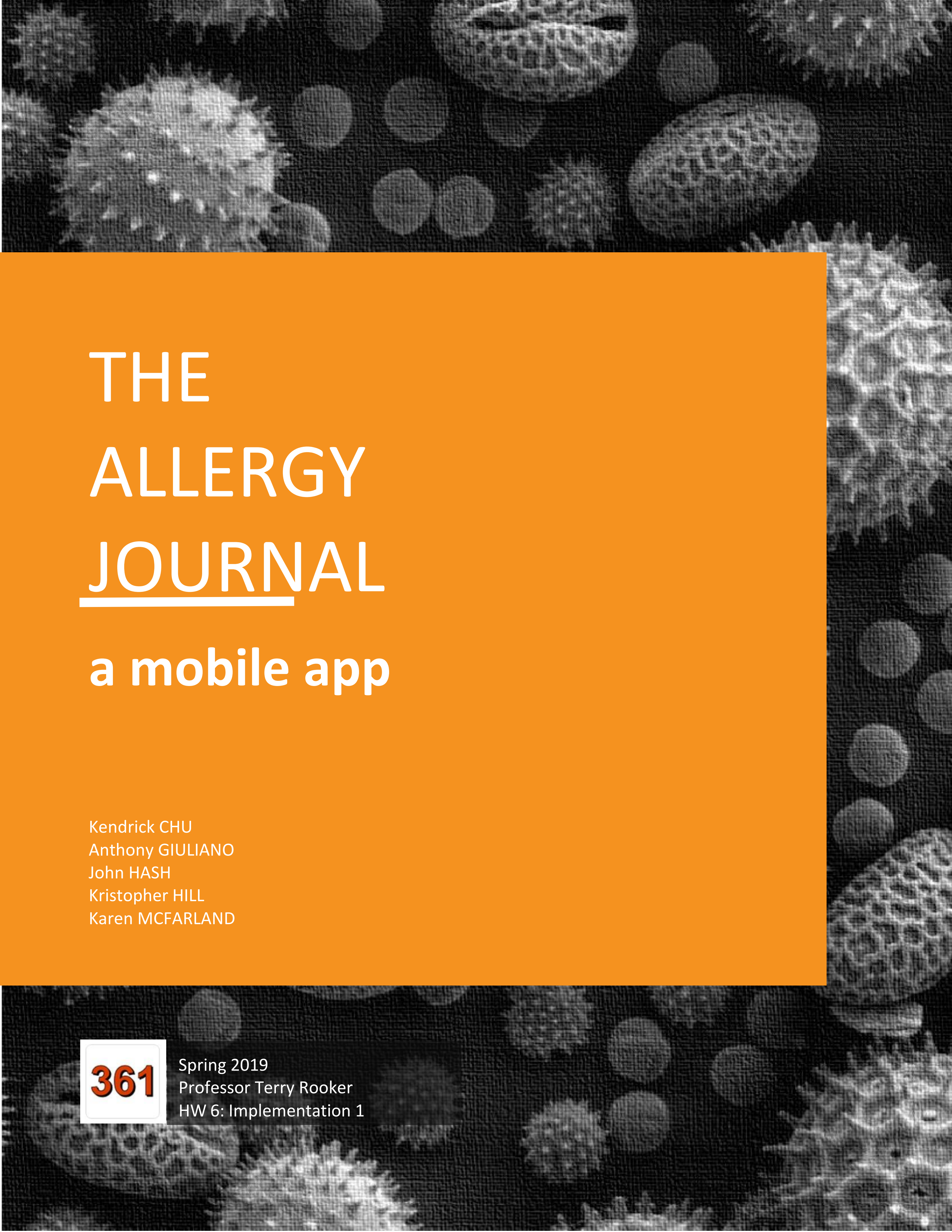


THE ALLERGY JOURNAL

a mobile app

Kendrick CHU
Anthony GIULIANO
John HASH
Kristopher HILL
Karen MCFARLAND

361

Spring 2019
Professor Terry Rooker
HW 6: Implementation 1

URL To Software Prototype

The application's homepage is found at the following url:
http://web.engr.oregonstate.edu/~hashj/appHome.html

The website application works on all major browsers (Chrome, Firefox, Safari, and Microsoft Edge). The page cannot be properly viewed in Internet Explorer. In order to view any of the pages that have links on the homepage, users should be either on the Oregon State University campus or connected via VPN. The homepage will load over any network.

User Stories Information:

Pair Programming

The following table shows the pairs of teammates who worked on a given user story's tasks together.

