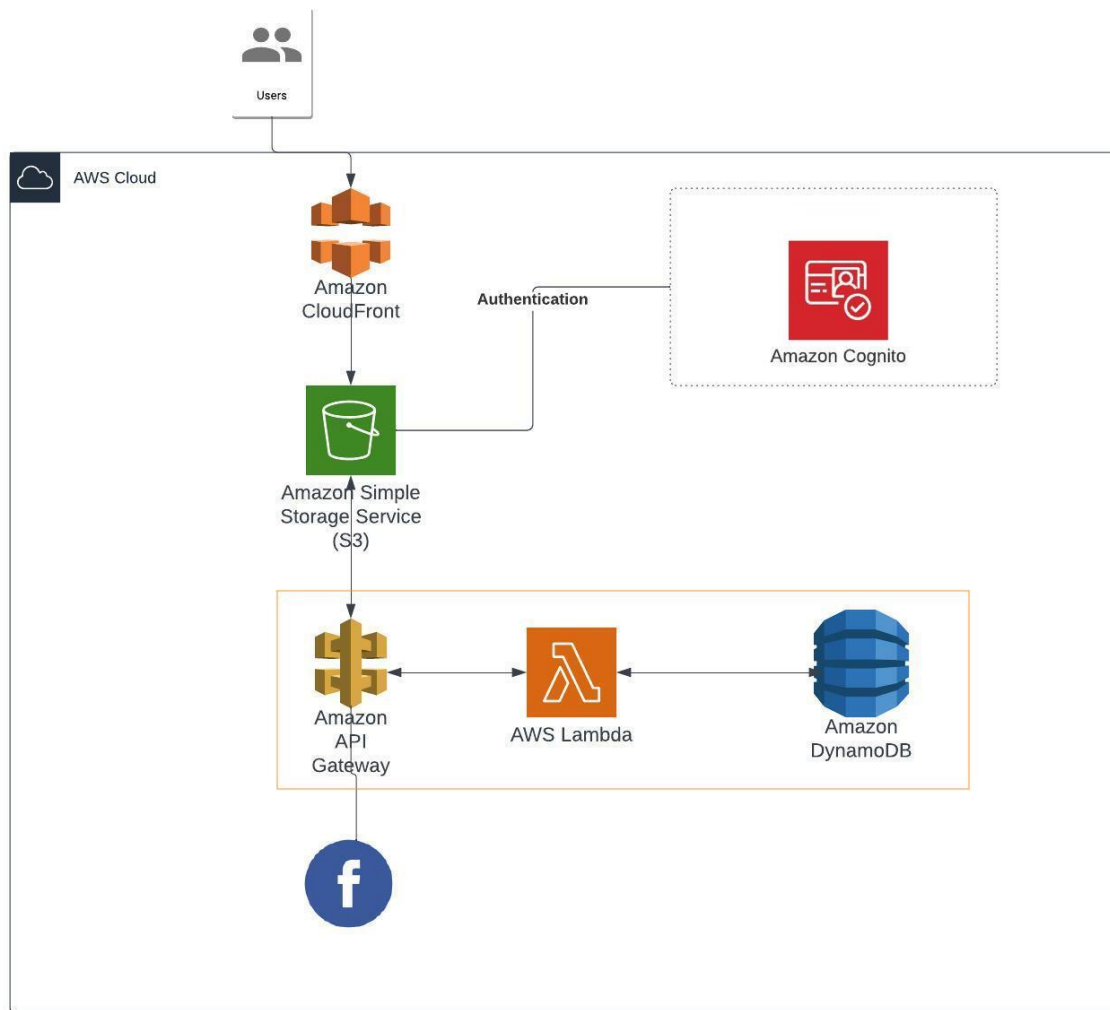


Assignment Full

Official System Architecture

This section describes the high level system design and the architecture of Coin 360. Coin 360 is a server-less web application deployed using Amazon webs services. Coin 360 is written in JavaScript using the react framework. Below is the system architecture of the application



Code Description

Dynamo DB:

The first dynamo db table is used to store a selected range of coins to be selected from when creating a new coin entry. The second table is used to store the record that is created by the user.

