RecipEZ Final Capstone Report

Cooper Hickmott

Table of Contents

Introduction

How do you share recipes with your friends and family? You might write them down on paper and exchange them in person, maybe you have a shared google drive storing them all… how much nicer would it be if you could use a website for that! RecipEZ will provide a place for users to create, share and view their friend's food and drink recipes from all around the world!

Users will login to their account using their Gmail, where they will then be able to upload individualized food/drink recipes. A user's profile page will act as their Cookbook where they can browse their own recipes. Users will be able to view each other's cookbooks through following/follower requests.

As a computer science major in the College of Engineering, I knew I wanted to develop a web application for my capstone project. Thus, I chose to create RecipEZ.

Outline

From my CSE education, I have experienced the beginning and middle stages of the software development lifecycle many times (requirements definition/initial plannings to developing and debugging the code). An area of the software development lifecycle I was lacking experience in, however, is the final stage of deploying the software service for production use. This report will investigate the different strategies and technologies available for deploying and hosting web services on the internet as well as determine the desirable option for hosting RecipEZ.

Additionally, this report will explain the tools and thought process I used for implementing both the front end (user interface) and the back end (data storing and fetching logic) of the website.

Web Hosting Investigation

RecipEZ requires reliable performance, meaning the website does not crash due to server/network errors. An additional requirement for RecipEZ is the hosting service being low cost, as this project is self-funded. RecipEZ should be secure; the website does not deal with sensitive data per say, but having a secure server is always best practice. An additional desire of recipe is scalability, meaning the website performs reliably as the user base grows and shrinks.

Upon investigation into the deployment of web services for production use, I found four distinct hosting strategies to choose from: shared, dedicated server, virtual private server and serverless. Shared web hosting is a hosting practice where multiple web services run on a single server. Each user is given a certain portion of the servers resources and it is one of the most used forms of web hosting. Dedicated server hosting consists of a web service running on a server that is only hosting a single website/application. The server is completely dedicated to the person/service that owns it, so there is no sharing of computing resources with other applications. Like shared hosting, virtual private server hosting involves many different services running on a single server machine; however, resources will not be shared in a first come first serve format. Instead, specific resources are assigned to a specific web service, providing certain advantages over shared hosting while not being too expensive ($10-$20 more per month than shared). Finally, serverless architectures allow one to build and deploy web applications and services without having to manage the physical server infrastructure. The programmer is responsible for defining which services they are using, then the actual implementation of hosting is abstracted away.

Shared Web Hosting

Shared web hosting is when a single server is the host for multiple websites. Each user is allocated a portion of the server's resources like disc space, databases, and operating system accounts. In an article titled 'Advantages and Disadvantages of Shared Web Hosting' written by the company Monster Host, the author discusses some of the advantages and disadvantages that are encountered when using shared web hosting. The article explains how "Shared hosting is less expensive than other types of hosting… because the physical server is being shared by many people" (Monster Host, 2019). When a developer uses shared web hosting, they outsource the heavy lifting aspects of setting up and maintaining a website to the service provider they choose. This is beneficial as lots of developers are not interested in the heavy lifting aspect of maintaining a server for a website. Shared hosting is also efficient. A website’s performance is determined by several factors such as disk space and bandwidth. Shared hosting gives users the option to buy more resources without having to go through a technical support team. Thus, developers can scale and optimize their website as desired. The final advantage discussed in the article is the ability to customize a website to suit one’s needs and desires through a control panel interface provided by most web hosting providers.

However, shared hosting comes with its own set of disadvantages. The main disadvantage for shared web hosting is security concerns. Hundreds, sometimes thousands of websites are using the same physical server. Thus, a cyber-attack can affect all those websites relying on this shared server. The issue is further propagated with the restriction of installing antivirus or firewall software for your website, as shared hosting provides you with a platform on a server, not the server itself. An additional disadvantage of shared hosting is that the resources are limited. A physical server does not have unlimited resources, thus each website using the server is allocated a set amount of those resources. If one website gets a lot of traffic, it will draw lots of the server’s resources and other website’s performance can be negatively impacted, meaning users would have difficulty accessing the application or the system could crash.

