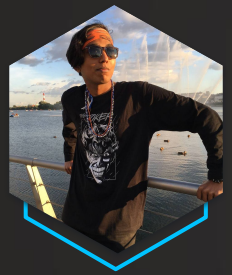
Innopolis University

Database Group Project

PostgreSQL to Google Firebase

Innopolis University

Team 🍀 Weed Detector 🔍

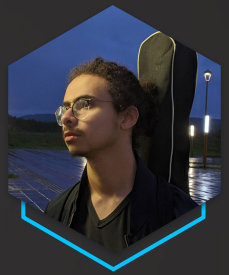


Parth Kalkar

BS19-02

Kamil Sabbagh

BS19-02



Rafik Hachana

BS19-04

Igor Mpore

BS19-06

innopolis university

AGENDA

01

Introduction

What is Google firebase and its Pros and Cons.

02

Installation

How to install it and how it operates in general

03

Migration

Scripts, components and bulking time (UML Diagram included)

04

Automatization

How migration can be automatized continuously

05

Queries

Comparing queries and performance

06

CONCLUSION

Conclusion and Questions from the audience will be WELCOME

innopolis university

01

Introduction

What is Google firebase and its Pros and Cons.

1.1 WHAT IS FIREBASE

The Firebase Realtime Database is a cloud-hosted NoSQL database that lets you store and sync data between your users in real-time. Data is stored as JSON and synchronized in real-time to every connected client. When you build cross-platform apps with iOS, Android, and JavaScript SDKs, all of your clients share one Realtime Database instance and automatically receive updates with the newest data.

1.2 KEY CAPABILITIES

innopolis university

Realtime

It uses data synchronization

Offline

It remains responsive even when offline

Accessibility

It can be accessed directly from a mobile device or web browser

Scalability

It can split your data across multiple database instances in the same Firebase project

Google indexing

Easy Google indexing by search

AdMob

It has focused in-application promoting

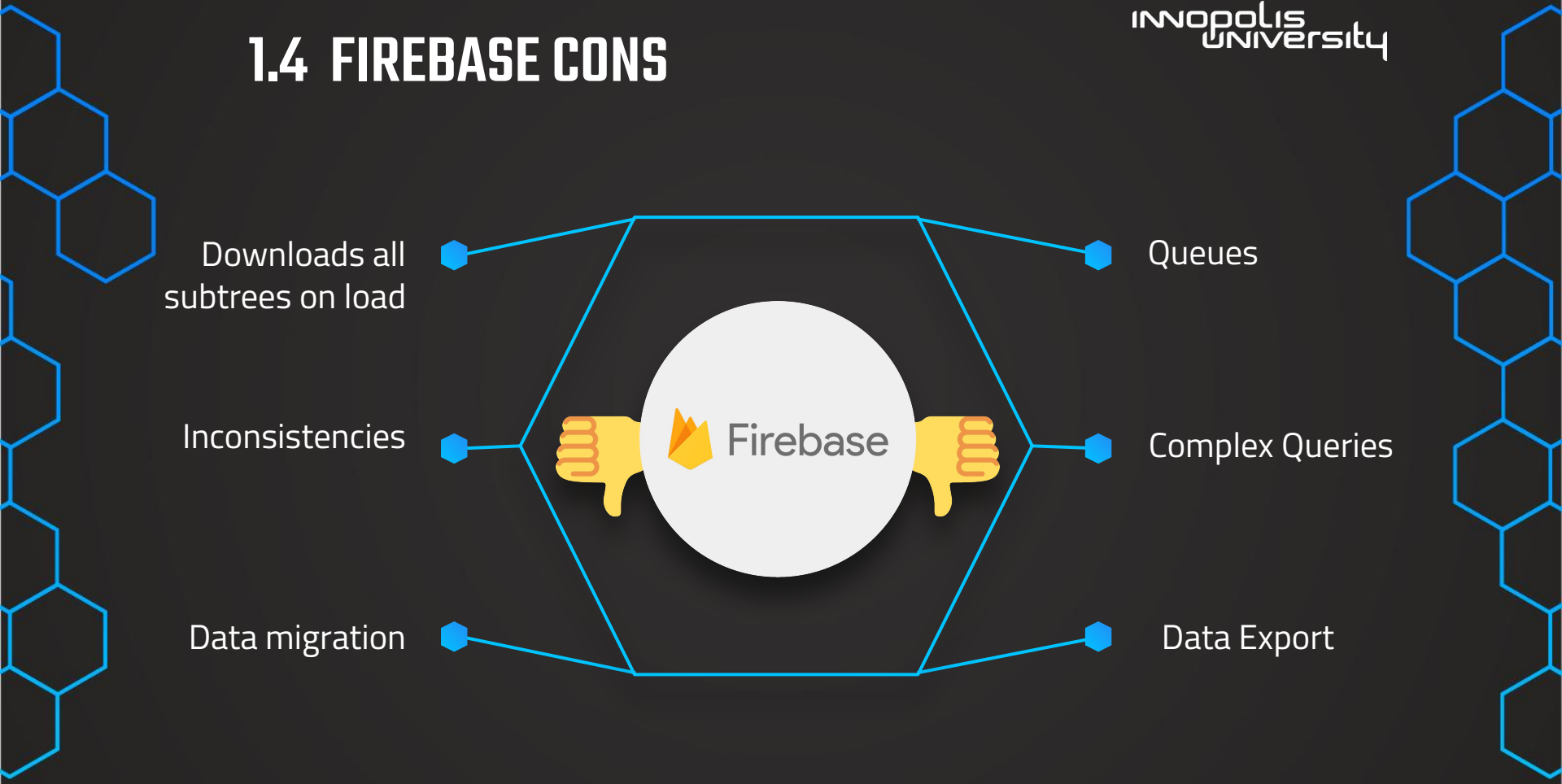
1.3 FIREBASE PROS

Innopolis University

- Authentication
- Easy Hosting
- Ability to Messaging
- Offers Analytics
- Storage facility
- Crash reporting

1.4 FIREBASE CONS

innopolis university



innopolis university

02

Installation

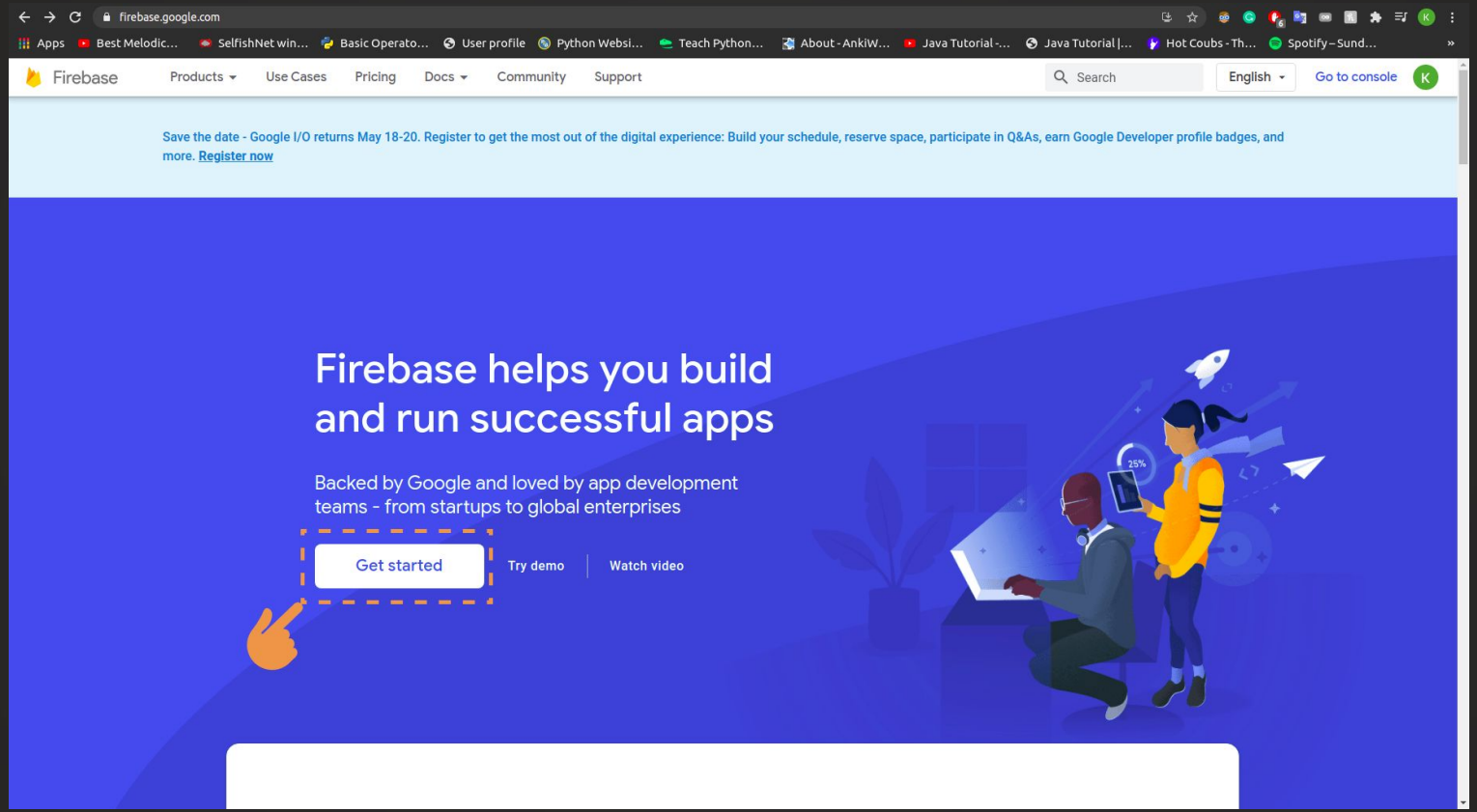
How to install it and how it operates in general.

