University Applications Manager

WEB APPLICATION USING PYTHON AND SQLITE

Tesfahun Tegene Boshe & Yenish Nurmuhammedov
UNIVERSITY OF WARSAW | DATA SCIENCE AND BUSINESS ANALYTICS/ MASTERS

Introduction

Setup

This web App was created based on the tutorial at Flask documentation.

The following steps were followed:

Create the application factory

A function with a flask instance was created.

Run the application.

Once the application factory was created, we run it on command prompt using the flask command.

```
$ export FLASK_APP=flaskr
$ export FLASK_ENV=development
$ flask run
```

Define and access the database.

As instructed in the requirements material, we used SQLite database. Python's sqlite3 module was used to create and connect the connection to the database. Package g was used to store data accessed by multiple functions during the request.

Create the tables.

A .sql file was created with commands to create all necessary tables. The file was executed using 'init_db' function in 'db .ipynb' file.

The database file was initialized after this.

Blueprints and Views

We used Blueprint module to create an organized and group related views and other code. Two blueprints were defined: auth and blog.

Auth Blueprint and views

- Register – registers a new user to the database
- Login – verifies user information and creates a session for a logged in user.
- Logout – clears the session. Frees up the memory.

Blog Blueprint

- Home – redirects the logged in user to the home page.
- Applications – displays the most recent application of the user.
- Get_post – a function for querying the data from database.
- Applynow – redirects to applynow.html file. This is where the user makes first draft of his application.
- Update – redirects to the page where the user can update his current application.
- Table – redirects to the page with applicants table and countries table.
- Profile – this returns the profile page for the logged in user.

- Delete – deletes the user input data from the database.
- Admin – the admin is able to see all applicants and grade the applications, as well as give final decisions.

Endpoints and URLS

Functions in the Blueprint usually redirect the user to a certain page, using redirect(url_for()) method.

Templates

Templates contain static data as well as placeholders for dynamic data.

Jinja template library is used to render templates.

- Base Layout: has the shared features of different templates. This helps us to write shorter html files and override specific sections.
- Other HTML files: all pages listed below have their html codes in templates folder.
- Static Files: the static files are saved in static folder. Each static file is associated with at least one static file.

Install the project.

Tis builds a distributable file to install in another environment. Setup. Ipynb and MANIFEST.in files handle installation. We used pip install to install the project in the virtual environment.

Deploy the app.

We hosted our web app at https://www.pythonanywhere.com/

The project layout

```
/WebApp
├── flaskr/
│   ├── __init__.py
│   ├── db.py
│   ├── schema.sql
│   ├── auth.py
│   ├── blog.py
│   ├── templates/
│   │   ├── base.html
│   │   ├── auth/
│   │   │   ├── login.html
│   │   │   └── register.html
│   │   └── blog/
│   │       ├── home.html
│   │       ├── applynow.html
│   │       └── applications.html
│   │       └── update.html
│   │       └── table.html
│   │       └── admin.html
│   │       └── profile.html
│   └── static/
│       └── admin.css
│       └── home.css
│       └── style.css
│       └── profile.css
│       └── table.css
│
├── venv/
├── setup.py
└── MANIFEST.in
```

