Internet of Things

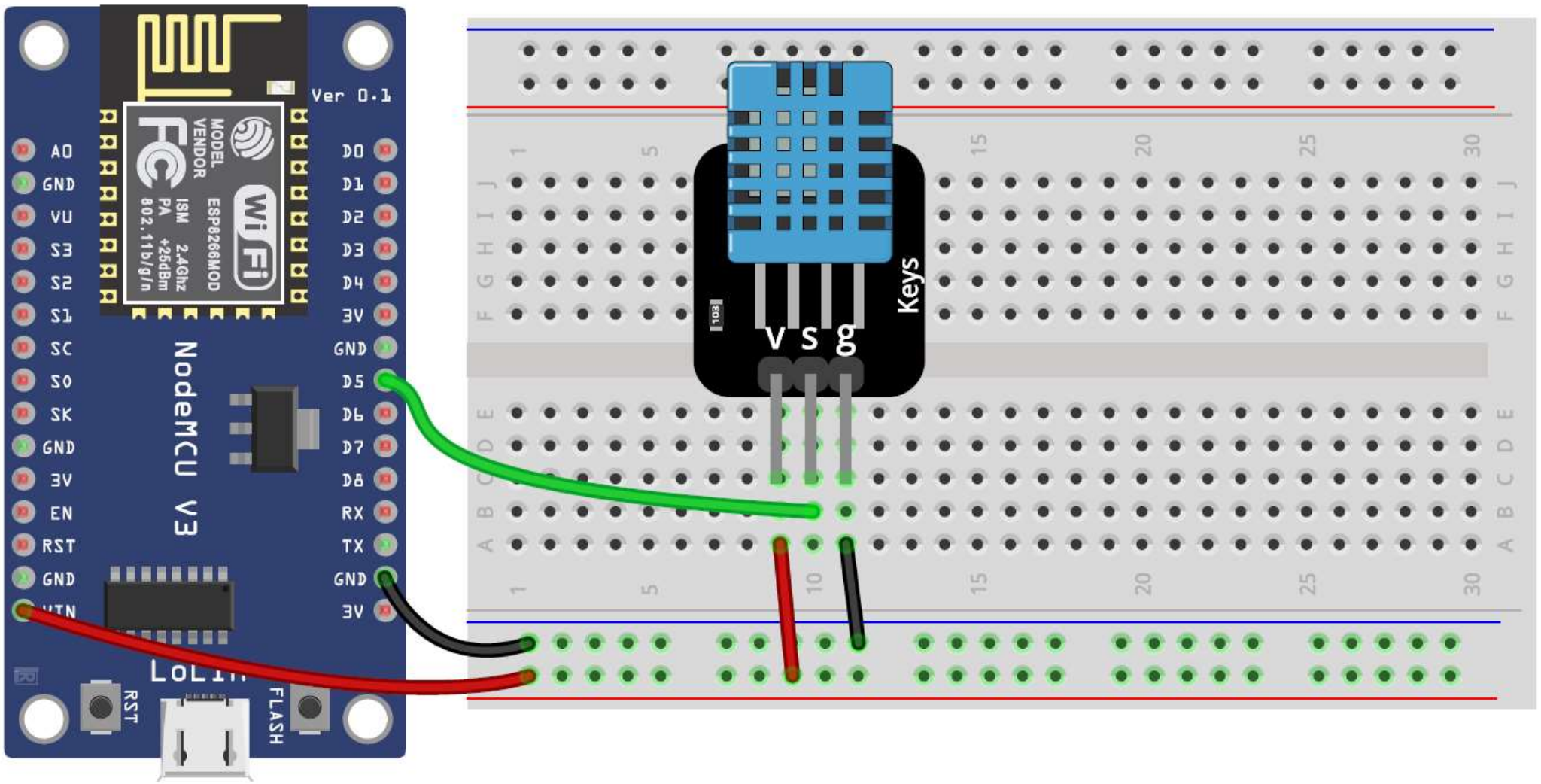
Dashboard Using Firebase, Plotly and Flask

IoT Team, BFCAI

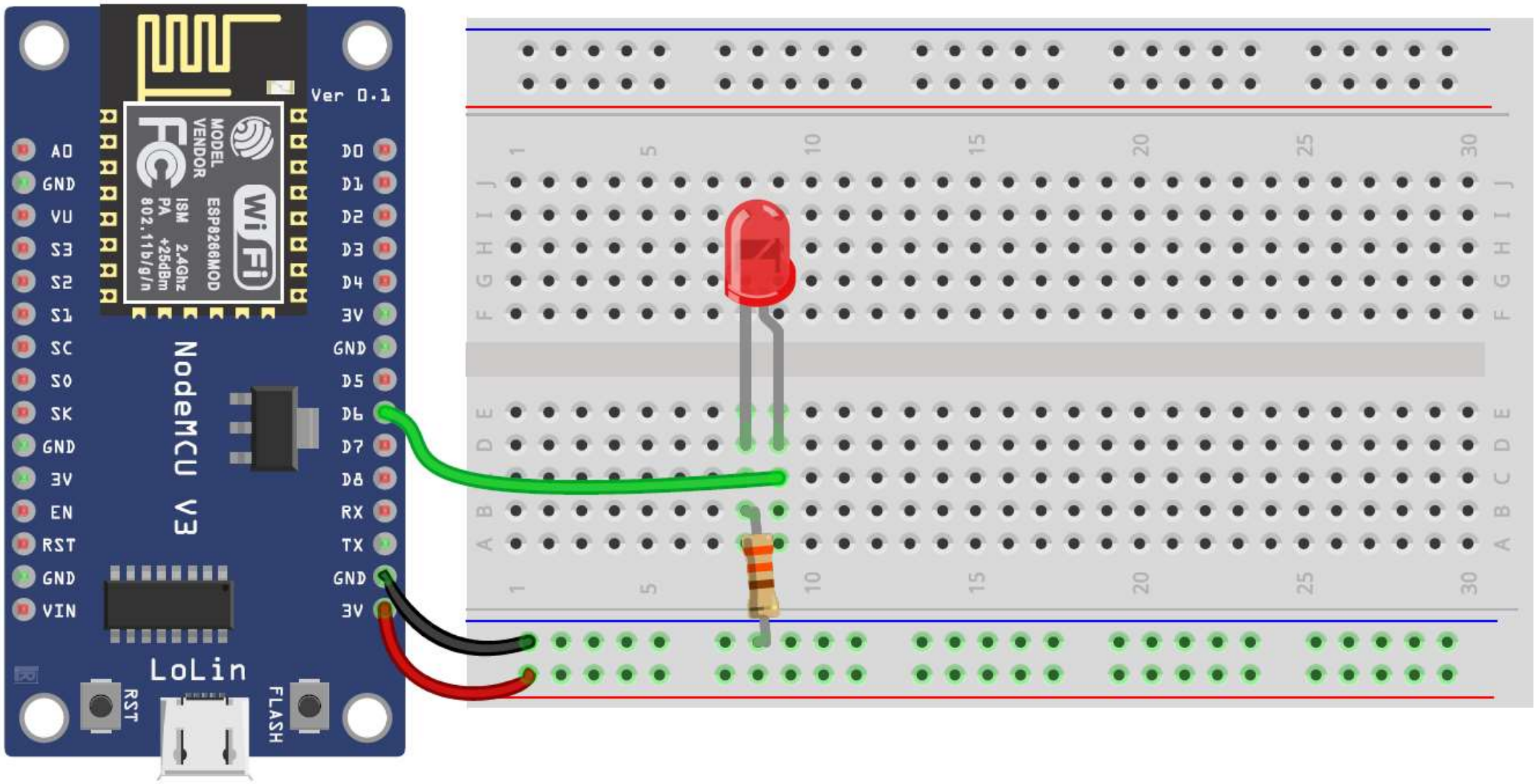
Firebase

- Firebase is Google's mobile and web application development platform that includes many services to manage data from iOS, Android, or web applications.
- This includes things like analytics, authentication, databases, configuration, file storage, push messaging, and the list goes on.
- The services are hosted in the cloud and scale with little to no effort on the part of the developer.

