TEST AUTOMATION (UCS662)

FOOD ON DOORS (FOOD DELIVERY WEBSITE)

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

SUBMITTED BY:

KASHVI JAIN(102153038)

NARINDER KAUR(102013190)

AASHI MEHTA(102103843)

BE Third Year, 3CO28

SUBMITTED TO:
Riya Sharma Ma’am

Computer Science and Engineering Department
TIET, Patiala
May 2024

TABLE OF CONTENTS:

ACKNOWLEDGEMENT

We are incredibly grateful to Thapar Institute of Engineering and Technology for their support and guidance throughout the development of "Food On Doors." Their endorsement and collaboration have been instrumental in bringing this innovative solution to fruition.

We extend our sincere appreciation to Ma’am Riya Sharma for her invaluable guidance and encouragement throughout the project. Her expertise in technology and unwavering support were crucial in shaping the core concept of "Food On Doors" and ensuring its alignment with the needs of the Thapar University community.

Finally, heartfelt thanks to our families and friends for their unwavering support. Their belief in us fueled our motivation and perseverance, allowing us to reach this milestone. We are confident that "Food On Doors" will revolutionize the campus dining experience and create a positive impact at Thapar University.

INTRODUCTION

Title: Enhancing Campus Convenience: Introducing Food On Doors - A Seamless Food Website

- In the hustle and bustle of university life, convenience is paramount. With this ethos in mind, we proudly present "Food On Doors," a pioneering food delivery website tailored specifically for the vibrant community of Thapar University. Our platform aims to revolutionize the dining experience on campus, offering a seamless and efficient solution for students and faculty to satisfy their culinary cravings without leaving the comfort of their dorms or offices.
- This report serves as a comprehensive overview of the development, testing, and implementation of Food On Doors.

• KEY FEATURES:

1. Website Development: An insight into the inception and evolution of Food On Doors, detailing the design principles, functionality, and user interface considerations that have shaped its development.

2. Testing Methodology: A comprehensive overview of our testing framework, including the utilization of Selenium and TestNG to conduct automated tests. We detail our testing strategy, scenarios, and methodologies adopted to ensure robust performance and reliability.

3. Test Results and Analysis: A detailed examination of test results, encompassing performance metrics, bug reports, and user feedback. Through this analysis, we identify areas of improvement and refinement, crucial for enhancing the overall user experience.

4. Ant Reports Generation: Exploring the process of generating Ant reports to facilitate comprehensive documentation and analysis of test results. These reports serve as a valuable tool for stakeholders, providing insights into the efficacy of our testing procedures and the overall health of the website.

PROBLEM STATEMENT:

Traditional food delivery services often lead to frustrations like order inaccuracies and delayed deliveries. Our challenge is to mitigate these issues by developing a ReactJS-based web application. This solution will streamline browsing and ordering. Ensuring reliability across various platforms and browsers, we'll use Selenium and TestNG testing methodologies and use tool ANT to generate tools.

SPECIAL REQUIREMENTS:

Based on the problem statement provided, here are the functional requirements for the food delivery web application:

Functional Requirements

- User Registration and Authentication:

  - Allow users to create accounts and log in securely.
  - Implement authentication mechanisms to ensure data privacy and security.

- Restaurant Browsing

  - Provide a user-friendly interface for browsing nearby restaurants.
  - Display restaurant details including menus, prices and rating.

- Order Placement

  - Enable users to add items to their cart and place orders seamlessly.
  - Allow customization of orders (e.g., adding special instructions, selecting meal options).

- Payment Integration

  - Integrate secure payment gateways to facilitate online payments.
  - Support various payment methods such as credit/debit cards, digital wallets, and cash on delivery.

- Feedback and Support:

  - Provide a mechanism for users to leave feedback and reviews for restaurants and their overall experience.
  - Offer customer support

NON-FUNCTIONAL REQUIREMENTS

- Performance:

  - Ensure fast loading times, even during peak usage periods.
  - Minimize latency in order processing.

- Scalability:

  - Design the application architecture to handle increasing user traffic and restaurant partnerships.

- Reliability:

  - Maintain high availability of the application with minimal downtime.

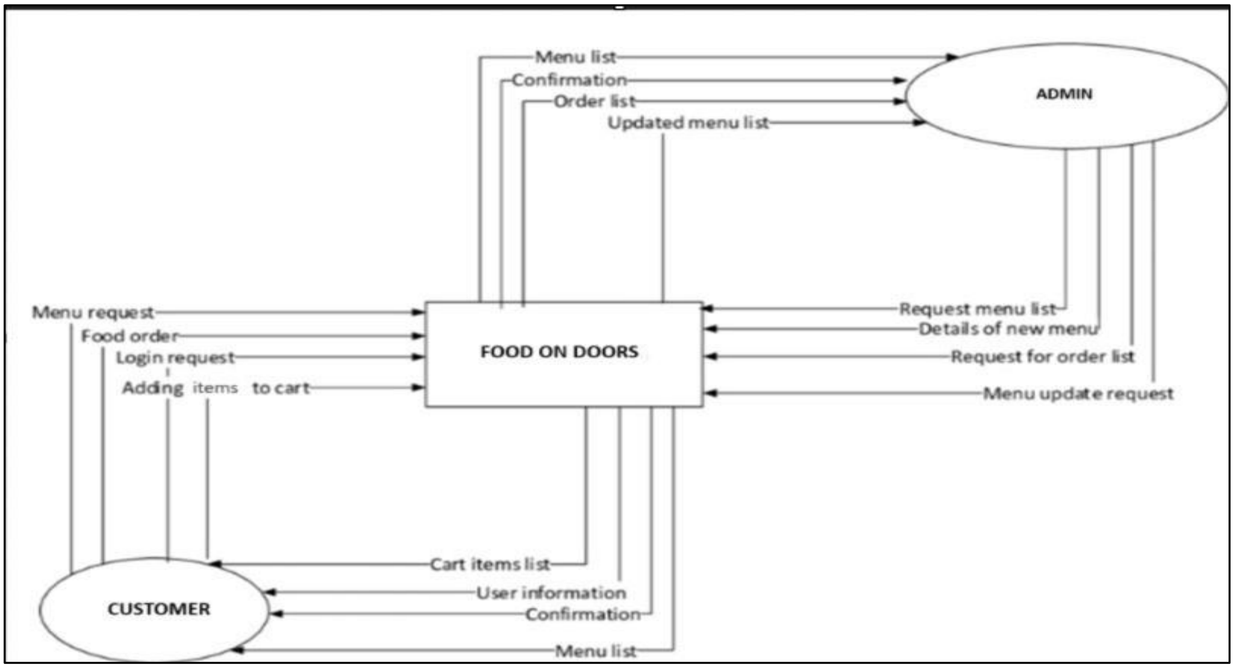
- Security:

  - Implement robust security measures to protect user data and payment information.

