A Project Report
on

COLLEGE ACTIVITIES TRACKER

Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

17WH1A0596	**Ms. AFREEN BEGUM**
17WH1A0598	**Ms. SANIA THAHASEEN**
18WH8A0501	**Ms. V.NIHARIKA**

Under the esteemed guidance of

Mr. K Naresh
Assistant Professor

Department of Computer Science and Engineering
BVRIT HYDERABAD
College of Engineering for Women
(NBA Accredited – EEE, ECE, CSE and IT)
(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)
Bachupally, Hyderabad – 500090

JUNE, 2021

DECLARATION

We hereby declare that the work presented in this project entitled **"COLLEGE ACTIVITIES TRACKER"** submitted towards completion of Project Work in IV year of B.Tech., CSE at 'BVRIT HYDERABAD College of Engineering for Women', Hyderabad is an authentic record of our original work carried out under the guidance of Mr. K Naresh, Assistant Professor, Department of CSE.

Sign. With date:
Ms.AFREEN BEGUM
(17WH1A0596)

Sign. With date:
Ms. SANIA THAHASEEN
(17WH1A0598)

Sign. With date:
Ms. K.NIHARIKA
(18WH8A0501)

BVRIT HYDERABAD
College of Engineering for Women
(NBA Accredited – EEE, ECE, CSE and IT)
(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)
Bachupally, Hyderabad – 500090

Department of Computer Science and Engineering

Certificate

This is to certify that the Project Work report on "**COLLEGE ACTIVITIES TRACKER**" is a bonafide work carried out by Ms. AFREEN BEGUM (17WH1A0596) ; Ms. SANIA TAHHASEEN (17WH1A0598) ; Ms. K.NIHARIKA (18WH8A0501) in the partial fulfillment for the award of B.Tech. Degree in **Computer Science and Engineering, BVRIT HYDERABAD College of Engineering for Women, Bachupally, Hyderabad**, affiliated to Jawaharlal Nehru Technological University Hyderabad, Hyderabad under my guidance and supervision.

The results embodied in the project work have not been submitted to any other University or Institute for the award of any degree or diploma.

Head of the Department
Dr. K.Srinivasa Reddy
Professor and HOD,
Department of CSE

Guide
Mr. K Naresh
Assistant Professor

External Examiner

Acknowledgements

We would like to express our sincere thanks to **Dr. K V N Sunitha, Principal**, **BVRIT HYDERABAD College of Engineering for Women**, for providing the working facilities in the college.

Our sincere thanks and gratitude to our **Dr. K.Srinivasa Reddy, Professor and HOD**, Department of CSE, **BVRIT HYDERABAD College of Engineering for Women** for all the timely support and valuable suggestions during the period of our project.

We are extremely thankful and indebted to our internal guide, **Mr. K Naresh Assistant Professor**, Department of CSE**, BVRIT HYDERABAD College of Engineering for Women** for his constant guidance, encouragement and moral support throughout the project.

Finally, we would also like to thank our Project Coordinator, all the faculty and staff of **CSE** Department who helped us directly or indirectly, parents and friends for their cooperation in completing the project work.

Ms. AFREN BEGUM

(17WH1A0596)

Ms. SANIA THAHASEEN

(17WH1A0598)

Ms. K.NIHARIKA

(18WH8A0501)

ABSTRACT

College Activities Tracker deals with the tracking of both students and faculty performance and the activities done by them in and out of the college. This program can facilitate us explore all the activities happening in the college and even get to know who has scheduled the activity. Different certificates of the students based on their performance are generated department wise and academic year wise. Our system has two users, faculty and student. Student will upload their certificates that they have participated in other college and faculty will schedule the activity and upload the respective student's certificate those who have participated in the activity.

College Activities Tracker is an web application that will allow the users (both faculty and students) to upload their documents and intimating each activity (hackathon, workshops, competitions) that they participate in and out of the college with certifications as the proof. And it should also allow the user to register for the ongoing or upcoming events through the form and after registration, A successfully registered message should be send to the user to their corresponding phone number. The faculty must be allowed to check or keep track of students and faculty performance and previews the data student wise, faculty wise, section wise, department wise, event wise, duration wise. And faculty should be allowed to add/update the upcoming event happening in the college.

Contents

1. INTRODUCTION

College Management System is an application that will aid users in uploading their Documents and intimating each activity that they perform in and out of the college with Certifications as the proof. This is a centralized system which keeps in track of students and faculty activities and previews the data student wise, faculty wise, section wise, Department wise, Event wise, Duration wise. It is mainly used to maintain the data for Department's weekly report.

1.1 Modules of the Project

We have three modules in our project.

- Admin
- Student
- Faculty

2. HARDWARE AND SOFTWARE SPECIFICATION:

Software Requirement:

1. Language - Java (JDK 1.7)
2. OS - Windows 7- 32bit
3. MySql Server
4. JDBC
5. JSP

Hardware Requirement :

1. 1 GB RAM
2. 80 GB Hard Disk
3. Above 2GHz Processor

3. TECHNOLOGY DESCRIPTION

3.1The Java Platform

A platform is the hardware or software environment in which a program runs. We've already mentioned some of the most popular platforms like Microsoft Windows, Linux, Solaris OS, and Mac OS. Most platforms can be described as a combination of the operating system and underlying hardware. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other hardware-based platforms.

3.2Java Runtime Environment

The Java Runtime Environment, or JRE, is the software required to run any application deployed on the Java Platform. End-users commonly use a JRE in software packages and Web browser plug-in. Sun also distributes a superset of the JRE called the Java 2 SDK(more commonly known as the JDK), which includes development tools such as the Java compiler, Javadoc, Jar and debugger.

One of the unique advantages of the concept of a runtime engine is that errors (exceptions) should not 'crash' the system. Moreover, in runtime engine environments such as Java there exist tools that attach to the runtime engine and every time that an exception of interest occurs they record debugging information that existed in memory at the time the exception was thrown (stack and heap values). These Automated Exception Handling tools provide 'root-cause' information for exceptions in Java programs that run in production, testing or development environments.

3.2.1 JSP:

Java Server Pages (**JSP**) is a Java technology that allows software developers to dynamically generate HTML, XML or other types of documents in response to a Web client request. The technology allows Java code and certain pre-defined actions to be embedded into static content.

The JSP syntax adds additional XML-like tags, called JSP actions, to be used to invoke built-in functionality. Additionally, the technology allows for the creation of JSP tag libraries that act as extensions to the standard HTML or XML tags. Tag libraries provide a platform independent way of extending the capabilities of a Web server.

JSPs are compiled into Java Servlet by a JSP compiler. A JSP compiler may generate a servlet in Java code that is then compiled by the Java compiler, or it may generate byte code for the servlet directly. JSPs can also be interpreted on-the-fly reducing the time taken to reload changes

Java Server Pages (JSP) technology provides a simplified, fast way to create dynamic web content. JSP technology enables rapid development of web-based applications that are server and platform-independent.

Architecture OF JSP

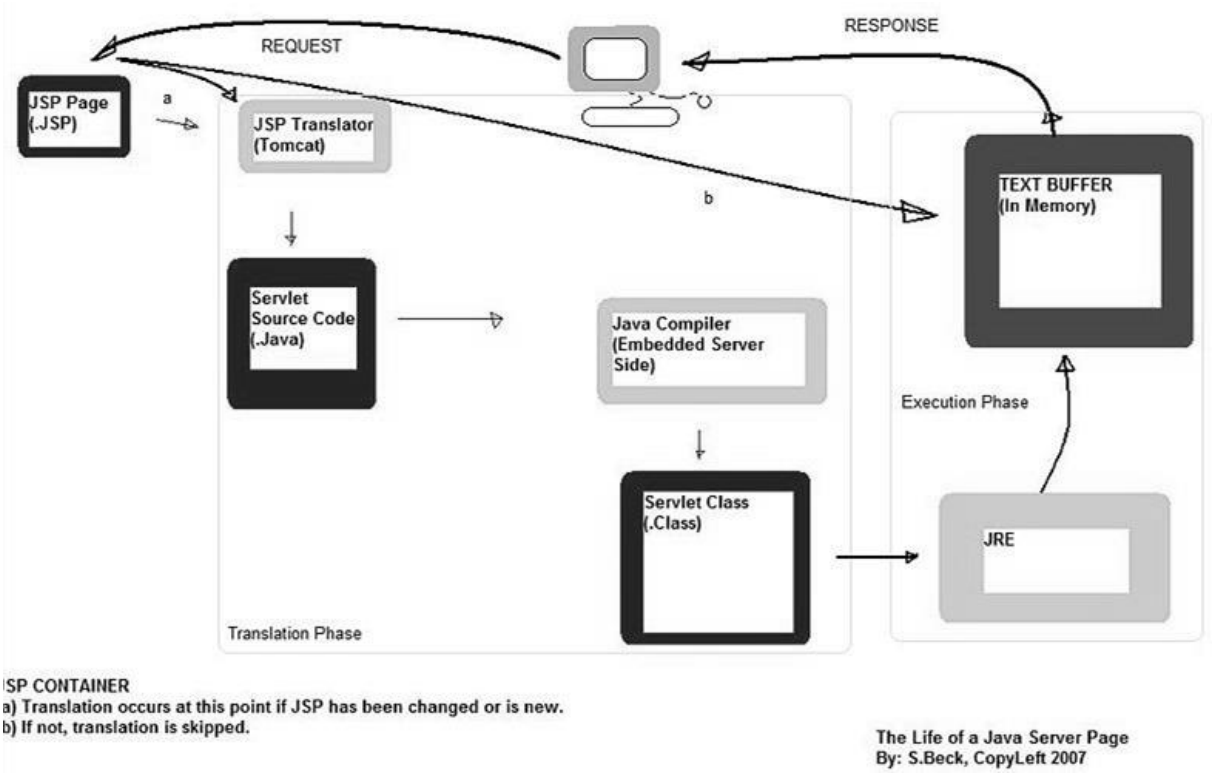

Fig: 1 Architecture of JSP

The Advantages of JSP

Active Server Pages (ASP). ASP is a similar technology from Microsoft. The advantages of JSP are twofold. First, the dynamic part is written in Java, not Visual Basic or other MS-specific language, so it is more powerful and easier to use. Second, it is portable to other operating systems and non-Microsoft Web servers. Pure Servlet. JSP doesn't give you anything that you couldn't in principle do with a Servlet. But it is more convenient to write (and to modify!) regular HTML than to have a zillion println statements that generate the HTML. Plus, by separating the look from the content you can put different people on different tasks: your Web page design experts can build the HTML,

leaving places for your Servlet programmers to insert the dynamic content.

Server-Side Includes (SSI). SSI is a widely-supported technology for including externally-defined pieces into a static Web page. JSP is better because it lets you use Servlet instead of a separate program to generate that dynamic part. Besides, SSI is really only intended for simple inclusions, not for "real" programs that use form data, make database connections, and the like. JavaScript. JavaScript can generate HTML dynamically on the client. This is a useful capability, but only handles situations where the dynamic information is based on the client's environment.

With the exception of cookies, HTTP and form submission data is not available to JavaScript. And, since it runs on the client, JavaScript can't access server-side resources like databases, catalogs, pricing information, and the like. Static HTML. Regular HTML, of course, cannot contain dynamic information. JSP is so easy and convenient that it is quite feasible to augment HTML pages that only benefit marginally by the insertion of small amounts of dynamic data. Previously, the cost of using dynamic data would preclude its use in all but the most valuable instances.

3.3 Servlets

The Java Servlet API allows a software developer to add dynamic content to a Web server using the Java platform. The generated content is commonly HTML, but may be other data such as XML. Servlet are the Java counterpart to non-Java dynamic Web content technologies such as PHP, CGI and ASP.NET. Servlet can maintain state across many server transactions by using HTTP cookies, session variables or URL rewriting.

The Servlet API, contained in the Java package hierarchy javax. Servlet defines the expected interactions of a Web container and a Servlet. A Web container is essentially the component of a Web server that interacts with the Servlet. The Web container is responsible for managing the lifecycle of Servlet, mapping a URL to a particular Servlet and ensuring that the URL requester has the correct access rights.

3.4 The Servlet Run-time Environment

A Servlet is a Java class and therefore needs to be executed in a Java VM by a service we call a Servlet engine. The Servlet engine loads the servlet class the first time the Servlet is requested, or optionally already when the Servlet engine is started. The Servlet then stays loaded to handle multiple requests until it is explicitly unloaded or the Servlet engine is shut down.

Some Web servers, such as Sun's Java Web Server (JWS), W3C's Jigsaw and Gefion Software's Lite Web Server (LWS) are implemented in Java and have a built-in Servlet engine. Other Web servers, such as Netscape's Enterprise Server, Microsoft's Internet Information Server (IIS) and the Apache Group's Apache, require a Servlet engine add-on module. The add-on intercepts all requests for Servlet, executes them and returns the response through the Web server to the client. Examples of Servlet engine add-ons are Gefion Software's WAI Cool Runner, IBM's Web Sphere, Live Software's JRun and New Atlanta's Servlet Exec.

All Servlet API classes and a simple Servlet-enabled Web server are combined into the Java Servlet Development Kit (JSDK), available for download at Sun's official Servlet site .To get started with Servlet I recommend that you download the JSDK and play around with the sample Servlet.

3.5 Life Cycle OF Servlet

- The Servlet lifecycle consists of the following steps:
- The Servlet class is loaded by the container during start-up.