Firebase: Sending Temperature Values



Firebase: Controlling an LED from Anywhere



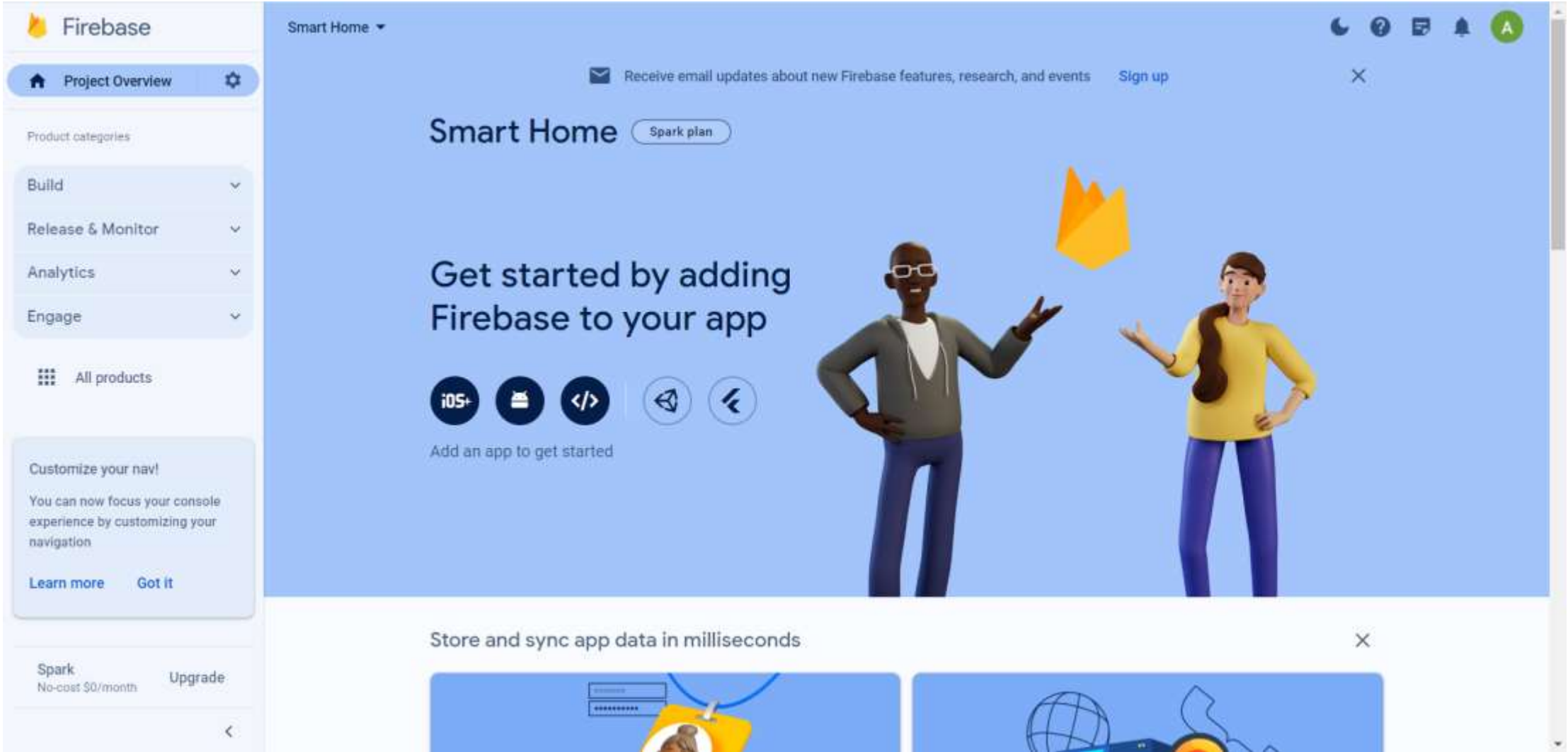
Firebase Admin SDK

- The Firebase Admin SDK provides a powerful set of tools for interacting with Firebase services, including the Realtime Database.
- In Python, you can use the Firebase Admin SDK to access and manipulate data in the Firebase Realtime Database.
- To use Admin Database API in Python, we need to install the following.

```
pip install firebase-admin
```

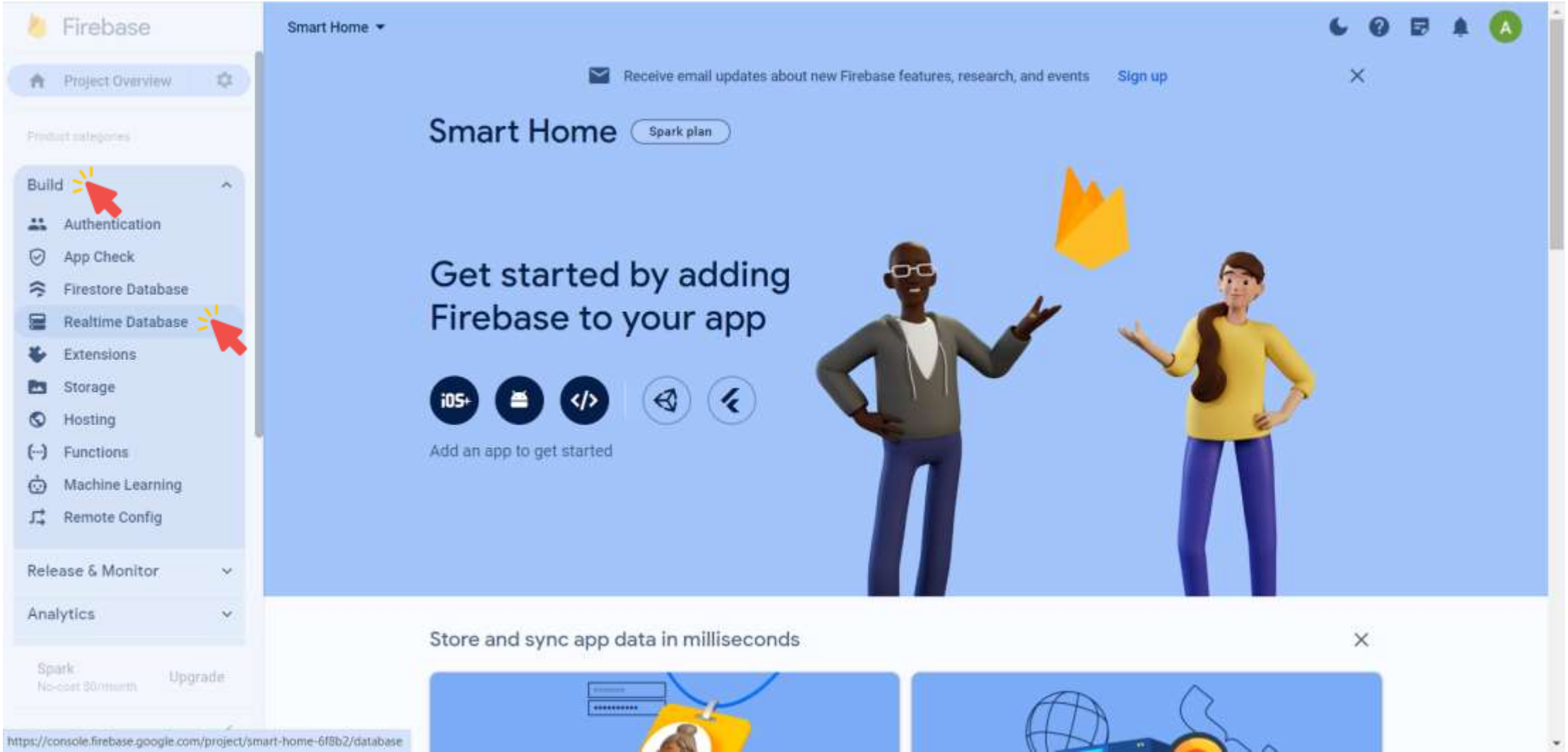
Firebase Admin SDK

- Open your Firebase project.