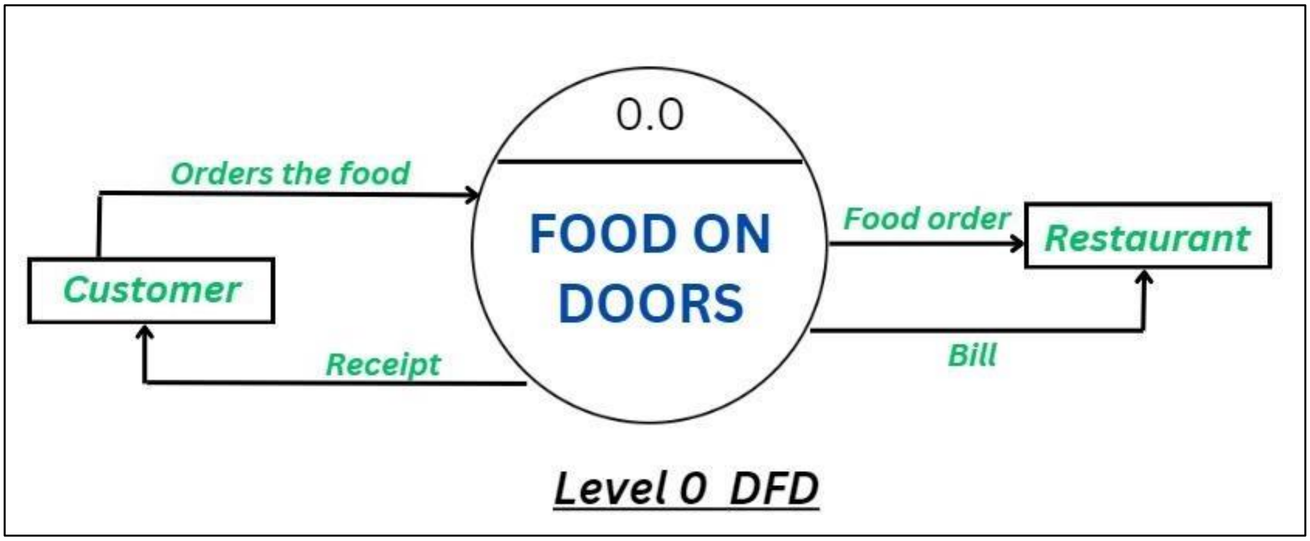
- Maintainability:

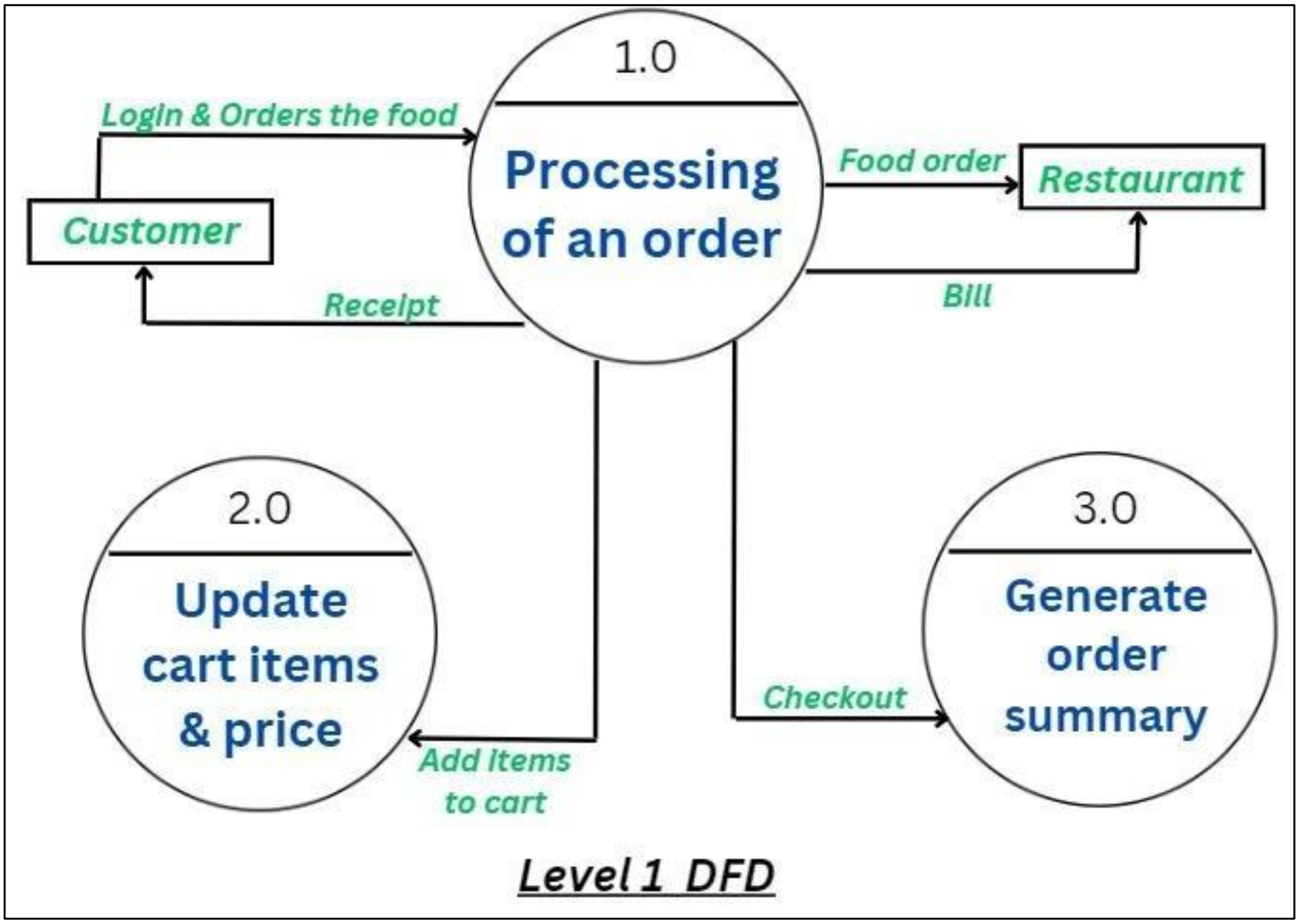
  - Write clean and modular code following best practices and coding standards.
  - Implement version control and documentation to facilitate future updates and maintenance tasks.

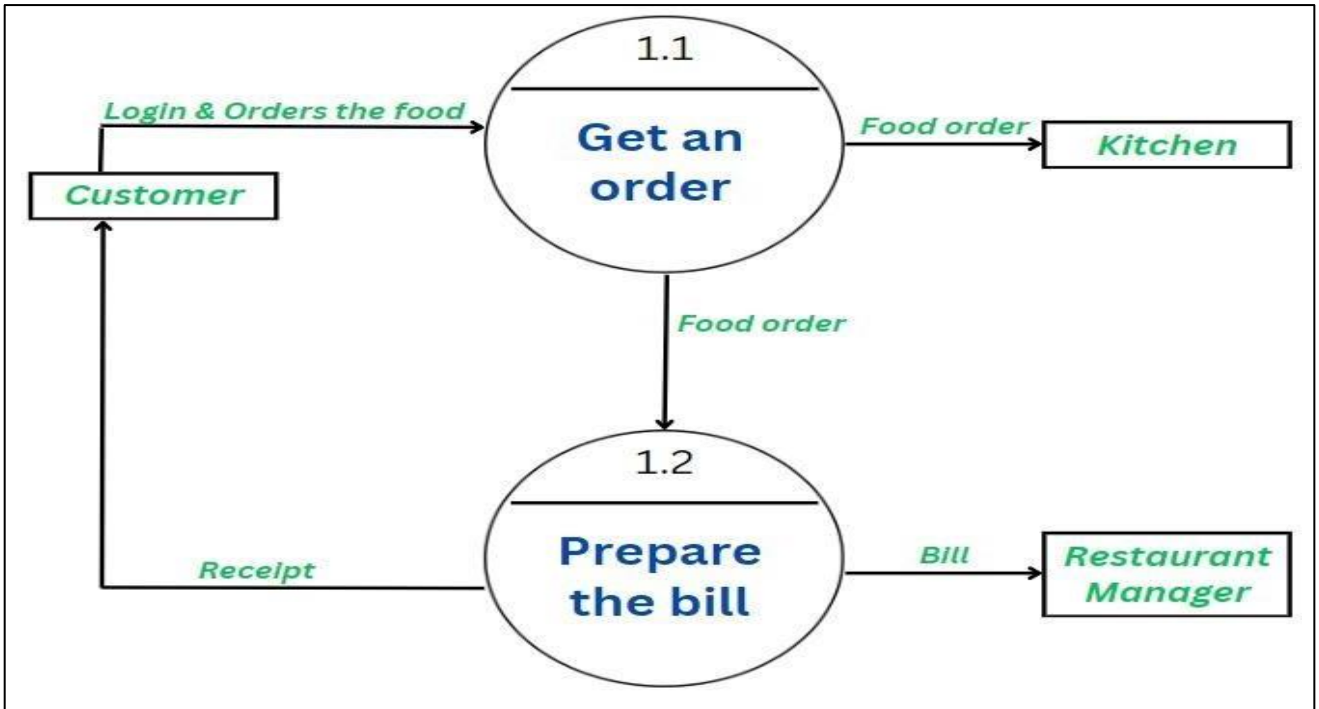
CONTEXT DIAGRAM:



DATA-FLOW DIAGRAM:






1.1
Get an order
Login & Orders the food
Customer
Food order
Kitchen
Food order
1.2
Prepare the bill
Receipt
Bill
Restaurant Manager

Level 2 DFD

System Specifications:

Hardware:

Server Infrastructure:

- High-performance servers with multi-core processors and sufficient RAM to handle concurrent user requests.
- SSD storage for fast data retrieval and reduced latency, ensuring smooth browsing and ordering experience.
- Reliable network infrastructure with adequate bandwidth to support seamless communication between clients and servers.

Software:

Programming Languages and Frameworks:

- Node.js and java script for server-side application logic and real-time communication between clients and servers, enabling efficient data exchange.

Testing Tools:

- Selenium for automated testing of web applications across different browsers, ensuring compatibility and consistency.
- TestNG for organizing and running test cases and suites in Java, providing a robust testing framework for comprehensive testing coverage.

Version Control:

- Git for version control to track changes in the application codebase and facilitate collaboration among developers.
- Hosting repositories on platforms like GitHub or Bitbucket for code management and version tracking, ensuring code integrity and transparency.

Tools Used:

Selenium code

```
package lklk;
import java.time.Duration;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.Alert;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;

public class khhh {
        @SuppressWarnings("deprecation")
        public static void main(String[] args)
        {
                System.setProperty("webdriver.chrome.driver",     "C:\\Users\\Punit\\eclipse\\jee-2023-
12\\eclipse\\chromedriver.exe");
                WebDriver driver = new ChromeDriver();
                driver.manage().window().maximize();
                driver.manage().timeouts().implicitlyWait(50, TimeUnit.SECONDS);

                driver.get("http://127.0.0.1:5500/index.html");
                WebElement signupButton = driver.findElement(By.xpath("/html/body/div/a[1]"));
                signupButton.click();
                WebElement emailInput = driver.findElement(By.id("signup-email"));
     // Enter an invalid email address
```

```
emailInput.sendKeys("invalidemail");
WebElement passwordInput = driver.findElement(By.id("signup-password"));
String password = "weakpassword";

passwordInput.sendKeys(password);
try {
    Thread.sleep(2000); // Adding a small wait for the alert to appear (you can replace this with WebDriverWait)
} catch (InterruptedException e) {
    e.printStackTrace();
}
boolean isValidPassword = validatePassword(password);

if(isValidPassword) {
    System.out.println("Password is valid!");
    driver.get("http://127.0.0.1:5500/index1.html");
} else
    System.out.println("Password is invalid!");
driver.navigate().refresh();
WebElement emailInput1 = driver.findElement(By.id("signup-email"));

emailInput1.sendKeys("kjain2_be21@thapar.edu");
// Find the password input field
WebElement passwordInput1 = driver.findElement(By.id("signup-password"));

// Enter a password that meets the specified constraints
passwordInput1.sendKeys("Test@123");

// Click the sign-up button
WebElement signUpButton1 = driver.findElement(By.xpath("//button[contains(text(),'Sign Up')]"));
signUpButton1.click();
```

```
        // Wait for the alert to appear
        try {
            Thread.sleep(2000); // Adding a small wait for the alert to appear (you can replace this with
WebDriverWait)
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        Alert alert = driver.switchTo().alert();
        alert.accept();

      WebElement addressElement = driver.findElement(By.id("add-address"));
      addressElement.click();

      // Wait for the alert to appear
      WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(8000));
      wait.until(ExpectedConditions.alertIsPresent());

      // Switch to the alert, enter "ThaparUniversity" in the input field, and accept the alert
      driver.switchTo().alert().sendKeys("Thapar University");
      driver.switchTo().alert().accept();

      // Wait for the alert to disappear
      wait.until(ExpectedConditions.not(ExpectedConditions.alertIsPresent()));

      // Find the "Your Address" box and get its text
      WebElement yourAddressElement = driver.findElement(By.className("your-address")); //
Assuming "your-address" is the class of the "Your Address" box
      String addressText = yourAddressElement.getText();
      System.out.println("Address displayed in 'Your Address' box: " + addressText);

      wait.until(ExpectedConditions.textToBePresentInElement(yourAddressElement, "Thapar
University"));
```

```
    System.out.println(driver.getTitle());
    try {
      Thread.sleep(5000); // Adding a small wait for the alert to appear (you can replace this with WebDriverWait)
    } catch (InterruptedException e) {
      e.printStackTrace();
    }

    WebElement LogoutButton = driver.findElement(By.xpath("//*[@id=\"circle\"]/a"));
        LogoutButton.click();
        try {
      Thread.sleep(5000); // Adding a small wait for the alert to appear (you can replace this with WebDriverWait)
    } catch (InterruptedException e) {
      e.printStackTrace();
    }

        WebElement LoginButton = driver.findElement(By.xpath("/html/body/div/a[2]"));
        LoginButton.click();
        try {
      Thread.sleep(5000); // Adding a small wait for the alert to appear (you can replace this with WebDriverWait)
    } catch (InterruptedException e) {
      e.printStackTrace();
    }

        WebElement emailInput2 = driver.findElement(By.id("login-email"));
    emailInput2.sendKeys("kjain2_be21@thapar.edu");
    WebElement passwordInput2 = driver.findElement(By.id("login-password"));
    passwordInput2.sendKeys("Test@123");
    WebElement Login = driver.findElement(By.xpath("//*[@id=\"login-form\"]/button "));
        Login.click();
        try {
```

```
      Thread.sleep(5000); // Adding a small wait for the alert to appear (you can replace this with
WebDriverWait)
    } catch (InterruptedException e) {
      e.printStackTrace();
    }
         Alert alert2 = driver.switchTo().alert();
    alert2.accept();

    driver.quit();


        }
        public static boolean validatePassword(String password) {
          // Validate the password based on the provided constraints
          String regex = "^(?=.*[0-9])(?=.*[a-z])(?=.*[A-Z])(?=.*[@#$%^&+=!()*-
])(?=\\S+$).{8,15}$";
          return password.matches(regex);
        }
  }
```

