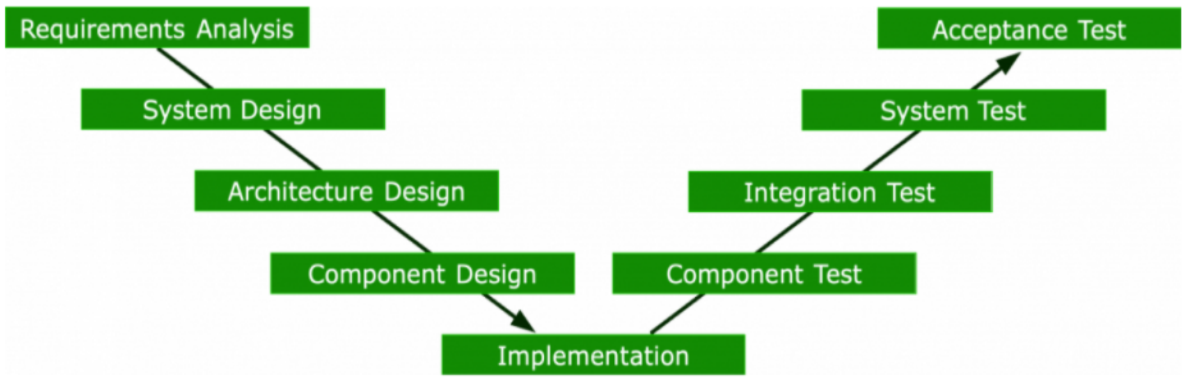
Test Cases and Automation Script :-

Made by : Mohit Ahirwar (B19CSE055) , Ram Khandelwal (B19CSE116)

OVERVIEW

We have used the V model in our Software Project and V model is a highly disciplined model of SDLC because it has the capability of parallel testing to each development phase.V model is nothing but an extension of the waterfall model and here testing is done to each of the phases and the testing is done parallely to that.

