Software Testing

Blackbox testing:

As the individual component started working, we did the integration of all the components. Later conducted black box testing of all the possible cases and functionalities. For example, as we update a profile, the corresponding database in phpmyadmin should also get updated. If the Boss declares a holiday, it should be visible to both employees and managers. We found a bug in age Updation by the birth date, which was corrected later.

Whitebox testing:

Codes under each role were tested manually by exchanging the codes written by us.
At this stage, we refined the code into a simpler possible version.

GUI Testing:

GUI testing for login page:

- All the buttons to login into either boss,employee or manager are working fine.
- Image in the background is displayed and aligned properly.
- Neither the content nor the images shrink, crop, or overlap when the user resizes the screen.

GUI testing for employee,manager,boss interface:

- All the buttons in the interface are working fine and properly aligned.
- Tables and text are properly aligned and displayed.
- Headings are aligned properly.

Unit Testing:

Sample Example: We have considered login as a Unit. Because it is a single functionality code snippet for logging into the website. We developed test cases on valid data

such as giving existing and correct email id and password stored in the database. Later tested with invalid data with mail ids not enrolled previously in the website which will not be available in the database storage. Also tested with the correct mail id and wrong password and vice versa of the situation. The software was able to identify such valid and invalid data and gave warning messages in case of invalid data. Also discussed among the teammates on how a single php file with more than a page containing higher functionalities could be separated into different php file snippets which gives more clear vision on testing.

```php
<!-- php script start -->
<?php
    $email_err = $pass_err = $login_Err = "";
    $email = $pass = "";

    if( $_SERVER["REQUEST_METHOD"] == "POST" ){

      if( empty($_REQUEST["email"]) ){
       $email_err = " <p style='color:red'> * Email Can Not Be Empty</p> ";
      }else {
       $email = $_REQUEST["email"];
      }

      if ( empty($_REQUEST["password"]) ){
       $pass_err =  " <p style='color:red'> * Password Can Not Be Empty</p> ";
      }else {
        $pass = $_REQUEST["password"];
      }

      if( !empty($email) && !empty($pass) ){

        // database connection
        require_once "../connection.php";

        $sql_query = "SELECT * FROM boss WHERE email='$email' && password = '$pass'  ";
        $result = mysqli_query($conn , $sql_query);

        if ( mysqli_num_rows($result) > 0 ){
         while( $rows = mysqli_fetch_assoc($result) ){
          session_start();
          session_unset();
          $_SESSION["email"] = $rows["email"];
          header("Location: dashboard.php?login-sucess");
         }
        }else{
          $login_Err = "<div class='alert alert-warning alert-dismissible fade show'>
          <strong>Invalid Email/Password</strong>
          <button type='button' class='close' data-dismiss='alert' >
            <span aria-hidden='true'>&times;</span>
          </button>
        </div>";
        }
```

Integration Testing:

Integration testing for logout:

Test case number	Input	Expected result	Actual result	Error
TC 1	Login into Boss	Should be able to access boss interface	As expected	No error
TC 2	Insert or delete an employee	Should be able to add an new employee or delete existing employee	As expected	No error
TC 3	Insert or delete a Manager	Should be able to add a new manager or delete exisiting manager	As expected	No error
TC4	Manage holidays	Should be able to add or delete holiday in holidays list	As expected	No error
TC 5	Manage employee leave	Should be able to accept or reject employee leave request	As expected	No error

TC 6	Logout from boss interface	Should logout from boss interface and get redirected to login page	As expected	No error

Similarly we conducted integration testing for employees and managers.

