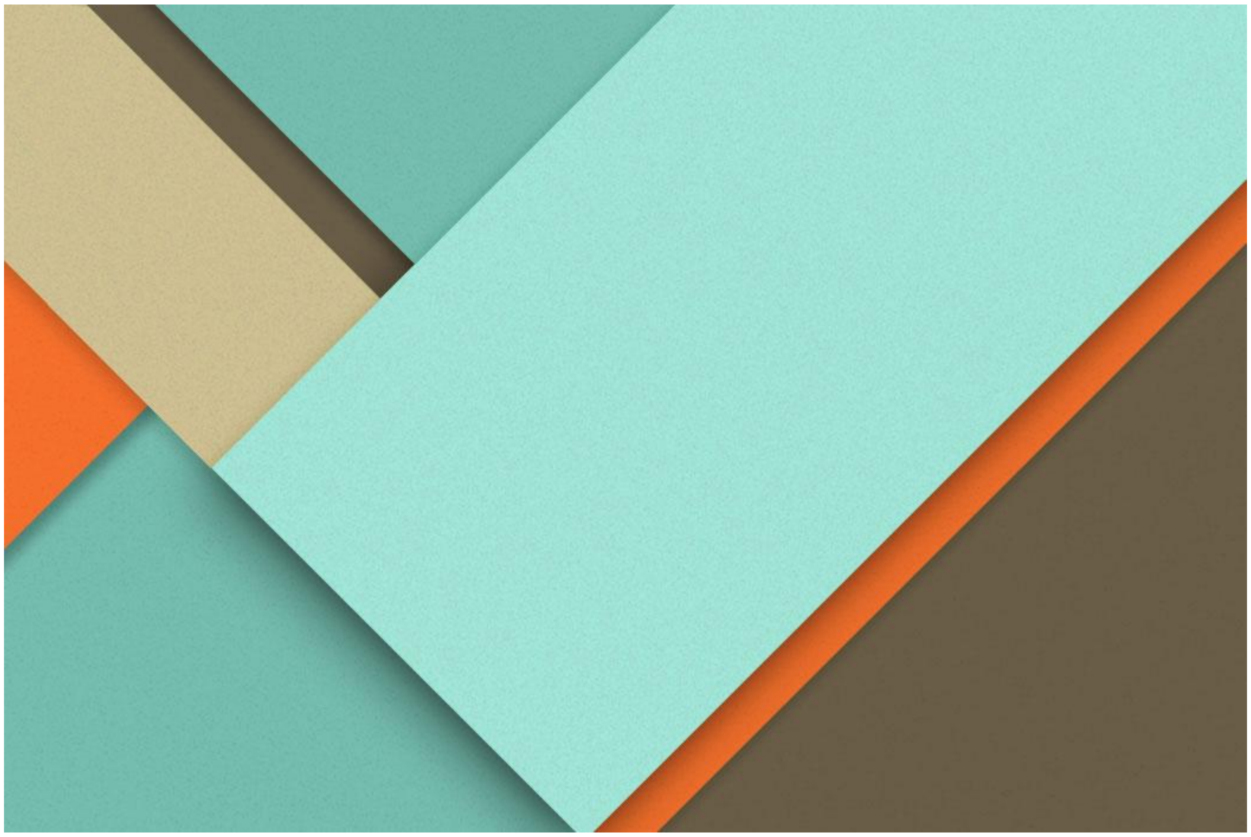


SOFTWARE TESTING REPORT

Essentials

OVERVIEW

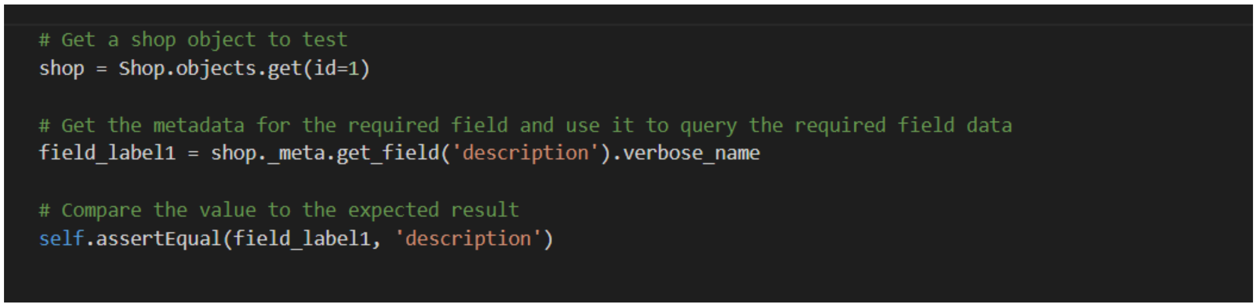
This project endeavors to create an application which will digitize the delivery of daily essential products. The scope of such an online marketplace is huge because every household requires daily essential products, in turn increasing the need for such applications. Convenience and quickness are the most impressive advantages of the system. It also removes the ambiguity in the billing.

