Identifying And Implementing the Testing Techniques for Web Applications Through a Case Study

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
in
Software Engineering

Submitted By
Sandeep Singh
(Roll No. 801131025)

Under the supervision of:
Dr. Shivani Goel
(Assistant Professor)

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

June 2013

Certificate

I hereby certify that the work which is being presented in the thesis entitled, "*Identifying And Implementing Testing Techniques for Web Applications Through a Case Study* ", in partial fulfillment of the requirements for the award of degree of Master of Engineering in *Software Engineering* submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Dr. Shivani Goel* and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

(Sandeep Singh)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(Dr. Shivani Goel)
Assistant Professor,
CSED,
Thapar University,
Patiala.

Countersigned by

(Dr. Maninder Singh)
Head
Computer Science and Engineering Department
Thapar University
Patiala

(Dr. S. K. Mohapatra)
Dean (Academic Affairs)
Thapar University
Patiala

Acknowledgement

First of all I would like to thank the Almighty, who has always guided me to work on the right path of the life. This work would not have been possible without the encouragement and able guidance of my supervisor Dr. Shivani Goel. I thank my supervisor for her time, patience, discussions and valuable comments. Her enthusiasm and optimism made this experience both rewarding and enjoyable.

I am equally grateful to Dr. Maninder Singh, *Associate Professor and Head*, Computer Science & Engineering Department, for motivation and inspiration that triggered me for the thesis work. I will be failing in my duty if I don't express my gratitude to, Dr. S. K. Mohapatra, *Senior Professor and Dean of Academic Affairs,* of the University for making provisions of infrastructure such as library facilities, computer labs equipped with net facilities, immensely useful for the learners to equip themselves with the latest in the field. I am also thankful to the entire faculty and staff members of Computer Science and Engineering Department for their direct-indirect help, cooperation, love and affection, which made my stay at Thapar University memorable.

Last but not least, I would like to thank my parents for their wonderful love and encouragement, without their blessings none of this would have been possible. I would also like to thank my elder brother Mandeep Singh, since they insisted that I should do so. I would also like to thank my close friends for their constant support.

Sandeep Singh

(801131025)

Abstract

In today's era of internet, there are many stores which are providing services online. The information about all the products like the brand, price, date of manufacturing, various offer schemes etc. are shown on the website of the shopping stores. Financial transactions may or may not be present in all the online stores for shopping. Feedback can help these stores to identify the areas where the customers are dissatisfied and need improvement. Getting a hard copy feedback form is tedious job and not all users are willing to do so. So online feedback system can help in collecting a large number of data and since it is in soft form, it is easy to analyze quickly. The present research work aims at developing an online feedback system for any store and reusing it for an online movie store.

Since a large number of users can view and access the online web application and provide feedback, an error in the web application itself can portray a bad impression on the customers/users. So presenting the web application to the users is important. To ensure this, testing the web application is important. Various types of testing techniques have been discussed in this research work. The ones suitable for web applications are also highlighted.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Application

Application is just a small program (software) which runs on machine like computers, mobiles etc. Application is abbreviated as apps. This shortened version of application became popular with the Apple iPhone, Android phones and Facebook, which allowed developers to use toolkits to develop applications for their products. An application can refer to a functional category such as payroll, inventory and billing application [1]. An application is another name for computer software. It is something developed by a company or single person that performs a task. Running this application on your computer allows you to utilize it [2]. It became popular with the consumer for mobile applications in smartphones and tablets after Apple debuted the iPhone 3G in 2008. It is just as correct to say "iPhone application" as it is "desktop computer app" [1].

1.2 Online Application

An online application is an application that is accessed by the users over a network such as internet. This term may also mean a computer software application that is coded in a browser-supported programming language (such as JavaScript, combined with a browser-rendered markup language like html) and reliant on a common web browser to render the application executable. An online application therefore is an application where there are two or more components in different places that talk to one another. Online applications can refer to software or programs which you can use through the internet and that don't need to be installed on your home computer. An example of this is Google docs or an online calculator or game, like the games on Facebook are known as online application.

1.3 Software Testing

According to *Edsger W Dijkstra*, testing can prove the presence of errors but not their absence. Software testing is an important phase of software development process that can be easily missed by software developers because of their limited time to complete the project [3]. Hence the software fails to complete the desire task at any state. The major goal of software testing is to discover errors in the software [10], with a secondary goal

of building confidence in the proper operation of the software when testing does not discover errors [11]. Computer software is a major component of an information system whose reliability is critical to the performance of an organization. The reliability of an information system has four facets: people, hardware, software and data. Among these four facets, the reliability of software is a joint responsibility of computer science and information system professionals. The former handles technical software and the latter application software. Regardless of the types of software involved, the reliability of software must be achieved through a continuous effort of planning, analysis, design, programming, testing, installation and maintenance. Most of the software errors detected during the testing phase originate in the early analysis phase. Therefore, software testing should start early in the system development process. Software testing is essential to ensure software quality. Software testing is iterative process which consists of [8]:

- Designing tests
- Executing tests
- Identifying problems
- Getting problems fixed

1.3.1 Difference between desktop application and web application

The following table 1.1 explains the main difference between a desktop application (Calculator program, Yahoo Messenger etc) and a web application (Meebo, Avairy, Google Docs etc).

Table 1.1 Differences between desktop application and web application [32]

S. No.	Functionality	Desktop Application	Web Application
1.	Performance	Faster	Slower
2.	Network Congestion	Depending o the data transfer and connections made to the server from various clients.	Depends
3.	User Interfaces, data binding etc.	Easy to build	Difficult to build
4.	Deployment and Maintenance	Complex. New versions of assemblies, configuration files and other required files must be deployed on all clients machines. Usually user	Easy. Need to deploy assemblies and configuration files on the server only. Transparent to the

		interaction required.	client.
5.	Robustness and Reliability	One client machine goes down, other users are still live.	Usually web servers are never down. However if the server goes down, all users are affected.
6.	Resources	Runs on the client machine.	Runs on a Web server.
7.	Catastrophic failure	User interaction required.	Usually user interaction not required.
8.	Framework dependency	All client machines have to install required versions of .NET framework and other required libraries.	Only server needs to have .NET framework and other required libraries.

1.3.2 Objective of Software Testing

The objective of software testing is to find problems and fix them to improve quality. Software testing typically represents 40% of a software development budget. There are four main objectives of testing [4]:

a. **Demonstration:** It show that the system can be used with acceptable risk, demonstrate function under special conditions and show that products are ready for integration or use.
b. **Detection:** It discovers defects, errors and deficiencies. Determine system capabilities and limitations quality of components, work products and the system.
c. **Prevention:** It provides information to prevent or reduce the number of errors, clarify system specifications and performance. It identifies ways to avoid risks and problems in the future.
d. **Improving quality:** By doing effective testing, we can minimize errors and hence improve the quality of the software.

1.3.3 Need for Testing

Well, while making food, it is ok to have something extra, people might understand and eat the things we made and may well appreciate our work. But this isn't the case with software project development [6]. If we fail to deliver a reliable, good and problem free software solution, we fail in our project and probably we may lose our client. So in order to make it sure, that we provide our client a proper software solution, we go for testing.

We check out if there is any problem, any error in the system, which can make software unusable by the client. We make software testers test the system and help in finding out the bugs in the system to fix on time [13]. We find out the problems and fix them and again try out all the potential problems. Testing is necessary because we all make mistakes. Some of those mistakes are unimportant, but some of them are expensive or dangerous. We need to check everything and anything we produce because things can always go wrong [16]. With testing we can save our time as well as money.

1.3.4 Phases in Software Testing

Although many test teams use test tools or scripts to automate testing activities, there's a lot about testing which is just simply labour intensive [27]. Here are just some of the activities involved:

I. **Planning and developing test cases**: Writing test plans documentation, prioritizing the testing based on assessing the risks, setting up test data, organising test teams [8].
II. **Setting up the test environment:** An application will be tested using multiple combinations of hardware and software and under different conditions. Also, setting up the prerequisites for the test cases themselves [8].
III. **Writing test harnesses and scripts:** Developing test applications to call the API directly in order to automate the test cases. Writing scripts to simulate user interactions [9].
IV. **Planning, writing and running load tests:** Non-functional tests to monitor an application's scalability and performance. Looking at how an application behaves under the stress of a large number of users [6].
V. **Writing bug reports:** Communicating the exact steps required to reproduce unexpected behaviour on a particular configuration. Reporting to development team with the results [8].

1.4 Test Planning and Process

To ensure effective testing proper test planning is important an effective testing process will comprise of the following steps [4]:

- Test Strategy and Planning
- Review Test Strategy to ensure its aligned with the Project Goals

- Design/Write Test Cases
- Review Test cases to ensure proper Test Coverage
- Execute Test Cases
- Track Defects
- Capture Relevant Metrics
- Analyze

The testing process and the cases should cover:

- All the scenarios that can occur when using the software application.
- Each business requirement that was defined for the project.
- Specific levels of testing should cover every line of code written for application.

1.5 Testing in Web Application

According to *Kota,* there are ten quick steps to test your web application [5] and their explanation is given below:

Step 1. Objectives

Whenever you want to test your web application, make sure your testing objectives are measurable. It will make your life a lot easier by having written objectives that your whole team can understand. In addition to documenting your objectives, make sure your objectives are prioritized. Decide about priority between minimal defects and time-to-market.

Here are two examples of how to determine priorities:

If you are building a medical web application that will assist in diagnosing illnesses, and someone could potentially die based on how correctly the application functions, you may want to make testing the correctness of the business functionality a higher priority than testing for navigational consistency throughout the application. If you are testing an application that will be used to solicit external funding, you may want to put testing the aspects of the application that impact the visual appeal as the highest testing priority. Your web application doesn't have to be perfect, it just needs to meet your intended customer's requirements and expectations.

Step 2. Testing Process and Reporting

Make sure that everyone on your testing team knows his or her role. Who should report what, when and to whom? In other words, define your testing process. Use the following questions to help you get started:

- How will issues be reported?
- Who can assign issues?
- How will issues be categorized?
- Who needs what report and when do they need it?
- Are team meetings scheduled in advance or scheduled as needed?

You may define your testing process and reporting requirements formally or informally, depending on your particular needs. The main point to keep in mind is to organize your team in a way that supports your testing objectives and takes into account the individual personalities on your team. One size never fits all when dealing with people.

Step 3. Tracking Testing Results

Once you start executing your test plans, you will probably generate a large number of bugs, issues, defects, etc. You will want a way to easily store, organize, and distribute this information to the appropriate technical team members. You will also need a way to keep management informed on the status of your testing efforts. If your company already has a system in place to track this type of information, don't try to reinvent the wheel. Take advantage of what's already in place. By using an online system, you can make it much easier on yourself by eliminating the need to install and maintain an off-the-shelf package.

Step 4. Setup a Test Environment

Set up a test environment that is separate from your development and production environment. This includes a separate web server, database server, and application server if applicable. You may or may not be able to utilize existing computers to setup a separate test environment. Create an explicitly defined procedure for moving code to and from your test environment and make sure the procedure is followed. Also, work with your development team to make sure each new version of source code to be tested is uniquely identified.

Step 5. Usability Testing

In usability testing, you'll be looking at aspects of your web application that affect the user's experience, such as:

- How easy is it to navigate through your web application?
- Is it obvious to the user which actions are available to him or her?
- Is the look-and-feel of your web application consistent from page to page, including font sizes and colors?

For instance, if a user forgets to fill in a required field, you might think it is a good idea to present the user with a friendly error message and change the color of the field label to red or some other conspicuous color. However, changing the color of the field label would not really help a user who has difficulty deciphering colors. The use of color may help most users, but you would want to use an additional visual clue, such as placing an asterisk beside the field in question or additionally making the text bold.

Step 6. Unit Testing

Unit testing is focused on verifying small portions of functionality. For example, an individual unit test case might focus on verifying that the correct data has been saved to the database when the Submit button on a particular page is clicked.

An important subset of unit testing that is often overlooked is range checking. That is, making sure all the fields that collect information from the user, can gracefully handle any value that is entered. Most people think of range checking as making sure that a numeric field only accepts numbers. In addition to traditional range checking make sure you also check for less common, but just as problematic exceptions.

Step 7. Verifying the HTML

Hyper Text Markup Language (HTML) is the computer language sent from your web server to the web browser on your users' computer to display the pages that make up your web application. The World Wide Web Consortium manages the HTML specification. One major objective of HTML is to provide the ability for anyone from anywhere to access information on the World Wide Web. This concept generally holds true if you conform strictly to the relevant version of the HTML specification that you will support. Unfortunately, in the real world, it is possible for a developer to inadvertently use a proprietary HTML tag that may not work for all of your intended users.

Step 8. Load Testing

In performing load testing, you want to simulate how users will use your web application in the real world. The earlier you perform load testing the better. Simple design changes can often make a significant impact on the performance and scalability of your web application. A good overview of how to perform load testing can be found on Microsoft's Developer Network (MSDN) website.

Step 9. User Acceptance Testing

By performing user acceptance testing, you are making sure your web application fits the use for which it was intended. Simply stated, you are making sure your web application makes things easier for the user and not harder. One effective way to handle user acceptance testing is by setting up a beta test for your web application.

Step 10. Testing Security

With the large number of highly skilled hackers in the world, security should be a huge concern for anyone building a web application. You need to test how secure your web application is from both external and internal threats. The security of your web application should be planned for and verified by qualified security specialists.

1.6 Software Reuse

Reusing an existing application saves a lot of development time and efforts and hence reduces the overall cost and time of application development [29]. Since the reused application is already tested, so the newly developed application from reuse of existing one is also of better quality. The test cases of reused application can also be reused to verify the testing of newly developed application.

1.7 Organization of Thesis

The rest of the thesis is organised as follows:

Chapter 2 covers the literature survey.

Chapter 3 covers the problem statement and the motivation for the thesis work.

Chapter 4 covers the work done during thesis.

Chapter 5 covers the experimental results.

Chapter 6 summarizes conclusion and future scope of the presented work.

Chapter 2
Literature Review

Literature on software testing has appeared since the beginning of computer science [12]. Software testing is the process that is used to measure the quality of developed software [33]. Software testing is an activity to check whether the actual results match the expected results and to ensure that the software system is defect free. Testing is of great importance to investigate ways of increasing the efficiency and effectiveness in quality [38, 39]. Software bugs can potentially cause monetary and human loss, history is full of such examples and that's why software testing is very important. There are many examples of software failures because of bad or no testing [7]:

- In 1985, Canada's Therac-25 radiation therapy machine malfunctioned due to software bug and delivered lethal radiation doses to patients. 3 people died and many others were critically injured.
- In May 1996, a software bug caused the bank accounts of 823 customers of a major U.S. bank to be credited with 920 million US dollars.
- In April 1999, a software bug caused the failure of a $1.2 billion military satellite launch.

Software testing is not a "silver bullet" that can guarantee the production of high quality software systems [14]. Testing is not limited to the detection of "bugs" in the software, but also increases confidence in its proper functioning and assists with the evaluation of functional and non-functional properties. Testing related activities encompass the entire development process and may consume a large part of the effort required for producing software [15].

2.1 History

The separation of debugging from testing was initially introduced by "Glen ford J. Myers" in 1979 [10]. Although his attention was on breakage testing ("a successful test is one that finds a bug") [10, 19] it illustrated the desire of the software engineering community to separate fundamental development activities, such as debugging, from that

of verification. Dr. David Gelperin and Dr. Bill Hetzel classified in 1988 the phases and goals in software testing in the following stages [21]:

- Until 1956 – Debugging oriented
- 1957-1978 – Demonstration oriented
- 1979-1982 – Destruction oriented
- 1983-1987 – Evaluation oriented
- 1988-2005 – Prevention oriented

2.2 What is Software Testing?

Software testing is any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results [17]. Software Testing is the process of executing a program or system with the intent of finding errors [20]. Software testing is very important part of software development. Almost 80% software fails because of not testing properly. The purpose of testing can be quality assurance, verification and validation, or reliability estimation. Software testing is a process of executing a program or application with the intent of finding the software bugs.

- It can also be stated as the process of validating and verifying that a software program or application or product.
- Meets the business and technical requirements that.

Let's break the definition of Software testing into the following parts [18]:

a. **Process:** Testing is a process rather than a single activity.
b. **All Life Cycle Activities:** Testing is a process that's take place throughout the Software Development Life Cycle (SDLC).
    - The process of designing tests early in the life cycle can help to prevent defects from being introduced in the code. Sometimes it's referred as "verifying the test basis via the test design".
    - The test basis includes documents such as the requirements and design specifications.
c. **Static Testing:** It can test and find defects without executing code. Static Testing is done during verification process. This testing includes reviewing of the

documents (including source code) and static analysis. This is useful and cost effective way of testing. For example: reviewing, walkthrough, inspection, etc.

d. **Dynamic Testing:** In dynamic testing the software code is executed to demonstrate the result of running tests. It's done during validation process. For example: unit testing, integration testing, system testing, etc.
e. **Planning:** We need to plan about what we want to do. We control the test activities, we report on testing progress and the status of the software under test.
f. **Preparation:** We need to choose what testing we will do, by selecting test conditions and designing test cases.
g. **Evaluation:** During evaluation we must check the results and evaluate the software under test and the completion criteria, which helps us to decide whether we have finished testing and whether the software product has passed the tests.

8. **Software products and related work products:** Along with the testing of code the testing of requirement and design specifications and also the related documents like operation, user and training material is equally important.

2.2.1 Why Software Fails?

We put a man on the moon. So why can't we make software that works [30]. Software fails even when it is produced on schedule within budget and meets the customer's specified software requirements [25].

