**A PROJECT REPORT ON**

XXXXX **Feedback System** XXXXX

**Submitted to XXXXXX University for the partial fulfillment of the   
requirement for the**

**Award of Degree for**

XXXXXXXXXXX **Course Name** XXXXXXXXXXXXX

**Done By**

# Mr. / Miss. XXXXXX

## XXXXX Institute of Management & Computer Sciences

**Hyderabad**

**CERTIFICATE**

This is to certify that Mr. / Miss XXXXXXXXXX, bearing Roll No. XXXXXXXXXXX have developed Software Project Titled XXXXXXXXXXXXXX for XXXXXXXXX **SOFTWARE SOLUTIONS** as a partial Fulfillment for the award of the Degree of XXXXXXXXXXXXXX.

**HEAD OF DEPARTMENT** **PRINCIPAL**

**XXX institute of Management**

**&**

**Computer Sciences**

**EXTERNAL**

#### ACKNOWLEDGMENT

My express thanks and gratitude and thanks to Almighty God, my parents and other family members and friends without whose uncontained support, I could not have made this career in XXXXXXXXXX.

I wish to place on my record my deep sense of gratitude to my project guide, **Mr. / Miss XXXXXXXXXX, XXXXX Software Solutions, Hyderabad** for his/her constant motivation and valuable help through the project work. Express my gratitude to **Mr. / Miss XXXXXXXXX**, Director of XXXXXXXXX **Institute of Management & Computer Sciences** for his/her valuable suggestions and advices throughout the XXXXXXXX course. I also extend my thanks to other Faculties for their Cooperation during my Course.

Finally I would like to thank my friends for their cooperation to complete this project.

XXXXXXXXXXXX

(XXXXXX)

**ABSTRACT**

Feedback System is a web based application. It will mainly deal with the Feedback information in

colleges. By using this application the students will sends feedbacks to administrator regarding their

issues.

**CONTENTS**

1. **INTRODUCTION**
   1. INTRODUCTION TO PROJECT
   2. ORGANIZATION PROFILE
   3. PURPOSE OF THE PROJECT
2. **SYSTEM ANALYSIS**
   1. INTRODUCTION
   2. ANALYSIS MODEL
   3. STUDY OF THE SYSTEM
   4. HARDWARE & SOFTWARE REQUIRMENT
   5. PROBLEMS IN EXISTING STSTEM
   6. SOLUTION OF THESE PROBLEMS IN PROPOSED SYSTEM
   7. INPUTS & OUTPUTS
3. **FEASIBILITY REPORT**
   1. TECHNICAL FEASIBILITY
   2. OPERATIONAL FEASIBILITY
   3. ECONOMIC FEASIBILITY
4. **SOFTWARE REQUIREMENT SPECIFICATIONS**
   1. FUNCTIONAL REQUIREMENTS
   2. NON-FUNCTIONAL REQUIREMENTS
   3. PERFORMANCE REQUIREMENTS
5. **SYSTEM DOVELOPEMENT ENVIRONMENT**
   1. INTRODUCTION TO JAVA
   2. SERVLETS,JSP
   3. JDBC
   4. HTML,JAVASCRIPT
   5. ORACLE 10g
6. **SYSTEM DESIGN**
   1. INTRODUCTION
   2. NORMALIZATION
   3. DATA DICTIONARY
   4. E-R DIAGRAM
   5. DATA FLOW DIAGRAM
   6. ACTIVITY DIAGRAM
   7. USE CASE DIAGRAM
   8. SEQUENCE DIAGRAM
   9. CLASS DIAGRAM
   10. COLLABARATION DIAGRAM
7. **OUTPUT SCREENS**
8. **SYSTEM TESTING AND IMPLEMENTATION** 
   1. INTRODUCTION
   2. STRATEGIC APPROACH OF SOFTWARE TESTING
   3. UNIT TESTING
9. **SYSTEM SECURITY**

9.1. INTRODUCTION

* 1. SECURITY IN SOFTWARE

1. **CONCLUSION**
2. **BIBLIOGRAPHY**

**Chapter-1**

**INTRODUCTION**

**1.1. INTRODUCTION TO PROJECT**

FEEDBACK SYSTEM is a java web based application. It will mainly deal with the Feedback information in colleges. By using this application the students will sends feedbacks to administrator regarding their issues.

The main aim of "FEEDBACK SYSTEM" is to improve the services of students and parents. It maintains the details attendance of students and also updating for the same. It also stores the details of various users of the colleges, personal information and exam details of the student.

The system security is taken care of by a login form, which is allows only authorized users to utilize the system.

* 1. **ORGANIZATION PROFILE**

###### Data point Info Solutions (P) Ltd logo

S i m p l i f y i n g S o l u t I o n s & O p p o r t u n i t i e s

# Business Proposition

**Data point** is incepted by young and ambitious team of Professional in the Industry with the Idea & motto of ***“Simplifying Solutions & opportunities”***. Data point is into IT Training (Corporate/Individual), Project assistance, Software Development and Placements. Data point is one among the very few companies in Hyderabad, which are spread across all the areas and technologies.

**Data point has been actively in the profession of sourcing IT professionals from the year 2001.** We have since placed scores of candidates from different skill sets, with varying levels of experience.

Data point started its journey initially as a Consulting Company and as a successful Placement Consultants as per the clients requirements we also emerged as a Corporate Training. Of-late we found that many engineering graduates are not being able to find jobs for themselves, despite increasing demand for IT professionals & Even our clients couldn’t able to find the suitable and potential candidates even in the fresher’s.

At this crucial point we found the gap which needed to be filled by Data point to improve our client satisfaction levels. The very decision of ***“Training (IT & Non-IT aspects) & providing Project assistance”*** to the fresher’s made Data point as a significant player in the market. Data point is assisting many colleges and Organizations in Training & Recruiting fresher.

At Data point, unlike other training institutes we know the Industry requisites and what an Organization expects from a candidate and henceforth we train our students accordingly so that they can get in to the market with more confidence. Data point as we already mentioned not only trains extensively on technologies but also on soft skills. **Data point also motivates the students to implement the projects on their own, which gives them a real time exposure towards the same.**

**Data point endeavors to be a pioneer in Recruiting and manpower consulting** thanks to strategic alliances with leading multinational companies in India and US of America. **Our technically competent, experienced, and certified consultants will help our clientele to get the right manpower at the right time.** We take pride in having top-notch companies who make enable us to have faith in the future through maintaining high quality in screening, hiring and management.

Data point has identified a number of areas of thrust in the emerging and ever growing IT industry and virtue of which, we would focus all our energies to get on to the fast track in the shortest possible period. We pursue requirements from leading Corporate in India and abroad.



## Mission

“Our mission is to identify, recruit and facilitate quality manpower who are technically strong, dynamic and determined, as we are, for the future belongs to those who think and prove global.”

# Why choose Data point for Academic Projects?

Our Project training is based on industry expectations and we will allow the students to do project in real time environment under the guidance of industrial experts. We afford quality training to student which is evinced by the fact that several colleges recommended their students for our extensive project training

# Why choose Data point for Placements?

We guarantee **reliable and productive candidates**, which is evinced by the fact that over **94% of our clientele** have done repeat businesses with us. We provide candidates at competitive bill rates, guaranteed not to change during the life of your assignment.

Once your positions are filled, they are there to stay.

**“Simplifying Opportunities”:** It’s more than just our slogan. Our entire business is geared toward helping our clients to successfully complete and implement their critical I.T initiatives in a timely and cost efficient manner. Our job is to provide you with “Excellent manpower which makes the organization” so that you can be more successful and continue to be the leader in the industry.

### SERVICES

The Company has identified the following as thrust areas:

* IT Solutions
* Training (On and Off Campus)
* Academic Project Assistance (All branches)
* Conducting Events (For Our Clients)
* Placement Services
* Outsourcing of Manpower

**For More Details please reach us:**

**Suresh Maguluri – Managing Director,**

**Data point Info Solutions (P) Ltd,**

**Call: 040 – 64618648 / 64518649 / 98497 53709.**



**Why**?????

* Established in 2001.
* An ISO 9001:2000 Certified Company.
* 8 Years of Expertise in Recruitments & Staffing.
* Having Own Software Development Division.
* Had a Clientele of about 50 MNCs.
* A unique company which provides opportunities to fresher’s.
* The Company, which helps you to meet the Industry expectations.
* Excellent track record in placing the Candidates of various levels.
* A company with the Coding standards of CMM level companies’.

**Chapter 2**

**SYSTEM ANALYSIS**

**2.1. INTRODUCTION**

After analyzing the requirements of the task to be performed, the next step is to analyze the problem and understand its context. The first activity in the phase is studying the existing system and other is to understand the requirements and domain of the new system. Both the activities are equally important, but the first activity serves as a basis of giving the functional specifications and then successful design of the proposed system. Understanding the properties and requirements of a new system is more difficult and requires creative thinking and understanding of existing running system is also difficult, improper understanding of present system can lead diversion from solution.

**2.2. ANALYSIS MODEL**

The model that is basically being followed is the SPIRAL MODEL, which states that the phases are organized in a linear order. First of all the feasibility study is done. Once that part is over the requirement analysis and project planning begins. If system exists one and modification and addition of new module is needed, analysis of present system can be used as basic model.

The design starts after the requirement analysis is complete and the coding begins after the design is complete. Once the programming is completed, the testing is done. In this model the sequence of activities performed in a software development project are: -

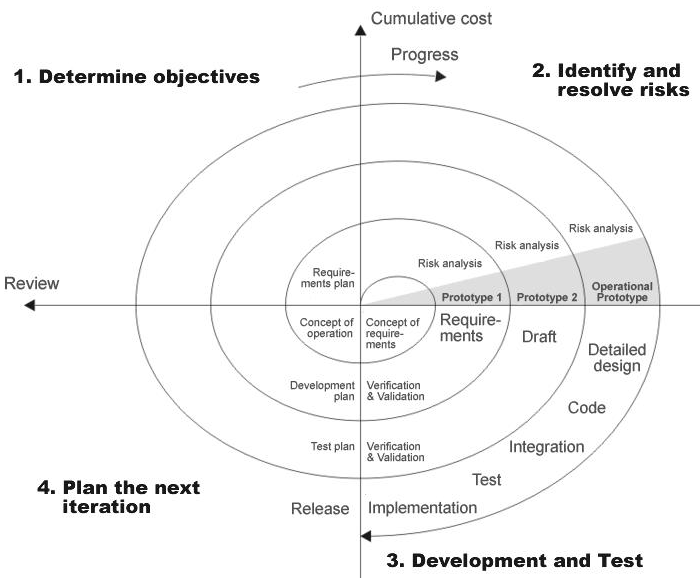
* Requirement Analysis
* Project Planning
* System design
* Detail design
* Coding
* Unit testing
* System integration & testing

Here the linear ordering of these activities is critical. End of the phase and the output of one phase is the input of other phase. The output of each phase is to be consistent with the overall requirement of the system.

SPIRAL MODEL was defined by Barry Boehm in his 1988 article, “A spiral Model of Software Development and Enhancement. This model was not the first model to discuss iterative development, but it was the first model to explain why the iteration models.

As originally envisioned, the iterations were typically 6 months to 2 years long. Each phase starts with a design goal and ends with a client reviewing the progress thus far. Analysis and engineering efforts are applied at each phase of the project, with an eye toward the end goal of the project.

**The following diagram shows how a spiral model acts like:**



**Fig 1.0: Spiral Model**

**2.3. STUDY OF THE SYSTEM**

To provide flexibility to the users, the interfaces have been developed that are accessible through a browser. The GUI’S at the top level have been categorized as

**Modules:**

After looking at the functional requirements it is identified that we can divide the system into mainly following modules:

* Admin Module
* Student Module
* Lecturer Module

**Administrators:**

In this modulehelps administrators get the most accurate information to make more effective decisions. Employees and administrators gain time saving administrative tools. Itequipped features makes it possible to create faculty, students, view the students and view the faculties.

**Add department**: admin can add department details.

**Add Faculty:** admin can add a faculty

**Add Student:** admin can add a student

**Feedback:** admin can view the feedback of all faculty members

**logout**

**Student Module:-**

This module maintains the information related to the Feedbacks and Self details.

**View profile:** student can login and view his profile

**Feedback:** student can send feed back

**Faculty Module:**

This module maintains the information related to the search feedbacks by year-wise, branch-wise, and semester-wise.

**View profile:** faculty can login and view his profile

**Feedback:** faculty feed back

**2.4. HARDWARE & SOFTWARE SPECIFICATIONS**

**Hardware Requirements:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Client Side** | | | |
|  | **Processor** | **RAM** | **Disk Space** |
| **Internet Explorer 6.0**  **Or Higher** | Computer with a 2.6GHz processor or higher (Pentium processor recommended) | 512MB Minimum | Minimum 20 GB |
| **Server Side** | | | |
| **Apache Tomcat** | Intel Pentium processor at 2.6GHz or faster | Minimum 512 MB Physical Memory; 1 GB Recommended | Minimum 20 GB |
| **Net Beans 6.8**  **Oracle10g** | Intel Pentium Processor at  2.6GHz or faster | Minimum 512 MB Physical Memory; (1 GB Recommended) | Minimum 20 GB |

**Software Interface:**

* **Client on Internet/Intranet:** Any web browser.
* **Web Server:** Apache Tomcat.
* **Database Server:** Oracle10g.
* **Development End:** NetBeans 6.8.

**2.5. EXISTING SYSTEM**

In the existing system the faculties have to take care of the students and collect all the feedback details and verification has to be taken manually.The existing system is manual and the manual system works in the following way:

**Limitations of the Existing system:**

* The faculty of particular department can assign because of that its take more time consuming to the faculty as well as student.
* If any personal details related to the student/staff have been tainted then the details has to be submitted to the college.
* If any personal details related to the student have been changed then the details has to be submitted to the College. This process is to be taken manually.
* Within a College the interactions between Principal (Admin), Faculties and the Students have been done manually.

Till now most of the schools has adopted the above manual system that produces lot of problems and at the same time the following disadvantages are there with the above system.

* The increasing complexity,
* producing the reports as desired is not possible,
* it is Manual system so it is time consuming

**2.6. PROPOSED SYSTEM**

To overcome all the difficulties of the existing system the management has proposed automated the whole system and the development of the new automated system contains the following activities, which try to automate the entire process keeping in view of the database integration approach.

**Advantages of Proposed System:**

* By developing this system we can provide better security to our data And reports will be gained accurately.
* Administrator is able to view create the faculty, students and he is also able to view the students and the faculty members also.
* It provides complete activity as automated system.
* It is not limited to a single system because it is aimed to develop for web based environment.
* The student information is available to the administrator and also to the faculty.

**2.7. INPUTS AND OUTPUTS**

The major inputs and outputs and major functions of the system are follows:

**Inputs:**

* Admin enter his user id and password for login.
* student enters his user id and password for login.
* New student/faculty gives his completed personnel, address and phone details to admin for registration.
* Administrator can view entire feedback.
* Admin enters department details.

