

**CHAPTER-1**  
**INTRODUCTION**

# **1.INTRODUCTION**

## **1.1 PURPOSE OF THE SYSTEM:**

Faculty takes lot of time to adjust their classes when they are on leave. They have to check manually timetables of all other faculty to know whether they are free or contact each and every faculty and ask them if they are free. Lot of time is wasted in this process. To overcome this problem, we designed a website in which we upload timetables of all faculties and using an algorithm we display the names and phone numbers of free faculties of selected slot. This solves the problem in couple of minutes.

## **1.2 EXISTING SYSTEM:**

A faculty is on leave or willing to take leave on a day. It is his/her responsibility to adjust their timetable. To adjust the timetable a faculty must contact each and every faculty and know whether they are free. It takes lot of time to consult each and every faculty and know their willingness to take the class.

## **1.3 PROPOSED SYSTEM:**

We designed a website to solve the above problem. In the website, faculty registers with his/her username, phone number and other details. He/She logs in to the website and checks for the free faculty available in those particular slots.

## **CHAPTER-2**

### **SYSTEM REQUIREMENTS**

## **2. SYSTEM REQUIREMENTS**

### **2.1 HARDWARE SYSTEM CONFIGURATION:**

Processor	- Pentium –III
Speed	- 1.1 GHz
RAM	- 256 MB (min)
Hard Disk	- 20 GB
Floppy Drive	- 1.44 MB
Key Board	- Standard Windows Keyboard
Mouse	- Two or Three Button Mouse
Monitor	- SVGA

### **2.2 SOFTWARE SYSTEM CONFIGURATION:**

Operating System	: WindowsXP/10
Application Server	: Tomcat7.X/8.X
Front End	: HTML, Jsp
Database	: Oracle, SqlDeveloper

### **2.3 MODULES OF THE SYSTEM:**

The Modules in the project:

1. User
2. Database

### **2.3.1 User:**

The faculty is the user adjusts his/her period by allocating the other free faculties in his/her period when he/she is on leave. The user login to the website and checks for free faculties by entering day, period and year.

### **2.3.2 Database:**

The database retrieves the free faculty for the user at his/her particular period and also his individual timetable. The individual timetables and the details of the faculty are stored in database.

**CHAPTER 3**  
**SYSTEM ANALYSIS**

## **3.SYSTEM ANALYSIS**

### **3.1 FEASIBILITY STUDY**

The feasibility of the project is analysed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the institute. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

3.1.1 Economical Feasibility

3.1.2 Technical Feasibility

3.1.3 Social Feasibility

#### **3.1.1 Economical Feasibility**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

#### **3.1.2 Technical Feasibility**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

#### **3.1.3 Social Feasibility**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him

familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

## **3.2 SOFTWARE ENVIRONMENT**

### **3.2.1 Java Server Pages (JSP)**

Java Server Pages (JSP) is a technology that helps software developers create dynamically generated web pages based on HTML, XML, or other document types. Released in 1999 by Sun Microsystems, JSP is similar to PHP and ASP, but it uses the Java programming language.

To deploy and run JavaServer Pages, a compatible web server with a servlet container, such as Apache Tomcat or Jetty, is required.

Architecturally, JSP may be viewed as a high-level abstraction of Java servlets. JSPs are translated into servlets at runtime, therefore JSP is a Servlet; each JSP servlet is cached and re-used until the original JSP is modified. JSP can be used independently or as the view component of a server-side model–view–controller design, normally with JavaBeans as the model and Java servlets (or a framework such as Apache Struts) as the controller. This is a type of Model 2 architecture. JSP allows Java code and certain pre-defined actions to be interleaved with static web mark-up content, such as HTML, with the resulting page being compiled and executed on the server to deliver a document. The compiled pages, as well as any dependent Java libraries, contain Java bytecode rather than machine code. Like any other Java program, they must be executed within a Java virtual machine (JVM) that interacts with the server's host operating system to provide an abstract, platform-neutral environment.

JSPs are usually used to deliver HTML and XML documents, but through the use of Output Stream, they can deliver other types of data as well.

The Web container creates JSP implicit objects like request, response, session, application, config, page, page Context, out and exception. JSP Engine creates these objects during translation phase. JSP pages use several delimiters for scripting functions. The most basic is `<% ... %>`, which encloses a JSP scriptlet. A scriptlet is a fragment of Java code that is run when the user requests the page. Other common delimiters include `<%= ... %>` for expressions, where the scriptlet and delimiters are replaced with the result of evaluating the expression, and directives, denoted with `<%@ ... %>`.



Java code is not required to be complete or self-contained within a single scriptlet block. It can straddle markup content, provided that the page as a whole is syntactically correct. For example, any Java if/for/while blocks opened in one scriptlet must be correctly closed in a later scriptlet for the page to successfully compile.

Content that falls inside a split block of Java code (spanning multiple scriptlets) is subject to that code. Content inside an if block will only appear in the output when the if condition evaluates to true. Likewise, content inside a loop construct may appear multiple times in the output, depending upon how many times the loop body runs.

### 3.2.2 CSS

Cascading Style Sheets (CSS) is a simple mechanism for adding style (e.g., fonts, colors, spacing) to Web documents. These pages contain information on how to learn and use CSS and on available software. They also contain news from the CSS working group. HTML was NEVER intended to contain tags for formatting a web page!

HTML was created to **describe the content** of a web page, like:

```
<h1>This is a heading</h1>
```

```
<p>This is a paragraph.</p>
```

When tags like <font>, and color attributes were added to the HTML 3.2 specification, it started a nightmare for web developers. Development of large websites, where fonts and color information were added to every single page, became a long and expensive process.

To solve this problem, the World Wide Web Consortium (W3C) created CSS.

CSS removed the style formatting from the HTML page!

There are three ways of inserting a style sheet:

- External style sheet
- Internal style sheet
- Inline style

#### 1.External Style Sheet

With an external style sheet, you can change the look of an entire website by changing just one file!

Each page must include a reference to the external style sheet file inside the <link> element.

The <link> element goes inside the <head> section.

An external style sheet can be written in any text editor. The file should not contain any html tags. The style sheet file must be saved with a .css extension.

## **2.Internal Style Sheet**

An internal style sheet may be used if one single page has a unique style.

Internal styles are defined within the <style> element, inside the <head> section of an HTML page.

## **3.Inline Styles**

An inline style may be used to apply a unique style for a single element.

To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.

## **Cascading Order**

All the styles in a page will "cascade" into a new "virtual" style sheet by the following rules, where number one has the highest priority:

1. Inline style (inside an HTML element)
2. External and internal style sheets (in the head section)
3. Browser default

So, an inline style has the highest priority, and will override external and internal styles and browser defaults.

### **3.2.3 Oracle**

An Oracle database is a collection of data treated as a unit. The purpose of a database is to store and retrieve related information. A database server is the key to solving the problems of information management. In general, a server reliably manages a large amount of data in a multiuser environment so that many users can concurrently access the same data. All this is accomplished while delivering high performance. A database server also prevents unauthorized access and provides efficient solutions for failure recovery.

