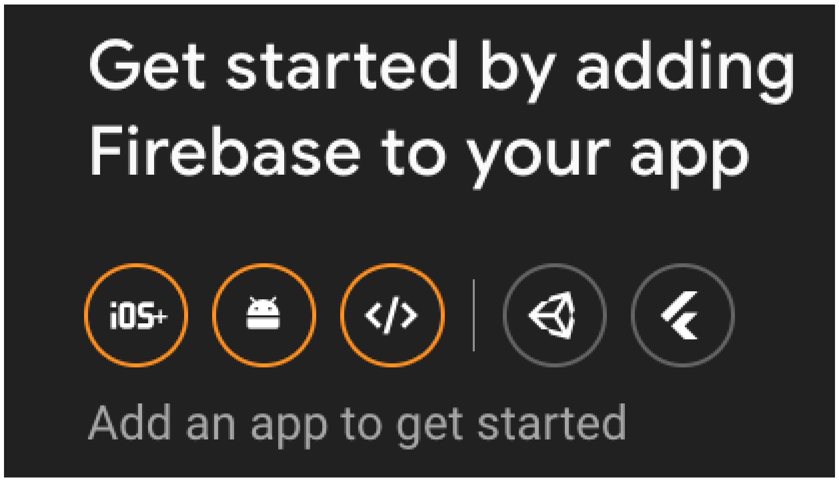
How to get firebase credentials

1.Visit https://firebase.google.com/

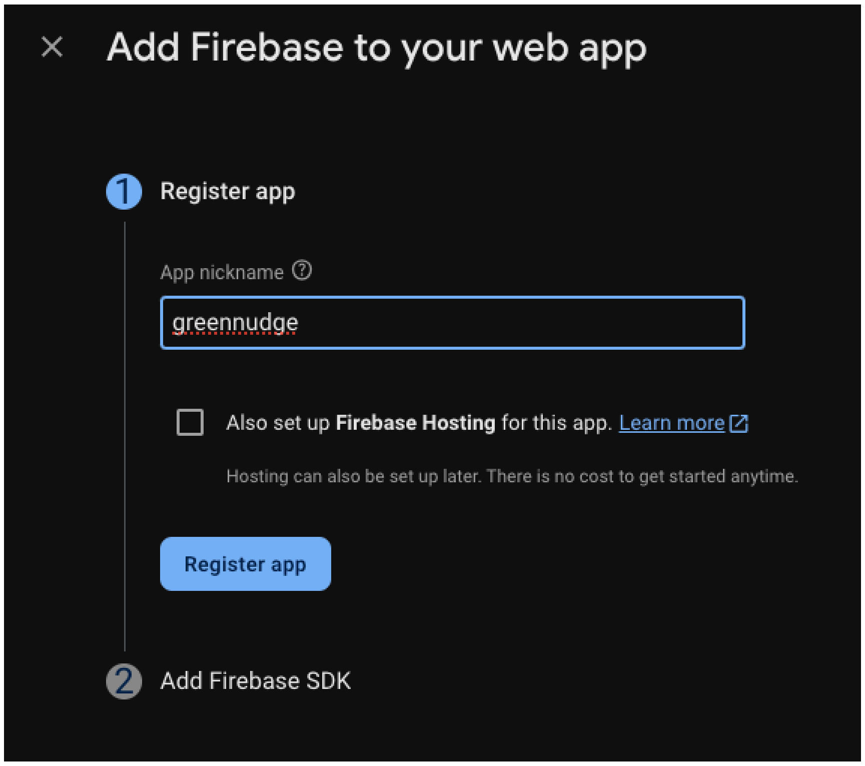
2. Press “get started”, press “create a project”, enter something like “greennudge”, press “continue”

3. Press “continue” (it is ok to turn off Google Analytics”)

4. Wait until the project is ready

5. pick “web” , it is the 3rd option 

Get started by adding Firebase to your app
iOS+
</>
Add an app to get started

6. enter the name of the web app  , hit “register”

Add Firebase to your web app

1 Register app

App nickname

greennudge

Also set up **Firebase Hosting** for this app. Learn more

Hosting can also be set up later. There is no cost to get started anytime.

Register app

2 Add Firebase SDK

7. You will get Firebase SDK. It looks like this. I will need those information:

```
// Import the functions you need from the SDKs you need

import { initializeApp } from "firebase/app";

// TODO: Add SDKs for Firebase products that you want to use

// https://firebase.google.com/docs/web/setup#available-libraries


// Your web app's Firebase configuration

const firebaseConfig = {

  apiKey: "AIzaSyAqm0ehFQxFnpoA511EiHjup-YMFmr4Dmw",

  authDomain: "greennudge-663c8.firebaseapp.com",

  projectId: "greennudge-663c8",

  storageBucket: "greennudge-663c8.firebasestorage.app",

  messagingSenderId: "532445301474",

  appId: "1:532984301474:web:7a5b86bf0cc5b40556f0fb"

};


// Initialize Firebase

const app = initializeApp(firebaseConfig);
```

Use Google Firebase and DynamoDB

Yes, you can definitely store multiple file locations for a single user by using an array of S3 locations in DynamoDB. Here's how you can modify your code to achieve this:

Step 1: Modify DynamoDB Table Structure

Ensure your DynamoDB table (`UserFiles`) can store an array of file locations. You don't need to change the table schema, but you'll be updating the items to include an array.

