Semester	T.E. Semester V – Computer Engineering
Subject	Software Engineering
Subject Professor In-charge	Dr. Sachin Bojewar
Assisting Teachers	Prof. Sneha Annappanavar
Laboratory	M313B

Student Name	Deep Salunkhe
Roll Number	21102A0014
TE Division	A

Title: Software Testing

.

Explanation: Testing in software development is a crucial phase that involves the evaluation of a software application to identify and fix defects, errors, and ensure that it meets the specified requirements and quality standards. Here are some key aspects of testing in software development:

- **Error Detection:** Testing helps in detecting errors or bugs in the software. These errors can be related to functionality, performance, security, or other aspects of the software.
- **Verification and Validation:** Testing verifies that the software meets the requirements outlined in the design and specification phase (verification) and validates that it fulfills the user's needs (validation).
- **Quality Assurance:** Testing is a fundamental part of quality assurance. It ensures that the software functions correctly, performs well, and is reliable.
- **Types of Testing:** There are various types of testing, including:
- **Unit Testing:** Testing individual components or functions of the code in isolation.
- **Integration Testing:** Ensuring that different components or modules work together correctly.
- **System Testing:** Evaluating the entire system's functionality.
- **Acceptance Testing:** Confirming that the software meets the user's acceptance criteria.
- **Performance Testing:** Checking how the software performs under different loads and conditions.
- **Security Testing:** Identifying vulnerabilities and weaknesses in the software's security.
- **Regression Testing:** Ensuring that new code changes do not introduce new defects.
- **Manual and Automated Testing:** Testing can be done manually, where testers execute test cases by hand, or through automated testing, where testing tools and scripts are used to run tests automatically.
- **Bug Tracking and Reporting:** When defects are found during testing, they are typically logged in a bug tracking system. This allows developers to prioritize and fix issues.
- **Iterative Process:** Testing is often an iterative process that continues throughout the software development life cycle. It's not just a one-time event but an ongoing activity.
- **Documentation:** Test plans, test cases, and test reports are important documentation that helps in managing and tracking the testing process.
- **User Acceptance Testing (UAT):** In some cases, end-users or stakeholders perform UAT to ensure the software aligns with their expectations.
- **Continuous Integration/Continuous Deployment (CI/CD):** Testing is integrated into the CI/CD pipeline, allowing for automated testing of code changes before they are deployed to production.

Testing is essential to ensure the reliability and quality of software products. It helps identify and address issues early in the development process, reducing the risk of defects reaching the end-users.

.

Implementation:

Test Case 1: User Registration and Authentication
Test Scenario 1.1: User Registration
- Test Case ID: TC01-UR
- Description: Verify that a new user can successfully register an account.
- Precondition: User is on the registration page.
- Test Steps: a. Enter valid registration information (name, email, password). b. Click the "Register" button.
- Expected Result: The user should be registered and redirected to the login page with a success message.

Test Scenario 1.2: User Login
- Test Case ID: TC02-UL
- Description: Verify that a registered user can log in with valid credentials.
- Precondition: User is on the login page and has a registered account.
- Test Steps: a. Enter valid login credentials (email and password). b. Click the "Login" button.
- Expected Result: The user should be logged in and redirected to the homepage.

Test Scenario 1.3: Invalid Login
- Test Case ID: TC03-IL
- Description: Verify that a user with invalid credentials cannot log in.
- Precondition: User is on the login page.
- Test Steps: a. Enter invalid login credentials (incorrect email or password). b. Click the "Login" button.
- Expected Result: The user should see an error message indicating that the login credentials are incorrect.

Test Case 2: Order Placement and Customization
Test Scenario 2.1: Adding Items to Cart
- Test Case ID: TC06-AC
- Description: Verify that users can add food items to their cart.
- Precondition: User is on the menu page.
- Test Steps: a. Click on a food item. b. Select customization options (if available). c. Click the "Add to Cart" button.
- Expected Result: The selected item should be added to the cart.

Test Scenario 2.2: Cart Management
- Test Case ID: TC07-CM
- Description: Verify that users can view and manage items in their cart.
- Precondition: User has items in their cart.
- Test Steps: a. Navigate to the cart page. b. Verify that the items in the cart are displayed. c. Remove an item from the cart.
- Expected Result: The cart should display the selected items, and the removed item should no longer appear.

Test Case 3: Time-Based Order Placement
Test Scenario 3.1: Immediate Pickup
- Test Case ID: TC08-IP
- Description: Verify that users can place an order for immediate pickup.
- Precondition: User has items in their cart.
- Test Steps: a. Proceed to checkout. b. Select "Immediate Pickup." c. Confirm the order.
- Expected Result: The order should be placed for immediate pickup, and the user should receive an order confirmation.