Common Distribution of Work

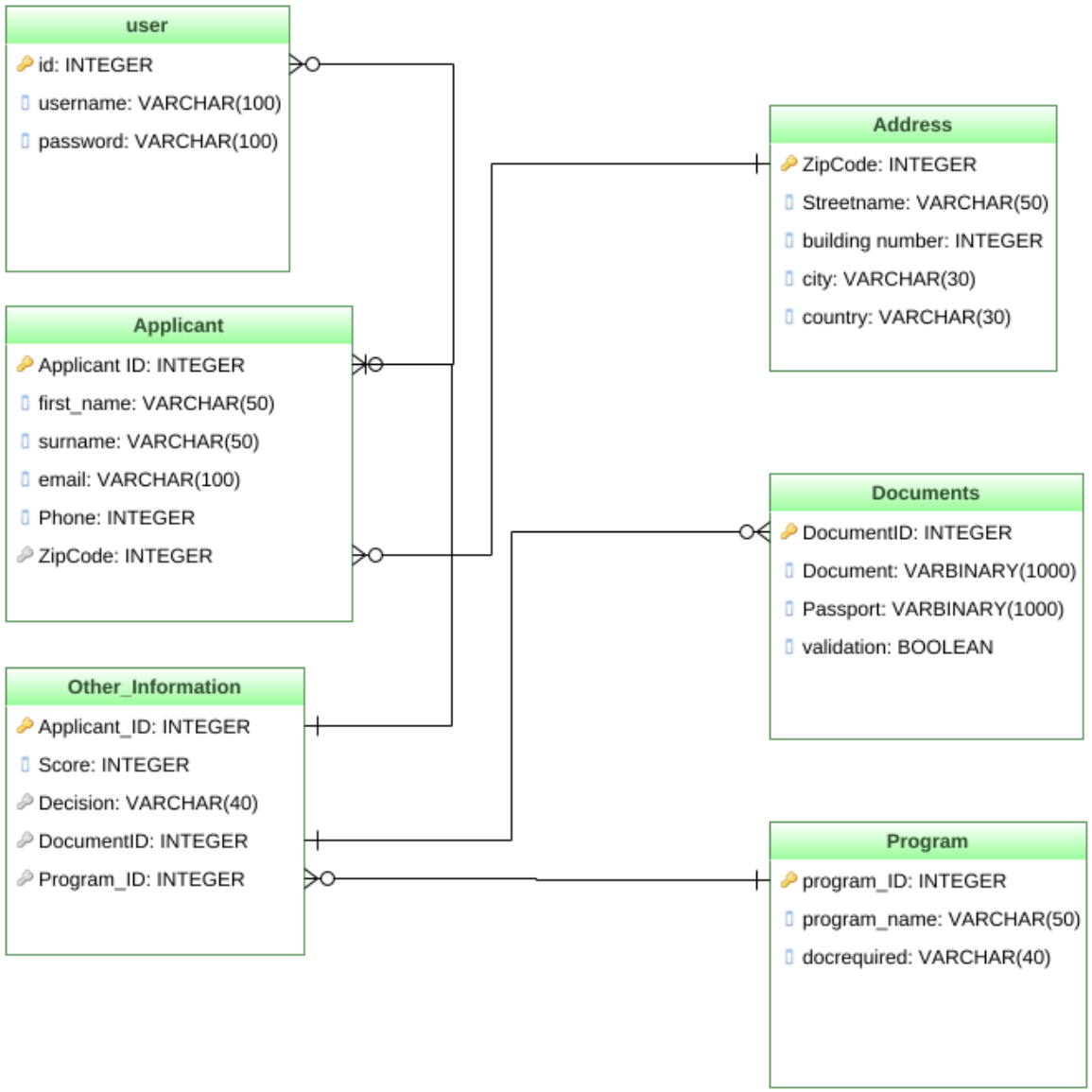
Decide Project Idea
Decide the Database Model
Design and Model
Decide the Queries

Distribution of Work

Tesfahun Tegene Boshe	Yenish Nurmuhammedov

Log in Page	Home Page
Log out Page	Profile Page
Sign up Page	Databases Model
Apply now Page	Queries
Connect Databases to the Website	
Queries	

