Table of Contents

Contribution

Jia Yin: 50%. Backend implementation.
Chuxuan Chen: 50%. Front end interface.

Links

URL of my website: http://recipe-finder.s3-website-us-east-1.amazonaws.com

Summary

This is a website that allows users to find recipes based on the ingredients they have, and they can also store their favourite recipes on their accounts.

Introduction

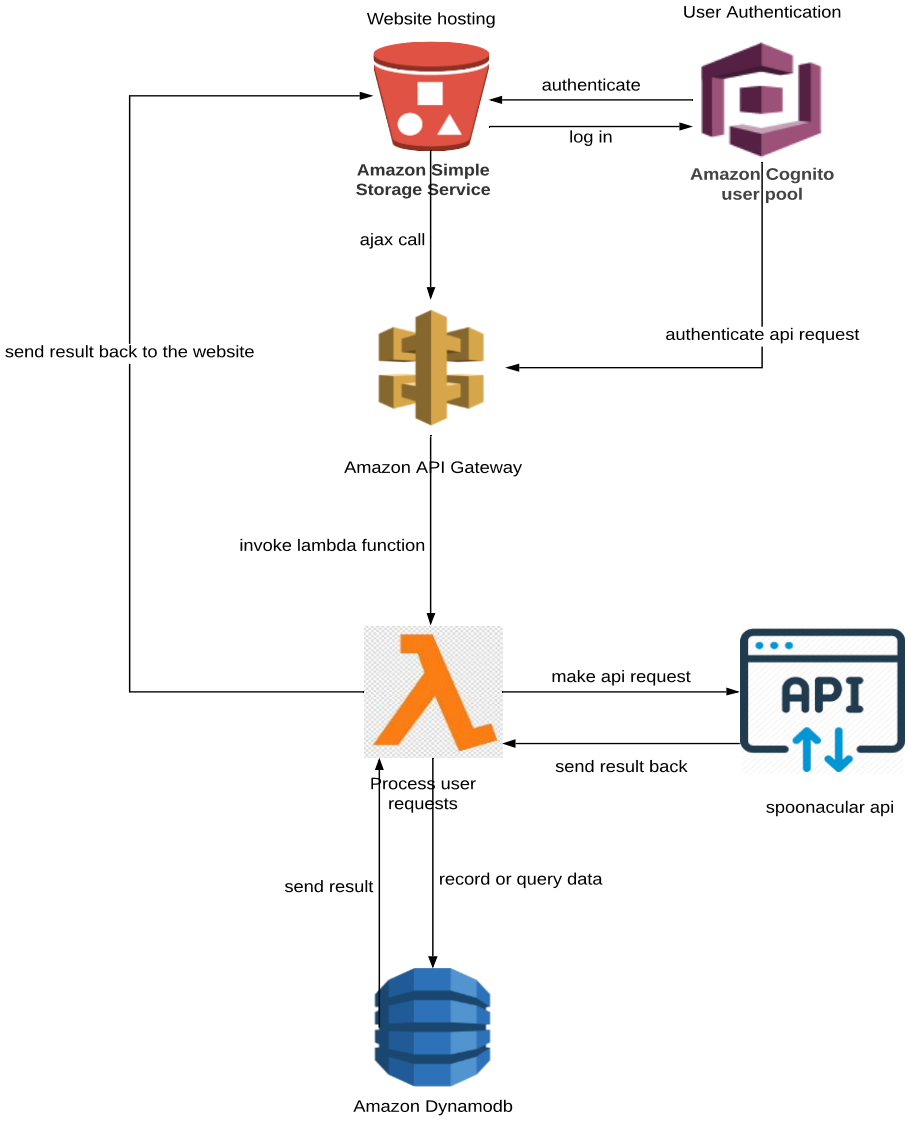
I cook myself most of the time, so my fridge is always filled with all kinds food and ingredients. But with all these ingredients, I often take a while to decide what to make. I wanted an app to make my life easier. So, I got the idea to make an app which can recommend recipes based on the ingredients I have. And I also wanted to store the recipes I like for later reference. Next time if I'm not sure what to make, I can just look for it in the favourites list, or search for the recipe by ingredients.

The website required the user to sign up using their emails and passwords. Once they are logged in, they can find recipes based on the ingredients they have, and if they've found something they like, they can add them to their favourite list.

Related Work

Taste.com does a similar thing as our app. Users can search for recipes by dish, keyword or ingredients. Myrecipes.com offers users the choice to search recipes by keywords.

