Project B - Group 2

http://assistance-for-dementia-sufferers.000webhostapp.com/ProjectB/accounts/login.php

Kevin Conner
Oregon State University
connerke@oregonstate.edu

Sean White
Oregon State University
whites3@oregonstate.edu

Josh Huff
Oregon State University
huffj@oregonstate.edu

Benjamin Carlson
Oregon State University
carlsobe@oregonstate.edu

Abstract

This paper describes the steps taken to form a solution using eXtreme Programming. The solution is to provide a web resource for those suffering from dementia. In this paper the user stories, priorities, and estimates of tasks are listed. Also, summaries are given for pair programming, bugs found, and acceptance testing. Two cycles are described in total.

1. User Stories

1. Establish account - The user can create an account and give access to trusted associates like family and caregivers.
2. Access account - The user or trusted associate can login to their account.
3. Log memory incidents - The user can easily note and store incidents of memory lapse, disorientation, etc. to their own account.
4. Log observed memory incidents - Trusted associates can log incidents to another person's account.
5. Track other factors - The user or trusted associate can log such information as sleep, stress, medications, etc.
6. Visualize patterns - The user or trusted associate can view memory incidents and/or other factors in some kind of graphic form to track increases over time or correlation between incidents and taking medication for example.
7. Access games - The user can select different types of brain-training games or puzzles from a menu.
8. Access game stats - The user or trusted associate can track game/puzzle statistics over time.
9. Schedule - The user or trusted associate can set up recurring reminders on a calendar for different events such as appointments, birthdays, taking medication, or anything else that they might want help remembering.
10. Upload items to virtual scrapbook - The user or trusted associate can upload personal items such as photos, video, audio recordings, blog posts, etc. including details about the items (names of people and places etc.)
11. View scrapbook - The user can browse through items in the scrapbook, or click on an item for more detail.

12. Follow step-by-step instructions - The user can access basic step-by-step illustrated instructions for tasks that they need help with, like getting dressed, morning hygiene routine, changing a lightbulb, etc.

2. Estimate of Tasks and Effort

Each story topic shows the total amount estimate of tasks and effort required measured in units. The units represent hours spent. 1 unit = 4 hours. These topics are broken and measured into sub-sections.

1. Establish account - 5 units
   3u - Create basic user database
   2u - Create basic account creation webpage
2. Access account - 2 units
   2u - Create alternative log-in page
3. Log memory incidents - 3 units
   2u - Create webpage to display logging options
   1u - PhP and database to implement note-taking
4. Log observed memory incidents - 3 units
   2u - Create webpage to display logging options
   1u - PhP and database to implement note-taking
5. Track other factors - 4 units
   1u - Create webpage to display logging options
   3u - Python script to implement note-taking
6. Visualize patterns - 2 units
   2u - Research report and graph generation. Excel?
7. Access games - 3 units
   3u - Integrating brain-training games with website
8. Access game stats - 2 units
   2u - Create webpage to display scores from games
9. Schedule - 4 units
   1u - Create webpage to display important dates
   3u - Integrating Google Calendar API with website
10. Upload items to virtual scrapbook - 6 units
    3u - Creating blog feature
    3u - Integrating media player with website
11. View scrapbook - 2 units
    2u - Integrating image gallery feature with website
12. Follow step-by-step instructions - 6 units
    5u - Creating static content for task help
    1u - Integrating content with website

3. Customer priority list

1. Establish account
2. Log memory incidents
3. Schedule
4. Associate accounts
5. Log observed memory incidents
6. Track other factors
7. Step-by-step instructions
8. Upload to scrapbook
9. View scrapbook
10. Visualize
11. Game creation

4. Pair Programming Summary (First Cycle)

Ben and Josh elected to implement the customer's story about a user tracking factors indirectly related to the presence and severity of dementia. They struggled with the concept, particularly with how to make the feature substantially different from the memory incident log that Kevin and Sean were developing. It seemed like a waste of time to make the same type of feature with only minor differences, so they looked more closely at the vision statement for information that would help clarify how tracking factors and a memory incident log differed as user stories. There was the possibility of violating YAGNI and creating an over-engineered solution when there was likely a more direct way to approach the problem.