2.1 Environment Preparation

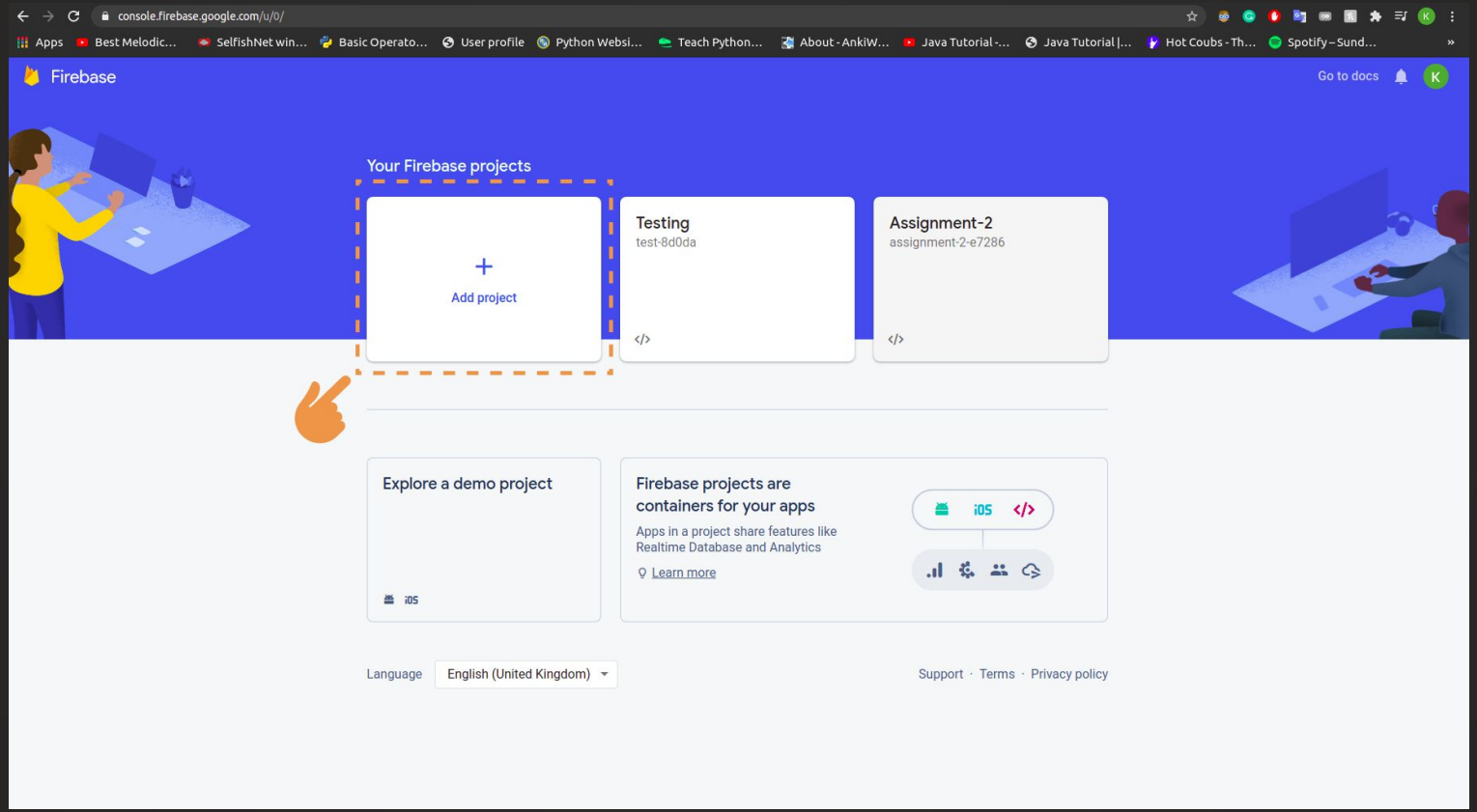
As talked about in the introduction, Firebase is a **cloud-hosted NoSQL database** and it's **GUI is a web based dashboard/portal** hence, as an administrator, you don't need to install anything locally on your machine to be able to access this tool. In this part, I'll take you through a series of steps to prepare your environment.

Step 1

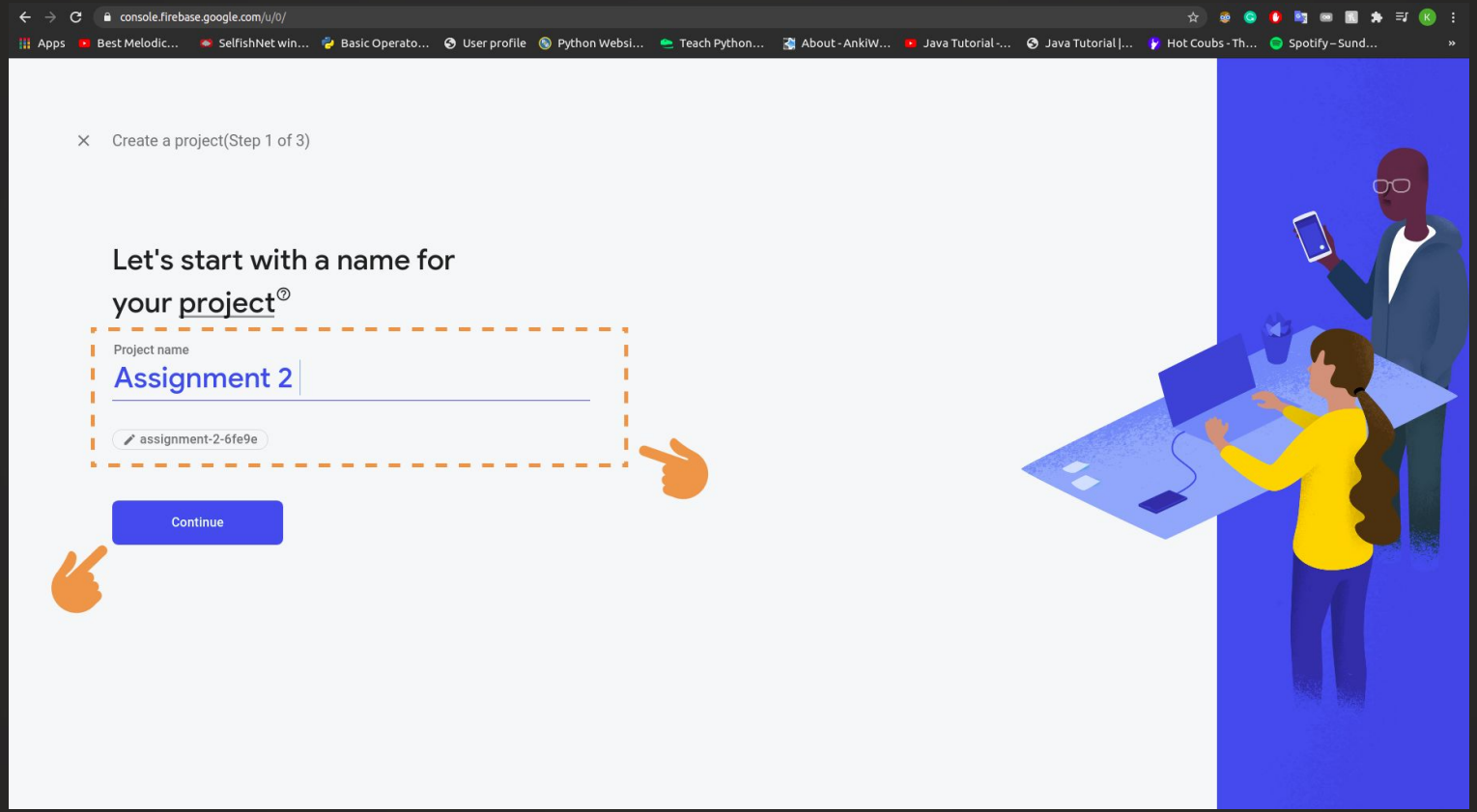
innopolis university



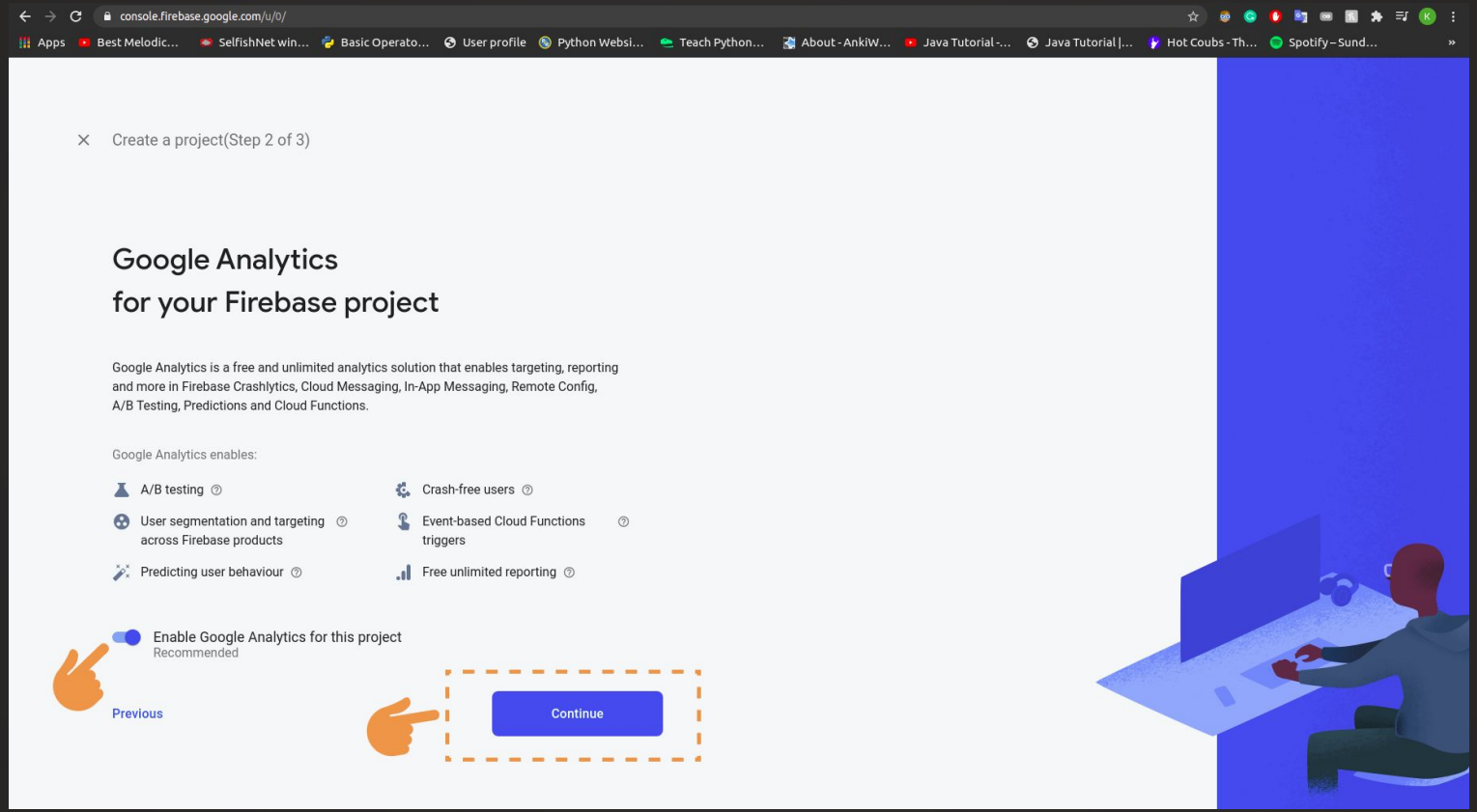
Step 2


console.firebase.google.com/u/0/
Apps
Best Melodic...
SelfishNet win...
Basic Operato...
User profile
Python Websi...
Teach Python...
About - AnkiW...
Java Tutorial -...
Java Tutorial |...
Hot Coubs - Th...
Spotify – Sund...
Firebase
Go to docs
K
Your Firebase projects
Add project
Testing
test-8d0da
Assignment-2
assignment-2-e7286
Explore a demo project
iOS
Firebase projects are containers for your apps
Apps in a project share features like Realtime Database and Analytics
Learn more
Language
English (United Kingdom)
Support · Terms · Privacy policy

