Department of Computer

Engineering Academic Term: First

Term 2023-24

Class: T.E /Computer Sem – V / Software Engineering

Practical No:	**8**
Title:	**Design test cases for performing black box testing**
Date of Performance:	14 - 09 - 23
Roll No:	9595
Team Members:	Atharva Dalvi

Rubrics for Evaluation:

Sr. No	Performance Indicator	Excellent	Good	Below Average	Total Score
1	On time Completion & Submission (01)	01 (On Time)	NA	00 (Not on Time)	
2	Theory Understanding(02)	02(Correct)	NA	01 (Tried)	
3	Content Quality (03)	03(All used)	02 (Partial)	01 (rarely followed)	
4	Post Lab Questions (04)	04(done well)	3 (Partially Correct)	2(submitted)	

Signature of the Teacher:

Lab Experiment 08

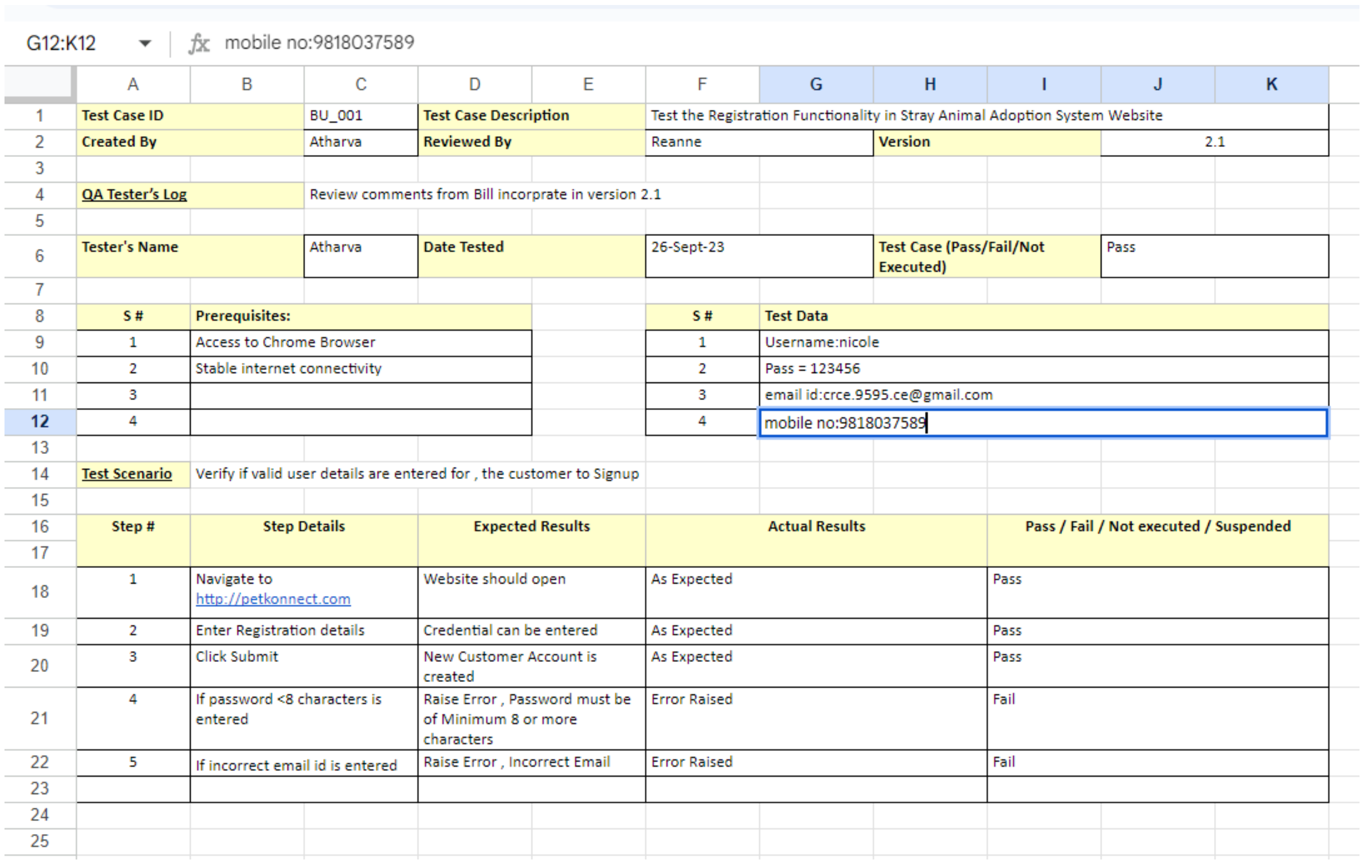
Experiment Name: Designing Test Cases for Performing Black Box Testing in Software Engineering

