# 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. All members contributed.

| | |
|---|---|
| Database setup, server setup | Aaron, Antonio, Wenwen |
| Pair programming | Aaron, Hunter, Antonio, Steve, Wenwen |
| Testing plan | Steve, Hunter, Wenwen |
| Burndown Diagram and refactoring | Steve and Antonio |
| Next week plan | All group members |

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

https://stormy-oasis-69997.herokuapp.com/

1. Brower wise, Firefox, Chrome and Safari 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 cellphone 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 #1:** Register a Food Truck

1. Which pair(s) of teammates worked on that user story's tasks?
   Aaron Berns and Steve Nowicki

2. What do the relevant unit tests do?
   The unit test will focus on the vendor registration for a user account. (Unit test plan see below #5)
   It will test the vendor's input data. It will also test the location function.

3. What problems, if any, did you encounter?
   We discovered that we need to provide error handling. We need to make all values in the form required, or the database will not accept the entry.

4. How long did each task require?
   There was some upfront work that needed to occur before this specific feature and most other features could be implemented.
   This included:
   1) Set up a Postgres database - 1 hour
   2) Set up an API server to access the database - 5 hours
   3) Using NodeJS and Jade, create a page with forms to collect user's input, and store it in database. - 4 hours
   4) Using Bootstrap to style webpage. -2 hours
   5) Do a unit test for user input. - 1hour

This up front work was completed by Aaron, Wenwen, and Antonio

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

See test plan below:

| Unit Test Results | | |
|---|---|---|
| ID | Test Cases | Pass/Fail |
| 1.1 | Test Case<br>Register with user account<br>Name: null<br>Email: null<br>Phone:null<br>Radius: null<br>Location: null<br>*An error will be displayed for the null input.* | Pass |
| 1.2 | Test Case<br>Register with user account<br>Name: Anna<br>Email: anna@live.com<br>Phone:1-321-993-0111<br>Radius: 20<br>Location: Times Square, New York, NY, United States<br>*An error will be displayed for the wrong phone number format.* | Pass |
| 1.3 | Test Case<br>Register with user account<br>Name: Chris<br>Email: anna@live.com<br>Phone: 4426768917<br>Radius: 21<br>Location: [key in zip code 95014 should have a list of address to pick]<br>*List of current address should be optional to choose from.* | Pass |
| 1.4 | Test Case<br>Register with user account<br>Name: Linda<br>Email: linda@123.com<br>Phone:3479901234<br>Radius: null<br>Location: Goodyear, AZ, United States<br>*An error will be displayed for the radium is null* | Pass |

| | | |
|---|---|---|
| 1.5 | Test Case Register with user account<br>Name: Linda<br>Email: linda@123.com<br>Phone:null<br>Radius: 50<br>Location: Apple Campus, Cupertino, CA, United States<br><br>*An error will be displayed for the phone input is null* | Pass |
| 1.6 | Test Case<br>Register with user account<br>Name: Anna Jean<br>Email: anna@live.com<br>Phone: 321-993-0113<br>Radius: 22<br>Location: Times Square, New York, NY, United States<br><br>*An error will be displayed for same email address and same phone number.* | Pass |
| 1.7 | Test Case<br>Register with user account<br>Name: George<br>Email: geo@live.com<br>Phone:8791274111<br>Radius: -11<br>Location: Apple Campus, Cupertino, CA, United States<br><br>*After first submission attempt, an error will be displayed under the form input stating that a positive number is required, input border will be red and submit button inactive until valid number is entered, at which point border turns green.* | Pass |
| 1.8 | Test Case<br>Register with user account<br>Name: George<br>Email: geo@live.com<br>Phone:87912742<br>Radius: 50<br>Location: Apple Campus, Cupertino, CA, United States<br><br>*An activation email should be sent from server.* | Fail |

| | | |
|---|---|---|
| 1.9 | Test Case<br>Login with a registered account name<br>Login<br>Name: George<br>*The user should be directed to their main menu* | Pass |
| 1.10 | Test Case<br>Login with an unregistered account name<br>Login<br>Name: Unregistered User<br>*The user should be shown an error on the login page* | Pass |
| 1.11 | Test Case<br>Register with account and re-register with identical information<br>Name: Same Vendor<br>Email: samevendor@gmail.com<br>Phone:5555555555<br>Radius: 20<br>Location:11218<br>*An error will be displayed stating that a vendor has already registered with that information* | Pass |
| 1.13 | Test Case<br>Register new account with negative value in delivery radius twice then positive value<br>Name: A Vendor<br>Email: avendor@gmail.com<br>Phone:5555555555<br>Radius: -20, -60, 60<br>Location:11218<br>*After first submission attempt, an error will be displayed under the form input stating that a positive value is required, input border will be red and submit button inactive until valid number is entered, at which point border turns green.* | Pass |
| 1.14 | Test Case | Pass |

