Student Event Manager

Software Engineering

Project Report

Submitted by:

Himani (17001570021)

Ishan Ahmad (17001570024)

Kumar Sidhant (17001570027)

Supervisor:

Dr. Aman Pal

2019

Department of Computer Science
Acharya Narendra Dev College

CERTIFICATE

This is to certify that the project entitled "**Student Event Manager**" has been done by: **Himani**, **Ishan Ahamad** and **Kumar Sidhant** of **Bachelor of Computer Science (Hons.)** during Semester-IV from **Acharya Narendra Dev College**, University of Delhi under the supervision of Dr. Aman Pal.

Dr. Aman Pal.

ACKNOWLEDGEMENT

This Project was jointly undertaken by Himani, Ishan Ahamad and Kumar Sidhant as their Semester-IV Software Engineering Project, under the guidance and supervision of Dr. Aman Pal. Our primary thanks goes to him, who poured over every inch of our project with painstaking attention and helped us throughout the working of the project. It is our privilege to acknowledge our deepest gratitude to him for his inspiration which has helped us immensely. We are extremely grateful to him for his unstilted support and encouragement in the preparation of this project.

Problem Statement:

Missing deadlines, fests and the conference. We were waiting for isn't uncommon for us and that is exactly the problem we are trying to solve.

Having everything sorted and arranged at a glance automatically and accurately is the only aim.

Students can select the categories of interest which are automatically added to their calendar ensuring they don't miss any important date.

Process Model:

We want to provide information to every student about every upcoming event in the college .

We can enhance functionality later on, by providing information about other events outside the college like events of other college's or some other state or district level competitions or other events.

Therefore our first goal is just to provide a basic functionality to users to manage their events in college, which can be evolved later. So we want to use an **INCREMENTAL** ***MODEL*** for our project.

1. Software Requirement Specification

Requirements analysis, also called requirements engineering, is the process of determining user

expectations for a new or modified product. These features, called requirements, must be quantifiable,

relevant and detailed. In software engineering, such requirements are often called functional

specifications.

1.1 Overall Description:

This section describes the software from different perspectives like functions, constraints, etc.

1.1.1 Product Functions:

SEM is a web application. It uses the college's login id to login in the system. If you are logged in it will notify you about upcoming events in two ways :

a)Details in notification section which can also be categorised on interest basis.

b)setting marks on the calendar for upcoming events.

1.1.2 User Characteristics

a)Any student of college is a user who needs to know about coming events, assignments in college. He can search for any event on the basis of his interest and can apply changes to his own calendar schedule. No educational background is required for using this for a student.

b)Any teacher or staff of college can be admin who enters for the details of events or submissions. He will be allowed to do so by using some special login id’s or formats. Knowledge about events should be there for the admin.

1.1.3 General Constraints

This application provides web access for all students and staff to manage their regular routine. The user interface will be intuitive enough that no extra training is required for any member. All online transactions and the storage of confidential member information will be done in a secure environment. Persistent storage for membership and event information will be maintained.

1.1.4 Assumptions and Dependencies

a) All students and administrator staff have knowledge about the internet and browsers.

1.2 External Interface Requirements

Defines the input and output process of the software.

1.2.1 User Interfaces

There are various graphical interfaces through which any user(students) can interact with the software , which are as followed:

Registration and login part

Notification part(all notifications will be displayed)

Categorised notifications

Calendar

1.2.2 Hardware Interfaces

Hardware interface can be any device through which a user can access the software . In our case it can be an android phone, a computer or a laptop; which provides access to a browser.

User device should have at least 1gb ram .

1.2.3 Software Interfaces

Any OS can act as a software interface which provides support to any of the browser therefore software interfaces can be android ,linux ,windows etc.

1.3 Functional Requirements

1.3.1 FR 1 Registration form

Registration form must contain fields for entering name ,rollno , college name (which should have ANDC as default value), course name , contact no, email id etc.

1.3.2 FR 2 login form

Login form contains two fields a)user id, b) password; and also contains a login button.

It will also have a forgot password link for which new password setting options will be provided after authentication of user through OTP or a link at registered email id.

1.3.3 FR 3 Event details form

First it should have categorised sections for types of events like a test,an assignment or any other cultural event of college .

For tests or submissions; it should have fields of course name, test or submission date , syllabus description .

For cultural events ; It should provide fields for event name , event date and time, venue, coordinators name and contact number, scheduled description about event .

FR 4 Notification look up

A table presenting event name, date and description.

1.4 Performance Requirement

It will have compatibility for mozilla firefox browsers. Changes to any field should be stored in the case of recent power breakup.

1.5 Design Constraints

1.6 Data Flow Diagram

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. A DFD is often used as a preliminary step to create an overview of the system, which can later be elaborated.

Context Diagram:

A context level DFD is the most basic form of DFD. It aims to show how the entire

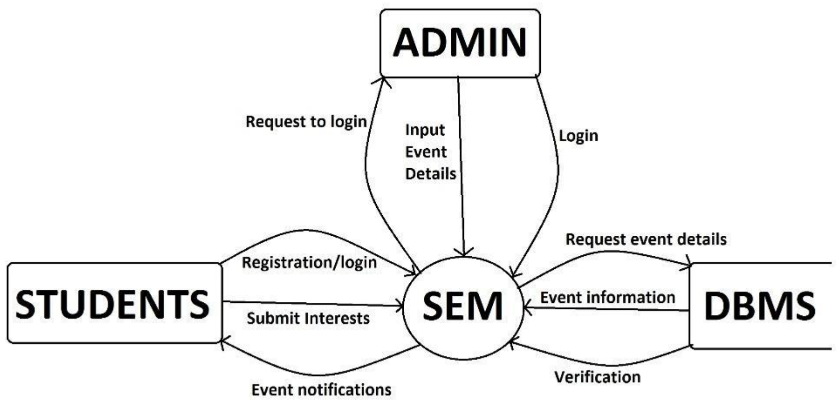


fig.1:CONTEXT LEVEL DFD

system works at a glance. There is only one process in the system and all the data flows either into or out of this process. Context level DFD's demonstrates the interactions between the process and external entities.When drawing Context Level DFD's, we must first identify the process, all the external entities and all the data flows.

DFD Level-1:

Level 1 DFD's aim to give an overview of the full system. Major processes are broken down into sub-processes. Level 1 DFD's also identifies data stores that are used by the major processes. When constructing a Level 1 DFD, start by examining the Context Level DFD. We must break up the single process into its sub-processes. We must then pick out the data stores from the text we are given and include them in our DFD. Like the Context Level DFD's, all entities, data stores and processes must be labelled.