Software Architecture



Amazon S3: Amazon S3 is used to deploy the website. Elastic beanstalk was considered for this purpose, but my educate account didn't support Elastic beanstalk. EC2 was also considered, but the maintenance for EC2 instance adds much complexity and this simple website doesn't require the scalability provided by EC2. Since the scale of the website is not that big, Amazon Lightsail and AWS Amplify Console were also considered, but again, the educate account don't support these services. S3 should be enough for this simple website.

Cognito User Pool: makes it easier to authenticate and manage users. I planned to integrate Amazon Simple Email Service with Cognito User Pool to send email verification emails. Compared to the built-in email delivery service in Cognito, Amazon Simple Email Service can manage higher email delivery volume, but it turned out the educate account doesn't support SES.

Amazon API Gateway: it's used to create resources and methods to associate Lambda function with the website. Amazon API Gateway provides security to the exposed API endpoint so only authenticated users can pass through to the lambda function. This was

accomplished by adding a Cognito authorizer to the API. Only the users in the Cognito user pool can access the API.

Lambda function: Lambda function makes the application serverless, so there's much less maintenance. It was used to process user requests and send the result back to the website.

Spoonacular API: All the recipes' information were obtained from this API. It has 360K+ recipes and extensive information about them. You can query recipes by ingredients, nutrients, and keywords etc. In this website, find recipes by ingredients is used.

DynamoDB: it's used to store users' favourite recipes, so they can be retrieved later.

Flow

The basic flow is that user signs up using email and password. Cognito records users' information in Cognito user pools. User then logs in using email and password; once the user was authenticated By Cognito user pool, the website redirects to the home page. The user then enters the ingredients, and click search. Javascript then makes an ajax call to the Amazon API Gateway's endpoint, and API uses the authentication tokens to verify the user. Once the api request was authenticated, users' input were sent to the lambda function. Based on the user's input, Lambda then sends an request to Spoonacular and receives the result. The result is sent back to the website and displayed.

When the user clicked the "add to favourites" button. One extra step is added to the previous flow. In the lambda function, the "favourited" recipe will be recorded in the Dynamodb table. And when user wants to view the "favourites" list, Lambda function queries the table and get all the user's favourite recipes.

Developer Manual

Create a amazon S3 bucket for website hosting

1. Create a bucket for web resources named "recipe-finder" leaving all settings as default. The instructions are in this link.

2. Upload web resources including HTML, CSS and Javascript files into the bucket following the instructions here.

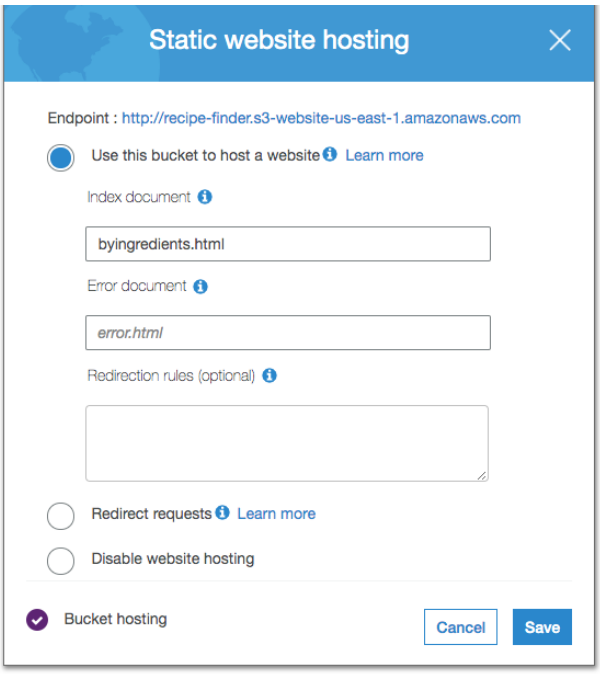
3. Add a bucket policy to allow users to view the website.

    1. Select the "recipe-finder" bucket
    2. Choose the "permissions" tab.
    3. Select "edit" and uncheck "Block public access to buckets and objects granted through *new* public bucket policies" and "Block public and cross-account access to buckets and objects through any public bucket policies"
    4. Switch to the "bucket policy " tab, and paste
    {

```
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Principal": "*",
    "Action": "s3:GetObject",
    "Resource": "arn:aws:s3:::recipe-finder/*"
  }
]
}
```

Into the editor. Click “save”.

