**Jewellery eCommerce Site**: Python-Django Framework Implementation Design Document

Version No: 1.0

Date: May 2019

Author: Ramya Babu

**Table of Contents**

[1. Introduction and Solution Overview 3](#_Toc26959745)

[2. High Level Functional Details 4](#_Toc26959746)

[3. High Level Technical Details 7](#_Toc26959747)

[1. Technical Implementation 9](#_Toc26959748)

[2. GITHUB location 10](#_Toc26959749)

[3. Advantages of using Django-python framework. 11](#_Toc26959750)

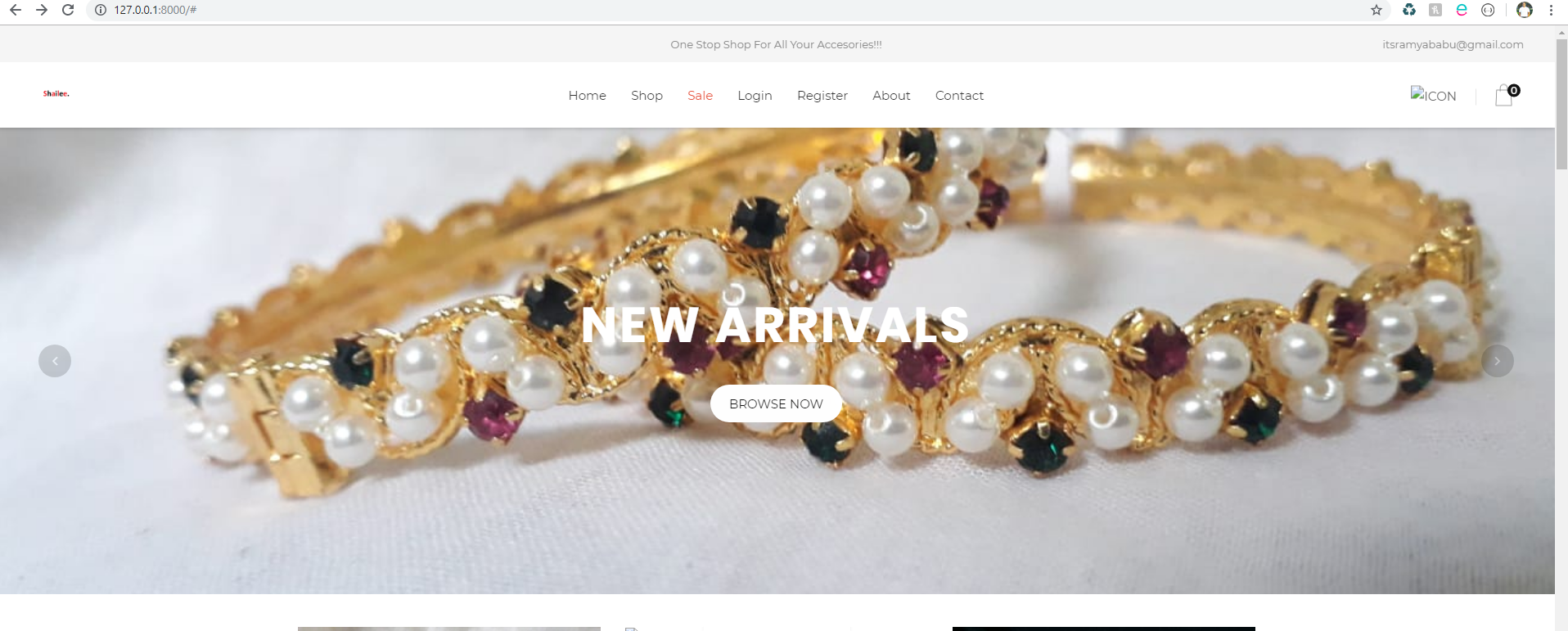
# Introduction and Solution Overview

This document is to provide technical and functional information on my eCommerce website which I created(from existing available free templates) using python-Django framework. This is to learn the framework and see how powerful it. I shall be adding more features to it in the future.

So this is hypothetical situation where a customer has low budget and wanted to release an eCommerce website for their customers quickly.