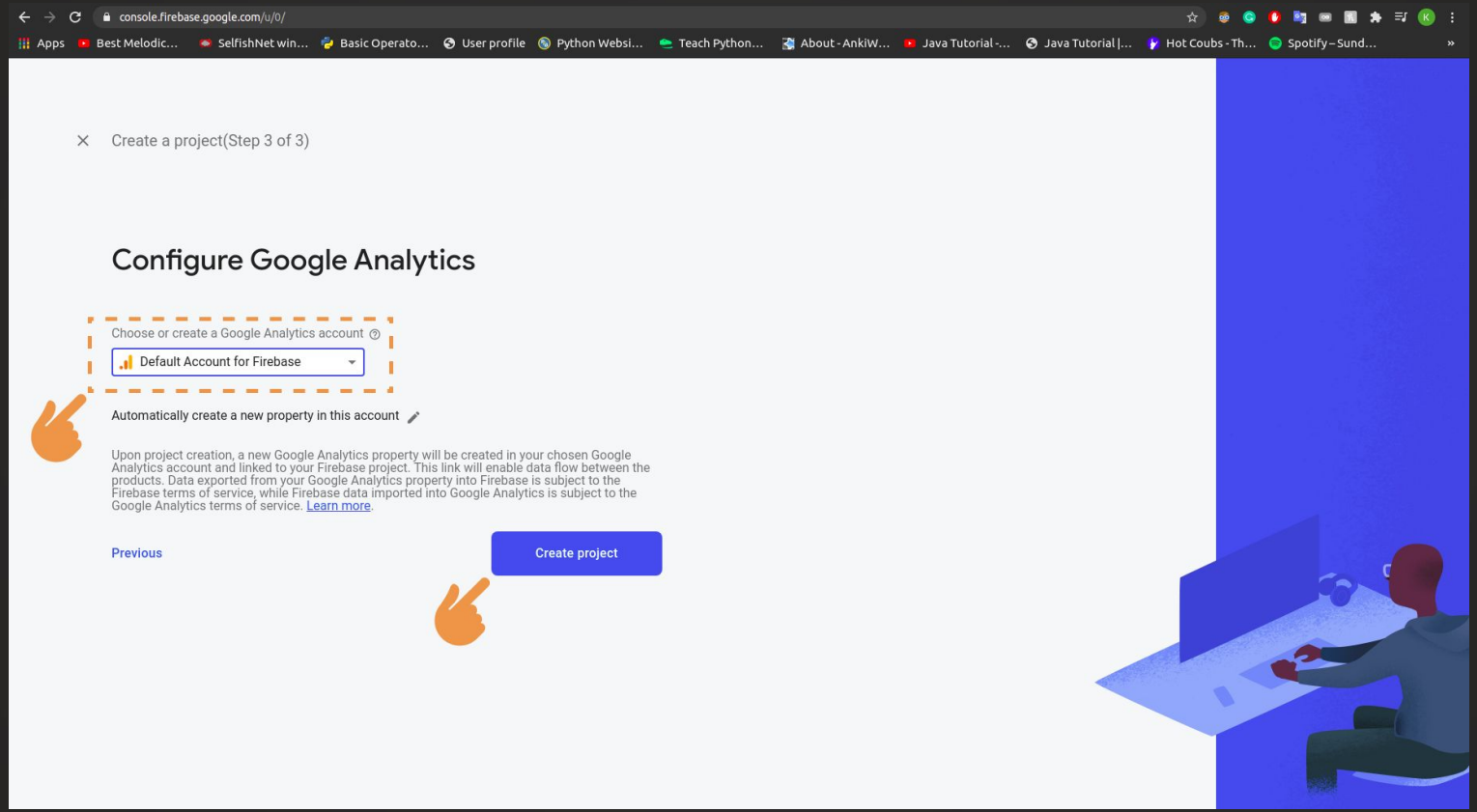
Step 3


console.firebase.google.com/u/0/
Apps
Best Melodic...
SelfishNet win...
Basic Operato...
User profile
Python Websi...
Teach Python...
About - AnkiW...
Java Tutorial -...
Java Tutorial |...
Hot Coubs - Th...
Spotify – Sund...
Create a project(Step 1 of 3)
Let's start with a name for your project
Project name
Assignment 2
assignment-2-6fe9e
Continue

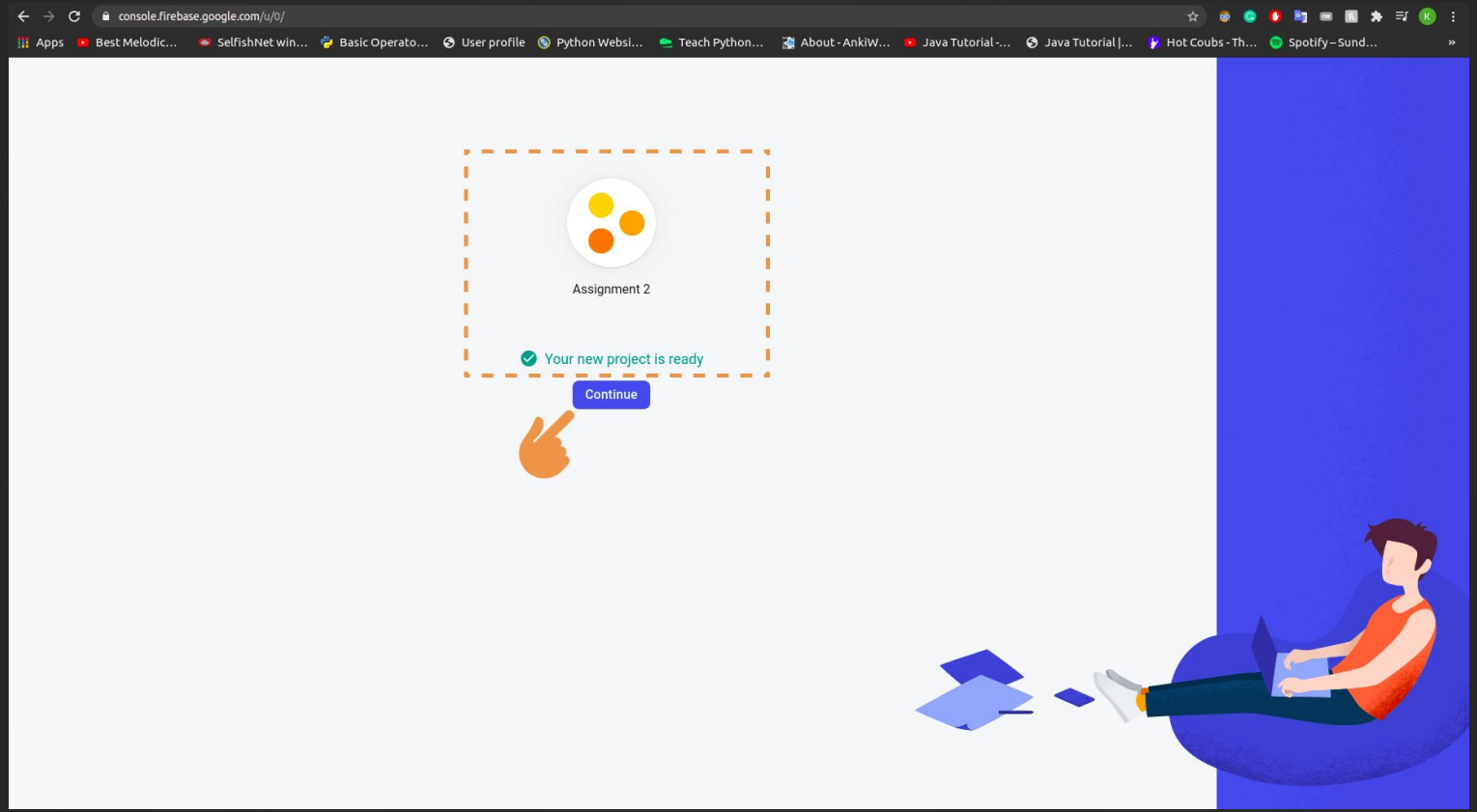
Step 4


console.firebase.google.com/u/0/
Apps
Best Melodic...
SelfishNet win...
Basic Operato...
User profile
Python Websi...
Teach Python...
About - AnkiW...
Java Tutorial -...
Java Tutorial |...
Hot Coubs - Th...
Spotify – Sund...
Create a project(Step 2 of 3)
Google Analytics
for your Firebase project
Google Analytics is a free and unlimited analytics solution that enables targeting, reporting and more in Firebase Crashlytics, Cloud Messaging, In-App Messaging, Remote Config, A/B Testing, Predictions and Cloud Functions.
Google Analytics enables:
A/B testing
Crash-free users
User segmentation and targeting across Firebase products
Event-based Cloud Functions triggers
Predicting user behaviour
Free unlimited reporting
Enable Google Analytics for this project
Recommended
Previous
Continue