The container calls the **init()** method. This method initializes the Servlet and must be called before the Servlet can service any requests. In the entire life of a Servlet, the **init()** method is called only once. After initialization, the Servlet can service client-requests.

Each request is serviced in its own separate thread. The container calls the **service()** method of the Servlet for every request.

The **service()** method determines the kind of request being made and dispatches it to an appropriate method to handle the request. The developer of the Servlet must provide an implementation for these methods. If a request for a method that is not implemented by the Servlet is made, the method of the parent class is called, typically resulting in an error being returned to the requester. Finally, the container calls the **destroy()** method which takes the Servlet out of service. The **destroy()** method like **init()** is called only once in the lifecycle of a Servlet.

3.5.1 Request and Response Objects

The **do Get** method has two interesting parameters: HttpServletRequest and HttpServletResponse. These two objects give you full access to all information about the request and let you control the output sent to the client as the response to the request. With CGI you read environment variables and stdin to get information about the request, but the names of the environment variables may vary between implementations and some are not provided by all Web servers.

The HttpServletRequest object provides the same information as the CGI environment variables, plus more, in a standardized way. It also provides methods for extracting HTTP parameters from the query string or the request body depending on the type of request (GET or POST). As a Servlet developer you access parameters the same way for both types of requests. Other methods give you access to all request headers and help you parse date and cookie headers.

Instead of writing the response to stdout as you do with CGI; you get an Output Stream or a PrintWriter from the HttpServletResponse. The OuputStream is intended for binary data, such as a GIF or JPEG image, and the PrintWriter for text output. You can also set all response headers and the status code, without having to rely on special Web server CGI configurations such as Non Parsed Headers (NPH). This makes your Servlet easier to install.

3.6 Servlet Config and Servlet Context:

There is only one Servlet Context in every application. This object can be used by all the Servlet to obtain application level information or container details. Every Servlet, on the other hand, gets its own ServletConfig object. This object provides initialization parameters for a servlet. A developer can obtain the reference to Servlet Context using either the ServletConfig object or Servlet Request object. All servlets belong to one servlet context. In implementations of the 1.0 and 2.0 versions of the Servlet API all servlets on one host belongs to the same context, but with the 2.1 version of the API the context becomes more powerful and can be seen as the humble beginnings of an Application concept. Future versions of the API will make this even more pronounced.

Many servlet engines implementing the Servlet 2.1 API let you group a set of

servlets into one context and support more than one context on the same host. The Servlet Context in the 2.1 API is responsible for the state of its servlets and knows about resources and attributes available to the servlets in the context. Here we will only look at how Servlet Context attributes can be used to share information among a group of servlets.

There are three Servlet Context methods dealing with context attributes: get Attribute, set Attribute and remove Attribute. In addition the servlet engine may provide ways to configure a servlet context with initial attribute values. This serves as a welcome addition to the servlet initialization arguments for configuration information used by a group of servlets, for instance the database identifier we talked about above, a style sheet

3.7 JDBC

Java Database Connectivity (JDBC) is a programming framework for Java developers writing programs that access information stored in databases, spreadsheets, and flat files. JDBC is commonly used to connect a user program to a "behind the scenes" database, regardless of what database management software is used to control the database. In this way, JDBC is cross-platform. This article will provide an introduction and sample code that demonstrates database access from Java programs that use the classes of the JDBC API, which is available for free download from Sun's site.

A database that another program links to is called a data source. Many data sources, including products produced by Microsoft and Oracle, already use a standard called Open Database Connectivity (ODBC). Many legacy C and Perl programs use ODBC to connect to data sources. ODBC consolidated much of the commonality between database management systems. JDBC builds on this feature, and increases the level of abstraction. JDBC-ODBC bridges have been created to allow Java programs to connect to ODBC-enabled database software.

JDBC Architecture

Two-tier and three-tier Processing Models

The JDBC API supports both two-tier and three-tier processing models for database access.

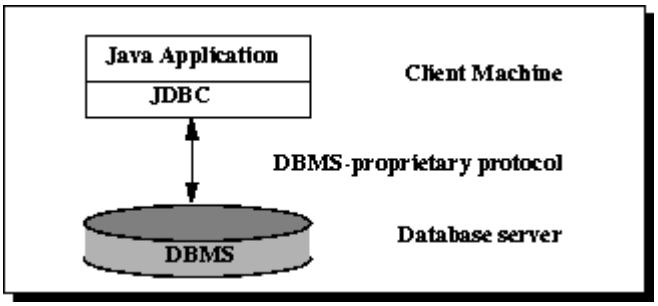

Fig 2: JDBC Architecture

In the two-tier model, a Java applet or application talks directly to the data source. This requires a JDBC driver that can communicate with the particular data source being accessed. A user's commands are delivered to the database or other data source, and the results of those statements are sent back to the user. The data source may be located on another machine to which the user is connected via a network. This is referred to as a client/server configuration, with the user's machine as the client, and the machine housing the data source as the server. The network can be an intranet, which, for example, connects employees within a corporation, or it can be the Internet.

In the three-tier model, commands are sent to a "middle tier" of services, which then sends the commands to the data source. The data source processes the commands and sends the results back to the middle tier, which then sends them to the user.

MIS directors find the three-tier model very attractive because the middle tier makes it possible to maintain control over access and the kinds of updates that can be made to corporate data. Another advantage is that it simplifies the deployment of applications. Finally, in many cases, the three-tier architecture can provide performance advantages.

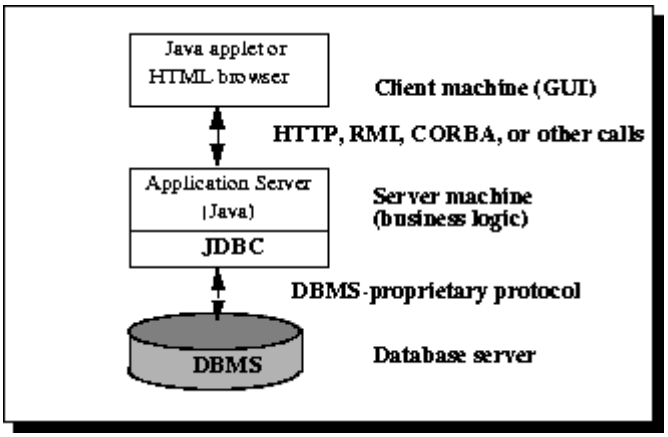

Fig 3: 3-tier Architecture

Until recently, the middle tier has often been written in languages such as C or C++, which offer fast performance. However, with the introduction of optimizing compilers that translate Java byte code into efficient machine-specific code and technologies such as Enterprise JavaBeans™, the Java platform is fast becoming the standard platform for middle-tier development. This is a big plus, making it possible to take advantage of Java's robustness, multithreading, and security features.

With enterprises increasingly using the Java programming language for writing server code, the JDBC API is being used more and more in the middle tier of a three-tier architecture. Some of the features that make JDBC a server technology are its support for connection pooling, distributed transactions, and disconnected row sets. The JDBC API is also what allows access to a data source from a Java middle tier.

4. TESTING

The various levels of testing are

1. White Box Testing
2. Black Box Testing
3. Unit Testing
4. Functional Testing
5. Performance Testing
6. Integration Testing
7. Objective
8. Integration Testing
9. Validation Testing
10. System Testing
11. Structure Testing
12. Output Testing
13. User Acceptance Testing

White Box Testing

White-box testing (also known as **clear box testing**, **glass box testing**, **transparent box testing**, and **structural testing**) is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality (i.e. black-box testing). In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses

inputs to exercise paths through the code and determine the appropriate outputs. This is analogous to testing nodes in a circuit, e.g. in-circuit testing (ICT).

While white-box testing can be applied at the unit, integration and system levels of the software testing process, it is usually done at the unit level. It can test paths within a unit, paths between units during integration, and between subsystems during a system–level test. Though this method of test design can uncover many errors or problems, it might not detect unimplemented parts of the specification or missing requirements.

White-box test design techniques include:

- Control flow testing
- Data flow testing
- Branch testing
- Path testing
- Statement coverage
- Decision coverage

White-box testing is a method of testing the application at the level of the source code. The test cases are derived through the use of the design techniques mentioned above: control flow testing, data flow testing, branch testing, path testing, statement coverage and decision coverage as well as modified condition/decision coverage. White-box testing is the use of these techniques as guidelines to create an error free environment by examining any fragile code.

These White-box testing techniques are the building blocks of white-box testing, whose essence is the careful testing of the application at the source code level to prevent any hidden errors later on. These different techniques exercise every visible path of the source code to minimize errors and create an error-free environment. The whole point of white-box testing is the ability to know which line of the code is being executed and being able to identify what the correct output should be.

Levels

1. Unit testing. White-box testing is done during unit testing to ensure that the code is working as intended, before any integration happens with previously tested code. White-box testing during unit testing catches any defects early on and aids in any defects that happen later on after the code is integrated with the rest of the application and therefore prevents any type of errors later on.

2. Integration testing. White-box testing at this level are written to test the interactions of each interface with each other. The Unit level testing made sure that each code was tested and working accordingly in an isolated environment and integration examines the correctness of the behaviour in an open environment through the use of white-box testing for any interactions of interfaces that are known to the programmer.
3. Regression testing. White-box testing during regression testing is the use of recycled white-box test cases at the unit and integration testing levels.

White-box testing's basic procedures involve the understanding of the source code that you are testing at a deep level to be able to test them. The programmer must have a deep understanding of the application to know what kinds of test cases to create so that every visible path is exercised for testing. Once the source code is understood then the source code can be analysed for test cases to be created. These are the three basic steps that white-box testing takes in order to create test cases:

1. Input, involves different types of requirements, functional specifications, detailed designing of documents, proper source code, security specifications. This is the preparation stage of white-box testing to layout all of the basic information.
2. Processing Unit, involves performing risk analysis to guide whole testing process, proper test plan, execute test cases and communicate results. This is the phase of building test cases to make sure they thoroughly test the application the given results are recorded accordingly.
3. Output, prepare final report that encompasses all of the above preparations and results.

Black Box Testing

Black-box testing is a method of software testing that examines the functionality of an application (e.g. what the software does) without peering into its internal structures or workings (see white-box testing). This method of test can be applied to virtually every level of software testing: unit, integration,system and acceptance. It typically comprises most if not all higher level testing, but can also dominate unit testing as well

Test procedures

Specific knowledge of the application's code/internal structure and programming knowledge in general is not required. The tester is aware of *what* the software is supposed

to do but is not aware of *how* it does it. For instance, the tester is aware that a particular input returns a certain, invariable output but is not aware of *how* the software produces the output in the first place.

Test cases

Test cases are built around specifications and requirements, i.e., what the application is supposed to do. Test cases are generally derived from external descriptions of the software, including specifications, requirements and design parameters. Although the tests used are primarily *functional* in nature, *non-functional* tests may also be used. The test designer selects both valid and invalid inputs and determines the correct output without any knowledge of the test object's internal structure.

Test design techniques

Typical black-box test design techniques include:

- Decision table testing
- All-pairs testing
- State transition tables
- Equivalence partitioning
- Boundary value analysis

Unit testing

In computer programming, **unit testing** is a method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures are tested to determine if they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. In procedural programming, a unit could be an entire module, but is more commonly an individual function or procedure. In object-oriented programming, a unit is often an entire interface, such as a class, but could be an individual method. Unit tests are created by programmers or occasionally by white box testers during the development process.

Ideally, each test case is independent from the others. Substitutes such as method stubs, mock objects, fakes, and test harnesses can be used to assist testing a module in isolation. Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended. Its implementation can vary from being very manual (pencil and paper)to being formalized as part of build automation.

Unit testing should be done in conjunction with other software testing activities, as they can only show the presence or absence of particular errors; they cannot prove a

complete absence of errors. In order to guarantee correct behaviour for every execution path and every possible input, and ensure the absence of errors, other techniques are required, namely the application of formal methods to proving that a software component has no unexpected behavior.

Software testing is a combinatorial problem. For example, every Boolean decision statement requires at least two tests: one with an outcome of "true" and one with an outcome of "false". As a result, for every line of code written, programmers often need 3 to 5 lines of test code.

Unit testing embedded system software presents a unique challenge: Since the software is being developed on a different platform than the one it will eventually run on, you cannot readily run a test program in the actual deployment environment, as is possible with desktop programs.

Functional testing

Functional testing is a quality assurance (QA) process and a type of black box testing that bases its test cases on the specifications of the software component under test. Functions are tested by feeding them input and examining the output, and internal program structure is rarely considered (not like in white-box testing). Functional Testing usually describes *what* the system does.

Functional testing differs from system testing in that functional testing "*verifies* a program by checking it against ... design document(s) or specification(s)", while system testing "*validate* a program by checking it against the published user or system requirements" (Kane, Falk, Nguyen 1999, p. 52).

Functional testing typically involves five steps .The identification of functions that the software is expected to perform

1. The creation of input data based on the function's specifications
2. The determination of output based on the function's specifications
3. The execution of the test case
4. The comparison of actual and expected outputs

Performance testing

In software engineering, **performance testing** is in general testing performed to determine how a system performs in terms of responsiveness and stability under a

particular workload. It can also serve to investigate measure, validate or verify other quality attributes of the system, such as scalability, reliability and resource usage.

Performance testing is a subset of performance engineering, an emerging computer science practice which strives to build performance into the implementation, design and architecture of a system.

Testing types

Load testing

Load testing is the simplest form of performance testing. A load test is usually conducted to understand the behavior of the system under a specific expected load. This load can be the expected concurrent number of users on the application performing a specific number of transactions within the set duration. This test will give out the response times of all the important business critical transactions. If the database, application server, etc. are also monitored, then this simple test can itself point towards bottlenecks in the application software.