The answer came when they realized the "track factors" data could easily be stored as enumerated responses, whereas the memory incident log could not. It was necessary for Kevin and Sean to give the user a text field, but the cases presented by the major factors could be reduced to structured and predictable input (which, conveniently, also minimizes the need for input validation). Additionally, because precision is largely unnecessary, and because it was in the best interests of the user, the answers were meant to be qualitative as opposed to quantitative (e.g. Q: "How did you sleep last night?", A: "Bad" -- not Q: "How many hours did you sleep last night?, A: "6-7 hours").

In keeping with the spirit of Agile and to satisfy the users' pronounced need for usability and simplicity, Ben and Josh decided to start with the idea of the user choosing from a small handful of unambiguous choices in answering questions about sleeping patterns, diet, medication, side effects, and overall mood. Much of the pair programming time was used deciding how to write unit tests that would meet the requirements of the story without inviting unnecessary complexity. For example, instead of asking the user to specify which side effects they suffered, the unit tests are concerned with the presence of side effects in general, because reviewing the frequency of side effects is likely more informational to family and healthcare providers than the exact nature of the side effects themselves.

At the outset of their first pair programming session, Ben brought Josh (primarily a user of a bare text editor and g++) up to speed with the use of the cloud9 IDE with a brief tutorial. Also, because web development is one of Josh's weaknesses, Ben took the lead in drafting the first iteration of the HTML while Josh prepared a simple draft of a SQL table to store the data gathered by the feature. Then, Ben provided a great deal of guidance while Josh implemented the PHP. They took the time to first ensure the SQL table worked by testing it directly, then ensured the PHP correctly passed the data to the table, and finally ensured that the radio buttons interfaced with the PHP.

Inevitably, there were lots of syntax errors, unexpected issues with credentials, speed bumps when using a new and unfamiliar IDE, and lots of "comment-out and do-over" mini-iterations. But the programming pair realized the advantage of having two sets of eyes on every bug, and because the pair worked through the logic of the features expected output and designed to well-defined unit tests, the coding was far easier than it would have been otherwise.

Sean and Kevin worked on making an accounts page and the back-end code that would be used for all pages to gain access to them. To solve this, sessions were researched and implemented. There was some confusion initially how to start the sessions and on which pages the code would need to be included. This was remedied by placing the code at the top of the page before all HTML code. It was ensured that users were redirected to the login page if they did not have the correct login information.

The memory incident page was updated to be associated with a user account. In this way each entry into the log will be associated to each individual user. This was done by using the session variable when the user puts in the username on login page. That variable is inserted via SQL into the database to link the entries with the account. Now the display log will only show the user's logs that they entered, instead of everyone's logs.

5. Unit Test Summary

The **"memory incident log" feature** is meant to accept text input from the user via a PHP page and pass them to a database via a SQL Insert.

Test Case: The user should be able to add a text entry describing an incident involving a lapse in memory or concentration.

Input (via text box)	**Result**	**Status**
"I could not remember my niece's name."	**Entry stands in SQL table**	**Verified**
"I forgot my phone number."	**Previous entry persists. New entry stands.**	**Verified**

Test Case: The user should be able to view the log of incidents involving memory.

Input	Result	Status
Clicking “Display Log”	**All previous entries persist**	**Verified**

The **“step-by-step instructions” feature** is meant to accept text input from the user via a PHP page and pass them to a database via a SQL Insert.

Test Case: The user should be able to add a text entry describing a process for future reference.

Input (via text box)	Result	Status
Instructions for making oatmeal	**Entry stands in SQL table**	**Verified**
Instructions for fastening a necktie	**Previous entry persists. New entry stands.**	**Verified**

Test Case: The user should be able to view the instructions in a legible format.

Input	Result	Status
Clicking “Display Instructions”	**All previous entries persist in a neatly formatted table.**	**Verified**

The **“track factors” feature** is meant to accept radio button input from the user via a PHP page and pass them to a database via a SQL Insert.

Test Case: The user should be able to indicate how well they slept.

Question:	**How did you sleep last night?**

Input (via radio button)	Result	Status
"Well. I slept like a baby"	**"good"**	**Verified**
"Ok. I tossed and turned a little"	**"okay"**	**Verified**

"Not well at all. I couldn't sleep"	**"bad"**	**Verified**

Test Case: The user should be able to indicate whether they took medicine that day.