Oracle Database is the first database designed for enterprise grid computing, the most flexible and cost-effective way to manage information and applications. Enterprise grid computing creates

large pools of industry-standard, modular storage and servers. With this architecture, each new system can be rapidly provisioned from the pool of components. There is no need for peak workloads, because capacity can be easily added or reallocated from the resource pools as needed.

The database has logical structures and physical structures. Because the physical and logical structures are separate, the physical storage of data can be managed without affecting the access to logical storage structures.

Each Oracle database has a data dictionary. An Oracle data dictionary is a set of tables and views that are used as a read-only reference about the database. For example, a data dictionary stores information about both the logical and physical structure of the database. A data dictionary also stores the following information:

- The valid users of an Oracle database
- Information about integrity constraints defined for tables in the database
- The amount of space allocated for a schema object and how much of it is in use

A data dictionary is created when a database is created. To accurately reflect the status of the database at all times, the data dictionary is automatically updated by Oracle in response to specific actions, such as when the structure of the database is altered. The database relies on the data dictionary to record, verify, and conduct ongoing work. For example, during database operation, Oracle reads the data dictionary to verify that schema objects exist and that users have proper access to them.

### **3.2.4 JDBC**

In an effort to set an independent database standard API for Java; Sun Microsystems developed Java Database Connectivity, or JDBC. JDBC offers a generic SQL database access mechanism that provides a consistent interface to a variety of RDBMSs. This consistent interface is achieved through the use of “plug-in” database connectivity modules, or drivers. If a database vendor wishes to have JDBC support, he or she must provide the driver for each platform that the database and Java run on.

To gain a wider acceptance of JDBC, Sun based JDBC’s framework on ODBC. As you discovered earlier in this chapter, ODBC has widespread support on a variety of platforms.

Basing JDBC on ODBC will allow vendors to bring JDBC drivers to market much faster than developing a completely new connectivity solution.

JDBC was announced in March of 1996. It was released for a 90 day public review that ended June 8, 1996. Because of user input, the final JDBC v1.0 specification was released soon after. The remainder of this section will cover enough information about JDBC for you to know what it is about and how to use it effectively. This is by no means a complete overview of JDBC. That would fill an entire book.

## **JDBC Goals**

Few software packages are designed without goals in mind. JDBC is one that, because of its many goals, drove the development of the API. These goals, in conjunction with early reviewer feedback, have finalized the JDBC class library into a solid framework for building database applications in Java.

The goals that were set for JDBC are important. They will give you some insight as to why certain classes and functionalities behave the way they do. The eight design goals for JDBC are as follows:

### **1. SQL Level API**

The designers felt that their main goal was to define a SQL interface for Java. Although not the lowest database interface level possible, it is at a low enough level for higher-level tools and APIs to be created. Conversely, it is at a high enough level for application programmers to use it confidently. Attaining this goal allows for future tool vendors to “generate” JDBC code and to hide many of JDBC’s complexities from the end user.

### **2. SQL Conformance**

SQL syntax varies as you move from database vendor to database vendor. In an effort to support a wide variety of vendors, JDBC will allow any query statement to be passed through it to the underlying database driver. This allows the connectivity module to handle non-standard functionality in a manner that is suitable for its users.

### **3. JDBC must be implemental on top of common database interfaces**

The JDBC SQL API must “sit” on top of other common SQL level APIs. This goal allows JDBC to use existing ODBC level drivers by the use of a software interface. This interface would translate JDBC calls to ODBC and vice versa.

#### **4. Provide a Java interface that is consistent with the rest of the Java system**

Because of Java's acceptance in the user community thus far, the designers feel that they should not stray from the current design of the core Java system.

#### **5. Keep it simple**

This goal probably appears in all software design goal listings. JDBC is no exception. Sun felt that the design of JDBC should be very simple, allowing for only one method of completing a task per mechanism. Allowing duplicate functionality only serves to confuse the users of the API.

#### **6. Use strong, static typing wherever possible**

Strong typing allows for more error checking to be done at compile time; also, less error appear at runtime.

#### **7. Keep the common cases simple**

Because more often than not, the usual SQL calls used by the programmer are simple SELECT's, INSERT's, DELETE's and UPDATE's, these queries should be simple to perform with JDBC. However, more complex SQL statements should also be possible.

Finally we decided to proceed the implementation using Java Networking..And for dynamically updating the cache table we go for MS Access database.Java has two things: a programming language and a platform. Java is a high-level programming language that is all of the following:

Simple

Architecture-neutral

Object-oriented

Portable

Distributed

High-performance

Interpreted

Multithreaded

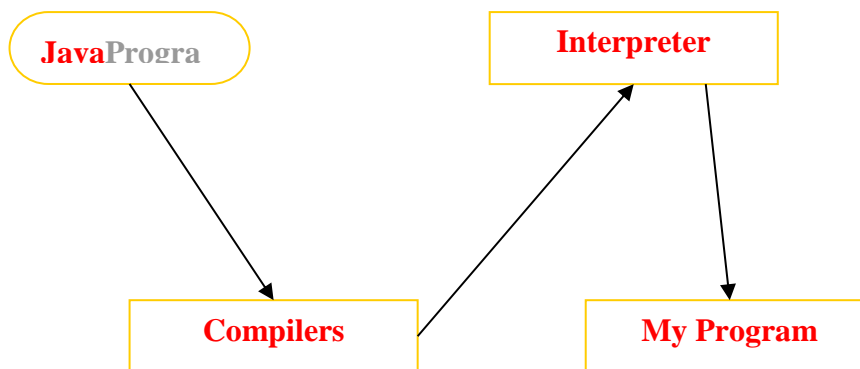
Robust

Dynamic

Java is also unusual in that each Java program is both compiled and interpreted. With a compiler you translate a Java program into an intermediate language called Java byte codes the platform-independent code instruction is passed and run on the computer.

Compilation happens just once; interpretation occurs each time the program is executed.

The figure illustrates how this works.



You can think of Java byte codes as the machine code instructions for the Java Virtual Machine (Java VM). Every Java interpreter, whether it's a Java development tool or a Web browser that can run Java applets, is an implementation of the Java VM. The Java VM can also be implemented in hardware.

Java byte codes help make “write once, run anywhere” possible. You can compile your Java program into byte codes on my platform that has a Java compiler. The byte codes can then be run any implementation of the Java VM. For example, the same Java program can run Windows NT, Solaris, and Macintosh.

### 3.2.5 Tomcat 6.0 web server

Tomcat is an open source web server developed by Apache Group. Apache Tomcat is the servlet container that is used in the official Reference Implementation for the Java Servlet and JavaServer

Pages technologies. The Java Servlet and JavaServer Pages specifications are developed by Sun under the Java Community Process. Web Servers like Apache Tomcat support only web components while an application server supports web components as well as business components (BEAs Weblogic, is one of the popular application server). To develop a web application with jsp/servlet install any web server like JRun, Tomcat etc to run your application.