User Story	Kendrick CHU	Anthony GIULIANO	John HASH	Kristopher HILL	Karen MCFARLAND
0.1 Environment Setup - Database				X	X
0.2 Environment Setup - Webpages	X	X	X	X	X
US 1: Add Product		X		X	
US 2: Document Product Use		X	X		
US 3: Document Reaction	X		X		
US 5: View Product Summary	--				--
US 6: View Reaction Summary	--			--	

Relevant Unit Tests

The following table shows the relevant unit tests that were conducted.

User Story	Relevant Unit Tests
0.1 Environment Setup - Database	After creating and populating the database, various CRUD statements were written to check that the table structure was correct and that the business data was also correct.
0.2 Environment Setup - Webpages	1) Page links: We tested the navigability of the website by testing all links. 2) Browser compatibility: Webpages were tested in all major browsers. 3) Persistence: We tested that all webpages were still accessible after logging out of the OSU server. 4) Accessibility and usability: We ensured the readability of fonts and clarity of purpose for each of the webpages.
US 1: Add Product	1) Browser compatibility: We tested that the webpage would load properly in all major browsers. 2) Check submit button: We tested that when we entered information into the text and number fields for the product name and the barcode, respectively, and then clicked the submit button, the information is sent to the database and put in the Products table. 3) INSERT SQL queries: We checked the Products table in the PHPMyAdmin database after submitting from the front-end and the correct information was where it was supposed to be in the database. 4) Required fields: We verified that the web form does not submit unless required fields contain values
US 2: Document Product Use	1) Browser compatibility: We tested that the webpage would load properly in all major browsers. 2) INSERT SQL queries: Even if the queries have not been integrated into the application at this stage, we have been testing them individually using either DB Fiddle or in PHPMyAdmin. 3) SELECT SQL query: We needed a query to select the proper product use instance id, and we tested them alone in Fiddle and PHPMyAdmin. 4) User cursor: We tested that the mouse cursor changes shape to indicate body image is clickable.

User Story	Relevant Unit Tests
US 3: Document Reaction	1) Browser compatibility: We tested that the webpage would load properly in all major browsers. 2) INSERT SQL queries: Even if the queries have not been integrated into the application at this stage, we have been testing them using either DB Fiddle or in PHPMyAdmin. 3) SELECT SQL query: We needed a query to select the proper product use instance id, and we tested them alone in Fiddle and PHPMyAdmin. 4) Build form functions: We tested the JavaScript code to ensure that each form builds. 5) Destroy form function: We tested the JavaScript code to ensure that the form is destroyed.
US 5: View Product Summary	N/A
US 6: View Reaction Summary	N/A

Problems encountered

The following table shows the problems that were encountered.

User Story	Problems Encountered
0.1 Environment Setup - Database	There was miscommunication about how the database needed to interact with the front-end, and so there was a lot of additional work done after the database was set up to make the requests being sent to it from the front-end recognizable to the database and do what they were supposed to do.
0.2 Environment Setup - Webpages	Miscommunication: At time of integration, we realized that we had not all agreed on how the webpages would be served. Some believed were going to combine all our server-side code into a single document and other thought we were keeping separate server-side code and linking them all together through the main website.
US 1: Add Product	There was confusion about how a user added a new product to their list of products, and how that was different than the list of approved products run by administrators of the service. So, the way that this part of the system functions in relation to the rest of the system is quite different than what it needs to be.

User Story	Problems Encountered
US 2: Document Product Use	Setting up the website and creating the client-side code provided few difficulties. The most difficult problems faced so far are getting the SQL queries properly executed within the server code. Taking this issue to the larger team proved effective, as we were able to figure it out as a group.
US 3: Document Reaction	Again, most difficulty here was establishing communication between the client code and the server code that contains the SQL queries.
US 5: View Product Summary	N/A
US 6: View Reaction Summary	N/A

Actual Time Required for Each Task

The following table shows the time require for each task in a given user story.

User Story	Time Spent on Tasks
0.1 Environment Setup - Database	DB setup and creation: 3 hours; DB Population: 2.5 hours; DB testing: 0.5 hours; Total hours: 6 hours
0.2 Environment Setup - Webpages	HTML, CSS: 2 hours; Total 2 hours
US 1: Add Product	HTML, CSS: .5 hours; Server-side JavaScript: .5 hours; SQL queries: .5 hours; Total: 1.5 hours
US 2: Document Product Use	HTML, CSS: 2 hours; Client-side JavaScript: 6 hours; Server-side JavaScript 1 hour; SQL queries: 3 hours; Total: 12 hours
US 3: Document Reaction	HTML, CSS: 2 hours; Client-side JavaScript: 6 hours; Server-side JavaScript 1 hour; SQL queries: 3 hours; Total: 12 hours
US 5: View Product Summary	This user story was not started. Therefore, no time was spent on its tasks during the sprint.
US 6: View Reaction Summary	This user story was not started. Therefore, no time was spent on its tasks during the sprint.

