MDA5104 OBJECT ORIENTED TECHNOLOGIES

ASSIGNMENT FIVE

NAME: YVONNE MAKENA

STUDENT ID: 21/01300

TASK

(i) Use a suitable web application framework such as Django, Flask etc, to demonstrate/simulate implementation of at least three components/subsystems of your proposed architectural pattern in assignment three (15 Marks)

(ii) Compile a document with screen shots of code and outputs as well as descriptions

SOLUTION:

Web Application used: Django which provides both front end tools of web application e.g. data selection, formatting, authentication mechanisms, display, URL management, a templating language, and backend tools for manipulating data source with ease.

My proposed architecture: Client – Server Architecture (3-Tier)...Image on the second last slide

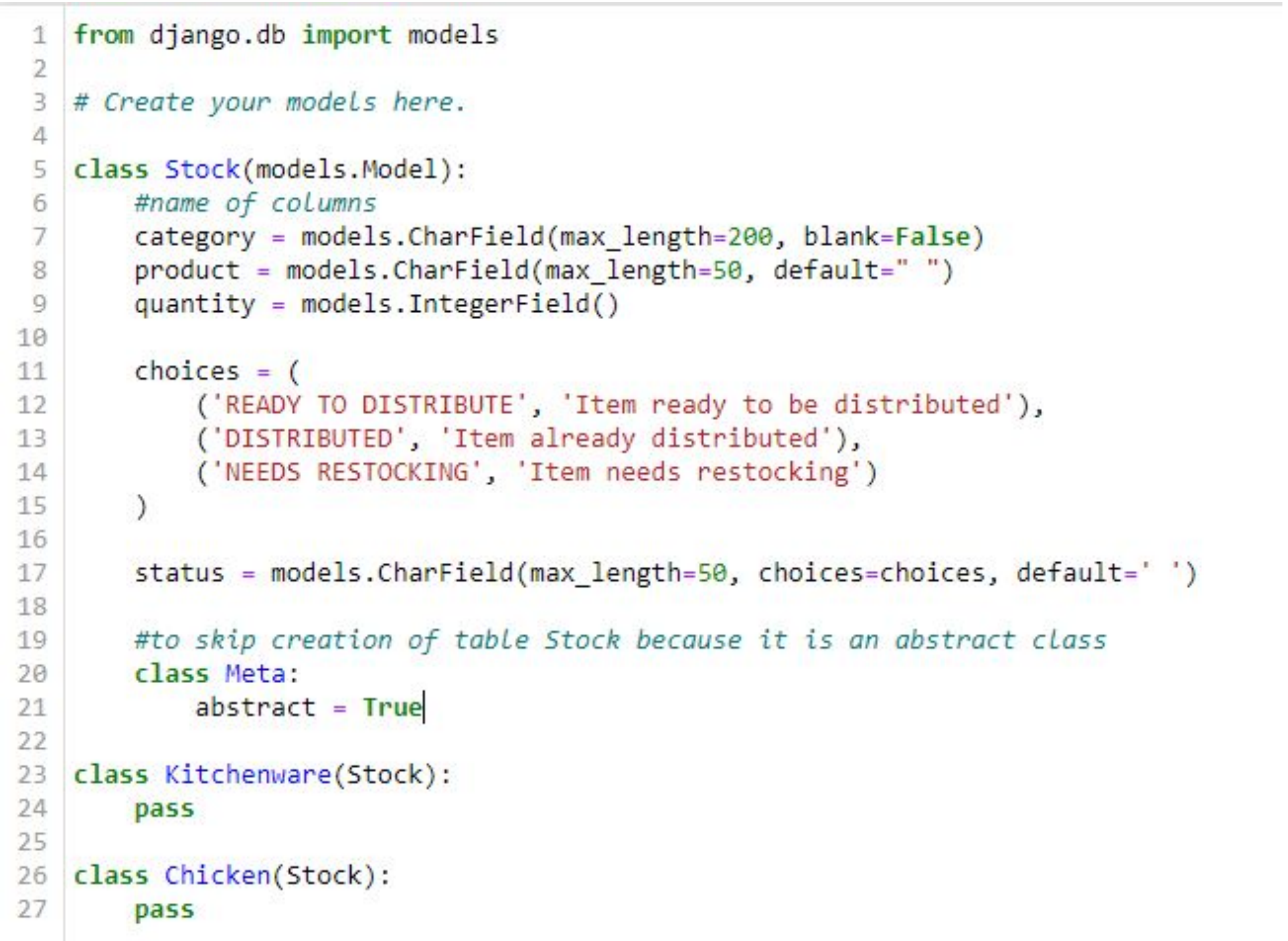
Components whose implementation is demonstrated:

I am going to simulate the implementation through an inventory management system since my proposed architecture was for a distribution company.

1. Data storage - Where data is stored and retrieved
2. Application logic - Performs detailed processing hence controls applications.
3. Presentation - Displays available information to the company users/clients. In this case, the client will be the director and procurement officer

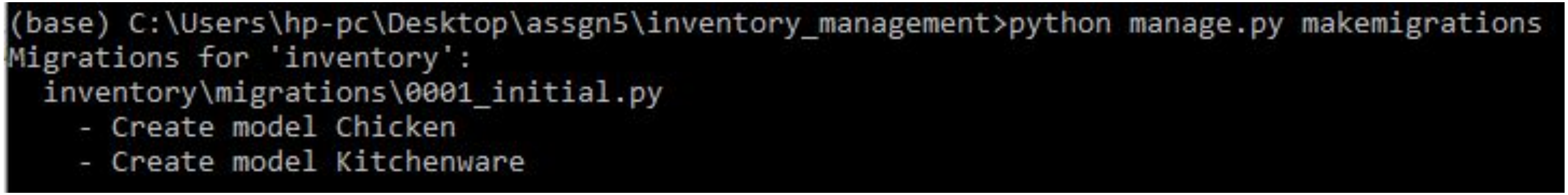
1. Data Storage

I have used models to define tables where data will be stored.



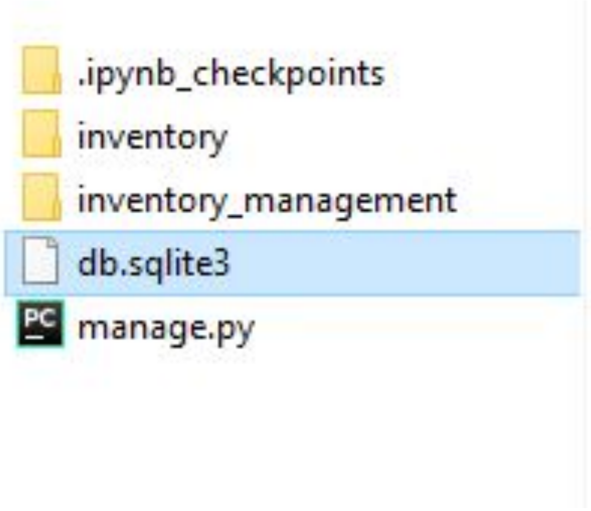
```python
from django.db import models

# Create your models here.

class Stock(models.Model):
    #name of columns
    category = models.CharField(max_length=200, blank=False)
    product = models.CharField(max_length=50, default=" ")
    quantity = models.IntegerField()

    choices = (
        ('READY TO DISTRIBUTE', 'Item ready to be distributed'),
        ('DISTRIBUTED', 'Item already distributed'),
        ('NEEDS RESTOCKING', 'Item needs restocking')
    )

    status = models.CharField(max_length=50, choices=choices, default=' ')

    #to skip creation of table Stock because it is an abstract class
    class Meta:
        abstract = True

class Kitchenware(Stock):
    pass

class Chicken(Stock):
    pass
```

Output:
Below is the output to show tables created after running python manage.py migrate in the anaconda command prompt



