PorkBun Product Guide

Khatanbuuvei Bold and Justin Chang
porkbun.herokuapp.com

Overview

PorkBun is a web application aimed at Princeton students, particularly undergraduates, who use the dining halls offered on campus. By offering the ability to quickly see matches between dining hall menus and one's preferences, we hope to reduce the repetitive nature that the dining halls are often associated with.

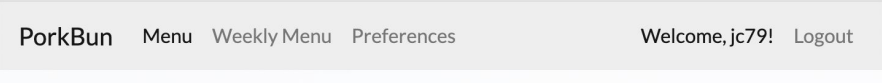
User Guide

PorkBun is very simple to access, as it only requires an Internet connection and a browser. Likewise, user experience is intended to be very similar to a service that many Princeton students are already familiar with: TigerMenus. By providing a easily accessible and familiar interface, we hope to make it easy for users to become acquainted to our website, lowering the effort needed to acquire new users within the Princeton community.

That being said, we've written the following descriptions to allow a user to make the best use of our site.

Pages

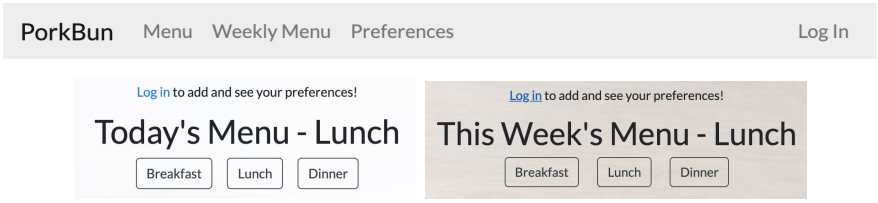
The PorkBun website has four main pages: 1) daily menu, 2) weekly menu, 3) preferences page, and 4) landing page. To navigate among these pages, simply click on the links in the navigation bar. The daily menu page is titled "Menu" and the landing page's link is the "PorkBun" banner in the top left.



Navigation bar with links to the four pages on the left side

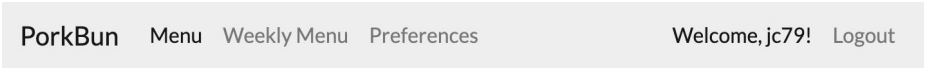
Logging In/Out

Much of the main functionality of our website requires logging in with a Princeton netid and password through the Central Authentication Service (CAS), much like other sites designed for Princeton students. If not logged in, there are multiple links that lead to the login page: 1) the link on the landing page, 2) the links on the top of the daily and weekly menu pages, and 3) the “Log in” button on the navigation bar. Additionally, if the user hasn’t logged in already, they will be asked to log in before accessing the “Preferences” page.



Different ways to log in. Top: navigation bar; Bottom: daily and weekly menu pages

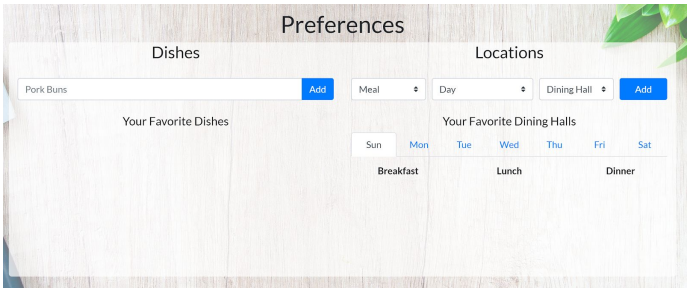
Unlike logging in, there is only one link to sign out of an account, which is the “Logout” link in the navigation bar that shows up when logged in.



