Mobile Development :

11 : Building the Beta Version : Part 2

Serverless Tech, User Auth, Intro to Firebase ...

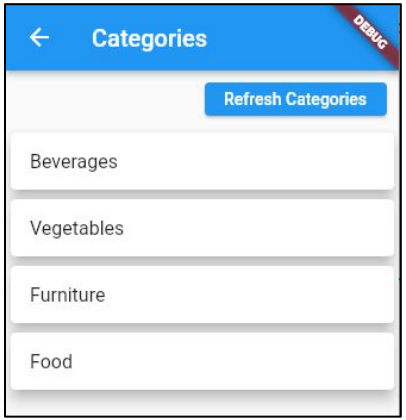
Professor Imed Bouchrika
National School of Artificial Intelligence
imed.bouchrika@ensia.edu.dz

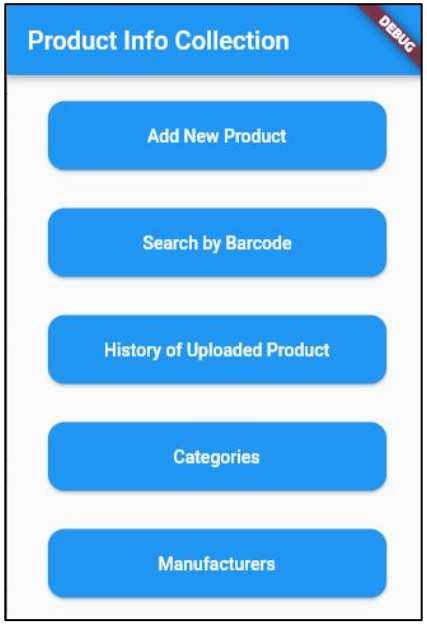
Outline :

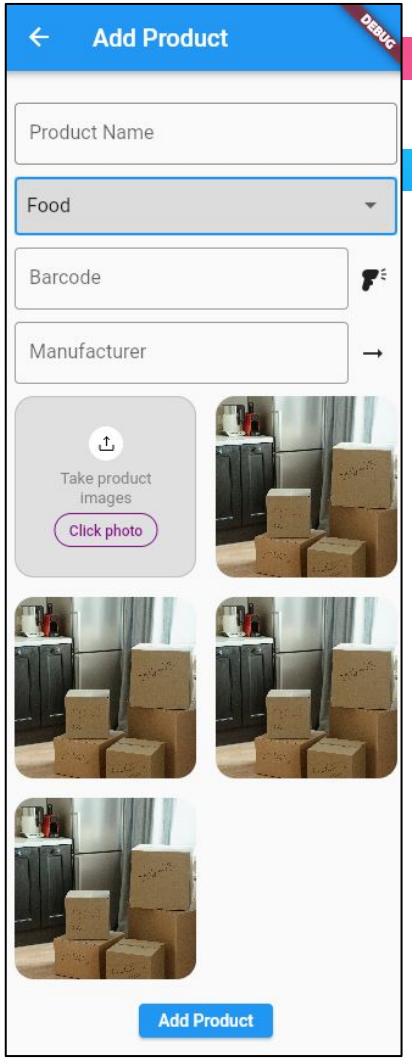
- ***Section 1 : Serverless Tech for Backends***
  - ***Python - Flask (Vercel / Git)***
  - ***Postgresql (Supabase)***
- ***Section 2 : User Authentication, Part 1***
  - ***Introduction & Motivation***
  - ***Design & Technologies***
  - ***Implementing from Scratch***
  - ***Using 3rd Party Authentication Services***
- ***Section 3 : Firebase Services***
  - ***Introduction, Setup and Configuration***

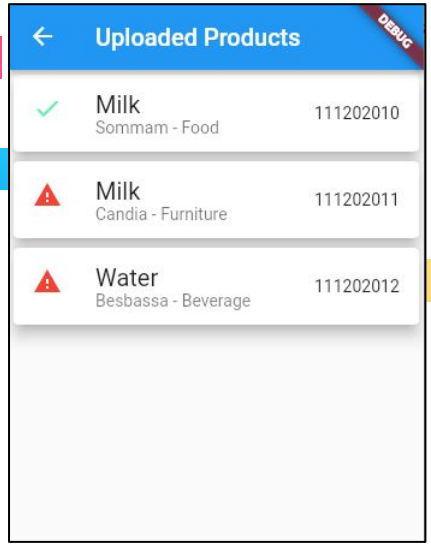
From MVP to Beta Version Case Study

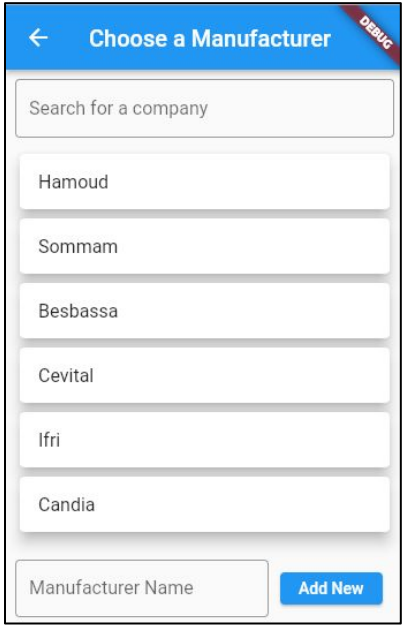
- **App for Collecting Product Descriptions**
  - The customer needs an App for their employees to collect information about products.
  - Information includes:
    - *Product name*
    - *Barcode*
    - *Product Manufacturer*
    - *Product Image**s***
  - The app should work in offline mode. Preferably, when there is internet connection, **collected data can be uploaded** and sync can be performed.
  - The staff, who are the app users can search for a product by barcode to see all product information being **uploaded by them or by other staff.**

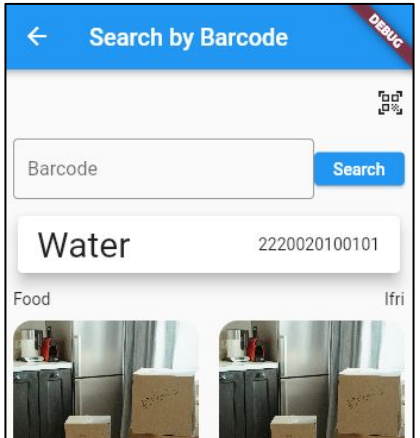

Categories
DEBUG
Refresh Categories
Beverages
Vegetables
Furniture
Food

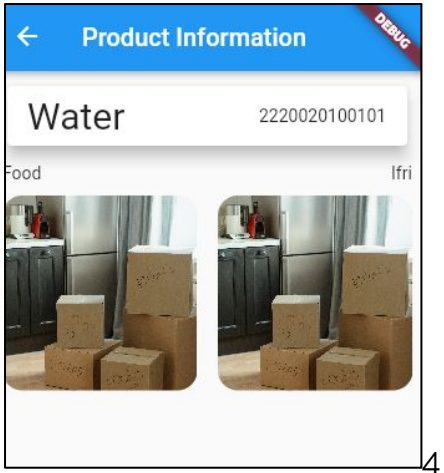

Product Info Collection
DEBUG
Add New Product
Search by Barcode
History of Uploaded Product
Categories
Manufacturers


Add Product
DEBUG
Product Name
Food
Barcode
Manufacturer
Take product images
Click photo
Add Product


Uploaded Products
DEBUG
Milk
Sommam - Food
111202010
Milk
Candia - Furniture
111202011
Water
Besbassa - Beverage
111202012

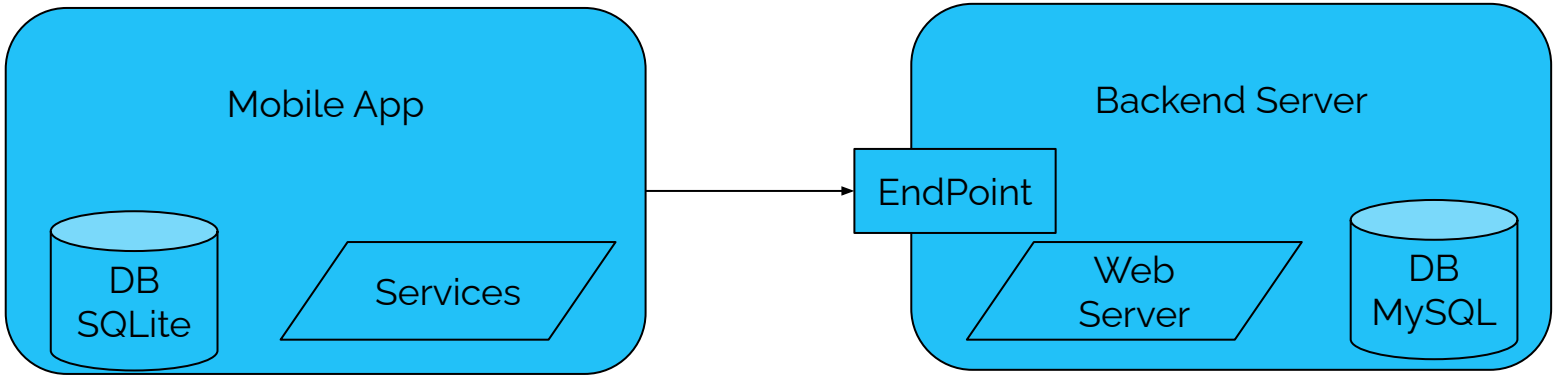

Choose a Manufacturer
DEBUG
Search for a company
Hamoud
Sommam
Besbassa
Cevital
Ifri
Candia
Manufacturer Name
Add New


Search by Barcode
DEBUG
Barcode
Search
Water
2220020100101
Food
Ifri


Product Information
DEBUG
Water
2220020100101
Food
Ifri

From MVP to Beta Version Case Study

- **Architecture of the Solution (not App)**



From MVP to Beta Version Case Study

- **Previous Implementation,**
  - **Linked to a Backend written in simple PHP**
    - Easy and fast to code API Endpoints
    - Instant to deploy and see results
    - Linked with existing MySQL / phpMyAdmin
    - Hosting is available (There are free providers)

