, comment on suspicious or dangerous activities, send message to staff if under 3 stars (2)
  3. display requester profile with feedback (2)

*NOTE: After initial meeting with client, client determined that the seventh user story (Verify eligibility of requester) was not something they wanted to pursue as they thought it might make the services inaccessible or difficult to access for some people who needed them.*

# Priority list

Priority assigned by the customer to each user story

**1st Priority: Account Creation**

create requester/volunteer user account general area, general area, email, phone (2)

**2nd Priority: Request repair**

name, give brief description, type of repair (dropdown), give schedule constraints/ general availability (maybe add calendar later), status of repair (3)

**3rd Priority: Accept request**

view list (3)

*NOTE: Our estimates were intentionally cautious, and we finished some other things during the fulfillment of these goals as noted in the summary of changes for the second XP cycle.*

# Summary of pair work

Organization, cowork, problems and solutions

## Organization & Cowork

In the first cycle we discussed our individual strengths and weaknesses and compared them to the tasks that were identified by the client as priorities.

We found that William felt very comfortable with web development having previously taken the course and also done an internship in that field so he became the driver for his pair. Zachary had not yet taken that course, but was keen to pick up a new skill so he made a good choice as copilot for the pair which was easily matched to the priorities which had been identified that dealt with web development such as assembling the architecture of the pages to login, make and view the requests, as well as the integration of the database to the front end functionality of those pages.

Niza and David both felt moderately comfortable with databases and the pair was chosen to create the database and write the adds and queries to fulfil the client’s priority of adding and viewing requests. David took primary responsibility as the driver with Niza copiloting.

John took of the primary responsibility as the driver for testing which worked well as his schedule was free towards the end of the week when testing could be completed. William volunteered to copilot with him as he would need to be walked through the environment set up in order to properly complete testing. At first William needed to pilot through the initial setup, and then John took the lead as pilot for testing.

The working method in general was to plan meetings using Doodle Polls or Gchat and then to meet on Google Chat or Skype which both have easy methods for screensharing.

## Problems & Solutions

One of the greatest problems for our group was scheduling. With one teammate abroad and another who was traveling during most of the week, setting up easy meetings was difficult. A great tool which we utilized to alleviate this problem was Doodle Polls which switch time zones according to the user and allow for easy scheduling.

Another problem was in the lack of attention we primarily gave to environment setup. Just getting a working environment greatly increased the work units put in by William and Zachary and threw off the group’s unit estimates. The team eventually persevered by following the agile method, telling the client about our miscalculations and putting in the extra work in a reasonable way.

The last problem which tied into this was putting the testing behind, luckily John was able to work around his schedule meet with William to complete things in a reasonable timeframe.

# Summary of unit tests and description of what kinds of bugs, if any, those unit tests enabled us to find

## Unit Tests (please access with OSU email account):

<https://docs.google.com/a/oregonstate.edu/spreadsheets/d/1z0yay0yO2-Og386QHtCr-j8UslGNAzF5uhvM92IJ7V8/edit?usp=sharing>

## Summary

The purpose of the Unit Testing for Cycle 1 was to uncover any functional or technical errors that were produced. From experience, when multiple teams work on different areas of a program there are many integration errors that may occur. The goal of this testing was to perform technical unit testing on each input from the user, to discover any functional errors that occur in the basic functionality of the application, to ensure that proper error messages are sent to the user when performing basic tasks, and to ensure that integration between the database and front end are adequate with no error. From the functional and technical unit testing, we were able to uncover a number of errors occurring from input validation, error handling, and object interaction.

For development cycle 1, 15 unit tests were created that covered a range of areas. 3 unit tests involved logging into accounts and the validation involved in it. 6 tests that involved the creation of accounts and input validation for the accounts. 2 tests involved ensuring that the buttons to load different areas were working properly, and the last 4 tests involved creating jobs from the “Job Creation” screen.

The unit tests performed for cycle 1 are in a separate document called “Unit Testing - XP Cycle 1” with all of the results and screenshots. For log-in validation a number of things were verified. The log-in functionality was confirmed to work and the user was redirected to the proper page. The other two tests that were performed on the log-in functionality were to verify input was properly validated. One test attempted to log into an account that didn’t exist and the other test attempted to log into an account with an incorrect password. The expected result of these tests were to return an error message that indicated the input problem to the user. However, the actual result returned a “This site can’t be reached” error. The method to fix this error should be to validate the input after the database is queried. An incorrect input should reload the page with an error pop-up or have a message displayed on screen with the prior input filled in so that the user can change it.