Objective:
The objective of this lab experiment is to introduce students to the concept of Black Box Testing, a testing technique that focuses on the functional aspects of a software system without examining its internal code. Students will gain practical experience in designing test cases for Black Box Testing to ensure the software meets specified requirements and functions correctly.
Introduction: Black Box Testing is a critical software testing approach that verifies the functionality of a system from an external perspective, without knowledge of its internal structure. It is based on the software's specifications and requirements, making it an essential part of software quality assurance.

Lab Experiment Overview:
1. Introduction to Black Box Testing: The lab session begins with an introduction to Black Box Testing, explaining its purpose, advantages, and the types of tests performed, such as equivalence partitioning, boundary value analysis, decision table testing, and state transition testing.
2. Defining the Sample Project: Students are provided with a sample software project along with its functional requirements, use cases, and specifications.
3. Identifying Test Scenarios: Students analyze the sample project and identify test scenarios based on its requirements and use cases. They determine the input values, expected outputs, and test conditions for each scenario.
4. Equivalence Partitioning: Students apply Equivalence Partitioning to divide the input values into groups that are likely to produce similar results. They design test cases based on each equivalence class.
5. Boundary Value Analysis: Students perform Boundary Value Analysis to determine test cases that focus on the boundaries of input ranges. They identify test cases near the minimum and maximum values of each equivalence class.
6. Decision Table Testing: Students use Decision Table Testing to handle complex logical conditions in the software's requirements. They construct decision tables and derive test cases from different combinations of conditions.
7. State Transition Testing: If applicable, students apply State Transition Testing to validate the software's behavior as it moves through various states. They design test cases to cover state transitions.
8. Test Case Documentation: Students document the designed test cases, including the test scenario, input values, expected outputs, and any preconditions or postconditions.
9. Test Execution: In a simulated test environment, students execute the designed test cases and record the results.
10. Conclusion and Reflection: Students discuss the importance of Black Box Testing in software quality assurance and reflect on their experience in designing test cases for Black Box Testing.

