Homework 6

Assistant for Recycling and Waste Management to Reduce Recyclables in Landfills

Noah Beach, Kevin Christensen, Emmet Cooke, Jacob Leno, Caleb Scott
CS361 Group 18 - Spring 2018

1. Contributions Summary

The work was divided among our group for HW6 as follows: Noah Beach - User Story 1/User Story 14/UML Sequence Diagrams/Week 2 Development Schedule, Kevin Christensen - User Story 3/User Story 5/Integration Testing, Emmet Cooke - User Story 2/User Story 14/Refactoring Discussion, Jacob Leno - User Story 1/User Story 2/Burndown Diagram/Integration Testing, Caleb Scott - User Story 3/User Story 5.

2. Software Description

2.1 Week One User Story Work

Week One Software URL

The software for week one work can be used at the following URL: http://cs361group18.noahbeach.com

User Story 1 Work Summary

Noah Beach and Jacob Leno worked in a pair to complete this user story. The user story implementation took approximately 10 hours to complete, and was estimated to take 8 hours total. Writing/testing the SQL queries took approximately 2 hour. Designing the user interface took approximately 3 hours. Writing the javascript to tie the user interface to the database and logging service took approximately 5 hours, this time included debugging.

To validate the SQL queries, each query was tested in phpMyAdmin with the correct inputs to verify that the queries were correct and would return the correct data. The javascript executing these queries was tested by sending GET requests with valid and invalid query parameters to verify that the code would respond with the correct data.

Problems encountered included issues with javascript function overloading that resulted in bad data being sent to the rendered template. This overloading also resulted in an error condition causing node to crash under certain circumstances. Refactoring of the code to no longer overload a function fixed the issue.

The current status of the user story is implemented from a user interface perspective. The underlying hardware/hardware interface for the system is not implemented but the control interface is ready to accept it.

User Story 2 Work Summary

Jake Leno and Emmet Cooke worked on this user story. The estimated time to complete this task was 4 hours and it took slightly under 4 hours to complete. Three hours of time was spent of creating the

functions to gather the data about the machines for the relevant buttons and the final hour was used to create the user interface.

The unit tests for this user story revolve around the use of the button on the user interface and testing whether it works. If the machine is sorting and the button is clicked, the machine should stop, and if the machine is not sorting and the button is clicked, the machine should begin sorting. The other unit tests involved gathering the relevant data for these buttons from the database.

There were few problems encountered with this task, however there are quite a few database queries among the functions and those were fine tuned to access the relevant data before deployment. Development of the buttons on the user interface was implemented without issue.

The current status of the user story is that it is completely implemented.

User Story 3 Work Summary

Caleb Scott, Kevin Christensen, and Emmet Cooke worked in various pairs to complete this user story.

To validate the SQL queries, each query was tested in phpMyAdmin with the correct inputs to verify that the queries were correct and would return the correct data. The javascript executing these queries was tested by sending GET requests with valid and invalid query parameters to verify that the code would respond with the correct data.

One issue that emerged during implementation and testing was getting the “Mark Read” button to function properly. The problem turned out to be a misplaced form element which, once placed in the proper order, functioned fine.

Setting up database access was very quick (<1 hour), since it had already been established for other stories at this point. The method of communication with the simulated hardware was decided to occur through the database, so the communication with the hardware took the form of a number of SQL queries to collect the necessary data to display to the UI. This process took approximately 3 hours with debugging. The UI design took approximately 2 hours to implement.

User Story 5 Work Summary

Caleb Scott and Kevin Christensen worked on this user story and its tasks. There were a number of unit tests developed for this story.

To validate the SQL queries, each query was tested in phpMyAdmin with the correct inputs to verify that the queries were correct and would return the correct data. The javascript executing these queries was tested by sending GET requests with valid and invalid query parameters to verify that the code would respond with the correct data.

The biggest issues encountered when implementing this story all related to remembering how to program in Javascript and HTML. Neither language are strong suits for Caleb and Kevin, so there were some hiccups remembering how to structure the code to get everything to work properly.

Writing the SQL queries and the supporting Javascript took about 4 hours. Creating the user interface took about 6 hours. The SQL queries and Javascript backend are implemented

User Story 14 Work Summary

Noah Beach and Emmet Cooke worked in a pair to complete this user story. The user story implementation took approximately 6 hours to complete, and was estimated to take 6 hours total. Writing/testing the SQL queries took approximately 2 hours. Designing the user interface took approximately 2 hours. Writing the javascript to tie the user interface to the database and logging service took approximately 2 hours since there was heavy code reuse from user story one, this time included debugging.

