Solution Design Document

Background:

Our focus is on creating wedding lists for couples. In a simplified scenario a couple creates a wedding list and adds different types of products to that which become available to their wedding guests for purchase as a present. After the wedding the couple can create their order and decide which gifts to have delivered to them from all the guests' purchases. Your task is to implement this scenario on a basic level.

Enablement -

The idea is to build a simple E- commerce website with easy navigations using Django Framework.

1. We start off by installing the necessary softwares and packages that are required to build the website.

```
pip install Django==3.0.8
```

2. Once Django is installed , create a project and an app.

```
django-admin startproject ProjectName
python manage.py startapp AppName
```

DB/ Django Models Architecture -

State - Third Normal Form(3NF)

We are going to create 4 new models to represent the Product list, Customer information, Customer Orders and Customer Order Line Items.

In addition to the custom models, we will be referring to the Django User model for user creation and the customer model shares a 1:1 relationship with the user model.

Model Definitions-

Note - *For future scalability, a custom model Customer has been created instead of using the inbuilt User Model.*

Product List -

Column	Type
Name	CharField(200)
Brand	CharField(200)
Price	FloatField
in_stock_quantity	IntegerField

Customer-

Column	Type
User	1:1
Name	Charfield(200)
email	EmailField

Order -

Column	Type
Customer	Foreign Key
Create_dt	DateTime
Order_fulfilled	CharField(100)