Database scheme



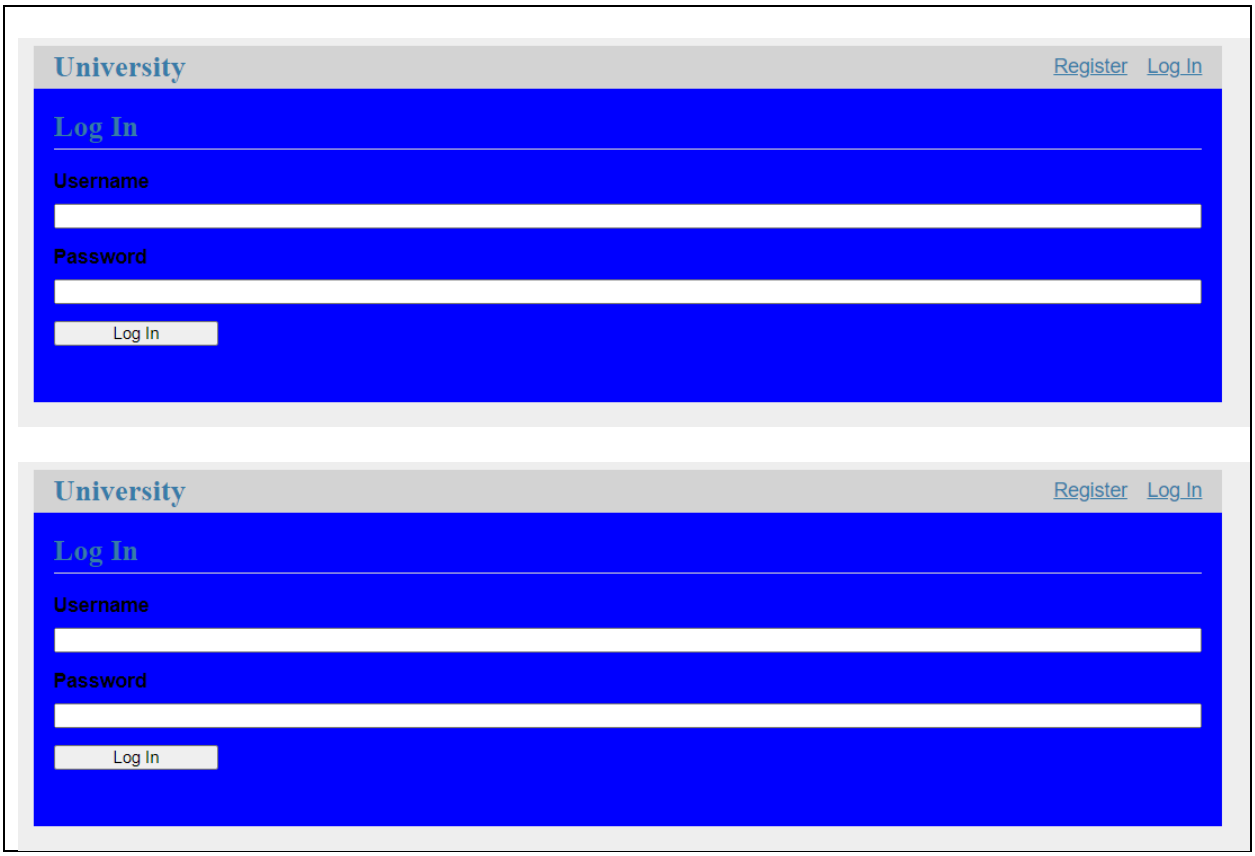
User Manual

Please follow these steps to get full advantage of our web app.

