**Validation Testing**

## **Software Testing - Validation Testing**

The process of evaluating software during the development process or at the end of the development process to determine whether it satisfies specified business requirements.

Validation Testing ensures that the product actually meets the client's needs. It can also be defined as to demonstrate that the product fulfills its intended use when deployed on appropriate environment.

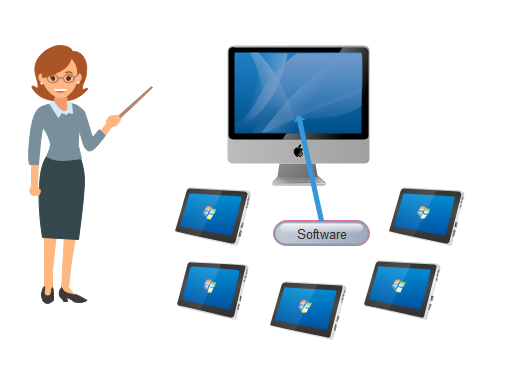
It answers to the question, Are we building the right product?

## **Validation Testing - Workflow:**

Validation testing can be best demonstrated using V-Model. The Software/product under test is evaluated during this type of testing.

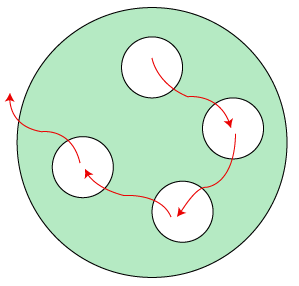
# System Testing

System Testing includes testing of a fully integrated software system. Generally, a computer system is made with the integration of software (any software is only a single element of a computer system). The software is developed in units and then interfaced with other software and hardware to create a complete computer system. In other words, a computer system consists of a group of software to perform the various tasks, but only software cannot perform the task; for that software must be interfaced with compatible hardware. System testing is a series of different type of tests with the purpose to exercise and examine the full working of an integrated software computer system against requirements.



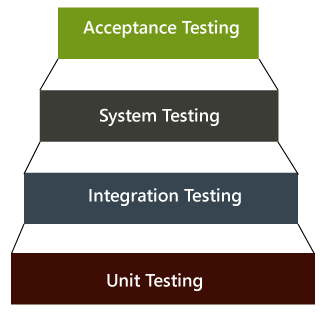
To check the end-to-end flow of an application or the software as a user is known as **System testing**. In this, we navigate (go through) all the necessary modules of an application and check if the end features or the end business works fine, and test the product as a whole system.

It is **end-to-end testing** where the testing environment is similar to the production environment.



There are four levels of [software testing](https://www.javatpoint.com/software-testing-tutorial): [unit testing](https://www.javatpoint.com/unit-testing), [integration testing](https://www.javatpoint.com/integration-testing), system testing and [acceptance testing](https://www.javatpoint.com/acceptance-testing), all are used for the testing purpose. Unit Testing used to test a single software; Integration Testing used to test a group of units of software, System Testing used to test a whole system and Acceptance Testing used to test the acceptability of business requirements. Here we are discussing system testing which is the third level of testing levels.

## **Hierarchy of Testing Levels**



There are mainly two widely used methods for software testing, one is **White box testing** which uses internal coding to design test cases and another is [black box testing](https://www.javatpoint.com/black-box-testing) which uses GUI or user perspective to develop test cases.

ADVERTISEMENT

ADVERTISEMENT

* White box testing
* Black box testing

**System testing falls under Black box testing** as it includes testing of the external working of the software. Testing follows user's perspective to identify minor defects.

System Testing includes the following steps.

* Verification of input functions of the application to test whether it is producing the expected output or not.
* Testing of integrated software by including external peripherals to check the interaction of various components with each other.
* Testing of the whole system for End to End testing.
* Behavior testing of the application via auser's experience

## **Example of System testing**

Suppose we open an application, let say **www.rediff.com**, and there we can see that an advertisement is displayed on the top of the homepage, and it remains there for a few seconds before it disappears. These types of Ads are done by the Advertisement Management System (AMS). Now, we will perform system testing for this type of field.

The below application works in the following manner:

* Let's say that Amazon wants to display a promotion ad on January 26 at precisely 10:00 AM on the Rediff's home page for the country India.
* Then, the sales manager logs into the website and creates a request for an advertisement dated for the above day.
* He/she attaches a file that likely an image files or the video file of the AD and applies.
* The next day, the AMS manager of Rediffmail login into the application and verifies the awaiting Ad request.
* The AMS manager will check those Amazons ad requests are pending, and then he/she will check if the space is available for the particular date and time.
* If space is there, then he/she evaluate the cost of putting up the Ad at 15$ per second, and the overall Ad cost for 10 seconds is approximate 150$.
* The AMS manager clicks on the payment request and sends the estimated value along with the request for payment to the Amazon manager.
* Then the amazon manager login into the Ad status and confirms the payment request, and he/she makes the payment as per all the details and clicks on the **Submit** and **Pay**
* As soon as Rediff's AMs manager gets the amount, he/she will set up the Advertisement for the specific date and time on the Rediffmail's home page.

