

Step Book

Crime Map Web App Flask

Episode 02 - Final

API Key Google . Add Map . Add Marker

What is “Step Book”?

Step Book is a book that contains the steps of how to build something. Does not contain a detailed description in it. This book is specially designed to focus on practical coding to make a real project.

How to use “Step Book”?

This is an important part of how I will show you how to use this book.

Since this book is only step by step, I recommend following it. You will find a point where you will ask "what does that mean blah-blah?". search in the Google search field. This step is much more practical and quicker to learn something. You continue to build a project, and when you get confused, you immediately find out with the help of Google or official documentation.

In this book, you will practice how to **improve your project building skills, critical thinking skills, and problem solving skills**. That's the point. If you're still confused about the steps, go back to Videos on our YouTube channel.

Source Code

If any time you need our source code, we have already provided it on our GitHub. Visit the following link:

<https://github.com/try-fullstack/pin-food-web-app-flask>

Support us at:

- **YouTube**
https://www.youtube.com/channel/UCLFXUhx6_Io7tznPtVdfiGQ
- **Patreon**
<https://www.patreon.com/TryFullStack>

Sorry, if our English is not perfect. We'll keep trying to make it easier to read.



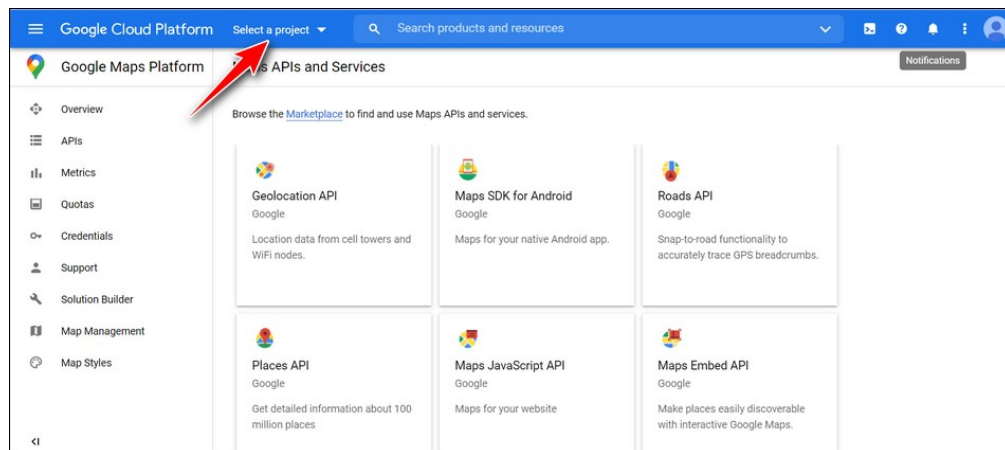
Table of Contents

API Key Google.....	1
Adding Map to Template.....	5
Adding Marker.....	7
Adding Form.....	8
Adding bootstrap style.....	9
Storing food stall data.....	11
Displaying existing food stall.....	13

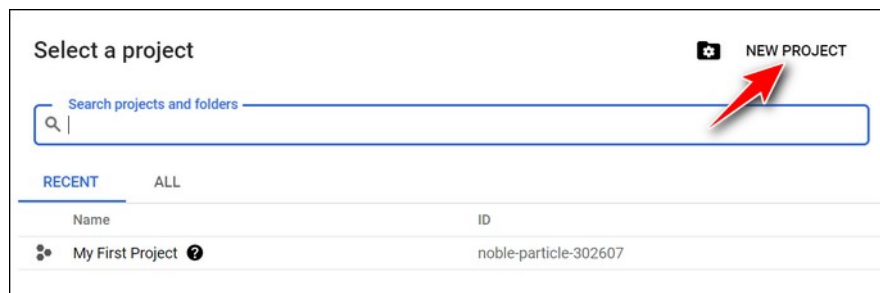
API Key Google

* **Note:** if you already have an API Key ready to use, you can skip this step

Open Google Cloud Platform and select "**Select a project**":




Then, the "**Select a Project**" dialog box appears. At the top, click **NEW PROJECT**:




In the **New Project** page, fill in the **Project name** with the desired project name (in the Location section, leave the value **No organization**) then click the **CREATE** button:


New Project

 You have 10 projects remaining in your quota. Request an increase or delete projects. [Learn more](#)


[MANAGE QUOTAS](#)

Project name * 
try-fullstack-project ?

