
# SkipQ – Orion (Cohort-5)

## Documentation

Documented by: Muhammad Shamaas

# Table of Contents

# Project Document

## 1 TRAINING OVERVIEW

In this course, I designed, developed and deployed a production-grade web application using MERN stack. The application consisted of a web crawler that monitored the health of various websites on a given cadence. I set up an Express application which served as the middleware API layer, set up a React application as the frontend, performed CRUD operations on MongoDB, integrated SSO (Github and OAuth), created tables with paginated data, performed search on the paginated data, created charts and graphs using HighCharts to display different CloudWatch metrics, and wrote unit test cases using Cypress. I also ran the application across multiple AWS Regions. I used CI/CD to automate multiple deployment stages (prod vs beta), wrote unit/integration tests to continuously deploy the application without human intervention and automated metrics to rollback a deployment in case of service degradation.

# 2 SPRINT 1 AND 2

## 2.1 OBJECTIVE

I used AWS CDK to build a canary in a Lambda function. This canary runs in one AWS Region and measures availability and latency when accessing a predefined, public web application. I pushed the code to versioning control repo and managed README files in markdown on GitHub. Then I extended the canary Lambda function into a web crawler to crawl a custom list (json file) of websites from an S3 bucket. The S3 bucket included webpages that should be crawled on the websites. I ran the crawler periodically on a 5 min cadence and wrote availability and latency metrics for each website to CloudWatch using CloudWatch's API. Then, I created a CloudWatch Dashboard to monitor website health, and set up alarms when availability or latency falls below prescribed thresholds. Every alarm was also published to SNS notifications with tags that were used to filter by metric type.

I learned about the following concepts:

- Cloud Computing and Infrastructure-as-Code (IaC)
- AWS Regions/AZs/Edge Services, Foundational services (EC2, S3, CloudFront), Microservice architecture
- AWS Services: IAM, Lambda
- Shell and Scripting tools, Vim, GitHub
- Writing code on AWS
- Introduction to the Art of Monitoring Web Applications
- AWS Monitoring Services: CloudWatch, SNS/SQS
- Scalability in web application

## 2.2 TECHNOLOGIES USED

### 2.2.1 AWS Cloud9 Environment

AWS Cloud9 is a cloud-based integrated development environment (IDE) that lets you write, run, and debug your code with just a browser. It includes a code editor, debugger, and terminal. Cloud9 comes prepackaged with essential tools for popular programming languages, including JavaScript, Python, PHP, and more, so you don't need to install files or configure your development machine to start new projects. Since your Cloud9 IDE is cloud-based, you can work on your projects from your office, home, or anywhere using an internet-connected machine. Cloud9 also provides a seamless experience for developing serverless applications

enabling you to easily define resources, debug, and switch between local and remote execution of serverless applications. With Cloud9, you can quickly share your development environment with your team, enabling you to pair program and track each other's inputs in real time.

### 2.2.2 AWS S3

Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance. Customers of all sizes and industries can use Amazon S3 to store and protect any amount of data for a range of use cases, such as data lakes, websites, mobile applications, backup and restore, archive, enterprise applications, IoT devices, and big data analytics. Amazon S3 provides management features so that you can optimize, organize, and configure access to your data to meet your specific business, organizational, and compliance requirements.

### 2.2.3 AWS IAM

AWS Identity and Access Management (IAM) provides fine-grained access control across all of AWS. With IAM, you can specify who can access which services and resources, and under which conditions. With IAM policies, you manage permissions to your workforce and systems to ensure least-privilege permissions.

### 2.2.4 AWS Lambda

AWS Lambda is a serverless, event-driven compute service that lets you run code for virtually any type of application or backend service without provisioning or managing servers. You can trigger Lambda from over 200 AWS services and software as a service (SaaS) applications, and only pay for what you use.

### 2.2.5 Github

GitHub is an online software development platform used for storing, tracking, and collaborating on software projects. It enables developers to upload their own code files and to collaborate with fellow developers on open-source projects.

### 2.2.6 AWS CloudWatch

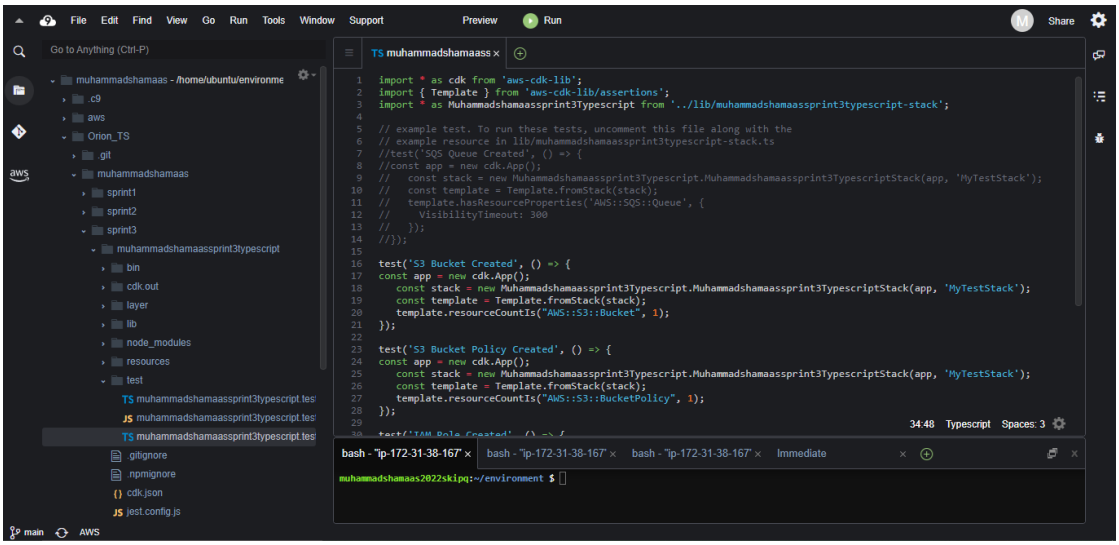
Amazon CloudWatch is a monitoring and observability service built for DevOps engineers, developers, site reliability engineers (SREs), IT managers, and product owners. CloudWatch provides you with data and actionable insights to monitor your applications, respond to system-wide performance changes, and optimize resource utilization. CloudWatch collects monitoring and operational data in the form of logs, metrics, and events. You get a unified view of operational health and gain complete visibility of your AWS resources, applications,

and services running on AWS and on-premises. You can use CloudWatch to detect anomalous behavior in your environments, set alarms, visualize logs and metrics side by side, take automated actions, troubleshoot issues, and discover insights to keep your applications running smoothly.

## 2.3 SPRINT BREAKDOWN

### 2.3.1 Task – 1: Create an EC2 environment

An EC2 instance was created to define architecture for the web health application. Stack file was used for creating CloudFormation template. AWS lambda function was used for monitoring of websites.

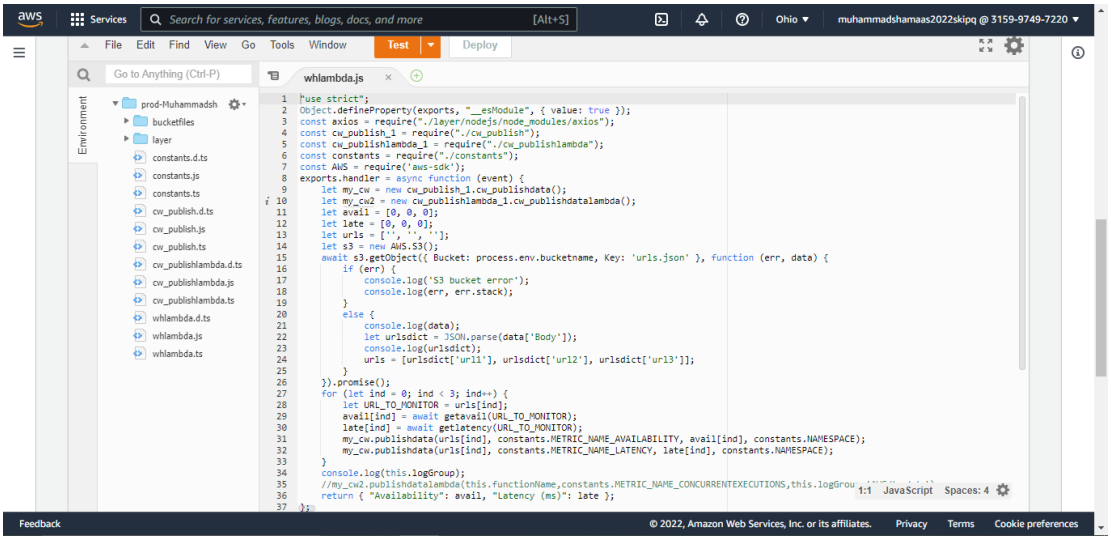



### 2.3.2 Task – 2: Schedule invocation of lambda function at fixed rate

The AWS web health lambda function was invoked every five minutes to find out the latency and availability values of each website.

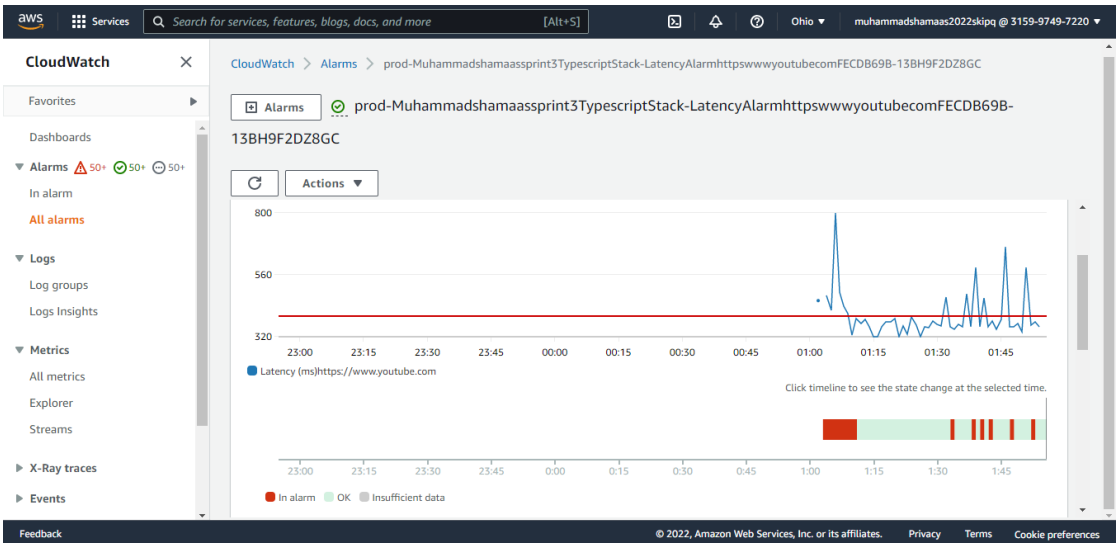
### 2.3.3 Task – 3: Write code for lambda function

AWS lambda function was called every minute to find out the latency and availability values of each website. The availability and latency values are stored in a dictionary and passed to the cloudwatch client. The cloudwatch client publishes the data to a given namespace.



