Food Trucks Fight Hunger

Group 24
Antonio Bermudez,bermudea@oregonstate.edu
Aaron Berns,bernsa@oregonstate.edu
Wenwen Dong,dongwen@oregonstate.edu
Steven Nowicki,nowickis@oregonstate.edu
Hunter Schallhorn,schallhh@oregonstate.edu

SUMMARY

Team uses Google Hangouts as main communication tool on daily basis. Team uses Google Docs to collaborate on the project.
Team uses Heroku as our server host, Postgresql as our database, Bootstrap as a styling library, Node JS for the backend, and Jade for templating.

What is the URL where your software can be tried out Describe any special instructions for using

Github link:
https://github.com/anberns/cs361group24
Software URL:
https://stormy-oasis-69997.herokuapp.com/

1. Brower wise, Firefox and Chrome should at least be ok.
2. Device wise, user should be able to visit using either desktop/laptop, tablet, or smartphone.
3. User should have wifi or cell phone data, with an internet connection.

Testing basic connectivity:

1. We tested mobile devices such as IPhone, Google Nexus 6P, Samsung A tablet.
2. We tested desktop computers such as Mac OS and Windows OS.

For each user story due today, describe if complete

1: **Story:** Post Donatable Food

1) Which pair(s) of teammates worked on that user story's tasks?
   Steve and Aaron
   Also:
   -Additional feedback regarding the database was provided by Hunter
   -Wenwen and Aaron added password validation

2) What do the relevant unit tests do?
   The test plan will focus on food vendor posting donation forms.
   It will test the input data validation, also the donation time should be greater than current time.

3) What problems, if any, did you encounter?

- Password problem. We want to make sure only user registered with validated password to be able to come to post donation.

4) How long did each task require?

Create table in database for available donation entities.
Modify table in database for available donation entities. - 1 hr
Create routes on the API server to get and post to a truck user's entities. - 1 hr
Create a web page to list and edit available food and details. - 4 hrs
Password validation -2 hrs

5) What is the current status (implemented? tested?)
Implemented and tested.

6) What is left to be completed?

Unit Tests		
ID	Test Cases	Pass / Fail
1.1	Description field left blank -Description: blank -Date: blank -Time: blank *blank field notification, form not submitted*	Pass
1.2	Date field left blank -Description: "Beans" -Date: blank -Time: blank *blank field notification, form not submitted*	Pass
1.3	Time field left blank -Description: "Beans" -Date: 1/1/18 -Time: blank *blank field notification, form not submitted*	Pass
1.4	Date entered has already passed -Description: "Beans" -Date: 1/1/18 -Time: 10:00 PM	Pass

	Date already passed notification, form not submitted	
1.5	All fields properly filled out -Description: “Beans” -Date: 1/1/18 -Time: 10:00 PM *Form submitted, alert generated with success message, clicking ‘ok’ takes user back to main food truck page*	Pass

2: **Story:** Confirm Food was Donated

1) Which pair(s) of teammates worked on that user story's tasks?
   Hunter and Aaron
2) What do the relevant unit tests do?

The test plan will focus on when a food truck user makes a donation, they will need to submit a form containing donation details to the recipient food bank user.
We will test if the vendor can send donation requests to nearby banks, if any donations have been posted.

3) What problems, if any, did you encounter?

We did not foresee that that a donation can be or not be requested multiple times to same shelter.

4) How long did each task require?
   a) Create a table in the database to store completed donations - 1h
   b) Create web page that allows food truck user to submit a donation to food bank user for approval. - 4h
   c) Create web page that notifies a bank of pending donations and allows them to approve a donation.
   d) Create a route in the API server to submit a donation for approval, adding the pending donation to the database - 2h
   e) Modify the get main menu route for the banks to check for pending donations
   f) Create a route in the API server to set a donation to “approved”

5) What is the current status (implemented? tested?)

Implemented and tested.

6) What is left to be completed?

Solve the problem and determine whether a donation request can or can not be sended multiple times to nearby shelters.