Logout button that shows up in the navigation bar once logged in

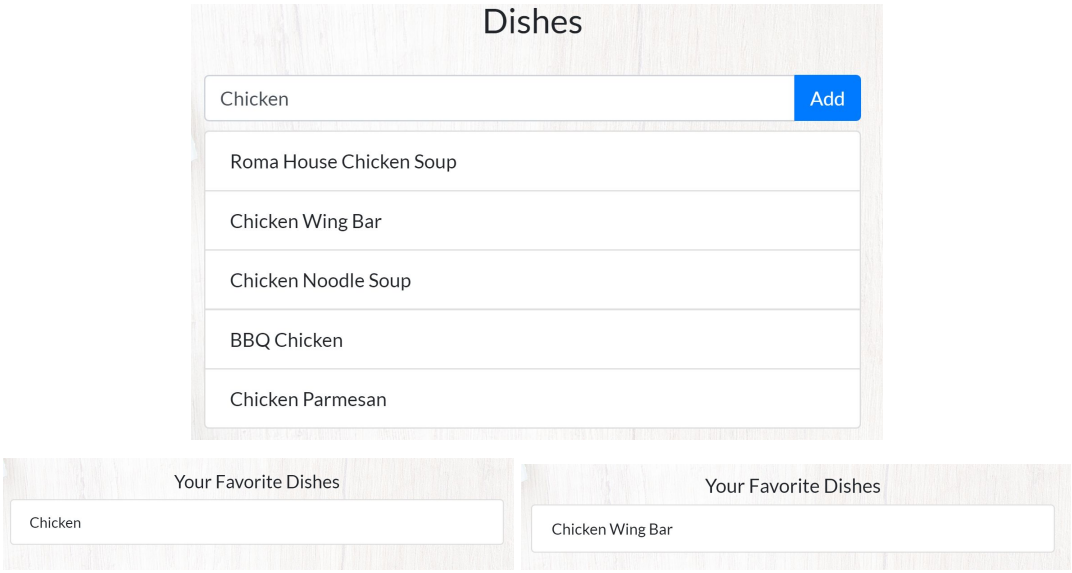
Adding/Removing Preferences

Once the user has logged in, they can access the Preferences page, where they set their preferred dishes and dining halls.



The “Preferences” interface

On the left side of the interface, the user can add and see their favorite dishes. There are two ways to do so: 1) by typing out a dish preference in the textbox, and clicking on the “Add” button, or 2) by typing a dish preference, and clicking on one of the suggestions that appear below the textbox. Once the user adds a favorite dish, the dish appears below “Your Favorite Dishes”.



Top: The suggestions shown upon typing “Chicken”, starting with “Roma House Chicken Soup”
Bottom left: after clicking the Add button; Bottom right: after clicking the suggestion “Chicken Wing Bar”

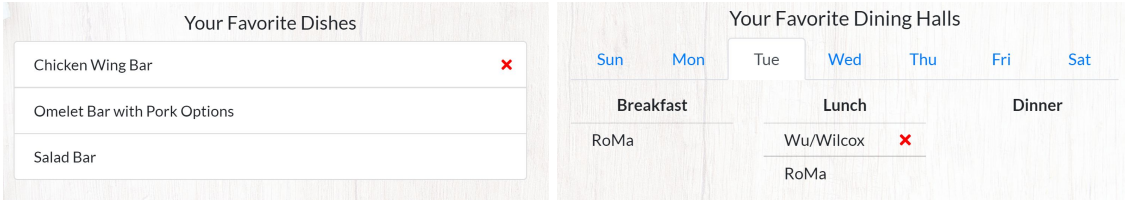
On the right side of the interface, the user adds their favorite dining halls for each day and meal time. To add a dining hall preference, the user selects the appropriate options from the dropdown menus entitled “Meal”, “Day” and “Dining Hall”, and clicks on the “Add” button on the right side of the dropdown menus. To make this process less tedious, we have provided the option to add a preference for all meals (breakfast, lunch and dinner), and/or for all days of the week simultaneously.

The dropdown menus

Once the user has added a preference, the user can see their location preferences under “Your Favorite Dining Halls”. This section has 7 tabs for each day of the week, which the user can select by clicking on the desired tab. Under each tab, there are three columns, listing the dining hall preferences for each meal of the day corresponding to the selected tab. The user can rank this list by simply dragging and dropping a dining hall to the desired position on the list.