Test Scenario 3.2: Scheduled Pickup
- Test Case ID: TC09-SP
- Description: Verify that users can schedule a specific pickup time for their order.
- Precondition: User has items in their cart.
- Test Steps: a. Proceed to checkout. b. Select "Scheduled Pickup" and choose a future pickup time. c. Confirm the order.

- Expected Result: The order should be scheduled for the chosen pickup time, and the user should receive an order confirmation.

Test Case 4: Real-Time Order Tracking
Test Scenario 4.1: Order Status Updates
- Test Case ID: TC10-OS
- Description: Verify that users receive real-time updates on their order status.
- Precondition: User has placed an order.
- Test Steps: a. Navigate to the order tracking page. b. Check for real-time updates on the order status (e.g., "Order received," "Preparing," "Out for delivery").
- Expected Result: The user should see accurate real-time updates on their order status.

Test Case 5: Payment Integration
Test Scenario 5.1: Payment Process
- Test Case ID: TC11-PP
- Description: Verify that users can successfully complete a payment transaction.
- Precondition: User has items in their cart and is on the checkout page.
- Test Steps: a. Choose a payment method (e.g., credit card). b. Enter valid payment details. c. Confirm the payment.
- Expected Result: The payment should be processed successfully, and the user should receive a payment confirmation.

Various testing tools: -

There are various testing tools available for different types and phases of software testing. The choice of testing tools depends on the specific testing needs and the technology stack of the project. Here's a list of some commonly used testing tools across different categories:

- **Unit Testing Tools:**
  - **JUnit**: A widely used framework for testing Java applications.
  - **NUnit:** A unit testing framework for .NET applications.
  - **PyUnit (unittest):** The built-in unit testing framework for Python.
  - **RSpec:** A behavior-driven development (BDD) framework for Ruby.

- **Integration Testing Tools:**
  - **Postman:** Used for testing RESTful APIs and web services.
  - **SoapUI:** A tool for testing SOAP and REST web services.
  - **Apache JMeter:** Primarily used for performance testing but can also handle integration testing.
  - **TestNG:** A testing framework inspired by JUnit for Java applications.

- **Functional Testing Tools:**
  - **Selenium:** Popular for automated web application testing.
  - **Cypress:** A modern end-to-end testing framework for web applications.
  - **Protractor:** Designed for testing AngularJS and Angular applications.
  - **Appium:** An open-source tool for mobile app testing.

- **Load and Performance Testing Tools:**
  - **Apache JMeter:** Can be used for load testing to simulate a large number of users.
  - **LoadRunner:** A performance testing tool by Micro Focus.
  - **Gatling:** An open-source load testing tool for web applications.

- **Security Testing Tools:**
  - **OWASP ZAP:** An open-source tool for finding vulnerabilities in web applications.
  - **Burp Suite:** A security testing tool for web application security assessment.
  - **Nessus:** A vulnerability scanner for network and web applications.

- **Test Management Tools:**

- **TestRail:** A test case management tool for organizing and tracking test cases.
- **qTest:** A test management and agile project management platform.
- **TestLink:** An open-source test management tool.

➢ **Continuous Integration/Continuous Deployment (CI/CD) Tools:**
- **Jenkins:** Widely used for automating the building, testing, and deployment of applications.
- **Travis CI:** A CI/CD service for GitHub repositories.
- **CircleCI:** A cloud-based CI/CD platform.

