DinnerWizard

Final Report

CS 450

Spring 2025

Jacob Vallery

jvallery@bellarmine.edu

04/27/2025

Executive Summary

DinnerWizard is a web-based application designed to help users quickly discover nearby dining options such as restaurants, cafes, and bars based on their current location, or desired location for future planning. The motivation behind this project came from the common need to make quick dining decisions in unfamiliar areas without being overwhelmed by excessive search results. By integrating the Google Maps and Places APIs, DinnerWizard allows users to enter a location through a search bar with autocomplete functionality, select a preferred place type, and define a search radius. The system then returns a filtered list of nearby establishments and displays them on a table next to a localized map.

Developed using HTML, CSS, and JavaScript, DinnerWizard also leverages Node.js for secure server-side processing and Docker for consistent containerized deployment. Visual Studio Code served as the primary development environment, offering integrated tools for coding, testing, and debugging.

The project met its goals by delivering a responsive, functional, and visually clean user interface that connects with Google's JavaScript Maps API as a backend service. API requests work reliably, and Docker ensures a stable runtime environment across systems. Although there was a brief delay in implementing the search algorithm, the project was completed on time and demonstrated effective time management and technical problem-solving.

Future improvements include migrating to the updated Google Places API, enhancing the map's interactivity with custom markers and animations, and adding features such as saving favorite places and user authentication. Overall, DinnerWizard showcases practical use of full-stack web development tools and APIs to solve a real-world problem in a user-centric way.

I. Project / Problem Introduction

Searching for places to grab coffee, eat lunch, or visit Saturday night can be challenging when you've arrived in a new area. The need to review these options quickly and efficiently is common when people find themselves in new locations. Classic Map applications can be time-consuming and return too many results, when all the user needs is a simple, clean list of establishments. DinnerWizard addresses this problem by building a web-based tool that connects with Google Maps API services to search for restaurants, cafes, or bars within a user-specified radius.

First, the user enters their location using the search bar the connects to google Autocomplete function within the Google Maps API. Additionally, users can customize their search by using dropdown menus and selecting the type of place (restaurant, café, bar), the search radius by miles (though the system calculates this by meters). The system then sends that request through the Google Places API to return a list of nearby establishments that match.

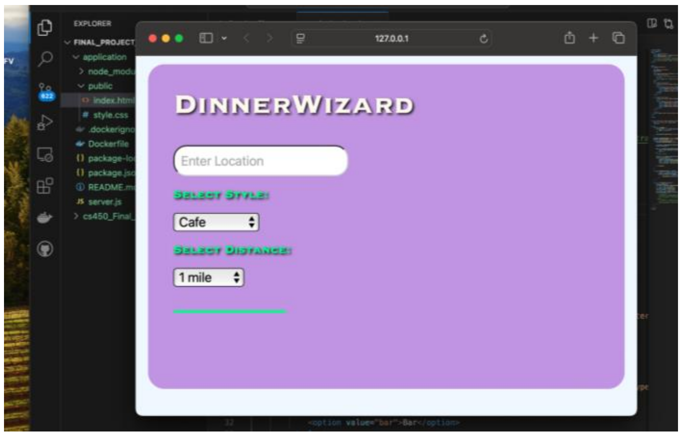
This is a JavaScript based application, so it requires a backend server to securely handle the Google API service requests. Both Node.js and Docker are implemented in DinnerWizard to ensure the application runs smoothly across different web browsers and that the data is handled efficiently. DinnerWizard enhances the search for new dining experiences and showcases the practical integration of Google Maps and Places APIs in a full-stack web environment.

II. Methods

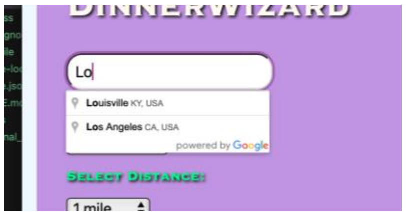
a. Project Idea

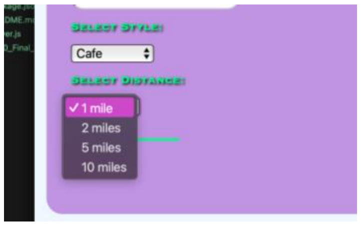
The primary goal of DinnerWizard is to create a user-friendly application that helps individuals quickly discover nearby restaurants based on their current location. I chose to develop this application because dining decisions are a daily need for people. I saw an opportunity to leverage geolocation and the Google Maps API to simplify this process. Additionally, this project broadened my skills in working with APIs, front-end development, and user interface design. DinnerWizard is targeted towards a broad audience, including locals looking for new dining options, travelers and new residents unfamiliar with their surroundings, or anyone wanting a quick and efficient way to find nearby restaurants.

