SOFTWARE ENGINEERING ASSIGNMENT-4

TEAM-X4B

Adarsh Liju Abraham(PES2UG20CS017)

Ananya Adiga(PES2UG20CS043)

Anirudh Chakravarty K(PES2UG20CS049)

Aanchal Agarwal(PES2UG20CS005)

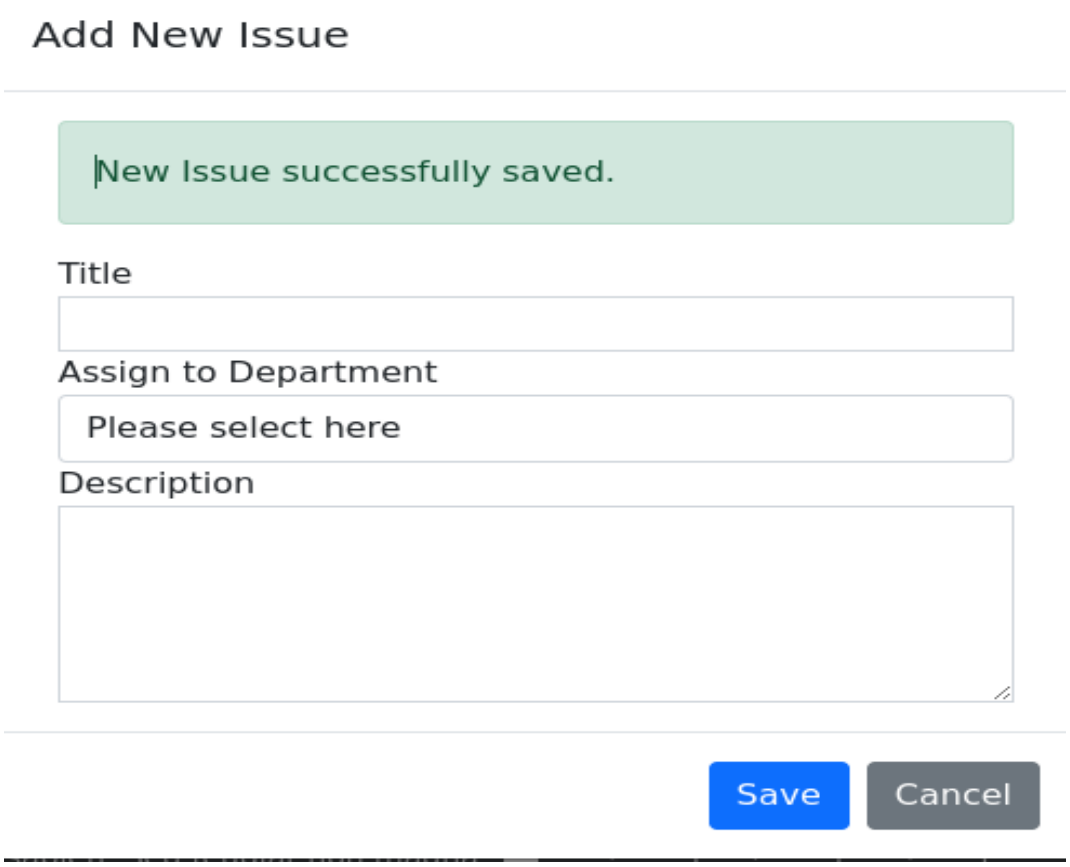
1) UNIT TESTING-

Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation. This testing methodology is done during the development process by the software developers and sometimes QA staff.

SUBMISSION OF A NEW ISSUE-

Submission success- Expected output- “submitting form…” and save issue

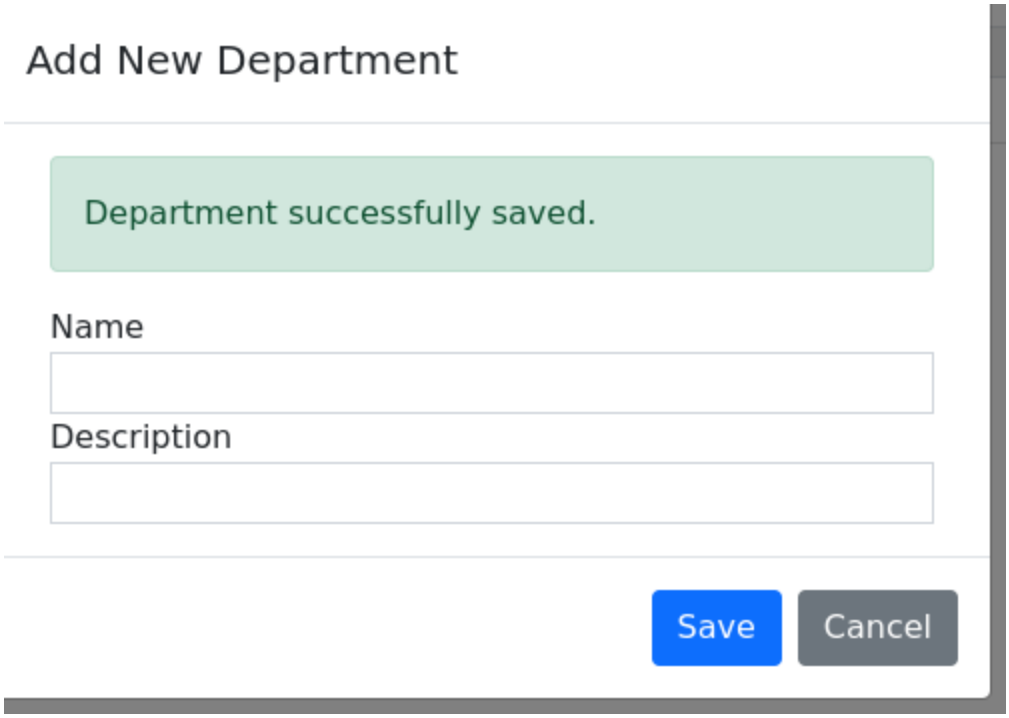
Submission failed- Expected output- “An error occurred”



SAVING DEPARTMENT INFORMATION-

Success- “department successfully saved” or “department successfully updated” and save to database