| | | |
|---|---|---|
| | Register new account with phone number other than 10 digits with dashes<br><br>Name: Another Vendor<br>Email: anothervendor@gmail.com<br>Phone: abc@#$1234, 123-456-78910, 555-555-5555<br><br>*After first submission attempt, an error will be displayed under the form input stating that a valid phone number is required, input border will be red and submit button inactive until valid number is entered, at which point border turns green.* | |

*Table 1: Unit test plan for vendor registration*

6. What is left to be completed?

At another iteration after the testing we figured out how to validate most of the input. We are able to validate below:

User input validations:

1- define the same vendor as the one who has same username and same phone number. Same vendor should not be able to register twice in the system.

2-define the vendor delivery radius as positive miles. Any negative radius should not be able to stored in the system.

3- Define the vendor's phone number should be of a length of 10 digits. Any symbol should not be allowed into the system.

We will have to have following be fixed in later stages, which we will discuss with customer.

User file upload:

1- vendor should be able to upload its business licenses either in .pdf or image type.

2- server should be able to store the file and allow verification with IRS server.

User input validations:

- Update of the vendor's activation status. Every vendor should be able to receive an activation email. Only after activated, the user's status will be updated.

**2 - Story #2:** Register a Food Bank

1 Which pair(s) of teammates worked on that user story's tasks?

Aaron Berns and Steve Nowicki

2 What do the relevant unit tests do?

The unit tests will focus on the bank registration for a user account. (Unit test plan see below #5)

It will test the bank's input data to see if it is properly validated. It will also test the location function.

3 What problems, if any, did you encounter?
We discovered that we need to provide error handling. We need to make all values in the form required, or the database will not accept the entry.

4. How long did each task require?

This included:

1- Set up a Postgres database  - 1 hour

2 - Using NodeJS and Jade, create a page with forms to collect user's input, and store it into server.  - 4 hours

3 - Using Bootstrap to style webpage. -1 hours

4 - Do a unit test for user input. - 1hour

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

| Unit Test Results | | |
|---|---|---|
| ID | Test Cases | Pass/Fail |
| 2.1 | Test Case<br>Register with bank account<br>Name: null<br>Email: null<br>Phone:null<br>Open at, close at: null<br>Location: null<br>*An error will be displayed for the null input* | Pass |
| 2.2 | Test Case<br>Register with user account<br>Name: Anna<br>Email: anna@live.com<br>Phone:12ew09235<br>Open at, close at: 11:00AM 12:30AM<br>Location: Times Square, New York, NY, United States<br>*An error will be displayed for the wrong phone number format.* | Pass |
| 2.3 | Test Case<br>Register with user account<br>Name: Jim<br>Email: jim@live.com<br>Phone: 230948750 | Pass |

| | | |
|---|---|---|
| | Open at, close at:08:00-- 8:00 PM<br>Location: Los Altos, CA, United States<br>*An error will be displayed for the incomplete time format.* | |
| 2.4 | Test Case<br>Register with user account<br>Name: Happy<br>Email: happy<br>Phone:3479901234<br>Open at, close at: 10:00AM 5:30PM<br>Location: Goodyear, AZ, United States<br>An error will be displayed for the email format is wrong. | Pass |
| 2.5 | Test Case<br>Register with user account<br>Name: Jim<br>Email: jim@live.com<br>Phone: 2309487501<br>Open at, close at:08:00-- 8:00 PM<br>Location: Los Altos, CA, United States<br>*An error will be displayed for the same email address and phone number.* | Pass |
| 2.6 | Test Case<br>Register with user account<br>Name: George<br>Email: geo@live.com<br>Phone:87912741<br>Open at, close at:10:00AM 8:00PM<br>Location:[key in 94088 to get a list of nearby poi]<br>A list of nearby POI from Google should be shown in a dropdown list. | Pass |
| 2.7 | Test Case<br>Login<br>Name: George<br>*The user should be directed to their main menu* | Pass |
| 2.8 | Test Case<br>Login<br>Name: Unregistered User<br>*The user should be shown an error on the login page* | Pass |