Section 1

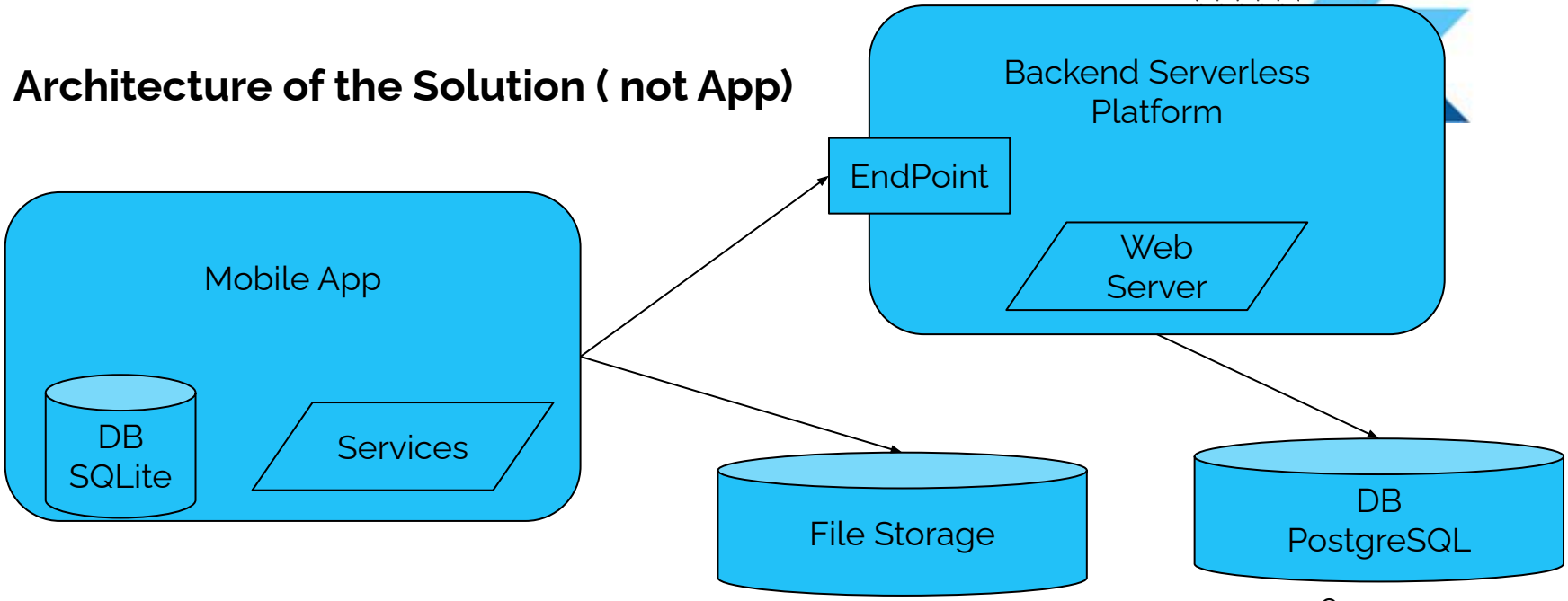
Serverless Technologies

Serverless Technology for Mobile App Backends

- **Improvement to the previous implementation:**
  - Use Python for Backend instead :
    - Existing Framework : **Flask** (Or even Django)
    - Link with PostgreSQL provided by Supabase
  - Use Better Storage Facilities : S3 Storage or Firebase Storage
  - Integrate Firebase Services
  - Users' Authentication.

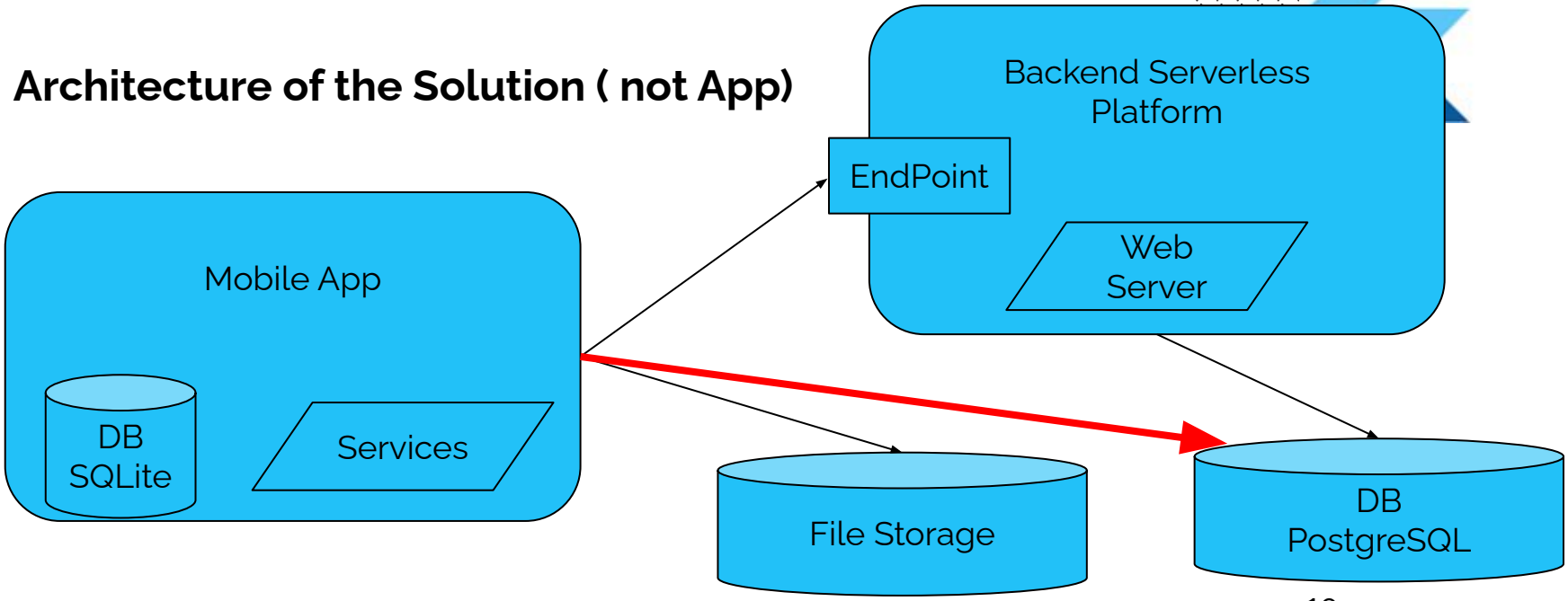
Serverless Technology for Mobile App Backends

- **Architecture of the Solution (not App)**



Serverless Technology for Mobile App Backends

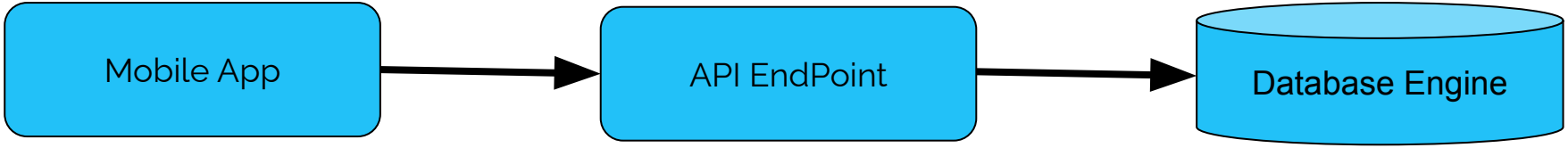
- **Architecture of the Solution (not App)**



Serverless Technology for Mobile App Backends

- **Architecture of the Solution (not App) : which one is better**
  - **Direct Database Access**

  - **Access to Database via EndPoint API**



Serverless Technology for

Personally, I always prefer access to the DB to be done via an EndPoint Server where extra business logic can be performed. This involves:

- Curation of Data,
- Validation
- Advanced Business Logic (Machine Learning..)
- Communication with Other services.
- Better control of security
- Better extensibility and easy to roll new features
- Easy to change the providers, technologies
    - ? Just the complexity you will be overloading the Endpoint, deploy more instances to overcome the problem

Direct Database Access from your mobile app :

- For simple and disposable apps having no intention to maintain them in the future.
- What happens if you are forced to change the database provider ? or just a database table **even a column name** inside the database table ?
- Security ? it is too complex to enforce and master. What happens if we can declasse your app and find your keys ? we will flood your database with requests and you end up paying a large bill.

```
    raise HTTPStatusError(message, request=request, response=self)
httpx.HTTPStatusError: Client error '429 Too Many Requests' for url 'https://rmuifxmxzskeidlcnnm
w.supabase.co/auth/v1/signup'
For more information check: https://httpstatuses.com/429
```

Serverless Technology for

Personally, I always prefer access to the DB to be done via an EndPoint Server where extra business lo

For secu

I do it even for the File Storage, though for Firebase, developers connect mobile apps directly with the Firebase Storage.

erverless
orm

eb
ver

DB
SQLite

Services

File Storage

DB
PostgreSQL

Serverless Technology for Mobile App Backends

- **Hello World in Flask**
  - Fask, Like Java Spark, it is a **python** micro framework for writing web applications.
  - Microframe does usually not use contain common libraries like database access/ORM, form validation…
  - It is written in Python, therefore you can important any python library you want from Machine Learning to other advanced libraries .
  - For a full framework :
    - Python ⇒ Django
    - Java ⇒ SpringBoot

Serverless Technology for Mobile App Backends

- **Hello World in Flask**

```
from flask import Flask
import json

app = Flask(__name__)

@app.route('/categories.get')
def get_categories():
    data=[{'id':1,'name':'Food'},{'id':2,'name':'Beverage'}]
    return json.dumps(data)


@app.route('/')
def index():
    return 'Welcome ENSIA Students from Flask!'

if __name__ == "__main__":
    app.run(port=8080)
```

Serverless Technology for Mobile App Bac

To get rid of the development message, you need to integrate your Flask App with a full Web Server Gateway Interface (WSGI) Like Waitress…

https://en.wikipedia.org/wiki/Web_Server_Gateway_Interface

- Hello World in Flask

```
from flask import Flask
import json

app = Flask(__name__)

@app.route('/categories.get')
def get_categories():
   data=[{'id':1,'name':'Food'},{'id':2,'name':'Beverage'}]
   return json.dumps(data)


@app.route('/')
def index():
   return 'Welcome ENSIA Students from Flask!'

if __name__ == "__main__":
   from waitress import serve
   serve(app, port=8080)
```

Serverless Technology for Mobile App Backends

- **Running the Hello World in Flask**
  - Install the required Python packages
    - pip3 install flask
    - pip3 install waitress
  - Run the Flask Application (File name is : hello.py)
    - python3 hello.py
      - Or sometimes
    - flask --app hello run

Serverless Technology for Mobile App Backends

- **Hello World in Flask**

```
from flask import Flask
import json

app = Flask(__name__)

@app.route('/categories.get')
def get_categories():
    data=[{'id':1,'name':'Food'},{'id':2,'name':'Beverage'}]
    return json.dumps(data)


@app.route('/')
def index():
    return 'Welcome ENSIA Students from Flask!'

if __name__ == "__main__":
    app.run(port=8080)
```

This is the route which **binds URL** to a function to perform a given business logic

__name__ is a special variable when the script is invoked directly (not imported), the __name is set as "__main__"

Serverless Technology for Mobile App Backends

- **Getting Variables from URLs**

```
from flask import Flask
import json

app = Flask(__name__)

@app.route('/categories.getById/<int:cat_id>')
def get_category_byId(cat_id):
    // some business logic here…
   return json.dumps(data)


@app.route('/')
def index():
   return 'Welcome ENSIA Students from Flask!'

if __name__ == "__main__":
   app.run(port=8080)
```

You can enclose URL variables inside **<type:variable_name>** or simply **<variable_name>**