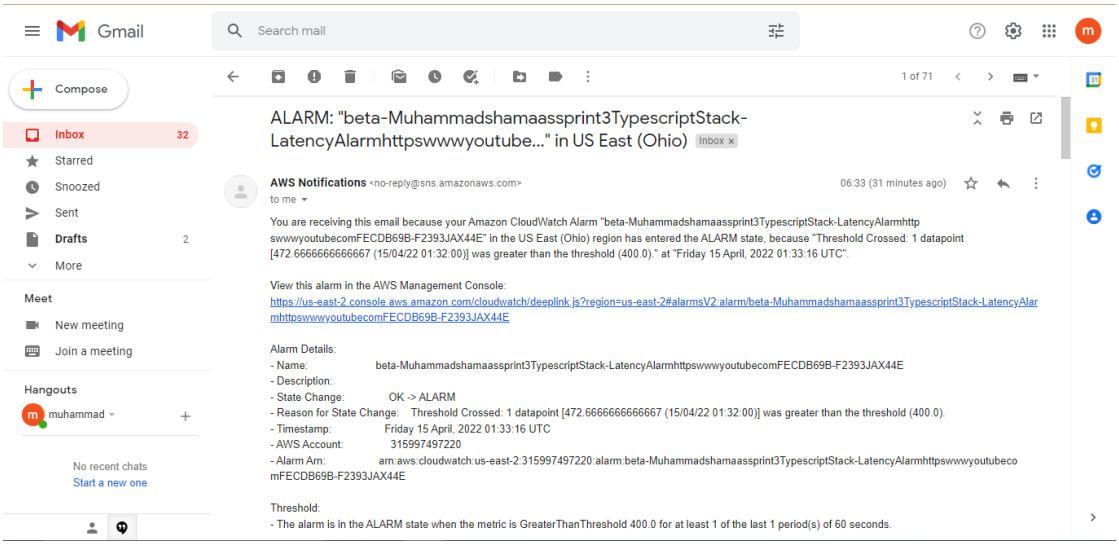

### 2.3.4 Task – 4: Install Cloudwatch alarms

A CloudWatch alarm was installed to alert user whenever the availability value was less than 1. A CloudWatch alarm was installed to alert user whenever the latency value was greater than 0.22s.




### 2.3.5 Task – 4: Configure SNS email alert

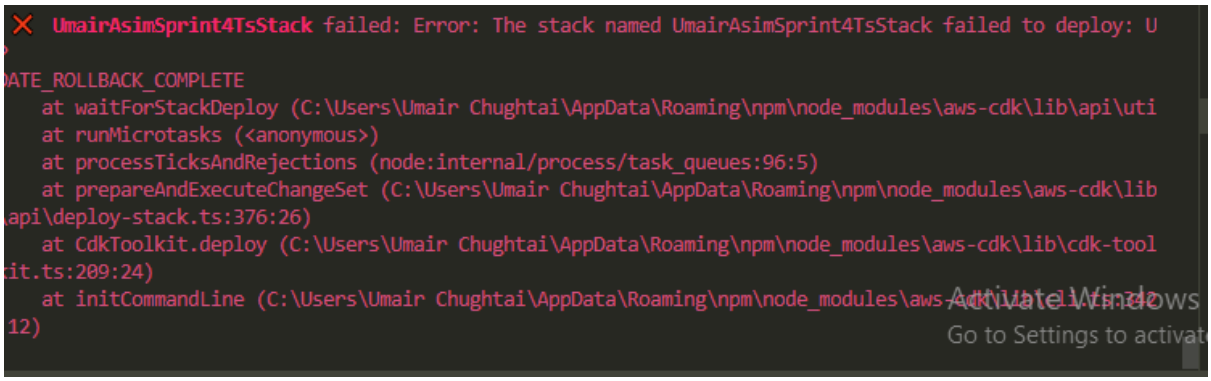
If any CloudWatch alarm was activated, the user was alerted using SNS email. The user had to subscribe to the latency alarm topic and availability alarm topic to receive SNS notification emails.




## 2.4 Issues/Errors and Troubleshooting

### 2.4.1 Issue: cdk deploy failed

- **Description**: The deployment of aws code failed




- Solutions: I executed the following commands:

1. npm run watch
2. cdk synth
3. cdk deploy

## 2.5 Function and classes Documentation

### 2.5.1 Create_lambda:

| createLambda(self,id,asset,handler,layer,role) | |
|---|---|
| Description | Creates the lambda function |
| Parameters | id (string): The user defined id for the lambda function.<br><br>asset (string): The folder path where the lambda |

| | |
|---|---|
| | function code was stored.<br><br>handler (string): The name of the function to execute when lambda function was invoked.<br><br>layer: The lambda function layer consisting of installed Node.js libraries required by the lambda function.<br><br>role: The IAM role for assigning the required AWS managed policies to Lambda function. |
| Returns | lambda function |

### 2.5.2 Lambda_handler:

| lambda_handler(event) | |
|---|---|
| Description | The lambda function code executed whenever the lambda function was invoked. |
| Parameters | event: describes what the event is and what it contains |
| Returns | It reads a dictionary of URLs stored in S3 bucket. Then, it visits each URL and records its latency and availability metric. These metrics are returned in a dictionary and published to CloudWatch namespace using aws-sdk<br>cloudwatch.putMetricData({<br>MetricData: [{<br>MetricName: metric_name+url,<br>Dimensions: [{Name: 'URL',Value: url}],<br>Value: metric_value<br>}],<br>Namespace: namespace<br>},callback_function); |

### 2.5.3 Create_CloudWatch_alarm:

| Create_CloudWatch_alarm (id,operator,threshold,period,metric) | |
|---|---|
| Description | It created a CloudWatch alarm to detect degraded health of website. |
| Parameters | id: User defined id for alarm.<br><br>operator: The comparison operator.<br><br>threshold: The threshold to trigger alarm.<br><br>period: The period to check alarm condition.<br><br>metric: The metric on which the alarm is installed. It was created using<br>cloudwatch.Metric({<br>metricName:my_metric_name, |

| | |
|---|---|
| | namespace:my_namespace,<br>period:my_period,<br>dimensionsMap: {'URL': my_url}<br>}); |
| Returns | It returned the CloudWatch alarm for the input metric. |

### 2.5.4 Alarm.addAlarmAction:

| Alarm.addAlarmAction (action) | |
|---|---|
| Description | It assigned SNS action for CloudWatch alarm. |
| Parameters | Alarm: The CloudWatch alarm.<br><br>action: The action taken when CloudWatch alarm was initiated e.g.<br>new cwactions.SnsAction(new sns.Topic(this,topic_id)) |
| Returns | N/A |

# 3 SPRINT 3

## 3.1 OBJECTIVE

I created a multi-stage pipeline having Beta/Gamma and Prod stage using CDK. This provided an introduction to CI/CD. CodePipeline was used for build and test, whereas CodeDeploy was used for CD. Beta and prod environments were setup in CodePipeline and deployed using CodeDeploy. I deployed the project code in us-east-2 region. Each stage had bake times, code-review, and test blockers. I wrote unit/integration tests for the web crawler. It tested CloudWatch metrics and alarms for operational health of the web crawler, including memory and time-to-process each crawler run. I also automated rollback to the last build if metrics were in alarm. I learned to manage README files and runbooks in markdown on GitHub. By integrating AWS CodePipeline with GitHub, I learned about building a release process by writing merge-blocking automated tests for the canary on CodePipeline. By building operational CloudWatch metrics for web crawler, rollback automation was achieved to allow rollback to last build.

I learned about the following concepts:

- Introduction to CI/CD
- AWS services: CodePipeline for build and test, CodeDeploy for CD
- Integrate AWS CodePipeline with GitHub
- Automated testing using PyTest
- Build a release process by writing merge-blocking automated tests for the canary on CodePipeline
- Build operational CloudWatch metrics for web crawler
- Write rollback automation allowing rollback to last build
- Setup beta and prod environments in CodePipeline and deploy using CodeDeploy

## 3.2 TECHNOLOGIES USED

### 3.2.1 AWS CloudWatch

Amazon CloudWatch is a monitoring and observability service built for DevOps engineers, developers, site reliability engineers (SREs), IT managers, and product owners. CloudWatch provides you with data and actionable insights to monitor your applications, respond to system-wide performance changes, and optimize resource utilization. CloudWatch collects monitoring and operational data in the form of logs, metrics, and events. You get a unified

view of operational health and gain complete visibility of your AWS resources, applications, and services running on AWS and on-premises. You can use CloudWatch to detect anomalous behavior in your environments, set alarms, visualize logs and metrics side by side, take automated actions, troubleshoot issues, and discover insights to keep your applications running smoothly.

### 3.2.2 AWS CodePipeline

AWS CodePipeline is a fully managed continuous delivery service that helps you automate your release pipelines for fast and reliable application and infrastructure updates. CodePipeline automates the build, test, and deploy phases of your release process every time there is a code change, based on the release model you define. This enables you to rapidly and reliably deliver features and updates. You can easily integrate AWS CodePipeline with third-party services such as GitHub or with your own custom plugin. With AWS CodePipeline, you only pay for what you use. There are no upfront fees or long-term commitments.

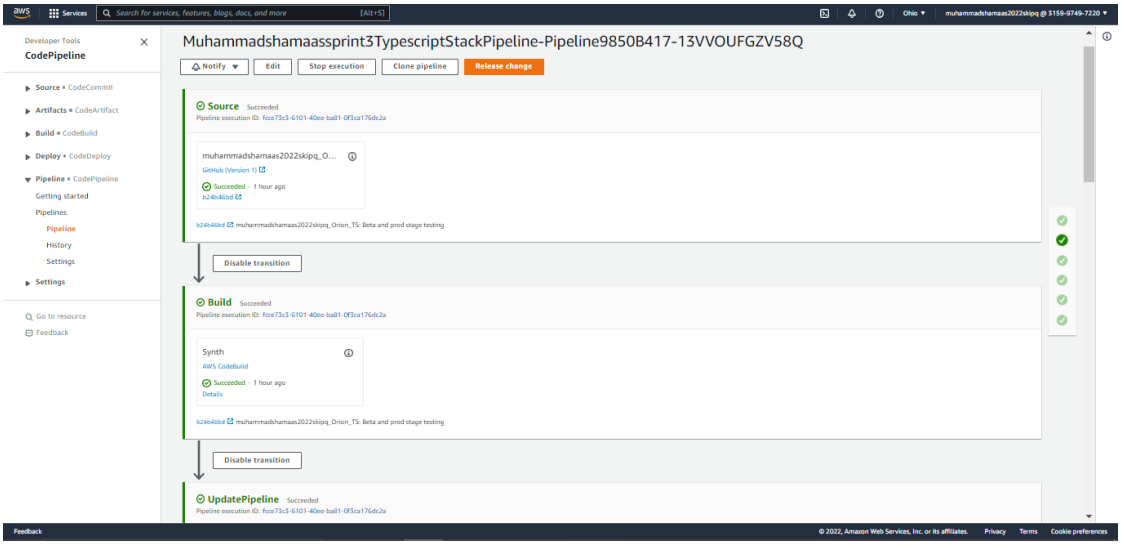
### 3.2.3 Github

GitHub is an online software development platform used for storing, tracking, and collaborating on software projects. It enables developers to upload their own code files and to collaborate with fellow developers on open-source projects.