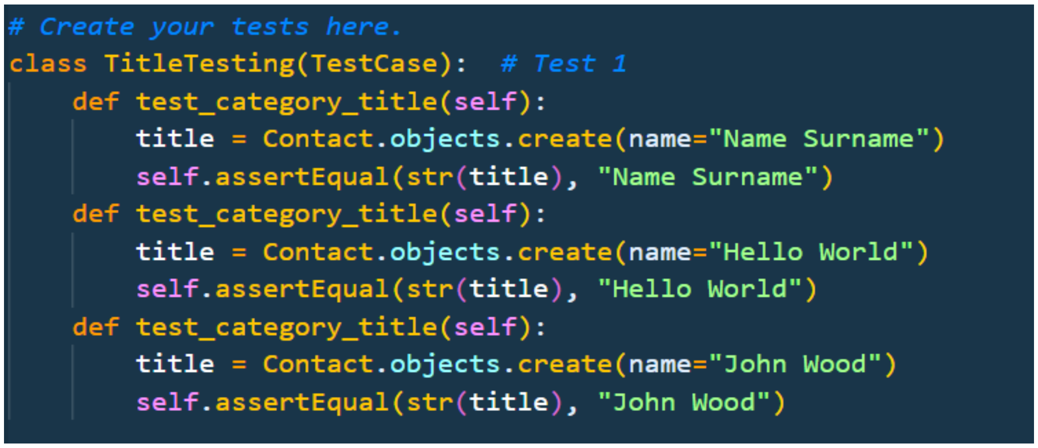


TYPES OF TESTING DONE IN OUR SOFTWARE PROJECT ACCORDING TO V-MODEL i.e, **TEST STAGES** :-

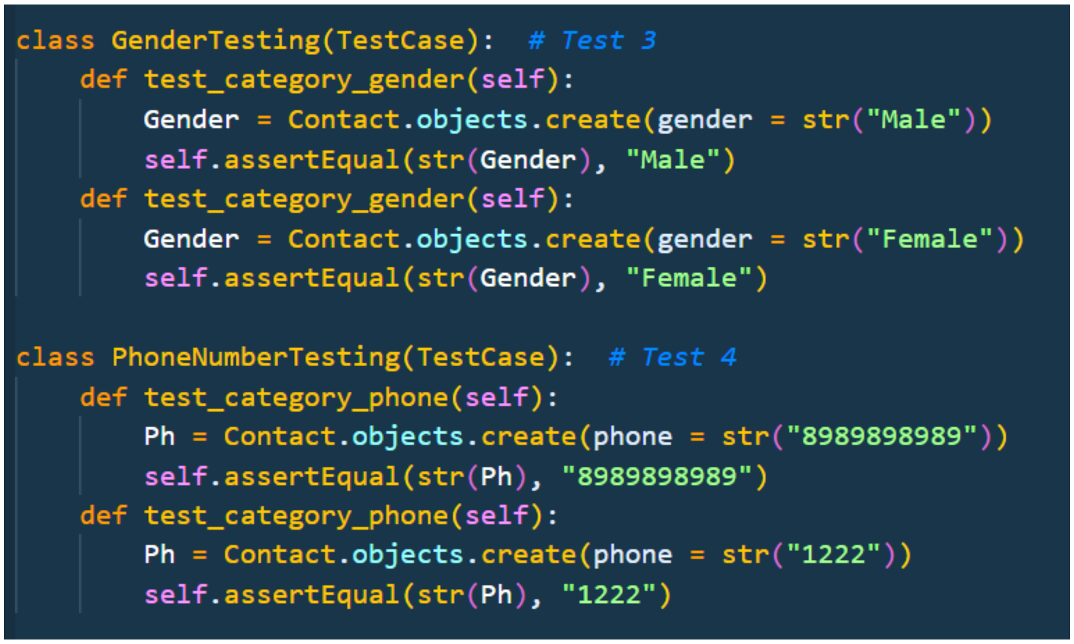
1. Unit Testing / Component Testing
2. Integration Testing
3. System Testing
4. Acceptance Testing

1. UNIT TESTING

Here we test the smallest components of our project i.e, testing modules of our software project like form filling,checking fields, email correctly filled and checking that from the database and then running the tests. We have created several test cases like Title Testing Class, Email Testing Class, Gender Testing Class and so on which is the smallest module which are unit components of our project.

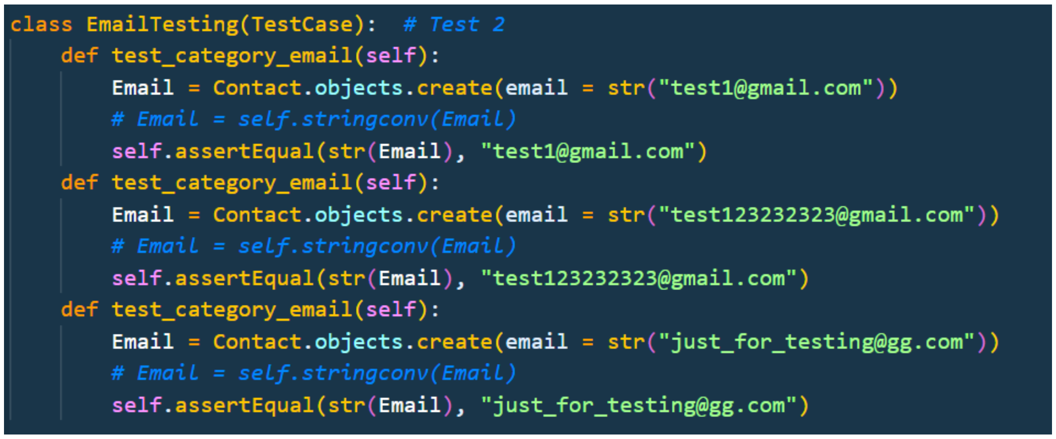


```python
# Create your tests here.
class TitleTesting(TestCase):  # Test 1
    def test_category_title(self):
        title = Contact.objects.create(name="Name Surname")
        self.assertEqual(str(title), "Name Surname")
    def test_category_title(self):
        title = Contact.objects.create(name="Hello World")
        self.assertEqual(str(title), "Hello World")
    def test_category_title(self):
        title = Contact.objects.create(name="John Wood")
        self.assertEqual(str(title), "John Wood")
```