Serverless Technology for Mobile App Backends

- **Getting GET/POST Variables**

```
from flask import Flask , request
import json

app = Flask(__name__)

@app.route('/user.login', methods=['GET', 'POST'] )
def users_login():
   username=request.form.get('username')
   password=request.form.get('password')
   //some more business logic
   return json.dumps(data)

@app.route('/')
def index():
   return 'Welcome ENSIA Students from Flask!'

if __name__ == "__main__":
   app.run(port=8080)
```

GET or POST variables can be accessed via the special variable :

request.form.get(VAR_NAME)

Serverless Technology for Mobile App Backends

- **Getting Binary Files from Uploads**

```
from flask import Flask
import json

app = Flask(__name__)

@app.route('/photo.upload', methods=['POST'] )
def users_login():
   if request.method=='POST':
         file=request.files.get('file_name')

   //some more business logic
   return json.dumps(data)

@app.route('/')
def index():
   return 'Welcome ENSIA Students from Flask!'

if __name__ == "__main__":
   app.run(port=8080)
```

Binary Data Files can be accessed via the special variable :

request.files.get(FILE_NAME)

Serverless Technology for Mobile App Backends

- **Logging**

```
from flask import Flask
import json

app = Flask(__name__)

app.logger.debug('A value for debugging')
app.logger.warning('A warning occurred (%d apples)', 42)
app.logger.error('An error occurred')
```

Serverless Technology for Mobile App Backends

- **How to run it : Server-Centric approach**
  - Get a dedicated Server or even virtual …
  - You need to manage it yourself
    - Upgrades
    - Security
  - Scaling, can be difficult to main
  - But, you have control over everything.
  - Cost : Cannot say (it may cost $5/month ..)

Serverless Technology for Mobile App Backends

- **Introduction to Serverless Technologies**
  - Even though, it is called serverless, the server is always there but:
    - Provided when there is a request from a user
    - You pay usually per request / processing duration / bandwidth transferred.
  - Serverless Technologies provided by the hosting provider where they have the infrastructure to deploy dynamically servers/frameworks on demand.

Serverless Technology for Mobile App Backends

- **Introduction to Serverless Technologies**
  - **Benefits..**
    - Maintenance of the technology taken care by the infrastructure provider
    - Scaling is always being taken care of. When more users, the hosting provider will deploy more resources dynamically, (but be prepared to pay more)
  - **Drawbacks..**
    - You have no control
    - Dependent on a technology provider

Serverless Technology for Mobile App Backends

- **Serverless Technology Providers :**
  - **Vercel**
    - Supports Python, Node.js, Go
    - PostgreSQL is given
  - **AWS Lambda**
    - Very mature technology supporting most languages
  - **Netlify**
    - Supports JavaScript and TypeScript
  - **Render**
    - Supports even Laravel, Django/Python..
  - **Firebase Functions**
  - ..

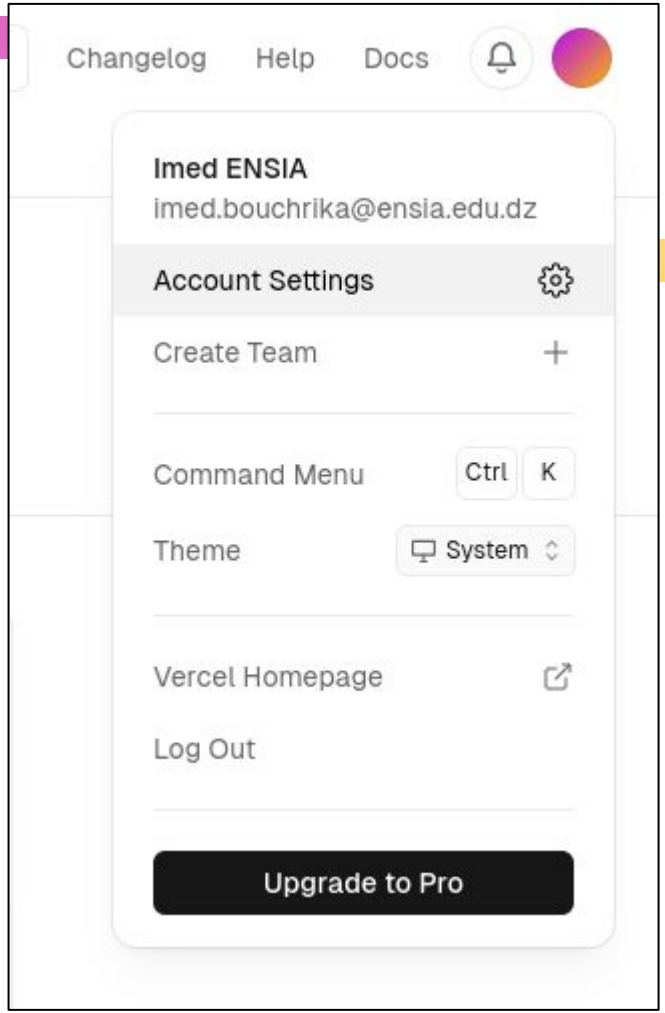
Serverless Technology for Mobile App Backends

- **Serverless Technology Providers :**
  - **Vercel**
    - Suppo
    - Postgr
  - **AWS Lamb**
    - Very m
  - **Netlify**
    - Supports JavaScript and TypeScript

Vercel will be tested for creating a mobile app backend using Python and Supabase Database

Serverless Technology for Mobile App Backends

- **Vercel & Git Integration**
  - You can always click on Account Settings
    - ///

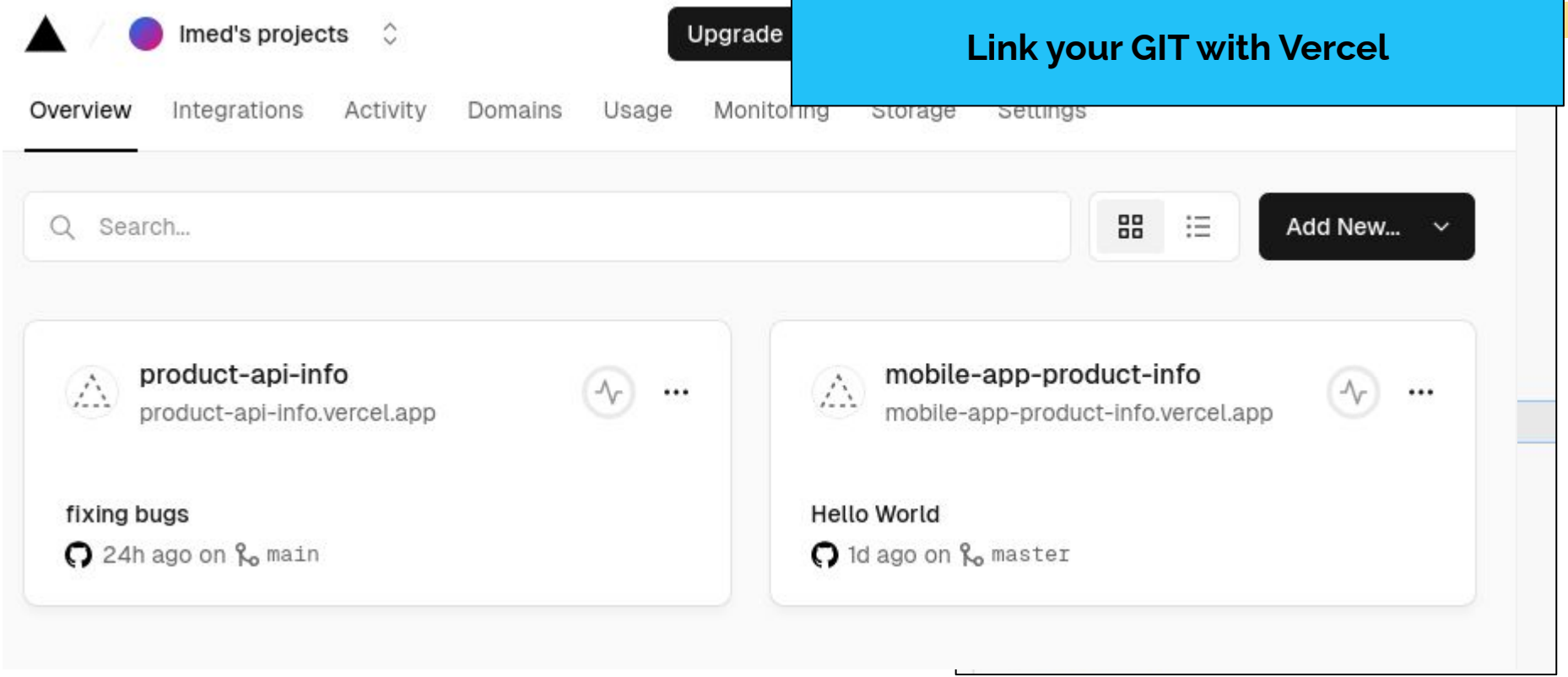


From MVP to Beta Ver
Case Study

Signup and create a project

Importantly,

Link your GIT with Vercel



Let's build something new.

To deploy a new Project, import an existing Git Repository or get started with one of our Templates.

Better to search for a template and clone

Continue with GitHub

Continue with GitLab

Continue with Bitbucket

Manage Login Connections

Import Third-Party Git Repository →

Clone Template

Next.js

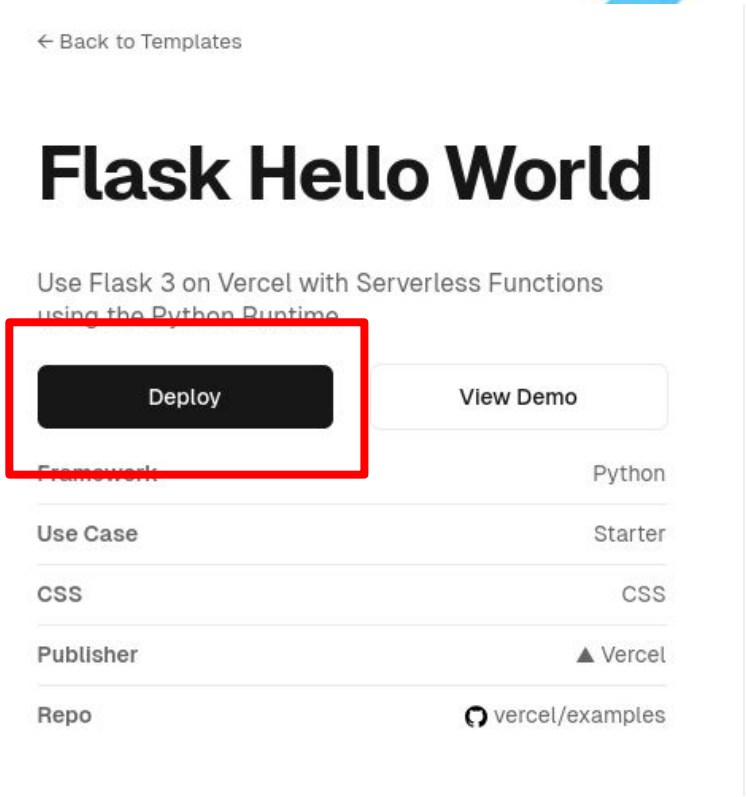
SvelteKit (v1)

Nuxt.js

Vite

Browse All Templates →

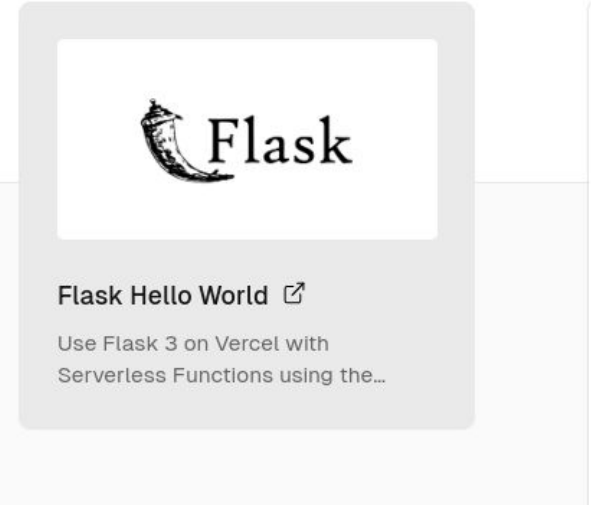
From MVP to Beta Version Case Study



Serverless Technology for Mobile App Backends

You're almost done.