**Outputs:**

* Admin can have his own home page.
* Student/faculty enter their own home page.
* The new user’s (student) data will be stored in the centralized database.
* Admin get feedback of all faculty .

**Chapter 3**

**Feasibility Report**

Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

* Technical Feasibility
* Operation Feasibility
* Economical Feasibility

## 3.1. Technical Feasibility

The technical issue usually raised during the feasibility stage of the investigation includes the following:

* Does the necessary technology exist to do what is suggested?
* Do the proposed equipments have the technical capacity to hold the data required to use the new system?
* Will the proposed system provide adequate response to inquiries, regardless of the number or location of users?
* Can the system be upgraded if developed?
* Are there technical guarantees of accuracy, reliability, ease of access and data security?

Earlier no system existed to cater to the needs of ‘Secure Infrastructure Implementation System’. The current system developed is technically feasible. It is a web based user interface for audit workflow at ABC Tech. Thus it provides an easy access to the users. The database’s purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security. The software and hard requirements for the development of this project are not many and are already available in-house at ABC Tech or are available as free as open source. The work for the project is done with the current equipment and existing software technology. Necessary bandwidth exists for providing a fast feedback to the users irrespective of the number of users using the system.

**3.2. Operational Feasibility**

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization’s operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. Some of the important issues raised are to test the operational feasibility of a project includes the following: -

* Is there sufficient support for the management from the users?
* Will the system be used and work properly if it is being developed and implemented?
* Will there be any resistance from the user that will undermine the possible application benefits?

This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits.

The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

**3.3. Economic Feasibility**

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs.

The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at ABC Tech, There is nominal expenditure and economical feasibility for certain.

**Chapter 4**

**SOFTWARE REQUIREMENT SPECIFICATION**

## 4.1. FUNCTIONAL REQUIREMENTS:

The services that this system should support for these users are summarized below:

This system should provide the administrator with the convenience such as adding a new faculty, view and manage the information about the faculty; students view the next information:

Admin maintain all the information about the project.

* Admin can add the students’ details and staff details.
* Admin can add department, view the staff details, and student details.
* Admin view the staff feedbacks and search the feedbacks by branch-wise, year-wise, and semester-wise.

This system should help the Faculty also.

* Faculty can login and he gets user id and password.
* Faculty can view the self details and search the feedback details by year-wise, semester-wise, and branch-wise.

This system should help the student’s also.

* Student can login and he gets the student id and password
* Student can view the self details and send to the feedbacks to administrator.

## 4.2. Non-Functional Requirements:

The system should be web-based system. Each user should have a user account. The system should ask the username and password to users. It doesn’t permit to unregistered user to access for Integrated Claim Settlement Services. The system should have Role based System functions access. Approval Process has to be defined. The system should have Modular customization components so that they can be reused across the implementation

# These are the mainly following:

* 24 X 7 availability
* Better component design to get better performance at peak time
* Flexible service based architecture will be highly desirable for future extension

**4.2.1. Performance**

They understand the importance of timing, of getting there before the competition. A rich portfolio of reusable, modular frameworks helps jump-start projects. Tried and tested methodology ensures that we follow a predictable, low - risk path to achieve results. Our track record is testimony to complex projects delivered within and evens before schedule.

**4.2.2. Security**

Its provides more security by setting username and password.

**4.2.3. Safety**

This application provides more safety to the users for accessing the databases and for performing the operations on the databases.

**4.2.4. Interfaces**

It provides the interface for accessing the database and also allows the user to do the manipulations on the databases.

**4.2.5. Reliability**

This entire project is depends on the Oracle.

**4.2.6. Accuracy**

Since the same table is created at different users account, the

Possibility of retrieving data wrongly increases. Also if the data is more,

Validations become difficult. This may result in loss of accuracy of data.

**4.2.7. Ease of Use**

Ever user should be comfortable of working with computer and internet browsing. He must have basic knowledge of English.

**4.2.8. Interoperability**

This provides the import and export facilities for sending one database to another database.

**4.2.9. Maintainability**

The key to reducing need for maintenance, while working, if possible to do essential tasks.

1. More accurately defining user requirement during system development.
2. Assembling better systems documentation.
3. Using more effective methods for designing, processing, and login and communicating information with project team members.
4. Making better use of existing tools and techniques.
5. Managing system engineering process effectively.

**4.2.10. Testability**

Testing is done in various ways such as testing the algorithm, programming code; sample data debugging is also one of following the above testing.

**4.2.11. Design Constraints**

During system testing the system is used experimentally used to ensure that the software does not fail, i.e., it will run according to its specification and in the way the users expect. Special test data are input for processing and the results examined. A limited number of users may be allowed to use the system to see whether they try to use it in unforeseen ways. It is preferable to discover any surprises before the organization implements the system.

**4.2.12. Cost Estimates**

**4.2.13. Preliminary Estimates**.

The project is decomposed into major structural systems or production equipment items, e.g. the entire floor of a building or a cooling system for a processing plant.

**4.2.14. Detailed Estimates**.

The project is decomposed into components of various major systems, i.e., a single floor panel for a building or a heat exchanger for a cooling system.

**4.2.15. Engineer's Estimates**.

The project is decomposed into detailed items of various components as warranted by the available cost data. Examples of detailed items are slabs and beams in a floor panel, or the piping and connections for a heat exchanger.

**Chapter 5**

**SYSTEM DEVELOPEMENT ENVIRONMENT**

# INTRODUCTION TO JAVA

**About Java**:

Initially the language was called as “oak” but it was renamed as “java” in 1995.The primary motivation of this language was the need for a platform-independent (i.e. architecture neutral)language that could be used to create software to be embedded in various consumer electronic devices.

* Java is a programmer’s language
* Java is cohesive and consistent
* Except for those constraint imposed by the Internet environment. Java gives the programmer, full control

Finally Java is to Internet Programming where c was to System Programming.

**Importance of Java to the Internet**

Java has had a profound effect on the Internet. This is because; java expands the Universe of objects that can move about freely in Cyberspace. In a network, two categories of objects are transmitted between the server and the personal computer. They are passive information and Dynamic active programs. in the areas of Security and probability. But Java addresses these concerns and by doing so, has opened the door to an exciting new form of program called the Applet.

**Applications and applets***.* An application is a program that runs on our Computer under the operating system of that computer. It is more or less like one creating using C or C++ .Java’s ability to create Applets makes it important. An Applet I san application, designed to be transmitted over the Internet and executed by a Java-compatible web browser. An applet I actually a tiny Java program, dynamically downloaded across the network, just like an image. But the difference is, it is an intelligent program, not just a media file. It can be react to the user input and dynamically change.

**Java Architecture**

Java architecture provides a portable, robust, high performing environment for development. Java provides portability by compiling the byte codes for the Java Virtual Machine, which is then interpreted on each platform by the run-time environment. Java is a dynamic system, able to load code when needed from a machine in the same room or across the planet.

# Compilation of code

## When you compile the code, the Java compiler creates machine code (called byte code)for a hypothetical machine called Java Virtual Machine(JVM). The JVM is supposed t executed the byte code. The JVM is created for the overcoming the issue of probability. The code is written and compiled for one machine and interpreted on all machines .This machine is called Java Virtual Machine.

Compiling and interpreting java source code.

**Source code**

**Pc compiler**

**Macintosh compiler**

**SPARC Compiler**

**Java Byte code**

**Platform independent**

**Java interpreter**

**Java interpretermacintosh**

**)))**

**Java interpreter(SPARC)**

During run-time the Java interpreter tricks the byte code file into thinking that it is running on a Java Virtual Machine. In reality this could be an Intel Pentium windows 95 or sun SPARCstation running Solaris or Apple Macintosh running system and all could receive code from any computer through internet and run the Applets.

**Simple**:

Java was designed to be easy for the Professional programmer to learn and to use effectively. If you are an experienced C++ Programmer. Learning Java will oriented features of C++ . Most of the confusing concepts from C++ are either left out of Java or implemented in a cleaner, more approachable manner. In Java there are a small number of clearly defined ways to accomplish a given task.

### Object oriented

Java was not designed to be source-code compatible with any other language. This allowed the Java team the freedom to design with a blank state. One outcome of this was a clean usable, pragmatic approach to objects. The object model in Java is simple and easy to extend, while simple types, such as integers, are kept as high-performance non-objects.

### Robust

The multi-platform environment of the web places extraordinary demands on a program, because the program must execute reliably in a variety of systems. The ability to create robust programs. Was given a high priority in the design of Java. Java is strictly typed language; it checks your code at compile time and runtime.

Java virtually eliminates the problems of memory management and deal location, which is completely automatic. In a well-written Java program, all run-time errors can and should be managed by your program.

**5.2. Servlets/JSP**

A Servlet Is a generic server extension. a Java class that can be loaded

Dynamically to expand the functionality of a server.Servlets are commonly used with web servers. Where they can take the place CGI scripts.

A servlet is similar to proprietary server extension, except that it runs inside a Java Virtual Machine (JVM) on the server, so it is safe and portable

Servlets operate solely within the domain of the server.

Unlike CGI and Fast CGI, which use multiple processes to handle separate program or separate requests, separate threads within web server process handle all servlets. This means that servlets are all efficient and scalable.

Servlets are portable; both across operating systems and also across web servers. Java Servlets offer the best possible platform for web application development.

Servlets are used as replacement for CGI scripts on a web server, they can extend any sort of server such as a mail server that allows servelts t extend its functionality perhaps by performing a virus scan on all attached documents or handling mail filtering tasks.

Servlets provide a Java-based solution used to address the problems currently associated with doing server-side programming including inextensible scripting solutions platform-specific API’s and incomplete interface.

Servlets are objects that conform to a specific interface that can be plugged into a Java-based server.Servlets are to the server-side what applets are to the server-side what applets are to the client-side-object byte codes that can be dynamically loaded off the net. They differ form applets in than they are faceless objects(with out graphics or a GUI component).They serve as platform independent, dynamically loadable,plugable helper byte code objects on the server side that can be used to dynamically extend server-side functionality.

For example an HTTP servlet can be used to generate dynamic HTML content when you use servlets to do dynamic content you get the following advantages:

* They’re faster and cleaner then CGI scripts
* They use a standard API( the servlet API)
* They provide all the advantages of Java (run on a variety of servers without needing to be rewritten)

**Attractiveness of servlets:**

They are many features of servlets that make them easy and attractive to tuse these include:

* Easily configure using the GUI-based Admin tool]
* Can be Loaded and Invoked from a local disk or remotely across the network.
* Can be linked together or chained, so that on servlet can call another servlet, or several servlets in sequence.
* Can be called dynamically from with in HTML, pages using server-side include-tags.
* Are secure-even when downloading across the network, the servlet security model and servlet and box protect your system from unfriendly behavior.,

**Advantages of the servlet API**

One of the great advantages of the servlet API is protocol independent. It assumes nothing about:

* The protocol being used to transmit on the net
* How it is loaded
* The server environment it will be running in
* These quantities are important, because it allows the Servlet API to be embedded in many different kinds of servers.There are other advantages to the servelt API as well These include:
* It’s extensible-you can inherit all your functionality from the base classes made available to you
* It’s simple small, and easy to use.

**Features of Servlets:**

* Servlets are persistent.Servlet are loaded only by the web server and can maintain services between requests.
* Servlets are fast. Since servlets only need to be l\loaded once, they offer much better performance over their CGI counterparts.
* Servlets are platform independent.
* Servlets are extensible Java is a robust, object-oriented programming language, which easily can be extended to suit your needs.
* Servlets are secure
* Servlets are used with a variety of client.

Servlets are classes and interfaces from tow packages,javax .servlet and javax.servlet.http.The java.servlet package contains classes t support generic, protocol-independent servlets.The classes in the javax.servelt.http package To and HTTP specific functionality extend these classes

Every servlet must implement the javax.servelt interface.Most servlets implement it by extending one of two classes.javax.servlet.GenericServlet or javax.servlet.http.HttpServlet.A protocol-independent servlet should subclass Generic-Servlet.while an Http servlet should subclass HttpServlet, which is itself a subclass of Generic-servlet with added HTTP-specific functionality.

Unlike a java program, a servlet does not have a main() method,Instead the server in the process of handling requests invoke certain methods of a servlet.Each time the server dispatches a request to a servlet, it invokes the servelts Service() method,

A generic servlet should override its service() method to handle requests as appropriate for the servlet.The service() accepts two parameters a request object and a response object .The request object tells the servlet about the request, while the response object is used to return a response

InContrast.anHttp servlet usually does not override the service() method.Instead it overrides doGet() to handle GET requests and doPost() to handle Post requests. An Http servlet can override either or both of these modules the service() method of HttpServlet handles the setup and dispatching to all the doXXX() methods.which iswhy it usually should not be overridden

The remainders in the javax.servlet and javax.servlet.http.package are largely support classes .The ServletRequest and ServletResponse classes in javax.servlet provide access to generic server requests and responses while HttpServletRequest and HttpServletResponse classes in javax.servlet provide access to generic server requests and responses while HttpServletRequest and HttpServletResponse in javax.servlet.http provide access a HTTP requests and responses . The javax.servlet.http provide contains an HttpSession class that provides built-in session tracking functionality and Cookie class that allows quickly setup and processing HttpCookies.

**Loading Servlets:**

Servlets can be loaded from their places. From a directory that is on the CLASSPATH. The CLASSPATH of the JavaWebServer includes service root/classes/, which is where the system classes reside

From the <SERVICE\_ROOT/servlets/directory.This is not in the server’s classpath. A class loader is used to create servlets form this directory.New servlets can be added-existing servlets can be recompiled and the server will notice these changes. From a remote location.For this a code base like <http://nine.eng/classes/foo/> is required in addtion to the servlet’s class name.Refer to the admin Gui docs on servlet section to see how to set this up.

Loading Remote Servlets

Remote servlets can be loaded by:

* Configuring the admin Tool to setup automatic loading of remote servlets.
* Selectiong up server side include tags in .html files
* Defining a filter chain Configuration

**Invoking Servlets**

A servlet invoker is a servlet that invokes the “server” method on a named servlet.If the servlet is not loaded in the server,then the invoker first loades the servlet(either form local disk or from the network) and the then invokes the “service” method.Also like applets,local servlets in the server can be identified by just the class name.In other words, if a servlet name is not absolute.it is treated as local.