## 3.3 SPRINT BREAKDOWN

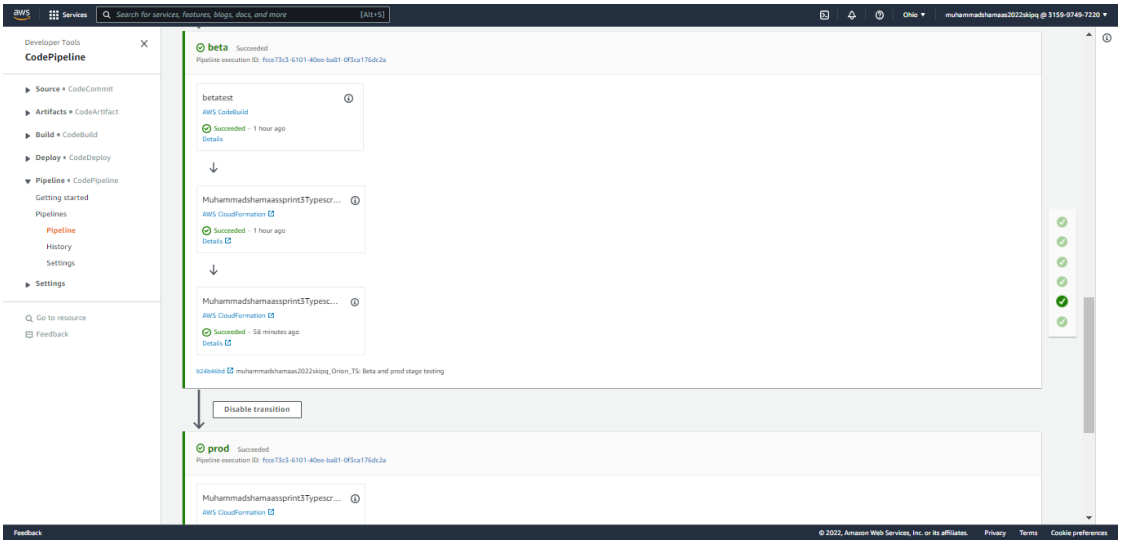
### 3.3.1 Task – 1: Create a CodePipeline

A multi-stage pipeline having beta and prod stage was built using CDK. In the source stage, the pipeline fetched code from Github repository. If this succeeded, the build stage was started to install dependencies using CodePipeline.ShellStep command. The CodePipeline then invoked the UpdatePipeline and Staging.




### 3.3.2 Task – 2: Invoke application stack file for two stages

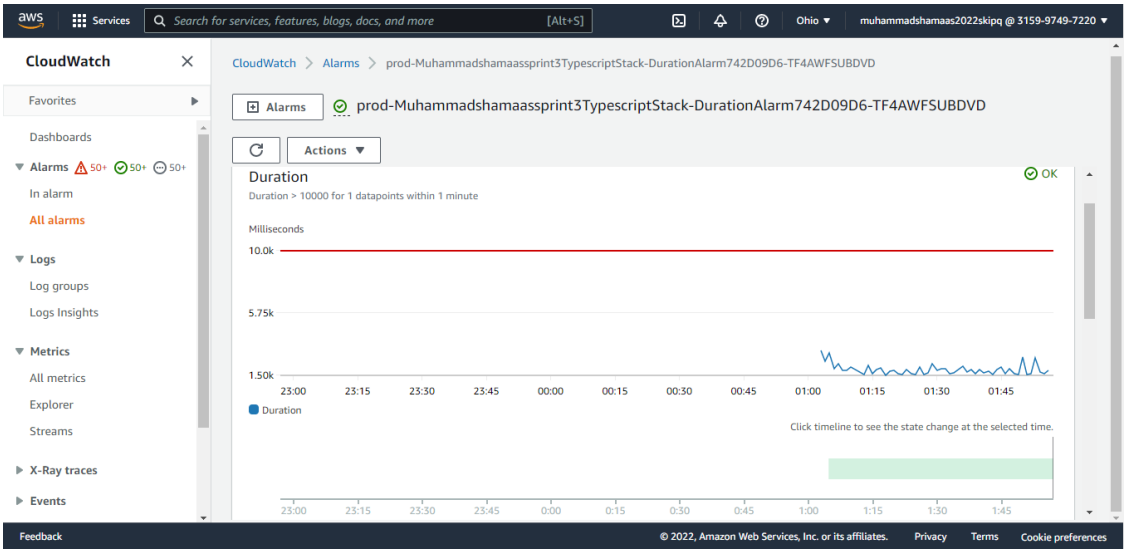
After the project environment was configured, CloudFormation template was generated for beta stage. Unit tests for the web crawler were executed in this stage. If the tests were passed, the CodePipeline proceeded to prod stage. Manual approval was required in prod stage.




### 3.3.3 Task – 3: Monitor health of lambda function

CloudWatch metrics were emitted for the operational health of the web crawler lambda function. Alarms for lambda function Duration and Concurrent Execution metrics were installed. Rollback to the last build was executed if metrics were in alarm. This was achieved using LambdaDeploymentGroup and the rollback configuration was:

codedeploy.LambdaDeploymentConfig.LINEAR_10PERCENT_EVERY_1MINUTE.




## 3.4 ISSUES/ERRORS AND TROUBLESHOOTING

### 3.4.1 Issue: Pipeline error in ShellStep

- **Description**: The dependencies were not installed in ShellStep.

```
[Container] 2022/06/07 13:13:50 Running command cd layers/nodejs

[Container] 2022/06/07 13:13:50 Running command npm ci
added 135 packages in 3.659s

[Container] 2022/06/07 13:14:07 Running command cd ../..

[Container] 2022/06/07 13:14:07 Running command npm install
npm WARN read-shrinkwrap This version of npm is compatible with lockfileVersion@1, but package-lock.json was generated for
lockfileVersion@2. I'll try to do my best with it!
npm ERR! code EBADPLATFORM
npm ERR! notsup Unsupported platform for fsevents@2.3.2: wanted {"os":"darwin","arch":"any"} (current:
{"os":"linux","arch":"x64"})
npm ERR! notsup Valid OS:    darwin
npm ERR! notsup Valid Arch:  any
npm ERR! notsup Actual OS:   linux
npm ERR! notsup Actual Arch: x64

npm ERR! A complete log of this run can be found in:
npm ERR!     /root/.npm/_logs/2022-06-07T13_14_08_985Z-debug.log

[Container] 2022/06/07 13:14:08 Command did not exit successfully npm install exit status 1
[Container] 2022/06/07 13:14:08 Phase complete: BUILD State: FAILED
[Container] 2022/06/07 13:14:08 Phase context status code: COMMAND_EXECUTION_ERROR Message: Error while executing command:
npm install. Reason: exit status 1
[Container] 2022/06/07 13:14:09 Entering phase POST_BUILD
[Container] 2022/06/07 13:14:09 Phase complete: POST_BUILD State: SUCCEEDED
[Container] 2022/06/07 13:14:09 Phase context status code:  Message:
[Container] 2022/06/07 13:14:09 Expanding base directory path: umair_asim/umair_asim_sprint4_ts/cdk.out
[Container] 2022/06/07 13:14:09 Assembling file list
[Container] 2022/06/07 13:14:09 Expanding umair_asim/umair_asim_sprint4_ts/cdk.out
[Container] 2022/06/07 13:14:09 Skipping invalid file path umair_asim/umair_asim_sprint4_ts/cdk.out
```

- **Solutions**: I added the following commands in ShellStep.

```
commands:[
    'cd muhammadshamaas/sprint3/muhammadshamaassprint3typescript',
    'npm ci',
    'npm run build',
    'npm install --save axios',
    'cd resources/layer/nodejs',
    'npm install axios',
```

```
    'cd .. && cd .. && cd ..',
    'export PATH=$PATH:$(npm get prefix)/bin',
    'npx cdk synth'
  ]
```

## 3.5 UNCTION AND CLASSES DOCUMENTATION

### 3.5.1 Create_CloudWatch_Duration_alarm:

| Create_CloudWatch_Duration_alarm (id,operator,threshold,period,metric) | |
|---|---|
| Description | It created a CloudWatch alarm to detect degraded health of website. |
| Parameters | id: User defined id for alarm.<br><br>operator: The comparison operator.<br><br>threshold: The threshold to trigger alarm.<br><br>period: The period to check alarm condition.<br><br>metric: The metric on which the alarm is installed. It was created using<br>cloudwatch.Metric({<br>metricName: 'Duration',<br>namespace: 'AWS/Lambda',<br>period:my_period,<br>dimensionsMap: {'FunctionName': whlambda.functionName}<br>}); |
| Returns | It returned the CloudWatch alarm for the Duration metric. |

### 3.5.2 Create_CloudWatch_ConcurrentExecutions_alarm:

| Create_CloudWatch_Duration_alarm (id,operator,threshold,period,metric) | |
|---|---|
| Description | It created a CloudWatch alarm to detect degraded health of website. |
| Parameters | id: User defined id for alarm.<br><br>operator: The comparison operator.<br><br>threshold: The threshold to trigger alarm.<br><br>period: The period to check alarm condition.<br><br>metric: The metric on which the alarm is installed. It was |

| | |
|---|---|
| | created using<br>cloudwatch.Metric({<br>metricName: 'ConcurrentExecutions',<br>namespace: 'AWS/Lambda',<br>period:my_period,<br>dimensionsMap: {'FunctionName':<br>whlambda.functionName}<br>}); |
| Returns | It returned the CloudWatch alarm for the ConcurrentExecutions metric. |

### 3.5.3 Create_LambdaDeploymentGroup:

| Create_LambdaDeploymentGroup (id,lambda_alias,configuration,alarms) | |
|---|---|
| Description | It created a LambdaDeploymentGroup to detect degraded health of Lambda and initiate rollback. |
| Parameters | id: User defined id for LambdaDeploymentGroup.<br><br>Lambda_alias: The alias for lambda function. It was made using:<br>alias=new aws-lambda.Alias(this, 'alias', {aliasName: 'prod', version: whlambda.currentVersion})<br><br>configuration: The configuration for rollback. This was set using:<br>codedeploy.LambdaDeploymentConfig.LINEAR_10PERCENT_EVERY_1MINUTE<br><br>alarms: The list of CloudWatch alarms to trigger rollback e.g. [DurationAlarm,ConcurrentExecutionsAlarm]. |
| Returns | It returned the LambdaDeploymentGroup for the rollback. |

# 4 SPRINT 4

## 4.1 OBJECTIVE

I built an Express Application with CRUD API functionality defined for the web crawler to create/ read/ update/ delete the list of webpages to crawl. First, I moved the JSON file from S3 to a MongoDB database. Then I implemented CRUD REST commands on MongoDB entries. The Express application ran inside an EC2 instance with a public IP and triggered an AWS Lambda which interacted with the MongoDB database. Finally, I registered the application with AWS Route 53 using a fully qualified domain name. I also wrote API documentation, README files and runbooks in markdown for GitHub.

I learned about the following concepts:

- Key backend concepts: Express, NodeJS, MongoDB
- Set up Express application
- Set up MongoDB
- Write a RESTful interface for reading/writing web crawler output/input
- Persist web crawler results in MongoDB
- Use cloud Domain Name System (DNS) web service.

## 4.2 TECHNOLOGIES USED

### 4.2.1 AWS API Gateway

Amazon API Gateway is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale. APIs act as the "front door" for applications to access data, business logic, or functionality from your backend services. Using API Gateway, you can create RESTful APIs and WebSocket APIs that enable real-time two-way communication applications. API Gateway supports containerized and serverless workloads, as well as web applications.

### 4.2.2 MongoDB

MongoDB is an open-source NoSQL database management program. NoSQL is used as an alternative to traditional relational databases. NoSQL databases are quite useful for working with large sets of distributed data. MongoDB is a tool that can manage document-oriented information, store or retrieve information.

### 4.2.3 AWS S3

Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance. Customers of all sizes and industries can use Amazon S3 to store and protect any amount of data for a range of use cases, such as data lakes, websites, mobile applications, backup and restore, archive, enterprise applications, IoT devices, and big data analytics. Amazon S3 provides management features so that you can optimize, organize, and configure access to your data to meet your specific business, organizational, and compliance requirements.

