Python Django Bootcamp

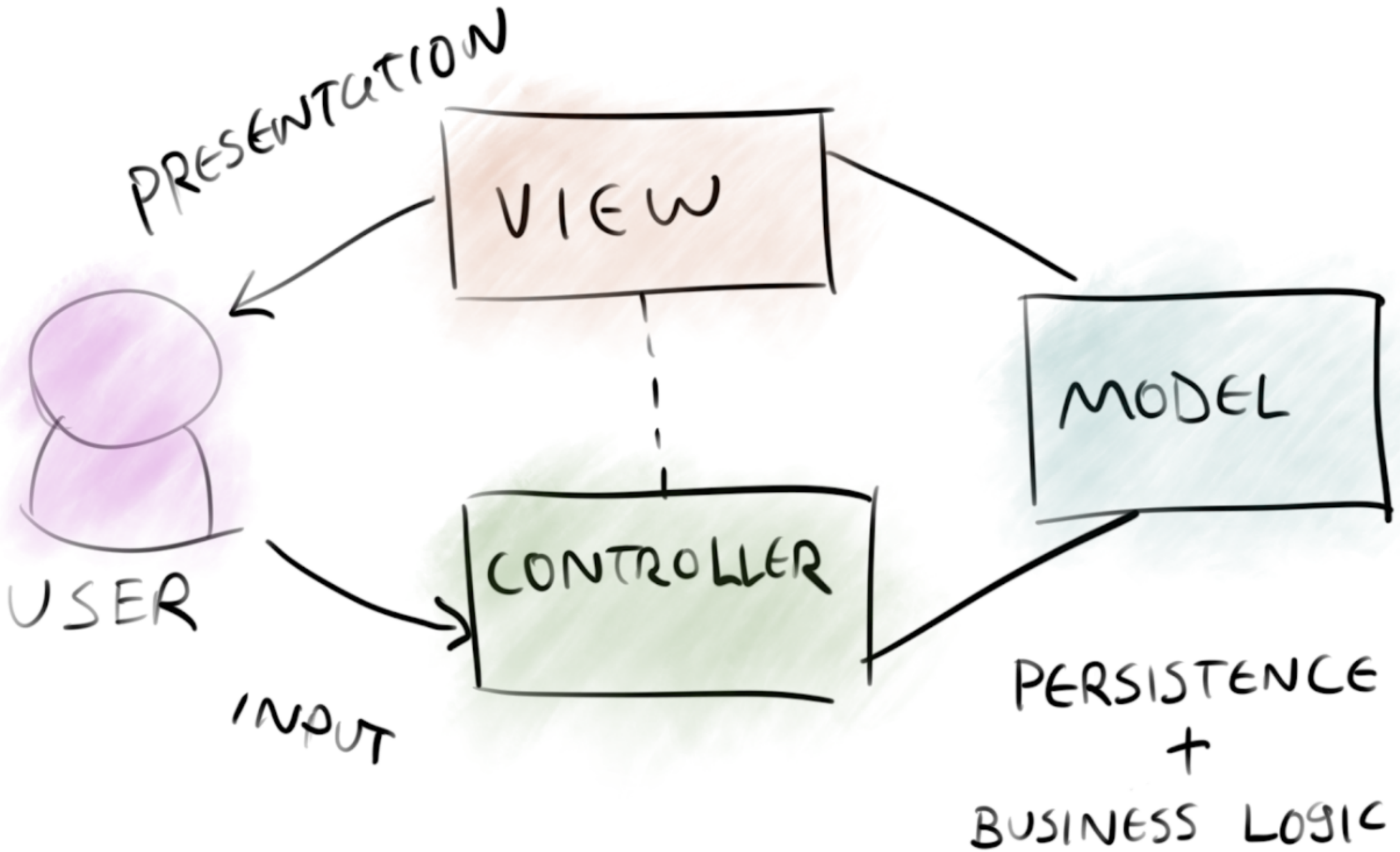
Reinert Yosua Rumagit, S.Kom., M.TI.
Andry Chowanda, S.Kom., MM., Ph.D., MBCS.

