

## **File Naming Web Application with Python and SQL**

**Subject:** Python and SQL: intro / SQL platforms

**Professor:** Dr. Robert Wojciechowski

**Authors:** Erim Çelen and Lashari Gochiashvili

**Warsaw, 2020**

# Contents

Introduction .....	3
Problem.....	3
Solution.....	4
Technology .....	4
Database Scheme.....	5
Web App.....	7
Web App - STEP 1: .....	8
Web App - STEP 2: .....	8
Web App - STEP 3: .....	9
Web App - STEP 4: .....	9
Web App - STEP 5: .....	10
Work Distribution.....	12
Conclusions .....	14
References .....	14

# Introduction

The purpose of this project is to create a web application that will help employees of organizations name all digital files according to their company naming convention. Project work is executed by students of Master of Data Science Erim Çelen and Lashari Gochiashvili during the class project in “Python and SQL: intro / SQL platforms” under the instructions of Dr. Robert Wojciechowski.

## Problem

Today digital files are generated every minute. Most of the organizations producing enormous brochures, database files, photoshop files, email campaigns, e-forms and etc. If we do not name these materials orderly, we will have a mess and we will struggle to find them whenever we need it. Some of the teams already have file naming conventions. This convention is standard for everyone in the whole organization.

Naming files you dont care about:

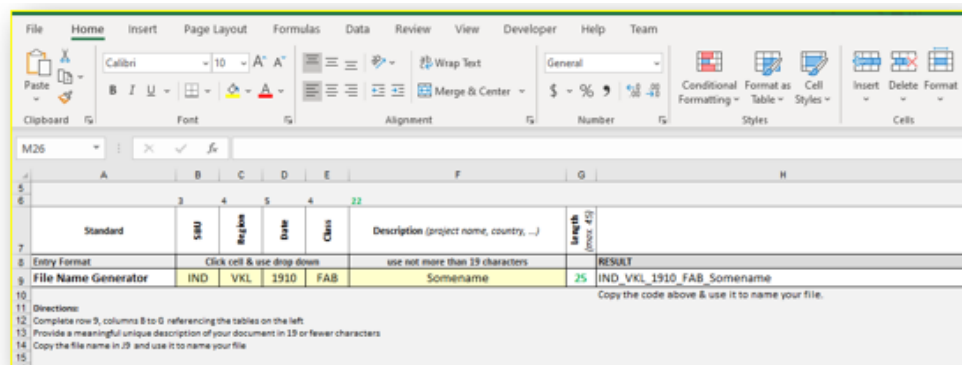


u7gtli8iguuuu

Source: <https://me.me/i/naming-files-you-dont-care-about-u7gtli8iguuu-f71c719385f744bca2bc5dbfa9391b40>

**Picture 1.**

There are a couple of tools people use. Such as “naming tools” built-in excel files. Here come versioning issues when the update in picklists is necessary. Also, sometimes people forget where the excel file is located. Especially the newest one.

