# Uploading Files to GitHub

**Step-by-Step Guide**

1. **Prepare Files:**

* Create a new folder on your computer.
* Place all the files you want to upload into this folder.

1. **Open Visual Studio Code:**

* Launch Visual Studio Code.
* Use the "Open Folder" option to open the folder you created in step 1.

1. **Initialize Git:**

* Open the integrated terminal in Visual Studio Code.
* Run the following command to initialize Git within the project folder:

****

1. **Add Files to Git:**

* In the terminal, navigate to the project folder if not already there.
* Run the following command to add all files to the staging area:

A black screen with white letters

Description automatically generated

1. **Create a New Branch:**

* Decide on a meaningful name for your new branch.
* Run the following command to create and switch to the new branch:

A black background with white text

Description automatically generated

1. **Commit Changes:**

In the terminal, use the following command to commit the staged changes with a descriptive message:



1. **Add Remote Repository:**

* On GitHub, create a new repository if you haven't already.
* Copy the remote repository URL.

A black background with white text

Description automatically generated

* In the terminal, run the following command to add the remote repository URL:

A black background with white text

Description automatically generated

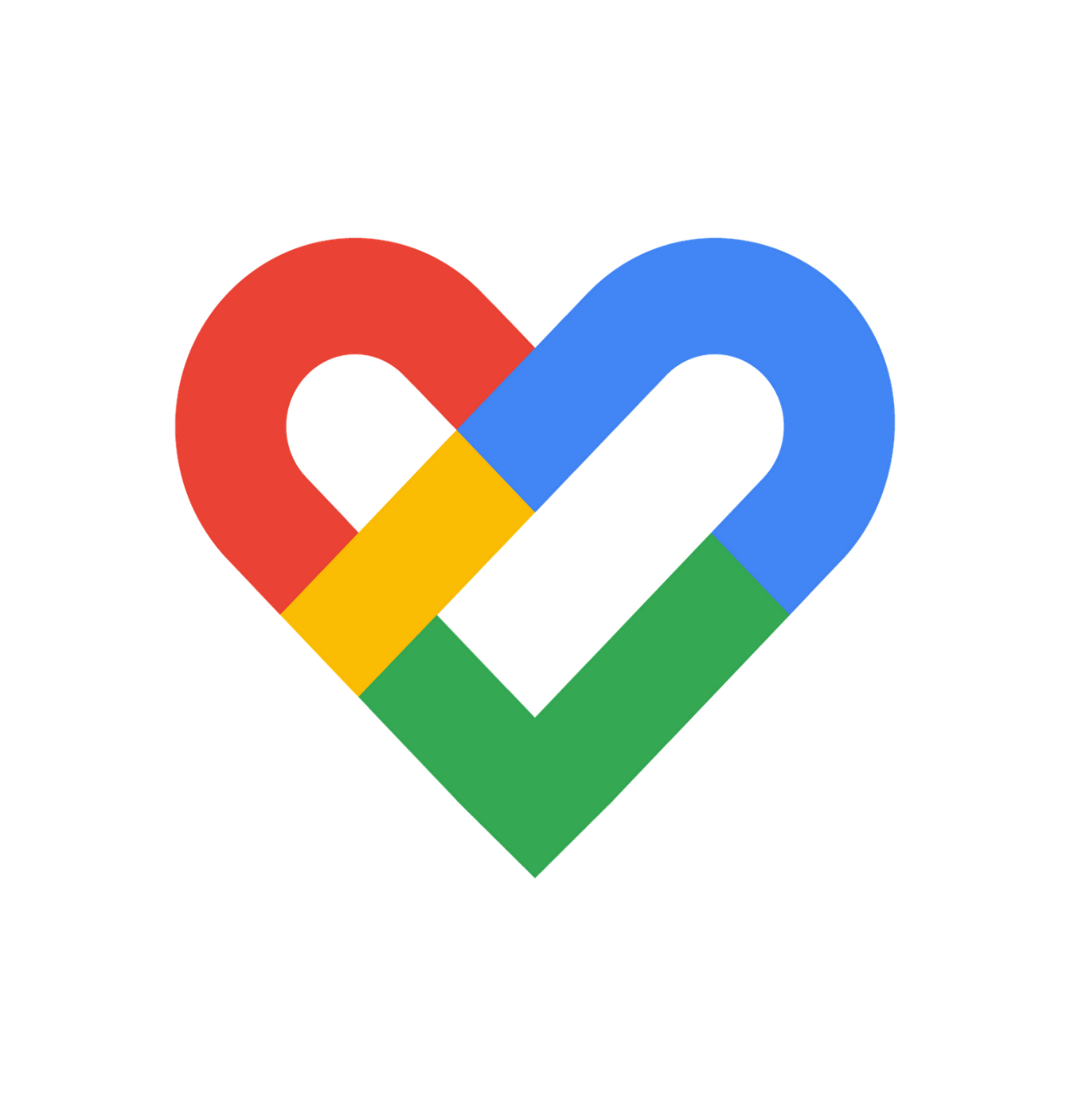
1. **Push Changes to GitHub:**

Use the following command to push your committed changes to GitHub:

A black background with white text

Description automatically generated

**DEVELOPER MANUAL**



**Google Fit Authorization**

**Pavan Kumar Etta Chaya Y K**

pavankumar@rhenix.org[chaya@rhenix.org](mailto:chaya@rhenix.org)

**Table of Contents**

Contents

[Uploading Files to GitHub 1](#_Toc148412186)

[1. Introduction 5](#_Toc148412187)

[2. Authorizing access to Google Fit 5](#_Toc148412188)

[2.1: Create a Project in the Google Cloud Console 5](#_Toc148412189)

[2.2: Enable the Google Fit API 7](#_Toc148412190)

[2.3: Create OAuth 2.0 Client Credentials 9](#_Toc148412191)

[3. Integration of OAuth 2.0 with Web Application 14](#_Toc148412192)

[3.1 Creating a Registration Page with HTML and CSS 14](#_Toc148412193)

[3.2 Implementing Serverless Backend 15](#_Toc148412194)

[3.3: Configuring Amazon API Gateway: 20](#_Toc148412195)

# Introduction

**Google Fit** is a powerful platform developed by Google that allows users to track and manage their fitness and wellness data. With Google Fit, users can gather data from various fitness trackers, smartwatches, and health-related apps, enabling them to monitor their activity levels, heart rate, sleep patterns, and more. To access and interact with user fitness data through the Google Fit API, proper authorization is essential. The authorization process ensures that only authorized applications can access a user's sensitive health information, maintaining the privacy and security of their data. By integrating with Google Fit, you can enhance your applications with the wealth of fitness data it provides. This integration opens opportunities to create personalized experiences, deliver targeted fitness recommendations, and help users achieve their health and wellness goals more effectively.

In this documentation, we will walk you through the step-by-step process of authorizing access to Google Fit. You will learn how to create a project in the Google Cloud Console, enable the Google Fit API, generate OAuth 2.0 client credentials, implement the authorization flow in your application, exchange authorization codes for access tokens, and make authorized API requests to retrieve and manage user fitness data. Following this documentation will empower you to seamlessly integrate Google Fit into your applications and leverage its rich set of fitness data and features. It will enable you to provide users with a personalized and data-driven fitness experience while adhering to the highest standards of privacy and security.

# Authorizing access to Google Fit

# 2.1: Create a Project in the Google Cloud Console

Creating a project in the Google Cloud Console is the initial step in the process of authorizing access to Google Fit.

* **Open the Google Cloud Console:**

Launch your preferred web browser and go to the **Google Cloud Console website** at <https://console.cloud.google.com/>.

* **Sign into your Google account:**

If you are not already signed in, enter your **Google account** credentials (email and password) to log in to the Google Cloud Console.

* **Create a new project:**

1. At the top of the Cloud Console page, you will see a **project drop-down menu**. Click on it to open the menu.
2. Click on "**New Project**".

**A screenshot of a computer

Description automatically generated**

**Fig1: Create a new project.**

**A screenshot of a computer

Description automatically generated**

**Fig 2: Create a new project.**

* **Enter project details:**

1. In the "**Project name**" field, provide a name for your project. Choose a descriptive name that represents your application or project.
2. If you want to associate the project with an **organization**, select the desired organization from the drop-down menu.
3. Leave the "**Location**" field as the default value.
4. Click the "**Create**"button to create the project.

A screenshot of a computer

Description automatically generated

**Fig 3: Enter Project Details**

* **Wait for project creation to complete:**

The Cloud Console will initiate the **project creation** process, which may take a few moments to complete. Once the project creation is finished, you will be redirected to the project dashboard.

A screenshot of a computer

Description automatically generated

**Fig 4: Project creation**

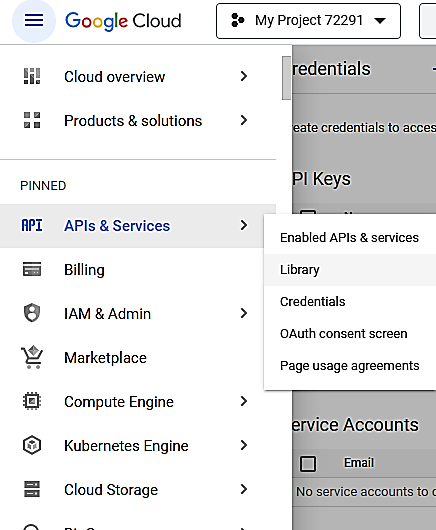
Creating a project in the Google Cloud Console establishes a dedicated workspace for managing your application's resources and configurations. It provides you with a unique project ID and allows you to set up billing, enable APIs, and manage permissions for your project.