Question: **Were you supposed to take medicine today?**

Input	**Result**	**Status**
"I took my medicine"	**"yes"**	**Verified**
"I forgot to take my medicine"	**"forgot"**	**Verified**
"I don't take medicine"	**"n/a"**	**Verified**

Test Case: The user should be able to indicate if they suffered side-effects.

Question: **Did your medication cause any side-effects?**

Input	**Result**	**Status**
"Yes"	**"yes"**	**Verified**
"No"	**"no"**	**Verified**
"I didn't take medicine today"	**"n/a"**	**Verified**

Test Case: The user should be able to indicate how their appetite was.

Question: **How many meals did you eat today?**

Input	**Result**	**Status**
"0"	**"zero"**	**Verified**
"1"	**"one"**	**Verified**
"2"	**"two"**	**Verified**
"3"	**"three"**	**Verified**

"4"	**"four or more"**	**Verified**

Test Case: The user should be able to indicate their overall mood.

Question: **What was your mood like today?**

Input	**Result**	**Status**
"I was happy"	**"happy"**	**Verified**
"I was sad"	**"sad"**	**Verified**
"I was angry or stressed"	**"stressed"**	**Verified**

Because the inputs are enumerated, most of the actual testing revolved around ensuring that the SQL table was correct, the addform.php page submitted the table data to the SQL, the factors.php page submitted the radio button selections to the addform.php page.

The **calendar** is designed to provide an accurate representation of the current month and year. In addition, the user will be able to both view and add appointments. Here are the unit tests:

Test	**Result**	**Status**
Accurate Month/Year	June/2017	Verified
View Existing Appointments	No appointments. (No appointments had been added yet)	Verified
Add New Appointments	Created new appointment on June 16	Verified
Highlight Current Day	June 9th was highlighted in grey.	Verified

The **login** feature enables a user their own account via sessions. It also blocks anyone who navigates to the site without having first logged in.

Test	**Result**	**Status**
Navigate to site without session	Redirected to login page	Verified

Attempt login with invalid credentials	Prompted to sign up for account	Verified
Create account	Information accepted into database	Verified
Log in with valid credentials	Directed to home page	Verified

The **scrapbook** feature enables users to upload their own images to the site. Images are then aggregated into a gallery for the user to view.

Test	**Result**	**Status**
Upload image	Image was successfully uploaded	Verified
View Scrapbook	Uploaded image appears in the gallery	Verified
Upload second image	Second image does not overwrite the first	Verified

6. Acceptance Testing

The Project B assignment requirements told us not to worry about having the customer involved for the acceptance testing phase, but we all believed having her input would be instructive and sent her our progress when we had a minimally viable iteration. Since our unit tests were successful and our integration testing went (mostly) without any significant hitches, it was dispiriting to have trouble when we deployed our PHP pages and shared them with the customer. That's because, while it worked for us on our respective machines, Aimee ran into issues.

Having Aimee as a customer was helpful for our first time practicing the Agile method. As a CS student, she has some insight into the sorts of issues that may arise in developing web apps. In particular, she was helpful in positing the theory that the errors may be caused by our using different browsers (i.e. we developed it in Chrome and she tested it in Firefox). However, Kevin tested it on Firefox prior to sharing the link with her, so we came to believe it was a matter of hosting.

After developing the pages in cloud9, we started using a third-party web hosting solution called 000webhostapp.com, which offered free hosting and support for MySQL and PHP. However, at the outset, it seemed to be unreliable. Our relatively light-weight pages were causing time-outs.

Even when the other features worked successfully, the web host was throwing opaque error messages about permissions, and it made the previously functional gallery feature a nightmare to debug. Since the feature worked fine when we developed it on cloud9, we deduced it wasn't a coding error of ours, but that the web host's configuration was not properly accommodating our pages. Perhaps if we were paying customers to a professional-grade web hosting company, we could get support and have the fine details worked out for us. As it stands, we are unlikely to get that level of support in a timely manner consistent with Agile principles.

Therefore, we determined that between spending a ton of time trying to bend the web host to our will and migrating, the easier fix was the latter. That is, until Kevin discovered an easy fix. By specifying a more generous connection timeout than the inadequate default, we were able to avoid a big hassle in migrating our content to a different host.