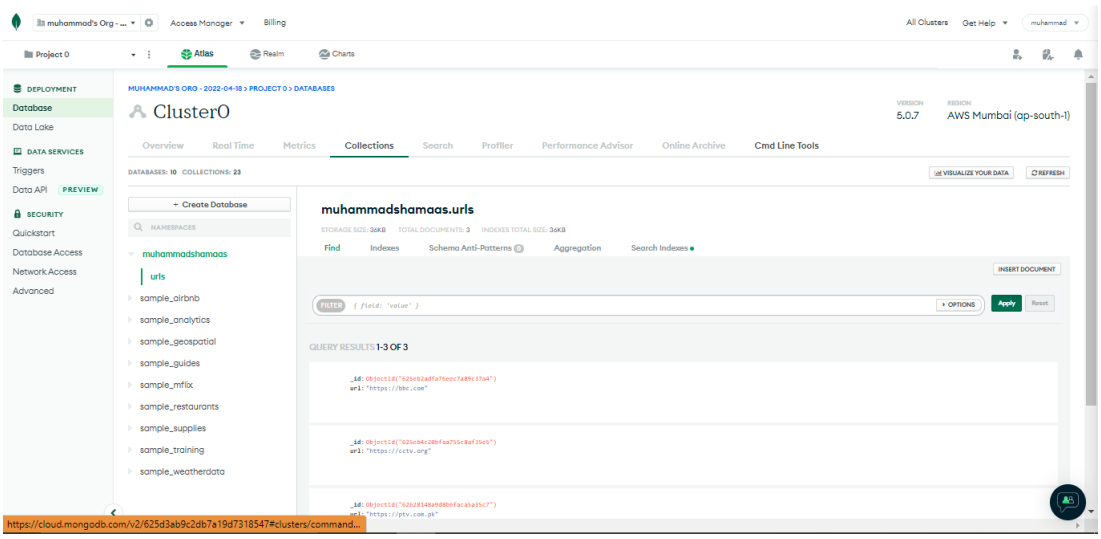
### 4.2.4 Express

Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications. With a myriad of HTTP utility methods and middleware at your disposal, creating a robust API is quick and easy. Express for web applications will give you added advantages, such as rapid application development and routing features. Express provides a thin layer of fundamental web application features, without obscuring Node.js features that you know and love.

## 4.3 SPRINT BREAKDOWN

### 4.3.1 Task – 1: Move the JSON file from S3 to a MongoDB database

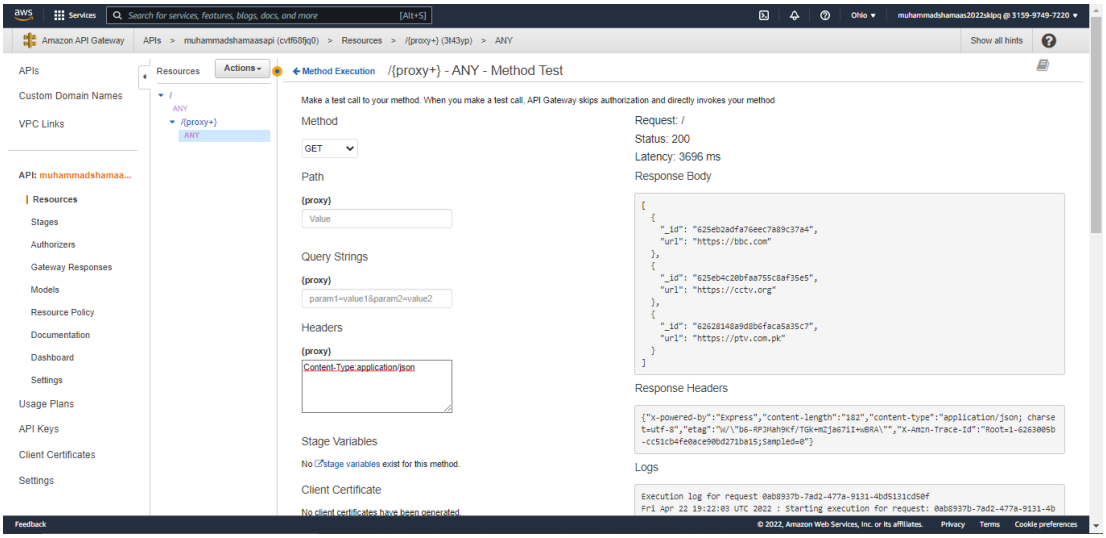
The list of websites was uploaded to MongoDB database so that it could be modified by the user. A cluster was setup for the project. Then, a database was initialized with one collection for storing URLs.




### 4.3.2 Task – 2: Implement CRUD REST commands on MongoDB entries

An Express application was set up to write a RESTful interface for CRUD REST API. Express application ran inside an EC2 instance with a public IP and it triggered an AWS Lambda which

interacted with the MongoDB database. The application was registered with AWS Route 53 using a fully qualified domain name. This allowed the user to modify the list of websites stored in MongoDB.




The GET request was executed by setting

1. Headers option as Content-Type:application/json.

The POST request was executed by setting

1. Headers option as Content-Type:application/json
2. Body option as {“url”:”https://skipq.org”}.

The DELETE request was executed by setting

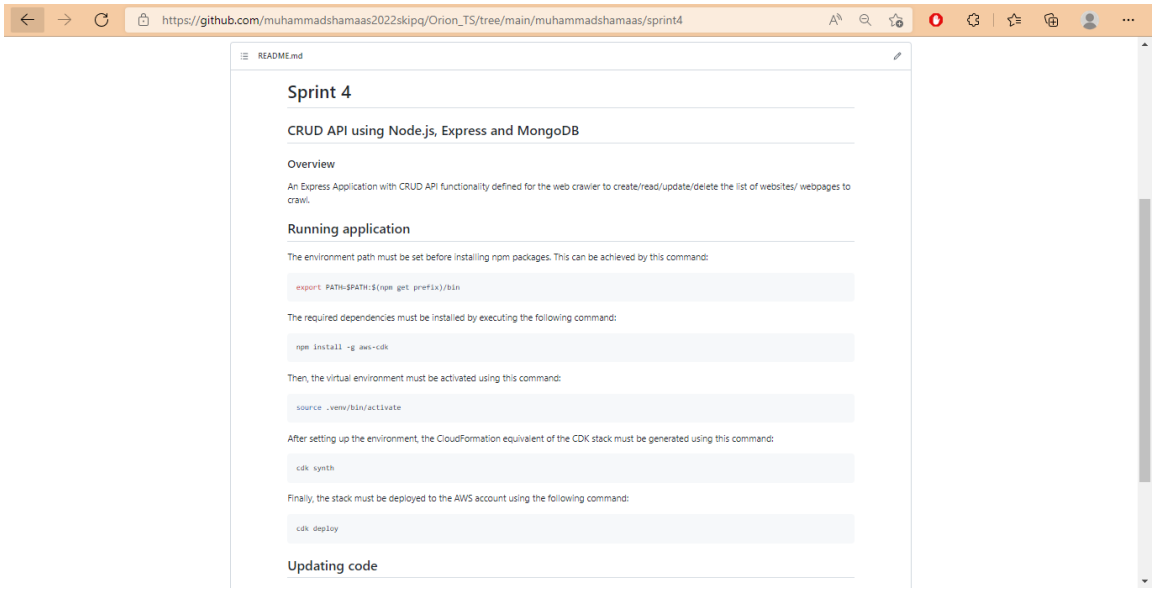
1. Headers option as Content-Type:application/json
2. Body option as {“url”:”https://skipq.org”}

The PUT request was executed by setting

1. Headers option as Content-Type:application/json
2. Body option as [{“url”:”https://skipq.org”}, {“url”:”https://bbc.com”}].

### 4.3.3 Task – 3: Complete documentation

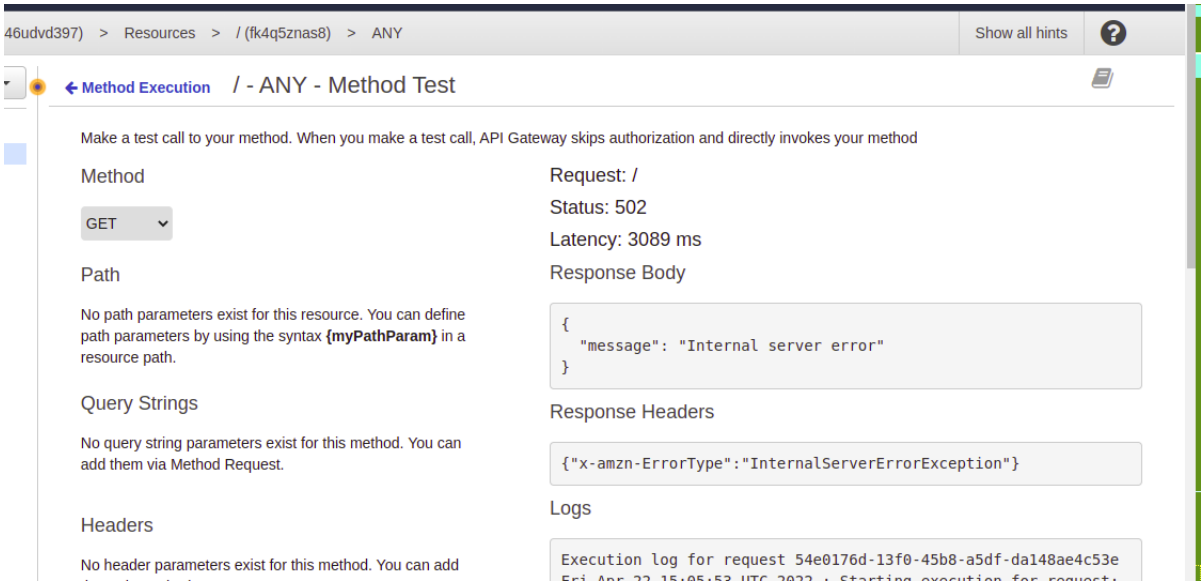
API documentation was completed and committed to GitHub. This involved managing README files and runbooks in markdown on GitHub.




## 4.4 ISSUES/ERRORS AND TROUBLESHOOTING

### 4.4.1 Issue

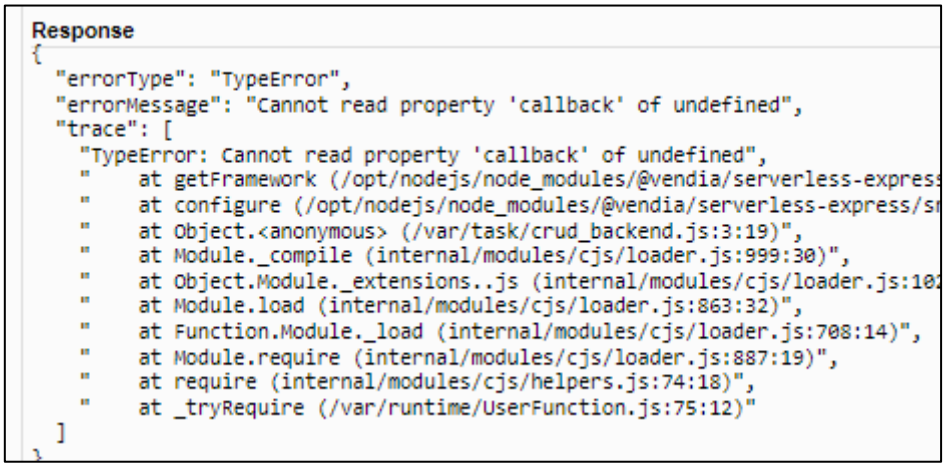
- **Description**: The GET request resulted in internal server error due to unhandled Node.js promise. The Express application GET request handler was passed the promise instead of response data.




- **Solutions**: The result of the promise was sent as response using res.json(result) inside promise.then function.

