AUTOMATED LAB ASSESSMENT SYSTEM

A PROJECT WORK
Submitted in partial fulfillment of requirements for the award of the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE AND ENGINEERING
by

K. JAYANTHI
(13331A0545)

S. SWATHI
(14335A0519)

K. B.VAISHNAVI
(13331A0563)

B.DURGA PRASAD
(13331A0512)

Under the esteemed guidance of -
B. S. Vamsi Krishna M. Tech., (Ph.D.)
Sr. Assistant Professor
Computer Science and Engineering Department
M. V. G. R. College of Engineering

Department of Computer Science and Engineering
MAHARAJ VIJAYARAM GAJAPATHIRAJ COLLEGE OF ENGINEERING
(Affiliated to Jawaharlal Nehru Technological University, Kakinada)
VIZIANAGARAM
2013 – 2017

MAHARAJ VIJAYARAM GAJAPATHIRAJ COLLEGE OF ENGINEERING VIZIANAGARAM

BONAFIDE CERTIFICATE

Certified that this is a bonafide record of the project work entitled **"AUTOMATION OF LAB ASSESSMENT SYSTEM"** being submitted by Ms. K. Jayanthi, Ms. S. Swathi, Ms. K. B. Vaishnavi, Mr. B. Durga Prasad, bearing Registration numbers 13331A0545, 14335A0519, 13331A0563, 13331A0512 respectively in partial fulfillment for the award of the degree of **"Bachelor of Technology"** in Computer Science and Engineering during the academic year 2016-2017.

B. S. Vamsi Krishna
MTech, (Ph.D.)
Sr. Assistant Professor
Dept. of Computer Science Engg,
MVGR College of Engineering (Autonomous),
Vizianagaram.

Prof. P. Sitharama Raju
Professor, Head of the Department
Dept. of Computer Science Engg,
MVGR College of Engineering
(Autonomous)
Vizianagaram.

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

As a note of acknowledgement, with great solemnity and sincerity, we offer our profuse thanks to **Mr. B. S. Vamsi Krishna**, Sr. Assistant Professor, Dept. of CSE, for guiding us all through our project work, giving a right direction and shape to our learning by extending his expertise and experience in the field of education. Really, we are indebted to him for his excellent and enlightened guidance.

We consider it our privilege to express our deepest gratitude to **Prof. P. Sitharama Raju**, Professor and Head of the Department for his valuable suggestions and constant motivation that greatly helped the project work to get successfully completed.

We also thank **Dr. K. V. L. Raju**, Principal, for extending his utmost support and cooperation in providing all the provisions for the successful completion of the project.

We sincerely thank all the members of the staff in the Department of Computer Science & Engineering for their sustained help in our pursuits.

We thank all those who contributed directly or indirectly in successfully carrying out this work.

K. Jayanthi(13331A0545)
S. Swathi(14335A0519)
K. B. Vaishnavi(13331A0563)
B. Durga Prasad(13331A0512)

ABSTRACT

A Daily Assessment procedure in a laboratory helps a faculty to assess a student's performance in that particular lab. A laboratory session usually involves few activities like recording attendance, giving questions, checking outputs, conducting viva, updating marks, answering queries and many more. Handling all these activities often creates a burden to the faculty and he/she might miss one or the other activity due to the time lapse and hardly finds time to concentrate on improving student's performance. So, there is a need for automating the lab assessment system. This system makes faculty's job easier by automating all the above activities. She can conduct viva and even handle discussion which reduces the unwanted dissonances. This allows the faculty to effectively evaluate the student's performance in comparatively less time and with minimal efforts. This leaves the faculty with an ample time to interact with the students with lower grades. Finally, he/she can print the report of the lab which includes all the student's details and marks including the total marks calculated by the system. This in-turn helps the faculty in calculating the marks for internal examination.

CONTENTS

1.INTRODUCTION………………………………………………………..1

1.1 NEED OF THE PROJECT………………………………………...2

1.2 EXISTING SYSTEM AND PROPOSED SYSTEM……………….....3

2. SYSTEM STUDY AND ANALSIS

2.1 USER REQUIREMENT……………………………………………..5

2.2 FUNCTIONAL REQUIREMENTS………………..……………..6

2.3 NON-FUNCTIONAL REQUIREMENTS………………….…………8

2.4 DIAGRAMS IN UML ………………………………………….…..8

2.4.1 USECASE DIAGRAM……………………………...…………9

2.4.2 SEQUENCE DIAGRAM…………………………………….11

2.4.3 ENTITY RELATIONSHIP DIAGRAM……….………….13

2.5 SOFTWARE AND HARDWARE REQUIREMENTS

2.5.1 SOFTWARE REQUIREMENTS……………………………14

2.5.2 HARDWARE REQUIREMENTS…………………...……14

3. SYSTEM DESIGN

3.1 IMPLEMENTATION DETAILS…………………………….………16

3.2 SCREENSHOTS…………………………………………………21

3.3 DATABASES AND TABLES USED IN THE SYSTEM………....31

4. UNDERLYING TECHNOLOGIES IN THE PROJECT

4.1 DOMAIN KNOWLEDGE……………………………………....33

4.2 HTML ………………………………………………………...…34

4.3 DATABASE…………………………………………………..35

4.4 CASCADING STYLE SHEET…………………………….....37

4.5 SQL………………………………………………………….....37

4.6 JAVASCRIPT……………………………………………….....39

4.7 PHP……………………………………………………………….....39

5. TESTING

5.1 TESTING PLAN……………………………………………………41

5.2 TESTING STRATEGIES…………………………………………..41

5.3 TEST CASES REPORT…………………………………...…….43

6. CONCLUSION ………………………………………..…………45

BIBILOGRAPHY

APPENDIX-A – REFERENCES………………………………………46

APPENDIX-B – CODE……………………………...……………47

Chapter 1

INTRODUCTION

Automated Lab Assessment System is a client-server based application. It is designed to automate three major activities in the laboratory session: Attendance, Assessment, Student-Faculty discussion. In this particular system, there are two users: Laboratory in-charge/Faculty and Student. A faculty registers to the system with his data. He/she later logs in into the system and must not log-out till the end of lab-session. Once logged in he/she must select the details of the laboratory session and add a new lab-session. Immediately, the system generates a unique id - “lab_id”. He/she can view a lab session which is added to the database and announce the lab_id to the students so that they can register to the system under that lab-session. Once students get registered, the in-charge can get the list of students registered and their details in the Attendance window. The student has to log-in to continue the lab-session.

In order to begin the assessments, the faculty has to choose Assessment in the menu bar. The lab in-charge has to upload the file (text/doc/pdf) with questions into the database. Once done, the students can download this file and start the assessment and upload their answers file (which includes the program and the respective output) to the database by the end of the lab-session. All the files sent by the students are listed on the faculty’s screen along with the description of the file. These files can be evaluated by downloading them even after the end of lab-session. This is very useful in the situations like absence of the actual in-charge and an alternate faculty is handling the lab who many not have the knowledge on the outputs. So, the in-charge can have the list of answer files in his profile and can later evaluate them.

By clicking on Viva on the menu bar, the faculty can conduct a viva session. The viva questions are multiple-choice questions, to be entered by the in-charge. He/she can view the questions while updating. After updating all the questions, a selected number of question are sent to the students once clicked on send button. As there is no timer, it is advisable to send the questions at the end of lab-session before the lab in-charge logs out to prevent any mal-practices. The student receives the questions with a text box to enter the option and submits them once answered. The system automatically calculates the marks and shows it on screen. Every student gets random questions which can hardly match.

The lab in-charge has a facility to look at individual marks by clicking on Marks in the menu bar. This mainly includes three types of marks: 1) Execution marks 2) Viva marks 3) Record marks. The execution marks can be updated based on evaluation of the assessment answer files uploaded by the students. The viva session marks are updated by the system automatically. Records are to be manually checked and marks are updated. Once all the marks are updated, the system calculates the individual total and updates it for the particular week. There is a facility to even reset the marks. The students who failed to attend the lab-session will get zero marks. Finally, this particular mark-list can be printed by the in-charge.

There is a Discussion forum which enables the students and lab in-charge to post all the queries and interact with each other during the lab-session and answer the queries of other students. Every post will include the details of the person posted. This forum is available only till the faculty is logged in to the system. All the data is washed off if he/she log outs. This reduces noise and clumsiness in the laboratory and even helps the in-charge to assess the students based on the queries they post.

1.1 NEED FOR THE PROJECT:

In a general manual lab assessment system, a lab in-charge has too many activities to be done before the completion of the lab session. She/he has to monitor the class, register the attendance, clarify the doubts of the pupil, make assessment by visiting each and every student's system, check the records, conduct viva and record those marks. Due to this burden, there are chances for him/her to commit many mistakes, he/she may overlook the attendance, may not get a chance to clarify everyone's doubts or assess in a correct way and may need additional human power to assist her/him. So, to avoid these situations, a system has been designed, to bring all the above-mentioned activities to the fingertips of the lab in-charge.

1.2 EXISTING SYSTEM AND PROPOSED SYSTEM:

Existing System:

In the present lab assessment system, handling the information like basic individual data of student, marks, attendance etc., are done manually. The faculty has to personally conduct the viva session and post the marks. It is pretty hard to handle these data and post them into different records to save them till the end of the semester.

In the existing system,

- Attendance might be incorrectly updated.
- The lab-session questions should be read out by the in-charge and people who are late might miss them.
- Student's work is not recorded during lab session by the in-charge for future evaluation.
- Correctness in calculations cannot be guaranteed.
- The burden that has been carried out by the Faculty is high.
- This System leads to some of the conflicts in analyzing the performance of the student.
- In-charge might cause disturbance to the students while answering the common queries and sometimes may not be able to reach out students, in this case the students discuss among themselves making the class noisy.

Proposed System:

Proposed System is an application which avoids more manual time that needed to be spent on record keeping and report generating. The application keeps the data in a centralized way which is available to the faculty conducting lab session. No specific training is required to Faculty members to use this application.

This proposed system overcomes all the problems in the current system and is

- Easy to maintain all the data.
- Easy to look for all the previous data.
- Time saving.
- Can print the final marks report**.**

Chapter 2

SYSTEM STUDY AND ANALYSIS

2.1 USER REQUIREMENT:

A Software requirements specification (SRS) - a requirements specification for a system software system - is a complete description of the behavior of a system to be developed. It includes a set of use cases that describe all the interactions the users will have with the software. In addition to the use cases, the SRS also contains non-functional requirements. On being requirements which impose constraints on the design or implementation.

SOFTWARE REQUIREMENTS SPECIFICATION:

Software requirement specification (SRS) is the starting point of the software development activity. The SRS means translating the minds of the clients (the input) into a formal document (the output of the requirement phase). Thus, the output of the phases set formally specific requirements, which hopefully are consistent and complete, while the input has none of these properties. “A procedure for identifying the requirements can there for be a best set of guidelines”.

The requirement specifications phase consists of two basic activities:

1.Problem or Requirement Analysis

2.Requirement Specification

The requirement specification phase terminates with the production of the validation and software requirement specification document.
Thus, there are three major parties interested in the new system: the client, the user and the developer. There is a communication gap between these people. A basic

purpose of this SRS is bridge this communication gap. SRS is a medium through which the client and the user needs are accurately specified; indeed, the SRS forms the basis of the software development.

2.2 FUNCTIONAL REQUIREMENTS:

1. STUDENT REGISTRATION:

Input: Student details.

Processing: The faculty gives a lab_id to the students in the lab. Only then the students can fill the details and get registered to the respective lab session. The entered data are stored in the student database and are used for further processing.

Output: If all the details are entered, student gets registered and redirects to login page.

2. STUDENT LOGIN:

Input: Registered registration number and system number.

Processing: The details entered here are validated with the details entered during registration. If matched student can access the system.

Output: In case of successful login, home page appears else login page to re-login.

3. GET QUESTIONS:

Input: No input

Processing: The question file updated by the faculty is displayed along with the description.

4. SUBMIT ANSWERS:

Input: Description of the file that includes number of outputs and the file of answers named after the student's registration number.

Processing: The file is saved on the website in uploads/answers folder and the path and description in the database.

Output: Message showing "uploaded".

5. VIVA:

Input: No input

Processing: Displays n number of questions from the database with an input textbox to enter the option value.

Output: On submitting the viva marks are displayed on screen and updated in the database.

6. FORUM:

Input: A new query or answer to the existing query by faculty or student.

Processing: All the data are updated to the database.

Output: Question id and the link to question table are displayed in a tabular format.

7. ADD LAB:

Input: Lab details.

Processing: If the entered data doesn't match any previous records, a new lab session is created along with a unique lab_id and tables are created in each database based on the lab id generated.

Output: List of databases in which tables are created.

8. VIEW LAB:

Input: Details of the lab, which a faculty wants to view.

Processing: If the lab is added previously in the lab database, then the faculty is redirected to home page with various functionalities. If not an error message is displayed.

Output: Faculty home page.

9. UPDATE QUESTIONS:

Input: Enters the question, options and correct option and clicks on add.

Processing: If none of the fields is empty then the question is updated to the databases which are further sent to the students.

Output: Question added.

10. ATTENDANCE:

Input: No input

Processing: The rows in the student database are retrieved.

Output: The list of registered students is displayed on the screen.

2.3 NON-FUNCTIONAL REQUIREMENTS:

1. **Safety Requirements:**

   The database may get crashed at any certain time due to virus or operating system failure. Therefore, it is required to take database backup.

2. **Security Requirements:**

   We are going to develop a secured database. There are two different categories of users namely administrator, faculty who will be viewing the information from the database. Depending upon the category of users the access right is decided. In this case Administrator takes the control of it by registering each actor.

2.4 DIAGRAMS IN UML:

A Diagram is the graphical presentation of a set of elements, most often rendered as a connected graph of vertices (things) and arcs (relationships).
UML includes such diagrams:

- **Use-case Diagram:** Shows actors, use-cases, and the relationships between them.
- **Class Diagram:** Shows relationships between classes and pertinent information about classes themselves.
- **Object Diagram:** Shows a configuration of objects at an instant in time.
- **Interaction Diagram:** Show an interaction between a group of collaborating objects. Two types: Collaboration diagram and sequence diagram

- **Entity-Relationship Diagram:** An entity relationship diagram(ERD) shows the relationships of entity sets stored in a database.
- **State Diagram:** Describes behavior of instances of a class in terms of states, stimuli, and transitions.
- **Activity Diagram:** Very similar to a flow chart—shows actions and decision points, but with the ability to accommodate concurrency.
- **Deployment Diagram:** Shows configuration of hardware and software in a distributed system.

2.4.1 USECASE DIAGRAM:

The purpose of use case diagram is to capture the dynamic aspect of a system. But this definition is too generic to describe the purpose. Because other four diagrams (activity, sequence, collaboration and State chart) are also having the same purpose. So,

we will look into some specific purpose which will distinguish it from other four diagrams. Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. So, when a system is analyzed to gather its functionalities use cases are prepared and actors are identified.

