Software Testing Analysis

Team 12
Akshat Goyal, Kanish Anand, Nikunj Nawal, and Sridhar M

1. **Acceptance Testing**

   Level of the software testing process where a system is tested for acceptability. The purpose of this test is to evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery.
   Acceptance Testing is the fourth and last level of software testing performed after System Testing and before making the system available for actual use.
   Blackbox type of testing is based entirely on software requirements and specifications. In testing, we just focus on the inputs and output of the software system without bothering about internal knowledge of the software program.
   Black Box Testing:-
   1. Predict Energy Consumption: It takes four inputs, from date, time and to date, time. If to date, time is less than or equal to from date, time, it shows an invalid date error otherwise it shows the table of predicted energy consumption values.
   2. Register: It has many fields like name, username, email, etc. All fields are necessary, if any field is left unfilled, or filled with invalid format, it shows error, otherwise it accepts the info and redirects to the home page. If we try to register again with the same username, email or mobile no., it shows error.

2. **System Testing**

   SYSTEM TESTING is a level of software testing where a complete and integrated software is tested. The purpose of this test is to evaluate the system's compliance with the specified requirements. System Testing is the third level of software testing performed after Integration Testing and before Acceptance Testing. Normally, independent Testers perform System Testing.

3. **Integration Testing**

   INTEGRATION TESTING is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing.
   - Big Bang is an approach to Integration Testing where all or most of the units are combined together and tested at one go. This approach is taken when the testing

team receives the entire software in a bundle. So what is the difference between Big Bang Integration Testing and System Testing? Well, the former test only the interactions between the units while the latter tests the entire system.

- Top-Down is an approach to Integration Testing where top-level units are tested first and lower-level units are tested step by step after that. This approach is taken when a top-down development approach is followed. Test Stubs are needed to simulate lower-level units which may not be available during the initial phases.
- Bottom-Up is an approach to Integration Testing where bottom level units are tested first and upper-level units step by step after that. This approach is taken when the bottom-up development approach is followed. Test Drivers are needed to simulate higher-level units which may not be available during the initial phases.
- Sandwich/Hybrid is an approach to Integration Testing which is a combination of Top-Down and Bottom-Up Approaches.

Sandwich was the most beneficial approach to integration testing for our product.
We tested the backend api part using drivers like postman and automated testing with pytest while the frontend part was tested using the dummy data or stubs simulating api outputs and then both parts were integrated easily and manually tested their output.
Sandwich Integration allowed us to work on many sub parts simultaneously. We didn't have to wait for one part to work to start the other part. It allows parallel testing and is very time saving.

4. Unit Testing

UNIT TESTING is a level of software testing where individual units/ components of the software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output. In procedural programming, a unit may be an individual program, function, procedure, etc. In object-oriented programming, the smallest unit is a method, which may belong to a base/ super class, abstract class or derived/ child class.
A tester, usually a developer as well, studies the implementation code of a certain field on a webpage, determines all legal (valid and invalid), AND illegal inputs and verifies the outputs against the expected outcomes, which is also determined by studying the implementation code.
In white box testing, we have to ensure that the test cases pass through all the lines of code to get our code fully tested. We made a flow graph for each api where the graph contains nodes as the functions or conditions and directed edges which points towards the called function. Test cases were made making sure that it passes through all the lines of code.

For example, we have ‘predict’ api which takes date and time range as input and returns the expected energy consumption predicted by ml model for that range. On calling this api,

1. It fetches the saved previous predicted data
    a. Then, if we already have that data it takes and returns that data.
    b. Otherwise it calls a function which predicts the energy consumption for that range
2. Then, it calls a function which saves this data and then returns the data.

This code has two different flows. Initially there was no saved data. On giving input date, time range first time, it followed 1 → 2 b). It was verified by the fact that there was saved data after the code was run in the database with the same value returned by the code. On giving the same input again, this time it followed 1 → 2 a) and gave the same data as earlier.
Making the flow graph helped in making the test cases and white box testing and ensured the correct working of the code.