# 2.2: Enable the Google Fit API

Enabling the **Google Fit API** is a crucial step in the process of authorizing access to Google Fit. By enabling the API, you allow your application to interact with and make use of the Google Fit functionality.

* **Access the Library section:**

1. In the left sidebar, click on "**APIs & Services**".
2. From the dropdown menu, select "**Library**".

****

**2**

**1**

**Fig 5: Access the Library section.**

* **Search for Google Fit API:**

In the Library section, you'll find a search bar. Enter "Google Fit API" in the **search** bar to locate the API**.**

**A screenshot of a computer

Description automatically generated**

**Fig 6: Search for Google Fit API**

* **Enable the API:**

Once you've found the Google Fit API, click on it. You'll be taken to the API details page. On this page, click the "**Enable**" button to enable the API for your project.

**A screenshot of a computer

Description automatically generated**

**Fig 7: Enable the API**

Enabling the Google Fit API ensures that your project has the necessary permissions to access and use Google Fit's features and data. This step is essential for your application to retrieve, store, and manage user fitness information through the Google Fit API.

# 2.3: Create OAuth 2.0 Client Credentials

Before proceeding with the creation of OAuth 2.0 client credentials, you need to configure the **OAuth Consent Screen**. The OAuth Consent Screen is a user-facing interface that displays information about your application and requests permission from users to access their Google Fit data.

**Access the OAuth Consent Screen:**

* In the left sidebar, click on "APIs & Services" and then select "OAuth consent screen”. You will be taken to the OAuth Consent Screen configuration page.

**A screenshot of a computer

Description automatically generated**

**2**

**1**

**Fig 8: Access the OAuth Consent Screen**

* On the OAuth Consent Screen configuration page, select the **appropriate user type** based on your application's intended users. The user type can be "External" if you are developing an application for users outside your organization, or "Internal" for users within your organization.

**A screenshot of a chat

Description automatically generated**

**Fig 9: Select the appropriate user type**

* Enter **application information**:

1. Fill in the required information on the OAuth Consent Screen configuration page.
2. Enter the "Application name", which is the name of your application as it will appear to users.
3. Optionally, you can upload an "Application logo" that will be displayed alongside the application name.
4. Provide a "Support email" address for users to reach out to you for support related to your application.

A screenshot of a question

Description automatically generated A screenshot of a computer

Description automatically generated **Fig 10: Enter Application Information**

* **Configure scopes** and additional information:

In the "Scopes" section, add the necessary scopes for accessing Google Fit data. These scopes define the level of access your application requires.

A screenshot of a computer

Description automatically generated **Fig 11: Configure Scopes**

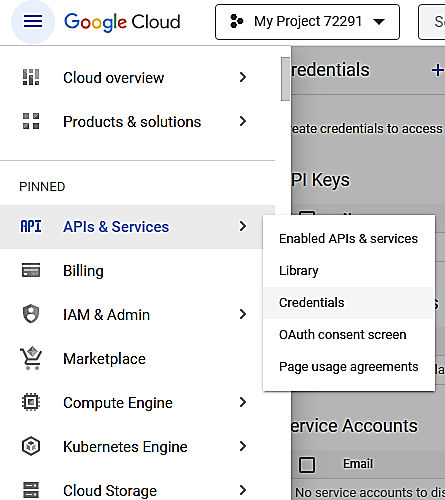
* Provide a "**Test user**" email address if you want to restrict access during the testing phase. Fill in any other additional information that might be required based on your application's requirements.
* Once you have provided all the necessary information, click the "**Save**" button to save your OAuth Consent Screen configuration. If you have filled in all the required fields and met the necessary criteria, you can submit your application for verification by clicking the "Submit for verification" button. After configuring the OAuth Consent Screen, you can proceed to create OAuth 2.0 client credentials.

**Create OAuth 2.0 client credentials:**

Creating **OAuth 2.0 client credentials** is an important step in the Google Fit authorization process. These credentials are required for your application to authenticate and securely communicate with the Google Fit API.

* **Access the Credentials section:**

In the left sidebar, click on "**APIs & Services**" and then select "**Credentials**”. You will be taken to the Credentials page where you can manage and create credentials for your project.



**2**

**1**

**Fig 12: Access the Credentials section.**

* **Create new OAuth client ID:**

On the Credentials page, click the "Create Credentials" button. From the drop-down menu, select "**OAuth client ID**". This will prompt you to configure the OAuth client ID settings for your application.

**1**

A screenshot of a computer

Description automatically generated

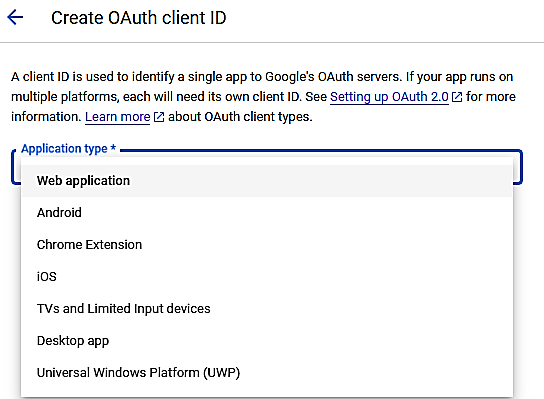
**2**

**Fig 13: Create new OAuth client ID.**

* **Choose application type:**

Select the appropriate application type based on the platform and requirements of your application.