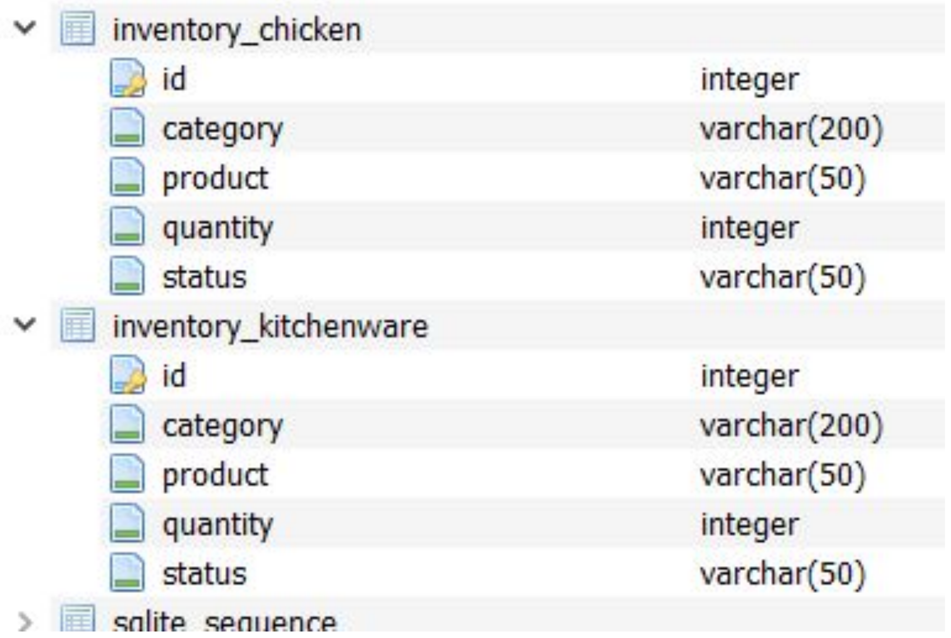
```
(base) C:\Users\hp-pc\Desktop\assgn5\inventory_management>python manage.py makemigrations
Migrations for 'inventory':
  inventory\migrations\0001_initial.py
    - Create model Chicken
    - Create model Kitchenware
```

The script also creates an sql database file



```
.ipynb_checkpoints
inventory
inventory_management
db.sqlite3
manage.py
```

Two tables created: Viewing from the DB Browser



inventory_chicken	
id	integer
category	varchar(200)
product	varchar(50)
quantity	integer
status	varchar(50)
inventory_kitchenware	
id	integer
category	varchar(200)
product	varchar(50)
quantity	integer
status	varchar(50)
sqlite_sequence	

2. Application Logic

Where all controls are defined. All processing of the system takes place.

1. views.py - to display all elements on the webpage . Fetch entries from the database and display on the table
2. forms.py - to enable entering information into the database
3. urls.py - where I got to link all the functions so that I can use them

In [ ]:

```python
from django.shortcuts import render, redirect, get_object_or_404
from .models import *
from .forms import *

# Create your views here.


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

def display_kitchenware(request):
    items = Kitchenware.objects.all()
    context = {
        'items': items,
        'header': 'Kitchenware',
    }
    return render(request, 'index.html', context)

def display_chicken(request):
    print(Chicken)
    items = Chicken.objects.all()
    context = {
        'items': items,
        'header': 'Chicken',
    }
    return render(request, 'index.html', context)


def add_item(request, cls):
    if request.method == "POST":
        form = cls(request.POST)

        if form.is_valid():
            form.save()
            return redirect('index')

    else:
        form = cls()
        return render(request, 'add_new.html', {'form' : form})


def add_kitchenware(request):
    return add_item(request, KitchenwareForm)

def add_chicken(request):
    return add_item(request, ChickenForm)

def edit_item(request, pk, model, cls):
    item = get_object_or_404(model, pk=pk)

    if request.method == "POST":
        form = cls(request.POST, instance=item)
        if form.is_valid():
            form.save()
            return redirect('index')
    else:
        form = cls(instance=item)

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



def edit_kitchenware(request, pk):
    return edit_item(request, pk, Kitchenware, KitchenwareForm)


def edit_chicken(request, pk):
    return edit_item(request, pk, Chicken, ChickenForm)


def delete_kitchenware(request, pk):

    template = 'index.html'
    Kitchenware.objects.filter(id=pk).delete()

    items = Kitchenware.objects.all()

    context = {
        'items': items,
    }

    return render(request, template, context)


def delete_chicken(request, pk):

    template = 'index.html'
    Chicken.objects.filter(id=pk).delete()

    items = Chicken.objects.all()

    context = {
        'items': items,
    }

    return render(request, template, context)