Please follow the steps to configure your Project and deploy it.



Create Git Repository

To ensure you can easily update your project after deploying it, a Git repository must be created. Every push to that Git repository will be deployed automatically.

Git Scope: imed5

Repository Name: flask-product-info-no-database

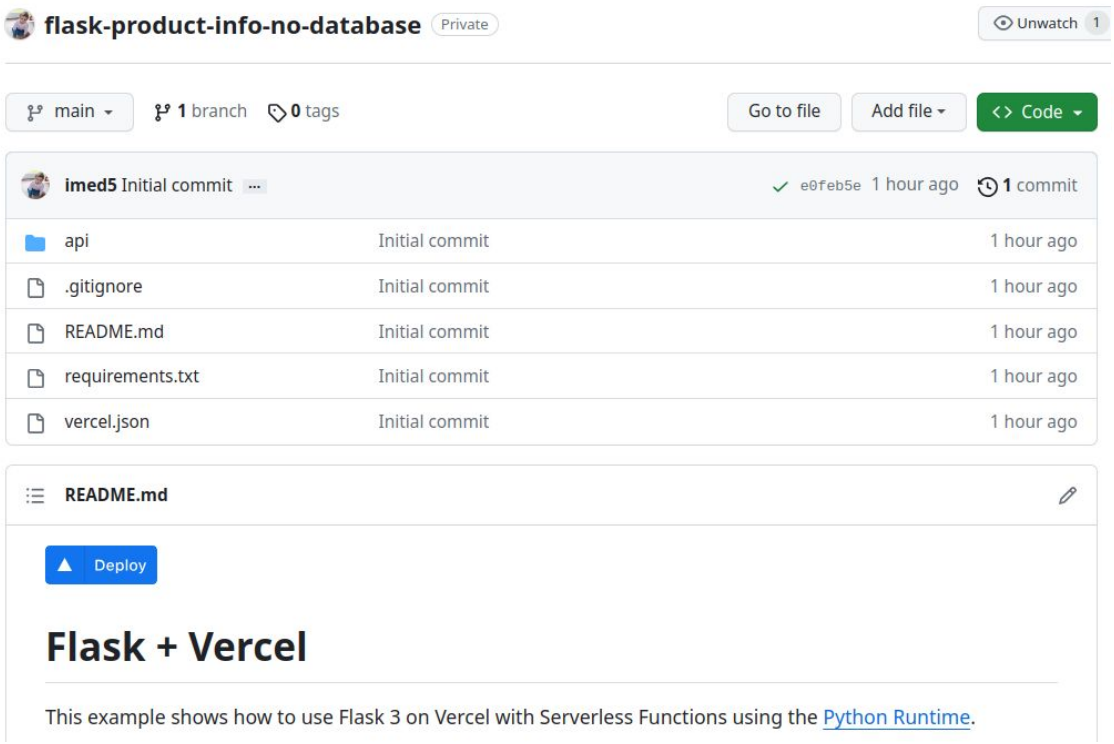
☑ Create private Git Repository

Create

Serverless Technology for Mobile App Backends

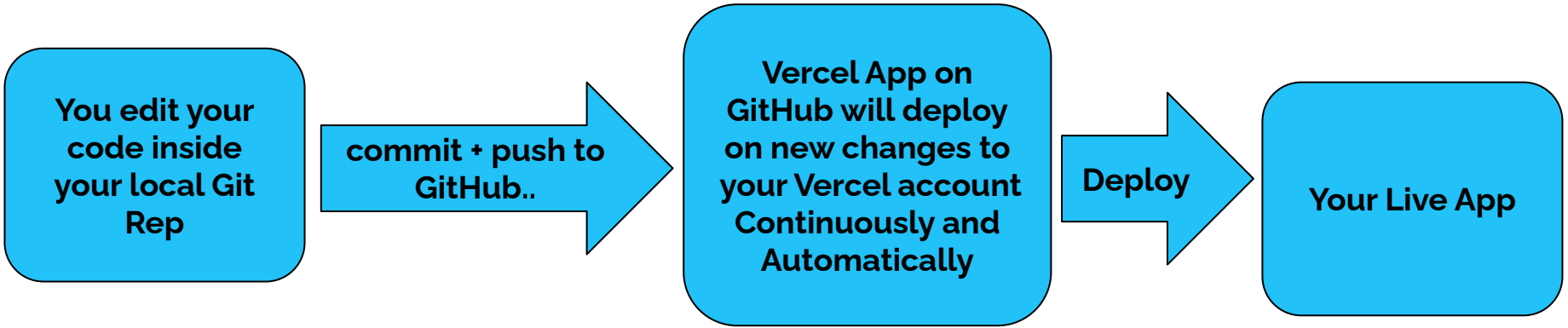
- **Git Repo is created.**

  (You can use an existing Repo)

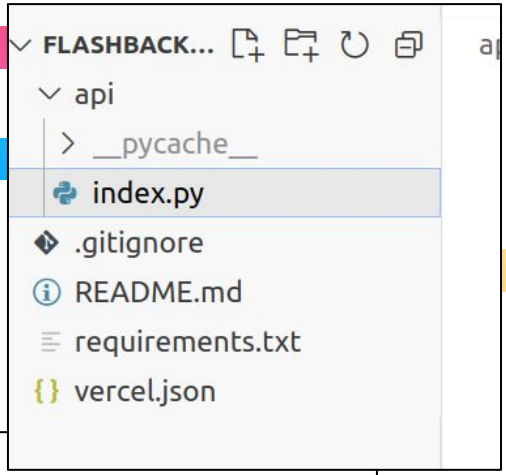


Serverless Technology for Mobile App Backends

- Continuous Integration/Continuous Delivery (CI/CD) :



Serverless Technology for Mobile App Backends



- **Vercel project structure**
  - vercel.json

```
{
 "rewrites": [
   { "source": "/(.*)", "destination": "/api/index" }
 ]
}
```

  - requirements.txt

```
Flask==3.0.0
supabase
```

Serverless Technology for Mobile App Backends

- **Develop, Run & Test LOCALLY**
  - To speed up
  - To avoid paying for the deployment/building time
- Remember for Serverless, you may not need the main app.run method

```
if __name__ == "__main__":
    app.run(port=8080)
```

- For local testing, use the command : **flask --app hello run**

  (hello.py is the main file name)

Serverless Technology for Mobile App Backends

- **Get GIT Folder setup on your local machine:**
  - Create a folder
  - Inside the folder, execute the following commands
    - git init
    - git remote add origin link_to_your_git_folder
      git remote add origin https://github.com/imed5/flask-product-info-no-database.git
    - git pull link_to_your_git_folder main
      git pull https://github.com/imed5/flask-product-info-no-database.git main
    - git branch --set-upstream-to=origin/main master
    - git pull

Serverless Technology for Mobile App Backends

- **Get GIT Folder setup on your local machine:**
  - git checkout -b feature/YYYY_MM_DD_NAME
  - git add .
  - git commit -m “description of the changes”
  - git push origin feature/YYYY_MM_DD_NAME
    - Make PR (Pull Request)
    - Merge
  - git checkout master
  - git pull

Imed's projects Hobby / product-api-info

Upgrade Feedback Changelog Help Docs

Project Deployments Analytics Speed Insights Logs Storage Settings

product-api-info

Git Repository Domains Visit

Production Deployment

The deployment that is available to your visitors.

Build Logs Runtime Logs Instant Rollback

```
Hello, World!
```

Deployment
product-api-info-p9yh6bz1l-imeds-projects-6ec09fae.vercel.app

Domains
product-api-info.vercel.app +2

Status
Ready

Created
7h ago by imed5

Source
main
eb35843 fixing bugs

Logs

Search, inspect, and share the runtime logs from your Vercel projects.

Filters Reset

- My Project Presets
- Team Project Presets
- **Timeline**

Past 30 minutes

- **Level**
  - Info 6
  - Warning 4
  - Error 0
- Environment
- **Function**

10 total logs found...

