CHAPTER No.
TESTING

Chapter No. INTRODUCTION OF TESTING:

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as-yet-undiscovered error. A successful test is one that uncovers an as-yet-undiscovered error. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently as expected before live operation commences.

The software testing process commences once the program is created and the documentation and related data structures are designed. Software testing is essential for correcting errors. Software testing is the critical element of software quality assurance and represents the ultimate the review of specification design and coding.

#. SOFTWARE TESTING:

It is the process of testing the functionality and correctness of software by running it. Process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as yet undiscovered error, successful test is one that uncovers an as yet undiscovered error.

In this project the testing is performed by running the various equipment and integrating with each other and finding all the errors and then debugging the occurred errors and correcting that errors.

#BLACK BOX TESTING:

Applies to software systems or module, tests functionality in terms of inputs and outputs at interfaces. Test reveals if the software function is fully operational with reference to requirements specification Failing this a systematic approach may be necessary. Equivalence partitioning is where the input to a program falls into a number of classes. E.g. positive numbers vs. Negative numbers. Programs normally behave the same way for each member of a class. Partitions exist for both input and output. Partitions may be discrete or overlap. Invalid data (i.e. outside the normal partitions) is one or more partitions that should be tested.

1.Black box testing is done to find incorrect or missing function

2. Interface error

3.Error in external databases

4.Performance error

5.Initialization and termination error

Black box testing is rarely exhaustive (because one doesn't test every value in an equivalence partition) and sometimes fails to reveal corruption defects caused by "weird" combination of inputs. Black box testing should not be used to try and reveal corruption defects caused, for example, by assigning a pointer to point to an object of the wrong type. Static inspection (or using a better programming language!) is preferable for this. In this testing we have tested all the requirement specifications that are required for our project, input functionality and also output functionality.

#WHITE BOX TESTING:

Knowing the internal workings i.e., to test if all internal operations are performed according to program structures and data structures. To test if all internal components have been adequately exercised. Testing based on knowledge of structure of component (e.g. by looking at source code). Advantage is that structure of code can be used to find out how many test cases need to be performed. Knowledge of the algorithm can be used to identify the equivalence partitions. Path testing is where the tester aims to exercise every independent execution path through the component. All conditional statements tested for both true and false cases. If a unit has n control statements, there will be up to 2n possible paths through it. This demonstrates that it is much easier to test small program units than large ones. Flow graphs are a pictorial representation of the paths of control through a program. Use flow graph to design test cases that execute each path. Static tools may be used to make this easier in programs that have a complex branching structure. Tools support. Dynamic program analysers instrument a program with additional code.

SOFTWARE TESTING STRATEGIES:

Testing needs to be planned to be cost and time effective. Planning is setting out standards for tests. Test plans set out the context in which individual engineers can place their own work. Typical test plan contains:

- Unit testing
- Integration testing
- Validation testing
- System testing

 # Unit testing:

It concentrates on each unit of the software as implemented in source code and is a white box oriented. Using the component level design description as a guide, important control paths are tested to uncover errors within the boundary of the module. In the unit testing, the step can be conducted in parallel for multiple components. In this testing we have tested all module individually for checking errors and analysing them.

Integration testing:

Here focus is on design and construction of the software architecture. Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unit tested components and build a program structure that has been dictated by design.
In this testing we have built(combined) all the module that are tested in unit testing and check the build module for checking errors.

#Validation testing:

In this, requirements established as part of software requirements analysis are validated against the software that has been constructed i.e., validation succeeds when software functions in a manner that can reasonably expected by the customer.

 # System testing:

System testing is testing conducted on a complete integrated system to evaluate the system's compliance with its specified requirements. System testing takes, as its input, all of the integrated components that have passed integration testing. The purpose of integration testing is to detect any inconsistencies between the units that are integrated together

 # PRINCIPLES OF SOFTWARE TESTING:

Software testing is an extremely creative and challenging task. Some important principles of software testing are as given: -

- All tests should be traceable to customer requirements.
- Testing time and resources should be limited i.e. avoid redundant testing.
- It is impossible to test everything.
- Use effective resources to test.
- Test should be planned long before testing begins i.e. after requirement phase.
- Test for invalid and unexpected input conditions as well as valid conditions.
- Testing should begin in “in the small” and progress towards testing “in the large”.
- For the most effective testing should be conducted by an independent party.
- Keep software static (without change meanwhile) during test.
- Document test cases and test results

 # BLACK-BOX TESTING FOR Project:

In this testing we have tested all internal operations i.e. checking all modules are working properly working by opening or running the project. And tested that modules working accordingly to program structure and data structure. The easiest way to start is by considering the project as a black box. We don’t have any idea about how it works, what is its specification.

#Test Cases:

Test case is a set of condition or variables under which a testing team or tester will determine whether an application or software system is working correctly or not. The test case can be broadly classified into 3 categories:

1. Formal Test Cases A formal written test case is a test where input is known and input is an expected output which is worked out before the test is executed.