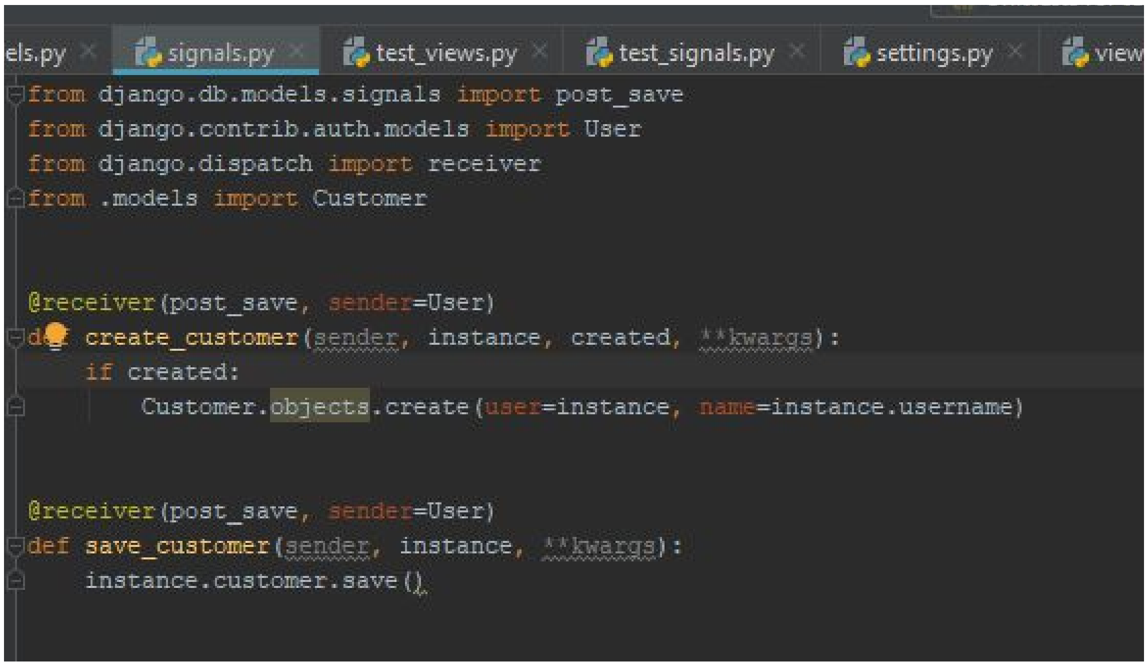
OrderLineItem -

Column	Type
Order	FK
Product	FK
Customer	FK
Quantity	IntegerField

Date_added	DateTimeField
Status	CharField(200)
Quantity_Purchased	IntegerField

Signals -

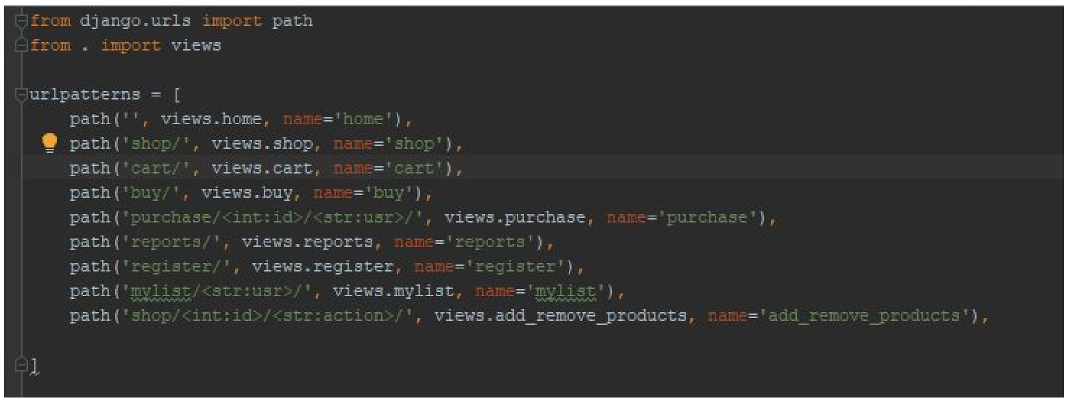
Once the models are defined and the necessary migrations are completed , create a signal on post save event to create a customer record when a new user is registered.



```
from django.db.models.signals import post_save
from django.contrib.auth.models import User
from django.dispatch import receiver
from .models import Customer


@receiver(post_save, sender=User)
def create_customer(sender, instance, created, **kwargs):
    if created:
        Customer.objects.create(user=instance, name=instance.username)


@receiver(post_save, sender=User)
def save_customer(sender, instance, **kwargs):
    instance.customer.save()
```

Url Config -

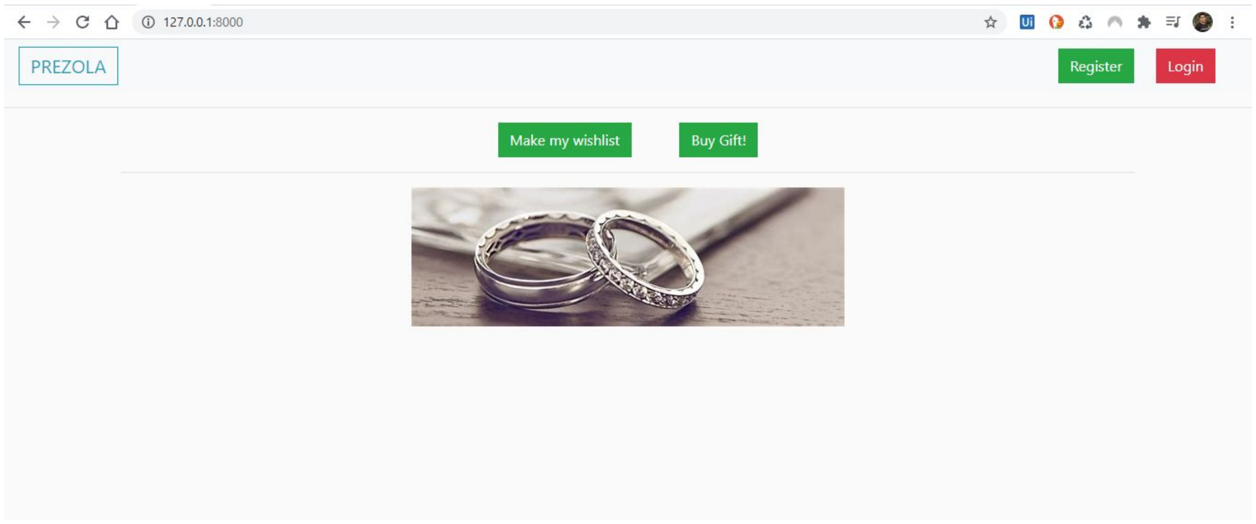
We will be defining 11 URLs for successful navigations.



