**Sample Flask Project:**

Install Sublime

Install MySQL (Server version: 5.7.17 MySQL Community Server (GPL))

set username and password

mkdir ecom

cd ecom

python3.6 -m venv myvenv

source myvenv/bin/activate

pip install --upgrade pip

mkdir ecom ( we will work inside ecom app of ecom project)

Create requirements.txt file in ecom project

Create create\_database.py in ecom project

Create .gitignore in ecom project

pip install --upgrade pip

pip install -r requirements.txt

ecom

|ecom

|static

|templates

|controllers

|managers

|exceptions

| ……

|routes.py

|\_\_init\_\_.py

|config.py

|api.py

|....

|requirements.txt

|create\_database.py

|.gitignore

Scaffolding in ecom app:

cd ecom

(scaffolding in ecom app now)

mkdir constants

mkdir controllers

mkdir datastore

mkdir managers

mkdir models

mkdir repositories

mkdir static

mkdir templates

mkdir utils

mkdir exceptions

vi config.py

vi config.py.sample

vi routes.py

vi api.py

vi \_\_init\_\_.py

------

Git

git init

git add README.md

git add .

git commit -m "first commit"

git remote add origin [git@github.com](mailto:git@github.com):ppundir/sample.git ( new repository created on git without readme)

git push -u origin master

FLASK\_APP=ecom flask db init

Creating directory /Users/pawanpundir/ecom/ecom/migrations ... done

Creating directory /Users/pawanpundir/ecom/ecom/migrations/versions ... done

Generating /Users/pawanpundir/ecom/ecom/migrations/alembic.ini ... done

Generating /Users/pawanpundir/ecom/ecom/migrations/env.py ... done

Generating /Users/pawanpundir/ecom/ecom/migrations/README ... done

Generating /Users/pawanpundir/ecom/ecom/migrations/script.py.mako ... done

Please edit configuration/connection/logging settings in '/Users/pawanpundir/ecom/ecom/migrations/alembic.ini' before proceeding.

Inside models, custom\_datatypes

<http://flask-sqlalchemy.pocoo.org/2.3/quickstart/>

**class Post(db.Model):  
 id = db.Column(db.Integer, primary\_key=True)  
 title = db.Column(db.String(80), nullable=False)  
 body = db.Column(db.Text, nullable=False)  
 pub\_date = db.Column(db.DateTime, nullable=False,  
 default=datetime.utcnow)  
  
 category\_id = db.Column(db.Integer, db.ForeignKey('category.id'),  
 nullable=False)  
 category = db.relationship('Category',  
 backref=db.backref('posts’))  
  
 def \_\_repr\_\_(self):  
 return '<Post %r>' % self.title  
  
  
class Category(db.Model):  
 id = db.Column(db.Integer, primary\_key=True)  
 name = db.Column(db.String(50), nullable=False)  
  
 def \_\_repr\_\_(self):  
 return '<Category %r>' % self.name**

After adding code for member and subscription:

FLASK\_APP=ecom flask db migrate

Generating /Users/pawanpundir/ecom/ecom/migrations/versions/b607c855545c\_.py ... done

Script.py.mako imports

from ecom.models import custom\_datatypes

Env.py

Context.configure add

user\_module\_prefix='custom\_datatypes.'

-----

Now thats base for any project.

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.

Most commonly, forks are used to either propose changes to someone else's project or to use someone else's project as a starting point for your own idea.

git clone [git@github.com](mailto:git@github.com):ppundir/sample.git

sudo mv sample/ ecommerce

(renamed repo)

rm -rf .git

git init

git add .

git commit -m "first commit"

git remote add origin [git@github.com](mailto:git@github.com):ppundir/ecommerce.git

git push -u origin master

Now work as needed.

python3.6 -m venv myvenv

source myvenv/bin/activate

pip install -r requirements.txt

Models:

Create files.

Delete migrations folder

Create config.py at level of config.py.sample and database

***ENV = 'development' # default***

***DEBUG = True # default False if ENV = production***

***SECRET\_KEY = '6c4b24e0cd361e65c94057dfa233fd16'***

***# SQLAlchemy***

***DB\_USER = 'root'***

***DB\_PASSWORD = '<password>'***

***DB\_HOST = 'localhost'***

***DB\_NAME = 'ecommerce'***

***# SQLALCHEMY\_DATABASE\_URI = 'mysql://<username>:<password>@<host>/<database>'***

***SQLALCHEMY\_DATABASE\_URI = 'mysql://%s:%s@%s/%s' % (DB\_USER, DB\_PASSWORD, DB\_HOST, DB\_NAME)***

***SQLALCHEMY\_TRACK\_MODIFICATIONS = False***

***SQLALCHEMY\_POOL\_RECYCLE = 600***

***SQLALCHEMY\_RECORD\_QUERIES = True***

***# redis***

***REDIS\_HOST = 'localhost'***

***REDIS\_PORT = 6379***

***REDIS\_DECODE\_RESPONSES = True***

***REDIS\_DB = ''***

***REDIS\_URL = 'redis://:@%s:%s/%s' % (REDIS\_HOST, REDIS\_PORT, REDIS\_DB)***

Comment manager code. Or controller code. We will write this all when needed.

Comment routes other than status

--

Create database ecommerce.