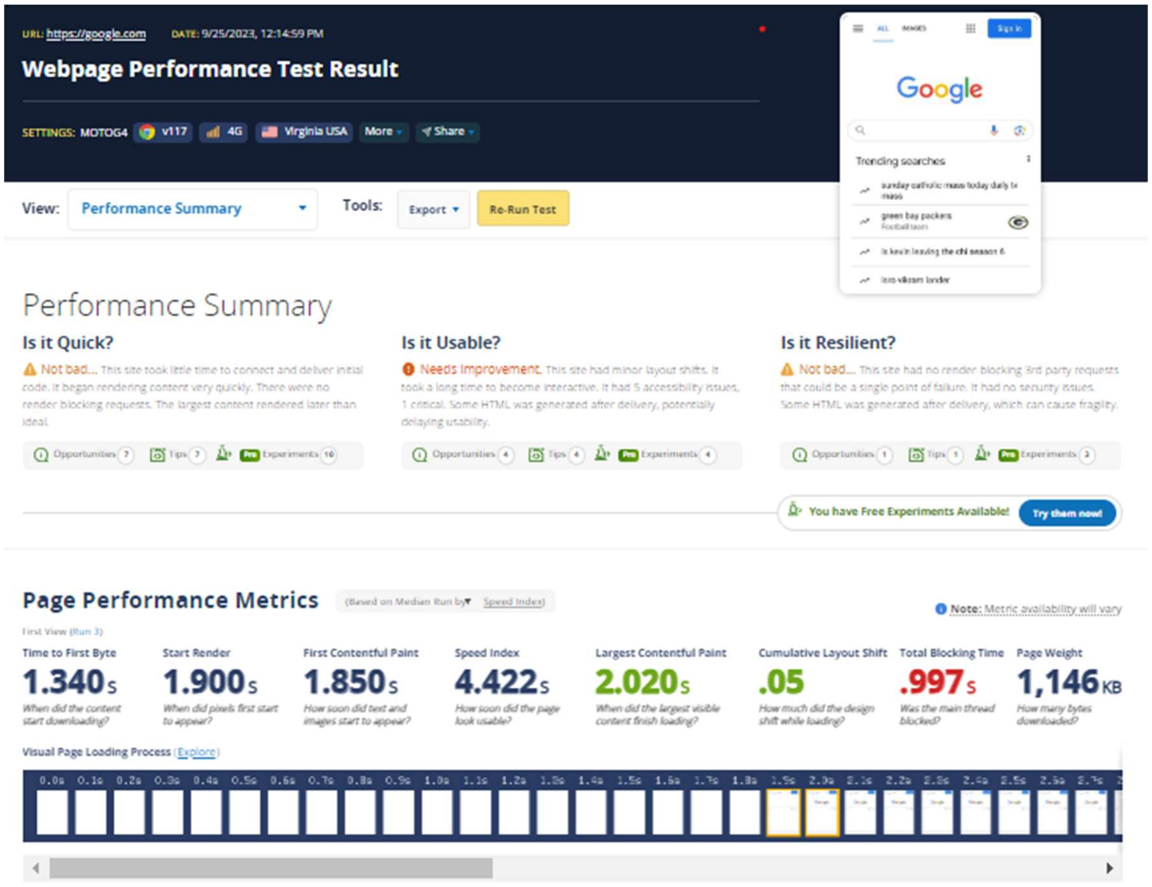
➢ **Code Analysis and Static Testing Tools:**
- **SonarQube:** Used for continuous inspection of code quality.
- **ESLint:** A static code analysis tool for identifying and fixing problems in JavaScript code.
- **Checkstyle:** Enforces a coding standard in Java.

➢ **Test Data Management Tools:**
- **Docker:** Used for creating and managing containers for testing environments.
- **Test Data Generator Tools:** Tools like Faker, DataFactory, and Mockaroo generate test data.

➢ **Cross-Browser Testing Tools:**
- **BrowserStack:** Provides a cloud-based platform for cross-browser testing.
- **Sauce Labs:** Offers a cloud-based testing platform for web and mobile apps.

.

End Result:


URL: https://google.com DATE: 9/25/2023, 12:14:59 PM

Webpage Performance Test Result

SETTINGS: MOTOG4 v117 4G Virginia USA

View: Performance Summary Tools: Export Re-Run Test

Performance Summary

Is it Quick?
Not bad... This site took little time to connect and deliver initial code. It began rendering content very quickly. There were no render blocking requests. The largest content rendered later than ideal.
Opportunities 7 Tips 7 Experiments 10

Is it Usable?
Needs Improvement. This site had minor layout shifts. It took a long time to become interactive. It had 5 accessibility issues, 1 critical. Some HTML was generated after delivery, potentially delaying usability.
Opportunities 4 Tips 4 Experiments 4

Is it Resilient?
Not bad... This site had no render blocking 3rd party requests that could be a single point of failure. It had no security issues. Some HTML was generated after delivery, which can cause fragility.
Opportunities 1 Tips 1 Experiments 2

You have Free Experiments Available! Try them now!

Page Performance Metrics (Based on Median Run by Speed Index)
Note: Metric availability will vary

First View (Run 3)

Time to First Byte 1.340s When did the content start downloading?
Start Render 1.900s When did pixels first start to appear?
First Contentful Paint 1.850s How soon did text and images start to appear?
Speed Index 4.422s How soon did the page look usable?
Largest Contentful Paint 2.020s When did the largest visible content finish loading?
Cumulative Layout Shift .05 How much did the design shift while loading?
Total Blocking Time .997s Was the main thread blocked?
Page Weight 1,146KB How many bytes downloaded?

Visual Page Loading Process (Explore)

Conclusion:

Software testing is essential for identifying and fixing defects in software, ensuring it meets quality standards, and mitigating risks. It involves creating test cases, collaborating with stakeholders, and continuously improving the software. Automation and user-centric testing are common practices, and it concludes with formal sign-off before software release.