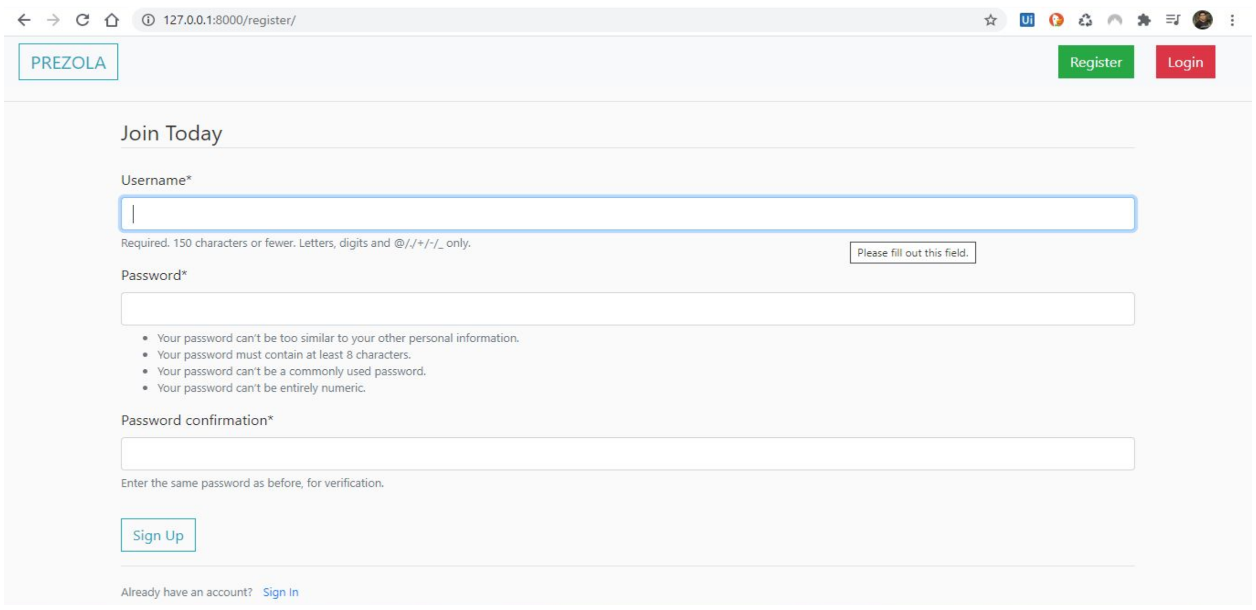
```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.home, name='home'),
    path('shop/', views.shop, name='shop'),
    path('cart/', views.cart, name='cart'),
    path('buy/', views.buy, name='buy'),
    path('purchase/<int:id>/<str:usr>/', views.purchase, name='purchase'),
    path('reports/', views.reports, name='reports'),
    path('register/', views.register, name='register'),
    path('mylist/<str:usr>/', views.mylist, name='mylist'),
    path('shop/<int:id>/<str:action>/', views.add_remove_products, name='add_remove_products'),

]
```

Key Navigations -

HomePage



RegistrationPage


127.0.0.1:8000/register/

PREZOLA

Register

Login

Join Today

Username*

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Please fill out this field.

