Krishna Institute of Engineering and Technology

TEST PLAN FOR

HI DRIVE

Team Members
Anmol Patel 2000290120027
Akshat Singh 2000290120018
Aman Verma 2000290120021

ChangeLog

Version	Change Date	By	Description
version number	Date of Change	Name of person who made changes	Description of the changes made
1.0	01/11/2023	Akshat Singh	Added contact Component
1.1	04/11/2023	Anmol Patel	Implemetation of user States

1 Introduction

This web application project for hiring drivers for one-way trips, long journeys, and permanent positions involves rigorous testing to ensure functionality, usability, performance, and security. This project employs various testing strategies, including functional, usability, security, and compatibility testing, and follows a structured testing process, from requirements analysis to user acceptance testing. Methodologies such as Agile and CI/CD facilitate efficient testing, while data security measures protect user information. In this introduction, we will delve deeper into the project's testing strategies, processes, workflows, and methodologies, as well as its commitment to data security for both customers and drivers.

1.1 Scope

1.1.1 In Scope

For this application on hiring drivers for one-way trips, long journeys, and permanent positions, testing should encompass a wide range of features, including both functional and non-functional requirements. Here are some of the key features and requirements that should be tested:

Functional Requirements:

1. User Registration and Authentication:
   - Ensure users can create accounts and log in securely.
   - Verify password reset and account recovery processes.

2. Driver Profile Management:
   - Test the ability for drivers to create and update their profiles with personal and licensing information.
   - Ensure drivers can specify their availability and job preferences.

3. Job Search and Booking:
   - Test the search functionality for finding drivers based on criteria such as location, vehicle type, and availability.
   - Verify the booking process, including payment and confirmation.

4. Ride Matching:
   - Ensure the application can match customers with suitable drivers based on preferences and availability.

5. Notifications and Communication:
   - Test the messaging system for communication between customers and drivers.
   - Verify the notifications for booking confirmations, updates, and cancellations.

6. Payment Processing:
   - Test payment methods and ensure secure and accurate transaction processing.

7. Admin Panel:
  - Test admin features for managing driver accounts, resolving disputes, and ensuring compliance with platform rules.

Non-Functional Requirements:

1. Performance:
  - Evaluate system responsiveness under various loads, especially during peak hours.
  - Assess the application's ability to handle concurrent users.

2. Security:
  - Conduct security testing to identify vulnerabilities and ensure data protection.
  - Verify the implementation of secure authentication and authorization mechanisms.

3. Usability:
  - Perform usability testing to assess the user-friendliness of both customer and driver interfaces.
  - Ensure that the application provides an intuitive and smooth user experience.

4. Compatibility:
  - Test the application on different browsers, devices, and operating systems to ensure cross-compatibility.

5. Scalability:
  - Assess the application's ability to scale as the user base grows.

6. Data Privacy and Compliance:
  - Ensure compliance with data protection regulations and best practices.
  - Verify the encryption of sensitive user data and regular security audits.

7. Reliability:
  - Test the system's reliability, including error handling and crash recovery.

8. Accessibility:
  - Ensure that the application is accessible to users with disabilities, following relevant accessibility standards.

9. Load Balancing:
  - Test load balancing mechanisms to distribute traffic evenly and optimize performance.

10. Error Handling and Recovery:
  - Verify how the application handles and recovers from errors, including failed transactions and system crashes.

1.1.2 Out of Scope

Some features, functional or non-functional requirements of the software that will NOT be tested are:

1. Server Hardware and Infrastructure: The physical hardware and infrastructure supporting the web application are usually not part of the testing process. These components are typically managed separately by IT and infrastructure teams.

2. Third-party Service Reliability: The reliability of third-party services like payment gateways or mapping services is not tested by the development team but is the responsibility of the service providers. However, integration with these services should be tested for compatibility.

3. Scalability Testing Beyond Project Scope: Although scalability testing is important, extensive scalability testing to handle large-scale operations, such as in case of massive user growth, might not be conducted as part of the initial testing phase. It is usually addressed as the system evolves.

4. Market Research and Customer Feedback: Gathering market research data and user feedback about the application's usability and features are not part of the testing process but are essential for ongoing improvements.

5. Customer and Driver Behavior Analysis: Analyzing the behavior and preferences of customers and drivers may not be a direct testing activity, but it informs product improvements and marketing strategies.