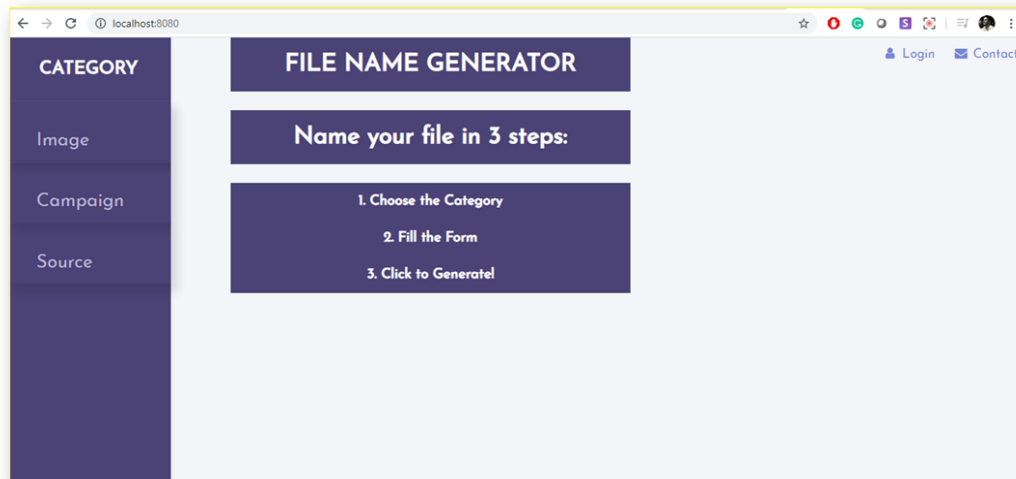


Source: my file with changed names

**Picture 2.** Excel built-in file naming tool

# Solution

We decided to tackle the above-mentioned problem and create a user-friendly web app to name any type of file that will be used by all employees to keep the consistency. The app should give the possibility to a limited number of admins to login and change the values in the picklists when applicable. The solution will have back-end in Python, we will use the Flask web framework and SQLite for databases.



**Picture 3.** File naming web application

# Technology

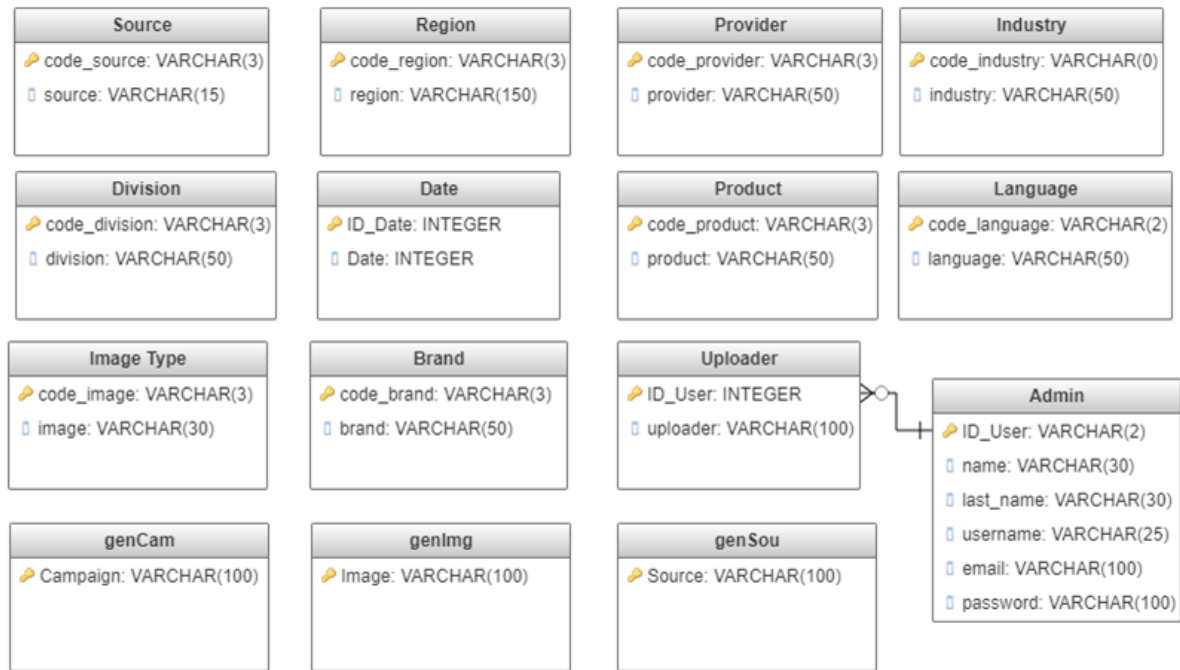
In our case, we used Python for back-end programming, Flask framework for web development, SQLite3 for database, CSS, HTML, and bootstrap for web app front-end. It is worth to mention and our team did not have any experience in any of the mentioned technologies. Our knowledge was based on attending the class of Intro in Python and SQL and reading online documentation on relevant online tech communities. Such as stackoverflow.com, python.org, quora.com and etc. This web site is a great first step for us in data-driven web applications developed in Python & SQL. From the beginning, we started using the Bottle web framework but due to the lack of documentation and community, we moved to Flask.



