FeastFinder

Cody Newton, Samuel Garmany, Jack Van Dyke, Tate Charboneau, Mason Esslinger, Christopher Chan

TABLE OF CONTENTS

01

IDENTIFYING THE PROBLEM

Problem Statement

Reaching a consensus on where to eat is a common challenge for groups, as differing tastes, dietary needs, and location preferences make finding a mutually agreeable option difficult and inefficient.

02

APPLICATION DESCRIPTION

Solution & Product Description

FeastFinder

FeastFinder transforms group dining decisions into a fun and efficient experience. It's a collaborative website enabling groups to easily find restaurants that everyone genuinely wants to try.

It also takes the pressure off by keeping choices private until the group agrees together.

How It Works

Users create or join a group, set shared preferences like location radius and cuisine types, and then individually swipe 'yes' or 'no' on curated restaurant options. FeastFinder instantly reveals the match – the restaurant everyone in the group approved – eliminating endless debate and ensuring everyone gets a say.

03

TECH TOOLS & DESIGN

HTML & CSS

Purpose:

- HTML structures the content of the application's user interface, defining elements like text, buttons, and images.
- CSS styles this structure, controlling the layout, colors, fonts, and overall visual appearance that the user interacts with.

Rating:

JavaScript

Purpose:

- JavaScript adds interactivity to the website, handling actions like swipes, button clicks, and dynamically updating what the user sees in their browser. It also manages communication between the user's browser (frontend) and the application's server (backend).

Rating:

NodeJS

Purpose:

- NodeJS serves as the backend runtime environment, allowing us to build the server-side logic of FeastFinder using JavaScript. It handles incoming requests from users, processes data (like matching swipes), and interacts with the databases.

Rating:

Bootstrap

Purpose:

- Bootstrap is a frontend framework that provides pre-built CSS and JavaScript components to quickly style the app's interface. This speeds up development and ensures a consistent, responsive design across different screen sizes (like phones and desktops).

Rating:

Handlebars

Purpose:

- Handlebars is a templating engine used to dynamically generate HTML pages on the server. It allows inserting data (like restaurant details or group information) into HTML templates before sending the final page to the user's browser.

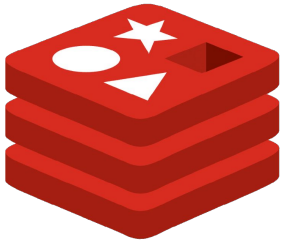
Rating:

Redis

Purpose:

- Redis is a fast, in-memory data store used for caching data and the status of ongoing group swiping sessions. This helps improve the application's performance and responsiveness by reducing load on the main database.

Rating:



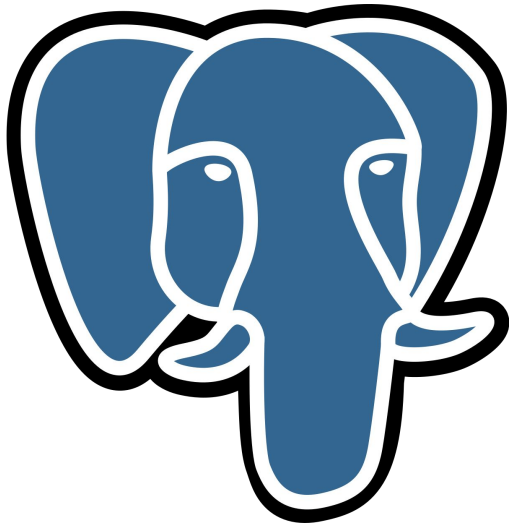
redis

PostgreSQL

Purpose:

- PostgreSQL is the primary relational database used for persistent storage of the application's core data. This includes essential information like user accounts, group memberships, friend lists, and potentially saved preferences or match history.

Rating:



Azure

Purpose:

- Azure is the cloud platform providing the infrastructure to host and run the entire FeastFinder application online. It supplies the necessary servers, databases, networking, and other services required to make the app accessible and scalable.

Rating:

VSCode

Purpose:

- VSCode is the code editor used by our team to write, edit, manage, and debug the project's source code. It provides tools and features that streamline the development workflow for HTML, CSS, JavaScript, NodeJS, and other project files.

Rating:

Docker/Docker Compose

Purpose:

- Docker packages the application components (NodeJS app, database, Redis, etc.) and their dependencies into containers, ensuring they run consistently across development, testing, and production environments.
- Docker Compose is used to define and manage these multiple containers together easily.

Rating:

Google Maps API

Purpose:

- The Google Maps API provides critical location-based functionality, primarily used to search for restaurants based on location, distance, and type. It also supplies details about these restaurants (address, hours, etc.) and is used to provide directions to the chosen restaurant.

