P01 - ArRESTed Development - Global Bites
Ben Rudinski, Tiffany Yang, Tim Ng, Endrit Idrizi
Team Bareustoph (bah-reh-us-tofh)
Pd. 5
TARGET SHIP DATE: {2024-12-15}
(Yes it's a Sunday but this is because it's better for our peace of mind 😊)

Overview: Global Bites is an interactive and immersive website that showcases traditional recipes for global holidays. Users can explore holidays worldwide, find associated recipes, and view high-quality food images for inspiration. The site features an interactive world map where users can click on countries to discover popular holidays, traditional dishes, cultural customs, and fun facts. Additionally, users can create accounts to save favorite recipes, share experiences, and engage with a community of global food enthusiasts.

Key Features Include

- Users can select countries and view holiday details, recipes, and cultural insights.
- Accounts allow users to save recipes, earn badges, and track cooking history.
- Forums, recipe sharing, and cooking challenges to connect users with other users.

Program Components:
Front-End Framework (FEF): Tailwind CSS
Backend Framework (BEF): Flask (Python)
Database: SQLite3
RESTful APIs:

- Calendarific API - Looks for upcoming holidays to "suggest" specific recipes.
- Spoonacular API - Provides recipe data based on various criteria
    - Search Recipes
    - Search Recipes by Nutrients
    - Search Recipes by Ingredients
- Unsplash API - Supplies high-quality images for recipes and ingredients.
- GoogleMaps API - Adds location-based functionality, mapping services, and geolocation capabilities into their websites, mobile applications, and other digital platforms.

Component Relationships:
Flask ↔ Python Middleware

- Flask routes user requests to the middleware, which processes data and returns responses.

Python Middleware ↔ SQLite

- Middleware queries and updates the database as needed.

HTML/CSS ↔ Flask

- The front end sends requests to Flask and displays data received from it.

Flask ↔ APIs

- Flask and middleware handle API calls to external services.

Component Map:

[User Interface]
↓
[Front End (HTML/CSS/JS)]
↓
[Flask Application]
↔
[Python Middleware]
↔
[SQLite Database]
↔
[External APIs]

Legend:

- ↓: Data/request flow
- ↔: Interaction between components

Website Pages:

Home Page (‘/home’)

- Highlights upcoming holidays with featured recipes.-
- Showcases user achievements and popular community posts.

Explore Map (‘/map’)

- Interactive world map to select countries and view related content.
- Displays holiday details, traditional dishes, cultural facts, and fun insights.
- The map will use Google Maps API with custom overlays to highlight countries. Clicking on a country displays a pop-up and then a link to a page with details about holidays and recipes.

Search Page (‘/search’)

- Advanced filters (cuisine, ingredients, dietary restrictions, difficulty level).

Recipe Page (‘/recipes’)

- Detailed recipe instructions, nutritional info, user comments, and ratings.

User Profile (‘/user’)

- Displays saved recipes, badges earned, cooking history, and personal notes.

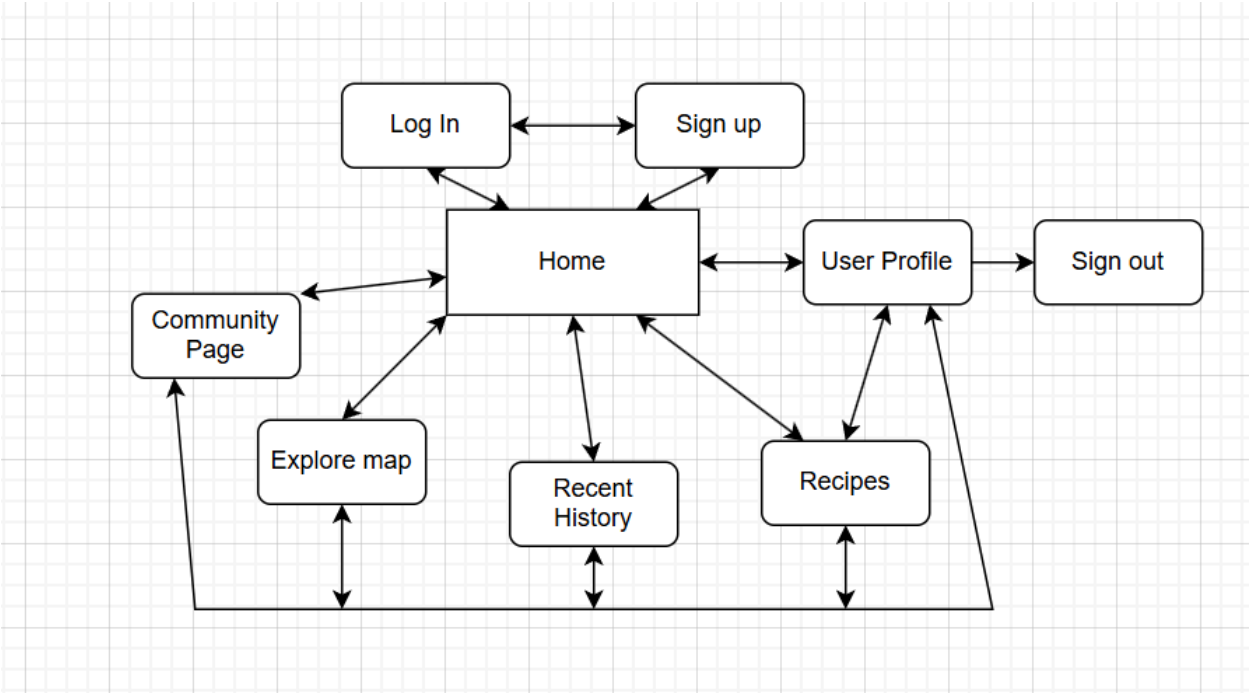
Community Hub (‘/community’)

- Forums, user-shared recipes, cooking challenges, and events.