Time	Status	Host	Request	Message
DEC 11 22:10:30.50	200	product…	[GET] /	
DEC 11 22:10:09.27	200	product…	[GET] /	
DEC 11 22:09:27.17	404	product…	[GET] /favicon.ico	
DEC 11 22:09:26.98	200	product…	[GET] /	
DEC 11 22:09:26.00	401	product…	[GET] /	
DEC 11 22:09:24.05	404	product…	[GET] /favicon.ico	
DEC 11 22:09:21.99	404	product…	[GET] /favicon.ico	
DEC 11 22:09:21.80	200	product…	[GET] /	
DEC 11 22:09:19.62	206	product…	[GET] /	using Web Server Gatewa…
DEC 11 22:09:19.62	200	product…	[GET] /	using Web Server Gatewa…

Show New Logs

[GET] /

Time	December 11 22:09:26.00 GMT+01…
Request Path	/
Status Code	401
Host	product-api-info-p9yh6bz1l-ime…
Request ID	z7mt2-1702328966465-04063…
Request User Agent	Vercelbot/0.1 (scr…
Level	Warning
Environment	preview
Branch	main

Serverless Technology for Mobile App Backends

- **Examples for getting categories**

https://flask-product-info-no-database.vercel.app/categories.get

```
from flask import Flask
import json

app = Flask(__name__)

@app.route('/categories.get')
def home():
   data=[
           {'id':1,'name':'Food'},
           {'id':2,'name':'Furniture'},
           {'id':3,'name':'Beverages'},
       ]
   return json.dumps(data)

@app.route('/')
def about():
   return 'Welcome ENSIA Students'
```

Serverless Technology for Mobile App Backends

- **Examples for getting categories**

Deploy/Run locally FIRST, execute in the shell the following command:

flask --app api/index run

```
from flask import Flask
import json

app = Flask(__name__)

@app.route('/categories.get')
def home():
   data=[
           {'id':1,'name':'Food'},
           {'id':2,'name':'Furniture'},
           {'id':3,'name':'Beverages'},
       ]
   return json.dumps(data)

@app.route('/')
def about():
   return 'Welcome ENSIA Students'
```

Serverless Technology for Mobile App Backends

- **Examples for getting**

To view/Test the App, Open a browser on the address

http://localhost:5000

Or

http://localhost:5000/categories.get

localhost:5000/categories.get

Paused

Relaunch to up

[{"id": 1, "name": "Food"}, {"id": 2, "name": "Furniture"}, {"id": 3, "name": "Beverages"}]

Serverless Technology for Mobile App Backends

- **Examples for getting categories**

```php
case "companies.add":{
    $data=json_decode($vars['companies']);
    $mapping=[];
    if (is_array($data)){
        for($i=0;$i<count($data);$i++){
            $line=$data[$i];
            $db->query("INSERT INTO companies (name) VALUES (?)",$line->name);
            $mapping[$line->id]=$db->lastInsertID();
        }
        $ret['status']='OK';
        $ret['mapping']=$mapping;
        echo json_encode($ret);
    }
    exit;
}break;
```

PHP Code

Serverless Technology for Mobile App B

- Examples for getting cat

```
case "companies.add":{
    $data=json_decode($vars['companies']);
    $mapping=[];
    if (is_array($data)){
        for($i=0;$i<count($data);$i++){
            $line=$data[$i];
            $db->query("INSERT INTO companie
            $mapping[$line->id]=$db->lastIns
        }
        $ret['status']='OK';
        $ret['mapping']=$mapping;
        echo json_encode($ret);
    }
    exit;
}break;
```

PHP

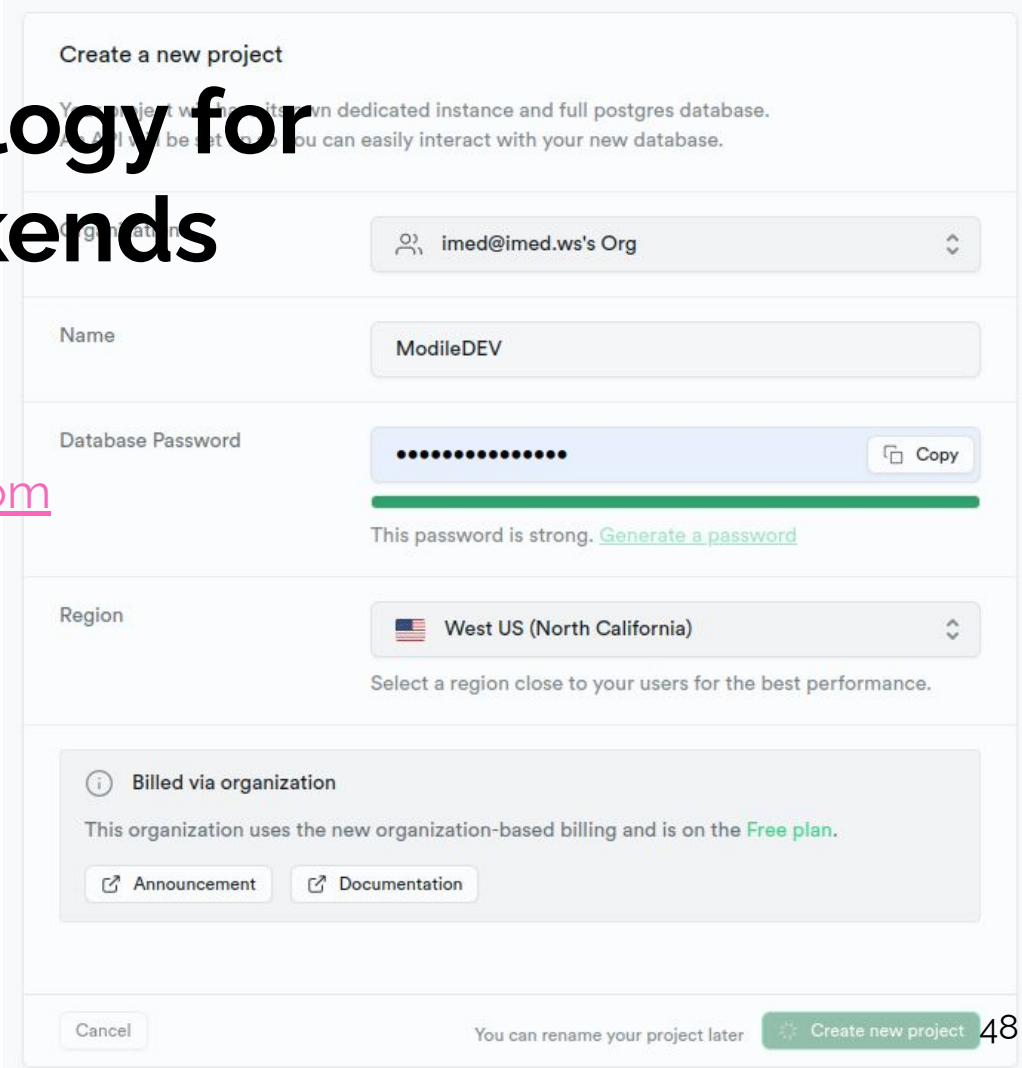
```
from flask import Flask, request
import json
app = Flask(__name__)

@app.route('/companies.add',methods=['GET','POST'])
def api_companies_create():
    print('Variables...'+str(request.args))
    if request.args.get('companies'):
        try:
            companies=json.loads(str(request.args.get('companies')))
            mapping={}
            for company in companies:
                print('Creating a company with the name '+company['name'])
                mapping[company['id']]=company['id']
            data={'status':'OK','message':'success','mapping':mapping}
            return json.dumps(data)

        except Exception as error:
            print(str(error))
            data={'status':'Error','message':'Exception ...','mapping':[]}
            return json.dumps(data)

    data={'status':'Error','message':'No data','mapping':[]}
    return json.dumps(data)
```

Serverless Technology for Mobile App Backends

- **Use of Relational Databases:**
  - As data has relation among them, relational database engine the recommended solution
  - Using NoSQL is only recommended when processing large data which is **already structured**
  - **PostgreSQL** another free relational database with its better performance compared to other dbms.
  - Supabase is another cloud provider offering free PostgreSQL hosting (To get started, when you grow, you pay..).

Serverless Technology for Mobile App Backends

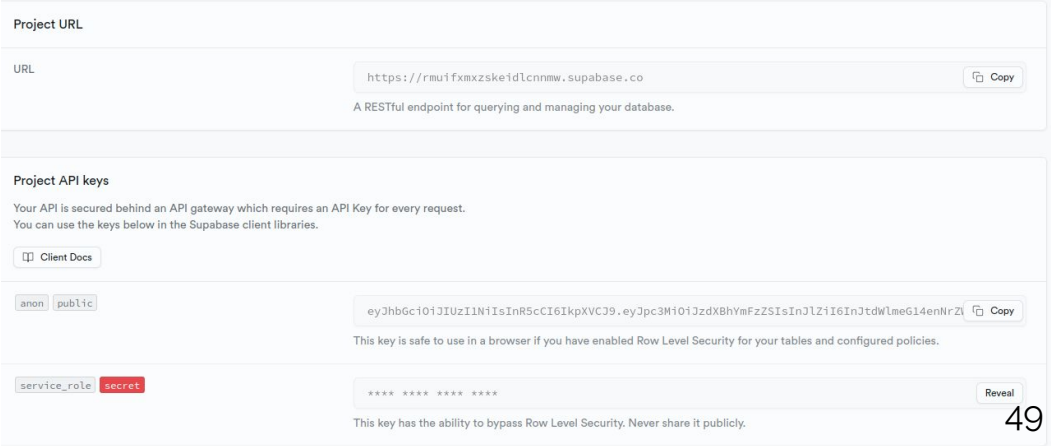
- **Steps to get Started :**
  1. Signup at www.supabase.com
  2. Create a Project and set a Password, Please keep the password at a safe place



Serverless Technology for Mobile App Backends

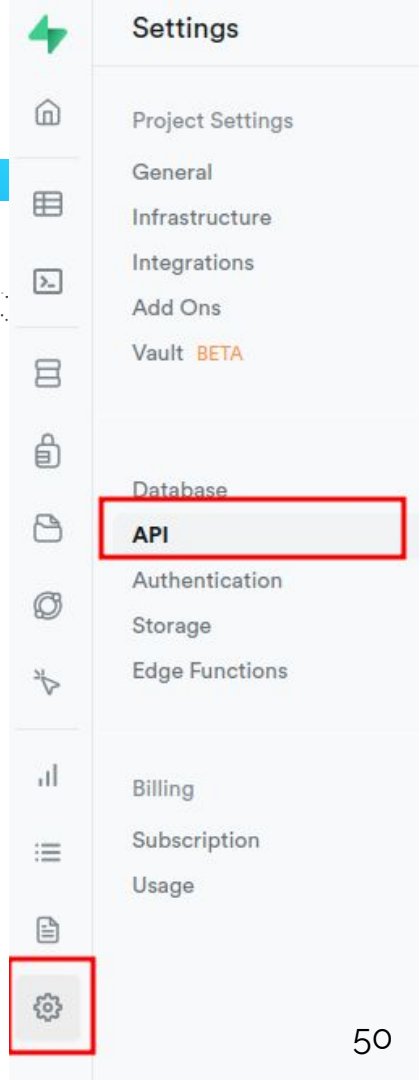
- **Steps to get Started :**

  1. Signup at www.supabase.com
  2. Create a Project and set a Password, Please keep the password at a safe place
  3. **Copy the URL and Key**