Password*

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

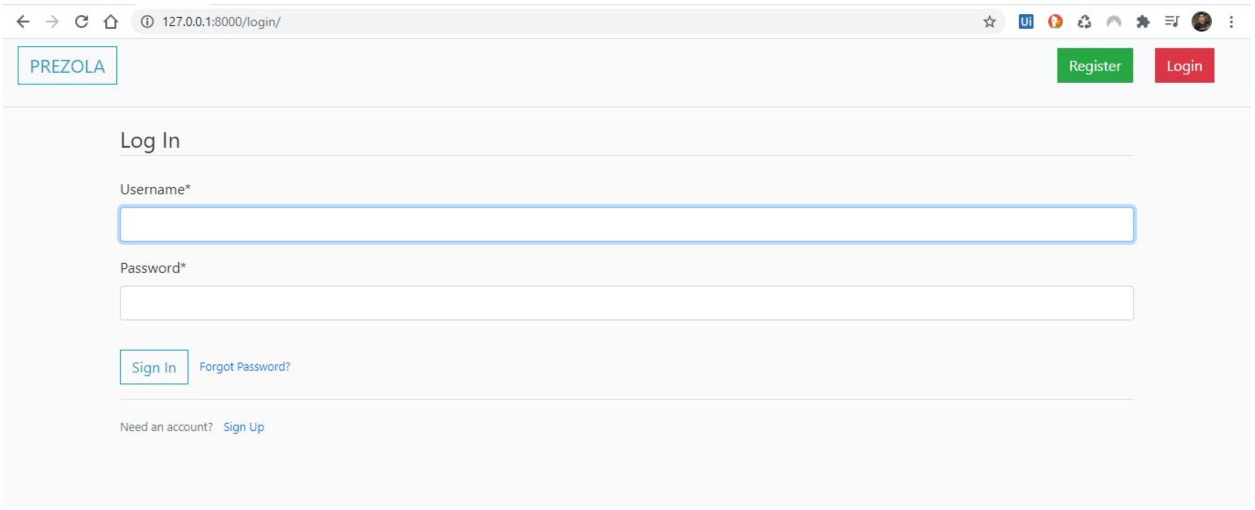
Password confirmation*

Enter the same password as before, for verification.