```

```python
from django import forms
from .models import *

class KitchenwareForm(forms.ModelForm):
    class Meta:
        model = Kitchenware
        fields = ('category', 'product', 'quantity', 'status')


class ChickenForm(forms.ModelForm):
    class Meta:
        model = Chicken
        fields = ('category', 'product', 'quantity', 'status')
```

```
from django.conf.urls import url
from .views import *

urlpatterns = [
    url(r'^$', index, name='index'),

    url(r'^kitchenware$', display_kitchenware, name='display_kitchenware'),
    url(r'^chicken$', display_chicken, name='display_chicken'),

    url(r'^add_kitchenware$', add_kitchenware, name='add_kitchenware'),
    url(r'^add_chicken$', add_chicken, name='add_chicken'),


    url(r'^kitchenware/edit_item/(?P<pk>\d+)$', edit_kitchenware,
name="edit_kitchenware"),
    url(r'^chicken/edit_item/(?P<pk>\d+)$', edit_chicken, name="edit_chicken"),

    url(r'^kitchenware/delete/(?P<pk>\d+)$', delete_kitchenware,
name="delete_kitchenware"),
    url(r'^chicken/delete/(?P<pk>\d+)$', delete_chicken, name="delete_chicken"),



]
```

3. Presentation

Where the interface of the website is defined. I have used html and css.

Html files used include:

- index.html - has the face of the website
- base.html -where template inheritance has taken place. Enable all html files to use the same navigation files
- add_new.html - defines the aspect of adding data. Uses POST method to add a new item that did not exist in the database
- edit_item.html - defines the aspect of editing data. Also POST method to make changes to items in the database.

In [ ]:

```
<!-- inherit base.html into index-->
{% extends 'base.html' %}

<!-- replace body from base.html-->
{% block body %}

<br>

    <div class="button-group">
      <a href="{% url 'display_kitchenware' %}" class="btn btn-primary btn-m
      <a href="{% url 'add_kitchenware' %}" class="btn btn-warning btn-sm" r

      <a href="{% url 'display_chicken' %}" class="btn btn-primary btn-md" r
      <a href="{% url 'add_chicken' %}" class="btn btn-warning btn-sm" role=

    </div>
 <br>


  <h4>Currently Viewing {{ header }}</h4>

  <table class="table table-hover">
    <thead>
      <tr>
        <th>id</th>
        <th>Category</th>
        <th>Product</th>
        <th>Quantity</th>
        <th>Status</th>
      </tr>
    </thead>

    <tbody>
       {% for item in items %}

      <tr>
        <td>{{ item.pk }}
        <td>{{ item.category }}</td>
        <td>{{ item.product }}</td>
        <td>{{ item.quantity }}</td>
        <td>{{ item.status }}</td>


       {% if header|lower == "kitchenware" %}
        <td>
            <a href="{% url 'edit_kitchenware' item.pk %}" class="btn btn-wa
            <a href="{% url 'delete_kitchenware' item.pk%}" class="btn btn-d
        </td>
        {% else %}
        <td>
            <a href="{% url 'edit_chicken' item.pk %}" class="btn btn-warnin
            <a href="{% url 'delete_chicken' item.pk%}" class="btn btn-dange
        </td>

        {% endif %}

      </tr>

      {% endfor %}
    </tbody>
  </table>

{% endblock %}
```