TestNG code:

```
package Tests;

import org.openqa.selenium.By;
import org.openqa.selenium.JavascriptExecutor;
//import org.openqa.selenium.Keys;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
//import org.openqa.selenium.support.ui.WebDriverWait;
//import org.testng.annotations.AfterMethod;
import org.testng.annotations.AfterTest;
//import org.testng.annotations.BeforeMethod;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.DataProvider;
import org.testng.annotations.Test;

//import io.github.bonigarcia.wdm.WebDriverManager;

public class FirstTest {
	private WebDriver driver;

	@BeforeTest
	public void setUp() {
		driver=new ChromeDriver();
		driver.manage().window().maximize();
	}
	@Test(priority=1, dataProvider="Register")
	public void projectSignUptest(String email, String pswd) throws InterruptedException {
		driver.get("http://127.0.0.1:5500/index.html");
		driver.findElement(By.xpath("/html/body/div/a[1]")).click();
		driver.findElement(By.id("signup-email")).sendKeys(email);
		Thread.sleep(1000);
```

```
        driver.findElement(By.id("signup-password")).sendKeys(pswd);
        Thread.sleep(1000);
        try {
              driver.findElement(By.xpath("//*[@id=\"signup-form\"]/button")).click();
          Thread.sleep(2000);
          if (driver.switchTo().alert().getText().equals("All fields are required")) {
            // Handle the expected alert
            driver.switchTo().alert().accept();
          }
          if(driver.switchTo().alert().getText().equals("Email already exists")) {
              driver.switchTo().alert().accept();
          }
          if(driver.switchTo().alert().getText().equals("Password must be between 8 and 15
characters, contain at least one letter, one number, and one special character (!@#$%^&*()_+), and
should not start with a special character.")) {
              driver.switchTo().alert().accept();
          }
          if(driver.switchTo().alert().getText().equals("Signed up successfully!")) {
              driver.switchTo().alert().accept();
              driver.findElement(By.xpath("//*[@id=\"circle\"]/a")).click();
                Thread.sleep(2000);
          }else{
            // Handle unexpected behavior (no alert or different alert message)
            System.err.println("Unexpected alert content or no alert found.");
          }
        } catch (Exception e) {
          // Catch other potential exceptions
          System.err.println("Unexpected error occurred: " + e.getMessage());
          // Log or handle the error as needed
        }
    }

    @Test(dependsOnMethods = "projectSignUptest",dataProvider="Login")
    public void projectLogintest(String email, String pswd) throws InterruptedException {
        driver.findElement(By.xpath("/html/body/div/a[2]")).click();
        driver.findElement(By.id("login-email")).sendKeys(email);
        Thread.sleep(1000);
        driver.findElement(By.id("login-password")).sendKeys(pswd);
        Thread.sleep(1000);
        try {
            driver.findElement(By.xpath("//*[@id=\"login-form\"]/button")).click();
         Thread.sleep(2000);
         if (driver.switchTo().alert().getText().equals("Invalid email or password. Please try again.")) {
           // Handle the expected alert
           driver.switchTo().alert().accept();
         }
         if(driver.switchTo().alert().getText().equals("Logged in successfully!")) {
            driver.switchTo().alert().accept();
            //driver.findElement(By.xpath("//*[@id=\"circle\"]/a")).click();
             Thread.sleep(2000);
         }else{
           // Handle unexpected behavior (no alert or different alert message)
           System.err.println("Unexpected alert content or no alert found.");
         }
        } catch (Exception e) {
         // Catch other potential exceptions
         System.err.println("Unexpected error occurred: " + e.getMessage());
         // Log or handle the error as needed
        }
    }

    @Test(dependsOnMethods = "projectLogintest",dataProvider="details")
    public void websitetesting(String cardnum, String exp, String cvv) throws InterruptedException {
        JavascriptExecutor js = (JavascriptExecutor) driver;
```

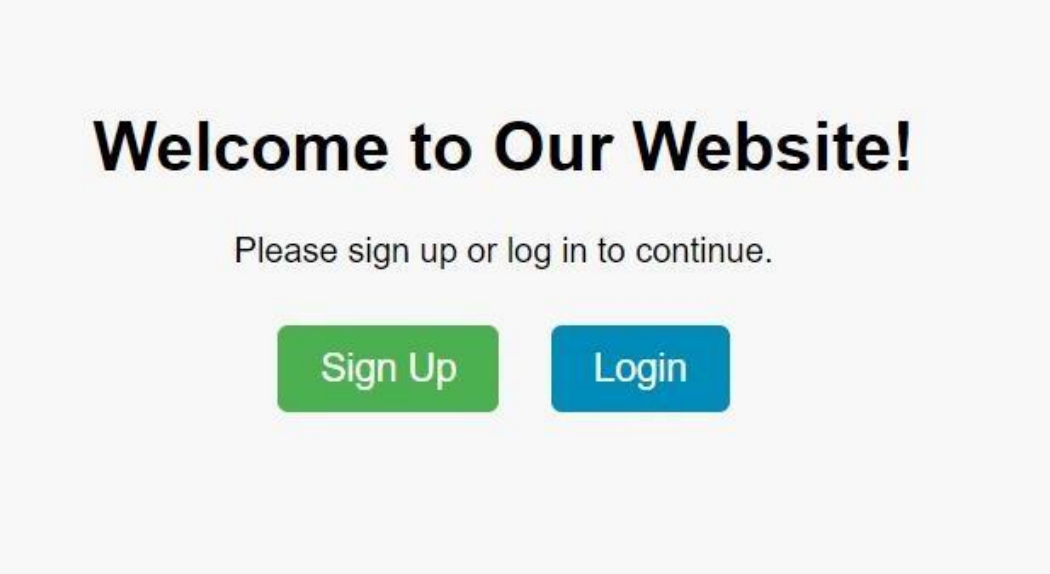
```
// Scroll down by 10000 pixels
js.executeScript("window.scrollBy(0, 10000)");
Thread.sleep(500);
js.executeScript("window.scrollBy(0, -10000)");
Thread.sleep(300);
driver.findElement(By.xpath("//*[@id=\"category-list\"]/div[4]/a")).click();
Thread.sleep(1000);
driver.findElement(By.xpath("//*[@id=\"22\"]")).click();
Thread.sleep(1000);
driver.findElement(By.xpath("//*[@id=\"cart-plus\"]")).click();
Thread.sleep(1000);
driver.findElement(By.xpath("//*[@id=\"table-body\"]/tr/td[3]/button[1]")).click();
Thread.sleep(1000);
try {
        if (driver.switchTo().alert().getText().equals("Currently no item in cart!")) {
      // Handle the expected alert
      driver.switchTo().alert().accept();
  }else{
      // Handle unexpected behavior (no alert or different alert message)
      System.err.println("Unexpected alert content or no alert found.");
  }

}catch (Exception e) {
  // Catch other potential exceptions
  System.err.println("Unexpected error occurred: " + e.getMessage());
  // Log or handle the error as needed
}

driver.findElement(By.xpath("//*[@id=\"category-list\"]/div[4]/a")).click();
Thread.sleep(1000);
driver.findElement(By.xpath("//*[@id=\"16\"]")).click();
Thread.sleep(1000);
driver.findElement(By.xpath("//*[@id=\"17\"]")).click();
```

```
          Thread.sleep(1000);
          driver.findElement(By.xpath("//*[@id=\"20\"]")).click();
          Thread.sleep(1000);
          driver.findElement(By.xpath("//*[@id=\"cart-plus\"]")).click();
          Thread.sleep(1000);
          driver.findElement(By.xpath("//*[@id=\"checkout\"]/button")).click();
          Thread.sleep(1000);
          driver.findElement(By.xpath("//*[@id=\"card-number\"]")).sendKeys(cardnum);
          Thread.sleep(1000);
          driver.findElement(By.xpath("//*[@id=\"expiry-date\"]")).sendKeys(exp);
          Thread.sleep(1000);
          driver.findElement(By.xpath("//*[@id=\"cvv\"]")).sendKeys(cvv);
          Thread.sleep(1000);
          driver.findElement(By.xpath("/html/body/section/div[2]/form/button")).click();
          Thread.sleep(2000);
    }

    @AfterTest
    public void tearDown() {
  driver.quit();
}

    @DataProvider
    public Object[][] Register()
    {
          Object[][] data = new Object[7][2];

          //empty password
          data[0][0] = "";
          data[0][1] = "";

          //already existing user
          data[1][0] = "1234@gmail.com";
```

```
                data[1][1] = "K@1234567#";

                //Password less than 8 digits
                data[2][0] = "kj@gmail.com";
                data[2][1] = "jaink@1";

                //Password more than 15 digits
                data[3][0]="kash@gmail.com";
                data[3][1]="Kashu@12345678910#";


                //Password has no lowercase letter
                data[4][0]="kash@gmail.com";
                data[4][1]="KASHVI@1234";

                //no uppercase
                data[5][0]="kash@gmail.com";
                data[5][1]="kashvi@1234";

                //all correct
                data[6][0] = "12346@gmail.com";
                data[6][1] = "Kash1@8906";

                return data;
        }

        @DataProvider
public Object[][] Login() {
    Object[][] data = new Object[1][2];
        //valid login
    data[0][0] = "12346@gmail.com";
    data[0][1] = "Kash1@8906";
```