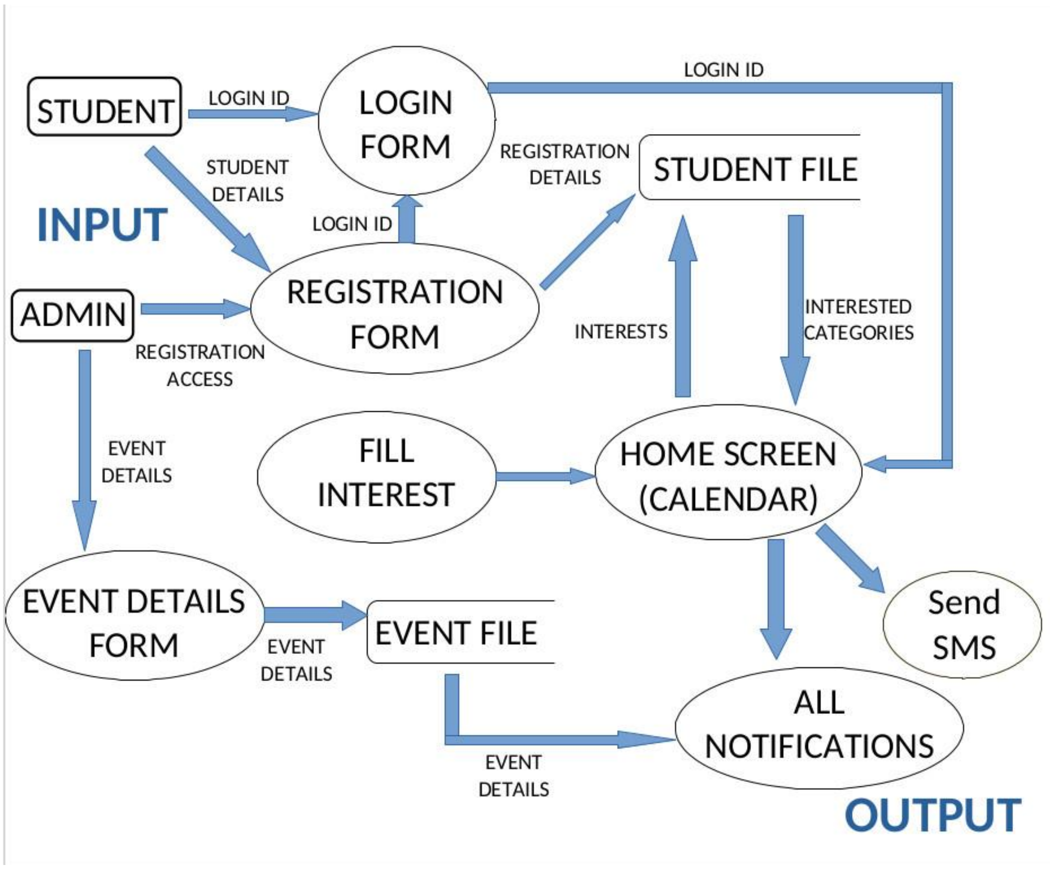


fig.2:LEVEL-1 DFD

DFD level-2:

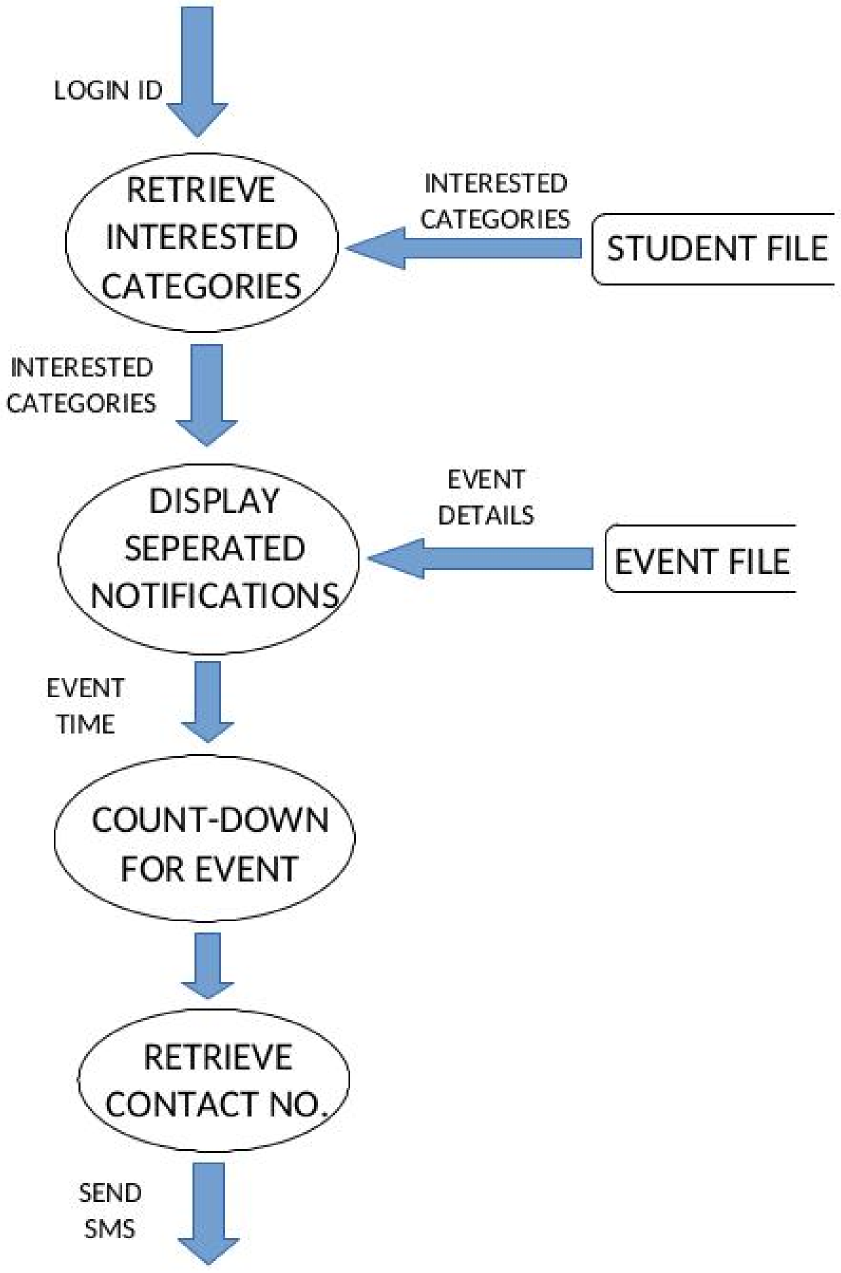


fig.3:LEVEL-2 DFD

1.7 Data Dictionary

A Data Dictionary is a repository of various data defined in the DFDs. The associated Data Dictionary states precisely the structure of each data flow in a given DFD. The ‘+’ symbol represents composition, ‘|’ means selection and ‘*’ means repetition.

Data	**Description**
Student details	Student_name + Roll_no + course+year [word+word+word+Digit]
Event details	EventName + Time + Venue + Description + Category [word + days + word + Line + word]
Intrested_category	Fest | Debate | Dancing | Singing | BSc(h)cs
Line	[word+word+word]*
word	[Character + Character + Character]*
Character	[Alphabet | Digit | special characters]
Alphabet	a-z | A-Z
Digit	0-9

TABLE.1:DATA DICTIONARY

2. Estimations

Size bye estimation is essential for Software Project Management. Helps the product manager to further predict the effort needed to build the project.

2.1 Function Points

Function-Oriented software metrics use a measure of the functionality delivered by the application as a normalization value. Since "functionality" cannot be measured directly, it must be derived indirectly using other direct measures. Thus, function points are used. Five information domain characteristics are determined and counts are provided in the appropriate table location. Information domain values are defined in the following manner:

Number of user inputs: Each user input that provides distinct application oriented data to the software is counted. Inputs should be distinguished from inquiries, which are counted separately.

Number of user outputs: Each user output that provides application oriented information to the user is counted. In this context, output refers to reports, screens, error messages, etc. Individual Data Items within a report are not counted separately.

Number of user inquiries: An inquiry is defined as an on-line input that results in the generation of some immediate software response in the form of an on-line output. Each distinct inquiry is counted.

Number of files: Each logical master file is counted.

Number of external interfaces: All machine readable interfaces that are used to transmit information to another system are counted.