5. Go to the “properties” tab
6. Click on “static web hosting”
7. Choose “Use this bucket to host a website”
8. Enter “byingredients.html” for the index document to set the home page of the website to “byingredients.html”



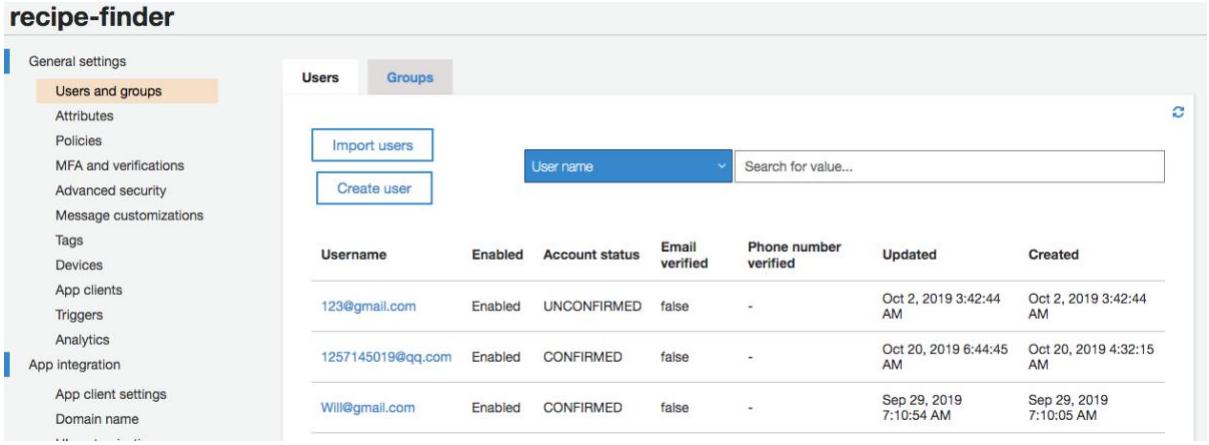
Create an Amazon Cognito user pool

1. Choose manage your user pools on the Cognito console
2. Select create a user pool
3. Enter a name for the user pool
4. Then click on “create user pool”
5. Add an app client to the user pool

1. Choose app clients from the left navigation bar on the console page
2. Select add an app client
3. Give the client name “recipe-finder”
4. Uncheck the Generate client secret
5. Select “create app client”
6. Add the following config.js file to the js directory

```
window._config = {
    cognito: {
        userPoolId: 'us-east-1_UZVGgKrkb', // e.g. us-east-2_uXboG5pAb
        userPoolClientId: '6hqkfjac9ojkfibrvk65ttgph6', // e.g. 25ddkmj4v6hfsfvruhpf
i7n4hv
        region: 'us-east-1' // e.g. us-east-2
    },
    api: {
        invokeUrl: '' // e.g. https://rc7nyt4tql.execute-api.us-west-
2.amazonaws.com/prod,
    }
};
```

7. Add auth.js file to the js directory. Jquery code inside will handle user sign-up and sign-in using Amazon Cognito SDK.
8. To test if the code works, go to the Cognito console after registering a user on the website.
9. Select “users and groups”
10. Check if the registered user is on the user list



Create a Dynamodb table

1. Go to the dynamodb console and choose “create table”
2. Enter “RecipeFinder” for table name, “UserName” for the partition key, and “RecipeID” for the sort key
3. Choose string for the partition key type, and number for the sort key type
4. Select the “use default settings” box and then select “create”
5. The resulting table should look like this

Table details

Table name	RecipeFinder
Primary partition key	UserName (String)
Primary sort key	RecipeID (Number)
Point-in-time recovery	DISABLED Enable
Encryption Type	DEFAULT Manage Encryption
KMS Master Key ARN	Not Applicable
Time to live attribute	DISABLED Manage TTL
Table status	Active
Creation date	October 6, 2019 at 4:09:41 PM UTC+11
Read/write capacity mode	Provisioned
Last change to on-demand mode	-
Provisioned read capacity units	5 (Auto Scaling Error)
Provisioned write capacity units	5 (Auto Scaling Error)
Last decrease time	-
Last increase time	-
Storage size (in bytes)	6.06 KB
Item count	18 Manage live count
Region	US East (N. Virginia)
Amazon Resource Name (ARN)	arn:aws:dynamodb:us-east-1:522511800038:table/RecipeFinder

Create a lambda function

1. Create an IAM role for the lambda function
    1. Choose “roles” on the IAM console and choose “create new rolw”
    2. Select “Lambda” for the role type
    3. Select “next:permissions”
    4. Add “AWSLambdaBasicExecutionRole” and all the policies related to Dynamodb in the box
    5. Enter “RecipeFinderLambda” for the role name
    6. Click “create role”
2. Create the lambda function
    1. Go to the lambda console and choose “create function”
    2. Select “author from scratch”
    3. Enter “find Recipe” for the function name
    4. Choose “Node.js 10.x” for the runtime
    5. Select “RecipeFinderLambda” for the role which was just created
    6. Paste the function code inside the inline editor
    7. Click “save”
3. Update the lambda function
    1. Since “request” library is necessary for the lambda function, “request” package must be imported in the nodejs code. To include libraries other than “aws-sdk”, functions must be uploaded through the aws CLI
    2. Install aws CLI following the instructions here
    3. Navigate to the function directory on the local computer
    4. Install “request” library by typing “npm install request” in the terminal
    5. Zip the whole directory by “zip -r function.zip .”
    6. Use “aws lambda update-function-code --function-name findRecipe --zip-file fileb://function.zip --region us-east-1” to upload the function to Lambda
4. Test the function
    1. If the function executes successfully, the logs should look like this