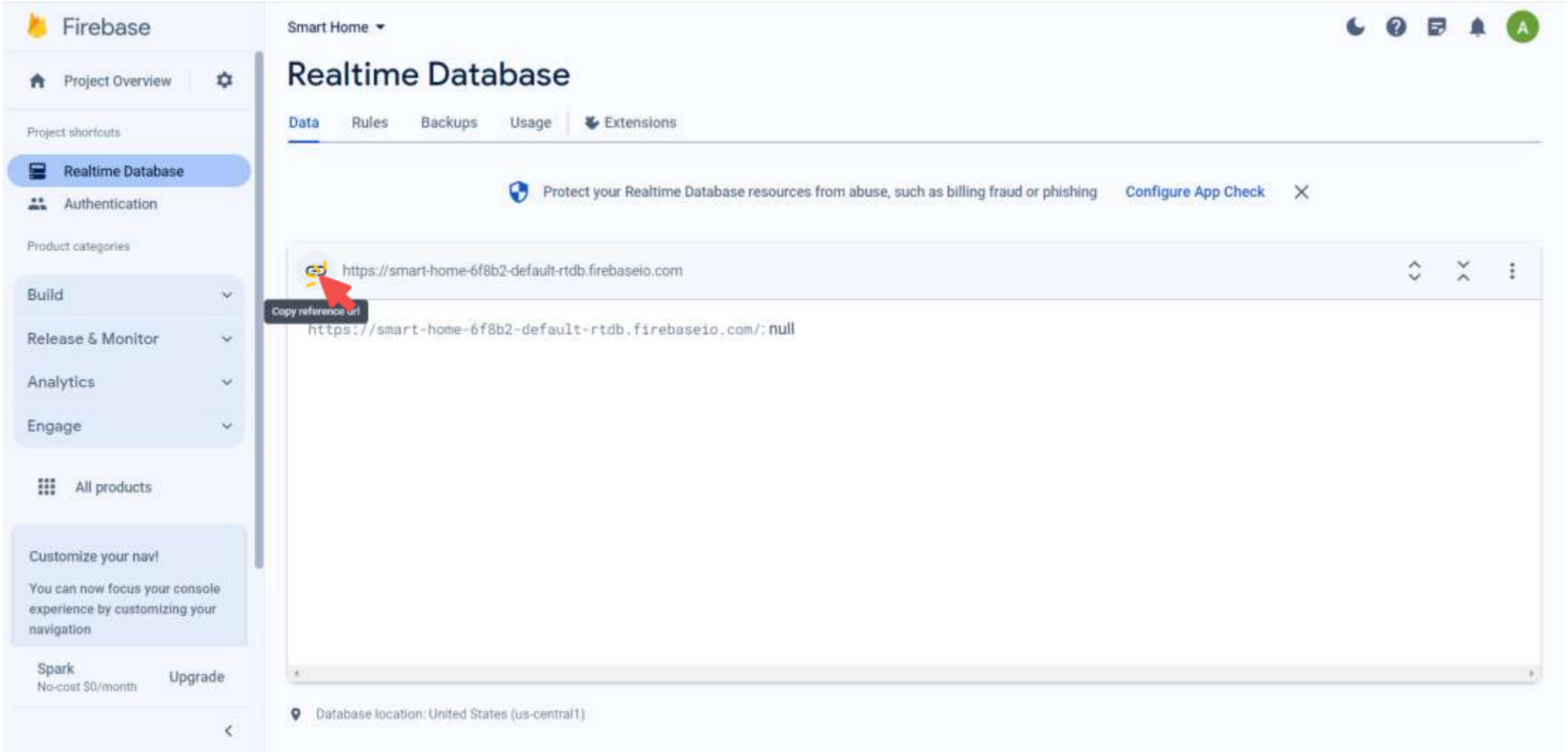
Firebase Admin SDK

- Expand Build tap, and go to Realtime Database.



Firebase Admin SDK

- Copy the database URL.

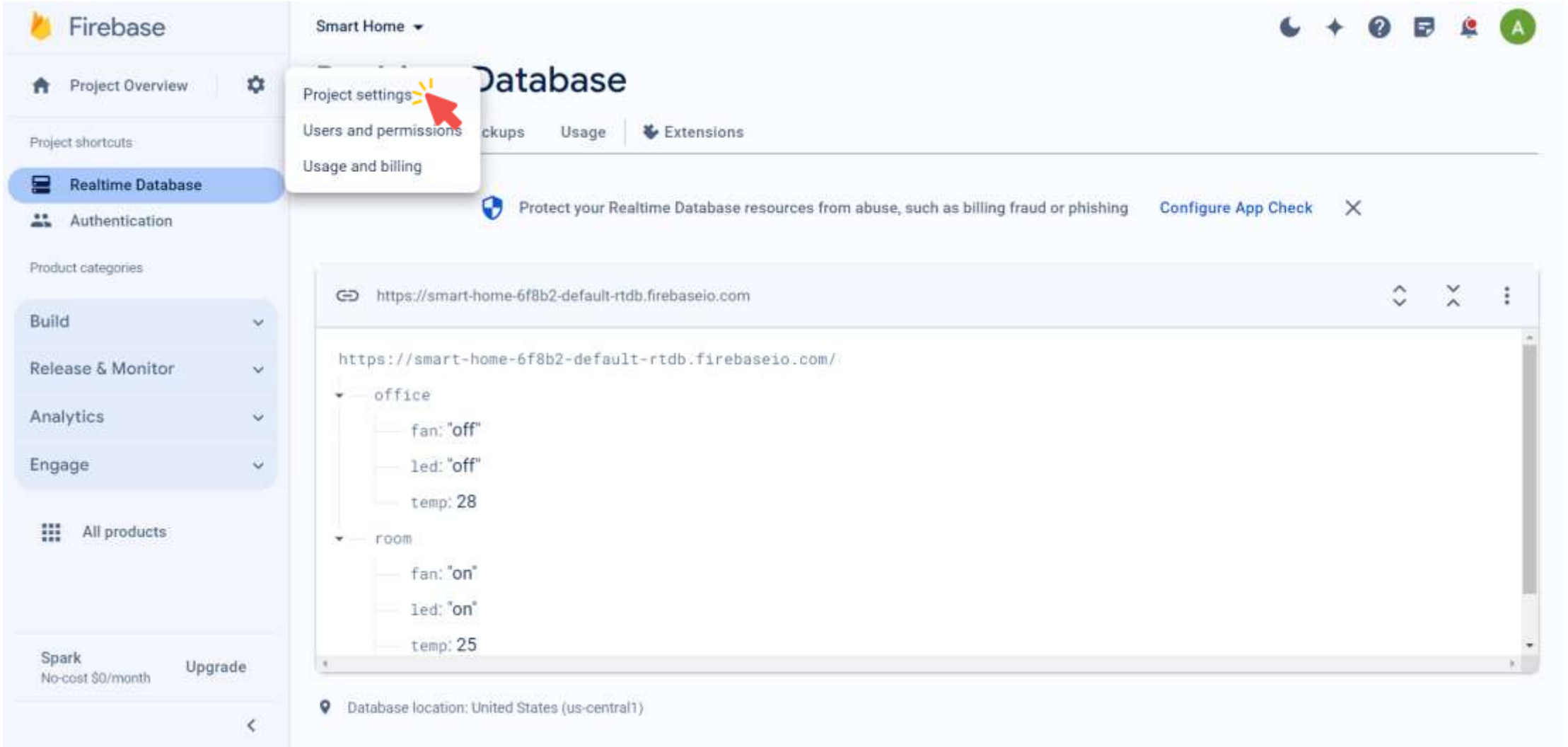


Firebase Admin SDK

- Create the following nodes in the database.