	Measurement factors	**Count**	**Simple**	**Averag e**	**High**	**Total**
	Number of user inputs	4	3	4	6	12
	Number of user outputs	2	4	5	7	8
	Number of user inquiries	3	3	4	6	9
	Number of internal logical files	2	7	10	15	14
	Number of external interfaces	0	5	7	10	0
	Count total					43

TABLE.2:FUNCTION POINT

Once these data have been collected, a complexity value is associated with each count. Organizations that use Function point methods develop criteria for determining whether a particular entry is simple, average, or complex. To compute function points (FP), the following relationship is used:

FP

= count total * [0.65 + 0.01 * $\sum$ (Fi)]

= 43 * [0.65 + 0.01*35]

=43 * 1

=43

Where count total is the sum of all FP entries.

The Fi (i=1 to 14) are "complexity adjustment values" based on responses to the following questions:

1. Does the system require reliable backup and recovery?	1
2. Are data communication required?	3
3. Are there distributed processing functions?	1
4. Is performance critical?	5
5. Will the system run in an existing, heavily utilized operational environment?	5
6. Does the system require on-line data entry?	3
7. Does the on-line data entry require the input transaction to be built over multiple screens or operations?	0
8. Are the master files updated on-line?	3
9. Are the inputs, outputs, files, or inquiries complex?	2
10. Is the internal processing complex?	3
11. Is the code designed to be reusable?	3
12. Are conversion and installation included in the design?	2
13. Is the system designed for multiple installations in different organizations?	2
14. Is the application designed to facilitate change and ease of use by the user?	2

TABLE.3:COMPLEXITY ADJUSTMENT VALUES

2.2 Efforts

Productivity = 100 (approx.)

Effort = FP/productivity

= 43/100= 0.43

3. Scheduling

Scheduling of a software project doesn’t defer greatly from any multitask engineering effort , therefore generalised project scheduling tools and techniques can be applied with little modification for software projects while creating a software project.

While creating a software project a timeline chart is generated.

A timeline chart can be generated for an entire project, all tasks are listed in the left hand column and the horizontal bars depict the duration of each task.

When multiple bars occur at the same time on the calender task concurrency is implied.

The diamond indicate the milestone is achieved.

Work tasks Week	1		2		3		4		5		6		7		8		9		10		11		12	
0.Identify problem & need of s/w	■																							
Milestone: Product Stmt. Defined	◆																							
1.Document SRS		■	■	■	■	■	■	■																
1.2 External Interface Requirements		■																						
1.3 Document Function Requirements		■	■	■																				
1.4 Identify Performance Requirements				■	■																			
1.5 Identify Design Constraints					■																			
1.6 Design Data Flow Diagram					■	■	■																	
1.7 Design Data Dictionary							■	■																
Milestone: SRS documented								◆																
2 Estimations									■	■														
2.1 Evaluate Function Point									■	■														
Milestone: Effort & cost evaluated																								
3 Scheduling											■	■												
Milestone: Generate Schedule Table												◆												
4 Risk Management													■	■										
Milestone: Risk Table Generated														◆										
5 Design															■	■	■	■						
5.1 Architectural designing															■	■	■							
5.2 Component designing																	■	■						
Milestone: Designs Documented																		◆						
6 Coding																			■	■	■	■		
7 Testing Complexity																					■	■		
Milestone: Testing process performed																						◆		
8 References																							■	
Milestone: Document all references																							◆	

TABLE.4:SCHEDULING CHART

4. Risk Management

A risk table provides you with a simple technique for risk projection. All risks are denoted in the first column of the table. Each risk is categorized according to type of risk. PS implies a Project Size risk, BU implies a Business risk, TE implies a Technological Environment risk and DE implies a Development Environment risk.

The impact is rated as:

1. Negligible
2. Marginal
3. Critical
4. Catastrophic

Risk	**Type**	**Probability**	**Impact**	**RMMM**
Size estimate may be significantly low	PS	30%	1	
Larger number of users than planned	PS	20%	3	
End-users resist system	BU	70%	3	
Delivery deadline will be tightened	BU	60%	2	
Technology will not meet expectations	TE	30%	2	
Lack of training tools	DE	20%	2	
Staff inexperienced	ST	50%	1	
Staff turnover will be high	ST	20%	2	

TABLE.5:RISK TABLE

5. Design

Depicts the functioning structure of the software at various levels.

5.1 Architectural Design

The first-level factoring results in a very high-level structure,where each subordinate has a lot of processing to do.To simplify these modules ,they must be factored into subordinate modules that will distribute the work of a module.Each of the input,output and transformation modules must be considered for factoring.

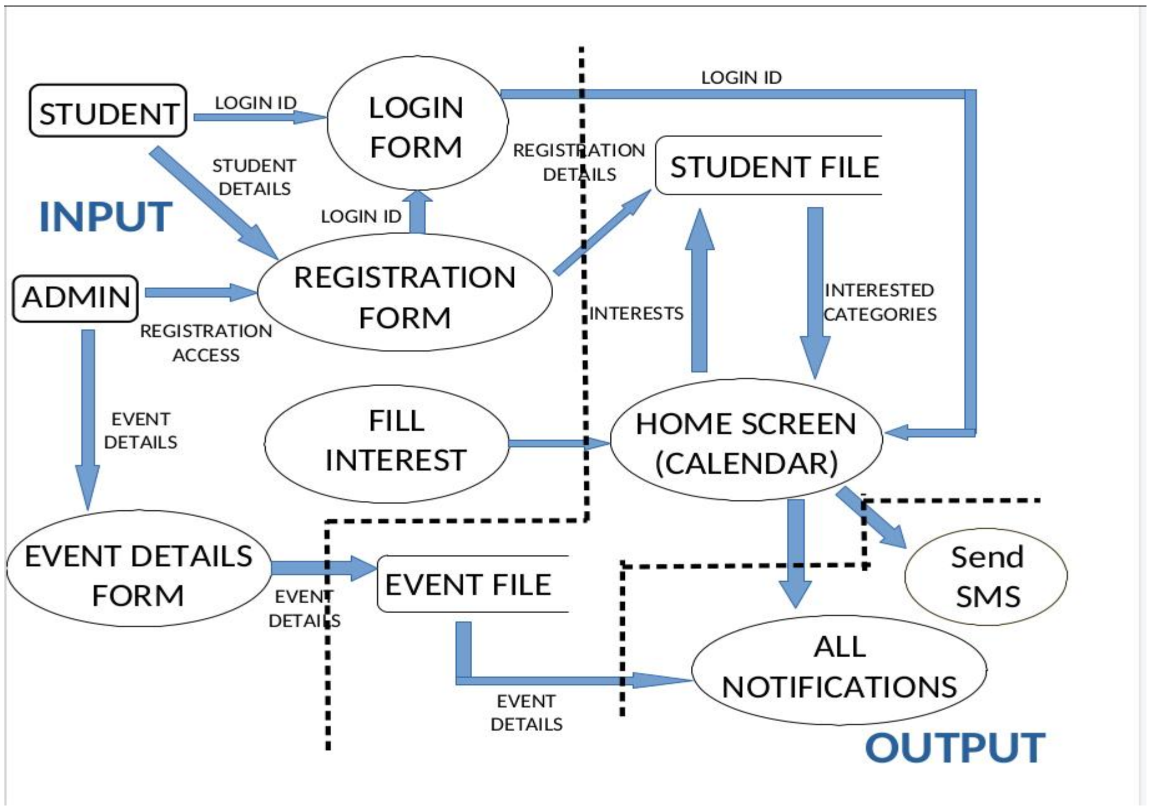


fig.4:DATA FLOW DIAGRAM FOR SE

1.Identify the input and output data elements:-

The most abstract input data elements are those data elements in the data flow diagram that are that are farthest removed from the physical inputs but can still be considered inputs to the system.The most abstract output data elements(MAO) by starting from the outputs in the data flow diagram and travelling toward the inputs.