Upon initialization, this is the application interface:

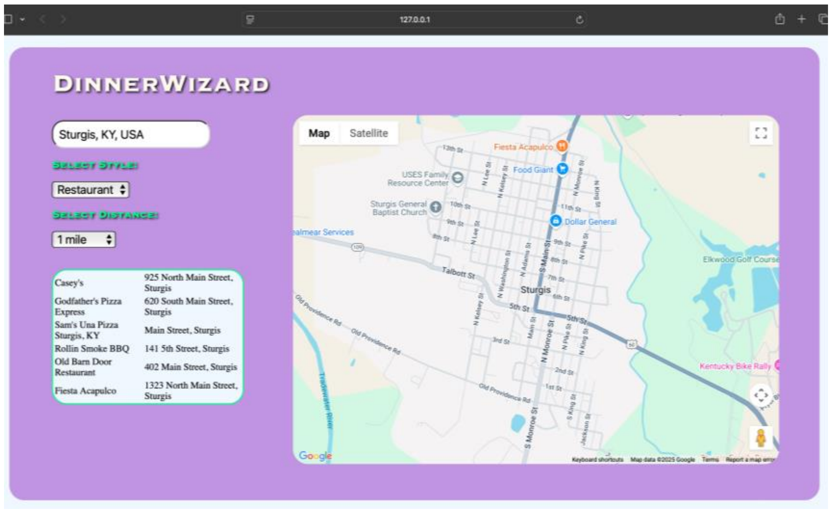


The predictive autocomplete feature begins the moment the user types the first letter into the search bar. The Map will update anytime the user inputs a new location through the search bar. General city names and specific addresses are accepted. Selecting the options from the drop-down bars will refine the search results, changing them will update the results in real time.





- Searching for restaurants in a 1-mile radius within Sturgis, Kentucky:



DinnerWizard

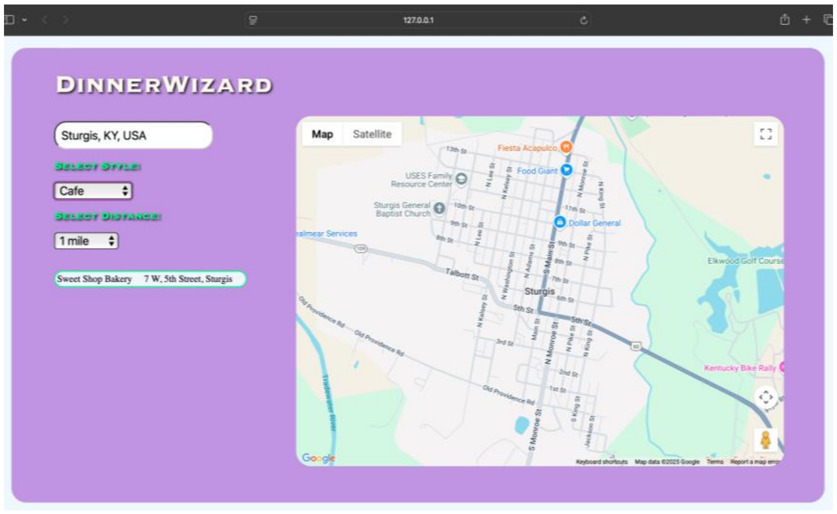
Sturgis, KY, USA

Select Style: Restaurant

Select Distance: 1 mile

Casey's	925 North Main Street, Sturgis
Godfather's Pizza Express	620 South Main Street, Sturgis
Sam's Una Pizza Sturgis, KY	Main Street, Sturgis
Rollin Smoke BBQ	141 5th Street, Sturgis
Old Barn Door Restaurant	402 Main Street, Sturgis
Fiesta Acapulco	1323 North Main Street, Sturgis

- Updating this request to search for cafés instead of restaurants:



DinnerWizard

Sturgis, KY, USA

Select Style: Cafe

Select Distance: 1 mile

Sweet Shop Bakery	7 W. 5th Street, Sturgis

b. *Tools/Materials*

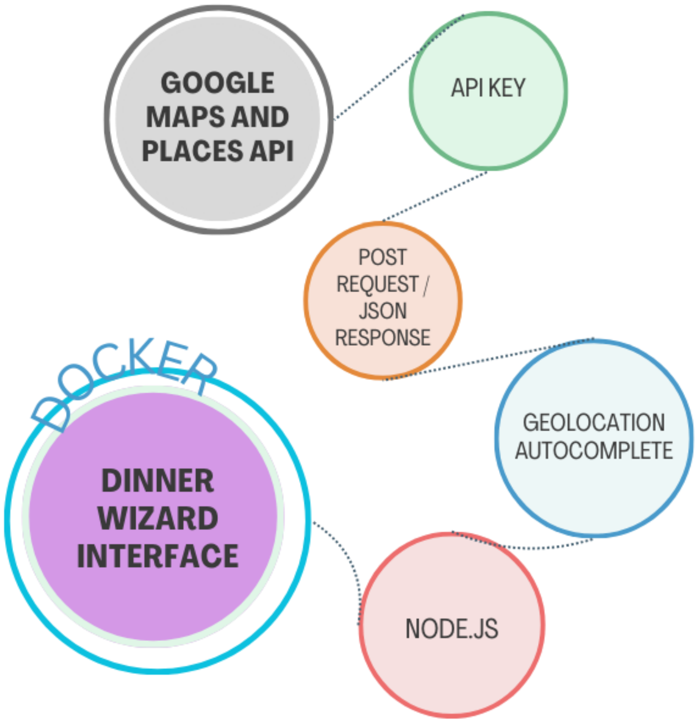
- Programming Languages – HTML, CSS, and JavaScript work together to make this application work. The front-end design is handled by HTML and CSS, while JavaScript handles functionality and API communication.
- VS Code - The primary IDE throughout the development process. It efficiently supported JavaScript, CSS, HTML files and more. This IDE has an integrated terminal allowing Node.js and Docker commands.
- Development Environment - The project was developed on a local machine using VS Code, with Docker installed to test containerized versions of the app. Node.js was used locally to serve and test the application, and browser-based debugging tools were used for front-end testing.
- Node.js - Used as the runtime environment for the server-side logic of the application. It enabled handling of API requests and processing of data in a scalable and efficient manner.
- Docker -Used to containerize the application, ensuring consistency across different environments and simplifying deployment. This allowed for the application to run with all dependencies packaged, reducing environment-related issues
- Google Maps API and Google Places API - These APIs were used to integrate location-based services into the application. The Google Maps API provided the map, while the Google Places API allowed for real-time restaurant searches based on user input and location data. Source code and usage examples were adapted from the Google Cloud Platform documentation.
- API Key – Google Developer Cloud and API service provided a unique key that allows DinnerWizard access to both API's used.

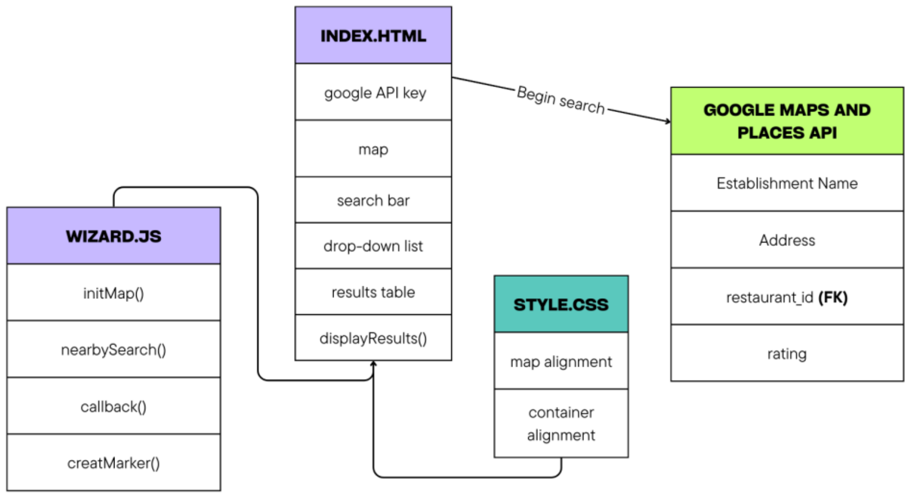
III. Results

a. *Project Results*

DinnerWizard successfully integrates Google Maps and Google Places APIs, using a unique API key that provides real-time results and displays them interactively as a list with a localized map. The web interface is clean and responsive, ensuring usability across various screen sizes. API calls to Google Maps and Places work consistently, returning accurate data for a variety of requests. With Docker, the application runs in a consistent environment. Node.js handles server-side processing, managing requests and serving responses efficiently. Overall, the project met its core objectives and delivered a working application that demonstrates effective use of third-party