The next set of tests were for creating user accounts. In these tests, The positive test of account creation with valid input information was confirmed to work as expected with the user redirected to the “Job Creation” screen. Of the validation tests, tests confirmed that special characters were allowed to be used. The final three tests for account creation were for no password provided as input, no username provided as input, and for a duplicate user name. For all of these tests, proper error handling should have occurred and provided the user with feedback on the error but for the lack of input tests, the next page failed to load with an error message indicating that “This site can’t be reached”. The final test for duplicate username allowed for a duplicate account to be created in the database instead of returning an error message. As for log-in, more input validation needs to occur to ensure that the user is receiving adequate feedback for their input.

Two tests were performed to ensure that navigational buttons were working properly. The button to load all jobs in a display was confirmed to work. The test to ensure that the “Pick Job” functionality worked failed. Instead of allowing a user to pick a job, the button returned an error message showing that “This site can’t be reached”.

The last set of tests that were created involved verifying that the “Request a Job” functionality worked properly. Four tests were created to cover both positive and negative inputs. Two of four of these jobs passed. Any input with blank descriptions returned an error screen indicating that “This site can’t be displayed” instead of error handling properly. As before, more input validation needs to occur to ensure adequate inputs.

# 

# 

# Summary of acceptance testing and description of what kinds of bugs, if any, acceptance testing enabled us to find

## Acceptance Tests (please access with OSU email account):

<https://docs.google.com/a/oregonstate.edu/spreadsheets/d/1WXfjFIkrFVzsTh4rIdGj9ieiRoYhf0rXGU9u_mz4glI/edit?usp=sharing>

## Summary

The goal of acceptance testing for XP cycle one was multi-faceted. One of the goals was to make sure that all application processes functioned properly with no errors so we created a number of multiple step test scenarios that mirrored the functionality of the application. This was to ensure that all of the functionality that was wanting to be delivered was tested again on a integration level instead of only on a unit testing level. The secondary goal was to get feedback from a subset of possible users to gain acceptance at the functionality, usability, and design of our application. For XP cycle week 1, these users were not the client that assigned us functionality requirements but possible users of a completed application. Both of these goals were accomplished and many improvements to the application and further increases in functionality will be able to be made based on feedback from this acceptance testing.

For this cycle of acceptance testing, five scenarios were provided to each user participating in the acceptance testing. These users performed each of the five scenarios, documented the performance of each of these scenarios, and supplied comments based on the scenarios about functionality, usability, and design. Within XP cycle 1, there are five main scenarios that need to be tested. The first scenario was for the user to create a user account. The second scenario was for the user to load the homepage a log-in to their user account. The third scenario was for the user to create a job using the “Create a Job” screen that appears after a user logs into their account. The fourth scenario is for the user to load the “Display Jobs” screen by pressing the button the appears in the “Create a Job” screen. The last scenario is for the user to “Pick a Job” from the “Display Jobs” screen.

As discussed previously, the primary goal of testing the five testing scenarios is to confirm functionality, to document errors not discovered in unit testing, and to confirm errors found in unit testing. Four of the five scenarios passed as expected with results appearing in the Acceptance Testing - XP Cycle 1 spreadsheet provided elsewhere in this group of files. All four of these scenarios were unit test cases in the Unit Testing for XP cycle 1 and the results from the Acceptance Testing confirmed these results. The last scenario results in the web page crashing with a “This site can’t be reached” error. The expected result was for the “Picked Job” to be assigned to the user.

The secondary goal of acceptance testing was to receive feedback from potential users about the functionality of the application, usability, and design in order to build a better application that meets the needs of the users and to ensure that the design decisions that were made conform to expected functionality. For acceptance testing - XP cycle 1, two potential users were selected to perform each of the five scenarios and to give comments about each scenario. Comments on the first scenario included a better designed account creation page as well as for the account creation page to be separated from the login screen as it may confuse some users. Otherwise the account creation and login area of the website is straight-forward and well put together. One of the comments that was repeated for multiple scenarios was to spend more time on developing the display of each of the screens to make them more visually appealing. Users also requested that a logout button appears on each of the screens after the user is already logged in (“Create Job” screen and “Pick A Job” screen). For scenario 3, both users requested for more options be added for “Repair Type”. The current options don’t allow for a wide variety during job creation.

# 2nd XP cycle: Summary of changes to the list of user stories, effort estimates, and priority list

The following changes were made to the user stories. The completed tasks are noted and marked in grey, the priorities are noted and marked in blue.The changes to the user stories were mostly connecting and modifying the existing stories.

The first and second user story were combined to create:

1. **Account Creation and Request**

A user provides their personal information and sets up an account with a password.

