https://www.csestack.org/render-open-pdf-file-django/

anastasia.karobia@cloudfactory.com

https://sleepy-blackwell-e00fe5.netlify.app/

```
Create an environment in windows and activate\

         python -m venv .venv

Activate environment

        .venv\Scripts\activate(windows)

        source env/bin/activate(linux)

https://realpython.com/django-setup/ Django set up

python manage.py makemigrations

python manage.py migrate

python manage.py createsuperuser

python manage.py runserver

Python commands https://cloud.google.com/python/django/appengine
```

Set Up a postgres db

sudo apt-get install postgresql postgresql-contrib

sudo apt-get install libpq-dev python3-dev

pip install psycopg2

sudo service postgresql restart

https://github.com/wanjugucode/pythonclass

https://github.com/wanjugucode/RegistrationApp
https://github.com/wanjugucode/RegistrationApp
student

sudo -u postgres psql

CREATE DATABASE expresskitchendb;
CREATE USER expresskitchen WITH ENCRYPTED PASSWORD 'AkiraChix2021';

ALTER ROLE expresskitchen SET client_encoding TO 'utf8';
ALTER ROLE expresskitchen SET default_transaction_isolation TO 'read committed';
ALTER ROLE expresskitchen SET timezone TO 'UTC';
GRANT ALL PRIVILEGES ON DATABASE expresskitchendb TO expresskitchen;

sudo apt-get install postgresql postgresql-contrib
sudo apt-get install libpq-dev python3-dev
sudo -u postgres psql
CREATE DATABASE xpresskitchendb;
CREATE USER myuseradmin WITH ENCRYPTED PASSWORD 'passkitchen2021';
ALTER ROLE myuseradmin SET client_encoding TO 'utf8';
ALTER ROLE myuseradmin SET default_transaction_isolation TO 'read committed';
ALTER ROLE myuseradmin SET timezone TO 'UTC';
GRANT ALL PRIVILEGES ON DATABASE xpresskitchendb TO myuseradmin;

Authentication

- **User account**
- **Registration**
- **Account management**
- **permissions**

Django all auth redux

Django packages

Django internalization

Base template

Template inheritance

Http sessions

Trainer and student add another attribute called user which is a onetoone field pointing to user model

From django.contrib.auth.models import User

NULLABLE

```
heroku login

heroku create aisha-akirachix

git remote add heroku

git add -a

git commit -m

git push heroku master

heroku run python manage.py migrate

andrew.kibe@kibandatopup.com
```

```
heroku run python manage.py createsuperuser

heroku ps:scale web=1

heroku open
```

```
python manage.py migrate --run-syncdb

Heroku logs --tail
```

```
heroku git:remote -a expresskitchen
```

```
heroku git:remote -a stock-management-akira

git remote -v

git add -A

git commit -m "deployment"

git push heroku master
```

```
heroku buildpacks:set heroku/python
heroku config:set DISABLE_COLLECTSTATIC=1
```

```
git add .; git commit -m "add requirements.txt"; git push heroku master
```

```
heroku run python3 manage.py createsuperuser
```

Software testing

Process of evaluating the functionality of a software application to ensure it meets the required specification

1. Manual testing
2. Automated testing - we write code that is executed to generate test results

Levels of automated testing

1. Unit testing test whether the individual components of a system are working as specified
2. Integration testing-Test whether components of a system that are supposed to work together are working as expected
3. System testing- test the whole system works as specified
4. User acceptance testing(UAT)-Test whether the user journey is working as specified

Browser stack-allows you test(UAT)Automation of UAT

Importance of software testing

To maintain the product or the system quality as it grows

It makes it easier to make a change or update the system despite the size

It increasing /maintain customer satisfaction

Cost effectiveness -

It helps teams work together effectively as the system grows

Tdd-test dreaming development

CODE COVERAGE-THE CODE U WRITE HOW MUCH IS COVERED BY TESTING

Testing view

Route

Sample data

Simple http request

Asserts the response status code 200

Deployment- Heroku

Database -structured data(tata we can query)

File storage- unstructured data(files,images, videos) binary data

Web server- mutable data

S3- is scalable infinitely

Rds-

Caching-

EC2-

PAAS-Platform as a service(has database, web services)

SRE site reliability engineer

Account

Heroku cli

Dependencies

Add config files

Update our settings

Deploy user interface

Sudo snap install --classic heroku

Sudo apt-get install libpq-dev

Pip install gunicorn

Pip install django-heroku

Procfile

Web: gunicorn schoolsystem .wsgi

Runtime.txt

Python -3.9.9

Requirement.txt

Pip freeze>requirement.txt

Heroku login

(apache, enginex)