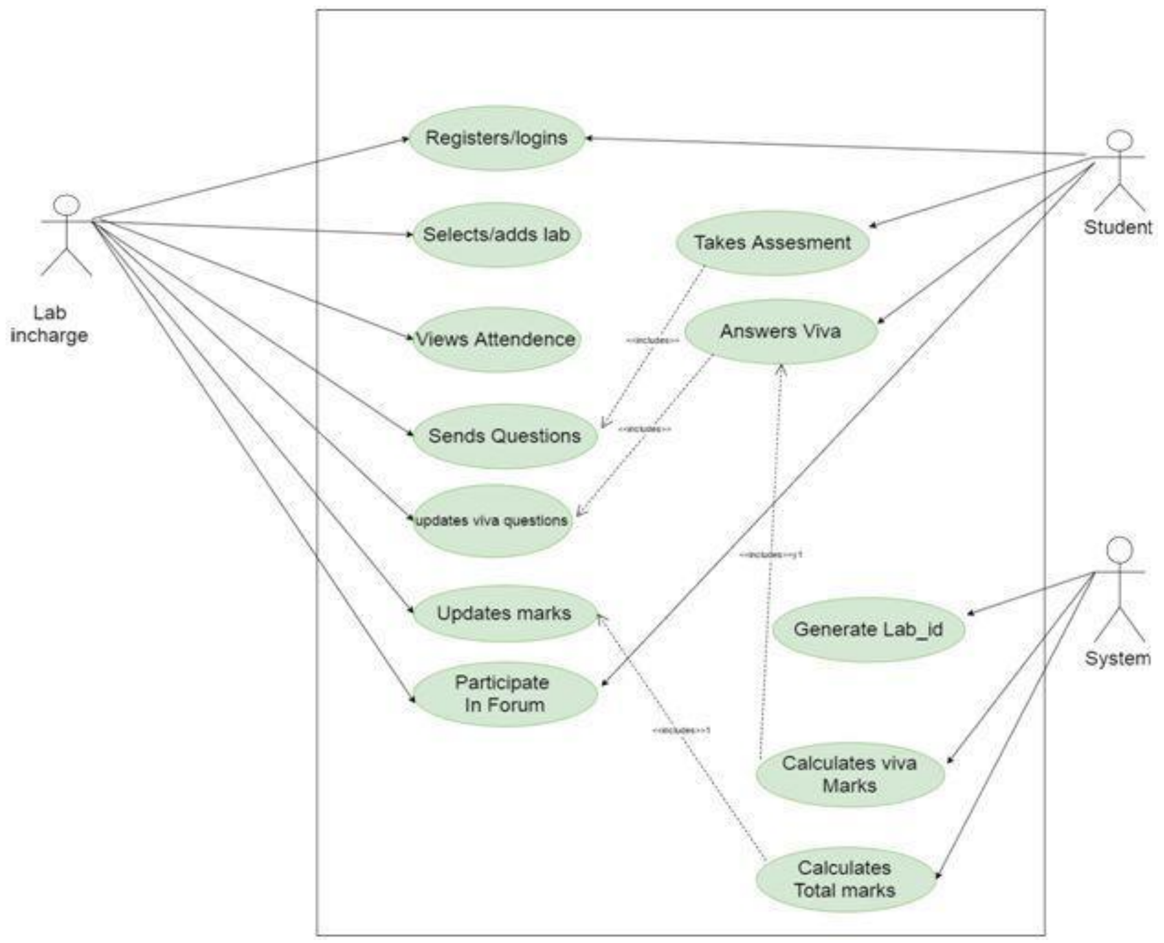


Fig 2.1 Use case diagram

Description:

In the above use case diagram the primary actors are student, lab in-charge. The automated lab assessment system is the secondary actor. The use cases are described in the following manner:

Lab in-charge:

- ➔ registers/logins into the system.
- ➔ selects a previous lab or adds new lab to the database. ØViews the attendance.
- ➔ Uploads the question file.
- ➔ Evaluates the answers received.
- ➔ Enters viva questions and sends them.

- ➔ Enters record and execution marks into the database. ØViews the complete mark's list.
- ➔ Participates in the query forum.

Student:

- ➔ Registers to the system.
- ➔ Starts an assessment
- ➔ Answers viva and views marks
- ➔ Participates in the forum.

System:

- ➔ Authenticates the faculty and student login.
- ➔ Sends random viva questions to the students.
- ➔ Calculates viva marks.
- ➔ Calculates the total marks.

<u>**Relations:**</u> The relations included in our system are:

Dependency:

- ➔ The use case calculates the total marks is dependent on updating marks by lab in-charge. Once the lab in-charge updates the record, viva and execution marks the total can be calculated.
- ➔ The use case registration of student is dependent on generation of lab_id by the system. The student can register only if he has the lab_id.
- ➔ The use case answers viva is dependent on updates viva questions.

2.4.2 SEQUENCE DIAGRAM:

The sequence diagram models the collaboration f objects based on a time sequence. It shows how the objects interact with others in a particular scenario of a use case. With the advanced visual modeling capability, you can create sequence diagram. Besides visual paradigm can generate sequence diagram from the flows of events which you have defined in the use case description.

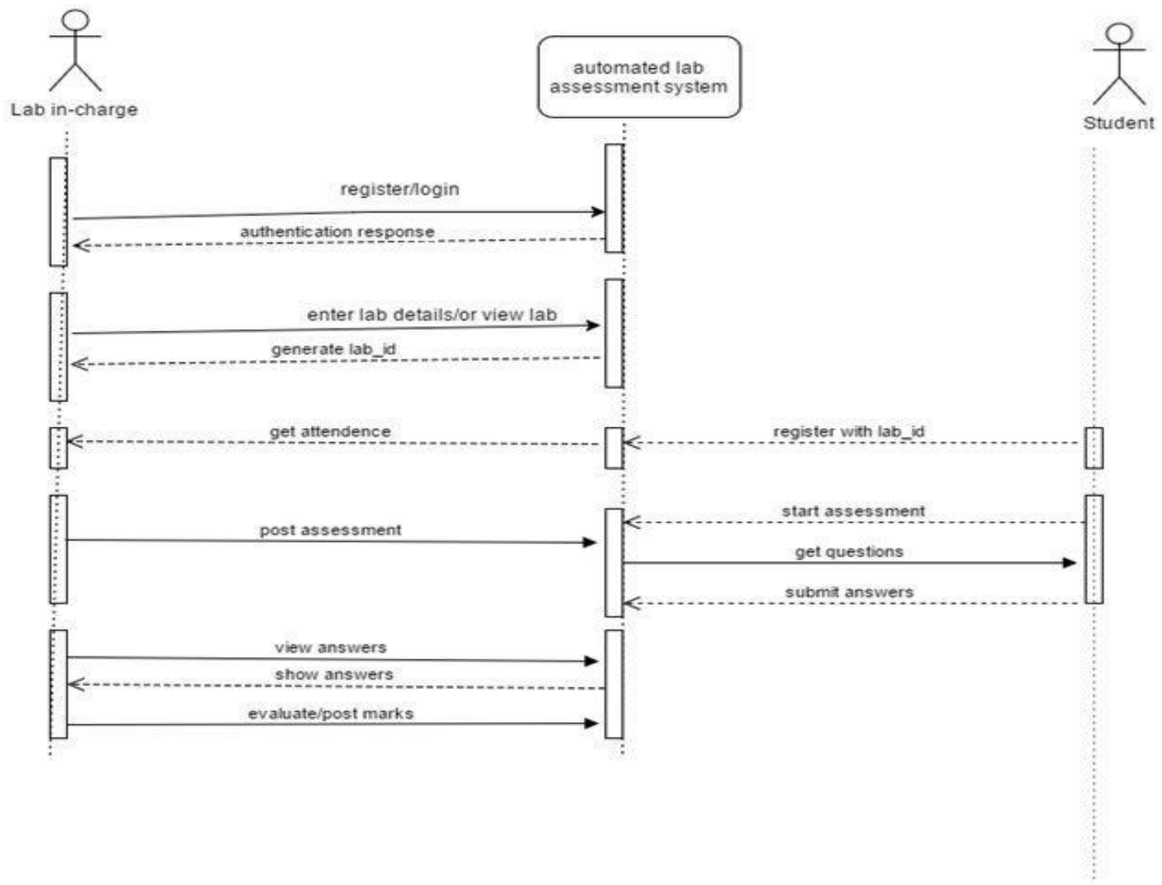


Fig:1.2.2- Sequence diagram-1

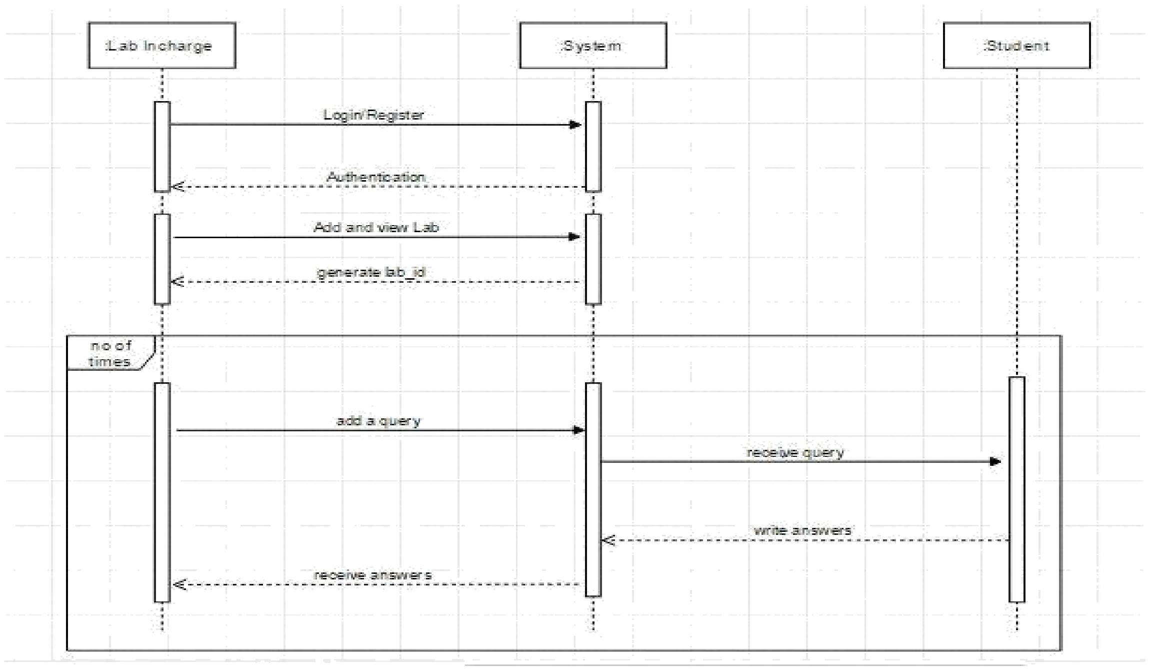


Fig:1.2.2- Sequence diagram-2

2.4.3 ENTITY-RELATION DIAGRAM:

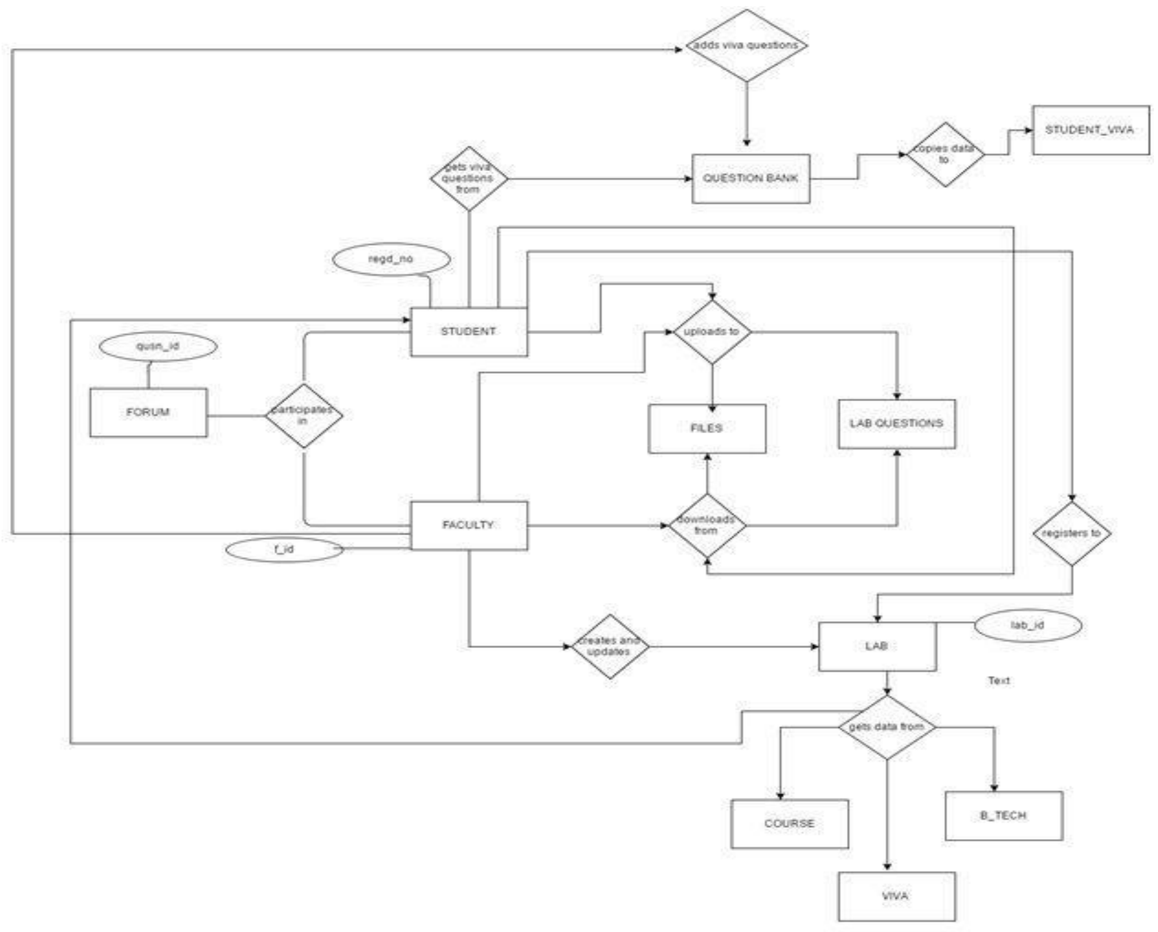


Fig:1.3- Entity-Relationship diagram

<u>Description:</u>

Databases uses:

- Student database (student_db)
- Faculty database (faculty_db)Lab database(lab_db)
- Question bank database(question_db)
- Forum database (forum_db)
- Lab Questions database (lab_questions_db)
- Files database (files_db)
- Viva database (viva)

- B_tech database
- Course database (course_db)
- Student viva database (student_viva)

2.5 SOFTWARE AND HARDWARE CONFIGURATION:

2.5.1 SOFTWARE REQUIREMENTS:

- Any windows Operating System.
- For the Database handling MYSQL must be installed.
- HTML for Front end.
- JavaScript for validations.
- The final application must be packaged in a setup program, so that system can be easily installed on the client's-machine.

2.5.2 HARDWARE REQUIREMENTS:

- Intel Pentium IV processor or equivalent or higher.
- 512 MB Ram or Higher.
- 20 GB HDD or Higher.
- Network Connectivity

Chapter 3

SYSTEM DESIGN

The most creative and challenging phase of the life cycle is system design. The term design describes a final system and the process by which it is developed. It refers to the technical specifications that will be applied in implementations the candidate system. The design may be defined as “the process of applying various techniques and principles for the purpose of defining a device, a process or a system in sufficient details to permit its physical realization”.

The designer’s goal is how the output is to be produced and in what format samples of the output and input are also presented. Second input data and database files have to be designed to meet the requirements of the proposed output. The processing phases are handled through the program Construction and Testing. Finally, details related to justification of the system and an estimate of the impact of the candidate system on the user and the organization are documented and evaluated by management as a step toward implementation.

The importance of software design can be stated in a single word “Quality”. Design provides us with representations of software that can be assessed for quality. Design is the only way that we can accurately translate a customer’s requirements into a finished software product or system without design we risk building an unstable system, that might fail it small changes are made or may be difficult to test, or one whose quality can’t be tested. So, it is an essential phase in the development of a software product.

The goal of the coding or programming phase is to translate the design of the system produced during the design phase into code in a given programming language, which can be executed by a computer and it performs computation specified by design.

3.1 IMPLEMENTATION DETAILS:

<u>FACULTY MODULE :</u>

A faculty / lab in-charge has two options: ***[faculty_main.html]***

1.Register

2. Login

<u>REGISTER</u>*: [faculty_register.html]*

Faculty can enter the system only if he is a registered user. He has to enter the details which gets updated into the faculty_table of faculty_db.