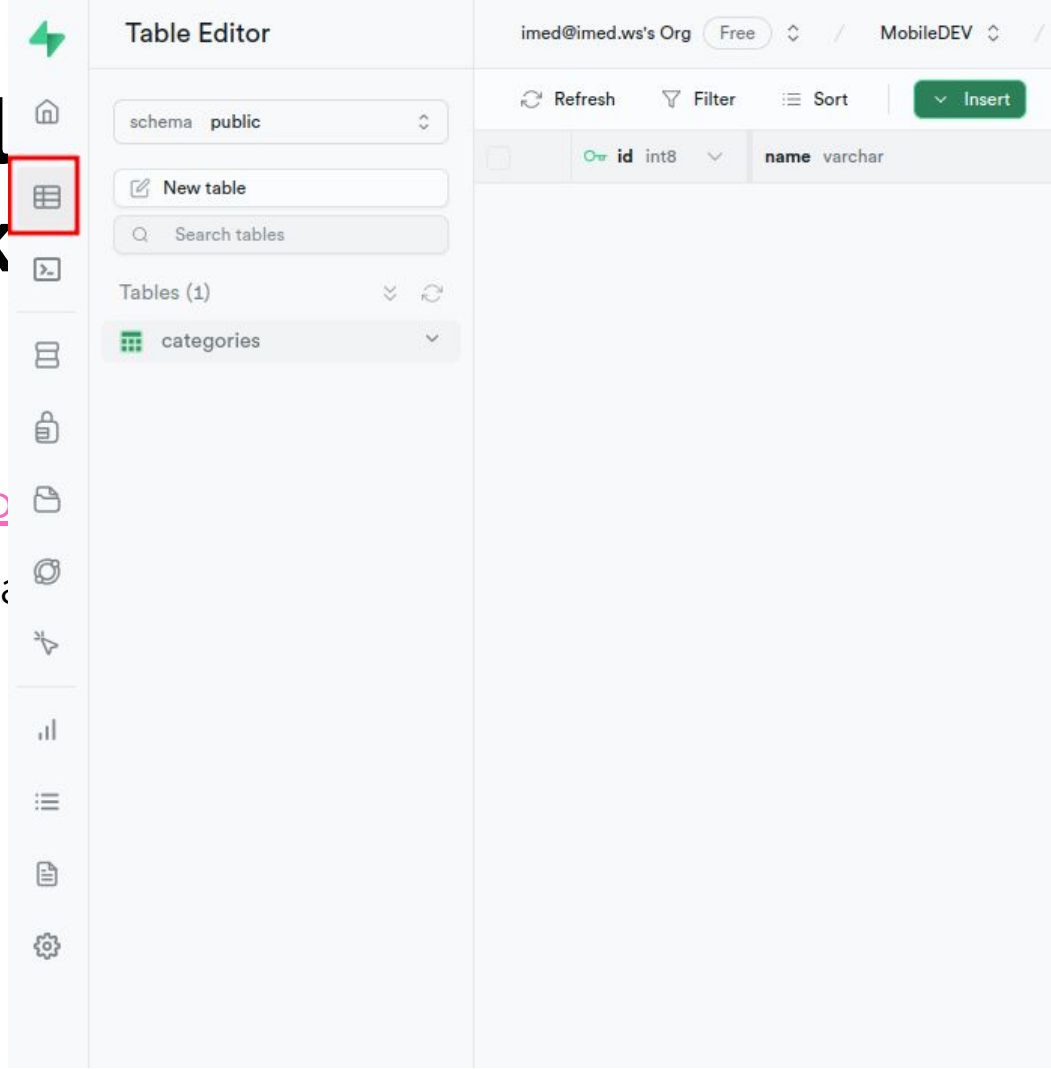
Serverless Technology for Mobile App Backends

- **Steps to get Started :**
  1. Signup at www.supabase.com
  2. Create a Project and set a Password, Please keep the password at a safe place
  3. **Copy the URL and Key**



Serverless Technol
Mobile App Back

- **Steps to get Started :**
  1. Signup at www.supabase.co
  2. Create a Project and set a Pa
     password at a safe place
  3. Copy the URL and Key
  4. **Open the Table Editor**
     Create Tables, insert
     data...

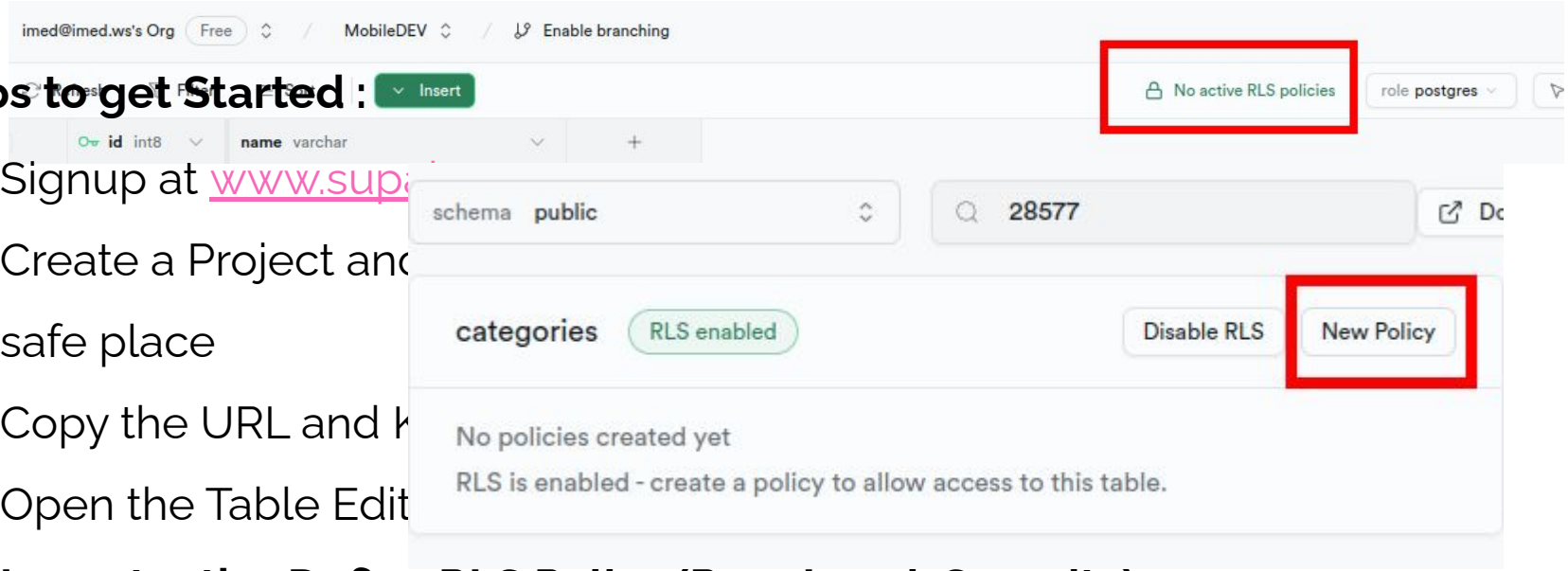


Serverless Technology for Mobile App Backends

- **Steps to get Started :**
  1. Signup at www.supabase.com
  2. Create a Project and set a Password, Please keep the password at a safe place
  3. Copy the URL and Key
  4. Open the Table Editor Create Tables, insert data…
  5. **Importantly : Define RLS Policy (Row-Level Security)**

Serverless Technology for Mobile App Backends

- **Steps to get Started :**
  1. Signup at www.supa
  2. Create a Project and
     safe place
  3. Copy the URL and k
  4. Open the Table Edit
  5. **Importantly : Define RLS Policy (Row-Level Security)**



Serverless Technology for Mobile App Backends

- **Linking with Flask/Vercel with PostgreSQL on Supabase**
  - Note that Vercel has its own PostgreSQL infrastructure also, but we aim to create a solution with many distributed components. **Vercel offers an integration with Supabase.**
  - You can either,
    - Use the python package developed by supabase
      - pip3 install supabase
      - Make sure to add it to the requirements.txt file
    - Use the common package for accessing PostgreSQL databases

Serverless Technology for Mobile App Backends

- **Linking with Flask/Vercel with PostgreSQL on Supabase**
  - Note that Vercel has its own PostgreSQL infrastructure also, but we aim to create a solution with many distributed components. **Vercel offers an integration with Supabase.**
  - You can either,
    - Use the python package developed by supabase
      - pip3 install supabase
      - Make sure to add it to the requirements.txt file
    - Use the common package for accessing PostgreSQL databases

Serverless Technology for Mobile App Backends

- **Linking with Flask**

```
from flask import Flask, request
import json
from supabase import create_client, Client

app = Flask(__name__)

#This is the very stupid way to store private/confidential data inside GIT
#Store inside a file to be ignored
url="https://yvnctxteoqxvamhbzpvd.supabase.co"
key="eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6Inl2bmN0eHRlb3F4dmFtaGJ6cHZkIiwi
cm9sZSI6ImFub24iLCJpYXQiOjE3MDIzMDE4ODQsImV4cCI6MjAxNzg3Nzg4NH0.4AoGpxSQF3-T4b_dJ2B5ZfJY1pukT7Gu8xbKq8pN9g
A"
supabase: Client = create_client(url, key)

@app.route('/categories.get')
def api categories get():
   response = supabase.table('categories').select("*").execute()
   return json.dumps(response.data)
```

Test and Run always Locally FIRST

flask --app api/index run

Serverless Technology for Mobile App Backends

- Linking with Flask

Table Editor

imed@imed.ws's Org Free / Product_Info_Mobile_App / Enable branching

schema public

New table

Search tables

Tables (1)

categories

Refresh Filter Sort Insert

id int8	name text
1	Food
2	Beverages

localhost:5000/categories.get

[{"id": 1, "name": "Food"}, {"id": 2, "name": "Beverages"}]

Serverless Technology for Mobile App Backends

- **Linking with Flask and PostgreSQL**
  - There are documentation on the SQL functions provided by the package supabase.

```
response = supabase.table('categories').select("*").execute()
```

    - https://supabase.com/docs/reference/python/introduction
  - If you want to try the common postgreSQL python package to be able to write raw SQL query, you can install
    - sudo pip3 install psycopg2-binary

Serverless Technology for

Settings

imed@imed.ws's Org Free / Product_Info_Mobile_App / Enable branching

Help Feed

Project Settings
General
Infrastructure
Integrations
Add Ons
Vault BETA

Database
API
Authentication
Storage
Edge Functions

Billing
Subscription
Usage

Database Settings

Connection info

Host	db.yvnctxteoqxvamhbzpvd.supabase.co	Copy
Database name	postgres	Copy
Port	5432	Copy
User	postgres	Copy
Password	[The password you provided when you created this project]	

Connection string

PSQL URI Golang JDBC .NET Nodejs PHP Python

- sudo pip3 install psycopg2-binary

```python
from flask import Flask, request
import json

app = Flask(__name__)

#This is the very stupid way to store private/confidential data inside GIT
#Store inside a file to be ignored
url="db.yvnctxteoqxvamhbzpvd.supabase.co"
password="Your Initial Password here…"

import psycopg2

@app.route('/categories.get')
def api_categories_get():
   conn=False
   try:
       conn = psycopg2.connect("dbname='postgres' user='postgres' host='"+url+"' password='"+password+"'")
   except Exception as error:
       print("I am unable to connect to the database")
       return 'cannot connect to database'+str(error)

   curs=conn.cursor()
   curs.execute("select id,name from categories")
   data=[]
   for record in curs:
       print(record)
       data.append({'id':record[0],'name':record[1]})

   return json.dumps(data)
```

Section 2

User Authentication

User Authentication for Flutter Apps

- **Authentication of Users**
  - **Do you need it ?**
    - The app data needs to be accessed from the web or other devices ?
    - To collect information about the user ?
    - The information shown in the app is too confidential ?
  - **How to authenticate ?**
    - By email and password ?
    - By other providers (OAuth (OpenAuthorization : Google, Fb..)
    - By Phone/Emailing OTP/SMS...

