# Litterbug – A Software Approach to Littering

DAVID MEDNIKOV, Oregon State University
HITESH VARMA, Oregon State University
KEVIN ALLEN, Oregon State University
ALEXANDER YFRAIMOV, Oregon State University

## 1. USER STORIES DOWNLOAD

The developed user stories can be downloaded from:
https://drive.google.com/open?id=1si0rwqLioRIXelkYCfROO1B9LIeQFyjM.

The file to be downloaded has individual folders for each user story that we have developed so far. They can be viewed in any browser but we recommend using Google Chrome.

## 2. USER STORIES DUE TODAY

### 2.1 Register for the App (User Case 1.1)

- **Who worked on this feature?** – Alex and David worked together on the "user registration for the app" user story.
- **What do the unit tests do?** – We created a form which upon clicking a "register" button submits form data to a website which parses the data submitted. We created a unit test that allowed us to verify the contents of the submission were correct. This gives us a high level of confidence that once the database is set up and we change the recipient from the data parsing website to our database, the database will successfully receive the proper information.
- **What problems were encountered?** – This was a basic post request of text data so this portion went smoothly. We did not run into any problems.
- **How long did each task require?** – The coding for the html took about half an hour. The Javascript portion took approximately an hour to both create and verify that it was functioning as expected including checking that the data was displaying correctly on the website which was parsing the data.
- **What is the current status?** – The functionality and page have been developed and our unit test has passed. However, we cannot fully implement and test this story until the database is up and operational.
- **What is left to be completed?** – We will also need to create a unit test to verify the potential situation of a user attempting to create a username that already exists in the database. That request should be rejected and the user should be informed as to the reason. In order to test that properly we need our database operational. Aside from that we have some minor styling changes to make the application more aesthetically pleasing to the end user.

This work is supported by Michael Johnson of Oregon State University.

### 2.2 Report a User (User Case 1.6)

- **Who worked on this feature?** – Kevin and Hitesh worked together on the "report a user" user story.
- **What do the unit tests do?** – The unit test confirmed that the correct data was being sent when the button was clicked. When submitted the form send its data to a website that parsed the data contained in the request and displayed it on the screen so that its contents could be confirmed correct. Once implemented, the submission location will be changed to the system's database.
- **What problems were encountered?** – We did not encounter any problems; the story is very simple and we both had experience in this type of web application.
- **How long did each task require?** – The coding of a simple user page only took an hour. The coding of the popup took a little longer at 2 hours and the final JS function to submit the took about an hour to write and verify that is was sending the correct information.
- **What is the current status?** – Currently this unit has been developed and tested but not implemented. It will need a working database built before any meaningful implementation can take place.
- **What is left to be completed?** – The user info page needs better styling and to adjusted to be populated via handlebars once that server is setup to provide the necessary information.

### 2.3 Exchange Points for Currency to be Donated (User Case 1.10)

- **Who worked on this feature?** – David and Kevin worked together on the "exchange points for currency" user story.
- **What do the unit tests do?** – The relevant tests ensure that we are successfully getting the currency exchange rates from the API, that users cannot spend more points than they currently have when exchanging points for currency, that users can successfully exchange points for currency, that their points and money values are immediately updated, and that the changes in points and money that a user has get stored in the database.
- **What problems were encountered?** – We had some issues getting the API to work. Once we found an API that worked well and sent back data that was easy to work with, the rest of the work (parsing exchange rates, sanitizing inputs by ensuring that the user has the number of points they are trying to trade, updating points and currency that the user has, etc.) were pretty easy to implement.
- **How long did each task require?** – Writing the code to get the user's points was quite easy, as expected this would only take 0.5 hours. Since the database is still not in operation, the points are currently generated randomly within a certain range. The code that pulled exchange rates from the API and calculated exchange rates from points to money took a bit longer than expected, about 1.5 hours total. Then, creating the page where the user could select a currency and trade points for money in that currency took about 1.5 hours as well.
- **What is the current status?** – This feature has been developed and is currently in testing by the customer. Until the database is up and running, we will not be able to fully implement it.
- **What is left to be completed?** – Other than improving the styling of the user interface and porting the page to a mobile app, the functionality and business logic is all there. We are just waiting on the database.

## 3. UML DIAGRAMS DEVELOPED THIS WEEK

### 3.1 Submit a Photo (User Case 1.4)

The diagram proved to be useful in determining the order of developing code. The sequence laid out a plan of individual small tasks that could be combined into the function to submit the photo to the database.

### 3.2 Earning Badges/Medals (User Case 1.7)

The UML sequence diagram was useful in that it served to make sure the code we produced was carrying out the tasks it needed to in the proper order. We are reliant on the output of other user stories (particularly the one that involves verifying that the user who reported a particular area cleaned actually cleaned the area). Once our function receives confirmation that an area was truly cleaned then it can go about issuing medals/badges. The UML helped us see this process and structure our code accordingly.