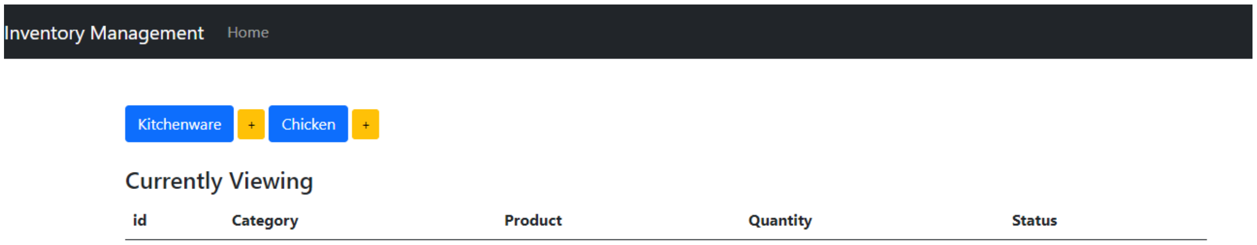
In [ ]:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Inventory</title>
  </head>

  {% load static %}
    <!-- link html with css-->
  <link rel="stylesheet" href="{% static '/css/style.css' %}"/>
  <link rel="stylesheet" href="{% static '/css/bootstrap.min.css' %}"/>

  <body>
 <!-- navigation-->
    <nav class="navbar navbar-expand navbar-dark bg-dark">
  <a class="navbar-brand" href="#">Inventory Management</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-t
    <span class="navbar-toggler-icon"></span>
  </button>

  <div class="collapse navbar-collapse" id="navbarsExample02">
    <ul class="navbar-nav mr-auto">
      <li class="nav-item active">
        <a class="nav-link" href="{% url 'index' %}">Home</a>
      </li>
    </ul>
  </div>
</nav>

    <br>
      <!-- bootstrap container where every other template is going to overri
    <div class="container">

      {% block body %}

      {% endblock%}

    </div>

  </body>
</html>

```

```
{% extends 'base.html' %}

{% block body %}

  <div class="container">
    <form method="POST">
        <br>
      {% csrf_token %}

<!--       <h4>{{ header }}</h4> -->

      {% for field in form %}
      <div class="form-group row">
        <label for="id_{{ field.name }}" class="col-2 col-form-label">{{ field.label }}
</label>
        <div class="col-10">
          {{ field }}
        </div>
      </div>
      {% endfor %}

      <div class="form-group row">
        <div class="offset-sm-2 col-sm-6">

          <button type="submit" class="btn btn-primary">Add Product</button>

<!--         </div> -->

    </form>
      </div>
    {% endblock %}
{% extends 'base.html' %}

{% block body %}

  <div class="container">
    <form method="POST">
        <br>
      {% csrf_token %}

<!--       <h4>{{ header }}</h4> -->
        <h3><u>Editing item</u></h3>

      {% for field in form %}
      <div class="form-group row">
        <label for="id_{{ field.name }}" class="col-2 col-form-label">{{ field.label }}</label>
        <div class="col-10">
          {{ field }}
        </div>
      </div>
      {% endfor %}

<!--       <div class="form-group row">
        <div class="offset-sm-2 col-sm-6"> -->