The id is the main component of the record

```

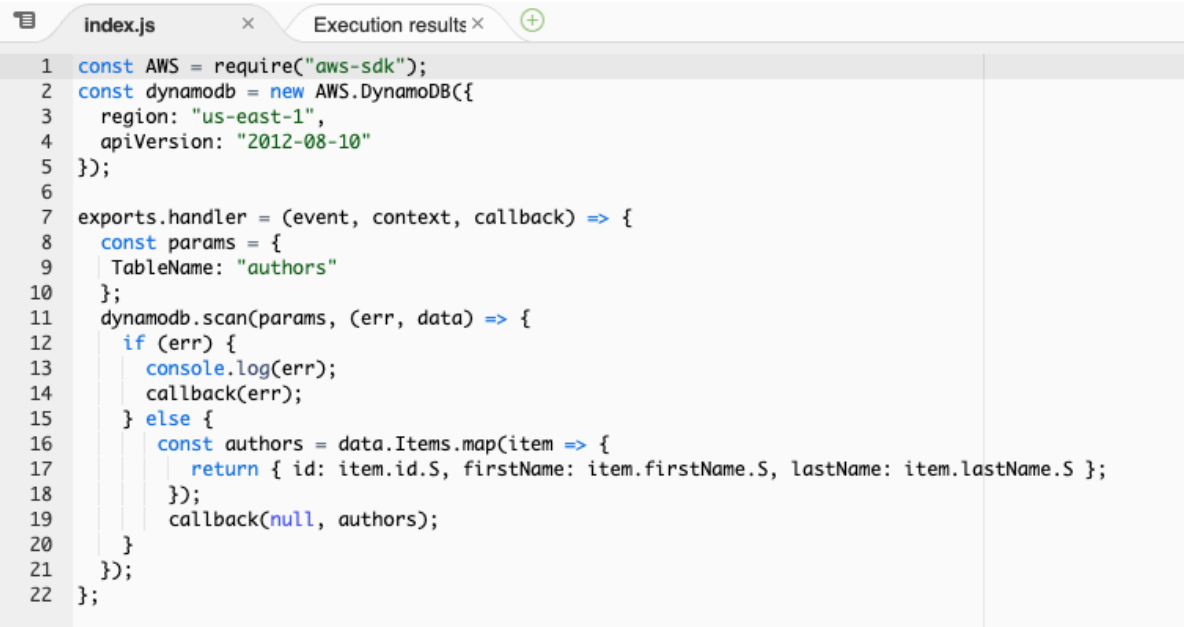
1 {
2   "id": "Solana",
3   "lastName": "",
4   "firstName": "Solana"
5 }

```

Lambda:

Five lambda roles were created for the following uses:

- Getting all records for the user to choose a coin from



```

index.js
Execution results
1 const AWS = require("aws-sdk");
2 const dynamodb = new AWS.DynamoDB({
3   region: "us-east-1",
4   apiVersion: "2012-08-10"
5 });
6
7 exports.handler = (event, context, callback) => {
8   const params = {
9     TableName: "authors"
10  };
11  dynamodb.scan(params, (err, data) => {
12    if (err) {
13      console.log(err);
14      callback(err);
15    } else {
16      const authors = data.Items.map(item => {
17        return { id: item.id.S, firstName: item.firstName.S, lastName: item.lastName.S };
18      });
19      callback(null, authors);
20    }
21  });
22 };

```

- Get all records that are saved and made by the user

```

1  const AWS = require("aws-sdk");
2  const dynamodb = new AWS.DynamoDB({
3    region: "us-east-1",
4    apiVersion: "2012-08-10"
5  });
6
7  exports.handler = (event, context, callback) => {
8    const params = {
9      TableName: "courses"
10   };
11   dynamodb.scan(params, (err, data) => {
12     if (err) {
13       console.log(err);
14       callback(err);
15     } else {
16       const courses = data.Items.map(item => {
17         return {
18           id: item.id.S,
19           title: item.title.S,
20           watchHref: item.watchHref.S,
21           authorId: item.authorId.S,
22           length: item.length.S,
23           category: item.category.S
24         };
25       });
26       callback(null, courses);
27     }
28   });
29 };