There are some basic terminologies are given below [26]:

a. **Error:** It is a discrepancy between a computed, observed or measured value or condition and the true, specified, or theoretically correct value or condition. This can be a misunderstanding of the internal state of the software, an oversight in terms of memory management, confusion about the proper way to calculate a value, etc. Errors also include error sources such as human misunderstandings, misinterpretation and so on [27].
b. **Failure:** The inability of a system or component to perform its required function within specified performance requirements. The incapacity of a system to conduct its required functions within clarified performance requirements [27].
c. **Bug:** A software bug is an error in the program that prevents it from operating properly. Some bugs may only affect a program under specific circumstances. Others

may be more serious and cause the program to unstable or even unstable. A simple mistake in the software code can cause major problems. For example, if the programmer accidentally coded a program to multiply together when it should only add them, the rest of the program will give a faulty result. Most programmers test software thoroughly and some hand over programs to quality assurance team to make certain there are no bugs in the code. Fixing bugs is called debugging [31].

d. **Fault:** A false, wrong step, process or data definition in a software product [27]. An incorrect step, process or data definition in a computer program which causes the program to perform in an unintended or unanticipated manner.

e. **Defect:** Commonly refers to several troubles with the software products, with its external behaviour or with its internal features. Commonly refers to several troubles with the software products, with its external behaviour or with its internal features [27].

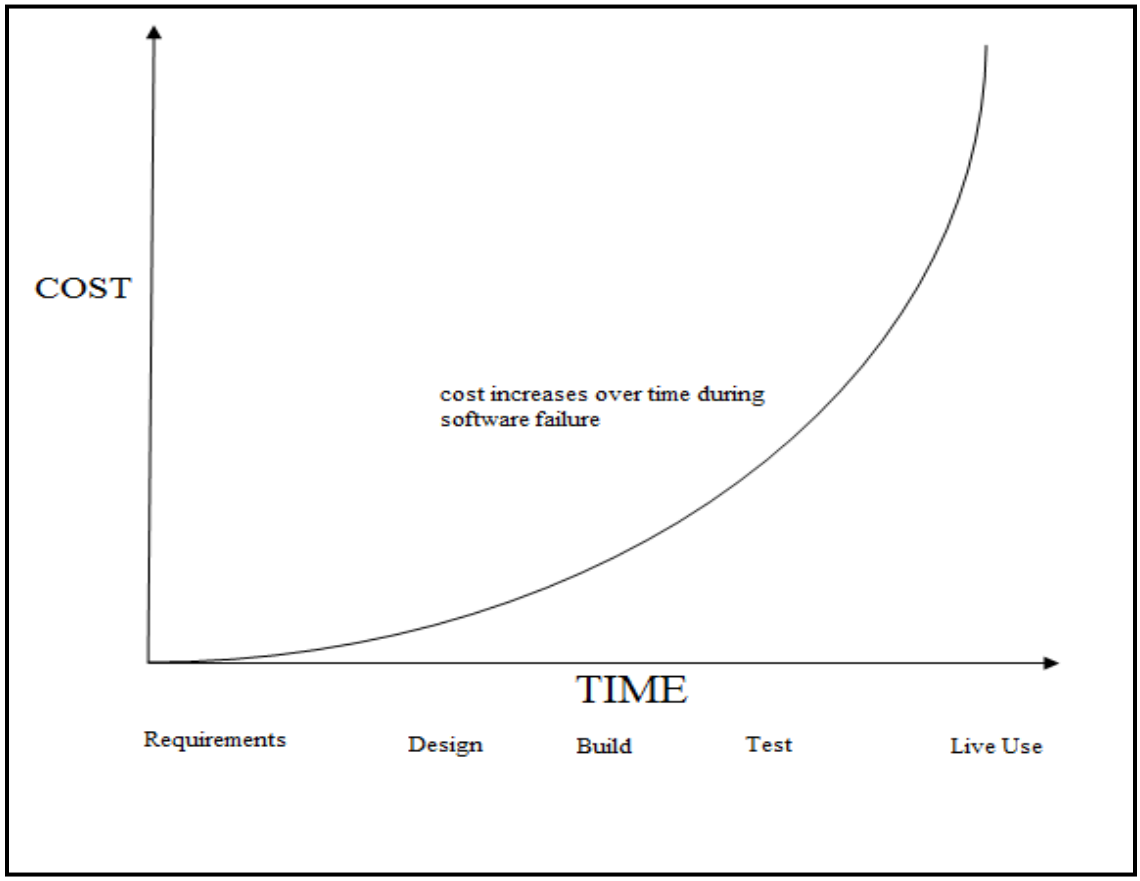


Figure 2.1 Cost of software failure [27]

2.2.2 How much testing is enough?

Testing everything (all combinations of inputs and preconditions) is not feasible except for trivial cases [24]. Instead of exhaustive testing, we use risks and priorities to focus testing efforts.

We've seen that testing helps us find defects and improve software quality. How much testing should we do? We have a choice: test everything, test nothing or test some of the software. Now, your immediate response to that may well be to say, 'Everything must be tested'. We don't want to use software that has not been completely tested, do we? This implies that we must exercise every aspect of a software system during testing. What we need to consider is whether we must, or even can, test completely [24].

We carry out a risk assessment to decide how much testing to do. We can then vary the testing effort based on the level of risk in different areas [34]. Additionally, testing should provide sufficient information to stakeholders to make informed decisions about the release of the software or system we're testing, for the next development step or handover to customers. The effort put into the quality assurance and testing activities needs to be tailored to the risks and costs associated with the project. Because of the limits in the budget, the time, and in testing we need to decide how we will focus our testing, based on the risks [24].

2.2.3 When can we meet our test objectives?

Testing activities should start as early as possible in the software or system development life cycle and should be focused on defined objectives.

We can use both dynamic testing and static testing as a means for achieving similar test objectives. Both provide information to improve both the system to be tested, and the development and testing processes. We mentioned above that testing can have different goals and objectives, which often include [24]:

- Finding defects
- Gaining confidence in and providing information about the level of quality
- Preventing defects.

2.2.4 Is the software defect free?

Testing can show that defects are present, but cannot prove that there are no defects. Testing reduces the probability of undiscovered defects remaining in the software but, even if no defects are found, it is not a proof of correctness.

2.2.5 If we don't find defects does that mean the users will accept the software?

Finding and fixing defects does not help if the system built is unusable and does not fulfill the users' needs and expectations.

There are another important principle we must consider; the customers for software the people and organizations that buy and use it to aid in their day-today tasks - are not interested in defects or numbers of defects, except when they are directly affected by the instability of the software. The people using software are more interested in the software supporting them in completing tasks efficiently and effectively. The software must meet their needs [24].

2.3 What are test design techniques?

By design we mean to create a plan for how to implement an idea and technique is a method or way for performing a task. So, Test Design is creating a set of inputs for given software that will provide a set of expected outputs. The idea is to ensure that the system is working well enough and it can be released with as few problems as possible for the average user [22]. These techniques are complementary and may be used as appropriate for any given test activity, regardless of which level of testing is being performed [23].

Broadly speaking there are two main categories of Test Design Techniques. They are:

1. Static Test Techniques

Static test techniques provide a great way to improve the quality and productivity of software development. It includes the reviews and provides the overview of how they are conducted. The primary objective of static testing is to improve the quality of software products by assisting engineers to recognize and fix their own defects early in the software development process [34].

- Static testing is the testing of the software work products manually.
- It starts early in the life cycle and so it is done during the verification process.

- It does not need computer as the testing of program is done without executing the program. For example: reviewing, walk through, inspection, etc.

2.3.1 What are the uses of Static Testing?

The uses of static testing are as follows:

- Since static testing can start early in the life cycle so early feedback on quality issues can be established [35].
- As the defects are getting detected at an early stage so the rework cost most often relatively low.
- Development productivity is likely to increase because of the less rework effort.
- Types of the defects that are easier to find during the static testing are: missing requirements, design defects, non-maintainable code and inconsistent interface specifications.
- Static tests contribute to the increased awareness of quality issues [24].

2. Dynamic Test Techniques

Under dynamic testing, code is executed. It checks for functional behaviour of software system, memory/CPU usage and overall performance of the system. Hence the name "Dynamic".

Main objective of this testing is to confirm that the software product works in conformance with the business requirements. This testing is also called as Execution technique or validation testing. Dynamic testing executes the software and validates the output with the expected outcome. Dynamic testing is performed at all levels of testing and it can be either black or white box testing [40].

Dynamic testing is a method of assessing the feasibility of a software program by giving input and examining output (I/O). The dynamic method requires that the code be compiled and run. The alternative method of software testing, static testing, does not involve program execution but an examination of the code and associated documents [36].

- This testing technique needs computer for testing.
- It is done during Validation process.
- The software is tested by executing it on computer. Ex: Unit testing, integration testing, system testing etc.

2.4 Difference between Static Testing and dynamic testing

Table 2.1 Difference between Static Testing and dynamic testing [40]	
Static Testing	**Dynamic Testing**
Testing done without executing the program.	Testing done by executing the program.
This testing does verification process.	Dynamic testing does validation process.
Static testing is about prevention of defects.	Dynamic testing is about finding and fixing the defects.
Static testing gives assessment of code and documentation.	Dynamic testing gives bugs/bottlenecks in the software system.
Static testing involves checklist and process to be followed.	Dynamic testing involves test cases for execution.
This testing can be performed before compilation.	Dynamic testing is performed after compilation.
Static testing covers the structural and statement coverage testing.	Dynamic testing covers the executable file of the code.
Cost of finding defects and fixing is less.	Cost of finding and fixing defects is high.
Return on investment will be high as this process involved at early stage.	Return on investment will be low as this process involves after the development phase.
More reviews comments are highly recommended for good quality.	More defects are highly recommended for good quality.
Requires loads of meetings.	Comparatively requires lesser meetings.

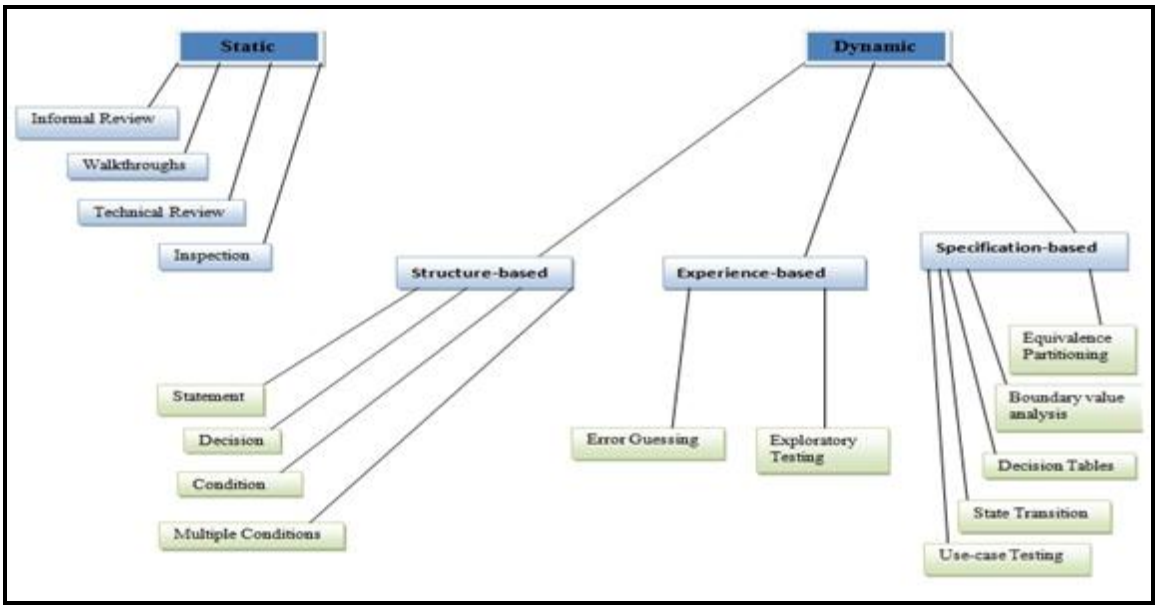


Figure 1.2 Static and Dynamic Testing Techniques [22]

2.5 Types of Testing

Testing is involved in every stage of software life cycle, but the testing done at each level of software development is different in nature and has different objectives. Testing is an activity that is used to discover errors and correct them, so that we are able to create a defect-free product for our customer. Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing also provides an independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs.

Many types of software testing are given below:

2.5.1 Acceptance Testing

It is formal testing conducted to determine whether or not a system satisfies its acceptance criteria and to enable the customer to determine whether or not to accept the system. It is usually performed by the customer [37]. A user story is not considered complete until it has passed its acceptance tests. A principal purpose of acceptance testing is that, once completed successfully and provided certain additional acceptance criteria

are met, the sponsors will then sign off on the system as satisfying the contract and deliver final payment [8].

2.5.2 Accessibility Testing

Accessibility testing is the technique of making sure that your product is accessibility compliant [42]. This testing which determines the usability of a product to the people having disabilities (deaf, blind, mentally disabled etc). The evaluation process is conducted by persons having disabilities [37].

2.5.3 Active Testing

Type of testing consists of introducing test data and analyzing the execution results. It is usually conducted by the testing teams [37]. Active testing has recently been introduced to effectively test concurrent programs. Active testing works in two phases. It first uses predictive off-the-shelf static or dynamic program analyses to identify potential concurrency bugs, such as data races, deadlocks, and atomicity violations. In the second phase, active testing uses the reports from these predictive analyses to explicitly control the underlying scheduler of the concurrent program to accurately and quickly discover real concurrency bugs, if any, with very high probability and little overhead [43,44, 45].

2.5.4 Ad-hoc Testing

Ad hoc testing is a term commonly used for the tests carried out without planning software and documentation. The tests are intended to be executed only once, unless a defect is discovered. This is a part of exploratory test, which is the least formal of test methods. This test is most often used as a complement to other types of tests such as regression tests [6]. Testing performed without planning and documentation - the tester tries to 'break' the system by randomly trying the system's functionality. It is performed by the testing teams [37].

2.5.5 Age Testing

Type of testing which evaluates a system's ability to perform in the future. The evaluation process is conducted by testing teams [37]. This type of testing tells us about the life of the software that means in future how will be the software work.

2.5.6 Agile Testing

The word "Agile" itself goes with the meaning "move quickly" and so the testing. In agile testing, no conventional testing practices are applicable to wait until the entire development cycle activities are completed, whereas the testing is closely intact with the development and is done in parallel as and when a piece of code is developed [50]. It is usually performed by the QA teams [37].

2.5.7 All-pairs Testing

All-pairs testing is an effective test case generation technique that is based on the observation that most faults are caused by interactions of at two factors. Pairwise-generated test suits cover all combinations of two therefore are much smaller than exhaustive ones yet still very effective in finding defects [48]. Combinatorial testing method that tests all possible discrete combinations of input parameters.

2.5.8 Alpha Testing

Alpha testing takes place at developers' sites, and involves testing of the operational system by internal staff, before it is released to external customers. Testing of an application when development is near completion, Minor design changes may still be made as a result of such testing [46].

2.5.9 API Testing

API (Application Programming Interface) testing is mostly used for the system which has collection of API that needs to be tested. The system could be system software, application software or libraries. API testing is different from other testing types as GUI is rarely involved in API Testing. Even if GUI is not involved in API testing, you still need to setup initial environment, invoke API with required set of parameters and then finally analyze the result. Setting initial environment become complex because GUI is not involved. In case of API, you need to have some way to make sure that system is ready for testing [47]. API testing requires understanding both API functionality and possessing good coding skills [41].

2.5.10 Assertion Testing

This type of testing consisting in verifying if the conditions confirm the product requirements. It is performed by the testing teams [37]. In software industry, before deliver a product, we need to run many test assertions to make sure the product met

customer's expectation. This test is a condition that must be tested to confirm conformance to a requirement [51].

2.5.11 Automated Testing

Every software development group tests its products, yet delivered software always has defects. Test engineers strive to catch them before the product is released but they always creep in and they often reappear, even with the best manual testing processes**.** Automated software testing is the best way to increase the effectiveness, efficiency and coverage of your software testing [48]. Testing technique that uses automation testing tools to control the environment set-up, test execution and results reporting. It is performed by a computer and is used inside the testing teams [37].

2.5.12 Backward Compatibility Testing

Backward compatibility refers to a hardware or software system that can successfully use interfaces and data from earlier versions of the system or with other systems. Backward compatibility testing is the kind of testing which ensures software packages built on your previous versions work on the latest versions [48]. Testing method which verifies the behaviour of the developed software with older versions of the test environment.

2.5.13 Basis Path Testing

This testing mechanism was proposed by *McCabe.* Its aim is to derive a logical complexity measure of a procedural design and use this as a guide for defining a basic set of execution paths [46]. Test cases which exercise basic set will execute every statement at least once identifying tests based on flow and paths of a program or system [8]. A testing mechanism which derives a logical complexity measure of a procedural design and use this as a guide for defining a basic set of execution paths. It is used by testing teams when defining test cases.

2.5.14 Benchmark Testing

Benchmark testing is a normal part of the application development life cycle. It is a team effort that involves both application developers and database administrators (DBAs), and should be performed against your application in order to determine current performance and improve it. If the application code has been written as efficiently as possible, additional performance gains might be realized from tuning the database and database

manager configuration parameters. You can even tune application parameters to meet the requirements of the application better [49].

2.5.15 Beta Testing

Beta testing takes places at customer site and involves testing by a group of customers who use the system at their own locations and feedback, before the system is releases to other customers. Testing is done when development and testing are essentially completed and to find final bugs and problems before final release [28]. Successful completion of Beta testing means customer acceptance of the software. It is typically done by end-users or others not by programmers or testers.

2.5.16 Big Bang Integration Testing

In big bang integration testing, individual modules of the programs are not integrated until everything is ready. This approach is seen mostly in inexperienced programmers who rely on 'Run it and see' approach. In this approach, the program is integrated without any formal integration testing, and then run to ensures that all the components are working properly [49]. With testing all components or modules are integrated simultaneously, after which everything is tested as a whole. Big Bang testing has the advantage that everything is finished before integration testing starts. The major disadvantage is that in general it is time consuming and difficult to trace the cause of failures because of this late integration.

2.5.17 Binary Portability Testing

Technique that tests an executable application for portability across system platforms and environments, usually for conformation to an API specification. It is performed by the testing teams [37].