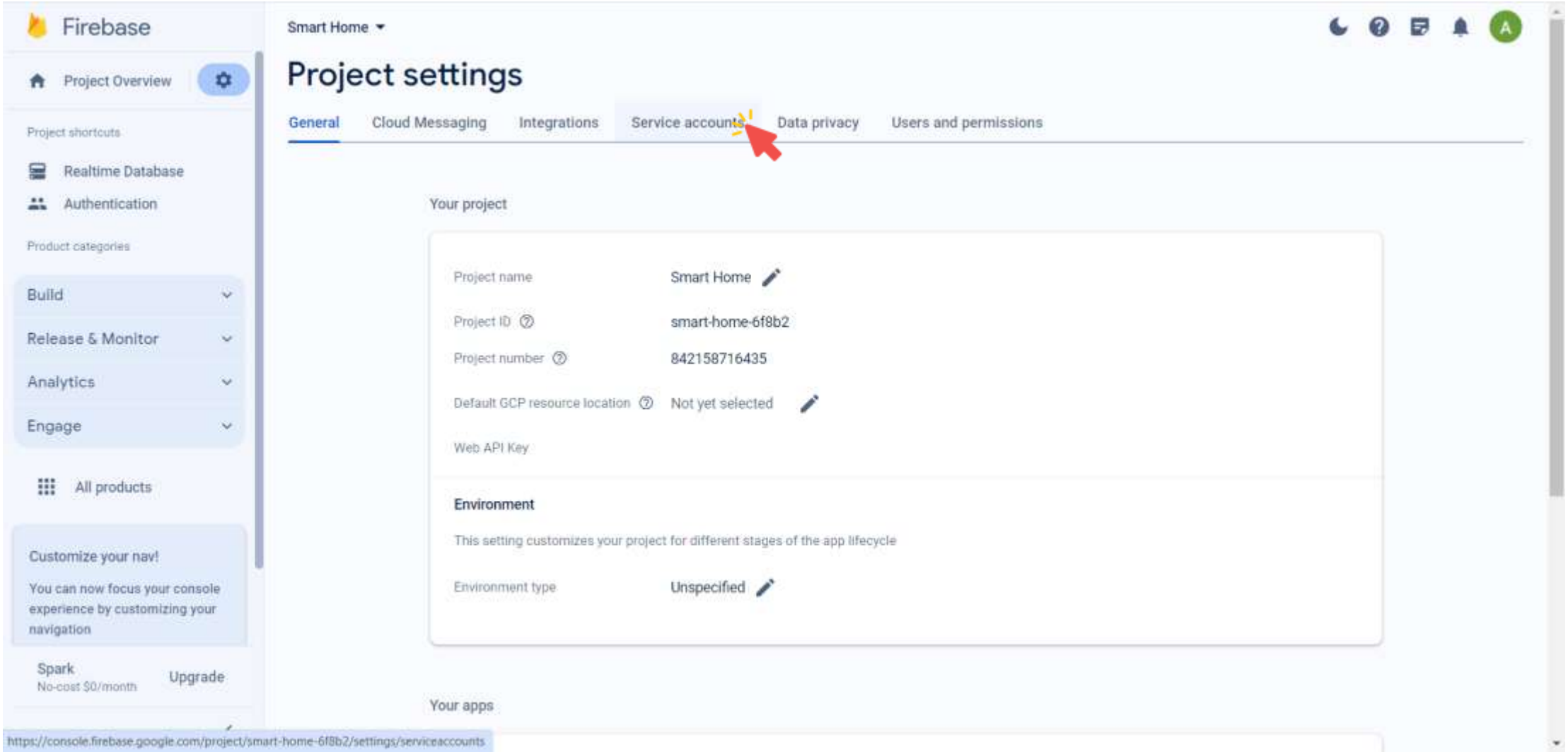
Firebase Admin SDK

- Goto Project settings.



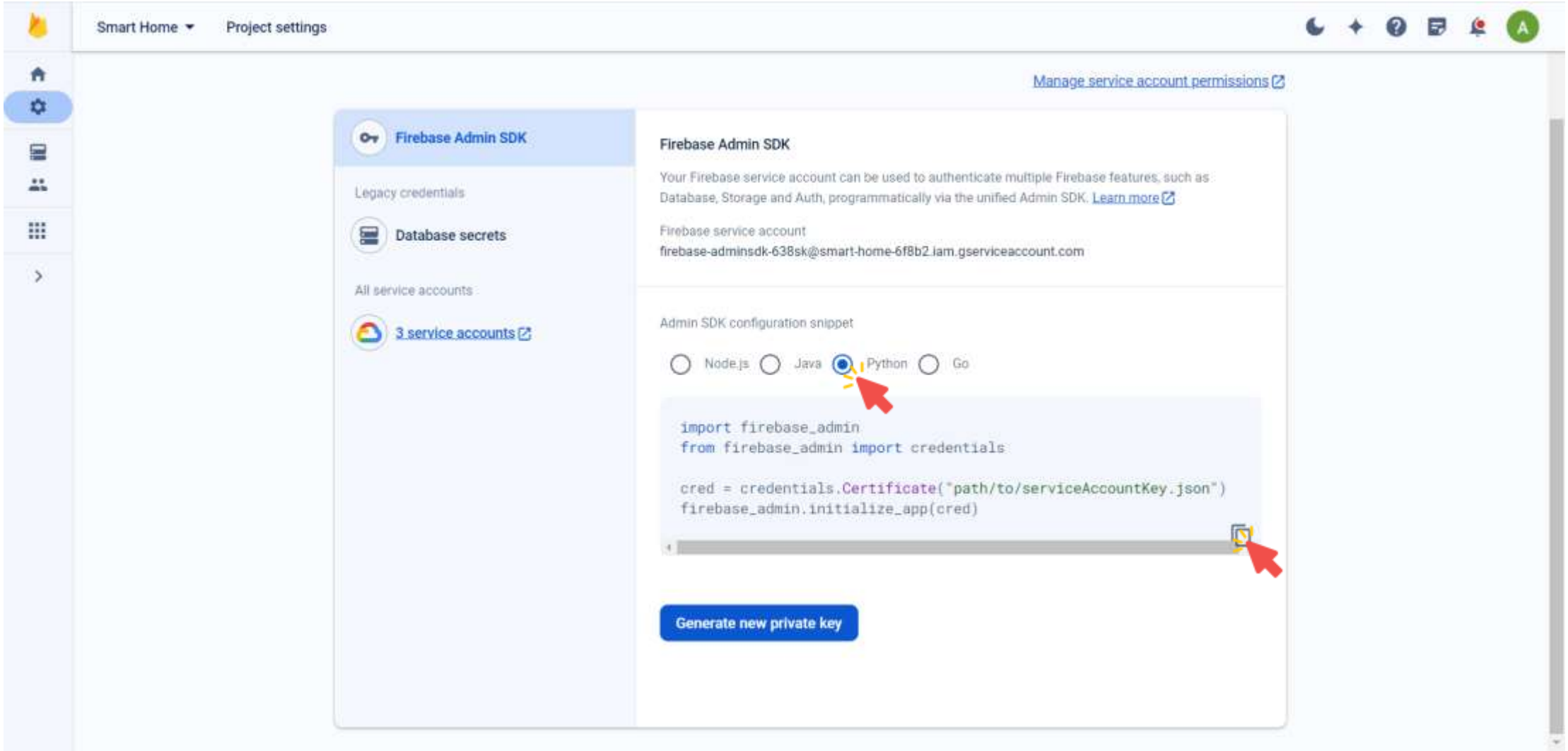
Firebase Admin SDK

- Click on Service accounts.



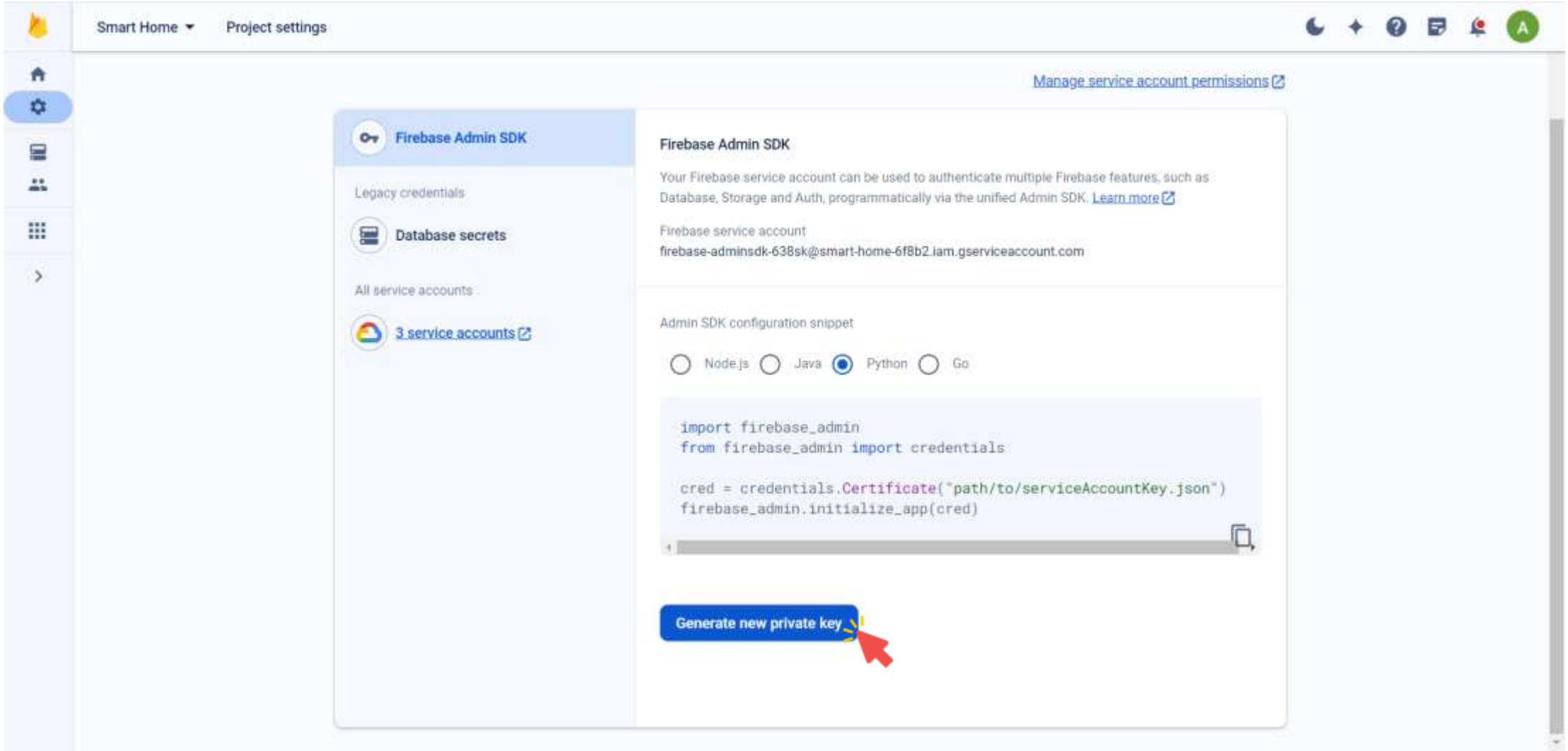
Firebase Admin SDK

- Choose Python and copy the code.