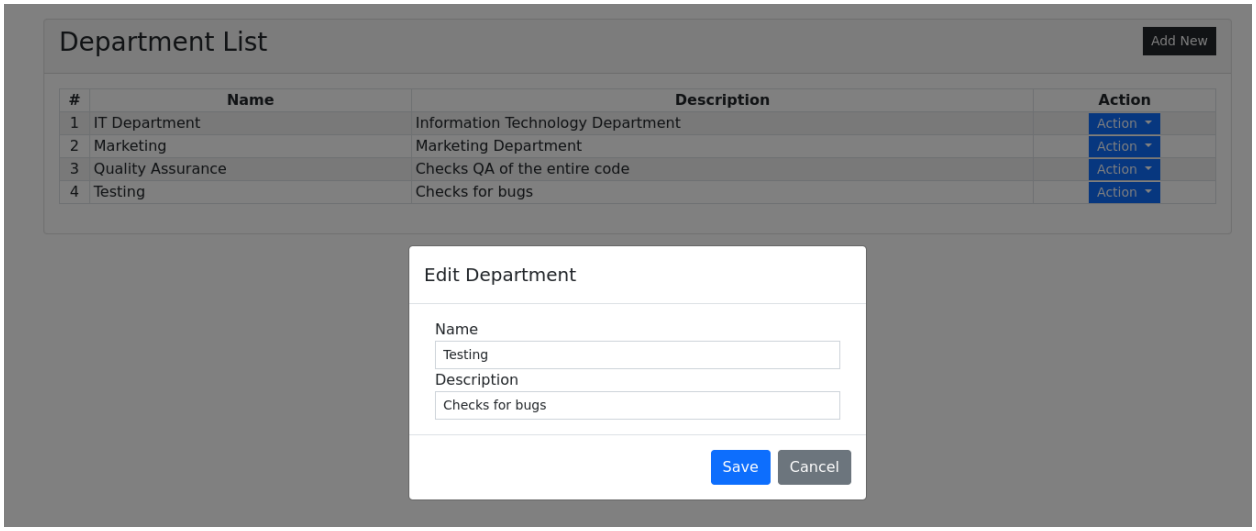
Failed- “saving new department failed” or “Updating department failed”



EDITING DEPARTMENT INFORMATION-

Delete data success- expected output- Ask for confirmation and delete data

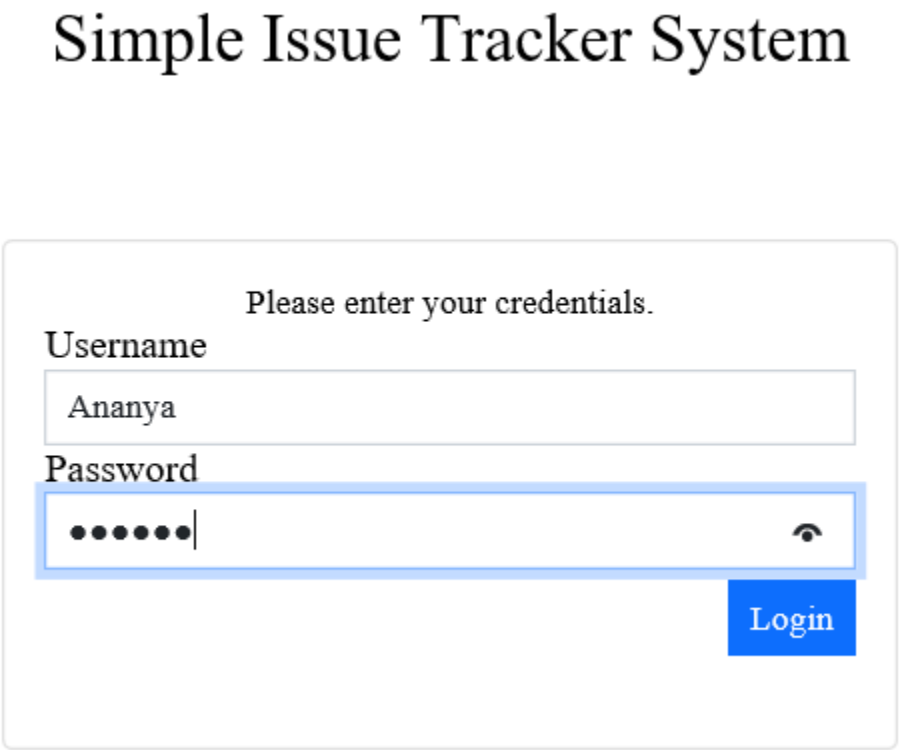
Delete data failed- expected output- “An error occurred”



LOGIN-

Login failed- expected output- “invalid username or password”

Login successful- expected output- “Login successful.”



LOGIN-

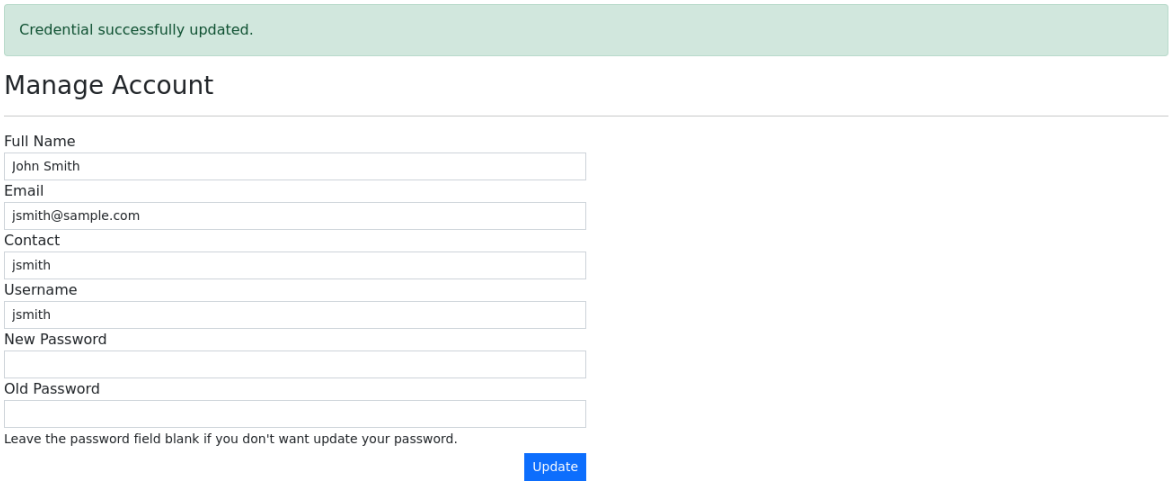
Login successful- expected output- “Logging in…”