* 1. create requester/volunteer user account general area, general area, email, phone (*completed*)
  2. profile page (2)
  3. user then continues on to become a requester and submits a request for a home repair to be fulfilled by a volunteer. (3) 1st Priority
  4. requester inputs name, give brief description, type of repair (dropdown), give schedule constraints/ general availability (maybe add calendar later), status of repair (*completed*)
  5. update status of request (*completed*)
  6. put on list/database (*completed*)

The ability to return which was integrated into the third original user story:

1. **Returning User Accepts Request**

A user has set up a user profile previously. The user returns to the site to becomes a volunteer and accepts a repair request, which connects the volunteer with the requester for purposes of scheduling, etc.

* 1. user returns to the site to becomes a volunteer (3) 2nd Priority
  2. view list (*completed*)
  3. requesting user is notified, conversation is initiated (through app) (30)
  4. starts conversation with requester to schedule repair (30)

It became necessary to specify what happens after the volunteer accepts the request which required modification of the fourth original user story:

1. **View Requests**

A user can view a list of open repair requests, sorted/filtered by location or repair type. Once a volunteer accepts a request, the request no longer appears in the list.

* 1. sees list or map of requests (3)
  2. based on capability/availability can accept request (*completed*)
  3. once a volunteer accepts a request, the request no longer appears in the list. (3) 3rd Priority
  4. A volunteer can log in and view requests already fulfilled, and pending requests already accepted, but not yet fulfilled.(3)
  5. starts conversation with requester to schedule repair (30)

*NOTE :The original fifth and sixth user stories remained unchanged and were not prioritized.*

# Summary of how division into pairs for the second XP cycle and comparison to first cycle

## Summary of Division into Pairs

The pair divisions went much the same as the first cycle with team members volunteering in the areas they had the most experience and copilots jumping in as needed.

For the second cycle William volunteered to be the driver in working with the database selections and viewing them. Niza became the copilot for this pair and together they worked to fulfill the client’s requests regarding the display of available versus completed jobs.

David offered to be the driver for the second group with Zachary copiloting. The two worked together to focus on getting sessions working in order to fulfil the client's requests that returning users would be able to perform requester and volunteer functions.

John continued his work on testing and Niza volunteered to copilot for this round.

The actual collaboration happened the same way as well with a mixture of Skype and Google Chat. Meeting were scheduled a little less formally, mostly through Gchat.

## Comparison of 1st and 2nd Cycle

The workflow of the second cycle was less formal with team members meeting with less scheduling and jumping in when needed. This may have been due to a better cohesion as a group or possibly our greater comfortability working with the agile process.

It was also a bit more collaborative during the second cycle with a flourishing Gchat discussion of the current state of the project and a lot of cross talk between groups as we collaborated putting everything together.

Overall, most team members thought that things went much more smoothly the second time around as we became more familiar with both the process and each other.

Probably the only way the first cycle went better was that it was a little less hectic as the deadline was not approaching as fast. This gave the team members more time for reflection and brainstorming ideas for improvement to discuss with the client. The second round of user stories was influenced more by this back and forth with the client which gave the team more of a sense of ownership and pride in the project.

# Reflection: Summary of ways XP was preferable to waterfall, ways waterfall was preferable to XP, and strengths and weaknesses of each process

## XP Preferable to Waterfall

XP had several preferable aspects but among our group the consensus was that the opportunity to learn from each other was one of the greatest benefits when compared with waterfall in which, though of course peer learning is possible, the practice is not as central. Being compelled to pair code was a challenge in many ways, but it gave us a way to expand our knowledge basis that simply didn’t happen with waterfall.

Another big benefit of XP over waterfall is the speed at which we were generating working code for our program. As one of our team members put it, “I’d rather spend eight hours actually coding than four hours making diagrams of something that might not even be put into production.”

The last major benefit we agreed on was the close relation we were able to establish with our client and the satisfaction of knowing that what we were producing was exactly what someone wanted. Again, this is also possible in waterfall, but since it isn’t as emphasised, we felt that none of us had the experience of working so closely with the client on our first projects.

## Waterfall Preferable to XP

We found that while waterfall had some drawbacks, there were also some aspects we missed. One of the main areas we struggled with XP was scheduling and in waterfall scheduling was much easier. Being able to work alone in waterfall allowed our first projects to actually devote more time to the projects with no need to wait for someone to meet up in order to get started.

Division of tasks was also much easier as people naturally just volunteered to work on their area of expertise and there was no need to balance skills. In XP at times team members sometimes felt under or over utilized depending on their pairing. Of course we were able to balance this on the second cycle.

## Strengths and Weaknesses of Each Process

XP Strengths: *fast, generating real usable product, learning opportunities, flexible*

XP Weaknesses: *difficult to coordinate, hard to estimate time, wait for team members/client*

Waterfall Strengths: *easy to get started and do work, thoughtful and well planned*

Waterfall Weaknesses: *a lot of paperwork before actual product, some work may not be used*