Stress testing

Stress testing is normally used to understand the upper limits of capacity within the system. This kind of test is done to determine the system's robustness in terms of extreme load and helps application administrators to determine if the system will perform sufficiently if the current load goes well above the expected maximum.

Soak testing

Soak testing, also known as endurance testing, is usually done to determine if the system can sustain the continuous expected load. During soak tests, memory utilization is monitored to detect potential leaks. Also important, but often overlooked is performance degradation. That is, to ensure that the throughput and/or response times after some long period of sustained activity are as good as or better than at the beginning of the test. It essentially involves applying a significant load to a system for an extended, significant period of time. The goal is to discover how the system behaves under sustained use.

Spike testing

Spike testing is done by suddenly increasing the number of or load generated by, users by a very large amount and observing the behaviour of the system. The goal is to determine whether performance will suffer, the system will fail, or it will be able to handle dramatic changes in load.

Configuration testing

Rather than testing for performance from the perspective of load, tests are created to determine the effects of configuration changes to the system's components on the system's performance and behavior. A common example would be experimenting with different methods of load-balancing.

Isolation testing

Isolation testing is not unique to performance testing but involves repeating a test execution that resulted in a system problem. Often used to isolate and confirm the fault domain.

Integration testing

Integration testing (sometimes called **integration and testing**, abbreviated **I&T**) is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

Purpose

The purpose of integration testing is to verify functional, performance, and reliability requirements placed on major design items. These "design items", i.e. assemblages (or groups of units), are exercised through their interfaces using black box testing, success and error cases being simulated via appropriate parameter and data inputs. Simulated usage of shared data areas and inter-process communication is tested and individual subsystems are exercised through their input interface.

Test cases are constructed to test whether all the components within assemblages interact correctly, for example across procedure calls or process activations, and this is done after testing individual modules, i.e. unit testing. The overall idea is a "building block" approach, in which verified assemblages are added to a verified base which is then used to support the integration testing of further assemblages.

Some different types of integration testing are big bang, top-down, and bottom-up. Other Integration Patterns are: Collaboration Integration, Backbone Integration, Layer Integration, Client/Server Integration, Distributed Services Integration and High-frequency Integration.

Big Bang

In this approach, all or most of the developed modules are coupled together to form a complete software system or major part of the system and then used for integration testing. The Big Bang method is very effective for saving time in the integration testing process. However, if the test cases and their results are not recorded properly, the entire integration process will be more complicated and may prevent the testing team from achieving the goal of integration testing.

A type of Big Bang Integration testing is called **Usage Model testing**. Usage Model Testing can be used in both software and hardware integration testing. The basis behind this type of integration testing is to run user-like workloads in integrated user-like environments. In doing the testing in this manner, the environment is proofed, while the individual components are proofed indirectly through their use.

Usage Model testing takes an optimistic approach to testing, because it expects to have few problems with the individual components. The strategy relies heavily on the component developers to do the isolated unit testing for their product. The goal of the strategy is to avoid redoing the testing done by the developers, and instead flesh-out problems caused by the interaction of the components in the environment.

For integration testing, Usage Model testing can be more efficient and provides better test coverage than traditional focused functional integration testing. To be more efficient and accurate, care must be used in defining the user-like workloads for creating realistic scenarios in exercising the environment. This gives confidence that the integrated environment will work as expected for the target customers.

Top-down and Bottom-up

Bottom up Testing is an approach to integrated testing where the lowest level components are tested first, then used to facilitate the testing of higher level components. The process is repeated until the component at the top of the hierarchy is tested.

All the bottom or low-level modules, procedures or functions are integrated and then tested. After the integration testing of lower level integrated modules, the next level of modules will be formed and can be used for integration testing. This approach is helpful only when all or most of the modules of the same development level are ready. This method also helps to determine the levels of software developed and makes it easier to report testing progress in the form of a percentage.

Top down Testing is an approach to integrated testing where the top integrated modules are tested and the branch of the module is tested step by step until the end of the related module.

Sandwich Testing is an approach to combine top down testing with bottom up testing.

The main advantage of the Bottom-Up approach is that bugs are more easily found. With Top-Down, it is easier to find a missing branch link

5. SYSTEM DESIGN

System design is the process of defining the components, modules, interfaces, and data for a system to satisfy specified requirements. System development is the process of creating or altering systems, along with the processes, practices, models, and methodologies used to develop them.

System design is the subject matter of the architectural domain. The primary activity of this phase is the translation of the OOA models into structural design entities. This includes the specification of mechanisms and structures for managing data, the control of the system as a whole, and performance requirements.

The creation of subsystems is managed by slicing large domains into smaller, manageable portions that can be worked on by two to four persons. Another approach to the creation of subsystems is the clustering of related objects into cohesive entities that can form a well defined interface to other subsystems.

5.1 Architecture Diagram

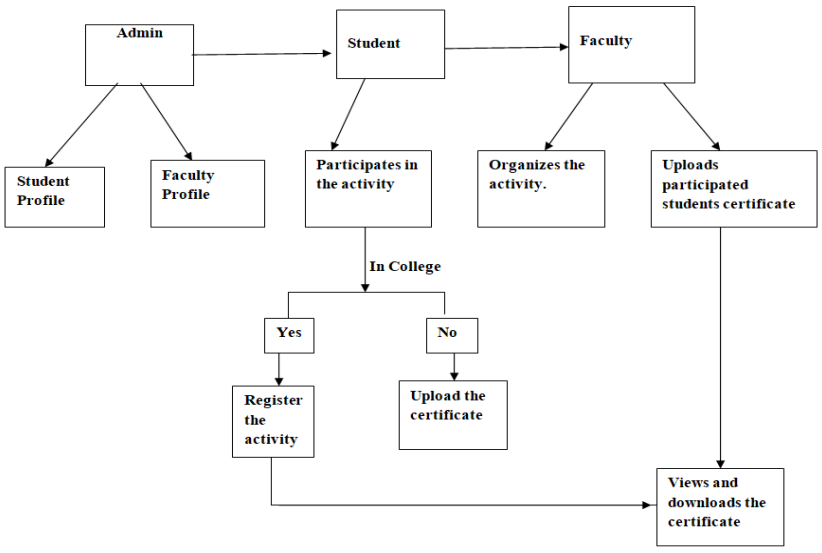

Fig 4: Architecture of College Activities Tracker

6. UML DIAGRAMS

UML is a standardized graphical display format for the visualization, specification, design and documentation of (software) systems. It offers a set of standardized diagram types with which complex data, processes and systems can easily be arranged in a clear, intuitive manner..