Current Status

The following table shows the current status of each user story in the sprint.

User Story	Current Completion Status
0.1 Environment Setup - Database	Fully implemented and tested. The database is connected to the front-end.
0.2 Environment Setup - Webpages	Fully implemented and tested. All web pages that have been created are accessible from the main, splash page and URLs are stable.
User Story	Current Completion Status
US 1: Add Product	Not fully implemented. Two of the subtasks were not implemented at all, and one subtask was implemented largely incorrectly.
US 2: Document Product Use	Not fully implemented. Two of the three major subtasks outlined in HW 5 are complete and tested. The Document Product Use webpage cannot communicate with the database at this point.
US 3: Document Reaction	Not fully implemented. Two of the three major subtasks outlined in HW5 are complete and tested. The page and forms are completed, and some data can be inserted and accessed from the database.
US 5: View Product Summary	Not implemented
US 6: View Reaction Summary	Not implemented

Remaining Work to Be Completed

The following table shows the remaining work to be completed for each user story in the sprint.

User Story	Current Completion Status
0.1 Environment Setup - Database	Nothing. This task is complete.
0.2 Environment Setup - Webpages	1) Navigation bar needs to be added to non-homepage webpages to allow easy access to other application features. 2) The layout of the pages and text will be slightly altered to make it more obvious to the users the order in which steps should be followed for entering their user data.

User Story	Current Completion Status
US 1: Add Product	1) Panel for submitting admin product add 2) Change methods so that customer adds product from select menu of products already in the database (SQL select statement) 3) Add a table in DB for tracking which products are being used by which users 4) Change SQL insert statement to add to new user-products table product id and user id, instead of adding product to products table
US 2: Document Product Use	1) SQL queries on server-side JavaScript file need to be updated to the correct, current versions based on database redesign choices. 2) Complete the remaining XMLHttprequests on the client-side. 3) Express routes need updates on server-side. 4) Convert the default values of the checkboxes in submission forms to match the datatypes of the corresponding fields in the database.
US 3: Document Reaction	1) SQL queries on server-side JavaScript file need to be updated to the correct, current versions based on database redesign choices. 2) XMLHttprequests that send user data to the database need to be completed. 3) Express routes need to be completed on server-side. 4) Find the simplest way to make sure that instanceID is not accidently increased by "1" each time a user adds data to the same reaction instance.
US 5: View Product Summary	The tasks for this user story were not started. It will be added to the next sprint.
US 6: View Reaction Summary	The tasks for this user story were not started. It will be added to the next sprint.

HW 5 Diagrams

Usefulness of Spikes and Diagrams

User Story 1

We actually did not reference the UML diagram for this user story while building the page for this story. Although we should have, because that would have made clear to us that we had misinterpreted what this page was supposed to do and we could have come closer to completing this unit this week instead of (hopefully) next week.

User Stories 2 and 3 (Image Map Spike)

Each of these required an interactive image for the user to click on. The spike we performed allowed us to explore methods for achieving this functionality and ultimately settle on using an image map of polygons overlaid on a body image that fulfilled our needs.

User Story 4 Spike

This spike was valuable in that in prevented us from wasting too much time trying to implement functionality that the OSU servers would no support. Camera access is not allowed, so further development of the application using the Scandit SDK would have been a large opportunity cost.

User Story 5

This user story was not developed further this week. It has been added to the next sprint.

User Story 6

This user story was not developed further this week. It has been added to the next sprint.