<u>LOGIN:</u> [*faculty_login.html]*

Only a registered user gets authentication to login into the system. One has to provide the same id and password while login. If login is successful, the system lead you to the home page.

The home page leaves you with two options***:[faculty_home.html]***

1. start a new lab session ***[add_lab.php].***
2. to view the previous lab details. ***[view_lab.php]***

<u>ADD LAB:</u>

If you choose to start a new lab, then fill in the details and click on "add lab*"* If there are no entries with the same details in the lab_details table of lab_db, a new lab id is generated. Lab_id is a unique id which is generated from year, semester, section, lab name, lab number. For example, 1311AHADOOP1 means '2013' batch '1-1' semester 'A' section 'HADOOP' lab number '1'. This unique id is used to create tables in databases student_db, lab_questions_db, files_db, forum_db, lab_db, question bank, viva. The new lab's details are updated in lab_details table of lab_db. Once the lab_id is generated, the faculty must share it among the students, so that they can register to the lab.

<u>VIEW LAB:</u>

If the lab is already added, you can visit the lab by using "view lab" button after filling the details***[view_lab.php].*** In the view_lab page we will have a navigation bar with 6 links.

1. Attendance
2. Assessment
3. Viva
4. Marks
5. Forum
6. Logout

<u>ATTENDANCE:</u> ***[att.php]***

This link shows the faculty, all the registered users for the lab session in a tabular form. That is the data present in the student_db is retrieved and displayed as a table.

<u>ASSESSMENT</u>: ***[assmt.php]***

In assessment, we will have two options.

1.Upload

2.Download

<u>UPLOAD:</u> ***[upload.php]***

This enables a faculty to upload his questions file. Once the faculty enters the description and sends the file, the description and path of the file is updated in lab_questions_db. The actual file is saved in the website at "assessment/questions".

<u>DOWNLOAD:</u> ***[download.php]***

This enables the faculty to download all the answers file submitted by the students in the lab. The files are saved at "assessment/answers" in the website but the path is saved at files_db.

VIVA: ***[viva.php]***

In this, there are 3 options:

1. Update questions
2. View questions
3. Send questions

UPDATE QUESTIONS: ***[update.html]***

The faculty must enter the question with 4 options and the correct option number as per the text boxes and add them to database. The questions must be adequate enough to be sent all students with less repetations. All the questions entered are stored in question_bank database with lab name and lab number as the table name.

VIEW QUESTIONS: ***[view_questions.php]***

All the questions entered are displayed on the screen to the faculty. This data is retrieved from question_bank database. This is helpful for reference of the faculty.

SEND QUESTIONS: ***[send.html]***

The faculty can enter number of questions to be sent. Once the send button is clicked, the complete table will be copied to student_viva database with lab_id as table name. Then, the questions are randomly sent to the students from student_viva database.

MARKS: ***[marks.php]***

This retrieves different data from different databases and displays to the faculty. The student regd_no and student name are retrieved from b_tech database. Based on the name system number is retrieved from student_db if the student is registered for the lab. Viva marks are automatically updated from viva_db . The execution and record marks will have text boxes for the faculty to fill-in. Once the data is entered, the details are updated to lab_db. We can also reset the details.

FORUM***: [monitor.html]***

Forum has two options:

1. Ask query
2. View queries and answer

<u>ASK QUERY:</u>

The query entered in the text area is updated in forum_db's question_list and a random question_id is generated. Now a table is created in the database based on the question_id and the question as the row head.

<u>VIEW AND ANSWER:</u>

Once the forum is refreshed, the tables in the database are viewed. Every question has a hyperlink, on clicked leads us to the table with question and answers. There will be a textarea which allows us to answer the question. On answering, the data is updated in the respective table and displayed to others.

<u>LOGOUT:</u> ***[logout.php]***

All the session variables expire and the tables in the forum_db are dropped.

<u>STUDENT MODULE :</u>

Student has two options:

1. Register
2. Login

<u>REGISTER:</u>

Student must register to the lab with the details required and lab_id provided by the faculty. All the details are inserted into student_db (table name as lab_id). Here the system_no must be unique as no two students are allotted the same system.Once the registration is successful, the student is redirected to the main page.

<u>LOGIN:</u>

Student must login with the regd_no and system _no. If successful will be accessing the home page. In the student_home.html page there is a button “start lab” which starts that day's lab session.

There are 3 options:

1.Forum

2. Daily Assessment

3. Viva

FORUM: *[forum.html]*

Forum has two options:

1. Ask query
2. View queries and answer

ASK QUERY:

The query entered in the text area is updated in forum_db’s question_list and a random question_id is generated. Now a table is created in the database based on the question_id and the question as the row head.

VIEW AND ANSWER:

Once the forum is refreshed, the tables in the database are viewed. Every question has a hyperlink, on clicked leads us to the table with question and answers. There will be a textarea which allows us to answer the question. On answering, the data is updated in the respective table and displayed to others.

DAILY ASSESSMENT:

GET QUESTIONS: *[get_questions.php]*

This field displays the question file along with the description sent by the faculty from lab_questions_db.

<u>SUBMIT ANSWERS: *[submit_ans.html]*</u>

This provides a way to upload the answers file named as student regd_no to the faculty. The file is saved to the website but the path is saved in files_db in the (lab_id as table name) table.

<u>VIVA:</u>

Ten question appear on screen with a textbox for answer. Once the questions are answered and submitted, the total marks are calculated by the system and displayed. These marks are updated to the viva database along with the regd_no. The viva hyperlink is disabled after usage of once to avoid refreshing of questions.

3.2 SCREENSHOTS:

FACULTY MODULE:

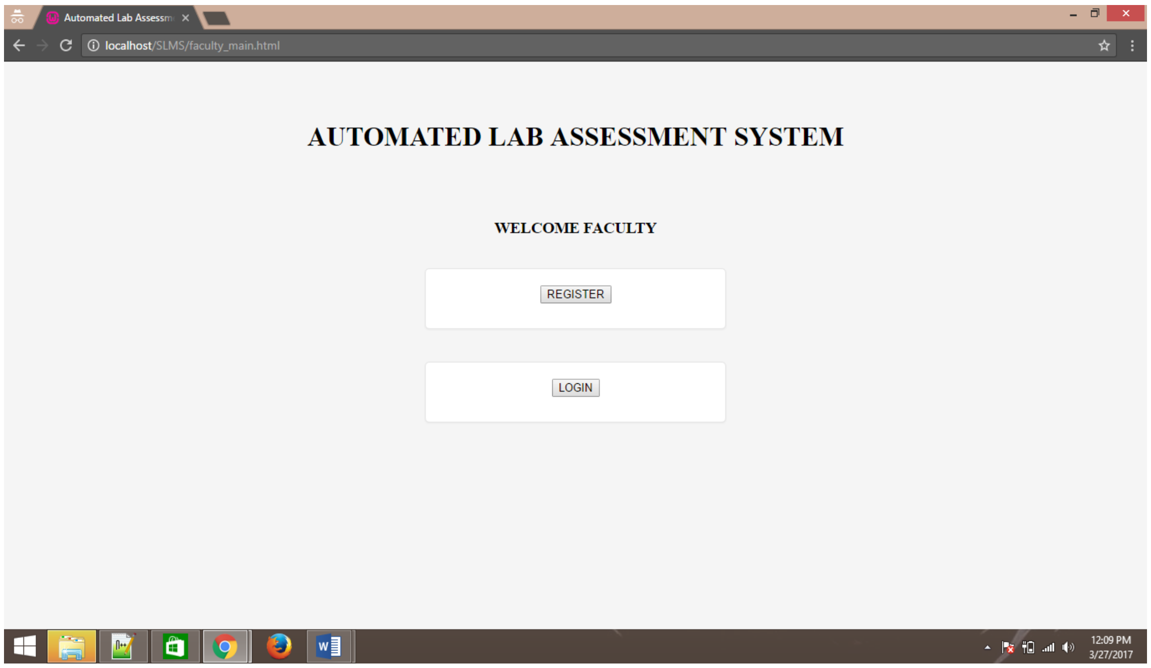


Figure-3.1 *faculty main page*

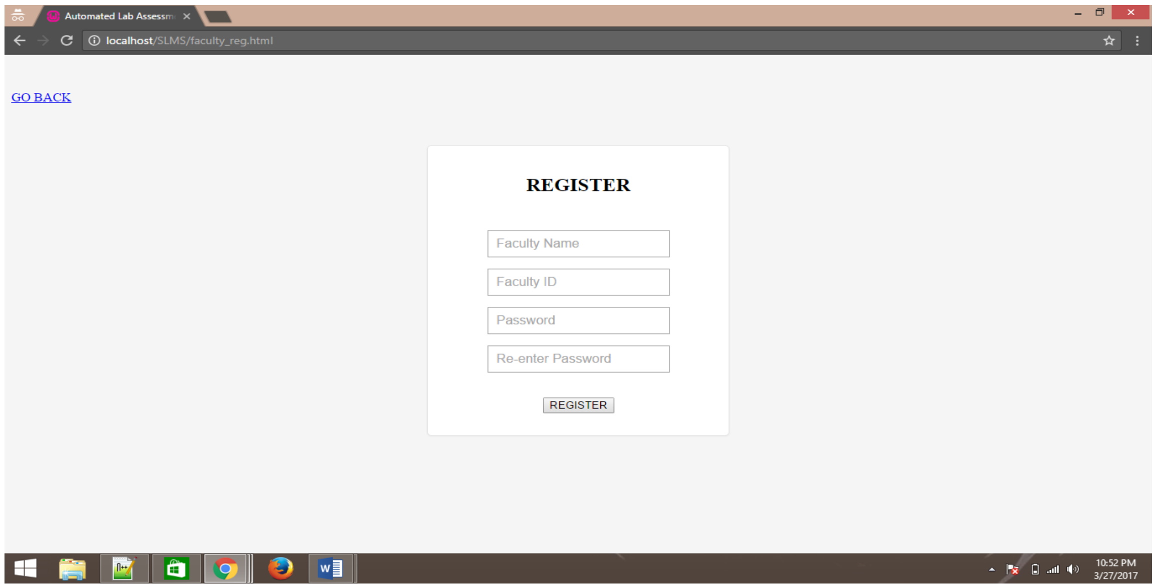


Figure-3.2 *Faculty registration page*

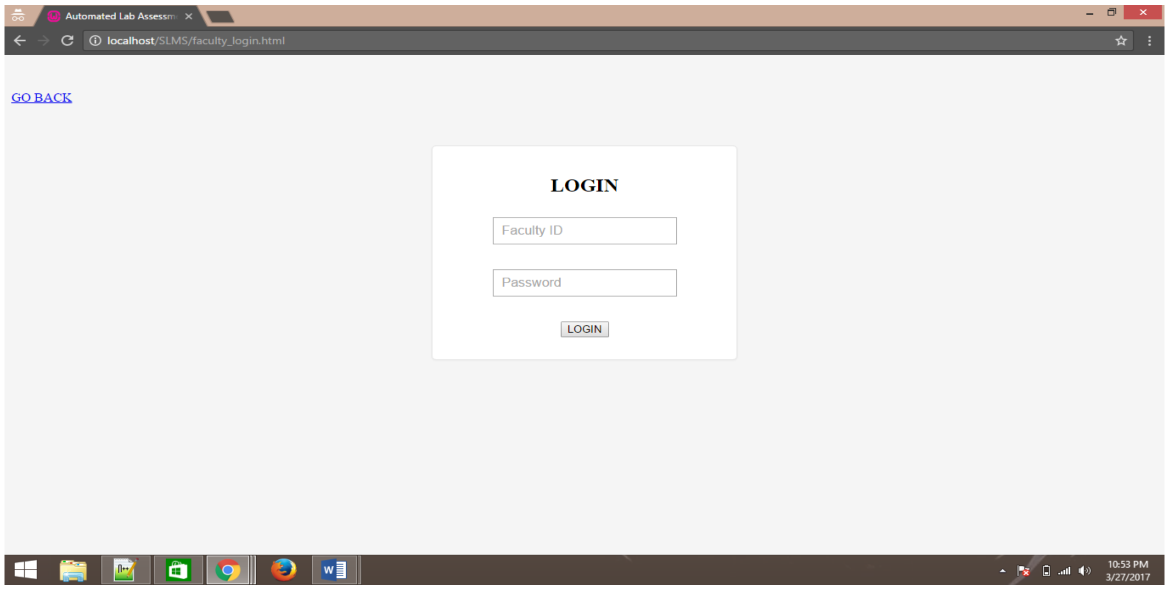
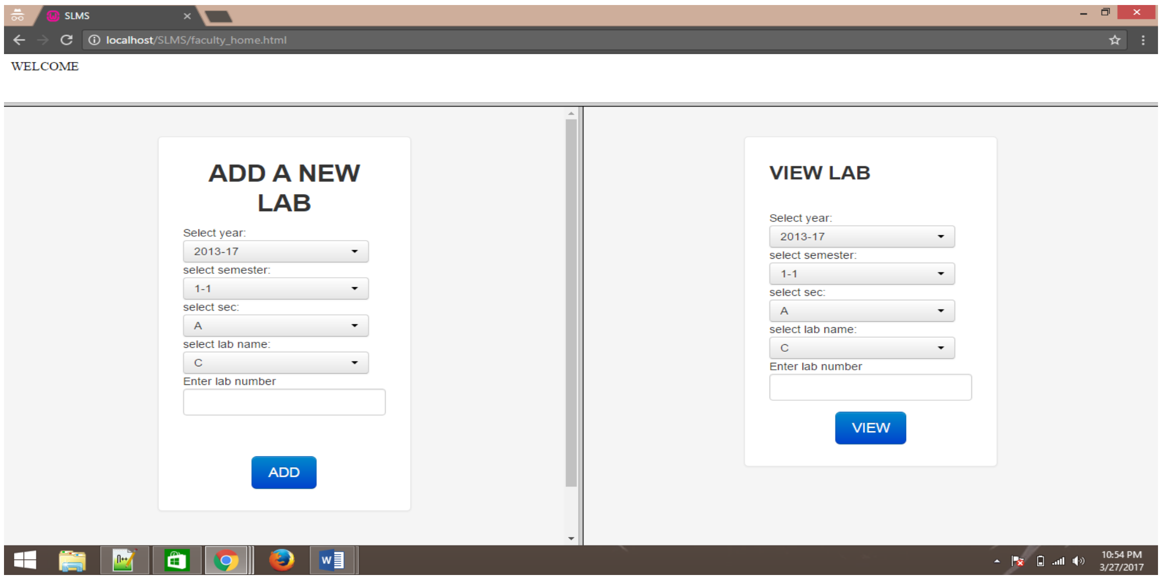


Figure-3.3 *faculty login page*



WELCOME

ADD A NEW LAB

Select year: 2013-17
select semester: 1-1
select sec: A
select lab name: C
Enter lab number