Rating:

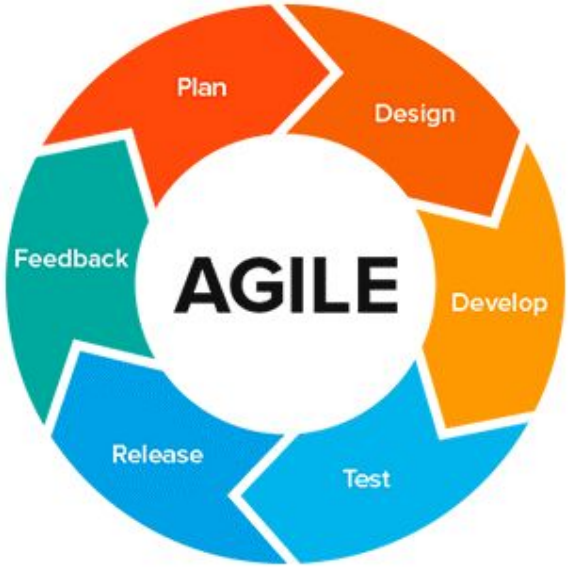
Google Maps

Agile Development Methodology

Purpose:

- Agile is the iterative project management approach used to build FeastFinder. It was a pretty inefficient project management tool for a small team, and resulted in more frustration and time wasted.

Rating:

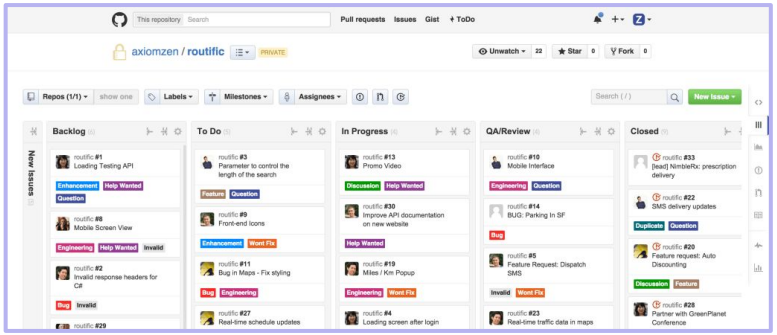


GitHub Projects

Purpose:

- We used GitHub projects for the creation of an Agile-based project board and to track the team's status and progress on tasks as the website was developed.

Rating:

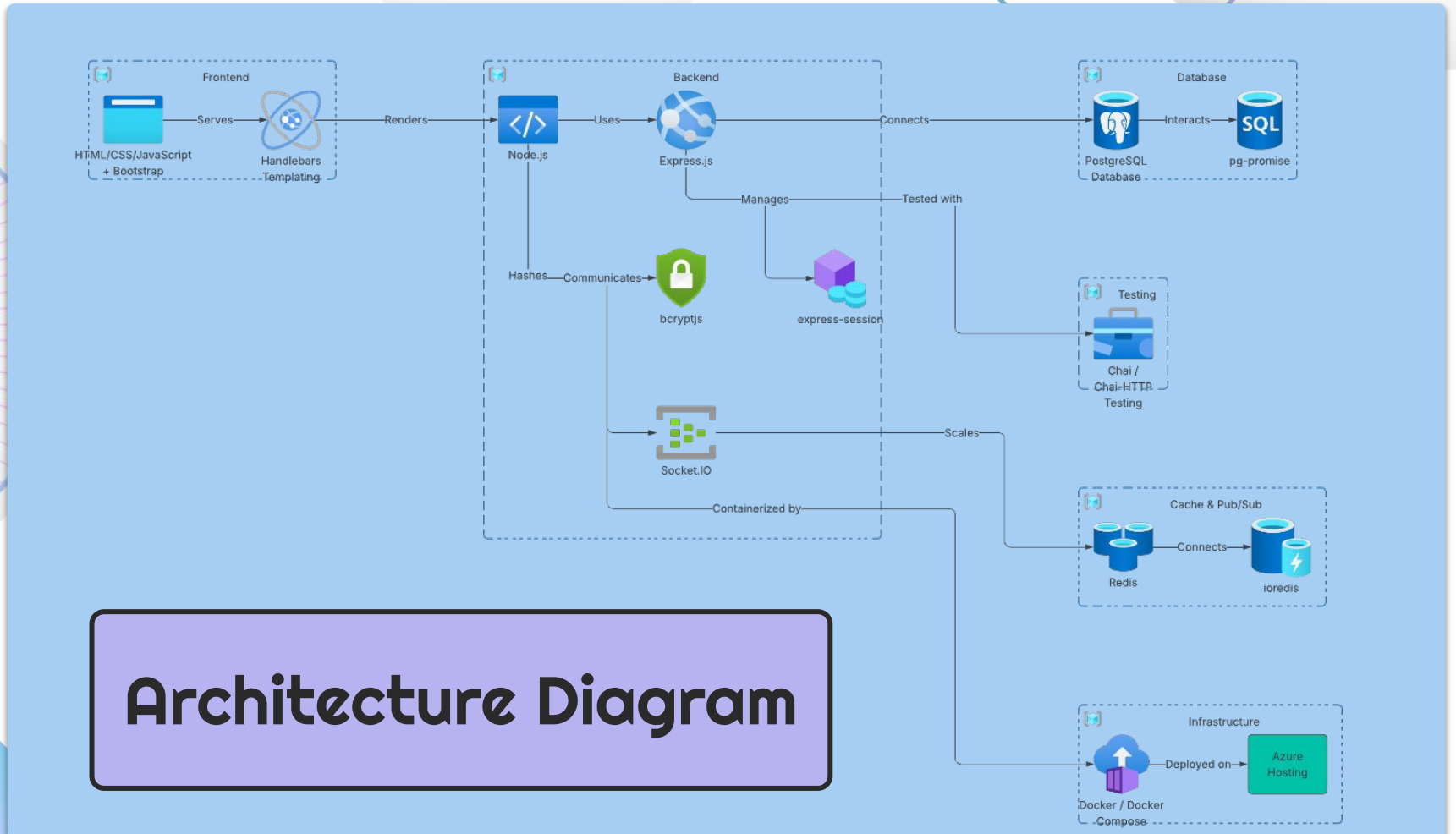


Mocha & Chai

Purpose:

- Mocha and Chai were used to run automated tests against the FeastFinder API endpoints, ensuring backend functionality behaved correctly. Mocha structured and executed the tests, while Chai provided assertions to validate the accuracy of the API responses.

Rating:



Architecture Diagram

04

CHALLENGES FACED

Challenge #1

Challenge:

Coordinating group swiping was initially difficult, as one user finishing didn't update others in the same session in real-time. This is because individual browsers can't directly share live status updates.

Solution:

We implemented Redis as a fast, intermediary data store on our server to manage this coordination. It maintains a shared session state for each active group, letting the backend track completions efficiently and know when to calculate and announce matches.

Challenge #2

Challenge:

Using Redis caused server instability with memory overcommits and errors when registration sent invalid user IDs, impacting sign-up and session handling.

Solution:

We adjusted kernel parameters for memory overcommits and made Redis more stateless to reduce memory load. We also added strict user ID validation before accessing Redis to prevent errors.

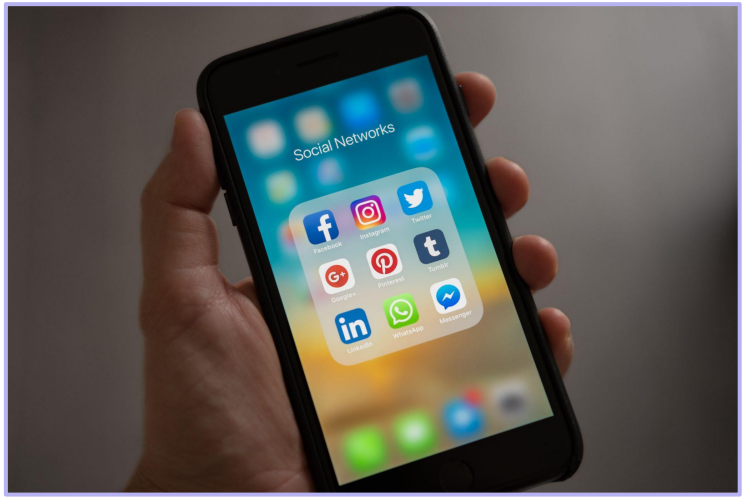
05

FUTURE SCOPE & ENHANCEMENTS

Future Enhancements

Turn the Website into a Mobile App

Developing a native mobile application would offer a more integrated and seamless user experience compared to the website. This would enable features like push notifications for group updates and potentially easier access to device location services.



Future Enhancements

API Improvements for Better Match-Making

This enhancement would involve adding logic to automatically broaden the search parameters (like distance or cuisine types) if the initial criteria yield zero matches for the group. This prevents users from hitting a dead end and increases the chances of finding an acceptable restaurant, even if it requires slight compromise.

Future Enhancements

Potential Paid Features

We could introduce an optional paid feature allowing one user within a group to bypass the matching algorithm after swiping is complete. This "Decider" could then secretly select any restaurant from the pool initially presented as the final choice.

06

LIVE DEMO