PRE-TEST

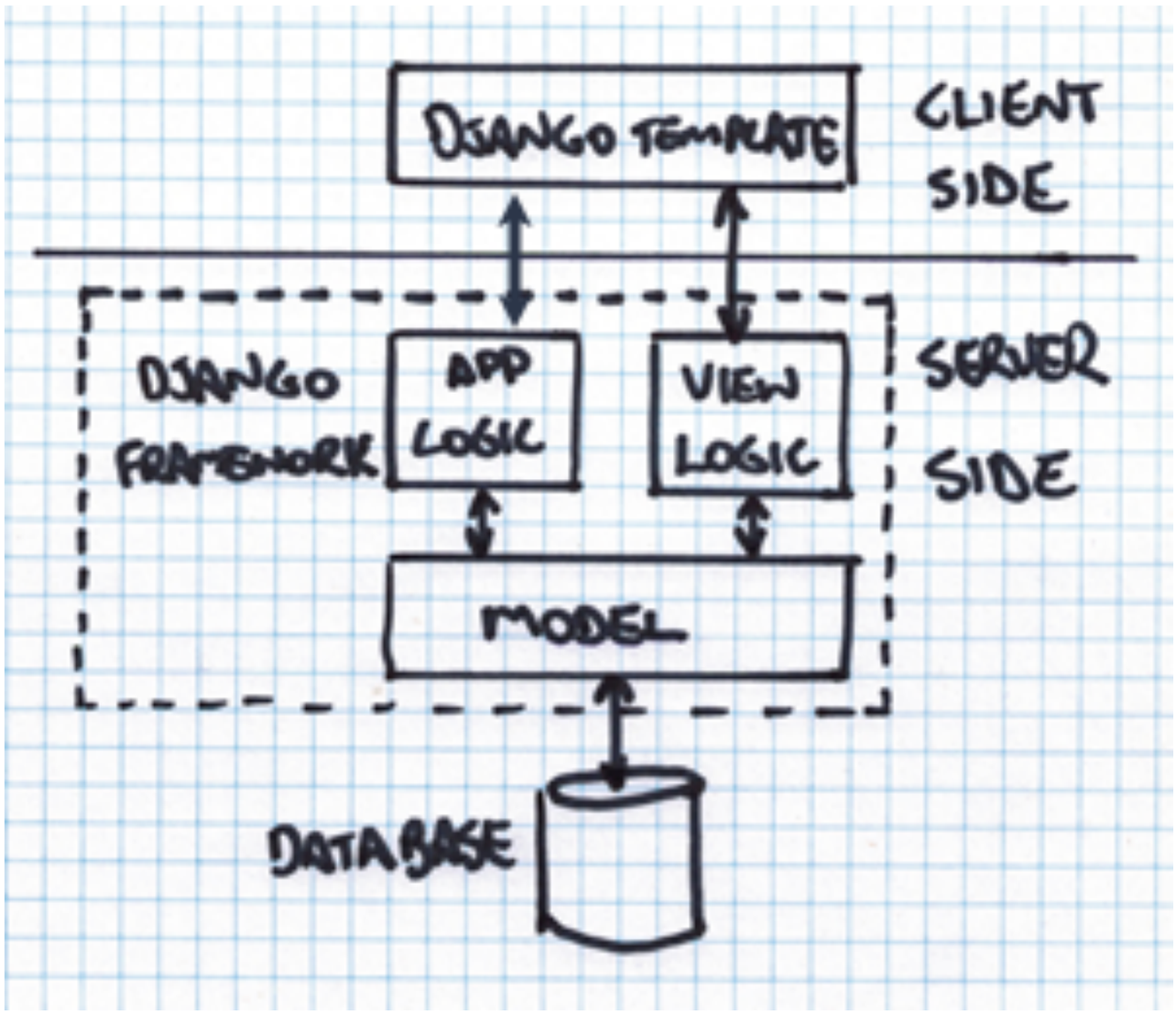
The Quest

- Understand Python Django Concept
- Create a (very) simple website
- Create a second (very) simple website
- Create a website that connect to SQLLite
- Create a website that shows data from DB
- Play around with Django Forms
- Create a website with CRUD Functions