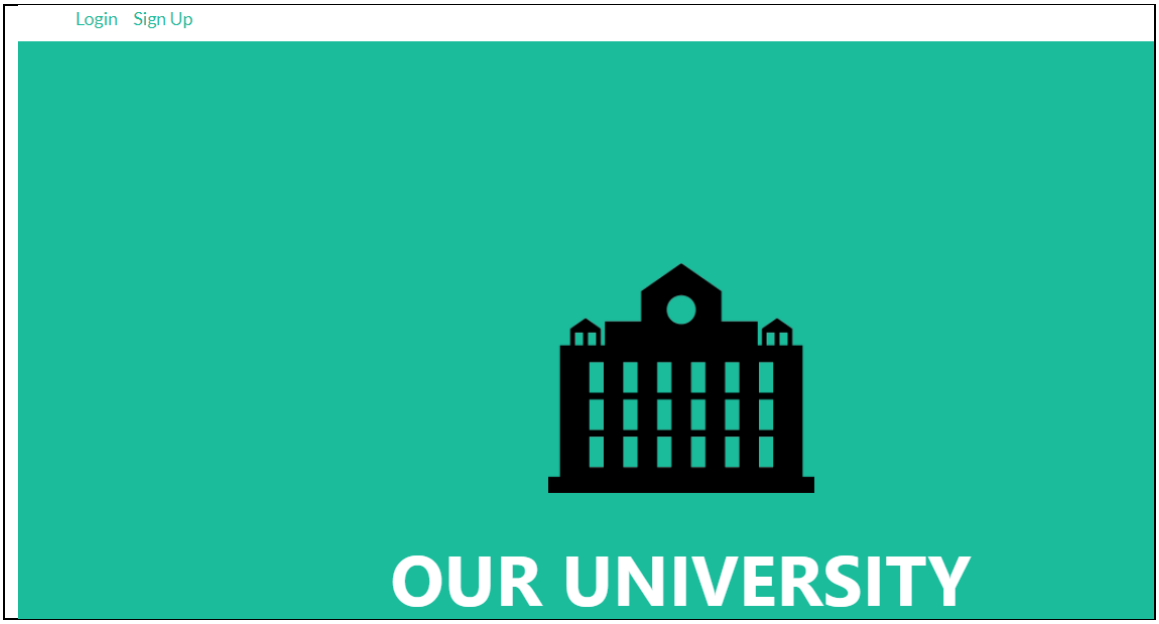
1. Click this link or type it in your browser.
2. Register
3. Login using the same credentials as register.
4. When you successfully login, you are redirected to home page where you have more options.
    1. To add an application: click ‘Apply now’ on top menu.
        a. In the new page, enter all necessary information and click submit.
        b. It directs you to update page where you can edit your applications.
        c. On the same page, you are also able to delete the application.
    2. To view statistics of all applicants:
        a. Click Data Table on the top menu. You should be able to see applicants table and countries table.
        b. Click ‘Home’ to come back to home page.
    3. If you are an admin user, you will have one more icon ‘Admin’ on the menu bar. This should open a table with all applicant information. Edit the last two columns ‘Score’ and “Decision’ and submit. This should update the database.
5. See your profile on “Profile” menu. Your profile has the latest updates of admin in addition to your user information.
6. Logout when you are done.
7. You can repeat the same process logging again with the same username or creating another account with a different username.