2.5.18 Black box Testing

Black box testing takes an external perspective of the test objects to derive test cases. These tests can be functional or non-functional, through usually functional. The test designer selects valid and invalid inputs and determines the correct output. There is no knowledge of the test object's internal structure. This method of test design is applicable to all levels of software testing: unit, integration, functional, system and acceptance testing [4]. The higher the level, and hence the bigger and more complex the box, the more one is forced to use black box testing to simplify. While this method can uncover

unimplemented parts of the specification, one cannot be sure that all existent paths are tested [8].

2.5.19 Bottom-up Integration Testing

Bottom-up testing is an approach to integration testing where the lowest level components are tested first, then used to facilitate the testing of higher level components. The process is repeated until the component at the top of hierarchy is tested. A bottom-up approach is piecing together systems to give rise to grander systems, thus making the original systems sub-systems of the emergent system. In a bottom-up testing approach the individual base elements of the system are first specified in great detail [4]. In bottom up integration testing, module at the lowest level are developed first and other modules which go towards the 'main' program are integrated and tested one at a time.

2.5.20 Boundary Value Testing

Boundary value analysis (BVA) is based on testing at the boundaries between partitions. Here we have both valid boundaries (in the valid partitions) and invalid boundaries (in the invalid partitions) [49]. Each set, or partition, contains values that are expected to be processed by the component in the same way. Partitioning of test data ranges is done in the test case design technique. It is important to consider both valid and invalid partitions when designing test cases [8]. It is performed by the QA testing teams.

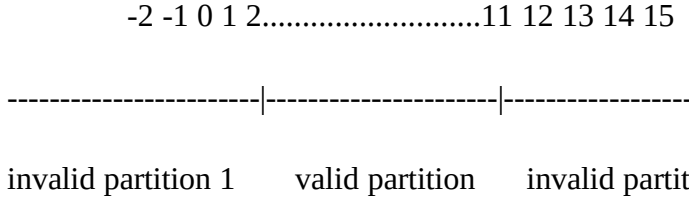


Figure 2.3 Boundary Value Analysis

2.5.21 Branch/Condition Coverage Testing

Branch coverage combines the requirements for decision coverage with those for condition coverage i.e. there must be sufficient test cases to toggle the decision outcome between true and false and to toggle each condition value between true [28]. Hence, a minimum of two test cases are necessary for each decision. Using the example (A or B), test cases (TT) and (FF) would meet the coverage requirement. However, these two tests do not distinguish the correct expression (A or B) from the expression A or from the expression B or from the expression (A or B) [37].

2.5.22 Breadth Testing

A test suite that exercises the full functionality of a product but does not test features in detail. This kind of testing just tests the software overview or test functionally to check the working of the software whether the software working well or not. It is performed by testing teams [37].

2.5.23 Code-driven Testing

Testing technique that uses testing frameworks that allow the execution of unit tests to determine whether various sections of the code are acting as expected under various circumstances [37]. Code driven test testing is a key feature of agile software development, where it is known as Test-driven development (TDD). Unit tests are written to define the functionality before the code is written. Only when all tests pass is the code considered complete. Proponents argue that it produces software that is both more reliable and less costly than code that is tested by manual exploration. It is considered more reliable because the code coverage is better, and because it is run constantly during development rather than once at the end of a waterfall development cycle [49].

2.5.24 Comparison Testing

Comparison testing means comparing your software with better one or your Competitor. While comparison testing we basically compare the performance of the software. For example, if you have to do comparison testing of PDF converter (Desktop Based Application) then you will compare your software with your competitor on the basis of [4]:

a. Speed of conversion PDF file into Word
b. Quality of converted file.

2.5.25 Compatibility Testing

It is a type of non-functional testing. Compatibility testing is a type of software testing used to ensure compatibility of the system/application/website built with various other objects such as other web browsers, hardware platforms, users (in case if it's very specific type of requirement, such as a user who speaks and can read only a particular language), operating systems etc [41]. This type of testing helps find out how well a system performs in a particular environment that includes hardware, network, operating system and other software etc.

2.5.26 Compliance Testing

The Compliance testing program, also known as the OGC Compliance Testing Program**,** is to increase systems interoperability while reducing technology risks. It accomplishes this by providing a process whereby compliance for OGC standards can be tested. This program provides a mechanism by which users and buyers of software that implements OGC standards can be certain that the software follows the mandatory rules of implementation as specified in the standard. It is usually performed by external companies which offer "Certified OGC Compliant" brand [55].

2.5.27 Component Testing

Component testing is also known as module and program testing. It finds the defects in the module and verifies the functioning of software. Before you work with the test suite editor and perform component testing, you should have experience with unit testing and the integration test client. The component testing documentation largely extends the unit testing documentation, so you should familiarize yourself with the topics on unit testing [52].

2.5.28 Concurrency Testing

Concurrency testing also known as multi user testing, is used to know the effects of accessing the application, code module or database by different users at the same time. It helps in identifying and measuring the problems in response time, levels of locking and deadlocking in the application [54]. We can accomplish this by studying the design documents. First check if report tables are used. Every unit that makes use of the report tables needs to be tested for concurrency control. Second we need to see if reservation of tickets or anything of that nature is involved, if so, we need to carry out concurrent testing.

2.5.29 Configuration Testing

Configuration testing is the process of testing a system with each of the supported software and hardware configurations. The Execution area supports configuration testing by allowing reuse of the assigned tests. You can create configuration unites with a set of assigned tests, and all execution plans that you add to the configuration suite will also have the set of tests assigned. You can also create configuration suites from existing execution plans and copy and paste or cut and paste execution plans in the Execution tree

into a configuration suite. Silk Central enables you to add or remove parameters, keywords, and manual testers to or from the configurations [53]. When you create a configuration suite out of an existing execution plan, all the results of the execution plan are preserved in the configuration suite.

2.5.30 Conformance Testing

Conformance testing is a process conducted to determine whether a product or system meets the requirements of a specified standard or standards created by a standards development organization consortium, trade association government agency or other entity to achieve connectivity or interoperability. Effective pre-conformance and conformance testing demands compliance with the latest industry specifications. Anite solutions deliver, cutting risk out of R&D programmes, enabling testing that's better, safer and quicker, significantly reducing time to market [41].

2.5.31 Context Driven Testing

Context-driven software testing is a set of values about test methodology. It is not itself a test technique. To be a context-driven tester is to approach each testing situation as if it were unique in important ways, and to develop the skills to react to situations with a broad and deep awareness of problems in projects and possible testing-related solutions to those problems [37].

2.5.32 Conversion Testing

Testing of programs or procedures used to convert data from existing systems for use in replacement systems. Without conversion testing you're making important decisions based on intuition, instead of fact. There are countless case studies that show how a minor change resulted in x% increase in sales. Conversion testing is all about optimization. You make small changes, measure the results, rinse and repeat [41]. It is usually performed by the QA teams.

2.5.33 CRUD Testing

CRUD testing is actually black box testing. CRUD testing stands for (Create, Read, Update and Delete) i.e. whether you can create or add data, whether you can read or access the data after it is saved once, or whether you can detect the data along with its relationship. CRUD testing is nothing but exploratory testing and this is performed by the

testers itself. Building a CRUD matrix and testing all object creations, reads, updates and deletions [56].

2.5.34 Decision Coverage Testing

It covers both the true and false conditions unlikely the statement coverage. A branch is the outcome of a decision, so branch coverage simply measures which decision outcomes have been tested. This sounds great because it takes a more in-depth view of the source code than simple statement coverage. A decision is an IF statement, a loop control statement (e.g. DO-WHILE or REPEAT-UNTIL), or a CASE statement, where there are two or more outcomes from the statement. With an IF statement, the exit can either be TRUE or FALSE, depending on the value of the logical condition that comes after IF [57]. It is typically made by the automation testing teams.

2.5.35 Dependency Testing

This type of testing which can be applied to components, web services, software services, software, APIs, intent of the dependency testing it to check the configuration, input and output sent by base application to dependent services or software are as per specification [58].

2.5.36 Destructive Testing

Destructive testing (DT) includes methods where your material is broken down in order to determine mechanical properties, such as strength, toughness and hardness. In practice it means, for example, finding out if the quality of a weld is good enough to withstand extreme pressure or to verify the properties of a material [41].

2.5.37 Domain Testing

Domain testing is a software testing technique, objective of domain testing is to select test cases of critical functionality of the software and execute them. Domain testing does not intend to run all existing cases [41].

2.5.38 Dynamic Testing

Dynamic testing is a term used in software engineering to describe the testing of the dynamic behaviour of code. That is dynamic analysis refers to the examination of the physical response from the system to variables that are not constant and change with time. In dynamic testing the software must actually be compiled and run, this is in

contrast to static testing. Dynamic testing is the validation portion of verification and validation [4].

2.5.39 End-to-end Testing

End-to-End testing is the process of testing transactions or business level products as they pass right through the computer system. Thus this generally ensures that all aspects of the business are supported by the systems under test. End-to-End testing is nothing but penetration testing. Whether our application builds is co existing with other existing system or not is called end to end testing [4].

2.5.40 Endurance Testing

Endurance testing is also known as soak testing, longevity testing. Endurance testing is usually performed by performance testing team. This testing is done to check if software can withstand expected load on continuous basis for reasonable longer duration of time, say about a week or a month [41]. During Endurance Testing, performance issue or memory leaks that can be discovered and this is the intent of the endurance testing [37].

2.5.41 Equivalence Partitioning Testing

Equivalence partitioning testing is also known as Equivalence Class Partitioning is software testing technique and not a type of testing by itself. Equivalence partitioning technique is used in black box and grey box testing types. Equivalence partitioning classifies test data into Equivalence classes as positive Equivalence classes and negative Equivalence classes, such classification ensures both positive and negative conditions are tested [41].

2.5.42 Error-Handling Testing

Error handling refers to the anticipation, detection and resolution of programming application and communications errors. Specialized programs, called error handlers are available for some applications. The best programs of this type forestall errors if possible, recover from them when they occur without terminating the application and save the error information to a log file. In programming, a development error is one that can be prevented. Such an error can occur in syntax or logic. Syntax errors, which are typographical mistakes or improper use of special characters, are handled by rigorous proof reading [9].

2.5.43 Exhaustive Testing

Exhaustive Testing, as the name suggests is very exhaustive. In this type of testing we try to check the output given by the software by entering all the possible inputs, in fact we use all the permutations and combinations of the inputs. Each element of code is verified under this process. This way we are able to find software endurance as well as its ability to handle. Exhaustive testing is usually done when the programs and the scope of project is small. For bigger projects exhaustive testing is impractical and is not used. Exhaustive testing is time consuming and costly, thus it has only theoretical significance, a teacher may employ it to guide his ward, who is learning basics of coding [35].

2.5.44 Fault injection Testing

Fault injection Testing is a type of testing that done by testing team with help of development team. Faults or errors are induced in the application and existing tests are executed, in case the introduced errors are caught, that means existing tests are good and robust else additional tests have to be included. Fault injection testing is very important in case of defence, aerospace, medical or any critical software that can lead to financial or loss of life [41].

2.5.45 Formal verification Testing

Formal verification testing helps confirm that your embedded system software models and code behave correctly. These verification methods rely on mathematically rigorous procedures to search through all possible execution paths of your model or code base to identify errors in your design [59]. It is usually performed by QA teams.

2.5.46 Functional Testing

Functional testing is a formal type of testing performed by testers. Functional testing focuses on testing software against design document, use cases and requirements document. Functional testing is a black box type of testing and does not require internal working of the software unlike white box testing [41].

2.5.47 Fuzz Testing

Fuzz testing or fuzzing is a software testing technique used to discover coding errors and security loopholes in software, operating systems or networks by inputting massive amounts of random data, called fuzz, to the system in an attempt to make it crash. If vulnerability is found, a tool called a fuzz tester (or fuzzer), indicates potential causes.

Fuzz testing was originally developed by Barton Miller at the University of Wisconsin in 1989 [60].

2.5.48 Glass box Testing

Glass box testing produces an altogether better application scanning experience. It improves accuracy, coverage, performance and reporting. Although a regular black box scan relies mostly on a server response to determine if vulnerability exists, glass box technology can scan the internal actions and structure of the web application. The result is scans that are more complete [37]. Glass-box-assisted scanning can also identify more complex vulnerabilities and report the exact location of the vulnerability in the source code.

2.5.49 Globalization Testing

The goal of globalization testing is to detect potential problems in application design that could inhibit globalization. It makes sure that code can handle all international support without breaking functionality that would cause either data loss or display problems. Globalization testing checks proper functionality of the product with any of the culture/locale setting using every type of international input possible. Proper functionality of the product assumes both a stable component that works according to design specification regardless of international environment setting or cultures/locales and the correct representation off data [28].

2.5.50 Gorilla Testing

Gorilla Testing is that which is always used to describe repetitive and boring (frustrating) Manual Testing process which a tester has already done a hundred times before. Due to this repetitive manual testing process Gorilla Testing is also known by the name Frustrating Testing. Actually Gorilla Testing is used in Software Testing to check functionality of one particular module heavily. As you know Software Testing is a very wide concept so concept of Gorilla Testing is also wide in nature [37].

2.5.51 Gray Box Testing

Gray box testing is a software testing technique that uses a combination of black box testing and white box testing. Gray box testing is not black box testing, because the tester does know some of the internal workings of the software under test. In gray box testing, the tester applies a limited number of test cases to the internal workings of the software

under test. In the remaining part of the gray box testing, one takes a black box approach in applying inputs to the software under test and observing the outputs. Gray box testing is a powerful idea [4].

2.5.52 GUI Software Testing

The aimed at testing the software GUI (Graphical User Interface) of the software meets the requirements as mentioned in the GUI mock-ups and Detailed designed documents. For e.g. checking the length and capacity of the input fields provided on the form, type of input field provided, e.g. some of the form fields can be displayed as dropdown box or a set of radio buttons. So GUI testing ensures GUI elements of the software are as per approved GUI mock-ups, detailed design documents and functional requirements [41]. Most of the functional test automation tools work on GUI capture and playback capabilities. This makes script recording faster at the same time increases the effort on script maintenance.

2.5.53 Hybrid Integration Testing

In hybrid integration testing approach, we try to leverage benefits of top-down and bottom-up, both the types of testing. While it is important to take benefit of both the approaches using hybrid integration techniques, we need to make sure that we do it thoroughly. Otherwise it will be very difficult to identify which modules are tested using top down and which one are tested using bottom up. You might end up missing some modules altogether in this case if proper caution is not exercised [41].

2.5.54 Install/uninstall Testing

Installation testing is like introducing a guest in your home. The new guest should be properly introduced to all the family members in order to feel him comfortable. If your installation is successful on the new system then customer will be definitely happy but what if things are completely opposite. If installation fails then our program will not work on that system not only this but can leave user's system badly damaged [37]. User might require to reinstall the full operating system.

2.5.55 Integration Testing

While software modules may function well by themselves when they are developed, getting them to work together efficiently and correctly is another matter. After they have been coded and tested individually, individual software components are combined to

form a final software product. During this integration effort, tests are performed on various grouping of components to determine how well they work together. Incompatibilities, errors and integrated and debugged so they function correctly as a whole [4].

2.5.56 Interface Testing

Software provides support for one or more interfaces like "Graphical user interface", "Command Line Interface" or "Application programming interface" to interact with its users or other software. Interfaces serves as medium for software to accept input from user and provide result. Approach for interface testing depends on the type of the interface being testing like GUI or API or CLI [41].

2.5.57 Internationalization Testing

Internationalization is the process of designing an application so that it can be adapted to various languages and regions without engineering changes. Internationalization testing poses a set of unique challenges, including [61]:

- Coping with an exponential increase in testing effort with every new language added, as the tests performed in one language needs to be repeated on every other language.
- Lengthening of time-to-market cycles due to the increased testing effort.
- Managing the increased complexity of testing the application on all languages, including cultural, formatting, and local requirements. This impacts the coverage and quality of testing performed on these applications.
- Improving return on investment from I18N testing, despite the increased cost of highly skilled resources (technical & language proficient) and the testing time.

2.5.58 Inter-Systems Testing

There is software, some built in the OS and many after market that can do in system "house cleaning" and testing. One example is PC Alert. It shows up as a small open eye in task bar and when we move mouse over it, reads out my CPU temp. If I right click it and go to the main page, I can test the inter system of every function of my computer as well as all the external systems as well. An inter system test, for example, could be a memory map, showing what is in memory at that moment. Another is a test of my fans, speed, efficiency etc [4].

2.5.59 Keyword-driven Testing

This is a software testing methodology for automated testing that separates the test creation process into two distinct stages: a Planning Stage and an Implementation Stage. Keyword-Driven Testing is also called Keyword Testing in Test Complete and consists of operations that correspond to automated testing actions. Each operation specifies the action to be simulated and the parameters associated with the action. The test engine executes the automated test, operation-by-operation, and thus simulates user actions on the application under test [37].

2.5.60 Load Testing

Load testing is the process of putting demand on a system or device and measuring its response. In mechanical systems it refers to the testing of the testing of a system to certify it under the appropriate regulations. Load testing is usually carried out a load 1.5 times the Safe Working Load. Periodic recertification is required. The term load testing is used in different ways in the professional software testing community. Load testing generally refers to the practice of modeling the expected usage of a software program by simulating multiple users accessing the program concurrently [28].

2.5.61 Localization Testing

Localization is the process of customizing a software application that was originally designed for a domestic market so that it can be released in foreign markets. This process involves translating all native language strings to the largest language and customizing the GUI so that it is appropriate for the target market [8].

2.5.62 Loop Testing

Loop Testing is a white box testing technique that focuses exclusively on the validity of loop constructs. This is the testing of a resource or resources multiple times under program control. The looping is controlled by the Diagnostic Controller. Loop testing is only supported when running in maintenance mode or service mode. Four classes of loops can be defined: Simple loops, Concatenated loops, nested loops, and unstructured loops [41].

2.5.63 Manual Scripted Testing

Manual scripted testing is the oldest and most rigorous type of software testing. In this particular type of testing, test cases are designed and reviewed by the team before

executing it [37]. There are many variation of this basic approach, test cases can be created at the basic functionality level or they can be created at the scenario level.