| | | |
|---|---|---|
| 2.9 | Test Case<br>Register with account and re-register with identical information<br>Name: Same Bank<br>Email: samebank@gmail.com<br>Phone:5555555555<br>Open At: 9:00<br>Close At: 8:00<br>Location:11218<br>*An error will be displayed stating that a vendor has already registered with that information* | Pass |
| 2.10 | Test Case<br>Register new account with phone number other than 10 digits with dashes<br>Name: Another Bank<br>Email: anotherbank@gmail.com<br>Phone: abc@#$1234, 123-456-78910, 555-555-5555<br>*After first submission attempt, an error will be displayed under the form input stating that a valid phone number is required, input border will be red and submit button inactive until valid number is entered, at which point border turns green.* | Pass |

*Table 2: Unit test plan for bank registration*

6: What is left to be completed?
At another iteration after the testing we figured out how to validate most of the input. We are able to validate below:

<u>User input validations:</u>

1 - Define the same bank user as the one who has same username and same phone number. Same bank user should not be able to register twice in the system.

2 - Define the bank’s phone number should be of a length of 10 digits. Any symbol should not be allowed into the system.

We might have to discuss with customer about following validations:

<u>User file upload:</u>

1- Banks should upload their business licenses either in .pdf or image type.

2- server should be able to store the file and allow verification with IRS server.

User input validations:

- Update of the bank's activation status. Every bank should be able to receive an activation email. Only after activated, the user's status will be updated.

**3 - Story #3:** Look for Food Banks Nearby

1. Which pair(s) of teammates worked on that user story's tasks?

Antonio Bermudez and Hunter Schallhorn

2. What do the relevant unit tests do?

The unit tests will test the displaying of nearby food banks (see test plan below)

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

We discovered we needed to handle errors for invalid registrations done in early testing. This included errors caused by banks without locations, vendors without locations, and vendors without a registered max travel distance

4. How long did each task require?

1 - Create a webpage that takes location information and submits it, displaying results of nearby food banks. - 4h
2 - Create route on the API server that queries database for banks with a location within a specified range from the user's current location. - 3h
3 - Testing. - 15m

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

Implemented and tested

| Unit Test Results | | |
|---|---|---|
| ID | Test Cases | Pass/Fail |
| 3.1 | Test Case<br>Vendor has a location and search radius<br>location: 1 Infinite Loop, Cupertino, CA 95014<br>max_dis: 25<br>*Banks within a 25 mile radius are shown* | Pass |
| 3.2 | Test Case<br>Vendor has an invalid location<br>location: null | Fail |

| | | |
|---|---|---|
| | *An error should be shown* | |
| 3.3 | Test Case<br>Vendor has an invalid max_dis<br>max_dis: null<br><br>*An error should be shown* | Fail |
| 3.4 | Test Case<br>A bank is within the max_dis a vendor<br>Vendor location: Infinite Loop, Cupertino, CA, United States<br>Vendor max_dis: 25<br>Bank location: Apple Campus, Cupertino, CA, United States<br><br>*The bank should be listed* | Pass |
| 3.5 | Test Case<br>A bank is outside of the max_dis of a vendor<br>Vendor location:<br>Vendor max_dis:<br>Bank location:<br><br>*The bank should not be listed* | Pass |

6. What is left to be completed?
Handle invalid vendor distance and location by adding an error message or cleaning up the database

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

**Stories 1 & 2**

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

Spike or diagram is useful before you sit down and program. Spike especially, it helps to identify the small problems that you can solve at one time.

At the time of programming, we didn’t use much of the sequence diagram. We mostly referred to the sql we created in the database. This is probably due to the fact that it is a relatively simple feature to set up a home page, and two registration pages.

At the time of testing, we do refer to the diagram. The design can tell where this problem belongs to, what other testing value could probably happen, and what else is missing from our original goal. After comparing the diagram to what we designed, we were able to identify some functions that were supposed to be implemented but were yet to be included.

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

When we were trying to carry out some unit tests, we figured that for the registration page, we would have to do a spike on how to upload a file into Postgresql.