6. Marketing and User Acquisition: Marketing and user acquisition strategies are not part of the testing process, although the application's performance in attracting and retaining users can be monitored.Quality Objective

1.2 Roles and Responsibilities

Detail description of the Roles and responsibilities of different team members like

- QA Analyst: Akshat Singh
- Test Manager: Anmol Patel
- Configuration Manager: Aman Verma
- Developers: Anmol Patel, Akshat Singh, Aman Verma
- Installation Team: Anmol Patel, Akshat Singh, Aman Verma

2 Test Methodology

2.1 Overview

We are using Agile model for this project to fulfill the requirements, context, and goals. This model is selected due to various factors:

1. Project Complexity: Choose Agile for complex, dynamic projects, and traditional methods for simpler ones.

2. Frequent Changes: Agile is ideal for rapid updates and adaptability. And in this project we need to do changes frequently.

3. Customer Involvement: Agile is favored for regular customer feedback. So it will help to resolve the problems easily.

4. Resource Availability: Consider the availability of skilled resources and their suitability for the chosen methodology.

5. Risk Tolerance: Integrate risk-based testing for high-risk elements.

6. Regulatory Compliance: Use Waterfall or V-Model for projects with strict regulatory requirements.

7. Testing Objectives: Select methodologies based on specific testing goals (e.g., security, performance).

8. Team Expertise: Teams with experience in specific methodologies tend to choose them.

2.2 Test Levels

Test Levels define the Types of Testing to be executed on the Application Under Test (AUT). The Testing Levels primarily depends on the scope of the project, time and budget constraints.

1. Unit testing
2. Integration testing
3. System testing
4. Acceptance testing

2.2.1 Unit Testing

Unit testing is the first level of testing. This testing is the most basic type of testing done by the developers before handing the software/product to the testing team.

Unit Testing Definition: Unit testing is a type of software testing in which individual units or components of the software are tested.
Primary Objective: The main objective of unit testing is to isolate each component of the software and then perform tests to illustrate that every individual component is accurately meeting the requirements and delivering the expected output.

2.2.2 Integration testing

Integration testing is the second level of testing. The testers, rather than the developers, mainly conduct this testing. This testing can be performed manually or using integration testing tools, such as Selenium.

Integration Testing Definition: Integration testing is a type of software testing in which individual software components (modules) are logically integrated (combined) and tested as a group.
Primary Objective: The main objective of integration testing is to verify whether individual modules, when combined (integrated), work correctly or not as a group.

2.2.3 System testing

System testing is the third level of testing. This level of testing assists you in identifying bugs and challenges while ensuring that the software will meet all specific requirements. A specialized testing team is usually in charge of this type of testing.

System Testing Definition: System testing is software testing in which all components are tested together (as a whole) to ensure that the final product meets the specified requirements.
Primary Objective: The main objective of this level of testing is to make sure that the software/product meets specified requirements and runs as smoothly as possible in its operating environment.

2.2.4 Acceptance testing

Acceptance testing is the last and final level of testing. This level of testing is broad in scope, ranging from simply finding spelling and cosmetic errors to discovering bugs that might produce a significant error in the software.

Acceptance Testing Definition: Acceptance testing is a type of software testing that determines whether or not the software should be released to the public.
Primary Objective: The main objective of acceptance testing is to evaluate whether the software complies with the end-user requirements and whether it is ready for deployment.

2.3 Suspension Criteria and Resumption Requirements

Suspension criteria define the criteria to be used to suspend all or part of the testing procedure while Resumption criteria determine when testing can resume after it has been suspended.

Suspension Criteria:

1. Critical Bugs: Pause testing for severe defects impacting core functionality or security.

2. Showstopper Issues: Suspend testing due to obstacles hindering test execution.

3. Resource Constraints: Stop testing if crucial testing resources become unavailable.

4. Legal/Compliance Concerns: Suspend testing for legal or compliance issues until resolved.

Resumption Requirements:

1. Bug Fix Verification: Verify critical bug fixes before resuming testing.

2. Environment Stability: Ensure a stable testing environment before resuming.

3. Resource Availability: Confirm the availability of necessary testing resources.

4. Test Data Validation: Validate the correctness and security of test data.

5. Test Plan Update: Update the test plan and cases if needed.

2.4 Test Completeness

We have tested all the criteria to check test Completeness and accuracy of the project. We have done

- 100% test is covered
- All Manual & Automated Test cases are executed
- All open bugs are fixed