Heroku login

Heroku create Anastasia-akirachix

Git remote add heroku

Git add -A

Git commit -m “my first commit”

Git push push heroku master /main

Heroku run python manage.py migrate

Create a database db and connect your project

Create a file storage

Testing-

ci/cd-

RestApi

API-Application programming interface

SOAP, REST, GraphQL- Protocol to create api

Rest- representation state transferer

-Transfer the state of any object

Data from - **XML,JSON, CSV to send**

JSON-Javascript object Notation

CRUD- Create, Read, Update, Delete

Restful-

CLIENT	**RESOURCE**
Browser	**Any object in our data model**
	(we perform a crud mechanism using API)

1. URL - to interact with a resource

2. HTTP method/verb

HTTP METHODS- They are included in the request- to let the server know what to do but by default server acknowledges **get**

GET-Retrieving information about one or more resource

POST- Create one or more resource

PUT- update a resource on the server

Delete- Deletes a resource on the server

Patch- partially updates a resource

HTTP STATUS CODE

Client (request) (URL method) server(resource)

Servers return a response with a status code and data

200- Okay()

201- Resource Created(post)

202- Accepted(delete)

204- No content(get)

300- Redirected to another resource

400- Bad request(post)

401-unauthorized/ no permission(get)

403- Forbidden

404- Resource not found on the server

405- method not allowed

409- conflict(issues with validation/validation did not pass a good API should tell the client where the conflict occur)

500- Server error(when there is a bug in the code(it crushes and return 500))

Django rest framework(companion framework that heavy lifts our work)

1. CREATE A SERIALIZER-**Serializers translates data format to and from JSON**
2. **CREATE A VIEW- i**t accepts and processes the request to a particular resource
3. Create a route/URL route for the view- redirect or response and request to the view

Serializer

- **Retrieving individual objects**
- **Editing data**
- **Shared functionality**

Retrieving individual objects

Types of views

List view- many objects

Detail view- individual object

Detail view that displays data of one student

Free bootstrap templates

Starting point- view page accepts

Good template nav bar

Create detailed view and editing view and template for

Trainer

Course

event

Editing objects

Building mobile applications using kotlin and implementing recyclerview to display large datasets.
How to query data from or db via terminal
Display data on a client(browser)
SQL-traditionally
ORM-
Timebase data quest
Command-line utility- //manage.py
Create a shell to update our data
Python manage.py shell

From student. models import student

Querryset-allows us to access data store in a model

Access all data stored in a model
students=Student.objects.all()
Students

For student in students: //iterate/loop through list
    Print(student.first_name)
    Print (student.age)
    Print (student.profile_pics.url)//image are stored in urls

Filtering- allows us to query a subset of data
Eg: Student whose age is 20yrst
student=Student.object.filter(age=20)
student=Student.object.filter(first_name='Anastasia')

Eg student aged 20yrs and above
student=Students.objects.filter(age__age=20) //__shortcut of greater than
EG All student from rwanda
student=Student.object.filter(country=Rwanda)

Students whose firstname starts with "A"
student=Student.objects.filter(first_name__startswith="A")

Number of all student
student=Student.objects.filter(count)
First record
student=Student.objects.filter(first)
student=Student.objects.filter(first).firstname
student=Student.objects.filter(last)

Modify data
student=Student.objects.get(first_name="Anastasia",last_name="Belyse")
student.first_name// prints the first name
student.first_name="Wanjugu"

Save-helps us to commit data to the database
student.save()
student=Student.objects.id(1)
Primarycase

Display data to a client
View renders a request
Querry to get the relevant data
Template to display data
Url root to the view-----apply any business logic

Instagram engineering //Assignment

{% for student in student%}

{% end for}

Saving image
Reference image

Images
How to send information to a database

From Django import forms
From models import student
Class student registration from (forms.model forms)=forms
Class meta
model=student
field=””

Python
4Form- class
3View
1. Template-HTML CSS js
5. Model
Function-business logic
2Url router

Presentation layer
Business logic layer-SERVER
Database layer
We use HTTP-helps to communicate with the client

Client

Akirachixbank.com
Ip address-identifies server
cloud-

Client tier-web browser
DATA TIER-database
How to know server location

Database
SQL serve
MongoDB
Firebase DB
Mysql
Sqlite
Oracle
SQL DB-store data in a structured format in tables=>which column, and rows
The table is an attribute and column and row are the instances
NoSQL DB-we store unstructured data
We structure it in the document where each data is a document
Eg is a dict
student={‘name’:rahma}
Jayson object for API

webserver(server)