**Picture 4.** Technologies used in our web application

## Database Scheme

The database used in our application contains 15 tables shown in **Scheme 1** below. 11 tables are for the picklists in the form when the user fills the form. These tables have a unique field ID that is the primary key with VARCHAR (3) type. Language table ID is VARCHAR (2). The second columns mostly containing the values for picklists. Type is VARCHAR (15), (30), (50), (100) and (150). The 3 tables are for storing generated names genCam for campaign names, genImg for image names, and genSou for source names. These tables have only one column that contains saved values. This is a VARCHAR (100) type column. The Admin table is connected with Uploader table with the ID\_User field. Admin table contains ID\_User field with VARCHAR (2), name VARCHAR (30), last\_name VARCHAR (30), username VARCHAR (25), email VARCHAR (100), and password VARCHAR (100).



**Scheme 1.** Database scheme for web application

The database is run from Python notebook (database.ipynb). After importing SQLite3 in Python notebook and connecting to the database file “database.db” then we create a cursor “c = conn.cursor()”. These operations we need to execute SQL queries such as:

**# creating tables (one example):**

```
c.execute("CREATE TABLE Admin (ID_User VARCHAR(2)primary key, Name VARCHAR(30), Surname VARCHAR(30), Username VARCHAR(25), [Email] VARCHAR(100), Password VARCHAR(100))")
```

**# creating lists (one example):**

```
listad = [('LG','Lasha','Gochiashvili','lasha','l.gochiashvili@student.uw.edu.pl','lasha123'),
          ('EC','Erim','Celen','erim','m.celen@student.uw.edu.pl','erim123')]
```

**# adding lists into tables (one example):**

```
c.executemany("INSERT INTO Admin VALUES (?,?,?,?,?)", listad)
```

**# commit to database and closing connection:**

```
conn.commit()
conn.close()
```

We also use query “SELECT” to pull password for the username when the user is trying to log in. In this way we have a function that compares the password:

```
c.execute('SELECT * FROM Admin WHERE Username = ?', [user_name])
data = c.fetchone()
password = data[5] # password is stored in column with index 5.
if (password_candidate == password):
    print('Password matched')
else:
    print('Password not matched')
    return render_template('login.html')
```

We also add new users in the database by running the query “INSERT INTO” table:

```
c.execute('INSERT INTO Admin(ID_User,Name,Surname,Username,Email>Password)
VALUES(?,?,?,?,?,?)', (id,name,surname,username,email,password))
```

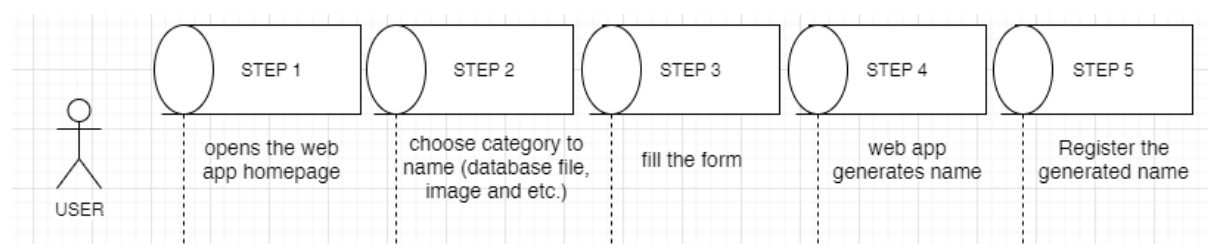
Saving results into SQL database after generate function:

```
c.execute('INSERT INTO genImg(Image) VALUES (?)', [generatedResult])
```

## Web App

In our web application, we have very simple steps for users. To name the file user should choose the category and select variables from the dropdown menu then hit the generate button. There are three categories on the sidebar and eight variables in the generator form. One out of eight variable is a text input as a unique description in the form to fill.