Sign Up

Already have an account? Sign In

Login Page



127.0.0.1:8000/login/

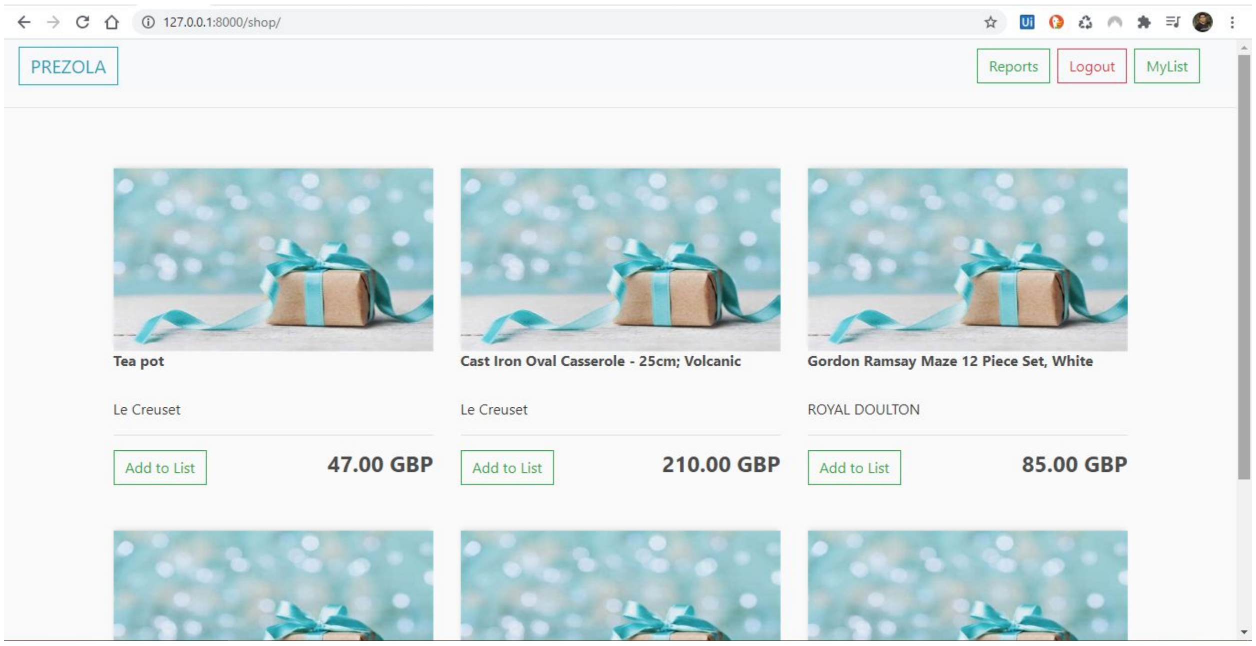
Log In

Username*

Password*

Need an account? Sign Up

Landing Page



127.0.0.1:8000/shop/

Tea pot

Le Creuset

47.00 GBP

Cast Iron Oval Casserole - 25cm; Volcanic

Le Creuset

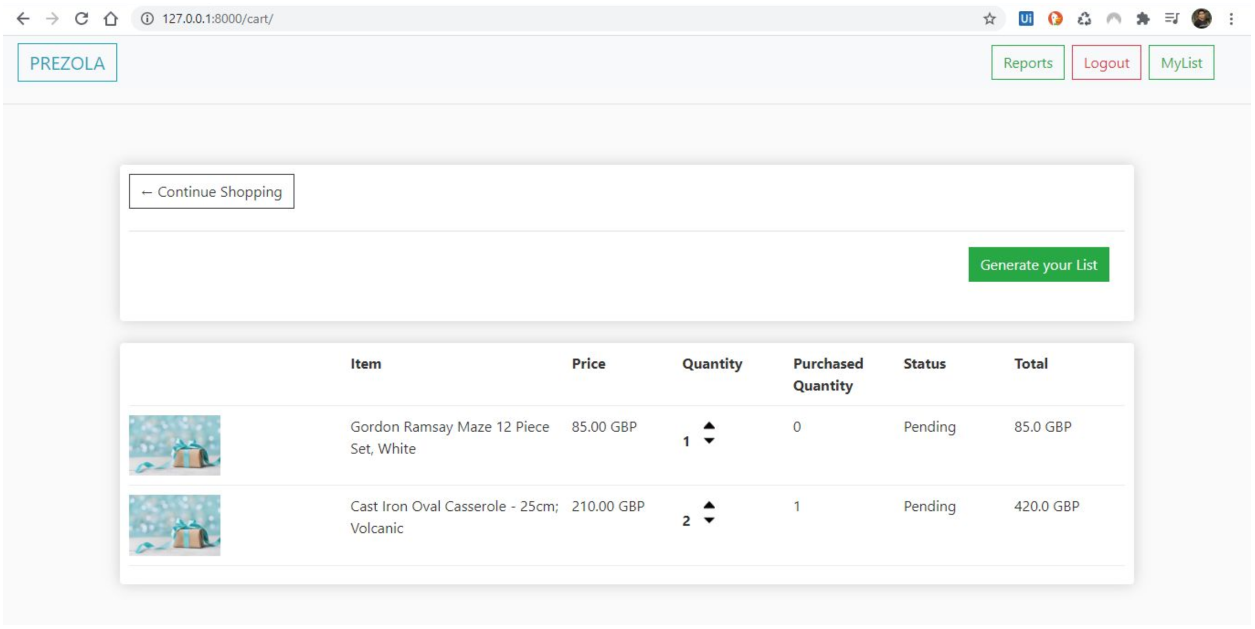
210.00 GBP

Gordon Ramsay Maze 12 Piece Set, White