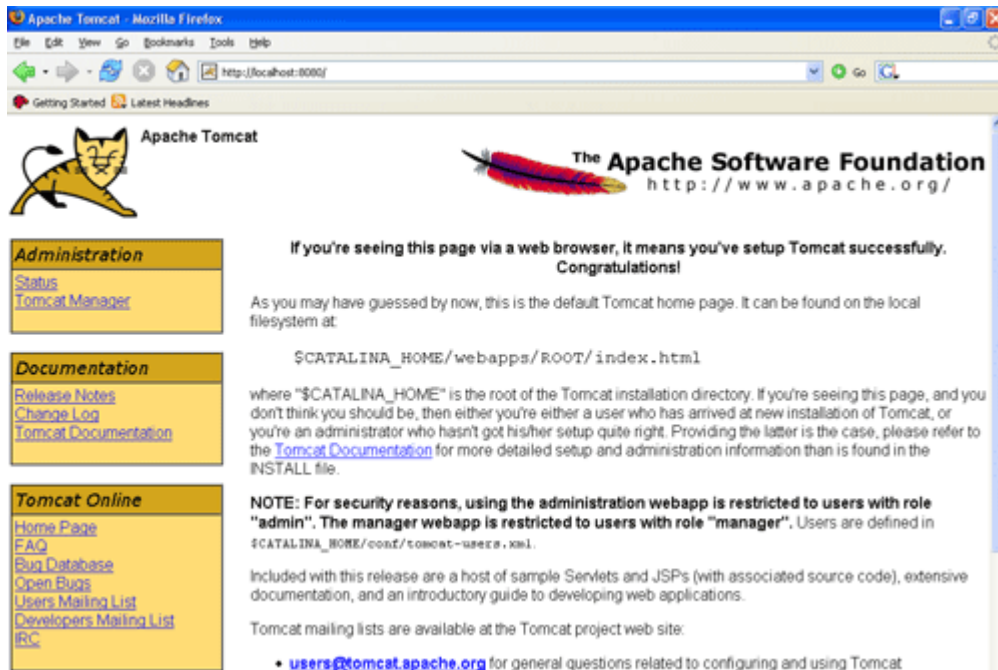


FIG 4.6.1 TOMCAT SERVER PAGE

# **CHAPTER-4**

## **SYSTEM DESIGN**



## **4.1 INPUT DESIGN**

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy.

Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

## **OBJECTIVES**

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.
2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.
3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

## **4.2 OUTPUT DESIGN**

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct

source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

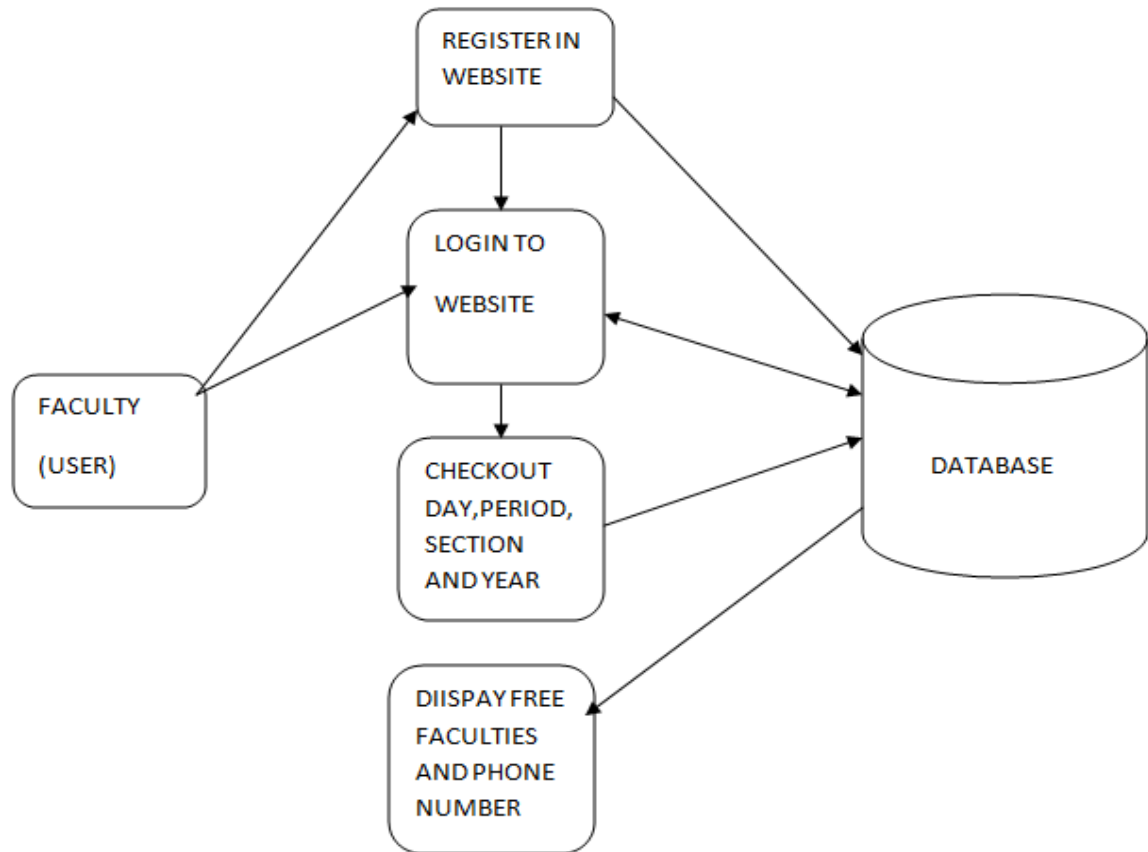
2. Select methods for presenting information.

3. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

### 4.3 SYSTEM ARCHITECTURE:



The above figure shows the architecture of our project classwork adjustment system. It shows how our system is designed and shows the flow among various elements throughout the system in an abstract view. It helps in visualizing the overall structure of the application.

### 4.4 UML CONCEPTS

The Unified Modelling Language (UML) is a standard language for writing software blue prints. The UML is a language for

- Visualizing
- Specifying
- Constructing
- Documenting the artefacts of a software intensive system.

The UML is a language which provides vocabulary and the rules for combining words in that vocabulary for the purpose of communication. A modelling language is a language whose vocabulary and the rules focus on the conceptual and physical representation of a system. Modelling yields an understanding of a system.

#### 4.4.1 Building Blocks of the UML

The vocabulary of the UML encompasses three kinds of building blocks:

- Things
- Relationships
- Diagrams

Things are the abstractions that are first-class citizens in a model; relationships tie these things together; diagrams group interesting collections of things.

##### 1. Things in the UML

There are four kinds of things in the UML:

- Structural things
- Behavioral things
- Grouping things
- Annotational things

**Structural things** are the nouns of UML models. The structural things used in the project design are:

First, a **class** is a description of a set of objects that share the same attributes, operations, relationships and semantics.

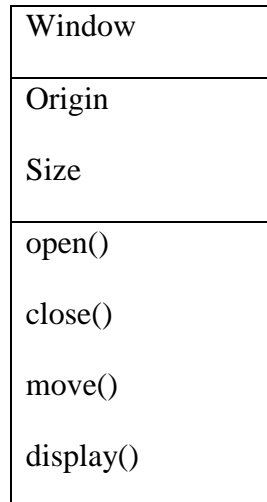


Fig: Classes

Second, a **use case** is a description of set of sequence of actions that a system performs that yields an observable result of value to particular actor.

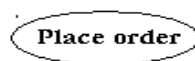


Fig: Use Cases

Third, a node is a physical element that exists at runtime and represents a computational resource, generally having at least some memory and often processing capability.