Project ID: try-fullstack-project. It cannot be changed later. [EDIT](#)

Location *
 No organization [BROWSE](#)

Parent organization or folder

[CREATE](#) [CANCEL](#) 


Next, we will be directed to the dashboard page. Select the project that we created:

Select a project  [NEW PROJECT](#)

[RECENT](#) [ALL](#)

	Name	ID
	try-fullstack-project ? 	try-fullstack-project
	My First Project ?	noble-particle-302607

Click the **Maps JavaScript API** on the dashboard page:

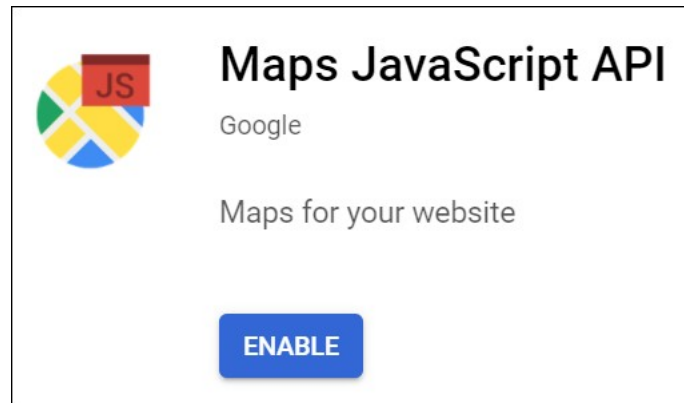


Maps JavaScript API

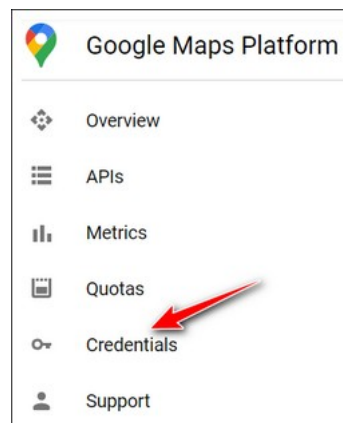
Google

Maps for your website

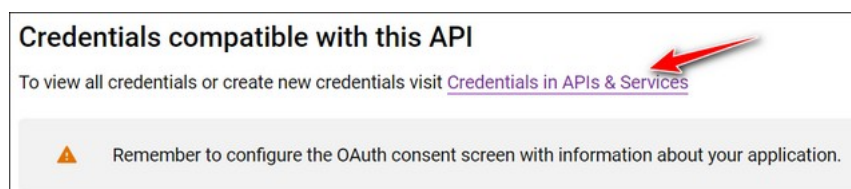
Click **ENABLE** button:



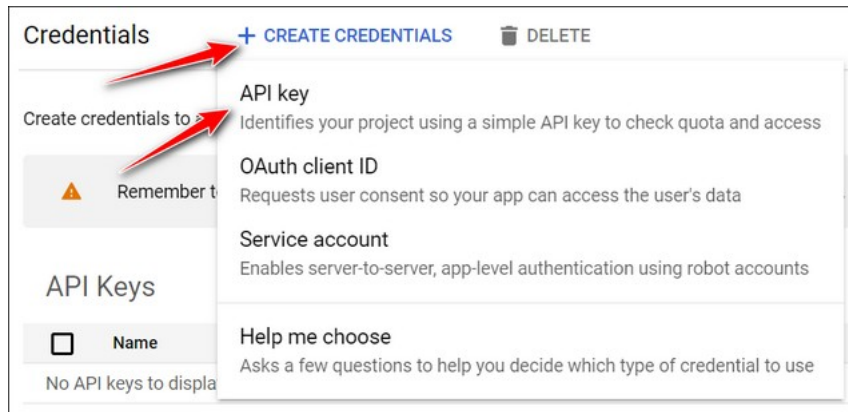
In the left sidebar menu, click **Credentials**:



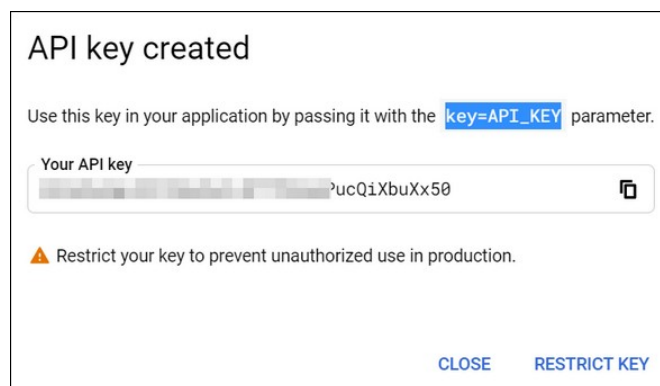
Click **Credentials in APIs & Services**:



Click **CREATE CREDENTIALS** → **API key**:



If the API key is generated successfully, copy the API key:



Adding Map to Template

Open the **templates/home.html** file and add the following code (Don't forget, on the yellow-red line, you can replace it with your API Key):

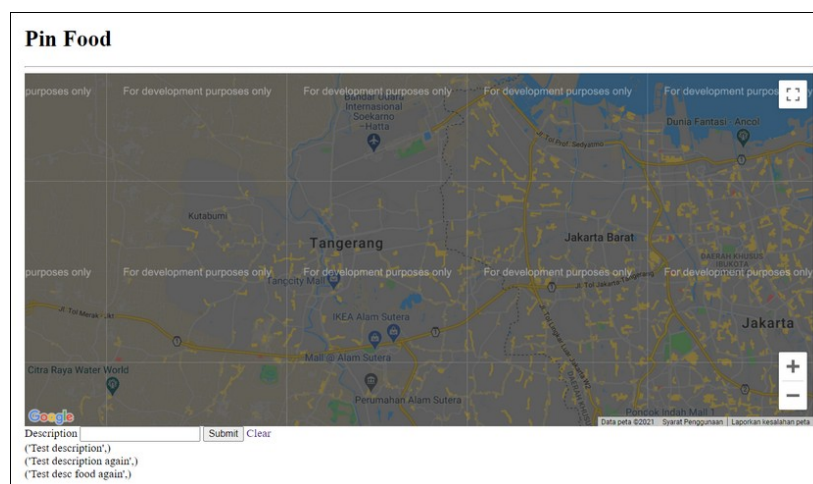
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Pin Food</title>
    <script
      type="text/javascript"
      src="https://maps.googleapis.com/maps/api/js?key=AIZA5yAgL-
hZt2myGy3_QPT7UGauPucQixbuXx50&callback=init">
    </script>
    <script type="text/javascript">
      function init() {
        var opt = {
          center: new google.maps.LatLng(
            -6.190059388922542,
            106.7599655019272
          ),
          zoom: 12
        };
        var map = new google.maps.Map(
          document.getElementById("maps"),
          opt
        );
      }
    </script>
  </head>
  <body onload="init()">
    <div class="container mt-4 mb-3">
      <div class="row">
        <div class="col-md-12">
          <h1 class="display-4">Pin Food</h1>
          <hr>
        </div>
      </div>
    </div>
    <div class="container mb-3">
      <div class="row">
        <div class="col-md-8">
          <div id="maps" style="width: 70%; height: 500px"></div>
        </div>
        <div class="col-md-4">
          <form action="/add" method="POST">
            <label>Description</label>
            <input type="text" class="form-control" name="description">
            <button class="btn btn-primary" type="submit">
              Submit
            </button>
            <a href="/clear" class="btn btn-outline-danger">
              Clear
            </a>
          </form>
        </div>
      </div>
    </div>
    <div class="mt-3">
      {% if data %}
```