For example, if you are developing a web application, choose "Web application". Other options include "Android" and "iOS" for mobile applications.



**Fig 14: Choose application type.**

* **Enter client details and authorized origins:**

1. Enter a name for your OAuth 2.0 client in the "Name" field. Choose a meaningful name that represents your application.
2. In the "Authorized JavaScript origins" field, specify the origins from which your application will make requests to the Google Fit API. These origins are typically the domains or URLs of your application.
3. In the "Authorized redirect URIs" field, enter the redirect URIs to which users will be redirected after granting authorization. These URIs must match the callback URLs defined in your application.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Fig 15: Enter client details and authorized origins.**

* **Generate client ID and client secret:**

Once you have entered the necessary details, click the "Create" button. Google Cloud Console will generate a **client ID and client secret** for your OAuth 2.0 client**. Make note of these credentials** as you will need them in the subsequent steps of the authorization process.

Note: Please do not share client IDs and secret IDs publicly.

A screenshot of a web page

Description automatically generated

**Fig 16: Generate client ID and client secret.**

# Integration of OAuth 2.0 with Web Application

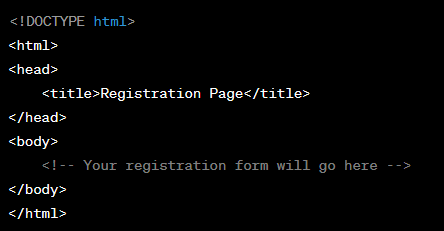
# 3.1 Creating a Registration Page with HTML and CSS

1. **Set Up Your Development Environment**

Before you begin, you'll need a **text editor** (e.g., Visual Studio Code, Sublime Text, Notepad++) and a web browser for testing. Make sure you have these tools ready.

1. **Create an HTML Document**

Open your text editor and **create a new HTML document**. You can start with the basic structure of an HTML page by using the following template:

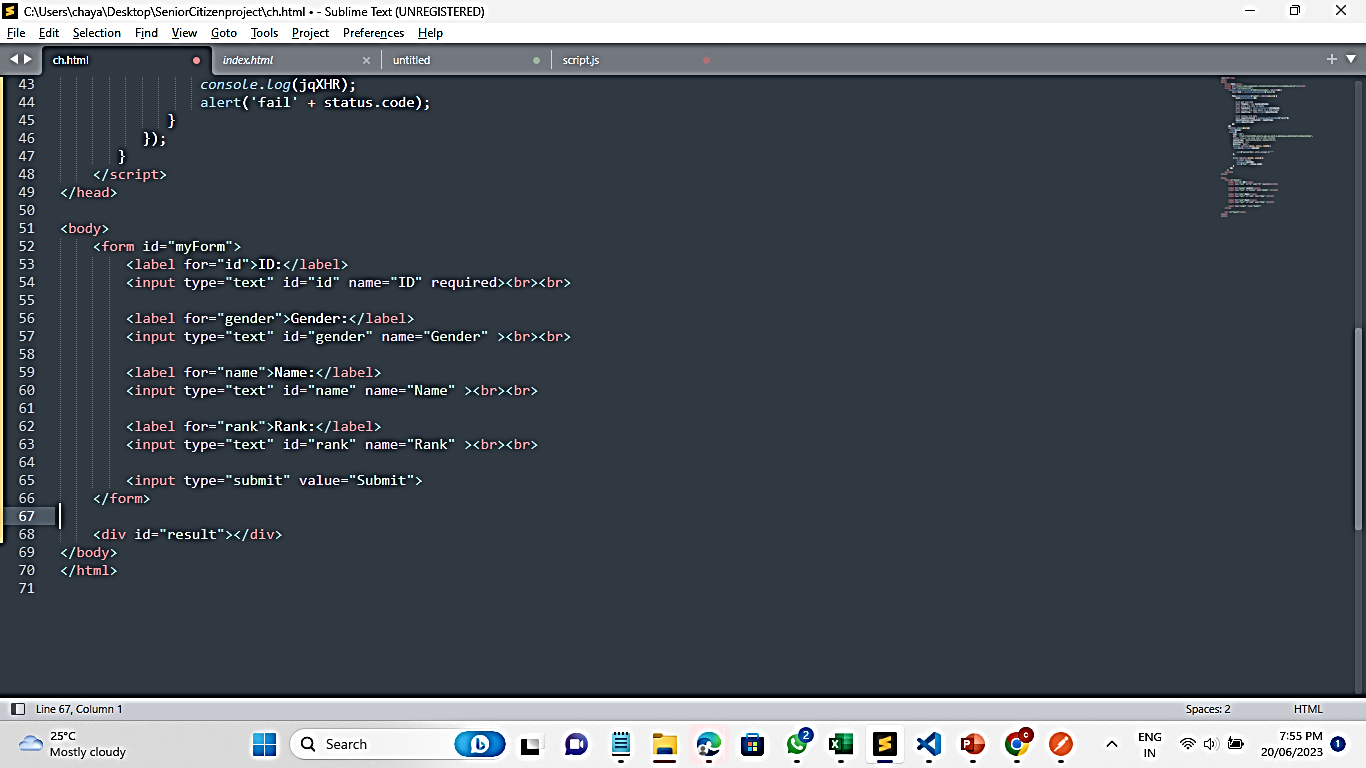


**Fig 17: Basic Structure of HTML.**

This sets up the basic structure of your HTML page with a title and an empty body section.

1. **Build the Registration Form**

Inside the <body> section, you can **create the registration form**. Here's an example of a simple registration form with common input fields:



**Fig 18: Simple Registration form**

This form includes fields for ID, Gender, Name, Rank. Make sure to customize it according to your requirements**.**

1. **Save and Test**

**Save your HTML document** with an appropriate name, such as "index.html." You can open it in a web browser to test how it looks and functions.

1. **Styling (Optional)**

To make your registration page visually appealing, you can **apply CSS styles**. You can either include an external CSS file or use inline styles directly within your HTML document.

1. **Form Submission**

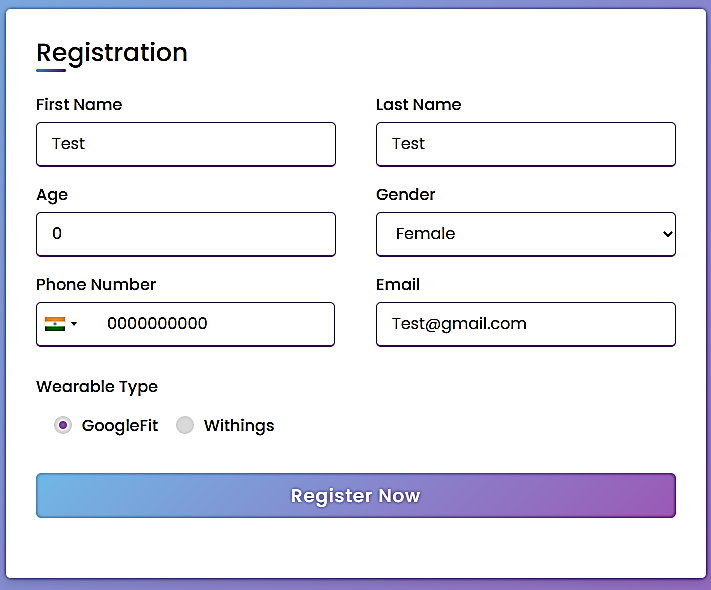
Once the registration form is completed, users can fill it out and **click the "Submit" button**. To process this data on the server, you would typically need a server-side script (e.g., PHP, Node.js, Python) to handle the form submission and store the data in your chosen database**.**

Upon clicking the 'Submit' button, the code should include a **Google Fit redirect URL**. This URL triggers a redirection to a Google account page, where the user will be prompted to select or create a Gmail account to proceed further and to allow access to the scope provided in the redirect URL.

"https://accounts.google.com/o/oauth2/v2/auth/oauthchooseaccount?redirect\_uri=https://rhenixhub.github.io/googleFit/thankyou.html&prompt=consent&response\_type=code&client\_id=\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*&scope=https://www.googleapis.com/auth/fitness.activity.read+https://www.googleapis.com/auth/fitness.blood\_pressure.read”

That's it! You've created a basic registration page using HTML. You can further enhance and customize it based on your project's requirements and design preferences. Top of Form

This is the registration page for our project.



**Fig 19: Registration Page**

# 3.2 Implementing Serverless Backend

To save user registration data from your registration page into DynamoDB using AWS Lambda functions and API Gateway, you can follow these steps:

**1. create DynamoDB Table**

Amazon DynamoDB is a cloud-based database service provided by Amazon Web Services (AWS) that stores and manages your data in a highly scalable and reliable way. Imagine it as a big, fast, and organized digital filing cabinet for your data.

1. Navigate to the **DynamoDB service**.

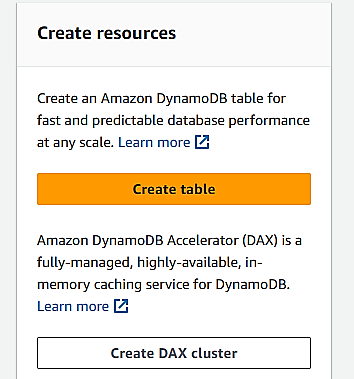
A screenshot of a computer

Description automatically generated

**1**

**Fig 20: DynamoDB service**

1. Click "**Create table**."



**2**

**Fig 21: DynamoDB service**

1. Define the **table name, primary key**, and other optional settings like provisioned capacity or on-demand capacity mode.
2. Configure other attributes based on the user registration data you want to store (e.g., first name, last name, age, gender, phone, email, and wearable type).