          <button type="submit" class="btn btn-primary">Edit Product</button>

<!--         </div> -->

    </form>
      </div>
    {% endblock %}
```

USER INTERFACE

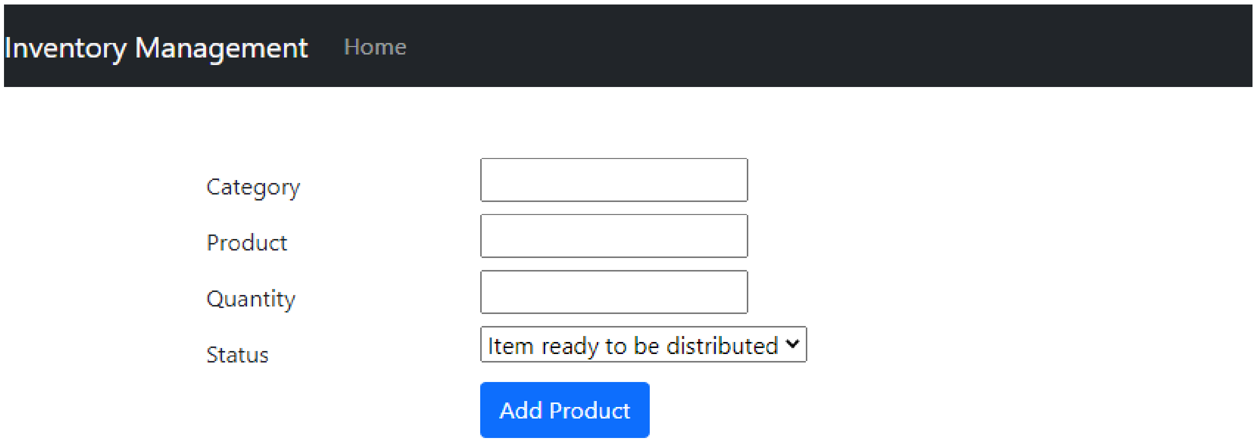
1. Home page



Currently Viewing

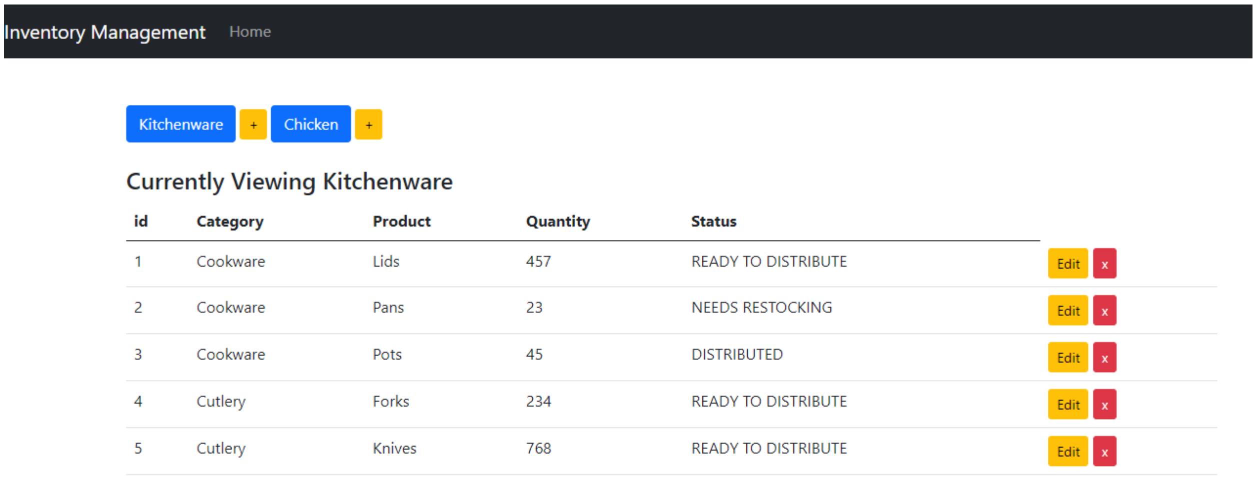
id	Category	Product	Quantity	Status

2. Add Interface



Category	
Product	
Quantity	
Status	Item ready to be distributed

3. Sample view after products have been added

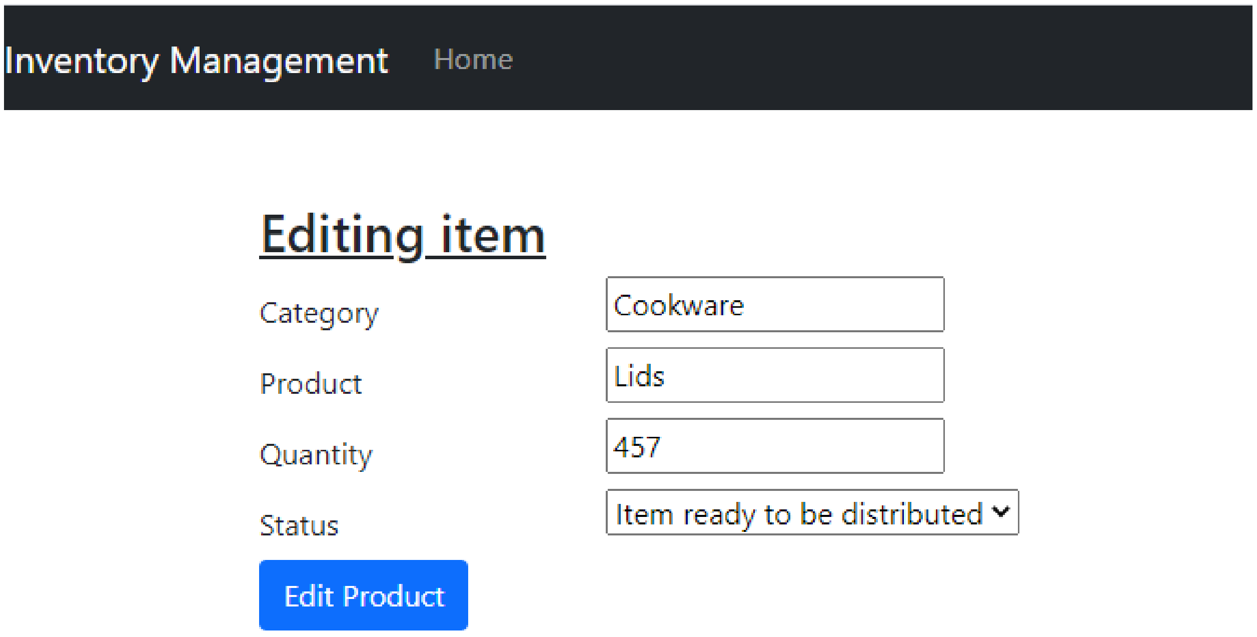

Inventory Management Home

Kitchenware + Chicken +

Currently Viewing Kitchenware