Login failed- expected output- “An error occurred” and save to terminal

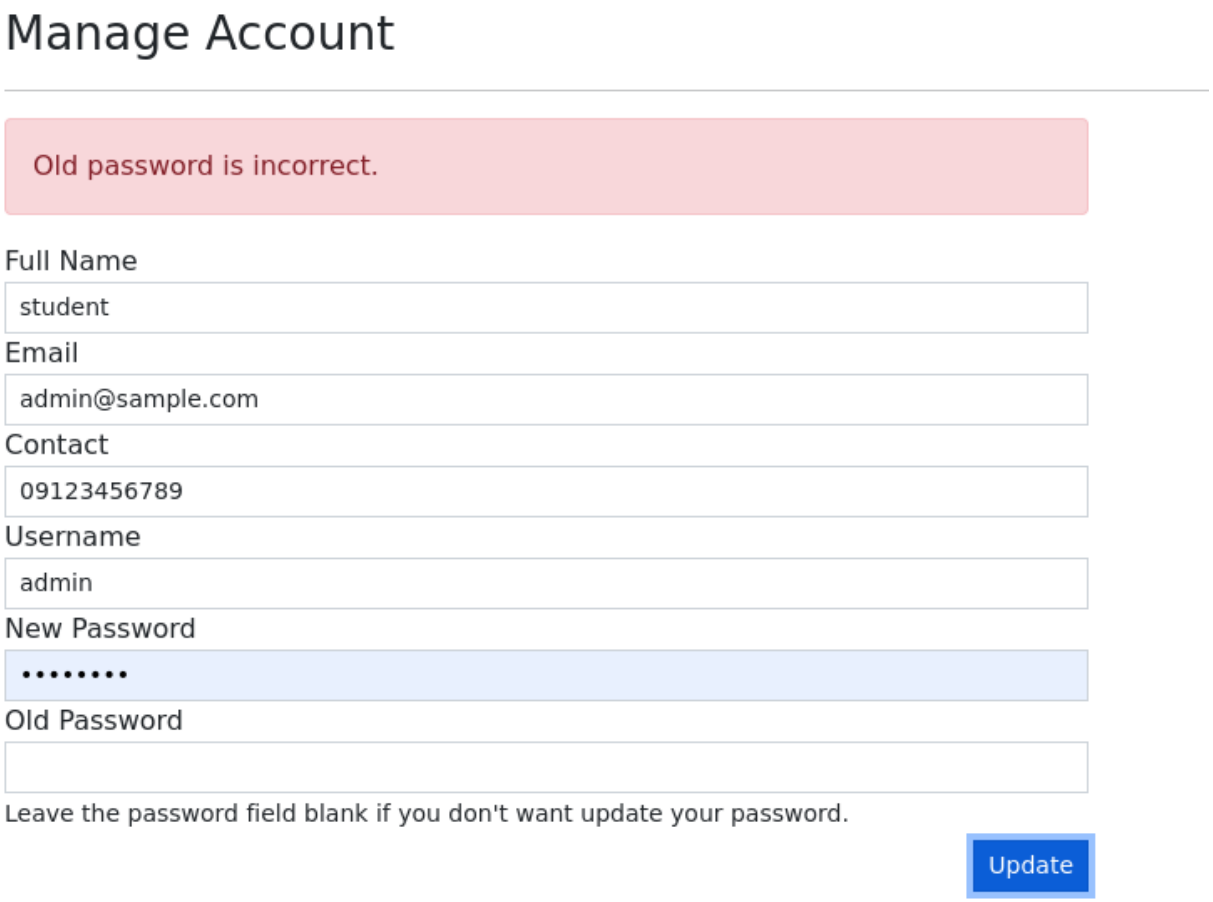
UPDATE PASSWORD-

Enter old password, if failed- “Old password is incorrect”

Update successful- “Credential successfully updated”



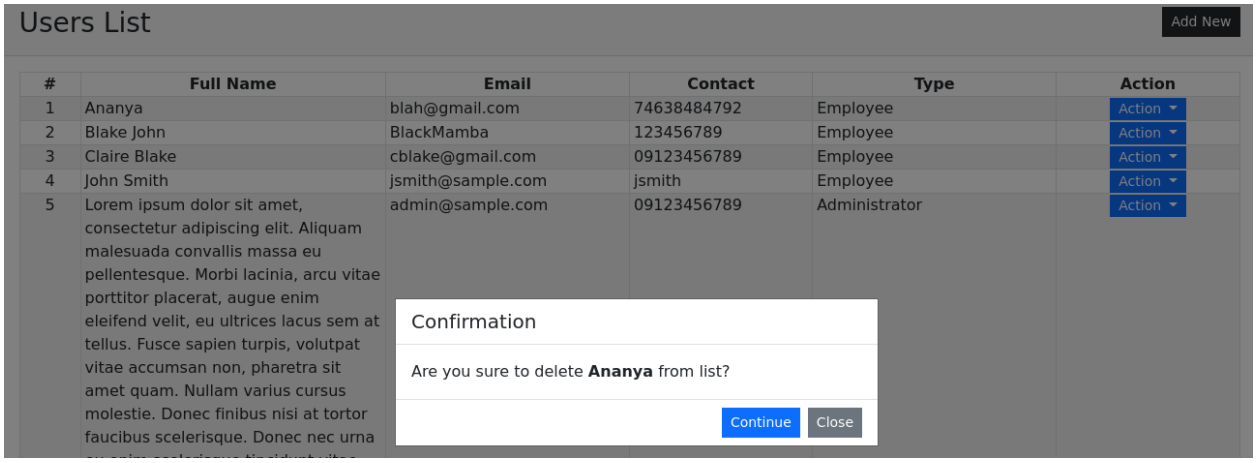
Update failed- “updating credentials failed” and display the error message