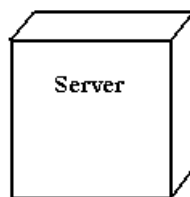


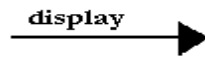
Fig: Nodes

Behavioral things are the dynamic parts of UML models. The behavioral thing used is:

### **Interaction:**

An interaction is a behaviour that comprises a set of messages exchanged among a set of objects within a particular context to accomplish a specific purpose. An interaction involves a number of

other elements, including messages, action sequences (the behaviour invoked by a message, and links (the connection between objects).



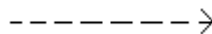
**Fig: Messages**

#### 4.4.2. Relationships in the UML:

There are four kinds of relationships in the UML:

- Dependency
- Association
- Generalization
- Realization

A **dependency** is a semantic relationship between two things in which a change to one thing may affect the semantics of the other thing (the dependent thing).



**Fig: Dependencies**

An **association** is a structural relationship that describes a set links, a link being a connection among objects. Aggregation is a special kind of association, representing a structural relationship between a whole and its parts.



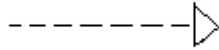
**Fig: Association**

A **generalization** is a specialization/ generalization relationship in which objects of the specialized element (the child) are substitutable for objects of the generalized element (the parent).



**Fig: Generalization**

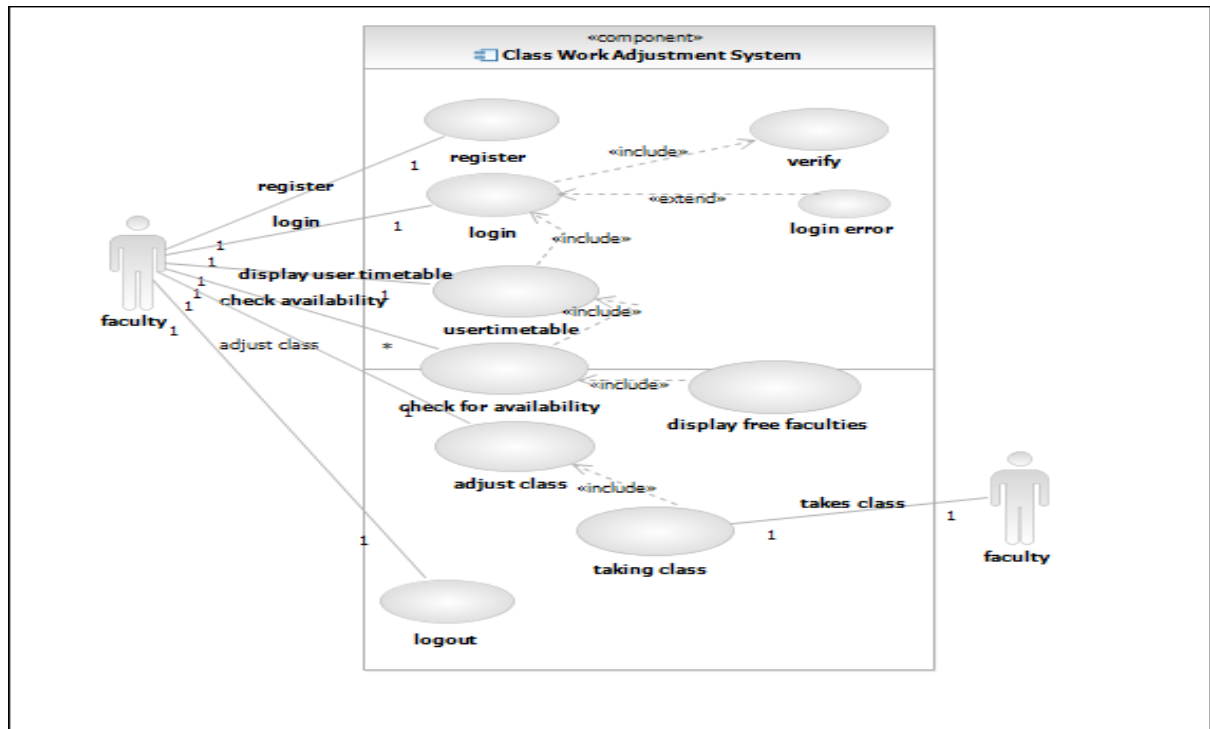
A **realization** is a semantic relationship between classifiers, where in one classifier specifies a contract that another classifier guarantees to carry out.



**Fig: Realization**

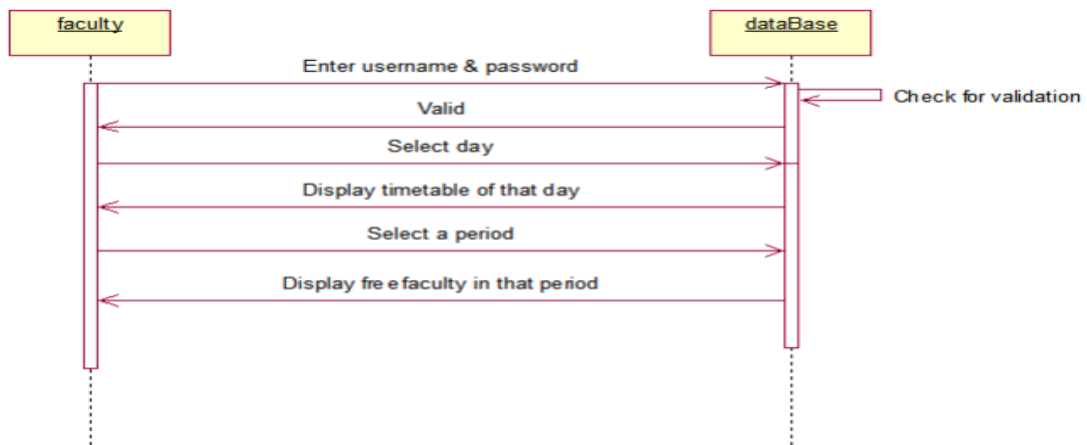
#### 4.4.3 UML DIAGRAMS:

##### 1.USECASE DIAGRAM:



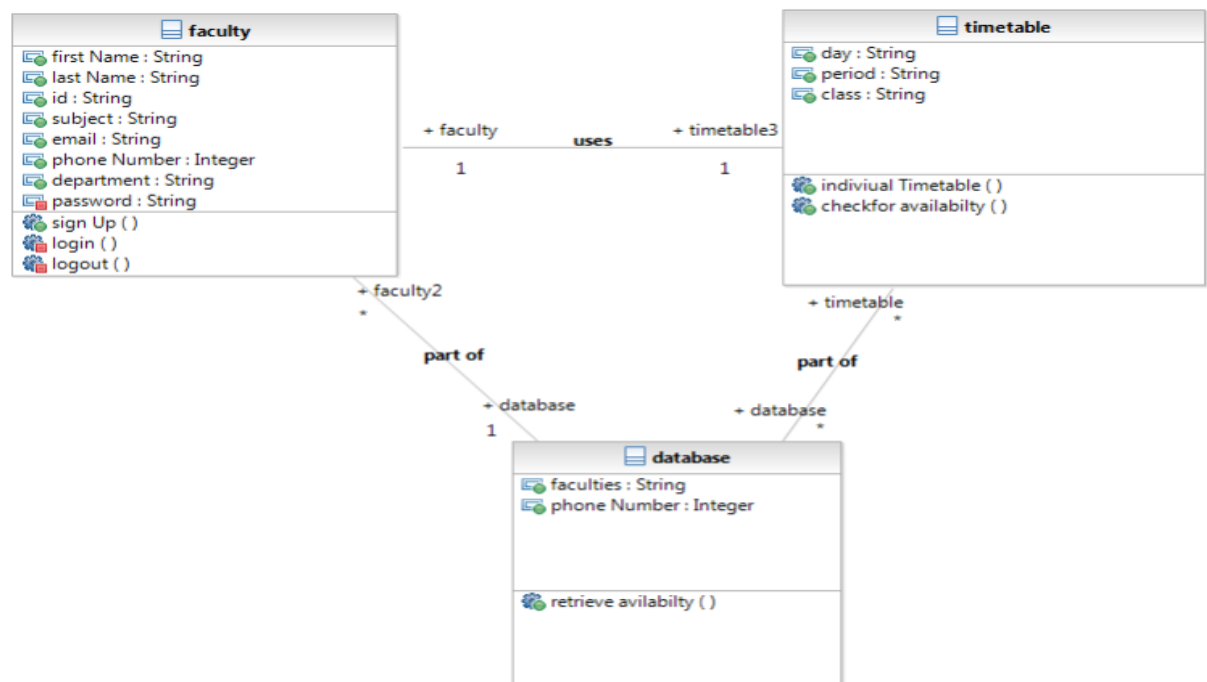
The above diagram is a usecase diagram of our project. It shows the set of actions done by the faculty (User). User registers and logins to the website then the user timetable will be displayed. He/she clicks on the slot which they want to adjust, then it checks for the free faculties available in that particular slot from the database.

## 2.SEQUENCE DIAGRAM:



Above diagram represents sequence diagram for our project. It shows the sequence of operations performed.

## 3.CLASS DIAGRAM:

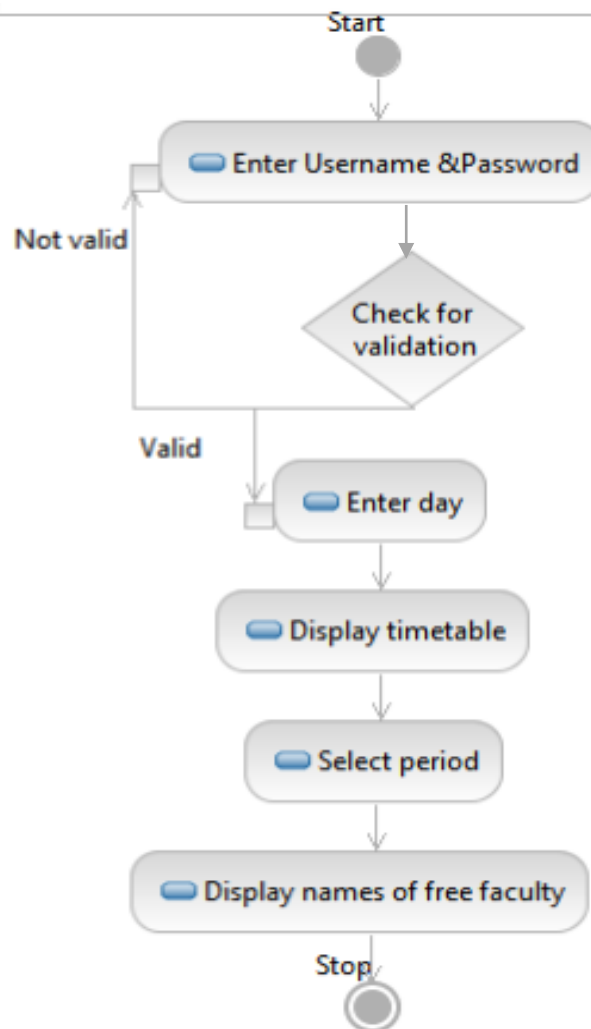


The diagram represents the class diagram for our system. It shows various classes present in our project. Each rectangle box represents a class and the upper portion of it represents class name and middle portion represents attributes of the class and the lower represents the functions performed by that class. **faculty** class contains all the details of the faculty entered at the time of registration,



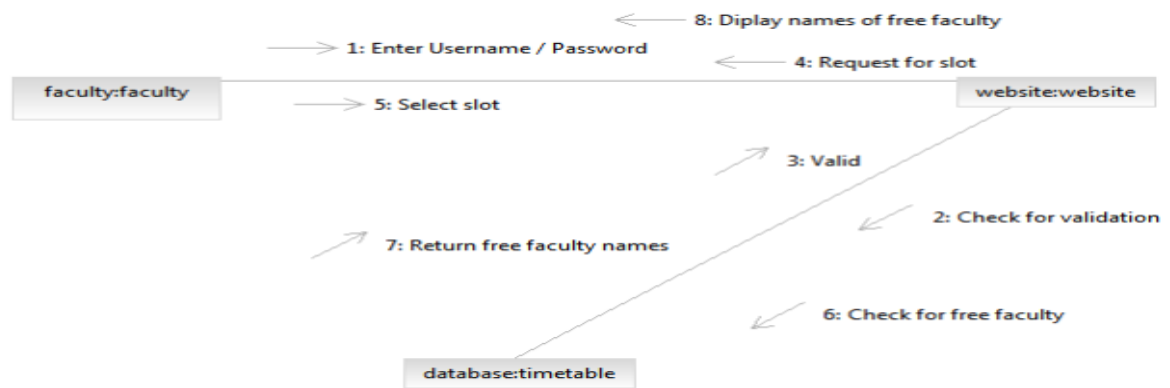
and retrieves the data from it while login. timeTable contains all the sections timetables. These are part of a database.

#### 4.ACTIVITY DIAGRAM:



The above diagram represents activity diagram of the system i.e., it represents the flow of activities in our project. Dot at the start represents starting and dot with circle represents ending and an activity is represented as curve sided rectangle. On seeing it we can understand the flow activities that has to be gone from start to end.

## 5. COLLABORATION DIAGRAM:



Above diagram represents collaboration diagram of our project. It is an illustration of the relationships and interactions among software objects.