Inputs to the system:-*Registration details, Login-id, Password, Event details, Interests of student.*

Outputs by the system:*Notifications of events(All and Interest based),SMS*

5.2 Component Design

In component design,the goal is to factorise the system into modules and submodules and write pseudo codes for them.

1.First-level factoring:-

The first-level factoring is straightforward,after the most abstract input and output data items are identified in the flow diagram.The main module is the overall control module,which will form the main program or procedure in the implementation of the design

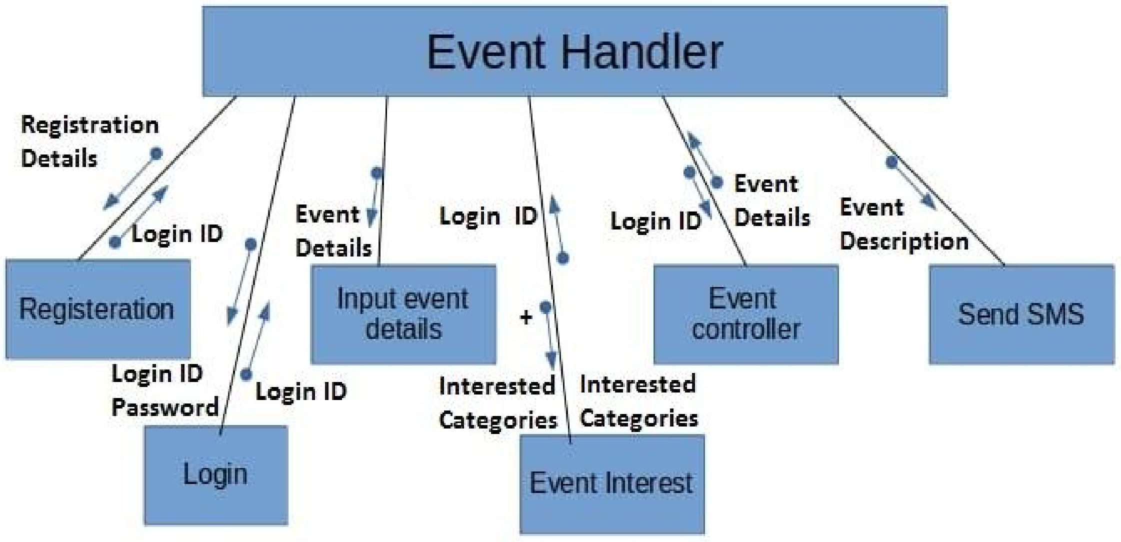


fig.5:FIRST LEVEL FACTORING

2.Factoring of input,output and transform branches:-

First level Modules must be factored into subordinate modules that will distribute the work of a module.Each of the input,output and transformation modules must be considered for factoring.

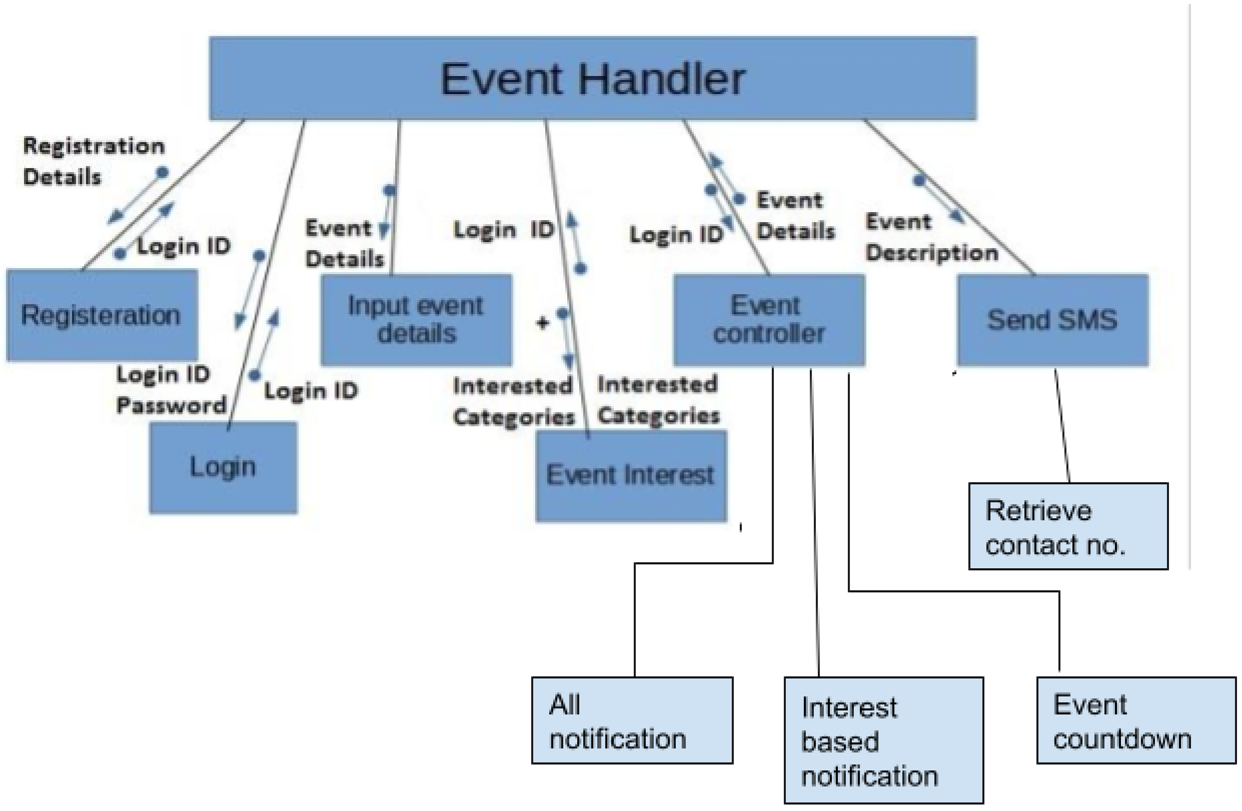


fig.6:SECOND LEVEL FACTORING

Iteration 1: Iteration 1 shows the structure chart for the software being delivered at first time to the client which has its basic functionality.

