Princeton Food Alert

Product Guide

Part 1: User Guide

What is Princeton Food Alert?

Princeton Food Alert sends notifications to Princeton users when preferred foods are being served in campus dining halls. Before the launch of Princeton Food Alert, students could only get dining menu information from a tedious manual inspection of the campus dining website. Princeton Food Alert provides a much more efficient system where students log their favorite food preferences and then receive notifications when one of their preferences is in a dining hall.

Login

Users access Princeton Food Alert by visiting *ptonfoodalert.herokuapp.com*. In order to use Princeton Food Alert, users must have a valid Princeton email address and netID. Upon entry to Princeton Food Alert, users will log in through CAS, or Central Authentication Service, using their Princeton netID and password. After logging in, users will be taken to the home screen of Princeton Food Alert which is the “Add Food” page.

Add Food

On the “Add Food” page, which is also the home page, users have the ability to add food preferences. By typing in the search box, users will generate updating lists of autocomplete entries based on foods which have appeared previously in the dining halls. The autocomplete function provides inspiration to users and also helps reduce user error such as spelling mistakes and increases the chances of a user getting a match. In addition, users can add items that do not appear in the autocomplete. A user might want to add a keyword (i.e. “avocado” so that it will match with any food item containing “avocado”), or rare foods that do not appear in the autocomplete entries,

such as “lobster”. After adding a preference, the user will receive an on-screen notification that states “New entry added!”. A user can delete a preference by clicking on the ‘X’ on the right side of the preference. The user will receive a on-screen notification that states “Entry removed.”

Popular Foods

The “Popular Foods” page lists the top ten most preferred food items for all users of Princeton Food Alert. The list updates every day and is another opportunity for users to get ideas about what food preferences to potentially add and to see what other student prefer. A user can add food preferences directly from the top ten list by clicking the “+” on the right side of the food item.

Emails

Users receive emails when a preferred food is in a dining hall based on their notification and dining hall preferences on the “Settings” page. An email from Princeton Food Alert includes how many preferred foods for a user are in a dining hall along with the date, meal time, and location of where the food will be served, the food name, and the food preference that resulted in the food match. Also included on the email is a link to the Princeton Food Alert website in order to let users directly update any preferences or settings.

Account Settings

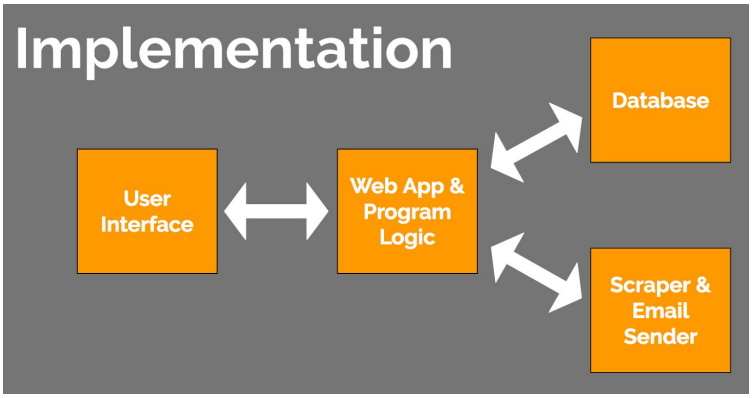
On the “Settings” page, users are able to select notification and dining hall preferences. Users can change any setting by clicking on the appropriate box to either “check” or “uncheck” a setting, and committing these changes with the “Apply” button. The default notification preferences are for a user to be notified one day before and one week before the item is in a dining hall. The default dining hall settings are for a user to receive notifications regarding foods in all dining halls. There are six dining halls at Princeton: Whitman, Butler-Wilson, Rocky-Mathey, Forbes, the Center for Jewish Life, and the Graduate College. Users can choose to receive notifications from all dining halls or can select certain dining halls to receive notifications from. If a user decides to no longer use Princeton

Food Alert, they can delete their account by clicking the “Delete Account” option on the bottom left corner of the “Settings” page. A user’s information will be deleted from our database. If a user wants to re-register with Princeton Food Alert they simply log in again.

Part 2: Developer Guide

Overview of System Components

The app consists of a front-end user interface, a back-end, and an intermediate logic module.



User Interface

The goal for the User Interface was something easy to use and intuitive. An oversized text box that contrasts with the background was used to capture the user's attention when they log in. Most of the UI was completed using Bootstrap, a web framework containing CSS and HTML based design templates for the various interface components along with JavaScript to handle user input. Each page has a different template that extends a core template by the name of “layout.html”. The different templates can be found in the “templates” folder while stylesheets and images used can be found in the “static” folder. In addition, on the website, the tab “Tell us what you think.” in the bottom right corner is a form for user feedback that submits to a Google form.

