Project Demo: Process Documentation

Overview

This document outlines the frontend process of a system designed to accept user inputs through a web interface, upload files to an AWS S3 bucket, insert data into a DynamoDB table, and trigger further processing including EC2 instance creation and file manipulation.

System Components

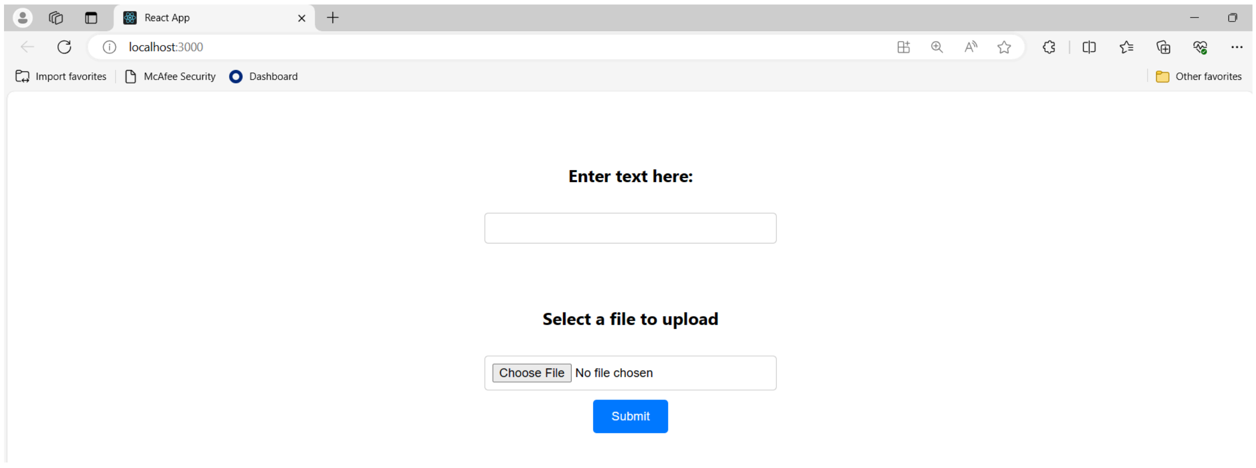
- Frontend Interface: Built with React.js
- Backend Services: AWS Lambda, DynamoDB, S3 Bucket
- API Gateway: fileUploaderGateway, DynamoDBInsertionAPI

Step-by-Step Process

1. User Interface Interaction

The frontend React UI presents the user with:

- An input text field.
- An input file uploader.
- A submit button.



Enter text here:

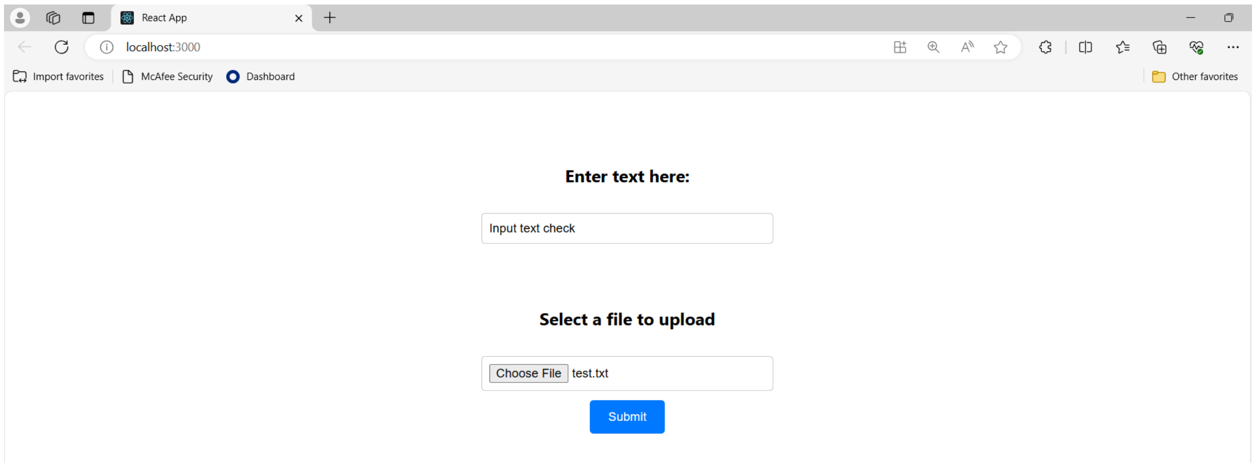
Select a file to upload

Choose File No file chosen

Submit

2. File and Text Submission

Users attach a file, enter text into the provided field, and click the submit button to initiate the process.