TESTING

UNIT TESTING

MODELS

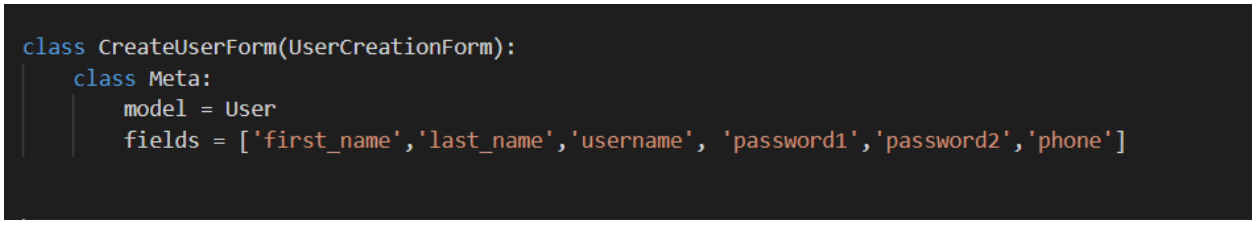
In django models we should test the labels for all the fields, because even though we haven't explicitly specified most of them, we have a design that says what these values should be. If we don't test the values, then we don't know that the field labels have their intended values.


```
# Get a shop object to test
shop = Shop.objects.get(id=1)

# Get the metadata for the required field and use it to query the required field data
field_label1 = shop._meta.get_field('description').verbose_name

# Compare the value to the expected result
self.assertEqual(field_label1, 'description')
```

This unit test case snippet checks that the values of the field labels (verbose_name) and that the content and size of the Shop's text is as expected.

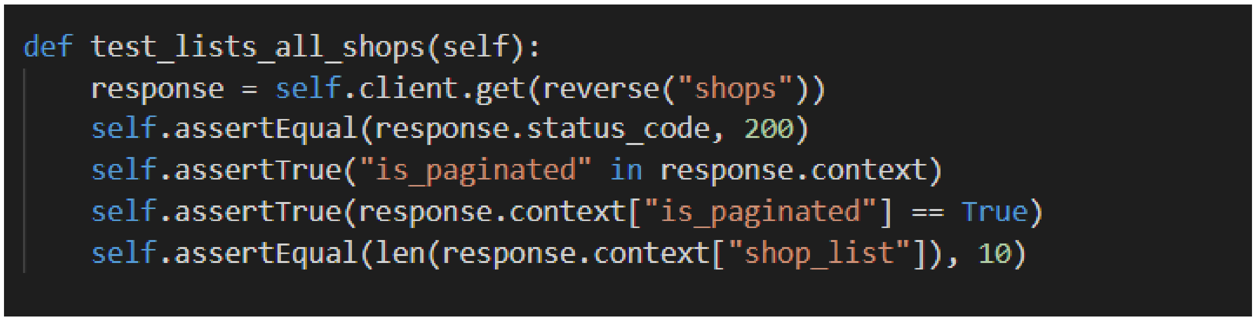
FORMS

We performed form testing to check all additional validation that is expected to be performed on the fields and any messages that code will generate for errors. The below unit test case snippet is for search form . We ensure that the user entered all the required fields. Then a check is performed whether the entered data is valid or not , if it's invalid appropriate errors are thrown, else the search will be performed. This way there will not be any unnecessary api calls and the service time on our website will be faster as only genuine requests will be completed .


```
class CreateUserForm(UserCreationForm):
    class Meta:
        model = User
        fields = ['first_name','last_name','username', 'password1','password2','phone']
```

