CSCI 5410

Serverless Data Processing

Activity - 1

Name: Shreya Kapoor

Banner ID: B00957587

Deployed URL: https://foodrecipe-manager.netlify.app/

Github URL: https://github.com/shreyakapoor08/Recipe-Management-Serverless

Table of Contents

Aim of the task

The Recipe Management application is designed to help users easily manage their favorite recipes. Built with ReactJS for the frontend, AWS Lambda for the backend, and DynamoDB for user data storage and S3 for images, this application allows users to add, edit, delete, and view recipes. Users can also upload images for each recipe, which are stored in an S3 bucket and displayed in the application.

The application has a simple interface with modals for adding and editing recipes. It provides clear messages for successful actions and errors, making it easy to use. A detailed view modal lets users see larger images and more details about each recipe. AWS Lambda functions handle the main operations (create, read, update, delete), ensuring the app runs efficiently and can scale as needed [1]. This application uses modern web technologies and cloud services to offer a reliable and easy-to-use recipe management tool.

Application Overview

The users can perform the following actions on the website.

1. **Viewing Recipes:**
   - The user opens the app and sees a list of their recipes.
   - Each recipe entry displays a title and a small image.

2. **Adding a New Recipe:**
   - The user clicks the "Add" button.
   - A modal window opens where the user can enter the recipe title and upload an image.
   - Upon submitting, the recipe image is uploaded to S3 and the dish name and S3 object are stored in DynamoDB.
   - The new recipe is then displayed in the list.

3. **Editing an Existing Recipe:**
   - The user clicks the "Edit" button next to a recipe.
   - A modal window opens with the current title and image.
   - The user updates the title and/or uploads a new image.
   - Upon saving, the updates are sent to DynamoDB, and if a new recipe image is uploaded, it replaces the old one in S3.
   - The updated recipe is then displayed in the list.

4. **Deleting a Recipe:**
   - The user clicks the "Delete" button next to a recipe.
   - A confirmation prompt appears.
   - Upon confirming, the dish is removed from DynamoDB and the recipe image is deleted from S3.
   - The recipe is removed from the list.

5. **Viewing Recipe Details:**
    - The user clicks on a recipe title or image.
    - A modal window opens showing a larger image and the full recipe details for a better view.

Thought Process:

The idea for the Recipe Management app came from a common problem: forgetting which recipe was used for a favorite dish. As someone who enjoys cooking and often looks for new recipes online, I realized the need for a way to store and organize recipes in one place. The app would make it easy to find and recreate dishes without searching through multiple websites.

Choice of Technologies

1. **ReactJS for Frontend**: ReactJS was chosen because it helps build dynamic and interactive user interfaces. It makes the app responsive and easy to use, which is important for tasks like adding and editing recipes.

2. **AWS Lambda for Backend**: AWS Lambda was chosen because it's cost-effective and can handle a large number of users without slowing down. It manages backend tasks efficiently, like saving and retrieving recipes [1].

3. **DynamoDB for Data Storage**: DynamoDB was chosen because it's fast and reliable. It stores recipe data securely and makes it easy to access when needed [1].

4. **S3 for Image Storage**: Amazon S3 was chosen because it's safe and reliable for storing images. It ensures that recipe images are stored securely and can be easily accessed.

Development Thought Process

1. **User Engagement**:
    - **Interactive Design**: The app uses pop-up windows for adding and editing recipes, so users stay focused on one task at a time.
    - **Visual Appeal**: Allowing users to add images to their recipes makes the app more interesting and enjoyable to use.

2. **Enhanced User Experience**:
    - **Feedback and Notifications**: The app gives clear messages when actions like adding or editing recipes are successful or if there's an issue.
    - **Smooth Interactions**: Loading indicators show when the app is working, and messages disappear after a few seconds to keep the app running smoothly.

3. **Future Enhancements**:
    - **Scalability and Extensibility**: The app is built to grow and add new features in the future, like sharing recipes with friends or creating groups for recipe sharing.