**Story 3**

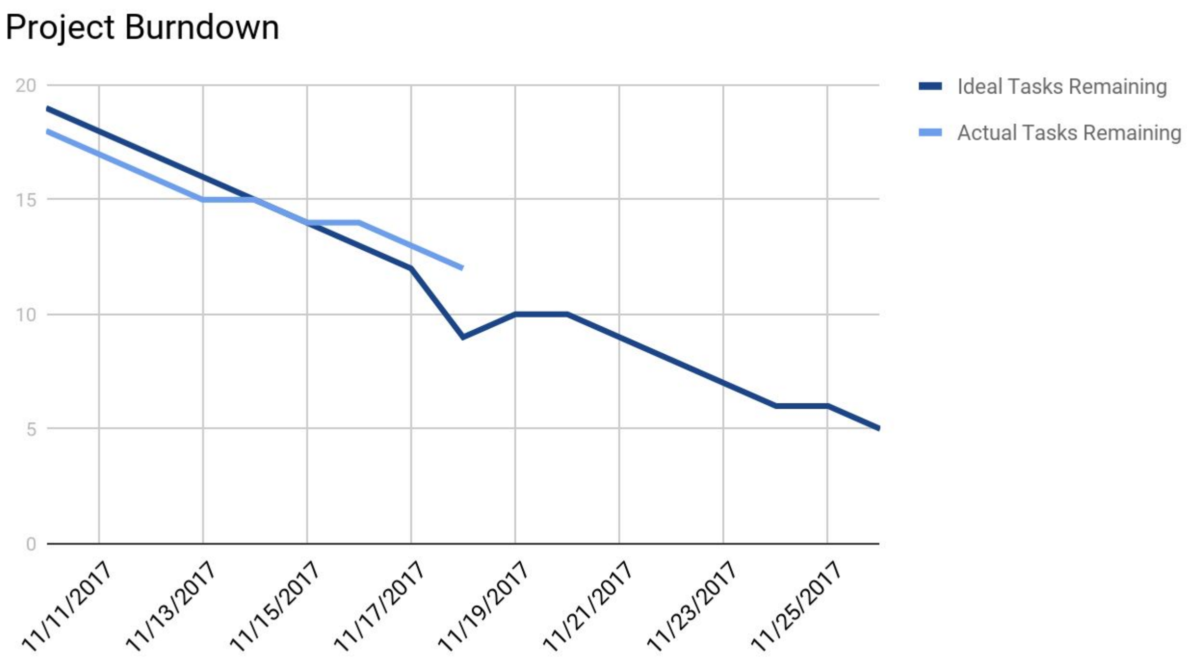
*Was the spike or diagram useful? Why or why not?*

The spike for the server side setup of nearby banks was a great help in implementing the response step by step. Each smaller step was a natural progression towards the final task and helped keep us on track of what to do next.

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

We used the Google Maps API to verify the distance between vendor and food bank to filter the list of food banks.  A useful diagram would have been a sketch of the call/response functionality between our server and the Google server.  This would have helped with the general overall framework of how the filter process was to function.  Additionally, a UML diagram of the object returned from the Google Maps API would have been useful so that the necessary information could have been accessed and process from the object in a faster manner.  Without the diagram it took some time to read and understand the API documentation.  This should have been something done beforehand.

## Provide a Burndown diagram



Notes:

- Our total of 19 tasks accounts for our 16 user stories, plus 3 up front tasks that were needed to get the environment up and running.
- The burndown of actual tasks remaining is to date, which is why Actual Tasks Remaining ends at 11/19/2017.
- The Ideal Tasks Remaining plot ends at 5. This accounts for tasks we recognized that we probably wouldn't get to from the beginning.

## Briefly describe any refactoring that you did

1. When building out the Registration features, we realized that we should combine them into one Welcome page that also included a Login feature. We also realized that after a user registers they should be brought to the Login screen.
2. After creating the food truck login, we realized we would run into a complication with the food bank login because these different types of users are stored in different databases and need to be shown different user interfaces when they log in. We resolved this by providing a single login page where the user can switch the type of user they are before they log in.
3. After the separate login for food trucks and food vendors issue was resolved, we realized that we didn't really need our userManager table which was intended to detect user

types. We realized that if user accounts of different types are kept in separate tables, we can always just query those separate tables when necessary.

4. After testing both registration processes, we realized a lot of form input validation was required, so we added that on the client side. We also ensured the combination of user name and phone would be unique by adding a check on the server.

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

We were unsure how to represent that an actual donation occurred within the app and consulted with the customer after they reviewed our current progress. She responded that having a button next to listed nearby banks to donate was a good idea and requested we implement some sort of approval of the donation on the bank side.