Functional Testing:

Test Case number	Test case	Expected result	Actual result	Error
TC 1	Login as employee,boss,manager	Successful login only when email and password matches	As expected	No error
TC 2	Privileges on boss	Boss should be able to edit employee,manager,holidays list and approval of employee leave request	As expected	No error
TC3	Privileges on manager	Manager should be able to edit employee details and approval of leave request	As expected	No error
TC 4	Privileges on employee	Employee should be able to apply for leave and able to see leave status	As expected	No error

System Testing:

Testcase number	Test case	Expected result	Actual result	Error
TC 1	Site display	All the features and images are displayed	As expected	No error
TC 2	Login	Employees,manager and boss are able to login into their respective dashboards successfully.	As expected	No error
TC 3	Boss interface	If the boss is able to edit manager and employee.If he is able to accept or reject employee leave request.He should also be able to edit holidays list.	As expected	No error
TC 4	Boss logout	He should be able to logout from his account and redirected to login portal	As expected	No error
TC 5	Manager interface	He should be able to edit employees and also have access to approve employee leave requests.	As expected	No error

TC 6	Manager logout	He should be able to logout from his account and redirected to login portal	As expected	No error
TC 7	Employee interface	He should be able to apply for leave.Leave status of the employee should be updated accordingly.	As expected	No error
TC 8	Employee logout	He should be able to logout from his account and redirected to login portal	As expected	No error

Static Testing:

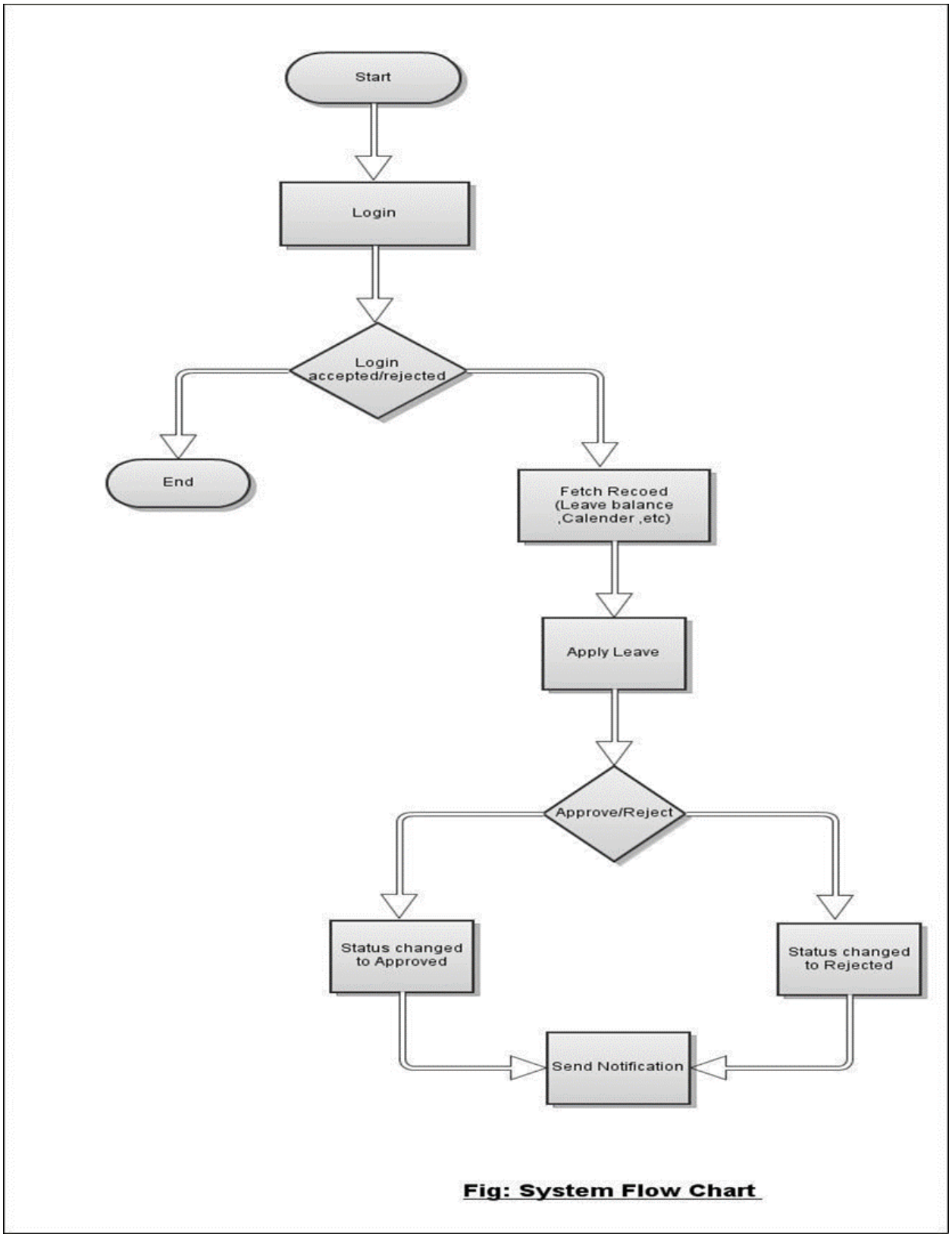

Fig: System Flow Chart

Cyclomatic complexity:
We have applied for leave as a unit. For this unit above the diagram is control flow graph.We are calculating cyclomatic complexity without execution by static testing.
Using R + 1 method
Therefore, R +1 = r1 + r2 + 1
= 3
Hence cyclomatic complexity is 3.
Using code:

```php
<?php
$reasonErr = $startdateErr = $lastdateErr = "";
$reason = $startdate = $lastdate = "";

if($_SERVER["REQUEST_METHOD"] == "POST"){

    if( empty($_REQUEST["reason"]) ){
        $reasonErr = "<p style='color:red'>* Reason is Required</p>";
    }else{
        $reason = $_REQUEST["reason"];
    }

    if( empty($_REQUEST["startDate"]) ){
        $startdateErr = "<p style='color:red'>* Start Date is Required</p>";
    }else{
        $startdate = $_REQUEST["startDate"];
    }

    if( empty($_REQUEST["lastDate"]) ){
        $lastdateErr = "<p style='color:red'>* Last Date is Required</p>";
    }else{
        $lastdate = $_REQUEST["lastDate"];
    }

        if( !empty($reason) && !empty($startdate) && !empty($lastdate) ){

            // database connection
            require_once "../connection.php";

            $sql = "INSERT INTO emp_leave( reason , start_date , last_date , email , status ) VALUES( '$reason' ,
            '$startdate' , '$lastdate' , '$_SESSION[email_emp]' , 'pending' )";
            $result = mysqli_query($conn , $sql);
            if($result){
                $reason = $startdate = $lastdate = "";
                echo "<script>
            $(document).ready( function(){
                $('#showModal').modal('show');
                $('#addMsg').text('leave Applied , Please Wait until it is approved!!');
                $('#linkBtn').attr('href', 'leave-status.php');
                $('#linkBtn').text('Check Leave Status');
                $('#closeBtn').text('Apply Another');
            })
        </script>
        ";
            }
```

Cyclomatic complexity:
CC = no. if statements + 1
= 7
Therefore, cyclomatic complexity is 7 for the given problem. Here we are not executing the code we are just looking at the code and finding the cyclomatic complexity and this is called static testing.

Dynamic Testing:

In order to ascertain whether each function operates effectively or not, we carried out dynamic testing by running the code. Additionally, we checked to see if the functionality met the requirements outlined in the SRS paper. targeting the creation of high-quality software, with as many test cases as is achievable.

Acceptance Testing:

We have given the code base for review and outlined our product's features and software needs. We also reviewed the neighboring team's code and filed bug reports for their project.
All of the user requirements have been satisfied in our project, and we have also planned for a few adjustments.
These modifications consist of:
1. We intended to provide helpful leave categories for employees, such as sick leave, marital leave, maternity leave, etc.
2. In addition to the leave information, we intended to include employee's personnel information.