A Client can Invoke Servlets in the Following Ways:

* The client can ask for a document that is served by the servlet.
* The client(browser) can invoke the servlet directly using a URL, once it has been mapped using the SERVLET ALIASES Section of the admin GUI
* The servlet can be invoked through server side include tags.
* The servlet can be invoked by placing it in the servlets/directory
* The servlet can be invoked by using it in a filter chain

**5.3. JDBC**

**What is JDBC?**

any relational database. One can write a single program using the JDBC API,and the JDBC is a Java Api for executing SQL,Statements(As a point of interest JDBC is trademarked name and is not an acronym; nevertheless,Jdbc is often thought of as standing for Java Database Connectivity. It consists of a set of classes and interfaces written in the Java Programming language.JDBC provides a standard API for tool/database developers and makes it possible to write database applications using a pure Java API

Using JDBC, it is easy to send SQL statements to virtually program will be able to send SQL .statements to the appropriate database. The Combination of Java and JDBC lets a programmer writes it once and run it anywhere.

***What Does JDBC Do?***

**Simply put,JDBC makes it possible to do three things**

* Establish a connection with a database
* Send SQL statements
* Process the results
* JDBC Driver Types
* The JDBC drivers that we are aware of this time fit into one of four categories
* JDBC-ODBC Bridge plus ODBC driver
* Native-API party-java driver
* JDBC-Net pure java driver
* Native-protocol pure Java driver

An individual database system is accessed via a specific JDBC driver that implements the java.sql.Driver interface. Drivers exist for nearly all-popular RDBMS systems, through few are available for free. Sun bundles a free JDBC-ODBC bridge driver with the JDK to allow access to a standard ODBC,data sources, such as a Microsoft Access database, Sun advises against using the bridge driver for anything other than development and very limited development.

JDBC drivers are available for most database platforms, from a number of vendors and in a number of different flavours. There are four driver categories

**Type 01-JDBC-ODBC Bridge Driver**

Type 01 drivers use a bridge technology to connect a java client to an ODBC database service. Sun’s JDBC-ODBC bridge is the most common type 01 driver. These drivers implemented using native code.

**Type 02-Native-API party-java Driver**

Type 02 drivers wrap a thin layer of java around database-specific native code libraries for Oracle databases, the native code libraries might be based on the OCI(Oracle call Interface) libraries, which were originally designed for **c/c++** programmers, Because type-02 drivers are implemented using native code. in some cases they have better performance than their all-java counter parts. They add an element of risk, however, because a defect in a driver’s native code section can crash the entire server

**Type 03-Net-Protocol All-Java Driver**

Type 03 drivers communicate via a generic network protocol to a piece of custom middleware. The middleware component might use any type of driver to provide the actual database access. These drivers are all java, which makes them useful for applet deployment and safe for servlet deployment

**Type-04-native-protocol All-java Driver**

Type o4 drivers are the most direct of the lot. Written entirely in java, Type 04 drivers understand database-specific networking. protocols and can access the database directly without any additional software

**JDBC-ODBC Bridge**

If possible use a Pure Java JDBC driver instead of the Bridge and an ODBC driver. This completely eliminates the client configuration required by ODBC.It also eliminates the potential that the Java VM could be corrupted by an error in the native code brought in by the Bridge(that is, the Bridge native library, the ODBC driver manager library, library, the ODBC driver library, and the database client library)

**WHAT IS The JDBC-ODBE Bridge ?**

The JDBC-ODBC Bridge is a Jdbc driver, which implements JDBC operations by translating them into ODBC operations. To ODBC it appears as a normal application program. The Bridge is implemented as the sun.jdbc.odbc Java package and contains a native library used to access ODBC.The Bridge is joint development of Intersolv and Java Soft

**5.4. HTML, JAVASCRIPT**

Hypertext Markup Language(HTML), the languages of the world wide web(WWW), allows users to produces web pages that included text, graphics and pointer to other web pages (Hyperlinks).

HTML is not a programming language but it is an application of ISO Standard 8879,SGML(Standard Generalized Markup Language),but

Specialized to hypertext and adapted to the Web. The idea behind Hypertext one point to another point. We can navigate through the information based on out interest and preference. A markup language is simply a series of items enclosed within the elements should be displayed.

Hyperlinks are underlined or emphasized works that load to other documents or some portions of the same document.

Html can be used to display any type of document on the host computer, which can be geographically at a different location. It is a versatile language and can be used on any platform or desktop

HTML provides tags(special codes) to make the document look attractive.

HTML provides are not case-sensitive. Using graphics,fonts,different sizes, color, etc.. can enhance the presentation of the document. Anything

That is not a tag is part of the document it self.

**Basic Html Tags**:

<!-- --> Specific Comments.

<A>………</A> Creates Hypertext links.

<B>………</B> Creates hypertext links.

<Big>……..</Big> Formats text in large-font

<Body>…….</Body> contains all tags and text in the Html-document

<Center>……</Center> Creates Text

<DD>………..</DD> Definition of a term.

<TABLE>……</TABLE> creates table

<Td>………..</Td> indicates table data in a table.

<Tr>………..</Tr> designates a table row

<Th>……….</Th> creates a heading in a table.

##### *A D V A N T A G E S:-*

* + 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.

**5.5. Database**

Oracle is a relational database management system, which organizes data in the form of tables. Oracle is one of many database servers based on RDBMS model, which manages a seer of data that attends three specific things-data structures, data integrity and data manipulation.

With oracle cooperative server technology we can realize the benefits of open, relational systems for all the applications. Oracle makes efficient use of all systems resources, on all hardware architecture; to deliver unmatched performance, price performance and scalability. Any DBMS to be called as RDBMS has to satisfy Dr.E.F.Codd’s rules.

**Features of Oracle:**

**Portable**

The Oracle RDBMS is available on wide range of platforms ranging from PCs to super computers and as a multi user loadable module for Novel NetWare, if you develop application on system you can run the same application on other systems without any modifications.

**Compatible**

Oracle commands can be used for communicating with IBM DB2 mainframe RDBMS that is different from Oracle, which is Oracle compatible with DB2. Oracle RDBMS is a high performance fault tolerant DBMS, which is specially designed for online transaction processing and for handling large database applications.

**Multithreaded Server Architecture**

Oracle adaptable multithreaded server architecture delivers scalable high performance for very large number of users on all hardware architecture including symmetric multiprocessors (sumps) and loosely coupled multiprocessors. Performance is achieved by eliminating CPU, I/O, memory and operating system bottlenecks and by optimizing the Oracle DBMS server code to eliminate all internal bottlenecks.

Oracle has become the most popular RDBMS in the market because of its ease of use

* Client/server architecture.
* Data independence.
* Ensuring data integrity and data security.
* Managing data concurrency.
* Parallel processing support for speed up data entry and online transaction processing used for applications.
* DB procedures, functions and packages.

**Dr.E.F.Codd’s Rules**

These rules are used for valuating a product to be called as relational database management systems. Out of 12 rules, a RDBMS product should satisfy at least 8 rules + rule called rule 0 that must be satisfied.

**RULE 0: Foundation Rule**

For any system to be advertised as, or claimed to be relational DBMS should manage database with in it self, with out using an external language.

**RULE 1: Information Rule**

All information in relational database is represented at logical level in only one way as values in tables.

**RULE 2: Guaranteed Access**

Each and every data in a relational database is guaranteed to be logically accessibility by using to a combination of table name, primary key value and column name.

**RULE 3: Systematic Treatment of Null Values**

Null values are supported for representing missing information and inapplicable information. They must be handled in systematic way, independent of data types.

**RULE 4: Dynamic Online Catalog based Relation Model**

The database description is represented at the logical level in the same way as ordinary data so that authorized users can apply the same relational language to its interrogation as they do to the regular data.

**RULE 5: Comprehensive Data Sub Language**

A relational system may support several languages and various models of terminal use. However there must be one language whose statement can express all of the following:

Data Definitions, View Definitions, Data Manipulations, Integrity, Constraints, Authorization and transaction boundaries.

**RULE 6: View Updating**

Any view that is theoretical can be updatable if changes can be made to the tables that effect the desired changes in the view.

**RULE 7: High level Update, Insert and Delete**

The capability of handling a base relational or derived relational as a single operand applies not only retrieval of data also to its insertion, updating, and deletion.

**RULE 8: Physical Data Independence**

Application program and terminal activities remain logically unimpaired whenever any changes are made in either storage representation or access method.

**RULE 9: Logical Data Independence**

Application programs and terminal activities remain logically unimpaired whenever any changes are made in either storage representation or access methods.

**RULE 10: Integrity Independence**

Integrity constraints specific to particular database must be definable in the relational data stored in the catalog, not in application program.

**RULE 11: Distributed Independence**

Whether or not a system supports database distribution, it must have a data sub-language that can support distributed databases without changing the application program.

**RULE 12: Non Sub-Version**

If a relational system has low level language, that low language cannot use to subversion or by pass the integrity rules and constraints expressed in the higher level relational language.

**Oracle supports the following Codd’s Rules**

Rule 1: Information Rule (Representation of information)-YES.

Rule 2: Guaranteed Access-YES.

Rule 3: Systematic treatment of Null values-YES.

Rule 4: Dynamic on-line catalog-based Relational Model-YES.

Rule 5: Comprehensive data sub language-YES.

Rule 6: View Updating-PARTIAL.

Rule 7: High-level Update, Insert and Delete-YES.

Rule 8: Physical data Independence-PARTIAL.

Rule 9: Logical data Independence-PARTIAL.

Rule 10: Integrity Independence-PARTIAL.

Rule 11: Distributed Independence-YES.

Rule 12: Non-subversion-YES.

**Chapter 6**

**SYSTEM DESIGN**

**6.1. INTRODUCTION**

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer’s goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirement have been specified and analyzed, system design is the first of the three technical activities -design, code and test that is required to build and verify software.

The importance can be stated with a single word “Quality”. Design is the place where quality is fostered in software development. Design provides us with representations of software that can assess for quality. Design is the only way that we can accurately translate a customer’s view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage.

During design, progressive refinement of data structure, program structure, and procedural details are developed reviewed and documented. System design can be viewed from either technical or project management perspective. From the technical point of view, design is comprised of four activities – architectural design, data structure design, interface design and procedural design.

**6.2. NORMALIZATION**

It is a process of converting a relation to a standard form. The process is used to handle the problems that can arise due to data redundancy i.e. repetition of data in the database, maintain data integrity as well as handling problems that can arise due to insertion, updation, deletion anomalies.

Decomposing is the process of splitting relations into multiple relations to eliminate anomalies and maintain anomalies and maintain data integrity. To do this we use normal forms or rules for structuring relation.

**Insertion anomaly**: Inability to add data to the database due to absence of other data.

**Deletion anomaly**: Unintended loss of data due to deletion of other data.

**Update anomaly**: Data inconsistency resulting from data redundancy and partial update

**Normal Forms**: These are the rules for structuring relations that eliminate anomalies.

**FIRST NORMAL FORM**:

A relation is said to be in first normal form if the values in the relation are atomic for every attribute in the relation. By this we mean simply that no attribute value can be a set of values or, as it is sometimes expressed, a repeating group.

**SECOND NORMAL FORM**:

A relation is said to be in second Normal form is it is in first normal form and it should satisfy any one of the following rules.

1. Primary key is a not a composite primary key
2. No non key attributes are present
3. Every non key attribute is fully functionally dependent on full set of primary key.

**THIRD NORMAL FORM**:

A relation is said to be in third normal form if their exits no transitive dependencies.

**Transitive Dependency**: If two non key attributes depend on each other as well as on the primary key then they are said to be transitively dependent.

The above normalization principles were applied to decompose the data in multiple tables thereby making the data to be maintained in a consistent state.

**6.4. E – R DIAGRAMS**

* + The relation upon the system is structure through a conceptual ER-Diagram, which not only specifics the existential entities but also the standard relations through which the system exists and the cardinalities that are necessary for the system state to continue.
  + The Entity Relationship Diagram (ERD) depicts the relationship between the data objects. The ERD is the notation that is used to conduct the date modeling activity the attributes of each data object noted is the ERD can be described resign a data object descriptions.
  + The set of primary components that are identified by the ERD are
  + Data object
  + Relationships
  + Attributes
  + Various types of indicators.

The primary purpose of the ERD is to represent data objects and their relationships.



**6.5. DATA FLOW DIAGRAMS**

A data flow diagram is graphical tool used to describe and analyze movement of data through a system. These are the central tool and the basis from which the other components are developed. The transformation of data from input to output, through processed, may be described logically and independently of physical components associated with the system. These are known as the logical data flow diagrams. The physical data flow diagrams show the actual implements and movement of data between people, departments and workstations. A full description of a system actually consists of a set of data flow diagrams. Using two familiar notations Yourdon, Gane and Sarson notation develops the data flow diagrams. Each component in a DFD is labeled with a descriptive name. Process is further identified with a number that will be used for identification purpose. The development of DFD’S is done in several levels. Each process in lower level diagrams can be broken down into a more detailed DFD in the next level. The lop-level diagram is often called context diagram. It consist a single process bit, which plays vital role in studying the current system. The process in the context level diagram is exploded into other process at the first level DFD.

The idea behind the explosion of a process into more process is that understanding at one level of detail is exploded into greater detail at the next level. This is done until further explosion is necessary and an adequate amount of detail is described for analyst to understand the process.

Larry Constantine first developed the DFD as a way of expressing system requirements in a graphical from, this lead to the modular design.

A DFD is also known as a “bubble Chart” has the purpose of clarifying system requirements and identifying major transformations that will become programs in system design. So it is the starting point of the design to the lowest level of detail. A DFD consists of a series of bubbles joined by data flows in the system.

**DFD SYMBOLS:**

In the DFD, there are four symbols

1. A square defines a source(originator) or destination of system data
2. An arrow identifies data flow. It is the pipeline through which the information flows
3. A circle or a bubble represents a process that transforms incoming data flow into outgoing data flows.
4. An open rectangle is a data store, data at rest or a temporary repository of data

Process that transforms data flow

Source or Destination of data

Data flow

Data Store

**CONSTRUCTING A DFD:**

Several rules of thumb are used in drawing DFD’S:

1. Process should be named and numbered for an easy reference. Each name should be representative of the process.
2. The direction of flow is from top to bottom and from left to right. Data traditionally flow from source to the destination although they may flow back to the source. One way to indicate this is to draw long flow line back to a source. An alternative way is to repeat the source symbol as a destination. Since it is used more than once in the DFD it is marked with a short diagonal.
3. When a process is exploded into lower level details, they are numbered.
4. The names of data stores and destinations are written in capital letters. Process and dataflow names have the first letter of each work capitalized

A DFD typically shows the minimum contents of data store. Each data store should contain all the data elements that flow in and out.

Questionnaires should contain all the data elements that flow in and out. Missing interfaces redundancies and like is then accounted for often through interviews.

**SAILENT FEATURES OF DFD’S**

1. The DFD shows flow of data, not of control loops and decision are controlled considerations do not appear on a DFD.
2. The DFD does not indicate the time factor involved in any process whether the dataflow take place daily, weekly, monthly or yearly.
3. The sequence of events is not brought out on the DFD.