APIs and full-stack web development practices. Below is a functionality flow chart and an ER diagram describing the services, functions and methods relationships to one another.





b. *Adherence to Timeline*

The development timeline mostly aligned with the proposed schedule. While the majority of tasks stayed on track, completing the search algorithm and handling JSON responses was delayed by about two weeks. This setback arose from underestimating the complexity of those specific tasks. Nevertheless, DinnerWizard was completed by the deadline. To improve future estimations, I will allocate additional buffer time for complex or unfamiliar tasks and incorporate flexibility into the timeline to accommodate unforeseen challenges.

c. Future Work

In the future, the first enhancement of the application will be migrating from the legacy Google Places API to the updated version, which offers improved performance and more detailed data. This will ensure compatibility with future updates and provide users with a richer experience. Additionally, the visual design and interactivity of the map can be elevated, such as adding custom map markers, animations, and filter controls directly on the map interface. These enhancements would make the application more engaging and user-friendly. Functions could be included to implement additional features like allowing users to save favorite restaurants, sort results by rating, and potentially include user authentication. Minor UI bugs and inconsistencies in how results are rendered across different screen sizes can be addressed to improve overall usability and responsiveness.

IV. Conclusion

DinnerWizard successfully addresses a common challenge faced by people navigating new areas. It offers a streamlined, user-friendly solution for discovering nearby dining options. Through the integration of Google Maps and Places APIs, and the use of Node.js and Docker, this project demonstrates both practical utility and technical proficiency. The application achieved its primary goals, including real-time search functionality, responsive design, and consistent performance across environments. Despite some minor timeline adjustments, all core features were implemented and tested successfully. DinnerWizard showcases effective use of API integration, front-end and back-end development, and containerization in a full-stack web project.

References

Building a Real-Time Location Tracking App with Python and Google Maps API in Python. (2025). Codemax.app. https://codemax.app/snippet/building-a-real-time-location-tracking-app-with-python-and-google-maps-api-in-python-3/

Chrisjshull, googlemaps. (2020). *js-api-loader/examples/index.html at main · googlemaps/js-api-loader*. GitHub. https://github.com/googlemaps/js-api-loader/blob/main/examples/index.html

Coding Shiksha. (2023, January 15). *Javascript Google Places API Project to Plot Nearby Places on Maps Using Autocomplete Location*. YouTube. https://www.youtube.com/watch?v=aFelEcWBqII

Geolocation: Displaying User or Device Position on Maps | Maps JavaScript API. (n.d.). Google for Developers. https://developers.google.com/maps/documentation/javascript/geolocation#maps_map_geolocation-javascript

Merrill, K (2019) *Blog: Three ways to add a map implementation to your app and when to use each*. (n.d.). Google Maps Platform. https://mapsplatform.google.com/resources/blog/three-ways-add-map-implementation- your-app-and-when-use-each/

Om Raghuvanshi. (2023, August 9). Unveiling Your Location: A Python Guide to Retrieve Current GPS Coordinates. Medium. https://medium.com/@omraghuvanshi1010/unveiling-your-location-a-python-guide-to-retrieve-current-gps-coordinates-d1ba282b44fd

Place Autocomplete | Places SDK for Android. (n.d.). Google Developers. https://developers.google.com/maps/documentation/places/android-sdk/autocomplete

Select Current Place and Show Details on a Map. (2020). Google for Developers. https://developers.google.com/maps/documentation/places/android-sdk/current-place-tutorial

vishuvaishnav. (2024). GitHub - vishuvaishnav/GPS_tracker_with_Python: The GPS Locator Using Python project. GitHub. https://github.com/VISHUVAISHNAV/GPS_tracker_with_Python

Yadav, A. (2023, March 30). Distance Between Two Geo-Locations in Python - AskPython. AskPython. https://www.askpython.com/python/examples/find-distance-between-two-geo-locations.