After the fix, the sign-up and log-in functions were error-free (as unit and integration testing had previously led us to believe they would be), and the step-by-step instructions and memory incident log worked correctly.

7. Second XP Cycle

7.1 Changes To List Of User Stories

The only changes are the remaining stories left. There aren't any new stories to report.

1. Visualize patterns - The user or trusted associate can view memory incidents and/or other factors in some kind of graphic form to track increases over time or correlation between incidents and taking medication for example.
2. Access games - The user can select different types of brain-training games or puzzles from a menu.
3. Access game stats - The user or trusted associate can track game/puzzle statistics over time.
4. Schedule - The user or trusted associate can set up recurring reminders on a calendar for different events such as appointments, birthdays, taking medication, or anything else that they might want help remembering.
5. Upload items to virtual scrapbook - The user or trusted associate can upload personal items such as photos, video, audio recordings, blog posts, etc. including details about the items (names of people and places etc.)
6. View scrapbook - The user can browse through items in the scrapbook, or click on an item for more detail.
7. Follow step-by-step instructions - The user can access basic step-by-step illustrated instructions for tasks that they need help with, like getting dressed, morning hygiene routine, changing a lightbulb, etc.

7.2 Changes To Estimates

Units are measured the same was as they were initially. The units represent hours spent. 1 unit = 4 hours. These topics are broken and measured into sub-sections. This list shows the current estimations for the second cycle on the remaining stories left.

1. Visualize patterns - 4 units

   4u - Research report and graph generation. Excel?

2. Access games - >= 8 units
   3u - Integrating brain-training games with website
   5u per game - Create game to integrate

3. Access game stats - 4 units
   2u - Create webpage to display scores from games
   2u - Retrieve states from game when it is completed

4. Schedule - 2 units
   2u - Create webpage to display and add important dates

-The original estimate was for 4u (16 hours). What changed this estimate was a decision not to build the calendar using the google calendar API. After some research, a simpler approach was identified that leverage the use of jQuery, AJAX, PHP, and MySQL. This different approach cut the estimate in half to just 2u (8 hours).

5. Upload items to virtual scrapbook - 4 units
   2u - Creating upload processor with php
   2u - Create form to get the image to process

6. View scrapbook - 2 units
   1u - Use SQL to retrieve pictures from database
   1u - Implement picture in html for display

7. Follow step-by-step instructions - 6 units
   3u - Create HTML form to insert instructions in database.
   3u - Use SQL to retrieve instructions from database and insert into table.

7.3 Changes To Priority List

After completion of the first cycle, the team submitted an update to the customer for feedback and direction on if any of the remaining priority stories should be modified. The customer didn’t see a reason to modify the remaining stories or the priority.

7.4 Pair Programming Summary (Second Cycle)

For the second cycle, we switched up the pairs. Kevin & Josh paired up and Ben & Sean paired up. The efforts in the second cycle were definitely more streamlined. We were all new to pair programming, and the anxiety of having a stranger watch us code in real-time was mostly forgotten. Still, after just one week, we saw a marked increase in productivity. We attribute the increase in both pace and relative ease of work to three factors: 1) our communication was less cumbersome, 2) our vague ideas were becoming tangible objects to work on, and 3) our experience with the code base gained from the first cycle helped us identify problems more quickly and more easily.

We also saw an improvement in our ability to communicate ideas with each other, using a kind of shorthand in chat that simply wouldn't work in an asynchronous environment — it would only be possible in a shared context made possible by pair programming. As early as the middle of the first cycle, the team had a more concrete concept of what the project was going to look like, and the shift from abstract discussion to hard implementation meant that the team spent less time creating new ideas and more time refining existing ones. We believe this is a correct reflection of XP, because we started with something small and viable that snowballed into something greater. The speed of the development also increased as a result of improved identification of bugs. Since all members were up to speed on the technologies used and more familiar with cloud9's interface, less time was spent on the administrative part of development and more time was available for actual coding and review. However, as Ben and Sean discovered, the process doesn't preclude bugs and problems entirely.