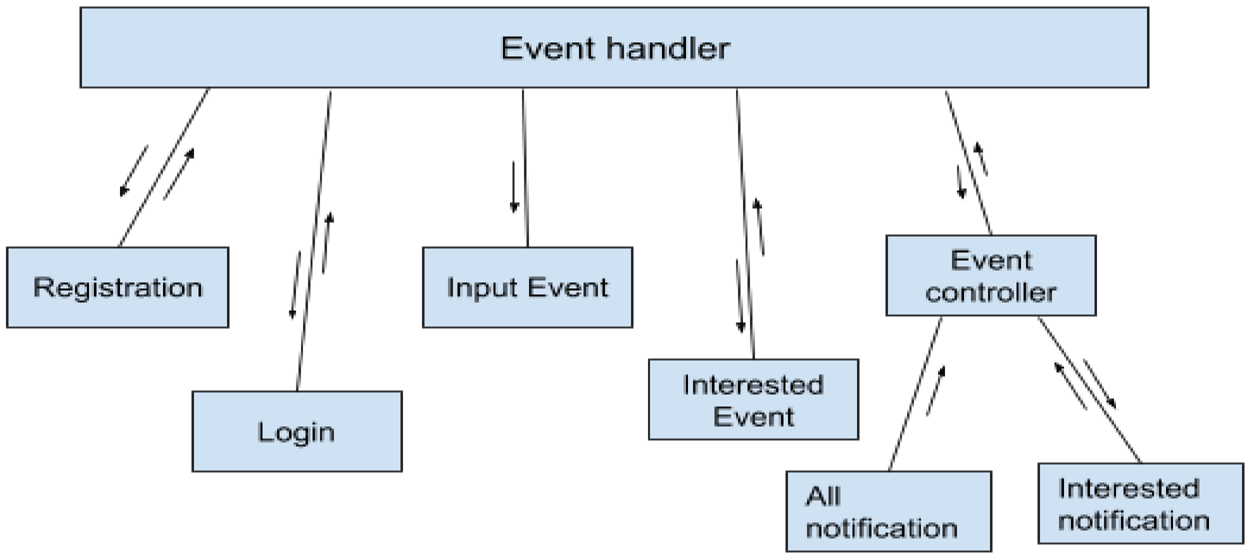


fig.7:FIRST ITERATION STRUCTURE CHART

Pseudo codes:

```
1. Registration
    inputs:-6(Name , rollno, course, year, password ,submit(button))
    output:- 1 (login id)
    steps:-1.Take value
            2.if name,rollno,password are not null
            3.then Save to database
            4.endif
2. Login
    inputs:-3 (login id, password, login (button))
    output:-1 (login id )
    steps:-1: Take input
            2.Match with values in database
            3.if matched->login
            4.else-> try again
            5.endif
```

3. **Input Event Details**

inputs:-6-8**(**Eventname, date, time, category, course & year, description, submit button)

steps:-1.Take inputs

```
   2.if(category == Test/assignment)
   3.then enable course and year fields
   4.else disable course and year fields
   5.Press (submit)hi
```

4. **EventInterest**

inputs:-3(login_id,)

```
 1.Choose all the categories you want.
 2. On save changes
 3. Delete all tuples leaving its course for that rollno
 4. Add new tuples for all the selected categories into the interested categories
```

5. **All notification**

```
1. Retrieve login id
2. Retrieve all events with Eventdate >= presentdate
3. while(!empty event list)
4. Display event notifications
```

5.endwhile

6. **Interest based notification**

inputs:-login id

outputs:-table showing events

steps:-1. Retrieve login id

2. Retrieve all events belonging to interested categories of login id with Eventdate >= presentdate

3. while(!empty event list)

4. Display event notifications

5.endwhile

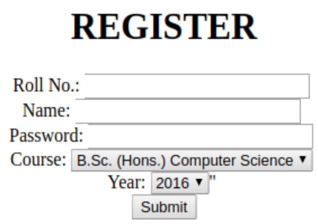
6. Coding

The objective of coding phase is to transform the design of a system into code in a high level language and then to unit test this code. Good software development organizations normally require their programme to adhere to some well-defined and standard style of coding called coding standards.

Registration.php:

```
<html>
<style>
.center {
  margin: 0;
  position: absolute;
  top: 50%;
  left: 50%;
  -ms-transform: translate(-50%, -50%);
  transform: translate(-50%, -50%);
}
</style>
<div class="center", align="center">
<body>
<h1>REGISTER</h1>
<form method="POST" action="REGISTER.php">
Roll No.:
<input type="text" name="RollNo" required>
<br>
```

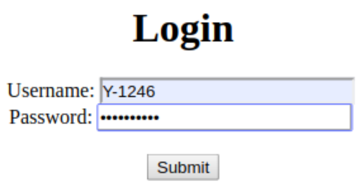
```
Name:
<input type="text" name="Name" required>
<br>
Password:
<input type="password" name="Password"
required>
<br>
Course:
<select name="Course">
    <option value="BSc(H)CS">B.Sc. (Hons.)
Computer Science</option>
     <option value="BSc(H)C">B.Sc. (Hons.)
Chemistry</option>
     <option value="BSc(H)M">B.Sc. (Hons.)
Mathematics</option>
     <option value="BSc(H)P">B.Sc. (Hons.)
Physics</option>
</select>
<br>
Year:
<select name="Year">
  <option value="2016">2016</option>
  <option value="2017">2017</option>
  <option value="2018">2018</option>
  <option value="2019">2019</option>
</select>"
<br>
<input type="submit" id="btn" value="Submit">
</form>
<?php
    $connect = new mysqli("localhost", "root",
"qwerty@asdfgh", "SEM");
    if ($connect->connect_error)
                die("ERROR: Unable to connect: "
. $connect->connect_error);
    $query="INSERT INTO STUDENT
VALUES(\"".$_POST["RollNo"]."\",\"".$_POST
["Name"]."\",\"".$_POST["Password"]."\",\"".$_
POST["Course"]."\",\"".$_POST["Year"]."\");";
    if($_POST["RollNo"] and
$_POST["Name"] and $_POST["Password"])
    {
                if
(!mysqli_query($connect,$query))
                {
                            echo("Error description: "
. mysqli_error($connect));
                }
    }
?>
</body>
</div>
</html
```


REGISTER

Roll No.:
Name:
Password:
Course: B.Sc. (Hons.) Computer Science
Year: 2016 "
Submit

Login.php:

```
<html>
<style>
.center {
  margin: 0;
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
}
</style>
<div class="center", align="center">
<body>
<?php
session_start();
if ($_SESSION["CHECK"] == "ERROR")
{
    echo "INVALID INPUT";
    $_SESSION["CHECK"] == "CLEAR";
}
session_destroy();
?>
<h1>Login </h1>
<form method="POST" action="CHECK.php">
Username:
<input type="text" name="Username">
<br>
Password:
<input type="password" name="Password">
<br>
<br>
<input type="submit" id="btn" value="Submit">
</form>
</body>
</div>
</html>
```



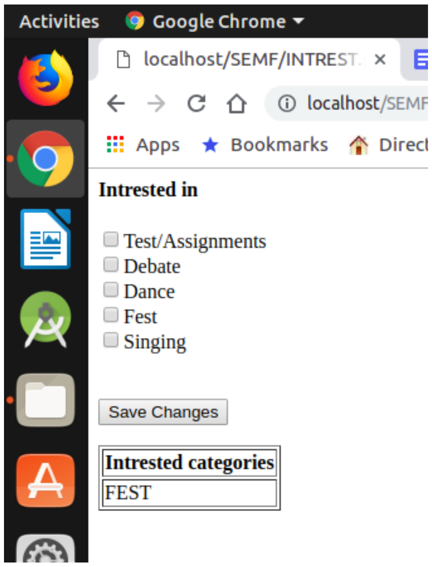
Check.php:

```
<html>
<body>
<?php
session_start();
function fun($u,$p)
{
$connect = new mysqli("localhost", "root",
"qwerty@asdfgh", "SEM");
    if ($connect->connect_error)
    {
            die("ERROR: Unable to connect: "
. $connect->connect_error);
    }
$query="SELECT NAME FROM
STUDENT WHERE RNO=\"".$u."\" AND PWORD=\"".$p."\"";
    $udetails=$connect->query($query);
    if($row=mysqli_fetch_array($udetails))
    {
            $_SESSION["UNAME"]=$u;
            echo "Homescreen";
            header("Location: CALENDER.php");
    }
    else
    {
    $_SESSION['CHECK'] = "ERROR";
    header("Location: LOGIN.php");
    }
    mysqli_close($connect);
}
$u=$_POST["Username"];
$p=$_POST["Password"];
fun($u,$p);
?>
</body>
</html>
```