```

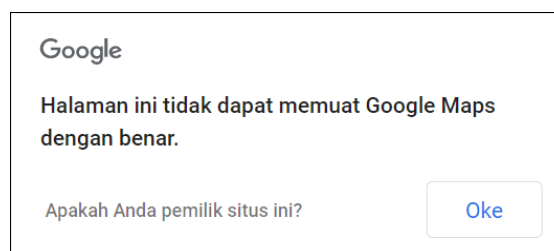
        {% for description in data %}
        <p>{{ description }}</p>
        {% endfor %}
    {% endif %}
</div>
</div>
</div>
</div>
</body>
</html>

```

Reload again **http://localhost:5000:**



If you find a message like:



Maybe you need to activate billing. Maybe for now you can ignore this message, but if you really want this message to go away, you'll need to enable Billing.

Adding Marker

Open the **templates/home.html** file and change the JavaScript code to:

```
var map;
var marker;

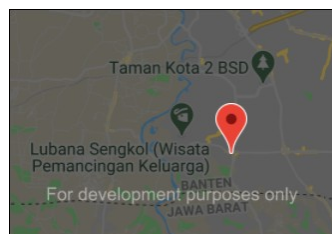
function init() {
  var opt = {
    center: new google.maps.LatLng(
      -6.190059388922542,
      106.7599655019272
    ),
    zoom: 12
  };

  map = new google.maps.Map(
    document.getElementById("maps"),
    opt
  );

  google.maps.event.addListener(map, 'click',
    function(event) {
      setMarker(event.latLng);
    }
  );
}

function setMarker(location) {
  if (marker) {
    marker.setPosition(location);
  } else {
    marker = new google.maps.Marker({
      position: location,
      map: map
    });
  }
}
```

Reload **http://localhost:5000/** again, try clicking anywhere on the map. The result will appear a red marker icon like:



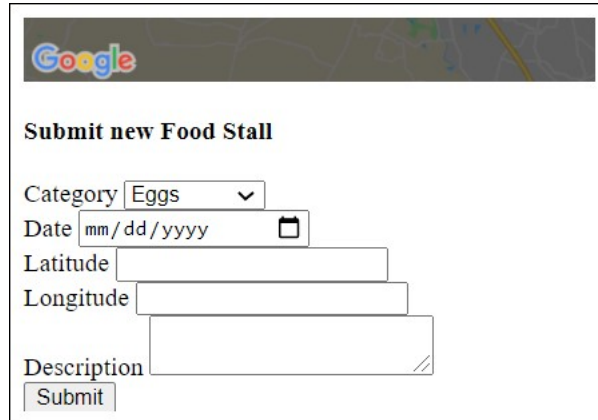
Adding Form

Still in the **templates/home.html** file and change the body to:

```
<body onload="init()">
  <div class="container mt-4 mb-3">
    ...
  </div>
  <div class="container mb-3">
    <div class="row">
      <div class="col-md-8">
        <div id="maps" style="width: 70%; height: 500px"></div>
      </div>
      <div class="col-md-4">
        <div class="card bg-light">
          <div class="card-body">
            <h4 class="card-title">Submit new Food Stall</h4>
            <form action="/savefood" method="post">
              <div class="form-group">
                <label for="category">Category</label>
                <select class="form-control"
                  name="category" id="category">
                  <option value="eggs">Eggs</option>
                  <option value="fish">Fish</option>
                  <option value="hotmeat">Hot Meat</option>
                  <option value="bread">Bread</option>
                  <option value="preserves">Preserves</option>
                  <option value="beverage">Beverage</option>
                </select>
              </div>
              <div class="form-group">
                <label for="date">Date</label>
                <input class="form-control"
                  name="date" id="date" type="date">
              </div>
              <div class="form-group">
                <label for="latitude">Latitude</label>
                <input class="form-control"
                  name="latitude" id="latitude" type="text">
              </div>
              <div class="form-group">
                <label for="longitude">Longitude</label>
                <input class="form-control"
                  name="longitude" id="longitude" type="text">
              </div>
              <div class="form-group">
                <label for="description">Description</label>
                <textarea class="form-control"
                  name="description" id="description">
                </textarea>
              </div>
              <button class="btn btn-primary" type="submit">
                Submit
              </button>
            </form>
          </div>
        </div>
      </div>
    </div>
  </div>
```

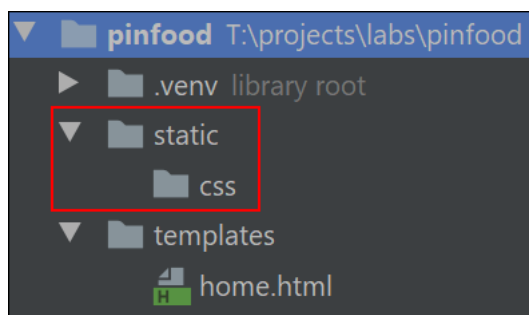
```
</div>
</body>
```

Refresh **http://localhost:5000** again. The form will appear at the bottom:



Adding bootstrap style

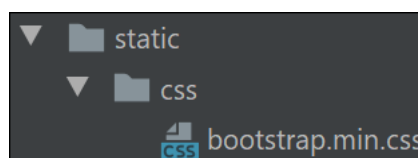
Next, create a folder structure like this:



Download **Bootstrap 4** at the following link:

<https://github.com/twbs/bootstrap/archive/v4.0.0.zip>

Then extract and copy **bootstrap.min.css** into **pinfood/static/css**:



Open the **templates/home.html** file and add the bootstrap css to it (put it after the script tag):