The various system test scenarios are as follows:

ADVERTISEMENT

ADVERTISEMENT

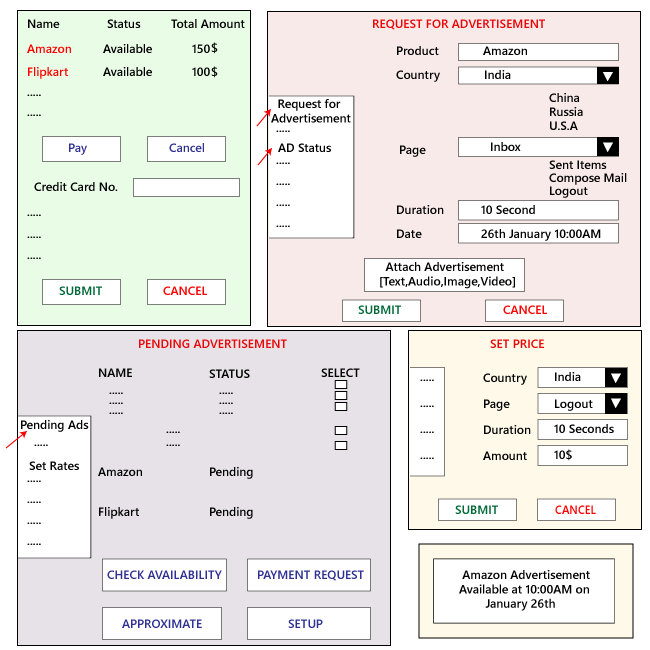
**Scenario1:** The first test is the general scenario, as we discussed above. The test engineer will do the system testing for the underlying situation where the Amazon manager creates a request for the Ad and that Ad is used at a particular date and time.

**Scenario2:** Suppose the Amazon manager feels that the AD space is too expensive and cancels the request. At the same time, the Flipkart requests the Ad space on January 26 at 10:00 AM. Then the request of Amazon has been canceled. Therefore, Flipkart's promotion ad must be arranged on January 26 at 10 AM.

After all, the request and payment have been made. Now, if Amazon changes their mind and they feel that they are ready to make payment for January 26 at 10 AM, which should be given because Flipkart has already used that space. Hence, another calendar must open up for Amazon to make their booking.

**Scenario3:** in this, first, we login as AMS manger, then click on Set Price page and set the price for AD space on logout page to 10$ per second.

Then login as Amazon manager and select the date and time to put up and Ad on the logout page. And the payment should be 100$ for 10 seconds of an Ad on Rediffmail logout page.



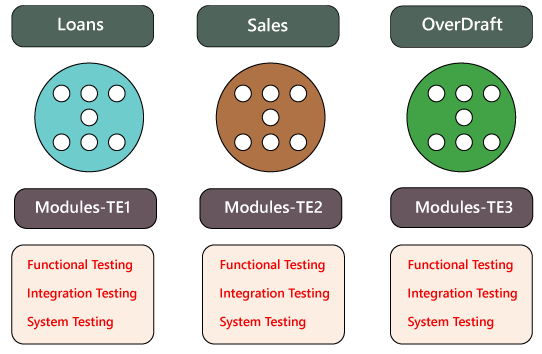
#### **Note: Generally, every test engineer does the functional, integration, and system testing on their assigned module only.**

As we can see in the below image, we have three different modules like **Loans, Sales, and Overdraft**. And these modules are going to be tested by their assigned test engineers only because if data flow between these modules or scenarios, then we need to clear that in which module it is going and that test engineer should check that thing.

Let us assume that here we are performing system testing on the interest estimation, where the customer takes the Overdraft for the first time as well as for the second time.