# High Level Functional Details

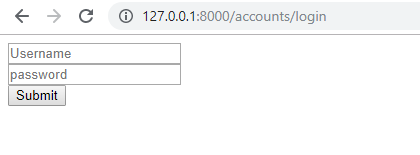
**My Home Page**



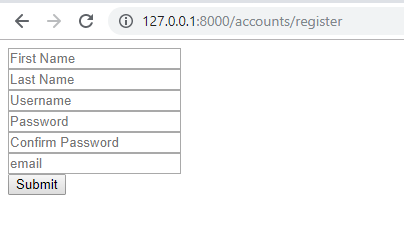
**Menus on the home page**



On click of **Login**, takes us to a login page where user can input the user name and password

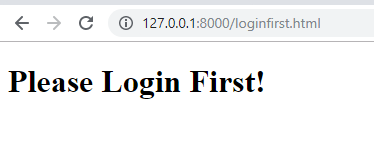


On click of **Register,** takes us to a registration page where user can register.

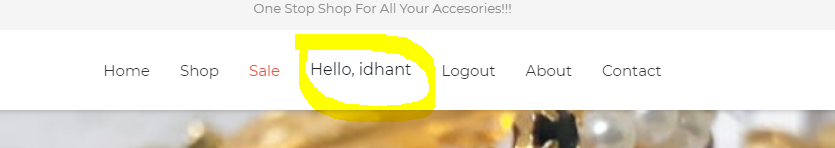


The details are stored in the DB. While login in, we are checking if the user exists or not.

On click of **Sale** menu item, if the user has not logged in, it will give this message,

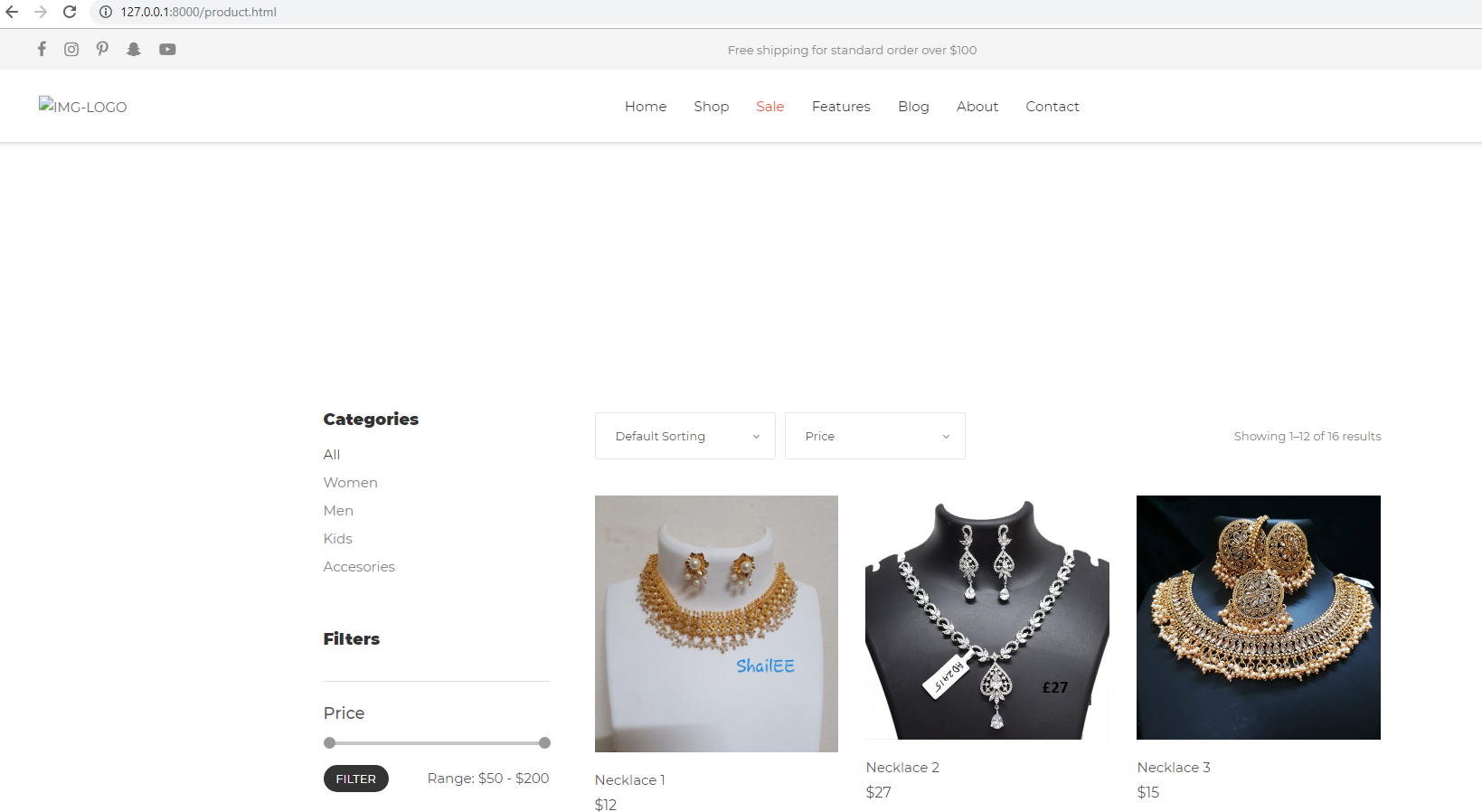


Once the user logs in, it will show the users name,



And the Register button is hidden and logout button is made visible.