```

- Get a single record that is saved and made by the user

```

1  const AWS = require("aws-sdk");
2  const dynamodb = new AWS.DynamoDB({
3    region: "us-east-1",
4    apiVersion: "2012-08-10"
5  });
6
7  exports.handler = (event, context, callback) => {
8    const params = {
9      Key: {
10        id: {
11          S: event.id
12        }
13      },
14      TableName: "courses"
15    };
16    dynamodb.getItem(params, (err, data) => {
17      if (err) {
18        console.log(err);
19        callback(err);
20      } else {
21        callback(null, {
22          id: data.Item.id.S,
23          title: data.Item.title.S,
24          watchHref: data.Item.watchHref.S,
25          authorId: data.Item.authorId.S,
26          length: data.Item.length.S,
27          category: data.Item.category.S
28        });
29      }
30    });
31  };

```

- Update(Edit) a chosen record

```

1  const AWS = require("aws-sdk");
2  const dynamodb = new AWS.DynamoDB({
3    region: "us-east-1",
4    apiVersion: "2012-08-10"
5  });
6
7  exports.handler = (event, context, callback) => {
8    const params = {
9      Item: {
10        id: {
11          S: event.id
12        },
13        title: {
14          S: event.title
15        },
16        watchHref: {
17          S: event.watchHref
18        },
19        authorId: {
20          S: event.authorId
21        },
22        length: {
23          S: event.length
24        },
25        category: {
26          S: event.category
27        }
28      },
29      TableName: "courses"
30    };
31    dynamodb.putItem(params, (err, data) => {
32      if (err) {
33        console.log(err);
34        callback(err);
35      } else {
36        callback(null, {
37          id: params.Item.id.S,
38          title: params.Item.title.S,
39          watchHref: params.Item.watchHref.S,
40          authorId: params.Item.authorId.S,
41          length: params.Item.length.S,
42          category: params.Item.category.S
43        });
44      }
45    });
46  };

```

- Delete a chosen record

```

1  const AWS = require("aws-sdk");
2  const dynamodb = new AWS.DynamoDB({
3      region: "us-east-1",
4      apiVersion: "2012-08-10"
5  });
6
7  exports.handler = (event, context, callback) => {
8      const params = {
9          Key: {
10             id: {
11                 S: event.id
12             }
13         },
14         TableName: "courses"
15     };
16     dynamodb.deleteItem(params, (err, data) => {
17         if (err) {
18             console.log(err);
19             callback(err);
20         } else {
21             callback(null, data);
22         }
23     });
24 };

```

API Gateway:

The Rest API is created and managed by AWS and is used for the front end services of the application the GET POST and DELETE methods are created to manage the database interactions using the Lambda function.



Developer/Use Manual

The application was originally developed using a local server but now is uploaded to AWS S3 to hold the files and also to be accessed by the public.

<https://d1asnh0towkht9.cloudfront.net/>

However the code can also be used locally using the following steps located in the ReadMe text file.

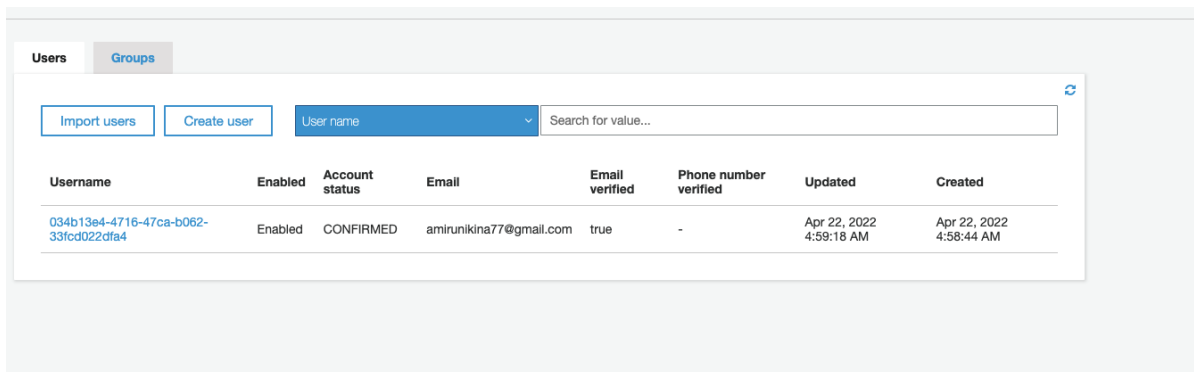

```

1 //To load the app on the local machine
2 Step 1: Download the files to chosen location(Ensure javascript is loaded)
3 Step 2: Open Command Terminal and navigate to the file
4 Step 3: First Run: "npm install" to download all node modules needed to run the application or else there will be an error
5 Step 4: Then Run: "npm start" to begin the server and launch the application locally

```

AWS Cognito:

AWS cognito is used to manage access to the web application, this also allows for log in and registration functionality.