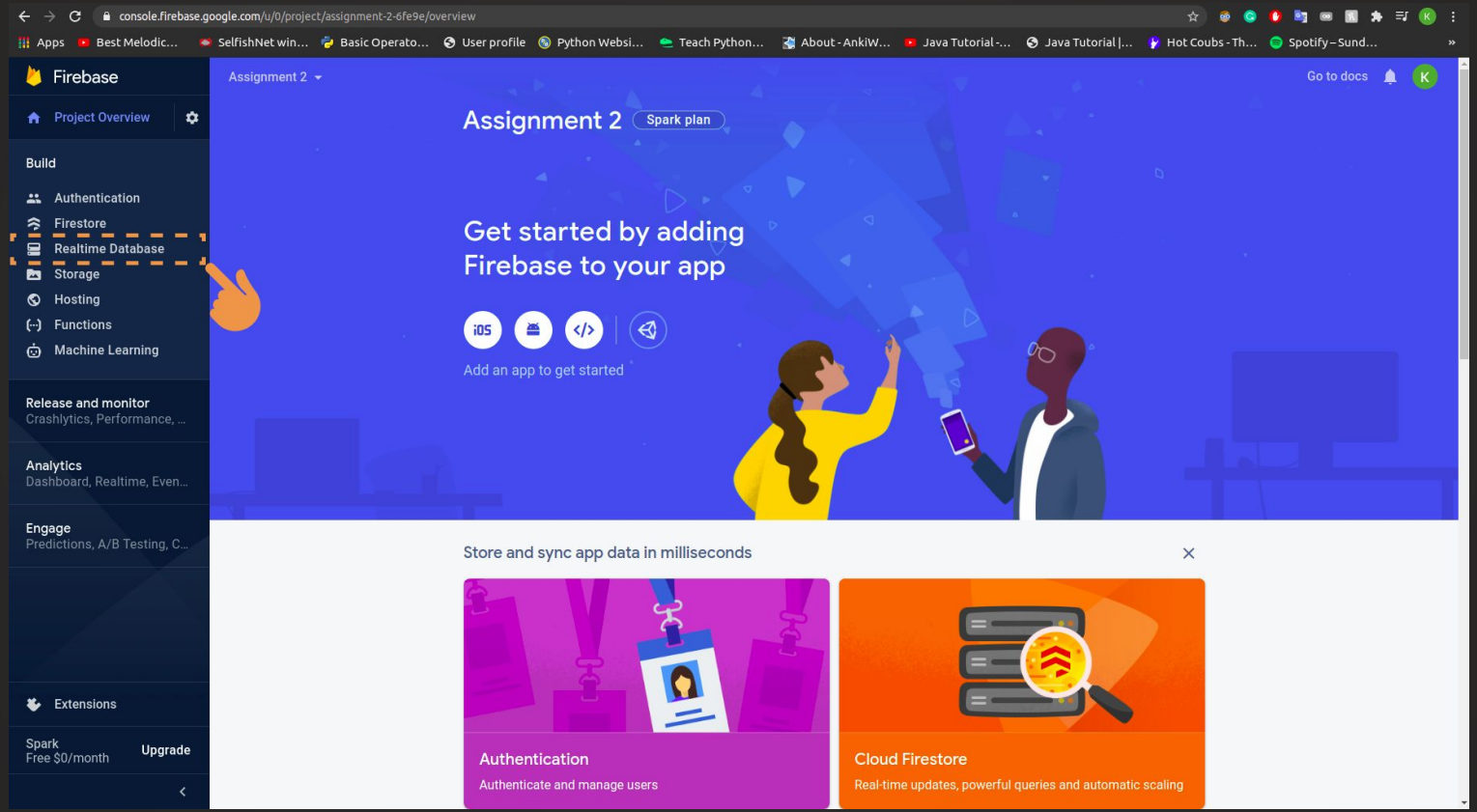
Step 5


console.firebase.google.com/u/0/
Apps
Best Melodic...
SelfishNet win...
Basic Operato...
User profile
Python Websi...
Teach Python...
About - AnkiW...
Java Tutorial -...
Java Tutorial |...
Hot Coubs - Th...
Spotify – Sund...
Create a project(Step 3 of 3)
Configure Google Analytics
Choose or create a Google Analytics account
Default Account for Firebase
Automatically create a new property in this account
Upon project creation, a new Google Analytics property will be created in your chosen Google Analytics account and linked to your Firebase project. This link will enable data flow between the products. Data exported from your Google Analytics property into Firebase is subject to the Firebase terms of service, while Firebase data imported into Google Analytics is subject to the Google Analytics terms of service. Learn more.
Previous
Create project

Step 6

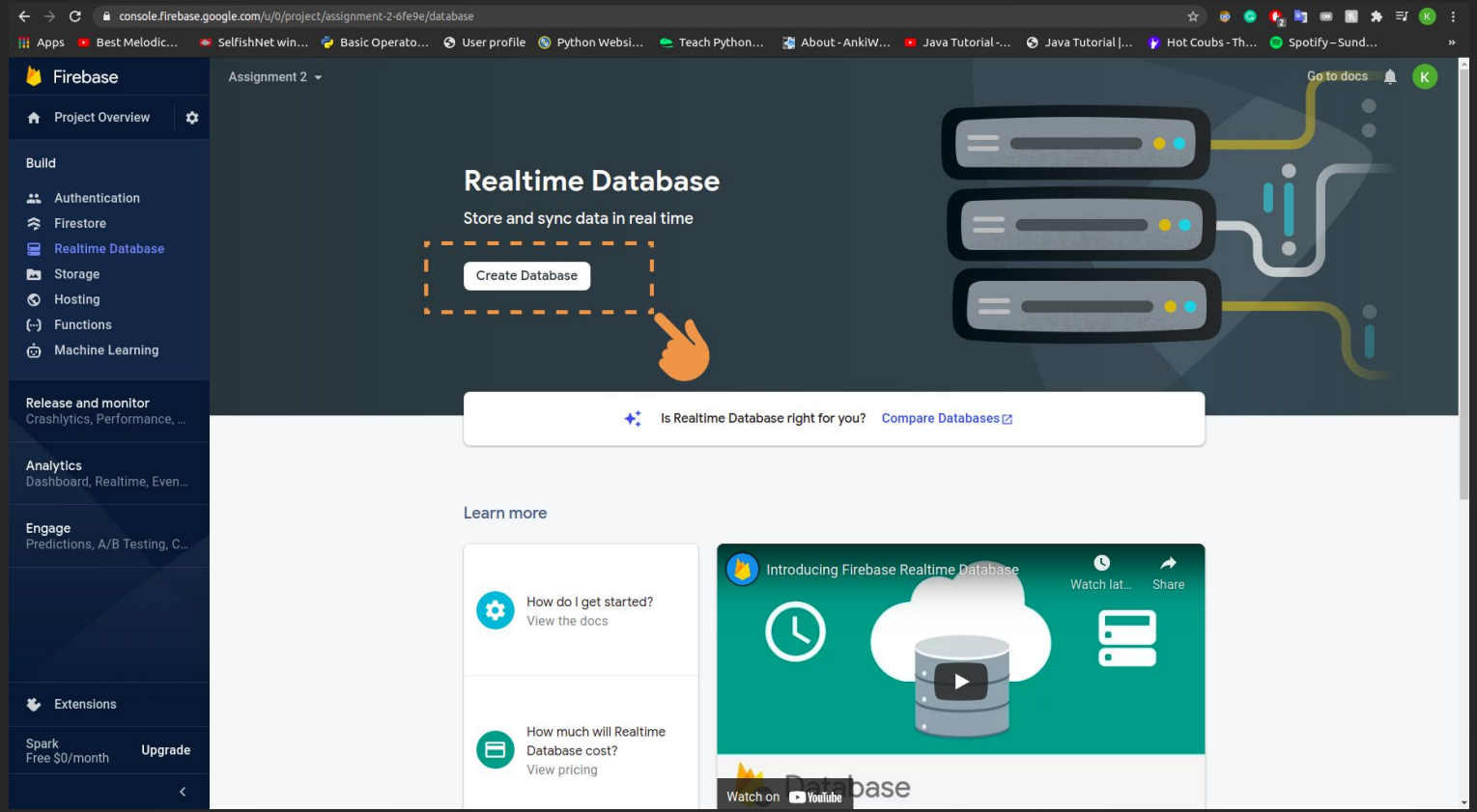

console.firebase.google.com/u/0/
Apps
Best Melodic...
SelfishNet win...
Basic Operato...
User profile
Python Websi...
Teach Python...
About - AnkiW...
Java Tutorial -...
Java Tutorial |...
Hot Coubs - Th...
Spotify – Sund...
Assignment 2
Your new project is ready
Continue