2. Informal Test Cases Informal test are mainly used in scenario testing which are generally of multiple steps and not written down like formal tests.

3. Typical Written Test Cases Typical Written Test Case contain the test case data documented in detail.

Test Case 1: Test Cases for Password

Test Case Description:

In this test case we are going to check and validate login page password and username

Fig No.#

Test Cases for Password:

1.Verify if the login is possible with a valid password.

2.Verify if the separate row for entering the password is visible.

3.Verify the limit of characters for password matches with the specified range.

4.Check if the password is masked or visible in the form of asterisks to ensure secured login.

5.Check if the backspace or delete keys help in removing entered information in case wrong credentials are entered.

6.Check if an error message appears for an invalid password.

7.Check if the login is possible with the new password after the password is reset.

8.Verify that login is not possible with the wrong credentials.

9.Verify that login is only possible within the specified time limit after the password is entered.

Test Case 2: Test Cases for Forget Password

Test Case Description:

In this test case we are going to check and validate login page password and username

Fig No.#

Test Cases for Forgot Password:

1. Check if the forgot password option is shown right after the wrong password is entered.

2. Verify if the forgot password link is working correctly and landing on the correct page.

3. Check if the forgot password link is directed to the right page (i.e. forgot password page).

4. Verify if the link to change the password is sent to the user's email id only.

5. Verify if the security questions asked are the same as the user entered during sign up.

6. Check if a wrong answer is entered to any security question, it should not proceed to the next question.

Test Case 3: Test Cases for Fan Regulator

Test Case Description:

In this test case we are going to check and validate Fan Regulator Slider module:

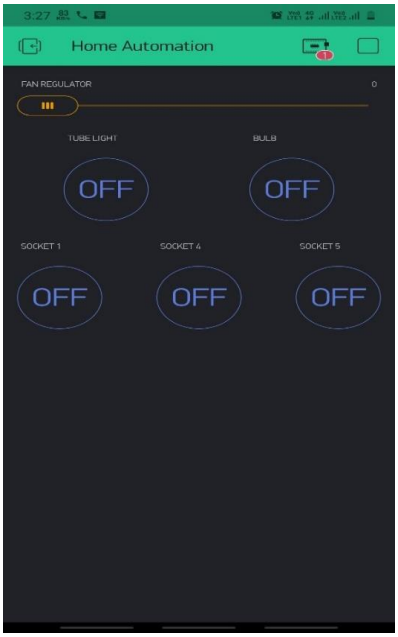

Fig No.#

Test Cases for Fan Regulator Slider module:

1.Check if circuit is closed and all appliances are connected properly

2.Verify if dimmer/regulator are working or not using multimeter

3.Check if the slider is moving or not

4.Verify that if slider is moving towards right side the speed of fan is increasing or vice-versa

5.Verify that only those appliances should be affected those are related to the slider and rest of appliances should remain as it is

Test Case 3: Test Cases for Tube light button

Test Case Description: -

In this test case we are going to check and validate Tube light button module

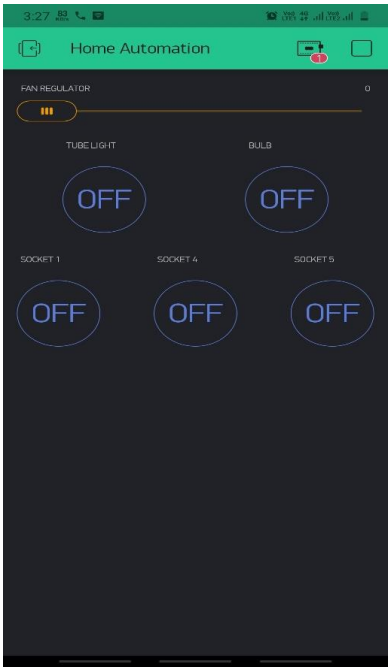

Fig No.#

Test Cases for Tube light module*:*

1.Check if circuit is closed and all appliances are connected properly

2.Verify that tube light is working properly or not using proper equipment

3.Check if the button is working or not i.e. it is getting on / off properly

4.Verify that after pressing button once it should switch on tube light and again pressing button it should be switch off

5.Verify that only those appliances should be affected those are related to the that on/off button and rest of appliances should remain as it is

Test Case 4: Test Cases for bulb button

Test Case Description:

In this test case we are going to check and bulb button module

Fig No.#

Test Cases for bulb button module:

1.Check if circuit is closed and all appliances are connected properly

2.Verify that bulb is working properly or not using proper equipment

3.Check if the button is working or not i.e. it is getting on / off properly

4.Verify that after pressing button once it should switch on bulb and again pressing button it should be switch off

5.Verify that only those appliances should be affected those are related to the that on/off button and rest of appliances should remain as it is

Test Case 5: Test Cases for Socket button module

Test Case Description:

In this test case we are going to check and validate socket button module

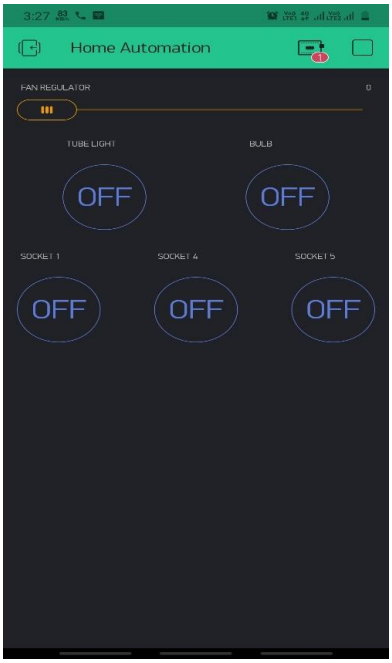

Fig No.#

Test Cases for socket button module:

1.Check if circuit is closed and all appliances are connected properly

2.Verify that connecting device is working properly or not using proper equipment

3.Check if the button is working or not i.e. it is getting on / off properly

4.Verify that after pressing button once it should switch on connecting and again pressing button it should be switch off

5.Verify that only those appliances should be affected those are related to the that on/off button and rest of appliances should remain as it is

Test Case 6: - ESP 32/ Node MCU ESP8266

Test Case Description:

In this test case we are going to check and validate ESP 32/ Node MCU ESP8266 module

Test case for ESP 32:

1.Test that can be done is to power up the board via a USB cable and to check that the red LED lights up as shown in the image below.

2.This confirms that the 3.3V power from the on-board regulator is working.

Test Case 6: - Relays

Test Case Description:

In this test case we are going to check RELAYS

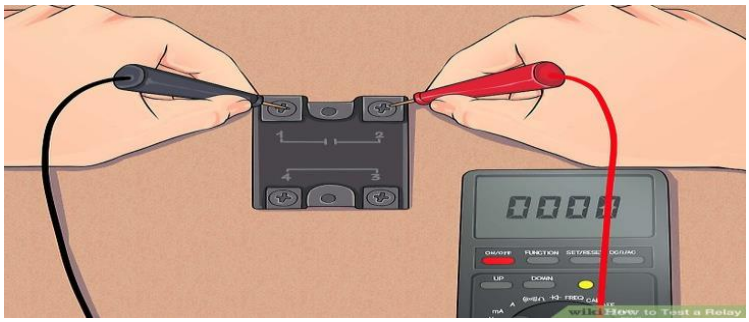

Fig No.#

Test case for RELAYS: -

1.Check the Relay's Coil Terminals Using a Multimeter, to check the relay whether it's good or faulty one, we need to check the resistance of a coil.

2. In the Relay, there is a coil and that coil has some resistance, using that resistance knowledge we can find that the relay is a good one or no.

Test Case 7: - Breadboard

Test Case Description:

In this test case we are going to check Breadboard

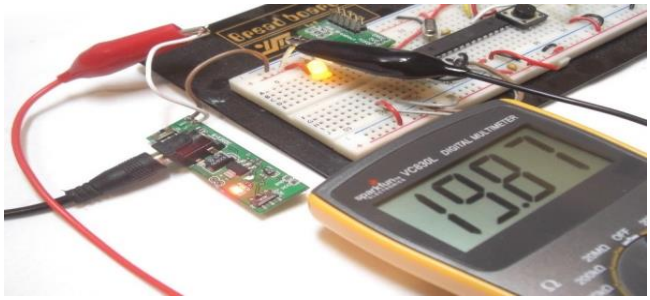

Fig No.#

Test Case for breadboard:

1.You test the isolation of one row from another by touching one probe to "A" and the other probe to the "A" hole in the row below.

2.The row is isolated from the one below it if the multimeter reads infinite.

3. If you question the functionality of other sections of the breadboard, repeat these tests in those places. Find the row number listed on the left side of the breadboard.

4.You test the continuity of the "+" column for a given bus by touching one of the multimeter probes to the top hole and touching the other probe to the bottom hole in the same column.

5.The multimeter will read around zero if the device under test has continuity.

Test Case 7: - Capacitor

Test Case Description:

In this test case we are going to check Capacitor

Fig No.#

Test case for Capacitor:

1.Make sure the capacitor is discharged: One of the main functions of a capacitor is to store power; Set the digital multimeter in a high Ohm range.

2.The ideal meter reading should be above 1000 Ohm = 1k.

3.Connect the meter leads to the capacitor terminals: For a polarized capacitor, connect the red probe to the positive terminal and the black probe to the negative terminal.

4.For a non-polarized capacitor, you can connect it either way. Note the digital resistance reading. The digital multimeter will start reading from zero and move towards infinity

5. It will then stop at a digital resistance value and return to the Open Line. Note the reading and check whether the reading is closer to the resistance value mentioned on the capacitor.