Web Application & Program Logic

The main idea behind Flask is creating a straightforward Python app to control a web interface in HTML. The Flask “render_template()” function takes a .html file as input and renders it in the browser. When users click on buttons or links on the webpage, Flask handles redirection to a Python function. Thus the app can respond to user input and behavior, and can render the necessary changes on the screen. The Flask app is divided into functions for each of the application screens, displaying the entries, “Popular Foods”, “About Us”, and “Settings”. Additional hidden functions take care of operations such as adding and removing entries. The autocomplete function was handled directly in the HTML. The program logic module is run every time a user interacts with the website, so caution should be taken to protect against all varieties of bad input and edge cases. Try/except statements are used liberally to guard against unlikely failures in the database.

Database

Princeton Food Alert uses a Heroku add-on called mLab, which hosts MongoDB databases, to store important app data such as user food preferences, notification settings, popular foods, and a dictionary between user NetIDs and names.

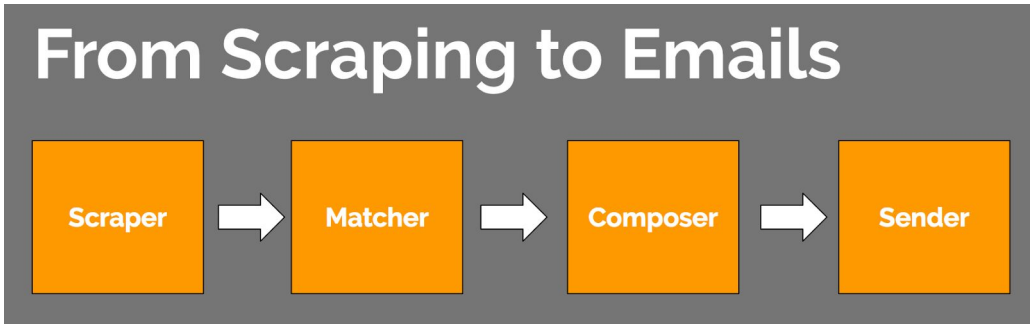
When a user logs into Princeton Food Alert and adds food preferences or changes their settings, a document in the “Users” collection is created using the user’s netID. Any time a user adds or deletes food preferences or notification settings, the MongoDB database is updated. In order to display a user’s current food preferences and current settings, the web app accesses a user’s document when they log into Princeton Food Alert.

Princeton Food Alert runs a script each day that updates the “Top_Ten” collection with the current most preferred food items. The web app then displays this list on the “Popular Foods” page. In order to display a user’s first name on the website and in order to include a user’s first name in

notification emails, both the web app and the email composer match a user's netID to their first name using the "NetIDs" collection.

Scraper & Email Sender

The Scraper and Email Sender is broken into four python files and run together with a shell script called "daily_script.sh".



The first file is "scraper.py" and outputs a tab-delimited table of the foods that will be in each dining hall for every day in the upcoming week. Each dining hall and day has a separate dining hall web page that gets scraped. The scraper uses the LXML library to break the webpage into a tree structure and find the parts with a "Recipe Desc", which is the name of the food item.

The output of scraper.py gets written to a file and then read by "matcher.py". The system writes/reads a file instead of piping so a developer can see the output of each stage for easier debugging. The matcher outputs a tab-delimited file listing every match between a user and a food item. Matches are only printed if every word of a user's food preference is found in a food item and the item is available on the correct days and dining halls specified by the user in their settings. Note that substrings do not match, only the whole word. So if a user inputs "ice" it will not match with "rice". Also note that the order of words does not matter. If a user inputs "BBQ Chicken" it will match with "Chicken BBQ". The output of matcher.py is then sorted alphabetically so the matches for each user are next to each other.

The sorted output is then read in by "email_composer.py", which takes the matches and writes it into an HTML format to be sent by email. All of the matches for one user are read in before

the email is written. The user's first name is also retrieved from the database and written into the email. If the user is not in the database then the netID is used instead.

The composed emails is then sent to "send_email.py" which reads in each email (which begins with the user's netID and ends with "<end of email>") and sends them.

Helpful Software Systems

There are several third-party products and services that were critical to the implementation of Princeton Food Alert. The first is Heroku, which provided a free and easy-to-use server for hosting our app, allowing it to be accessible 24/7. Heroku has built-in Git support which allowed for pushing changes to the server after testing them locally. Additionally, Heroku's Scheduler service allowed scheduling of daily or hourly tasks such as scraping dining hall websites and sending emails. We also used a free MongoDB account, which allowed efficient storage and retrieval of all user information. For our feedback form, we referred to the following: https://github.com/izhan/gform_feedback. Finally, we used Awesomplete, a lightweight implementation of the autocomplete feature for text input fields. After producing a list of food options available in the dining hall, Awesomplete created a drop-down autocomplete on the main search bar of our app.

For Questions and Comments, please contact the Food Alert team:

Melana Hammel mhammel@princeton.edu
Antonio Papa agpapa@princeton.edu
Thomas Clark thclark@princeton.edu
Ramon Ayala ra4@princeton.edu

Happy Eating!