# **CHAPTER-5**

## **CODING**

## HOME.CSS

```
body, html {  
    height: 100%;  
}*  
  
{  
    margin:0;  
    padding: 0;  
    font-family: century gothic;  
}  
  
header{  
    background-image:url(snist1.jpg);  
    height:100vh;  
    background-size: cover;  
    background-position: center;  
}  
  
ul  
{  
    float:right;  
    list-style-type: none;  
    margin-top:25px;  
}  
  
ul li{  
    display: inline-block;  
}  
  
ul li a{  
    text-decoration: none;
```

```
    color: 191970;

    padding: 5px 20px;

    border: 1px solid transparent;

    transition: 0.6s ease;

}

ul li a:hover{

    background-color: #fff;

    color: 191970;

}

ul li active.a

{

    background-color: #000;

    color: #000;

}

.logo img

{

    float: left;

    width: 150px;

    height: auto;

}


.main

{

    max-width: 1200px;

    margin: auto;
```

```
}
```

```
.login-page{
```

```
    width:360px;
```

```
    padding:10% 0 0;
```

```
    margin:auto;
```

```
}
```

```
.form{
```

```
    position:relative;
```

```
    z-index: 1;
```

```
    background:rgba(7,40,195,0.8);
```

```
    max-width: 360px;
```

```
    margin:0 auto 100px;
```

```
    padding:45px;
```

```
    text-align: center;
```

```
}
```

```
.select
```

```
{
```

```
    font-family: "Roboto",sans-serif;
```

```
    outline:1;
```

```
    background: #f2f2f2;
```

```
    width:100%;
```

```
    border: 0;
```

```
    margin:0 0 15px;
```

```
    padding: 15px;
```

```
    box-sizing: border-box;
```

```
    font-size: 14px;
```

```
}
```

```
.option
```

```
{
```

```
font-family: "Roboto",sans-serif;
```

```
outline:1;
```

```
background: #f2f2f2;
```

```
width:100%;
```

```
border: 0;
```

```
margin:0 0 15px;
```

```
padding: 15px;
```

```
box-sizing: border-box;
```

```
font-size: 14px;
```

```
}
```

```
.form input
```

```
{
```

```
font-family: "Roboto",sans-serif;
```

```
outline:1;
```

```
background: #f2f2f2;
```

```
width:100%;
```

```
border: 0;
```

```
margin:0 0 15px;
```

```
padding: 15px;
```

```
box-sizing: border-box;
```

```
font-size: 14px;
```

```
}
```

```
.form button{
```

```
font-family: "Roboto",sans-serif;

text-transform: uppercase;

outline: 0;

background: #4CAF50;

width: 100%;

border:0;

padding: 15px;

color:#ffffff;

cursor: pointer;

}

.form button:hover,.form button:active{

    background: #43A047;

}
```

```
.form .message{

    margin:15px 0 0;

    color: aliceblue;

    font-size: 12px;

}
```

```
.form .message a{

    color:#4CAF50;

    text-decoration: none;

}
```

HOME PAGE

<!DOCTYPE html>



```
<html lang="en">

<head>

<title>Page Title</title>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

* {

    box-sizing: border-box;

}

/* Style the body */

body {

    font-family: Arial, Helvetica, sans-serif;

    margin: 0;

background-color: #ddd;

}

/* Header/logo Title */

.header {

    padding: 80px;

    text-align: center;

    background-image:url(snist1.jpg);

    color: white;

}

/* Increase the font size of the heading */
```

```
.header h1 {  
    font-size: 40px;  
}
```

```
/* Style the top navigation bar */
```

```
.navbar {  
    overflow: hidden;  
    background-color: #333;  
}
```

```
/* Style the navigation bar links */
```

```
.navbar a {  
    float: left;  
    display: block;  
    color: white;  
    text-align: center;  
    padding: 14px 20px;  
    text-decoration: none;  
}
```

```
/* Right-aligned link */
```

```
.navbar a.right {  
    float: right;  
}
```

```
/* Change color on hover */
```

```
.navbar a:hover {  
    background-color: #ddd;  
    color: black;
```

```
img {  
    display: block;  
    margin-left: auto;  
    margin-right: auto;  
}
```

```
pre  
{  
font-size: 20px;  
}
```

```
.footer {  
    padding: 20px;  
  
    background: #ddd;  
}
```

```
@media screen and (max-width: 400px) {  
    .navbar a {  
        float: none;  
        width: 100%;  
    }  
}
```

.logo img

```
{  
    float:center;  
    width:150px;  
    height:auto;  
}
```

</style>

</head>

<body>

<div class="header">

<div class="logo">



</div>

<h1>SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY</h1>

<p> CLASS WORK ADJUSTMENT</p>

</div>

<div class="navbar">

<a href="home.jsp">HOME</a>

<a href="register.jsp">REGISTER</a>

<a href="login.jsp">LOGIN</a>

<a href="home.jsp" class="right">LOGOUT</a>

</div>

<center> <h2>Dr. Aruna Varanasi</h2>

<h5>Professor & HOD CSE Dept.</h5>

</center>

<div style="height:200px;"></div>

<p>Dr. Aruna Varanasi is a hardworking and enthusiastic researcher. The Department of Computer Science and Engineering has taken leaps and bounds of progress under her headship. She is a persistent pursuer to get things done by all the faculty and staff in the department and possess very remarkable HR skills. She has got a number of research papers published in leading journals. She has rendered her services in Government sector for six years after which she shifted to teaching. She now has 14 years of teaching experience in addition to her industrial experience. She has great passion in devising solutions to security issues in Information Technology and has been doing lot of research work in Cryptography, Image cryptography, Information Security and Bioinformatics. She has 19 International Journal publications to her credit. She has also been a resource person for several other institutions for the benefit of students of those institutions. Dr. Aruna Varanasi is a self motivated and self driven leader and has taken several initiatives for the development of the department.

Dr. Aruna Varanasi has received "Best Teacher in Computer Science and Engineering Department, AP & TS" award by ISTE, during December, 2017.</p>

<div class="footer">

<pre >

Address

Corporate Office

Sreenidhi Institute of Science & Technology

#1-2-288/23/1

Yamnampet, Ghatkesar

Domalguda, Hyderabad - 500

029

Hyderabad - 501 301, Telangana

Telangana, India

info@sreenidhi.edu.in  
0395

Phone: +91 40 2763 1236 / 2764

Fax: +91 40 2764 0394 / 2764 0394

</pre>

</div>

</body>

</html>

REGISTER.JSP

<html>

<head>

<style>

\* {

box-sizing: border-box;

}

/\* Style the body \*/

body {

font-family: Arial, Helvetica, sans-serif;

margin: 0;

```
background-color: #ddd;

}
```

```
/* Header/logo Title */
```

```
.header {

    padding: 80px;

    text-align: center;

    background-image:url(snist1.jpg);

    color: white;

}
```

```
/* Increase the font size of the heading */
```

```
.header h1 {

    font-size: 40px;

}
```

```
/* Style the top navigation bar */
```

```
.navbar {

    overflow: hidden;

    background-color: #333;

}
```

```
/* Style the navigation bar links */
```

```
.navbar a {

    float: left;

    display: block;
```

```
    color: white;

    text-align: center;

    padding: 14px 20px;

    text-decoration: none;
}
```

```
/* Right-aligned link */
```

```
.navbar a.right {

    float: right;

}
```

```
/* Change color on hover */
```

```
.navbar a:hover {

    background-color: #ddd;

    color: black;

}
```

```
/* Column container */
```

```
/* Create two unequal columns that sits next to each other */
```

```
img {

    display: block;

    margin-left: auto;

    margin-right: auto;

}
```



```
.footer {  
    padding: 20px;  
    text-align: center;  
    background: #ddd;  
}
```

/\* Responsive layout - when the screen is less than 400px wide, make the navigation links stack on top of each other instead of next to each other \*/

```
@media screen and (max-width: 400px) {  
    .navbar a {  
        float: none;  
        width: 100%;  
    }  
}
```

```
.logo img  
{  
    float:center;  
    width:150px;  
    height:auto;  
}
```