While working on the calendar feature, Ben and Sean decided early on that it would be too much time- and work-intensive to build a calendar from scratch, particularly since there are so many existing calendars that would serve our purposes just as well. After some research online, they discovered some tutorials to help them assemble a basic calendar using our chosen tech stack (and a bit of jQuery). Unfortunately, the tutorial they chose was designed as a "tease" to lure customers into a paid course on web programming, and the pair was left to "fill in the blanks" on their own.

Some issues involved in implementing this user story included the absence of a CSS style sheet and a preponderance of jQuery, which the pair did not have much experience with. This inexperience made implementation, especially debugging, a challenge. Namely, they initially overlooked the absence of a critical jQuery library.

Most aggravating was the fact the tutorial itself had incorrect code! When trying to add functionality to the "add event" button, the pair discovered that there was a bug in the code that broke the calendar. Because this code came from outside of the group's XP process — and because the pair made the fair assumption that the code drafted for educational purposes would be correct — it did not benefit from the improvement in overall workflow that other stories did. Still, after some careful debugging, they identified the offending code and adjusted it to make the button function properly.

Sean and Kevin implemented the Gallery page. An upload handler was used to take a picture that was chosen to be uploaded by the user that is saved in the database. The handler checks for errors or lands the user on a success page if there aren't any errors. From here SQL is used to save the file name as text associated with the account owner in the database. SQL is used again to retrieve all the pictures the account holder has uploaded and they are all displayed in the Gallery page. The SQL just retrieves the file name as text. The full address was not working properly to call the files so a function was created to cut part of the address to properly call and display the pictures. Some changes were made in the error handling was optimized based on file size of images uploaded.

8. Reflection

After using the waterfall method, the adoption of XP practices was liberating. Though we were careful to avoid “hacking” and consequently winding up with a dog house instead of a high-rise, it was fun to go from loosely outlining what the user stories meant to sketching out pseudocode in a matter of hours, as opposed to a matter of weeks.

Naturally, the feeling of freedom XP gave us didn’t come without some baggage. It was comforting to have plenty of time to plan, discuss, and even go back to square one if a group member correctly determined that a feature was not panning out in a manner befitting the “grand scheme.” With XP, the planning phase of each one-week cycle is at no leisurely pace. Instead, the planning is highly compressed because the emphasis is on turning up with a working product, albeit a simple one. Sometimes it was frustrating to imagine that the features we were developing might end up being uncomfortably outside the customer’s range of expectations. These moments of doubt were reminders of the guidance all the deliberation in Project A provided to us. Of course, one programmer’s safety blanket is another’s straitjacket.

Waterfall lends itself to a hierarchical and bureaucratic process. Project A group members were nothing short of terrified at the prospect of advancing to the next phase of development, because each phase was another opportunity to have missed crucial details and to have no means of rectifying it. In waterfall, each step is a Rubicon to be crossed. In contrast, the bite-sized nature of user stories in XP meant that development could at times be “embarrassingly parallel.” It’s conceivable (and exciting!) that, with a crew of a dozen or so programming pairs, we might have been capable of developing all the user stories at once in a given week or two. Certainly, there would be a back-loaded effort to integrate all those stories into a cohesive and functional whole, but it’s a definite improvement over the highly serial and regimented waterfall process.

A critic of the waterfall method might think of Publilius Syrus’ maxim, “It is a bad plan that admits of no modification.” A critic of eXtreme Programming might think like Euripides, who said, “A bad beginning makes a bad ending.” Neither Publilius Syrus nor Euripides were software engineers (probably), so they wouldn’t necessarily be authorities on the subject. As is the case with most engineering matters, comparing waterfall to XP is really a discussion of trade-offs and matching the nature of a task with the appropriate tools.

The waterfall method—the three-piece suit with a pocket square—has its place in the creation of software with clear and fully defined requirements, and XP—the activewear and Nike cross-trainers—has its place in the creation of software with nebulously defined or evolving requirements. Adopting the waterfall method means measuring twice and cutting once to get things right, and adopting the XP method means getting something small done right away and adding to it until it's ready to go to market and beat the other guys to the punch.

Individual taste aside, there is little reason one method is "better" than the other. Waterfall and XP remain viable options given the right circumstances, because if there was a clear choice across all domains, one of them would have disappeared long ago.