Once user has logged it, they will be allowed to go to the Sale page, the pictures here are coming from the DB (postgres sql). I inserted all records from the django admin page.



# High Level Technical Details

This section will provide a high level steps involved in the development of this application. Since we have a short time to release the website, I decided to pick an existing free templates available in the market for eCommerce and make a few changes to suit our customer. Hence, the web site is an existing template used and blended that in Django using python.

Pick an existing HTML free template. Change the necessary for our client and blend it with Django using jinja (in red) format.

Example 1  
*<a href="index.html" class="logo">*

*<img src="{% static 'images/icons/Shailee.png' %}" alt="IMG-LOGO">*

*</a>* Example 2

*{% if user.is\_authenticated %}*

*<li>Hello,* ***{{****user.first\_name****}}****</li>*

*<li> <a href="accounts/logout">Logout</a> </li>*

*{% else %}*

*<li> <a href="accounts/login">Login</a> </li>*

*<li> <a href="accounts/register">Register</a></li>*

*{% endif %}*

Additionally create Login/Logout features for the customers.

Uso Django framework to handle various paths in the site. Use postgresSQL as backend to store data related to the catalogue. Django gives easy methods to insert data via models.

When users register, this is how I pushed data to the DB.

*def register(request):*

*if request.method == 'POST':*

*first\_name = request.POST['first\_name']*

*last\_name = request.POST['last\_name']*

*username = request.POST['username']*

*password1 = request.POST['password1']*

*password2 = request.POST['password2']*

*email = request.POST['email']*

*if (password1==password2):*

*if User.objects.filter(username=username).exists():*

*messages.info(request,'Username taken..')*

*return redirect('register')*

*elif User.objects.filter(email=email).exists():*

*messages.info(request,'email is taken...')*

*return redirect('register')*

*else:*

*user = User.objects.create\_user(username=username,first\_name=first\_name,last\_name=last\_name,password=password1,email=email)*

*user.save();*

*return redirect('login')*

*else:*

*messages.info(request,'Password is not matching..')*

*return redirect('register')*

*else:*

*return render(request,'register.html')*

# Technical Implementation

This implementation is on windows. Installed Python(v3.7). Installed the django package required in python by saying pip install django. Below are the steps to achieve a working eCommerce website. Django comes with a light weight server to test our application which we run on 127.0.0.1:8000

1. Created a test environment to use for my project.
   1. Install the package required for creating the virtual env. Pip install virtualenvwrapper-win
   2. mkvirtualenv ramyatestenv.
   3. workon ‘envname’
2. Always use this environment for development of this project. What ever we need for this project, install here.
3. Install django here. Pip install –django.
4. Created a new folder for my project. Go to that folder and type command ***“django –admin startproject Jewell”***
5. We got three files after this command. Init, urls.py and manage.py, settings.py . These are the files we shall be manipulating to achieve our requirements. The project urls.py will be inheriting all the urls of app we create.
6. We will create app for each functionality (as these apps can be re-used in other projects). App will have certain functionality which we will code in the view.py file.
   1. The app also has models.py which will define our DB table structure and its an easy way to form a structure of the table and use it to push data to the tables.
   2. It will have its own urls.py (which we create) to define our website paths.
7. All the html files are to be kept in a folder. The path to these files need to be done in settings.py. All the static data of the project needs to be kept in specific folders and the path has to be configured in settings.py
8. Hence, we have 3 main things which is virtual environment, project and app to start with.
9. The high level flow of how django works is as follows,

**Check if this path is in urls.py**

**Go to views.py to run the functionality**

1. To test the changes, run “python manage.py runserver”.
2. All the code related to my project has been embedded below,

# GITHUB location

https://github.com/ramya-babu/eCommerce\_PythonDjano.git

# Advantages of using Django-python framework.

1. Django is fast and it’s a full stack framework.
2. All components are available related to any operations you want to do. E.g creare website, have database, form validations etc. Drawback here is, all the responsibilities lies on the developer from end to end since all components are available with us.
3. Provides security to have a safe web site.
4. It highly scalable.
5. We do have other frameworks for python. Still exploring ☺