Images from the application

1. Login/register pages

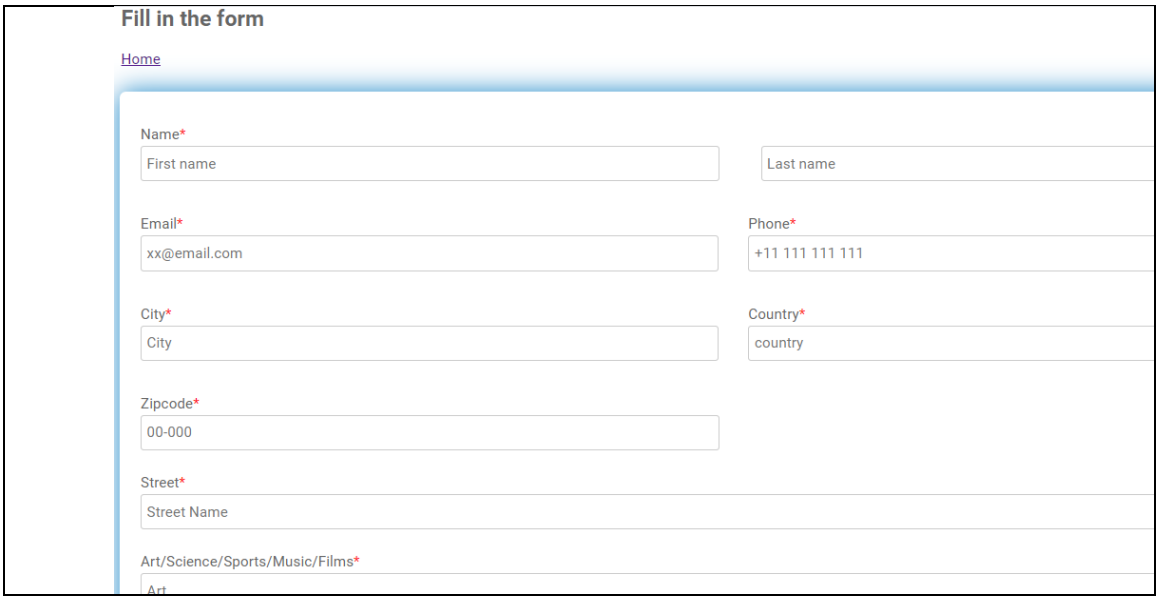

University
Register
Log In
Log In
Username
Password
Log In
University
Register
Log In
Log In
Username
Password
Log In