Step 2: Update `Upload.js` to Handle Multiple File Uploads

Modify your `Upload.js` to append new file locations to the existing array in DynamoDB:

```
import React, { useState } from 'react';
import './Upload.css';
import * as XLSX from 'xlsx';
import { Light as SyntaxHighlighter } from 'react-syntax-highlighter';
import json from 'react-syntax-highlighter/dist/esm/languages/hljs/json';
import { tomorrowNight } from 'react-syntax-highlighter/dist/esm/styles/hljs';
import AWS from 'aws-sdk';

// Register the JSON syntax
SyntaxHighlighter.registerLanguage('json', json);

AWS.config.update({
  accessKeyId: process.env.REACT_APP_AWS_ACCESS_KEY_ID,
  secretAccessKey: process.env.REACT_APP_AWS_SECRET_ACCESS_KEY,
  region: process.env.REACT_APP_AWS_REGION,
});

const s3 = new AWS.S3();
const dynamoDB = new AWS.DynamoDB.DocumentClient();

const Upload = ({ userUid }) => {
  const [fileData, setFileData] = useState(null);
  const [fileName, setFileName] = useState('');
  const [fileSize, setFileSize] = useState(0);
  const [jsonSize, setJsonSize] = useState(0);
  const [dragOver, setDragOver] = useState(false);
  const [error, setError] = useState('');

  const handleFileUpload = (file) => {
    setError(''); // Reset error message
    const validExtensions = ['xlsx', 'xls', 'numbers'];
    const fileExtension = file.name.split('.').pop().toLowerCase();
    if (!validExtensions.includes(fileExtension)) {
      setError('Unsupported file type. Please upload a .xlsx/.xls/.numbers file.');
      return;
    }

    setFileName(file.name); // Set the file name
    setFileSize(file.size); // Set the file size
    const reader = new FileReader();

    reader.onload = (e) => {
      const data = new Uint8Array(e.target.result);
      const workbook = XLSX.read(data, { type: 'array' });

      const sheetsData = {};
      workbook.SheetNames.forEach(sheetName => {
        const worksheet = workbook.Sheets[sheetName];
        sheetsData[sheetName] = XLSX.utils.sheet_to_json(worksheet);
      });

      setFileData(sheetsData);
      setJsonSize(new Blob([JSON.stringify(sheetsData)]).size); // Set the JSON size

      // Upload JSON to S3
      const jsonBlob = new Blob([JSON.stringify(sheetsData)], { type: 'application/json'
});
      const s3Params = {
        Bucket: 'my-app-uploads', // Replace with your S3 bucket name
        Key: `${userUid}/${file.name}.json`,
        Body: jsonBlob,
        ContentType: 'application/json',
      };

      s3.upload(s3Params, async (err, data) => {
        if (err) {
          setError('Error uploading file to S3');
          console.error(err);
          return;
        }

        // Retrieve existing file locations from DynamoDB
        const getParams = {
          TableName: 'UserFiles',
          Key: { uid: userUid },
        };

        try {
          const result = await dynamoDB.get(getParams).promise();
          const existingFiles = result.Item ? result.Item.fileLocations : [];

          // Append new file location to the array
          const updatedFiles = [...existingFiles, data.Location];

          // Store updated file locations in DynamoDB
          const updateParams = {
            TableName: 'UserFiles',
            Key: { uid: userUid },
            UpdateExpression: 'set fileLocations = :fileLocations',
            ExpressionAttributeValues: {
              ':fileLocations': updatedFiles,
            },
          };
          await dynamoDB.update(updateParams).promise();
          console.log('File metadata stored in DynamoDB');
        } catch (dynamoErr) {
          setError('Error storing file metadata in DynamoDB');
          console.error(dynamoErr);
        }
      });
    };

    reader.readAsArrayBuffer(file);
  };

  const handleDragOver = (event) => {
    event.preventDefault();
    setDragOver(true);
  };

  const handleDragLeave = () => {
    setDragOver(false);
  };

  const handleDrop = (event) => {
    event.preventDefault();
    setDragOver(false);
    const file = event.dataTransfer.files[0];
    if (file) handleFileUpload(file);
  };

  const handleInputChange = (event) => {
    const file = event.target.files[0];
    if (file) handleFileUpload(file);
  };

  return (
    <div
      className={`upload-container ${dragOver ? 'drag-over' : ''}`}
      onDragOver={handleDragOver}
      onDragLeave={handleDragLeave}
      onDrop={handleDrop}
    >
      <h2>Upload Your Emissions Data</h2>
      <p>We will evaluate the data and get back to you with your customized scenario
modelling.</p>
      {error && <p className="error">{error}</p>}
      <button onClick={() => document.getElementById('fileInput').click()}>
        {fileName ? (
          `Uploaded File: ${fileName} (${(fileSize / 1024).toFixed(2)} KB)`
        ) : (
          <>
```

```
            <span>+</span>
            Click or Drag a file here to upload
          </>
        )}
      </button>
      <input
        id="fileInput"
        type="file"
        onChange={handleInputChange}
        style={{ display: 'none' }}
      />
      {fileData && (
        <>
          <hr />
          <h3>Thank you! We'll evaluate your data shortly. You can preview the parsed data
below.</h3>
          <p>JSON Size: {(jsonSize / 1024).toFixed(2)} KB</p>
          <p>Space Saved: {((fileSize - jsonSize) / 1024).toFixed(2)} KB ({((1 - jsonSize /
fileSize) * 100).toFixed(2)}%)</p>
          <SyntaxHighlighter language="json" style={tomorrowNight}>
            {JSON.stringify(fileData, null, 2)}
          </SyntaxHighlighter>
        </>
      )}
    </div>
  );
};

export default Upload;
```

Explanation:

1. Retrieve Existing File Locations: Before updating DynamoDB, retrieve the existing file locations for the user.
2. Append New File Location: Append the new file location to the existing array.
3. Update DynamoDB: Update the DynamoDB item with the new array of file locations.

This way, each user can have multiple file locations stored in DynamoDB, allowing them to upload multiple files. Let me know if you need any further assistance!