### 3.3 Confirm a Cleaned Area (User Case 1.5)

The UML diagram that was designed for this user story was helpful because it helped us understand the components that would be involved in this functionality and what data will be sent between them. By having a solid understanding of the order of data transfer, we were able to develop the individual components in a logical order. For example, first we had to build the functionality to pull a map from the API and overlay it with colored areas. One benefit of developing this first was that it will be re-used for other features as well. After that, we just had to build the functionality for a simple form where the user either selects “cleaned” or “not cleaned” and post that back to the server, which then saves the feedback report to the database and reports users if necessary. If the user selects “cleaned”, they will also be prompted to enter a rating of 1-5 stars to also be sent back to the server.

### 3.4 Comment on a User’s Activity (User Case 1.9)

The UML sequence diagram helped us with breaking down this user story into smaller pieces rather than tackling it at once. We will need to pull a user’s activity from the database and show it to the different user. Then there will need to be an interface in the app where the user can enter a comment on a particular task and submit it to the server. Our server will then store that comment to the database for other users to see. Before creating the UML diagram, we did not have a detailed understanding of how this feature would be implemented, but splitting it up into modular pieces that transferred data between one another helped us plan out a logical solution.

### 3.5 Diagrams We Wish We Had

While we felt that our UML diagrams were helpful in developing many aspects of the system, we would benefit from creating more detailed diagrams for processes that are part of the user stories. For example, there are multiple steps required to get a photo from the user in the “submit a photo” user case, and will require a lower-level diagram to develop it efficiently. Similarly, the “confirm a cleaned area” user case will require a diagram for the multi-step process that both obtains the map from an API and uses database information to overlay colors on the map in real time. We felt that our diagrams for the badges/medals and commenting features were adequate.

# 4. BURNDOWN REPORT

## 4.1 Development Progress Data

| 3/Aug | 4/Aug | 5/Aug | 6/Aug | 7/Aug | 8/Aug | 9/Aug | 10/Aug | 11/Aug | 12/Aug |
|---|---|---|---|---|---|---|---|---|---|
| 336 | 336 | 336 | 336 | 336 | 336 | 336 | 336 | 336 | 336 |
| | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 10 | 41 | 100 | 150 | 175 | 225 | 250 | 300 | 384 |
| 284 | 308 | 344 | 436 | 486 | 511 | 561 | 586 | 636 | 720 |