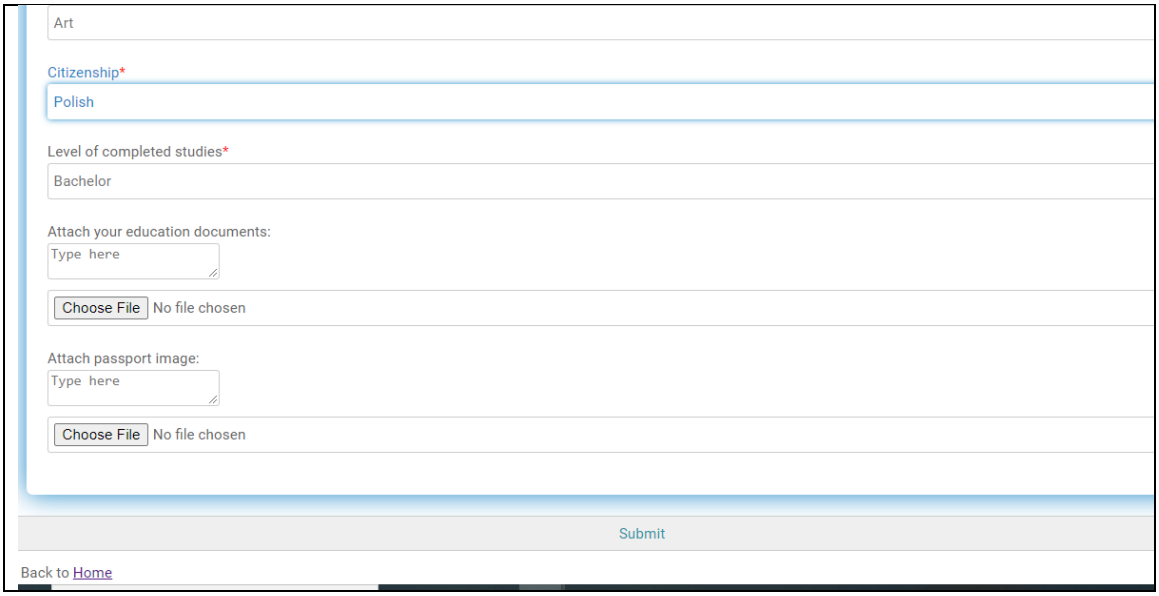
2. Home page


Login
Sign Up
OUR UNIVERSITY

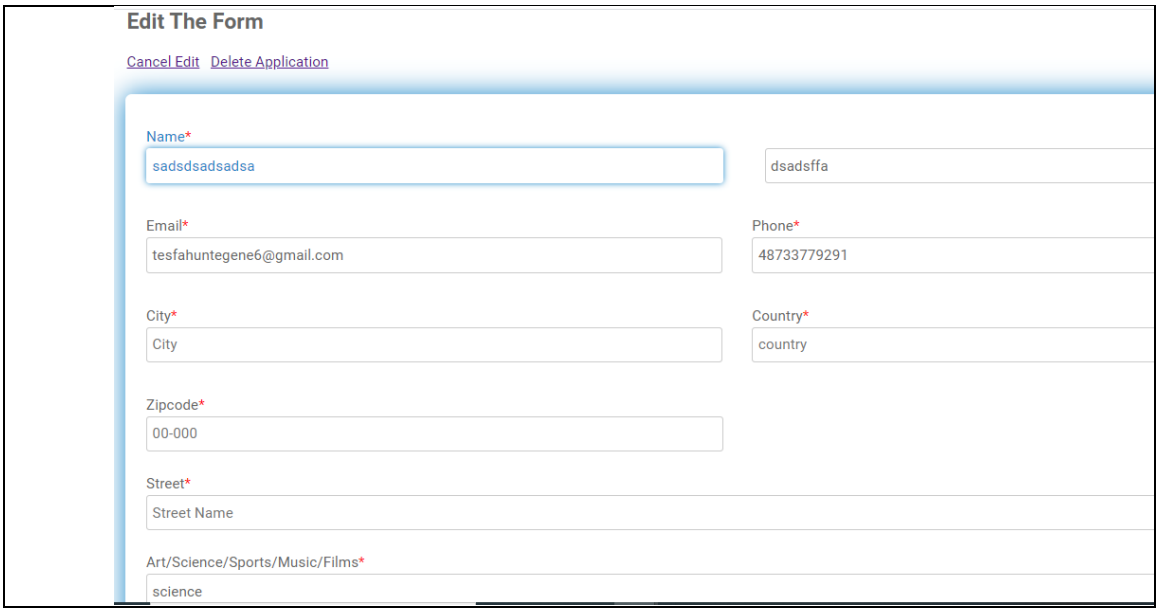
3. Admin Home page

4. Apply now.

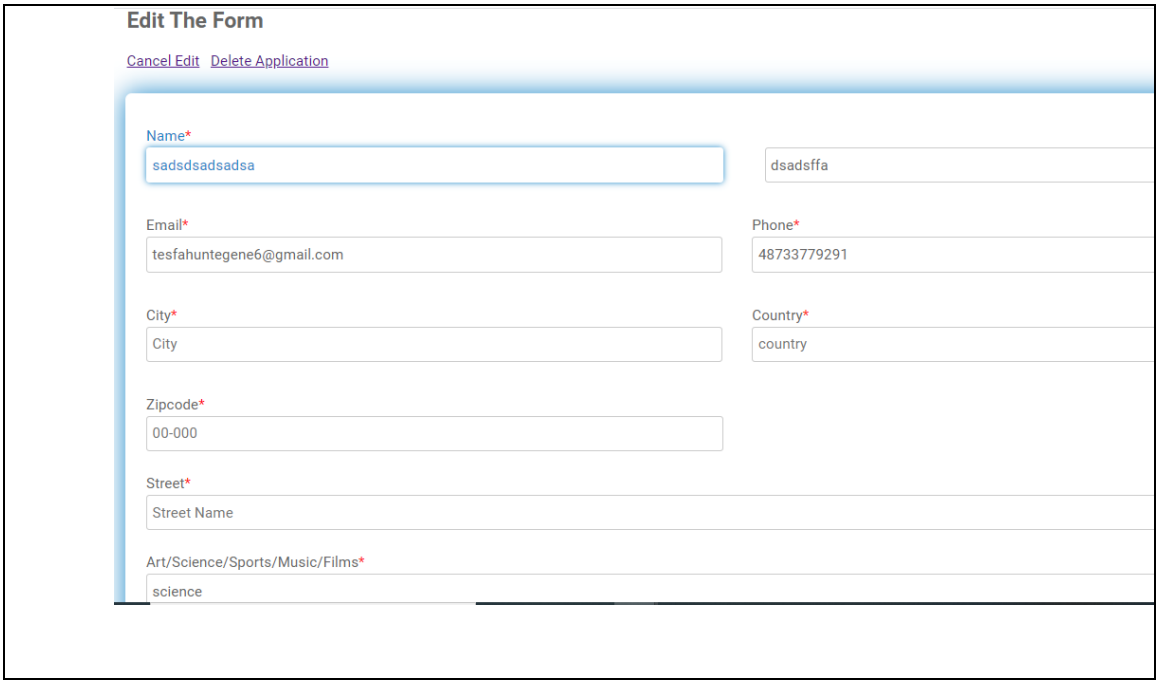




5. Applications



6. Edit applications.



7. Data Table

Email	Last Name	First Name	Count
tesfahuntegene6@gmail.com	BOSHE	TESFAHUN	Poland
tesfahuntegene56@gmail.com	dfsdfsfs	fdssfsdf	Poland
tesfahuntegene6@gmail.com	dsadsffa	sadsdsadsadsa	Poland