2.5.64 Manual Support Testing

System commence when transactions originate and conclude with use of the results of processing. The manual part of the system requires the same attention to testing, as does the automated segment. Although the timing and testing methods may be different, the objectives of manual testing remain the same as testing automated segment of the application system [62].

2.5.65 Modularity-driven Testing

The test script modularity framework requires the creation of small, independent scripts that represent modules, sections, and functions of the application under test. These small scripts are then used in a hierarchical fashion to construct larger tests, realizing a particular test case. Of all the frameworks, this one should be the simplest to grasp and master. It is a well-known programming strategy to build an abstraction layer in front of a component to hide the component from the rest of the application [37].

2.5.66 Mutation Testing

Mutation testing is a method of software testing, which involves modifying program's source code in small ways. These, so-called mutations, are based on well-defined mutation operators that either mimic typical programming errors or force the creation of valuable tests. The purpose is to help the tester develop effective tests or locate weaknesses in the test data used for the program or in sections of the code that are seldom or never accessed during execution [4].

2.5.67 Navigation testing

Navigation means jumping to a new webpage on a click to a link. Navigation testing is carried out to ensure that the user can move around the application in the manner intended [19]. That is, it is important to check that on clicking all the links, the user is able to get the right webpage or section of the webpage. The navigational characteristics of the web application are all the internal and external links within the application. Navigation testing is used to check whether all hyperlinks are functioning correctly, whether there are any broken links/hyperlinks which may lead the user to a wrong or no

webpage and to ensure whether there is a smooth transition between screens of the web application or not.

2.5.68 Negative Testing

Also known as "test to fail" testing method where the tests' aim is showing that a component or system does not work [37]. These are functional and non-functional tests that are intended to break the software by entering incorrect data like incorrect date, time or string or upload binary file when text files supposed to be upload or enter huge text string for input fields etc [41].

2.5.69 Non-functional Testing

Software are built to fulfill functional and non-functional requirements, non-functional requirements like performance, usability, localization etc., There are many types of testing like compatibility testing, compliance testing, localization testing, usability testing, volume testing etc., that are carried out for checking non-functional requirements [41].

2.5.70 Operational Testing

It is also known as pre go live testing is usually performed by software testers. As the name suggests, Operational readiness testing intends to validate the production environment after new version of the software is deployed in production environment [63]. Software testers test the existing and new functionality to certify that the software is ready to be used by end users.

2.5.71 Orthogonal Array Testing

It is a black box testing technique. Orthogonal array Testing is statistical and systematic way of Software testing. Orthogonal array Testing technique helps to minimize the number of test cases and maximize test coverage by grouping set of test conditions. Orthogonal testing is effective in case of GUI testing, Configuration testing where there are multiple input parameters and testing one parameter at a time would lead to large number of test cases [41].

2.5.72 Pair Testing

Software development technique in which two team members work together at one keyboard to test the software application. One does the testing and the other analyzes or reviews the testing. This can be done between one Tester and Developer or Business

Analyst or between two testers with both participants taking turns at driving the keyboard [37].

2.5.73 Parallel Testing

Testing technique which has the purpose to ensure that a new application which has replaced its older version has been installed and is running correctly [37]. It is a software testing technique, where in you test two or more versions of the software "the current version" and "previous version or versions" of the software together to see the differences of existing functionality [41].

2.5.74 Passive Testing

Testing technique consisting in monitoring the results of a running system without introducing any special test data. Passive testing, which consists of testers following a script in order to get information about the software. Even worse if a single engine bursts, the plan will become obsolete instantly [41].

2.5.75 Path Testing

It is a type of software testing technique that is used as part of white box testing approach. Objective of path testing is to exercise and test each of the branch statements. These testing techniques are applied by developers while performing Unit testing [41]. The goal to satisfy coverage criteria for each logical path through the program [37].

2.5.76 Penetration Testing

It is a type of security testing, also known as pentest in short. Penetration testing is done to tests how secure software and its environments Hardware, Operating system and network) are when subject to attack by an external or internal intruder. Intruder can be a hacker or malicious programs. Pentest uses methods to forcibly intrude (by brute force attack) or by using a weakness (vulnerability) to gain access to a software or data or hardware with an intent to expose ways to steal, manipulate or corrupt data, software files or configuration Usually they are conducted by specialized penetration testing companies [41].

2.5.77 Performance Testing

It is a type of software testing and part of performance engineering that is performed to check some of the quality attributes of software like Stability, reliability, availability.

Performance testing is carried out by performance engineering team. Unlike Functional testing, Performance testing is done to check non-functional requirements [37].

2.5.78 Qualification Testing

Testing against the specifications of the previous release, usually conducted by the developer for the consumer, to demonstrate that the software meets its specified requirements [37].

2.5.79 Ramp Testing

Type of testing consisting in raising an input signal continuously until the system breaks down. It may be conducted by the testing team or the performance engineer [37].

2.5.80 Recovery Testing

Testing technique which evaluates how well a system recovers from crashes, hardware failures, or other catastrophic problems. This testing is the forced failure of the software in a variety of ways to verify that recovery is performed [8].

2.5.81 Regression Testing

This is a type of software testing that is carried out by software testers as functional regression tests and developers as Unit regression tests. Objective of regression tests are to find defects that got introduced to defect fix (es) or introduction of new feature(s). Regression tests are ideal candidate for automation [41].

2.5.82 Requirements Testing

Testing technique which validates that the requirements are correct, complete, unambiguous, and logically consistent and allows designing a necessary and sufficient set of test cases from those requirements. We throw out a net and try to capture all these criteria Blitzing, Rapid Application Development (RAD), Joint Application Development (JAD) etc [4].

2.5.83 Sanity Testing

This is a type of testing that is carried out mostly by testers and in some projects by developers as well. Sanity testing is a quick evaluation of the software, environment, network, external systems are up & running, software environment as a whole is stable enough to proceed with extensive testing. Sanity tests are narrow and most of the time sanity tests are not documented [41].

2.5.84 Scalability Testing

It is a non functional test intended to test one of the software quality attributes i.e. "Scalability". Scalability test is not focused on just one or few functionality of the software instead performance of software as a whole. Scalability testing is usually done by performance engineering team. Objective of scalability testing is to test the ability of the software to scale up with increased users, increased transactions, increase in database size etc [41].

2.5.85 Scenario Testing

Scenario testing is done to make sure that the end to end functioning of software is working fine, or all the business process flows of the software are working fine. In scenario testing the testers put themselves in the end users shoes and figures out the real world scenarios or use cases which can be performed on the software by the end user [37].

2.5.86 Security Testing

It is a type of software testing carried out by specialized team of software testers. Objective of security testing is to secure the software is to external or internal threats from humans and malicious programs. Security testing basically checks, how good is software's authorization mechanism, how strong is authentication, how software maintains confidentiality of the data [41].

2.5.87 Smoke Testing

This is a type of testing that is carried out by software testers to check if the new build provided by development team is stable enough i.e. major functionality is working as expected in order to carry out further or detailed testing. Smoke testing is intended to find "show stopper" defects that can prevent testers from testing the application in detail. Smoke testing carried out for a build is also known as building verification test [37].

2.5.88 Stability Testing

It is a non functional test intended to test one of the software quality attributes i.e. "Stability". Stability testing focuses on testing how stable software is when it is subject to loads at acceptable levels, peak loads, loads generated in spikes, with more volumes of data to be processed. Scalability testing will involve performing different types of

performance tests like load testing, stress testing, spike testing, soak testing, spike testing etc [8].

2.5.89 Statement Testing

Statement testing is a white box, dynamic testing technique. It requires examination of the source code and the creation of tests that will exercise individual statements. The project plan should indicate the proportion of statements that should be tested, this is usually expressed as a percentage of "statement coverage" [41].

2.5.90 Static Testing

This is a form of testing where in approaches like reviews, walkthroughs are employed to evaluate the correctness of the deliverable. In static testing software code is not executed instead it is reviewed for syntax, commenting, naming convention, size of the functions and methods etc [41].

2.5.91 Storage Testing

Testing type that verifies the program under test stores data files in the correct directories and that it reserves sufficient space to prevent unexpected termination resulting from lack of space. It is usually performed by the testing team [37].

2.5.92 Stress Testing

Testing technique in which software is subjected to peak loads and even to a break point to observe how the software would behave at breakpoint. Stress testing also tests the behaviour of the software with insufficient resources like CPU, Memory, Network bandwidth, Disk space etc. Stress testing enables to check some of the quality attributes like robustness and reliability [41].

2.5.93 Structural Testing

White box testing technique which takes into account the internal structure of a system or component and ensures that each program statement performs its intended function. It is usually performed by the software developers [37].

2.5.94 System Testing

This testing technique includes multiple software testing types that will enable to validate the software as a whole (software, hardware and network) against the requirements for which it was built. Different types of tests (GUI testing, Functional testing, Regression

testing, Smoke testing, load testing, stress testing, security testing, stress testing, ad-hoc testing etc.,) are carried out to complete system testing [41].

2.5.95 System Integration Testing

As the name suggests, focus of System integration testing is to test for errors related to integration among different applications, services, third party vendor applications etc., As part of SIT, end-to-end scenarios are tested that would require software to interact (send or receive data) with other upstream or downstream applications, services, third party application calls etc [8].

2.5.96 Thread Testing

Thread testing is the testing of system by running multiple users as one process. It is basically used for load testing. In simple terms, when testers focus on testing individual logical execution paths in context of entire system, it is called as Thread Testing. It is also an integration testing methodology that is gaining popularity [8].

2.5.97 Top Down Integration Testing

This is one of the integration testing approaches. Top down integration testing is an incremental testing approach for integration testing where in testing of top level modules are done first before moving on to testing of branch modules. Top down integration testing helps to find design issues in the beginning of the Integration test stage [41].

2.5.98 Upgrade Testing

Testing technique that verifies if assets created with older versions can be used properly and that user's learning is not challenged. It is performed by the testing teams [37]. It verifies that the software will work properly after the upgrades.

2.5.99 Unit Testing

This is a type of testing that is performed by software developers [37]. Unit testing follows white box testing approach where developer will test units of source code like statements, branches, functions, methods OR class, interface in OOP (object oriented programming). Unit testing usually involves in developing stubs and drivers.

2.5.100 Usability Testing

This is a type of software testing that is performed to understand how user friendly the software is. Objective of usability testing is to allow end users to use the software, observe their behaviour, their emotional response (whether users liked using software or

were they stressed using it? etc.,) and collect their feedback on how the software can be made more useable or user friendly and incorporate the changes that make the software easier to use [37].

2.5.101 User Interface Testing

Type of testing which is performed to check how user-friendly the application is. Basically in this type of testing test, how user interacts with application, testing application elements like fonts, layouts, button, images, colors etc [41]. And verifies that it is according to the user requirements or not.

2.5.102 Volume Testing

Testing which confirms that any values that may become large over time (such as accumulated counts, logs, and data files), can be accommodated by the program and will not cause the program to stop working or degrade its operation in any manner[8]. It is usually conducted by the performance engineer.

2.5.103 Vulnerability Testing

It involves identifying, exposing the software, hardware or network Vulnerabilities that can be exploited by hackers and other malicious programs likes viruses or worms. Vulnerability Testing is key to software security and availability [41]. With increase number of hackers and malicious programs, Vulnerability Testing is critical for success of a Business.

2.5.104 White box Testing

White box testing is also known as clear box testing, transparent box testing and glass box testing. White box testing is a software testing approach, which intends to test software with knowledge of internal working of the software [37]. White box testing approach is used in Unit testing which is usually performed by software developers [41]. White box testing intends to execute code and test statements, branches, path, decisions and data flow within the program being tested. White box testing and Black box testing complement each other as each of the testing approaches has potential to uncover specific category of errors.

2.5.105 Workflow Testing

This type of testing executes workflows and tasks using individual test data. If a start transaction or start form is defined for starting a workflow, this is executed during the test

[41]. If the workflow or task is started by a triggering event, you must manually enter values for the import parameters that are normally entered by a binding from the event container. To start execution, you must be a possible agent for the workflow or task.

2.6 Summary

There are many types of testing that can be used to test any software but there are some specific testing techniques will be used in testing the web applications such as, *"Functional Testing", "Security Testing", "Black Box Testing", "Alpha Testing", "Beta Testing", "Database Testing", "Top Down Testing", "Performance Testing", "White Box Testing", "Integration Testing", "Navigation Testing", "Acceptance Testing", "Feasibility Study Chart".* To use all these types of testing techniques a web application, *"Online Movie Store Rent/Buy and Feedback"* is developed and used. On this web application these testing techniques are applied.

Chapter 3
Problem Statement

3.1 Problem Definition

The study of various software development process models reveal that in almost all these models, software testing is included as one phase, but testing is required at each phase and not a particular stage. The main purpose of software testing is to uncover errors which are not simply syntax errors in code but various other types of errors in all the document produced during the software development, e.g. software requirements document, design document, test plan etc. Various types of testing techniques have been developed till date, but which type of testing technique will be suitable and sufficient for checking a particular document in which phase of software development life cycle (SDLC) is not yet clear. Also the web application differs from a desktop application. Developing and testing a web application is complex, difficult and challenging. The objective of present research is

1. To design a cycle for testing in various SDLC phases of developing a web application so that it can be applied throughout the development of a web application.

2. Apply the proposed testing techniques for a sample web application developed with reuse.

3.2 Justification

By categorizing which type of testing to be applied at which phase of software development life cycle will help us plan for testing in that phase efficiently and to take full advantage of all types of testing techniques to improve the quality in that phase and consequently the overall quality of the software project. Also, in order to take the advantages of software reuse, the sample web application has been developed using reuse. The similar test cases can be used for all applications developed with reuse.

Chapter 4
Work Done

4.1 Apply testing during SDLC phases

The six stages of the SDLC are designed to build on one another, taking the outputs from the previous stage, adding additional effort, and producing results that leverage the previous effort and are directly traceable to the previous stages. This top-down approach is intended to result in a quality product that satisfies the original intentions of the customer.

1. Feasibility analysis: A feasibility study looks at the viability of an idea with an emphasis on identifying potential problems and attempts to answer one main question: Will this idea work and should we proceed with it? Basically it includes analysis of project requirements in terms of input data and desired output.

2. Requirement analysis and specification: It includes gathering, analyzing, validating, and specifying requirements. At the end of this phase, the Software Requirement Specification (SRS) document is prepared. SRS is a formal document that acts as a written agreement between the development team and the customer. SRS acts as input to the design phase and includes functional, performance, software, hardware, and network requirements of the project.

3. Design: In designing phase, the translation of the requirements specified in the SRS into a logical structure that can be implemented in a programming language. The output of the design phase is a design document that acts as an input for all the subsequent SDLC phases.

4. Coding: As the name implies basically it **i**ncludes implementation of the design specified in the design document into executable programming language code. The output of the coding phase is the source code for the software that acts as input to the testing and maintenance phase.

5. Testing: The testing process starts with a test plan that recognizes test-related activities, such as test case generation, testing criteria, and resource allocation for testing. It includes detection of errors in the software.

6. Maintenance: This phase is the last phase of SDLC that includes implementation of changes that software might undergo over a period of time, or implementation of new requirements after the software is deployed at the customer location. The maintenance phase also includes handling the errors that may exist in the software even after the testing phase.

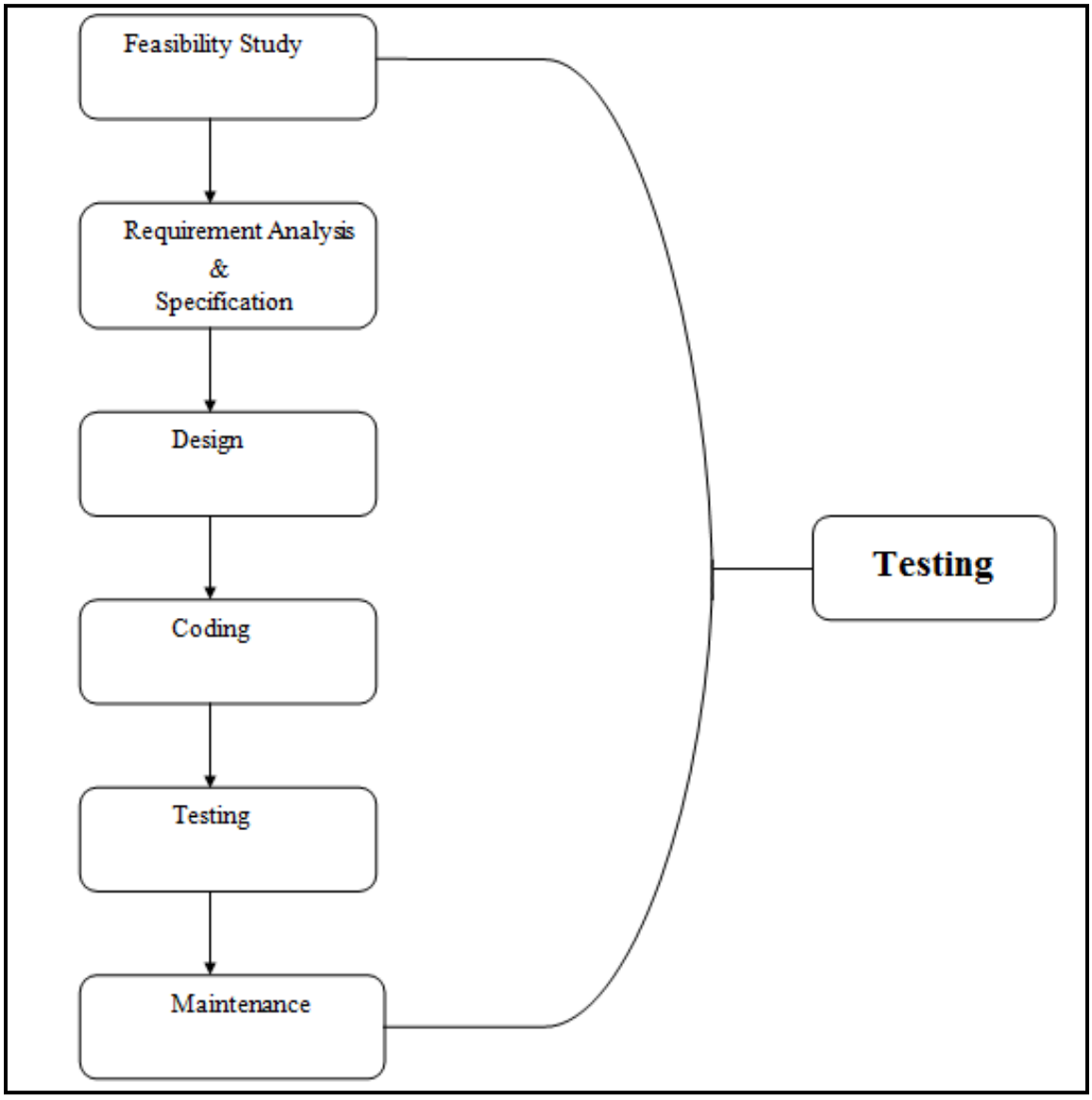


Figure 4.1 Testing Apply during SDLC phases

4.2 Proposed testing technique for SDLC phases

This is the proposed testing technique with respect to phases of SDLC. There is a web application named as *"Online Movie Store Rent/Buy and Feedback"* has been taken for applying the proposed testing technique.