Enter text here:

Input text check

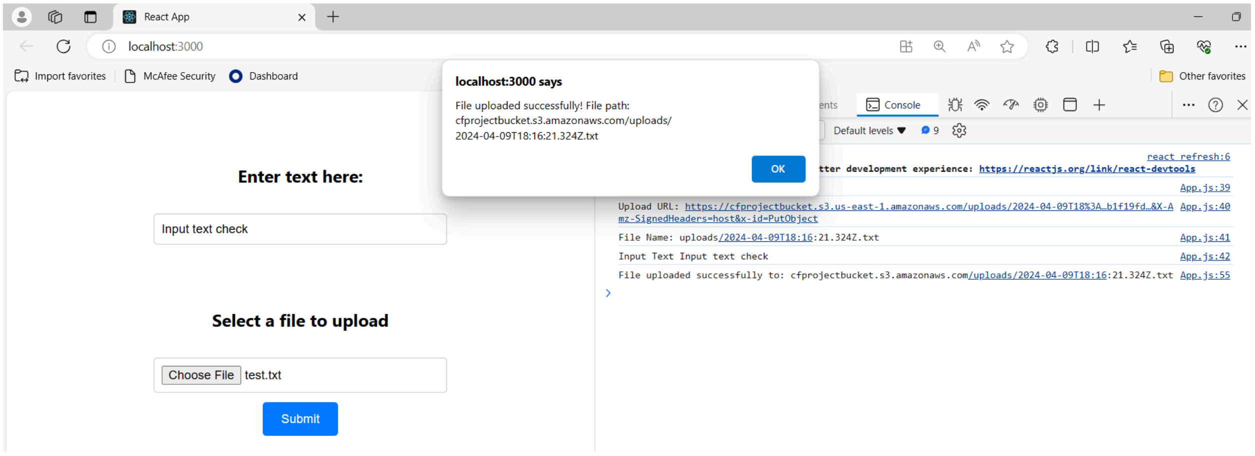
Select a file to upload

Choose File test.txt

Submit

3. Pre-signed URL Generation

Upon submission, the frontend calls a Lambda function named `fileuploaderSystem`, which retrieves a pre-signed URL from the `fileUploaderGateway` API Gateway.



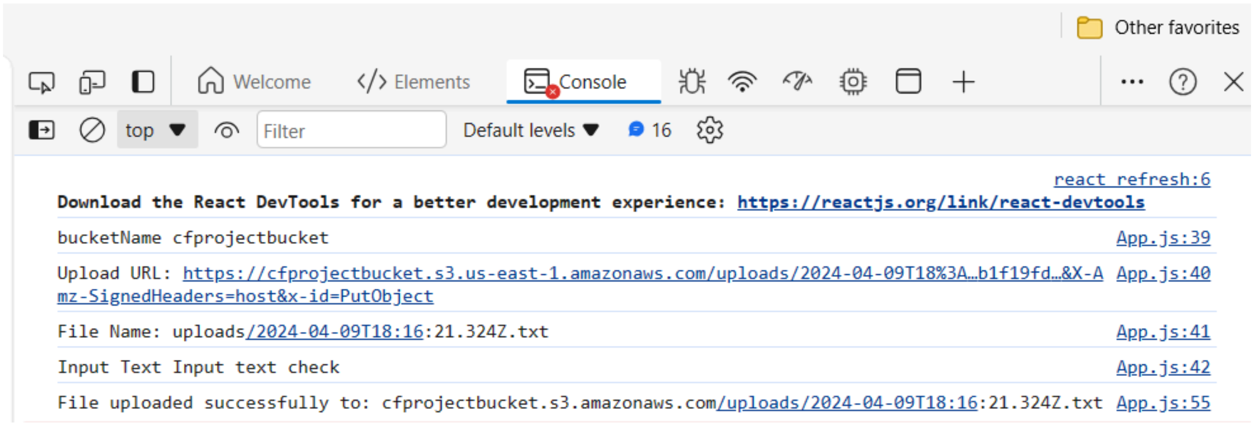
localhost:3000 says

File uploaded successfully! File path:
cfprojectbucket.s3.amazonaws.com/uploads/
2024-04-09T18:16:21.324Z.txt

```
                                                                              react_refresh:6
tter development experience: https://reactjs.org/link/react-devtools
                                                                                  App.js:39
Upload URL: https://cfprojectbucket.s3.us-east-1.amazonaws.com/uploads/2024-04-09T18%3A…b1f19fd…&X-A  App.js:40
mz-SignedHeaders=host&x-id=PutObject
File Name: uploads/2024-04-09T18:16:21.324Z.txt                                   App.js:41
Input Text Input text check                                                       App.js:42
File uploaded successfully to: cfprojectbucket.s3.amazonaws.com/uploads/2024-04-09T18:16:21.324Z.txt  App.js:55
>
```