User Authentication for Flutter Apps

- **Authentication of Users**
  - **D**
  - **H**
    - By email and password ?
    - By other providers (OAuth (OpenAuthorization : Google, Fb..)
    - By Phone/Emailing OTP/SMS…

ices ?

Remember : The phone is now more and more personal !(Than passwords)

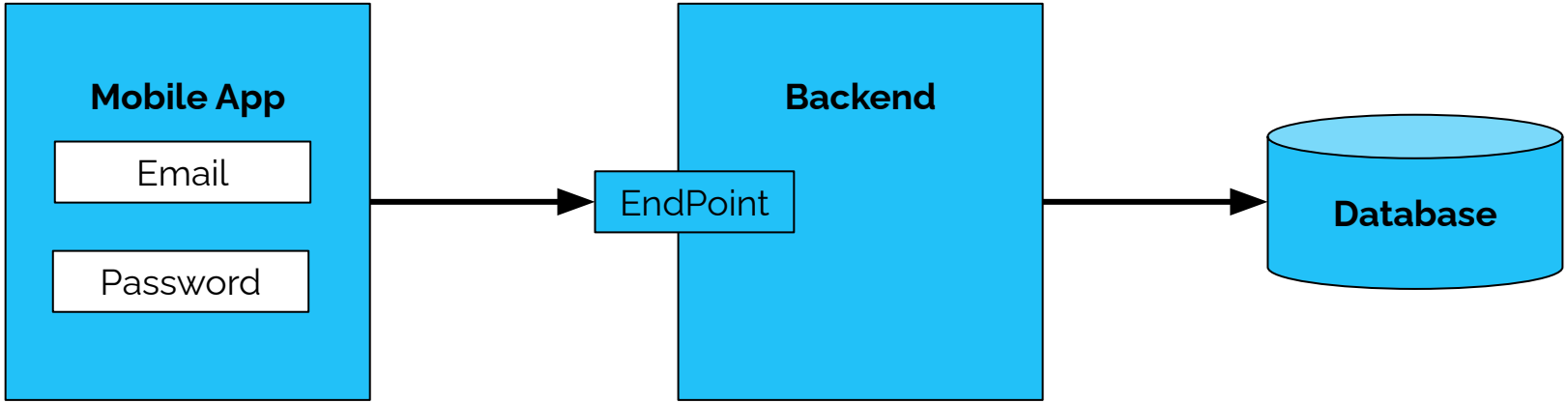
Users are protecting their phones with fingerprint, patterns, pins, face recognition… (But, they are ok in sharing their passwords)

User Authentication for Flutter Apps

- **How to Authenticate Users:**
  - Students rush to use Firebase Auth +Supabase Authentication module where Authentication can be integrated using 10 lines of code.
  - BUT
    - It is highly recommended for the purpose of learning, that you implement it yourself from **scratch**

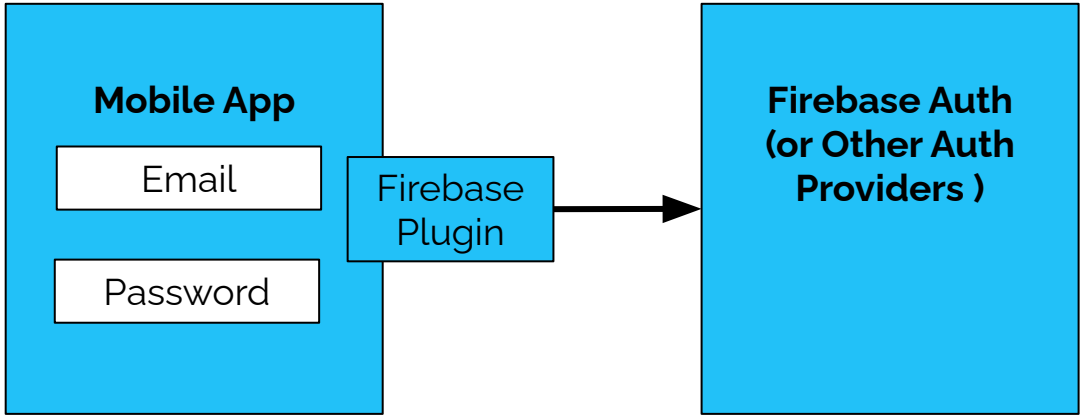
User Authentication for Flutter Apps

- **Implementing Users' Authentication using Email and Password**
  - **Classical Architecture**



User Authentication for Flutter Apps

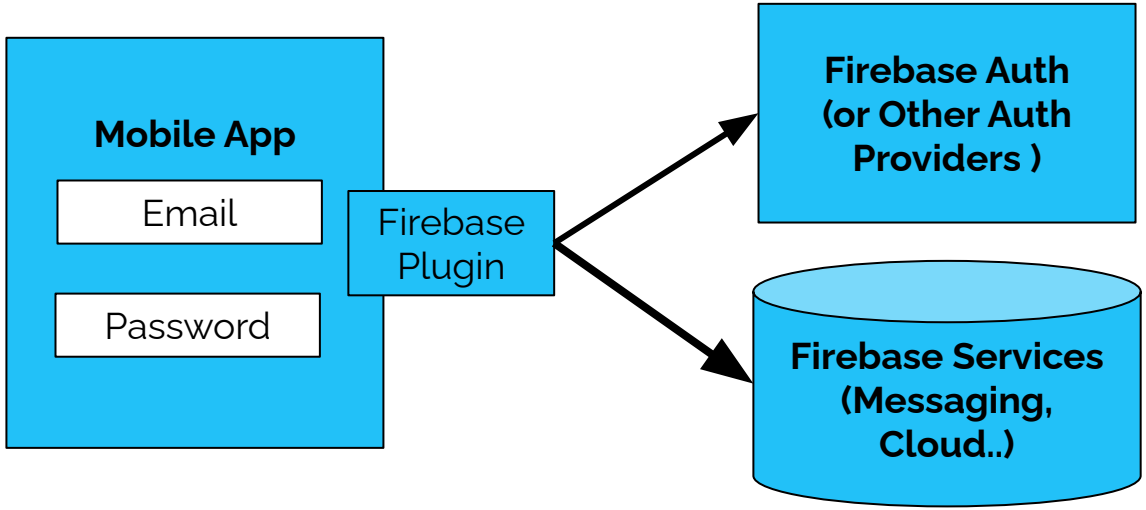
- **Implementing Users' Authentication using Email and Password**
  - **Using 3rd Party Authentication Services**



Remember, we don't store password if we decide to outsource authentication to 3rd party providers

User Authentication for Flutter Apps

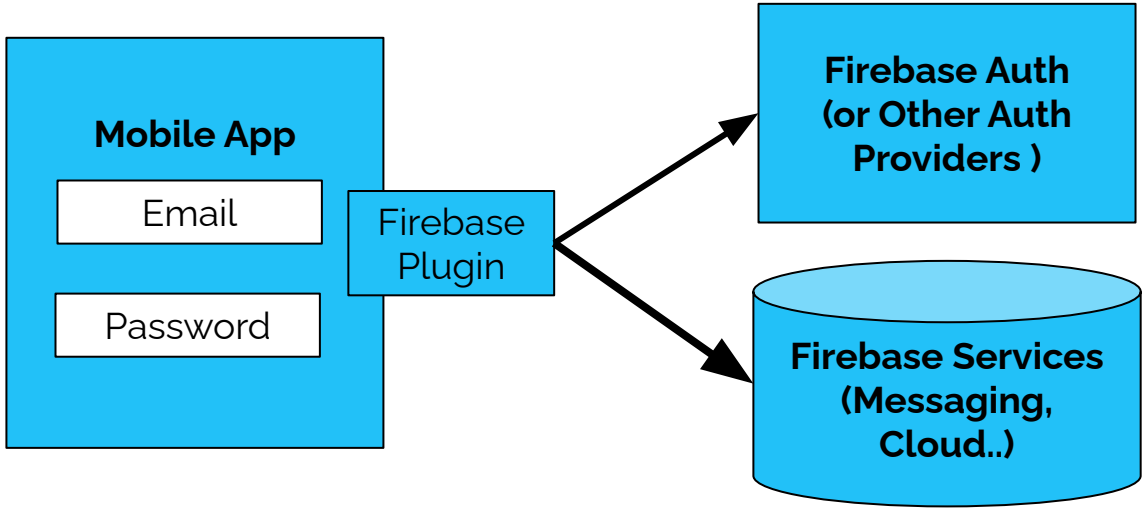
- **Implementing Users' Authentication using Email and Password**
  - **Using 3rd Party Authentication Services**



User Authentication for Flutter Apps

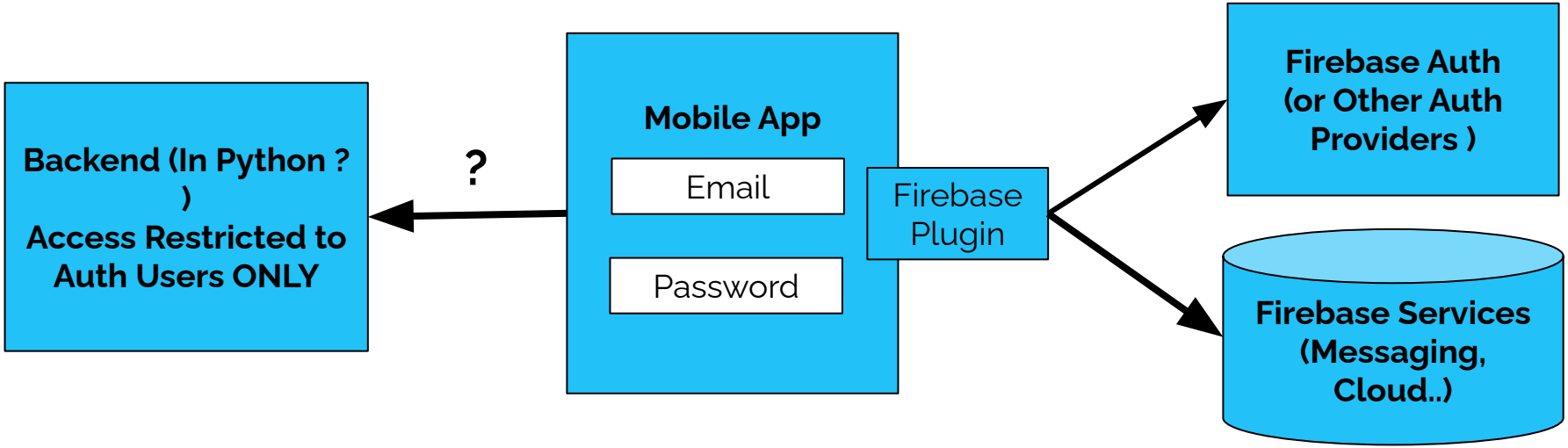
- **Implementing Users' Authentication using Email and Password**
  - **Using 3rd Party Authentication Services**

It is easy to integrate other services in Firebase whilst ensuring only access to Authenticated users