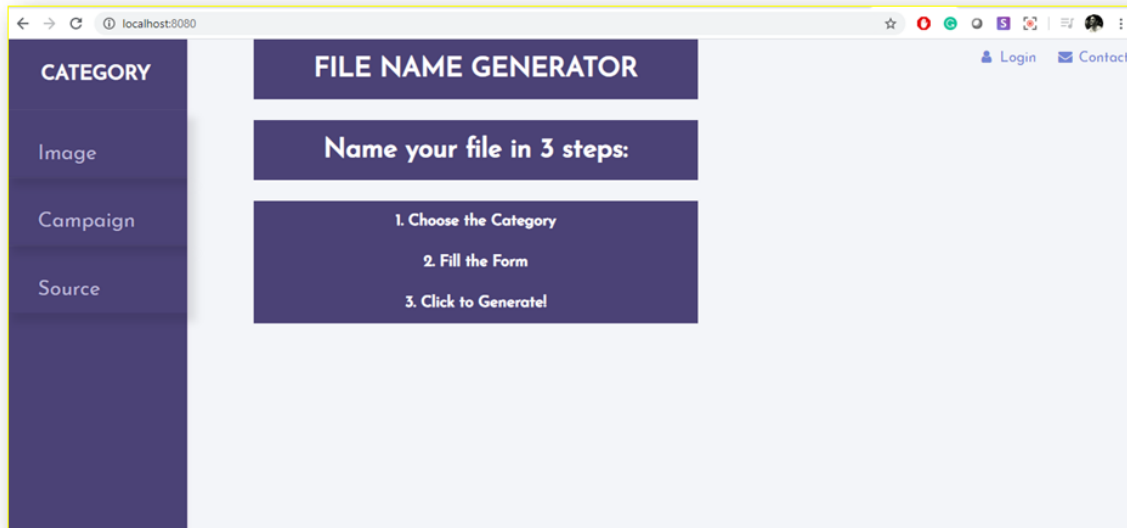
After creating generated names user can register names to the database for statistics. **Scheme 2** is the best description of an average user path on our web application.



**Scheme 2.** User’s path in the web application

## Web App - STEP 1:

The user opens the web app homepage. Ideally, the web app is linked to the internal use website where only one organization employees can access it. As the web app is using company-specific dropdown values they should be customized by the admin according to the host company naming convention.



**Picture 5.** The main page of our website

## Web App - STEP 2:

Depending on the need user can choose category of the file he or she wants to name. As we used the marketing team's problem to solve with our app, we created three categories Image, Campaign, and Source. These are main assets marketing teams are using daily basis to name. Each category has its form to be filled for name generation as shown in **Picture 6**.

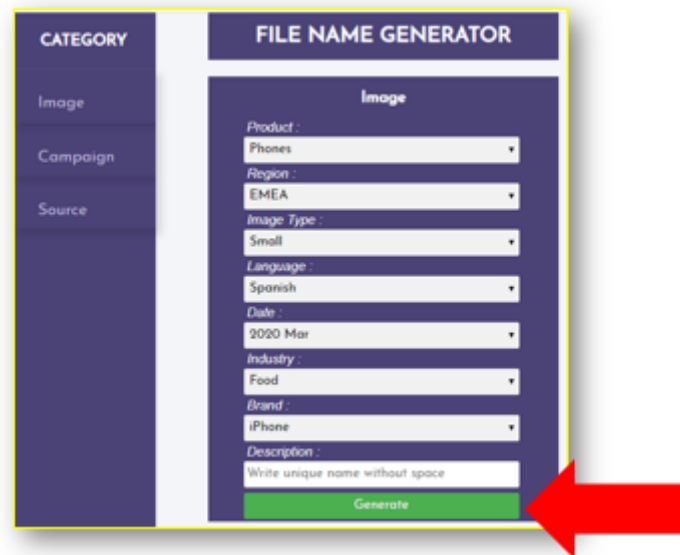


**Picture 6.** Categories and the relevant forms



### Web App - STEP 3:

After choosing category, a user is selecting fields from the dropdown menu that describes the file the best. Seven fields to be selected and the eighth should be filled with a unique name.