6.1 Class Diagram

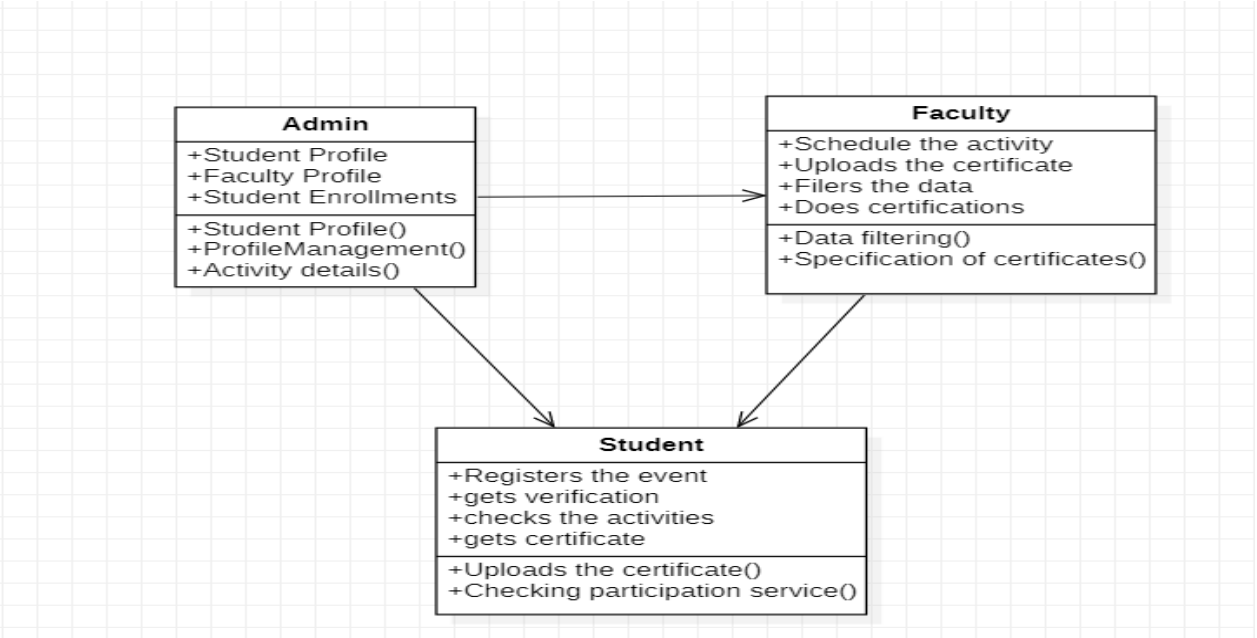

Fig 5: Class diagram

6.2 Sequence Diagram

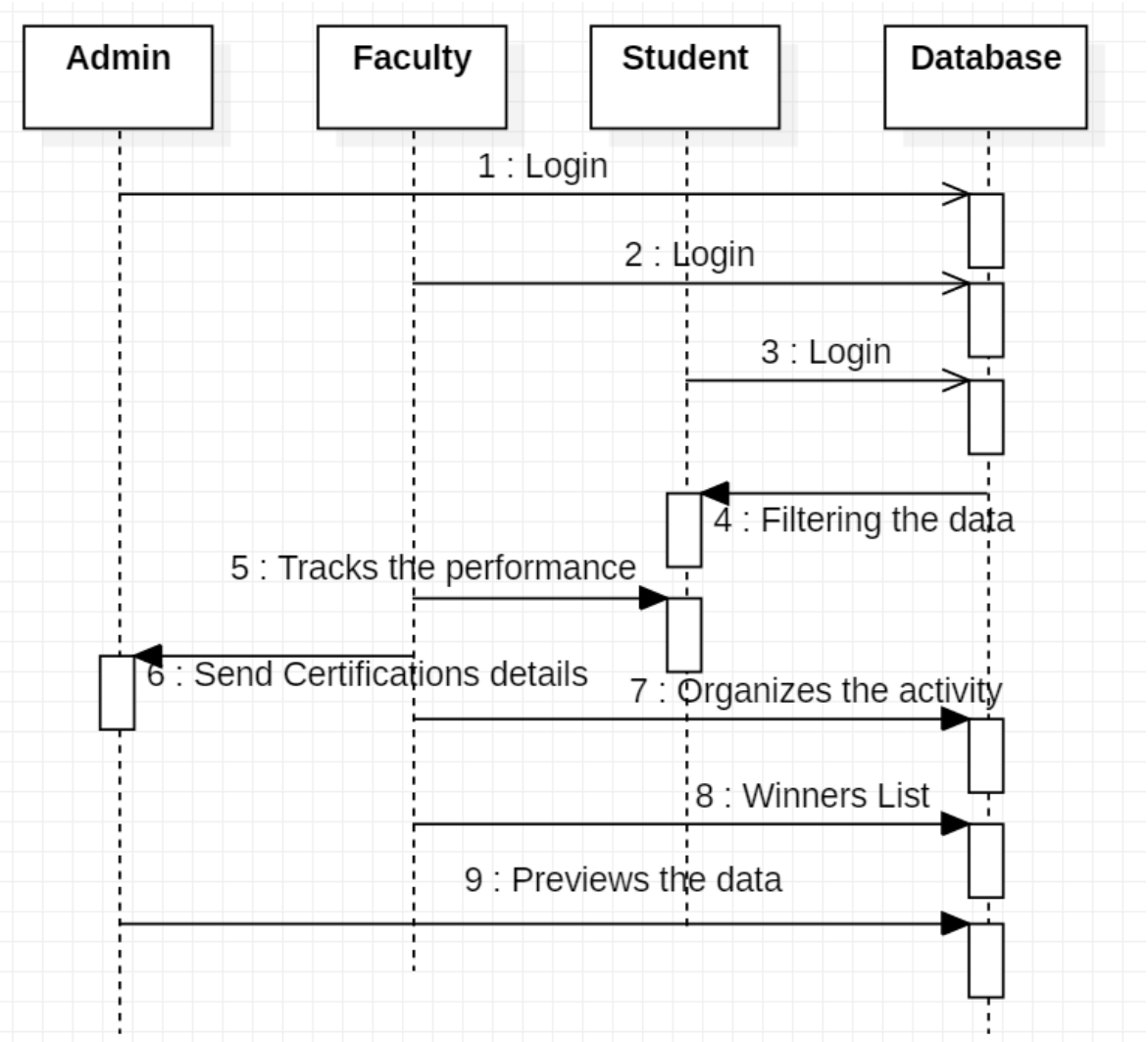

Fig 6: Sequence diagram

6.3 Activity Diagram

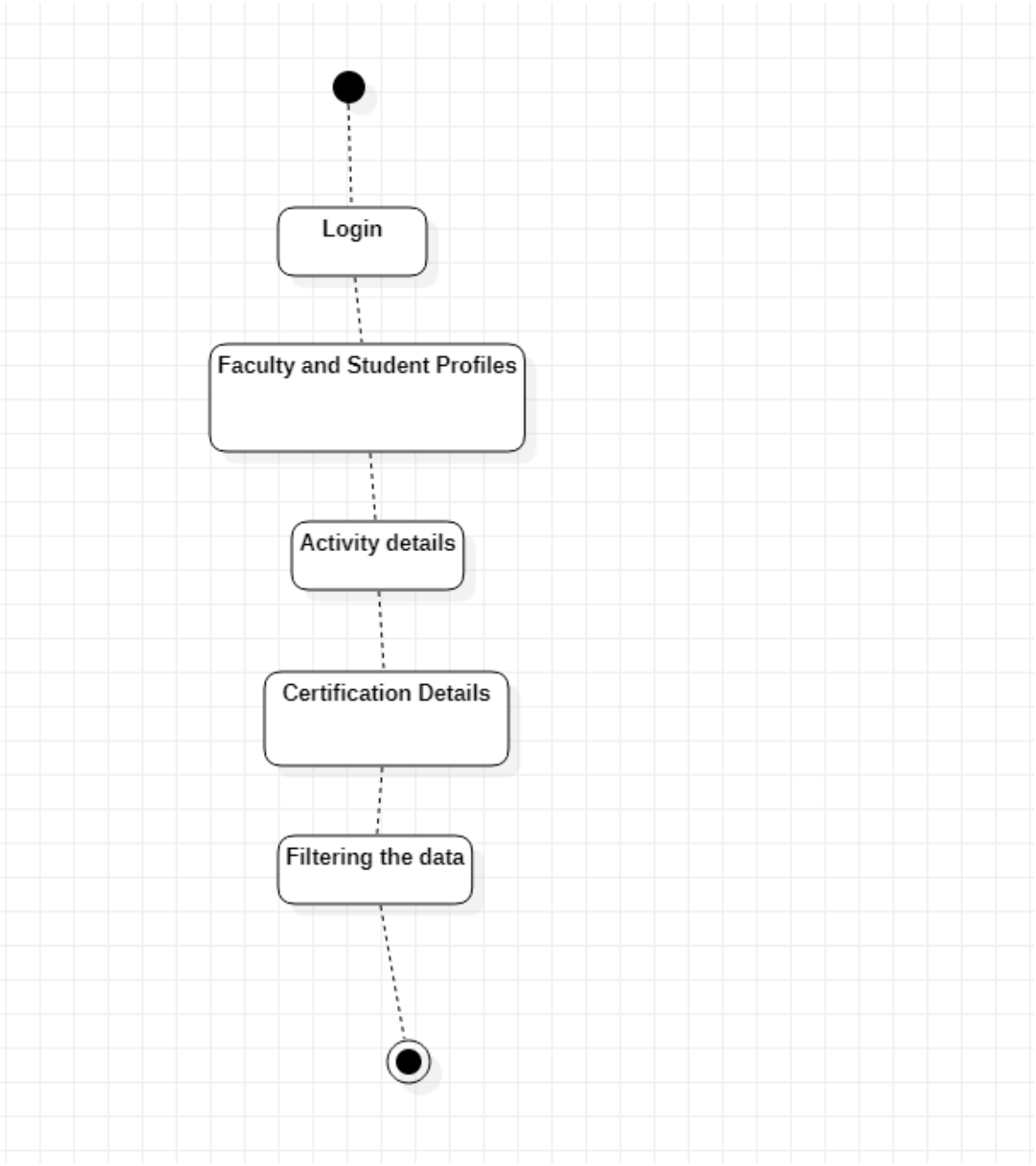

Fig 7: Activity diagram

6.4 Use Case Diagram

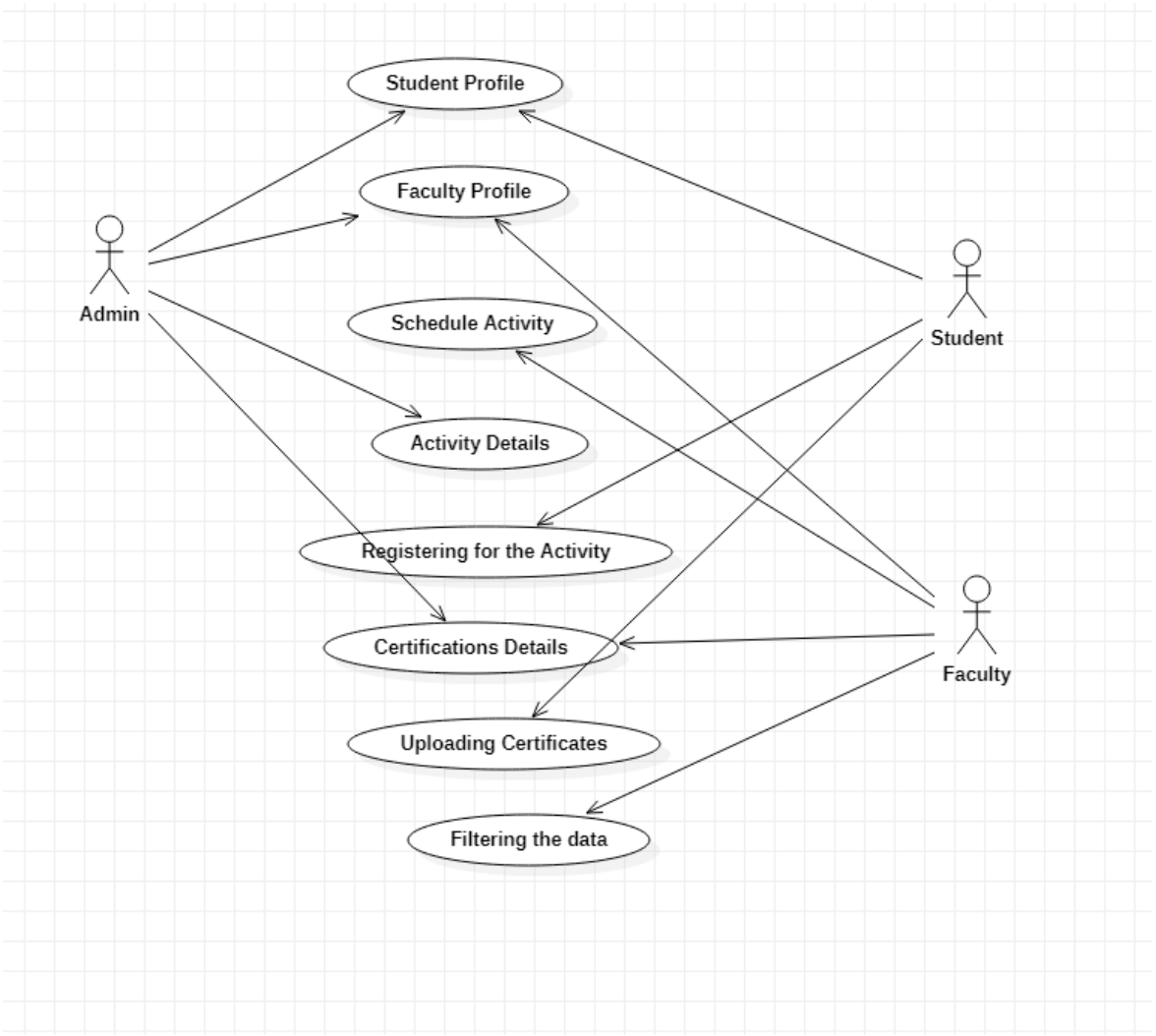

Fig 8: Use Case diagram

7. Verification and Validation

Verification and Validation are independent procedures that are used together for checking that a product, service, or system meets requirements and specifications and that it full fills its intended purpose. These are critical components of a quality management system such as ISO 9000. The words "verification" and "validation" are sometimes preceded with "Independent" (or IV&V), indicating that the verification and validation is to be performed by a disinterested third party.

It is sometimes said that validation can be expressed by the query "Are you building the right thing?" and verification by "Are you building it right?"In practice, the usage of these terms varies. Sometimes they are even used interchangeably.

The PMBOK guide, an IEEE standard, defines them as follows in its 4th edition

- "**Validation.** The assurance that a product, service, or system meets the needs of the customer and other identified stakeholders. It often involves acceptance and suitability with external customers. Contrast with *verification*."
- "**Verification**. The evaluation of whether or not a product, service, or system complies with a regulation, requirement, specification, or imposed condition. It is often an internal process. Contrast with *validation*."
    - Verification is intended to check that a product, service, or system (or portion thereof, or set thereof) meets a set of initial design specifications. In the development phase, verification procedures involve performing special tests to model or simulate a portion, or the entirety, of a product, service or system, then performing a review or analysis of the modeling results. In the post-development phase, verification procedures involve regularly repeating tests devised specifically to ensure that the product, service, or system continues to meet the initial design requirements, specifications, and regulations as time progresses. It is a process that is used to evaluate whether a product, service, or system complies with regulations, specifications, or conditions imposed at the start of a development phase. Verification can be in development, scale-up, or production. This is often an internal process.
    - Validation is intended to check that development and verification procedures for a product, service, or system (or portion thereof, or set thereof) result in a product, service, or system (or portion thereof, or set thereof) that meets initial requirements. For a new development flow or verification flow, validation

procedures may involve modeling either flow and using simulations to predict faults or gaps that might lead to invalid or incomplete verification or development of a product, service, or system (or portion thereof, or set thereof). A set of validation requirements, specifications, and regulations may then be used as a basis for qualifying a development flow or verification flow for a product, service, or system (or portion thereof, or set thereof). Additional validation procedures also include those that are designed specifically to ensure that modifications made to an existing qualified development flow or verification flow will have the effect of producing a product, service, or system (or portion thereof, or set thereof) that meets the initial design requirements, specifications, and regulations; these validations help to keep the flow qualified. It is a process of establishing evidence that provides a high degree of assurance that a product, service, or system accomplishes its intended requirements. This often involves acceptance of fitness for purpose with end users and other product stakeholders. This is often an external process.

7.1 Activities

Verification of machinery and equipment usually consists of design qualification (DQ), installation qualification (IQ), operational qualification (OQ), and performance qualification (PQ). DQ is usually a vendor's job. However, DQ can also be performed by the user, by confirming through review and testing that the equipment meets the written acquisition specification. If the relevant document or manuals of machinery/equipment are provided by vendors, the later 3Q needs to be thoroughly performed by the users who work in an industrial regulatory environment. Otherwise, the process of IQ, OQ and PQ is the task of validation. The typical example of such a case could be the loss or absence of vendor's documentation for legacy equipment or do-it-yourself (DIY) assemblies (e.g., cars, computers etc.) and, therefore, users should endeavour to acquire DQ document beforehand. Each template of DQ, IQ, OQ and PQ usually can be found on the internet respectively, whereas the DIY qualifications of machinery/equipment can be assisted either by the vendor's training course materials and tutorials, or by the published guidance books, such as *step-by-step* series if the acquisition of machinery/equipment is not bundled with on- site qualification services. This kind of the DIY approach is also applicable to the qualifications of software, computer operating systems and a manufacturing process. The most important and critical task as the last step of the activity is to generating and archiving machinery/equipment qualification reports for auditing purposes, if regulatory compliances are mandatory.

Qualification of machinery/equipment is venue dependent, in particular items that are shock sensitive and require balancing or calibration, and re-qualification needs to be conducted once the objects are relocated. The full scales of some equipment qualifications are even time dependent as consumables are used up (i.e. filters) or springs stretch out, requiring recalibration, and hence re-certification is necessary when a specified due time lapse Re-qualification of machinery/equipment should also be conducted when replacement of parts, or coupling with another device, or installing a new application software and restructuring of the computer which affects especially the pre-settings, such as on BIOS, registry, disk drive partition table, dynamically-linked (shared) libraries, or an ini file etc., have been necessary. In such a situation, the specifications of the parts/devices/software and restructuring proposals should be appended to the qualification document whether the parts/devices/software are genuine or not.

Torres and Hyman have discussed the suitability of non-genuine parts for clinical use and provided guidelines for equipment users to select appropriate substitutes which are capable to avoid adverse effects. In the case when genuine parts/devices/software are demanded by some of regulatory requirements, then re-qualification does not need to be conducted on the non-genuine assemblies. Instead, the asset has to be recycled for non-regulatory purposes.

When machinery/equipment qualification is conducted by a standard endorsed third party such as by an ISO standard accredited company for a particular division, the process is called certification. Currently, the coverage of ISO/IEC 15408 certification by an ISO/IEC 27001 accredited organization is limited; the scheme requires a fair amount of efforts to get popularized.

8. SYSTEM TESTING

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic.

As a rule, system testing takes, as its input, all of the "integrated" software components that have passed integration testing and also the software system itself integrated with any applicable hardware system(s). The purpose of integration testing is to detect any inconsistencies between the software units that are integrated together (called *assemblages*) or between any of the *assemblages* and the hardware. System testing is a more limited type of testing; it seeks to detect defects both within the "inter-assemblages" and also within the system as a whole.

System testing is performed on the entire system in the context of a Functional Requirement Specification(s) (FRS) and/or a System Requirement Specification (SRS). System testing tests not only the design, but also the behavior and even the believed expectations of the customer. It is also intended to test up to and beyond the bounds defined in the software/hardware requirements specification

Types of tests to include in system testing

The following examples are different types of testing that should be considered during System testing:

- Graphical user interface testing
- Usability testing
- Software performance testing
- Compatibility testing
- Exception handling
- Load testing
- Volume testing
- Stress testing
- Security testing
- Scalability testing
- Sanity testing
- Smoke testing

- Exploratory testing
- Ad hoc testing
- Regression testing
- Installation testing
- Maintenance testing Recovery testing and failover testing.
- Accessibility testing, including compliance with:
  - Americans with Disabilities Act of 1990
  - Section 508 Amendment to the Rehabilitation Act of 1973
  - Web Accessibility Initiative (WAI) of the World Wide Web Consortium (W3C)

Although different testing organizations may prescribe different tests as part of System testing, this list serves as a general framework or foundation to begin with.