The affordability and convenience provided by shared web hosting are its main advantages, while security concerns and performance unreliability are its main disadvantages. Shared web hosting meets RecipEZ’ need of being low cost but fails to meet RecipEZ’ hosting need of reliable performance.

Dedicated Web Server Hosting

Dedicated web server hosting uses a server that hosts a single website/application. This server is completely committed to the individual who owns it and does not share any computing resources with another site such as bandwidth and processing power. In an article titled “Pros & Cons of Dedicated Server Hosting: Is It Worth To Get One?”, the author describes the advantages and disadvantages encountered when using dedicated web server hosting. The article begins by declaring “One of the advantages of dedicated hosting is no security threat from neighboring sites as they don’t have (any neighbors)” (Yadav, 2022). When comparing dedicated hosting to shared, it is clear dedicated has an advantage when it comes to security threats from neighboring sites (sites on the same server), as there are not any neighboring sites on a dedicated server. Additionally, dedicated servers allow for extensive control over the server as the user is paying for sole ownership of the resource. Users are free to install any software or applications they desire to fulfill their business needs or current demand of work. The final major advantage for dedicated servers is performance reliability. Unlike shared hosting, traffic fluctuations do not impact your website performance, as well as not having to share resources with any other website.

However, dedicated web server hosting comes with its own set of disadvantages. Dedicated servers are expensive, as they are exclusively owned by an individual. Additionally, using a dedicated server requires the technical knowledge to set up, run and manage the server as these responsibilities now fall on the developer. Finally, dedicated servers are not meant to scale. The dedicated server is already a large machine, so if your web application needs to grow, it may require moving to a larger server rather than simply allocating more computing resources.

The security and reliable performance provided by dedicated web hosting are its main advantages, while the high cost and technical knowledge required are its main disadvantages. Dedicated web hosting meets RecipEZ' need of having reliable performance but fails to meet RecipEZ' hosting need of low cost.

Virtual Private Server Hosting

Virtual private server (VPS) hosting, like shared web hosting, consists of multiple websites using a single physical server. However, unlike shared web hosting, resources are not allocated in a first come first serve format. Instead, specific resources are assigned to an account, and that account can use them however they desire. In an article titled "What is VPS? Pros & Cons of VPS hosting", the author describes how "virtual private server hosting offers a middle option (between shared and dedicated) with some of the benefits of dedicated hosting without the high cost" (Ehinger, 2020). VPS hosting can be either managed: the hosting company handles all server related issues, or unmanaged: the responsibility of server administration is handled by the developer. One of the advantages to VPS hosting is increased security compared to shared hosting. Users can secure their partition of the server however they choose, and their application will not be in a vulnerable position due to someone else's actions on the server. Additionally, VPS hosting allows for scalability, as an individual can upgrade to a larger set of resources. Finally, VPS hosting offers reliable performance, as web applications are allocated a set amount of resources. VPS does not have higher performance than dedicated server hosting though. Some of the cons for VPS hosting are the cost, limited resources, and security concerns as described in the previous paragraphs. These cons are expected however, as VPS is the middle ground between shared and dedicated hosting, thus it will fall in between for cost, resource availability and security protocols.

The security customizability, reliable performance and scalability provided by VPS web hosting are its main advantages, while the medium/high cost, limited resources and neighboring site security concerns are its main disadvantages. VPS web hosting meets RecipEZ' need of having reliable performance but fails to meet RecipEZ' hosting need of low cost.