ROYAL DOULTON

85.00 GBP

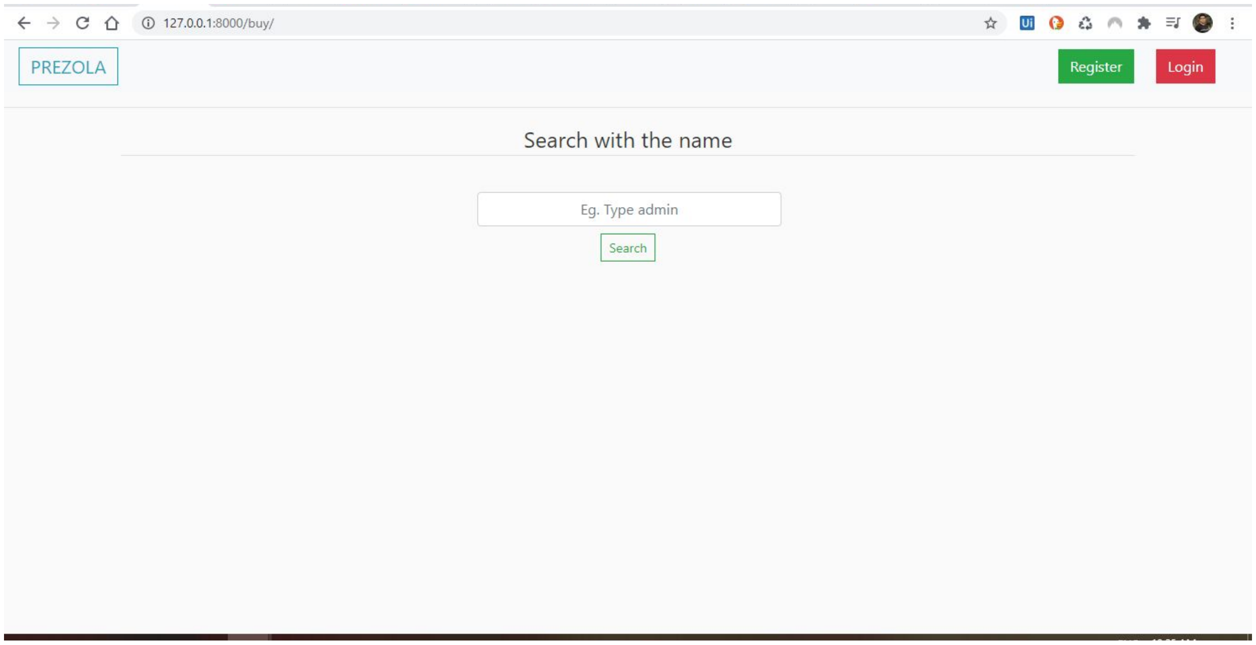
Customer Checkout View



127.0.0.1:8000/cart/

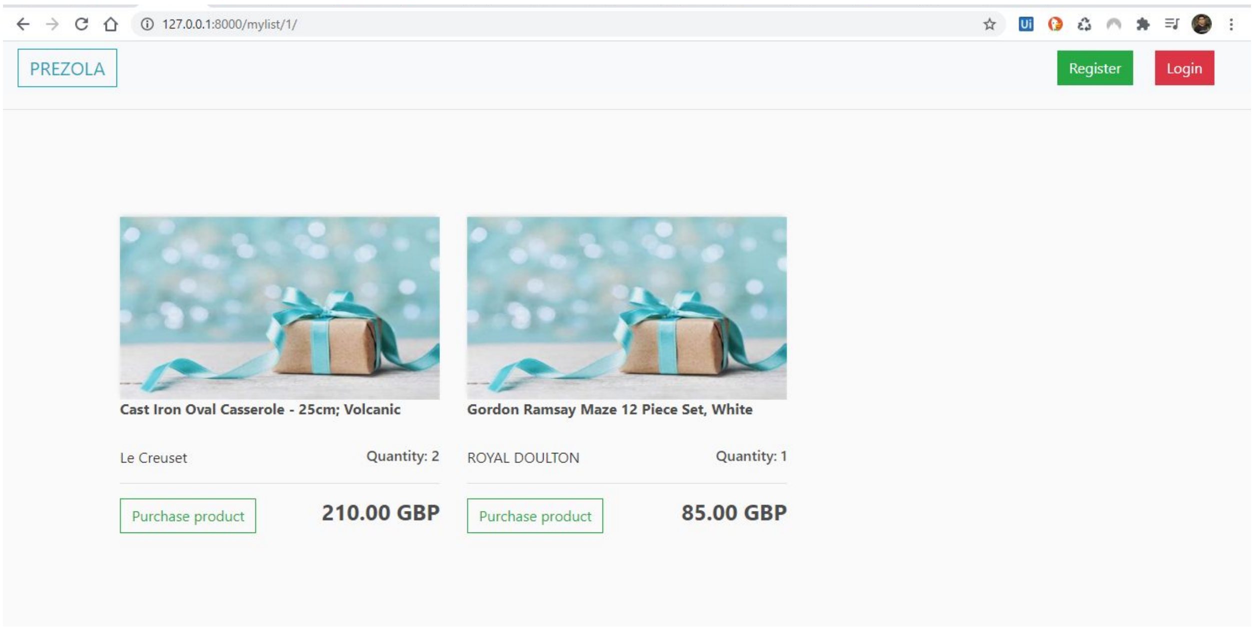
	Item	Price	Quantity	Purchased Quantity	Status	Total
	Gordon Ramsay Maze 12 Piece Set, White	85.00 GBP	1	0	Pending	85.0 GBP
	Cast Iron Oval Casserole - 25cm; Volcanic	210.00 GBP	2	1	Pending	420.0 GBP