Interest.php:(for filling interest)

```
<html>
<body>
<h4>Intrested in</h4>
<form method="post" action="INTREST.php">
<input type="checkbox" name="category" value="Test/Assignments">Test/Assignments
<br>
<input type="checkbox" name="category" value="Debate">Debate
<br>
<input type="checkbox" name="category" value="Dance">Dance
<br>
<input type="checkbox" name="category" value="FEST">Fest
<br>
<input type="checkbox" name="category" value="Singing">Singing
<br><br><br>
<input type="Submit" name="save" value="Save Changes">
<br>
<!--<input type="submit" name="back" value="back">
<br>-->
</form>
<?php
session_start();
    $connect = new mysqli("localhost", "root", "qwerty@asdfgh", "SEM");
    if ($connect->connect_error)
    {
            die("ERROR: Unable to connect: " . $connect->connect_error);
    }
    $query="SELECT CATEGORY FROM INTEREST WHERE RNO=\"".$_SESSION["UNAME"]."\";";
    $category=$connect->query($query);
    if (!mysqli_query($connect,$query))
            echo("Error description: " . mysqli_error($connect));
    echo "<table border='1'>";
    echo "<tr>";
    echo "<th>Intrested categories</th>";
    echo "</tr>";
```

```
while($row=mysqli_fetch_array($category))
    {
            echo "<tr>";
            echo "<td>", $row["CATEGORY"]
,"</td>";
            echo "</tr>";
    }
    echo "</table>";
    if(isset($_POST["save"]))
    {
            if(isset($_POST["category"]))
    {
      $query="INSERT INTO INTEREST
VALUES(\"".$_SESSION["UNAME"]."\",\"".$_
```



Intrested in

- [ ] Test/Assignments
- [ ] Debate
- [ ] Dance
- [ ] Fest
- [ ] Singing

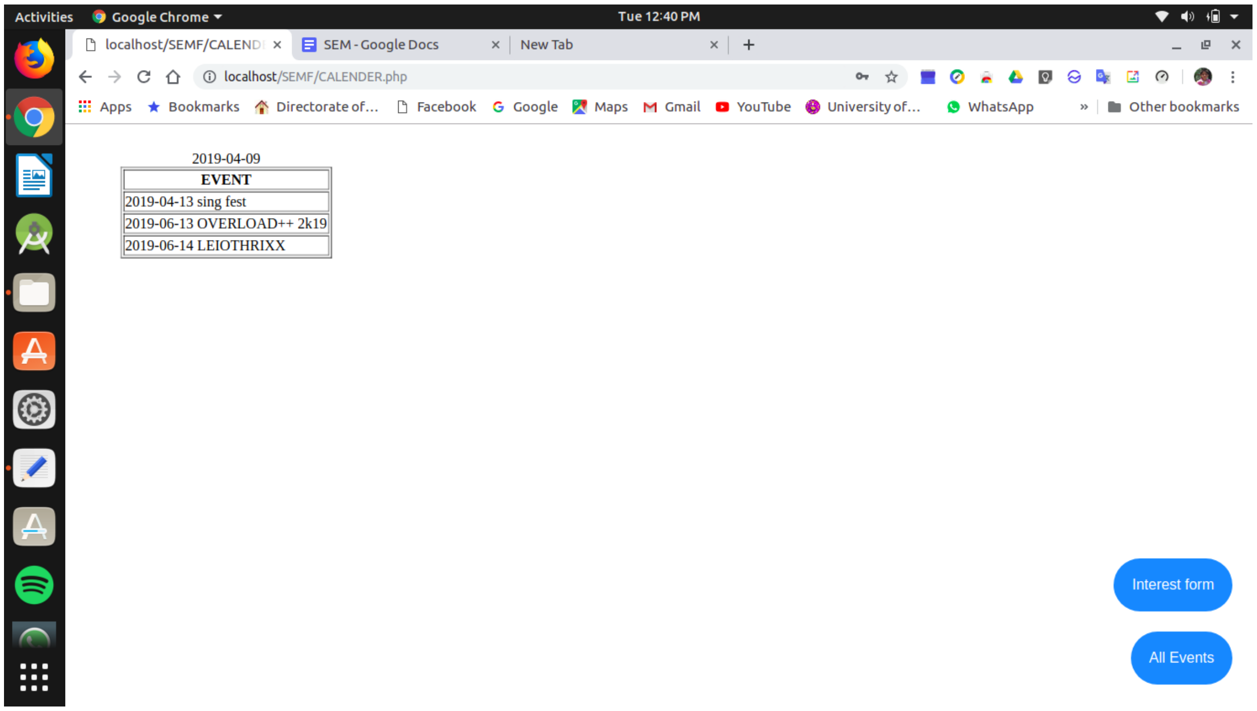
Intrested categories
FEST

```
POST["category"]."\");";
 if (!mysqli_query($connect,$query))
    echo("Error description: " .
mysqli_error($connect));
            }else echo "nothing selected";
            header("Location:
CALENDER.php");
    }
    //after saving new intrest go back to
Calender
?>
</body>
</html>
```

CALENDER.php:(for interest based notification)

```
<html>
<style>
.center {
  margin: 0;
  position: absolute;
  top: 10%;
  left: 10%;
  transform: translate(-30%, -30%);
}
.button {
  background-color: #009FFF;
  border: none;
  color: white;
  padding: 20px;
  text-align: center;
```

```
 text-decoration: none;
  display: inline-block;
  font-size: 16px;
  margin: 4px 2px;
  position:absolute;
  right:20;
}

.button1 {border-radius: 50px;bottom:20;}
.button2 {border-radius: 50px;bottom:100;}
</style>
<form action="INTREST.php" >
<input class="button button2" type="Submit"
name="i" id="btn" value="Interest form">
</form>
<form action="ALL.php">
<input type="submit" class="button button1"
id="btn" value="All Events">
</form>
<div class="center", align="center">

<?php
session_start();
function fun($u)
{
	$connect = new mysqli("localhost", "root",
"qwerty@asdfgh", "SEM");
	if ($connect->connect_error)
	{
		die("ERROR: Unable to connect: "
. $connect->connect_error);
	}
	$cdate=date("Y-m-d");
	echo $cdate;

	//RETRIEVING EVENTS BELONGING
TO INTERESTED CATEGORIES
	$query="SELECT DISTINCT
EC.ENAME,E.DATE FROM INTEREST I,
EVENT_CATEGORY EC,EVENT E WHERE
I.RNO=\"".$u."\" AND
I.CATEGORY=EC.CATEGORY AND
E.ENAME=EC.ENAME AND
E.DATE>='".$cdate."' ORDER BY E.DATE;";
	$events=$connect->query($query);
	if(!mysqli_query($connect,$query))
	{
		echo mysqli_error($connect);
	}
	echo "<table border='1'>";
	echo "<tr>";
	echo "<th>EVENT</th>";
	echo "</tr>";
	while($row=mysqli_fetch_array($events))
	{
		echo "<tr>";
		echo "<td>",
$row['DATE'],"\t",$row['ENAME'], "</td>";
		echo "</tr>";
	}
	echo "</table>";
	mysqli_close($connect);
}
$u=$_SESSION["UNAME"];
fun($u);
session_close();
?>
</body>
</div>
</html>
```