## 4.2 Burndown Diagram

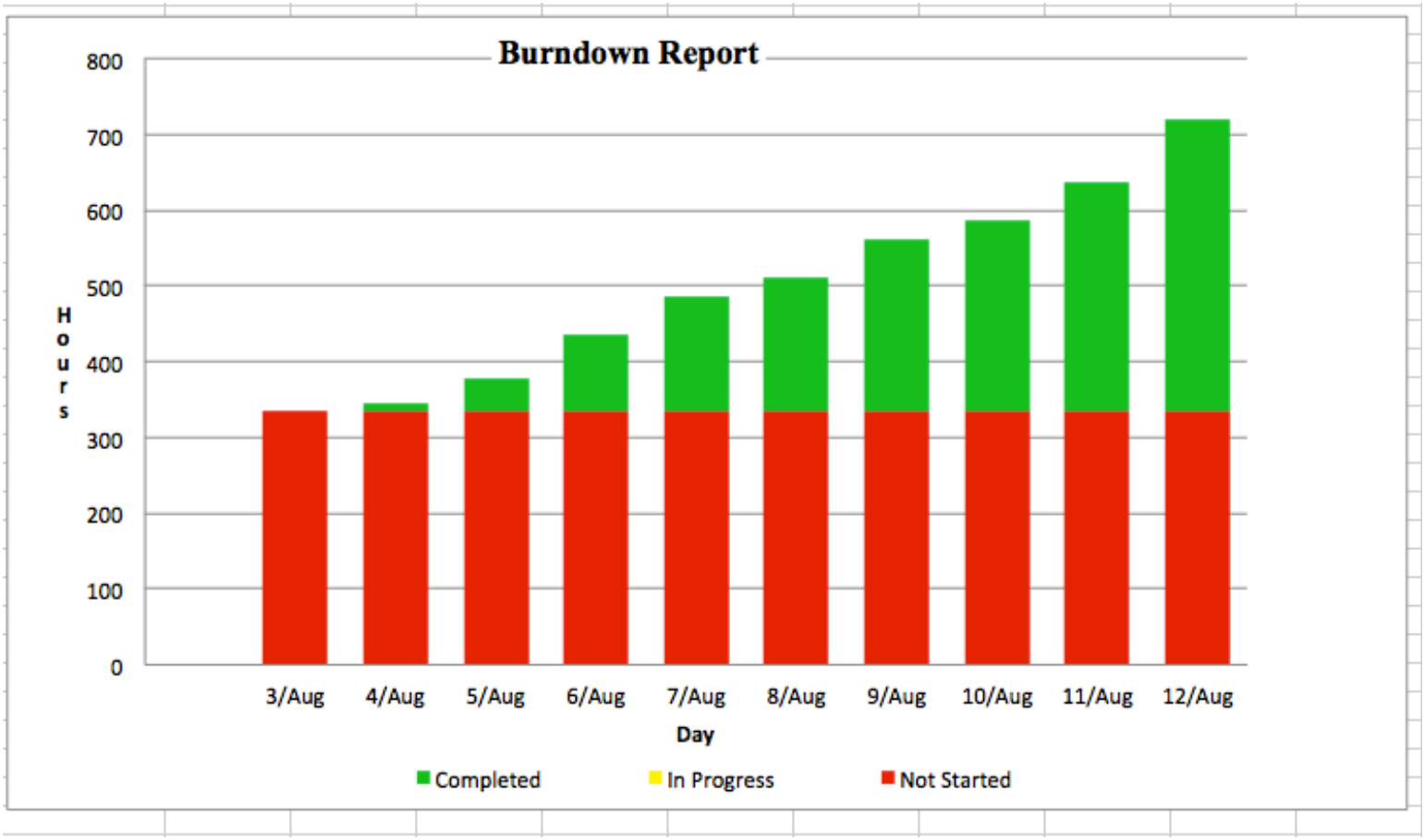


Fig. 1. Burndown Diagram

## 5. REFACTORING

### 5.1 Register for the App (User Case 1.1)

There was not much refactoring needed with the user registration story. For the sake of convenience (that is, our customer's convenience), we included all our Javascript inside script tags inside the .html files we submitted to him for review. In the finished product we will be stripping this Javascript out and placing it in separate .js files. The same is true of any CSS/styling code. We wanted to make things as simple as possible for our customer as they review the different components of the system we are constructing for them and we felt that this was the best way.

It also bears mentioning that given that our database is not yet online there are certain parts in the implementation that are acting as placeholders for the eventual code that will handle the communication with the database. For instance, in the "earn a badge" implementation there are some values hard coded which will be replaced with values obtained from the database. Once that new code comes in there is a potential for more refactoring to take place.

### 5.2 Report a User (User Case 1.6)

Not much refactoring was needed. Most of this code was similar to previous project. After completion and passing of the unit tests a few variables had their names changed to better reflect what they represented.

### 5.3 Exchange Points for Currency to be Donated (User Case 1.10)

Similar to the two user stories above, there was not much refactoring needed. We did modify the way that we parsed the response from the API, which removed a few lines of code. Other than that, the logic was pretty straightforward. Some optional further refactoring that could be done would be to validate point amounts in real-time so that a user is notified that they do not have the number of points that they are trying to exchange as soon as they enter the number. Currently, our system does not check to see if the user has enough points until after they click the "Exchange Points" button.

## 6. QUESTIONS FOR THE CUSTOMER

### 6.1 Register for the App (User Case 1.1)

Given that the user registration story is quite straightforward there was no need to consult with the customer for this. All the details of this were discussed in a prior meeting with the customer.

### 6.2 Report a User (User Case 1.6)

After creating the page we weren't sure how much user information should be displayed. One question for the client would be whether or not to allow users to share their general location with other users. This can be helpful with meeting up for group events but also poses some liability for sharing locations with potentially dangerous people. We have not yet had the opportunity to ask the customer this question as it came up after our meeting.

### 6.3 Exchange Points for Currency to be Donated (User Case 1.10)

The only question for the customer that comes to mind (other than UI-related questions) would be whether there are further currencies that he wants the page to support. Currently, the page supports exchanging points for dollars (US), euros, pounds (UK), and Canadian dollars. We perceived these to be the currencies used in countries where the app was launched, but if the customer wanted to add

other currencies (such as Kroner, Yen, Yuan, etc.) we can implement that quite easily. Similar to the user story above we have not yet had the opportunity to ask the customer about this one since it came to mind after our meeting.

## 7. INTEGRATIONS TESTS

### 7.1 Register for the App (User Case 1.1)

In regards to the registration user story, which was implemented the week prior, we will need to make sure it can post its values to a database. The integration testing will consist of us entering in a first name, last name, date of birth, username and password and clicking Register. At that point, we will check the entries in our users table in our SQL database making sure that all the entries posted correctly. As an additional component of the integration testing we will test that in the event a user attempts to create the same username as a previously registered user, the database will reject the request to add the duplicated entry and that the end user is appropriately alerted.