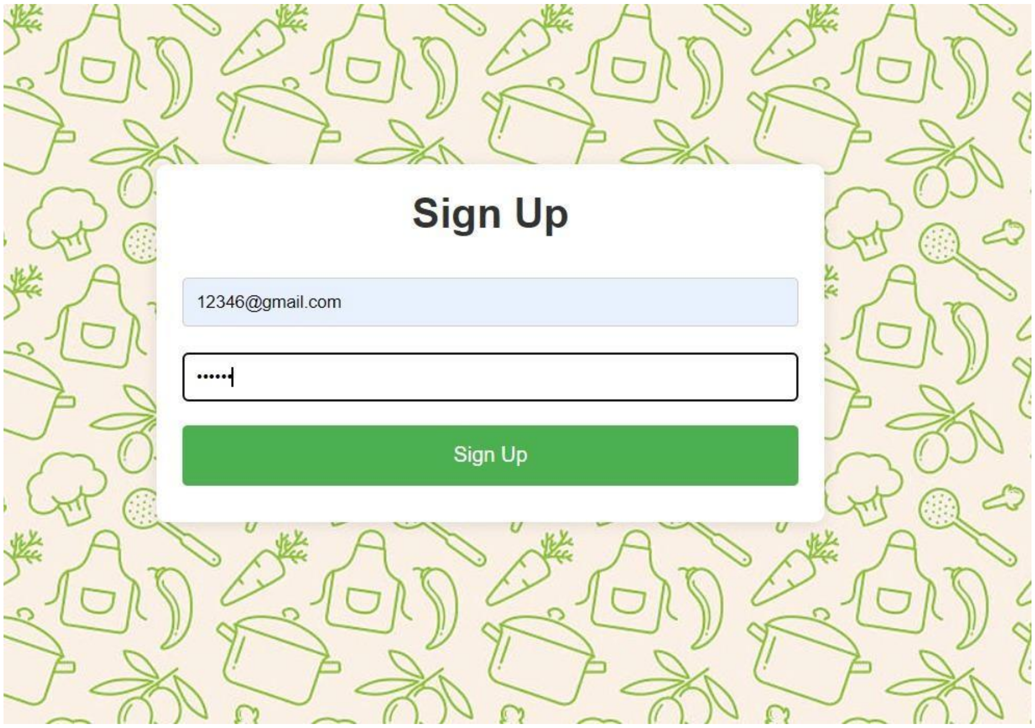
```
        return data;
    }

            @DataProvider
    public Object[][] details() {
        Object[][] data = new Object[1][3];
        data[0][0] = "12345678910";
        data[0][1] = "02/24";
        data[0][2] = "12345";
        return data;
    }
}
```

Sample Screenshots



Welcome to Our Website!

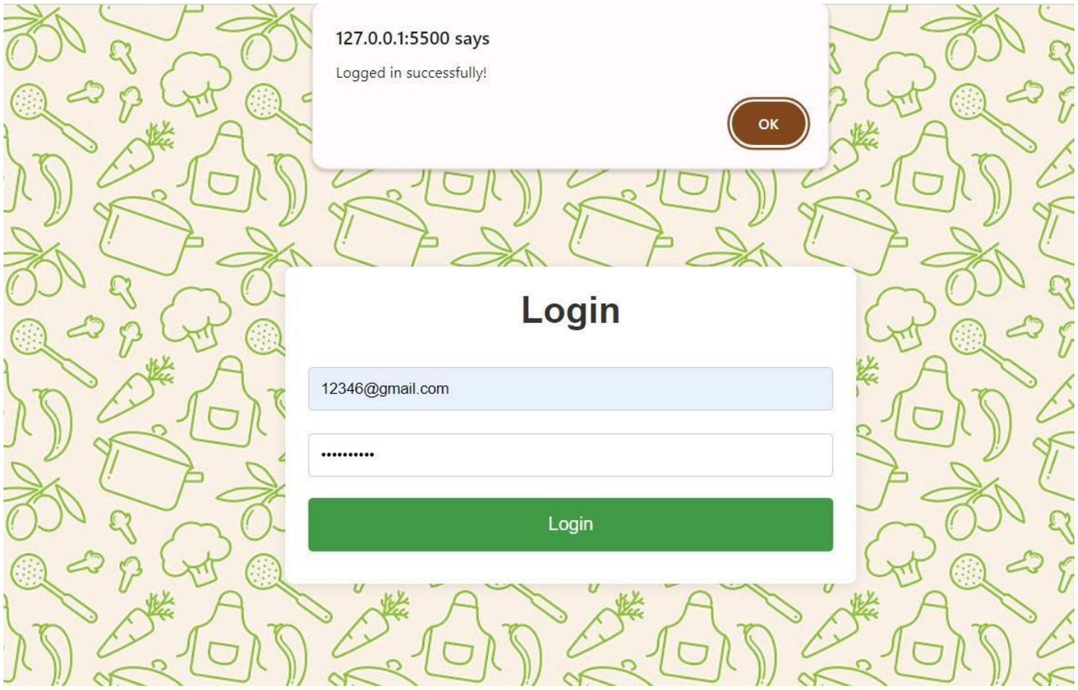
Please sign up or log in to continue.



Sign Up

12346@gmail.com

••••••



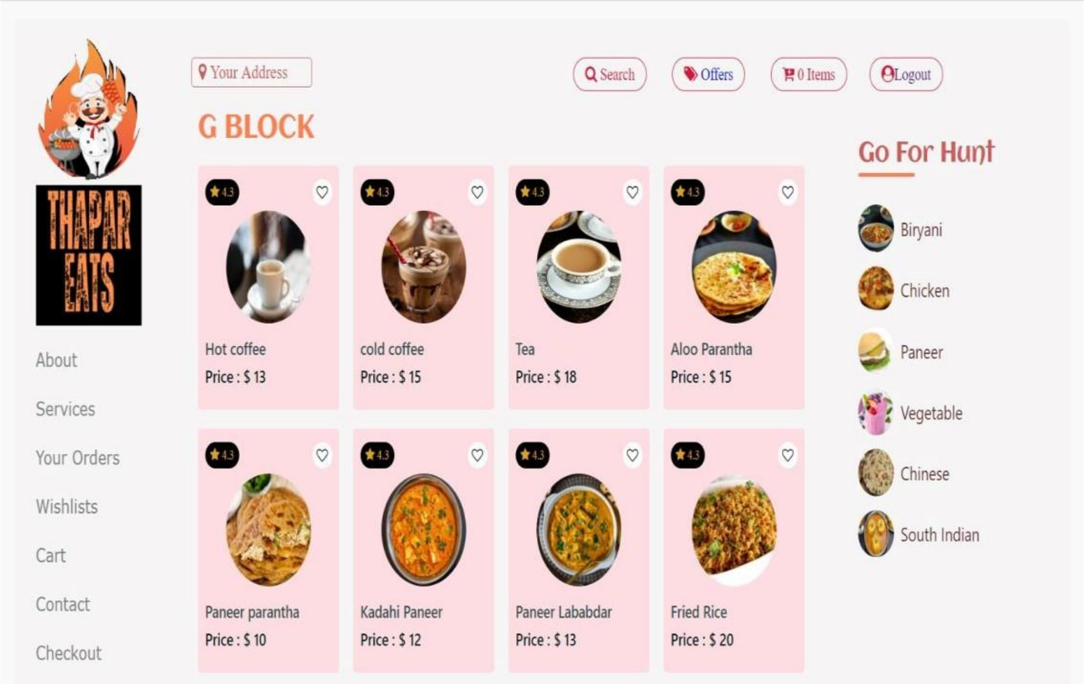
127.0.0.1:5500 says

Logged in successfully!