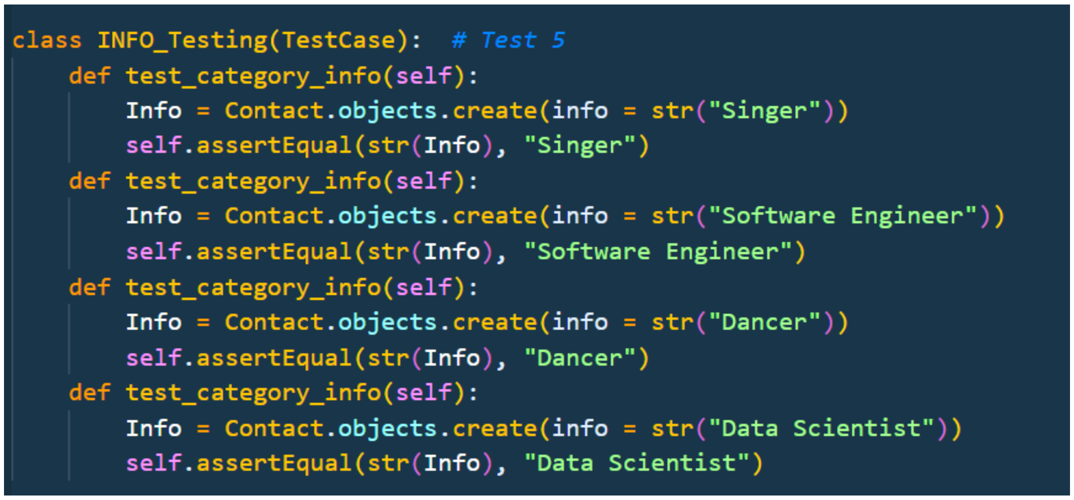


```python
class GenderTesting(TestCase):  # Test 3
    def test_category_gender(self):
        Gender = Contact.objects.create(gender = str("Male"))
        self.assertEqual(str(Gender), "Male")
    def test_category_gender(self):
        Gender = Contact.objects.create(gender = str("Female"))
        self.assertEqual(str(Gender), "Female")

class PhoneNumberTesting(TestCase):  # Test 4
    def test_category_phone(self):
        Ph = Contact.objects.create(phone = str("8989898989"))
        self.assertEqual(str(Ph), "8989898989")
    def test_category_phone(self):
        Ph = Contact.objects.create(phone = str("1222"))
        self.assertEqual(str(Ph), "1222")
```



```python
class EmailTesting(TestCase):  # Test 2
    def test_category_email(self):
        Email = Contact.objects.create(email = str("test1@gmail.com"))
        # Email = self.stringconv(Email)
        self.assertEqual(str(Email), "test1@gmail.com")
    def test_category_email(self):
        Email = Contact.objects.create(email = str("test123232323@gmail.com"))
        # Email = self.stringconv(Email)
        self.assertEqual(str(Email), "test123232323@gmail.com")
    def test_category_email(self):
        Email = Contact.objects.create(email = str("just_for_testing@gg.com"))
        # Email = self.stringconv(Email)
        self.assertEqual(str(Email), "just_for_testing@gg.com")
```