Guest Purchase view



127.0.0.1:8000/buy/

Guest Landing View



Cast Iron Oval Casserole - 25cm; Volcanic

Le Creuset Quantity: 2

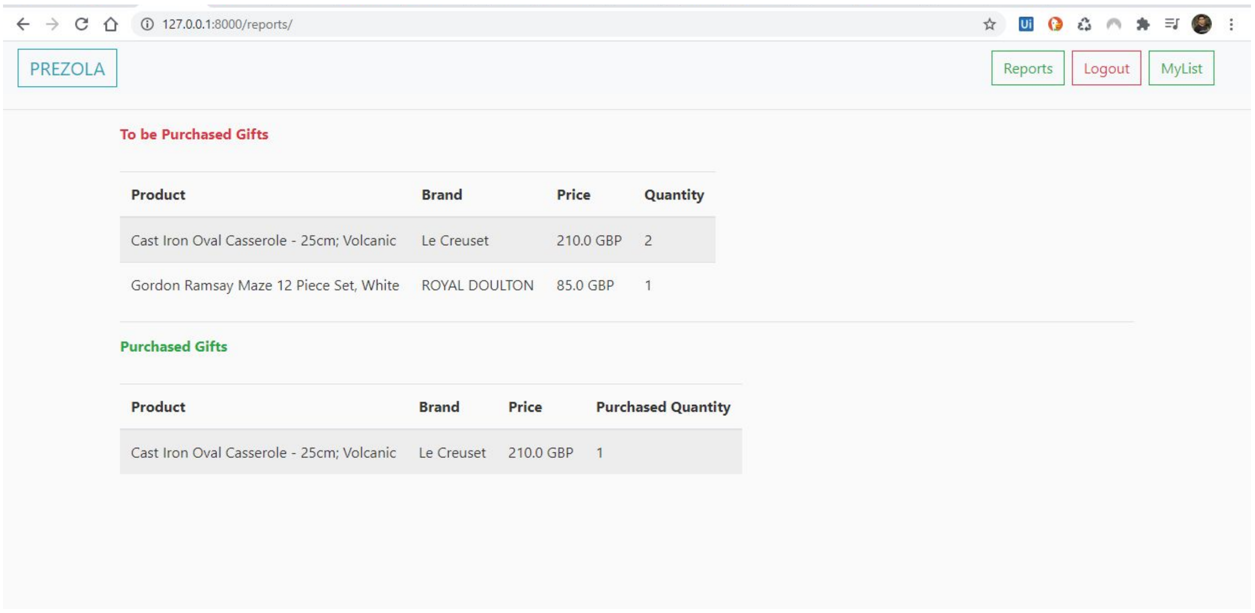
Purchase product 210.00 GBP

Gordon Ramsay Maze 12 Piece Set, White

ROYAL DOULTON Quantity: 1

Purchase product 85.00 GBP

Reports View



To be Purchased Gifts

Product	Brand	Price	Quantity
Cast Iron Oval Casserole - 25cm; Volcanic	Le Creuset	210.0 GBP	2
Gordon Ramsay Maze 12 Piece Set, White	ROYAL DOULTON	85.0 GBP	1

Purchased Gifts

Product	Brand	Price	Purchased Quantity
Cast Iron Oval Casserole - 25cm; Volcanic	Le Creuset	210.0 GBP	1

Views -

We will be using function based views to handle all the logic for us.

The idea is to ignore the django out of box functionality (Class Based views) as stated in the Technical test requirement document.

Excerpt from the requirement document -
Stay away from frameworks/boilerplates that handle everything for you - or use them only as a thin layer - so we can see how you structure applications yourself.

Home view - Displays the home page with a route to home.html template.
Template - home.html
url - home

Shop view -
Use the @login_required decorator to ensure only logged in users can access the landing page of the website.

- Template - shop.html
- url - shop

Add_remove_products -
Update order line items of a customer. Handles the orders and line items status logic and purchased quantity updates.

- Template - shop.html/ cart.html
- redirect url's - shop, cart

cart -
Displays the cart data for a specific user.

- Template - cart.html
- url - cart

mylist-
Displays the list of products selected by the customer.

- Template - shop.html
- url - shop

buy-
Display the list of products for the guest to purchase.

- Template - shop.html
- url - shop

Future Enhancements -

1. Guest/Anonymous user checkout option.
2. Create a blog and allow the couples and their guests to share the event updates
3. Surprise us option. (Allowing the users to surprise the couple with their own choice of gifts from the products available in the website.

Existing Code enhancement -

1. Modify Product price data type to DemicalField
2. Use Javascript and Ajax call for client server interaction.
3. Proper unit tests to have the full test coverages.