2019-04-09

EVENT
2019-04-13 sing fest
2019-06-13 OVERLOAD++ 2k19
2019-06-14 LEIOTHRIXX

ALL.php:(for displaying all notifications)

```
<html>
<style>
.center {
  margin: 0;
  position: absolute;
  top: 10%;
  left: 10%;
  transform: translate(-30%, -30%);
}
.button {
  background-color: #009FFF;
  border: none;
  color: white;
  padding: 20px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
  font-size: 16px;
  margin: 4px 2px;
  position:absolute;
  right:20;
}
.button1 {border-radius: 50px;bottom:20;}
.button2 {border-radius: 50px;bottom:100;}
</style>
<form action="INTREST.php">
<input type="Submit" class="button button2" id="btn" value="Interest Form">
</form>
<form action="CALENDER.php">
<input type="submit" class="button button1" id="btn" value="My Calender">
</form>
<div class="center", align="center">
<?php
session_start();
```

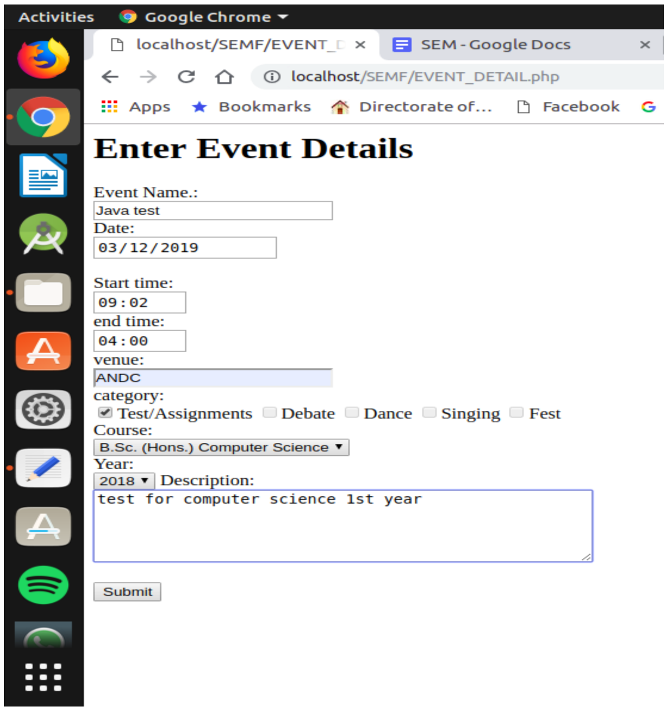
```
function fun($u)

{

    $connect = new mysqli("localhost", "root", "qwerty@asdfgh", "SEM");

    if ($connect->connect_error)

    {

        die("ERROR: Unable to connect: " . $connect->connect_error);

    }

    //RETRIEVING EVENTS BELONGING TO INTERESTED CATEGORIES

    $cdate=date("Y/m/d");

    $query="SELECT DISTINCT EC.ENAME,E.DATE FROM EVENT_CATEGORY EC,EVENT E, CATEGORIES C WHERE E.ENAME=EC.ENAME AND EC.CATEGORY=C.CATEGORY AND C.COURSE=FALSE AND E.DATE>='".$cdate."' ORDER BY E.DATE;";

    $events=$connect->query($query);

    if(!mysqli_query($connect,$query))

    {

        echo mysqli_error($connect);

    }

    echo "<table border='1'>";

    echo "<tr>";

    echo "<th>EVENT</th>";

    echo "</tr>";

    while($row=mysqli_fetch_array($events))

    {

        echo "<tr>";

        echo "<td>", $row['DATE'],"\t",$row['ENAME'], "</td>";

        echo "</tr>";

    }

    echo "</table>";

    mysqli_close($connect);

}

$u=$_SESSION["UNAME"];

fun($u);

session_close();

?>

</body>

</div>

</html>
```

EVENT_DETAIL.php:

```
<html>

<body>

<h1>Enter Event Details</h1>

<form method="POST" action="EVENT_DETAIL.php">

Event Name.: <br>

<input type="text" name="Eventname" align=left required>

<br>

Date: <br>

<input type="date" name="date" align=right required>

<br><br>

Start time:<br>

<input type="time" name="stime" required>

<br>

end time:<br>

<input type="time" name="etime" required>

<br>

venue: <br>

<input type="text" name="venue" required>

<br>

category:<br>

<input type="checkbox" name="category[]" value="Test/Assignments" oninput="handleSelect()"
```

```
id='ch1'>Test/Assignments

<input type="checkbox" name="category[]" value="Debate" id='ch2' >Debate

<input type="checkbox" name="category[]" value="Dance" id='ch3'  >Dance

<input type="checkbox" name="category[]" value="Singing" id='ch4'>Singing

<input type="checkbox" name="category[]" value="FEST" id='ch5'>Fest

<br>

Course:<br>

<select name="Course" id="cid" disabled>

<option value="BSc(H)CS">B.Sc. (Hons.) Computer Science</option>

<option value="BSc(H)C">B.Sc. (Hons.) Chemistry</option>

<option value="BSc(H)M">B.Sc. (Hons.) Mathematics</option>

<option value="BSc(H)P">B.Sc. (Hons.) Physics</option>

</select><br>

Year:<br>

<select name="Year" id="yid" disabled>

  <option value="2016">2016</option>

  <option value="2017">2017</option>

  <option value="2018">2018</option>

  <option value="2019">2019</option>

</select>

<script>

        function handleSelect()

        {

  var checking=document.getElementById('ch1');

        if(checking.checked){
document.getElementById('cid').disabled=false;
document.getElementById('yid').disabled=false;
document.getElementById('ch2').disabled=true;
document.getElementById('ch3').disabled=true;
document.getElementById('ch4').disabled=true;
document.getElementById('ch5').disabled=true;

                }else{
document.getElementById('cid').disabled=true;

        document.getElementById('yid').disabled=true;

document.getElementById('ch2').disabled=false;
document.getElementById('ch3').disabled=false;
document.getElementById('ch4').disabled=false;
document.getElementById('ch5').disabled=true;

                }

        return;

        }

</script>

Description: <br>

<textarea maxlength="200" rows=5 cols=50 wrap=hard name="desc"></textarea>

<br>

<br>

<input type="submit" name="submit" value="Submit">

</form>

        <?php

        if(isset($_POST["submit"]))

        {

        $connect = new mysqli("localhost", "root", "qwerty@asdfgh", "SEM");

        if ($connect->connect_error)

        {

        die("ERROR: Unable to connect: " . $connect->connect_error);

        }

        $query="INSERT INTO EVENT VALUES(\"".$_POST["Eventname"]."\",\"".$_POST["date"]."\",\"".$_POST["stime"]."\",\"".$_POST["etime"]."\",\"".$_POST["venue"]."\",\"".$_POST["desc"]."\");";

        if (!mysqli_query($connect,$query))

        {

        echo("Error description: " . mysqli_error($connect));

        }

        if(isset($_POST["category"])){
```

```
foreach($_POST["category"] as
$selected)
{
if($selected=="Test/Assignment")
                    {

$query="INSERT INTO EVENT_CATEGORY
VALUES(\"".$_POST["course"]."\",\"".$_POST["Ev
entname"]."\");";

                                        if
(!mysqli_query($connect,$query))

echo("Error description: " . mysqli_error($connect));
                    }
                    else
{
   $query="INSERT INTO EVENT_CATEGORY
VALUES(\"".$selected."\",\"".$_POST["Eventname"]
."\");";
 if (!mysqli_query($connect,$query))
                         echo("Error description:
" . mysqli_error($connect));}
                    }
               }}
      ?>
      </body>
      </div>
      </html>
```


localhost/SEMF/EVENT_DETAIL.php

Enter Event Details

Event Name.: Java test
Date: 03/12/2019
Start time: 09:02
end time: 04:00
venue: ANDC
category: [✓ Test/Assignments] Debate Dance Singing Fest
Course: B.Sc. (Hons.) Computer Science
Year: 2018 Description: test for computer science 1st year

Submit

7. Testing

The Testing Phase involves an independent investigation conducted to provide an unbiased view of the software quality and also appreciation of the underlying risks. Testing is the process of executing a program with the intent of finding errors. A Test Case is a combination of inputs and expected outputs. The outputs from the executed program are then compared with the expected outputs to uncover errors and detect anomaly.