DELETE USER DETAILS-

If successful- expected output- ask for confirmation, delete details and reload log

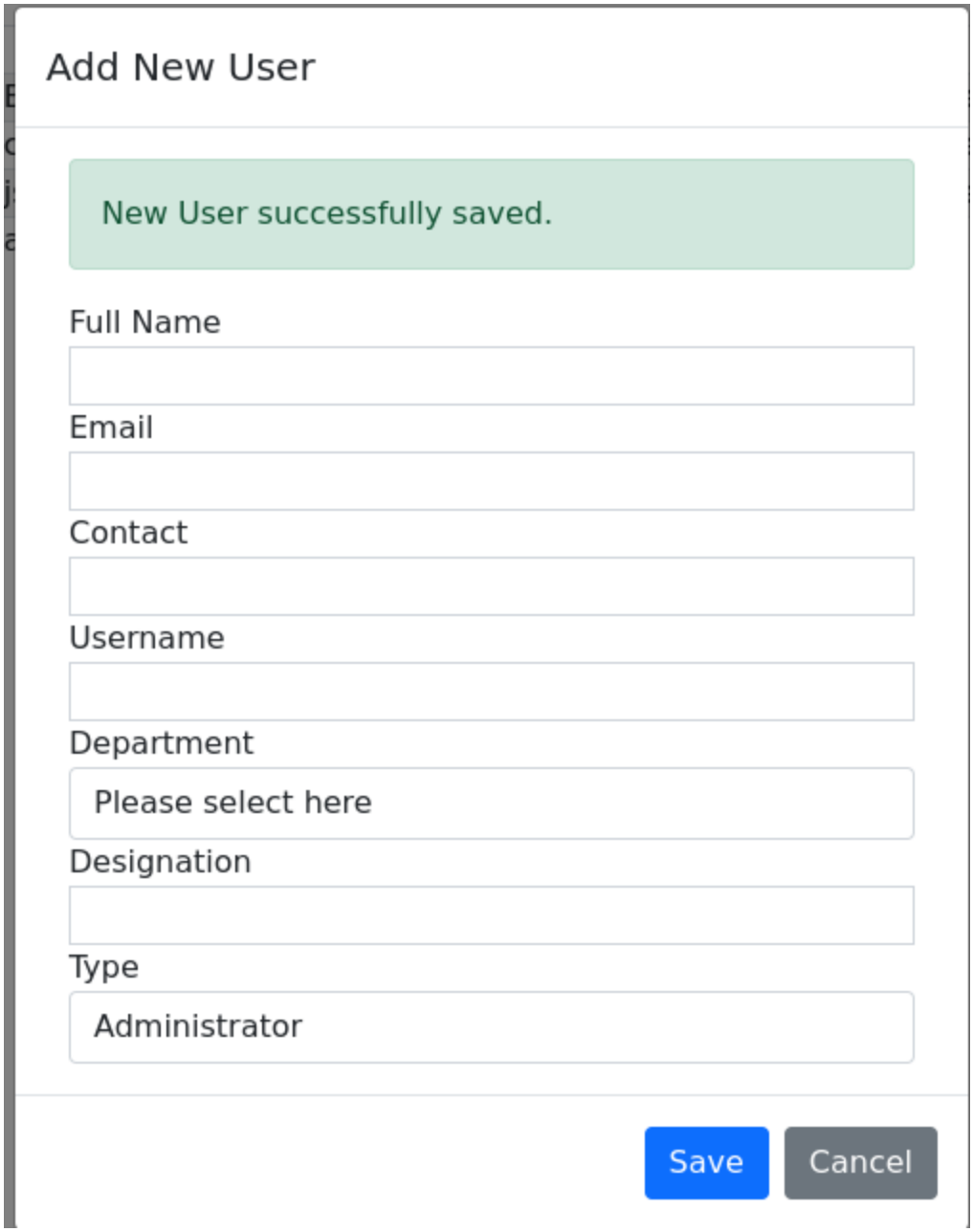
If failed- expected output- “An error occurred”



SUBMITTING USER FORM-

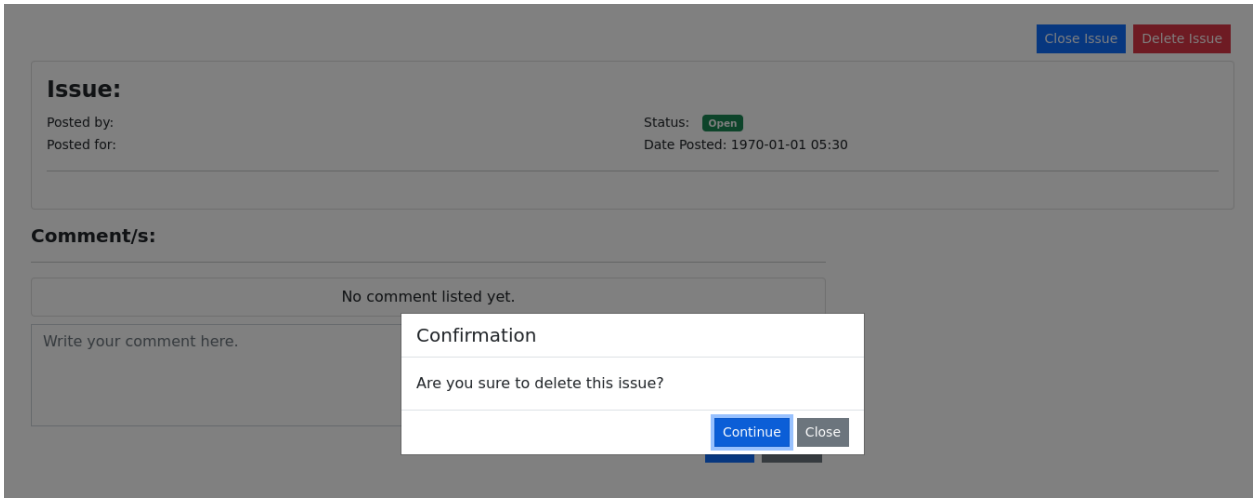
Successful- expected output- “submitting form…” and save user details