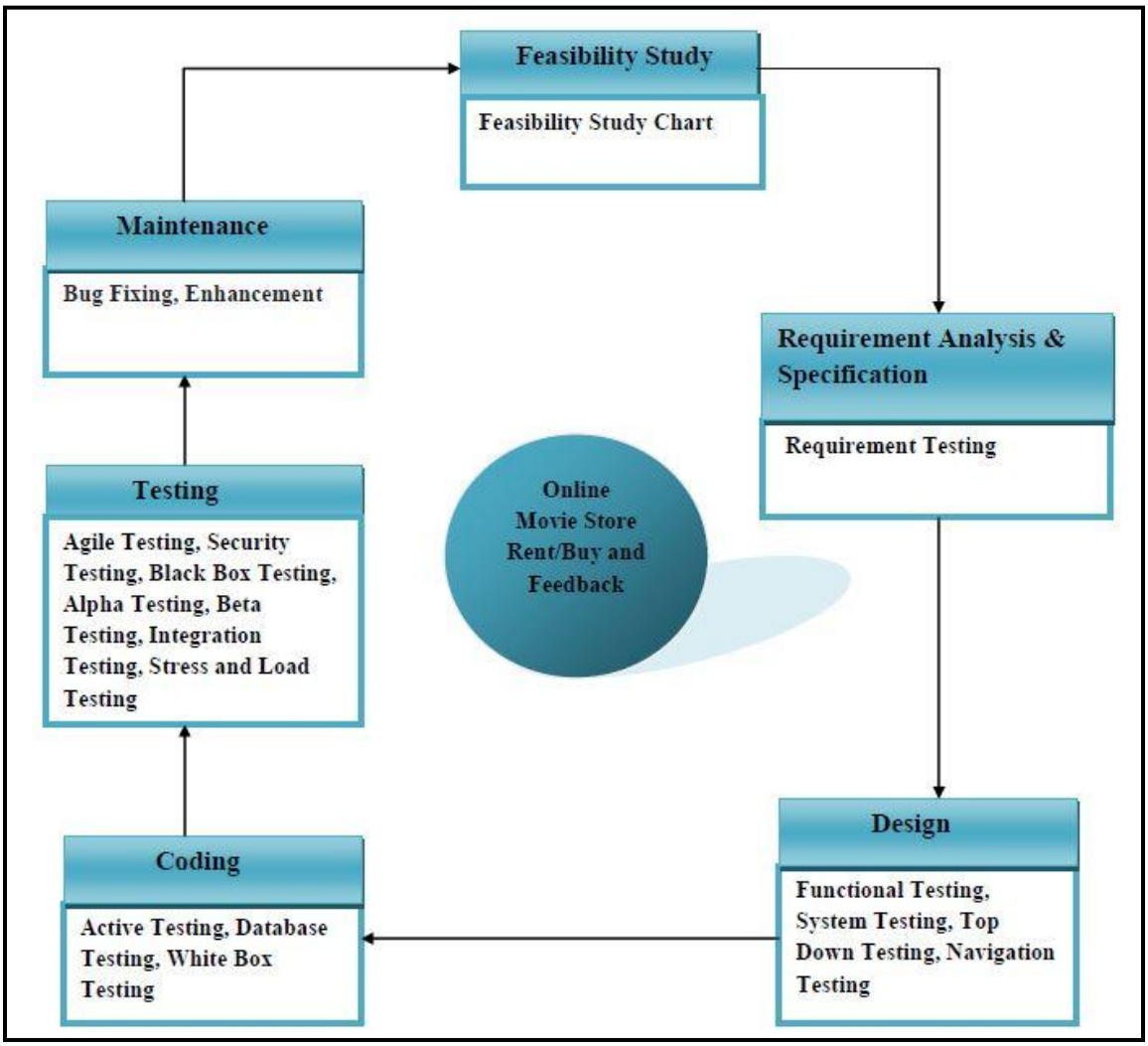


Figure 4.2 Testing applied on SDLC phases

4.3 Screen shots of web application

1. Home Page: Home page is the first page of the web application. This page displays the different categories of movies such as Action, Drama, Horror etc. There are some tabs are on home page such as Feedback, Login etc.

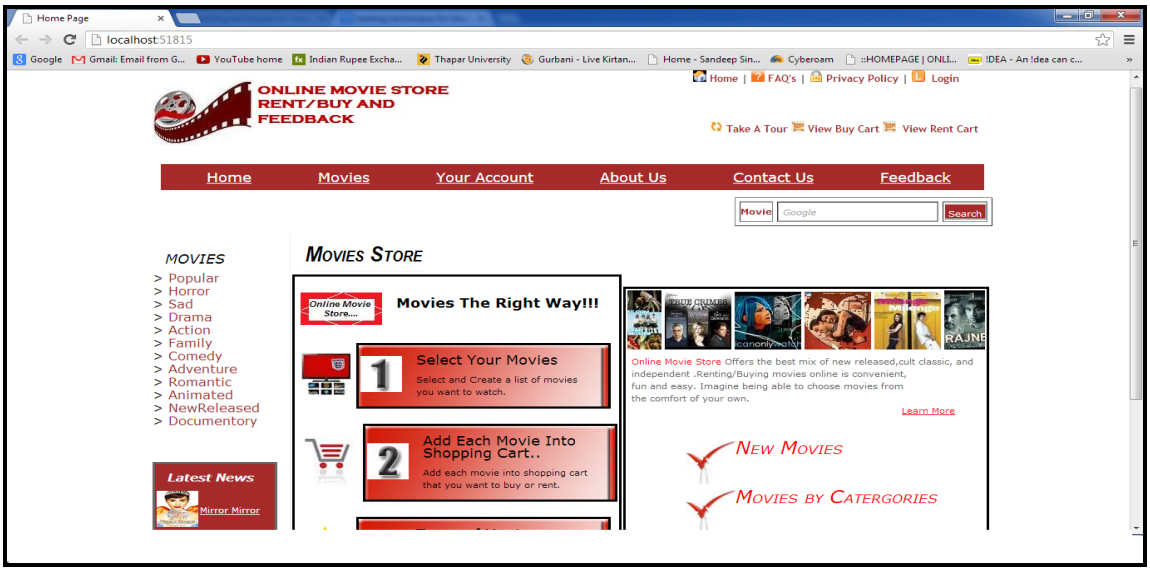


Figure 4.3 Home Page Screen

2. All Movies: This page contains the details about movies such as description about the movies, name of the movies, image, view and a buy or rent button are also displayed on this page so that user can easily buy or rent movies from this page.

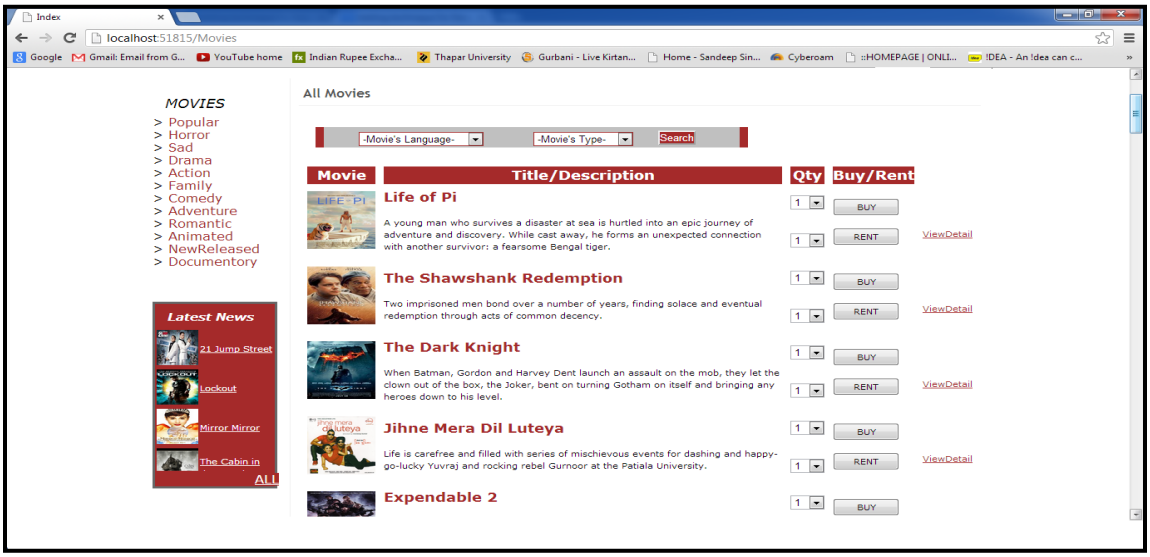


Figure 4.4 All Movies Screen

3. Register new user: Register page contains the fields that are necessary for the registration for the user such as user name, email, and password. User name should be different from the already exists user names and password should be minimum of six characters. A valid email should be required, otherwise the application will give errors.

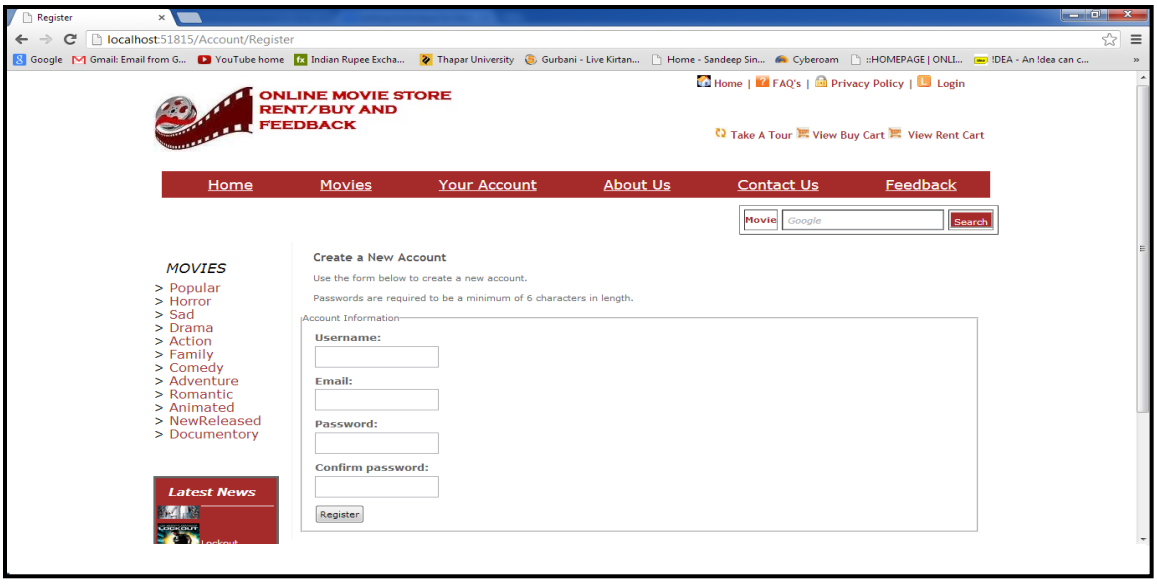


Figure 4.5 Registration Screen

4. Logon user: A logon screen will comes when admin wants to change in the web application and when a registered user wants to give feedback for the movies.

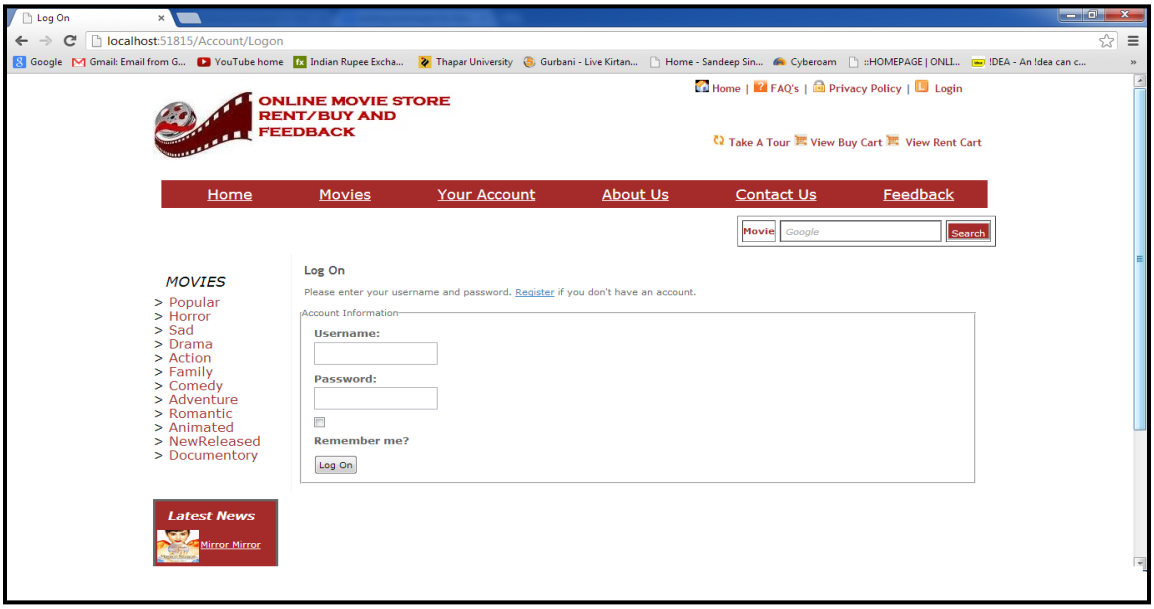


Figure 4.6 Logon Screen

5. Add New Movies: This page is an admin page in which admin add the new movies, first he has to login in the web application after that he can add new movies. On this page there are some fields about the movies such as name of the movie, actor name, releasing date, price etc.

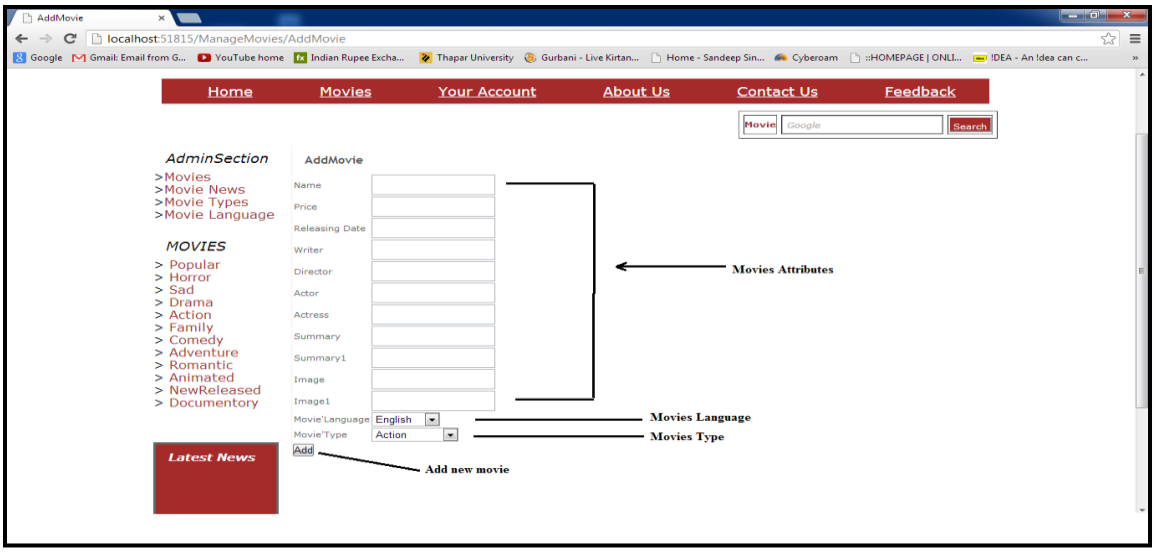


Figure 4.7 Add New Movie Screen

6. Edit and Delete Movies: On this page admin edit and delete the movies as per the requirements. This page also contains the details about the movies.