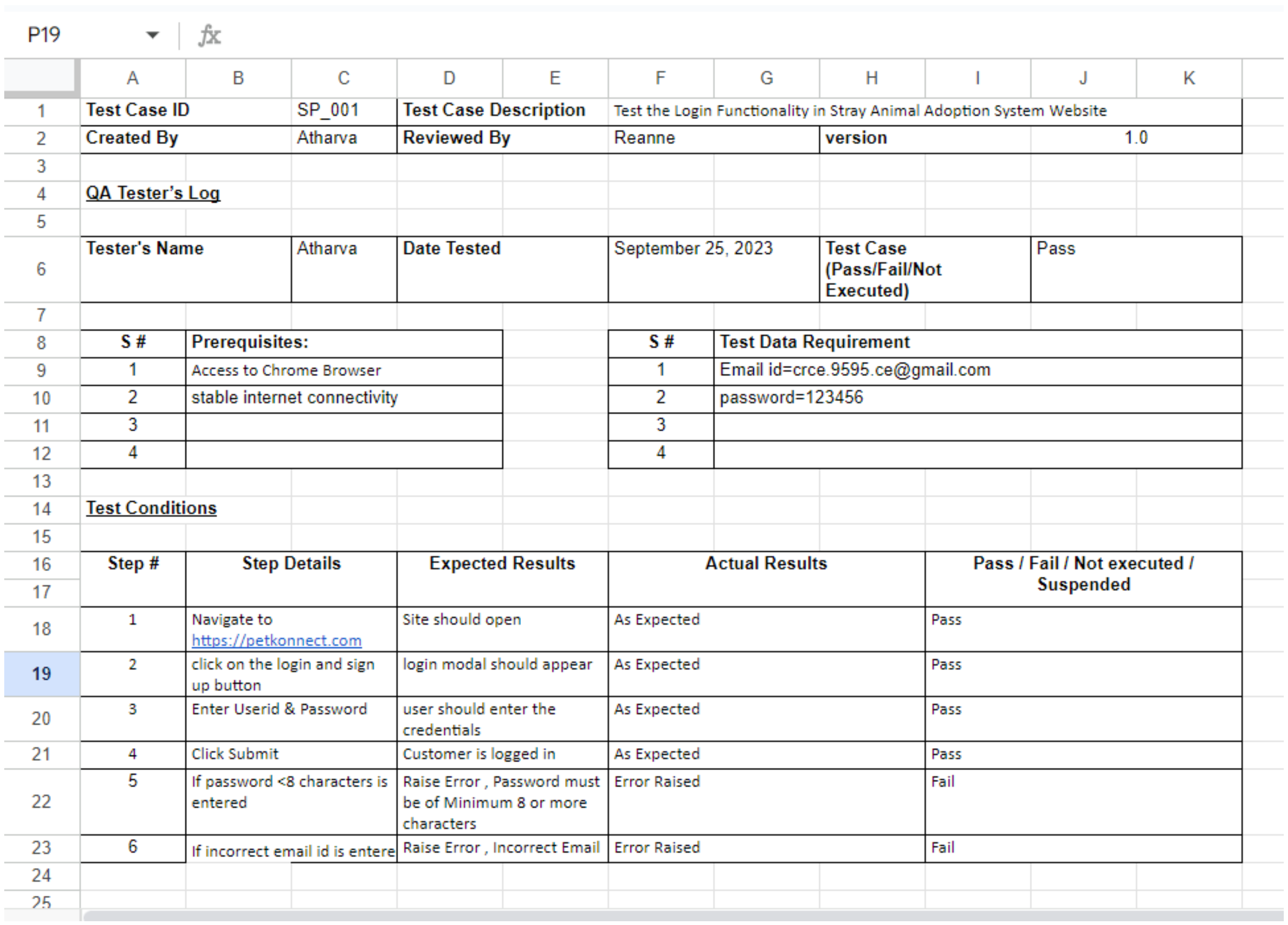
Learning Outcomes:
By the end of this lab experiment, students are expected to:
Understand the concept and significance of Black Box Testing in software testing.
Gain practical experience in designing test cases for Black Box Testing based on functional requirements.
Learn to apply techniques such as Equivalence Partitioning, Boundary Value Analysis, Decision Table Testing, and State Transition Testing in test case design.
Develop documentation skills for recording and organizing test cases effectively.
Appreciate the role of Black Box Testing in identifying defects and ensuring software functionality.
Pre-Lab Preparations:
Before the lab session, students should familiarize themselves with Black
Box Testing concepts, Equivalence Partitioning, Boundary Value Analysis, Decision Table Testing, and State Transition Testing techniques.
Materials and Resources:
Project brief and details for the sample software project
Whiteboard or projector for explaining Black Box Testing techniques
Test case templates for documentation
Conclusion:
The lab experiment on designing test cases for Black Box Testing provides students
with essential skills in verifying software functionality from an external perspective. By applying various Black Box Testing techniques, students ensure comprehensive test coverage and identify potential defects in the software. The experience in designing and executing test cases enhances their ability to validate software behavior and fulfill functional requirements. The lab experiment encourages students to incorporate Black Box Testing into their software testing strategies, promoting
robust and high-quality software development. Emphasizing test case design in Black Box Testing

empowers students to contribute to software quality assurance and deliver reliable and customer-oriented software solutions.



G12:K12 fx mobile no:9818037589

	A	B	C	D	E	F	G	H	I	J	K
1	**Test Case ID**		BU_001	**Test Case Description**		Test the Registration Functionality in Stray Animal Adoption System Website					
2	**Created By**		Atharva	**Reviewed By**		Reanne		**Version**		2.1	
3											
4	**QA Tester's Log**		Review comments from Bill incorprate in version 2.1								
5											
6	**Tester's Name**		Atharva	**Date Tested**		26-Sept-23		**Test Case (Pass/Fail/Not Executed)**		Pass	
7											
8	**S #**	**Prerequisites:**				**S #**	**Test Data**				
9	1	Access to Chrome Browser				1	Username:nicole				
10	2	Stable internet connectivity				2	Pass = 123456				
11	3					3	email id:crce.9595.ce@gmail.com				
12	4					4	mobile no:9818037589				
13											
14	**Test Scenario**	Verify if valid user details are entered for , the customer to Signup									
15											
16	**Step #**	**Step Details**		**Expected Results**		**Actual Results**			**Pass / Fail / Not executed / Suspended**		
17											
18	1	Navigate to http://petkonnect.com		Website should open		As Expected			Pass		
19	2	Enter Registration details		Credential can be entered		As Expected			Pass		
20	3	Click Submit		New Customer Account is created		As Expected			Pass		
21	4	If password <8 characters is entered		Raise Error , Password must be of Minimum 8 or more characters		Error Raised			Fail		
22	5	If incorrect email id is entered		Raise Error , Incorrect Email		Error Raised			Fail		
23											
24											
25											