Testing is often referred to as “verification and validation”. Verification refers to the set of activities that ensures that software correctly implements a specific function. Validation refers to a different set of activities that ensures the software that has been built is traceable to customer requirements.

Any engineered product can be tested using any of the following two approaches:

1. Black-Box Testing- Black-Box testing, also called behavioral testing, focuses on the functional requirement of the software. That is, Black-Box testing enables the software engineer to derive a set of input conditions that will fully exercise all functional requirements of a program. It attempts to find errors in the following categories:

a. Incorrect or missing functions

b. Interface errors

c. Errors in data structures or external database access

d. Behavior or performance errors

e. Initialization and termination errors

2. White-Box Testing- White-Box testing, also called glass-box testing, is a test case design philosophy that uses the control structure described as part of component-level design to derive test cases. Using these methods, the software engineer can derive test cases that:

a. Guarantee that all independent paths within a module have been exercised at least once

b. Execute all logical decisions on their true and false sides

c. Execute all loops at their boundaries and within their operational bounds

d. Exercise internal data structures to ensure their validity

Registration:

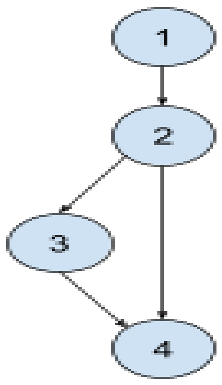


fig.8:FLOW GRAPH OF REGISTRATION MODULE

No.of independent paths:2

1-2-3-4

1-2-4

Predicate nodes=1

Regions=2

Cyclomatic complexity=2

Input	**Description**	**Expected**	**Actual**	**Result**
Name|Roll_no |Password=Null	Name=”abc” roll_no=”123” password=”” other values are filled	values should not be submitted and error massage should be displayed	Error Displayed	pass
Name+Roll_no +Password=Fille d	Name=”abc” roll_no=”123” password=”” other values are filled	All values submitted and success massage should be displayed	Successfully submitted displayed	pass

TABLE.6:TEST CASES FOR REGISTRATION

Login:

No. of independent paths:2

1-2-3-5

1-2-4-5

Predicated nodes=1

Regions=2

Cyclomatic complexity=2

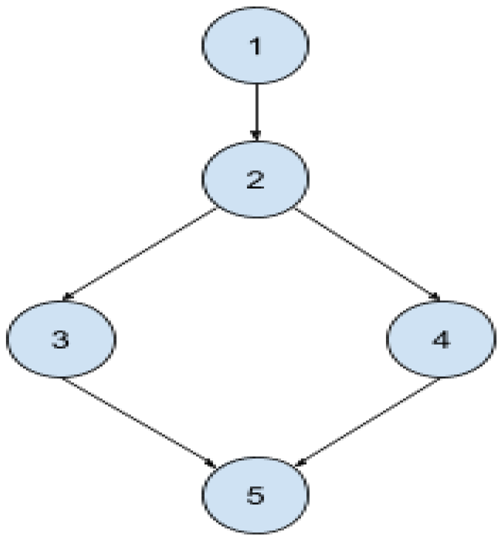


fig.9:FLOW GRAPH OF LOGIN MODULE

Input	**Expected**	**Actual**	**Result**
Username=”abc” Password=”abc”	user directed to home page	user directed to home page	pass
Username=”abc” Password=”abd”	massage for wrong username or password	massage displayed	pass

TABLE.7:TEST CASES FOR LOGIN

Input Event details:

No. Of independent paths: 2

1-2-3-6

1-2-4-5-6

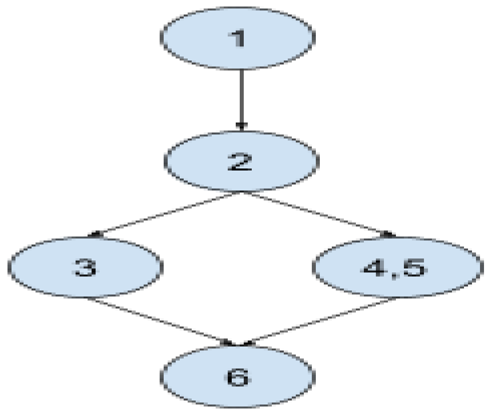


fig.10:FLOW GRAPH OF INPUT EVENT DETAILS

Cyclomatic complexity=2

Inputs	**Description**	**Expected**	**Result**
Category=test/ assignment	all values filled with category =test/assignment	Course and year field enabled	Pass
Category!=test/ assignment	all values filled with category!= test/assignment	Course and year fields disabled	Pass

TABLE.8:TEST CASES FOR INPUT EVENT DETAILS

Interest based notification:

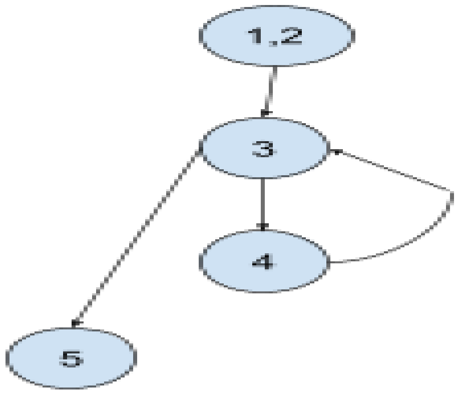


fig.11:FLOW GRAPH OF INTEREST BASED NOTIFICATIONS

No. of independent paths=2

1-2-3-5

1-2-3-4-5-3

Predicate nodes=2, No. of regions=2

Cyclomatic complexity=2

Inputs	**Description**	**Expected**	**Actual**	**Result**
no event retrieved from database	user doesnt have any interested category or notification belonging to interested category	nothing displayed	nothing displayed	pass
some events retrieved from databases	user have any interested category or notification belonging to interested category	should display all notification after date	all displayed	pass

ALL notification:

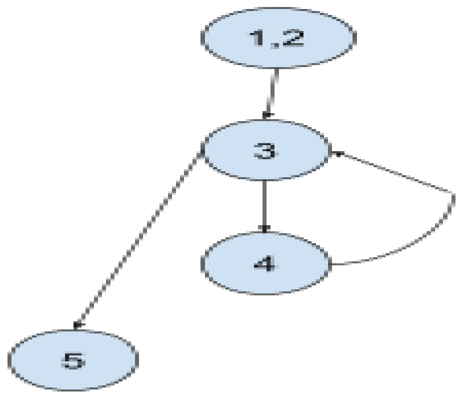


fig.11:FLOW GRAPH OF INTEREST BASED NOTIFICATIONS

No. of independent paths=2

1-2-3-5

1-2-3-4-5-3

Predicate nodes=2

No. of regions=2

Cyclomatic complexity=2

Inputs	Description	Expected	Actual	Result
no event retrieved from database	notification doesn’t present for some events	nothing displayed	nothing displayed	pass
some events retrieved from databases	notification present for some events	should display all notification after date	all displayed	pass

8. References

1. Software Engineering- A Practitioner's Approach by Roger S. Pressman: 6 th Edition McGraw Hill, 2005

2. An Integrated Approach to Software Engineering by Pankaj Jalote: 3 rd Edition Springer, 2005

3. PHP REFERENCE:Beginner to Intermediate PHP5:Mario lurig

4.HTML AND CSS design and building websites: John Duckett