</style>

<script>

```
function validLogin(){  
if (document.form.fn.value == ""){
```

```
alert ( "Please enter Login Name." );

return false;

}

if (document.form.p.value == ""){

alert ( "Please enter password." );

return false;

}

alert ( "Welcome " + document.form.fn.value);

return true;

}

</script>

<link rel="stylesheet" type="text/css" href="home.css">

</head>

<body>

    <div class="header">

        <div class="logo">

        </div>

        <h1>SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY</h1>

        <p> CLASS WORK ADJUSTMENT</p>

    </div>

    <div class="navbar">

        <a href="home.jsp">HOME</a>

        <a href="register.jsp">REGISTER</a>

        <a href="login.jsp">LOGIN</a>
```

<a href="home.jsp" class="right">LOGOUT</a>

</div>

<div class="login-page">

<div class="form">

<form class="register-form" action="reg.jsp">

<input type="text" placeholder="username" name="fn">

<input type="text" placeholder="phonenumbr" name="ph">

<input type="text" placeholder="branch" name="br">

<input type="text" placeholder="subject1" name="s1">

<input type="text" placeholder="subject2" name="s2">

<input type="text" placeholder="subject3" name="s3">

<input type="password" placeholder="password" name="p">

<input type="submit" value="create">

</form>

</div>

</div>

</header>

<div class="footer">

<h2> Copyright &copy; 2018 SNIST. All Rights Reserved. </h2>

</div>

</body>

</html>

## REGISTERBACKEND.JSP

<% @ page language="java" import="java.sql.\*"%>

<% @ page language="java" import="java.util.\*"%>

<%

String name = request.getParameter("fn");

session.setAttribute("name",name);

String phone =request.getParameter("ph");

session.setAttribute("phone",phone);

String branch = request.getParameter("br");

session.setAttribute("branch",branch);

String s1 = request.getParameter("s1");

session.setAttribute("s1",s1);

String s2 = request.getParameter("s2");session.setAttribute("s2",s2);

String s3 = request.getParameter("s3");session.setAttribute("s3",s3);

```
String password = request.getParameter("p");
```

```
try {
```

```
    Class.forName("oracle.jdbc.driver.OracleDriver");
```

```
    Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE",  
"system", "system");
```

```
    PreparedStatement ps = con.prepareStatement("insert into register values(?,?,?,?,?,?,?)");
```

```
    ps.setString(1, name);
```

```
    ps.setString(2, phone);
```

```
    ps.setString(3, branch);
```

```
    ps.setString(4,s1);
```

```
    ps.setString(5,s2);
```

```
    ps.setString(6,s3);
```

```
ps.setString(7,password);
```

```
ps.executeUpdate();
```

```
response.sendRedirect("timetable.jsp");
```

```
} catch (Exception e) {
```

```
    out.println(e);
```

```
}
```

#### TIMETABLE.JSP

```
<% @ page language="java" import="java.lang.*" %>
```

```
<% @ page language="java" import="java.sql.*" %>
```

```
<% @ page language="java" import="java.util.*" %>
```

```
<%
```

```
String ph=(String)session.getAttribute("phone");
```

```
out.println(ph);
```

```
Class.forName("oracle.jdbc.driver.OracleDriver");
```

```
Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE",
"system", "system");
```

```
String sqlSelectRecord="";
```

```
sqlSelectRecord="SELECT                                distinct
s.facultyname,s.day,s.period1,s.period2,s.period3,s.period4,s.period5,s.period6,s.period7,s.period
8,s.period9"+ " FROM indivualtt s ,register f" +" where s.phno="+ph+"";
```

```
Statement s = con.createStatement();
```

```
ResultSet rsSelectRecord = s.executeQuery(sqlSelectRecord);
```

```
%>
```

```
<html>
```

```
<head>
```

```
<title>SubQuery in JDBC Query</title>
```

```
<style>
```

```
* {
```

```
    box-sizing: border-box;
```

```
}
```

```
/* Style the body */
```

```
body {
```

```
    font-family: Arial, Helvetica, sans-serif;
```

```
    margin: 0;
```

```
background-color: #ddd;

}
```

```
/* Header/logo Title */
```

```
.header {

    padding: 80px;

    text-align: center;

    background-image:url(snist1.jpg);

    color: white;

}
```

```
/* Increase the font size of the heading */
```

```
.header h1 {

    font-size: 40px;

}
```

```
/* Style the top navigation bar */
```

```
.navbar {

    overflow: hidden;

    background-color: #333;

}
```

```
/* Style the navigation bar links */
```

```
.navbar a {

    float: left;

    display: block;
```



```
    color: white;

    text-align: center;

    padding: 14px 20px;

    text-decoration: none;
}
```

```
/* Right-aligned link */
```

```
.navbar a.right {

    float: right;

}
```

```
/* Change color on hover */
```

```
.navbar a:hover {

    background-color: #ddd;

    color: black;

}
```

```
/* Column container */
```

```
/* Create two unequal columns that sits next to each other */
```

```
img {

    display: block;

    margin-left: auto;

    margin-right: auto;

}
```

```
.footer {  
    padding: 20px;  
    text-align: center;  
    background: #ddd;  
}
```

/\* Responsive layout - when the screen is less than 400px wide, make the navigation links stack on top of each other instead of next to each other \*/

```
@media screen and (max-width: 400px) {
```

```
    .navbar a {  
        float: none;  
        width: 100%;  
    }  
}
```

```
.logo img  
{  
    float:center;  
    width:150px;  
    height:auto;  
}
```

```
td, th {  
    border: 1px solid #ddd;  
    padding: 8px;
```

```
background-color:#a8a49e;

}
```

```
tr:hover {background-color: #ddd;}
```

```
th {

    padding-top: 12px;

    padding-bottom: 12px;

    text-align: left;

    background-color:#af6b4c;

    color: white;

    border: 1px solid black;

    margin-top:"50% ";

    margin-bottom: 50px;

    margin-right:50px;

    margin-left: 50px;

}
```

```
</style>
```

```
<link rel="stylesheet" type="text/css" href="home.css">
```

```
</head>
```

```
<body>
```

```
<div class="header">
```

```
    <div class="logo">
```



</div>

<h1>SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY</h1>

<p> CLASS WORK ADJUSTMENT</p>

</div>

<div class="navbar">

<a href="home.jsp">HOME</a>

<a href="register.jsp">REGISTER</a>

<a href="login.jsp">LOGIN</a>

<a href="home.jsp" class="right">LOGOUT</a>

</div>

<table class="login-page">

<tr>

<th>facultyname</th>

<th>day</th>

<th>period1</th>

<th>period2</th>

<th>period3</th>

<th>period4</th>

<th>period5</th>

<th>period6</th>

<th>period7</th>

<th>period8</th>

<th>period9</th>

```

</tr>

<%
int cnt=1;
while(rsSelectRecord.next())
{
    out.println(cnt);
%>
<tr>
    <td><%=rsSelectRecord.getString("facultyname")%>&nbsp;</td>
    <td><%=rsSelectRecord.getString("day")%>&nbsp;</td>
    <td><%=rsSelectRecord.getString("period1")%>&nbsp;</td>
    <td><%=rsSelectRecord.getString("period2")%>&nbsp;</td>
    <td><%=rsSelectRecord.getString("period3")%>&nbsp;</td>
    <td><%=rsSelectRecord.getString("period4")%>&nbsp;</td>
    <td><%=rsSelectRecord.getString("period5")%>&nbsp;</td>
    <td><%=rsSelectRecord.getString("period6")%>&nbsp;</td>
    <td><%=rsSelectRecord.getString("period7")%>&nbsp;</td>
    <td><%=rsSelectRecord.getString("period8")%>&nbsp;</td>
    <td><%=rsSelectRecord.getString("period9")%>&nbsp;</td>

</tr>

<%
cnt++; // increment of counter
} // End of while loop
%>
</table>