To validate the SQL queries, each query was tested in phpMyAdmin with the correct inputs to verify that the queries were correct and would return the correct data. The javascript executing these queries was tested by sending GET requests with valid and invalid query parameters to verify that the code would respond with the correct data.

No noteworthy problems were encountered implementing this user interface as code reuse from user story one made a smooth implementation possible.

The current status of the user story is implemented from a user interface perspective. The underlying hardware/hardware interface for the system is not implemented but the control interface is ready to accept it.

2.2 Integration Testing

User Story 1 : Integration of Database + Conveyor Motion + UI

Test Case	Input Values	Driver Functions	Expected Outcomes	Observed Outcomes
Click belt start icon and belt start alarm is sent to the DB.	Click start button	machines.js router.get('/belt start/:mName/:idMachine/:idBelt'	Belt is started / Alarm is sent to the DB notifying the user	The alarm was sent to the database and displayed to the user correctly

Click belt stop icon and belt stop alarm is sent to the DB.	Click stop button	machines.js router.get('/beltstop/:mName/:idMachine/:idBelt',	Belt is stopped / Alarm is sent to the DB notifying the user	The alarm was sent to the database and displayed to the user correctly
Click Machine Controls Icon from Machine Information page	Click wrench icon	machines.js router.get('/control/:idMachine',	Page should navigate to the machine controls page	Page navigates to the machine controls page
Test sql query in getOneMachine function	Click wrench icon	Machines.js getOneMachine	The query should extract the requested machine from the database	The query gets the correct machine and returns it to the function

In implementing the belt controls and integrating it with the database a change to the structure of the alarms table was made. The id in the alarms table was set as being a foreign key that referenced the id for the machine table. This constraint was removed and the idMachine attribute of the alarms table was made a foreign key that referenced the id in the machine table. The name of the getMachines function was also changed. Originally there were two functions with the same name (one was overloaded) but javascript does not allow this. The second getMachines function was renamed to getOneMachine

User Story 2 : Integration of Database + Hardware IO + UI

Test Case	Input Values	Driver Functions	Expected Outcomes	Observed Outcomes
Click bin sorting start icon and bin sorting start alarm is sent to the DB.	Click start button	machines.js router.get('/binstart/:mName/:idMachine/:idBin'	Bin sorting is started / Alarm is sent to the DB notifying the user	The alarm was sent to the database and displayed to the user correctly

Click bin sorting stop icon and bin sorting stopped alarm is sent to the DB.	Click stop button	machines.js router.get('/binstop/:mName/:idMachine/:idBin'	Bin sorting is stopped / Alarm is sent to the DB notifying the user	The alarm was sent to the database and displayed to the user correctly
Test sql query in getMachines function	Click machines on the navbar	machines.js getMachines	The query should extract all the current machines in the database	The query gets the correct machine names from the database and returns them to the function
Test sql query in getMachineAlarms function	Click wrench icon	machines.js getMachineAlarms	The query should extract all the current alarms in the database	The query gets all the correct alarms and the corresponding information

For the sorting controls page two more bins were added as example bins to the user. These were added to provide a better visualization of a real implementation of this interface. A display of all the machine alarms was also added to the bottom of the page to make it easier to see what signals were being sent and what alarms were being added to the database.

User Story 5 - Integration of Database + Add/Remove/Update Data + UI:

Test Case	**Input Values**	**Driver Functions**	**Expected Outcomes**	**Observed Outcomes**
Input recycling type/bin and add to database	Text, Click Button	modType.js router.get('/addType')	The input should add the recycling type with the bin location to the database	The information was successfully added to the database

Input recycling type/bin and remove from database	Text, Click Button	modType.js router.get('/removeType')	The input should remove the recycling type with the bin location from the database	The information was successfully removed from the database
Input recycling type/bin and edit within database	Text, Click Button	modType.js router.get('/updateType')	The input should update the database record for the items input	The information was successfully updated in the database

For implementing the ability for the operators or technicians to make changes to the type of recyclables allowed by the system and which bins they are sorted into the group created a front end to allow the technician to input data which is connected to the database tables that hold the types of recyclables and their locations. This was done so the records can be quickly and easily maintained by any trained personnel instead of requiring a database administrator to make the modifications in the event of equipment malfunction or sudden influx of a specific recyclable.