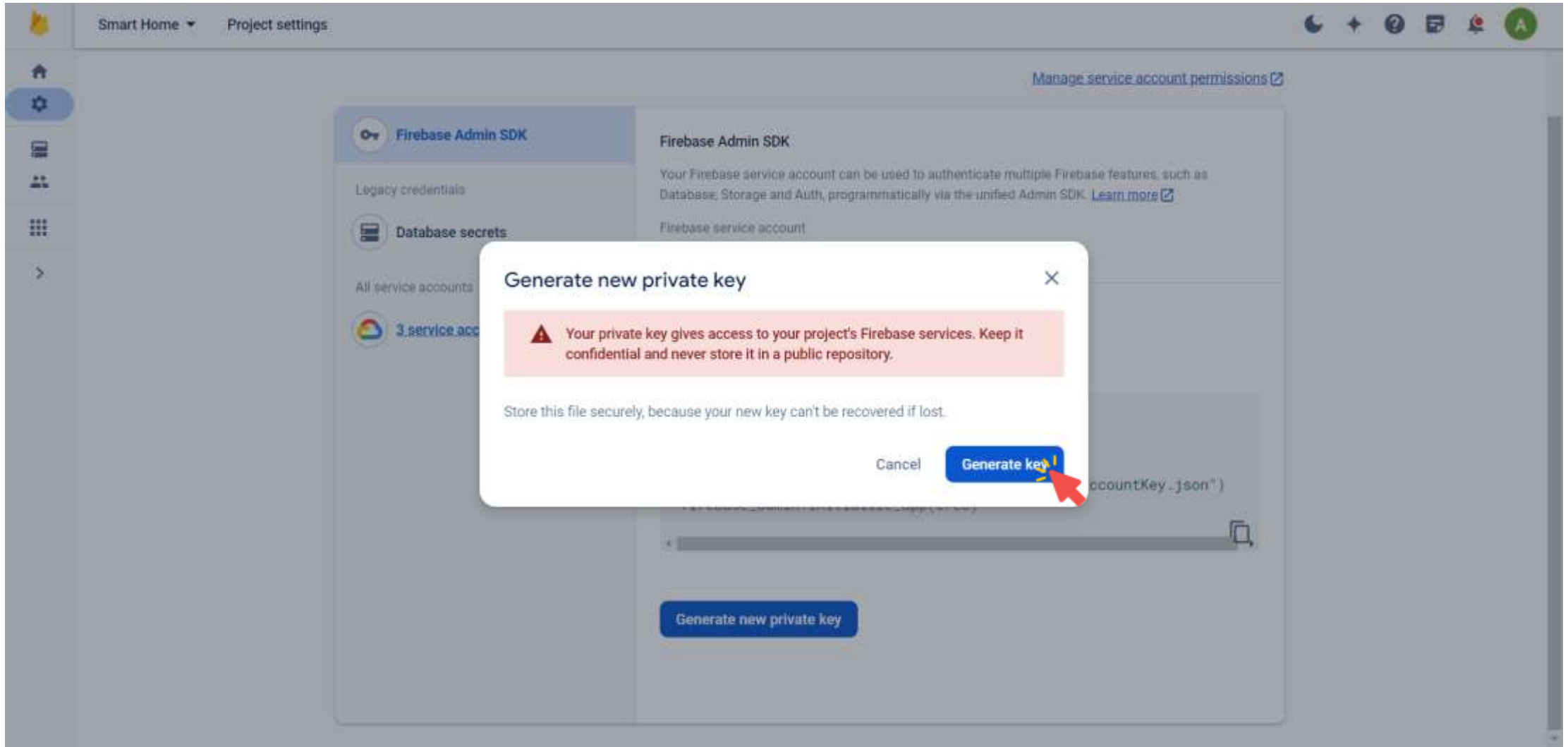
Firebase Admin SDK

- Click on Generate new private key.



Firebase Admin SDK

- Click on Generate key.



Firebase Admin SDK

- This will download a JSON file that contains the private key.
- For simplicity, rename the file as `key.json`.

Firebase Admin SDK: Imports

- The `firebase_admin` module provides functionality for working with Firebase services.

```
import firebase_admin
```

- The `credentials` submodule contains functions for managing credentials.

```
from firebase_admin import credentials
```

- The `db` submodule from provides access to the Firebase Realtime Database.

```
from firebase_admin import db
```

Firebase Admin SDK: Connecting to the Realtime Database

- The following initializes a credentials object using a service account key file (`key.json`) for authentication.

```
cred = credentials.Certificate("key.json")
```

- Initialize the Firebase Admin SDK with the provided credentials and database URL.

```
firebase_admin.initialize_app(cred,
                              {'databaseURL':
'https://smart-home-6f8b2-default-rtdb.firebaseio.com/'})
```

Firebase Admin SDK: Getting Data from the Realtime Database

- This following creates a reference to the root of the Firebase Realtime Database, which allows you to read, write, and manipulate data at the root level of the database.

```
ref = db.reference('/')
```

- This following retrieves the current data stored at the root.

```
ref.get()
```

```
Out: {'office': {'fan': 'off', 'led': 'off', 'temp': 28},
      'room': {'fan': 'on', 'led': 'on', 'temp': 25, 'tv': 'on'}}
```

Firebase Admin SDK: Getting Data from the Realtime Database

- This code retrieves the current data stored at the `office` child node.

```
ref.child('office').get()
```

```
Out: {'fan': 'off', 'led': 'off', 'temp': 28}
```

- This code retrieves the current data stored at the `room` child node.

```
ref.child('room').get()
```

```
Out: {'fan': 'on', 'led': 'on', 'temp': 25, 'tv': 'on'}
```

Firebase Admin SDK: Getting Data from the Realtime Database

- This code retrieves the current data stored at the `tv` child node under the `room` child node.

```
ref.child('room').child('tv').get()
```

```
Out: 'on'
```

- The following does exactly the same.

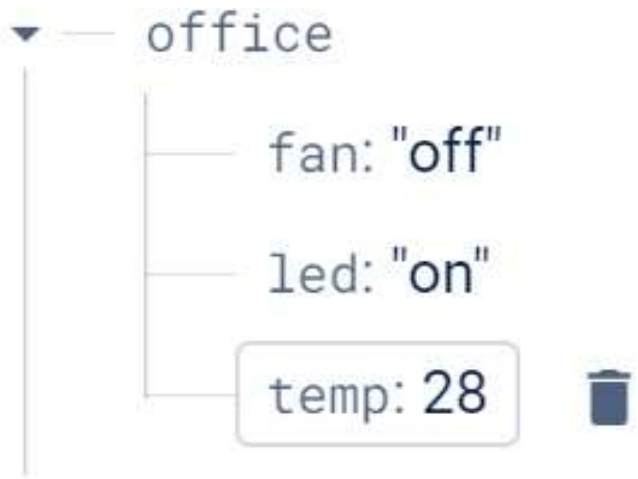
```
ref.child('room/tv').get()
```

```
Out: 'on'
```

Firebase Admin SDK: Sending Data to the Realtime Database

- This code sets the value of the `led` child node under the `office` child node to "on" in the Firebase Realtime Database.

```
ref.child('office/led').set("on")
```



Firebase Admin SDK: Sending Data to the Realtime Database

- These lines of code add and set the values of the `gas_level` and `oven` child nodes under the kitchen node to “low” and “off ”, respectively.

```
ref.child('kitchen/gas_level').set("low")
ref.child('kitchen/oven').set("off")
```