Failed- expected output- “An error occurred”

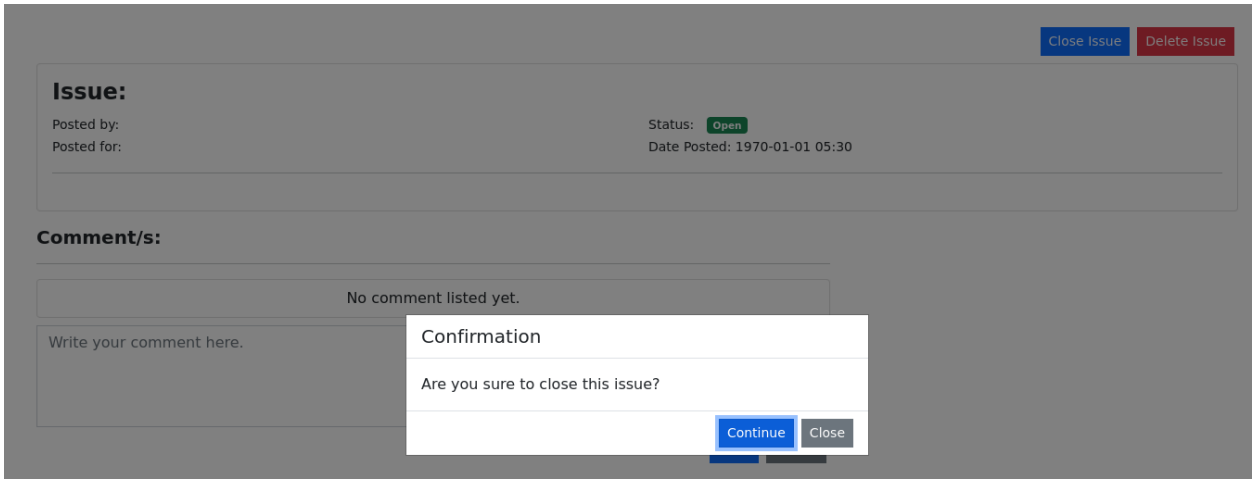


DELETING AN ISSUE-

Expected output- “Are you sure you want to delete this issue?”



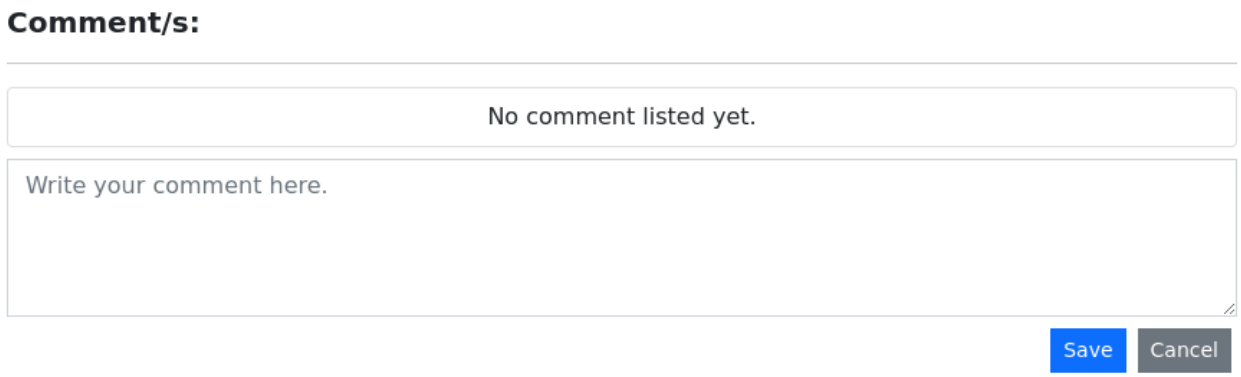
CLOSING AN ISSUE- “Are you sure you want to close this issue?”



ADDING A COMMENT-

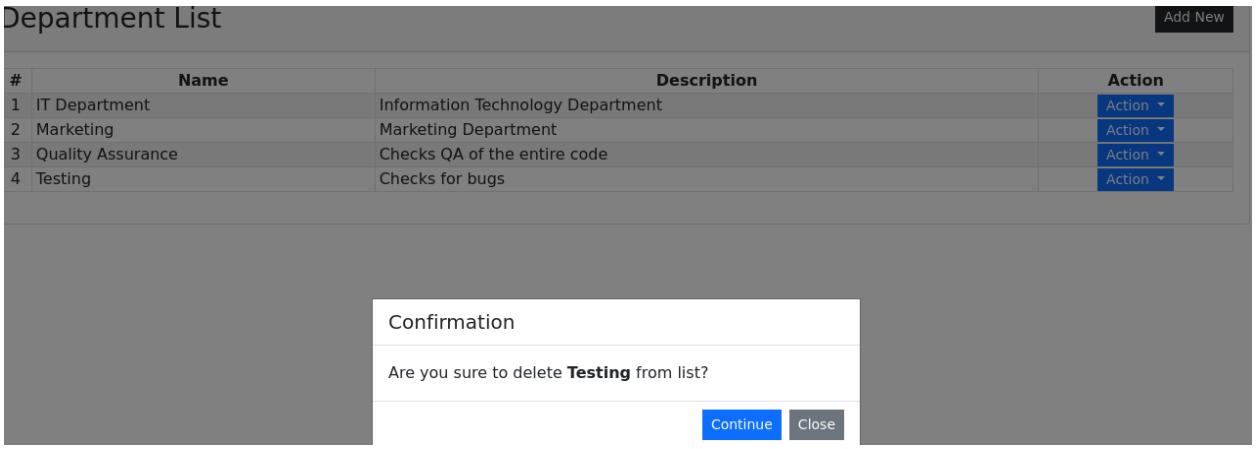
Successful- expected output- “Saving…” and save the comment in the database

Failed- expected output- “An error occurred”



DELETING A DEPARTMENT-

Expected output- “Are you sure to delete ‘department’ from list?” and delete it from the database and reload