User Story 14 - Integration of Database + Load Sensor Package + UI:

Test Case	Input Values	Driver Functions	Expected Outcomes	Observed Outcomes
Click machine name hyperlink from Machine Information link	Click hyperlink	Machines.js router.get('/:id Machine'	Page should navigate to individual machine information page	Page navigates to individual machine information page
Machine Bin Utilization populated with values	Page load	machines.js Machine.handlebars router.get('/:id Machine'	Machine Bin Utilization graph should be populated with values that represent the current bin level	On page load the graph is populated with values

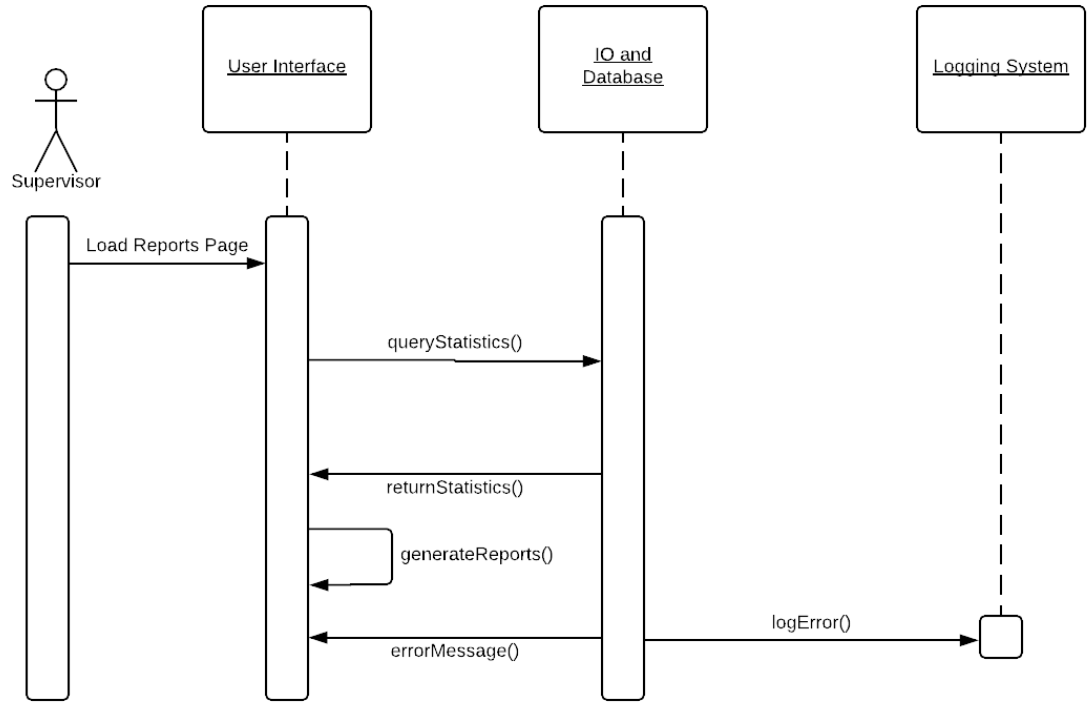
2.3 Burndown Diagram

The following burndown diagram represents the outstanding work vs time left to complete.