Structure Testing:

It is concerned with exercising the internal logic of a program and traversing particular execution paths.

Output Testing:

- Output of test cases compared with the expected results created during design of test cases.
- Asking the user about the format required by them tests the output generated or displayed by the system under consideration.
- Here, the output format is considered into two was, one is on screen and another one is printed format.
- The output on the screen is found to be correct as the format was designed in the system design phase according to user needs.
- The output comes out as the specified requirements as the user's hard copy.

User acceptance Testing:

- Final Stage, before handling over to the customer which is usually carried out by the customer where the test cases are executed with actual data.
- The system under consideration is tested for user acceptance and constantly keeping touch with the prospective system user at the time of developing and making changes whenever required.
- It involves planning and execution of various types of test in order to demonstrate that the implemented software system satisfies the requirements stated in the requirement document.

9. IMPLEMENTATION

StudentRegister.java

```
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Properties;
import java.util.Random;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.Multipart;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;
import javax.mail.internet.MimeMultipart;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

/**
 *
 * @author user
 */
@WebServlet(urlPatterns = {"/studentRegservlet"})
public class studentRegservlet extends HttpServlet {

    static Properties properties = new Properties();
   static
   {
     properties.put("mail.smtp.host", "smtp.gmail.com");
     properties.put("mail.smtp.socketFactory.port", "465");
     properties.put("mail.smtp.socketFactory.class",
                    "javax.net.ssl.SSLSocketFactory");
     properties.put("mail.smtp.auth", "true");
     properties.put("mail.smtp.port", "465");
   }
    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            /* TODO output your page here. You may use following sample code. */
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet studentRegservlet</title>");
```

```
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Servlet studentRegservlet at " + request.getContextPath() + "</h1>");
            out.println("</body>");
            out.println("</html>");
        }
    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        HttpSession session1=request.getSession();
            Connection con=null;
            Statement st=null;
            ResultSet rs1=null;

            try
            {
            String reguser=request.getParameter("username");
            //String regpass=request.getParameter("password");
            String reggender=request.getParameter("gender");
            String regdob=request.getParameter("dob");
            String regdob1=request.getParameter("dob1");
            String regdob2=request.getParameter("dob2");
            String regage=request.getParameter("age");
            String regcollegename=request.getParameter("collegename");
           // String regaddress=request.getParameter("address");
            //String regstate=request.getParameter("state");
            String regnation=request.getParameter("nation");
            String regmobile=request.getParameter("mobile");
            String regemail=request.getParameter("email");


            Class.forName("com.mysql.jdbc.Driver");
            con=DriverManager.getConnection("jdbc:mysql://localhost:3306/events","root","password");
            st=con.createStatement();
            int rs=st.executeUpdate("Insert into studentregservlet(username,dob,gender,dept,collegename,email,mobile,nation) VALUES('"+reguser+"','"+regdob+","+regdob1+","+regdob2+"','"+reggender+"','"+regage+"','"+regcollegename+"','"+regemail+"','"+regmobile+"','"+regnation+"')");
            if(rs>0)
```

```
            {
                response.sendRedirect("index.jsp");
                final String  from="afreenkhan2199@gmail.com";
                final String password="KHANS@123";

                Session session = Session.getInstance(properties, new javax.mail.Authenticator()
                {
                  @Override
                  protected PasswordAuthentication getPasswordAuthentication() {
                  return new PasswordAuthentication(from, password);
                  }});

                Message message = new MimeMessage(session);
                message.setFrom(new InternetAddress(from));
                message.setRecipients(Message.RecipientType.TO,
                InternetAddress.parse(regemail));
                message.setText("Successfully Registered for the event "+"\n"+"Kindly check below your Username and Event Name"+"\n"+"USERNAME: "+reguser+"\n"+"EVENT: "+regnation);
                Multipart multipart = new MimeMultipart();
                Transport.send(message);

            }
            else
            {
                response.sendRedirect("error.jsp");
            }
        }catch(ClassNotFoundException | SQLException e){
            System.out.println(e);
        } catch (MessagingException ex) {
            Logger.getLogger(studentRegservlet.class.getName()).log(Level.SEVERE, null, ex);
        }

    }



    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>

}
```

StudenRegister.jsp

```
<%--
    Document   : studentregistration
    Created on : May 19, 2021, 4:15:47 AM
    Author     : user
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.Connection"%>
<!DOCTYPE html>
<html>
```

```
   <head>
      <title>event management Page</title>
      <style type="text/css">

h3{font-family: Calibri; font-size: 22pt; font-style: normal; font-weight: bold; color:SlateBlue;
text-align: center; text-decoration: underline }
table{font-family: Calibri; color:white; font-size: 11pt; font-style: normal;
text-align:center; background-color: SlateBlue; border-collapse: collapse; border: 2px solid navy}
table.inner{border: 0px}body {
              background: #2F0916;
}
    </style>
   </head>
   <body>
      <center>
         <form name="form1" method="post" action="studentRegservlet">
            <br>
            </br>
            <table>
               <tr>
                  <td>
                     <div align="center">
                        <font size="10" face="Colonna MT">student Registration Form</font>
                     </div>
                  </td>
               </tr>
            </table>
            <br>
            </br>

<table align="center" cellpadding = "10">

<!----- First Name ----------------------------------------------------------->
<tr>
<td>
<div align="center">USERNAME</div>
</td>
<td>
<label for="username"></label>
<input type="text" name="username" id="username" maxlength="30">
(Max 30 Characters a-z And A-Z)
</td>
</tr>

<!----- Last Name ----------------------------------------------------------->

<!----- Date Of Birth ----------------------------------------------------------->
<tr>
   <td><div align="center">DATE OF BIRTH</div></td>

<td>
<select name="dob" id="dob">
<option value="-1">Day:</option>
<option value="1">1</option>
<option value="2">2</option>
<option value="3">3</option>

<option value="4">4</option>
<option value="5">5</option>
<option value="6">6</option>
<option value="7">7</option>
<option value="8">8</option>
<option value="9">9</option>
<option value="10">10</option>
<option value="11">11</option>
<option value="12">12</option>
```

```
<option value="13">13</option>
<option value="14">14</option>
<option value="15">15</option>
<option value="16">16</option>
<option value="17">17</option>
<option value="18">18</option>
<option value="19">19</option>
<option value="20">20</option>
<option value="21">21</option>

<option value="22">22</option>
<option value="23">23</option>
<option value="24">24</option>
<option value="25">25</option>
<option value="26">26</option>
<option value="27">27</option>
<option value="28">28</option>
<option value="29">29</option>
<option value="30">30</option>

<option value="31">31</option>
</select>

<select id="dob1" name="dob1">
<option value="-1">Month:</option>
<option value="January">Jan</option>
<option value="February">Feb</option>
<option value="March">Mar</option>
<option value="April">Apr</option>
<option value="May">May</option>
<option value="June">Jun</option>
<option value="July">Jul</option>
<option value="August">Aug</option>
<option value="September">Sep</option>
<option value="October">Oct</option>
<option value="November">Nov</option>
<option value="December">Dec</option>
</select>

<select name="dob2" id="dob2">

<option value="-1">Year:</option>
<option value="2012">2014</option>
<option value="2011">2013</option>
<option value="2012">2012</option>
<option value="2011">2011</option>
<option value="2010">2010</option>
<option value="2009">2009</option>
<option value="2008">2008</option>
<option value="2007">2007</option>
<option value="2006">2006</option>
<option value="2005">2005</option>
<option value="2004">2004</option>
<option value="2003">2003</option>
<option value="2002">2002</option>
<option value="2001">2001</option>
<option value="2000">2000</option>

<option value="1999">1999</option>
<option value="1998">1998</option>
<option value="1997">1997</option>
<option value="1996">1996</option>
<option value="1995">1995</option>
<option value="1994">1994</option>
<option value="1993">1993</option>
```

```
<option value="1992">1992</option>
<option value="1991">1991</option>
<option value="1990">1990</option>

<option value="1989">1989</option>
<option value="1988">1988</option>
<option value="1987">1987</option>
<option value="1986">1986</option>
<option value="1985">1985</option>
<option value="1984">1984</option>
<option value="1983">1983</option>
<option value="1982">1982</option>
<option value="1981">1981</option>
<option value="1980">1980</option>
</select>
</td>
</tr>
 <!-----------------Gender---------------------------------------------------->
 <tr>
                    <td>
                       <div align="center">GENDER</div>
                    <td>
<label for="gender"></label>
<input type="text" name="gender" id="gender">
</td>
</tr>

<!-----------------college name---------------------------------------------------->
 <tr>
                    <td>
                       <div align="center">COLLEGE NAME</div>
                    <td>
<label for="college name"></label>
<input type="text" name="college name" id="college name">
</td>
</tr>


<!-------------------AGE---------------------------------------------------->
<tr>
<td>
<div align="center">Department</div>
<td>
<label for="age"></label>
<input type="text" name="age" id="age">
</td>
</tr>

<!----- Email Id ---------------------------------------------------------->
 <tr>
<td>
<div align="center">EMAIL ID</div>
</td>
<td>
<label for="email"></label>
<input type="text" name="email" id="email">
</td>
</tr>

<!----- Mobile Number ----------------------------------------------------------->
<td>
<div align="center">MOBILE</div>
</td>
<td>
<label for="mobile"></label>
<input type="text" name="mobile" id="mobile" maxlength="10"/>
```

```
(10 Digit Number)
</td>
</tr>
<!----- City ---------------------------------------------------------->

<!----- Pin Code ---------------------------------------------------------->

<!----- Country ---------------------------------------------------------->
<tr>
<td>
<div align="center">Select Event</div>
</td>
<td>
<label for="nation"></label>
<select name="nation" id="nation">
   <option value="web mining">----select----</option>
   <%
      try{
         String query="SELECT name FROM prinlog  ";
               Class.forName("com.mysql.jdbc.Driver");
               Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/events","root","password");
               Statement st=con.createStatement();
               ResultSet rs1=st.executeQuery(query);
               while(rs1.next()){
                  %>
                  <option><%=rs1.getString("name")%></option>


                  <%

               }
      }
      catch(Exception ex){
         ex.printStackTrace();
         out.println("Error" +ex.getMessage());
      }
      %>

</select>
</td>
</tr>
<center>
   <table>
<tr>
<td>
<div align="center">
<input type="submit" name="submit" id="submit" value="submit">
</div>
</td>
</tr>
</center>
  </table>
</table>

      </center>
   </body>
</html>
```

Certificate Upload Code

AdUpCer.java

```
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.servlet.ServletException;
import javax.servlet.annotation.MultipartConfig;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import javax.servlet.http.Part;

/**
 *
 * @author LENOVO
 */

@WebServlet(urlPatterns = {"/FirUpload"})
@MultipartConfig(fileSizeThreshold=1024*1024*10,     // 10 MB
                 maxFileSize=1024*1024*50,         // 50 MB
                 maxRequestSize=1024*1024*100)
public class AdUpCer extends HttpServlet {
    InputStream is = null;
    String fileName;

    private static final long serialVersionUID = 1L;
    //File directory = new File("D:/temp/");
    //private final String UPLOAD_DIRECTORY = "D:/Files/";
    //ServletFileUpload upload;
    String names=null;
    Part names1;
    String fsize,ftype;

    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            /* TODO output your page here. You may use following sample code. */
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet AdUpCer</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Servlet AdUpCer at " + request.getContextPath() + "</h1>");
            out.println("</body>");
            out.println("</html>");
        }
    }
```

```
    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {


        HttpSession se=request.getSession();

        String dpt=se.getAttribute("dpt").toString();
        System.out.println(">>>>"+dpt);
        String year=se.getAttribute("year").toString();
        System.out.println(">>>>"+dpt);

        try{

            System.out.println("Going to condition");
      Part filePart = request.getPart("file1");
      String ff=filePart.getName();
      System.out.println(">>>>"+filePart.getName());

       fileName = getFileName(filePart);
          System.out.println("FileName:3 4e------>"+fileName);
      InputStream is = filePart.getInputStream();
      System.out.println(fileName);
      String outputfile = this.getServletContext().getRealPath("/");
      String outputfile1=outputfile.concat(dpt+"/"+fileName);
      System.out.println("......"+outputfile1);
      FileOutputStream os = new FileOutputStream (outputfile1);
      int ch = is.read();
            while (ch != -1) {
                os.write(ch);
                ch = is.read();
            }
            os.close();
            Class.forName("com.mysql.jdbc.Driver");
           Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/events","root","password");
           Statement st=con.createStatement();
            //int rs=st.executeUpdate("crime", "insert into crecd(cname,cage,cadd,filename) values('"+uname+"','"+cage+"','"+cadd+"','"+fileName+"')");

           int rs=st.executeUpdate("insert into certificate (filename,deptnm,year) values('"+fileName+"','"+dpt+"','"+year+"')");
```

```
        if(rs>0){
            response.sendRedirect("DeptSel.jsp");

        }


       }catch(IOException | ServletException e){

          System.out.println(e);

       } catch (ClassNotFoundException ex) {
          Logger.getLogger(AdUpCer.class.getName()).log(Level.SEVERE, null, ex);
       } catch (SQLException ex) {
          Logger.getLogger(AdUpCer.class.getName()).log(Level.SEVERE, null, ex);
       }


    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>
 private String getFileName(Part part) {
        String contentDisp = part.getHeader("content-disposition");
        System.out.println("content-disposition header= "+contentDisp);
        String[] tokens = contentDisp.split(";");
        for (String token : tokens) {
            if (token.trim().startsWith("filename")) {
                return token.substring(token.indexOf("=") + 2, token.length()-1);
            }
        }
        return "";
    }
}
```