### 4.4.2 Issue

**Description**: The header of the API request was missing. The server was not able to parse the request body.




**Solutions**: The headers field was changed to Content-Type:application/json.

## 4.5 FUNCTION AND CLASSES DOCUMENTATION

### 4.5.1 dbcreate(url): dbcreate({“url”:”https://skipq.org”})

| dbcreate(url) | |
|---|---|
| Description | Adds url to the MongoDB database |
| Parameters | url (Dictionary): JSON format dictionary with new URL |
| Returns | Result of database.collection.insertOne request |

### 4.5.2 dbdelete(url): dbdelete({“url”:”https://skipq.org”})

| dbdelete(url) | |
|---|---|
| Description | Deletes url from the MongoDB database |
| Parameters | url (Dictionary): JSON format dictionary with target URL |
| Returns | Result of database.collection.deleteOne request |

### 4.5.3 dbread(url): dbread({“url”:”https://skipq.org”})

| dbread(url) | |
|---|---|
| Description | Reads url from the MongoDB database |
| Parameters | url (Dictionary): JSON format dictionary with target URL |
| Returns | Result of database.collection.find request |

#### 4.5.4 dbupdate(url1,url2): dbupdate({“url”:”https://skipq.org”}, {“url”:”https://bbc.com”})

| dbupdate(url1,url2) | |
|---|---|
| Description | Replaces url1 with url2 in the MongoDB database |
| Parameters | url1 (Dictionary): JSON format dictionary with target URL<br>url2 (Dictionary): JSON format dictionary with new URL |
| Returns | Result of database.collection.replaceOne request |

# 5 SPRINT 5

## 5.1 OBJECTIVE

I built a frontend application using React.js and a Material UI design system for the MongoDB CRUD API. Frontend application had a text field where users could type in a URL and then perform one of the following operations:

1. Add a new URL to the DB (MongoDB). The message “Already Exists” was returned if the added URL was already present in the DB. When implementing this operation, I checked that the value entered in the text field was a valid URL.
2. Delete a URL from the DB. “Not Found” error was returned if the URL was not present in the DB. When implementing this operation, I checked that the value entered in the text field was a valid URL.
3. Search the DB for a particular URL. “Not Found” error was returned if the URL was not present in the DB. If the URL was found then it showed a table with the URL and all its sub-URLs. The table was paginated with only 5 entries showing per page. It also plotted the CloudWatch metrics for Latency and Availability for that URL in the past 1 week. I used HighCharts.js library for plotting the time-series data. These graphs remained visible as the user scrolled through pages. If the URL had child links then it plotted the 10 child links using D3.js library’s Hierarchy graphs. The graphs remained visible as the user scrolled through pages.

I learned about the following concepts:

- Create a frontend app with React
- Connect frontend React app with a backend database (MongoDB)
- Use graph libraries to visualize data: HighCharts, D3 Hierarchy.

## 5.2 TECHNOLOGIES USED

### 5.2.1 React

React is a JavaScript library for creating user interfaces. The react package contains only the functionality necessary to define React components. It is typically used together with a React renderer like react-dom for the web, or react-native for the native environments. React makes it painless to create interactive UIs, design simple views for each state in the application, and React efficiently updates and renders just the right components when data changes. Declarative views make the code more predictable and easier to debug. It uses encapsulated components that manage their own state, to make complex UIs. Since component logic is written in JavaScript instead of templates, we can easily pass rich data through the app and keep state out of the DOM. We can develop new features in React without rewriting existing code. React can also render on the server using Node and power mobile apps using React Native.

### 5.2.2 Cheerio

Cheerio parses markup and provides an API for traversing/manipulating the resulting data structure. It does not interpret the result as a web browser does. Specifically, it does not produce a visual rendering, apply CSS, load external resources, or execute JavaScript which is common for a SPA (single page application). This makes Cheerio much, much faster than other solutions. If your use case requires any of this functionality, you should consider browser automation software like Puppeteer and Playwright or DOM emulation projects like JSDom.

### 5.2.3 d3

D3 (or D3.js) is a JavaScript library for visualizing data using web standards. D3 helps you bring data to life using SVG, Canvas and HTML. D3 combines powerful visualization and interaction techniques with a data-driven approach to DOM manipulation, giving you the full capabilities of modern browsers and the freedom to design the right visual interface for your data.

### 5.2.4 Highcharts

Highcharts is a JavaScript charting library based on SVG rendering. This project includes Stock, the financial charting package, the Maps package for geo maps and the Gantt package. This package is intended for supporting client-side JavaScript charting through bundlers like Parcel or Webpack and environments like Babel or TypeScript. If you intend to generate static charts on the server side, use the Highcharts node.js Export Server instead.

## 5.3 SPRINT BREAKDOWN

### 5.3.1 Write React app code

Frontend application had a text field where users could type in a URL and then perform one of the following operations:

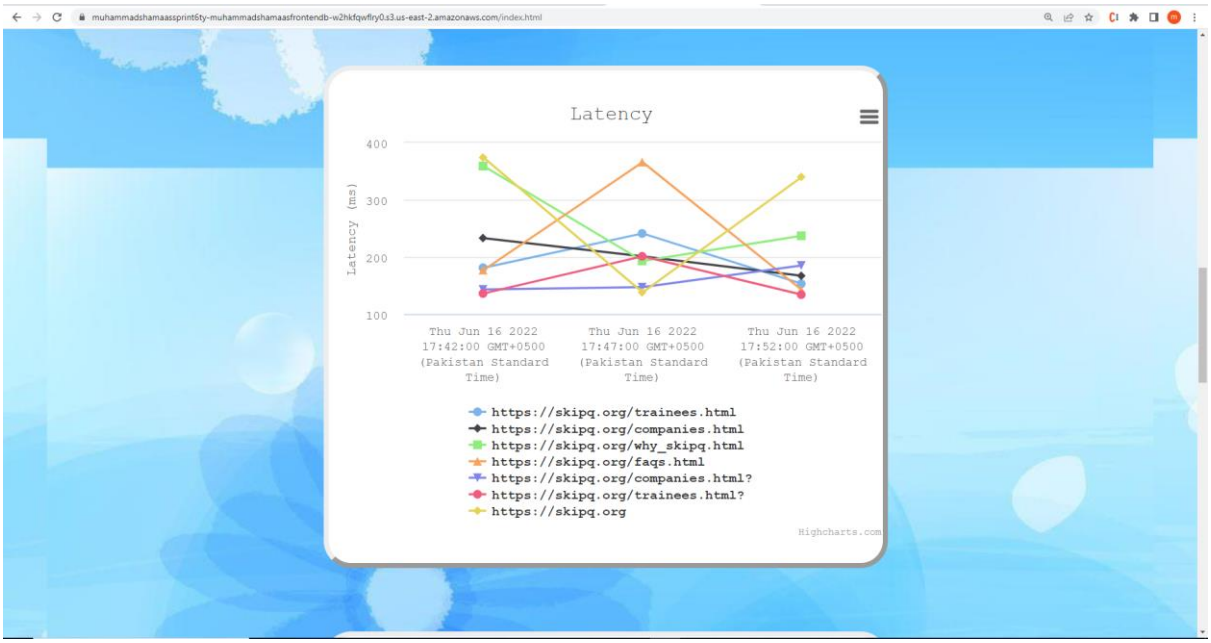
1. Add a new URL to the DB (MongoDB). The message “Already Exists” was returned if the added URL was already present in the DB. When implementing this operation, I checked that the value entered in the text field was a valid URL.

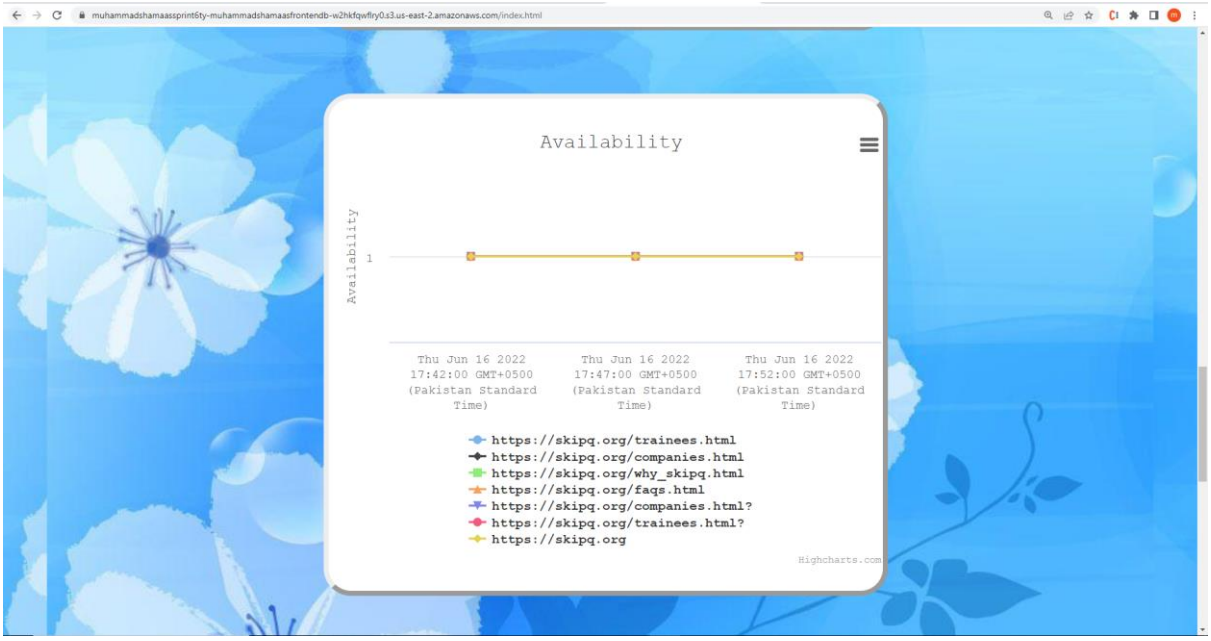
2. Delete a URL from the DB. “Not Found” error was returned if the URL was not present in the DB. When implementing this operation, I checked that the value entered in the text field was a valid URL.

3. Search the DB for a particular URL. “Not Found” error was returned if the URL was not present in the DB.

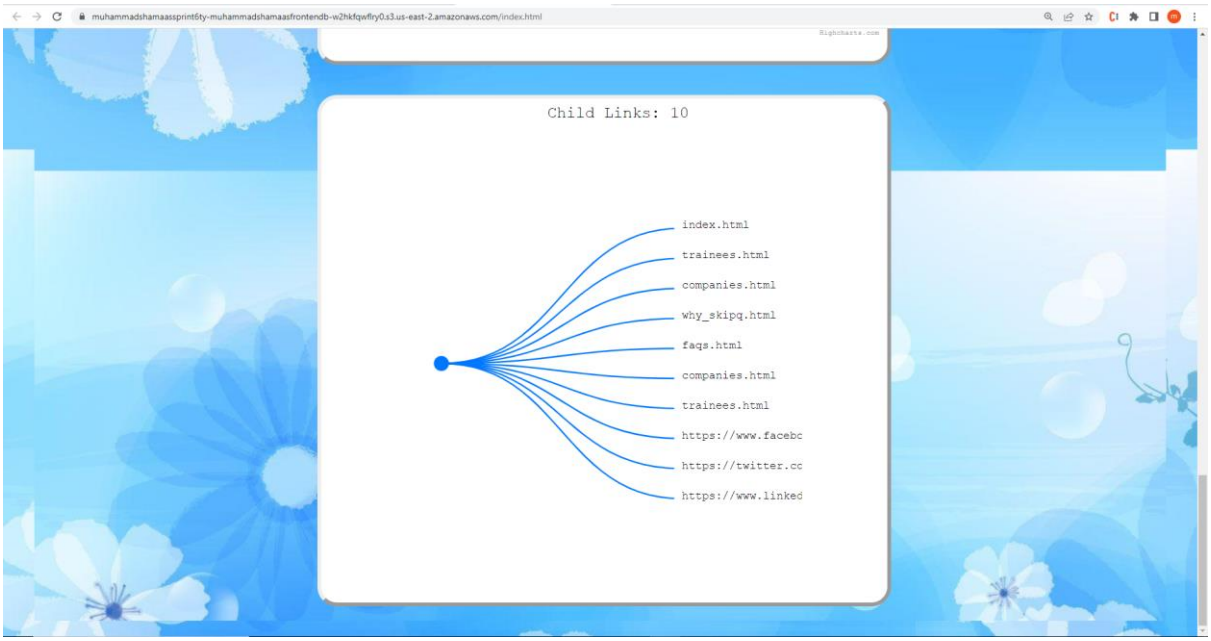

If the URL was found then it showed a table with the URL and all its sub-URLs. The table was paginated with only 5 entries showing per page.