Unit Tests

ID	Test Cases	Pass / Fail
2.1	Food truck with no donations attempts to donate to a ‘nearby bank’ *No donations available notification*	Pass
2.2	First of multiple donations checked only -Donation 1: “Beans” : checked -Donation 2: “Potatoes” : unchecked -Donation 3: “Rice” : unchecked *Donation sent, waiting for confirmation message* *Checked donation appears on food bank page*	Pass
2.3	Second of multiple donations checked only -Donation 1: “Beans” : unchecked -Donation 2: “Potatoes” : checked -Donation 3: “Rice” : unchecked *Donation sent, waiting for confirmation message* *Checked donation appears on food bank page*	Pass
2.4	Third of multiple donations checked only -Donation 1: “Beans” : unchecked -Donation 2: “Potatoes” : unchecked -Donation 3: “Rice” : checked *Donation sent, waiting for confirmation message* *Checked donation appears on food bank page*	Pass
2.5	First and Second of multiple donations checked only -Donation 1: “Beans” : checked -Donation 2: “Potatoes” : checked -Donation 3: “Rice” : unchecked *Donation sent, waiting for confirmation message* *Checked donation appears on food bank page*	Pass
2.6	First and Third of multiple donations checked only -Donation 1: “Beans” : checked -Donation 2: “Potatoes” : unchecked -Donation 3: “Rice” : checked *Donation sent, waiting for confirmation message* *Checked donation appears on food bank page*	Pass
2.7	Second and Third of multiple donations checked only	Pass

	-Donation 1: “Beans” : unchecked -Donation 2: “Potatoes” : checked -Donation 3: “Rice” : checked *Donation sent, waiting for confirmation message* *Checked donation appears on food bank page*	
2.8	All of multiple donations checked -Donation 1: “Beans” : checked -Donation 2: “Potatoes” : checked -Donation 3: “Rice” : checked *Donation sent, waiting for confirmation message* *Checked donation appears on food bank page*	Pass
2.9	Single donation added to bank that already has donations waiting to be confirmed -Bank has: “beans”, “potatoes”, “rice” -Donation 1: “ground beef”: checked *Donation sent, waiting for confirmation message* *Checked donation appears on food bank page in addition to prior donations waiting for confirmation*	Pass
2.10	Two additional donations added to bank that already has donations waiting to be confirmed -Bank has: “beans”, “potatoes”, “rice” -Donation 1: “ground beef” : checked -Donation 2: “pork chops” : checked *Donation sent, waiting for confirmation message* *Checked donation appears on food bank page in addition to prior donations waiting for confirmation*	Pass
2.11	Three additional donations added to bank that already has donations waiting to be confirmed -Bank has: “ground beef”, “pork chops” -Donation 1: “beans” : checked -Donation 2: “potatoes” : checked -Donation 3: “rice” : checked *Donation sent, waiting for confirmation message* *Checked donation appears on food bank page in addition to prior donations waiting for confirmation*	Pass
2.12	Attempted addition of donation already sent to bank for confirmation	Fail

	-Bank has: “ground beef”, “pork chops” -Donation 1: “ground beef” : checked -Donation 2: “pork chops” : checked *Duplicate donation message, form not submitted*	*Form submitted but duplicate are not added to database and therefore not shown on bank page*

3: **Story:** Browse Donation History

1) Which pair(s) of teammates worked on that user story's tasks?
   Steve and Antonio
2) What do the relevant unit tests do?
   The test plan focuses on browsing all of a user’s donations that have been made under different conditions.

3) What problems, if any, did you encounter?
   Formatting the date in a proper display format. This required parsing it server-side.

4) How long did each task require?
   Create a web page, using Node JS and Jade, that requests and displays history of most recent donations. - 4 hrs
   Create routes to query and return donation history, using Node JS. - 1 hr

5) What is the current status (implemented? tested?)
   Implemented and tested.

6) What is left to be completed?
   Though test 3.3 passes with respect to this story, it reveals that a user can post ‘duplicate’ donations - although the back end does not consider them duplicate because of each donation’s unique id. In a future iteration, it should be decided whether the user should be allowed to make duplicate entries with the Post Donation feature, or whether the application should prevent them from doing so.

Unit Tests		
ID	Test Cases	Pass / Fail
3.1	Description: Browse Donations when no donations have been made by the user.	Pass

	Donation page appears, but no donations are listed	
3.2	Description: Browse Donations when 1 donation has been made by the user. *Donation page appears,donation is listed*	Pass
3.3	Description: Browse Donations when 2 donations with same description, date, and time have been made by the user. *Donation page appears, both donations are listed.*	Pass

4: **Story:** View details of a donation

1) Which pair(s) of teammates worked on that user story's tasks?
   Steve and Wenwen
2) What do the relevant unit tests do?
   When a user is browsing donations, they should be able to click the Details link next to each donation. A new page should appear that lists the details of that donation.
3) What problems, if any, did you encounter?
   Figuring out how to pass a specific donation id was tricky, but was resolved by use of a query string in the url.
4) How long did each task require?
   Add functionality to donation history webpage that allows for a donation to be selected and submitted to server. The server should return donation details, which should be displayed in the web page UI. - 3 hrs
   Create route to query and return specific donation information, using Node JS and Jade. - 1 hr

5) What is the current status (implemented? tested?)
   Implemented and tested.