AdCerUP.jsp

```
        <%--
    Document   : FirUpl
    Created on : 1 Feb, 2019, 4:59:51 PM
    Author     : Prabu
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
        <meta name="description" content="Demo for the tutorial: Styling and Customizing File Inputs the Smart Way" />
                            <meta name="keywords" content="cutom file input, styling, label, cross-browser, accessible, input type file" />
                            <meta name="author" content="Osvaldas Valutis for Codrops" />
                            <link rel="shortcut icon" href="favicon.ico">
                            <link rel="stylesheet" type="text/css" href="css/normalize_2.css" />
                            <link rel="stylesheet" type="text/css" href="css/demo_2.css" />
                            <link rel="stylesheet" type="text/css" href="css/component_1.css" />
    </head>
```

```
    <script>(function(e,t,n){var   r=e.querySelectorAll("html")[0];r.className=r.className.replace(/(^|\s)no-
js(\s|$)/,"$1js$2")})(document,window,0);</script>
            <style>
.button {
  padding: 15px 25px;
  font-size: 15px;
  text-align: center;
  cursor: pointer;
  outline: none;
  color: #000000;
  background-color: #F5B7B1;
  border: none;
  border-radius: 15px;
  box-shadow: 0 9px #999;
}

.button:hover {background-color: #F5B7B1}

.button:active {
  background-color: #F5B7B1;
  box-shadow: 0 5px #666;
  transform: translateY(4px);
}
</style>
      </head>
      <%

      HttpSession se=request.getSession();

      String dpt=request.getParameter("dept");

      se.setAttribute("dpt", dpt);

      %>

      <script>
         function valid()
         {
            var a1=document.form1.file1.value;

            if(a1=="")
               {
                  alert("Choose file");
                  return false;
               }
               return true;
         }
      </script>




   <body>
      <form action="AdUpCer" method="post" name="form1" id="form1" enctype="multipart/form-data">
                        <div class="container">
                                 <header class="codrops-header">
                  <h1><font size="8" face="Californian FB"> Upload Page</font></h1>

                                 </header>
                                 <div class="content">


                                          <div class="box">
                                                   <input     type="file"     name="file1"     id="file-5"
class="inputfile inputfile-4" data-multiple-caption="{count} files selected" multiple />
                           <button name="Submit" class="button">Upload</button>
```

```
                    </div>

                                </div>
                        </div><!-- /container -->

                        <script src="js/custom-file-input.js"></script>

            </form>
    </body>
</html>
```

Certificate Download Code

Cerdown.java

```
import java.io.FileInputStream;
import java.io.IOException;
import java.io.PrintWriter;
import static java.lang.System.out;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.text.ParseException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

/**
 *
 * @author LENOVO
 */
public class Cerdown extends HttpServlet {

    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            /* TODO output your page here. You may use following sample code. */
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet Cerdown</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Servlet Cerdown at " + request.getContextPath() + "</h1>");
            out.println("</body>");
            out.println("</html>");
        }
```

```
    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit
the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {


        HttpSession se=request.getSession();
        PrintWriter out = response.getWriter();
        String s1=request.getParameter("c1");
        try{

            Class.forName("com.mysql.jdbc.Driver");
            Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/events","root","password");
            Statement st=con.createStatement();
            ResultSet rs=st.executeQuery("select * from certificate where sno='"+s1+"'");
            if(rs.next()){
            String filename=rs.getString("filename");
             String dpt=rs.getString("deptnm");
                System.out.println("FileName: "+filename);

            String outputfile = this.getServletContext().getRealPath(dpt+"/"+filename);
                FileInputStream filetoDownload=new FileInputStream(outputfile);
                response.setHeader("Content-Disposition","inline; filename="+filename);
                response.setContentType("application/x-msdownload");
                int i;
                while ((i=filetoDownload.read()) != -1) {
                out.write(i);

               // response.sendRedirect("downldcer.jsp");

                }
                filetoDownload.close();
            out.close();
            }
        }catch(ClassNotFoundException | SQLException | IOException e){

            System.out.println(e);

        }
```

```
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>

}
```

downldcer.jsp

```
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.Connection"%>
<!DOCTYPE html>
<html lang="en" class="no-js">
<head>
<meta charset="UTF-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Student information</title>
<meta name="description" content="Sticky Table Headers Revisited: Creating functional and flexible sticky
table headers" />
<meta name="keywords" content="Sticky Table Headers Revisited" />
<meta name="author" content="Codrops" />
<link rel="shortcut icon" href="../favicon.ico">
<link rel="stylesheet" type="text/css" href="css/normalize.css" />
<link rel="stylesheet" type="text/css" href="css/demo.css" />
<link rel="stylesheet" type="text/css" href="css/component.css" />
<!--[if IE]>
<script src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>
<![endif]-->
</head>
<style>
.button {
display: inline-block;
border-radius: 4px;
background-color: #000000;
border: none;
color: #FFFFFF;
text-align: center;
font-size: 20px;
padding: 5px;
width: 200px;
transition: all 0.5s;
cursor: pointer;
margin: 5px;
}

.button span {
cursor: pointer;
display: inline-block;
position: relative;
transition: 0.5s;
}

.button span:after {
content: '\00bb';
```

```
position: absolute;
opacity: 0;
top: 0;
right: -20px;
transition: 0.5s;
}

.button:hover span {
padding-right: 25px;
}

.button:hover span:after {
opacity: 1;
right: 0;
}
</style>


<body align="center">
<form name="formm" action="downldcer.jsp" method="POST" onsubmit="return one()">
<div class="container">
<!-- Top Navigation -->

<header>
<h1>Certificate Download</h1>
</header>

<nav class="codrops-demos">
<a>Student Regno :
<input type ="text" name="dept" style = "height:50px; width: 100px"/>

</a>    <br><br>

<button class="button" type="submit" name="Student Status" style="vertical-
align:middle"><span>Download Certificate </span></button>

</nav>

<div class="component">

<table>
<thead>
<tr>


<th>Select</th>
<th>Name</th>
<th>EventName</th>

<th>Department</th>
<th>Year</th>

<th>File Name</th>


</tr>
</thead>
<tbody>
<%
try{
String d1=request.getParameter("dept");

String query="SELECT * FROM certificate WHERE deptnm='"+d1+"' ";
Class.forName("com.mysql.jdbc.Driver");
Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/events","root","password");
Statement st=con.createStatement();
```

```
ResultSet rs1=st.executeQuery(query);
while(rs1.next())


{
%>
<tr>
<td><input type="radio" name="c1" value="<%=rs1.getString("sno")%>" > </td>

<td><%=rs1.getString("name") %></td>
<td><%=rs1.getString("eventname") %></td>
<td><%=rs1.getString("year") %></td>

<td><%=rs1.getString("deptnm") %></td>
<td><%=rs1.getString("filename") %></td>


</tr>
<%

}


}catch(Exception e){

System.out.println(e);
}


%>
</tbody></table>

<button class="button" type="submit" name="Book Search" style="vertical-align:middle" onclick="form.action='Cerdown';"><span>Download It </span></button>


</div>

</div><!-- /container -->

<script src="http://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js"></script>
<script src="http://cdnjs.cloudflare.com/ajax/libs/jquery-throttle-debounce/1.1/jquery.ba-throttle-debounce.min.js"></script>
<script src="js/jquery.stickyheader.js"></script>
</form>
<a href="index.jsp" class="button" style="vertical-align:middle">Exist</a>
</body>
</html>
```