Login

12346@gmail.com

••••••••••



G BLOCK

4.3 Hot coffee Price : $ 13

4.3 cold coffee Price : $ 15

4.3 Tea Price : $ 18

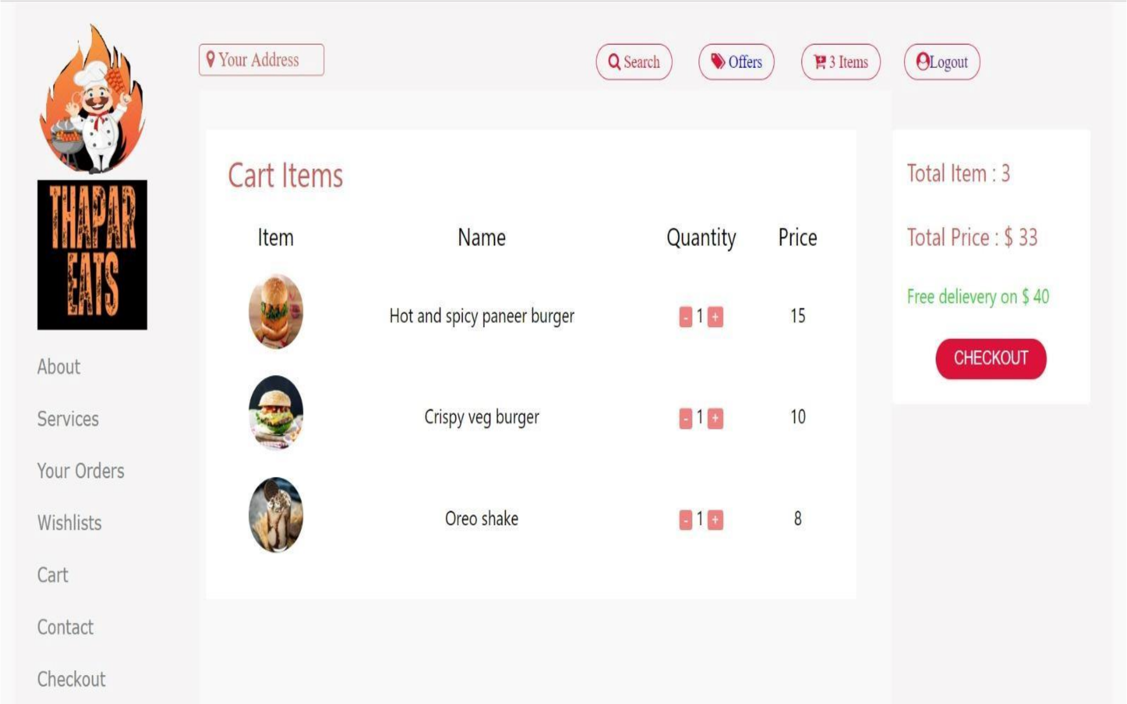
4.3 Aloo Parantha Price : $ 15

4.3 Paneer parantha Price : $ 10

4.3 Kadahi Paneer Price : $ 12

4.3 Paneer Lababdar Price : $ 13

4.3 Fried Rice Price : $ 20



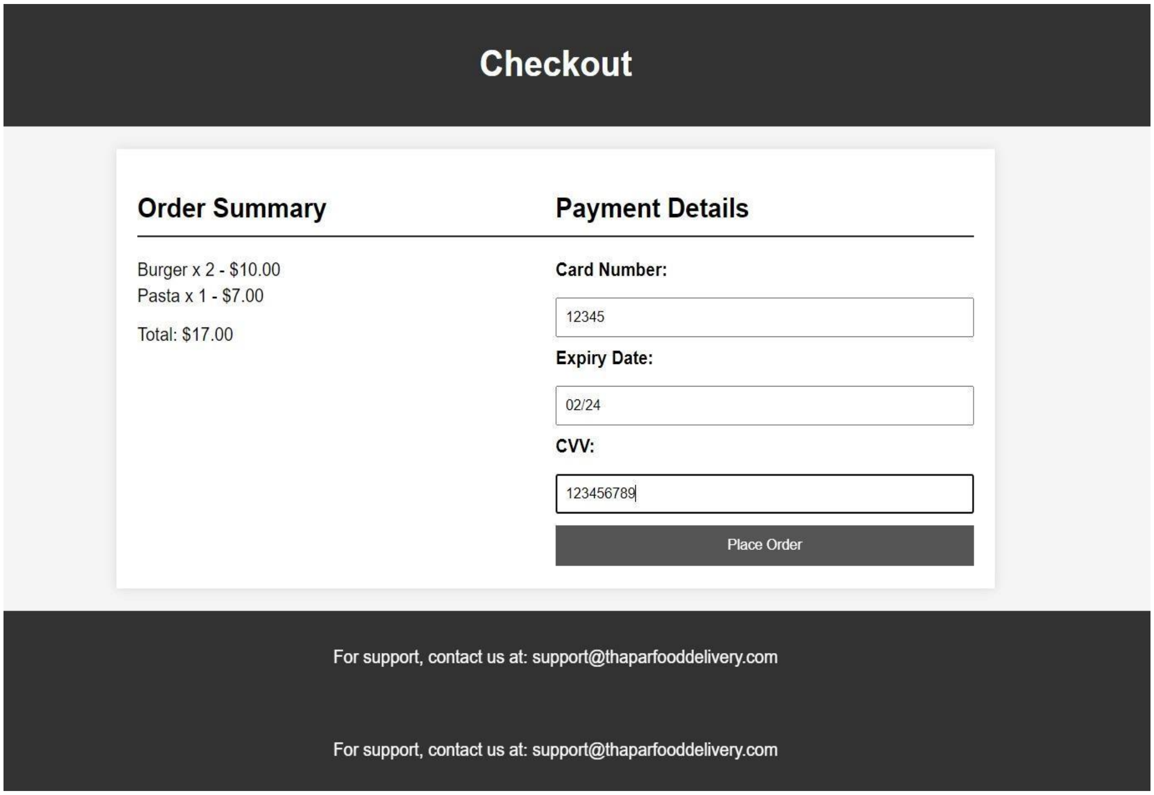
Cart Items

Item	Name	Quantity	Price
	Hot and spicy paneer burger	- 1 +	15
	Crispy veg burger	- 1 +	10
	Oreo shake	- 1 +	8