User Authentication for Flutter Apps

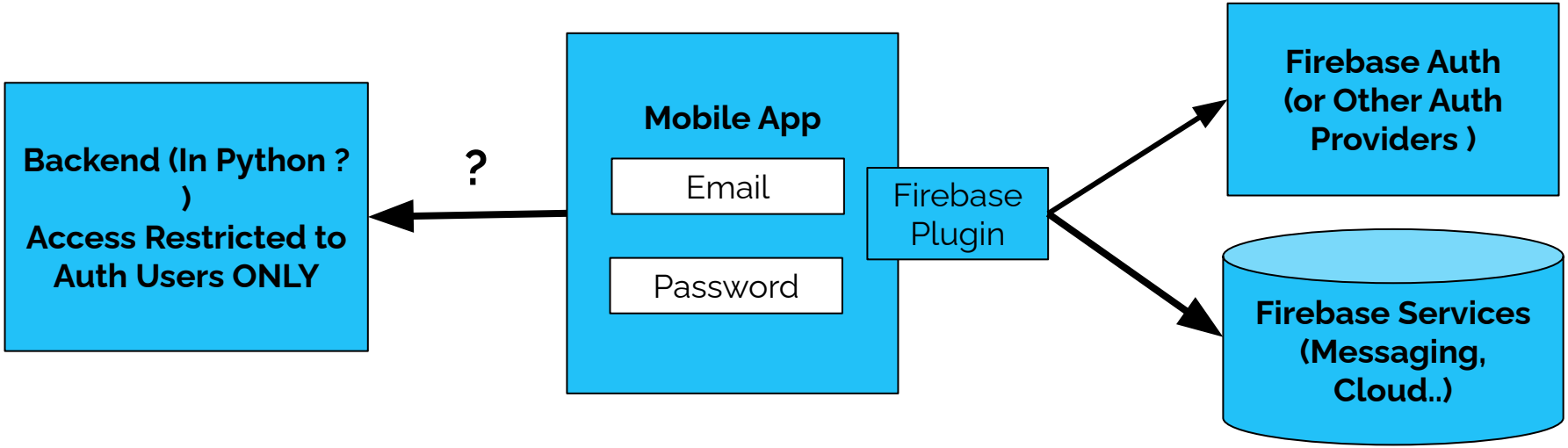
- Implementing Users' Authentication using Email and Password
  - Using 3rd Party Authentication Services



User Authentication for

How to tell a backend : **This is a legitimate user** who needs access to his **private data ?**

- **Im**
  - **Using 3rd Party Authentication Services**



User Authentication for Flutter Apps

- **Implementing Users' Authentication using Email and Password**
  - Implement the API EndPoints
    - **users.signup**
    - **users.login**
    - users.forgetpass
    - users.update
    - users.signup.verify
    - users….

User Authentication for Flutter Apps

- **Implementing Users' Authentication using Email and Password**
  - Implement the API EndPoints
    - **users.signup**
    - **users.login**
    - users.forgetpass
    - users.update
    - users.signup.verify
    - users….

Do we need users.signout ?

```python
@app.route('/users.signup',methods=['GET','POST'])
def api_users_signup():
    email= request.args.get('email')
    password= request.args.get('password')
    error =False
    if (not email) or (len(email)<5): #You can even check with regx
        error='Email needs to be valid'
    if (not error) and ( (not password) or (len(password)<5) ):
        error='Provide a password'
    if (not error):
        response = supabase.table('users').select("*").ilike('email', email).execute()
        if len(response.data)>0:
            error='User already exists'
    if (not error):
        response = supabase.table('users').insert({"email": email, "password": password}).execute()
        print(str(response.data))
        if len(response.data)==0:
            error='Error creating the user'
    if error:
        return json.dumps({'status':500,'message':error})

    return json.dumps({'status':200,'message':'','data':response.data})
```

```python
@app.route('/users.login',methods=['GET','POST'])
def api_users_login():
   email= request.args.get('email')
   password= request.args.get('password')
   error =False
   if (not email) or (len(email)<5): #You can even check with regx
       error='Email needs to be valid'
   if (not error) and ( (not password) or (len(password)<5) ):
       error='Provide a password'
   if (not error):
       response = supabase.table('users').select("*").ilike('email',
                                                                    email).eq('password',password).execute()
       if len(response.data)>0:
           return json.dumps({'status':200,'message':'','data':response.data})

   if not error:
        error='Invalid Email or password'

   return json.dumps({'status':500,'message':error})
```

User Authentication for Flutter Apps

- **Implementing Users' Authentication using Email and Password**
  - Flutter Code ?
    - What to modify ?
    - How to know that the user is logged in and have a valid account ?
    - What packages needed to get started

```
static Future<String> signupUser(Map data) async {
  Map<String, dynamic> form_data = {
    'email': data['email'],
    'password': data['password']
  };
  var response = await dio.post(api_endpoint_user_sign,
      data: FormData.fromMap(form_data));

  String error_msg = '';
  Map ret_data = jsonDecode(response.toString());
  if (ret_data['status'] == 200) {
    Map<String, dynamic> data = ret_data['data'];
    if (prefs != null) {
      prefs!.setString("user_id", "${data['id']}");
      prefs!.setString("user_email", data['email']);
      prefs!.setString("user_password", data['password']);
    }
    return 'success';
  }
  error_msg = ret_data?['message'];
  return 'Error : $error_msg';
}
```

```
static Future<String> loginUser(String email, String password) async {
  //verify::
  Map<String, dynamic> form_data = {'email': email, 'password': password};

  var response = await dio.post(api_endpoint_user_login,
      data: FormData.fromMap(form_data));

  print(response);
  String error_msg = '';
  Map ret_data = jsonDecode(response.toString());
  if (ret_data['status'] == 200) {
    Map<String, dynamic> data = ret_data['data'];
    if (prefs != null) {
      prefs!.setString("user_id", "${data['id']}");
      prefs!.setString("user_email", data['email']);
      prefs!.setString("user_password", data['password']);
    }
    return 'success';
  }
  error_msg = ret_data?['message'];
  return 'Error : $error_msg';
}
```

```
class UserAuthentication {
 static Future<User?> getLoggedUser() async {
   String? uid = prefs?.getString("user_id");
   String? email = prefs?.getString("user_email");
   String? password = prefs?.getString("user_password");
   if (uid != null &&
       email != null &&
       password != null &&
       uid.isNotEmpty &&
       email.isNotEmpty &&
       password.isNotEmpty) {
     return User(
         uid: uid, name: 'Your name', email: email, password: password);
   } else
     return null;
 }
```

User Authentication for Flutter Apps

- **Implementing Users' Authentication using Email and Password**
  - Flutter Code

How to implement user logging out ?

User Authentication for Flutter Apps

- **Supabase Authentication instead ? (Or even Firebase Auth)**
  - You have two options:
    - Let your App communicates with the Supabase/Firebase Auth to signup/login users **directly**
      - OR
    - Your app talks to Endpoints which communicate to 3rd Party authentication services.

User Authentication for Flutter Apps

- **Supabase Authentication instead ? (Or even Firebase Auth)**
  - You have two options:

```
response=supabase.auth.sign_up({"email": email,"password": password})
```

    - OR
      - **Your app talks to Endpoints which communicate to 3rd Party authentication services.**

Section 3

Firebase Services, Part 1

Firebase Services : Introduction

- **Firebase is :**
  - **“**a Backend-as-a-Service (BaaS) app development platform that provides hosted backend services such as a realtime database, cloud storage, authentication, crash reporting, machine learning, remote configuration, and hosting for your static files.**“**

https://docs.flutter.dev/data-and-backend/firebase?

Firebase Services : Introduction

- **Services Provided by Firebase :**
  - Messaging
  - Remote Config
  - Database (NoSQL)
  - File Storage
  - Authentication
  - Machine Learning
  - Analytics
  - Functions
  -

Available plugins

Product	Plugin name	iOS	Android	Web	Other Apple (macOS, etc.)
Analytics	firebase_analytics	✓	✓	✓	beta
App Check	firebase_app_check	✓	✓	✓	beta
Authentication	firebase_auth	✓	✓	✓	beta
Cloud Firestore	cloud_firestore	✓	✓	✓	beta
Cloud Functions	cloud_functions	✓	✓	✓	beta
Cloud Messaging	firebase_messaging	✓	✓	✓	beta
Cloud Storage	firebase_storage	✓	✓	✓	beta
Crashlytics	firebase_crashlytics	✓	✓		beta
Dynamic Links	firebase_dynamic_links	✓	✓		
In-App Messaging	firebase_in_app_messaging	✓	✓		
Firebase installations	firebase_app_installations	✓	✓	✓	beta
ML Model Downloader	firebase_ml_model_downloader	✓	✓		beta
Performance Monitoring	firebase_performance	✓	✓	✓	
Realtime Database	firebase_database	✓	✓	✓	beta
Remote Config	firebase_remote_config	✓	✓	✓	beta

List of all firebase plugins for flutter : https://firebase.google.com/docs/flutter/setup?platform=ios#available-plugins

Firebase Services : Introduction

- **Steps to get Started**
  - Install the Firebase CLI :
    - https://firebase.google.com/docs/cli#install-cli-mac-linux
  - Link your Firebase Account
    - **firebase login**
  - Follow all instructions at:

    https://firebase.google.com/docs/flutter/setup?platform=ios

Lecture Demo Apps

- MVP for User Login/Signup
  - https://www.dropbox.com/scl/fo/bom5d3ym1194goed92bj7/h?rlkey=6qxuxmdde5kw5qy3jkj9i8tee&dl=0
- Flask Endpoint API without Database on Vercel
  - https://www.dropbox.com/scl/fo/euj0j7eznw7rt0d8aujwa/h?rlkey=fi2t6ghkhoy1nwekahwe6uzg8&dl=0
- Flask Endpoint API on Vercel with Supabase Database
  - https://www.dropbox.com/scl/fo/ktsg4151hwdhdq0e2n84h/h?rlkey=54lavf63p7gkmx01gurtxtn16&dl=0
- Flask Endpoint for User Authentication with PostgreSQL/Supabase
  - https://www.dropbox.com/scl/fo/d415mbetwnw746hlof6yl/h?rlkey=dnmmj8b4ujgwnrvs4x58mqi33&dl=0
- Flutter APP with Authentication (Signup/Login) using Endpoints
  - https://www.dropbox.com/scl/fo/hfbqekrc98ozdn2m64luc/h?rlkey=x2hr3oozn7mjy7ndoloscd3l3&dl=0

Resources

- https://www.prisma.io/dataguide/serverless/serverless-comparison
- https://flask.palletsprojects.com/en/3.0.x/quickstart/
- https://firebase.google.com/docs/firestore/billing-example#small-50k-installs
- https://medium.com/google-developer-experts/firebase-authentication-flutter-80e8f00338ac
- https://firebase.flutter.dev/docs/auth/usage/
- https://www.youtube.com/watch?v=u52TWx410U4&ab_channel=Droidmonk