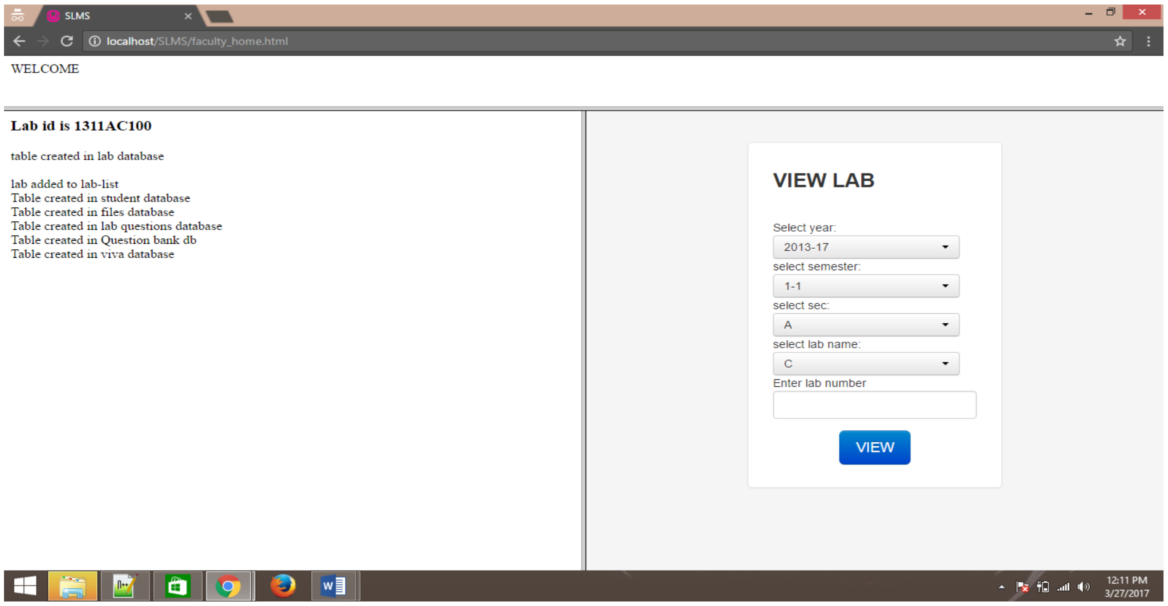
ADD

VIEW LAB

Select year: 2013-17
select semester: 1-1
select sec: A
select lab name: C
Enter lab number

VIEW

Figure-3.4 *faculty home page*



WELCOME

Lab id is 1311AC100

table created in lab database

lab added to lab-list
Table created in student database
Table created in files database
Table created in lab questions database
Table created in Question bank db
Table created in viva database

VIEW LAB

Select year: 2013-17
select semester: 1-1
select sec: A
select lab name: C
Enter lab number

VIEW

Figure-3.5 *adding a lab session*

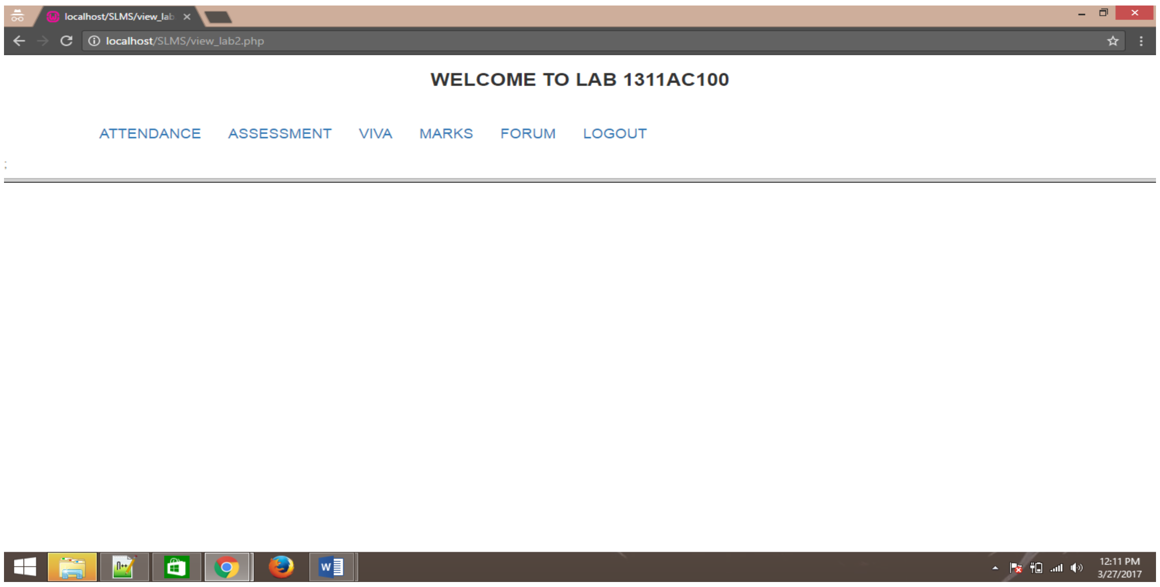
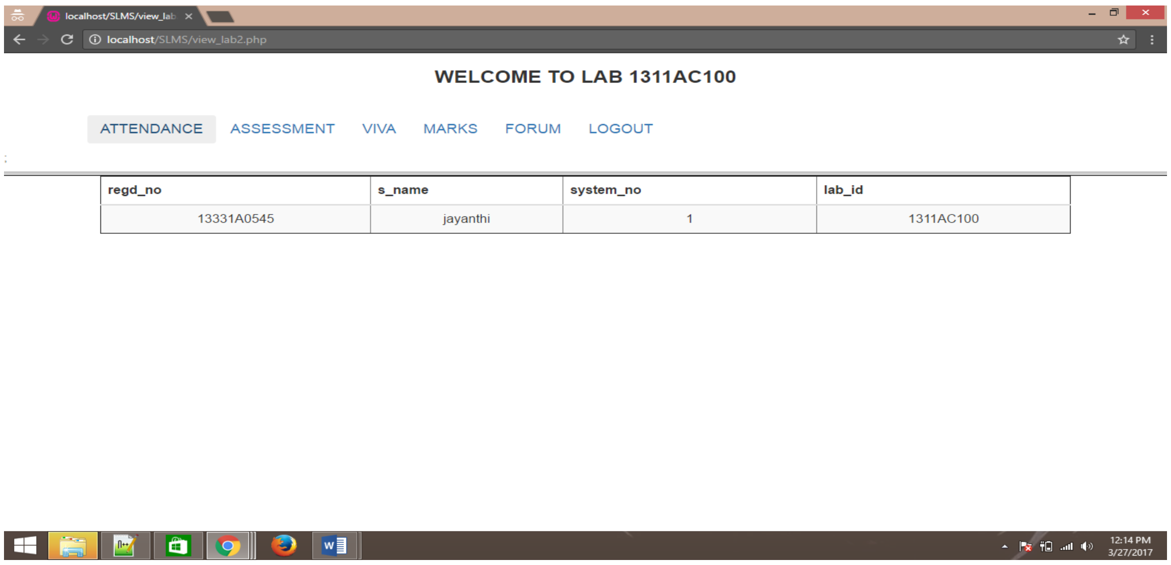
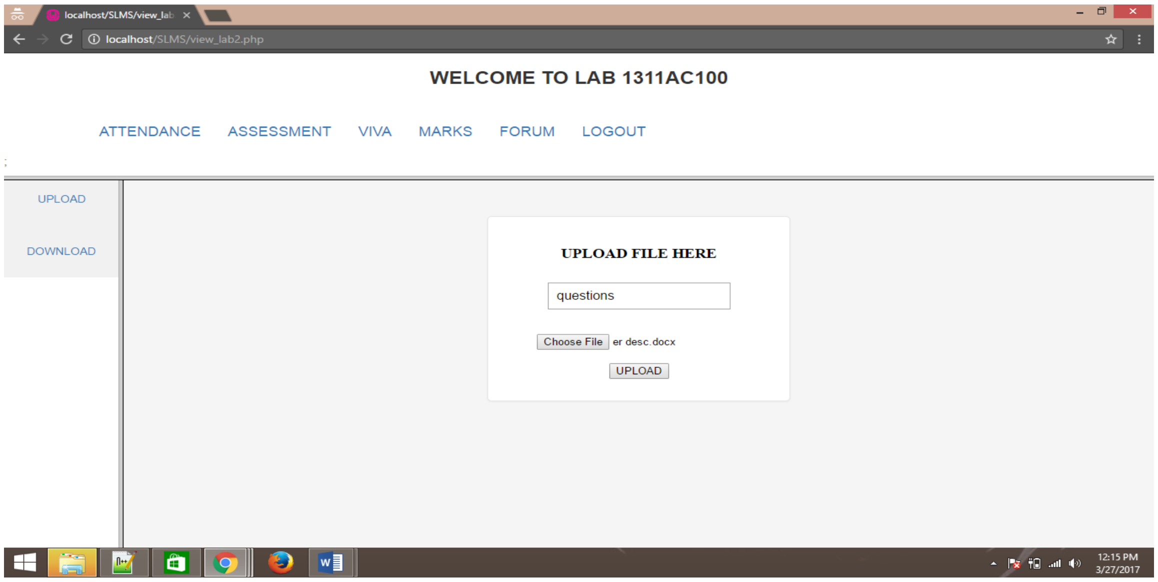


Figure-3.6 *view a lab details*



regd_no	s_name	system_no	lab_id
13331A0545	jayanthi	1	1311AC100

Figure-3.7 *attendance*


WELCOME TO LAB 1311AC100

ATTENDANCE ASSESSMENT VIVA MARKS FORUM LOGOUT

UPLOAD
DOWNLOAD

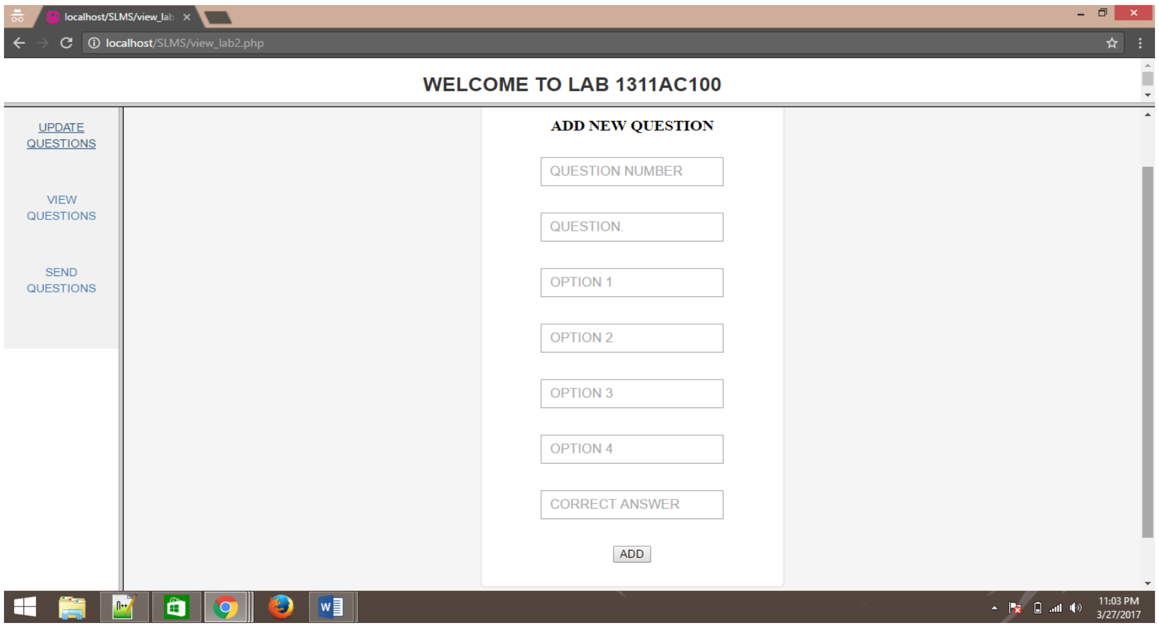
UPLOAD FILE HERE

questions

Choose File er desc.docx

UPLOAD

Figure-3.8 *uploads question file*


WELCOME TO LAB 1311AC100

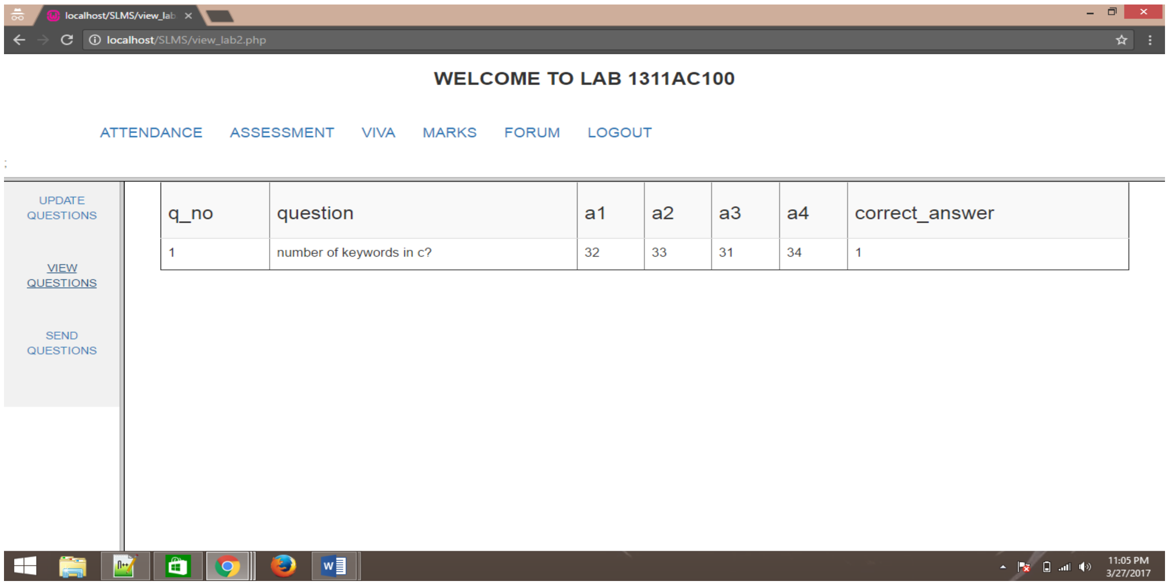
UPDATE QUESTIONS
VIEW QUESTIONS
SEND QUESTIONS

ADD NEW QUESTION

QUESTION NUMBER
QUESTION.
OPTION 1
OPTION 2
OPTION 3
OPTION 4
CORRECT ANSWER

ADD

Figure-3.9 *Updating viva questions*



q_no	question	a1	a2	a3	a4	correct_answer
1	number of keywords in c?	32	33	31	34	1