6) What is left to be completed?
   In a future iteration, the failure of test 4.3 should be considered. A fix would require supporting sessions across browser tabs.

Unit Tests		
ID	Test Cases	Pass / Fail
4.1	Description: View Details of a basic donation.	Pass

	Donation Details page appears, listing ID, DESCRIPTION, DATE, and TIME.	
4.2	Description: View Details of a donation with a long description (398 characters) *Donation Details page appears, listing ID, DESCRIPTION, DATE, and TIME. Long DESCRIPTION should be properly displayed on page.*	Pass
4.3	Description: Attempt to open Details in a new browser tab with right-click. *Donation Details page appears, listing ID, DESCRIPTION, DATE, and TIME.*	Fail

5: **Story:** Generate Donation Report

1) Which pair(s) of teammates worked on that user story's tasks?
   Wenwen and Hunter
2) What do the relevant unit tests do?
   When a food truck user enters a date range, he should be able to receive a list of donations they made during that date range. A new page should appear that lists the details of that donation.
3) What problems, if any, did you encounter?
   The report does not indicate if the donation has or has not been accepted by food banks. After implementing the 'Confirm Food is donated', there will require a future iteration to include more information from the bank side.
4) How long did each task require?
   - Add functionality to donation history web page that accepts range criteria, submits range data within query to database, and displays returned results in the web page UI. -3h
   - Create route to query and return applicable donation information, using Node JS and Jade. -1h
5) What is the current status (implemented? tested?)
   Implemented and tested.
6) What is left to be completed?
   In a future iteration, the failure of test 4.3 should be considered. A fix would require supporting with input device connection, mail server API, etc.

Unit Tests

ID	Test Cases	Pass / Fail
5.1	Description: After login, vendor enter 2017-11-27 to 2017-11-29. *Donation Details page appears, listing ID, DESCRIPTION, DATE, and TIME.*	Pass
5.2	Description: After get the report, vendor can view each donation details made within 2017-11-27 to 2017-11-29. *Donation Details page appears, listing ID, DESCRIPTION, DATE, and TIME. Long DESCRIPTION should be properly displayed on page.*	Pass
5.3	Description: Attempt to print out the report or download or send to email. *Donation Report allows print, download or send to email.*	Fail

For each spike and UML sequence diagram that you developed this week, answer the following

1 Story: Post Donatable Food

Was the spike or diagram useful? Why or why not?

The diagram for this Story was useful, as there were multiple steps leading up to a database submission. Interestingly, the diagram also stated that we need password validation for posting to the database, which is something we had forgotten during Week 6.

Were there any diagrams that you wish that you had? Why or why not?

It would have been helpful to have a fully formed diagram of the donations table and an ERD that conveyed its final state. We did have a diagram of our original donations table in an ERD, but as development occurred the table changed. Since it was a relatively simple table, it was not difficult to work without an updated ERD, but if this project were to continue we would need to create one.

2 Story: Confirm Food was Donated

Was the spike or diagram useful? Why or why not?

Yes. The diagram leads us to design another schema to implement confirmation function. The diagram also indicate how the data flow works.

Were there any diagrams that you wish that you had? Why or why not?

We did not foresee the difference in implementation required between confirming a single donation versus confirming multiple donations at once. It would have been very helpful to have separate diagrams for each scenario, or a single diagram that accounted for both scenarios.

3 Story: Browse Donation History

Was the spike or diagram useful? Why or why not?
Yes, the diagram mapped out exactly what needed to happen for this feature to work. Additionally, this diagram also reminded us that a user needed to be logged in with a password in order to use this feature.

Were there any diagrams that you wish that you had? Why or why not?
No, the diagram that was provided was very useful and contained all of the information needed to implement the story.

4 Story: View details of a donation

Was the spike or diagram useful? Why or why not?
Yes. This was the same diagram used for the Browse Donation History story, although just a specific portion of it was needed for this story. What was most useful about the diagram is that it emphasized that a specific donation id needed to be submitted to the database, in order to retrieve the details about that donation. With that goal in mind, it was just a matter of implementation.