**TYPES OF DATA FLOW DIAGRAMS**

1. Current Physical
2. Current Logical
3. New Logical
4. New Physical

**CURRENT PHYSICAL:**

In Current Physical DFD process label include the name of people or their positions or the names of computer systems that might provide some of the overall system-processing label includes an identification of the technology used to process the data. Similarly data flows and data stores are often labels with the names of the actual physical media on which data are stored such as file folders, computer files, business forms or computer tapes.

**CURRENT LOGICAL:**

The physical aspects at the system are removed as much as possible so that the current system is reduced to its essence to the data and the processors that transform them regardless of actual physical form.

**NEW LOGICAL**:

This is exactly like a current logical model if the user were completely happy with the user were completely happy with the functionality of the current system but had problems with how it was implemented typically through the new logical model will differ from current logical model while having additional functions, absolute function removal and inefficient flows recognized.

**NEW PHYSICAL:**

The new physical represents only the physical implementation of the new system.

**RULES GOVERNING THE DFD’S**

**PROCESS**

1. No process can have only outputs.
2. No process can have only inputs. If an object has only inputs than it must be a sink.
3. A process has a verb phrase label.

**DATA STORE**

1. Data cannot move directly from one data store to another data store, a process must move data.
2. Data cannot move directly from an outside source to a data store, a process, which receives, must move data from the source and place the data into data store
3. A data store has a noun phrase label.

**SOURCE OR SINK**

The origin and /or destination of data

1. Data cannot move direly from a source to sink it must be moved by a process
2. A source and /or sink has a noun phrase land

**DATA FLOW**

1. A Data Flow has only one direction of flow between symbols. It may flow in both directions between a process and a data store to show a read before an update. The later is usually indicated however by two separate arrows since these happen at different type.
2. A join in DFD means that exactly the same data comes from any of two or more different processes data store or sink to a common location.
3. A data flow cannot go directly back to the same process it leads. There must be at least one other process that handles the data flow produce some other data flow returns the original data into the beginning process.
4. A Data flow to a data store means update (delete or change).
5. A data Flow from a data store means retrieve or use.

A data flow has a noun phrase label more than one data flow noun phrase can appear on a single arrow as long as all of the flows on the same arrow move together as one package.

**LEVEL-1: SYSTEM LEVEL DATA FLOW**

A level-1 DFD describes the next level of details within the system, detailing the data flows between subsystems, which make up the whole.

### DFD for Administrator

**Admin authentication**

**Admin Master**

**Admin Master**

**Admin Master**

**Maintain login info**

**Admin Master**

**Check Modified Details of Admin**

**Check for login Details**

### DFD For Faculty:

staff Master

Member Authentication

staff Master

Check for the login details.

staff Master

Maintain login info

**LEVEL-2: Subsystem level DATA FLOW**

All the projects are feasible given unlimited resources and infinite time. It is both necessary and prudent to evaluate the feasibility of the project at the earliest possible time. Feasibility and the risk analysis are pertained in many ways.

**DFD for Administrator**

**Insert Staff**

**Check for Staff Info**

**Admin Master**

**Admin Master**

**Admin Master**

**Provide**

**security**

#### Admin Master

**Staff profiles view**

**DFD For Faculty**

**Insert Timetable**

**Check for Lecture info**

**Staff Master**

**Staff Master**

**Staff Master**

**Assigning time table each class**

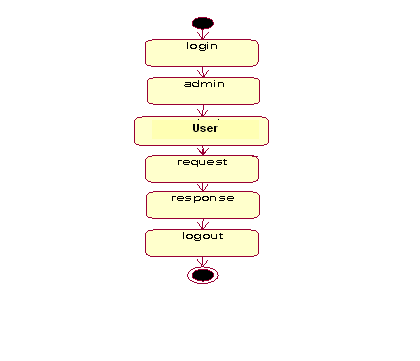
#### Staff Master

**Check for Class**

**6.6. ACTIVITY DIAGRAMS**

A State diagram/Activity diagram is a specification of the sequences of states that an object or an interaction goes through in response to events during its life, together with its responsive action. Every state diagram is having one entry and exit state. And the state can have any number of sub-states. The above state diagram represents, how admin will interact with other objects, and how he will perform actions and change his state.

**Admin Activity Diagram:**

****

**For student:**

login

MyProfile

Send Feedback

Logout

**For staff:**

login

MyProfile

View Feedback

Logout

**6.7. USE CASE DIAGRAMS**

USECASE is a description of a set of sequence of actions that a system performs that yields an observable result of value to a particular things in a model. User is an actor and these are use cases are login, view work details, assign work, approval link, view voter request details, view ward member and helper details.

**Identification of actors:**

**Actor:** Actor represents the role a user plays with respect to the system. An actor interacts with, but has no control over the use cases.

An actor is someone or something that:

* Interacts with or uses the system.
* Provides input to and receives information from the system.
* Is external to the system and has no control over the use cases.

Actors are discovered by examining:

* Who directly uses the system?
* Who is responsible for maintaining the system?
* External hardware used by the system.
* Other systems that need to interact with the system.

**Questions to identify actors:**

* Who is using the system? Or, who is affected by the system? Or, which groups need help from the system to perform a task?
* Who affects the system? Or, which user groups are needed by the system to perform its functions? These functions can be both main functions and secondary functions such as administration.
* Which external hardware or systems (if any) use the system to perform tasks?
* What problems does this application solve (that is, for whom)?
* And, finally, how do users use the system (use case)? What are they doing with the system?

**Identification of use cases:**

**Use case:** A use case can be described as a specific way of using the system from a user’s (actor’s) perspective.

**Admin Use case Diagram**

login

Add Faculty

Add student

View student faculty

View feedback

admin

logout

**Faculty Use Case Diagram:**

login

My Profile

View Feedback

Logout

Faculty

**Student Use case Diagram**

login

My Profile

Send Feedback

Logout

Student

**6.8. SEQUENCE DIAGRAMS**

A sequence diagram is a graphical view of a scenario that shows object interaction in a time-based sequence what happens first, what happens next. Sequence diagrams establish the roles of objects and help provide essential information to determine class responsibilities and interfaces.

There are two main differences between sequence and collaboration diagrams: sequence diagrams show time-based object interaction while collaboration diagrams show how objects associate with each other.

**Object:** An object has state, behavior, and identity. The structure and behavior of similar objects are defined in their common class. Each object in a diagram indicates some instance of a class. An object that is not named is referred to as a class instance. The object icon is similar to a class icon except that the name is underlined. An object's concurrency is defined by the concurrency of its class.

**Message:** A message is the communication carried between two objects that trigger an event. A message carries information from the source focus of control to the destination focus of control. The synchronization of a message can be modified through the message specification. Synchronization means a message where the sending object pauses to wait for results.

**Link:** A link should exist between two objects, including class utilities, only if there is a relationship between their corresponding classes. The link is depicted as a straight line between objects or objects and class instances in a collaboration diagram. If an object links to itself, use the loop version of the icon.

**For admin**

****

**For Lecturer**

****

**For student:**

****

**6.9. CLASS DIAGRAM**

**Identification of analysis classes:**

A class is a set of objects that share a common structure and common behavior (the same attributes, operations, relationships and semantics). A class is an abstraction of real-world items.

There are 4 approaches for identifying classes:

1. Noun phrase approach:
2. Common class pattern approach.
3. Use case Driven Sequence or Collaboration approach.
4. Classes , Responsibilities and collaborators Approach
5. **Noun Phrase Approach:**

The guidelines for identifying the classes:

* 1. Look for nouns and noun phrases in the use cases.
  2. Some classes are implicit or taken from general knowledge.
  3. All classes must make sense in the application domain; Avoid computer implementation classes – defer them to the design stage.
  4. Carefully choose and define the class names.

After identifying the classes we have to eliminate the following types of classes:

1. Redundant classes.
2. Adjective classes.
3. **Common class pattern approach:**

The following are the patterns for finding the candidate classes:

* 1. Concept class.
  2. Events class.
  3. Organization class
  4. Peoples class
  5. Places class
  6. Tangible things and devices class.

1. **Use case driven approach:**

We have to draw the sequence diagram or collaboration diagram. If there is need for some classes to represent some functionality then add new classes which perform those functionalities.

1. **CRC approach:**

The process consists of the following steps:

* 1. Identify classes’ responsibilities ( and identify the classes )
  2. Assign the responsibilities
  3. Identify the collaborators.

**Identification of responsibilities of each class:**

The questions that should be answered to identify the attributes and methods of a class respectively are:

1. What information about an object should we keep track of?
2. What services must a class provide?

**Identification of relationships among the classes:**

Three types of relationships among the objects are:

**Association:** How objects are associated?

**Super-sub structure:** How are objects organized into super classes and sub classes?

**Aggregation:** What is the composition of the complex classes?

**Guidelines for identifying the tentative associations:**

* A dependency between two or more classes may be an association. Association often corresponds to a verb or prepositional phrase.
* A reference from one class to another is an association. Some associations are implicit or taken from general knowledge.

**Super-sub class relationships**

Super-sub class hierarchy is a relationship between classes where one class is the parent class of another class (derived class).This is based on inheritance. This hierarchy is represented with Generalization.

**Guidelines for identifying the super-sub relationship, a generalization are**

1***.* Top-down*:*** Look for noun phrases composed of various adjectives in a class name. Avoid excessive refinement. Specialize only when the sub classes have significant behavior.

2.**Bottom-up*:*** Look for classes with similar attributes or methods. Group them by moving the common attributes and methods to an abstract class. You may have to alter the definitions a bit.

3.**Reusability*:*** Move the attributes and methods as high as possible in the hierarchy.

4**. Multiple inheritances*:*** Avoid excessive use of multiple inheritances. One way of getting benefits of multiple inheritances is to inherit from the most appropriate class and add an object of another class as an attribute

`The class diagram is core to object-oriented design.  It describes the types of objects in the system and the static relationships between them.

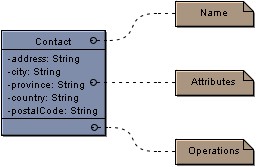
**Packages**

Packages allow you to break up a large number of objects into related groupings.  In many object oriented languages (such as Java), packages are used to provide scope and division to classes and interfaces.  In the UML, packages serve a similar, but broader purpose.

Package

**Classes**

The core element of the class diagram is the class.  In an object oriented system, classes are used to represent entities within the system; entities that often relate to real world objects.



The *Contact* class above is an example of a simple class that stores location information.

Classes are divided into three sections:

**Top**: The **name**, **package** and **stereotype** are shown in the upper section of the class

**Centre**: The centre section contains the attributes of the class.

**Bottom**: In the lower section are the **operations** that can be performed on the class.

**Attributes**

An **attribute** is a property of a class.  In the example above, we are told that a *Contact* has an address, a city, a province, a country and a postal code.  It is generally understood that when implementing the class, functionality is provided to set and retrieve the information stored in attributes. The format for attributes is:

*Visibility name: type = default Value*

The visibility is as follows:

|  |  |
| --- | --- |
| **-** | Private |
| **+** | Public |
| **#** | Protected |
| **~** | Package |

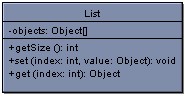
In object oriented design, it is generally preferred to keep most attributes private.

**Static**: attributes that are static only exist once for all instances of the class.  In the example above, if we set *city* to be static, any time we used the *Contact* class the *city* attribute would always have the same value.

**Final:** if an attribute is declared final, it's value cannot be changed.  The attribute is a constant.

**Operations**

The **operations** listed in a class represent the functions or tasks that can be performed on the data in the class.



In the *List* class above, there is one attribute (a private array of Objects) and three operations.

The format for operations is:

*visibility name (parameters): type*

The format is very similar to that of the attribute except with the removal of a default value and the addition of parameters.

Parameters take the format:

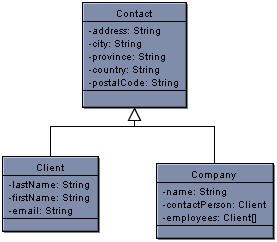
*direction name: type = default value*

The direction can be one of *in*, *out*, *input* or it can be unspecified.

In Visual Case you can show and hide the parameter list for a class or all classes on a diagram.  If the list is hidden and an operation has parameters, three dots are shown (...) to indicate that parameters exist, but are hidden.  Sometimes operations have numerous parameters that need not be shown all the time.

**Generalization**

The **generalization** link is used between two classes to show that a class incorporates all of the attributes and operations of another, but adds to them in some way.



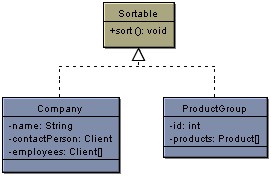
In the above diagram, we again see our *Contact* class, only now with two child classes.  We can say that *Client* and *Company* **inherit**, **generalize** or **extend** *Contact*.  In each of *Client* and *Company* all of the attributes in *Contact* (address, city, etc.) exist, but with more information added.  In the above situation *Contact* is said to be the **super class** of *Client* and *Company*.

Above, *OntarioTaxCalculator* redefines or **overrides** the implementation of the method in BasicTaxCalculator.  Essentially, the code is different but the operation is called the same way.

Sometimes you may want to force children to override methods in a parent class.  In this case you can define the methods in the super class as **abstract**.  If a class has abstract operations, the class itself is considered abstract.  Abstract methods and classes are shown in italics.  Not all of the operations in an abstract class have to be abstract.

**Interfaces**

Many object oriented programming languages do not allow for multiple inheritance.  The **interface** is used to solve the limitations posed by this.  For example, in the earlier class diagram *Client* and *Company* both generalize *Contact* but one or the other child classes may have something in common with a third class that we do not want to duplicate in multiple classes.



The interface *Sort able*, is used in the above example to show that both *Company* and *Product* implement the *sort* operation.  We can say that *Company* and *Product* **implement** *Sort able* or that they are *Sort able*.  Because Product already generalizes *Contact*, we could not also allow it to generalize *Sort able*.  Instead, we made *Sort able* an interface and added a **realization** link to show the implementation.

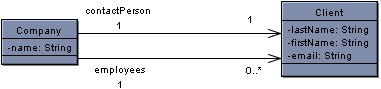
Interfaces are very similar to abstract classes with the exception that they do not have any attributes.  As well, unlike a class, all of the operations in an interface have no implementation.  The child Classes *Company* and *Product* are forced to implement the *sort* operation in its entirety.

**Associations**

Classes can also contain references to each other.  The *Company* class has two attributes that reference the Client class.



Although this is perfectly correct, it is sometimes more expressive to show the attributes as **associations**.



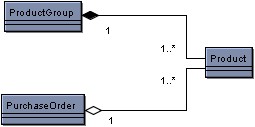
The above two associations have the same meaning as the attributes in the old version of the *Contact* class.