Challenges Faced

When working on the Recipe Management app, I encountered a challenge with uploading images. Initially, only JPEG images could be uploaded because the S3 bucket was set up to accept only "image/jpeg" files. This meant that other image formats, like PNG, couldn't be uploaded. To fix this, I needed to make the app able to handle different image types, so I used something called image MIME types [2]. The problem was that the app could only handle one type of image, so users couldn't upload images in formats other than JPEG. To solve this, I added a feature that lets the app detect the type of image being uploaded. This way, it can work with different formats like JPEG or PNG. Sample snippet of code from the application using image mime is below [2].

```
# Handling image upload if 'image' and 'image_mime' are provided in the request
    if 'image' in body and 'image_mime' in body:
        image_data = base64.b64decode(body['image'])

        # Generating the S3 key for the image using the item ID and image MIME type
        s3_key = f"{item_id}.{body['image_mime'].split('/')[-1]}"
```

Implementation Strategy

To make this work, I changed how the app's backend handles image uploads. I added code to check the type of each image that's uploaded. If it's a JPEG or PNG, the app knows what to do with it. By adding support for different image formats, the app became more useful for users. They can now upload images in formats other than JPEG, which makes managing recipes easier.

Output Screenshots:

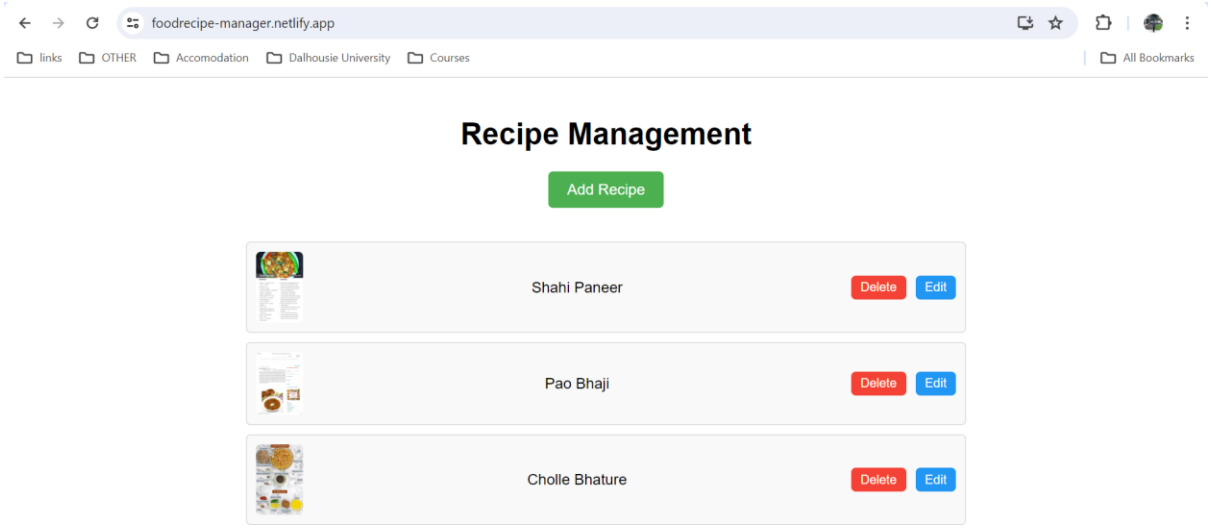

Figure 1: Recipes List

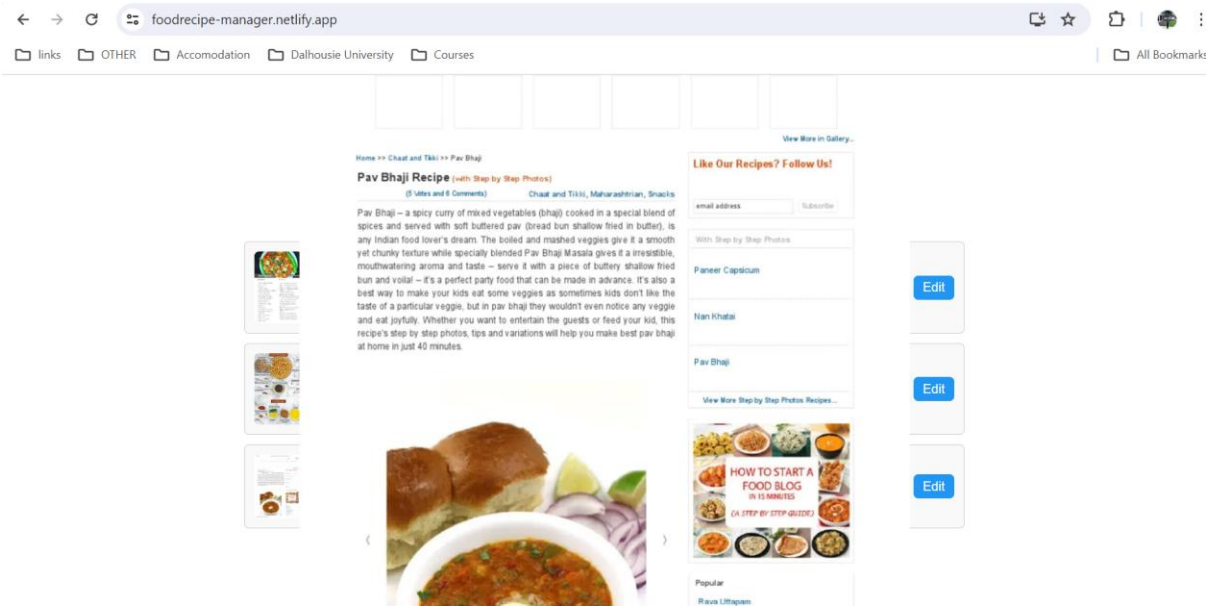

Figure 2: Recipe Preview

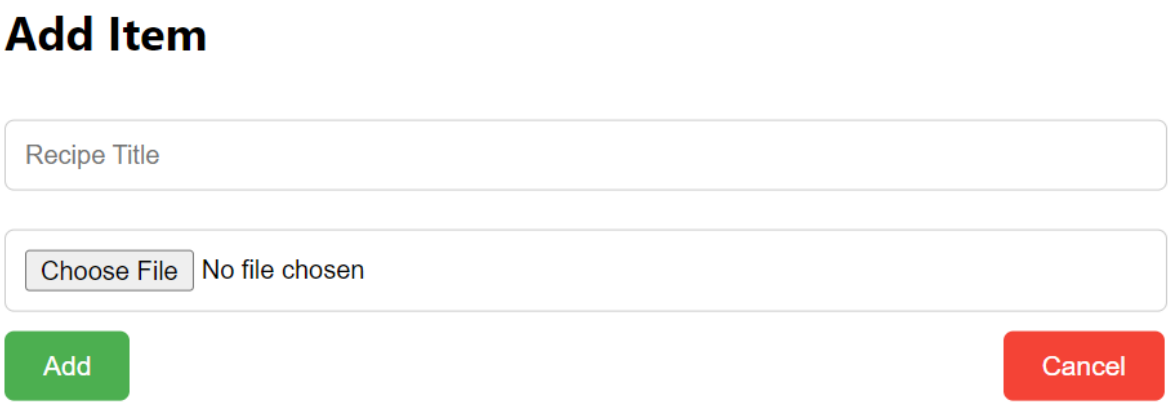

Figure 3: Add new dish

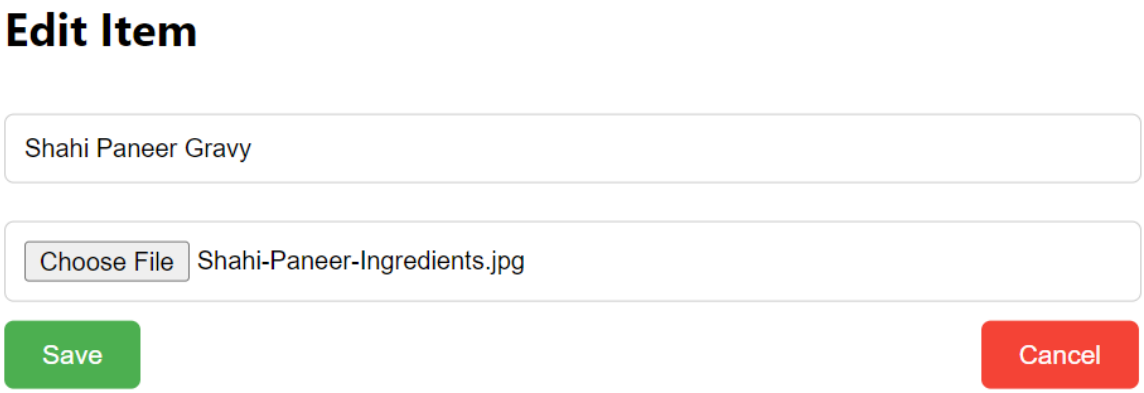

Figure 4: Edit New Dish

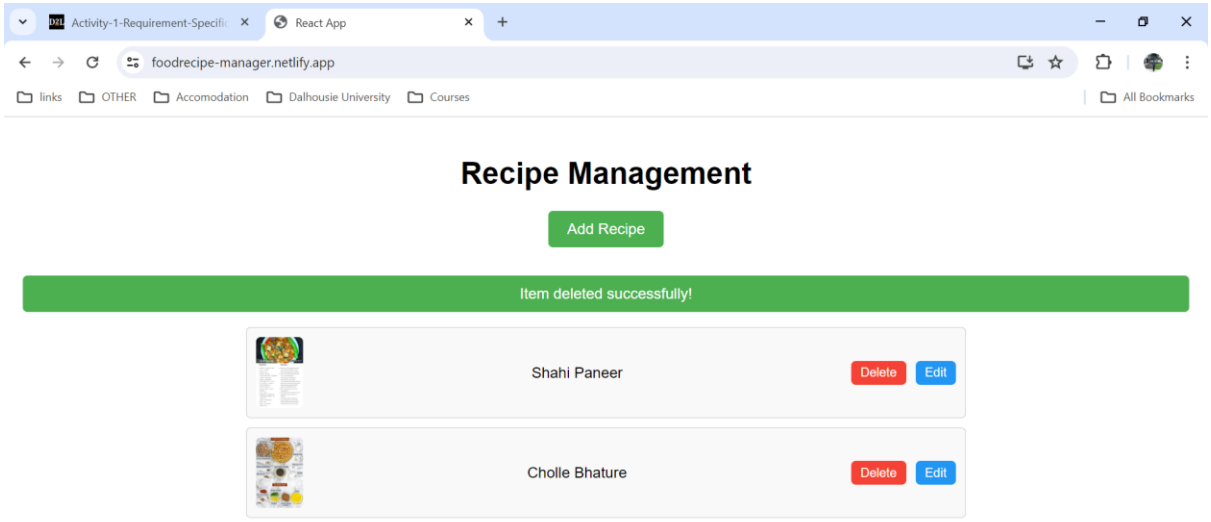

Figure 5: Delete Dish

References:

[1] M. Jayashanka, “Building a Serverless CRUD API with AWS Lambda, DynamoDB, and IAM policies,” *Medium*, Sep 30, 2023. [Online]. Available: https://medium.com/@madhurajayashanka/building-a-serverless-crud-api-with-aws-lambda-dynamodb-and-iam-policies-40fa655787a. [Accessed: June 05, 2024].

[2] “MIME types (IANA media types),” *MDN Web Docs*. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types [Accessed: June 05, 2024].