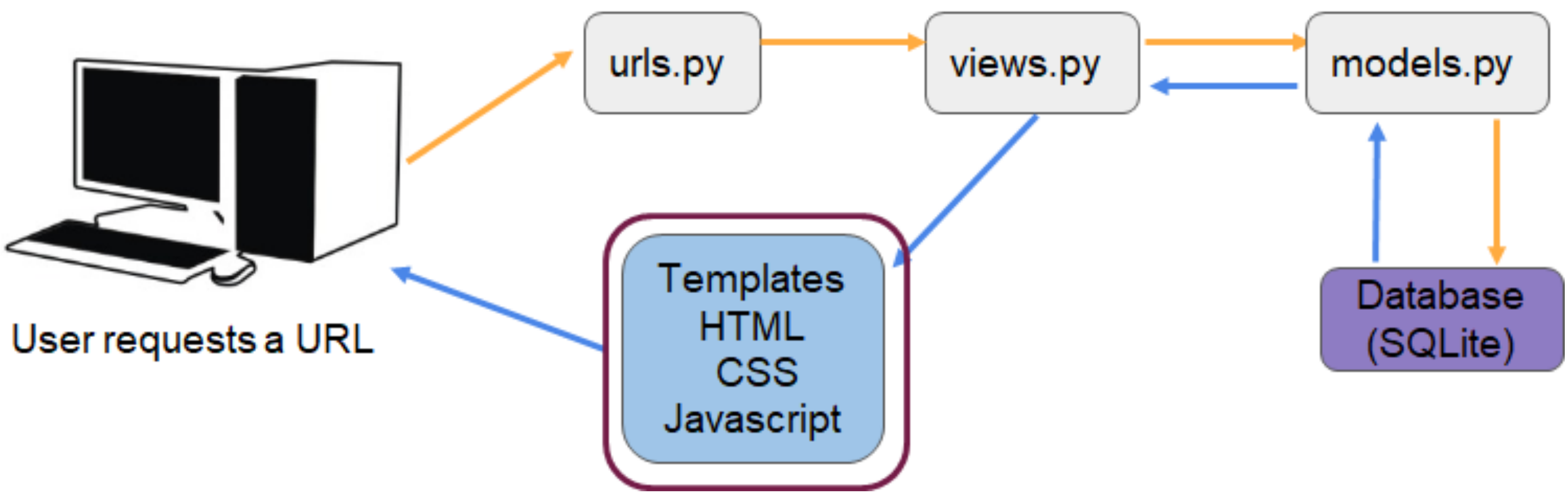
Review: The MVC



Django: The MVT



Django Request Flow



The Quest

- ~~Understand Python Django Concept~~
- Create a (very) simple website
- Create a second (very) simple website
- Create a website that connect to SQLLite
- Create a website that shows data from DB
- Play around with Django Forms
- Create a website with CRUD Functions

Create a (very) Simple Website

- Create a project
- Check (Run) the Web Server
- Create an app
- Create a view page
- Learn more on URLs Mapping

Check (Run) the Web Server

```
python manage.py runserver
```

```
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 17 unapplied migration(s). Your project may not work properly until yo
u apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
November 10, 2019 - 12:04:23
Django version 2.2.7, using settings 'myFirstWebsite.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Create a Project