CATEGORY	FILE NAME GENERATOR
Image	<p><b>Image</b></p> <p>Product : Phones</p> <p>Region : EMEA</p> <p>Image Type : Small</p> <p>Language : Spanish</p> <p>Date : 2020 Mar</p> <p>Industry : Food</p> <p>Brand : iPhone</p> <p>Description : Write unique name without space</p> <p>Generate</p>
Campaign	
Source	

**Picture 7.** Form for image naming

### Web App - STEP 4:

After clicking “Generate” in the previous step, the web app generates name by concatenating names in the form that is filled by the user. Now user can copy the generated name and use it to name the file.

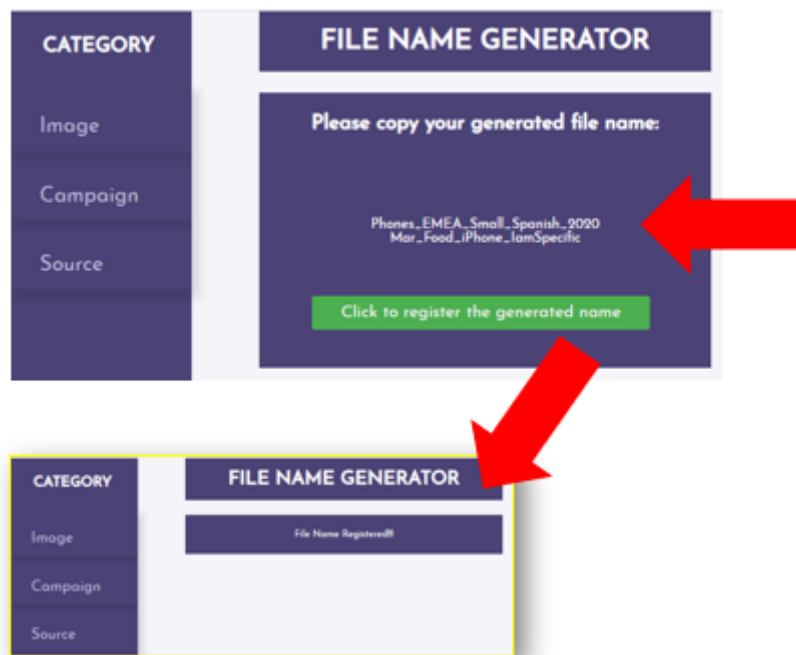


CATEGORY	FILE NAME GENERATOR
Image	<p>Please copy your generated file name:</p> <p>Phones_EMEA_Small_Spanish_2020 Mar_Food_iPhone_IamSpecific</p> <p>Click to register the generated name</p>
Campaign	
Source	