P19 fx

Test Case ID	SP_001	**Test Case Description**	Test the Login Functionality in Stray Animal Adoption System Website		
Created By	Atharva	**Reviewed By**	Reanne	**version**	1.0

QA Tester's Log

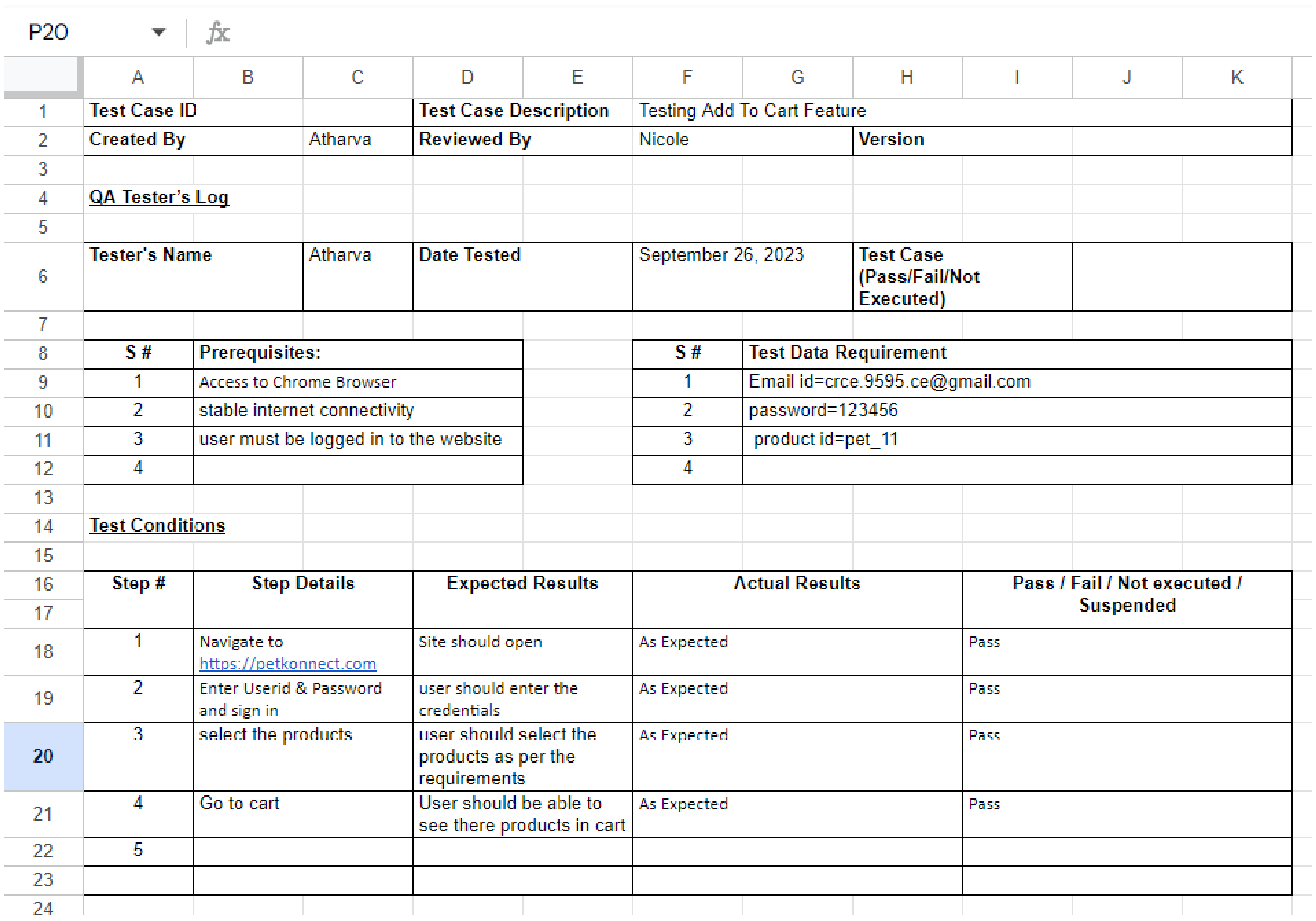
Tester's Name	Atharva	**Date Tested**	September 25, 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Access to Chrome Browser
2	stable internet connectivity
3	
4	