4. File Upload to S3 Bucket

With the pre-signed URL, the React application uploads the input file directly to the specified AWS S3 bucket.



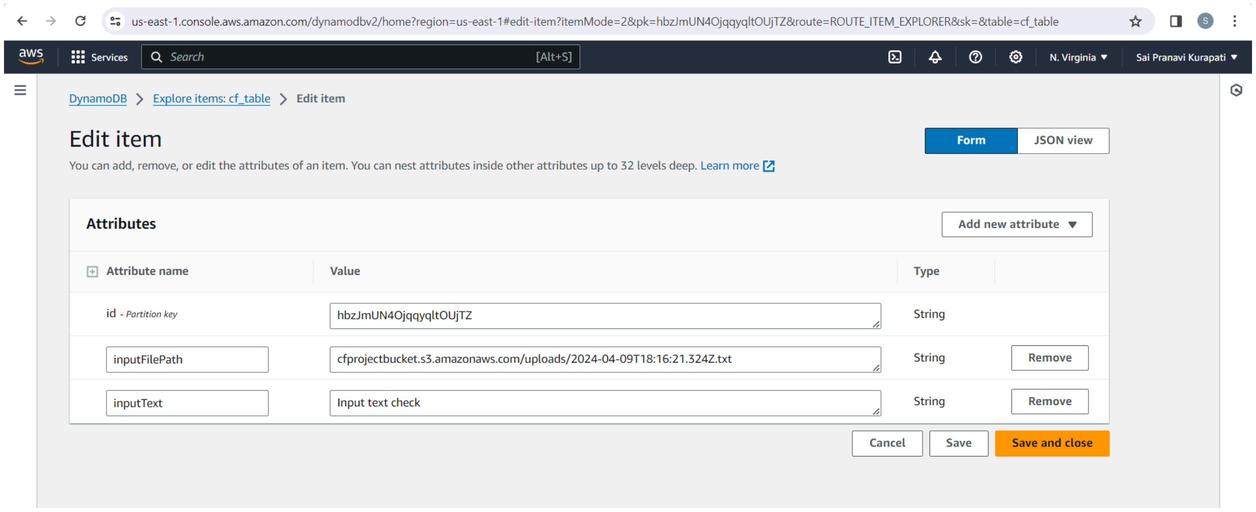
```
                                                                              react_refresh:6
Download the React DevTools for a better development experience: https://reactjs.org/link/react-devtools
bucketName cfprojectbucket                                                        App.js:39
Upload URL: https://cfprojectbucket.s3.us-east-1.amazonaws.com/uploads/2024-04-09T18%3A…b1f19fd…&X-A  App.js:40
mz-SignedHeaders=host&x-id=PutObject
File Name: uploads/2024-04-09T18:16:21.324Z.txt                                   App.js:41
Input Text Input text check                                                       App.js:42
File uploaded successfully to: cfprojectbucket.s3.amazonaws.com/uploads/2024-04-09T18:16:21.324Z.txt  App.js:55
```

5. Data Insertion into DynamoDB

Next, the React app makes a POST API call to the `DynamoDBInsertionAPI` (via API Gateway) to insert the submission details into a DynamoDB table named `cf_table`. The handling Lambda function is `insertToDynamoDBTable`.

This insertion includes:

- A unique ID (generated using nano ID).
- The input text.
- The input file path (constructed as `bucketname/filename`).



us-east-1.console.aws.amazon.com/dynamodbv2/home?region=us-east-1#edit-item?itemMode=2&pk=hbzJmUN4OjqqyqltOUjTZ&route=ROUTE_ITEM_EXPLORER&sk=&table=cf_table