Serverless Hosting

Serverless hosting architectures allow one to build and deploy web applications and services without having to manage the physical server infrastructure they run on. In a book titled "Hands-On Serverless Applications with Kotlin" written by Hardik Trivedi and Ameya Kulkarni, the authors describe serverless computing as "allowing (one) to build and run applications and services without thinking about servers… everything required to run and scale your application with high availability is handled for you" (Kulkarni, 2018). This paradigm allows developers to focus more on the front-end functionality of a website or app, without worrying about back-end administration functionality. Developers are still required to write all the logic they require for their backend system; however, they do not manage how or where this code is executed on physical servers. In an article titled "Serverless Computing: Uses, Advantages, and Disadvantages", the author defines four attributes all true serverless applications have: "Variable in price, Self-maintaining, Scalability and Reliability" (Okta, 2019). Variable in price refers to serverless architectures only charging the user for what resources they consume (when the code is triggered by an event request), unlike a dedicated server which might charge by the hour. Self-maintainability refers to the user not having to worry about server maintenance. Scalability refers to how serverless architectures are designed to scale up and down continuously with no client programming intervention. Finally, this technology is reliable as the products have built-in high availability and fault tolerance features. Some of the disadvantages associated with serverless technologies are Security/Privacy. You are entrusting another company to manage the resources your application uses; thus, the security and privacy protocols used are inherently up to that company. Additionally, it can be technically challenging to debug serverless architectures as there are multiple distinct places to look for a bug.

The variability in cost, reliable performance, scalability, and self-maintainability provided by serverless architectures for web hosting are its main advantages, while outsourcing the security and data privacy implementations to a third party are its main disadvantages. Serverless web hosting meets RecipEZ' need of having reliable performance and meets RecipEZ' hosting need of low cost. Additionally, serverless web hosting meets RecipEZ' desire of scalability.

Web Host Investigation Results

Below is a table summarizing the hosting options discussed previously. Some of the key columns to note are Cost, Performance, Security Provided and Scalability. As stated before, RecipEZ requires a web hosting option with low cost and high performance. Additionally, RecipEZ desires a scalable and secure platform.

Type of Hosting	Cost	Performance	Security Provided	Scalability	Technical Knowledge Required	Customizability
Shared	Low	Low/Medium	Low	Low	Low	High
Virtual Private Server	Medium	Medium	Medium	High	Low	High
Dedicated	High	High	High	Low	High	High
Serverless	Depends on resource usage	High	High	High	High	High

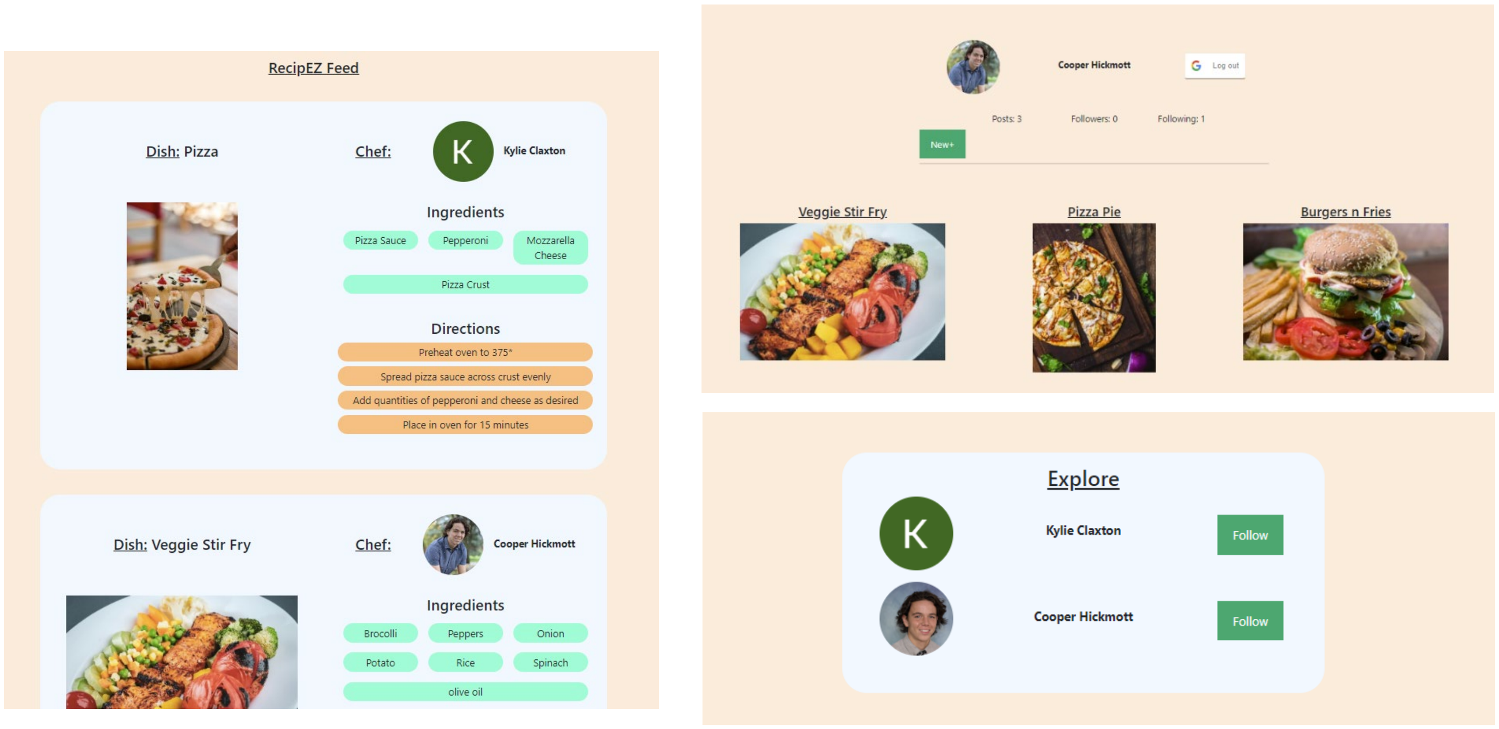
Table 1: Serverless Hosting Desirable Option for RecipEZ