10. OUTPUT

10.1 Screenshots of output:

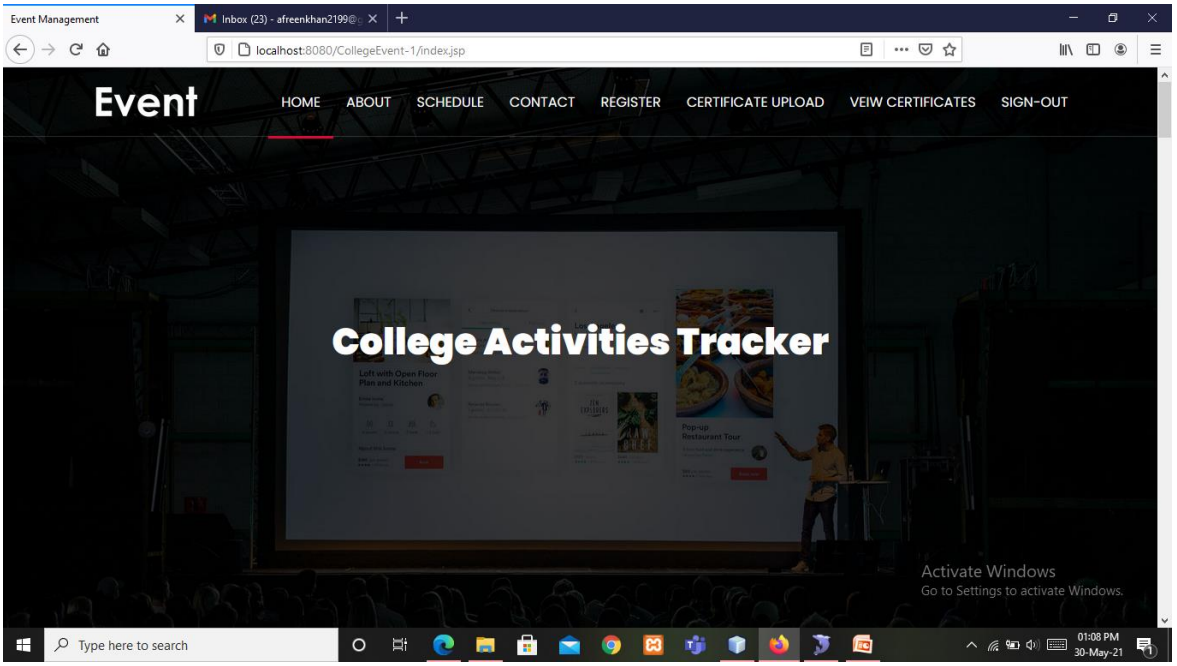

Fig 9: Home Page

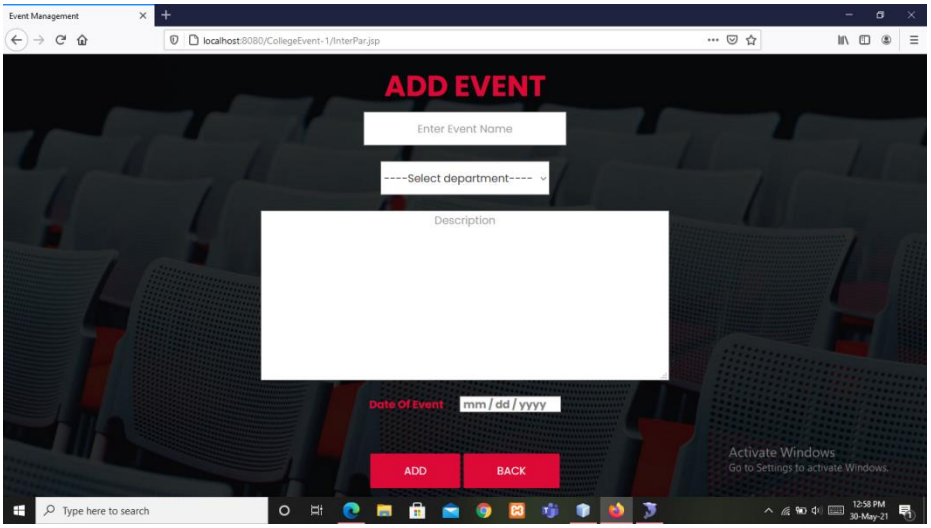

Fig 10: Adding event form

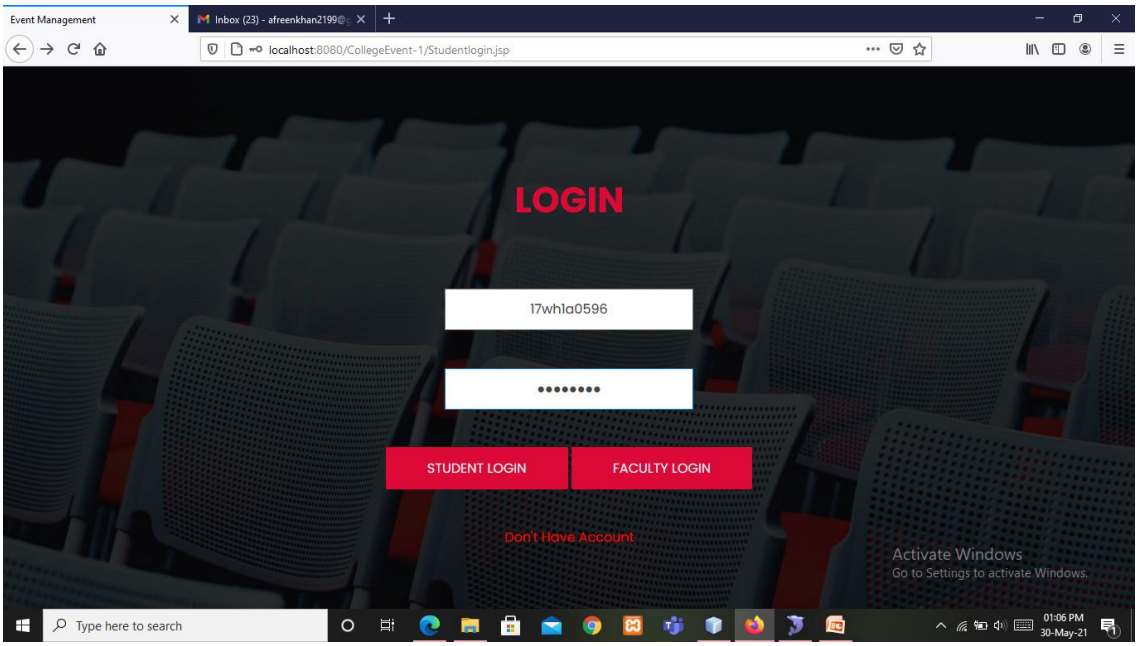

Fig 11: Login form

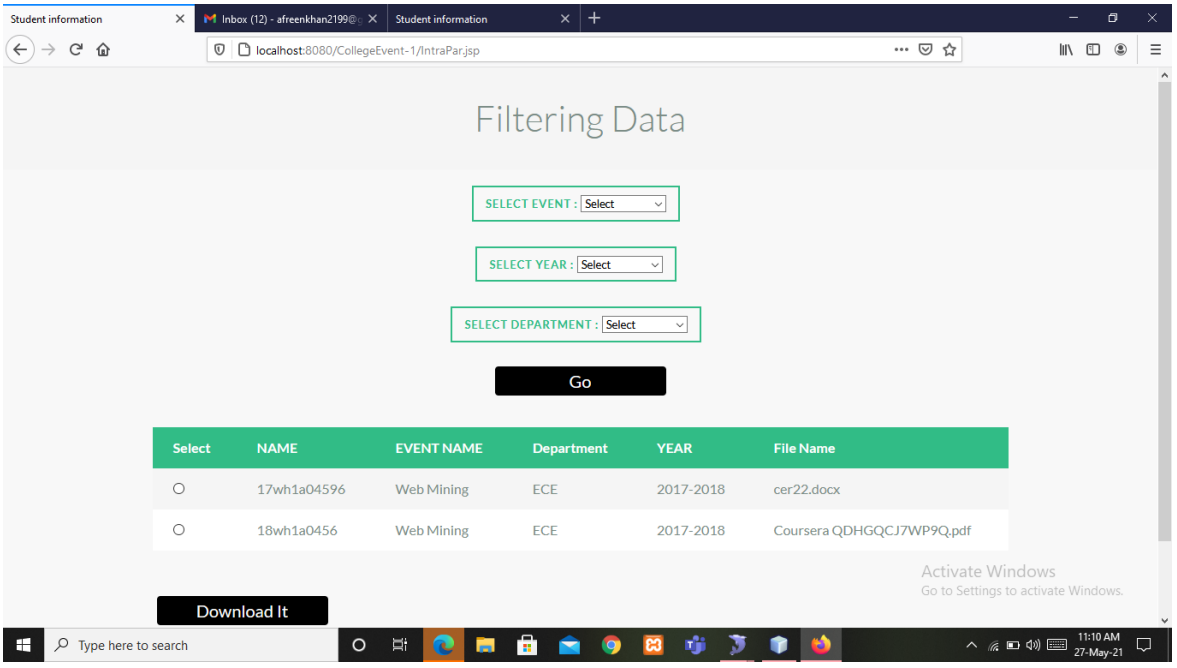

Fig 12: Data Filtering Form

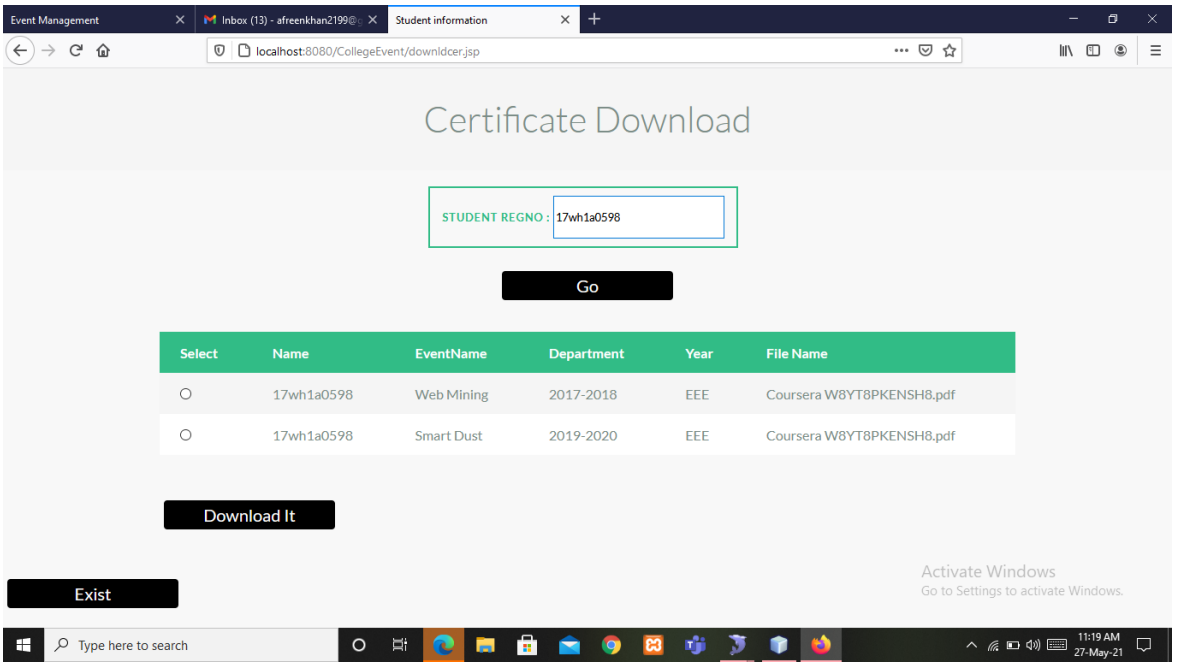

Fig 13: Certificate Download form

Fig 14: Event details

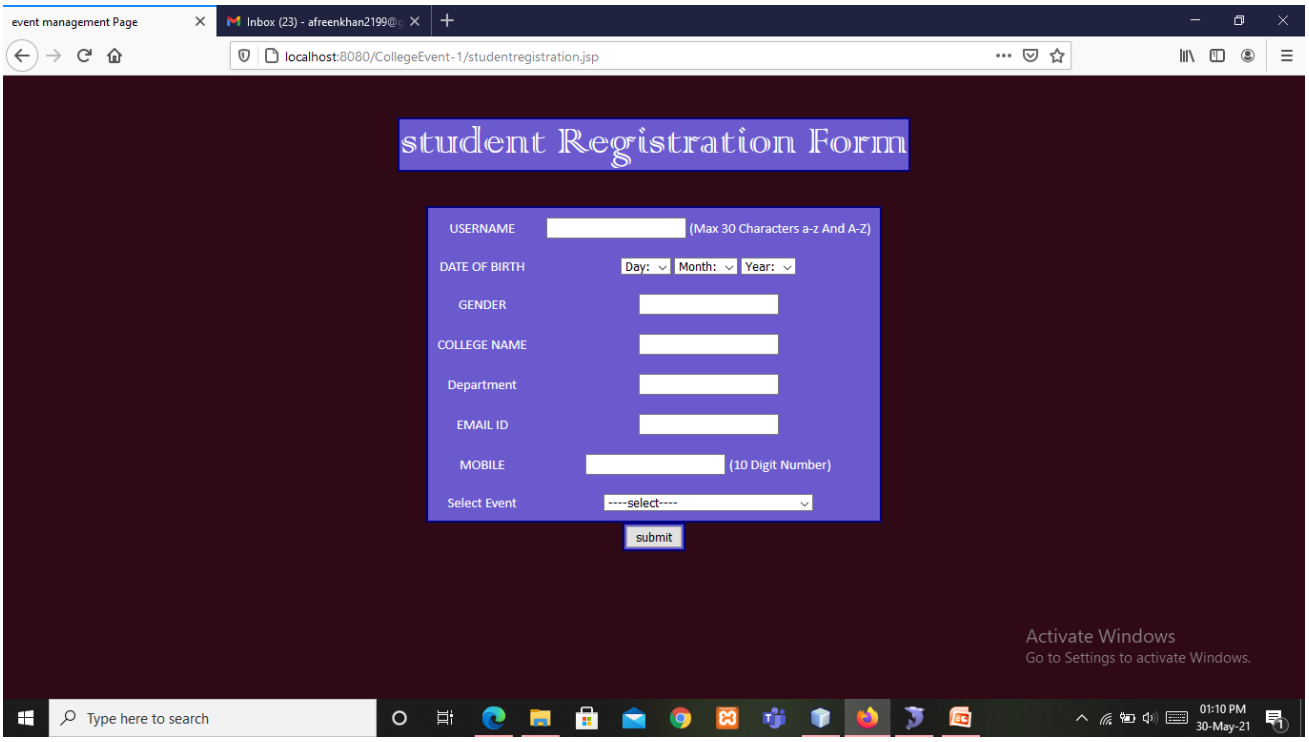

Fig 15: Student Registration Form

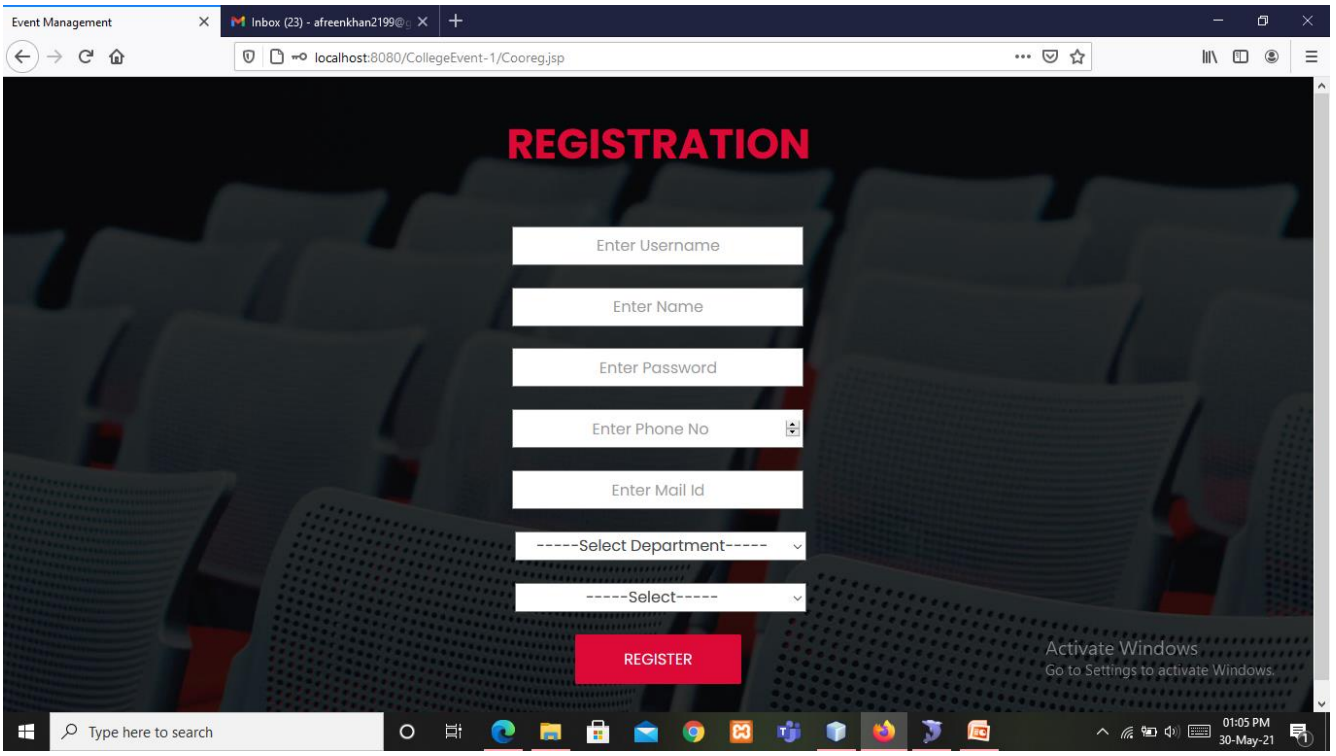

Fig 16: Event Registration form

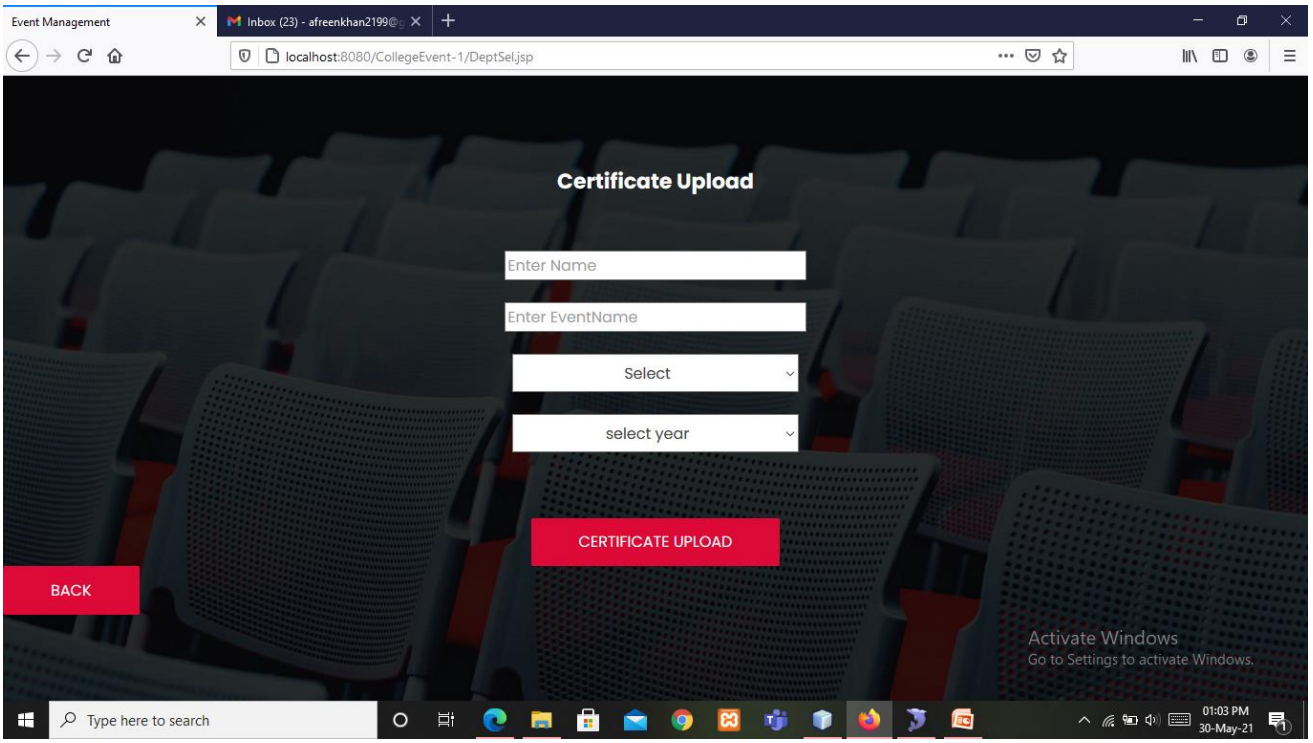

Fig 17: Certificate Upload form

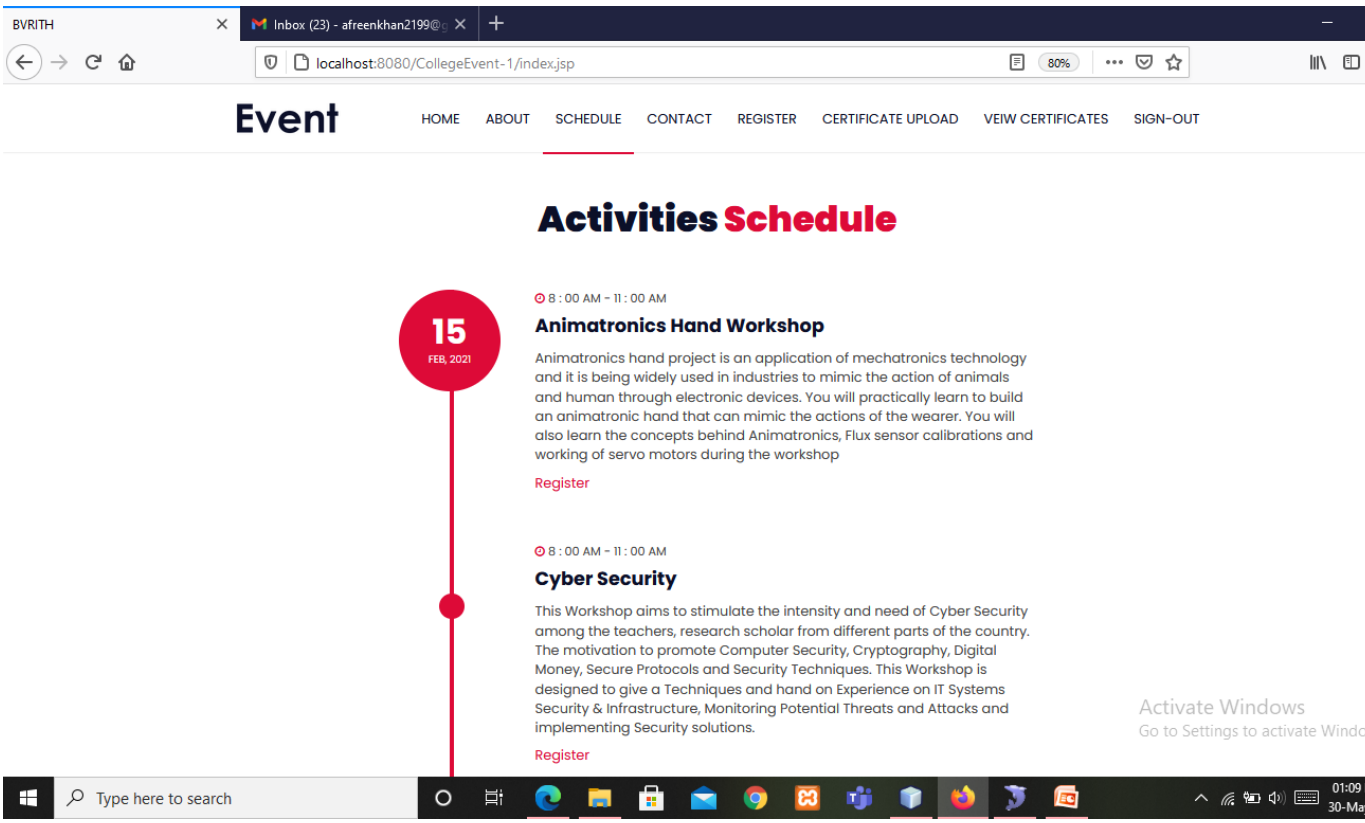

Fig 18: Details of Scheduled Activities

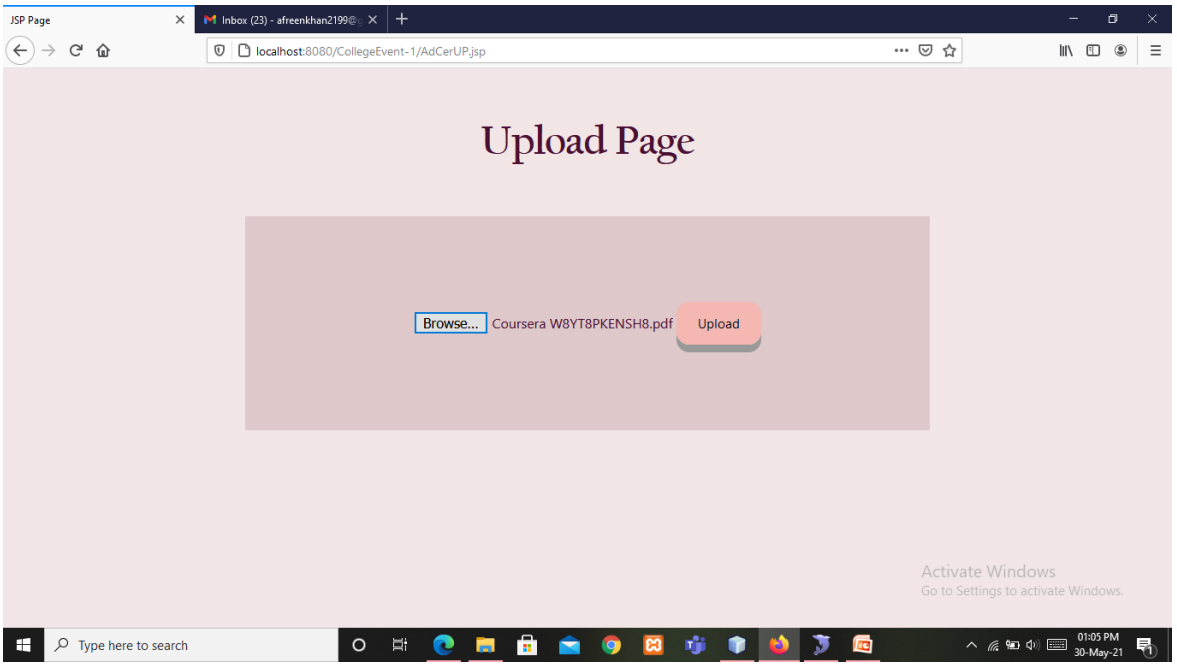

Fig 19: Upload Page

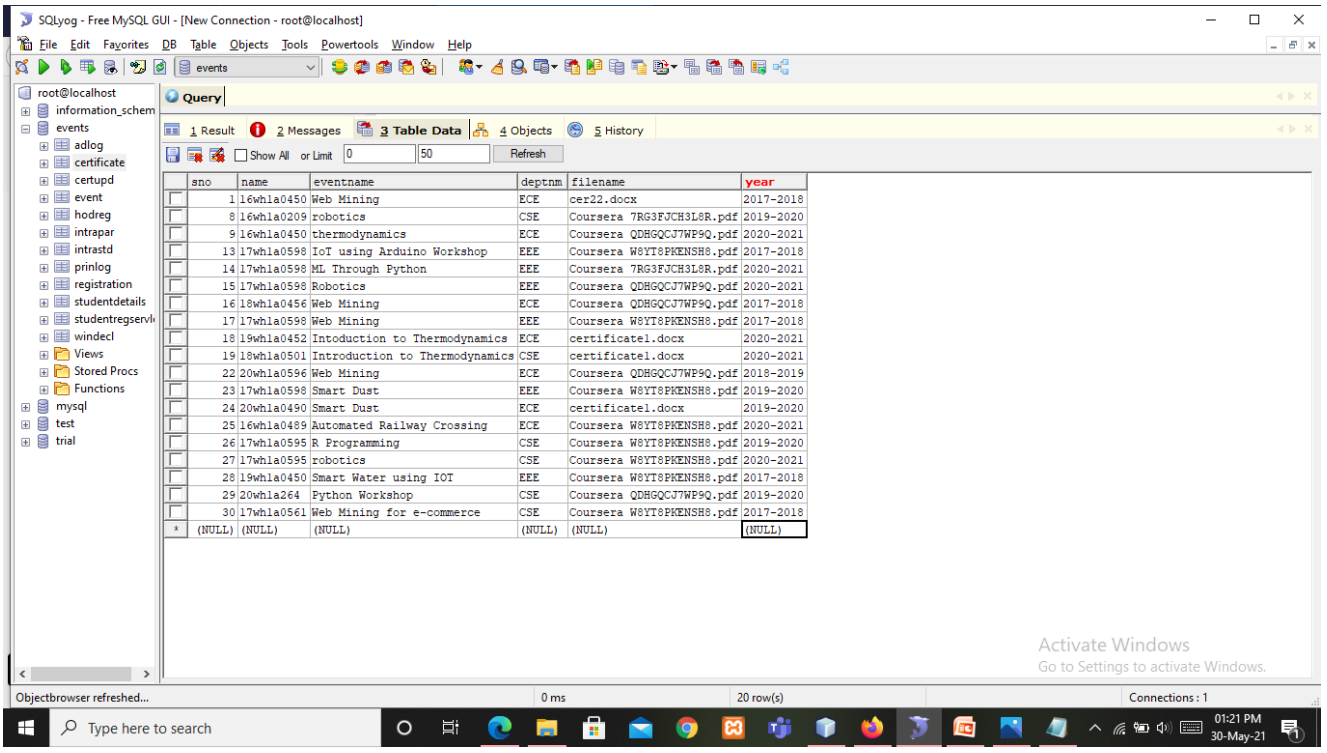

Fig 20: Database of the Students Participated

11. CONCLUSION

The overall speedup is considerably higher than the manual work. The management system enhances the functionalities of the routine works of the management in a number of ways. The computerization helps the users a lot to minimize the working time with ease. The management staffs get information in desired manner. Data retrieval is also easy and fast. The data maintaining has been made quite simple such as searching of records and records maintenance. It's portable to preview the status of participation of students and to track their performance.

Students now can easily register for the activity and get their certifications. They can even provide their participation proof from other colleges and get notified for their excellence.

12. FUTURE SCOPE

This portal is used by the admin so there is no data leakage and it can handle securely. Delivering such software to the management it helps to take place task with ease and that's why it reduces time, money on manpower and efforts. We have event module through which students to get the notices of upcoming events. It is an open source application so that others can edit and transform this system application according to their needs can be a future enhancement in project.

13. REFERENCES

[1] Reddy, G. S., Srinivasu, R., Rikkula, S. R., & Rao, V. S. (2009). Management information system to help managers for providing decision making in an organization. International Journal of Reviews in Computing, 1-6.

[2] Sivarajah, K., & Achchuthan, S. (2013). Entrepreneurial Intention among Undergraduates: Review of Literature. European Journal of Business and Management, 5(5), 172-186.

[3] Lucey, T. (2005). Management information systems. Cengage Learning EMEA.

[4] M. Ashok Kumar; College Activity Management System Second International Conference on Intelligent Computing and Control Systems (ICICCS) 2018.

[5]Omkar Tiware, Prof. Kirti Rajadnya, Siddhesh Shinde College Activity Management International Research Journal of Engineering and Technology (IRJET) 2018.