Firebase Admin SDK: Sending Data to the Realtime Database

- If the data does not exist, calling `set()` will add a new node with the data.
- If data already exists, calling `set()` will overwrite it with the new data.

```
ref.child('temp_sensor').set(25)
```

```
temp_sensor: 25
```

```
ref.child('temp_sensor').set(27)
```

```
temp_sensor: 27
```

Firebase Admin SDK: Sending Data to the Realtime Database

- This code deletes the `temp_sensor` node from the Realtime Database.

```
{
    "kitchen": {
      "gas_level": "low",
      "oven": "off"
    },
    "office": {
      "fan": "off",
      "led": "on",
      "temp": 28
    },
    "room": {
      "fan": "on",
      "led": "on",
      "temp": 25,
      "tv": "on"
    }
  }
```

Firebase Admin SDK: Sending Data to the Realtime Database

- The `push()` function is used to generate a unique key for a new child node and then set data at that location.
- The following will add five child nodes to the `temp_sensor` ensures uniqueness of keys.

```
ref.child('temp_sensor').push({'timestamp': 1, 'temp': 26})
ref.child('temp_sensor').push({'timestamp': 2, 'temp': 27})
ref.child('temp_sensor').push({'timestamp': 3, 'temp': 28})
ref.child('temp_sensor').push({'timestamp': 4, 'temp': 26})
ref.child('temp_sensor').push({'timestamp': 5, 'temp': 25})
```

Firebase Admin SDK: Sending Data to the Realtime Database

- The `push()` is used when you want to add data in a way that ensures uniqueness of keys and preserves order.

Firebase Admin SDK: Getting and Preprocessing Data

- This will get the data stored at the `temp_sensor` node.

```
temp_sensor = ref.child('temp_sensor').get()
```

Out:

```
{'-NwSfZyWwTj6enKMsg_J': {'temp': 26, 'timestamp': 1},
 '-NwSf_2SHOxEjoh3SV_S': {'temp': 27, 'timestamp': 2},
 '-NwSf_7I6WQMgDa57Fje': {'temp': 28, 'timestamp': 3},
 '-NwSf_C0rNmz2wSOnXDh': {'temp': 26, 'timestamp': 4},
 '-NwSf_GmdxwhQlYD0IZN': {'temp': 25, 'timestamp': 5}}
```

Firebase Admin SDK: Getting and Preprocessing Data

- This code iterates over the items in the `temp_sensor` dictionary retrieved from the Firebase Realtime Database.
- For each child node under `temp_sensor`, it extracts the `temp` and timestamp values and appends them to separate lists.

```
timestamp = []
temp = []

for key, val in temp_sensor.items():
    temp.append(val['temp'])
    timestamp.append(val['timestamp'])
```

Firebase Admin SDK: Getting and Preprocessing Data

In:
```
timestamp
```

Out:
```
[1, 2, 3, 4, 5]
```

In:
```
temp
```

Out:
```
[26, 27, 28, 26, 25]
```

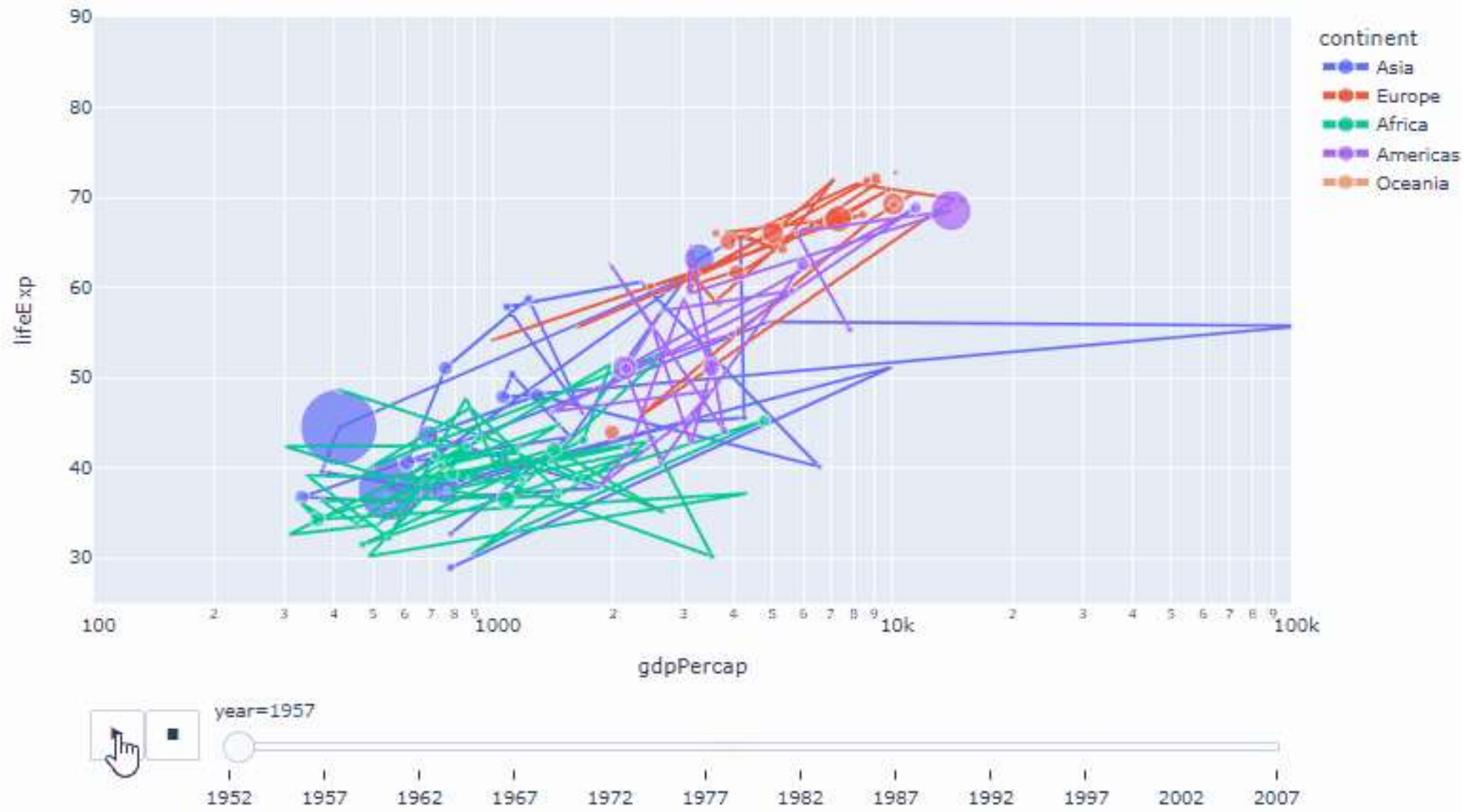
Plotly Python

- The `plotly.py` is an interactive, open-source, and browser-based graphing library for Python.
- Built on top of `plotly.js` which is a high-level charting library.
- `plotly.js` ships with over 30 chart types, including scientific charts, 3D graphs, statistical charts, SVG maps, financial charts, and more.
- Plotly graphs can be viewed in Jupyter notebooks, standalone HTML files, or integrated into Dash applications.
- Plotly Python may be installed using pip.