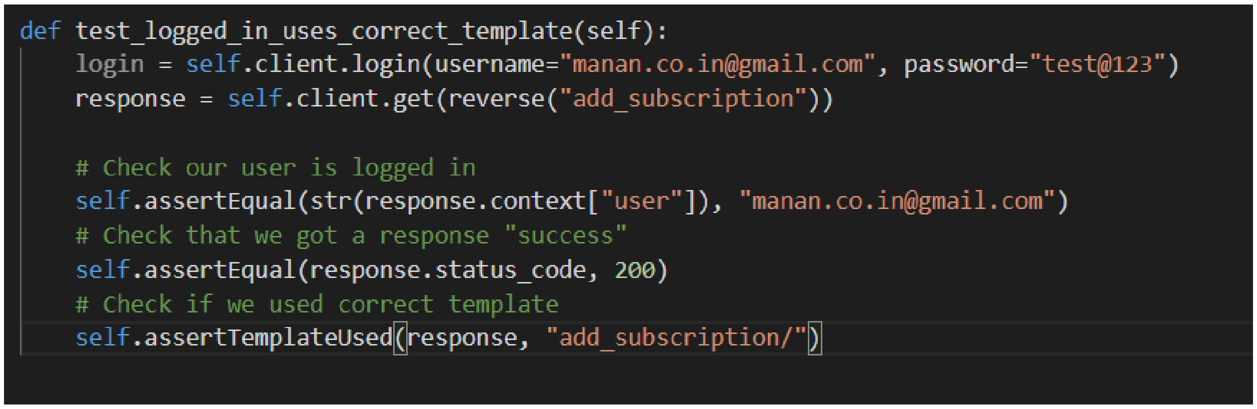
VIEWS

View unit testing allows us to confirm that our template is getting all the data it needs. In other words we can check that we're using the intended template and what data the template is getting, which goes a long way to verifying that any rendering issues are solely due to the template.



```python
def test_lists_all_shops(self):
    response = self.client.get(reverse("shops"))
    self.assertEqual(response.status_code, 200)
    self.assertTrue("is_paginated" in response.context)
    self.assertTrue(response.context["is_paginated"] == True)
    self.assertEqual(len(response.context["shop_list"]), 10)
```

Below is the snippet attached for those views in our website that are only restricted to logged in users like New post view, direct messaging view, donation view etc. So by testing we have made sure that only logged in users can access that page and redirect all the anonymous users to the login page.



```python
def test_logged_in_uses_correct_template(self):
    login = self.client.login(username="manan.co.in@gmail.com", password="test@123")
    response = self.client.get(reverse("add_subscription"))

    # Check our user is logged in
    self.assertEqual(str(response.context["user"]), "manan.co.in@gmail.com")
    # Check that we got a response "success"
    self.assertEqual(response.status_code, 200)
    # Check if we used correct template
    self.assertTemplateUsed(response, "add_subscription/")
```

MANUAL TESTING

Three test accounts were created to test the flow of entire web application

Buyer Account

- Multiple subscriptions were created for the buyers.
- Subscriptions were then added to cart and edited and deleted multiple times.
- Shops were searched on the basis of category.
- Shops were searched on the basis of location.
- Shops were searched on the basis of name.

Seller Account

- Shop was registered by the seller.
- Products were added for the shop.
- Notifications were sent to the buyer.
- Subscription was modified by the seller as per availability.
- Order was modified by the seller as per availability .

Admin

- Verified the registration details of all sellers.
- Only after account verification by admin, sellers can set up their shop.
- Tracked any suspicious recurrent transactions from the same account.
- 2 Test shops were created and test products were added and linked to them.
- CRUD operations were performed on every entity.

Taking the account through the complete process from signing up to testing every feature ensured that none of the features were breaking in production and that they were all working properly.

We also ran the tests in various devices and resolutions to make sure the UI did not vary incorrectly. This process was repeated with varying the inputs and corner case inputs to check for integrity of the application. We employed various techniques to test input validation, including bug checks.

INTEGRATION TESTING

We ran all unit test modules. We found that all the modules are loosely coupled and don't require separate integration testing. To perform manual integration testing,we have individually tested all the modules and made sure that they are working properly in combination with each other. We prepared a top-down flowchart of all the modules to check the flow of our web application.

SYSTEM TESTING

We performed manual System testing on a complete integrated system to evaluate the compliance of the system with all the requirements. We tested the design and behavior of the system and also the expectations of the customer.

1) USER REGISTRATION

TEST CASES	RESULTS
User registration with new email id	Successful user registration
User registration with already registered email id	User already existing error
User registration with incorrect email id	Enter valid email id

2) LOGIN

TEST CASES	RESULTS
Login with correct email and password	Logged in Successfully
Login with incorrect email and password	Incorrect credentials error
Login with correct email and incorrect password	Incorrect credentials error

3) CREATE SUBSCRIPTION TEST CASES RESULT

TEST CASES	RESULTS
Access create subscription without login Redirects to login page	Redirects to login page
Access create subscription after login	Create subscription page opens
Create subscription with all valid information as per the required constraints	Subscription created successfully
Invalid or missed data while creating subscription	Fill all required fields or enter valid data error

4) CART TEST CASES RESULT

TEST CASES	RESULTS
Access cart without login Redirects to login page	Redirects to login page
Access cart after login	Cart page opens
Create cart with all valid information as per the required constraints	Cart created successfully
Invalid or missed data while creating cart	Fill all required fields or enter valid data error

5) ORDERS TEST CASES RESULT

TEST CASES	RESULTS
Access orders without login Redirects to login page	Redirects to login page
Access order after login	Order page opens
Create order with all valid information as per the required constraints	Order created successfully
Invalid or missed data while creating order	Fill all required fields or enter valid data error

6) PRODUCTS TEST CASES RESULT

TEST CASES	RESULTS
Access products without login Redirects to login page	Redirects to login page
Access products after login	Product page opens
Create products with all valid information as per the required constraints	Product created successfully
Invalid or missed data while creating product	Fill all required fields or enter valid data error