Step 7

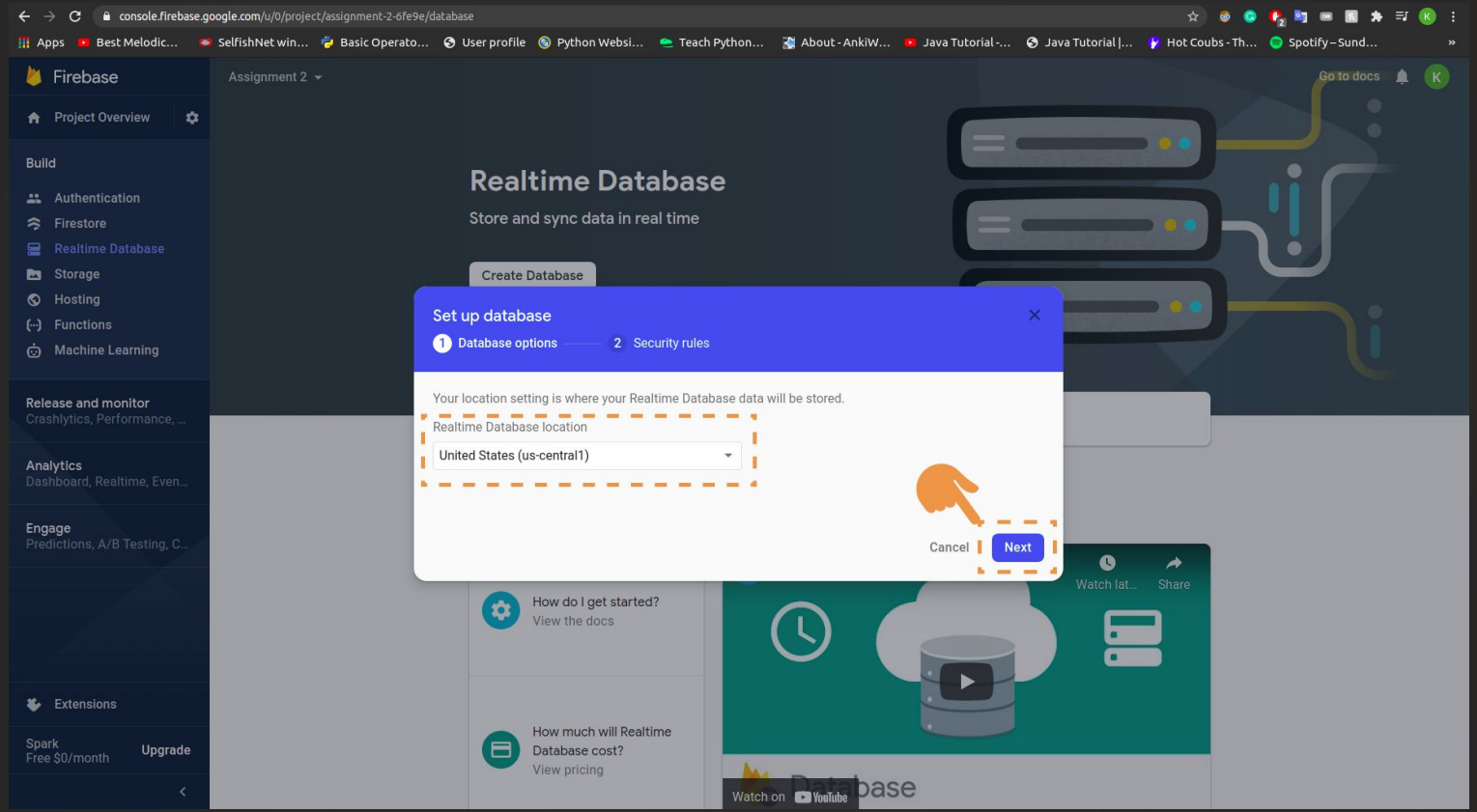

console.firebase.google.com/u/0/project/assignment-2-6fe9e/overview
Apps
Best Melodic...
SelfishNet win...
Basic Operato...
User profile
Python Websi...
Teach Python...
About - AnkiW...
Java Tutorial -...
Java Tutorial |...
Hot Coubs - Th...
Spotify – Sund...
Firebase
Assignment 2
Project Overview
Build
Authentication
Firestore
Realtime Database
Storage
Hosting
Functions
Machine Learning
Release and monitor
Crashlytics, Performance, ...
Analytics
Dashboard, Realtime, Even...
Engage
Predictions, A/B Testing, C...
Extensions
Spark
Free $0/month
Upgrade
Assignment 2
Spark plan
Go to docs
Get started by adding
Firebase to your app
iOS
Add an app to get started
Store and sync app data in milliseconds
Authentication
Authenticate and manage users
Cloud Firestore
Real-time updates, powerful queries and automatic scaling

Step 8

innopolis university

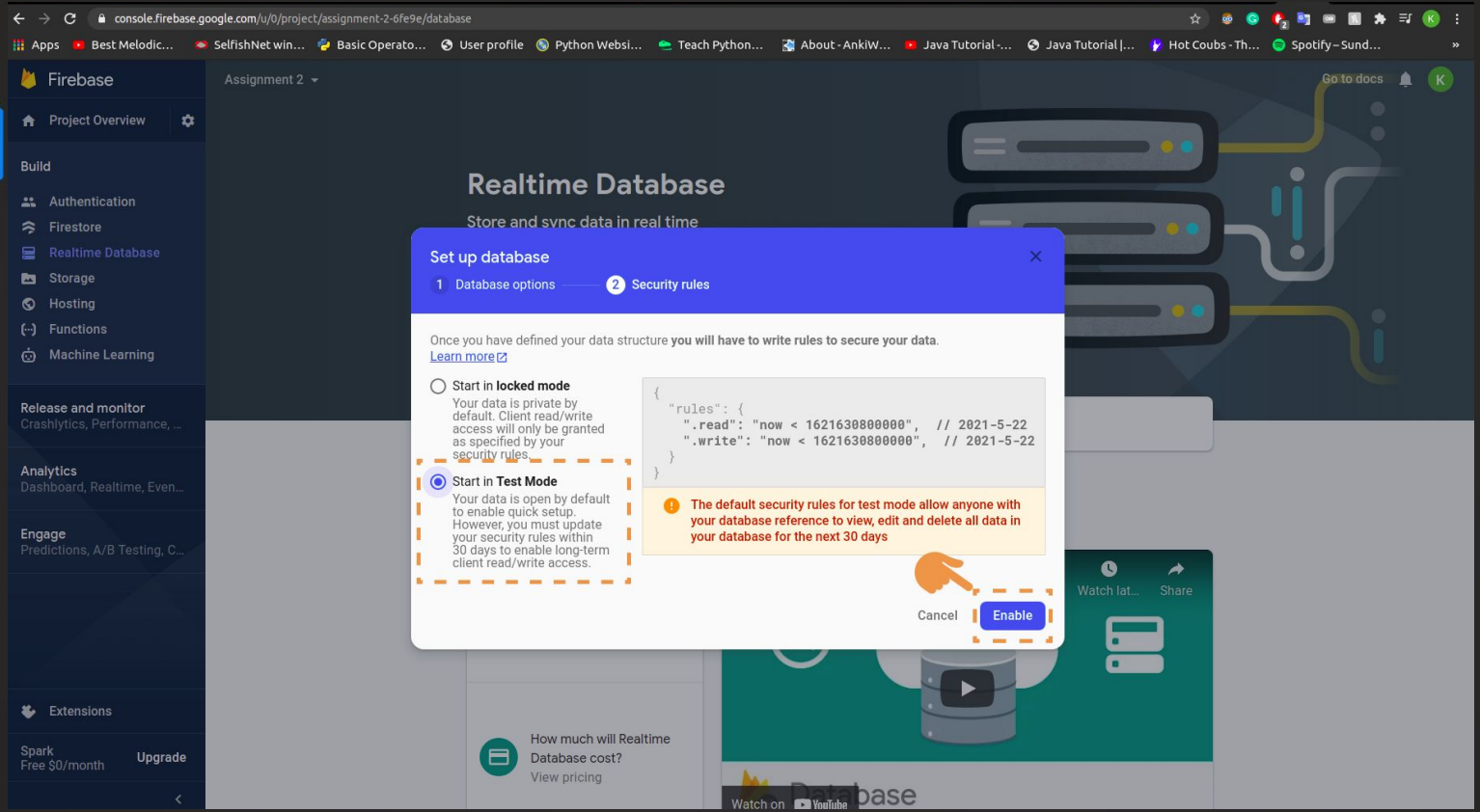


Step 9

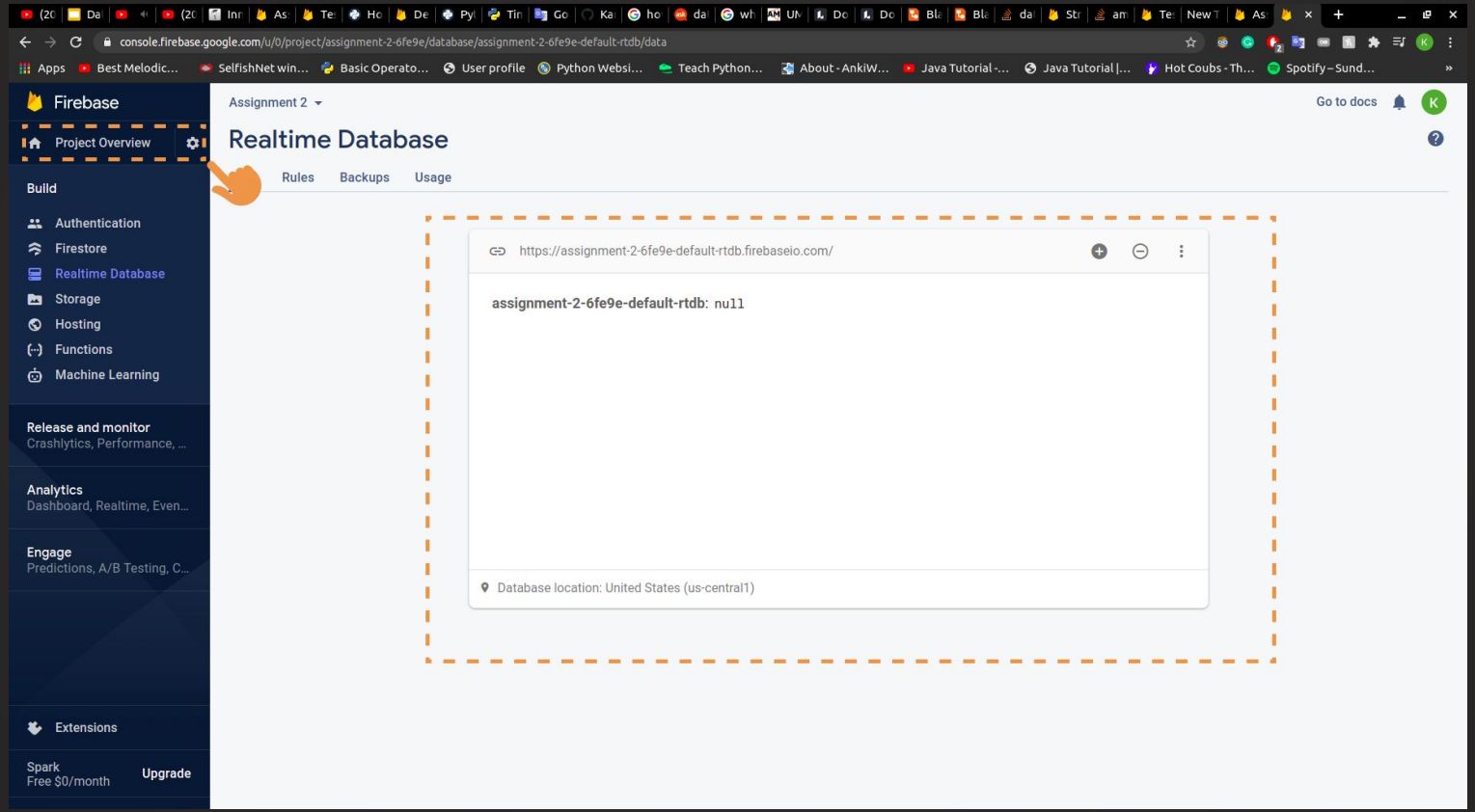

console.firebase.google.com/u/0/project/assignment-2-6fe9e/database
Firebase
Assignment 2
Go to docs
Project Overview
Build
Authentication
Firestore
Realtime Database
Storage
Hosting
Functions
Machine Learning
Release and monitor
Crashlytics, Performance, ...
Analytics
Dashboard, Realtime, Even...
Engage
Predictions, A/B Testing, C...
Extensions
Spark
Free $0/month
Upgrade
Realtime Database
Store and sync data in real time
Create Database
Set up database
1 Database options
2 Security rules
Your location setting is where your Realtime Database data will be stored.
Realtime Database location
United States (us-central1)
Cancel
Next
How do I get started?
View the docs
How much will Realtime Database cost?
View pricing
Watch lat...
Share
Watch on YouTube