A screenshot of a computer

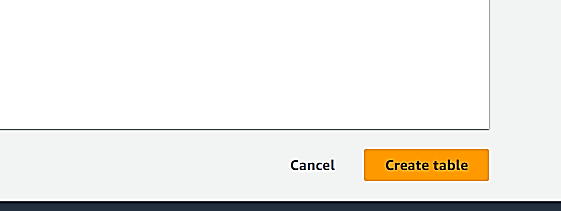
Description automatically generated

**4**

**3**

**Fig 22: Give table name and primary Key.**

1. Review and **create the table**.



**5**

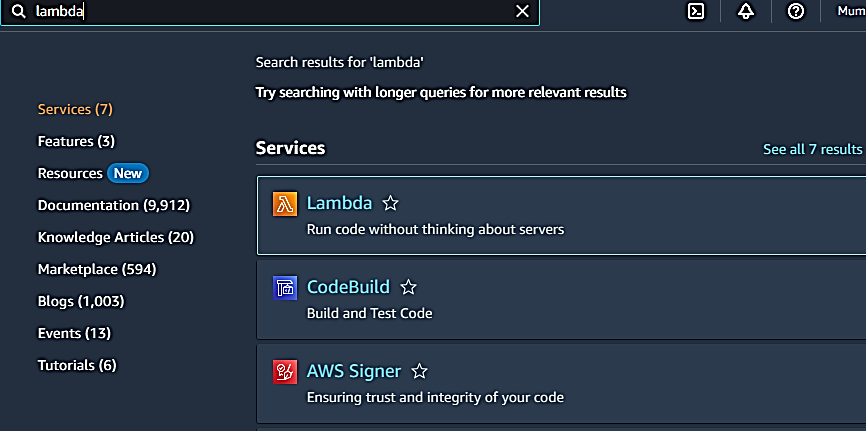
**Fig 23: Click on Create table.**

**2. Create Lambda Functions:**

AWS Lambda is a serverless compute service offered by Amazon Web Services (AWS) that allows you to run code without provisioning or managing servers. Lambda is designed to help developers build applications in a scalable, cost-effective, and low maintenance way.

Steps to create a Lambda Function:

* Navigate to the **AWS Lambda service** in the AWS Management Console.



**1**

**Fig 24: Click on Lambda service.**

* Click "**Create function**."

**2**

A screenshot of a computer

Description automatically generated

**Fig 25: Click on Create function.**

* Choose the "**Author from scratch**" option.

A screen shot of a computer

Description automatically generated

**3**

**Fig 26: Choose Author from scratch.**

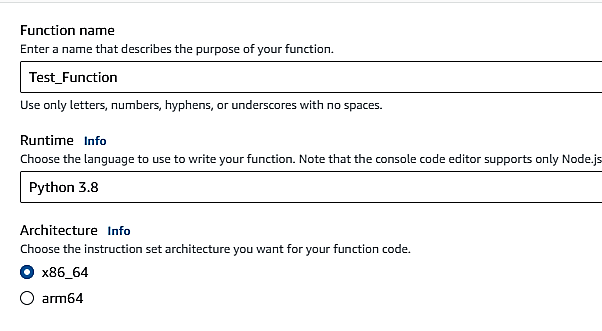
* Configure the function as follows:
  + **Function name**: Give your Lambda function a descriptive name.
  + Runtime: Choose the runtime compatible with your **programming language** (e.g., Node.js, Python, Java).

Note: For Project we used “Python3.8”

* + Execution role: Create a new **role** with basic Lambda permissions and DynamoDB access. This role allows the Lambda function to interact with DynamoDB.

Note: For Project we used “userrole”

* Click "**Create function**."

****

**5**

**4**

**Fig 28: Give Function name and Runtime.**

**A screenshot of a computer

Description automatically generated**

**7**

**6**

**Fig 29: Choose Execution role.**

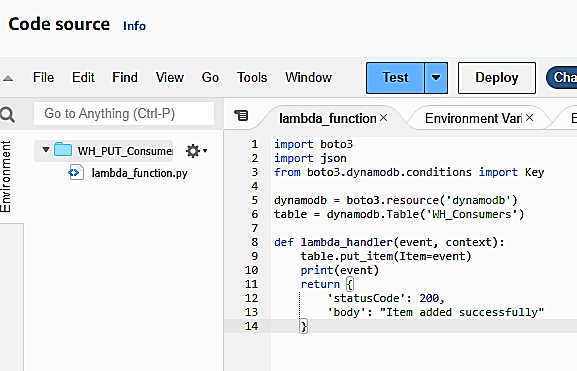
**A screenshot of a computer

Description automatically generated**

**8**

**Fig 30: Click on Create function.**

* In the function **code editor**, write the code for your Lambda function to perform the desired CRUD operation. For example, if it's the function for creating a document, write code to insert data into DynamoDB.
* **Deploy** your Lambda function.