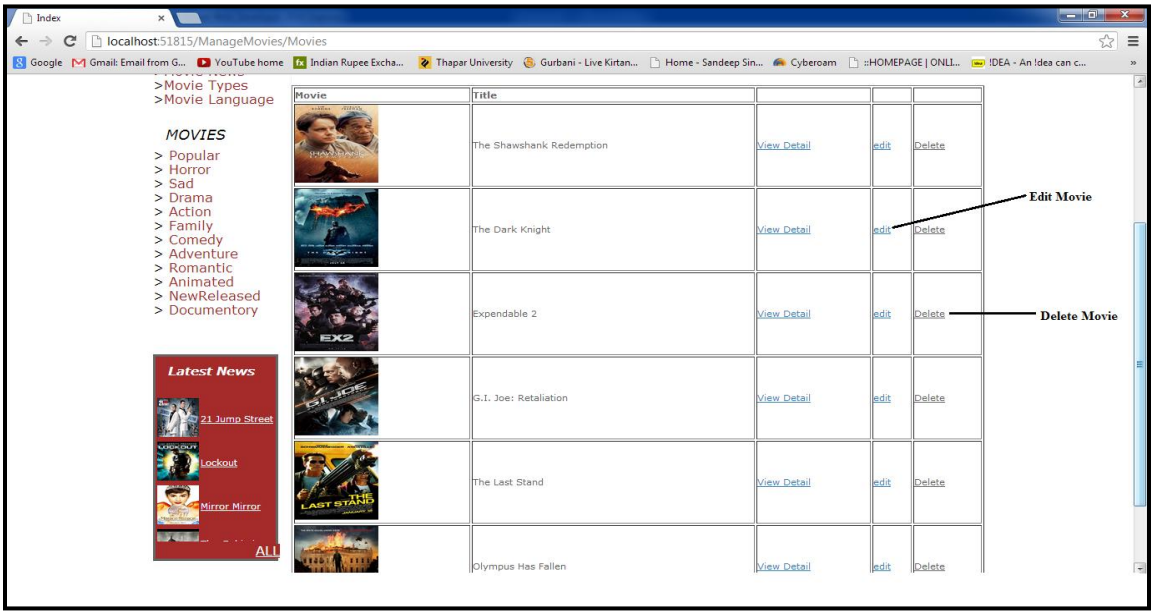


Figure 4.8 Edit and Delete Movies Screen

7. Update Movie: On this page admin can update the movies details such as he can change the image of the movie, summary of the movie and he can change the price of the movie.

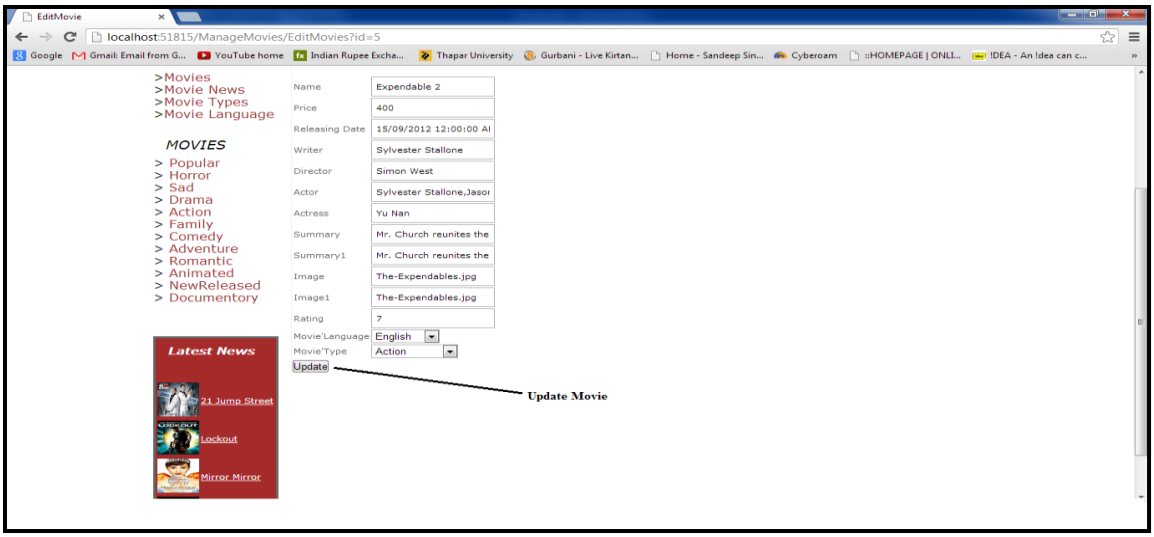


Figure 4.9 Update Movie Screen

8. Feedback Page: This is the page where registered user gives their feedback to different-different movies. There are some options for feedback such as how was the Story of the movie, Performance by the actors, Music, overall rating etc. With these options user can give their feedback for a particular movie.

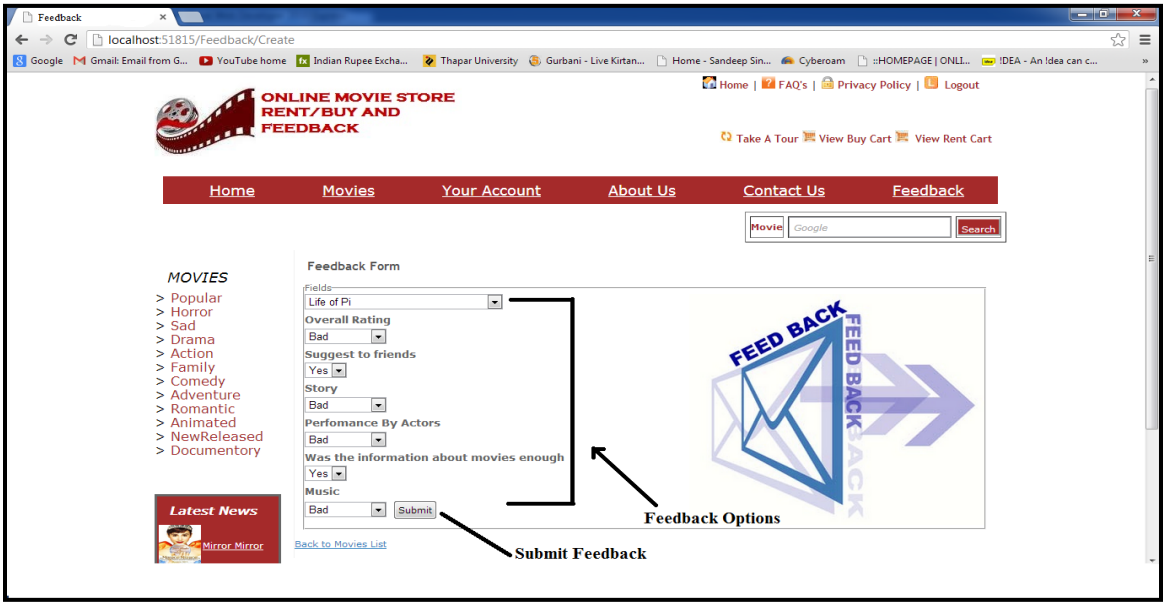
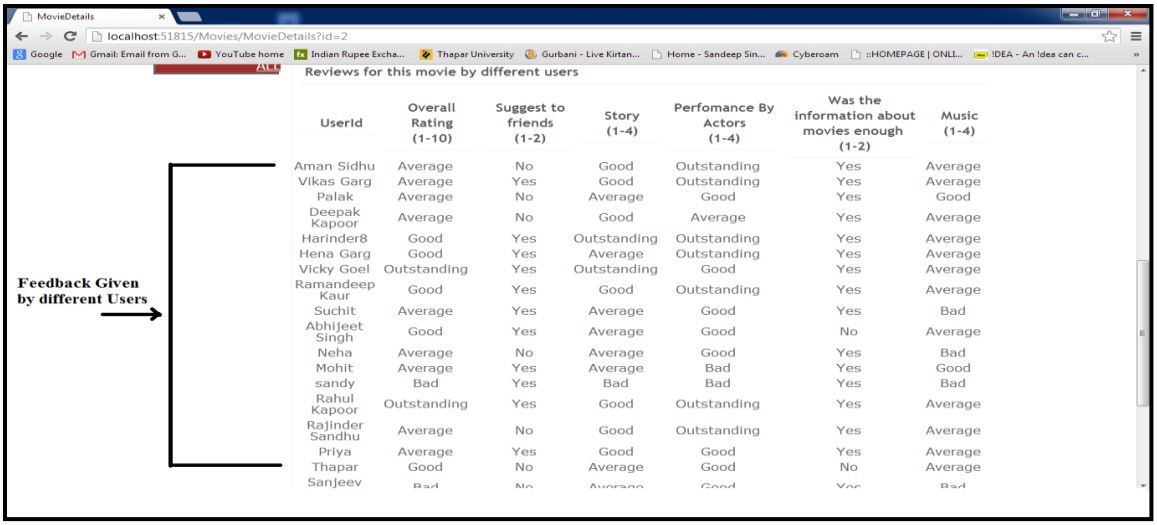


Figure 4.10 Feedback Form Screen

9. Given Feedback: This is the page on which feedback are given by the different users for a particular movie. This page shows that which user gives the feedback for this

particular movie. On this page all user names are displayed who give their feedback for this movie.



UserId	Overall Rating (1-10)	Suggest to friends (1-2)	Story (1-4)	Perfomance By Actors (1-4)	Was the information about movies enough (1-2)	Music (1-4)
Aman Sidhu	Average	No	Good	Outstanding	Yes	Average
Vikas Garg	Average	Yes	Good	Outstanding	Yes	Average
Palak	Average	No	Average	Good	Yes	Good
Deepak Kapoor	Average	No	Good	Average	Yes	Average
Harinder8	Good	Yes	Outstanding	Outstanding	Yes	Average
Hena Garg	Good	Yes	Average	Outstanding	Yes	Average
Vicky Goel	Outstanding	Yes	Outstanding	Good	Yes	Average
Ramandeep Kaur	Good	Yes	Good	Outstanding	Yes	Average
Suchit	Average	Yes	Average	Good	Yes	Bad
Abhijeet Singh	Good	Yes	Average	Good	No	Average
Neha	Average	No	Average	Good	Yes	Bad
Mohit	Average	Yes	Average	Bad	Yes	Good
sandy	Bad	Yes	Bad	Bad	Yes	Bad
Rahul Kapoor	Outstanding	Yes	Good	Outstanding	Yes	Average
Rajinder Sandhu	Average	No	Good	Outstanding	Yes	Average
Priya	Average	Yes	Good	Good	Yes	Average
Thapar	Good	No	Average	Good	No	Average
Sanjeev	Bad	No	Average	Good	Yes	Bad

Figure 4.11 Given feedback Screen

4.4 Software testing techniques that are applied on web application with test cases

There are some sorts of software testing techniques are available that are mostly applied on software. Here are some types of software testing that applied on a web application named as Online Movie Feedback software application.

1. Functional Testing
2. Security Testing
3. Black Box Testing
4. Alpha Testing
5. Beta Testing
6. Database Testing
7. Top Down Testing
8. Performance Testing
9. White Box Testing
10. Navigation Testing
11. Integration Testing
12. Acceptance Testing

13. Feasibility Study Chart

1. **Functional Testing:** Functional testing verifies that each function of the software application operates in conformance with the requirement specification. This testing mainly involves black box testing and it is not concerned about the source code of the application.

Table 4.1 Functional Testing

Sr. No	Task	Input	Expected Output	Actual Output	Result
1.	Search using Google	Enter any movie name	Should display the details about movie	Display the details about movie	Pass
		Enter any movie name	Should display the details about movie	Details not shown about entered movie	Fail
2.	Click on tabs that are on home page	Click on any tab	Go to the clicked page	Go to the clicked page	Pass
		Click on any tab	Go to the clicked page	Clicked page not shown	Fail
3.	Adding, Deleting, Updating movies	Do any operation i.e. adding, deleting	Operation successful	Operation successful	Pass
		Do any operation i.e. adding, deleting	Operation successful	Operation not successfully done	Fail
4.	Click on latest news listed on home page	Click on that	Display news about the movies	Display news about the movies	Pass
		Click on that	Display news about the movies	News not displayed	Fail
5.	Click on movies types that are categorised on home page	Click on any type of movies i.e. drama, action, comedy	Show the movies details as clicked	Shows the movies details as per clicked	Pass
		Click on any type of movies i.e. drama, action, comedy	Show the movies details as clicked	Details did not shown	Fail
6.	Click on other links such as	Click on that links	Display the desired	Display the desired	Pass

	FAQ, Privacy Policy, Take a tour		information	information	
		Click on that links	Display the information	Information not shown	Fail

2. Security Testing: Security Testing tests the ability of the system/software to prevent unauthorized access to the resources and data. There are some security concepts that need to be covered by security testing like authentication and authorization.

Table 4.2 Security Testing

Sr. No	Task	Input	Expected Output	Actual Output	Result
1.	Wrong user name/password	Wrong User name/password	Error message	Error message	Pass
		Wrong User name/password	Error message	Login successful	Fail
2.	Name already exits while registering new user	Enter any name	Error message that name already exists	Error message that name already exists	Pass
		Enter any name	Error message that name already exists	Register with already existed name	Fail
3.	Proper format of email while new user registration	Enter email	Email accepted with desired format	Email accepted	Pass
		Enter email	Email without desired format	Email accepted	Fail
4.	Password should have minimum six characters	Enter password with minimum six characters	Password should be accepted	Password accepted	Pass
		Enter password with less than six characters	Password should not be accepted	Password accepted	Fail
5.	Conform password while new user registration	Enter password	Both passwords should match	Passwords matched	Pass
		Enter password	Both passwords should be matched	Password did not match	Fail

6.	Click on view buy cart/rent cart without login then redirect to login page	Click on that	Redirect to login page	Redirect to login page	Pass
		Click on that	Redirect to login page	Display shopping cart page/ Login successfully	Fail
7.	Redirect to login page while click on Your Account tab	Click on tab your account	Redirect to login page	Redirect to login page	Pass
		Click on tab your account	Redirect to login page	Display shipping/billing page/Login successfully	Fail

3. Black Box Testing: Black Box testing doesn't require knowledge of internal code/structure of the system/software. This kind of testing done with just put the input and gets the output nothing else.

Table 4.3 Black Box Testing

Sr. No	Task	Input	Expected Output	Actual Output	Result
1.	Search movies by entering language and their types	Enter movie type and language	Show movies as per their types	Show movies as per their types	Pass
		Enter movie type and language	Show movies as per their types	Movies not shown	Fail
2.	Search using Google	Enter any movie name	Search result should be from Google	Search result from Google	Pass
		Enter any movie name	Search result should be from Google	Result not shown	Fail
3.	Click on latest news and get movie details	Click on that link	Should display details about the movies	Displayed details about the movies	Pass
		Click on that link	Should display details about the movies	Did not display details about the movies	Fail
4.	Click on	Click on these	Movies should be	Movies added to	Pass

	rent/buy buttons	buttons	added to shopping cart	shopping cart	
		Click on these buttons	Movies should be added to shopping cart	Movies not added	Fail

4. Alpha Testing: Alpha Testing is done to ensure confidence in the product or for internal acceptance testing. Alpha testing is done at the developer's site by independent test team. Alpha Testing is mostly done for COTS (Commercial Off the Shelf) software to ensure internal acceptance before moving the software for beta testing and also ensure application is as per the user requirements.

Table 4.4 Alpha Testing

Sr. No	Task	Input	Expected Output	Actual Output	Result
1.	Click on all the links and tabs that are in application	Click on those links	Functionally should be working	Functionally working	Pass
		Click on those links	Functionally should be working	If any link does not working well	Fail
2.	Entering user name, password and email new user as well as registered user	Enter the details	Operation should be complete	Operation omplete	Pass
3.	Adding new movies	Add movies by entering details	Adding movies should be successful	Adding movies successful	Pass
4.	Editing old movies	Editing movies by enter new details about the movies	Editing should be successful	Editing done	Pass
5.	Delete old movies	Delete old movies that are not visited from long time	Deleting should be successful	Deleting done	Pass
6.	Login/logout	Login by entering user details and logout	Login and logout should be successful	Login and logout done	Pass

5. Beta Testing: Beta Testing is done after alpha testing. Testing done by the existing users, customers and end users at the external site without developers involvement is

known as beta testing. User has to ensure about the application that this is it what actually needed.

Table 4.5 Beta Testing

Sr. No	Task	Input	Expected Output	Actual Output	Result
1.	Entering login details	Enter valid user name & password	Login successfully	Login successful	Pass
		Enter valid user name & password	Login successfully	Login failed	Fail
2.	Entering registering details	Enter required details	Register new user	New user registered	Pass
		Enter required details	Register new user	Registration fail	Fail
3.	Ensure all links are working well that are categorised on home page	Click on every link	Functionally working well	Functionally working well	Pass
4.	Fill the account details	Enter required account details	Details accepted	Accepts details	Pass

6. Database Testing: Database testing involves the tests to check the exact values which have been retrieved from the database by the web or desktop application. Data should be matched correctly as per the records are stored in the database.

Table 4.6 Database Testing

Sr. No	Task	Input	Expected Output	Actual Output	Result
1.	Check all queries that are integrated with other queries	Check all queries using break point	Working well	Working well	Pass
		Check all queries using break point	Working well	If any query does not work well	Fail
2.	Check all fields in database tables	Check name, data type etc	As per the rules	As per the rules	Pass
3.	Check the min/max values of all	Check their values are according to rules	Values are as per rules	Values are as per rules	Pass

	the fields				
4.	Ensure entry of names, address etc are correct in the database	Check names etc	Names are as per specified type	Names are in specified type	Pass
		Check names etc	Names are as per specified type	Names are not in given format	Fail
5.	Retrieval of data from database tables correctly	Retrieve some data like details of movies	Get the details about movies	Gets the details about movies	Pass

7. Top Down Testing: In top-down testing technique, high level modules are integrated and tested first. Testing then continues hierarchically to the bottom level with the help of testing strategy.

Table 4.7 Top Down Testing

Sr. No	**Task**	**Input**	**Expected Output**	**Actual Output**	**Result**
1.	Start with UI and ensure overall looking of an application is matching with the expected result	Check overall looking of an application	User Interface(UI) should be matched with the desire output	Matched with desired output	Pass
		Check overall looking of an application	UI should be matched with the desire output	Overall looking not matching with desired output	Fail
2.	Database tables and their fields	Check database entries	All entries should be as per rules	All entries as per rules	Pass
3.	Ensure integration of all modules with other are well	Check the integration of modules	Integration are well functioned	Integration are good	Pass
		Check the integration of modules	Integration are well functioned	Integration does not work well	Fail
4.	All the classes of an application are working well	Check classes of the entire application	Should working well	Working well	Pass

5.	All the functions of an application are working well	Check all functions	Should working well	Working well	Pass

8. Performance Testing: Performance Testing is done to determine the software characteristics like response time at which the system/software operates. The purpose of doing performance testing is to ensure that the software meets the specified performance criteria, and figure out which part of the software is causing the software performance go down.