```
django-admin startproject myFirstWebsite
```

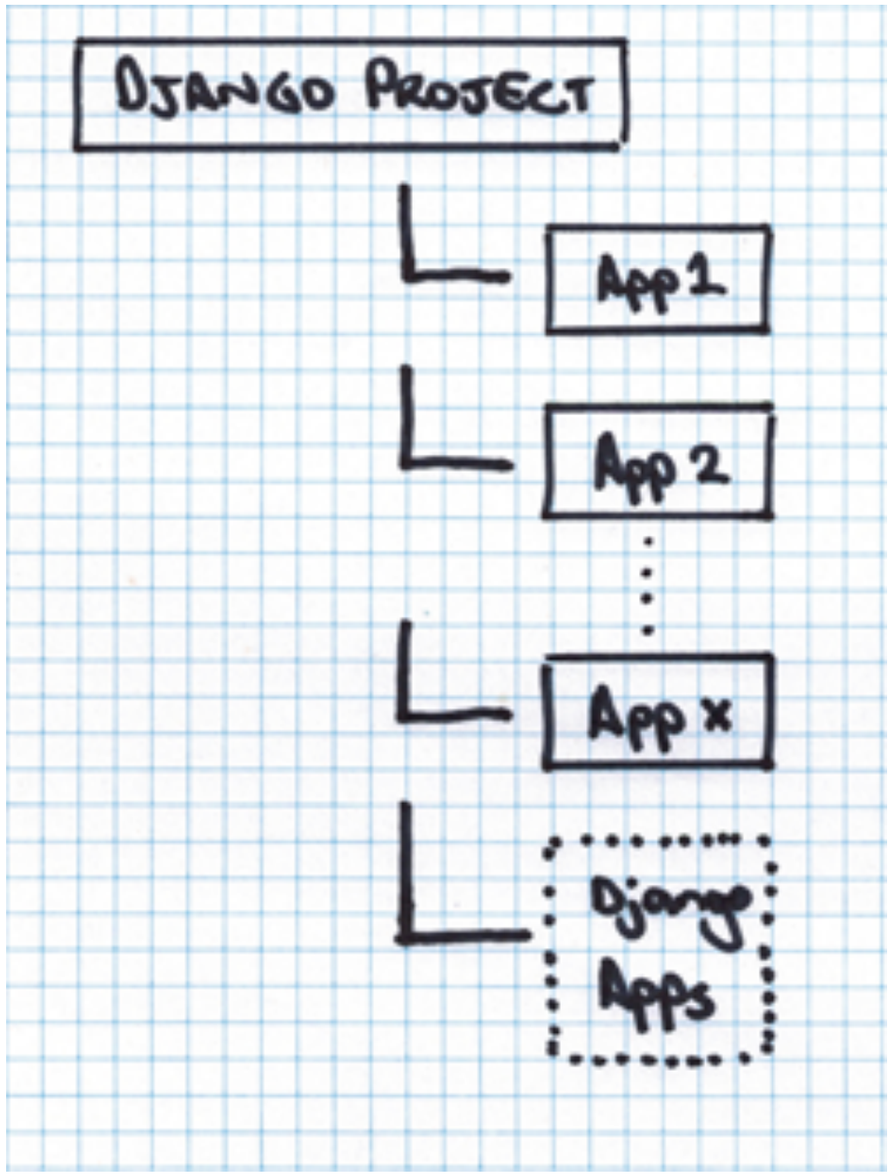
```
manage.py
myFirstWebsite
    settings.py
    urls.py
    wsgi.py
```

Create an App

```
manage.py startapp HelloWorld
```

```
HelloWorld
manage.py
myFirstWebsite
```

```
HelloWorld
    admin.py
    apps.py
    migrations
    models.py
    tests.py
    views.py
```



Create a View Page

- Register the app (setting.py)

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'HelloWorld'
]
```

- Modify the view page (views.py)

```
from django.shortcuts import render
from django.http import HttpResponse

def index(request):
    return HttpResponse("Quest: Hello World!")
```

- Map the URLS (urls.py)

```
from django.contrib import admin
from django.urls import path
from HelloWorld import views

urlpatterns = [
    path('', views.index, name='index'),
    path('admin/', admin.site.urls),
]
```

- Run the server

Learn More on URLs Mapping

- Function views

  1. Add an import: from my_app import views

  2. Add a URL to urlpatterns: path('', views.home, name='home')

- Class-based views

  1. Add an import: from other_app.views import Home

  2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')

- Including another URLconf

  1. Import the include() function: from django.urls import include, path

  2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))

Mini Boss

- Create a new project with 2 apps, with include URLs where:
  - The first app showing a text: *This is the italic page!*
  - The second app showing a text: **This is the bold page!**

The Quest

- ~~Understand Python Django Concept~~
- ~~Create a (very) simple website~~
- Create a second (very) simple website
- Create a website that connect to SQLLite
- Create a website that shows data from DB
- Play around with Django Forms
- Create a website with CRUD Functions

Create a (very) Simple Second Website

- Create a project
- Create an app
- Create an HTML file
- Create a view page

Create an HTML file

```html
<!DOCTYPE html>
<html>
    <head>
        <title>My Second Website - Index</title>
    </head>
    <body>
        <h1>Hello, Welcome to my Second Website</h1>
        <p> {{ my_variable }} </p>
    </body>
</html>
```