Execution result: succeeded (logs)

▼ Details

The area below shows the result returned by your function execution. Learn more about returning results from your function.


```
{
  "statusCode": 201,
  "body": "{\"Recipes\":\"{\\\"results\\\":[{\\\"id\\\":29559,\\\"title\\\":\\\"Ginger Beef with Broccoli\\\",\\\"readyInMinutes\\\":45,\\\"servings\\\":4,\\\"image\\\":\\\"ginger-beef-with-broccoli-29559.jpg\\\",\\\"imageUrls\\\":[\\\"ginger-beef-with-broccoli-29559.jpg\\\"]},{\\\"id\\\":44499,\\\"title\\\":\\\"Ginger Beef Salad\\\",\\\"readyInMinutes\\\":45,\\\"servings\\\":4,\\\"image\\\":\\\"ginger-beef-salad-
```

Create an API using AWS api gateway

1. Create a REST API
    1. Choose api gateway from the AWS management console
    2. Select “create API”
    3. Click “new API”
    4. Enter “RecipeFinder” for the api name
    5. Leave everything as default and click “create API”
2. Create a Cognito user pool authorizer to secure API
    1. Select authorizers for the API just created
    2. Click “choose new authorizer”
    3. Choose “cognito” for the type
    4. Select the region where I created the Cognito User pool
    5. Enter “recipe-finder” for the Cognito User Pool input
    6. Then enter “Authorization” for Token Source
    7. Click on “create”
3. Create a resource and a method for the API
    1. Choose resources for the API just created
    2. Then choose “create resource” from “actions” dropdown
    3. Enter “find” for the resource name
    4. Tick “Enable API Gateway CORS”
    5. Select “create resource”
    6. Under the new “/find” resource, choose “create method”
    7. Choose “post” and click the checkmark on the right side
    8. Choose “lambda function” for the integration type
    9. Tick “Use Lambda Proxy integration”
    10. Choose the region where the lambda function was created
    11. Enter the name of the function “findRecipe”
    12. Click “save” and click “ok” on the popup to give API gateway to call the lambda function
    13. Choose “Method Request” under the resource
    14. Choose “WildRydes Cognito user pool authorizer” from the “authorization dropdown” then click the checkmark button
4. Deploy the API
    1. Choose “deploy API” from the action dropdown
    2. From the “Deployment stage drop-down list”, choose “new stage”
    3. Enter “prod” for the stage name
    4. Click “deploy”
    5. Copy the invoke URL of the API
    6. Add the URL to the config.js file under the api key
    7. The final config.js file is shown below

```
window._config = {
    cognito: {
        userPoolId: 'us-east-1_UZVGgKrkb',
        userPoolClientId: '6hqkfjac9ojkfibrvk65ttgph6',
        region: 'us-east-1'
```

```
  },
  api: {
    invokeUrl: 'https://y1ph0o0x8i.execute-api.us-east-1.amazonaws.com/prod'
  }
};
```

User Manual

1. Sign up using email and password.
2. Because the educate account doesn't support Amazon Simple Email Service, the user has to be confirmed manually on the Amazon Cognito Console. So, to make it easier. Please use the following user name and password to log in(this user name has been manually confirmed on the Cognito Console):

   User name: test@gmail.com
   Password: Qwer1234**
3. To find recipes, enter ingredients separated by comma. For example, "beef,ginger".
4. Recipes will then be displayed, you can click on each recipe to get a detailed information of the recipe
5. There's a "add to favourites" button for each recipe. Once it's clicked, it will be added to the "favourites" list
6. To find my "favourties" recipes, click on the "favourites" link on the top left corner
7. Again, to get a detailed description of a recipe, simply click on the recipe
8. To log out, click on the log out button on the top right corner

References

https://aws.amazon.com/getting-started/projects/build-serverless-web-app-lambda-apigateway-s3-dynamodb-cognito/module-1/

https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-user-identity-pools.html

https://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html

https://aws.amazon.com/lambda/

https://aws.amazon.com/dynamodb/getting-started/

https://rapidapi.com/spoonacular/api/recipe-food-nutrition