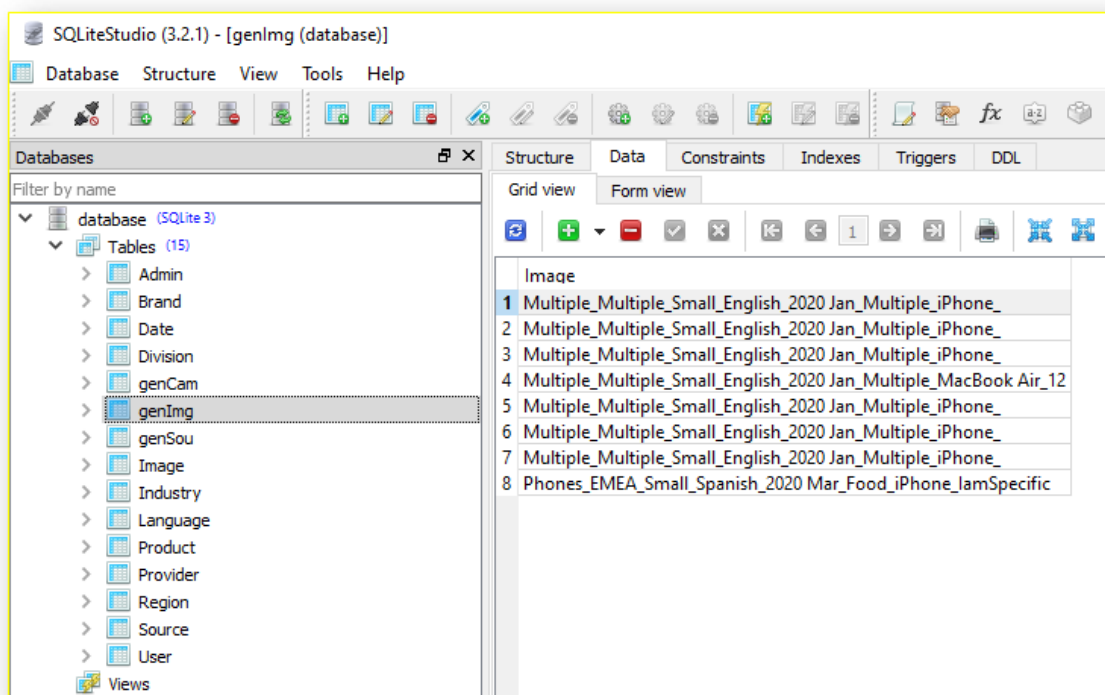
**Picture 8.** Generated image name

## Web App - STEP 5:

For statistics, the user can click the button and register the name in the database. Admin can see generated names in the database that were registered by users.

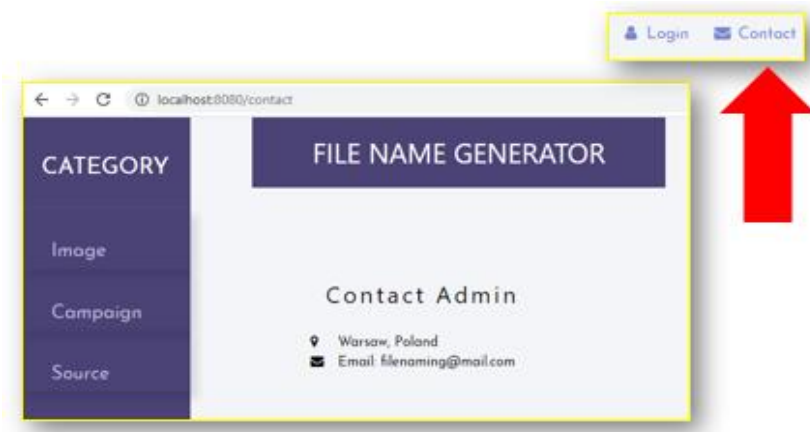


**Picture 9.** User clicked the register button to register the generated name



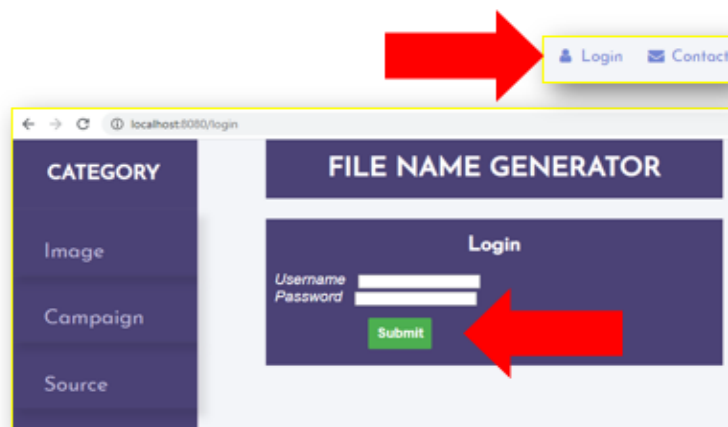
**Picture 10.** Generated image names in the database

Users can contact an admin in case of any issues by going to the contact webpage.