Step 10

innopolis university

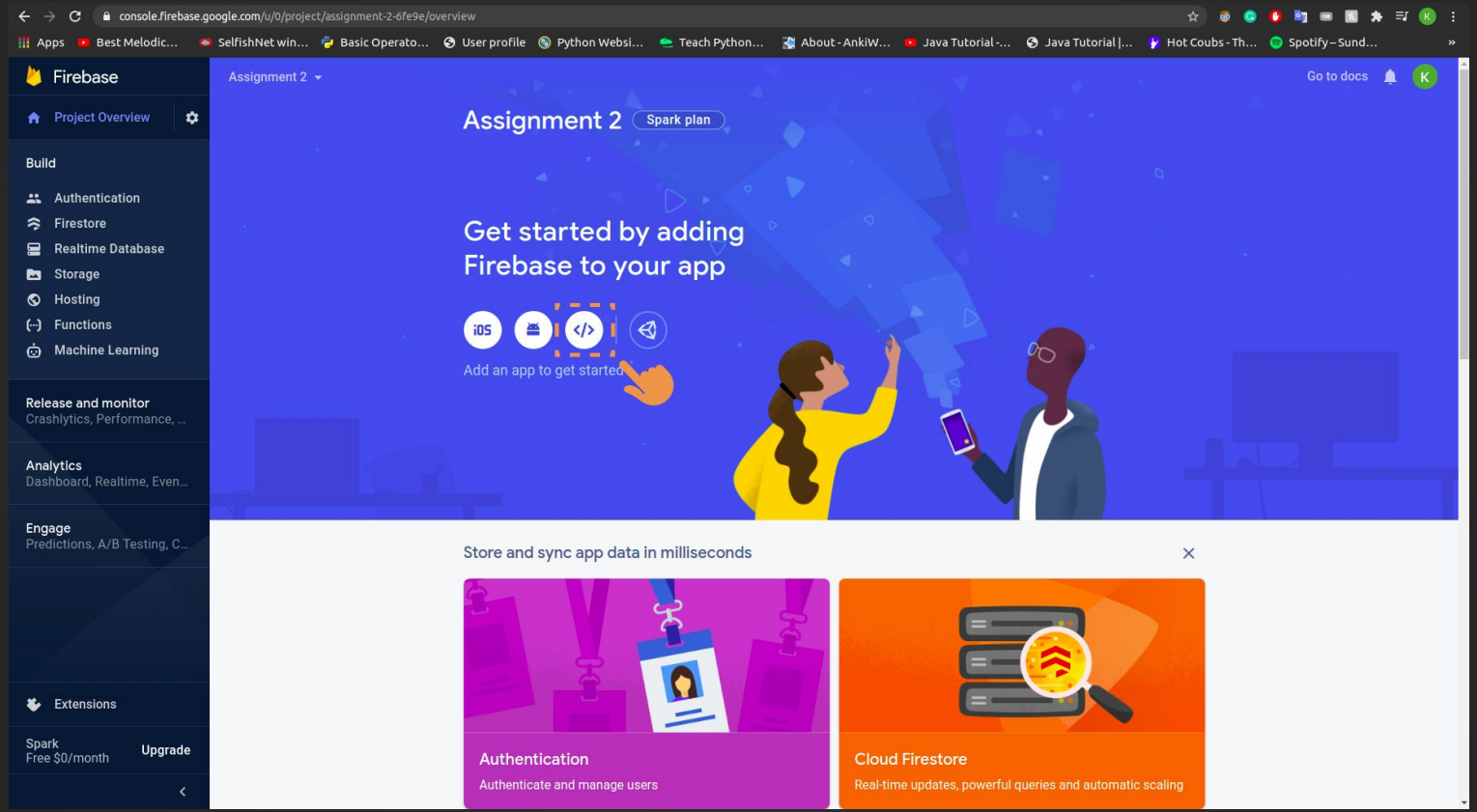


Step 11

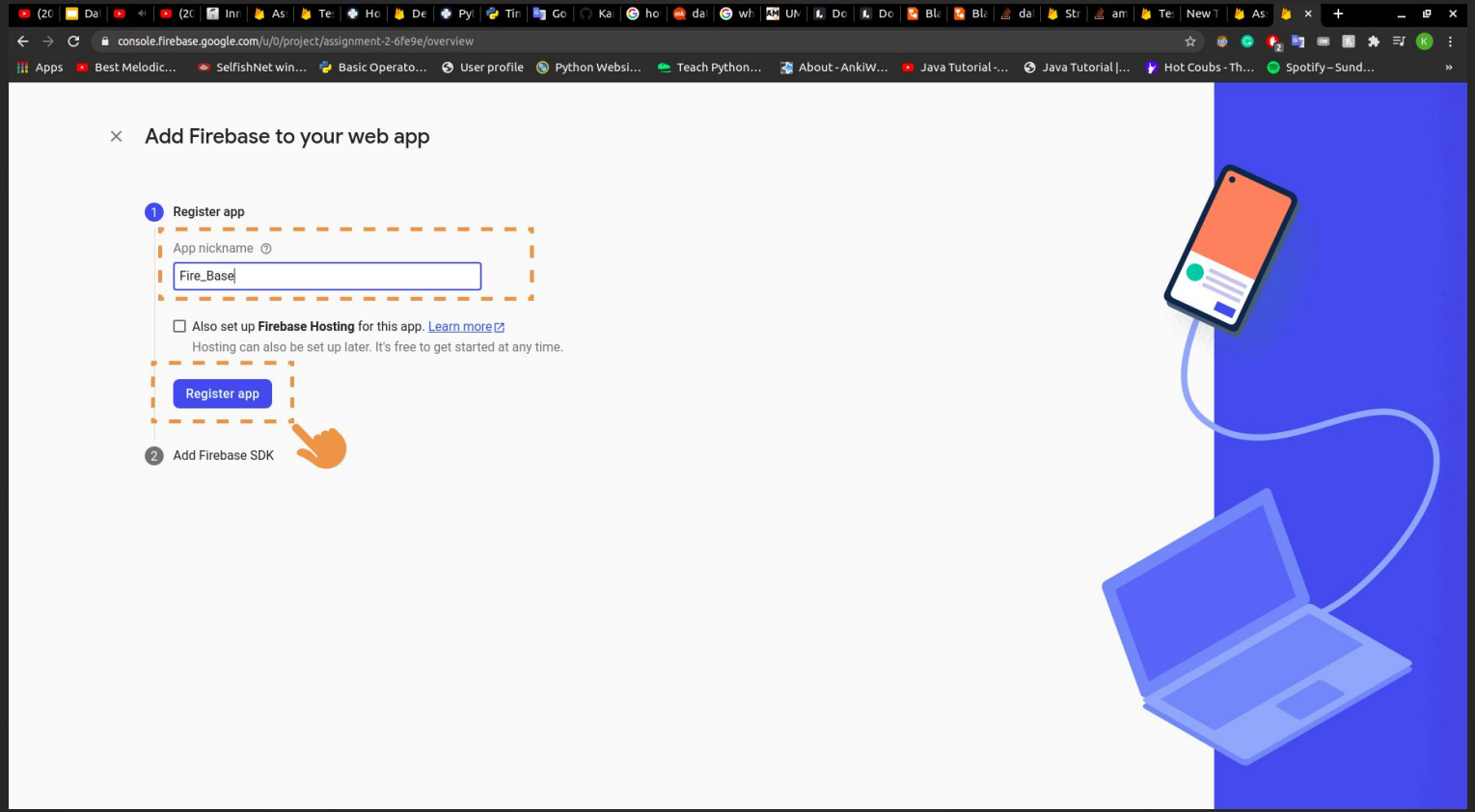

console.firebase.google.com/u/0/project/assignment-2-6fe9e/database/assignment-2-6fe9e-default-rtdb/data
Firebase
Project Overview
Build
Authentication
Firestore
Realtime Database
Storage
Hosting
Functions
Machine Learning
Release and monitor
Crashlytics, Performance, ...
Analytics
Dashboard, Realtime, Even...
Engage
Predictions, A/B Testing, C...
Extensions
Spark
Free $0/month
Upgrade
Assignment 2
Go to docs
Realtime Database
Rules
Backups
Usage
https://assignment-2-6fe9e-default-rtdb.firebaseio.com/
assignment-2-6fe9e-default-rtdb: null
Database location: United States (us-central1)

Step 13

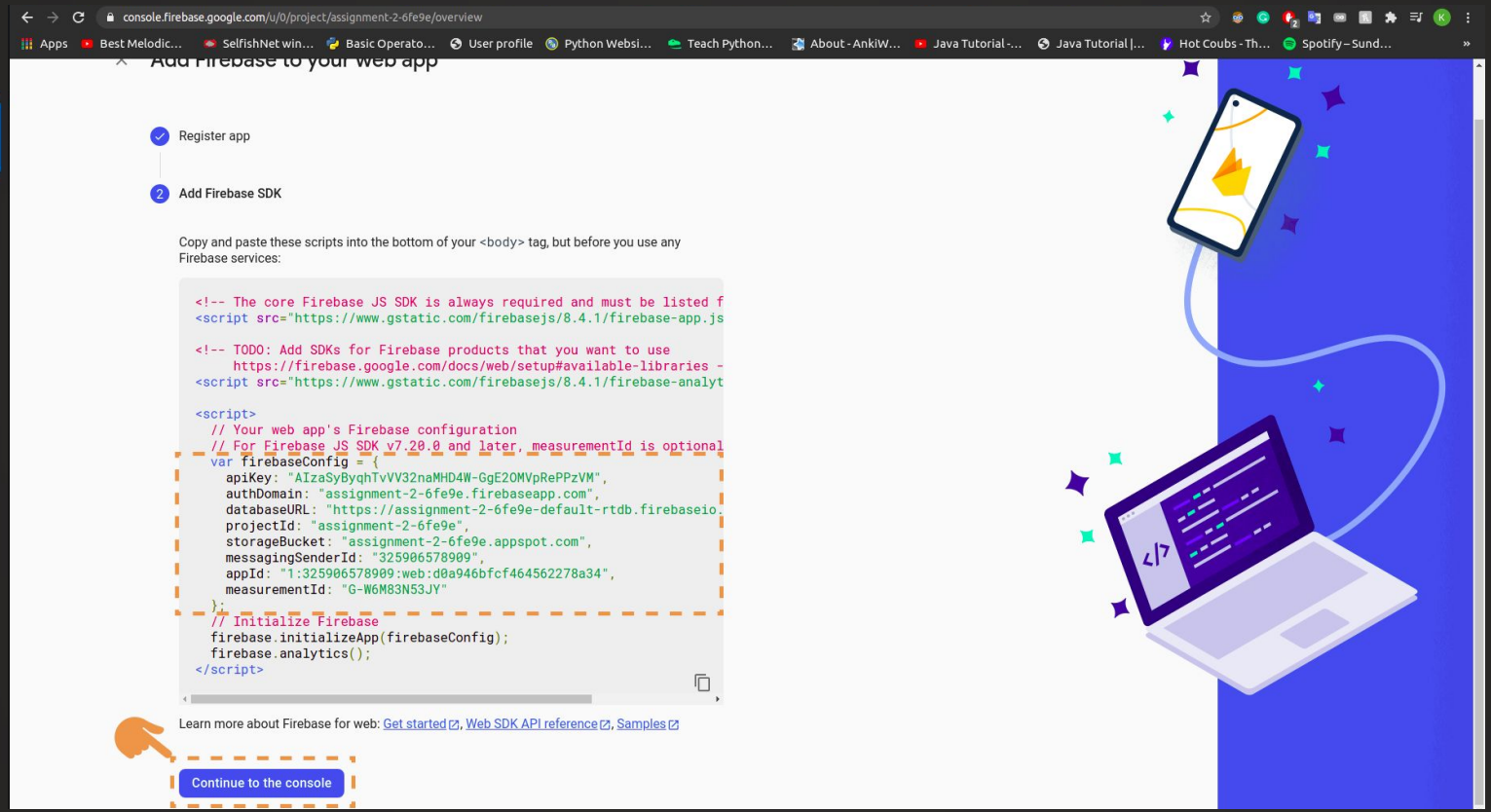
innopolis university



Step 14


console.firebase.google.com/u/0/project/assignment-2-6fe9e/overview
Add Firebase to your web app
1 Register app
App nickname
Fire_Base
Also set up Firebase Hosting for this app. Learn more
Hosting can also be set up later. It's free to get started at any time.
Register app
2 Add Firebase SDK