Table 4.8 Performance Testing

Sr. No	Task	Input	Expected Output	Actual Output	Result
1.	Adding new movies and get the details about the movies	Add some new movies	Adding successfully and get the details about the movies	Adding successful and get the details about the movies	Pass
2.	Registration of new user	Enter the required details for registration	Successful registration	Successful registration	Pass
		Enter the required details for registration	Successful registration	Registration failed	Fail
3.	Login/logout	Click on login and logout	Successful login and logout	Successful login and logout	Pass
		Click on login and logout	Successful login and logout	Failed login and logout	Fail

9. White Box Testing: This testing is a test case design method that uses the control structure of the procedural design to derive test cases. Using white-box testing methods, the software engineer can derive test cases that:

a. Guarantee that all independent paths within a module have been exercise at least once.
b. Exercise all logical decisions on their true and false sides.
c. Execute all loops at their boundaries and within their operational, and
d. Exercise internal data structures to ensure their validity.

Table 4.9 White Box Testing

Sr. No	Task	Input	Expected Output	Actual Output	Result
1.	Ensure all the functions are executing as per the requirements with the help of break points	Click on all links and tabs	Working should be well	Working well	Pass
		Click on all links and tabs	Working should be well	Failed links and tabs	Fail
2.	Check database queries	Check all database queries	Should be working well	Working well	Pass

10. Navigation Testing: Navigation testing is carried out to ensure that the user can move around the application in the manner intended. It is important to check all the links and ensure that the user is going to the right place.

By testing the navigational characteristics of the application check the following:

i. Do all links within the application work?
ii. Do all hyperlinks function correctly?
iii. Are there any broken links/hyperlinks?
iv. Is there a smooth transition between screens?

Table 4.10 Navigation Testing

Sr. No	Task	Input	Expected Output	Actual Output	Result
1.	Ensure user is going to desired page.	Click on all the tabs & hyperlinks from different pages	User must go on desired page or desired hyperlink should be open	User is going on desired page	Pass
		Click on all the tabs & hyperlinks from different pages	User must go on desired page or desired hyperlink should be open	User is not getting the desired location	Fail
2.	Smoothly navigation	Click on all hyperlinks and check that all hyperlinks are	Getting less time to open and working very smoothly	Getting less time to open and working very smoothly	Pass

		working smoothly and getting less time to open.			
		Click on all hyperlinks and check that all hyperlinks are working smoothly and getting less time to open.	Getting less time to open and working very smoothly	They getting more time to open and hanging	Fail

11. Integration Testing: Before we begin Integration Testing it is important that all the components have been successfully tested independently. In its simplest form, two units that have already been tested are combined into a component and the interface between them is tested. A component, in this sense, refers to an integrated aggregate of more than one unit.

In this application, there are some new components that integrate with old components are:

I. Login component
II. Password validation component
III. Email validation component

Table 4.11 Integration Testing

Sr. No	Task	Input	Expected Output	Actual Output	Result
1.	Ensure all the old components work properly with new components	Check all old modules that are integrated with other	Integration should be well	Integration is working well	Pass
2.	Enter values in new components and check that it is as per the requirements	Check new components	New components should be working well	They are working well	Pass

12. Acceptance Testing: Acceptance Testing Services validate end-to-end business process, system transactions and user access, confirms the system or application is

functionally fit for use and behaves as expected. Also, identifies areas where user needs are not included in the system or the needs are incorrectly specified or interpreted in the system.

Table 4.12 Acceptance Testing

Sr. No	Task	Input	Expected Output	Actual Output	Result
1.	All the links and tabs are working correctly as per the expectations	Click on all links and tabs	Should be working well	All links and tabs are working well	Pass
		Click on all links and tabs	Should be working well	Links and tabs are working correctly	Fail
2.	Login and register new user	Enter desire details	Successful login and registration	Login and registration successful	Pass
		Enter desire details	Successful Login and registration	Failed while login and registration	Fail
3.	Search using Google	Enter any movie name	Get the details about the movie	Gets the details about the movie	Pass

13. Feasibility Study Chart:

Table 4.13 Feasibility Study Chart

Sr. No	Requirements	Operational Feasibility	Technical Feasibility
1.	Search movie using movie language and type	✓	✓
2.	Manage movies i.e. add new movies, update, delete movies, edit etc	✓	✓
3.	Login/Register	✓	✓
4.	Can one user give feedback for same movie	✗	✗
5.	Search using Google	✓	✓

6.	Contact us through phone	✓	✓
7.	Contact us through Email	✓	✗
8.	Online pay	✗	✗
9.	Frequently asked questions	✓	✓
10.	Get details about the movies i.e. releasing date, actor name, director name etc	✓	✓

4.4 Summary of all types of testing

Although these above types of testing techniques will be applied on web application with the suitable test cases but there are some other types of testing techniques will also applied on web application. These testing techniques are also for increase the quality of the web application.

Table 4.14 Testing Summary

Sr.No	Testing Types	Apply
1.	White Box Testing	✓
2.	Black Box Testing	✓
3.	Functional Testing	✓
4.	Alpha Testing	✓
5.	Beta Testing	✓
6.	Security Testing	✓
7.	Exhaustive Testing	✗
8.	Database Testing	✓
9.	Performance Testing	✓
10.	Top Down Testing	✓
11.	Content Testing	✓
12.	Navigation Testing	✓
13.	Accessibility Testing	✗
14.	Agile Testing	✓

15.	Stress and Load Testing	✓
16.	Boundary value analysis	✓
17.	Automated Testing	✗
18.	Comparison Testing	✗
19.	Compatibility Testing	✓
20	Integration Testing	✓
21.	Acceptance Testing	✓
22.	Component Testing	✓
23.	Age Testing	✗
24.	Vulnerability Testing	✓
25.	User Interface Testing	✓
26.	Upgrade Testing	✓
27.	System Testing	✓

4.5 Feedback Table Testing

Table 4.15 Feedback Table Testing

Sr. No	Task	Input	Expected Output	Actual Output	Result
1.	Are all the movies in feedback table	Check all the movies in table	All movies should be in table	All movies are in table	Pass
		Check all the movies in table	All movies should be in table	Movies are not in table	Fail
2.	Check all feedback options have their proper values	Check all options one by one	All options have their proper values	All options have their proper values	Pass
		Check all options one by one	All options have their proper values	They don't have their proper values	Fail
3.	Submit the feedback	Click on submit query button	A thanks message should be displayed	A thanks message is displayed	Pass
		Click on submit query button	A thanks message should be displayed	Thanks message did not display	Fail

4.	Feedback to the same movie by same user is not allowed	Give feedback to same movie that you have already given	A error message should be displayed	A error message displayed	Pass
		Give feedback to same movie that you have already given	A error message should be displayed	Again feedback submitted by same user	Fail
5.	You have to login first for feedback	Click on feedback link without login	Redirect to login page	Redirect to login page	Pass
		Click on feedback link without login	Redirect to login page	Login Successful	Fail
6.	A link for go to back movies lists	Click on that link	That page should be displayed	Page displayed	Pass
		Click on that link	That page should be displayed	Page not displayed	Fail
7.	Check all reviews after submitting reviews by different-2 users	Go to any page such as action, comedy, adventure etc	Reviews should be displayed as per given by users	Reviews are as per given by users	Pass
		Go to any page such as action, comedy, adventure etc	Reviews should be displayed as per given by users	Reviews are not in that form	Fail

Chapter 5
Experimental Results

This chapter covers the experimental results of the work done for thesis. There are two applications have been used. One is generic *"Feedback Form Application"* and second is *"Online Movie Store Rent/Buy and Feedback".* The generic feedback application is used to develop movies feedback application. This generic feedback application can be reused to develop any feedback system.

5.1 Generic Feedback Application

In generic feedback application, there are some generic categories considered such as Amul, JW Marriott Hotel, Columbia Asia Hospital, and Movies etc. On the basis of those categories there are some options as: Overall Rating which can be Bad/Average/Good /Outstanding, Quality of Goods which can be Bad/Average/Good/Outstanding, Services which can be Bad/Average/Good/Outstanding, Satisfactory which can be Yes/No and at last there is option that is Suggest to friend which can be Yes/No. The user can select any category from the given options. For example, he can select Amul, JW Marriott Hotel, Columbia Asia Hospital, and Movies from the categories and on the basis of that selected categories that particular options will be displayed on the screen. These options are generic options for the generic feedback form as shown in Figure 5.1

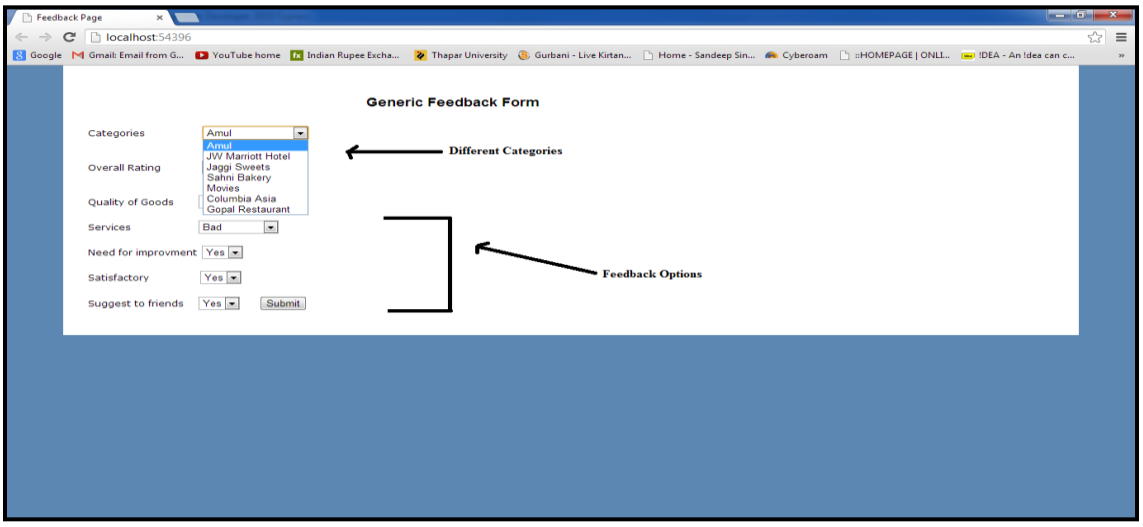


Figure 5.1 Generic Feedback Form

Table 5.1 Generic feedback application options

Categories	Overall Rating	Quality of Goods	Services	Satisfactory	Suggest to Friend
JW Marriott	Bad/Average/ Good/Outstanding	Bad/Average/ Good/Outstanding	Bad/Average/ Good/Outstanding	Yes/No	Yes/No
The Oberoi	Bad/Average/ Good/Outstanding	Bad/Average/ Good/Outstanding	Bad/Average/ Good/Outstanding	Yes/No	Yes/No
Grand Hyatt	Bad/Average/ Good/Outstanding	Bad/Average/ Good/Outstanding	Bad/Average/ Good/Outstanding	Yes/No	Yes/No
Jaggi Sweets	Bad/Average/ Good/Outstanding	Bad/Average/ Good/Outstanding	Bad/Average/ Good/Outstanding	Yes/No	Yes/No
Sahni Restaurant	Bad/Average/ Good/Outstanding	Bad/Average/ Good/Outstanding	Bad/Average/ Good/Outstanding	Yes/No	Yes/No
Gopal Restaurant	Bad/Average/ Good/Outstanding	Bad/Average/ Good/Outstanding	Bad/Average/ Good/Outstanding	Yes/No	Yes/No
Columbia Asia	Bad/Average/ Good/Outstanding	——	Bad/Average/ Good/Outstanding	Yes/No	——
PGI	Bad/Average/ Good/Outstanding	——	Bad/Average/ Good/Outstanding	Yes/No	——
Movies	Bad/Average/ Good/Outstanding	——	——	Yes/No	Yes/No
University	Bad/Average/ Good/Outstanding	——	——	Yes/No	Yes/No
Shopping Centre	Bad/Average/ Good/Outstanding	Bad/Average/ Good/Outstanding	Bad/Average/ Good/Outstanding	Yes/No	Yes/No

5.2 Online Feedback System for Movies

Online feedback system for movies helps users to give feedback for the different movies. This can help others to get the feedback about the movies. For any store who gives movies on rent or allows customers to purchase online, can surely be benefitted from positive feedback about movies. In the developed application, there are some movies

listed in the movie field. The users can select any movie from that field and give their feedback for the particular movie. Like generic feedback application, in this application there are also some options listed such as Overall Rating, Story, Performance by Actors, Music and last is Suggest to Friend.

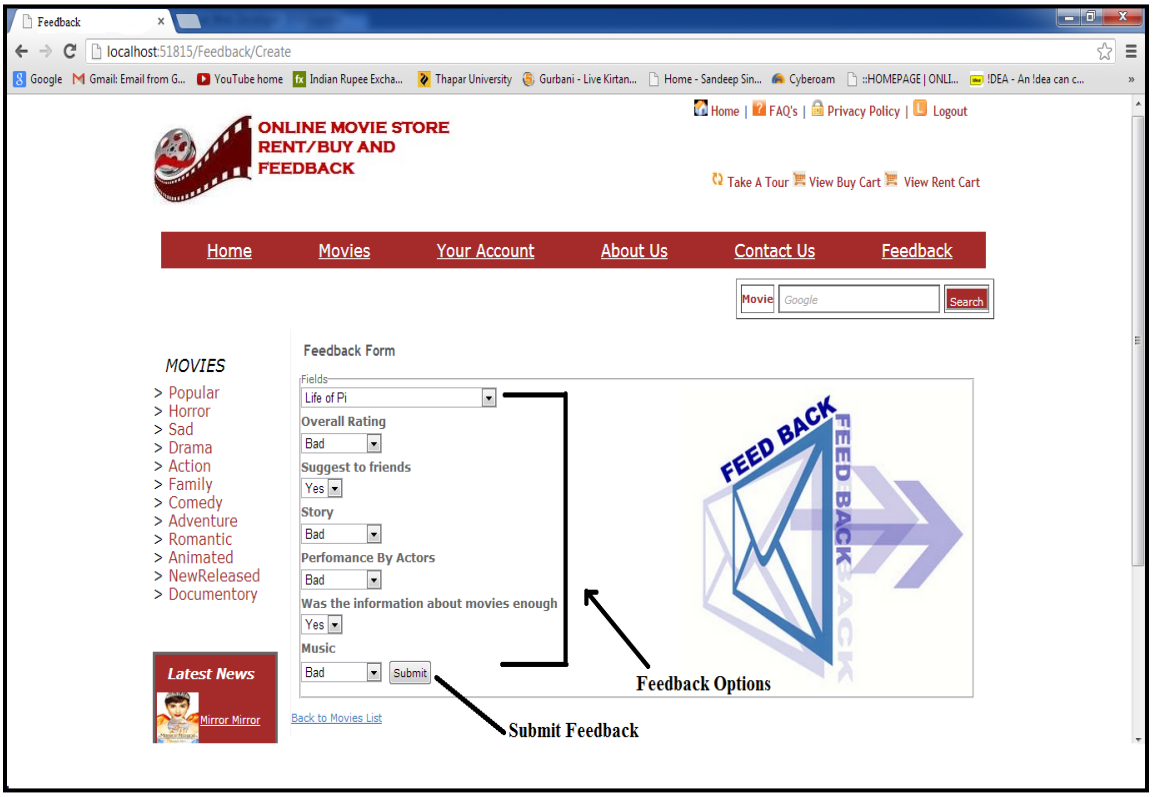


Figure 5.2 Feedback Form Screen

The given table 5.2 shows the details about the different movies according to all the options that are listed on the feedback form. Basically this table shows that how many users give feedback to particular movie as per different options. There are some movies listed here:

Table 5.2 Details of movies as per options by different users

Sr. No	Name	*CT	*US	*OR	*ST	*PF	*EI(Y)	*MU	*SF(Y)
1.	Life of Pi	Adventure	28	6	2/4	2.5/4	17/28	2/4	22/28
2.	The Shawshnk Redemptin	Sad	26	7	2/4	2/4	19/26	1/4	18/26

3.	The Dark Knight	Action	31	8.5	2.5/4	3/4	22/31	2/4	29/31
4.	Jihne Mera Dil Luteya	Comdey	23	7	3/4	3/4	15/23	3/4	19/23
5.	Expendable 2	Action	28	6	2/4	2.5/4	15/28	1.5/4	17/28
6.	Kung Fu Master	Drama	21	5.5	2/4	3/4	18/21	1/4	9/21
7.	Talaash	Drama	15	6.5	3/4	3/4	9/15	2.5/4	11/15
8.	G.I. Joe 2	Action	20	5	2/4	2.5/4	13/20	1/4	12/20
9.	Carry on Jatta	Comedy	21	8	3/4	3/4	16/21	3.5/4	19/21
10.	Lincoln	Drama	17	5.5	2/4	3.5/4	11/17	1.5/4	8/17
11.	Skyfall	Action	27	7	3/4	4/4	15/27	2/4	20/27
12.	Broken City	Drama	29	7.5	2.5/4	3/4	14/29	1.5/4	21/29
13.	The Last Stand	Adventure	31	8	3.5/4	3.5/4	16/31	2/4	25/31
14.	Stand Up Guys	Action	22	6.5	2.5/4	1.5/4	12/22	2.5/4	15/22
15.	A Good Day to Die Hard	Action	16	5	2/4	2/4	7/16	1/4	9/16
16.	Snitch	Horror	18	5.5	2.5/4	1.5/4	12/18	1/4	10/18
17.	Jack the Giant Slayer	Adventure	23	7.5	3/4	3.5/4	11/23	2.5/4	16/23
18.	Oz the Great and Powerful	Animated	28	6	2/4	2.5/4	15/28	1.5/4	13/28
19.	The Incredible Burt Wonderstone	Family	21	5.5	2/4	3/4	12/21	1/4	8/21
20.	Olympus Has Fallen	Romantic	15	6.5	3/4	3/4	7/15	2.5/4	9/15
21.	Dabangg 2	Drama	20	5	2/4	2.5/4	11/20	1/4	9/20
22.	Jab Tak Hai Jaan	Romantic	21	8	3/4	3/4	12/21	3.5/4	14/21
23.	1920: Evil Returns	Horror	17	5.5	2/4	3.5/4	13/17	1.5/4	9/17

24.	Luv Shuv Tey Chicken Khurana	Drama	33	7.5	3/4	3.5/4	20/33	2.5/4	26/33
25.	Religulous	Family	33	6.5	2.5/4	2/4	18/33	2/4	18/33
26.	This Film Is Not Yet Rated	Documentary	24	7.5	3.5/4	3.5/4	15/24	2.5/4	19/24
27.	Mama	Horror	19	5.5	2.5/4	3/4	8/19	1.5/4	11/19
28.	Control Room	Action	33	8.5	3.5/4	3/4	26/33	2.5/4	29/33
29.	Pink Saris	Sad	28	6	2/4	2.5/4	18/28	2/4	17/28
30.	Confessions of a Superhero	Documentary	26	7	2/4	2/4	10/26	1/4	21/26
31.	Arakshaka	Horror	20	6	2.5/4	3/4	9/20	2/4	13/20
32.	Happiness Never Comes Alone	Romantic	23	7	3/4	3/4	8/23	3/4	19/23
33.	Fish Story	Family	28	6	2/4	2.5/4	14/28	1.5/4	21/28
34.	Thuppakki	Sad	21	5.5	2/4	3/4	8/21	1/4	9/21
35.	Ice Age	Animated	15	6.5	3/4	3/4	9/15	2.5/4	11/15
36.	Midnight in Paris	Romantic	20	5	2/4	2.5/4	13/20	1/4	11/20
37.	Moonrise Kingdom	Documentary	21	8	3/4	3/4	10/21	3.5/4	16/21
38.	Kung Fu Panda	Animated	17	5.5	2/4	3.5/4	9/17	1.5/4	7/17
39.	A Thousand Words	Drama	21	8	3/4	3/4	19/21	3.5/4	18/21
40.	Toy Story 3	Animated	17	5.5	2/4	3.5/4	11/17	1.5/4	9/17

* **CT**- Category, **OR**- Overall Rating, **ST**- Story, **PF**- Performance by Actors, **EI (Y)**- Enough Information (Yes), **MU**- Music, **SF (Y)**- Suggest to Friend (Yes)

* **1**= Bad, **2**= Average, **3**= Good, **4**= Outstanding

The overall ratings of number of movies are shown in the figure 5.3. The number of users who have provided the feedback is also highlighted. Though there is no direct relation between number of users providing the feedback. The positive feedback increases the overall rating.