Back To Home

8. Admin

Applicants Table

Email	Applicant ID	Program	Document	Passport	Score	Decison
aaa	1	aaa	chrome links.txt	chrome links.txt	100	PASS
bbb	2	bbb	chrome links.txt	chrome links.txt	100	PASS

Submit

Back To Home

9. Profile

Name*
sadsdsadsadsa
dsadsffa

Email*
tesfahuntegene6@gmail.com

Phone*
48733779291

Score*
100

Decision*
PASS

Back To Home

SQL queries

These and similar queries were used. Some of them shortened on this document.

1. Create tables:

We created 6 tables, using commands like this. This one creates a table called Applicant with columns (applicantID, fistname,lastname,email,phone,zipcode) and a foreign key zipcode connects it to address table.

```
CREATE TABLE Applicant (
  applicantID INTEGER PRIMARY KEY AUTOINCREMENT,
  firstname TEXT NOT NULL,
  lastname TEXT NOT NULL,
  email TEXT NOT NULL,
  phone INTEGER NOT NULL,
  zipcode INTEGER NOT NULL,
  FOREIGN KEY (zipcode) REFERENCES Address (zipCode)
);
```

2. Select columns, entries

```
db = get_db()
students = db.execute(
        "SELECT firstname, lastname, email,phone,A.zipcode,applicantID"
        " FROM Applicant A JOIN user u ON A.applicantID = u.id"
  ).fetchall()
```

Select from multiple tables

```
dataTable = db.execute(
        "SELECT firstname, lastname, email,country,programName"
        " FROM Applicant"
        " INNER JOIN Address ON applicantID = AddressID"
        " INNER JOIN Program ON applicantID = programID"
    ).fetchall()
```

Select calculated values

```
CountryData = db.execute(
        "SELECT country, count(*) as N_applicants, max(score) as maximumScore, min(score) as
minimumScore"
        " FROM Address"
        " INNER JOIN OtherInformation ON AddressID = applicantID"
    ).fetchall()
```

Insert into tables

```
db = get_db()

db.execute(

            "INSERT INTO Applicant (firstname, lastname,
email,phone,zipcode,applicantID) VALUES (?, ?, ?,?, ?, ?)",

            (firstname, lastname,email, phone,zipcode, g.user["id"]),
        )
```

Update tables

```
db = get_db()
        db.execute(
            "UPDATE Applicant SET firstname = ?, lastname = ?, email = ?, phone = ?, zipcode = ? WHERE applicantID = ?", (firstname, lastname, email,phone,zipcode,id)
        )
```

Delete from tables

```
get_post(id)
    db = get_db()
    # db.execute("DELETE FROM post WHERE id = ?", (id,))
    db.execute("DELETE FROM Applicant WHERE applicantID = ?", (id,))
    db.commit()
```

Create views

```
dataAdmin = db.execute(
        "CREATE VIEW dataAdmin AS SELECT firstname, lastname, email,country,programName,document,passport,score,decision,a.applicantID"
        " FROM Applicant a"
        " INNER JOIN Address ON a.applicantID = AddressID"
        " INNER JOIN OtherInformation OI ON OI.applicantID = InfoID"
        " INNER JOIN Program p ON InfoID = p.programID"
        " INNER JOIN Documents d ON d.documentID = p.programID"
    ).fetchall()
```

Create trigger

```
db.execute("Create trigger MyTrigger after delete on Applicants \
    BEGIN DELETE FROM Address; END;“)
```

References

1. https://flask.palletsprojects.com/en/1.1.x/tutorial/
2. https://codepen.io/ace-subido/pen/Cuiep
3. https://colorlib.com/wp/template/fixed-header-table/
4. https://freefrontend.com/bootstrap-profiles/