“SW Engineering CSC 648/848 Fall 2019”

Milestone 4

Team 104

CERES

Member Name	Member Role
U Khyoi Nu (unu@mail.sfsu.edu)	Team Lead
Srushti Buddhadev	Backend Lead
David Fang	Frontend Lead
Darnell Kenebrew	Github Master/Backend
Brian Lopez	Scrum Master/Frontend
Jonathan Munoz	Frontend/Backend

History Table
December 3, 2019, version 1.0, First Draft
December 17, 2019, Version 2.0, Second Draft

1. Product Summary:

Product name: Ceres

List of committed functions:

We have a total of **18** committed functions.

1. User can sign up
2. User can sign in
3. User can sign out
4. User can view currently available items in their fridge
5. User can search for a particular item in the inventory
6. User can view sorted inventory items based on expiration date, calories, quantity, and alphabetical order of the item name
7. User can input items into a fridge by uploading an image of a receipt
8. User can review and edit items read from a receipt before confirming to add the items into their inventory
9. User can add a new fridge to their profile
10. User can remove themselves from a fridge
11. User can remove members from a fridge
12. User can switch between the fridges they belong to
13. User can send, accept, and decline invitations to a fridge
14. User can view shopping lists for different fridges
15. User can add, remove, and edit shopping list contents
16. User can get suggestions for recipes based on current items in their inventory
17. User can view the ingredients of a recipe and add the missing items to a shopping list
18. User can be redirected to a link that provides the full recipe

Unique Feature: A unique feature in our product is the multiple sharable fridges feature. The reason this feature is unique is because it allows a user to be a part of a group that we call a “fridge”. Similar to real life, a user is not confined to a sole fridge, but can be a part of multiple fridges (e.g. home, work etc.) with each fridge having different users. Every fridge contains at least one user, and all users by default have a fridge that they can share or invite others into. Every fridge comes with a grocery list that is meant to be filled out by the users of said fridge. All users in a specific fridge have access to the shopping list and can add or remove items from the shopping list. Members of other fridges cannot interact or see other’s fridges as they must be invited to obtain access to the fridge.

URL: http://ec2-54-183-138-25.us-west-1.compute.amazonaws.com/

2. QA Test Plan:

I. Unit Test:

The features we plan to test are the receipt image upload and the recipe recommendation based on current items in the inventory. We will use Python unittest module of Django that will run on a Macbook Pro. We will be using python package coverage.py for analyzing test coverage. The test cases will create an object of Images model, check if the response from the recipe page matches the expected outcome. The failed test cases will be put into Github as issues.

II. Integration Test:

The functional requirements to be tested are the sign in, sign up, sign out, search for a recipe, redirection to a recipe, create a grocery list, view grocery lists and manage their contents, receipt upload, send/accept and decline invitations, notification of expiration, and update password. We are testing **9** features out of **18**. For our integration test, our hardware and software setup is:

1. Apple iPhone X
   Running on ios 12.1.2
   Safari Browser
2. Apple iPhone 8
   Running on ios 10.1.2
   Safari Browser
3. Apple iPhone 6s
   Running on ios 12.4.1
   Google Chrome Browser
4. Samsung Galaxy core prime
   Android 4.4.4(Kitkat)
   Google Chrome Browser
5. Apple Macbook Pro
   Running on MacOs Mojave
   Google Chrome Browser
6. Windows Computer
   Running on Windows 8.1
   Google Chrome Browser

The link we used to run these tests:
http://ec2-54-183-138-25.us-west-1.compute.amazonaws.com/