Create a View Page

```python
from django.shortcuts import render

def index(request):
    my_dict = {'my_variable' : 'This is variables passed from view'}
    return render(request, 'index.html', context=my_dict)
```

The Quest

- ~~Understand Python Django Concept~~
- ~~Create a (very) simple website~~
- ~~Create a second (very) simple website~~
- Create a website that connect to SQLLite
- Create a website that shows data from DB
- Play around with Django Forms
- Create a website with CRUD Functions

Create a Website that Connect to SQLLite

- Create models
- Migrate models to database
- Manage admin page
- Connect to mySQL DB

Create Models

```python
from django.db import models

class User(models.Model):
    userName = models.CharField(primary_key=True, max_length=100)
    firstName = models.CharField(max_length=100)
    lastName = models.CharField(max_length=100)
    address = models.CharField(max_length=200)
    dob = models.DateField()
    phoneNumber = models.IntegerField()
```

Migrate Models to DB

```
python manage.py migrate
```

```
python manage.py makemigrations myIndexApp
```

```
python manage.py migrate
```

Manage Admin Page

```
from django.contrib import admin
from myIndexApp.models import User

admin.site.register(User)
```

```
python manage.py createsuperuser
```

Connect to MySQLDB

```
'default': {
    'ENGINE': 'django.db.backends.mysql',
    'NAME': 'myDatabase',
    'USER': 'root',
    'PASSWORD': '',
    'HOST': 'localhost',   # Or an IP Address that your DB is hosted on
    'PORT': '3306',
},
```

Mini Boss 2

- Use the project from Quest: Create a Website that Connect to SQLLite and change the database to mySQLDB.

The Quest

- ~~Understand Python Django Concept~~
- ~~Create a (very) simple website~~
- ~~Create a second (very) simple website~~
- ~~Create a website that connect to SQLLite~~
- Create a website that shows data from DB
- Play around with Django Forms
- Create a website with CRUD Functions

Create a Website that Shows Data from DB

- Load data from models
- Create an HTML File

Load Data from Models

```python
from django.shortcuts import render
from myIndexApp.models import User

def index(request):
    mydata = User.objects.all()
    userDataDict = {'usersData':mydata}
    return render(request, 'index.html', context=userDataDict)
```

Create an HTML File

```
<!DOCTYPE html>
<html>
    <head>
        <title>My Fourth Website - Index</title>
    </head>
    <body>
        <h1>Hello, Welcome to my Fourth Website</h1>
        <p> Here are the list of the users data: </p>
        {% if usersData %}
            <table>
                <thead>
                    <th>User Name</th>
                    <th>First Name</th>
                    <th>Last Name</th>
                    <th>Address</th>
                    <th>DOB</th>
                    <th>Phone Number</th>
                </thead>
                {% for user in usersData %}
                    <tr>
                        <td>{{user.userName}}</td>
                        <td>{{user.firstName}}</td>
                        <td>{{user.lastName}}</td>
                        <td>{{user.address}}</td>
                        <td>{{user.dob}}</td>
                        <td>{{user.phoneNumber}}</td>
                    </tr>
                {% endfor %}
                </table>
            {% else %} <p>NO USERS FOUND!</p>
            {% endif %}
    </body>
</html>
```

The Quest

- ~~Understand Python Django Concept~~
- ~~Create a (very) simple website~~
- ~~Create a second (very) simple website~~
- ~~Create a website that connect to SQLLite~~
- ~~Create a website that shows data from DB~~
- Play around with Django Forms
- Create a website with CRUD Functions

Play Around with Django Forms

- Create a Form
- Create a View
- Create an HTML for Index
- Create an HTML for Form
- Link form with Models
- Update the View

Create a Form

```python
from django import forms

class formUser(forms.Form):
    userName = forms.CharField()
    firstName = forms.CharField()
    lastName = forms.CharField()
    address = forms.CharField(widget=forms.Textarea)
    dob = forms.DateField(widget=forms.SelectDateWidget)
    phoneNumber = forms.CharField()
```