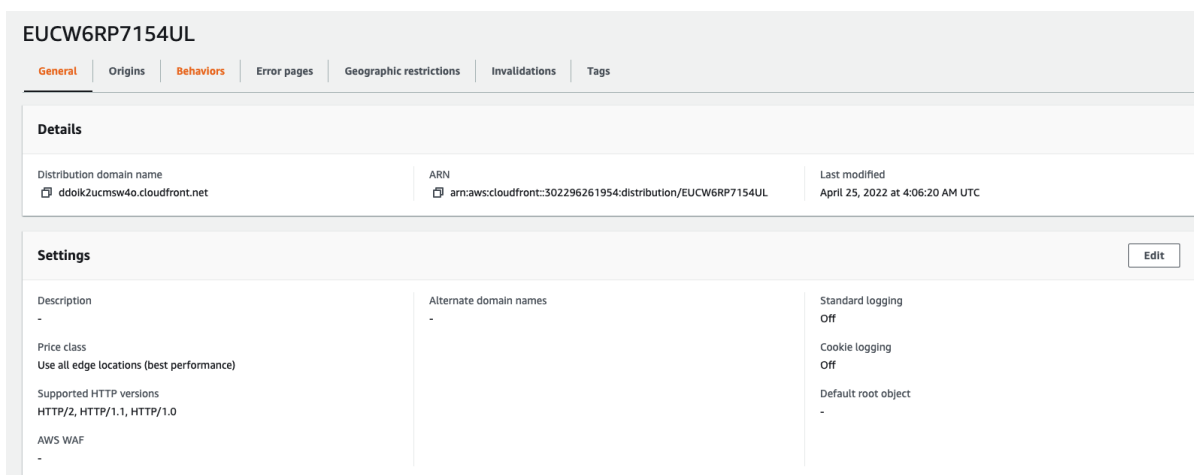


The screenshot shows the 'Users' tab in the AWS Cognito console. It includes buttons for 'Import users' and 'Create user', a search bar, and a table of existing users.

Username	Enabled	Account status	Email	Email verified	Phone number verified	Updated	Created
034b13e4-4716-47ca-b062-33fcd022dfa4	Enabled	CONFIRMED	amirunikina77@gmail.com	true	-	Apr 22, 2022 4:59:18 AM	Apr 22, 2022 4:58:44 AM

AWS CloudFront:

AWS Cloud front is used to allow the application to load faster

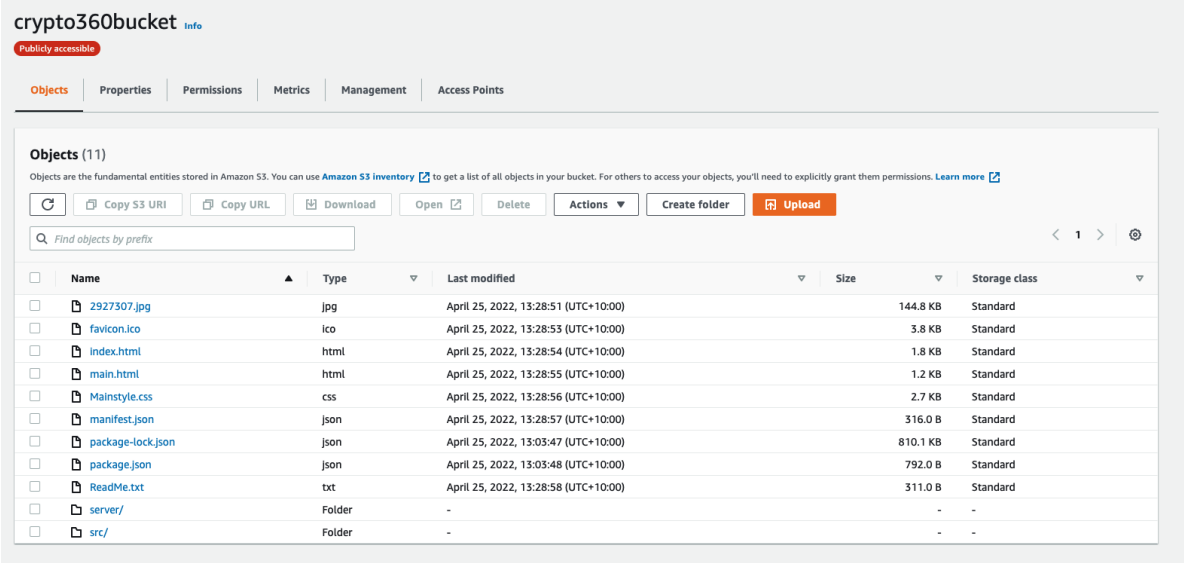


The screenshot shows the AWS CloudFront console for a distribution named 'EUCW6RP7154UL'. It includes tabs for 'General', 'Origins', 'Behaviors', 'Error pages', 'Geographic restrictions', 'Invalidations', and 'Tags'. The 'Details' section shows the distribution domain name, ARN, and last modified date. The 'Settings' section shows various configuration options like description, price class, supported HTTP versions, and logging.

EUCW6RP7154UL		
General	Origins	Behaviors
Details Distribution domain name: ddoik2ucmsw4o.cloudfront.net ARN: arn:aws:cloudfront::302296261954:distribution/EUCW6RP7154UL Last modified: April 25, 2022 at 4:06:20 AM UTC		
Settings Description: - Price class: Use all edge locations (best performance) Supported HTTP versions: HTTP/2, HTTP/1.1, HTTP/1.0 AWS WAF: - Alternate domain names: - Standard logging: Off Cookie logging: Off Default root object: -		

AWS S3:

Stores the files for the application including the media



The screenshot displays the AWS S3 console interface for a bucket named 'crypto360bucket'. The bucket is publicly accessible. The 'Objects' tab is selected, showing a list of 11 objects. The objects include various file types such as .jpg, .ico, .html, .css, .json, .txt, and folders. The table lists the object name, type, last modified date, size, and storage class.

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	2927307.jpg	jpg	April 25, 2022, 13:28:51 (UTC+10:00)	144.8 KB	Standard
<input type="checkbox"/>	favicon.ico	ico	April 25, 2022, 13:28:53 (UTC+10:00)	3.8 KB	Standard
<input type="checkbox"/>	index.html	html	April 25, 2022, 13:28:54 (UTC+10:00)	1.8 KB	Standard
<input type="checkbox"/>	main.html	html	April 25, 2022, 13:28:55 (UTC+10:00)	1.2 KB	Standard
<input type="checkbox"/>	Mainstyle.css	css	April 25, 2022, 13:28:56 (UTC+10:00)	2.7 KB	Standard
<input type="checkbox"/>	manifest.json	json	April 25, 2022, 13:28:57 (UTC+10:00)	316.0 B	Standard
<input type="checkbox"/>	package-lock.json	json	April 25, 2022, 13:03:47 (UTC+10:00)	810.1 KB	Standard
<input type="checkbox"/>	package.json	json	April 25, 2022, 13:03:48 (UTC+10:00)	792.0 B	Standard
<input type="checkbox"/>	ReadMe.txt	txt	April 25, 2022, 13:28:58 (UTC+10:00)	311.0 B	Standard
<input type="checkbox"/>	server/	Folder	-	-	-
<input type="checkbox"/>	src/	Folder	-	-	-

References:

[1]"Facebook for Developers", *Developers.facebook.com*, 2022. [Online]. Available: <https://developers.facebook.com>. [Accessed: 25- Apr- 2022].

[2]"Module Five", *Amazon Web Services, Inc.*, 2022. [Online]. Available: <https://aws.amazon.com/getting-started/hands-on/build-serverless-web-app-lambda-apigateway-s3-dynamodb-cognito/module-5/>. [Accessed: 25- Apr- 2022].

[3]E. Herrera and B. Services, "Build a Serverless Web App on AWS Services | Pluralsight | Pluralsight", *Pluralsight.com*, 2022. [Online]. Available: <https://www.pluralsight.com/guides/building-a-serverless-web-app-on-aws-services>. [Accessed: 25- Apr- 2022].