Test id #	Test Description	Date Tested	Test Scenario	Test Data	Test Results
1	User Sign Up	12/2/19	Enter Username Enter Email Enter Password	<Any name> <Valid Email> <Valid Password>	Successful Sign Up (PASS)
2	User Sign In	12/2/19	Enter Username Enter Password	<Existing Email> <Existing Password>	Successful Sign In (PASS)
3	User Sign Out	12/2/19	Click On Sign Out On Menu Bar	<Click link>	Successful Sign Out (PASS)
4	User searches for a particular item in inventory	12/2/19	Enter item name	<Item name>	Searched Results Displayed Successful (PASS)
5	User gets recipe suggestions based on current inventory, view ingredients and add missing items to shopping list	12/2/19	Click on Recipe link on Menu Bar Click on Add to Shopping List Button for the item that needs to be added	<Click Link> <Click Button>	Recipes displayed successfully (PASS) Items added to shopping cart successfully
6	User redirected to full recipe page	12/2/19	Click on View Full Recipe button on a particular recipe on Recipe page	<Click Link>	Link Successfully Redirected (PASS)

7	User add new fridge	12/2/19	Enter fridge name Click on Create	<Fridge Name> <Click on Create Button>	New fridge creation Successful (PASS)
8	User can view all of their available shopping lists and manage its contents	12/2/19	Click on Shopping List on menu bar Click on View Info button of a shopping list Enter Item name Click on Add button Click on +/- button to increase/decrease quantity/Type in the quantity Click on Confirm Changes Button	<Click On Shopping List link> <Click On View Info button of Shopping List> <Enter Item name> <Click on Add button> <Click On Plus/Minus Button On Item> <Enter item quantity> <Click on Confirm Changes Button>	Link Redirects To Shopping List page Successful (PASS) Link Redirects To Manage List page Successful (PASS) Item Is Incremented Or Decremented In Quantity Successful (PASS) Item Manually Added Successful (PASS) Confirm Button saved changes Successful (PASS)
9	Receipt upload	12/2/19	Click on Choose File Click on Upload Edit item name and	<Click On Choose File Button> <Choose Receipt Image File From	Upload Button Allows User To Search For A Photo Successful (PASS) Image Can Be Chosen And Is Successfully

			quantity if necessary Click on Save Form	Computer/Phone> <Click On Upload Image Button> <Click on Save Form Button>	Uploaded And Saved Into The Database Successful (PASS)

3. Beta Test Plan:

Our beta test plan for our product is to release it to a small set of users, to see if any bugs arise during their time with it. During this time our users will consist of our parents, siblings, cousins and friends. In order to track user data we will ask our participants to take notes on issues, their experience, and their overall opinion on how the product handles real-world tasks. In order to collect user feedback we will be available through email as it is the easiest form of communication for us to get back to our users as well as in person because our beta testers are people that we know. Our plan to collect the behavior of our product such as crashes, user interaction with the product, and user's ability to navigate through our product is for users to communicate with us either personally or through any of our communication channels.

Our System setup looks like:

1. Apple iPhone X
   Running on ios 12.1.2
   Safari Browser
2. Apple iPhone 8
   Running on ios 10.1.2
   Safari Browser
3. Apple iPhone 6s
   Running on ios 12.4.1
   Google Chrome Browser
4. Samsung Galaxy core prime
   Android 4.4.4(Kitkat)
   Google Chrome Browser
5. Apple Macbook Pro
   Running on MacOs Mojave
   Google Chrome Browser
6. Windows Computer
   Running on Windows 8.1
   Google Chrome Browser

The URL of our starting point for beta testing is:
http://ec2-54-183-138-25.us-west-1.compute.amazonaws.com/

4. Code Review:

a. The coding style we chose was the Pep-8 coding style. We have provided all team members with a link to the documentation for Pep-8 and how to follow it correctly, the link is: https://www.python.org/dev/peps/pep-0008. In order to enforce this coding style, we as a team decided to review each other's code before merging to our branches. These code reviewer roles have been assigned to each of the product leads, the front end lead reviews all front end code while the back end lead reviews all of the back end code. The front end and back end leads must ensure themselves to be on the same page when reviewing code as each one can have a different perspective on what is and what is not acceptable. Another method that we use to enforce our coding style is pull request review on github which has proved to be quite useful for us. As a team we make sure that all submitted code is reviewed by at least one person, but if more members of the team are available we recommend everyone to review all code that is submitted.

b. An example of our code:

Front End:

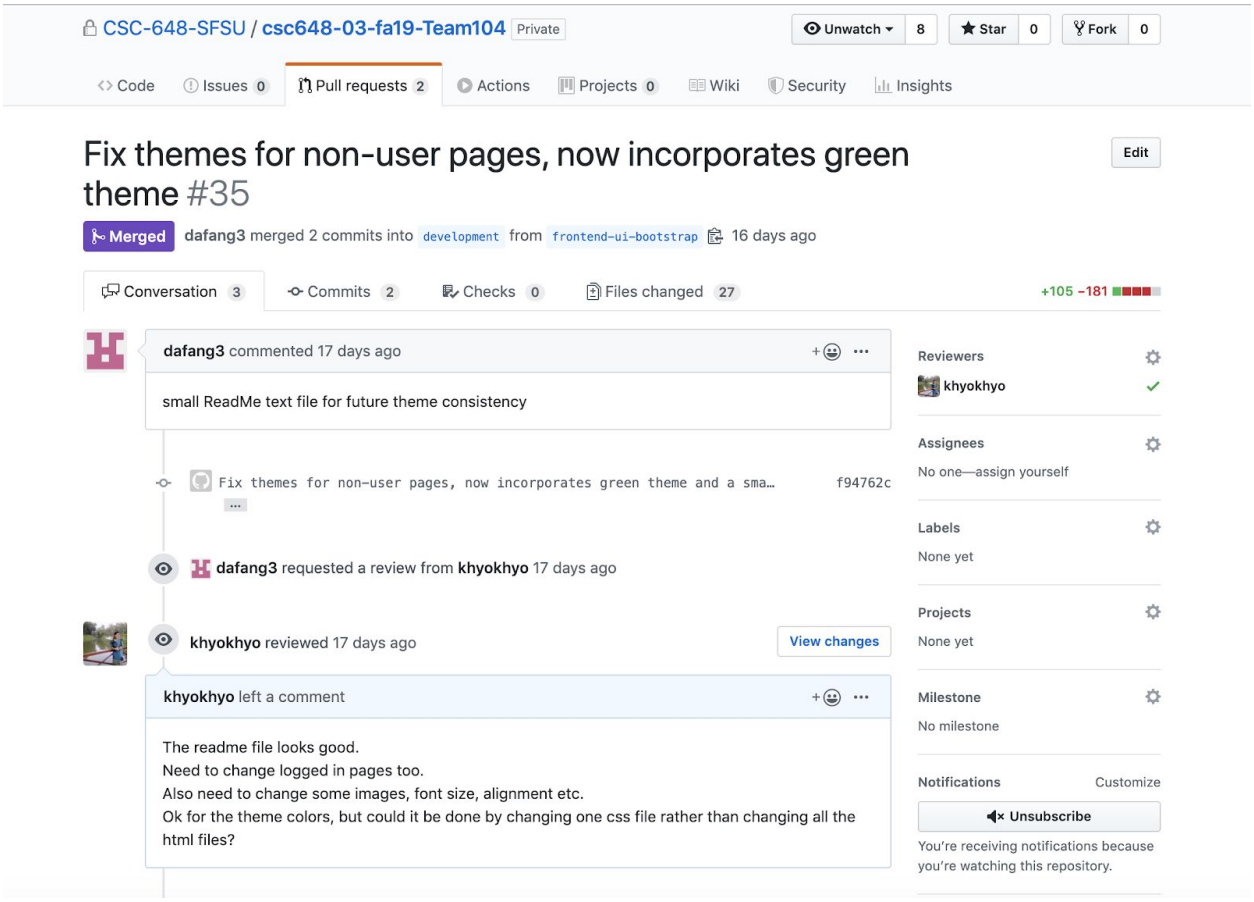
```
<!-- User Signin Page -->
{% extends "site/base.html" %}
{% load crispy_forms_tags %}
{% block content %}
    <div class="content-section">
        <form method="POST">
            {% csrf_token %}
            <fieldset class="form-group text-color-green">
                <legend class="border-bottem mb-4">Log In</legend>
                {{ form|crispy }}
            </fieldset>
            <div class="form-group">
                <button class="btn btn-outline-info button-color" type="submit">Log in</button>
            </div>
        </form>
        <div class="border-top pt-3">
            <small class="text-muted">
            Need an account? <a href="{% url 'signup' %}" class="ml-2 text-color-green">Sign Up</a>
            </small>
        </div>
    </div>
{% endblock content %}
```

```python
from django import forms
from .models import *
from .models import FoodDetails
from .models import Fridges
from django.contrib.auth.forms import UserCreationForm
from django.contrib.auth import get_user_model

# User model extending default User model of django
User = get_user_model()

# Form for milestone 2 vertical prototype to enter values into food_details table
class FoodDetailsForm(forms.ModelForm):
    class Meta:
        model = FoodDetails
        fields = ("__all__")

# Form for milestone 2 vertical prototype to search values from food_details table
class FoodDetailsSearchForm(forms.ModelForm):
    class Meta:
        model = FoodDetails
        fields = ("__all__")

# Form for milestone 3 search values from food_details table
class InventorySearchForm(forms.ModelForm):
    class Meta:
        model = FoodDetails
        fields = ("__all__")

# Form for creating fridge
class FridgeCreateForm(forms.ModelForm):
    class Meta:
        model = Fridges
        fields = ['name']

# Form for user signup
class UserRegisterForm(UserCreationForm):
    email = forms.EmailField()

    class Meta:
        model = User
        fields = ['username', 'email', 'password1', 'password2']

# Form for uploading receipt images
class ImagesForm(forms.ModelForm):

    class Meta:
        model = Images
        fields = ['path']