Total Item : 3

Total Price : $ 33

Free delievery on $ 40



Checkout

Order Summary

Burger x 2 - $10.00
Pasta x 1 - $7.00

Total: $17.00

Payment Details

Card Number: 12345

Expiry Date: 02/24

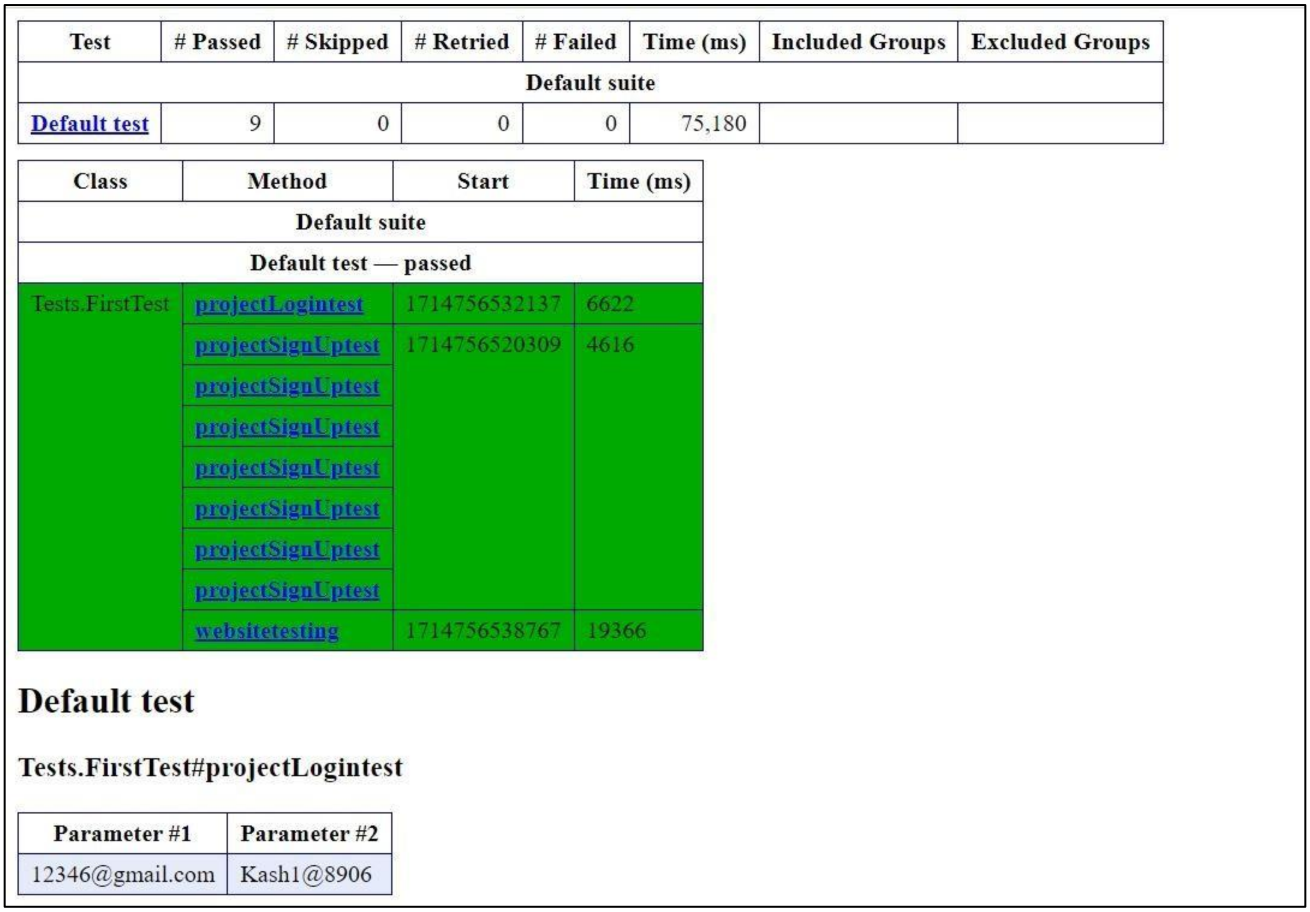
CVV: 123456789

For support, contact us at: support@thaparfooddelivery.com

For support, contact us at: support@thaparfooddelivery.com

OUTPUT REPORTS

Emailable Report



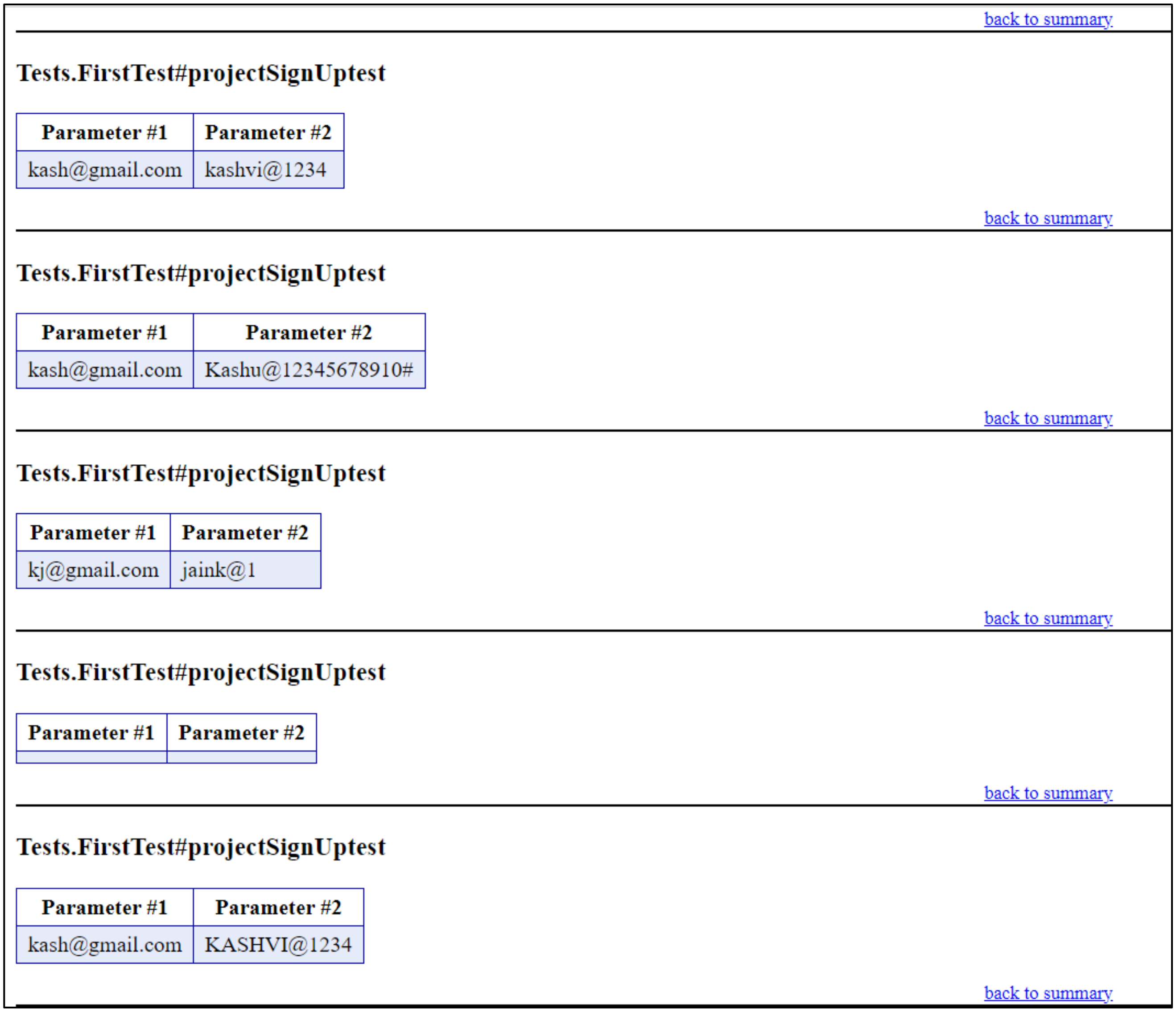
Test	# Passed	# Skipped	# Retried	# Failed	Time (ms)	Included Groups	Excluded Groups
Default suite							
Default test	9	0	0	0	75,180		