It also plotted the CloudWatch metrics for Latency and Availability for that URL in the past 1 week. I used HighCharts.js library for plotting the time-series data. These graphs remained visible as the user scrolled through pages.

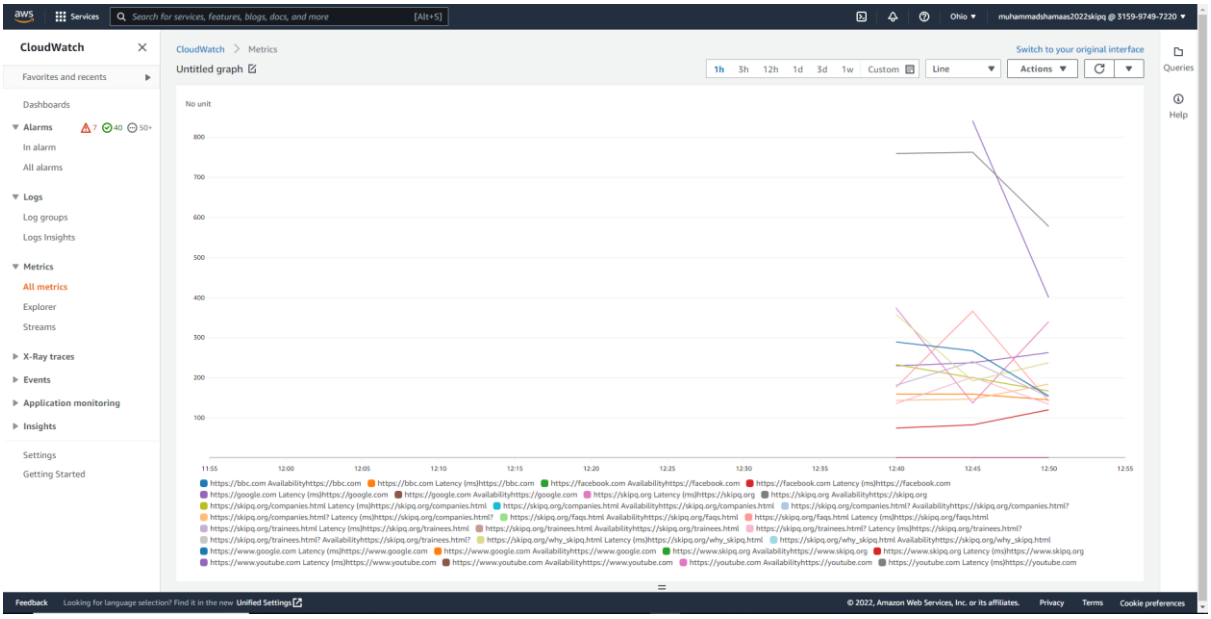


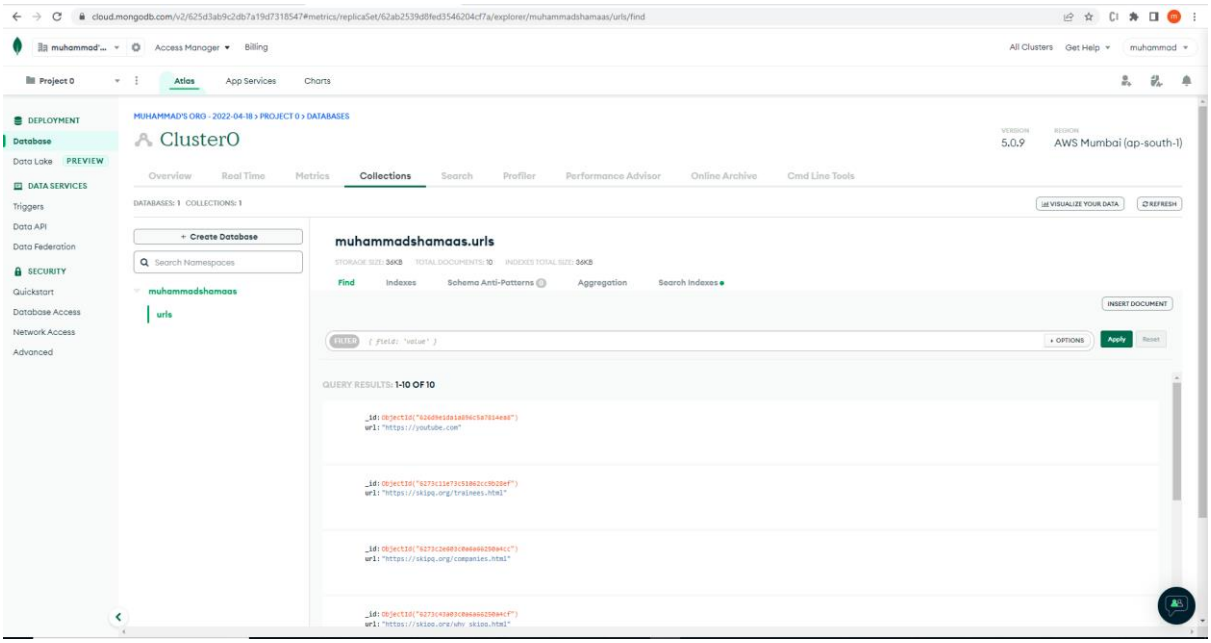

If the URL had child links then it plotted the 10 child links using D3.js library's Hierarchy graphs. The graphs remained visible as the user scrolled through pages.




The availability and latency metrics were retrieved from AWS Cloudwatch.

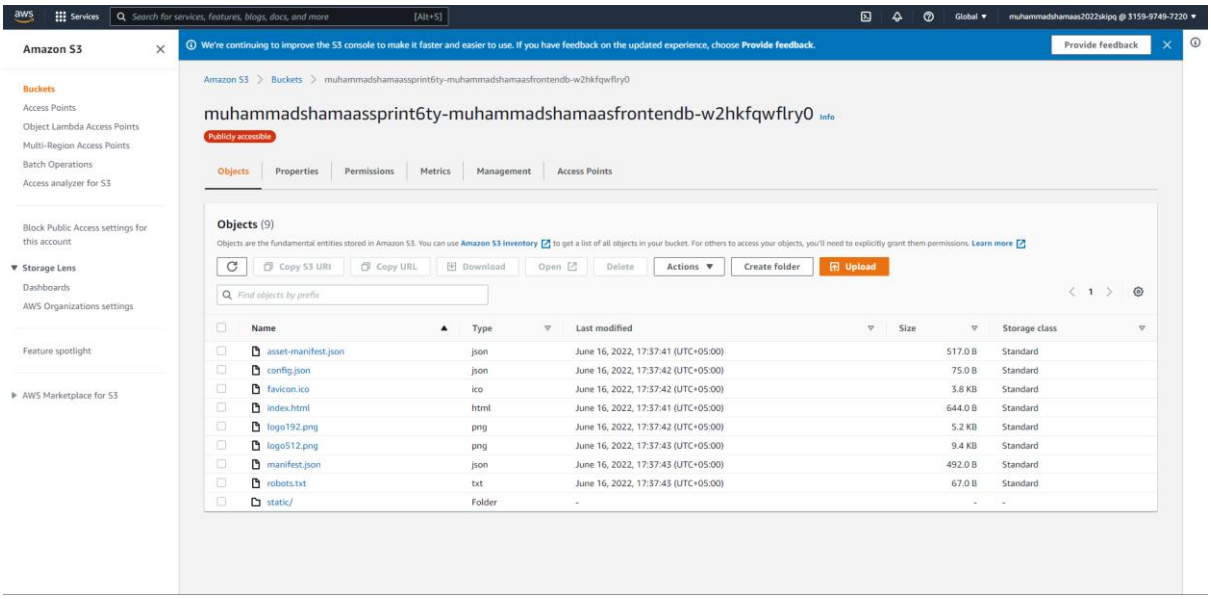



Whenever a URL was added or deleted, the application updated the entries in the MongoDB database.

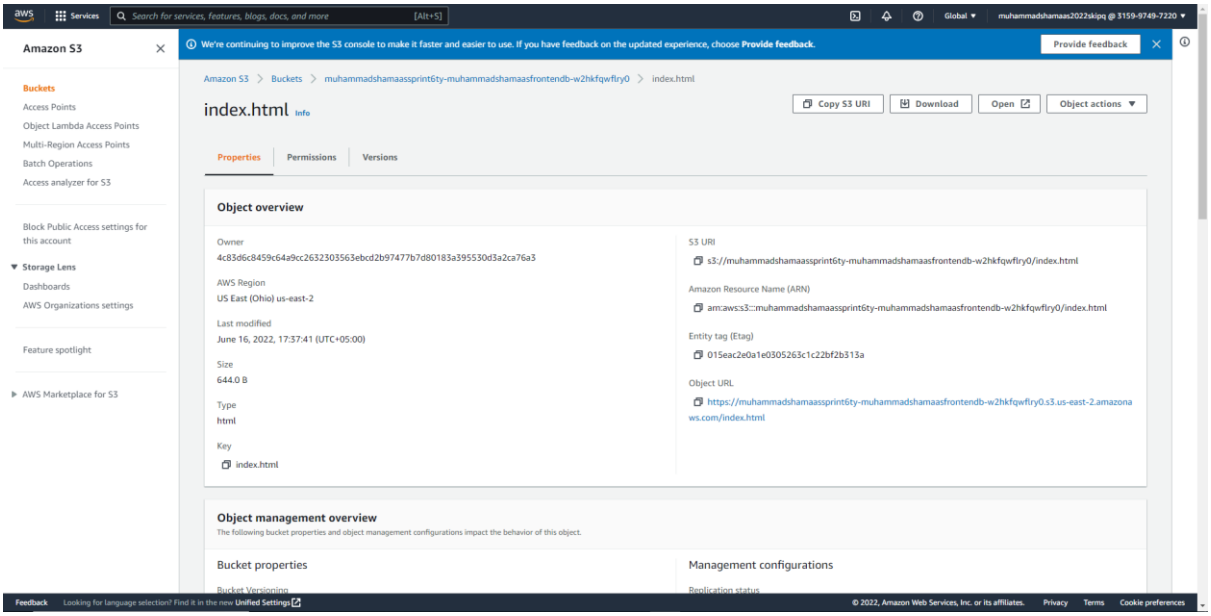



### 5.3.2 Build React app and upload to S3 bucket

The react application was built using npm run build. The build directory was stored in an S3 bucket so that it was publicly accessible.




The react application was accessed using the URI of index.html file stored in the front-end S3 bucket.