### 7.2 Report a User (User Case 1.6)

This page will need to be tested for integration after creation of the database. Once the database has been made and populated the page will need to pull information from the database to fill the user page. We also will need to ensure that the report type (abuse, hate speech, etc.) selected by the user will be successfully stored in our database. Other than that the user page is almost only concerned with its own content.

### 7.3 Exchange Points for Currency to be Donated (User Case 1.10)

We will need an integration test to ensure that currency results successfully come back from the API and can be parsed before being displayed to the user. Thankfully the response from the API is formatted in pretty simple JSON, but we will still need to ensure that our system is able to parse it without any issues and have a way to prevent an exception if there are any parsing errors. There shouldn't be any reason that a requested currency is absent from the JSON response, so we will not need to prepare for issues where a key is not available in an object. Either way, we will write code to handle that kind of issue.

Similar to the other user stories, we will also need to perform integration tests on our connection to the database to ensure that we are able to read, write, update, and delete from the database successfully.

## 8. SCHEDULE FOR WHAT TO COMPLETE NEXT WEEK

Our goal will still be to finish development of the "verify that a user cleaned an area", "earn badges/medals", "comment on a user's activity", and "submit a photo" user cases by the end of the week. Fortunately, most of our planning from last week will still work for us, but now we have specified who will be working with each person.

### 8.1 Submit a Photo (User Case 1.4)

Kevin and David will be implementing the ability to attach a photo when submitting a task that an area was cleaned. This will be done next week. Since the easiest part of this will be the UI, we will first develop the app page where users can attach a photo when submitting a task for an area they cleaned. After that, we will write the code to store the photo in the database and associate it with the task entry itself. This should still take 2 days, as expected in last week's assignment.

### 8.2 Earning Badges/Medals (User Case 1.7)

Alex and Kevin will be implementing the earning badge/medals functionality next week. This particular user story has a dependency on another one of the user stories in the sense that it relies heavily on the output of that other user story. That other user story, once implemented, will introduce the function of verifying that a user cleaned an area. This should still take 2 days, as expected in last week's assignment.

In order for the badge/medal functionality to execute properly it must receive confirmation from the verification function to ensure we don't issue rewards to users who may be making false reports about cleaning an area in an effort to acquire badges/medals. This means that we will need to work closely with the other pair who will be implementing that function so we know what we should be expecting as output from that verification function.

### 8.3 Confirm a Cleaned Area (User Case 1.5)

David and Hitesh will be implementing the ability to verify that a user did in fact clean an area next week. The fastest task to implement will be the notifications for users asking them to verify that a green area that they are located in or near that was recently marked as cleaned, was actually cleaned. Then we would write the code where the second user is able to leave some sort of feedback about the task and submit it to our server. Last, it should trigger the system to give or revoke badges, medals, or other incentives to the user that initially submitted the task. This should still take 2 days, as expected in last week's assignment.

### 8.4 Comment on a User's Activity (User Case 1.9)

Hitesh and Alex will be implementing the ability to comment on a user's activity next week. The first step will be to write code for the database where the user's list of tasks can be stored. This should be done first since it is easier to do than designing the profile page. Then the profile page with list of tasks should be designed, with space left to comment. The last step would be to write code to actually save user comments in the database. This needs to be done last because it is the most difficult part and we want to complete easier tasks first when possible. This should still take 2 days, as expected in last week's assignment.

## 9. MEETING WITH THE CUSTOMER

The customer was able to meet with us on the evening of Tuesday, August 7th. The customer was satisfied with our progress so far and is testing the features.

## 10. SUMMARY OF GROUP MEMBER CONTRIBUTIONS

- David Mednikov
    - Worked on “Exchange Points for Currency”, “Confirm a Cleaned Area”, and “Submit a Photo” user stories
    - Answered questions about “Exchange Points for Currency” user story
    - Created UML diagram for “Confirm a Cleaned Area” user story
    - Compiled the Google doc into the ACM format and created zip file containing features that have been developed so far
- Hitesh Varma
    - Worked on “Report a User”, “Comment on a User’s Activity”, and “Confirm a Cleaned Area” user stories
    - Created UML diagram for “Comment on a User’s Activity” user story
    - Created Burndown diagram
- Alexander Yfraimov
    - Worked on “Register for the App”, “Earn Badges/Medals”, and “Comment on a User’s Activity” user stories
    - Answered questions about “Register for the App” user story
    - Created UML diagram for “Earn Badges/Medals” user story
- Kevin Allen
    - Worked on “Report a User”, “Earn Badges/Medals”, and “Submit a Photo” user stories
    - Answered questions about “Report a User” user story
    - Created UML diagram for “Submit a Photo” user story