id	Category	Product	Quantity	Status	
1	Cookware	Lids	457	READY TO DISTRIBUTE	Edit x
2	Cookware	Pans	23	NEEDS RESTOCKING	Edit x
3	Cookware	Pots	45	DISTRIBUTED	Edit x
4	Cutlery	Forks	234	READY TO DISTRIBUTE	Edit x
5	Cutlery	Knives	768	READY TO DISTRIBUTE	Edit x

4. Edit Interface

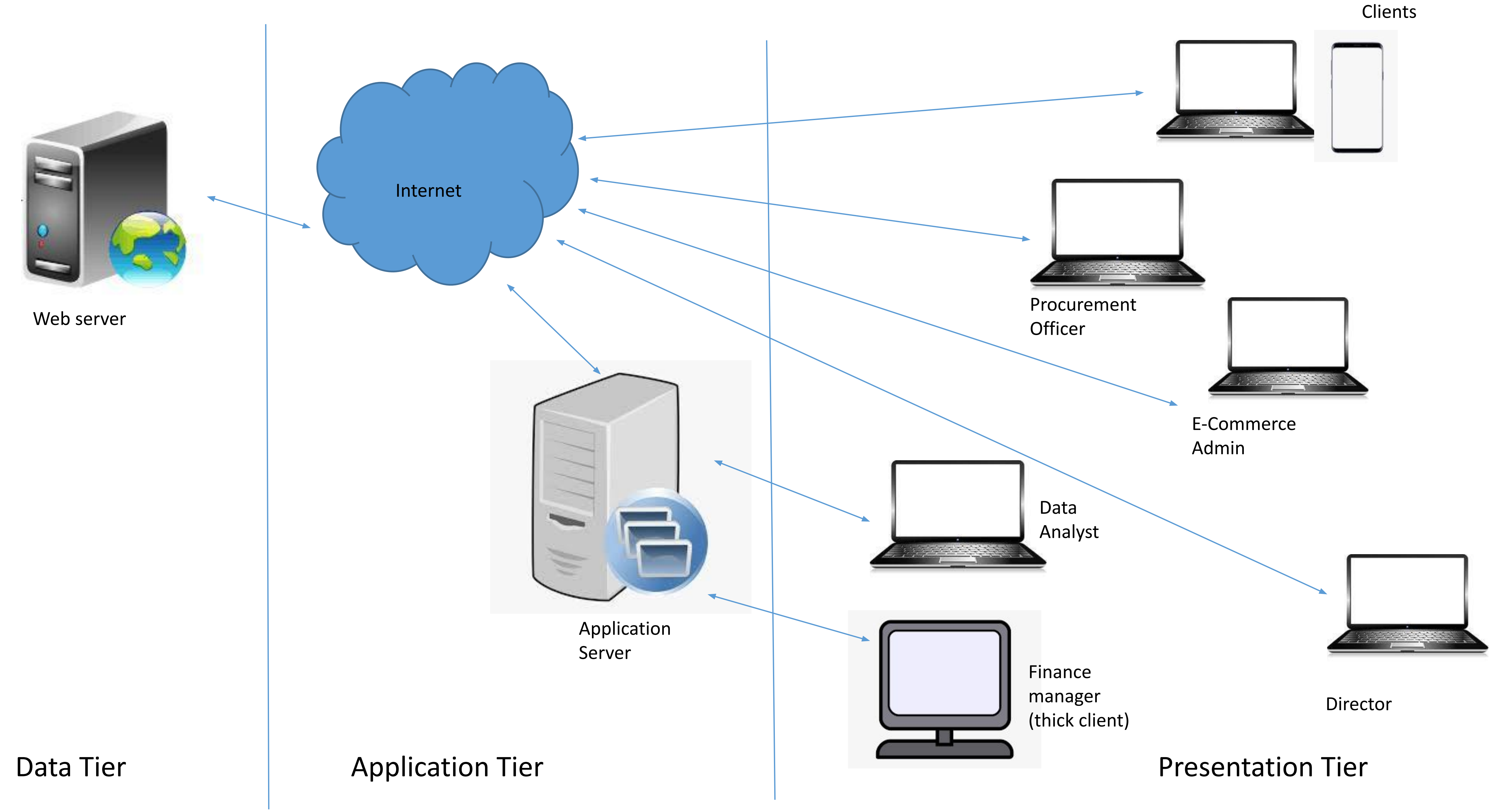

Inventory Management Home

Editing item

Category	Cookware
Product	Lids
Quantity	457
Status	Item ready to be distributed

Edit Product

Client – Server Architecture for a Distribution Company (3-Tier)



References

Class Notes:

https://djangocentral.com/create-a-hello-world-django-application/

https://simpleisbetterthancomplex.com/series/2017/09/11/a-complete-beginners-guide-to-django-part-2.html

https://medium.com/ayuth/how-to-use-django-in-jupyter-notebook-561ea2401852#:~:text=After%20that%20create%20a%20jupyter,as%20python%20manage.py%20shell%20.

https://djangobook.com/mdj2-django-templates/

https://www.simplifiedpython.net/django-templates-tutorial

https://docs.djangoproject.com/en/3.1/intro/tutorial01/

https://www.edureka.co/blog/django-tutorial/

https://djangoforbeginners.com/hello-world/

Youtube tutorials