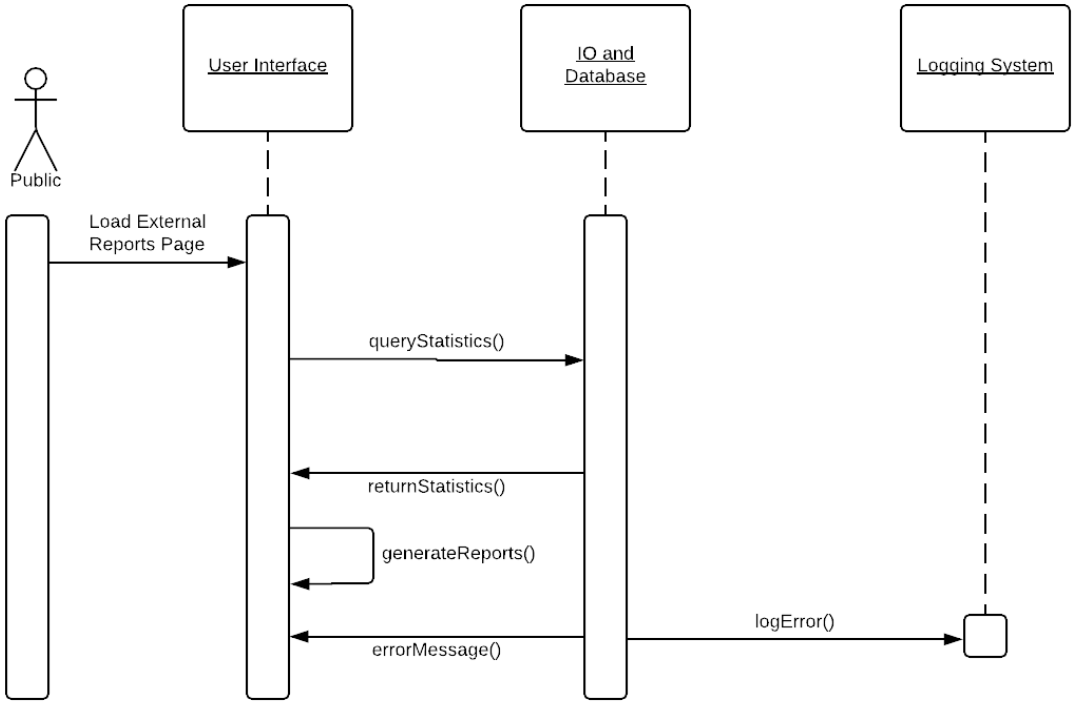
Burndown Diagram (Figure 2.3a)

2.3 Week Two User Story UML Sequence Diagrams

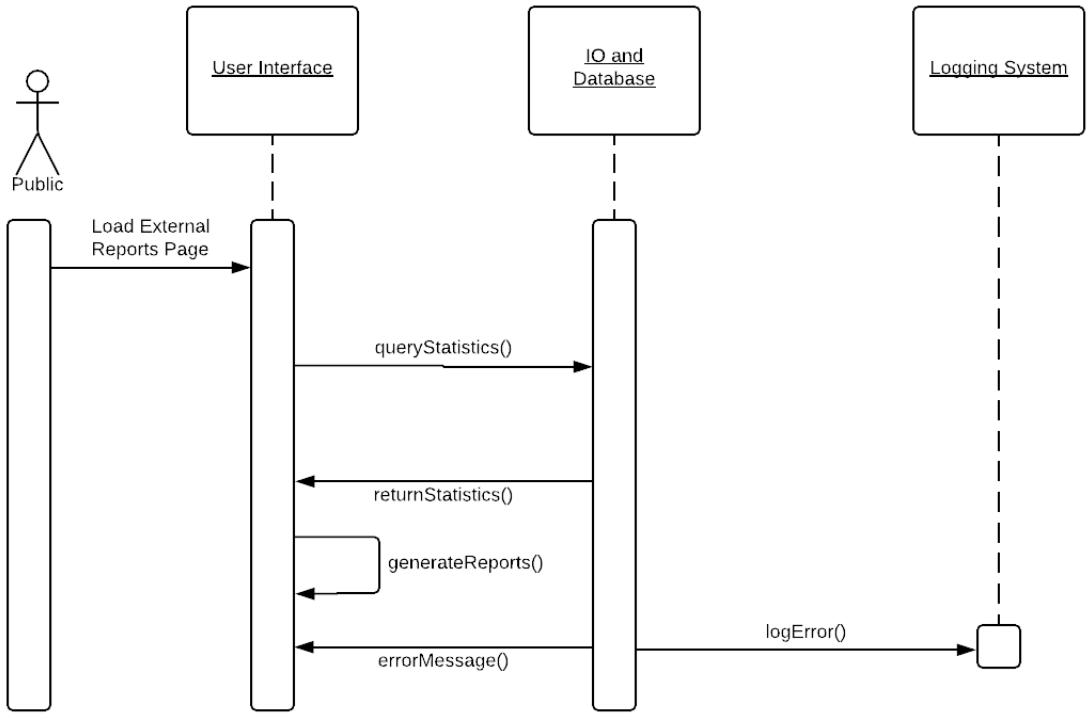
User Story 4 Sequence Diagram (Figure 2.2a)