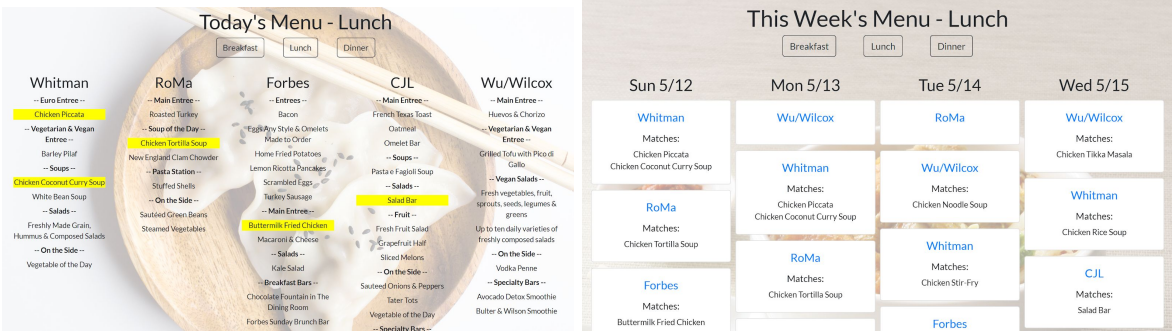
“Your Favorite Dining Halls” section, with the Tuesday tab selected

To remove a dining hall preference, the user simply hovers over a preference with their cursor. Then, a red “X” icon appears to the right of the listed preference, on which the user can click to remove said preference.



Examples of the remove icon shown upon hovering over preferences

After setting their preferences, the user will see that dishes matching their preferences are highlighted in the daily and weekly menus, and that the menus are ordered according to their preferences.



Left: daily menu; Right: weekly menu

Developer Guide

In terms of the internal structure of our system, the server runs on a Node.js environment, using the Express web framework, and a MongoDB database. The website is hosted on Heroku. We will go more in detail about each of these aspects below.

NodeJS Server

The minimal Express framework simplified our development process considerably. There are three main scripts that handle most of the server-side routing duties.

1. Index (src/index.js): This script configures Express settings, creates a cookie session, handles the main routes, such as /landing, /menu, and /week, and most importantly, starts the server.

2. Authentication (src/controllers/auth.js): This script handles all requests related to authentication. Particularly, it creates an Express router that interacts with CAS, and ensures that the user is logged in before accessing certain pages.

3. Preferences (src/controllers/prefs.js): This script handles all requests related to preferences. Particularly, it creates a router that handles the addition and removal of user preferences, and interacts with the database.

EJS Views

Thanks to the large NPM community, we were able to find packages that suited our needs throughout the development process, one of which was EJS. EJS allows us to pass JavaScript values easily from our backend scripts to our web pages, which made it simple to display dynamic values such as the menus and preferences on screen.

The .ejs files are styled with external CSS files, and we use jQuery to dynamically interact with the DOM for a fast and responsive user experience.

Database

We used a MongoDB Atlas cloud database. The database has 2 collections: “dishes” and “users”.

The dishes collection consists of documents that represent dishes that appear on the online Princeton Dining Services menus, and are updated periodically. Each document specifies the dish name, and the number of times that dish has appeared on the menus.

The users collection consists of documents that store the dish and location preferences of users. Each document contains the user's Net ID, and their dish preferences as an array, and their location preferences as an object.

Scripts

Our system is supported by several scripts located in the src/scripts directory. These scripts handle duties such as populating and interacting with the database, and also contains the algorithms we used for PorkBun. The scripts are listed below.

1. scraper.js: Provides functions that scrape the menus provided by Dining Services, and add them to the database. The scraping functions are run daily on a schedule.

2. menu.ejs: Provides various functions that we use in both the daily and weekly menus to rank dining halls, and highlight dishes.

3. db/db_dishes.js, db/db_users.js: Provides functions that allow us to interact with the database in various ways. Mainly, these scripts create, add, and remove user and dish documents.

Deployment

PorkBun is hosted on Heroku. We used the New Relic add-on on Heroku. New Relic is a performance monitoring tool that works by pinging the website at a given interval, which prevents Heroku from disabling the server. This is necessary because the server must be active for the scraping scripts to start.