FIXEROO

Developer Manual

CREATE A DYNAMO DB TABLE

Create a dynamo DB table at https://console.aws.amazon.com/dynamodb/
Call the table ‘fixeroo-handymen’
set ‘email’ as the primary key

CREATE AN S3 BUCKET

Create an S3 bucket via the S3 Console
Call the bucket ‘handymen-avatars’
Deselect the box ‘bock all public access’
Go to permission tab
Attach the following bucket policy

```
{
    "Version": "2008-10-17",
    "Statement": [
        {
            "Sid": "AllowPublicRead",
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::handymen-avatars/*"
        }
    ]
}
```

CREATE A LAMBDA FUNCTION

Create a Lambda function called ‘get-handymen’
Under Permissions choose Change default execution role.
Select Create a new role from AWS policy templates.
For role name write ‘fixeroo-role’
From the policy templates choose ‘Simple microservice permissions’ to allow the lambda function to interact with DynamoDB
Choose create function

Click on the Fixeroo-role to be taken to its IAM management console
Under permissions, click ‘Attach Policies’ and add S3 write access to allow the lambda function to write to S3
Open index.js of the lambda function and copy in the following code:

```
const AWS = require("aws-sdk");
AWS.config.update({region: 'us-east-1'});
var s3 = new AWS.S3({apiVersion: '2006-03-01'});
var uploadParams = {Bucket: 'handymen-avatars', Key: '', Body: ''};
var fs = require('fs');

const dynamo = new AWS.DynamoDB.DocumentClient();

exports.handler = async (event, context) => {
  let body;
  let statusCode = 200;
  const headers = {
    "Content-Type": "application/json",
    "Access-Control-Allow-Origin" : "*", // Required for CORS support to work
    "Access-Control-Allow-Credentials" : true, // Required for cookies, authorization
headers with HTTPS
    "Access-Control-Allow-Methods" : "GET, POST, PUT, OPTIONS",
    "Access-Control-Allow-Headers" : "Origin, Content-Type, Accept, Response-Type"
  };

  try {
    switch (event.routeKey) {
      case "DELETE /handymen/{email}":
        await dynamo
          .delete({
            TableName: "fixeroo-handymen",
            Key: {
              email: event.pathParameters.email
            }
          })
          .promise();
        body = `Deleted item ${event.pathParameters.email}`;
        break;
      case "GET /handymen/{email}":
        body = await dynamo
          .get({
            TableName: "fixeroo-handymen",
            Key: {
              email: event.pathParameters.email
            }
          })
          .promise();
        break;
      case "GET /handymen":
        body = await dynamo.scan({ TableName: "fixeroo-handymen" }).promise();
        break;
      case "PUT /handymen":
        let requestJSON = JSON.parse(event.body);
        await dynamo
          .put({
            TableName: "fixeroo-handymen",
            Item: {
              email: requestJSON.email,
              firstName: requestJSON.firstName,
              lastName: requestJSON.lastName,
              password: requestJSON.password
            }
          })
          .promise();
        body = `Put item ${requestJSON.email}`;
        break;
      case "PUT /handymen/{email}":
        let request = JSON.parse(event.body);

        await dynamo
          .put({
            TableName: "fixeroo-handymen",
            Key: {
              email: event.pathParameters.email
            },
            Item: {
              email: request.email,
              firstName: request.firstName,
              lastName: request.lastName,
              password: request.password,
              field: request.field,
              experience: request.experience,
              price: request.price,
              neatness: request.neatness,
              safety: request.safety,
              pets: request.pets,
              policeCheck: request.policeCheck,
            }
          })
          .promise();
        body = `Put item ${request.email}`;
        break;
      case "PUT /handymen/{email}/review":
        let requestReview = JSON.parse(event.body);

        let user = await dynamo
          .get({
            TableName: "fixeroo-handymen",
            Key: {
              email: event.pathParameters.email
            }
          })
          .promise();

        var review = [{
          rating: requestReview.rating,
          message: requestReview.message
        }]

        var upsertExpr = (user.Item.reviews == undefined) ? " :r" :
"list_append(#reviews, :r)";

        await dynamo
          .update({
            TableName: "fixeroo-handymen",
            Key: {
              email: event.pathParameters.email
            },
            ProjectionExpression:"#reviews",
            ExpressionAttributeNames:{
                "#reviews": "reviews"
            },
            UpdateExpression: "SET #reviews = " + upsertExpr ,
            ExpressionAttributeValues:{
                ":r": review
            }
          })
          .promise();
        body = `Put item ${requestReview.email}`;
        break;
      case "PUT /image":
        let file = JSON.parse(event.body);
        console.log("filename: " + file.key);

        uploadParams.Body = Buffer.from(file.data, "base64"),
        uploadParams.Key = file.key;

        // call S3 to retrieve upload file to specified bucket
        await s3.upload(uploadParams, function (err, data) {
          console.log("UPLOADING");
          if (err) {
            console.log("Error", err);
          } if (data) {
            console.log("Upload Success", data.Location);
          }
          else{
            console.log("Error", err);
          }
          console.log("DONE");
        }).promise();
        body = `Put item ${file.key}`;
        break;
      default:
        throw new Error(`Unsupported route: "${event.routeKey}"`);
    }
  } catch (err) {
    statusCode = 400;
    body = err.message;
  } finally {
    body = JSON.stringify(body);
  }

  return {
    statusCode,
    body,
    headers
  };
};
```

Press deploy.