```
<link
  type="text/css"
  rel="stylesheet"
  href="{{url_for('static', filename='css/bootstrap.min.css')}}"
/>
```

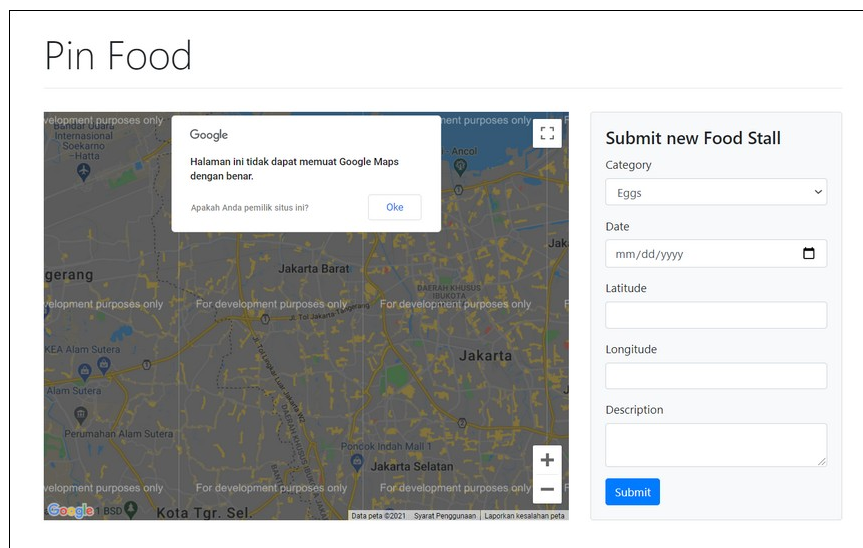
Now we can change parts:

```
<div id="maps" style="width: 70%; height: 500px"></div>
```

To be like:

```
<div id="maps" style="height: 100%; width: 100%;"></div>
```

Refresh again **http://localhost:5000**:

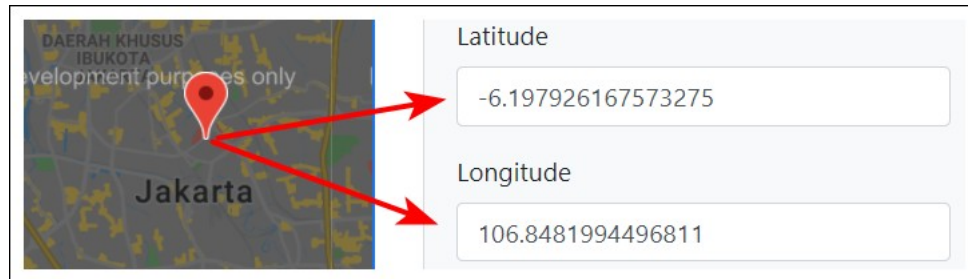


Next, add the following JavaScript code inside the **setMarker** function:

```
function setMarker(location) {
  if (marker) {
    // ...
  } else {
    // ...
  }

  document.getElementById('latitude')
    .value = location.lat();
  document.getElementById('longitude')
    .value = location.lng();
}
```

Refresh again **http://localhost:5000**. Try clicking any location on the map, long and lat values will appear in the Latitude and Longitude fields:



Storing food stall data

Open the **pinfood.py** file and replace the **add** function with **save_food** function:

```
@app.route("/savefood", methods=['POST'])
def save_food():
    category = request.form.get("category")
    date = request.form.get("date")
    latitude = float(request.form.get("latitude"))
    longitude = float(request.form.get("longitude"))
    description = request.form.get("description")
    DB.add_food(category,
                date,
                latitude,
                longitude,
                description)
    return home()
```

Open the **dbhelper.py** file and replace the **add_input** method with the **add_food**:

```
def add_food(self, category, date, latitude, longitude, description):
    connection = self.connect()

    try:
        query = "INSERT INTO foods (" \
                "category, " \
                "date, " \
                "latitude, " \
                "longitude, " \
                "description) \
                VALUES (%s, %s, %s, %s, %s)"

        with connection.cursor() as cursor:
            cursor.execute(query,
                           (category,
                            date,
                            latitude,
                            longitude,
                            description))
            connection.commit()

    except Exception as e:
```

Adding Marker

```
print(e)

finally:
    connection.close()
```

Refresh again **http://localhost:5000/**. Try entering data in the form and click submit. Data will be stored in the database:

id	latitude	longitude	date	category	description	updated_at
7	-6.214027	106.840599	2021-01-24 00:00:00	eggs	Umm yummy egs.	2021-01-24 13:49:49
8	-6.215885	106.835030	2021-01-24 00:00:00	fish	I like Fish food in here	2021-01-24 13:50:48

Displaying existing food stall

Open the **dbhelper.py** file and change the **get_all_inputs** method to **get_all_foods**:

```
from datetime import datetime

class DBHelper:

    # ...

    def get_all_foods(self):
        connection = self.connect()
        try:
            query = "SELECT latitude, " \
                    "longitude, " \
                    "date, " \
                    "category, " \
                    "description FROM foods;"

            with connection.cursor() as cursor:
                cursor.execute(query)

            named_foods = []

            for food in cursor:
                named_food = {
                    'latitude': food[0],
                    'longitude': food[1],
                    'date': datetime.strptime(food[2], '%Y- %m-%d'),
                    'category': food[3],
                    'description': food[4]
                }

                named_foods.append(named_food)

            return named_foods

        finally:
            connection.close()
```

Next, open the **pinfood.py** file and change the body of the **home** function to:

```
import json

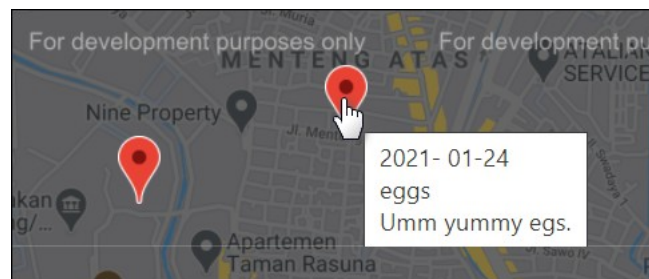
@app.route("/")
def home():
    foods = DB.get_all_foods()
    foods = json.dumps(foods)
    return render_template("home.html", foods=foods)
```

Then, open the **templates/home.py** file and add the following **placeFood** function inside the script tag:

```
<script type="text/javascript">
```

```
function init() {  
  // ....  
  placeFood({foods | safe});  
}  
  
// ...  
  
function placeFood(foods) {  
  for (i=0; i<foods.length; i++) {  
    new google.maps.Marker( {  
      position: new google.maps.LatLng(  
        foods[i].latitude,  
        foods[i].longitude),  
      map: map,  
      title: foods[i].date + "\n" +  
        foods[i].category + "\n" + foods[i].description  
    });  
  }  
}  
}  
</script>
```

Refresh again **http://localhost:5000/** and the result will appear all the food stalls in the map:



** **Note:** For optimal results, try adding some data and deleting previous data that has many NULL values.*