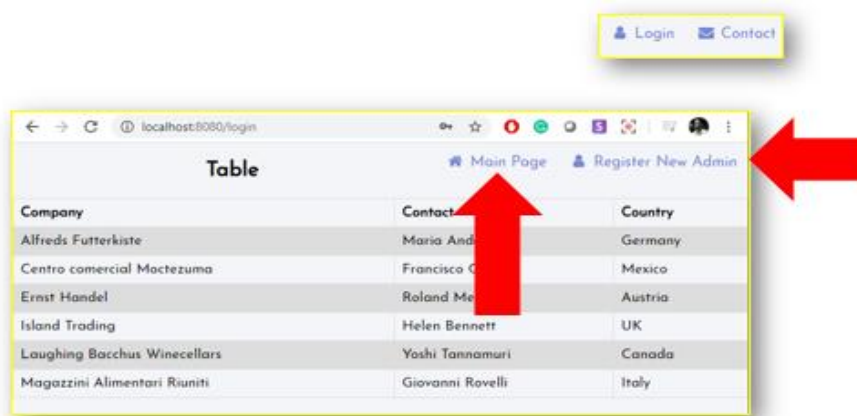
Picture 11. Contact page

A web application will have a couple of admins. Admin can log in to the web app and change the picklist after implementing additional changes to our web app admin panel.



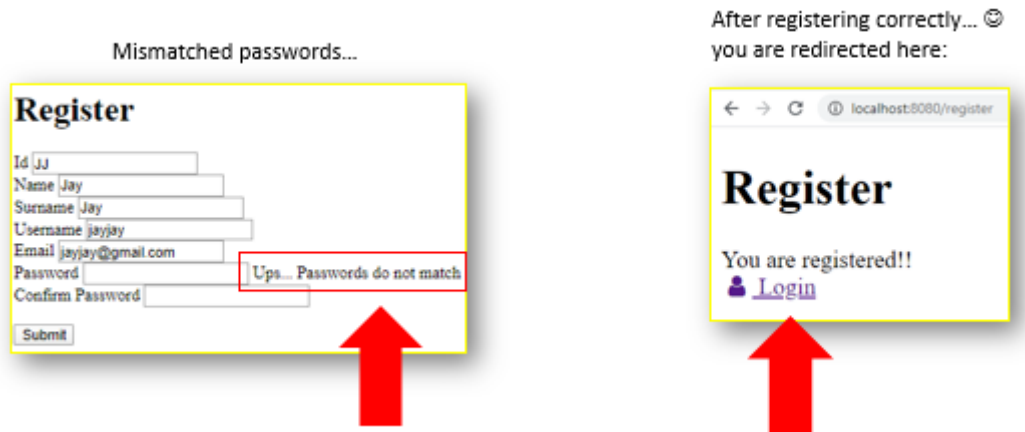
Picture 12. Login page

Admin can register other admins and go to the main page from the admin panel.



Picture 13. Admin panel

After registration admin can log in and go to the login page. Registration form validation is given in the below picture.



**Picture 14.** Registration form

## Work Distribution

The work plan is given in the below table. Almost all part of the work was equally split from the beginning to the end. Work included planning, preparation, materials review, documentation reading, watching youtube videos, community blog reviews, reviewing best practices and may others. Also, during the project, we worked individually on our parts and also we had teamwork on the challenging parts that the individual was not able to resolve. The project taught us the importance of teamwork, preparation, perseverance and focusing on solving problems whatever it takes. All this with a cheerful and smiley mood.