The first association (the top one) represents the old *contact Person* attribute.  There is one contact person in a single *Company*.  The **multiplicity** of the association is one to one meaning that for every *Company* there is one and only one *contact Person* and for each *contact Person* there is one *Company*.  In the bottom association there are zero for each company.  Multiplicities can be anything you specify.  Some examples are shown:

|  |  |
| --- | --- |
| **0** | Zero |
| **1** | One |
| **1..\*** | one or many |
| **1..2, 10..\*** | one, two or ten and above but **not** three through nine |

The arrows at the end of the associations represent their **navigability.**  In the above examples, the *Company* references *Clients*, but the *Client* class does not have any knowledge of the *Company*.  You can set the navigability on either, neither or both ends of your associations.  If there is no navigability shown then the navigability is unspecified.

**Aggregation and Composition**



The above example shows an **aggregation** association and a **composition** association.

The **composition** association is represented by the solid diamond.  It is said that *Product Group* is **composed** of *Products*.  This means that if a *Product Group* is destroyed, the *Products* within the group are destroyed as well.

The **aggregation** association is represented by the hollow diamond.  *Purchase Order* is an **aggregate** of *Products*.  If a *Purchase Order* is destroyed, the *Products* still exist.

If you have trouble remembering the difference between composition and aggregation, just think of the alphabet.  Composition means destroy and the letters 'c' and 'd' are next to each other.

**Dependencies**

A **dependency** exists between two elements if changes to one will affect the other.  If for example, a class calls an operation in another class, then a dependency exists between the two.  If you change the operation, than the dependent class will have to change as well.  When designing your system, the goal is to minimize dependencies.

PackageDiagram

To help clarify the dependencies in your design, you may wish to draw a **Package Diagram**.  A package diagram is essentially a class diagram with only packages and dependencies showing.  Dependencies can exist between any components in the UML however at the highest level, dependencies will exist between packages.  Within a package, the dependencies may be too numerous to specify.  That is not to say that numerous dependencies are okay.  Even within a package you want to limit the dependencies, however between packages in particular you should be strict about the number of dependencies that exist.  In general, the fewer the dependencies the more **scaleable** and **maintainable** your system will be

Administrator



UserName : Varchar2



Password : Varchar2



Staff()



Lecturer()



Student()

Faculty



User Name : varchar



Password : varchar



Lecturer class assign()



Student Info()



Marks Entry()



Time table Entry()



Holidays()



Student



Username : varchar



Password : varchar



Mark reports()



View schedules()



View class Tme tables()



Staff assigning Info()



Profile()

Dependency

Dependency

**6.10. COLLOBORATION DIAGRAMS**

Collaboration diagram is an interaction diagram or communication diagram. A Communication diagram models the interactions between objects or parts in terms of sequenced messages. Communication diagrams represent a combination of information taken from class, sequence and [Use Case Diagrams](http://en.wikipedia.org/wiki/Use_case_diagram) describing both the static structure and dynamic behavior of a system.

Communication diagrams use the free-form arrangement of objects and links as used in Object diagrams. In order to maintain the ordering of messages in such a free-form diagram, messages

Are labeled with a chronological number and placed near the link the message is sent over. Reading a communication diagram involves starting at message 1.0, and following the messages from object to object.

Communication diagrams show a lot of the same information as sequence diagrams, but because of how the information is presented, some of it is easier to find in one diagram than the other. Communication diagrams show which elements each one interacts with better, but sequence diagrams show the order in which the interactions take place more clearly.

### Basic Collaboration Diagram Symbols and Notations

##### Class roles

Class roles describe how objects behave. Use the UML object symbol to illustrate class roles, but don't list object attributes.

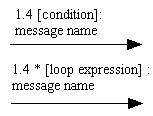
Class roles

##### Association roles

Association roles describe how an association will behave given a particular situation. You can draw association roles using simple lines labeled with stereotypes.  
Association roles

##### Messages

Unlike sequence diagrams, collaboration diagrams do not have an explicit way to denote time and instead number messages in order of execution. Sequence numbering can become nested using the Dewey decimal system. For example, nested messages under the first message are labeled 1.1, 1.2, 1.3, and so on



Admin



**For student:**

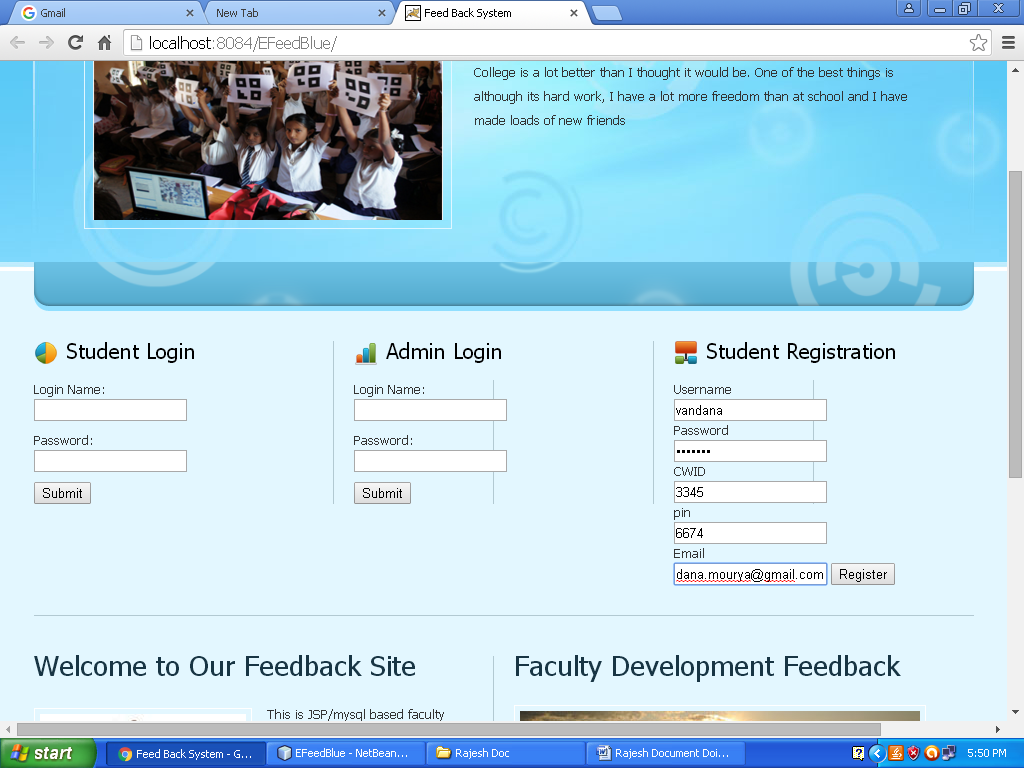
****

**Chapter 7**

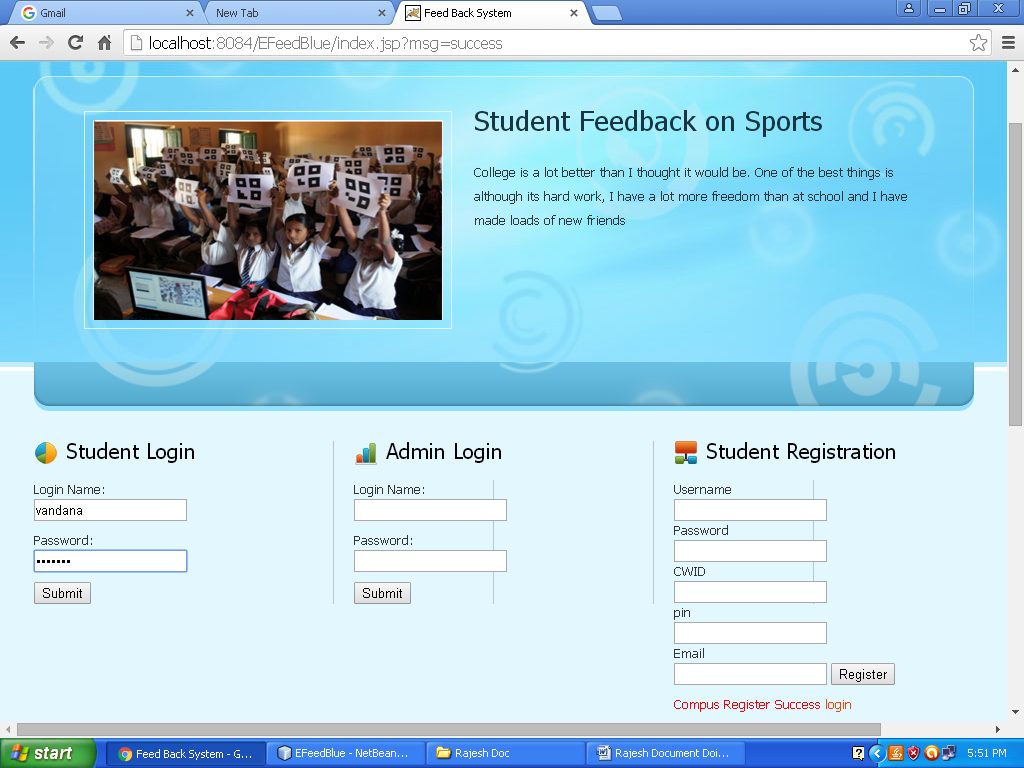
**OUTPUT SCREENS**



Home Screen



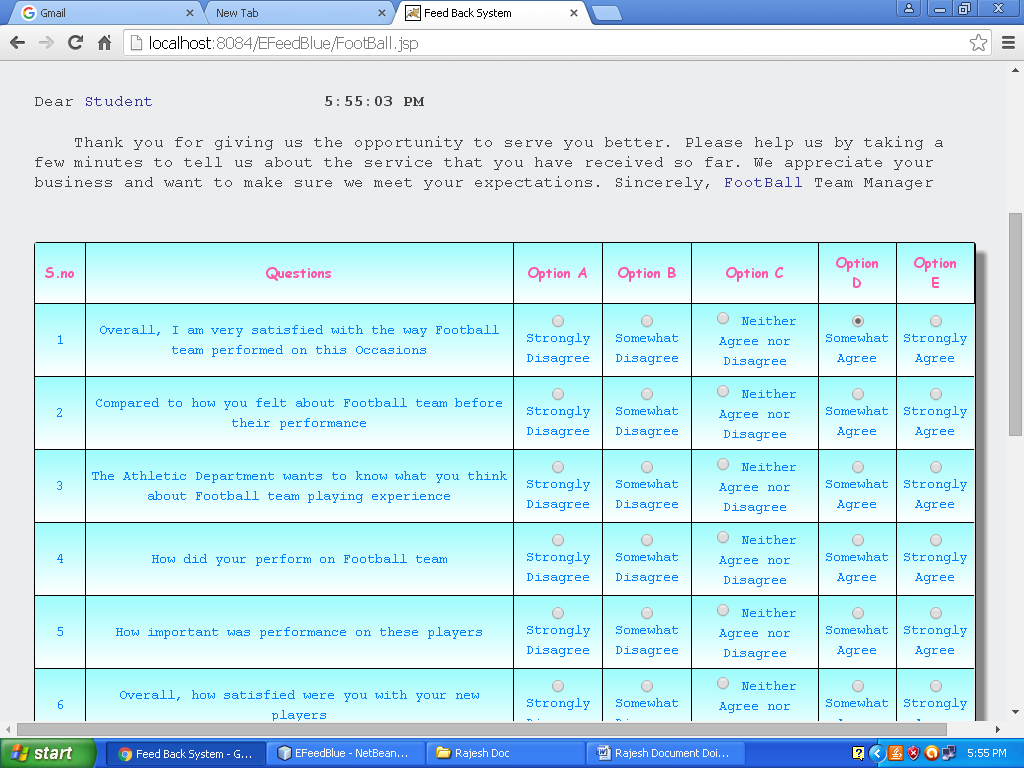
Student Registration



Student Login



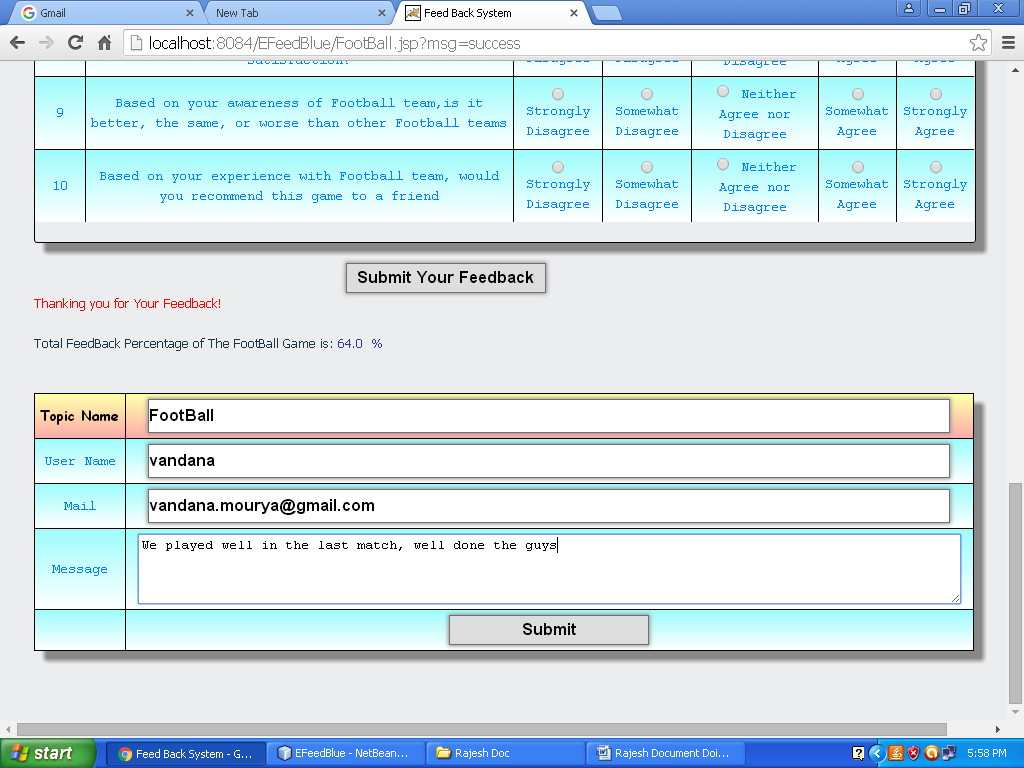
Stage -1: Student Home Page in that student can select the menus and he will get appropriate forms. Using those menus student can give the feedback.



Stage -2: Feedback form for Sports.