Thus, RecipEZ will be implemented using Serverless technologies, as this option provides reliable performance while being low cost. At this point, the next step for RecipEZ is developing the website.

Front-End Implementation

User Interface Workflow

RecipEZ has three main pages: feed, explore and profile. The feed page acts at the "homepage" for RecipEZ. Here, recipes posted by the logged in user or users that they follow will appear, that they can then scroll and view as desired. The explore page will show other users on the website that the logged in user is not currently following, providing them with a clickable button to follow said users. Finally, the profile page will display user information, as well as the titles and images for recipes uploaded by that user. A preview of these pages can be seen in the images below (Feed – left, Profile – top right, Explore – bottom right).



RecipEZ Feed

Dish: Pizza
Chef: Kylie Claxton
Ingredients: Pizza Sauce, Pepperoni, Mozzarella Cheese, Pizza Crust
Directions:
Preheat oven to 375°
Spread pizza sauce across crust evenly
Add quantities of pepperoni and cheese as desired
Place in oven for 15 minutes

Dish: Veggie Stir Fry
Chef: Cooper Hickmott
Ingredients: Brocolli, Peppers, Onion, Potato, Rice, Spinach, olive oil

Cooper Hickmott
Log out
Posts: 3 Followers: 0 Following: 1
New+
Veggie Stir Fry
Pizza Pie
Burgers n Fries

Explore
Kylie Claxton Follow
Cooper Hickmott Follow

Figure 1: RecipEZ User Interface

Technologies Used and Implementation Logic

The front-end user interface for RecipEZ was developed using React with Redux-Sagas. React is a JavaScript library for building user interfaces based on reusable components. Redux is a JavaScript library for managing and centralizing application state. Redux-Sagas is a JavaScript middleware library that allows the redux store to interact with resources outside of itself asynchronously. The Redux store is a JavaScript object that contains the requested data for the web application. User interface components are able to "subscribe" to the store, re-rendering whenever there is an update made to the store. This allows the developer to have fine grain control over how the components on the website interact with each other and the user. All data fetching and manipulation logic makes use of redux store and redux-sagas middleware.

I chose to store session information in local browser storage. This allows a logged in user to close and reopen their browser without having to login to the website again.