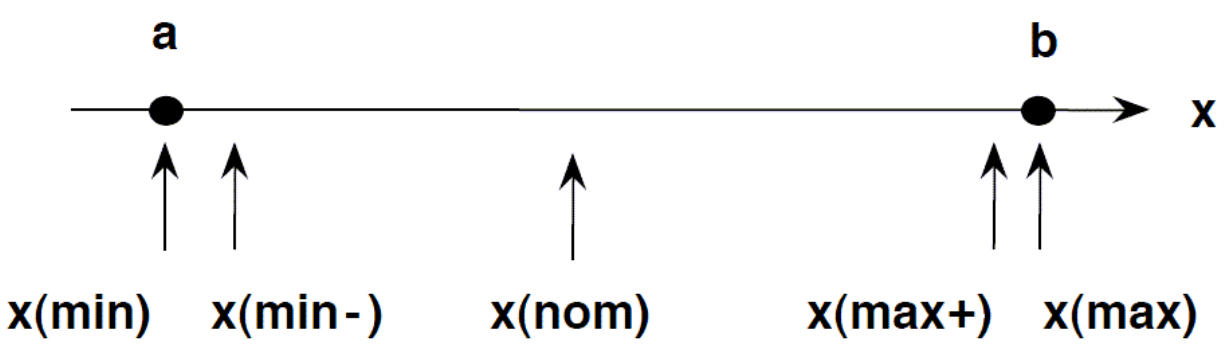
Dynamic Testing

2) Boundary Value Analysis

Boundary testing is the process of testing between extreme ends or boundaries between partitions of the input values.

- So these extreme ends like Start- End, Lower- Upper, Maximum-Minimum, Just Inside-Just Outside values are called boundary values and the testing is called “boundary testing”.
- The basic idea in normal boundary value testing is to select input variable values at their:

1. **Minimum**
2. **Just above the minimum**
3. **A nominal value**
4. **Just below the maximum**
5. **Maximum**



Login Credentials

Username and Password :

X(min) ✔️

<u>Testcase</u>
Username : a
Password : a

X(min-) ✔️

<u>Testcase</u>
Username : ab
Password : ab

X(max) ✔️

<u>Testcase</u>
Username : “It is a very large string”
Password : “It is a very large string”

Mutation Testing

Mutation testing, also known as code mutation testing, is a form of white box testing in which testers change specific components of an application's source code to ensure a software test suite will be able to detect the changes. Changes introduced to the software are intended to cause errors in the program. 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.
Mutation operators are changes made to the original code in order to generate mutants,those changes can be made by modifying expressions, changing, adding or removing operators and/or statements. These

operators can be arithmetic, relational, conditional, logical, assignment, among others.

Examples

```
function login(){
    extract($_POST);
    $sql = "SELECT * FROM user_list where username = '{$username}' and `password`
    $qry = $this->query($sql)->fetchArray();
    if(!$qry){
        $resp['status'] = "failed";
        $resp['msg'] = "Invalid username or password.";
    }else{
        $resp['status'] = "success";
        $resp['msg'] = "Login successfully.";
        foreach($qry as $k => $v){
            if(!is_numeric($k))
            $_SESSION[$k] = $v;
        }
    }
    return json_encode($resp);
```

Normal Code

```
function login(){
    extract($_POST);
    $sql = "SELECT * FROM user_list where username = '{$username}' and `password`
    $qry = $this->query($sql)->fetchArray();
    if($qry){
        $resp['status'] = "failed";
        // $resp['msg'] = "Invalid username or password.";
    }else{
        $resp['status'] = "success";
        // $resp['msg'] = "Login successfully.";
        foreach($qry as $k => $v){
            if(!is_numeric($k))
            $_SESSION[$k] = $v;
        }
    }
    return json_encode($resp);
```

Mutant Code

```php
if(!empty($password) && md5($old_password) != $_SESSION['password']){
    $resp['status'] = 'failed';
    $resp['msg'] = "Old password is incorrect.";
}else{
    $sql = "UPDATE `user_list` set {$data} where user_id = '{$_SESSION['user_id']}'";
    $save = $this->query($sql);
    if($save){
        $resp['status'] = 'success';
        $_SESSION['flashdata']['type'] = 'success';
        $_SESSION['flashdata']['msg'] = 'Credential successfully updated.';
        foreach($_POST as $k => $v){
            if(!in_array($k,array('id','old_password')) && !empty($v)){
                if(!empty($data)) $data .= ",";
                if($k == 'password') $v = md5($v);
                $_SESSION[$k] = $v;
            }
        }
    }else{
        $resp['status'] = 'failed';
        $resp['msg'] = 'Updating Credentials Failed. Error: '.$this->lastErrorMsg();
        $resp['sql'] =$sql;
    }
}
return json_encode($resp);
```

Normal Code

```php
if(empty($password) && md5($old_password) != $_SESSION['password']){
    // $resp['status'] = 'failed';
    // $resp['msg'] = "Old password is incorrect.";
}else{
    $sql = "UPDATE `user_list` set {$data} where user_id = '{$_SESSION['user_id']}'";
    // $save = $this->query($sql);
    if($save){
        $resp['status'] = 'success';
        // $_SESSION['flashdata']['type'] = 'success';
        // $_SESSION['flashdata']['msg'] = 'Credential successfully updated.';
        // foreach($_POST as $k => $v){
        //     if(!in_array($k,array('id','old_password')) && !empty($v)){
        //         if(!empty($data)) $data .= ",";
        //         if($k == 'password') $v = md5($v);
        //         $_SESSION[$k] = $v;
        //     }
        // }
    }else{
        $resp['status'] = 'failed';
        $resp['msg'] = 'Updating Credentials Failed. Error: '.$this->lastErrorMsg();
        $resp['sql'] =$sql;
    }
}
return json_encode($resp);
```

Mutant Code

```php
function save_user(){
    extract($_POST);
    $data = "";
    foreach($_POST as $k => $v){
    if(!in_array($k,array('id'))){
        if(!empty($id)){
            if(!empty($data)) $data .= ",";
            $data .= " `{$k}` = '{$v}' ";
            }else{
                $cols[] = $k;
                $values[] = "'{$v}'";
            }
        }
    }
    if(empty($id)){
        $cols[] = 'password';
        $values[] = "'".md5($username)."'";
    }
    if(isset($cols) && isset($values)){
        $data = "(".implode(',',$cols).") VALUES (".implode(',',$values).")";
    }
```