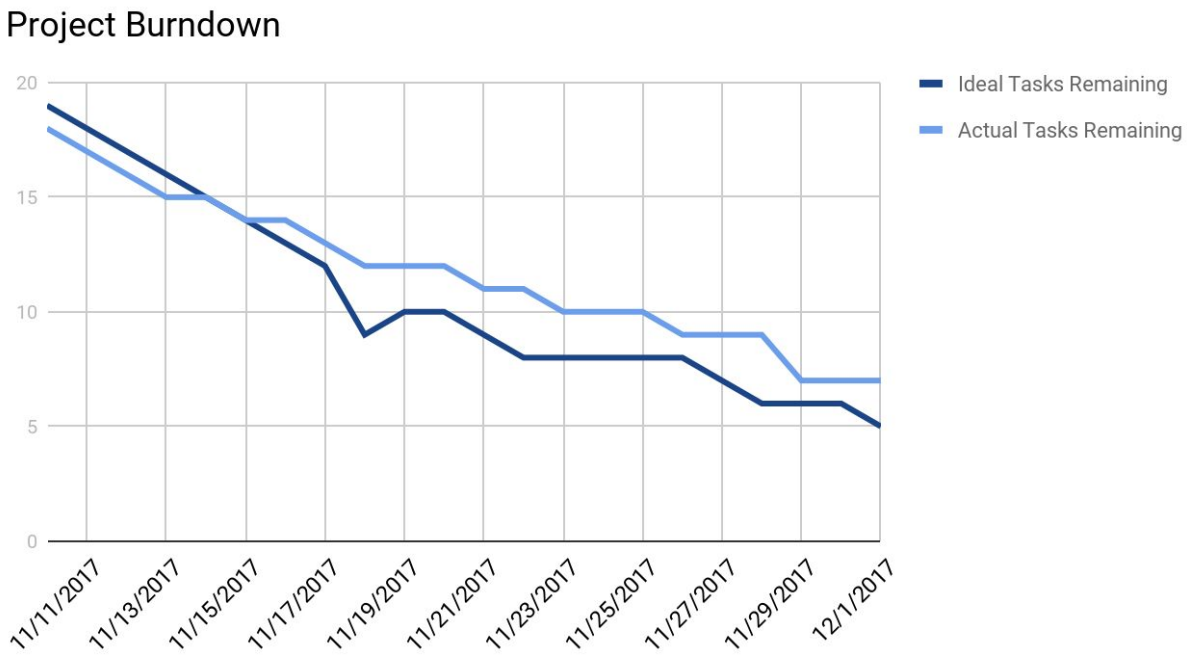
Were there any diagrams that you wish that you had? Why or why not?
We did not foresee that a user might want to View donation details in another browser tab. To implement this functionality, it would be useful to have a separate diagram.

5 Story: Generation donation report

Was the spike or diagram useful? Why or why not?
For implementing donation report, I used a lot of the ERD. This is much more visible compared to reading through the documents.

Were there any diagrams that you wish that you had? Why or why not?
The functionality is not implemented yet, but diagrams detailing how to print, download or email the donation report would be useful in future iterations. Users might eventually need to use this information for tax purposes, but we did not get to this yet.

Provide a Burndown diagram



Briefly describe any refactoring that you did

1. After last week's iteration, our grader pointed out that our user accounts don't require a password to log in. While this was never explicitly requested by the customer, we thought it would be good to require a password for posting to the database, so we added passwords to user accounts.
2. Posting a donatable item initially used the datetime-local input field type. Though this was supported by the Postgres database, it was discovered that Firefox does not currently support this input type. As a result, we changed to two input fields: date and time. Subsequently, the views for browsing donatable items and viewing donatable items had to be changed.
3. The donations table did not initially have any foreign key to reference the vendor who made the donation. As a result, vendor_id was added to the donations table.
4. We added a new table in the database for completed donations that have been confirmed by food banks.

If you had to ask the customer any questions, indicate what those questions were and what the customer's response was

We did not have any questions for the customer this week. In our talks with her last week and the weeks before, she answered all questions that we needed going into this week.

Briefly describe all integration tests that you did on the system, the test results, and any changes that you made (or will make) to the system as a result.

1: When testing vendor's donation submit, we added password to protect user's information.
To fix this, we added password attribute in both vendor and shelter's DB design.

2: When testing date types, we find that Firefox does not support the datetime-locale input type.
To fix this, we updated the DB input type and views for related pages.

3: When implementing donation completed by vendor, we do not have a DB schema to reflect the data flow.
To fix this, we designed a completed donation table schema.

4: When getting the current available donations seeking confirmation for the food banks, we realized we needed to display a description of a donation.
To fix this, we ensured the donation description was added to the completed_doantions database table and amended the database query and route so that donation descriptions would be passed to bank's main menu to be displayed.

Briefly summarize the contribution of each of your team members.

All members contributed.

Pair programming	All team members
Spike and refactoring	All team members
Testing	All team members