To deploy the front-end so it can be publicly accessible via the internet, I used AWS Amplify serverless service. This service prompted me to simply connect the GitHub repository containing all the RecipEZ code, and then upon any new commits to the repo, Amplify updates and redeploys your code to a server. Thus, users are able to access your website at a given URL.

Back-End Implementation

Technologies Used

The back-end system for RecipEZ was implemented entirely using AWS serverless technologies. I used a total of five services, that when combined together form a complete back-end for a website. The services used were API Gateway, Lambda, DynamoDB, S3, and IAM. API Gateway is a service for developers to manage RESTful APIs. This service handles all tasks involved with accepting and processing HTTP requests, as well as provides a clean user interface to see the various methods you have defined for a given resource. Lambda is a service where developers organize code into functions, that are then invoked from other AWS services. An example would be API Gateway calling a Lambda function on an HTTP request event, where the lambda function handles the necessary logic for that specific endpoint. DynamoDB is a NoSQL database service that provides fast and predictable performance, with seamless scalability. DynamoDB record entries are in a <Key, Value> format, thus querying logic is handled in lambda functions using pure JavaScript. S3 is a cloud object storage service that provides fast and predictable performance with seamless scalability. I used S3 to store images uploaded by users for their recipes. Finally, IAM is a service that helps secure control access to AWS resources.

Implementation Logic

I chose to outsource session management with Google Sign-In, as I trust their security protocols more than any system I would have developed myself for this project. Sign-In provides a session token to the user, which lambda functions are then able to use to perform session management logic with DynamoDB tables. All API requests are passed through an *Authorizer* function, which verifies an authentic session exists for the given request in the database. All API request logic is handled in Node.js lambda functions. These functions access DynamoDB table (key, value) pairs to query desired request information. Finally, I implemented an independent file upload API using API Gateway, Lambda, and S3 AWS services to store user uploaded images. CI/CD was handled through GitHub actions, where pushes to the Main branch triggered the serverless API to redeploy.

Conclusion

The main goal/focus of this capstone project, besides creating a fun recipe sharing website, was exploring and learning how to use serverless technologies to implement a websites backend. Serverless hosting architectures allow one to build and deploy web applications without having to manage the physical server infrastructure they run on. Developers are still required to write all the logic they require for their backend system; however, they do not need to worry about how or where this code is executed on physical servers as that is the point of cloud computing. I have found that all serverless technologies have four main properties: Variable in price, Self-maintaining, Scalability and Reliability. Variable in price refers to serverless architectures only charging the user for what resources they consume (when the code is triggered by an event request), unlike a dedicated server which might charge by the hour. Self-maintainability refers to the user not having to worry about server maintenance. Scalability refers to how serverless architectures are designed to scale up and down continuously with no client programming intervention. Finally, this technology is reliable as the products have built-in high availability and fault tolerance features. Throughout this project, I have found the AWS serverless technology to be extremely pleasant to work with, and I am confident I will use serverless architectures in the future to deploy web applications.

References

Ehinger, B. (2020, October). *What is VPS? Pros & Cons of VPS Hosting*. Retrieved from hostingjournalist: https://hostingjournalist.com/what-is-vps-pros-cons-of-vps-hosting/

Kulkarni, H. T. (2018). *Applications with Kotlin - Develop scalable and cost-effective web applications useing AWS Lambda and Kotlin.* Packt Publishing.

Monster Host. (2019). *Advantages And Disadvantages Of Shared Web Hosting*. Retrieved from Monster Host: https://monsterhost.com/advantages-and-disadvantages-of-shared-hosting/

Okta. (2019). *Serverless Computing: Uses, Advantages, and Disadvantages*. Retrieved from okta: https://www.okta.com/identity-101/serverless-computing/

Yadav, S. (2022, September). *Pros and Cons of Dedicated Web Server Hosting*. Retrieved from netforchoice.com: https://www.netforchoice.com/blog/pros-and-cons-of-dedicated-server-hosting/