Normal Code

```php
function save_user(){
    extract($_POST);
    // $data = "";
    foreach($_POST as $k => $v){
    if(in_array($k,array('id'))){
        if(!empty($id)){
            if(!empty($data)) $data .= ",";
            $data .= " `{$k}` = '{$v}' ";
            }else{
                // $cols[] = $k;
                // $values[] = "'{$v}'";
            }
        }
    }
    if(empty($id)){
        // $cols[] = 'password';
        // $values[] = "'".md5($username)."'";
    }
    if(isset($cols) && isset($values)){
        $data = "(".implode(',',$cols).") VALUES (".implode(',',$values).")";
    }
```

Mutant Code

While Testing with Mutant Codes , the test cases returned opposite values of boolean value indicating its working properly

Our TestCases are in python

```python
import os
true= '✔️'
false= '❌'
test=[]

db=open("db/issue_tracker_db.db","r")
if db:
   print("Database file exists")
   test.append(true)

else:
   print("Database file does not exist")
   test.append(false)

db.close()

db_file=open("db/issue_tracker_db.db","r")
if os.stat("db/issue_tracker_db.db").st_size == 0:
   print("Database file is empty")
   test.append(false)
else:
   print("Database file is not empty")
   test.append(true)

db_file.close()
```

```python
css=os.path.exists("css")
if css:
   print("CSS file exists")
   test.append(true)
else:
   print("CSS file does not exist")
   test.append(false)

print("------------------------------------------------------------
--")
for i in test:
   if i == true:
       print("Test Passed",end="✔️\n")
   else:
       print("Test Failed",end="❌\n")

print("------------------------------------------------------------
--")
```

Normal Output

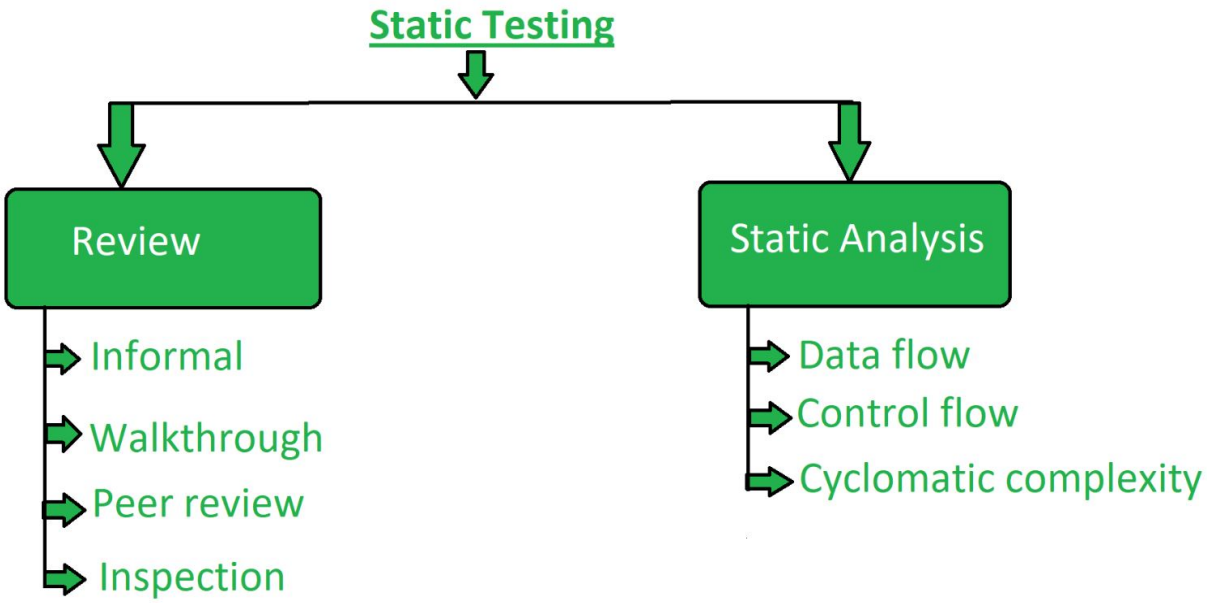
```
•$ /bin/python3 /home/adarsh/bug_tracking_system/issue_tracker/issue_tracker/check.py
 Database file exists
 Database file is not empty
 CSS file exists
 ------------------------------------------------------------
 Test Passed✔
 Test Passed✔
 Test Passed✔
 ------------------------------------------------------------
```

Mutant Output:

```
$ /bin/python3 /home/adarsh/bug_tracking_system/issue_tracker/issue_tracker/check.py
Database file exists
Database file is not empty
CSS file exists
------------------------------------------------------------
Test Failed✗
Test Failed✗
Test Failed✗
------------------------------------------------------------
```

4) Static Testing

Static Testing is a type of a software testing method which is performed to check the defects in software without actually executing the code of the software application. There are mainly two type techniques used in Static Testing:



1. **Review:**

In static testing review is a process or technique that is performed to find the potential defects in the design of the software. It is a process to detect and remove errors and defects in the different supporting documents like software

requirements specifications. People examine the documents and sort out errors, redundancies and ambiguities.

Review is of four types:

- **Informal:**

  In informal review the creator of the documents put the contents in front of the audience and everyone gives their opinion and thus defects are identified in the early stage.

- **Walkthrough:**

  It is basically performed by an experienced person or expert to check the defects so that there might not be a problem further in the development or testing phase.

- **Peer review:**

  Peer review means checking documents of one-another to detect and fix the defects. It is basically done in a team of colleagues.

- **Inspection:**

  Inspection is basically the verification of documents of higher authority like the verification of software requirement specifications (SRS).

2. **Static Analysis:**

Static Analysis includes the evaluation of the code quality that is written by developers. Different tools are used to do the analysis of the code and comparison of the same with the standard.