Login/Signup Page (‘/login’ & ‘/register’)

- User authentication and account creation.
- Passwords will be hashed using **bcrypt** in the Flask application for secure storage.

Website map:



Database Organization:

Users

```
user_id (primary key)
username
password_hash
join_date
badges
```

Recipes

```
recipe_id(primary key)
title
ingredients
instructions
cuisine
image_url
```

Recent History

```
history_id(primary key)
user_id
recipe_id
interaction_type ("viewed, "cooked", or "saved")
user_notes
```

Comments

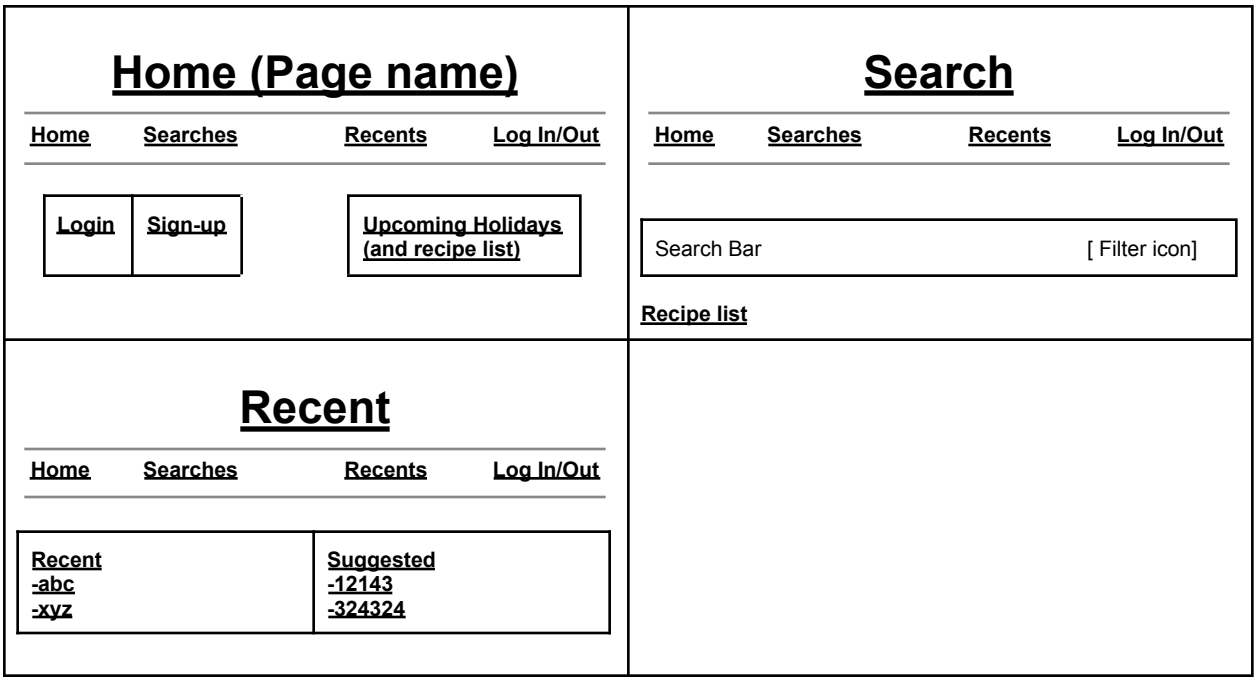
```
comment_id (primary key)
user_id (foreign key)
recipe_id (foreign key)
text
timestamp
```

Badges

```
badge_id(primary key)
badge_name
description
icon_url
```

***Badge images will be stored locally or fetched from Unsplash for default designs. Custom badges may use user-uploaded images, stored in the SQLite database.*

Site Map:



WHY TAILWIND?

We are using Tailwind CSS because:

- Allows rapid styling without writing custom CSS.

- Easy to create designs that work on all devices.
- Highly customizable to fit the project's unique needs.
- Works seamlessly with modern JavaScript frameworks and libraries.

We plan to make use of:

- To ensure the site looks good on all screen sizes.

- Build reusable components for buttons, cards, and forms.
- Enhance user experience with smooth interactions**.**

API USAGE DETAILS

- **Calendarific API**
    - Used to fetch upcoming holidays based on date and country.
    - Suggest recipes related to these holidays.
- **Spoonacular API**
    - Obtain detailed recipe data.
    - Use filters for dietary restrictions and ingredient availability.
- **Unsplash API**
    - Fetch high-quality images for recipes and backgrounds.
    - Enhance visual appeal of the site.

- **Google Maps API**
    - Create an interactive world map.
    - Allow users to select countries and view associated content.

Task Designations:
Button-Mashing, Feature-Wrangling, and Making Sure Everything Actually Works: *Ben*
Database Whisperer (and Chief Leak Plugger): *Tiffany*
CSS Sorcery and Stylin' Shenanigans + HTML: *Endrit*
Flask Mastery (a.k.a. Server Wizardry): *Tim*

Cuisine input/classification by user recommendation

- Recipe pages have missing information? Please contribute button
- Edit -> dropdown menu of popular cuisine types
- Added to database (text array)
- Shows the majority of the user's inputs on the page as the recipe type

Possible shopping list thing if we have extra time to implement this