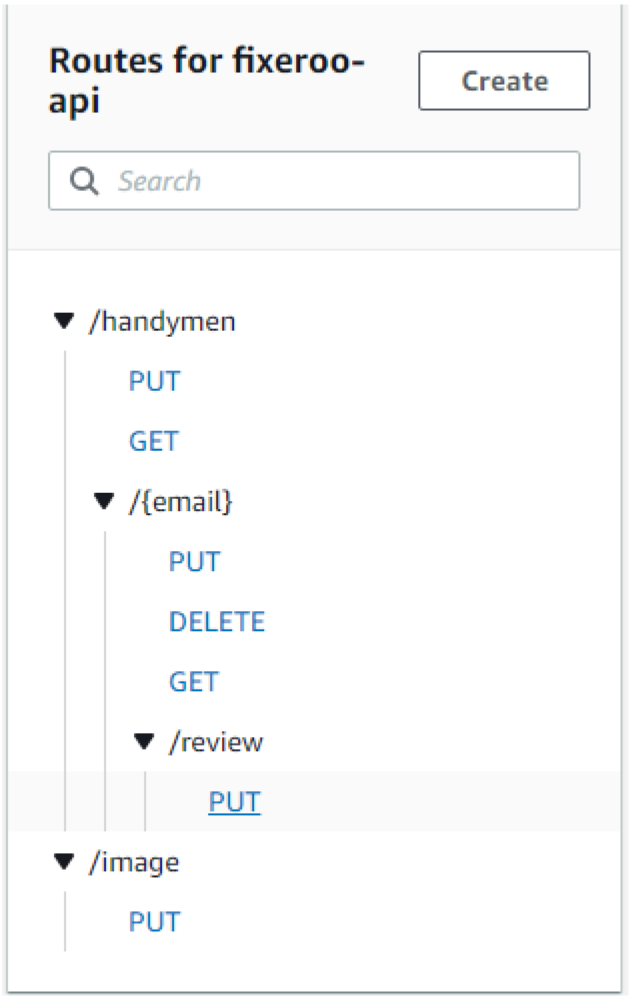
CREATE AN HTTP API

Navigate to API gateway console at https://console.aws.amazon.com/apigateway

Select create API, HTTP API , and then 'build'

Enter 'fixeroo-api as API name

Create the following routes:



```
/handymen
    PUT
    GET
    /{email}
        PUT
        DELETE
        GET
        /review
            PUT
/image
    PUT
```

Now, create an integration with the lambda function

Choose API. Choose Integrations.

Choose Manage integrations and then choose Create.

For Integration type, choose Lambda function.

For Lambda function, enter get-handymen

Choose Create.

Now attach the integration to the routes:

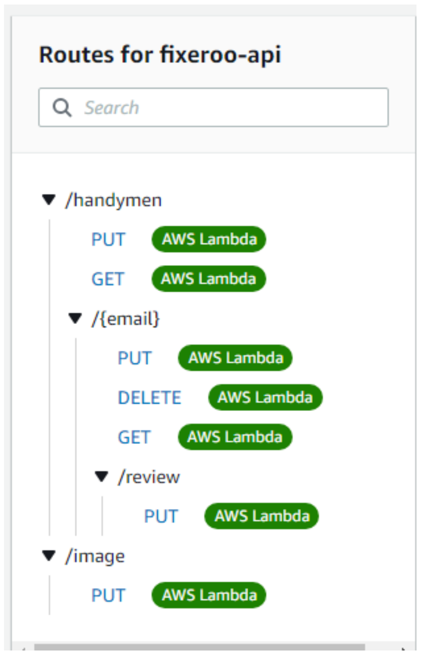
Choose your API. Choose Integrations. Choose a route.

Under Choose an existing integration, choose get-handymen

Choose Attach integration.

Repeat for all the routes.

The routes should show that they are attached to Lambda Integration



Routes for fixeroo-api

```
▼ /handymen
    PUT  AWS Lambda
    GET  AWS Lambda
  ▼ /{email}
      PUT     AWS Lambda
      DELETE  AWS Lambda
      GET     AWS Lambda
    ▼ /review
        PUT  AWS Lambda
▼ /image
    PUT  AWS Lambda
```

FIXEROO FRONTEND

Now that the API is set up, along with the Lambda functions that interact with DynamoDB and S3, the backend of the site is ready to be interacted with via a client!

Download the zip file called 'fixeroo-frontend' and extract
Type in `'npm install'` to install all dependencies
To run the code on localhost type `'npm run dev'`
To create a dist folder to upload the code to an S3 bucket type `'npm run prod'`