Create a View

```python
from django.shortcuts import render
from myFormApp import forms

def index(request):
    return render(request, 'index.html')

def formUserView(request):
    form = forms.formUser()
    if request.method == 'POST':
        form = forms.formUser(request.POST)
        if form.is_valid():
            print("Validation Success!")

    return render(request, 'form.html',{'form':form})
```

Create an HTML for Index

```html
<!DOCTYPE html>
<html>
    <head>
        <title>My Fifth Website - Index</title>
    </head>
    <body>
        <h1>Hello, Welcome to my Fifth Website</h1>
        <p> Here are the list of the users data: </p>
        {% if usersData %}
            <table>
                <thead>
                    <th>User Name</th>
                    <th>First Name</th>
                    <th>Last Name</th>
                    <th>Address</th>
                    <th>DOB</th>
                    <th>Phone Number</th>
                </thead>
            {% for user in usersData %}
                <tr>
                    <td>{{user.userName}}</td>
                    <td>{{user.firstName}}</td>
                    <td>{{user.lastName}}</td>
                    <td>{{user.address}}</td>
                    <td>{{user.dob}}</td>
                    <td>{{user.phoneNumber}}</td>
                </tr>
            {% endfor %}
            </table>
        {% else %} <p>NO USERS FOUND!</p>
        {% endif %}
        <p> please visit <a href="/form"> here </a> to register</p>
    </body>
</html>
```

Create an HTML for Form

```html
<!DOCTYPE html>
<html>
    <head>
        <title>My Fifth Website - User Form</title>
    </head>
    <body>
        <form method="POST">
            {{ form.as_p }}
            <input type="submit" name="formInputUser" value="Submit">
            {% csrf_token %}
        </form>
    </body>
</html>
```

Link Forms with Models

```python
def formInputUserView(request):
    form = forms.newUserForm()
    if request.method == 'POST':
        form = forms.newUserForm(request.POST)
        if form.is_valid():
            form.save(commit=True)
            return index(request)
        else:
            print("Validation Error!")
    return render(request, 'form.html',{'form':form})
```

Update the View

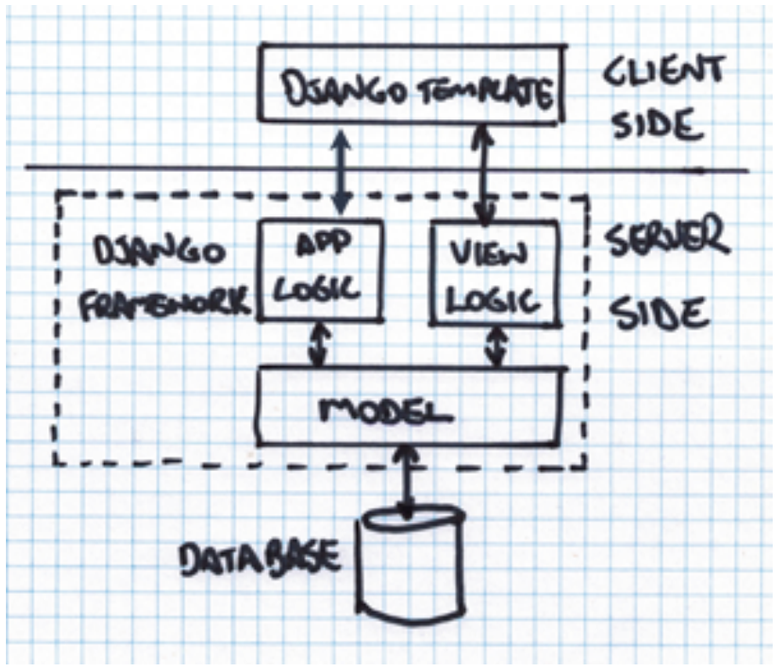
```
def index(request):
    myUserData = User.objects.all()
    myUserDataDict = {'usersData':myUserData}
    return render(request, 'index.html', context=myUserDataDict)
```