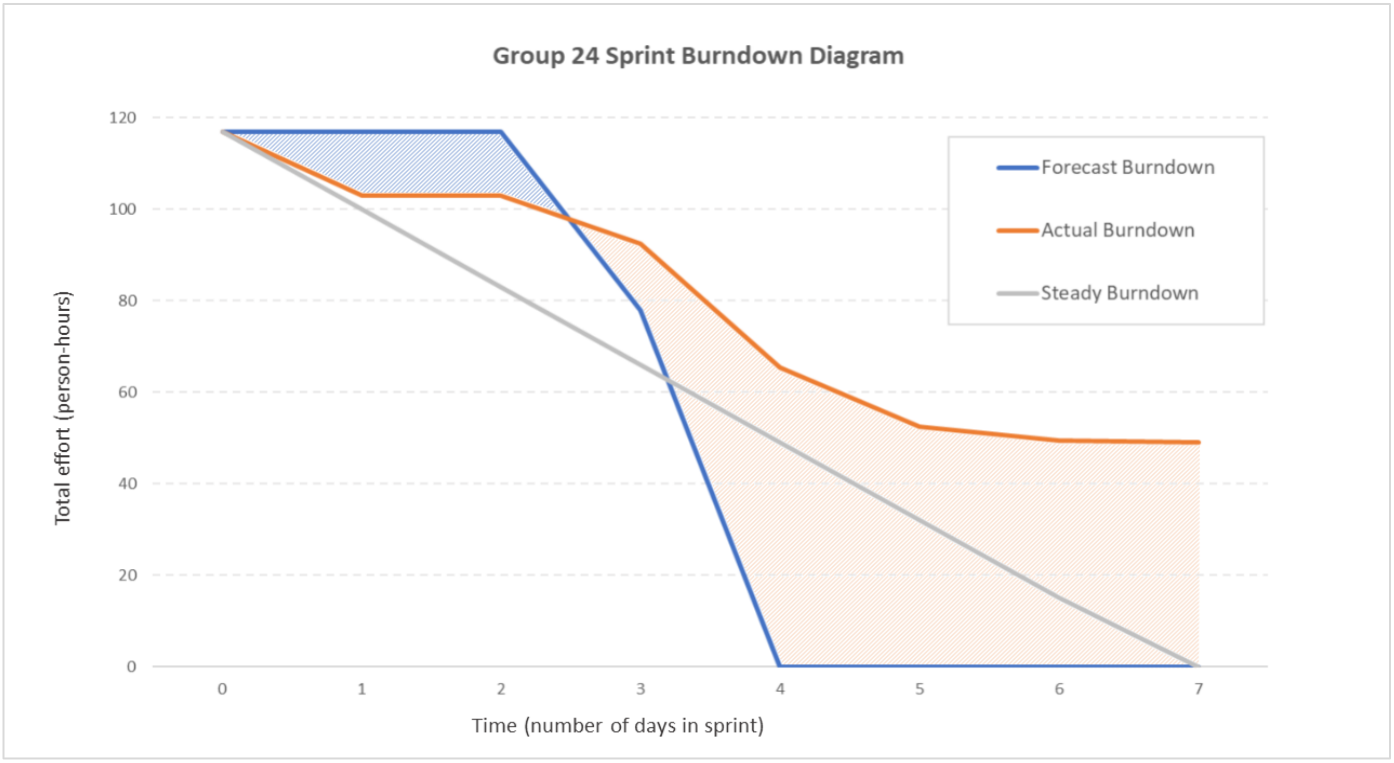
Sufficiency of HW 5 Diagrams

Looking over the problems encountered during our sprint, a few diagrams might have helped clarify and resolve issues faster, which could have led to a more productive sprint. For use case 0.1 (Environmental Setup - Databases) and use case 1 (Add Product), data flow diagrams would have been helpful. In particular, a sequence diagram showing the data flows would have been useful to specifically identify the data that was needed to support tasks and to bring clarity to the data requirements and alignment on what would be needed in the actual code and when. The data flow diagrams would have given us more insight in what information needed to be stored in the database, how this data should be structured, and how the parts of the application would interact with the data.

Specifically, for use case 1, there was confusion with whether the “Add A Product” page should have a list of already approved products for users to select, or if users could add directly to the product table in the database to track the products they use. A data flow diagram showing exactly how the data would interact with the “Add A Product” page, such as the page querying for product data to display to the user to select, or having the page insert data directly into the product table, would have been useful in planning for this use case.

Use cases 3 (Document Product Use) and 4 (Document Reaction) also would have benefitted from data flow diagrams, because the main issues encountered in these use cases were connecting to the back-end database and providing the data necessary to insert data into the relevant tables. When these parts of the application were created, we did not know what information was needed to be sent to the database, which ended up proving to be a major problem in getting these parts of the application running in a small time frame. A data flow diagram specifying what data was needed from these pages, and where this data needed to be inserted into the database would have cleared some of these issues and would have made implementation of these use cases easier.