Step 15


console.firebase.google.com/u/0/project/assignment-2-6fe9e/overview
Add Firebase to your web app
Register app
2 Add Firebase SDK
Copy and paste these scripts into the bottom of your <body> tag, but before you use any Firebase services:
<!-- The core Firebase JS SDK is always required and must be listed f
<script src="https://www.gstatic.com/firebasejs/8.4.1/firebase-app.js
<!-- TODO: Add SDKs for Firebase products that you want to use
https://firebase.google.com/docs/web/setup#available-libraries -
<script src="https://www.gstatic.com/firebasejs/8.4.1/firebase-analyt
<script>
// Your web app's Firebase configuration
// For Firebase JS SDK v7.20.0 and later, measurementId is optional
var firebaseConfig = {
apiKey: "AIzaSyByqhTvVV32naMHD4W-GgE2OMVpRePPzVM",
authDomain: "assignment-2-6fe9e.firebaseapp.com",
databaseURL: "https://assignment-2-6fe9e-default-rtdb.firebaseio.
projectId: "assignment-2-6fe9e",
storageBucket: "assignment-2-6fe9e.appspot.com",
messagingSenderId: "325906578909",
appId: "1:325906578909:web:d0a946bfcf464562278a34",
measurementId: "G-W6M83N53JY"
};
// Initialize Firebase
firebase.initializeApp(firebaseConfig);
firebase.analytics();
</script>
Learn more about Firebase for web: Get started, Web SDK API reference, Samples
Continue to the console

2.2 Connecting with Python

innopolis university

"import pyrebase"

```
43    #connecting to FireBase DataBase
44    firebaseConfig = {
45        'apiKey': "AIzaSyAfSnRHC913I0_zR5LdUZCk-xZs_YFio54",
46        'authDomain': "assignment-2-e7286.firebaseapp.com",
47        'databaseURL': "https://assignment-2-e7286-default-rtdb.firebaseio.com",
48        'projectId': "assignment-2-e7286",
49        'storageBucket': "assignment-2-e7286.appspot.com",
50        'messagingSenderId': "926718650475",
51        'appId': "1:926718650475:web:14adeb6a5d847e4594292e",
52        'measurementId': "G-GWLPMN10JZ"}
```

Innopolis University

03

Migration

Scripts, components and bulking time (UML Diagram included)

3. MIGRATION APPROACHES

Throughout this assignment, we found out that there are two approaches of migrating data from Postgresql to Google Firebase. Those two approaches are:

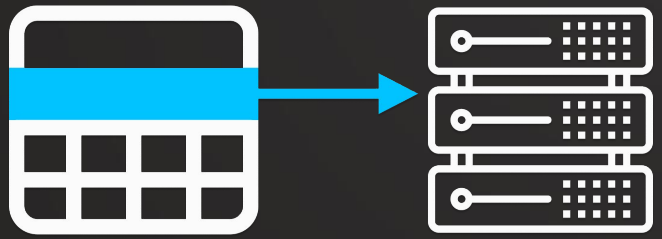
1. Naive approach

2. Optimized approach

3.1 NAIVE APPROACH

Installing each component (the tables in this example) from the original Database on PGadmin and uploading each row from each table in the original DB to the new one. Any entry to a new table will automatically create a new table in FireBase.

Uploading row by row



Bulking time + installation time

40 min

3.2 OPTIMIZED APPROACH

First, we install all the data from PGadmin. Each row will be represented as a dictionary that contains names of the columns and the matching data. Then all these rows will be added together forming a table also represented as a dictionary with the row index and its data. Finally, the complete table will be represented as a dictionary containing the table names and all their columns.

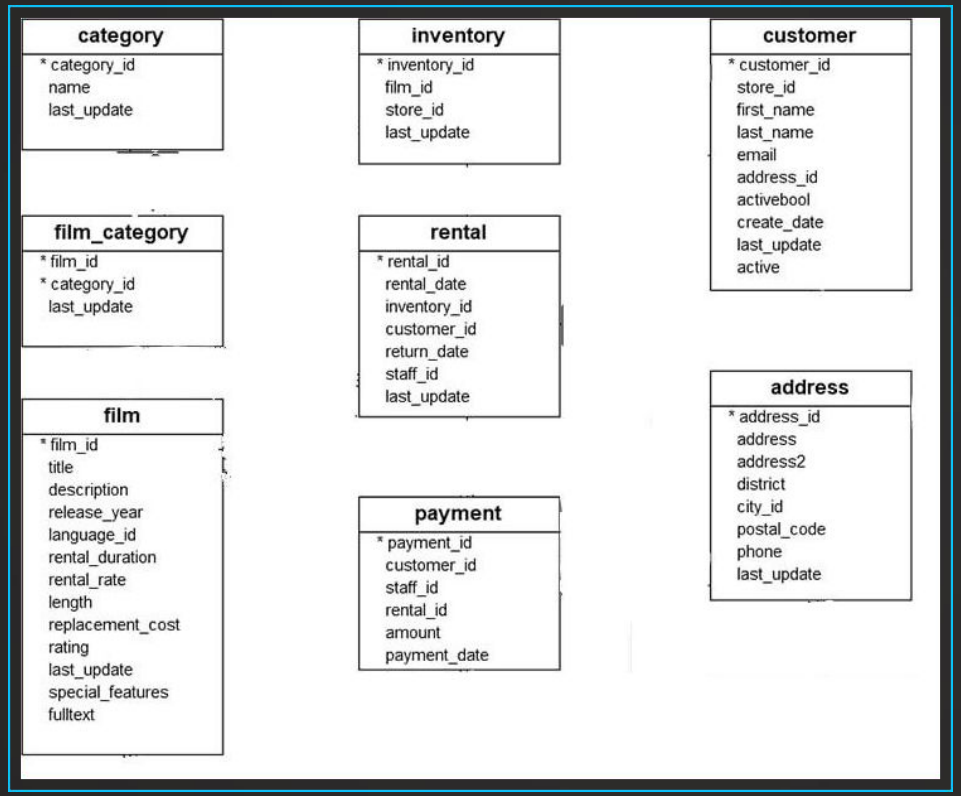
Uploading DB as a dictionary

```
DataBases{
 table  : {
index : {
column : data }}}
```