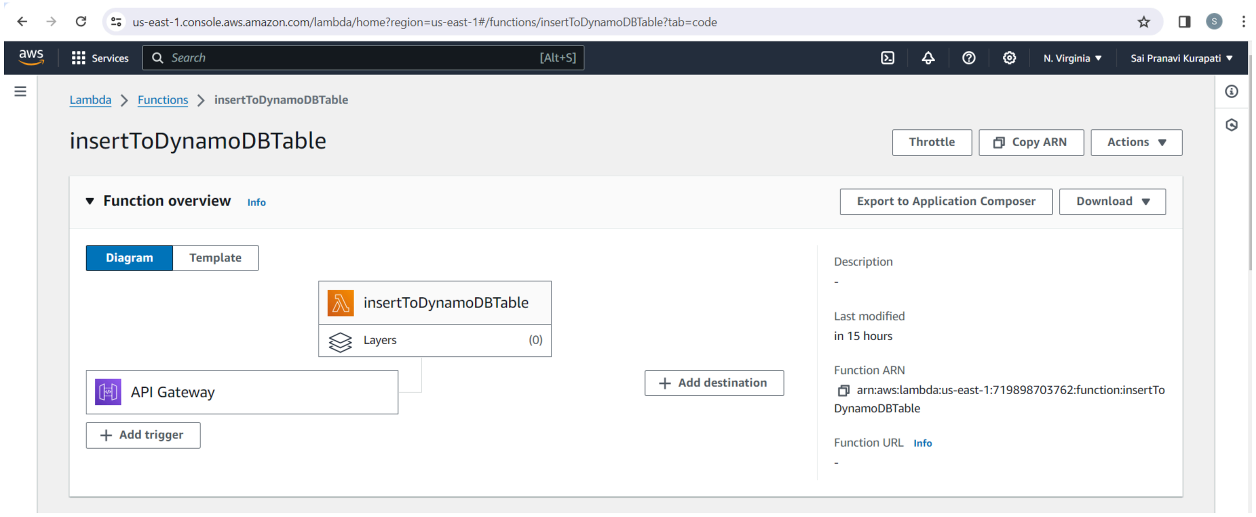
DynamoDB > Explore items: cf_table > Edit item

Edit item

You can add, remove, or edit the attributes of an item. You can nest attributes inside other attributes up to 32 levels deep. Learn more

Attributes

Attribute name	Value	Type
id - Partition key	hbzJmUN4OjqqyqltOUjTZ	String
inputFilePath	cfprojectbucket.s3.amazonaws.com/uploads/2024-04-09T18:16:21.324Z.txt	String
inputText	Input text check	String



us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/functions/insertToDynamoDBTable?tab=code