Ruby
Php

Java
Kotlin
Nodejs
Go
Python
C lang

Presentation tier/client tier (user inteface)
Html
Javascript
Css

Web application frameworks
-framework-is a set library/modules/software (collection of modules) which has a set of rule which make it easy to build a web app in a structured manner
It is a set of reusable rules-very quick and very quick

Presentation layer
Html
CSS
Js

Business logic
Python
Php
Go
Nodejs
Ruby
Java

Django
Flask
Loralvel
Rora
Phoenix
Cake PHP

React js
Vue
angular

Django framework
A WAF for building a web app using a python programming language

It follows a philosophy known as mvc=(model view controller)
Model-we define the data model
View-business logic
Controller-presentation
WHY USE DJANGO

- It is the most popular python framework
- One of the most popular web application framework
- It is very first
- It is secure
- It is easy to learn
- It has alot of batteries included-build in functionalities that u require
- (file storage)(login directly)(catching -store data near to the user)
- **www. Django project.org**
  How

  Python package index-how to do that
  Python package index-py pi
  Pip-package installer for python

Ubuntu software
Sudo apt-get
Npm-node package module
Pypi.org- check all available software

Install-python package index

Pip --version
python3 --version
Pip3 --version

install Virtual environment -libraries,

1. It is an isolated environment with a computer with its own dependencies-you can install software that is available to that environment
2. You can have multiple virtual environments in a single machine

python3 -m venv env1 -dir- to check
PWD
ls env2-ls
pip --version

cd desktop/python_class
CREATE FOLDER
mkdir python_web
cd
Create environment

Activate source env1/bin/activatedeactivate

Pip freeze
Create a Django project=
django-admin startproject AkiraChix
Akirachix is the directory

Web app
Web pl

Create a virtual env3
install Django
create a project school system
Make it a repo push to git hub share the link

8th/July/2021
Db-sqlite =it is the database
Manage. py-command line utility for managing the project locally….to lunch the webserver
Asgi- Asynchronous server gateway interface
Url.py- All project URLs are stored here(Uniform Resource Locator)
Setting. py-Store the configuration of the projects
WSGI- web server gateway interface(synchronous-await)

Compress
Thumbnail-small version of these picture
Store
Return
Asynchronous is nonblocking
Synchronous is blocking

An app is a package

Python manage.py startup student - to create an app
admin .py-allows us to create an admin interface for our projects
Apps. py-configuration of the app
Migration-contains database
Models. py-defines attributes and behavior of our entities
Test. py-creates test for the app
Views. py-business logic of web
Template-lendering

Data type

attribute	datatypre	Django datatype
		Char field Number field date/time Integerfield

python manage.py makemigration student-creates migrations
Install student app inside setting.py
python manage.py migrate
Python manage.py runserver

From models import student
Add to installed apps
Admin

Data model design design

Name	Description	Python datatype	Django modelfield
age	Information on how to old a student is	int	PositiveSmallInteger

Student ,teacher ,course calender

Django project.org model fields

1. Object-oriented programming paradigm
2. Functional

Attributes -
Behaviors-methods

How code is organized in python

Modules- a collection of classes,functions...
a file that contains python codes and we save it with. Py extension
It contains class functions flow control data types and data structures
Identified with.py
Multiple modules form **package**
Package is a directory of python modules
Python is an interpreted language-ie we don't compile
How to import-From pkg. module import, class

Instance variable

We create a constructor
def--init--(self ,name,age):

self.name=name
self.age=age

Assigning value(attribute) to a class once it is created
Class method add behavior to our classes
The first argument of a class method is always self
A class method is a normal python function

Question

1. Make python class a git repository
2. Create these 3 modules:-car.py,dog.py,bank.py
3. Create these classes in the respective, cat, dog, bank account
4. Each class should accept 2 instance variables
5. Each class should have two methods.

The attribute of a bank acc
Deposit
Withdraw
Borrow
Show balance
Update

The attribute of a bank acc

1. Name
2. Phone no
3. Id no
4. Pin
5. Email
6. Address
7. Account no
8. Balance

behavior

1. Deposit
2. Withdraw
3. Borrow
4. Show balance
5. Update
6. Transfer
7. Freeze
8. close

From module import Class
From module import Func_name
From the module. module import class
Standard library
Import math
Import calendar

Import math
dir(math)

math.log(10)math.tan(60)math.sin(11)