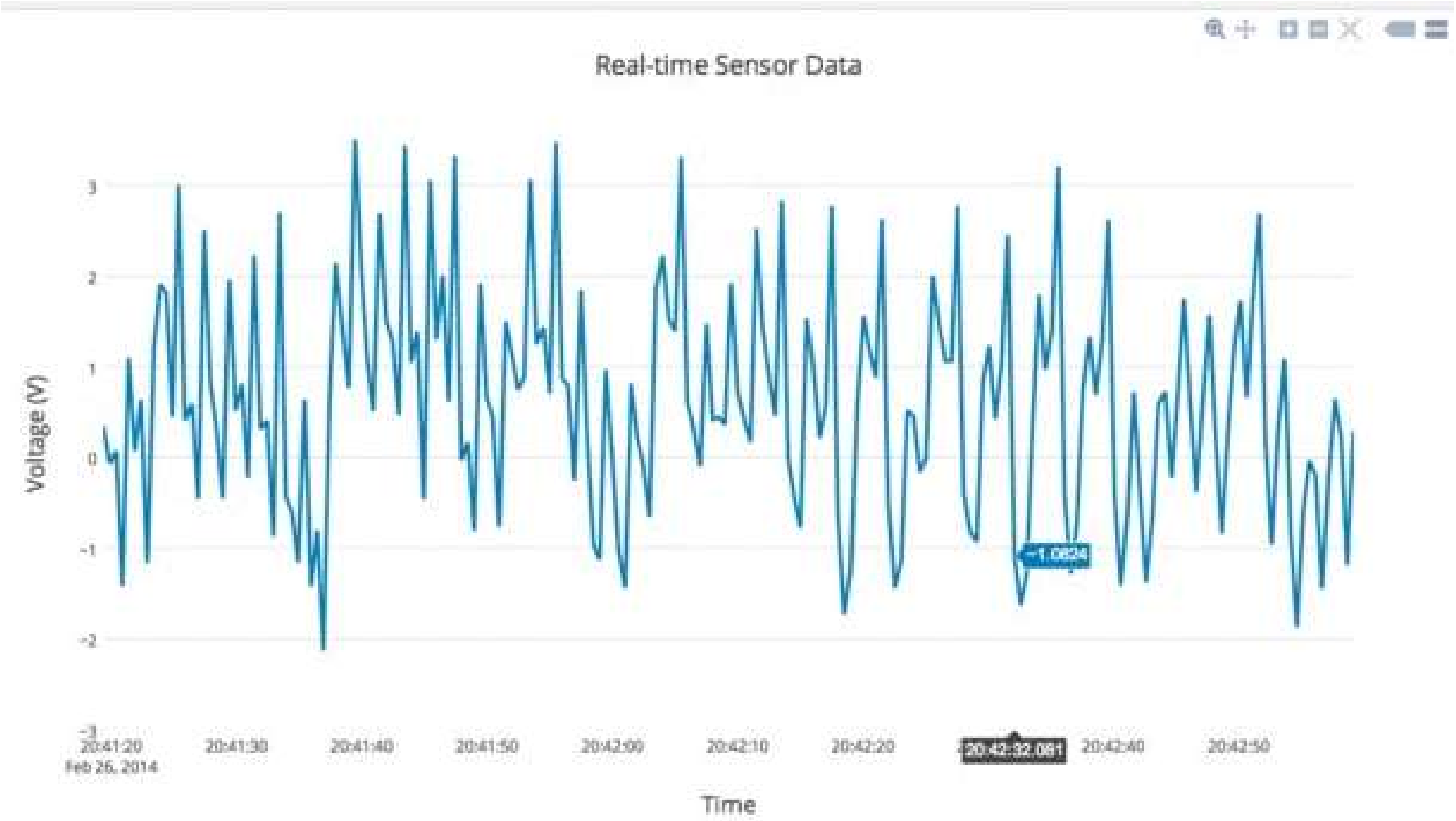
```
pip install plotly
```

Plotly Python

Plotly Python



Plotly Python



Plotly Python

- By importing `plotly.graph_objects`, you gain access to a powerful set of tools for creating interactive graphs in Python.

```
import plotly.graph_objects as go
```

- This creates a scatter plot using Plotly's `graph_objects` module.

```
data = go.Scatter(x=timestamp, y=temp)
```

- The following creates a figure object and assigns it to the variable `fig`.

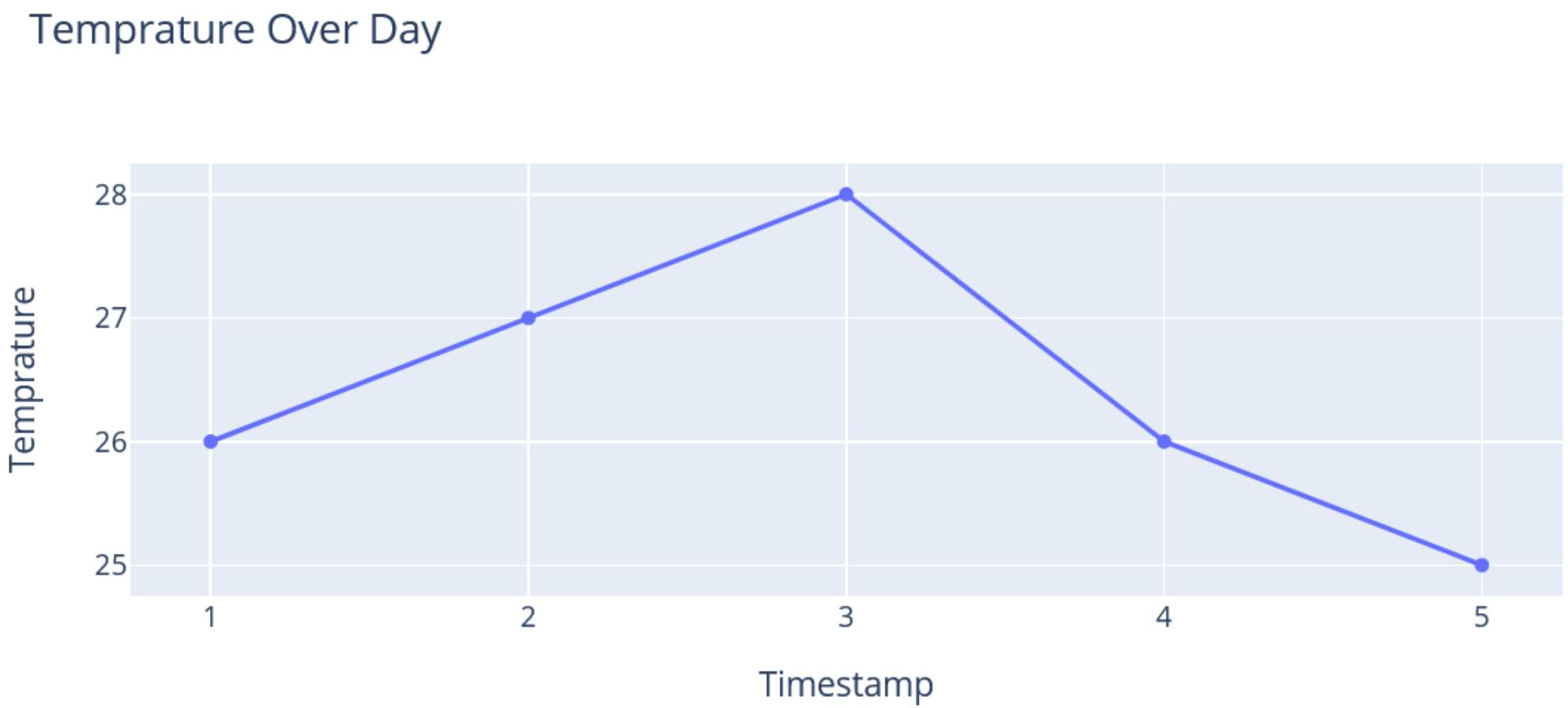
```
fig = go.Figure(data=data)
```

- The following lines of code update the layout properties of the figure.

```
fig.layout.title = 'Temprature Over Day'
fig.layout.xaxis.title = 'Timestamp'
fig.layout.yaxis.title = 'Temprature'
```