FLASK\_APP=ecom flask db init

Fix errors if prompted

When all fine, we get:

*Creating directory /Users/pawanpundir/ecommerce/ecom/migrations ... done*

*Creating directory /Users/pawanpundir/ecommerce/ecom/migrations/versions ... done*

*Generating /Users/pawanpundir/ecommerce/ecom/migrations/script.py.mako ... done*

*Generating /Users/pawanpundir/ecommerce/ecom/migrations/env.py ... done*

*Generating /Users/pawanpundir/ecommerce/ecom/migrations/README ... done*

*Generating /Users/pawanpundir/ecommerce/ecom/migrations/alembic.ini ... done*

*Please edit configuration/connection/logging settings in '/Users/pawanpundir/ecommerce/ecom/migrations/alembic.ini' before proceeding.*

Script.py.mako imports

from ecom.models import custom\_datatypes

Env.py

Context.configure add

user\_module\_prefix='custom\_datatypes.'

After \*\*current\_app.extensions['migrate'].configure\_args,

FLASK\_APP=ecom flask db migrate

Now it creates alembic version

**FLASK\_APP=ecom flask db upgrade**

After this success, we shd see tables

Start app:

uwsgi \

--master \

--protocol=http \

--socket 0.0.0.0:80 \

--catch-exceptions \

--reload-on-exception \

--enable-threads \

--buffer-size 32768 \

-w ecom:app -p 4 -O 2 \

--py-autoreload 1

**Use above (vimp)**

**Test** [**http://localhost:8080/**](http://localhost:8080/status) **work now….**

**Lets say in process we forgot to add product image. How to add.**

Change model, FLASK\_APP=ecom flask db migrate, FLASK\_APP=ecom flask db upgrade.

All inserts.

Static images with images (static/product/images)

Templates.

Can search for hoverable drop down menu

<https://www.w3schools.com/howto/howto_css_dropdown.asp>

---

Deployment:

<https://aws.amazon.com/free/>

Ec2: 750 hours per month of Linux, RHEL, or SLES t2.micro instance usage

s3:5 GB of Standard Storage

Rds: 750 Hours per month of db.t2.micro database usage (applicable DB engines)

Create a free account

Enter debit card/ credit card details.

**Create key pair**

Launch a virtual machine

chmod 400 ~/Downloads/mmmec.pem

Ec2 - **Ubuntu Server 18.04 LTS**

**with key pair**

**And you are connected.**

**ssh -i ~/Downloads/mmmec.pem ubuntu@3.16.43.189**

**Rds: Only enable options eligible for RDS Free Usage Tier**

**DB instance identifier - ritkart**

**Master username - ritkart**

**Master password - ritiksingh**

**Vpc security group mmmec created**

**Db name- ecommerce**

**---**

**Amazon elastic cache - of cache.t2micro Node usage**

**Redis create -**

**Name - mmmkart**

**Select t2.micro**

**Replicas -0**

**Subnet group - mmmkart-redis**

**Subnet - useeast2b**

**On box:**

**Sudo apt-get update**

**sudo apt install mysql-client-core-5.7**

**Add to security group - mysql/aurora anywhere**

**mysql -h mmmkart.cosxvkza0zpq.us-east-2.rds.amazonaws.com -u mmmut -p**

**sudo apt-get install redis-tools**

**Change security group to anywhere**

**----**

**git clone** [**git@github.com**](mailto:git@github.com)**:ppundir/ecommerce.git**

**Ssh-keygen**

**Add to ssh keys public--github(settings)**

**Sudo apt-get install python3-venv**

python3.6 -m venv myvenv

source myvenv/bin/activate

sudo apt-get install mysql-client libmysqlclient-dev

sudo apt-get install python3-dev

sudo apt-get install gcc libc6-dev linux-headers-4.15.0-32-generic libffi-dev

pip install -r requirements.txt

Create config.py

**FLASK\_APP=ecom flask db upgrade**

**Inserts**

**Sudo su if want on 80 for uwsgi start**

**--**

**Razorpay in test mode**

**Generate api keys**

**Key\_id: rzp\_test\_4NWrZjcfBP6KdH**

**Key\_secret: WdkIGvGOP0wYtL6RLFFDD4cA**

**pip install razorpay**

[**https://github.com/razorpay/razorpay-python**](https://github.com/razorpay/razorpay-python)

**https://github.com/razorpay/razorpay-python-testapp/blob/master/templates/app.html**

**For the test payment itself, you can use card number 5104015555555558 with any CVV, and any future expiry date.**

[**https://dashboard.razorpay.com/#/app/paymentpages**](https://dashboard.razorpay.com/#/app/paymentpages) **-- custom payment page**

https://aws.amazon.com/console/

Signin using root creds Welcome!23 / jolly.ranu@gmail.com

<https://goo.gl/HHFghr>

Add in config.py file below 2 lines for razorpay integration.

RAZORPAY\_KEY = 'rzp\_test\_5RWzckcBnnYm8W'

RAZORPAY\_SECRET = 'nScNNjGpjCUyFRjKxmuTUZOe'

Concept on dictionary and list complex scenario

category\_map = { "men" : [{"name" : "wallets", "id" : "323232323"}, {}, {}] ,

"women" : [{"name":"sarees","id" : "32323"},{"name":"jewellery","id" : "444"},{}]

category\_map = {}

for elem in result:

if elem[0] in category\_map:

category\_map[elem[0]].append({"name": elem[1],"id" : elem[2]})

else:

category\_map[elem[0]] = [{"name": elem[1],"id" : elem[2]}]