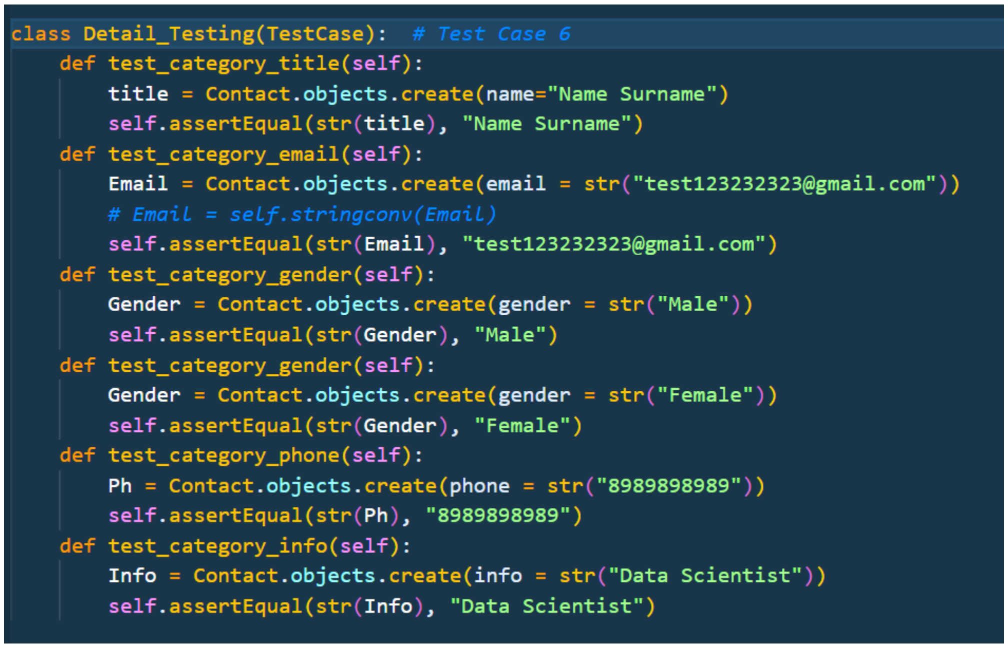


```
class INFO_Testing(TestCase):  # Test 5
    def test_category_info(self):
        Info = Contact.objects.create(info = str("Singer"))
        self.assertEqual(str(Info), "Singer")
    def test_category_info(self):
        Info = Contact.objects.create(info = str("Software Engineer"))
        self.assertEqual(str(Info), "Software Engineer")
    def test_category_info(self):
        Info = Contact.objects.create(info = str("Dancer"))
        self.assertEqual(str(Info), "Dancer")
    def test_category_info(self):
        Info = Contact.objects.create(info = str("Data Scientist"))
        self.assertEqual(str(Info), "Data Scientist")
```

2. INTEGRATION TESTING

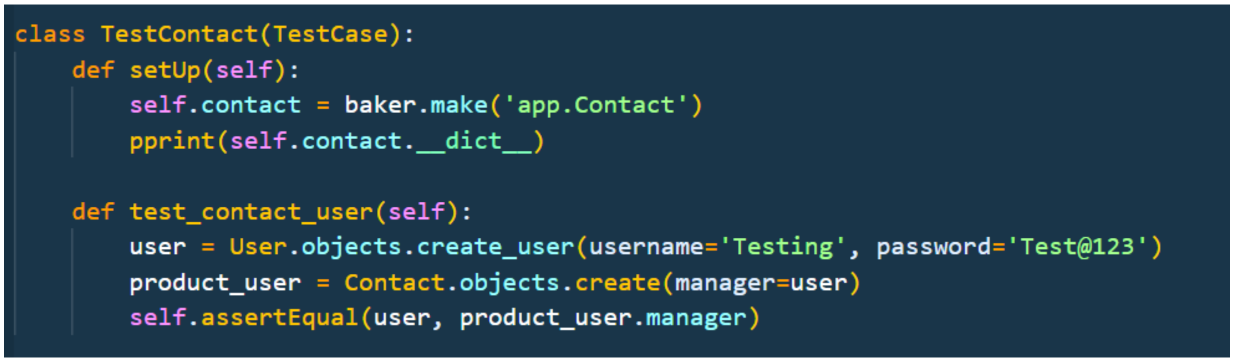
The work of integration testing is to verify that the components which are independently built and developed are working correctly i.e, testing that individual component(s) only. This testing occurs only after the completion of the unit testing.

Here we have build a class named as Details Testing so we are testing that filling a contact in our application will result into the same values stored in it so it is just verifying the details :-



```
class Detail_Testing(TestCase):  # Test Case 6
    def test_category_title(self):
        title = Contact.objects.create(name="Name Surname")
        self.assertEqual(str(title), "Name Surname")
    def test_category_email(self):
        Email = Contact.objects.create(email = str("test123232323@gmail.com"))
        # Email = self.stringconv(Email)
        self.assertEqual(str(Email), "test123232323@gmail.com")
    def test_category_gender(self):
        Gender = Contact.objects.create(gender = str("Male"))
        self.assertEqual(str(Gender), "Male")
    def test_category_gender(self):
        Gender = Contact.objects.create(gender = str("Female"))
        self.assertEqual(str(Gender), "Female")
    def test_category_phone(self):
        Ph = Contact.objects.create(phone = str("8989898989"))
        self.assertEqual(str(Ph), "8989898989")
    def test_category_info(self):
        Info = Contact.objects.create(info = str("Data Scientist"))
        self.assertEqual(str(Info), "Data Scientist")
```