****

**10**

**9**

**Fig 31: Write Lambda Function and Deploy.**

# 3.3: Configuring Amazon API Gateway:

* Navigate to the Amazon **API Gateway** service in the AWS Management Console.

A screenshot of a computer

Description automatically generated

**1**

**Fig 32: Click on API Gateway**

* Click "**Create API**."

A screenshot of a computer

Description automatically generated

**2**

**Fig 33: Click on Create API**

* Choose the "**REST API**" option.

A close-up of a computer screen

Description automatically generated

**3**

**Fig 35: Click on Build**

* Configure your API, including the **API name** and default stage name and Click "**Create API**."

A screenshot of a computer

Description automatically generated

**7**

**6**

**5**

**4**

**Fig 36: Configure API**

* Create **resources** and **methods** for each CRUD operation.

**Create Resources**

Resources represent the endpoints of your API. They are organized hierarchically to match the structure of your API.

* In the "**Actions**" menu, click "**Create Resource**”.
* Enter a **name** for the resource (e.g., "/users")."
* Enable API gateway CORS.
* Click "**Create Resource**."

A screenshot of a web page

Description automatically generated

**2**

**1**

**Fig 37: Creating Resource**

A screenshot of a computer

Description automatically generated

**4**

**5**

**3**

**Fig 38: Creating Resource**

**Create Methods**

Methods define how API Gateway handles incoming requests for a resource. Common methods are GET, POST, PUT, DELETE, etc.

* Select the resource you just created.
* Click "**Create Method**."
* Choose the **HTTP method** (e.g., GET, POST) you want to associate with the resource.

A screenshot of a computer

Description automatically generated

**1**

**Fig 39: Creating Methods**

**A screenshot of a method type

Description automatically generated**

**2**

**Fig 40: Choose Method Type**

* Configure each method to use Lambda integration. Choose the Lambda function created earlier as the integration target for each method.

**3**

**A screenshot of a computer

Description automatically generated**

**4**

**Fig 41: Choose Lambda function to integrate.**

* Deploy the API to create an accessible endpoint.

**A screenshot of a computer

Description automatically generated**

**1**

**Fig 42: Click on Deploy API.**

**A screenshot of a computer

Description automatically generated**

**3**

**2**

**Fig 43: Select stage and deploy.**

* After deployment, you can test your API using the provided Invoke URL

**A close-up of a computer screen

Description automatically generated**

**Fig 44: Test API using URL .**

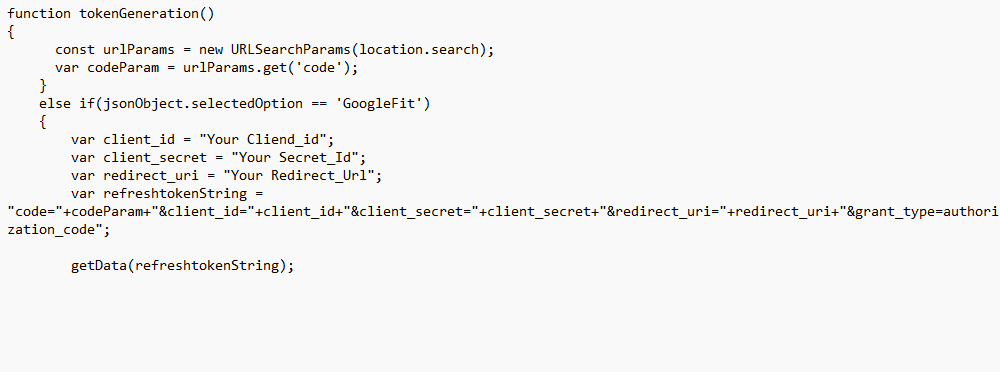
# 3.4: Thank You Page:

After the user clicks "Continue" on the Google account page, the redirection will lead them to a "**Thank You" page**. On this page, an OAuth token will be generated.

**Key parts of Code:**

**1. Token Generation Function:**

JavaScript function is called when the page loads. It handles the exchange of authorization codes for access tokens and refresh tokens. It also simulates user registration data and conditionally calls functions for Google Fit.



**2. Data Insertion Function (insert Data function):**



A blue rectangular box with black text

Description automatically generated

**Fig 20: Thank you Page**

# 3.5: Fetching Google Fit Data:

The script should design to fetch various health-related data from Google Fit, including step counts, calories expended, and heart rate data.

Before using the script, you need the following prerequisites:

* Python installed on your system.
* Necessary Python packages (requests, Json, csv, pandas).

