Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

Academic Year : 2024-25

Experiment No. 10
Develop test cases for White Box Testing for a given code
Name: Krisha Chikka
Std/Div: TE/1 Roll no. : 30
Date of Performance: 30/09/2024
Date of Submission: 07/10/2024

Aim: To develop test cases for White Box Testing for a given code.

Objective: To analyse the code of Currency Detector and develop test cases for White Box Testing.

Theory:

White Box Testing is software testing technique in which internal structure, design and coding of software are tested to verify flow of input-output and to improve design, usability and security. In white box testing, code is visible to testers so it is also called Clear box testing, Open box testing, Transparent box testing, Code-based testing and Glass box testing.
It is one of two parts of the Box Testing approach to software testing. Its counterpart, Blackbox testing, involves testing from an external or end-user type perspective. On the other hand, White box testing in software engineering is based on the inner workings of an application and revolves around internal testing.

The term "Whitebox" was used because of the see-through box concept. The clear box or Whitebox name symbolizes the ability to see through the software's outer shell (or "box") into its inner workings. Likewise, the "black box" in "Black Box Testing" symbolizes not being able to see the inner workings of the software so that only the end-user experience can be tested.

Steps:

STEP 1) UNDERSTAND THE SOURCE CODE

The first thing a tester will often do is learn and understand the source code of the application. Since white box testing involves the testing of the inner workings of an application, the tester must be very knowledgeable in the programming languages used in the applications they are testing. Also, the testing person must be highly aware of secure coding practices. Security is often one of the primary objectives of testing software. The

tester should be able to find security issues and prevent attacks from hackers and naive users who might inject malicious code into the application either knowingly or unknowingly.

Step 2) CREATE TEST CASES AND EXECUTE

The second basic step to white box testing involves testing the application's source code for proper flow and structure. One way is by writing more code to test the application's source code. The tester will develop little tests for each process or series of processes in the application. This method requires that the tester must have intimate knowledge of the code and is often done by the developer. Other methods include Manual Testing, trial, and error testing and the use of testing tools as we will explain further on in this article.

Types of White Box Testing

White box testing encompasses several testing types used to evaluate the usability of an application, block of code or specific software package. There are listed below —

- **Unit Testing:** It is often the first type of testing done on an application. Unit Testing is performed on each unit or block of code as it is developed. Unit Testing is essentially done by the programmer. As a software developer, you develop a few lines of code, a single function or an object and test it to make sure it works before continuing Unit Testing helps identify a majority of bugs, early in the software development lifecycle. Bugs identified in this stage are cheaper and easy to fix.
- **Testing for Memory Leaks**: Memory leaks are leading causes of slower running applications. A QA specialist who is experienced at detecting memory leaks is essential in cases where you have a slow running software application.

Advantages of White Box Testing

- Code optimization by finding hidden errors.
- White box tests cases can be easily automated.
- Testing is more thorough as all code paths are usually covered.
- Testing can start early in SDLC even if GUI is not available.

Disadvantages of White Box Testing

- White box testing can be quite complex and expensive.
- Developers who usually execute white box test cases detest it. The white box testing by developers is not detailed can lead to production errors.
- White box testing requires professional resources, with a detailed understanding of programming and implementation.

- White-box testing is time-consuming, bigger programming applications take the time to test fully.

Solution:

Module Name	User Interface Module
Test Case ID	UI_01
Tester Name	Krisha Chikka
Test Case Description	Test the voice command interaction for visually impaired users.
Prerequisites	The system should be installed, and voice command features should be enabled.
Environmental Information	OS: Android/iOS, System: Mobile device with microphone.

Test Scenario:	The user should be able to give voice commands to interact with the system.

Test Case ID	Test Steps	Test Inputs	Expected Results	Actual Results	Status	Comments
UI_01	1) Open the App	Voice Command 'Open App'	The app recognizes the command and proceeds to open the App.	Recognized and moved to the next step.	PASS	voice recognition system accurately detects the command.
	2) Access Camera	Voice Command "Open Camera."	The app recognizes the command and proceeds to open the Camera.	The camera opens without delay.	PASS	No issues Found.
	3) Capture Image	Voice Command "Take Photo."	The app recognizes the command and proceeds to take photo.	The camera captures the image and confirms the action with audio feedback.	PASS	No issues Found.
	4) Exit App	Voice Command "Exit App."	The app recognizes the command and proceeds to exit.	The app closes completely.	PASS	No issues Found.
	5) Help Command	Voice Command "Help."	The app recognizes the command and provides the list of commands.	The app provides a list of available voice commands.	PASS	Ensure that the help information is comprehensive and easy to understand.

Module Name	Image Capture Module
Test Case ID	IC_01
Tester Name	Krisha Chikka
Test Case Description	Test the camera's ability to capture images of currency notes
Prerequisites	Camera permissions must be enabled.
Environmental Information	OS: Android/iOS, System: Mobile device with camera.

Test Scenario:	The user should be able to capture an image of a currency note through the app.

Test Case ID	Test Steps	Test Inputs	Expected Results	Actual Results	Status	Comments
IC_01	1) Initiate Image Capture	Open the camera interface.	The camera is activated, ready for capturing images.	Camera Opened	PASS	The camera view is clear.
	2)Place Note for Capture.	Position an Indian 100 Rs. note under the camera.	The camera detects the note within the frame.	The camera detects the note within the frame.	PASS	No issues found
	3) Capture Image	Capture button press	The image is captured and stored in the application's database.	Image captured.	PASS	No issues found

Module Name	Image Recognition Module
Test Case ID	IR_01
Tester Name	Krisha Chikka
Test Case Description	Test the system's ability to recognize different currency notes.
Prerequisites	The system should have access to a currency database and machine learning
Environmental Information	OS: Android/iOS, System: Mobile device with internet connectivity.

Test Scenario:	The user should be able to scan a note and receive feedback on its denomination.

Test Case ID	Test Steps	Test Inputs	Expected Results	Actual Results	Status	Comments
IR_01	1) Capture a clear image of a currency note.	Indian 100 rs. note	The system should accurately recognize the note as 100 rs.	Recognized correctly	PASS	No issues found
	2) Capture an Unclear or Folded Image of a Note	Crumpled 200 rs. note	The system should either reject the note or ask for better image	Asked/Prompts for a better image	PASS	No issues found

Module Name	Feedback Module
Test Case ID	FB_01
Tester Name	Krisha Chikka
Test Case Description	Test the system's audio and haptic feedback mechanisms.
Prerequisites	System should be connected to audio and haptic hardware.
Environmental Information	OS: Android/iOS, System: Mobile device with speakers and vibration motor.

Test Scenario:	The user should receive audio and/or vibration feedback after scanning a currency note.

Test Case ID	Test Steps	Test Inputs	Expected Results	Actual Results	Status	Comments
FB_01	1) Open Camera	Open camera command	The System should open camera	Opened Camera	PASS	No issues found
	2) Scan the currency note	10 rs. note	The system should announce '10 Rs.' and provide haptic feedback	Announced and vibrated	PASS	No issues found

Conclusion:

In this experiment, we successfully developed test cases for White Box Testing by thoroughly analyzing the source code of the Currency Detector application. By focusing on internal structures, we were able to verify input-output flow and enhance the code's design, usability, and security. White Box Testing allowed us to identify and rectify hidden errors, ensuring code optimization. Through this testing, we gained a deeper understanding of the application's inner workings, highlighting the importance of security and efficiency in the software development process.