It also helps in following identification of following defects:

(a) Unused variables
(b) Dead code
(c) Infinite loops
(d) Variable with undefined value
(e) Wrong syntax

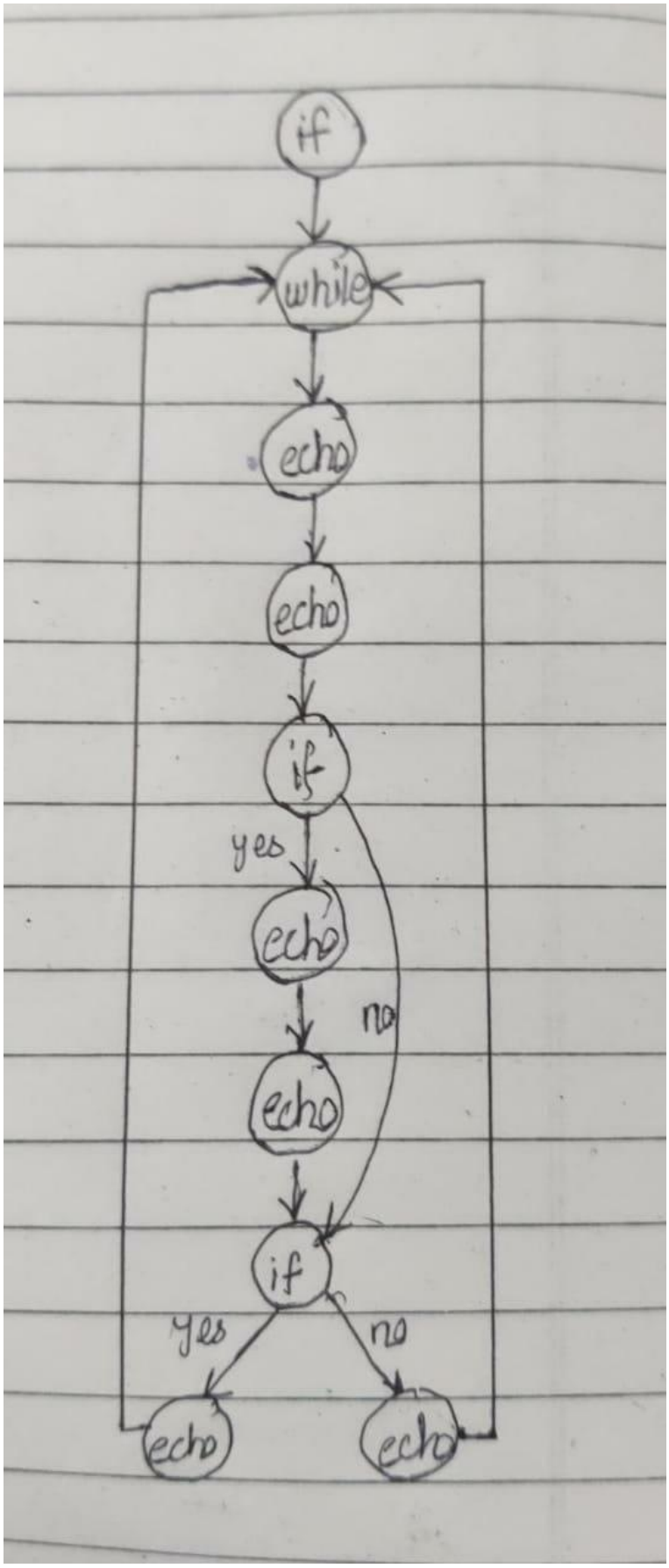
Static Analysis is of three types:

- **Data Flow:**
  Data flow is related to stream processing.
- **Control Flow:**
  Control flow is basically how the statements or instructions are executed.
- **Cyclomatic Complexity:**
  Cyclomatic complexity is the measurement of the complexity of the program that is basically related to the number of independent paths in the control flow graph of the program.

Cyclomatic Complexity

Strict cyclomatic complexity has been performed on each and every file that we have on our ammo. Cyclomatic complexity tells us the upper bound for the minimum number of possible statements that could be executed. It is very crucial to find out if your code has optimum cyclomatic complexity in order to keep it in check and make sure that the code does not suffer from executing infinitely and making sure there is a proper termination. The main part of our code is the issues.php file which reads out how many issues are generated along with which it displays the information about each issue. The code for the following program is as follows. The formula for the follows is E-N+(2*P).

```php
<?php
    $where = "";
    if($_SESSION['type'] != 1){
        $where = " where i.user_id = '{$_SESSION['user_id']}' ";
    }
    $sql = "SELECT i.*,u.fullname, d.name from issue_list i inner join department_list d on i.department_id = d.rowid inner join user_list u on i.u
    $qry = $conn->query($sql);
    // sql query to array
    while($row = $qry->fetchArray()):
?>
<div class="col">
    <div class="w-100 bg-dark text-light bg-gradient opcaity-70 py-3 px-2">
        <h5 class="truncate-1 border-bottom border-light" title="<?php echo $row['title'] ?>"><b><?php echo $row['title'] ?></b></h5>
        <small class="truncate-3">
        <?php
            echo $row['description'];
        ?>
        </small>
        <?php if($_SESSION['type'] == 1): ?>
            <small>By: <?php echo $row['fullname'] ?></small>
            <small>To: <?php echo $row['name'] ?></small>
        <?php endif ?>
        <div class="w-100 d-flex justify-content-between align-items-middle mt-3">
            <?php
            if($row['status'] == 0 ){
                echo "<span class='w-auto badge bg-success'>Open</span>";
            }else{
                echo "<span class='w-auto badge bg-danger'>Closed</span>";
            }
            ?>
            <a class="btn btn-sm btn-primary col-auto py-0 rounded-0" href="./?page=view_issue&id=<?php echo md5($row['issue_id']) ?>">View</a>
        </div>
    </div>
</div>
<?php endwhile; ?>
```


if
while
echo
echo
if
yes
echo
no
echo
if
yes
no
echo
echo

The number of edges(E) = 12

The number of nodes(N) = 10(AKA the control flow)

The total number of statements executed = 10

The number of exit points(P) = 2

E-N+2*P = 12-10+(2*2) = 2+4 = **6← cyclomatic complexity**

5) ACCEPTANCE TESTING-

The project was explained to the peer team and the accepted testing was done by Aditi Jain and Amisha Mathew.