Figure-3.10 *viewing questions*

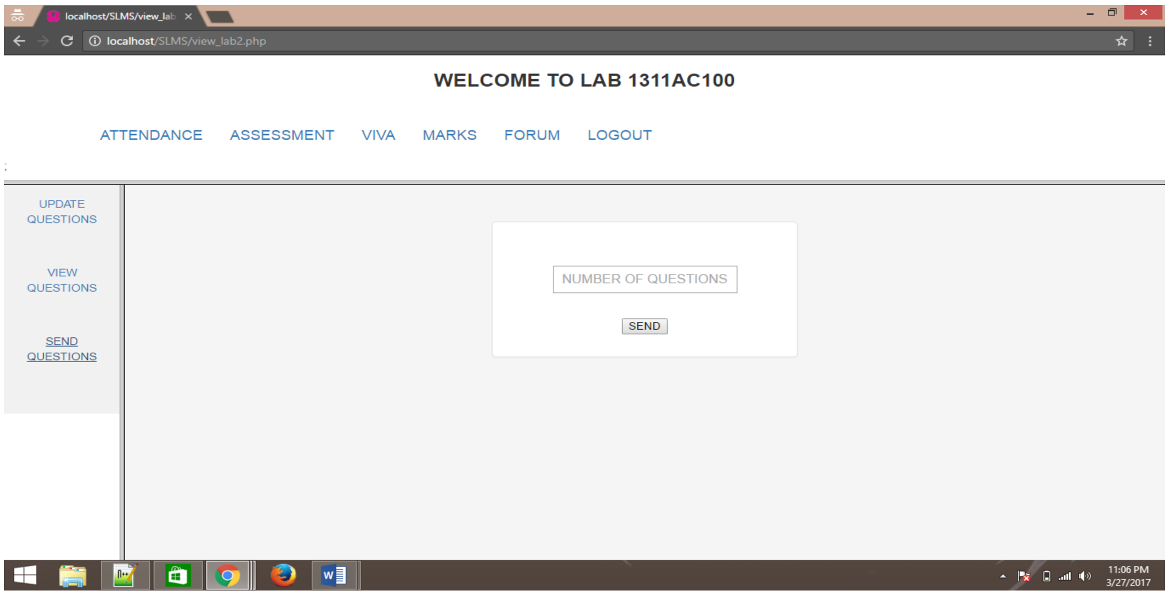
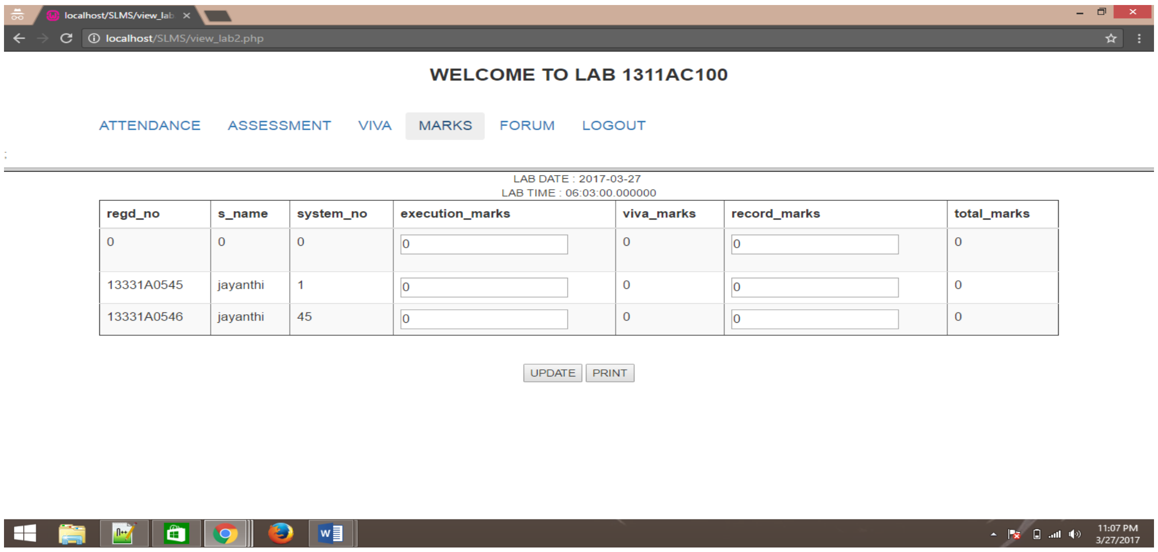


Figure-3.11 *sending questions*



localhost/SLMS/view_lab2.php

WELCOME TO LAB 1311AC100

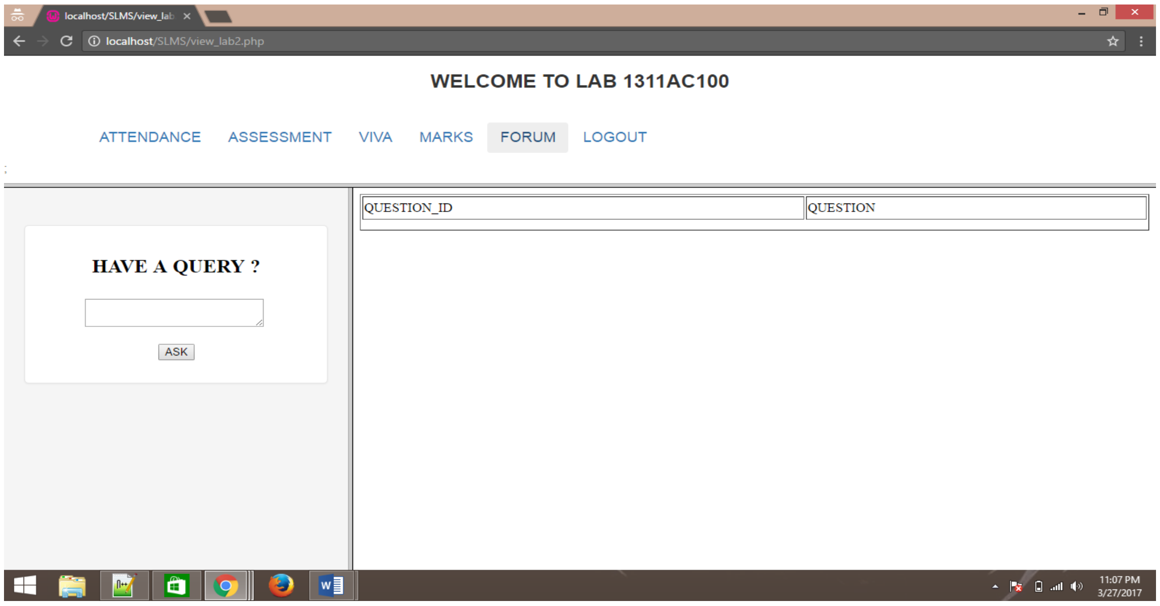
ATTENDANCE ASSESSMENT VIVA MARKS FORUM LOGOUT

LAB DATE : 2017-03-27
LAB TIME : 06:03:00.000000

regd_no	s_name	system_no	execution_marks	viva_marks	record_marks	total_marks
0	0	0	0	0	0	0
13331A0545	jayanthi	1	0	0	0	0
13331A0546	jayanthi	45	0	0	0	0

UPDATE PRINT

Figure-3.12 marks list



localhost/SLMS/view_lab2.php

WELCOME TO LAB 1311AC100

ATTENDANCE ASSESSMENT VIVA MARKS FORUM LOGOUT

HAVE A QUERY ?

ASK

QUESTION_ID	QUESTION

Figure-3.13 *Discussion forum*

STUDENT MODULE:

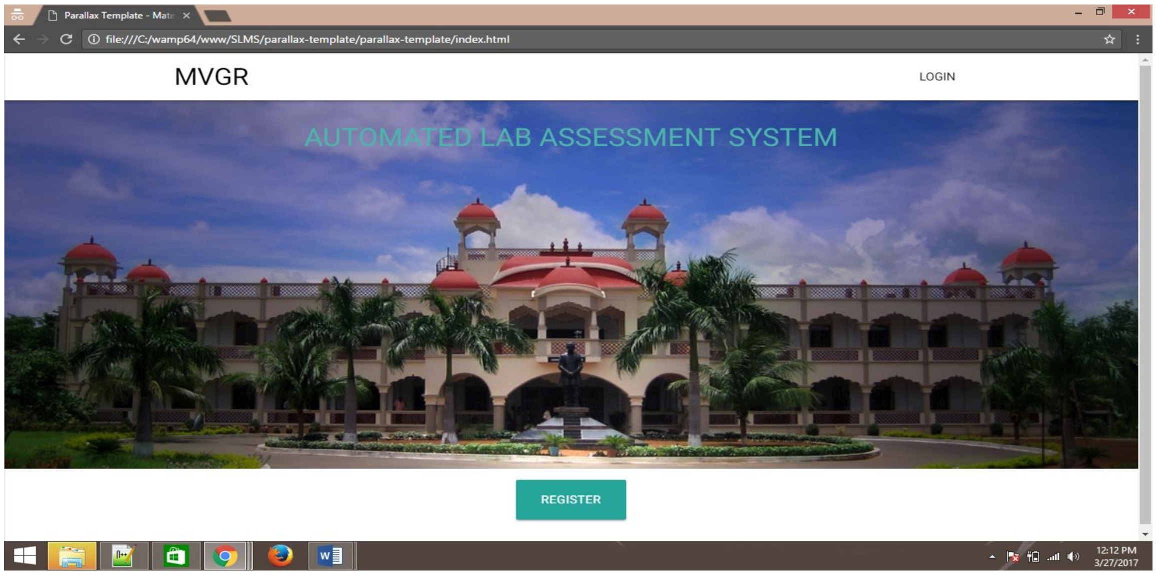

Parallax Template - Mat
file:///C:/wamp64/www/SLMS/parallax-template/parallax-template/index.html
MVGR
LOGIN
AUTOMATED LAB ASSESSMENT SYSTEM
REGISTER
12:12 PM
3/27/2017

Figure-3.14 *student main page*

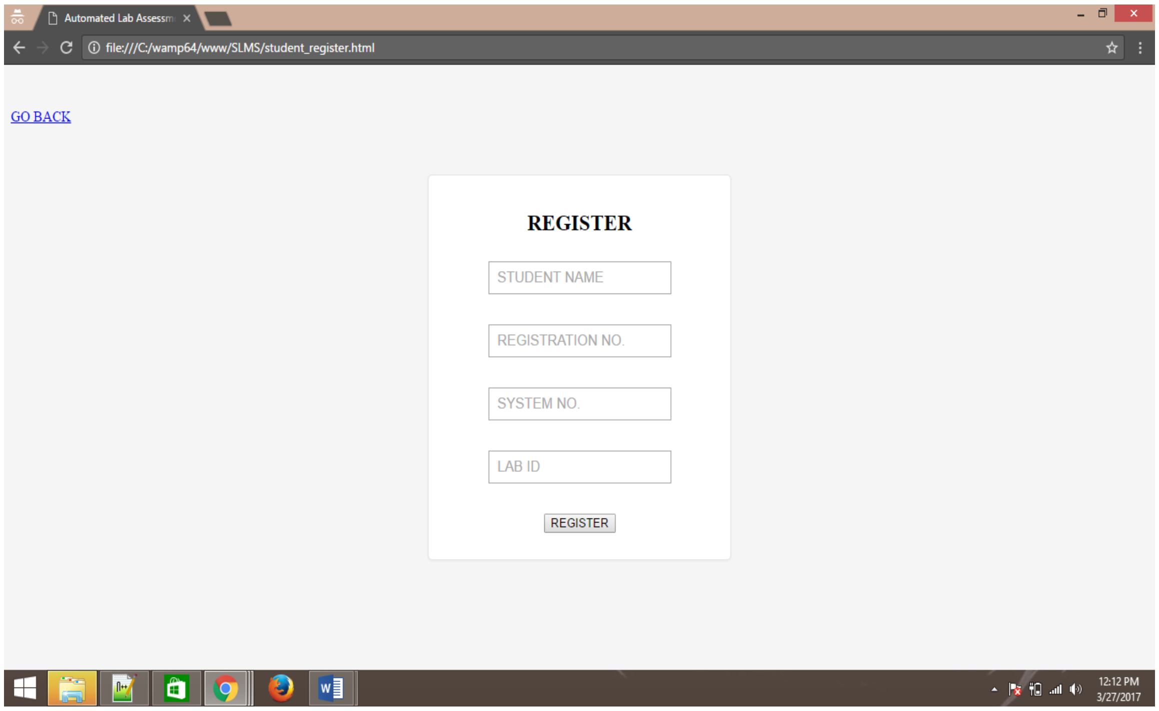


Figure-3.15 *registration page*

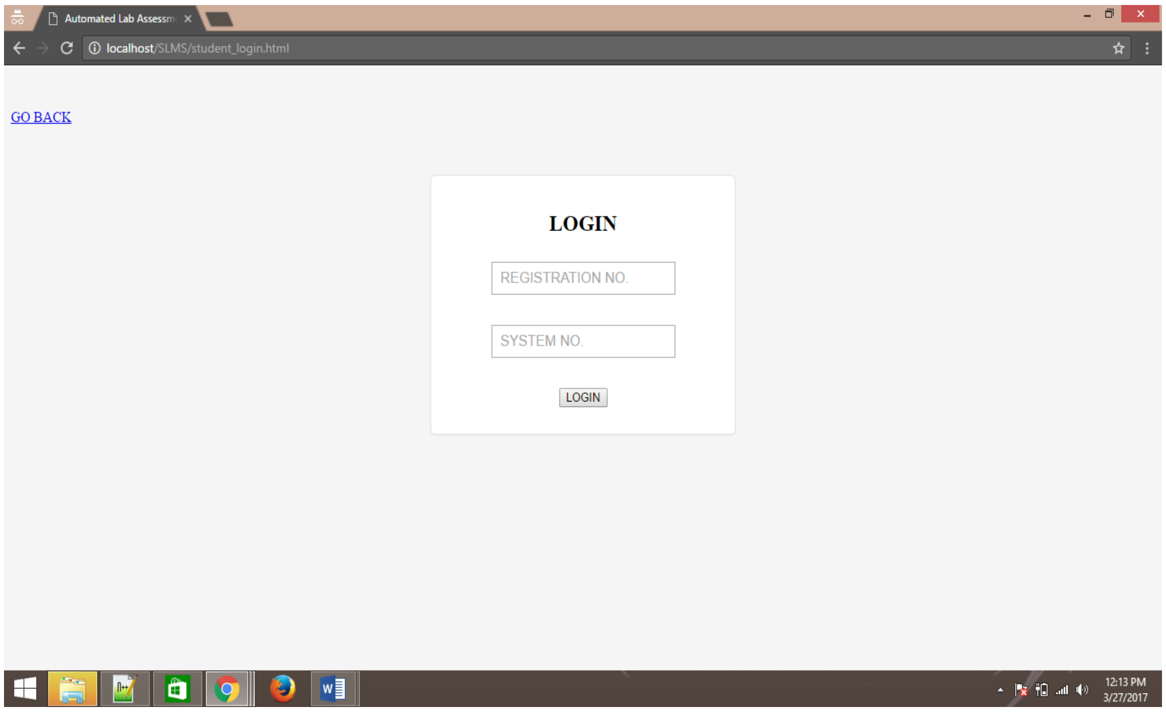


Figure-3.15 *student login page*

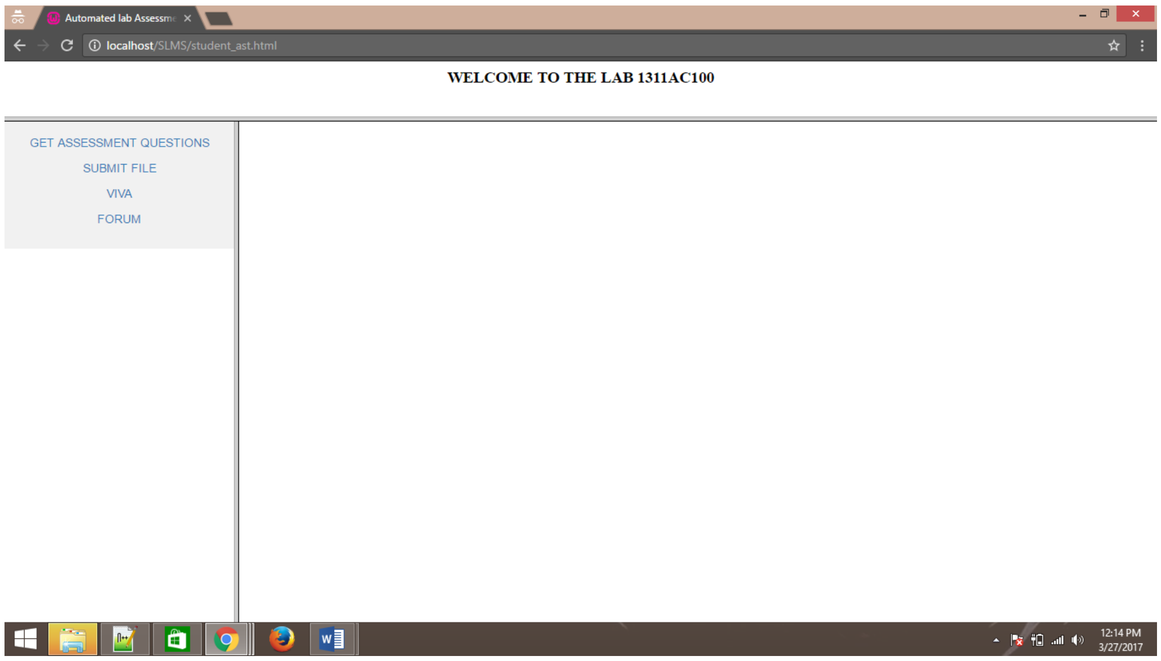
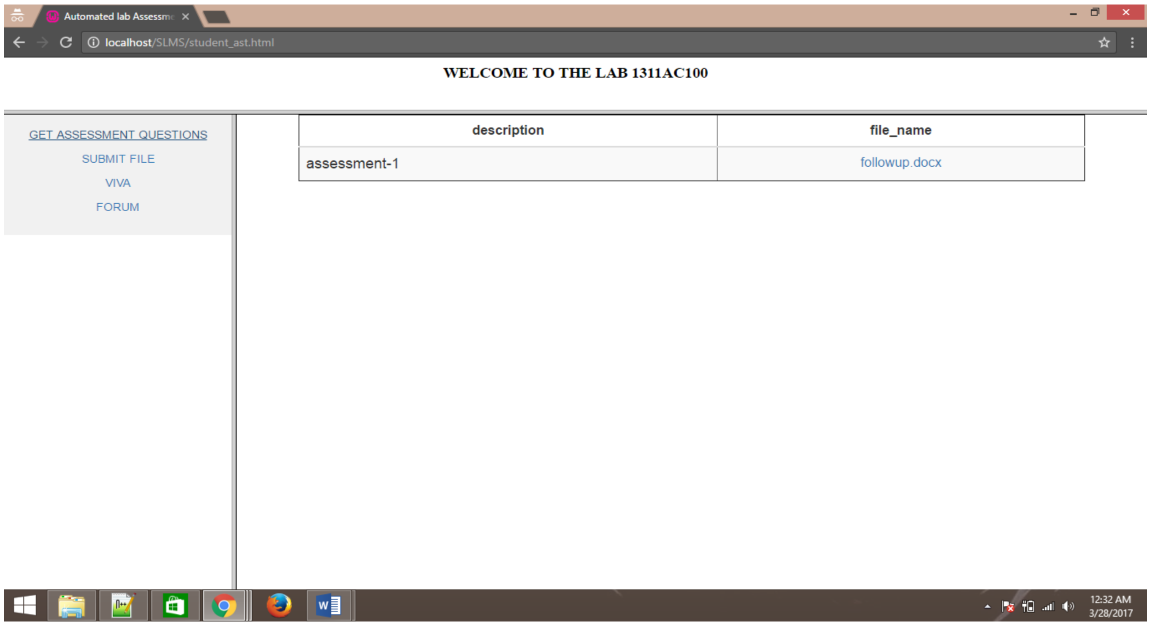