Here we are testing the login module i.e, verifying the login details and credentials that are registered are able to login :-



```python
class TestContact(TestCase):
    def setUp(self):
        self.contact = baker.make('app.Contact')
        pprint(self.contact.__dict__)

    def test_contact_user(self):
        user = User.objects.create_user(username='Testing', password='Test@123')
        product_user = Contact.objects.create(manager=user)
        self.assertEqual(user, product_user.manager)
```

3. SYSTEM TESTING

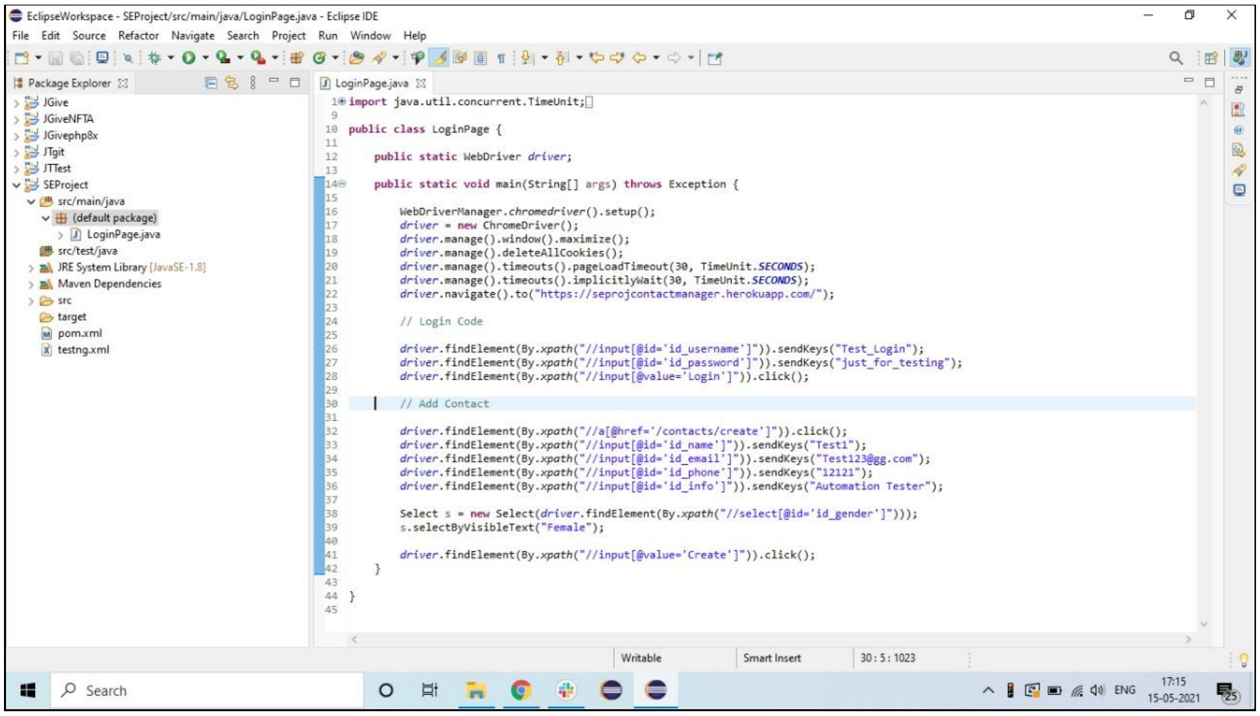
It covers the whole project/system testing. We have completed the unit testing and integration testing so now in system testing it is important to test all the modules because there are some modules which cannot be tested in integration testing and hence system testing is done.

The test environment is kept close to the environment for the final production and work on system requirements to run this application, system specifications and hardware. Since it is a web application, we don't need to bother much with the hardware specifications because nowadays all have access to the internet and a browser.