3 Test Deliverable

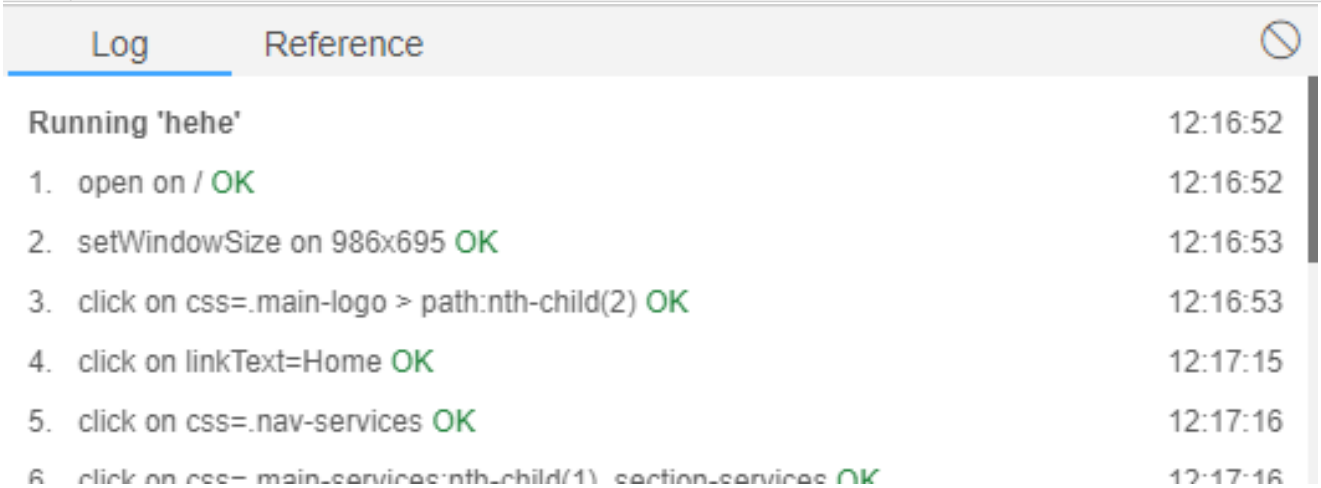
Here are the sample deliverable

Manual Testing Decision Table

	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6	Rule 7	Rule 8	Rule 9
Conditions	TC01	TC02	TC03	TC04	TC05	TC06	TC07	TC08	TC09
UserID	Blank	Valid	Invalid	Blank	Blank	Invalid	Valid	Invalid	Valid
Password	Blank	Blank	Blank	Valid	Invalid	Valid	Invalid	Invalid	Valid
Actions									
Login Successfully									Execute
Output	Execute	Execute	Execute	Execute	Execute	Execute	Execute	Execute	

Automated Testing (Selenium)

	Command	Target	Value
1	✓ *open*	/	
2	✓ *set window size*	986x695	
3	✓ *click*	css=.main-logo > path:nth-child(2)	
4	✓ *click*	linkText=Home	
5	✓ *click*	css=.nav-services	
6	✓ *click*	css=.main-services:nth-child(1) .section-services	
7	✓ *click*	css=.closeBtn > path:nth-child(2)	



7	*click*	css=.main-services:nth-child(3) .section-services	
8	*click*	css=.overlay	
9	*click*	css=.overlay	
10	*click*	css=.first_fif > .size:nth-child(4)	
11	*click*	linkText=Delhi NCR	
12	*mouse over*	linkText=Delhi NCR	
13	*mouse out*	css=.first_fif > .size:nth-child(5) > a	
14	*click*	linkText=Ezi Drive	
15	*mouse over*	linkText=Ezi Drive	
16	*mouse out*	css=.first_six_sec > a	

4 Resource & Environment Needs

4.1 Testing Tools

Testing tools used for project are:

1. Testing: Selenium Tool

4.2 Test Environment

Here's an outline of the key components and considerations for the test environment:

Software and Tools:

5. Operating Systems: Use the same operating system versions as in production, whether it's Linux, Windows, or another OS.

6. Web Server Software: Install the web server software (e.g., Apache, Nginx) used in production.

7. Database Management System: We have used MongoDB in this project.

8. Application Codebase: Deploy the application codebase, ensuring it matches the version in production.

9. Test Automation Tools: Implement test automation tools such as Selenium, for automated testing.

5 Terms/Acronyms

Make a mention of any terms or acronyms used in the project

TERM/ACRONYM	DEFINITION
API	Application Program Interface
AUT	Application Under Test