Lambda > Functions > insertToDynamoDBTable

insertToDynamoDBTable

Function overview

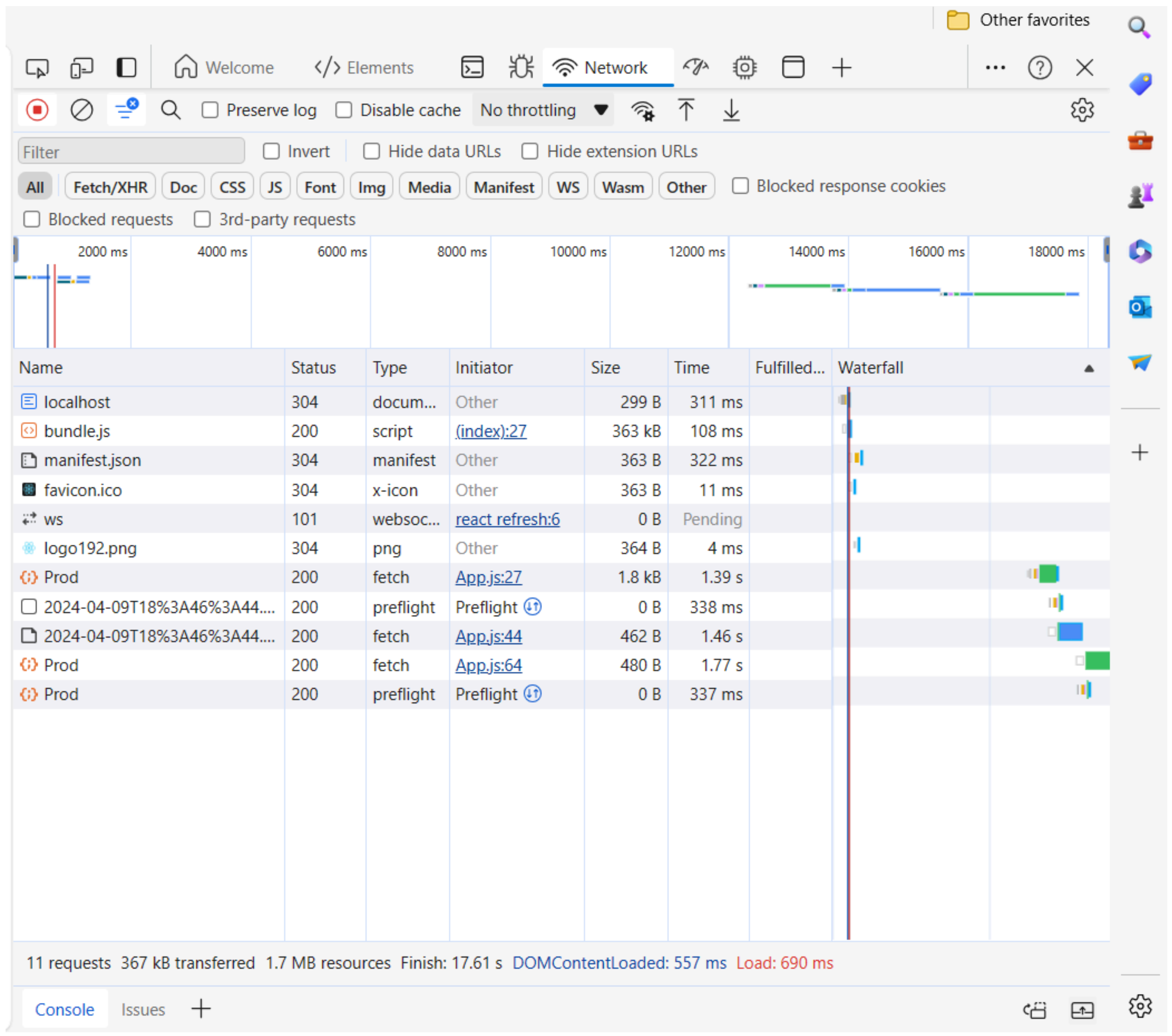
Description
-

Last modified
in 15 hours

Function ARN
arn:aws:lambda:us-east-1:719898703762:function:insertToDynamoDBTable

Function URL
-

```
Data sent to Lambda successfully: UwHHuJ2nN6iJvyaDSvrv_ Test  cfprojectbucket.s3.amazonaws.com/uploa App.js:78
ds/2024-04-09T18:40:31.448Z.txt
```



Preserve log ☐ Disable cache ☐ No throttling

Filter ☐ Invert ☐ Hide data URLs ☐ Hide extension URLs

All Fetch/XHR Doc CSS JS Font Img Media Manifest WS Wasm Other ☐ Blocked response cookies

☐ Blocked requests ☐ 3rd-party requests

Name	Status	Type	Initiator	Size	Time	Fulfilled...	Waterfall
localhost	304	docum...	Other	299 B	311 ms		
bundle.js	200	script	(index):27	363 kB	108 ms		
manifest.json	304	manifest	Other	363 B	322 ms		
favicon.ico	304	x-icon	Other	363 B	11 ms		
ws	101	websoc...	react refresh:6	0 B	Pending		
logo192.png	304	png	Other	364 B	4 ms		
Prod	200	fetch	App.js:27	1.8 kB	1.39 s		
2024-04-09T18%3A46%3A44....	200	preflight	Preflight	0 B	338 ms		
2024-04-09T18%3A46%3A44....	200	fetch	App.js:44	462 B	1.46 s		
Prod	200	fetch	App.js:64	480 B	1.77 s		
Prod	200	preflight	Preflight	0 B	337 ms		

11 requests 367 kB transferred 1.7 MB resources Finish: 17.61 s DOMContentLoaded: 557 ms Load: 690 ms

6. DynamoDB Event Trigger

The insertion into `cf_table` triggers an event, which in turn calls the Lambda function `dynamoDBEventtoCreateEC2`.

7. EC2 Instance Creation and Processing

The `dynamoDBEventtoCreateEC2` function:

- Creates an EC2 instance.
- Executes a script to:
    - Retrieve the input file path and text using the event's ID.

    - Download the file from the S3 bucket.
    - Merge the input text with the file.
    - Upload the modified file back to the S3 bucket.
- Terminates the instance upon completion of the script.

![Insert screenshot of the Lambda function or EC2 instance script]

Conclusion

This document has detailed the frontend process for submitting user inputs, processing files, and triggering backend workflows within our project. This approach demonstrates the integration of various AWS services with a React frontend to achieve a seamless file processing and data management solution.