## 5.4 ISSUES/ERRORS AND TROUBLESHOOTING

### 5.4.1 Issue: React version not supported

**Description**: d3 did not support react version 18.1.0

```
PS C:\Users\Shehreen\Desktop\Orion_TS\shehreen\sprint5_shehreen\frontend> npm install d3
npm ERR! code ERESOLVE
npm ERR! ERESOLVE could not resolve
npm ERR!
npm ERR! While resolving: @material-ui/core@4.12.4
npm ERR! Found: react@18.1.0
npm ERR! node_modules/react
npm ERR!   peer react@">=16.8.0" from @emotion/react@11.9.0
npm ERR!   node_modules/@emotion/react
npm ERR!     peer @emotion/react@"^11.0.0-rc.0" from @emotion/styled@11.8.1
npm ERR!     node_modules/@emotion/styled
npm ERR!       peerOptional @emotion/styled@"^11.3.0" from @mui/material@5.6.3
npm ERR!       node_modules/@mui/material
npm ERR!         peer @mui/material@"^5.0.0" from @mui/icons-material@5.6.2
npm ERR!         node_modules/@mui/icons-material
npm ERR!         1 more (the root project)
npm ERR!       3 more (@mui/styled-engine, @mui/system, the root project)
npm ERR!     peerOptional @emotion/react@"^11.5.0" from @mui/material@5.6.3
npm ERR!     node_modules/@mui/material
npm ERR!       peer @mui/material@"^5.0.0" from @mui/icons-material@5.6.2
npm ERR!       node_modules/@mui/icons-material
npm ERR!         @mui/icons-material@"^5.6.2" from the root project
npm ERR!       1 more (the root project)
npm ERR!     3 more (@mui/styled-engine, @mui/system, the root project)
npm ERR!   peer react@">=16.8.0" from @emotion/styled@11.8.1
npm ERR!   node_modules/@emotion/styled
npm ERR!     peerOptional @emotion/styled@"^11.3.0" from @mui/material@5.6.3
npm ERR!     node_modules/@mui/material
npm ERR!       peer @mui/material@"^5.0.0" from @mui/icons-material@5.6.2
npm ERR!       node_modules/@mui/icons-material
npm ERR!         @mui/icons-material@"^5.6.2" from the root project
npm ERR!       1 more (the root project)
npm ERR!     peerOptional @emotion/styled@"^11.3.0" from @mui/styled-engine@5.6.1
npm ERR!     node_modules/@mui/styled-engine
npm ERR!       @mui/styled-engine@"^5.6.1" from @mui/system@5.6.3
npm ERR! Conflicting peer dependency: react@17.0.2
npm ERR! node_modules/react
npm ERR!   peer react@"^16.8.0 || ^17.0.0" from @material-ui/core@4.12.4
npm ERR!   node_modules/@material-ui/core
```

**Solutions**: I uninstalled react 18 and installed react version 17.

## 5.5 FUNCTION AND CLASSES DOCUMENTATION

### 5.5.1 getURL (url): getURL(”https://skipq.org”)

| getURL(url) | |
|---|---|
| Description | Returns data of url including sub-URLs, CloudWatch metrics and child links. |
| Parameters | url (string): The input URL |
| Returns | Displays paginated material-UI table for sub-URLs of url.<br>Displays HighChart.js graphs of CloudWatch Availability and Latency metrics.<br>Displays d3.js graph of child links. |

### 5.5.2 addURL (url): addURL(”https://skipq.org”)

| addURL(url) | |
|---|---|
| Description | Adds url to MongoDB database. |
| Parameters | url (string): The input URL |
| Returns | Result of database entry update. |

### 5.5.3 deleteURL (url): deleteURL(”https://skipq.org”)

| deleteURL(url) | |
|---|---|
| Description | Deleted url from MongoDB database. |
| Parameters | url (string): The input URL |
| Returns | Result of database entry update. |

# 6 SPRINT 6

## 6.1 OBJECTIVE

I added authentication to the app through a login screen that used OAuth method (Auth0). Authentication allowed users to login using username/password, GitHub account, or Gmail account. I also wrote 15 test cases using Cypress for testing the functionality of React application.

## 6.2 TECHNOLOGIES USED

### 6.2.1 Auth0

Auth0 can be used to implement authentication with multiple identity providers, including social (e.g., Google, Facebook, Microsoft, LinkedIn, GitHub, Twitter, etc), or enterprise (e.g., Windows Azure AD, Google Apps, Active Directory, ADFS, SAML, etc.), log in users with username/password databases, passwordless, or multi-factor authentication, link multiple user accounts together, generate signed JSON Web Tokens to authorize your API calls and flow the user identity securely, access demographics and analytics detailing how, when, and where users are logging in and enrich user profiles from other data sources using customizable JavaScript rules.

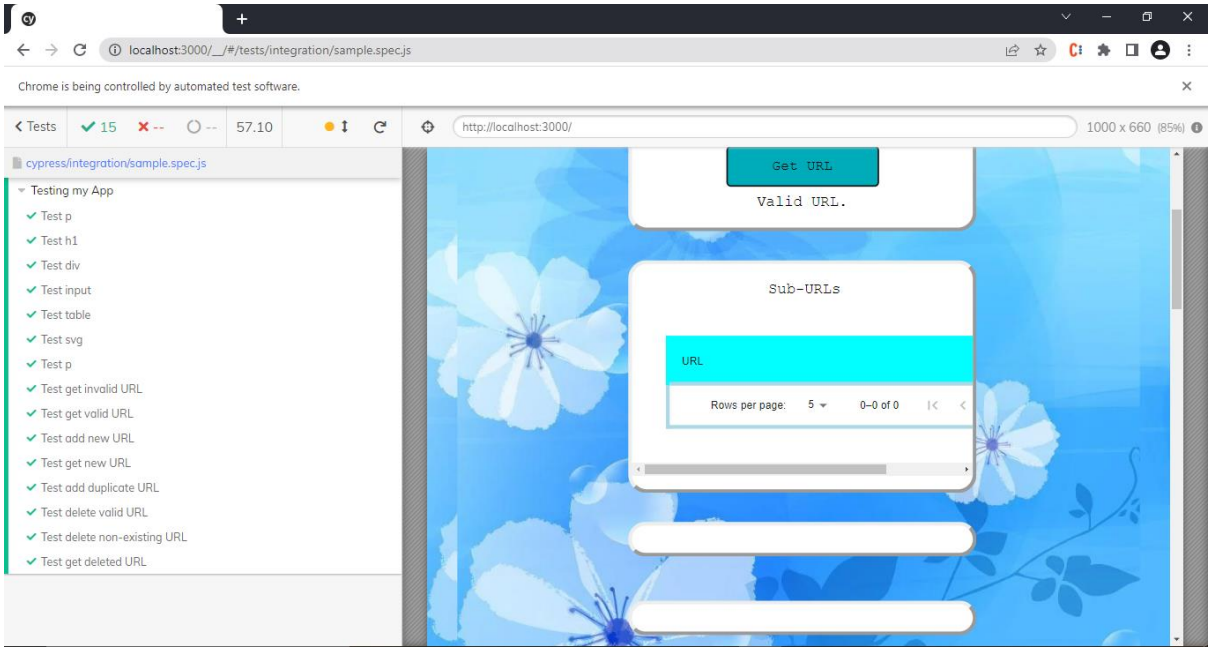
### 6.2.2 Cypress

Cypress is a next generation front end testing tool built for the modern web. We address the key pain points developers and QA engineers face when testing modern applications. We make it possible to set up tests, write tests, run tests and debug Tests. Cypress is most often compared to Selenium; however Cypress is both fundamentally and architecturally different. Cypress is not constrained by the same restrictions as Selenium. This enables you to write faster, easier and more reliable tests. Our users are typically developers or QA engineers building web applications using modern JavaScript frameworks. Cypress enables you to write all types of tests: End-to-end tests, Integration tests, Unit tests. Cypress can test anything that runs in a browser.

## 6.3 SPRINT BREAKDOWN

### 6.3.1 Task – 1: Functional testing using Cypress

The testing of the CRUD app was performed using Cypress. I wrote 15 test cases using Cypress for testing the functionality of React application.




- Initialization Step

  beforeEach(() => {

  cy.visit('http://localhost:3000')

  })

- Functional tests

  1. "Test p"

     cy.get("p").first().should('have.text', "")

  2. "Test h1"

     cy.get("h1").should('have.text', "Front End")

  3. "Test div"

     cy.get("div").should('have.length', 12)

  4. "Test input"

     cy.get("input").should('have.length', 4)

  5. "Test table"

     cy.get("Table").should('have.length', 1)

  6. "Test svg"

     cy.get("svg").should('have.length', 6)

  7. "Test p"

     cy.get("p").should('have.length', 5)

8. "Test get invalid URL"

```
cy.get("input").last().click()
cy.get("p").first().should('have.text', "Invalid URL.")
```

9. "Test get valid URL"

```
const newItem="https://skipq.org"
cy.get(".url").type(`${newItem}`)
cy.get("input").last().click()
cy.get("p").first().should('have.text', "Valid URL.")
```

10. "Test add new URL"

```
const newItem="https://bing.com"
cy.get(".url").type(`${newItem}`)
cy.get("#addbutton").click()
cy.get("p").first().should('have.text', '"Added"')
```

11. "Test get new URL"

```
const newItem="https://bing.com"
cy.get(".url").type(`${newItem}`)
cy.get("input").last().click()
cy.get("p").first().should('have.text', "Valid URL.")
```

12. "Test add duplicate URL"

```
const newItem="https://bing.com"
cy.get(".url").type(`${newItem}`)
cy.get("#addbutton").click()
cy.get("p").first().should('have.text', '"Already Exists"')
```

13. "Test delete valid URL"

```
const newItem="https://bing.com"
cy.get(".url").type(`${newItem}`)
cy.get("#deletebutton").click()
cy.get("p").first().should('have.text', '"Deleted"')
```

14. "Test delete non-existing URL"

```
const newItem="https://bing.com"
cy.get(".url").type(`${newItem}`)
cy.get("#deletebutton").click()
```

```
cy.get("p").first().should('have.text', '"Not Found"')
```

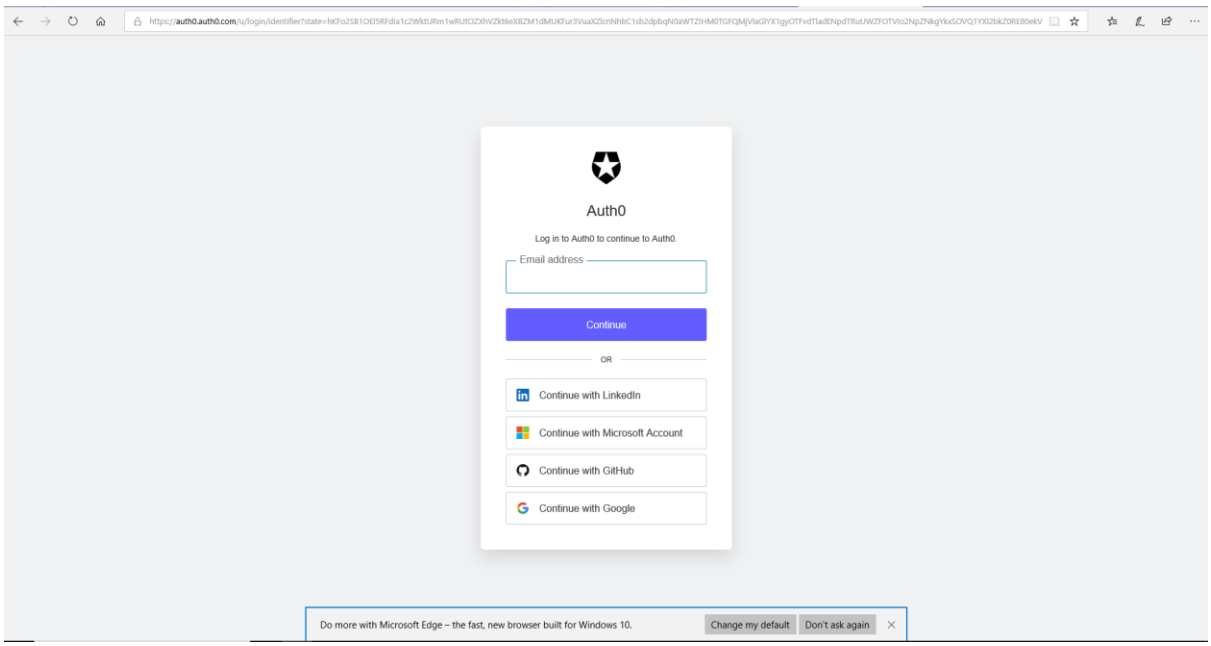
15. "Test get deleted URL"

```
const newItem="https://bing.com"
cy.get(".url").type(`${newItem}`)
cy.get("input").last().click()
cy.get("p").first().should('have.text', 'Valid URL.')
```