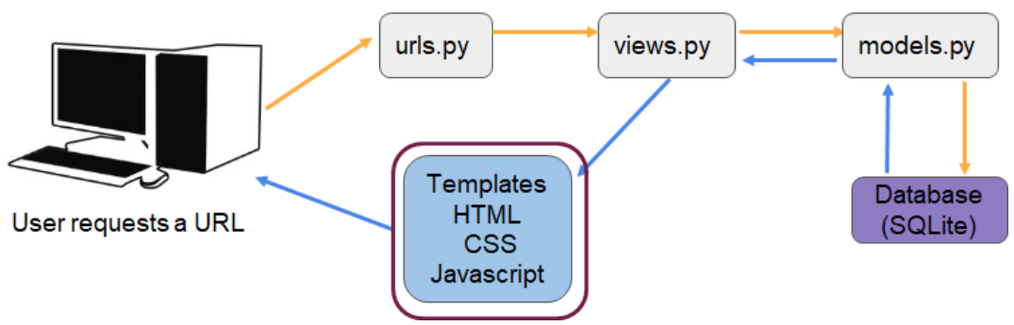
The Quest

- ~~Understand Python Django Concept~~
- ~~Create a (very) simple website~~
- ~~Create a second (very) simple website~~
- ~~Create a website that connect to SQLLite~~
- ~~Create a website that shows data from DB~~
- ~~Play around with Django Forms~~
- Create a website with CRUD Functions

Create a website with CRUD Functions

- Models: Use Previous Models
- Views: Create View
- Template: Edit Index HTML
- Template: Create HTML for Update & Delete
- Template: Edit URL





Create View: View Detail

```
def viewDetailView(request, key):
    userDetail = User.objects.get(userName__exact=key)
    return render(request, 'viewDetail.html', {'usersData':userDetail})
```

Edit the Index HTML

```
<!DOCTYPE html>
<html>
    <head>
        <title>My Sixth Website - Index</title>
    </head>
    <body>
        <h1>Hello, Welcome to my Sixth Website</h1>
        <p> Here are the list of the users data: </p>
        {% if usersData %}
            <table>
                <thead>
                    <th>User Name </th>
                    <th>Edit</th>
                    <th>Delete</th>
                </thead>
            {% for user in usersData %}
                <tr>
                    <td><a href= {% url "viewDetail" user.userName %}> {{user.userName}} </a></td>
                    <td><a href= {% url "update" user.userName %}> Edit </a> </td>
                    <td><a href= {% url "delete" user.userName %}> Delete </a> </td>
                </tr>
            {% endfor %}
            </table>
        {% else %} <p>NO USERS FOUND!</p>
        {% endif %}
        <p> please visit <a href="/form"> here </a> to register</p>
    </body>
</html>
```

Create View Detail HTML

```
<!DOCTYPE html>
<html>
    <head>
        <title>My Sixth Website - View Details</title>
    </head>
    <body>
        <h1> User Details </h1>
        {% if usersData %}
            <p>user name = {{usersData.userName}} </p>
            <p>first name = {{usersData.firstName}} </p>
            <p>last name = {{usersData.lastName}} </p>
            <p>DOB = {{usersData.dob}} </p>
            <p>address = {{usersData.address}} </p>
            <p>phone number = {{usersData.phoneNumber}} </p>
        {% else %}
            <p> NO DETAILS TO SHOW! </p>
        {% endif %}
        <a href={% url "index" %}> Back </a>
    </body>
</html>
```

Edit URLs

```python
from django.contrib import admin
from django.urls import path
from myApp import views

urlpatterns = [
    path('', views.index, name='index'),
    path('form/', views.formInputUserView, name='formUser'),
    path('viewDetail/<str:key>', views.viewDetailView, name='viewDetail'),
    path('delete/<str:key>', views.deleteUserView, name='delete'),
    path('update/<str:key>', views.updateUserView, name='update'),
    path('admin/', admin.site.urls),
]
```

Create View: Delete

```
def deleteUserView(request, key):
    userDelete = User.objects.get(userName__exact=key)
    if request.method == 'POST':
        userDelete.delete()
        return index(request)
    return render(request, 'delete.html', {'usersData':userDelete})
```