You can install the required Python packages using the following command:

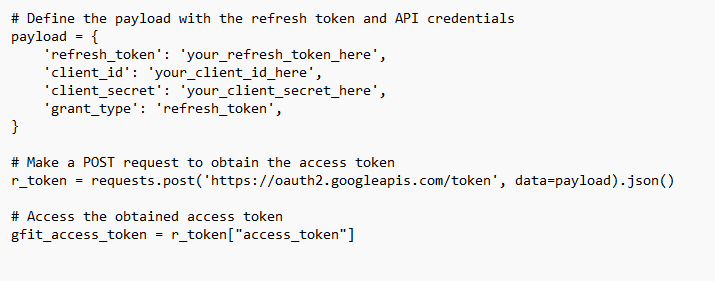
A black screen with white text

Description automatically generated

**Key parts of the code:**

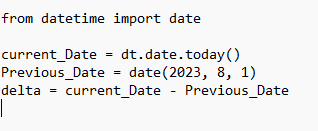
**Access Token Retrieval:**

* This section is responsible for obtaining the access token using the provided refresh token and API credentials.



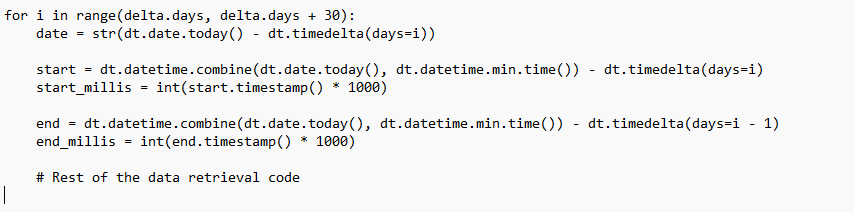
**Date Range Calculation:**

* This section calculates the date range for data retrieval, typically fetching data for the last 30 days.



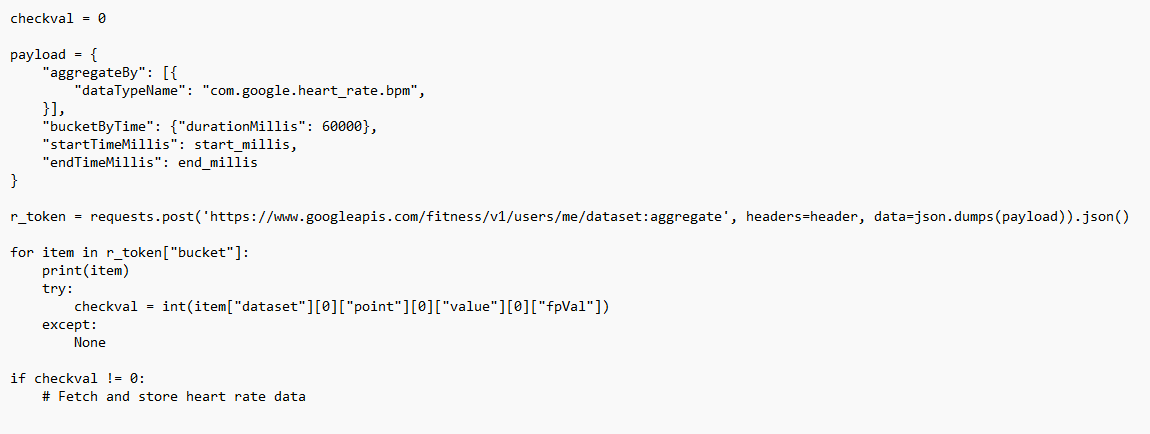
**Data Retrieval Loop:**

* The main loop iterates through the specified date range, allowing data retrieval for each day.



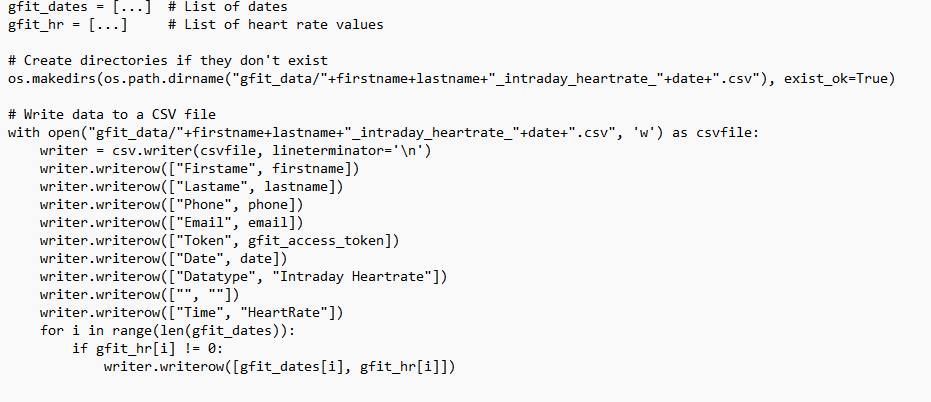
**Heart Rate Data Retrieval:**

* This section checks if heart rate data is available for the specified day and fetches it if available.



**Data Storage:**

* The script stores the retrieved data in CSV files.



**-----------------------------------------------------------END---------------------------------------------------------**