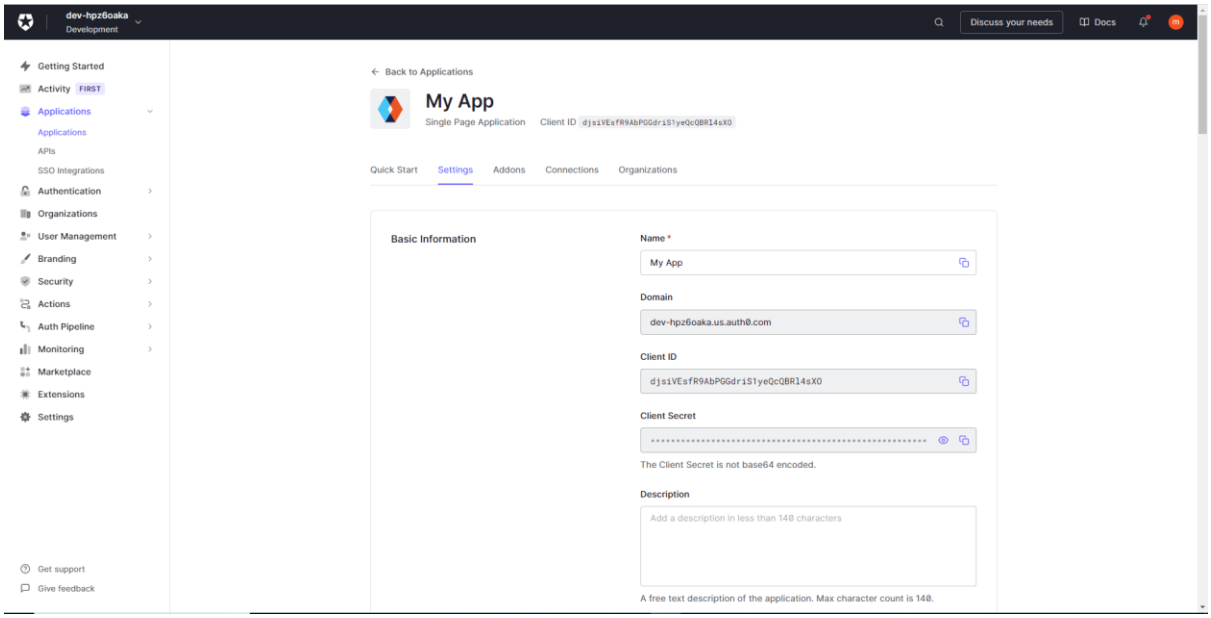
### 6.3.2 Task – 1: Authorization using Auth0

The authorization was performed using Auth0. I added authentication to the app through a login screen that used OAuth method (Auth0). Authentication allowed users to login using username/password, GitHub account, or Gmail account.

The login button directed user to authorization page. The logout button was used to exit the application.



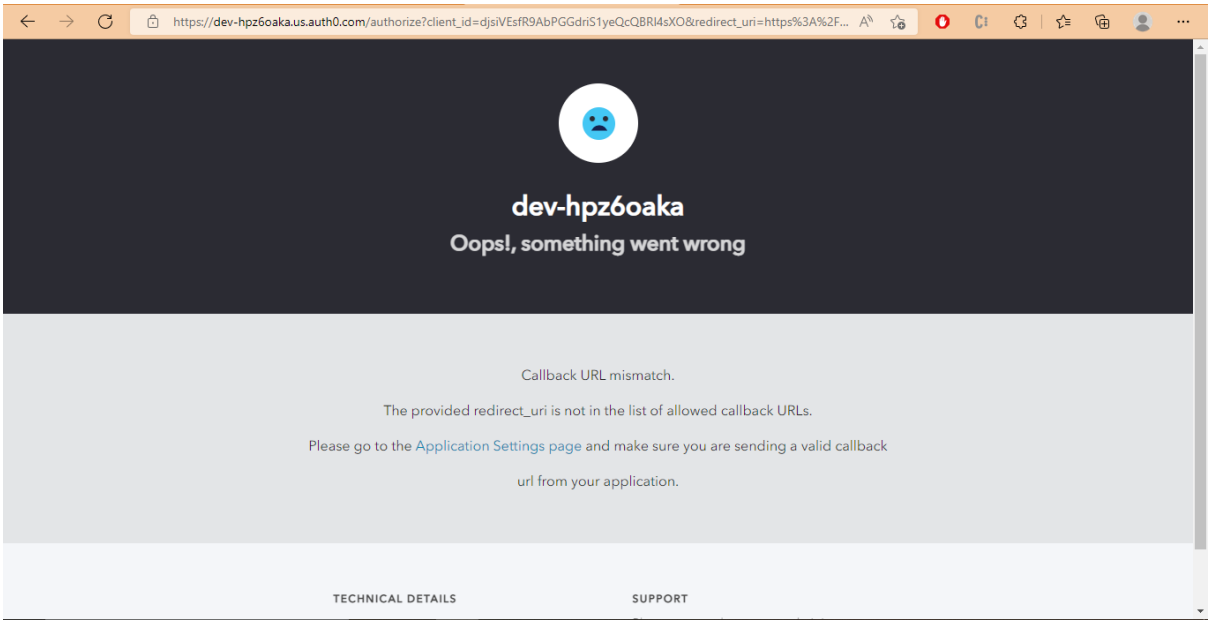

The login button directed the user to Auth0 authentication page. This was generated based on the application domain and ClientID.



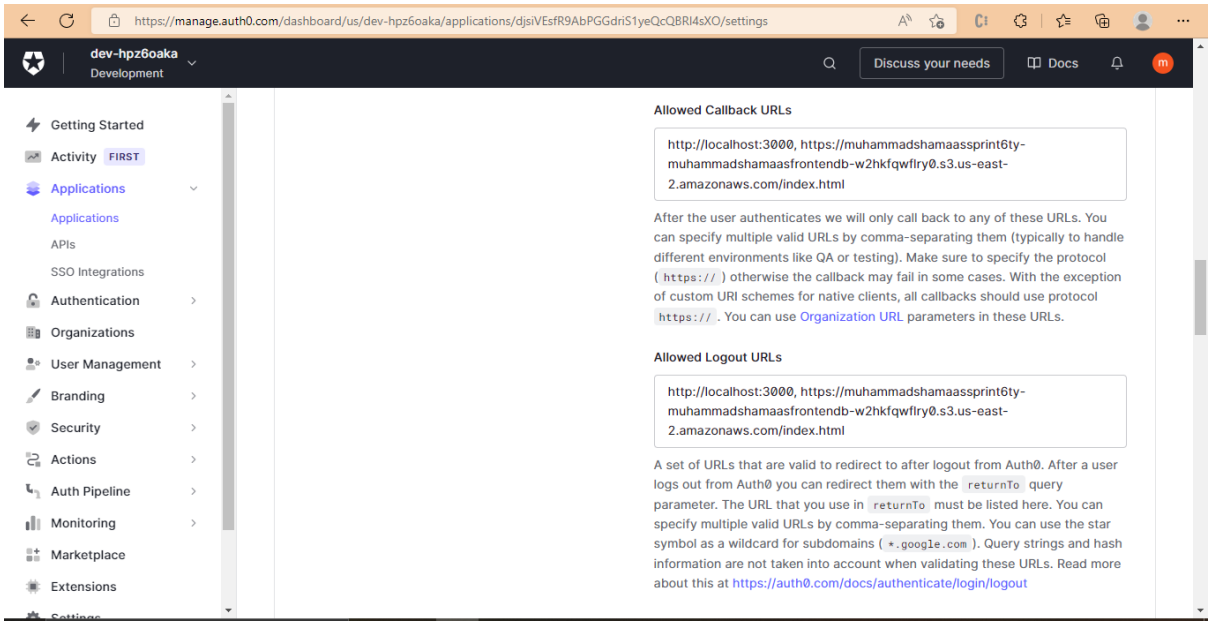

## 6.4 Issues/Errors and Troubleshooting

### 6.4.1 Issue: Callback URL mismatch

**Description**: Auth0 was not given allowed callback URLs list and allowed logout URLs.




**Solutions**: The lists of allowed callback and logout URLs was updated to include URI of index.html file.




## 6.5 FUNCTION AND CLASSES DOCUMENTATION

### 6.5.1 Conditional rendering for Auth0 authentication

```
if (!isAuthenticated){
        return (<LoginButton></LoginButton>);}
else{
        return (<html>…<LogoutButton></LogoutButton>…</html>);
}
```

The login button directed user to authorization page.

```
const LoginButton=()=>{
  const {loginWithRedirect, isAuthenticated}=useAuth0();
  return(!isAuthenticated && (<button onClick={()=>{loginWithRedirect();}}>Login</button>
    ));
}
```

The logout button was used to exit the application.

```
const LogoutButton=()=>{
  const {logout,isAuthenticated}=useAuth0();
  return(isAuthenticated && (<button onClick={()=>{logout();}}>Logout</button>));
}
```

# 7 REFERENCES


https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Programming.Errors.html

https://blog.hubspot.com/website/what-is-github-used-for

https://github.com/

https://www.mongodb.com/

https://aws.amazon.com/cloud9/

https://aws.amazon.com/s3/

https://aws.amazon.com/ec2/

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

https://aws.amazon.com/iam/

https://aws.amazon.com/cloudwatch/

https://aws.amazon.com/codepipeline/

https://aws.amazon.com/api-gateway/

https://aws.amazon.com/secrets-manager/

https://aws.amazon.com/sns/

https://aws.amazon.com/cloudformation/

https://www.npmjs.com/package/@auth0/auth0-react

https://www.npmjs.com/package/@emotion/react

https://www.npmjs.com/package/@emotion/styled

https://www.npmjs.com/package/@mui/icons-material

https://www.npmjs.com/package/@mui/material

https://www.npmjs.com/package/@mui/styled-engine-sc

https://www.npmjs.com/package/@testing-library/jest-dom

https://www.npmjs.com/package/@testing-library/user-event

https://www.npmjs.com/package/aws-cdk-lib

https://www.npmjs.com/package/aws-sdk

https://www.npmjs.com/package/axios

https://www.npmjs.com/package/body-parser

https://www.npmjs.com/package/cheerio

https://www.npmjs.com/package/d3

https://www.npmjs.com/package/express

https://www.npmjs.com/package/got

https://www.npmjs.com/package/highcharts

https://www.npmjs.com/package/mongodb

https://www.npmjs.com/package/react

https://www.npmjs.com/package/react-dom

https://www.npmjs.com/package/react-scripts

https://www.npmjs.com/package/styled-components

https://www.npmjs.com/package/util

https://www.npmjs.com/package/web-vitals

https://www.npmjs.com/package/cypress