Bulking time + installation time

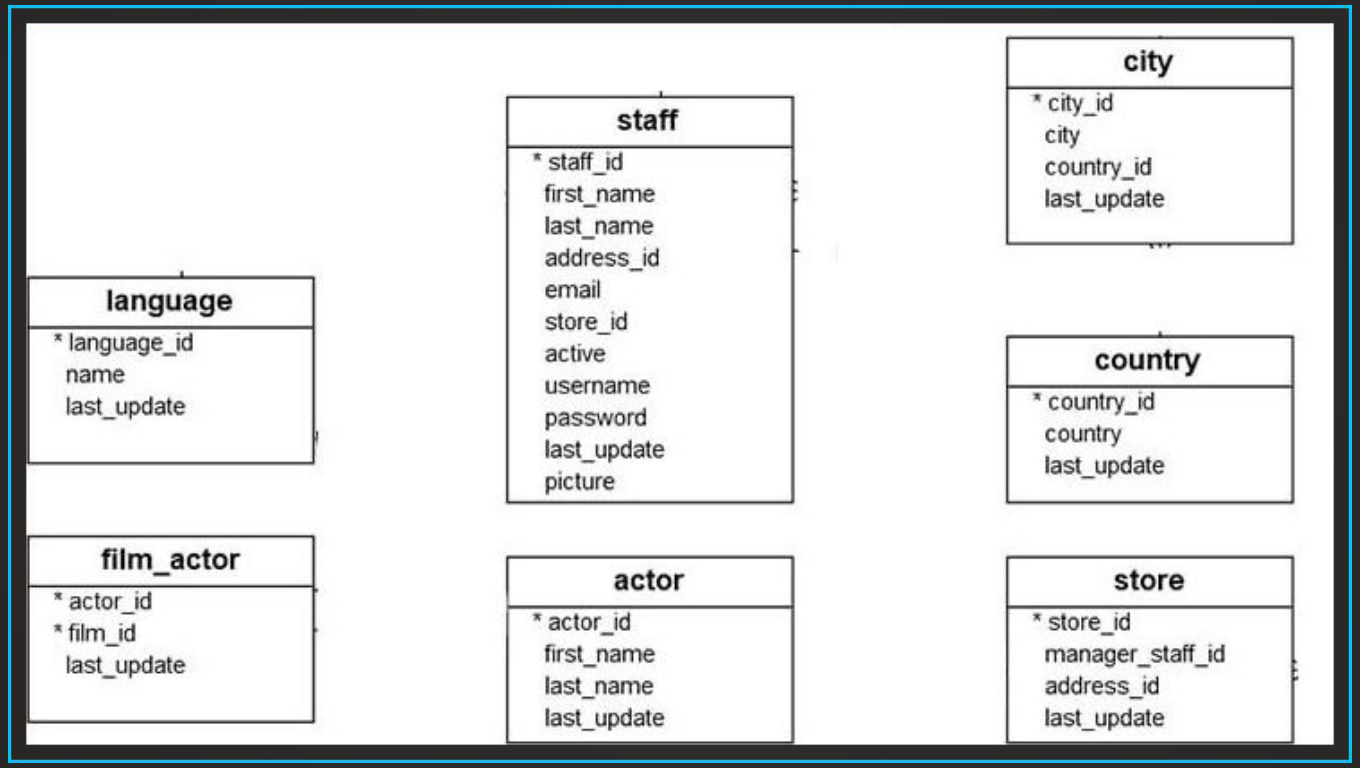
6.5 sec

3.3 UML DIAGRAM (1)



UML Diagram 1 of DVD Rental Tables in Google firebase

3.3 UML DIAGRAM (2)



UML Diagram 2 of DVD Rental Tables in Google firebase

innopolis university

04

Automatization

How migration can be automatized continuously

4.1 Automatization in Firebase (1)

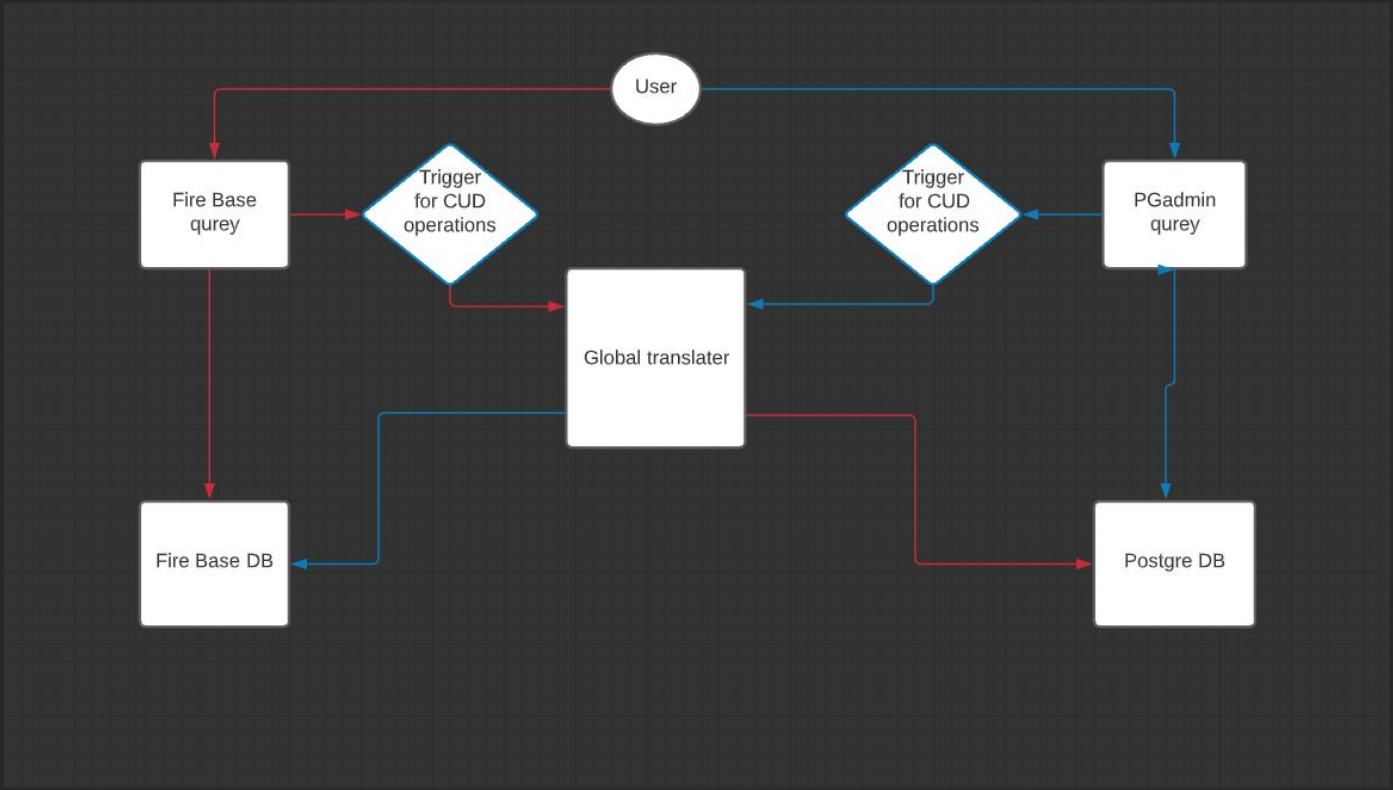
To have a real automatized migration, all the data should be completely synced with the data in PGadmin in real-time.

A user can either query to the FireBase or PGadmin. For the sake of presentation, we will check the case where a user is performing a **deleting operation** in PGadmin. As soon as the query runs, a trigger will rise. This trigger will send the query to a Global translator which will translate the Postgres query to a firebase query that will be implemented to the Firebase too.

This applies to any **Create**, **Update**, and **Delete** query.

And vice versa from any Database to another.

4.1 Automatization in Firebase (2)



Automatization process in Firebase

innopolis university

05

Queries

Comparing queries and performance

5.1 Query 2 (From Assignment)

Let's start simple...

Insertion, Update, Deletion

5.1.1 Insertion query (1)

Insertion Query format in Firebase:

```
db.reference(PATH).set(VALUE)
```

5.1.1 Insertion query (2)

SQL  0.044s

```sql
INSERT INTO actor(actor_id, first_name, last_name, last_update)
VALUES (1000, 'John', 'Smith', '2013-05-26 14:47:57.620000');
```

5.1.1 Insertion format (3)

Firebase Realtime DB

0.849s

```
def insert_actor_with_id(actor_id,first_name, last_name, last_update):
    # Preparing the new entry
    entry = {
        'actor_id': actor_id,  # Assuming the IDs are 1-indexed
        'first_name' : first_name,
        'last_name' : last_name,
        'last_update' : last_update
            }

    # The insert operation
    db.reference('actor').child(str(actor_id)).set(entry)
```

5.1.1 Insertion format (4)

Firebase Realtime DB (with auto ID)

```
def insert_actor(first_name, last_name, last_update):
    # This is for having a manual id
    all_actors = dict(db.reference("actor").get())
    size = len(all_actors)

    # Preparing the new entry
    entry = {
        'actor_id': size + 1,  # Assuming the IDs are 1-indexed
        'first_name' : first_name,
        'last_name' : last_name,
        'last_update' : last_update
            }

    # The insert operation
    db.reference('actor').child(str(size+1)).set(entry)
```

5.1.2 Selection query

Selection Query format in Firebase:

```
db.reference(PATH)
.order_function()
.criteria_function()
```

5.1.3 Update query(1)

SQL  0.055s

```
UPDATE staff
SET store_id = 2
WHERE store_id = 1;
```

5.1.3 Update query(2)

Firebase Realtime DB  0.472s

```
def update_store(old_store_id,new_store_id):
    print(old_store_id)
    # Getting all staff members that work in the store with id 'old_store_id'
    staff = db.reference('staff').order_by_child('store_id').equal_to(old_store_id).get()

    # Converting to the appropriate format
    staff = dict(staff)
    staff = list(staff.values())

    # Updating them one by one
    for x in staff:
        print(x)
        db.reference('staff').child(str(x['staff_id'])).child('store_id').set(new_store_id)
```

5.1.3 Delete query(1)

SQL  0.052s

```sql
DELETE FROM store
WHERE store_id = 1;
```

5.1.3 Delete query(2)

Firebase Realtime DB  0.230s

```
def delete_store(store_id):
    # Simple and easy
    db.reference('store').child(str(store_id)).delete()
```

Live demo

5.2 Query 1 (From Assignment)

SQL  33.701s

```
SELECT *,
  (SELECT COUNT(*) FROM rental r2, payment p2 WHERE r2.rental_id= p2.rental_id AND p2.amount<p.amount)
  AS count_smaller_pay
FROM rental r, payment p
WHERE r.rental_id = p.rental_id;
```

5.2 Query 1 (From Assignment)

Aggregate function

```
SELECT *,
  (SELECT COUNT(*) FROM rental r2, payment p2 WHERE r2.rental_id= p2.rental_id AND p2.amount<p.amount)
  AS count_smaller_pay
FROM rental r, payment p
WHERE r.rental_id = p.rental_id;
```

Join

5.2 Query 1 (From Assignment)

The Firebase RealTime Database doesn’t support either of them.

We need to do the **operations manually**.

5.2 Query 1 (From Assignment)

innopolis university

The slowest join ever 

```python
def query_1_join_operation_slow():
    payments = list(db.reference('payment').get())

    res = []

    #The slow part : getting the rental information directly from the database
    for i in payments:
        rental = list(dict(db.reference('rental').order_by_child('rental_id').equal_to(i['rental_id']).get()).values())
        tmp = i
        # In case there is no corresponding rental
        if len(rental) == 0:
            continue

        # We are assuming that there is only one rental for each payment
        for j in rental[0]:
            tmp[j] = rental[0][j]
        res.append(tmp)
    return res
```

5.2 Query 1 (From Assignment)

Innopolis University

The fastest join

```
def query_1_local():
    # Acquiring all tables
    payments = list(db.reference('payment').order_by_child('rental_id').get())
    rentals = list(db.reference('rental').order_by_child('rental_id').get())

    # The tables are sorted, so we can use 2 pointers to match the rental_id
    p_i = 0
    joint_table = []
    for i in rentals:
        while payments[p_i]['rental_id'] < i['rental_id']:
            p_i += 1
        if payments[p_i]['rental_id'] == i['rental_id']:
            tmp = payments[p_i]
            for j in i:
                tmp[j] = i[j]
            joint_table.append(tmp)
```

5.2 Query 1 (From Assignment)

Innopolis University

Doing Aggregation

3.725s

```
# Making the aggregation
payments_by_amount = list(db.reference('payment').order_by_child('amount').get())
for i in joint_table:

    # Binary search the answer
    low = 0
    high = len(payments_by_amount)
    while low < high:
        if low < 0:
            low = -1
            break
        mid = (low + high) // 2
        if payments_by_amount[mid]['amount'] < i['amount']:
            low = mid + 1
        else:
            high = mid
    i['count_smaller_pay'] = low

return joint_table
```

5.2 Query 1 (From Assignment)

Innopolis University

Using denormalization

```
def query_1_with_extra_table():
    table = db.reference('payment_rental_smaller_pay').get()
    if table is None:
        print("The table doesn't exist. Going to compute the table content...")
        new_table = query_1_local()
        db.reference('payment_rental_smaller_pay').set(new_table)
        return new_table
    return table
```

5.2 Query 1 (From Assignment)

Innopolis University

Using denormalization (with listeners)

```
db.reference('payment').listen(cb)
```

First time  5.694s

Subsequent times  1.930s

innopolis university

06

Time for Questions

All questions about Google Firebase are welcome

innopolis university

Thank you for your attention!