S #	Test Data Requirement
1	Email id=crce.9595.ce@gmail.com
2	password=123456
3	
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to https://petkonnect.com	Site should open	As Expected	Pass
2	click on the login and sign up button	login modal should appear	As Expected	Pass
3	Enter Userid & Password	user should enter the credentials	As Expected	Pass
4	Click Submit	Customer is logged in	As Expected	Pass
5	If password <8 characters is entered	Raise Error , Password must be of Minimum 8 or more characters	Error Raised	Fail
6	If incorrect email id is entere	Raise Error , Incorrect Email	Error Raised	Fail



P20

Test Case ID		**Test Case Description**	Testing Add To Cart Feature		
Created By	Atharva	**Reviewed By**	Nicole	**Version**	

QA Tester's Log

Tester's Name	Atharva	**Date Tested**	September 26, 2023	**Test Case (Pass/Fail/Not Executed)**	

S #	Prerequisites:
1	Access to Chrome Browser
2	stable internet connectivity
3	user must be logged in to the website
4	

S #	Test Data Requirement
1	Email id=crce.9595.ce@gmail.com
2	password=123456
3	product id=pet_11
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to https://petkonnect.com	Site should open	As Expected	Pass
2	Enter Userid & Password and sign in	user should enter the credentials	As Expected	Pass
3	select the products	user should select the products as per the requirements	As Expected	Pass
4	Go to cart	User should be able to see there products in cart	As Expected	Pass
5				

a) Create a set of black box test cases based on a given set of functional requirements, ensuring adequate coverage of different scenarios and boundary conditions.

Understand the Requirements: Begin by thoroughly understanding the functional requirements of the software. This involves reviewing the software specification documents and any other relevant information.

Identify Test Scenarios: Identify different scenarios based on the requirements. These scenarios should cover a range of inputs, conditions, and expected outputs. Consider positive and negative scenarios.

Boundary Conditions: Pay special attention to boundary conditions and edge cases. These are critical for ensuring comprehensive testing.

Equivalence Partitioning: Group input data into equivalence classes, which represent sets of inputs that should produce the same behavior. Test at least one case from each equivalence class.

Decision Tables: Create decision tables to cover all possible combinations of conditions and actions, especially in cases of complex conditional logic.

Use Case Testing: If applicable, create test cases based on use cases that represent real-world interactions with the software.

Error Handling: Include test cases that focus on error-handling scenarios, such as input validation and handling unexpected situations.

State Transitions: For systems with states, create test cases that cover state transitions and ensure the system behaves correctly at each state.

Interface Testing: Test the interfaces where the software interacts with external systems or components. Ensure data is correctly passed and received.

Performance and Stress Testing: Include test cases to evaluate the software's performance under load, stress, or other conditions that may affect its functionality.

b) Evaluate the effectiveness of black box testing in uncovering defects and validating the software's functionality, comparing it with other testing techniques

Black box testing is effective in many ways, but it's essential to compare it to other testing techniques to understand its strengths and weaknesses:
Defect Uncovering: Black box testing is effective in uncovering defects related to incorrect functionality, usability, security, and compatibility issues.

User Perspective: It focuses on testing the software from a user's perspective, ensuring it meets user expectations.

Independence: Testers do not need to know the internal code, making it independent of the programming language or implementation.

Validation: It validates that the software functions as specified in the requirements.
However, black box testing has limitations:

Limited Code Coverage: It may not fully exercise all paths within the code, potentially missing certain defects.

Limited Control: Testers have limited control over the internal logic and data, making it challenging to target specific code segments.

Integration Challenges: It may struggle to uncover integration issues between different components. Comparing it with white box testing (which focuses on internal code structure) and gray box testing (combining aspects of both), black box testing is less effective at code-level defects but highly effective for functional and user experience testing.

c) Assess the challenges and limitations of black box testing in ensuring complete test coverage and discuss strategies to overcome them.

Challenges in black box testing include:

Incomplete Coverage: Ensuring comprehensive test coverage can be challenging. To overcome this, prioritize critical and high-risk areas and use techniques like pairwise testing to reduce the number of test cases needed.

Lack of Internal Knowledge: Testers may not have access to the code or detailed knowledge of the software's internal structure. Encourage collaboration and knowledge sharing between development and testing teams to bridge this gap.

Data Variability: Handling a wide range of input data can be complex. Use data generation tools or data-driven testing to address this challenge.

Dynamic Behavior: For software with dynamic behavior, it's challenging to anticipate all possible states. Use state transition testing and exploratory testing to address dynamic scenarios.

Regression Testing: Maintain a strong regression test suite to ensure that changes in the software don't introduce new defects.