Figure-3.16 *student home page*



Automated lab Assessm

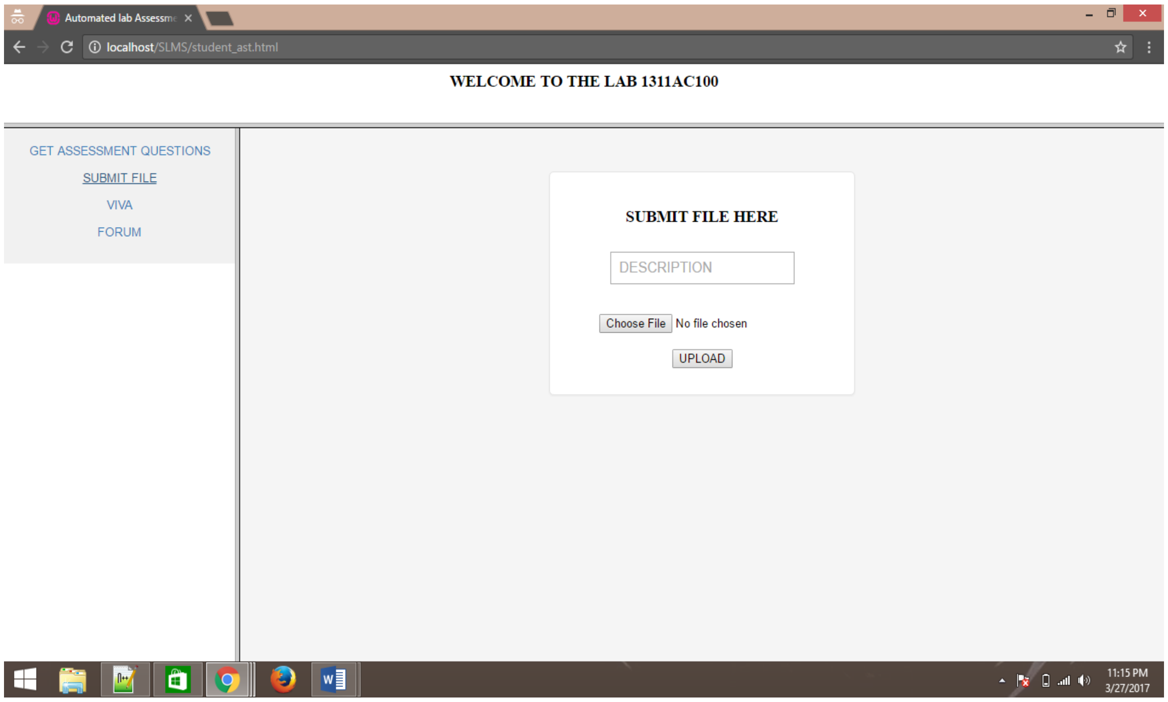
localhost/SLMS/student_ast.html

WELCOME TO THE LAB 1311AC100

GET ASSESSMENT QUESTIONS
SUBMIT FILE
VIVA
FORUM

description	file_name
assessment-1	followup.docx

Figure-3.17 *student receiving questions file*



Automated lab Assessm

localhost/SLMS/student_ast.html

WELCOME TO THE LAB 1311AC100

GET ASSESSMENT QUESTIONS
SUBMIT FILE
VIVA
FORUM

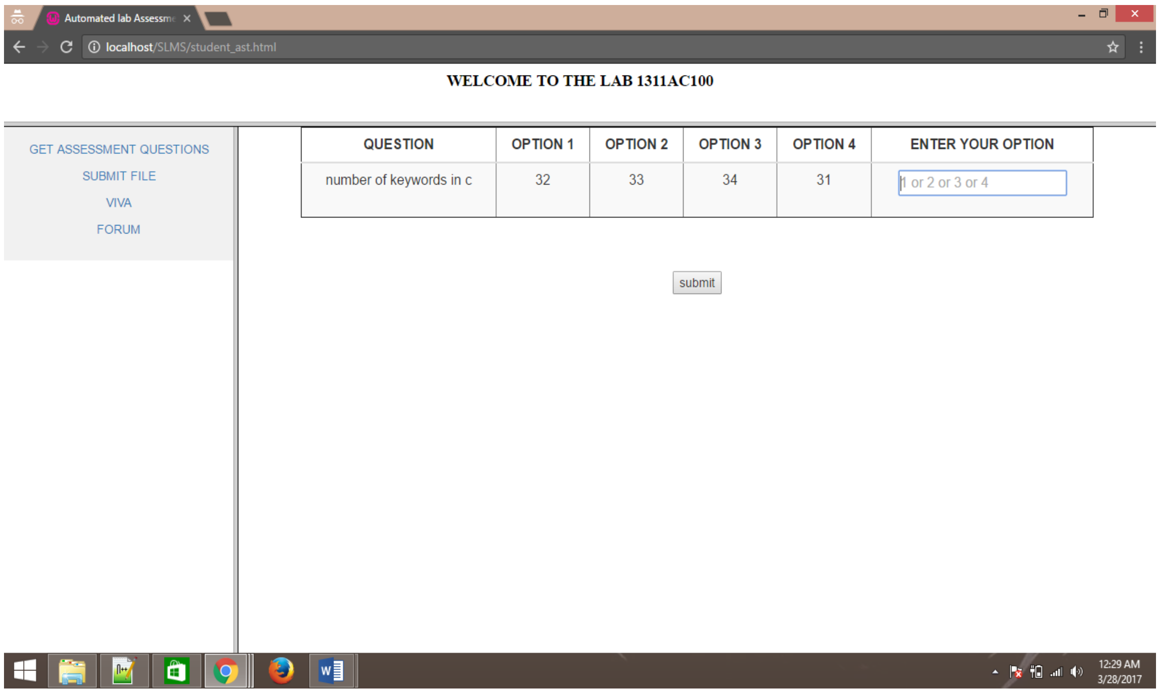
SUBMIT FILE HERE

DESCRIPTION

Choose File No file chosen

UPLOAD

Figure-3.18 *student file submitting page*


WELCOME TO THE LAB 1311AC100

QUESTION	OPTION 1	OPTION 2	OPTION 3	OPTION 4	ENTER YOUR OPTION
number of keywords in c	32	33	34	31	1 or 2 or 3 or 4

Figure-3.19 *receiving viva questions*

3.3 DATABASES AND TABLES:

lab_id= 'year'.'semester'.'section'.'lab name'.'lab number'
table_name='lab_name'.'lab_number'

DATABASE NAME	PREDEFINED TABLES
course_db	course_list
faculty_db	Faculty_table, *lab_id*
files_db	*lab_id*
forum_db	Qusn_list, *lab_id*
lab_db	Lab_list, *lab_id*
lab_questions_db	*lab_id*
noq	Noqtable

question_bank	*table_name*
student_db	*lab_id*
student_viva	*lab_id*
viva	*lab_id*

→ Table names in italic are created dynamically when the new lab session is added.

Chapter 4

UNDERLYING TECHNOLOGIES IN THE PROJECT

4.1 DOMAIN KNOWLEDGE:

Web Application:

A web application is an application that is accessed over a network such as the Internet or an Intranet. The turn may also mean a computer software application that is coded in a browser supported 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.

Web applications are popular due to the ubiquity of web browsers, and convenience of using a web browser as a client, sometimes called a thin client. The ability to update and maintain web applications without distributing and installing software on potentially thousands of client computers is a key reason for their popularity, as is the inherent support for cross platform compatibility.

Hyperlinks:

A hyperlink is a reference in a hypertext document to another document or other resource. It is similar to a citation or a literature. Combined with a data network and suitable access protocol, it can be used to fetch the resource referenced. This can done be saved, viewed, or displayed a spot of the referencing document. The most common type of hyperlink is the URL used in World Wide Web.

Web Server:

A web server is a computer (or specialized software running on a computer) responsible for serving the web pages to a client (usually a web browser) when a user requests the page. The browser sends the request to the web server, which responds after a brief “negotiation” (using HTTP), and sometimes some extra processing, the web server sends the page back to the browser for rendering.

GUI:

An acronym for Graphical User Interface, this term refers to a software front-end meant to provide an attractive and easily to use interface between a computer and application.

4.2 HTML:

HTML also known as HYPERTEXT MARKUP language is most commonly used markup language to create web pages. A markup language provides way to describe the structure of text based information on web page. Hypertext though similar to regular text has one additional advantage that is when you click the hypertext present on web page you are directed to another webpage on internet. The hypertext is called hyper because the navigation through the pages using hypertext is not linear. It means that if you click the hypertext present on webpage you are directed to relevant page on website or internet which is not necessary the next page on website. Worldwide is organization that defines new specification for HTML and responsible to update the language. HTML allow you to format arrange and group text display text as link and add images and multimedia to webpage. It also allow you to create and work with style sheet, controls, embedded scripting language code in webpage. HTML is written in form of tags which are by pair of angular brackets(<>) and some text placed between these brackets. Most of the elements has two basic properties :

Attributes: They have an opening tags (<element name) and closing tag(</element-name). Some HTML <hr> does not have content. Attributes Contents are collectively known as HTML markup. Most of the HTML attributes are name value pairs, separated by= sign. Some attributes such as map attribute of the <img> element is written without specifying value for it. Attribute value should be enclosed within single or double quotes.

Syntax:<element_nameattribute_name=”attribute_value”>content</element_ name>

Basic HTML tags:

<!_ _ -> Specifies comments

<A>…………..</A> creates the hypertext links

<B>……………</B> formats the text as bold

<BIG>…………</BIG> formats text in large font

<BODY>……..</BODY> contains all tags and text in HTML document

<FORM>………</FORM> encloses a Fill-out form

<FRAME>……</FRAME> defines a particular frame in a set of frames

<H#>…………….</#H> creates headings of different levels(1-6)

<HEAD>……….</HEAD> contains tags that specify information about a document

<SCRIPT>……..</SCRIPT> contains client-side or server-side script

<TABLE>……….</TABLE> creates a table

<TD>………..</td> indicates table data in a table

<TH>…………</TH> creates a heading in a table

<TR>………….</TR> designates a table row

Advantages:

A HTML document is small and hence easy to send over the net. it is small because it does not include formatted information.

HTML is platform independent.

HTML tags are not case-sensitive.

4.3 DATABASE:

A database management system(DBMS) is computer software designed for the purpose of managing databases, a large set of structured data, and run operations on the data requested by numerous users. Typical examples of DBMSs include Oracle, DB2, Microsoft Access, Microsoft SQL Server, Firebird, PostgreSQL, MySQL, SQLite, FileMaker and Sybase adaptive, server enterprise. DBMSs are typically used by database administrators in the creation of Database systems. Typical examples of DBMS use include accounting, human resources and customer support systems.

Originally found only in large companies with the computer hardware needed to support large data sets, DBMSs have more recently emerged as a fairly standard part of any company back office.

4.3.1: Description:

A DBMS is a complex set of software programs that controls the organization, storage, management, and retrieval of data in a database. A DBMS includes:
A modeling language to define the schema of each database hosted in the DBMS, according to the DBMS data model.

The four most common types of organizations are the hierarchical, network, relational and object models. Inverted lists and other models are also used. A given database management system may provide one or more of the four models. The optimal structure depends on the natural organization of the application's data, and on the application's requirements (which include reliability, maintainability, scalability, and cost). The dominant model use today is the ad hoc one embedded in SQL, despite the objections of purists who believe this model is a corruption of the relational model, since it violates several of its fundamental principles for the sake of practicality and performance. Many DBMSs also support the open Database Connectivity API that supports a standard way for programmers to access the DBMS. Data structures optimized to deal with very large amounts of data stored on a permanent data storage device. A database query language and report writer to allow users to interactively interrogate the database, analyze its data and update it according to the users privileges on data. It also controls the security of the database. Data security prevents unauthorized users from viewing or updating the database. Using passwords users are allowed access to the entire database or subsets of it called sub schemes. If the DBMS provides a way to interactively enter and update the database, as well as interrogate it, this capability allows for managing personal databases.

However, it may not leave an audit trail of actions or provide the kinds of controls necessary in a multiuser organization. These controls are only available when a set of application programs are customized for each data entry and updating function. A transaction mechanism, that ideally would guarantee the ACID properties, in order to ensure data integrity, despite concurrent user accesses, and faults. It also maintains the integrity of the data in the database. The DBMS can maintain the integrity of the database by not allowing more than one user to update the same record at the same time. The DBMS can help prevent duplicate records via unique index constraints. The DBMS accepts requests for data from the application program and instructs the operating system to transfer the appropriate data.

When a DBMS is used, information systems can be changed much more easily as the organization's information requirements change. New categories of data can be added to the database without disruption to the existing system. Database servers are specially designed computers that hold the actual databases and run only the DBMS and related software. Database servers are usually multiprocessor computers, with RAID disk arrays used for storage. Connected to one or more servers via a high-speed channel, hardware database accelerators are also used in large volume transaction processing environments.
DBMSs are found at the heart of most database applications. Sometimes DBMSs are built around a private multitasking kernel with built-in networking support although nowadays these functions are left to the operating system.

4.4 CASCADING STYLE SHEET:

Cascading Style Sheets(CSS) is a style sheet language used for describing the presentation semantics (the look and formatting) of a document written in a markup language. It’s most common application is to style web pages written in HTML and XHTML, but the language can also be applied to any kind of XML document, including plain XML, SVG and XUL.

4.5 SQL-STRUCTURED QUERY LANGUAGE:

SQL stands for Structured Query Language. SQL is used to communicate with a database. According to ANSI (American National Standards Institute), it is the standard language for relational database management systems. SQL statements are used to perform tasks such as update data on a database, or retrieve data from a database. Some common relational database management systems that use SQL are: Oracle, Sybase, Microsoft SQL Server, Access, Ingres, etc. Although most database systems use SQL, most of them also have their own additional proprietary extensions that are usually only used on their system.