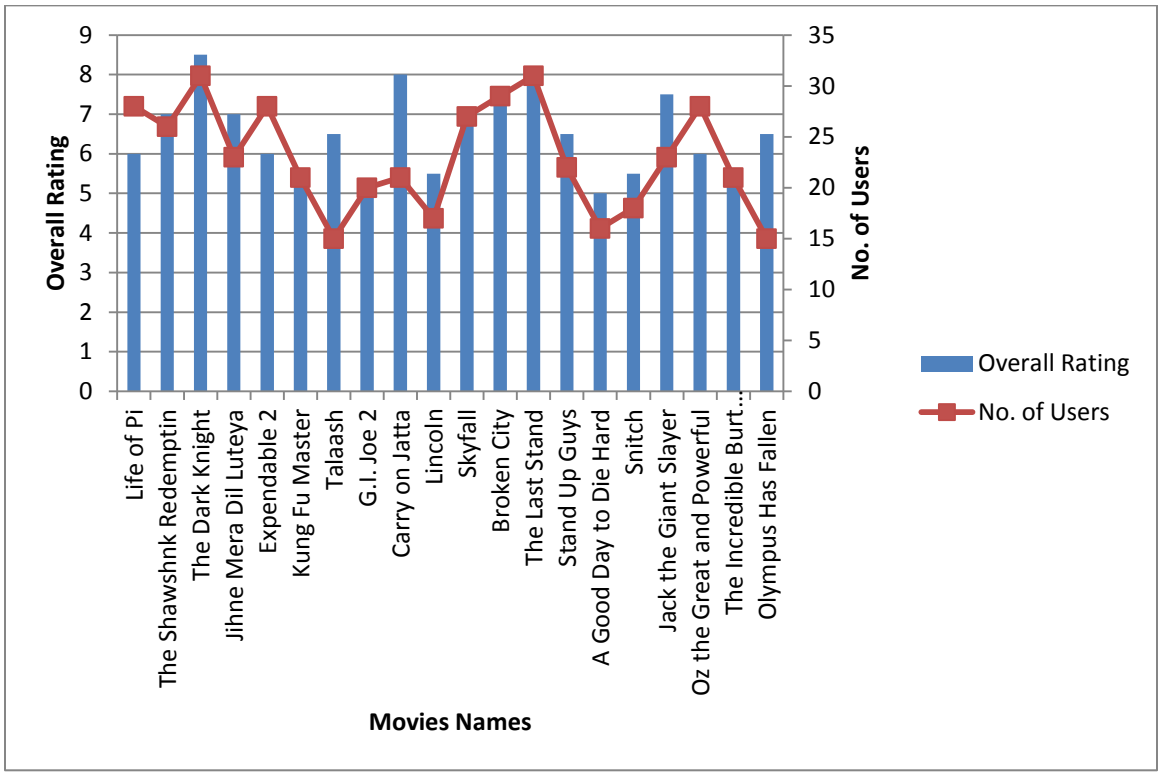


Figure 5.3 Feedback graph through some options

Chapter 6
Conclusion and Future Scope

Feedback is an important aspect in identifying the areas for improvement in any business. Online web applications provide an easy method of getting quick inputs for feedback. A generic feedback system application has been designed in this thesis. This can be used for many stores which can get feedback online and from analysis of the results, improvement can be done in a particular area. One complete application online movie feedback system has been developed and tested. Various types of testing techniques have been applied. The results of feedback have been summarized.

Presently the feedback system has options only for feedback system. The future scope is to be used it for online applications for rent or buy online. Additional information can be added for feedback of online transaction systems as well.

References

[1] Davis, Z., "Definition of: app", Internet: http://www.pcmag.com/encyclopedia/ term/37865/ app, Aug. 21, 2010 [Jan. 28, 2013].

[2] Warthen, D., "What are Computer Applications?" Internet: http://answers.ask .com/computers/Other/what are computer applications, [Dec.21. 2012].

[3] Shubpreet Kaur, "Testing Anomalies in Multiple and Multilevel Inheritance" M.E. Thesis, Thapar University, Patiala (Punjab), 2011.

[4] Jain, D., "Software Engineering Principle and Practices", First Ed. by Oxford University Press, ISBN-13:978-0-19-569484-0, 2009.

[5] Kota, K., "Testing Your Web Application", A Quick 10-Step Guide, Internet: http://www.adminitrack.com/articles/testing_web_apps.pdf, 2005 [Mar. 8, 2013].

[6] Beizer, B., "Black-Box Testing Technique for Functional Testing of software and system" New York Wiely, ISBN: 0471120944, pp. 135-136, 1995.

[7] Bucur, S. *et al.*, "Parallel symbolic execution for automated real-world software testing",In Proceedings of the sixth conference on Computer systems of ACM, pp. 183-198. April, 2011.

[8] Youddha Beer Singh, "Role of Testing in Phases of SDLC and Quality." M.E. Thesis, Thapar University, Patiala (Punjab), 2009.

[9] Joe W. Duran, Semeon, C. Ntafos, "An Evaluation of Random Testing", IEEE Transactions on Software Engineering, Vol.SE-10, No.4, pp. 438-443, July 1984.

[10] Myers, Glenford J., "The Art of Software Testing", John Wiley and Sons, ISBN 0- 471-04328-1, 1979.

[11] Watson, H. and McCabe, J., "McCabe Software", NIST Special Publication 500-235, September 1996.

[12] Whittaket, J.A., "What Is Software Testing? And Why Is It So Hard?", Florida Inst. Of Technology, Melbourne, FL,USA, IEEE SOFTWARE 0740-7459/00, 2000.

[13] IEEE "Standard Glossary of Software Engineering Terminology" IEEE Std 610.12-1990, IEEE Computer Society, Dec. 10, 1999.

[14] Kapfhammer, G., "Software Testing", The Computer Science and Engineering Handbook, CRC Press, May, 2004.

[15] Bertolino, Antonia, and Eda Marchetti. "A brief essay on software testing." Software Engineering, The Development Process. Wiley-IEEE Computer Society Press, (2005).

[16] Kumar, S., "Why is testing necessary?", Internet: http://istqbexamcertification .com/why- is- testing-necessary/, [Mar. 3, 2013].

[17] Hetzel, William C., "The Complete Guide to Software Testing", 2nd Ed. Wellesley Publication, Mass: QED Information Sciences, ISBN: 0894352423, pp. 390-392, 1998.

[18] Kumar, S., "What is Software Testing?", Internet: http://istqbexamcertification .com/what- is-a-software-testing/, [Mar. 3, 2013].

[19] Pressman, R.S., "Software Engineering: A Practitioner's Approach", 7th Ed., Tata Mcgraw Hill,

[20] Myers, Glenford J., "The Art of Software Testing", Wiley publication, ISBN: 0471043281, pp. 11-13, 2004.

[21] Gelperin, D. and Hetzel, W., "The Growth of Software Testing", CACM 31 (6): 687. doi:10.1145/62959.62965, ISSN 0001-0782, Volume 31 Issue 6, June 1988.

[22] Kumar, S., "What is test design technique?" Internet: http://istqbexamcertification .com/- what-is-test-design-technique/, [Mar. 3, 2013]

[23] Reddy, G.C., "Test Design Techniques", Internet: http://www.gcreddy.com /2012/10/test-design-techniques.html#.Uaw0h0Cl1dx, Oct. 25, 2011 [Feb. 2, 2013].

[24] Veenendaal, E.N. *et al.* "Static Techniques" in Foundations of Software Testing, 2nd Ed., Cengage Learning, Andover, UK, pp. 57-58, 2008.

[25] Gotterbarn, D., "Reducing software failures: addressing the ethical risks of the software development lifecycle", Australasian Journal of Information Systems, Vol. 9, No. 2, pp.1- 2, 2002.

[26] Gleen, H., "Difference between defect, error, bug, failure and fault" Internet:http://tfortesting.wordpress.com/2012/09/03/difference-between-defect-error- bug- failure-and-fault/, Sep. 3, 2012 [Jan. 9, 2013].

[27] Estrella, A., "Definitions and Meaning: Error, Fault, Failure and Defect", Internet: http://www.vietnamesetestingboard.org/zbxe/?document_srl=602412/, Dec. 21, 2012 [Nov. 2, 2012]

[28] Chilarege, R., "Software Testing Best Practices", IBM Research - Technical Report RC21457 Log 96856 4/26, 1999.

[29] Mili, H., Mili, A., Yacoub, S. And Addy, E., "Reuse Based Software Engineering: Techniques, Organizations and Measurement", Wiley 2002.

[30] Susan, K. Land , "Learning From Software Failure" Internet: http://spectrum.ieee .org/computing/software/learning-from-software-failure, Sep 01, 2005 [Jan. 29. 2013].

[31] McHale, J., "What is software bug?" Internet: http://curiosity.discovery .com/question/what-is-software-bug/, [Apr. 7, 2013].

[32] Chand, M., "Windows Forms Application versus Web Application", Internet: http://www.c-sharpcorner.com/blogs/416/windows-forms-application-versus webapllication.aspx, Oct. 03, 2007 [Nov. 25, 2012].

[33] "Portal: Software Testing" Internet: http://www.znu.ac.ir/members/afsharchim /lectures /Software Testing/, [Feb. 20, 2013].

[34] Kumar, S. "What is a static test technique?" Internet: http://istqbexamcertification.com/what-is-a-static-test-technique/,[Mar.20, 2013].

[35] Peena, P. "Exhaustive Testing" Internet: http://www.ianswer4u .com/2012/06/exhaustive-testing-in-software-testing.html#ixzz2VYbpdi3c/, [May. 2, 2013].

[36] Rouse, M. and Wigmore, I., "dynamic testing" Internet: http://whatis.techtarget. com/definition/dynamic- testing/, Dec, 2012 [Mar. 25, 2013].

[37] Guru, **"**100 types of Software Testing Types**"** Internet: http://www.guru99 .com/types-of-software-testing.html, Apr, 2012 [Feb. 5, 2013].

[38] Myers, G. J.,"A Controlled Experiment in Program Testing and Code Walkthroughs/Inspection", Communications of the ACM, Vol. 21, No. 9, Sept 1978.

[39] Basili, Victor R. and Richard, W. Selby, "Comparing the effectiveness of software testing strategies", IEEE Transactions on Software Engineering, Vol. 13, No. 12, ISSN-0098- 5589, pp.1278-1296, Dec. 1987.

[40] Guru, "Static Vs Dynamic Testing" Internet: http://www.guru99.com/static-dynamic-testing.html/, [Mar. 28, 2013].

[41] Pardhan T. "All-Types-Of-Software-Testing" Internet: http://www.software testingsoftware .com/all-types-of-software-testing/, July. 26, 2012 [Mar. 21, 2013].

[42] Ramdeo, A., "Accessibility Testing" Internet: http://www.testinggeek .com/accessibility-testing/, May. 5, 2011 [Apr. 25, 2013].

[43] Park, C., "A randomized dynamic program analysis technique for detecting real deadlocks", In ACM Sigplan Notices, Vol. 44, No. 6, pp. 110-120. ACM, 2009.

[44] Sen, K., “Randomized active atomicity violation detection in concurrent programs”, In 16th ACM SIGSOFT International Symposium on Foundations of software engineering, pp. 135-145, 2008.

[45] Sen, K., “Race directed random testing of concurrent programs”, In PLDI '08: Proceedings of the 2008 ACM SIGPLAN conference on Programming language design and implementation, pp. 11-21, New York, NY, USA, 2008.

[46] IEEE Standard for Software Unit Testing, IEEE Standards Board, The Institute of Electrical and Electronics Engineers, Inc 345 East 47th Street, New York, NY 10017, USA, July. 28, 1986

[47] Sulagna, “What is API testing?” Internet: http://www.scribd.com/doc/9808382/ Introduction-to-API-Testing, [May. 2, 2013].

[48] Cohen, David M., et al., “The Combinatorial Design Approach to Automatic Test Generation”, IEEE Software September 1996.

[49] Cohen, David M., *et al*. "The combinatorial design approach to automatic test generation." Software, IEEE 13.5 pp: 83-88, 1996.

[50] Chowdhary, H., “Agile Testing” Internet: http://www.codeproject.com/ Articles/230489/Agile-Testing-Best-to-Keep-Customers-Happy, Jul. 27, 2011 [Jan. 5, 2013].

[51] Qihui, Q., “What is test assertion” Internet: http://www.thoughtworks.com/ insights/articles/test-assertions, Oct. 19, 2010 [Jan. 25, 2013].

[52] Zheng, W., & Bundell, G., “Model-based Software Component Testing: A UML-based Approach”, ICIS 2007, 6th IEEE/ACIS International Conference in Computer and Information Science pp. 891-899, July 2007.

[53] Lun, L. and Xin, Chi., “Relationship between Testing Criteria for Architecture Configuration Testing Based on Wright Specification”, In Computational Intelligence and Software Engineering (CiSE), 2010 International Conference on, pp. 1- 4, 2010.

[54] Wang, C., Said, M., & Gupta, A. "Coverage Guided Systematic Concurrency Testing". In Proceedings of the 33rd International Conference on Software Engineering pp. 221- 230, May 2011.

[55] Mattila, "Compliance Testing" Internet: http://www.atis.org/glossary/definition.aspx?id= 6520 , [May. 2, 2013].

[56] Kropp, N. P., *et al*, "Automated Robustness Testing of the- Shelf Software Component". In 28th Annual International Symposium on Fault-Tolerant Computing, 1995.

[57] Jones, James A., and Mary Jean Harrold. "Test-suite reduction and prioritization for modified condition/decision coverage." Software Engineering, IEEE Transactions on 29.3 (2003): 195-209.

[58] Schertz, C. *et al*., "Efficient integration testing using dependency analysis", Microsoft Research, TechReport MSR-TR-2005-94, 2005.

[59] Moler, C., "Formal Testing" Internet: http://www.mathworks.com/discovery/formal-verification.html, [Apr. 16, 2013].

[60] Pak, Brian S., "Hybrid Fuzz Testing: Discovering Software Bugs via Fuzzing and Symbolic Execution", PhD diss., Carnegie Mellon University, Pennsylvania, United States, 2012.

[61] Abufardeh, S., & Magel, K., "Software Internationalization: Testing Methods for Bidirectional Software", In IEEE NCM'09 Fifth International Joint Conference on INC, IMS and IDC, pp. 226-231,Aug, 2009.

[62] Leitner, A. *et al.,* "Reconciling Manual and Automated Testing: The Autotest Experience", In 40th Annual Hawaii International Conference on System Sciences, pp. 261-262. January 2007.

[63] Xu, Chong-yu, "Operational testing of a water balance model for predicting climate change impacts", Agricultural and Forest Meteorology, Vol. 98, ISSN: 0168-1923, pp. 295-304, Elsevier Science, Dec, 1999.

List of Publications

1. S. Singh and S. Goel, *"CBSE versus COTS Based Software Development",* International Conference on Recent Trends in Computing, SRM University, NCR Campus, Modinagar, Ghaziabad, India, ISSN: 978-93-81583-67-8, October 04, 2012, pp. 135-138.

2. Sandeep Singh and Shivani Goel, *"CBSE versus COTS Based Software Development",* 3rd Annual International Conference on Software Engineering &Applications (SEA 2012) held at Singapore, 20 Nov, 2012. (Accepted)

3. Sandeep Singh and Shivani Goel, *"Phases of Testing An Online Application",* International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE), Volume 3, Issue 6, June 2013, pp. 508-511 (Impact Factor: 2.080).