## 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 food bank user registration, we found that the form *does not* submit to the database under the following conditions:**

a. String is entered for Phone number
b. Phone number is left blank
c. Open time or close time are left blank
d. Open date is left blank

To fix this, we changed the phone data type to a string and made all of these fields required.

**2. When testing food bank registration we found that the form *does* submit to the database under the following conditions.**

1. Email is left blank
2. Name is left blank

To fix this, we made these fields required.

**3. When testing food vendor registration we found that the form *does not* submit to the database under the following conditions:**

a. String is entered for Phone or Max distance
b. Phone or Max distance are left blank

To fix this, we added validation to data types and made these fields required.

**4. When testing food vendor registration, we found that the form *does* submit under the following conditions:**

a. Name or email left blank.

To fix this, we made these fields required.

**5. Login page testing**
Test 1: Enter unregistered user name, same name twice
Results: Not found message, login page refreshes

Test 2: Enter two different unregistered user names, then registered name
Results: Not found message, page refresh for each unregistered user, registered user able to log in

Test 3: Enter registered user name with incorrect case
Results: Not found message, page refreshes

Test 4: Enter nothing and submit form
Results: Logs in and displays welcome message without username
To fix this, we made this field required.

**6. Vendor Registration Page**
Test 1: Submit blank form
Result: Invalid syntax integer error
To fix this, we provide front end input validation.

**Vendor Registration Page**
Test 1: Submit blank form
Result: Invalid syntax integer error
To fix this, we provide front end input validation.

## What is your new schedule for what to complete next week? What user stories will be  done, what is your time estimate for each of these user stories?

Next week we plan on tackling posting donatable food and the stories that center around a user’s donation history. We will implement the browsing history, donation details request, and report creation features. These stories were initially scheduled for this week but are being pushed back. For a food truck to be able to view their previous donations, there must be a way to indicate to the system that a donation occurred, so we are also implementing the confirmation of a donation next week.

**Story:** Post Donatable Food
**Description:** A food truck user should be able to post what food they have available to donate. The posting should include details about the food, such as type and perishability.

**Due/Priority: High (Second week)**
**Tasks**

| **Task** | **Time Estimate** |
|---|---|
| Create table in database for available donation entities. | 4 hrs |
| Create routes on the API server to get and post to a truck user’s entities. | 4 hrs |
| Create a web page to list and edit available food and details. | 4 hrs |

**Story:** Confirm Food was Donated
**Description:** When a food truck user makes a donation, they will need to submit a form containing donation details to the recipient food bank user. The recipient food bank user then needs to approve the donation.
**Due/Priority: Low (Second week)/Won’t get to**
**Tasks:**

| **Task** | **Time Estimate** |
|---|---|
| Create a table in the database to store completed donations | 2 hrs |
| Create web page that allows food truck user to submit a donation to food bank user for approval. | 4 hrs |
| Create web page that notified a bank of pending donations and allows them to approve a donation. | 2 hrs |
| Create a route in the API server to submit a donation for approval, adding the pending donation to the database | 2 hrs |
| Modify the get main menu route for the banks to check for pending donations | 1 hr |
| Create a route in the API server to set a donation to “approved” | 1 hr |

**Story:** Browse Donation History
**Description:** The food truck user should be able to browse recent donations in a list form. This list form will not display donation details.
**Tasks:**

| Task | Time Estimate |
|---|---|
| Create a web page, using Node JS and Jade, that requests and displays history of most recent donations. | 3 hrs |
| Create routes to query and return donation history, using Node JS. | 1 hr |

**Story:** View details of a donation
**Description:** The food truck user should be able to view a donation they have made through the web page. When they view it, basic details should be displayed.
**Tasks:**

| Task | Time Estimate |
|---|---|
| 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 |

**Story:** Generate Donation Report
**Description:** The food truck user should be able to enter a date range and receive a list of donations they made during that date range.
**Tasks:**

| Task | Time Estimate |
|---|---|
| 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. | 3 hrs |

| | |
|---|---|
| Create route to query and return applicable donation information, using Node JS and Jade | 1 hr |

## In one sentence, briefly summarize whether your customer was willing and able to talk with you by email in a timely fashion.

The customer was contacted early Saturday morning to ask for a few clarifications for the upcoming week and to show off what we had completed and responded immediately.