Writing Automation Scripts for System Testing :-

We are using selenium app for automation testing and writing scripts for testing our software project.

We have written a script for automation testing of login into the application and adding a contact by filling all the details and then going to the home page for further testing.



```java
import java.util.concurrent.TimeUnit;

public class LoginPage {

    public static WebDriver driver;

    public static void main(String[] args) throws Exception {

        WebDriverManager.chromedriver().setup();
        driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.manage().deleteAllCookies();
        driver.manage().timeouts().pageLoadTimeout(30, TimeUnit.SECONDS);
        driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
        driver.navigate().to("https://seprojcontactmanager.herokuapp.com/");

        // Login Code

        driver.findElement(By.xpath("//input[@id='id_username']")).sendKeys("Test_Login");
        driver.findElement(By.xpath("//input[@id='id_password']")).sendKeys("just_for_testing");
        driver.findElement(By.xpath("//input[@value='Login']")).click();

        // Add Contact

        driver.findElement(By.xpath("//a[@href='/contacts/create']")).click();
        driver.findElement(By.xpath("//input[@id='id_name']")).sendKeys("Test1");
        driver.findElement(By.xpath("//input[@id='id_email']")).sendKeys("Test123@gg.com");
        driver.findElement(By.xpath("//input[@id='id_phone']")).sendKeys("12121");
        driver.findElement(By.xpath("//input[@id='id_info']")).sendKeys("Automation Tester");

        Select s = new Select(driver.findElement(By.xpath("//select[@id='id_gender']")));
        s.selectByVisibleText("Female");

        driver.findElement(By.xpath("//input[@value='Create']")).click();
    }

}
```

Here we are using Selenium Application for automation testing and coding to open web applications and automatically tests the login functionality, homepage functionality and adding contacts by filling all the required fields and hence it tests the software in an automated manner.

Code for automation testing :-

```java
import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.Select;
import io.github.bonigarcia.wdm.WebDriverManager;

public class LoginPage
{
public static WebDriver driver;
public static void main(String[] args) throws Exception {
WebDriverManager.chromedriver().setup();
driver = new ChromeDriver();
driver.manage().window().maximize();
driver.manage().deleteAllCookies();
driver.manage().timeouts().pageLoadTimeout(30, TimeUnit.SECONDS);
driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
driver.navigate().to("https://seprojcontactmanager.herokuapp.com/");
// Given navigation path to our software project web application
// Login Code
driver.findElement(By.xpath("//input[@id='id_username']")).sendKeys("Te
st_Login");

driver.findElement(By.xpath("//input[@id='id_password']")).sendKeys("jus
t_for_testing");

driver.findElement(By.xpath("//input[@value='Login']")).click();

// Add Contact
driver.findElement(By.xpath("//a[@href='/contacts/create']")).click();

driver.findElement(By.xpath("//input[@id='id_name']")).sendKeys("Test1")
;

driver.findElement(By.xpath("//input[@id='id_email']")).sendKeys("Test12
3@gg.com");

driver.findElement(By.xpath("//input[@id='id_phone']")).sendKeys("12121"
);

driver.findElement(By.xpath("//input[@id='id_info']")).sendKeys("Automat
ion Tester");

Select s = new
Select(driver.findElement(By.xpath("//select[@id='id_gender']")));
                s.selectByVisibleText("Female");
        driver.findElement(By.xpath("//input[@value='Create']")).click();
        }
}
```

it becomes very helpful to rerun and retest the application on just one click. It is very helpful and reduces manual efforts.

Added all codes created from selenium in the zip file and also added the video demonstrating automation in that zip.

4. ACCEPTANCE TESTING

Acceptance testing are mostly manual tests but there may be some automated parts. This is the testing done only once when we are completed with the development part. It majorly focuses on the workflow of the application, user interaction. Using test automation saves a lot of time and we can use it here also.These tests can be external or internal based on end-user or client.To decide the final release feedback is also taken from the customers i.e, to cover usability, user friendliness and user convenient aspects of the application.

Stakeholder	**Agree/Disagree**
Requirements all fulfilled	Agreed except one
Successfully deployed	Agreed
Performance as expected	Agreed
Quality attributes focused	Agreed
Operational Web app	Agreed