Plotly Python

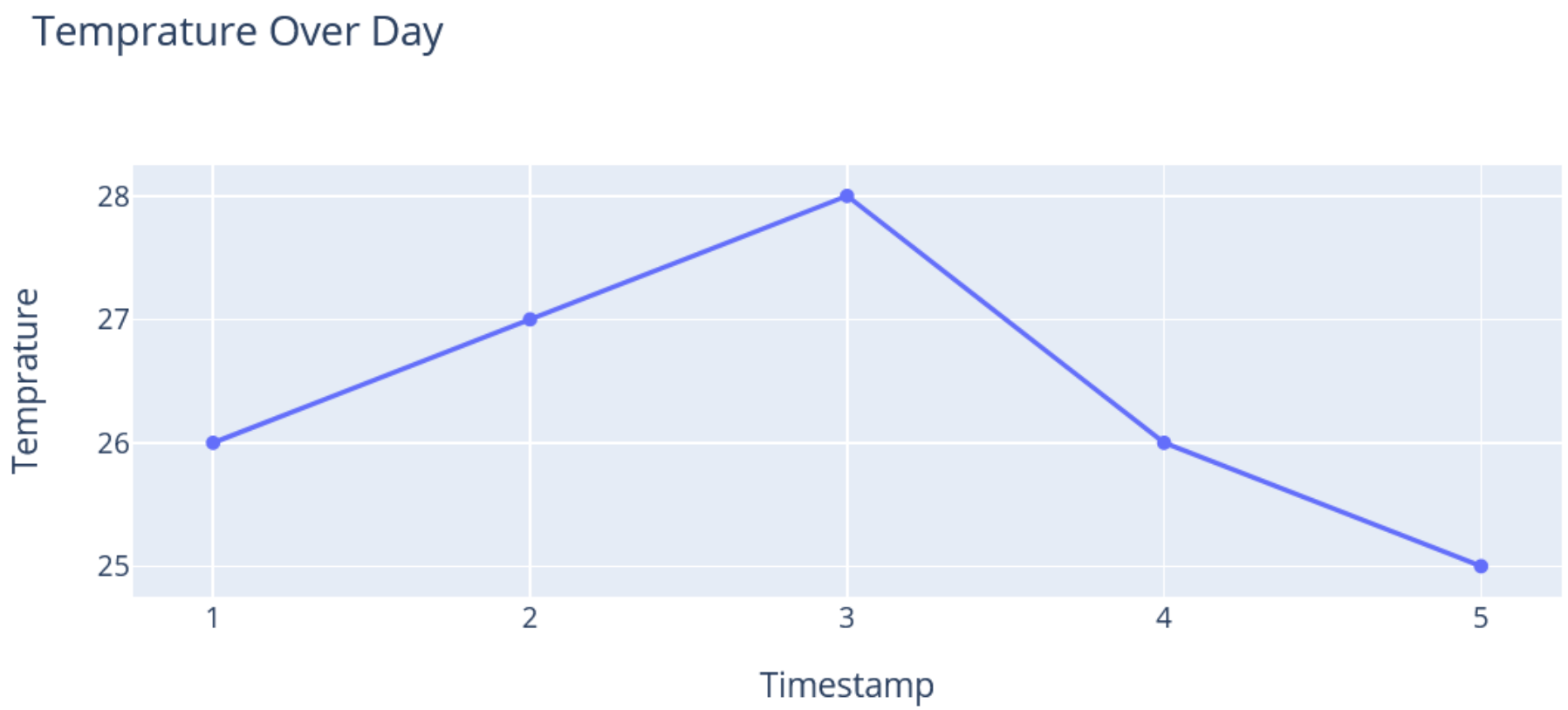
```
fig.show()
```



Plotly Python

- Converting the figure to JSON format is useful while rendering the figure in HTML pages.

```
fig_json = fig.to_json()
```



Flask Web Application Framework

- Flask is a lightweight web application framework.
- It is one of the most popular Python web application frameworks.
- It is designed to make getting started quick and easy.

```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello, World!"

if __name__ == "__main__":
    app.run()
```

Simple Dashboard Using Plotly and Flask

- This function retrieves temperature data from a Firebase and returns the timestamps and temperature values as lists.

```
def get_temp_values():
    ref = db.reference('/')
    temp_sensor = ref.child('temp_sensor').get()

    timestamp = []
    temp = []

    for key, val in temp_sensor.items():
        temp.append(val['temp'])
        timestamp.append(val['timestamp'])

    return timestamp, temp
```

Simple Dashboard Using Plotly and Flask

- The function encapsulates the process of creating a Plotly figure representing temperature data over time.

```
def viz_temp(timestamp, temp):
    data = go.Scatter(x=timestamp, y=temp)
    fig = go.Figure(data=data)

    fig.layout.title = 'Temprature Over Day'
    fig.layout.xaxis.title = 'Timestamp'
    fig.layout.yaxis.title = 'Temprature'

    return fig
```

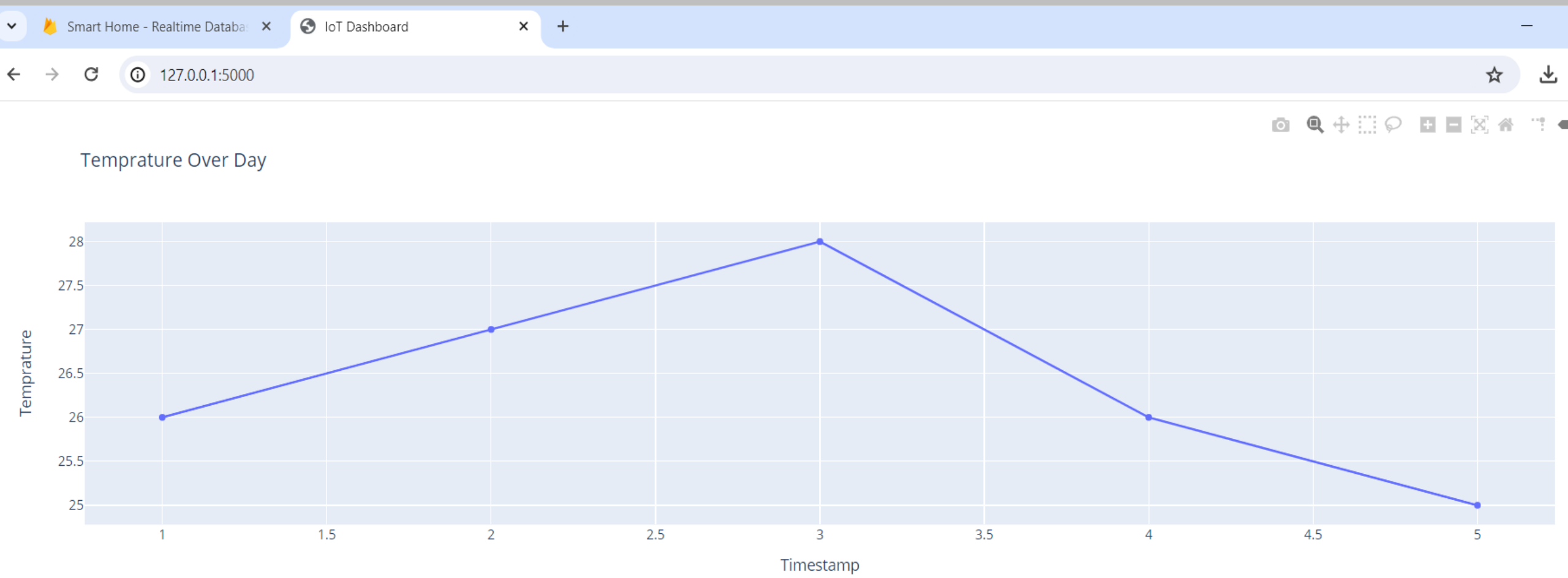
Simple Dashboard Using Plotly and Flask

```
from flask import Flask, render_template

# Create a Flask application object
app = Flask(__name__)

# Define a route for the root URL ('/')
@app.route('/')
def index():
    # Get timestamp and temperature values
    timestamp, temp = get_temp_values()

    # Visualize temperature data and get a Plotly figure
    fig = viz_temp(timestamp, temp)

    # Convert the Plotly figure to JSON format
    fig_json = fig.to_json()

    # Render the HTML template 'index.html', passing the Plotly figure JSON data
    return render_template('index.html', fig_json=fig_json)

# Check if the script is being run directly
if __name__ == '__main__':
    # Start the Flask server
    app.run()
```

Simple Dashboard Using Plotly and Flask

```
<!DOCTYPE html>
<html>

<head>
    <title>IoT Dashboard</title>
    <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
</head>

<body>
    <div id="temp_fig"></div>

    <script>
        var fig_json = {{ fig_json | safe }};
        Plotly.newPlot('temp_fig', fig_json.data, fig_json.layout);
    </script>
</body>

</html>
```

Simple Dashboard Using Plotly and Flask



The Flask Server Running on http://127.0.0.1:5000

Flask Tutorial

133 

Learn Python in Arabic #133 - Flask - Intro And Your First Page

Elzero Web School • 55K views • 3 years ago

134 

Learn Python in Arabic #134 - Flask - Create Html Files

Elzero Web School • 32K views • 3 years ago

135 

Learn Python in Arabic #135 - Flask - Create And Extends HTML Templates

Elzero Web School • 24K views • 3 years ago

136 

Learn Python in Arabic #136 - Flask - Jinja Template

Elzero Web School • 22K views • 3 years ago

137 

Learn Python in Arabic #137 - Flask - Advanced Css Task Using Jinja

Elzero Web School • 18K views • 3 years ago

138 

Learn Python in Arabic #138 - Flask - Skills Page Using List Data

Elzero Web School • 16K views • 3 years ago

References and Tutorials

- How to Get Started with Firebase Using Python
- Introduction to the Admin Database API
- plotly.py
- Plotly Open-Source Graphing Library for Python
- Plotly JavaScript Open-Source Graphing Library
- Plotly - Getting Started in JavaScript
- Flask Tutorial - Elzero Web School