Sports Feedback and Calculated the Feedback Score



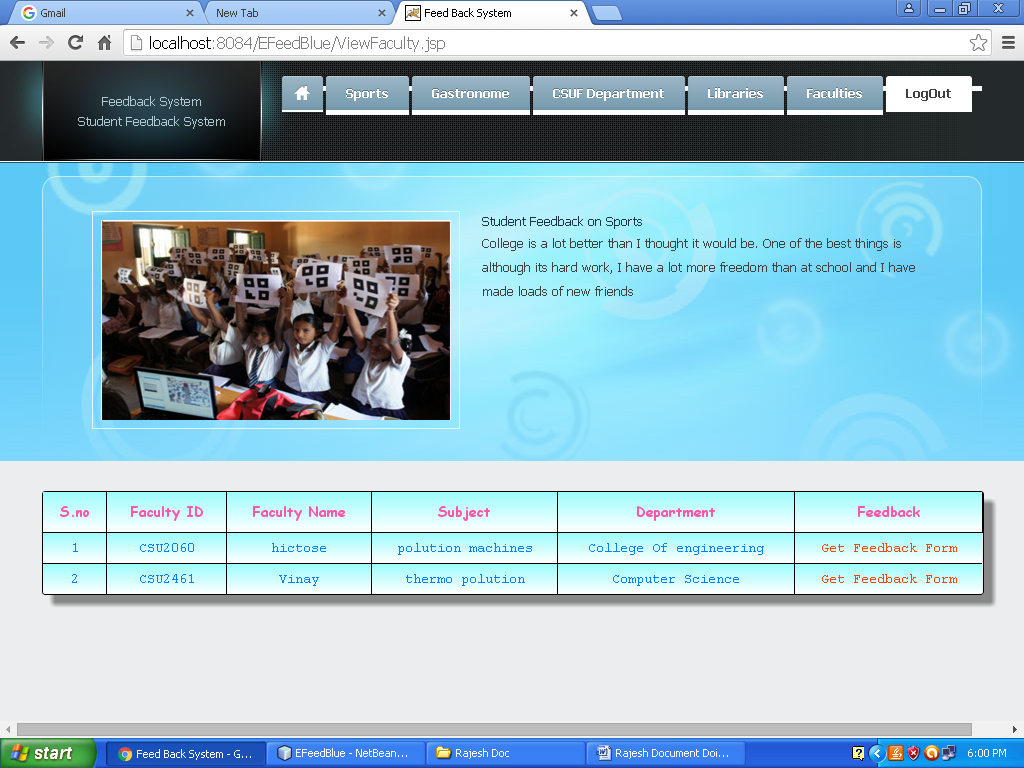
Descriptive Feedback form



College Of Education Form



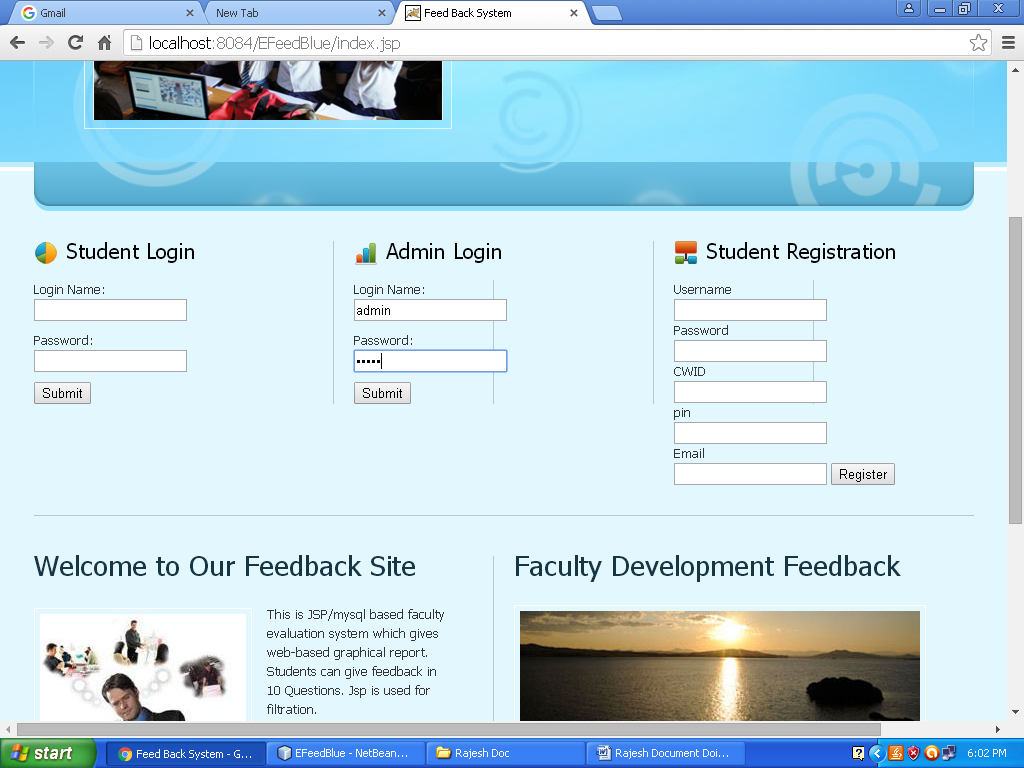
Faculty Feedback Form



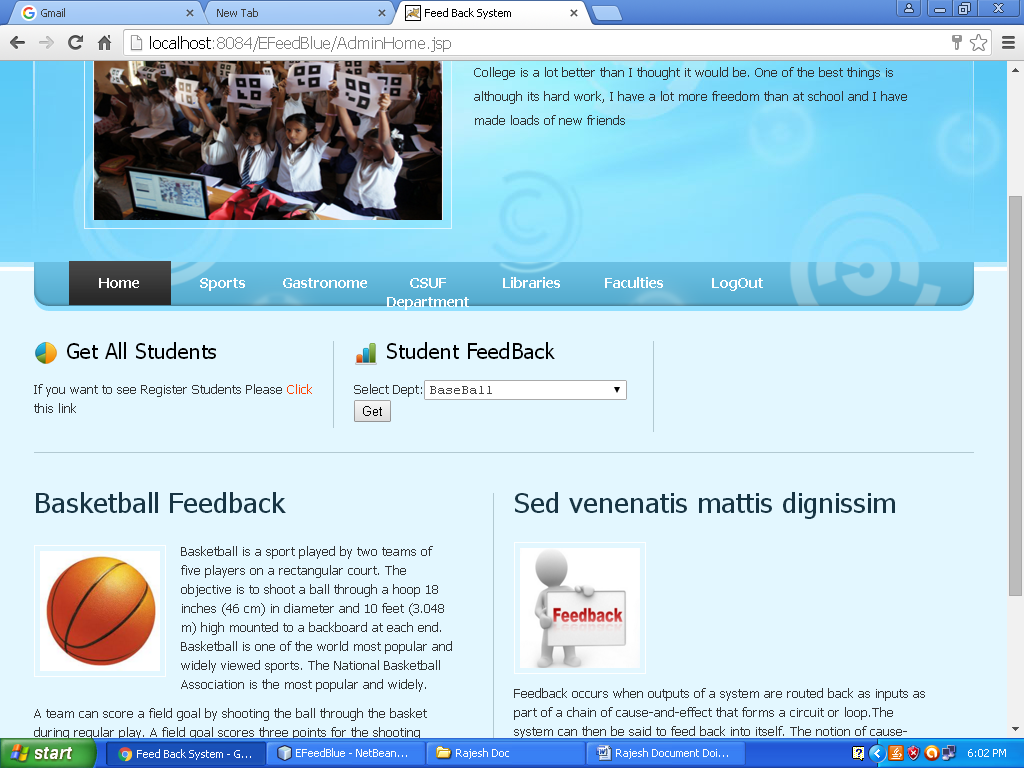
List The No Of Faculties



Selected Faculty Feedback Form



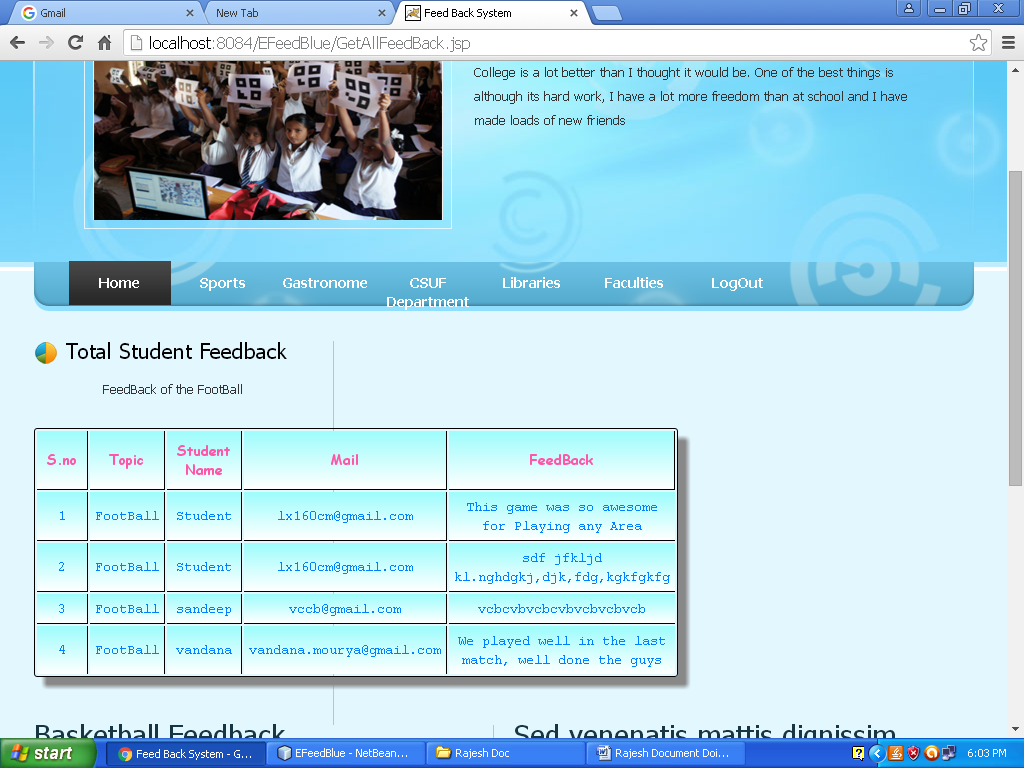
Admin Login Form



Admin Home Page



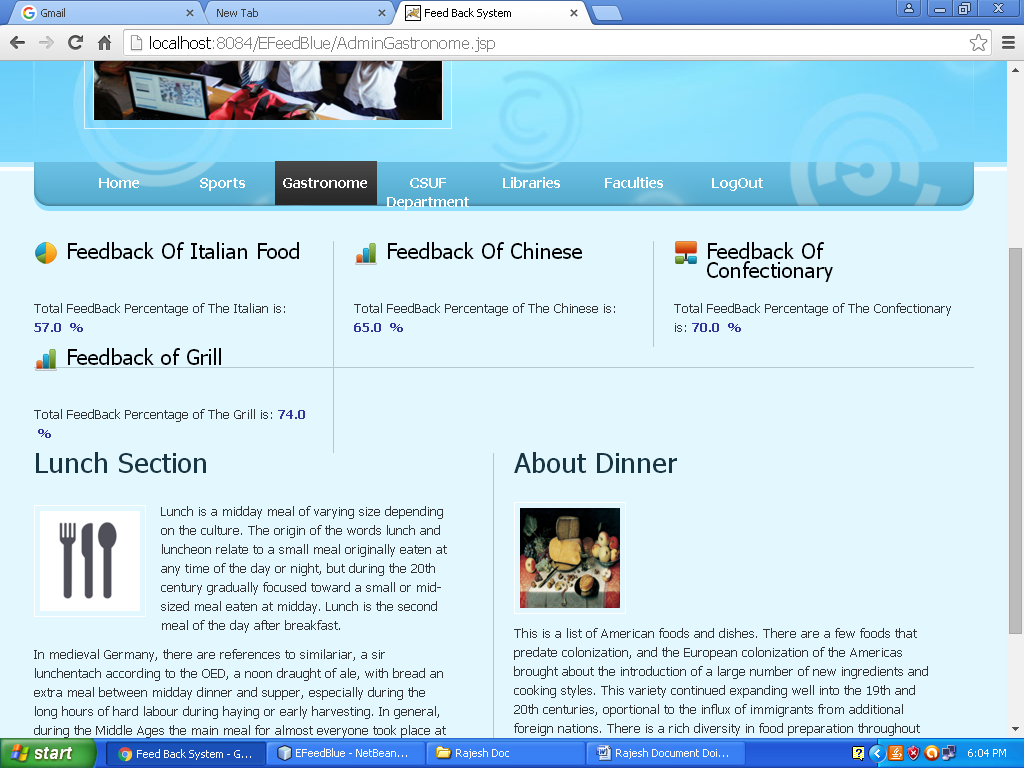
Getting all Student Data



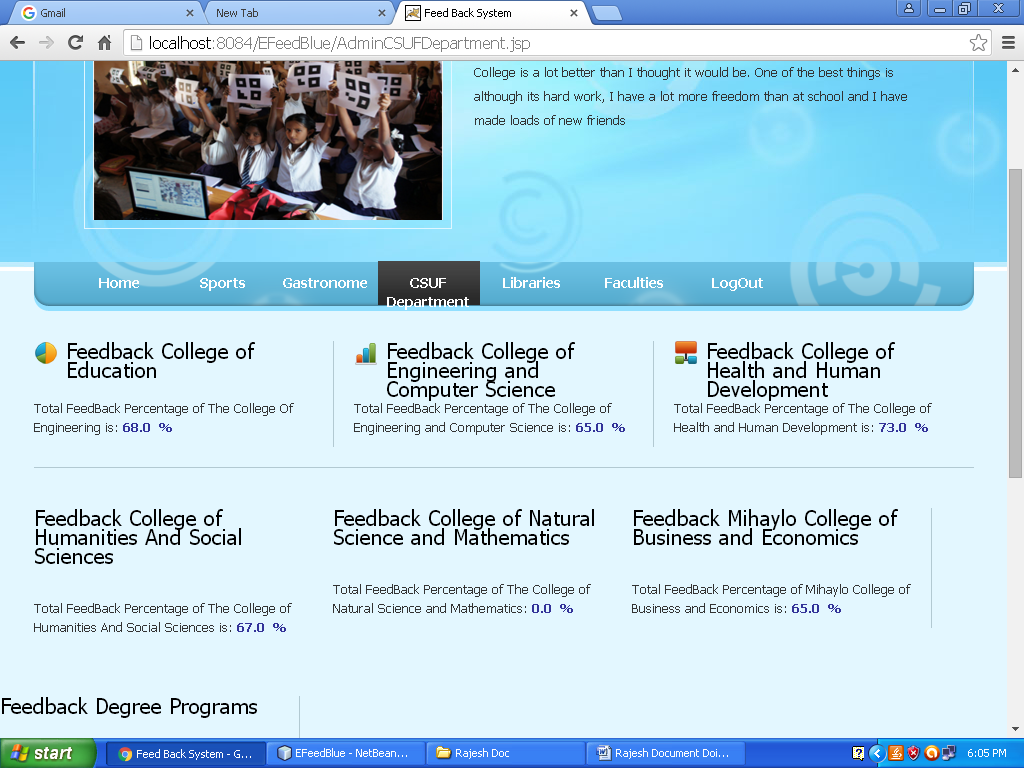
Descriptive Feedback Form



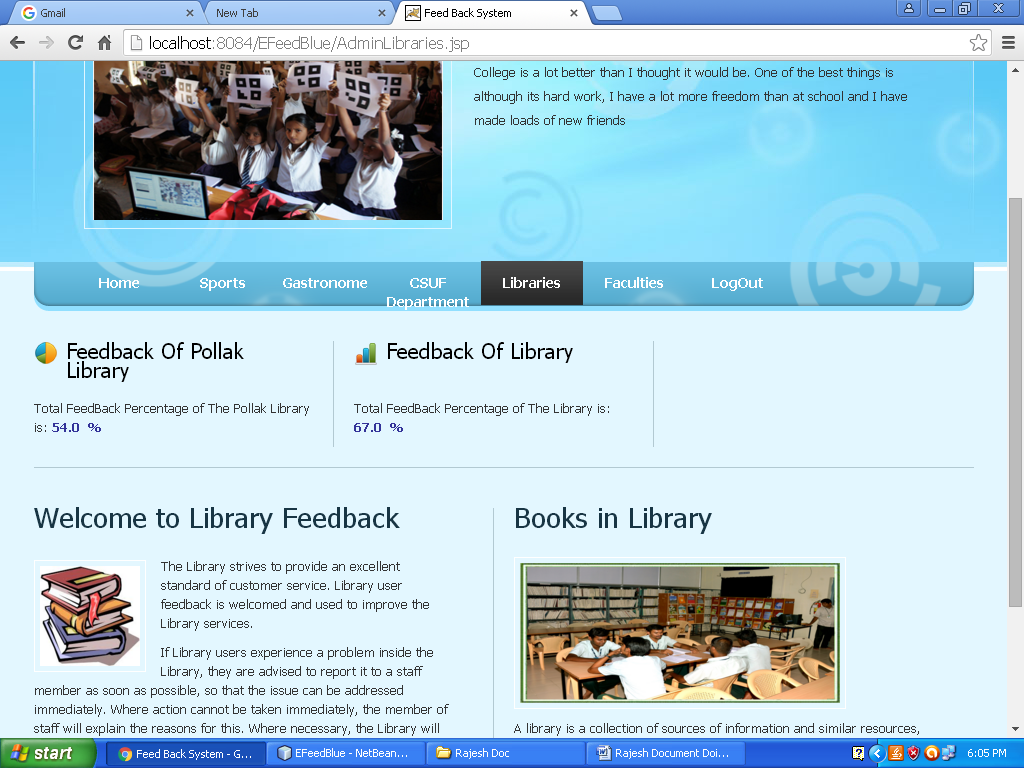
Sports Feedback Scores



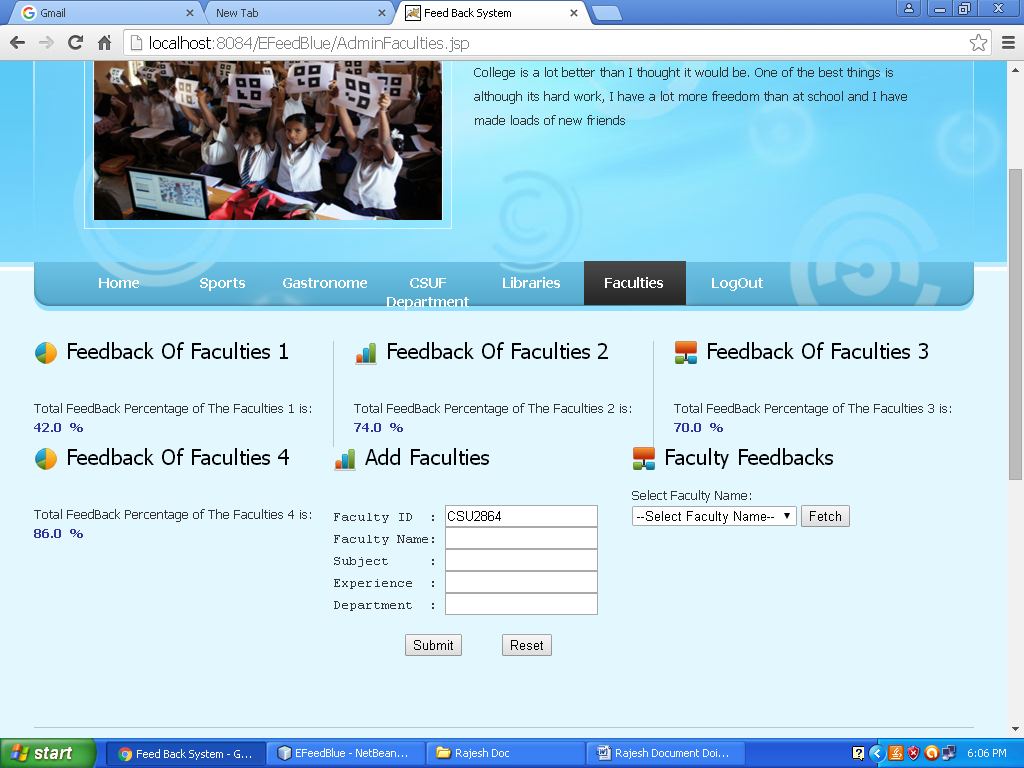
Gastronome Feedback Form



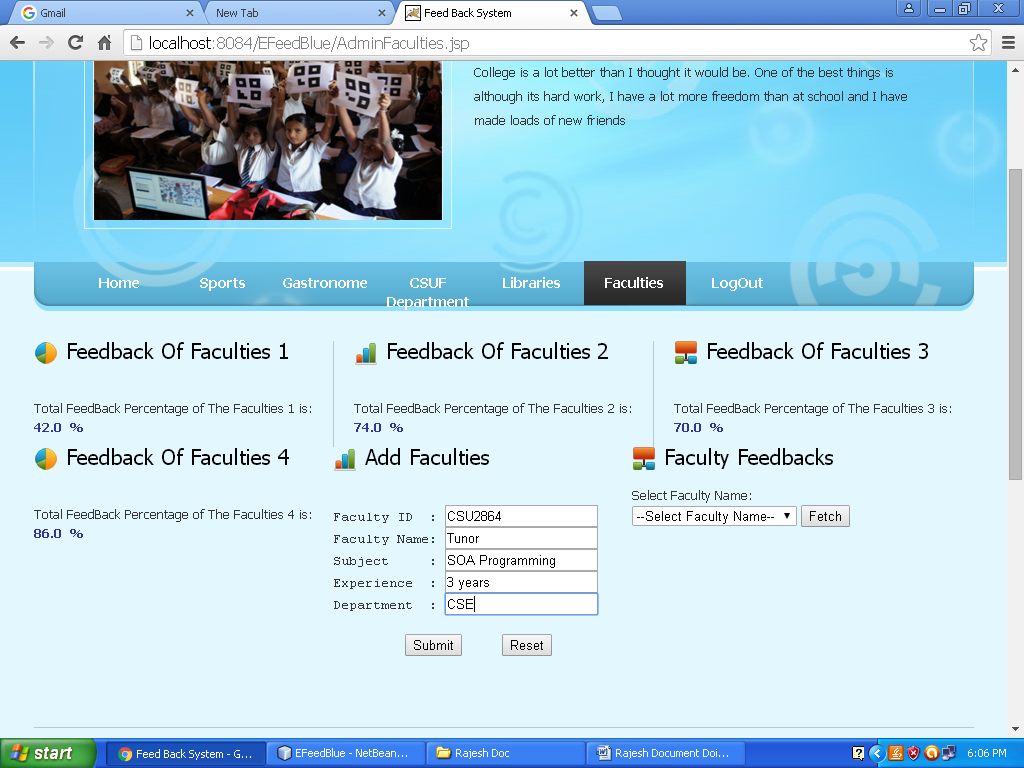
CSUF Department Feedback Form



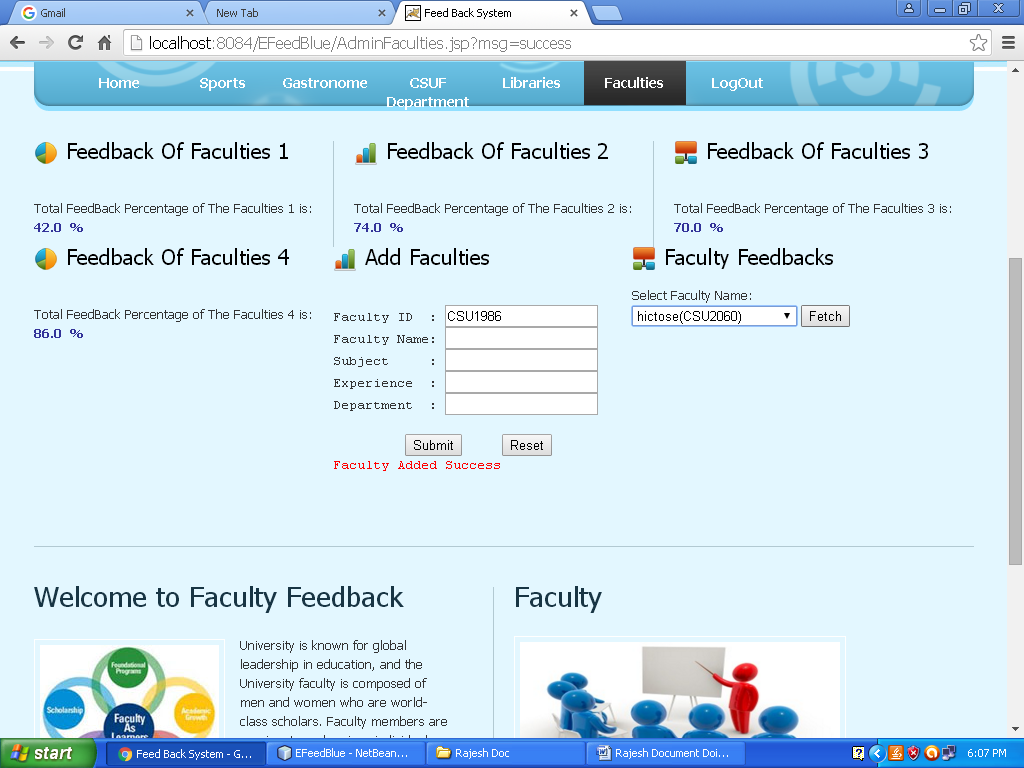
Libraries Feedback Score



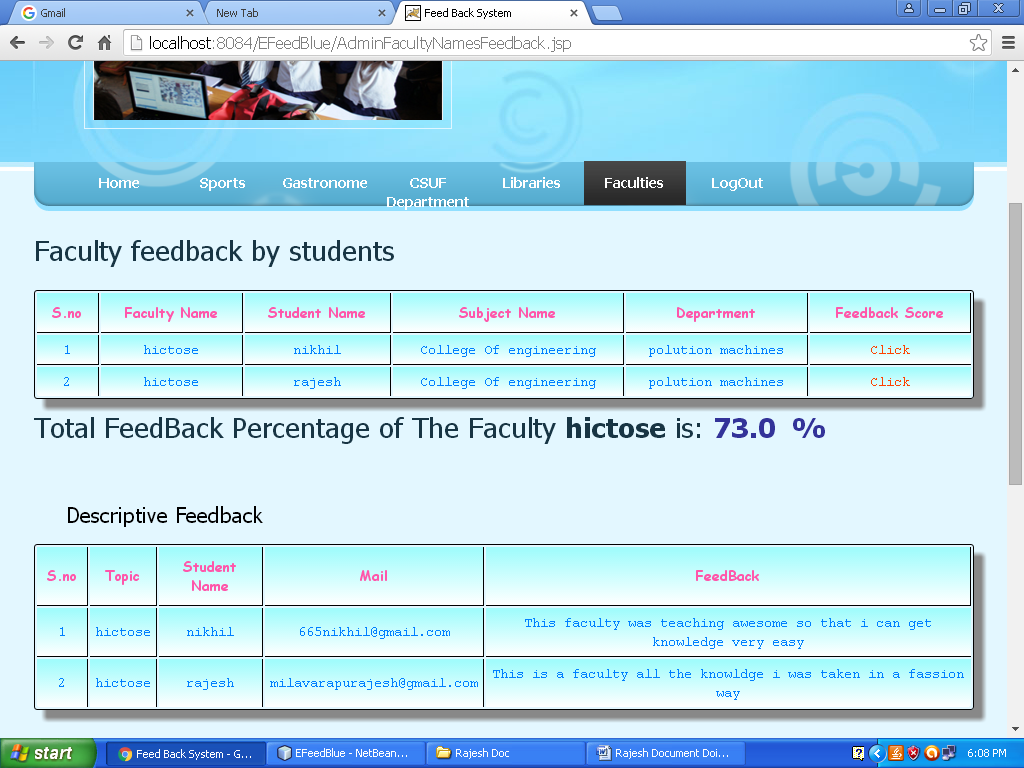
Faculties Feedback Scores



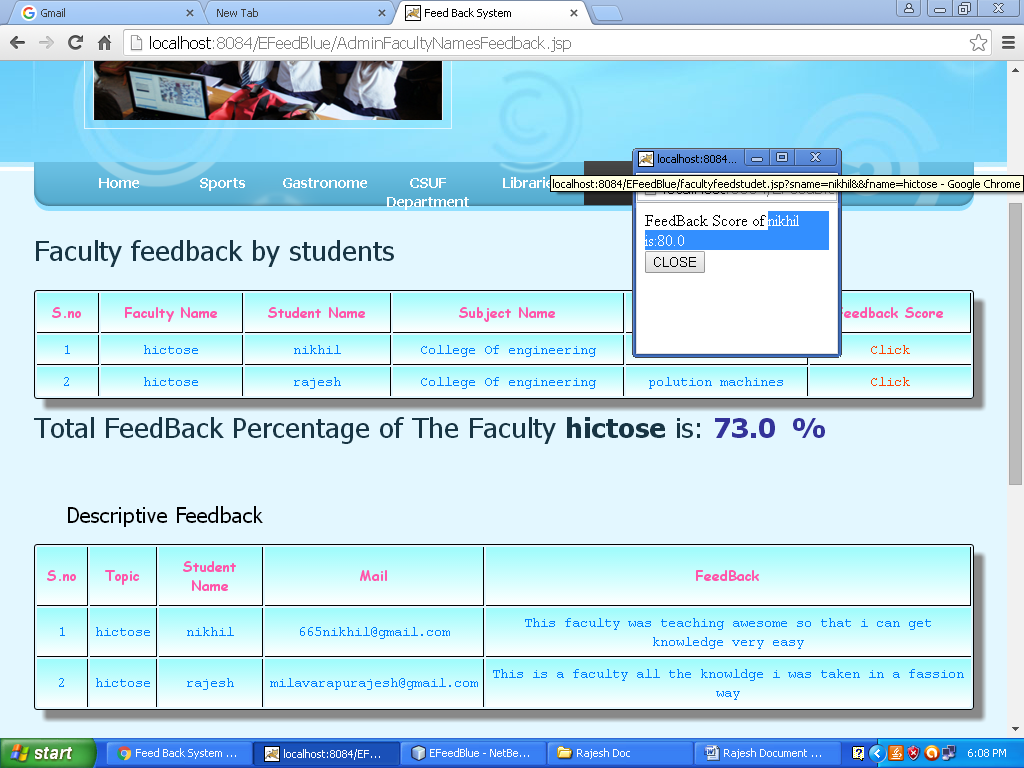
Adding Faculty Data



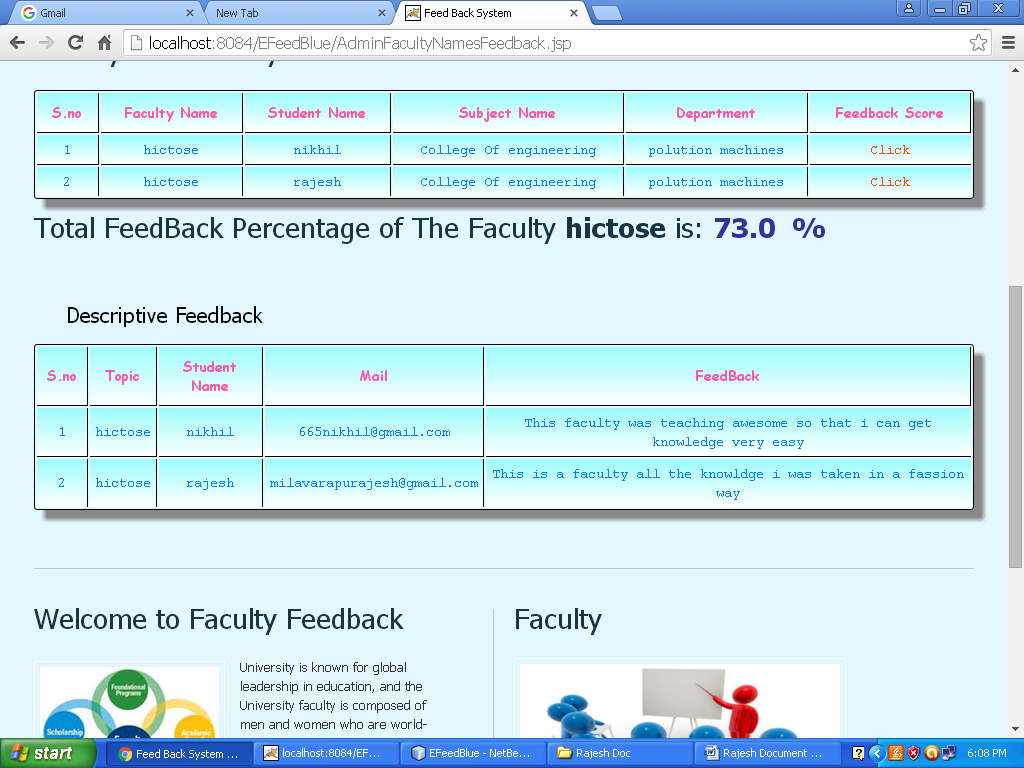
Selecting Faculty And Getting his feedback



Feedback of the Particular Faculty



Student Individual Feedback Score calculation



Descriptive Feedback of the Faculty by student

**Chapter 8**

**SYSTEM TESTING AND IMPLEMENTATION**

**8.1. INTRODUCTION**

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. In fact, testing is the one step in the software engineering process that could be viewed as destructive rather than constructive.

A strategy for software testing integrates software test case design methods into a well-planned series of steps that result in the successful construction of software. Testing is the set of activities that can be planned in advance and conducted systematically. The underlying motivation of program testing is to affirm software quality with methods that can economically and effectively apply to both strategic to both large and small-scale systems.

**8.2. STRATEGIC APPROACH TO SOFTWARE TESTING**

The software engineering process can be viewed as a spiral. Initially system engineering defines the role of software and leads to software requirement analysis where the information domain, functions, behavior, performance, constraints and validation criteria for software are established. Moving inward along the spiral, we come to design and finally to coding. To develop computer software we spiral in along streamlines that decrease the level of abstraction on each turn.

A strategy for software testing may also be viewed in the context of the spiral. Unit testing begins at the vertex of the spiral and concentrates on each unit of the software as implemented in source code. Testing progress is done by moving outward along the spiral to integration testing, where the focus is on the design and the construction of the software architecture. Talking another turn on outward on the spiral we encounter validation testing where requirements established as part of software requirements analysis are validated against the software that has been constructed. Finally we arrive at system testing, where the software and other system elements are tested as a whole.

UNIT TESTING

MODULE TESTING

SUB-SYSTEM TESING

SYSTEM TESTING

ACCEPTANCE TESTING

Component Testing

Integration Testing

User Testing

**8.3. Unit Testing**

Unit testing focuses verification effort on the smallest unit of software design, the module. The unit testing we have is white box oriented and some modules the steps are conducted in parallel.

**1. WHITE BOX TESTING**

This type of testing ensures that

* All independent paths have been exercised at least once
* All logical decisions have been exercised on their true and false sides
* All loops are executed at their boundaries and within their operational bounds
* All internal data structures have been exercised to assure their validity.

To follow the concept of white box testing we have tested each form .we have created independently to verify that Data flow is correct, All conditions are exercised to check their validity, All loops are executed on their boundaries.

**2. BASIC PATH TESTING**

Established technique of flow graph with Cyclomatic complexity was used to derive test cases for all the functions. The main steps in deriving test cases were:

Use the design of the code and draw correspondent flow graph.

Determine the Cyclomatic complexity of resultant flow graph, using formula:

V(G)=E-N+2 or

V(G)=P+1 or

V(G)=Number Of Regions

Where V(G) is Cyclomatic complexity,

E is the number of edges,

N is the number of flow graph nodes,

P is the number of predicate nodes.

Determine the basis of set of linearly independent paths.

**3. CONDITIONAL TESTING**

In this part of the testing each of the conditions were tested to both true and false aspects. And all the resulting paths were tested. So that each path that may be generate on particular condition is traced to uncover any possible errors.

**4. DATA FLOW TESTING**

This type of testing selects the path of the program according to the location of definition and use of variables. This kind of testing was used only when some local variable were declared. The *definition-use chain* method was used in this type of testing. These were particularly useful in nested statements.