Edit the Index HTML

```
<!DOCTYPE html>
<html>
    <head>
        <title>My Sixth Website - Index</title>
    </head>
    <body>
        <h1>Hello, Welcome to my Sixth Website</h1>
        <p> Here are the list of the users data: </p>
        {% if usersData %}
            <table>
                <thead>
                    <th>User Name </th>
                    <th>Edit</th>
                    <th>Delete</th>
                </thead>
            {% for user in usersData %}
                <tr>
                    <td><a href= {% url "viewDetail" user.userName %}> {{user.userName}} </a></td>
                    <td><a href= {% url "update" user.userName %}> Edit </a> </td>
                    <td><a href= {% url "delete" user.userName %}> Delete </a> </td>
                </tr>
            {% endfor %}
            </table>
        {% else %} <p>NO USERS FOUND!</p>
        {% endif %}
        <p> please visit <a href="/form"> here </a> to register</p>
    </body>
</html>
```

Create Delete HTML

```html
<!DOCTYPE html>
<html>
    <head>
        <title>My Sixth Website - User Delete</title>
    </head>
    <body>
        <form method="POST">
            Are you sure you want to delete user: "{{ usersData.userName }}" ?
            <input type="submit" name="formDeleteUser" value="Submit" />
            {% csrf_token %}
        </form>
        <p><a href={% url "index" %}> Cancel </a></p>
    </body>
</html>
```

Edit URLs

```python
from django.contrib import admin
from django.urls import path
from myApp import views

urlpatterns = [
    path('', views.index, name='index'),
    path('form/', views.formInputUserView, name='formUser'),
    path('viewDetail/<str:key>', views.viewDetailView, name='viewDetail'),
    path('delete/<str:key>', views.deleteUserView, name='delete'),
    path('update/<str:key>', views.updateUserView, name='update'),
    path('admin/', admin.site.urls),
]
```

Create View: Update

```
def updateUserView(request, key):
    userUpdate = User.objects.get(userName__exact=key)
    form = forms.newUserForm(instance=userUpdate)
    if request.method == 'POST':
        form = forms.newUserForm(request.POST, instance=userUpdate)
        if form.is_valid():
            form.save(commit=True)
            return index(request)
        else:
            print("Validation Error!")
    return render(request, 'form.html',{'form':form})
```

Edit the Index HTML

```
<!DOCTYPE html>
<html>
    <head>
        <title>My Sixth Website - Index</title>
    </head>
    <body>
        <h1>Hello, Welcome to my Sixth Website</h1>
        <p> Here are the list of the users data: </p>
        {% if usersData %}
            <table>
                <thead>
                    <th>User Name </th>
                    <th>Edit</th>
                    <th>Delete</th>
                </thead>
            {% for user in usersData %}
                <tr>
                    <td><a href= {% url "viewDetail" user.userName %}> {{user.userName}} </a></td>
                    <td><a href= {% url "update" user.userName %}> Edit </a> </td>
                    <td><a href= {% url "delete" user.userName %}> Delete </a> </td>
                </tr>
            {% endfor %}
            </table>
        {% else %} <p>NO USERS FOUND!</p>
        {% endif %}
        <p> please visit <a href="/form"> here </a> to register</p>
    </body>
</html>
```

Edit URLs

```python
from django.contrib import admin
from django.urls import path
from myApp import views

urlpatterns = [
    path('', views.index, name='index'),
    path('form/', views.formInputUserView, name='formUser'),
    path('viewDetail/<str:key>', views.viewDetailView, name='viewDetail'),
    path('delete/<str:key>', views.deleteUserView, name='delete'),
    path('update/<str:key>', views.updateUserView, name='update'),
    path('admin/', admin.site.urls),
]
```

The Quest

- ~~Understand Python Django Concept~~
- ~~Create a (very) simple website~~
- ~~Create a second (very) simple website~~
- ~~Create a website that connect to SQLLite~~
- ~~Create a website that shows data from DB~~
- ~~Play around with Django Forms~~
- ~~Create a website with CRUD Functions~~

Final Boss

- SiReksa is planning to create a website with Python-Django Framework for their Tabungan Reksadana. Where the customers are be able to:
    - See the their detail: including, but not limited to: username, full name, address, date registered, cash in hand, portfolio, exp, and level
    - Update their details, limited to full name, and address.
    - Buy Reksadana (manual input), customer is required to input: the name, the unit price, the unit number. This will also update customer's cash in hand, portfolio, exp, and level
    - List all the Reksadana they have
    - Sell all Reksadana, where the list will be deleted. This will also update customer's cash in hand, portfolio, exp, and level
- You are free to use class for the views.