ADVERTISEMENT

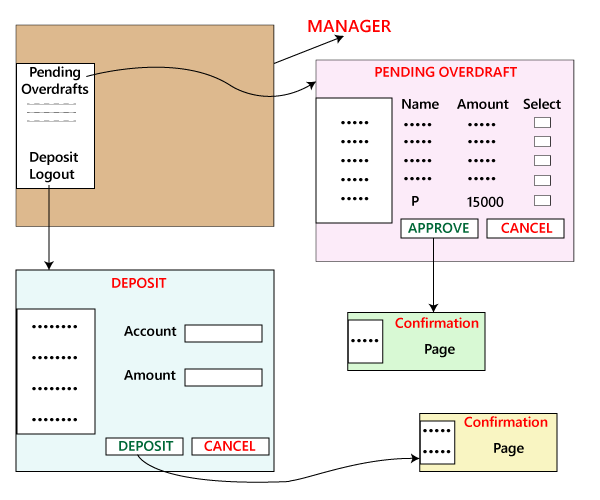
ADVERTISEMENT



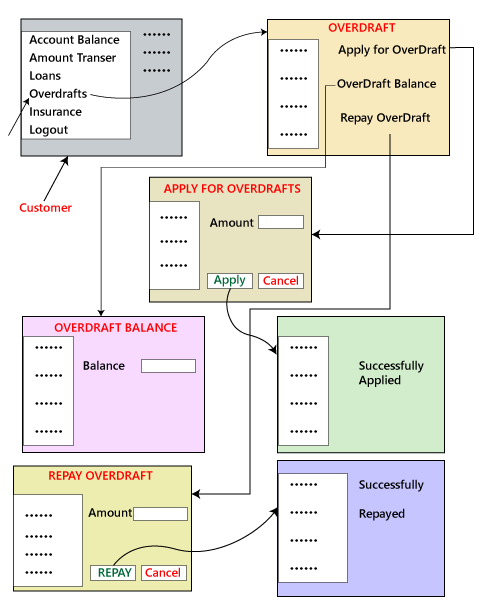
ADVERTISEMENT

In this particular example, we have the following scenarios:

**Scenarios 1**



* First, we will log in as a User; let see P, and apply for Overdraft Rs15000, click on apply, and logout.
* After that, we will log in as a Manager and approve the Overdraft of P, and logout.
* Again we will log in as a P and check the Overdraft balance; Rs15000 should be deposited and logout.
* Modify the server date to the next 30 days.
* Login as P, check the Overdraft balance is 15000+ 300+200=15500, than logout
* Login as a Manager, click on the Deposit, and Deposit Rs500, logout.
* Login as P, Repay the Overdraft amount, and check the Overdraft balance, which is Rs zero.
* Apply for Overdraft in Advance as a two-month salary.
* Approve by the Manager, amount credit and the interest will be there to the processing fee for 1st time.
* Login user → Homepage [Loan, Sales, Overdraft] → Overdraft page [Amount Overdraft, Apply Overdraft, Repay Overdraft] →Application
* Login manager → Homepage [Loan, Sales, Overdraft] → Overdraft page [Amount Overdraft, Apply Overdraft, Repay Overdraft, Approve Overdraft]→ Approve Page →Approve application.
* Login as user P → Homepage [Loan, Sales, Overdraft] → Overdraft page [Amount Overdraft, Apply Overdraft, Repay Overdraft] →Approved Overdraft →Amount Overdraft
* Login as user P→Homepage [Loan, Sales, Overdraft] → Overdraft page [Amount Overdraft, Apply Overdraft, Repay Overdraft] →Repay Overdraft → with process fee + interest amount.



**Scenario 2**

ADVERTISEMENT

ADVERTISEMENT

Now, we test the alternative scenario where the bank provides an offer, which says that a customer who takes Rs45000 as Overdraft for the first time will not charge for the Process fee. The processing fee will not be refunded when the customer chooses another overdraft for the third time.

We have to test for the third scenario, where the customer takes the Overdraft of Rs45000 for the first time, and also verify that the Overdraft repays balance after applying for another overdraft for the third time.

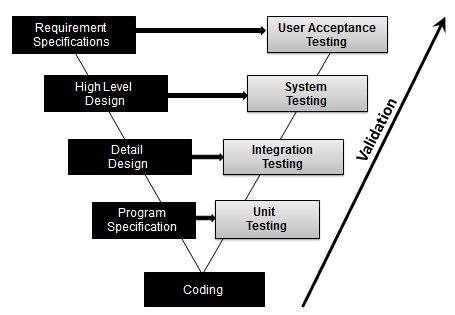
**Scenario 3**

In this, we will reflect that the application is being used generally by all the clients, all of a sudden the bank decided to reduce the processing fee to Rs100 for new customer, and we have test Overdraft for new clients and check whether it is accepting only for Rs100.

But then we get conflicts in the requirement, assume the client has applied for Rs15000 as Overdraft with the current process fee for Rs200. Before the Manager is yet to approve it, the bank decreases the process fee to Rs100.

Now, we have to test what process fee is charged for the Overdraft for the pending customer. And the testing team cannot assume anything; they need to communicate with the Business Analyst or the Client and find out what they want in those cases.

Therefore, if the customers provide the first set of requirements, we must come up with the maximum possible scenarios.



## **Activities:**

* Unit Testing
* Integration Testing
* System Testing
* User Acceptance Testing