RESPONSIBLE PERSON	
MEÇ	Mehmet Erim Çelen
LG	Lashari Gochiashvili

#	ACTION	RESPONSIBLE
<b>1 Proposal</b>		
1.1	Brainstorming about ideas I	MEÇ
1.2	Brainstorming about ideas II	LG
1.3	Mockup of final idea I	MEÇ
1.4	Mockup of final idea II	LG
1.5	Work plan breakdown I	MEÇ
1.6	Work plan breakdown II	LG
1.7	Writing proposal I	MEÇ
1.8	Writing proposal II	LG

<b>2 Web app</b>	
2.1 Login	LG
2.2 Registration	MEÇ
2.3 Web interface - categories	MEÇ
2.4 Web interface - forms	MEÇ
2.5 Web app - connecting categories with forms	MEÇ
2.6 Web app - name generator	MEÇ
2.7 Web interface - save/register button for generated name	MEÇ
2.8 Create data for each table - in scheme and SQL	LG
2.9 Create data for each table - in python	MEÇ
2.10 Routing in python I	MEÇ
2.11 Routing in python II	LG
2.12 HTML templates I	MEÇ
2.13 HTML templates II	LG
2.14 CSS I	MEÇ
2.15 CSS II	LG
2.16 Test web app	LG
2.17 Create environment to run web app	LG
<b>3 Report</b>	
3.1 Introduction	LG
3.2 Databases scheme with description	LG
3.3 User manual describing the use of application with screenshots	MEÇ
3.4 Final SQL queries	MEÇ
3.5 How work was distributed among team members	LG
<b>4 Presentation</b>	
4.1 Slides about introduction	LG
4.2 Slides about database scheme	LG
4.3 Demo of final app	MEÇ
4.4 Slide about how work was done	LG

**Table 1.** Work distribution

# Conclusions

Real problem with its solution identified - it is very encouraging that our solution can solve a problem that is real in companies today. This could be an example for us on how we can tackle the problems we see around us.

We learned lots of technologies while working on building the solution. We heard stories about from zero to hero. This story became a little real for us as we experienced how a team from scratch to create something with zero knowledge. It was a great journey for us to create something useful and learn a lot of programming, frameworks, libraries, etc.

Without good community and documentation, it is not easy to use even “cool” technology. Due to this problem, we switched “bottle” with “flask”. We learned a lesson that in the future we will choose the technology that is well supported by the community and the internet is full of documentation or best practices.

Teamwork is key. Great things in the world are not done by individuals but by teams.

Lots of areas to develop for this web app. E.g. admin panel, better front end and etc. There are plenty of work arounds to be done to make this web app much better.

# References

Most of the knowledge we acquired from Stackoverflow.com & youtube.com

<http://flask.palletsprojects.com/en/1.1.x/>

<https://www.youtube.com/watch?v=MwZwr5Tvyxo>

<https://wiki.python.org/moin/WebFrameworks>

Python Flask From Scratch series: <https://www.youtube.com/watch?v=zRwy8gtgJ1A>

<https://jinja.palletsprojects.com/en/2.10.x/>

[https://www.w3schools.com/w3css/w3css\\_examples.asp](https://www.w3schools.com/w3css/w3css_examples.asp)

[https://training.talkpython.fm/courses/explore\\_flask/building-data-driven-web-applications-in-python-with-flask-sqlalchemy-and-bootstrap](https://training.talkpython.fm/courses/explore_flask/building-data-driven-web-applications-in-python-with-flask-sqlalchemy-and-bootstrap)

<https://flask-wtf.readthedocs.io/en/stable/install.html>

<https://www.palletsprojects.com/p/flask/>

<http://bottlepy.org/docs/dev/>

<https://flask-wtf.readthedocs.io/en/stable/>

<https://hackersandslackers.com/flask-routes/>

<https://www.sqlite.org/docs.html>