However, the standard SQL commands such as "Select", "Insert", "Update", "Delete", "Create", and "Drop" can be used to accomplish almost everything that one needs to do with a database.

Structured Query Language (SQL) is the language used to manipulate relational databases. SQL is tied very closely with the relational model. In the relational model, data is stored in structures called relations or tables. SQL statements are issued for the purpose of:

Data definition: Defining tables and structures in the database (DDL used to create, alter and drop schema objects such as tables and indexes).

Data manipulation: Used to manipulate the data within those schema objects (DML Inserting, Updating, deleting the data, and Querying the Database).

List of SQL statements that can be issued against an Oracle database schema are:

- AUDIT - Track the changes made to a table (DDL)
- COMMENT - Add a comment to a table or column in a table (DDL)
- COMMIT - Make all recent changes permanent (DML - transactional)
- CREATE - Create new database objects such as tables or views (DDL)
- DELETE - Delete rows from a database table (DML)
- DROP - Drop a database object such as a table, view or index (DDL)
- GRANT - Allow another user to access database objects such as tables or views (DDL)
- INSERT - Insert new data into a database table (DML)
- No AUDIT - Turn off the auditing function (DDL)
- REVOKE - Disallow a user access to database objects such as tables and views (DDL)
- ROLLBACK - Undo any recent changes to the database (DML - Transactional)
- SELECT - Retrieve data from a database table (DML)
- TRUNCATE - Delete all rows from a database table (cannot be rolled back) (DML)

4.6 JAVASCRIPT:

JavaScript is a script-based programming language that was developed by Netscape Communication Corporation. JavaScript was originally called Live Script and renamed as JavaScript to indicate its relationship with Java. JavaScript supports the development of both client and server components of Web-based applications. On the client side, it can be used to write programs that are executed by a Web browser within the context of a Web page. On the server side, it can be used to write Web server programs that can process information submitted by a Web browser and then update the browser's display accordingly.

Even though JavaScript supports both client and server Web programming, we prefer JavaScript at Client side programming since most of the browsers supports it. JavaScript is almost as easy to learn as HTML, and JavaScript statements can be included in HTML documents by enclosing the statements between a pair of scripting tags
<SCRIPTS>.. </SCRIPT>.
<SCRIPT LANGUAGE = "JavaScript">
JavaScript statements
</SCRIPT>
Here are a few things we can do with JavaScript:

- ➔ Validate the contents of a form and make calculations.
- ➔ Add scrolling or changing messages to the Browser's status line.
- ➔ Animate images or rotate images that change when we move the mouse over them.
- ➔ Detect the browser in use and display different content for different browsers.
- ➔ Detect installed plug-ins and notify the user if a plug-in is required.
- ➔ We can do much more with JavaScript, including creating entire application.
- ➔ In this project, we mainly use it for validation of text fields.

4.7 PHP

PHP is a server scripting language, and a powerful tool for making dynamic and interactive Web pages. It is a widely-used, free, and efficient alternative to competitors such as Microsoft's ASP.PHP code may be embedded into HTML or HTML5 code, or it can be used in combination with various web template systems, web content management systems and web frameworks. PHP code is usually processed by a PHP interpreter implemented as a module in the web server or as a Common Gateway Interface (CGI) executable. The web server combines the results of the interpreted and executed PHP code, which may be any type of data, including images, with the generated web page. PHP code may also be executed with a command-line interface (CLI) and can be used to implement standalone graphical applications.

- ➔ In our project, we use PHP to handle forms.
- ➔ PHP functions are used to access the MySQL database and to manipulate the data records in it.
- ➔ It is used throughout the project to develop back end actions and interact with the server.

Chapter 5

TESTING

5.1 TESTING PLAN:

Software Testing is a critical element of software quality assurance and represents the ultimate review of specifications, design and coding. Testing presents an interesting anomaly for the software engineering.

Testing Objectives Include:

- Testing is a process of executing a program with the intent of finding an error.
- A good test case is one that has a probability of finding an as yet undiscovered error.
- A successful test is one that uncovers an undiscovered error.

Testing Principles:

- All rests should be traceable to end user requirements.
- Tests should be planned long before testing begins.
- Testing should begin on a small scale and progress towards testing in large.
- Exhaustive testing is not possible.
- To be most effective testing should be conducted by an independent third party.

5.2 TESTING STRATEGIES:

A strategy for software testing integrates software test cases into a series of well planned steps that result in the successful construction of software. Software testing is a boarder topic for what is referred to as verification and validation. Verification refers to the set of activities that ensure that the software correctly implements a specific function.

Unit Testing:

Unit testing focuses verification effort on the smallest unit of software design that is the module. Using procedural design description as a guide, important control paths are tested to uncover errors within the boundaries of the module. The unit test is normally white box testing oriented and the step can be conducted in parallel for multiple modules.

Integration Testing:

Integration testing is a systematic technique for constructing the program structure, while conducting test to uncover errors associated with the interface. The objective is to take unit tested methods and build a program structure that has been dictated by design.

Top-Down Integration:

Top down integration is an incremental approach for construction of program structure. Module are integrated by moving downward through the control hierarchy, beginning with the main control program. Modules subordinate to the main program are incorporated in the structure either in the breadth-first or depth-first manner.

Bottom-Up Integration:

This method as the name suggests, begins constructive and testing with atomic modules i.e..; modules at the lowest level. Because the modules are integrated in the bottom up manner the processing required for the modules subordinate to a given level is always available and for stubs is eliminated.

Validation Testing:

At the end of integration testing software is completely assembled as a package. Validation testing is the next stage, which can be defined as successful when the software function in the manner reasonably expected by the customer. Reasonable expectations are those defined in the software requirements specification. Information contained in those sections form a basis for validation testing approach.

System testing:

System testing is actually a series of difficulty tests whose primary purpose is to fully exercise the computer-based system. Although each test has a different purpose, all work to verify that all system elements have been properly integrated to perform allocated functions.

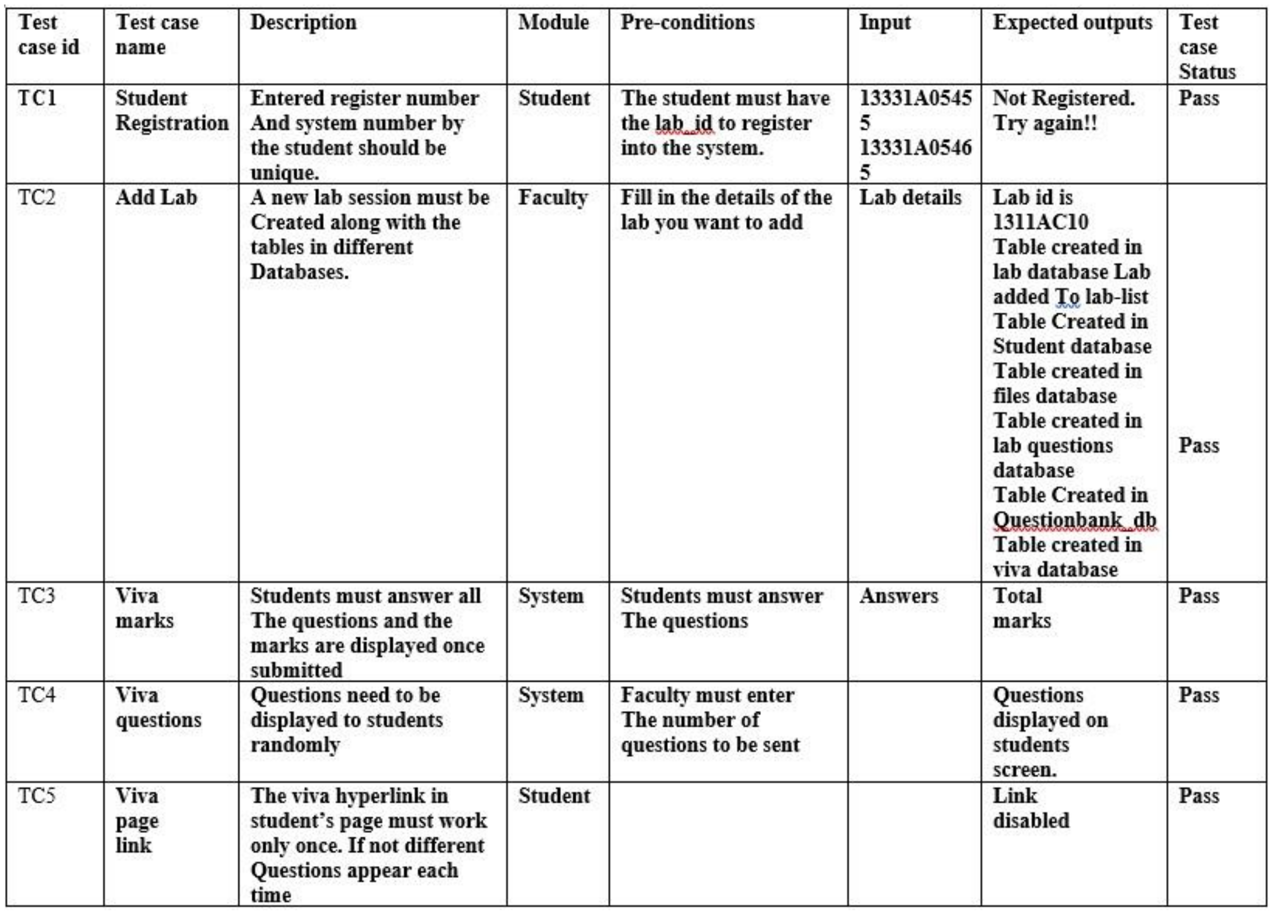
Security Testing:

Attempts to verify the protection mechanisms built into the system.

Performance Testing:

This method is designed to test runtime performance of the software within the context of an integrated system.

5.3 TEST CASES REPORT:



Test case id	Test case name	Description	Module	Pre-conditions	Input	Expected outputs	Test case Status
TC1	Student Registration	Entered register number And system number by the student should be unique.	Student	The student must have the lab_id to register into the system.	13331A05455 13331A05465	Not Registered. Try again!!	Pass
TC2	Add Lab	A new lab session must be Created along with the tables in different Databases.	Faculty	Fill in the details of the lab you want to add	Lab details	Lab id is 1311AC10 Table created in lab database Lab added To lab-list Table Created in Student database Table created in files database Table created in lab questions database Table Created in Questionbank_db Table created in viva database	Pass
TC3	Viva marks	Students must answer all The questions and the marks are displayed once submitted	System	Students must answer The questions	Answers	Total marks	Pass
TC4	Viva questions	Questions need to be displayed to students randomly	System	Faculty must enter The number of questions to be sent		Questions displayed on students screen.	Pass
TC5	Viva page link	The viva hyperlink in student's page must work only once. If not different Questions appear each time	Student			Link disabled	Pass

TC6	**Updating marks**	**The record and execution Marks entered must be Updated once clicked on update.**	**faculty**	**Marks must be entered.**	**marks**	**Updated marks list**	**Pass**
TC7	**Uploading questions**	**The faculty uploads Question file without Any description**	**Faculty**	**Upload a file**	**Empty description**	**Description needed!**	**Pass**
TC8	**Uploading answers**	**The student uploads the file with description. The file gets saved in website Folder**	**Student**	**Enter description Upload file**		**Uploaded. (file saved)**	**Pass**
TC9	**Forum**	**All the Questions in the forum database are deleted once the lab session ends and faculty logouts**	**Faculty**	**The faculty must logout**		**Forum database is empty.**	**Pass**

Chapter 6

CONCLUSION

In comparison with a manual system, the benefit under a computer system is considerable into saving of man power, working hours and effort. It can be observed that information required can be obtained with ease and accuracy in computerized system.

Automation of Lab Assessment system is done in such a way that, it is beneficial for the faculty members and students through which they can track the performance of the students and maintain proper records of it. The system is thoroughly tested against various inputs and is concluded to be efficiently working. The required functionalities of the system are met.

This application can be used by any institute as it can be modified easily. Additional features can be added to the system without interrupting its normal functionality.

BIBILOGRAPHY

APPENDIX – A:

References:

The following mentioned books are referred in order to design the system:

1. www.w3schools.com
2. www.tutorialspoint.com
3. https://www.w3schools.com/Bootstrap/bootstrap_templates.asp
4. www.stackoverflow.com
5. www.wikipedia.org

APPENDIX-B

SAMPLE CODE:

Add_lab.php

```
<?php
echo "
<html lang=\"en\">
  <head>
    <meta charset=\"utf-8\">
    <title>Automated Lab Assessment System</title>
    <meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0\">
    <meta name=\"description\" content=\"\">
    <meta name=\"author\" content=\"\">
    <!-- Le styles -->
             <link rel=\"stylesheet\"
href=\"//maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css\">
  <link rel=\"stylesheet\" href=\"//netdna.bootstrapcdn.com/twitter-
bootstrap/2.3.1/css/bootstrap-combined.min.css\">
  <link rel=\"stylesheet\" type=\"text/css\" media=\"screen\"
href=\"//cdnjs.cloudflare.com/ajax/libs/bootstrap-select/1.7.5/css/bootstrap-
select.min.css\">
    <link href=\"../assets/css/bootstrap.css\" rel=\"stylesheet\">
    <style type=\"text/css\">
      body {
        padding-top: 40px;
        padding-bottom: 40px;
        background-color: #f5f5f5;
      }
      .form-signin {
        max-width: 300px;
        padding: 19px 29px 29px;
        margin: 0 auto 20px;
        background-color: #fff;
        border: 1px solid #e5e5e5;
        -webkit-border-radius: 5px;
           -moz-border-radius: 5px;
                border-radius: 5px;
        -webkit-box-shadow: 0 1px 2px rgba(0,0,0,.05);
           -moz-box-shadow: 0 1px 2px rgba(0,0,0,.05);
                box-shadow: 0 1px 2px rgba(0,0,0,.05);
      }
      .form-signin .form-signin-heading,
      .form-signin .checkbox {
        margin-bottom: 10px;
      }
      .form-signin input[type=\"text\"],
      .form-signin input[type=\"password\"] {
        font-size: 16px;
        height: auto;
        margin-bottom: 15px;
        padding: 7px 9px;
      }
    </style>
    <link href=\"../assets/css/bootstrap-responsive.css\" rel=\"stylesheet\">
    <!-- HTML5 shim, for IE6-8 support of HTML5 elements -->
    <!--[if lt IE 9]>
      <script src=\"../assets/js/html5shiv.js\"></script>
    <![endif]-->
  </head>
  <body>
  <script src=\"//ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js\"></script>
                <script src=\"//netdna.bootstrapcdn.com/twitter-
bootstrap/2.3.1/js/bootstrap.min.js\"></script>
```

```
                <script src=\"//cdnjs.cloudflare.com/ajax/libs/bootstrap-
select/1.7.5/js/bootstrap-select.min.js\"></script>
          <script type=\"text/javascript\">
               $(document).ready(function() {
               $('.selectpicker').selectpicker();
              });
          </script>
  </body>
</html>";
session_start();
$error="";
$dbuser="root";
$server="localhost";
            $conn=mysqli_connect($server,$dbuser,$error);
            if(!$conn)
                        echo "connection failed";
            if(mysqli_select_db($conn,"course_db"))
            {
                        echo "<div class=\"container\">";
                        echo "<form class=\"form-signin\" action=\"add_lab1.php\"
method=\"POST\">";
                        echo "<center><h2> ADD A NEW LAB </h2></center>";
                         echo "Select year:
                                    <select class=\"selectpicker\" name=\"year\">
                                    <option value=\"13\">2013-17</option>
                                    <option value=\"14\">2014-18</option>
                                    <option value=\"15\">2015-19</option>
                                    <option value=\"16\">2016-20</option>
                                    </select><br>";
                        echo "select semester:
                                    <select class=\"selectpicker\" name=\"sem\">
                                    <option value=\"11\">1-1</option>
                                    <option value=\"12\">1-2</option>
                                    <option value=\"21\">2-1</option>
                                    <option value=\"22\">2-2</option>
                                    <option value=\"31\">3-1</option>
                                    <option value=\"32\">3-2</option>
                                    <option value=\"41\">4-1</option>
                                    <option value=\"42\">4-2</option>
                                    </select><br>";
                        echo "select sec:
                                    <select class=\"selectpicker\" name=\"sec\">
                                    <option value=\"A\">A</option>
                                    <option value=\"B\">B</option>
                                    </select><br>";
                        echo "select lab name:";
                        $a="select lab_name from course_list";
                        $b=mysqli_query($conn,$a);
                        echo " <select name=\"lab_name\" class=\"selectpicker\">";
                        while($a2=mysqli_fetch_array($b))
                        {
                          echo"<option value=\"".$a2[0]."\">".$a2[0]."</option>";
                        }
                        echo "</select><br>";
                        echo "Enter lab number";
                        echo" <input type=\"text\" class=\"input-block-level\"
name=\"lab_no\"><br><br>";
                        echo "<br>";
                        echo "<center>  <button class=\"btn btn-large btn-primary\"
type=\"submit\" name=\"add\">ADD</button> </center>";
                        echo "</form>";
                          echo " </div> <!-- /container -->";
```

```
                    echo "<br><br><footer><center>";
                    echo "<a href=\"add_students.html\" target=\"_blank\">ADD CLASS
</a>";
                    echo "</footer>";
            }
?>
```