Sprint 1 Burndown diagram



As shown in our burndown chart, we were not able to complete all of the user stories that we included in Sprint 1. The area shaded in blue shows that initially we were ahead of schedule, but as indicated by the area shaded in orange, we fell behind in our schedule. The steep drops shown in our diagram indicate the we did not sufficiently subdivide our tasks and assign corresponding progressive due dates. Our tasks were not very granular and they also fell on 1 of 2 due dates that we used. Another issue that our diagram surfaced was that our time estimates were not as accurate as we had hoped. Underestimating the time to complete tasks also led to us falling behind schedule. The entire team recognized these issues and we have created more granularity in our tasks for Sprint 2 and adjusted our time estimates. We expect the actual burndown line for Sprint 2 to be more gradual as a result. We also fully expect to successfully complete all of the tasks in our upcoming sprint.

Refactoring

Even in this short sprint, we did need to do some refactoring. When we started to integrate the database with the record a reaction and record a product usage pages, we realized that we had unnecessary complexity in or database design. We altered a table and removed columns that were needed as we decided to change how the body part selections were inserted into the database. Initially it was going to be a two-step process so that we could keep all of the body parts selected together in one instance. We decided that it would be easier to directly handle the auto-incrementing of the id in our code instead of letting the database handle it for us. Thus, we removed complexity but maintained the same required functionality.

Questions for Customer During the Sprint

We did not need to ask our customer any questions during the sprint.

Integration Tests

Database/Product Add Integration

a. Form submission with empty fields: We tested inserting products with empty fields. There aren't any constraints on the database side to prevent this, so we added `required` attributes to our form fields to prevent empty rows.

b. Duplicate form submissions: We tested how adding a product responds to duplicate form submissions, both accidental and unintentional. The form seems particularly vulnerable to duplicate submissions, but we don't have any uniqueness requirements on our product records. We'll have to discuss options with the team, whether it's at the database level, or whether the server looks for an existing product before trying to insert a new one.

Database/Reaction Instance Integration

a. Returning row id from database: Returning the correct instance ID is crucial to keeping user data associated with the correct reaction instance or product use instance. We tested the integration of the database with the server and client side JavaScript functions to ensure that the correct id was returned.

b. Inserting user reaction data into database: Several parts work together in this case including the database, SQL query, XMLHttprequest on the client-side, and Express on the server-side. While testing whether data could be properly inserted, we found that the HTML checkboxes return values different than we anticipated. We assumed they were boolean values (e.g. checked = true) but found out that check equals "on." We now need to convert "on" and "off" to boolean values in the next sprint so that they can match the datatype in the database fields.

UI, HTML forms, and Client JS

a. Interaction with image map overlaid on cartoon body: We tested the edges of the image map polygons to ensure that relevant major body regions were covered by the map. We tested that each major body region was selectable. This constitutes testing the integration of the user's visual experience with the code that makes the image clickable.

b. Correct body sub region form loads: We tested all the major body regions to ensure that the correct form loads the right of the body image and that the proper sub regions appear in the form. This tests the integration of the image map with the JavaScript functions that build the tables. During this testing we found we needed to build a function to destroy the previous forms before building another. We also found that we needed to nest certain functions that added event listeners inside another function that was called when the DOM content loaded.

Sprint 2 (during Week 10):

- 0.1 Environment Setup - Webpages:
  [0.5 hrs] {Karen, Kendrick} Due: Tuesday, June 4, 2019
  - Incorporate URL: View recent products used [0.5 hrs]

- 1. User Story: As a person with allergies, I want to be able to add a product by manually entering product information.
  [3 hrs][Epic A] {Kris, Anthony} Due: Tuesday, June 4, 2019
  - Include button/link to page/pop-up/dynamic form for admin request to add product [1 hr]
  - Create select menu for user to pick product in db from what they entered [1.5 hr]
  - Write SQL query to insert product data into corresponding table(s) [0.5 hr]

- 2. User Story: As a person with allergies, I want to capture information each time I use a product from my personal inventory about which product I used, when I used it and where I used it, so that I can collect data on my product usage.
  [5.5 hrs][Epic B] {John, Anthony} Due: Tuesday, June 4, 2019
  - Modify SQL queries to insert product usage date into corresponding table(s) [0.5 hrs]
  - Continue writing supporting client-side JavaScript functions for capturing user data, creating appropriate URLs, and creating XMLHttprequests to the server [2.5 hrs]
  - Update existing database queries and server-side Express routes [2.5 hrs]