```

Back End:

```python
from django.shortcuts import render, redirect
from django.http import HttpResponse
from django.http import HttpResponseRedirect
from django.contrib.auth.decorators import login_required
from django.db.models import Q
from .models import FoodDetails
from .forms import FoodDetailsForm
from .forms import *

"""
Vertical prototype for milestone 2
Displays all the food entries in food_details table
Also takes input from the FoodDetailsForm and saves to database
"""
def food(request):
    foods = FoodDetails.objects.all()
    if request.method == 'POST':
        form = FoodDetailsForm(request.POST)
        if form.is_valid():
            try:
                form.save()
                return redirect()
            except:
                pass
    else:
        form = FoodDetailsForm()
    return render(request, 'food/food-details.html', {'form': form, 'foods': foods})

"""
Vertical prototype for milestone 2
Search for a food item from food_details table
"""
def search(request):
    if request.method == 'GET':
        query = request.GET.get('name')
        submitbutton = request.GET.get('submit')
        if query is not None:
            lookups = Q(name__icontains=query)

            results = FoodDetails.objects.filter(lookups).distinct()

            context = {'results': results,
                       'submitbutton': submitbutton}

            return render(request, 'food/food-search.html', context)
        else:
            return render(request, 'food/food-search.html')

    else:
        return render(request, 'food/food-search.html')

```

Pull Request Review:



5. Self-Check:

- Application shall be developed, tested and deployed using tools and servers reviewed by Class TA (Nicholas Olegovich Stepanov) in M0 (some may be provided in the class, some may be chosen by the student team but all tools and servers have to be reviewed by class TA). (Done)
- Application shall be optimized for mobile browsers. (Done)
- Data shall be stored in the team's chosen database technology on the team's deployment server. (Done)
- Privacy of users shall be protected and all privacy policies will be appropriately communicated to the users. (Done)
- Application shall be very easy to use and intuitive. (Done)
- Pay functionality, if any (e.g. paying for goods and services) shall not be implemented. (Done)
- Site security: basic best practices shall be applied (Done)
- Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development (On track)
- Standard desktop use optimized (Done)
- The website shall prominently display the following exact text on all pages *"SFSU Software Engineering Project CSC 648-848, Fall 2019. For Demonstration Only"* at the top of the WWW page. (Important so as to not confuse this with a real application). (Done)