**5. LOOP TESTING**

In this type of testing all the loops are tested to all the limits possible. The following exercise was adopted for all loops:

* All the loops were tested at their limits, just above them and just below them.
* All the loops were skipped at least once.
* For nested loops test the inner most loop first and then work outwards.
* For concatenated loops the values of dependent loops were set with the help of connected loop.
* Unstructured loops were resolved into nested loops or concatenated loops and tested as above.

Each unit has been separately tested by the development team itself and all the input have been validated.

**8.4. Test Cases**

Test Case Report1

(Use one template for each test case)

|  |  |  |  |
| --- | --- | --- | --- |
| **GENERAL INFORMATION** | | | |
| **Test Stage:** | Unit Functionality Interface  Performance Acceptance | | |
| **Test Date:** | 1/08/2017 | **System Date, if applicable:** | 1/08/2017 |
| **Tester:** | vasu | **Test Case Number:** | 1 |
| **Test Case Description:** | Unit testing focuses on verifying the effort on the smallest unit of software-module. The local data structure is examined to ensure that the date stored temporarily maintains its integrity during all steps in the algorithm’s execution. Boundary conditions are tested to ensure that the module operates properly at boundaries established to limit or restrict processing. | | |
| **Results:** | Pass(OK) Fail |
| **INTRODUCTION** | | | |
| **Requirement(s) to be tested:** | Username Text field and Password Text field and the Authority Text Area. | | |
| **Roles and Responsibilities:** | Gathering the Requirements of the Project Designing and Testing. | | |
| **Set Up Procedures:** | By Installing Net Beans. | | |
| **ENVIRONMENTAL NEEDS** | | | |
| **Hardware:** | PC with Minimum 20GB Hard Disk and 1GB RAM. | | |
| **Software:** | Windows XP/2000, ORACLE, NetBeans IDE 7.0. | | |
| **TEST** | | | |
| **Test Items and Features:** | Username and Password. | | |
| **Procedural Steps:** | If the User enters the correct username and password it will be redirected to another appropriate page so that we can confirm test is accepted. | | |
| **Expected Results of Case:** | If the page is redirected we can confirm the result of this Test case is succeeded. | | |

Test Case Report2

(Use one template for each test case)

|  |  |  |  |
| --- | --- | --- | --- |
| **GENERAL INFORMATION** | | | |
| **Test Stage:** | Unit Functionality Interface  Performance Acceptance | | |
| **Test Date:** | 2/08/2017 | **System Date, if applicable:** | 2/08/2017 |
| **Tester:** | vasu | **Test Case Number:** | 2 |
| **Test Case Description:** | Unit testing focuses on verifying the effort on the smallest unit of software-module. The local data structure is examined to ensure that the date stored temporarily maintains its integrity during all steps in the algorithm’s execution. Boundary conditions are tested to ensure that the module operates properly at boundaries established to limit or restrict processing. | | |
| **Results:** | Pass(OK) Fail |
| **INTRODUCTION** | | | |
| **Requirement(s) to be tested:** | The registered Details should match with corresponding Student\_ID and Password, Send feedbacks. | | |
| **Roles and Responsibilities:** | Gathering the Requirements of the Project Designing and Testing. | | |
| **Set Up Procedures:** | By Installing NetBeans IDE 6.0. | | |
| **ENVIRONMENTAL NEEDS** | | | |
| **Hardware:** | PC with Minimum 20GB Hard Disk and 1GB RAM. | | |
| **Software:** | Windows XP/2000, ORACLE, NetBeans IDE 6.0. Windows XP/2000, ORACLE, NetBeans IDE 6.0. | | |
| **TEST** | | | |
| **Test Items and Features:** | Customer Loan, Payment Details | | |
| **Procedural Steps:** | If the User enters the correct username and password it will be redirected to another appropriate page so that we can confirm test is accepted. | | |
| **Expected Results of Case:** | If the page is redirected we can confirm the result of this Test case is succeeded. | | |

**Chapter 9**

**System Security**

# 9.1. Introduction

The protection of computer based resources that includes hardware, software, data, procedures and people against unauthorized use or natural

Disaster is known as System Security.

System Security can be divided into four related issues:

* Security
* Integrity
* Privacy
* Confidentiality

**SYSTEM SECURITY** refers to the technical innovations and procedures applied to the hardware and operation systems to protect against deliberate or accidental damage from a defined threat.

**DATA SECURITY** is the protection of data from loss, disclosure, modification and destruction.

**SYSTEM INTEGRITY** refers to the power functioning of hardware and programs, appropriate physical security and safety against external threats such as eavesdropping and wiretapping.

**PRIVACY** defines the rights of the user or organizations to determine what information they are willing to share with or accept from others and how the organization can be protected against unwelcome, unfair or excessive dissemination of information about it.

**CONFIDENTIALITY** is a special status given to sensitive information in a database to minimize the possible invasion of privacy. It is an attribute of information that characterizes its need for protection.

## 9.2. SECURITY IN SOFTWARE

System security refers to various validations on data in form of checks and controls to avoid the system from failing. It is always important to ensure that only valid data is entered and only valid operations are performed on the system. The system employees two types of checks and controls:

**CLIENT SIDE VALIDATION**

Various client side validations are used to ensure on the client side that only valid data is entered. Client side validation saves server time and load to handle invalid data. Some checks imposed are:

* JavaScript in used to ensure those required fields are filled with suitable data only. Maximum lengths of the fields of the forms are appropriately defined.
* Forms cannot be submitted without filling up the mandatory data so that manual mistakes of submitting empty fields that are mandatory can be sorted out at the client side to save the server time and load.
* Tab-indexes are set according to the need and taking into account the ease of user while working with the system.

**SERVER SIDE VALIDATION**

Some checks cannot be applied at client side. Server side checks are necessary to save the system from failing and intimating the user that some invalid operation has been performed or the performed operation is restricted. Some of the server side checks imposed is:

* Server side constraint has been imposed to check for the validity of primary key and foreign key. A primary key value cannot be duplicated. Any attempt to duplicate the primary value results into a message intimating the user about those values through the forms using foreign key can be updated only of the existing foreign key values.
* User is intimating through appropriate messages about the successful operations or exceptions occurring at server side.
* Various Access Control Mechanisms have been built so that one user may not agitate upon another. Access permissions to various types of users are controlled according to the organizational structure. Only permitted users can log on to the system and can have access according to their category. User- name, passwords and permissions are controlled o the server side.
* Using server side validation, constraints on several restricted operations are imposed.

**Chapter 10**

**CONCLUSION**

Feedback System is a web based application. It will mainly deal with the Feedback information system in colleges. By using this application the students will sends feedbacks to administrator regarding their issues.

**BENEFITS:**

The project is identified by the merits of the system offered to the user. The merits of this project are as follows: -

* It’s a web-enabled project.
* This project offers user to enter the data through simple and interactive forms. This is very helpful for the client to enter the desired information through so much simplicity.
* The user is mainly more concerned about the validity of the data, whatever he is entering. There are checks on every stages of any new creation, data entry or updating so that the user cannot enter the invalid data, which can create problems at later date.
* User is provided the option of monitoring the records he entered earlier. He can see the desired records with the variety of options provided by him.
* From every part of the project the user is provided with the links through framing so that he can go from one option of the project to other as per the requirement. This is bound to be simple and very friendly as per the user is concerned. That is, we can say that the project is user friendly which is one of the primary concerns of any good project.
* Data storage and retrieval will become faster and easier to maintain because data is stored in a systematic manner and in a single database.
* Decision making process would be greatly enhanced because of faster processing of information since data collection from information available on computer takes much less time then manual system

**Chapter 11**

**FURURE ENHANCEMENT**

In this project in future we will implement to get the message alerts to mobile students to get the username and password.

**Chapter 12**

**BIBLIOGRAPHY**

**The Sites** (In, I assure you, random order)

[http://www.javaalmanac.com](http://www.javaalmanac.com/) - The online counterpart of the Java Developer's Almanac - highly recommended for quick code snippets, and if you like the site, buy the book.   
  
[http://www.onjava.com](http://www.onjava.com/) - O'Reilly's Java website. New articles weekly.   
  
[http://java.sun.com](http://java.sun.com/) - The official Java developer website - new articles posted weekly.   
  
<http://www.java.net> - The Java community website hosted by Sun Microsystems.   
  
[http://www.javaworld.com](http://www.javaworld.com/) - One of the originals. Weekly updates of Java articles.   
  
<http://www.devx.com/java> - Java articles hosted at DevX.   
  
<http://www.sys-con.com/java> - The Java Developers Journal online magazine website.