Class	Method	Start	Time (ms)
Default suite			
Default test — passed			
Tests.FirstTest	projectLogintest	1714756532137	6622
	projectSignUptest	1714756520309	4616
	projectSignUptest		
	projectSignUptest		
	projectSignUptest		
	projectSignUptest		
	projectSignUptest		
	projectSignUptest		
	websitetesting	1714756538767	19366

Default test

Tests.FirstTest#projectLogintest

Parameter #1	Parameter #2
12346@gmail.com	Kash1@8906



back to summary

Tests.FirstTest#projectSignUptest

Parameter #1	Parameter #2
kash@gmail.com	kashvi@1234

back to summary

Tests.FirstTest#projectSignUptest

Parameter #1	Parameter #2
kash@gmail.com	Kashu@12345678910#

back to summary

Tests.FirstTest#projectSignUptest

Parameter #1	Parameter #2
kj@gmail.com	jaink@1

back to summary

Tests.FirstTest#projectSignUptest

Parameter #1	Parameter #2

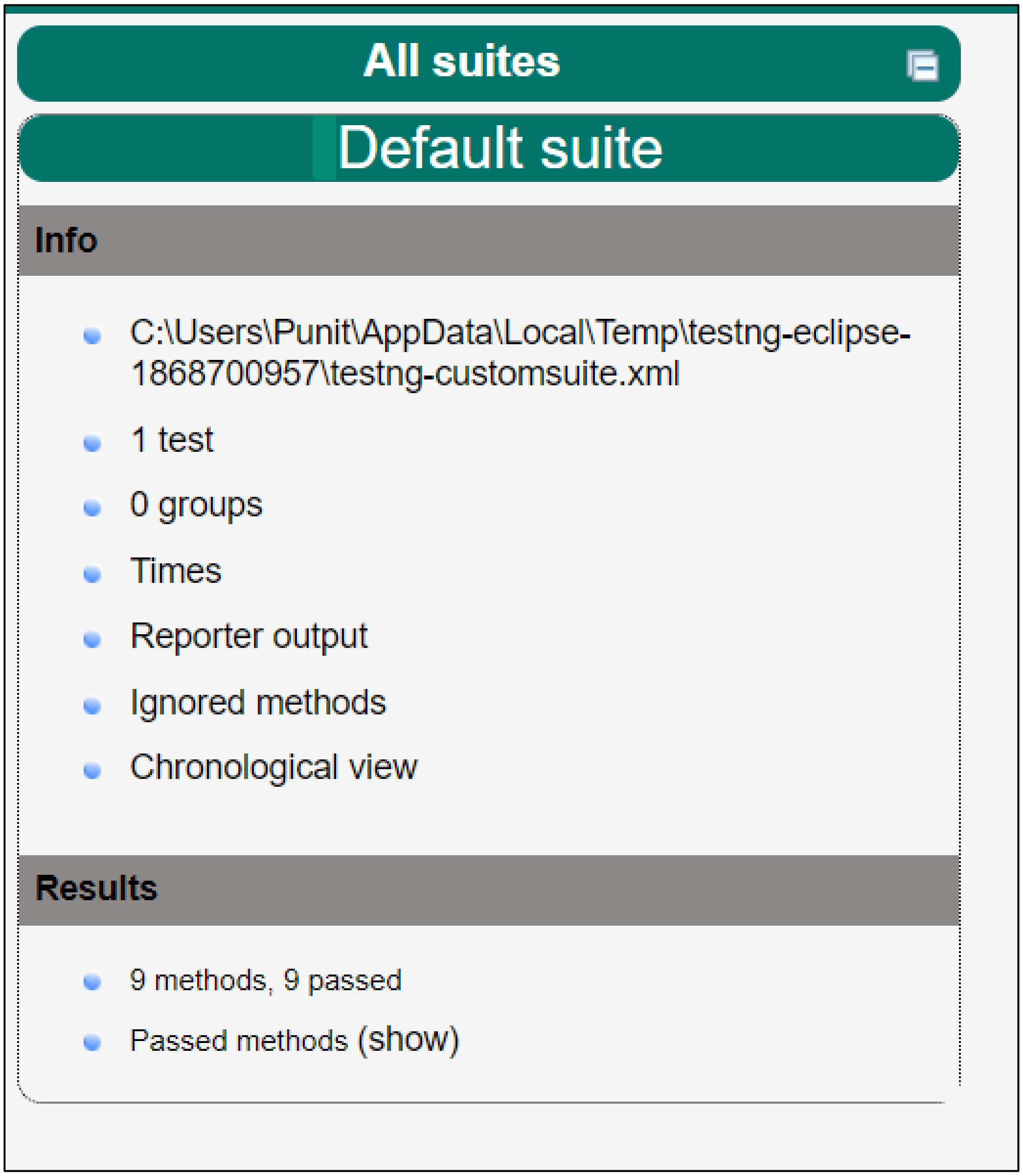
back to summary

Tests.FirstTest#projectSignUptest

Parameter #1	Parameter #2
kash@gmail.com	KASHVI@1234

back to summary

Index Report



Default suite

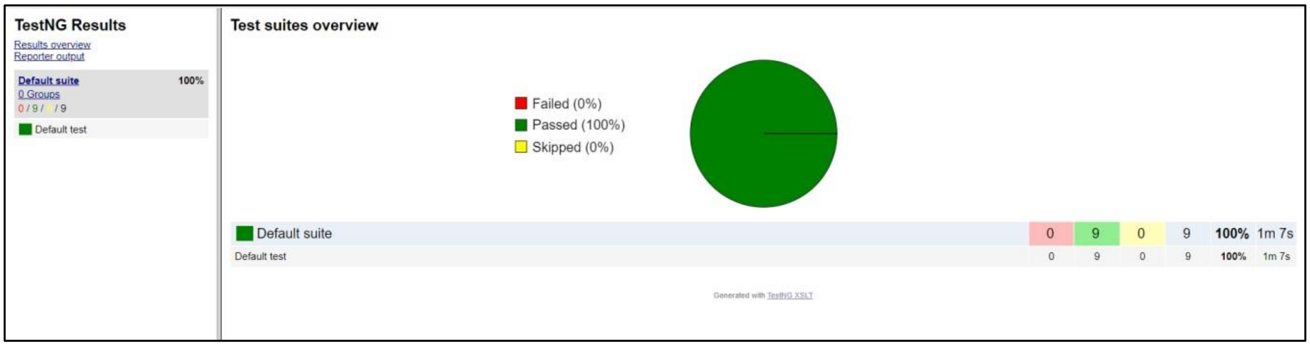
Info

- C:\Users\Punit\AppData\Local\Temp\testng-eclipse-1868700957\testng-customsuite.xml
- 1 test
- 0 groups
- Times
- Reporter output
- Ignored methods
- Chronological view

Results

- 9 methods, 9 passed
- Passed methods (show)

XSLT Report



TestNG Results

Results overview
Reporter output

Default suite 100%
0 Groups
0 / 9 / 0 / 9

🟩 Default test

Test suites overview

🟩 Default suite	0	9	0	9	**100%**	1m 7s
Default test	0	9	0	9	**100%**	1m 7s

Generated with TestNG XSLT

CONCLUSION

In conclusion, our project "Food On Doors" has reached a significant milestone with the successful completion of the testing phase. Through meticulous planning, implementation, and execution, we have ensured the robustness and reliability of our food delivery website.

The utilization of Selenium and TestNG for automated testing has proven to be instrumental in validating the functionality of our platform across different browsers and platforms. By simulating various user interactions and scenarios, we have identified and addressed potential issues, ensuring a seamless user experience.

As we move forward, we remain committed to delivering an exceptional food delivery experience to the Thapar University community. With the successful completion of the testing phase, we are one step closer to achieving our goal of revolutionizing campus dining and enhancing convenience for students and faculty alike.