- 3. User Story: As a person with allergies I want to record an allergic reaction: The allergy tracker app users can record an instance of an allergic reaction to a topical agent, like soap, lotion, makeup, etc.
  [5.5 hrs][Epic D] {John, Kendrick} Due: Tuesday, June 4, 2019
  - Modify SQL queries to insert reaction data into corresponding table(s) [0.5 hrs]
  - Continue writing supporting client-side JavaScript functions for capturing user data, creating appropriate URLs, and creating XMLHttprequests to the server [2.5 hrs]
  - Update existing database queries and server-side Express routes [2.5 hrs]

- 4. User Story: As a person with allergies, I want to quickly view a summary of recent products that I've used.
  [8 hrs] [Epic E] {Karen, Kendrick} Due: Tuesday, June 4, 2019
  - Create web page to show recent products used [6 hrs]
  - Write SQL query to retrieve needed information [0.5 hrs]
  - Test that the query behaves as expected [0.5hrs]
  - Incorporate query into web page [1 hr]

  -

- 5. User Story: As a person with allergies, I want to access a complete list of instances that I've recorded of allergic reactions I've had to a particular topical agent.
  [3 hrs][Epic E] {Kris, Kendrick} Due: Tuesday, June 4, 2019
  - Create web page to show all instances of allergic reactions, filtered by topical agent/product [2 hrs]
  - Write SQL query to retrieve allergic reaction data [1 hrs]

Implementation Plan

User Story	Task	Assigned To	Time Estimate	Due Date
0.1 Environment Setup - Webpages	Incorporate URL: View recent products used	Karen; Kendrick	0.5	6/4/2019
1. User Story: Add Product	Include button/link to page/pop-up/dynamic form for admin request to add product	Kris; Anthony	1	6/4/2019
1. User Story: Add Product	Create select menu for user to pick product in db from what they entered	Kris; Anthony	1.5	6/4/2019
1. User Story: Add Product	Write SQL query to insert product data into corresponding table(s)	Kris; Anthony	0.5	6/4/2019
2. User Story: Record Product Use	Modify SQL queries to insert product usage date into corresponding table(s)	John; Anthony	0.5	6/4/2019
2. User Story: Record Product Use	Continue writing supporting client-side JavaScript functions for capturing user data, creating appropriate URLs, and creating XMLHttprequests to the server	John; Anthony	2.5	6/4/2019
2. User Story: Record Product Use	Update existing database queries and server-side Express routes	John; Anthony	2.5	6/4/2019
3. User Story: Record Reaction	Modify SQL queries to insert reaction data into corresponding table(s)	John; Kendrick	0.5	6/4/2019
3. User Story: Record Reaction	Continue writing supporting client-side JavaScript functions for capturing user data, creating appropriate URLs, and creating XMLHttprequests to the server	John; Kendrick	2.5	6/4/2019
3. User Story: Record Reaction	Update existing database queries and server-side Express routes	John; Kendrick	2.5	6/4/2019
4. User Story: View product Summary	Create web page to show recent products used	Karen; Kendrick	6	6/4/2019
4. User Story: View product Summary	Write SQL query to retrieve needed information	Karen; Kendrick	0.5	6/4/2019
4. User Story: View product Summary	Test that the query behaves as expected	Karen; Kendrick	0.5	6/4/2019
4. User Story: View product Summary	Incorporate query into web page	Karen; Kendrick	1	6/4/2019
5. User Story: View reaction Summary	Create web page to show all instances of allergic reactions, filtered by topical agent/product	Kris; Kendrick	2	6/4/2019
5. User Story: View reaction Summary	Write SQL query to retrieve allergic reaction data	Kris; Kendrick	1	6/4/2019

Customer Availability

Our customer was willing and able to meet with us. We informed him about our unfinished tasks and he reviewed our progress on Sprint 1. He agreed that we should carry those unfinished tasks into Sprint 2. He also continues to be very generous with the team with his feedback and we appreciate it.

Contributions of team members

	Kendrick CHU	Anthony GIULIANO	John HASH	Kristopher HILL	Karen MCFARLAND
Pair programmed to implement tasks from Sprint 1	X	X	X	X	X
Provided updates on the progress of Sprint 1 tasks	X	X	X	X	X
Attributed Sprint 2 user stories (due date, tasks, dependencies, story points)	X	X	X	X	X
Created Sprint 2 Implementation plan	X	X	X	X	X
Helped write the HW submission	X	X	X	X	X
Burndown diagram					X
Stitch together document					X
Review final draft			X		X
Submit document					X