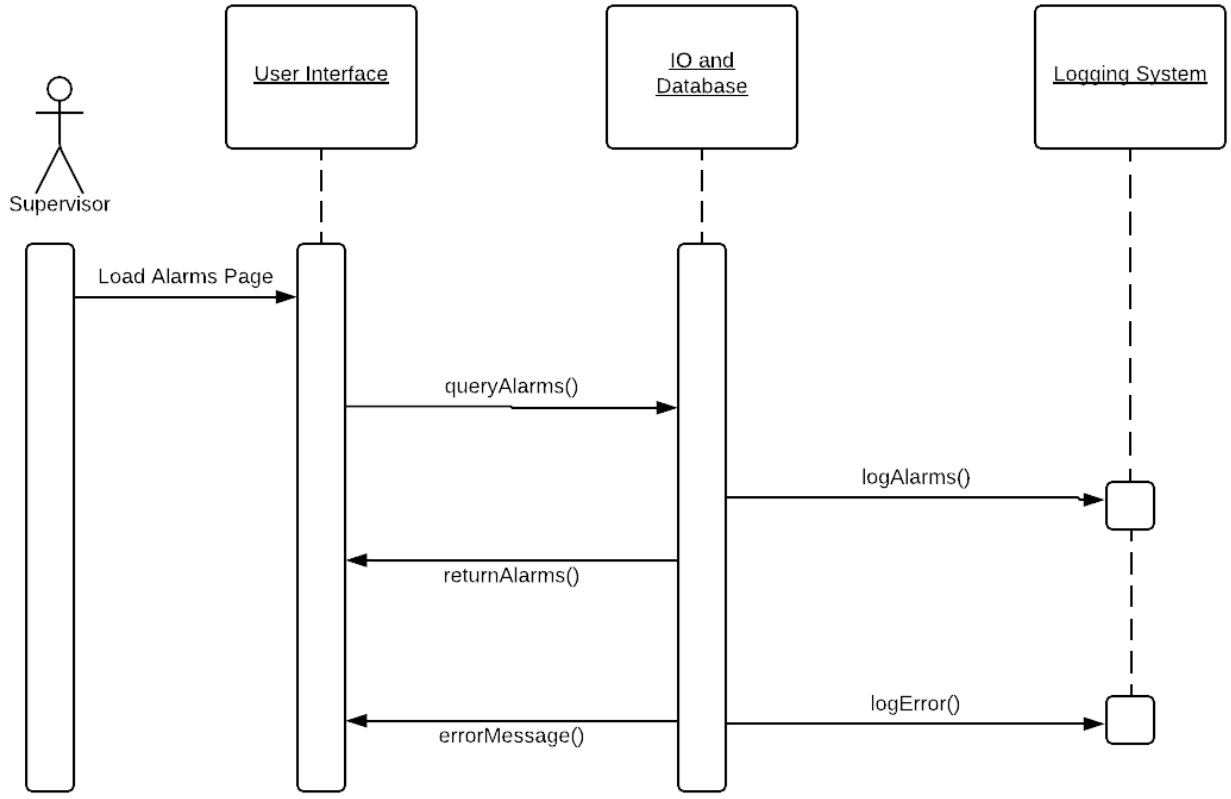
User Story 6 Sequence Diagram (Figure 2.2b)



User Story 7 Sequence Diagram (Figure 2.2c)



User Story 15 Sequence Diagram (Figure 2.2d)



3. Week Two Implementation Plan

3.1 User Story Week Two Work Allocation

User Story Four:

The estimated time for User Story Four is (10) hours and will be pair programmed by Caleb Scott and Noah Beach. The task will be broken into the following steps:

- Setup database communication (6/4)
- Design user interface (6/4)
- Generate Reports (6/5)
- Log interface object access or error message(s) (6/6)

User Story Six:

The estimated time for User Story Six is (10) hours and will be pair programmed by Emmet Cooke and Kevin Christensen. The task will be broken into the following steps:

- Setup database communication (6/4)
- Design user interface (6/5)
- Generate Reports (6/6)

- Log interface object access or error message(s) (6/7)

User Story Seven:

The estimated time for User Story Seven is (10) hours and will be pair programmed by Jacob Leno and Noah Beach. The task will be broken into the following steps:

- Setup database communication (6/4)
- Design user interface (6/4)
- Generate Reports (6/5)
- Log interface object access or error message(s) (6/6)

User Story Fifteen:

The estimated time for User Story Fifteen is (7) hours and will be pair programmed by Caleb Scott and Emmet Cooke. The task will be broken into the following steps:

- Setup database communication (6/4)
- Design user interface (6/4)
- Connect interface objects to simulated hardware (6/5)
- Log interface object access or error message(s) (6/6)

4. Customer Interaction

4.1 Customer Communication and Interaction

No customer interaction was attempted this week.