<u>Faculty_login.html :</u>

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Automated Lab Assessment System</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="">
    <meta name="author" content="">
    <!-- Le styles -->
    <link href="../assets/css/bootstrap.css" rel="stylesheet">
    <style type="text/css">
      body {
        padding-top: 40px;
        padding-bottom: 40px;
        background-color: #f5f5f5;
      }
      .form-signin {
        max-width: 300px;
        padding: 19px 29px 29px;
        margin: 0 auto 20px;
        background-color: #fff;
        border: 1px solid #e5e5e5;
        -webkit-border-radius: 5px;
           -moz-border-radius: 5px;
```

```
            border-radius: 5px;
        -webkit-box-shadow: 0 1px 2px rgba(0,0,0,.05);
           -moz-box-shadow: 0 1px 2px rgba(0,0,0,.05);
                box-shadow: 0 1px 2px rgba(0,0,0,.05);
      }
      .form-signin .form-signin-heading,
      .form-signin .checkbox {
        margin-bottom: 10px;
      }
      .form-signin input[type="text"],
      .form-signin input[type="password"] {
        font-size: 16px;
        height: auto;
        margin-bottom: 15px;
        padding: 7px 9px;
      }
    </style>
    <link href="../assets/css/bootstrap-responsive.css" rel="stylesheet">

    <!-- HTML5 shim, for IE6-8 support of HTML5 elements -->
    <!--[if lt IE 9]>
      <script src="../assets/js/html5shiv.js"></script>
    <![endif]-->
    <!-- Fav and touch icons -->
    <link rel="apple-touch-icon-precomposed" sizes="144x144" href="../assets/ico/apple-
touch-icon-144-precomposed.png">
    <link rel="apple-touch-icon-precomposed" sizes="114x114" href="../assets/ico/apple-
touch-icon-114-precomposed.png">
      <link rel="apple-touch-icon-precomposed" sizes="72x72" href="../assets/ico/apple-
touch-icon-72-precomposed.png">
```

```
                    <link rel="apple-touch-icon-precomposed" href="../assets/ico/apple-touch-
icon-57-precomposed.png">
                              <link rel="shortcut icon" href="../assets/ico/favicon.png">
  </head>
  <header><a href="faculty_main.html">GO BACK </a></header><br><br><br>
  <body>
    <div class="container">
      <form class="form-signin" action="http://localhost/SLMS/faculty_login.php"
target="_top" method="POST">
       <center>  <h2 class="form-signin-heading">LOGIN</h2><br>
        <input type="text" class="input-block-level" name="faculty_id"
placeholder="Faculty ID"><br><br>
        <input type="password" class="input-block-level" name="faculty_pwd"
placeholder="Password"><br><br>
       <button class="btn btn-large btn-primary" type="submit"
name="f_login">LOGIN</button> </center>
      </form>
    </div> <!-- /container -->
    <!-- Le javascript
    <!-- Placed at the end of the document so the pages load faster -->
    <script src="../assets/js/jquery.js"></script>
    <script src="../assets/js/bootstrap-transition.js"></script>
    <script src="../assets/js/bootstrap-alert.js"></script>
    <script src="../assets/js/bootstrap-modal.js"></script>
    <script src="../assets/js/bootstrap-dropdown.js"></script>
    <script src="../assets/js/bootstrap-scrollspy.js"></script>
    <script src="../assets/js/bootstrap-tab.js"></script>
    <script src="../assets/js/bootstrap-tooltip.js"></script>
    <script src="../assets/js/bootstrap-popover.js"></script>
    <script src="../assets/js/bootstrap-button.js"></script>
    <script src="../assets/js/bootstrap-collapse.js"></script>
    <script src="../assets/js/bootstrap-carousel.js"></script>
    <script src="../assets/js/bootstrap-typeahead.js"></script>
</body></html>
```

Faculty_login.php

```
<?php
session_start();
$server="localhost";
$dbuser="root";
$error="";
if(isset($_POST['f_login']))
{
        $conn=mysqli_connect($server,$dbuser,$error);
        if(!$conn)
                echo " connection failed" ;
        if(mysqli_select_db($conn,"faculty_db"))
        {
                $f_id=$_POST['faculty_id'];
                $f_pwd=$_POST['faculty_pwd'];
                $q="select faculty_pwd from faculty_table where faculty_id=\"$f_id\"";
                $r=mysqli_query($conn,$q);
                if(!$r)
                        echo "query failed";
                if(mysqli_num_rows($r)==0)
                {
                        echo "<script type=\"text/javascript\">
                                        alert (\"Enter correct id\");
                                 </script> ";
                   header("Location: faculty_main.html");
                }
                else
                {$x="select faculty_name from faculty_table where faculty_id=\"$f_id\"";
                $y=mysqli_query($conn,$x);
                $dwt=mysqli_fetch_array($y,MYSQL_NUM);
                if(!$y)
                        die(mysqli_error($conn));
                $_SESSION['f_name']=$dwt;

                        $row= mysqli_fetch_array($r,MYSQL_ASSOC);
                        if($row['faculty_pwd']==$f_pwd)
                        {
                                header("location:faculty_home.html");
                        }
                        else
                                echo "<script type=\"text/javascript\">
               alert (\"login failed\");
                                         </script> ";
                }

        }
        if(mysqli_select_db($conn,"forum_db"))
        {
 $c="CREATE TABLE qusn_list(question_id VARCHAR(10),question
VARCHAR(300))";
         mysqli_query($conn,$c);
        }
}
?>
```

Database file :

```
SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET time_zone = "+00:00";
-- Database: `b_tech’
```

```
CREATE DATABASE IF NOT EXISTS `b_tech` DEFAULT CHARACTER SET latin1
COLLATE latin1_swedish_ci;
USE `b_tech`;
-- Table structure for table `13a`
CREATE TABLE `13a` (
  `s_name` varchar(10) NOT NULL,
  `regd_no` varchar(10) NOT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
-- Dumping data for table `13a`
INSERT INTO `13a` (`s_name`, `regd_no`) VALUES
ALTER TABLE `13a`
  ADD PRIMARY KEY (`regd_no`);
-- Database: `course_db`
CREATE DATABASE IF NOT EXISTS `course_db` DEFAULT CHARACTER SET
latin1 COLLATE latin1_swedish_ci;
-- Table structure for table `course_list`
CREATE TABLE `course_list` (
  `semester` varchar(10) NOT NULL,
  `lab_name` varchar(20) NOT NULL,
  `no_of_labs` int(10) NOT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
INSERT INTO `course_list` (`semester`, `lab_name`, `no_of_labs`) VALUES
('1-1', 'C', 10),
-- Database: `faculty_db`
CREATE DATABASE IF NOT EXISTS `faculty_db` DEFAULT CHARACTER SET
latin1 COLLATE latin1_swedish_ci;
USE `faculty_db`;
-- Table structure for table `faculty_table`

CREATE TABLE `faculty_table` (
  `faculty_id` varchar(20) NOT NULL,
  `faculty_name` varchar(20) NOT NULL,
  `faculty_pwd` varchar(10) NOT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
INSERT INTO `faculty_table` (`faculty_id`, `faculty_name`, `faculty_pwd`) VALUES
 ('01', 'Vamsi krishna', 'jayanthi');
ALTER TABLE `faculty_table`
  ADD PRIMARY KEY (`faculty_id`);
-- Database: `files_db`
CREATE DATABASE IF NOT EXISTS `files_db` DEFAULT CHARACTER SET
latin1 COLLATE latin1_swedish_ci;
USE `files_db`;
-- Table structure for table `1311ac01`
CREATE TABLE `1311ac01` (
  `regd_no` varchar(20) NOT NULL,
  `description` varchar(30) NOT NULL,
  `file_name` varchar(20) DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
INSERT INTO `1311ac01` (`regd_no`, `description`, `file_name`) VALUES
('13331A0545', '2 outputs', '13331A0545.txt');
-- Database: `forum_db`
CREATE DATABASE IF NOT EXISTS `forum_db` DEFAULT CHARACTER SET
latin1 COLLATE latin1_swedish_ci;
-- Table structure for table `qusn_list`
CREATE TABLE `qusn_list` (
  `question_id` varchar(10) DEFAULT NULL,
  `question` varchar(300) DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
-- Database: `lab_db`
CREATE DATABASE IF NOT EXISTS `lab_db` DEFAULT CHARACTER SET latin1
COLLATE latin1_swedish_ci;
USE `lab_db`;
```

```
-- Table structure for table `1311ac01`
CREATE TABLE `1311ac01` (
  `regd_no` varchar(20) DEFAULT NULL,
  `s_name` varchar(20) DEFAULT NULL,
  `system_no` int(10) DEFAULT NULL,
  `execution_marks` varchar(20) DEFAULT NULL,
  `viva_marks` varchar(20) DEFAULT NULL,
  `record_marks` varchar(20) DEFAULT NULL,
  `total_marks` varchar(20) DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
-- Dumping data for table `1311ac01`
INSERT INTO `1311ac01` (`regd_no`, `s_name`, `system_no`, `execution_marks`,
`viva_marks`, `record_marks`, `total_marks`) VALUES
('0', '0', 0, '0', '0', '0', '0'),
('13331A0545', 'jayanthi', 1, '4', '0', '10', '14');
-- Table structure for table `lab_list`
CREATE TABLE `lab_list` (
  `year` varchar(10) NOT NULL,
  `semester` int(6) NOT NULL,
  `section` varchar(6) NOT NULL,
  `lab_name` varchar(10) NOT NULL,
  `lab_no` int(6) NOT NULL,
  `lab_date` date NOT NULL,
  `lab_time` time(6) NOT NULL,
  `lab_id` varchar(100) NOT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
-- Dumping data for table `lab_list`
INSERT INTO `lab_list` (`year`, `semester`, `section`, `lab_name`, `lab_no`, `lab_date`,
`lab_time`, `lab_id`) VALUES
 ('13', 11, 'A', 'C', 1, '2017-03-27', '18:03:00.000000', '1311AC01');
- Indexes for table `lab_list`
```

```
ALTER TABLE `lab_list`
  ADD UNIQUE KEY `lab_id` (`lab_id`);
-- Database: `lab_questions_db`
CREATE DATABASE IF NOT EXISTS `lab_questions_db` DEFAULT CHARACTER
SET latin1 COLLATE latin1_swedish_ci;
USE `lab_questions_db`;
-- Table structure for table `1311ac01`
CREATE TABLE `1311ac01` (
  `description` varchar(30) NOT NULL,
  `file_name` varchar(20) DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
-- Dumping data for table `1311ac01`
INSERT INTO `1311ac01` (`description`, `file_name`) VALUES
('assessment-1', 'final.docx');
-- Indexes for dumped tables
-- Indexes for table `1311ac01
ALTER TABLE `1311ac01`
  ADD UNIQUE KEY `file_name` (`file_name`);
-- Database: `noq`
CREATE DATABASE IF NOT EXISTS `noq` DEFAULT CHARACTER SET latin1
COLLATE latin1_swedish_ci;
USE `noq`;
-- Table structure for table `noqtable`
CREATE TABLE `noqtable` (
  `lab_id` varchar(200) NOT NULL,
  `noq` int(10) NOT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
-- Dumping data for table `noqtable`
INSERT INTO `noqtable` (`lab_id`, `noq`) VALUES
('1311AST1', 3),
('1311AC100', 1),
('1311AC01', 1);
-- Indexes for dumped tables
-- Indexes for table `noqtable;
ALTER TABLE `noqtable`
  ADD UNIQUE KEY `lab_id` (`lab_id`);
-- Database: `question_bank`
CREATE DATABASE IF NOT EXISTS `question_bank` DEFAULT CHARACTER
SET latin1 COLLATE latin1_swedish_ci;
USE `question_bank`;
CREATE TABLE `c_01` (
  `q_no` int(11) DEFAULT NULL,
  `question` varchar(30) NOT NULL,
  `a1` varchar(20) DEFAULT NULL,
  `a2` varchar(20) DEFAULT NULL,
  `a3` varchar(20) DEFAULT NULL,
  `a4` varchar(20) DEFAULT NULL,
  `correct_answer` int(11) DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
-- Dumping data for table `c_01`
INSERT INTO `c_01` (`q_no`, `question`, `a1`, `a2`, `a3`, `a4`, `correct_answer`)
VALUES
(1, 'father of c', 'dennis ', 'gandhi', 'babbage', 'einstein', 1);
-- Indexes for dumped tables
-- Indexes for table `c_01`
ALTER TABLE `c_01`
  ADD UNIQUE KEY `q_no` (`q_no`);
-- Database: `student_db`
CREATE DATABASE IF NOT EXISTS `student_db` DEFAULT CHARACTER SET
latin1 COLLATE latin1_swedish_ci;
USE `student_db`;
-- Table structure for table `1311ac01`
```

```
CREATE TABLE `1311ac01` (
  `regd_no` varchar(20) NOT NULL,
  `s_name` varchar(30) NOT NULL,
  `system_no` int(11) DEFAULT NULL,
  `lab_id` varchar(200) DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
-- Dumping data for table `1311ac01`
INSERT INTO `1311ac01` (`regd_no`, `s_name`, `system_no`, `lab_id`) VALUES
('13331A0545', 'jayanthi', 1, '1311AC01');
-- Database: `student_viva`
CREATE DATABASE IF NOT EXISTS `student_viva` DEFAULT CHARACTER SET
latin1 COLLATE latin1_swedish_ci;
USE `student_viva`;
-- Table structure for table `1311ac01`
CREATE TABLE `1311ac01` (
  `q_no` int(11) DEFAULT NULL,
  `question` varchar(30) NOT NULL,
  `a1` varchar(20) DEFAULT NULL,
  `a2` varchar(20) DEFAULT NULL,
  `a3` varchar(20) DEFAULT NULL,
  `a4` varchar(20) DEFAULT NULL,
  `correct_answer` int(11) DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
-- Dumping data for table `1311ac01`
INSERT INTO `1311ac01` (`q_no`, `question`, `a1`, `a2`, `a3`, `a4`, `correct_answer`)
VALUES
(1, 'father of c', 'dennis ', 'gandhi', 'babbage', 'einstein', 1);
-- Indexes for table `1311ac01`
--
ALTER TABLE `1311ac01`
  ADD UNIQUE KEY `q_no` (`q_no`)
```