```

```
</body>
```

```
</html>
```

```
<%
```

```
try{
```

```
    if(rsSelectRecord!=null)
```

```
    {
```

```
        rsSelectRecord.close();
```

```
    }
```

```
    if(con!=null)
```

```
    {
```

```
        con.close();
```

```
    }
```

```
}
```

```
catch(Exception e)
```

```
{
```

```
    e.printStackTrace();
```

```
}
```

```
%>
```

```
%>
```

## LOGIN.JSP

```
<html>

  <head>

    <style>

      * {

        box-sizing: border-box;

      }

      /* Style the body */

      body {

        font-family: Arial, Helvetica, sans-serif;

        margin: 0;

        background-color: #ddd;

      }

      /* Header/logo Title */

      .header {

        padding: 80px;

        text-align: center;

        background-image:url(snist1.jpg);

        color: white;

      }

      /* Increase the font size of the heading */

      .header h1 {
```

```
    font-size: 40px;
}
```

```
/* Style the top navigation bar */
```

```
.navbar {
    overflow: hidden;
    background-color: #333;
}
```

```
/* Style the navigation bar links */
```

```
.navbar a {
    float: left;
    display: block;
    color: white;
    text-align: center;
    padding: 14px 20px;
    text-decoration: none;
}
```

```
/* Right-aligned link */
```

```
.navbar a.right {
    float: right;
}
```

```
/* Change color on hover */
```

```
.navbar a:hover {
```



```
background-color: #ddd;

color: black;

}
```

```
/* Column container */
```

```
/* Create two unequal columns that sits next to each other */
```

```
img {

display: block;

margin-left: auto;

margin-right: auto;

}
```

```
.footer {

padding: 20px;

text-align: center;

background: #ddd;

}
```

```
/* Responsive layout - when the screen is less than 400px wide, make the navigation links stack
on top of each other instead of next to each other */
```

```
@media screen and (max-width: 400px) {

.navbar a {

float: none;
```

```
        width: 100%;

    }

}

.logo img

{

    float:center;

    width:150px;

    height:auto;

}

</style>

    <script>

function validLogin(){

if (document.form.fn.value == ""){

alert ( "Please enter Login Name." );

return false;

}

if (document.form.p.value == ""){

alert ( "Please enter password." );

return false;

}

alert ( "Welcome " + document.form.fn.value);

return true;

}

</script>

<link rel="stylesheet" type="text/css" href="home.css">

    </head>
```

```
<body>

  <div class="header">

    <div class="logo">

    </div>

    <h1>SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY</h1>

    <p> CLASS WORK ADJUSTMENT</p>

  </div>


  <div class="navbar">

    <a href="home.jsp">HOME</a>

    <a href="register.jsp">REGISTER</a>

    <a href="login.jsp">LOGIN</a>

    <a href="home.jsp" class="right">LOGOUT</a>

  </div>


  <div class="login-page">

    <div class="form">

      <form name="form" class="login-form" method="post" onsubmit="return
validLogin();" action="log.jsp">

        <input type="text" placeholder="username" name="fn" value="">

        <input type="number" placeholder="phononumber" name="ph" value="">

        <input type="password" placeholder="password" name="p" value="">

        <input type="submit" value="login">

      </form>

    </div>

  </div>

</div>
```

</form>

</div>

</div>

</header>

<div class="footer">

<h2> Copyright &copy; 2018 SNIST. All Rights Reserved. </h2>

</div>

</body>

</html>

LOGBACKEND.JSP

<%--

Document : log.jsp

Created on : 18 Oct, 2018, 7:11:22 PM

Author : chand

--%>

<% @page contentType="text/html" pageEncoding="UTF-8"%>

<% @ page language="java" import="java.sql.\*"%>

<%

String name = request.getParameter("fn");

```
String phone =request.getParameter("ph");
```

```
session.setAttribute("phone",phone);
```

```
String password = request.getParameter("p");
```

```
try {
```

```
Class.forName("oracle.jdbc.driver.OracleDriver");
```

```
Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE",  
"system", "system");
```

```
Statement st=con.createStatement();
```

```
ResultSet rs=st.executeQuery("select * from register where facultyname='"+name+"'");
```

```
if(rs.next()) {
```

```
    if(rs.getString(7).equals(password))
```

```
    {
```

```
        response.sendRedirect("display.jsp");
```

```
    }
```

```
else
```

```
{
```

```

        out.println("Invalid password try again and phno");

        response.sendRedirect("login.jsp");

    }

}

    } catch (Exception e) {

        out.println(e);

    }

```

```

%>

```

#### DISPLAY.JSP

```

<% @ page language="java" import="java.lang.*" %>

<% @ page language="java" import="java.sql.*" %>

<%

```

```

    String ph=(String)session.getAttribute("phone");

```

```

    out.println(ph);

```

```

    Class.forName("oracle.jdbc.driver.OracleDriver");

```

```

    Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE",
"system", "system");

```

```
String sqlSelectRecord="";

sqlSelectRecord="SELECT                                distinct
s.facultyname,s.day,s.period1,s.period2,s.period3,s.period4,s.period5,s.period6,s.period7,s.period
8,s.period9"+ " FROM individt s ,register f" +" where s.phno="+ph+"";

Statement s = con.createStatement();
```

```
ResultSet rsSelectRecord = s.executeQuery(sqlSelectRecord);
```

```
%>
```

```
<html>
```

```
<head>
```

```
<style>
```

```
* {
```

```
    box-sizing: border-box;
```

```
}
```

```
/* Style the body */
```

```
body {
```

```
    font-family: Arial, Helvetica, sans-serif;
```

```
    margin: 0;
```

```
background-color: #ddd;
```

```
}
```

```
/* Header/logo Title */
```

```
.header {
```

```
padding: 80px;

text-align: center;

background-image:url(snist1.jpg);

color: white;

}
```

```
/* Increase the font size of the heading */
```

```
.header h1 {

    font-size: 40px;

}
```

```
/* Style the top navigation bar */
```

```
.navbar {

    overflow: hidden;

    background-color: #333;

}
```

```
/* Style the navigation bar links */
```

```
.navbar a {

    float: left;

    display: block;

    color: white;

    text-align: center;

    padding: 14px 20px;

    text-decoration: none;

}
```



```
/* Right-aligned link */
```

```
.navbar a.right {  
    float: right;  
}
```

```
/* Change color on hover */
```

```
.navbar a:hover {  
    background-color: #ddd;  
    color: black;  
}
```

```
/* Column container */
```

```
/* Create two unequal columns that sits next to each other */
```

```
img {  
    display: block;  
    margin-left: auto;  
    margin-right: auto;  
}
```

```
.footer {  
    padding: 20px;  
    text-align: center;  
    background: #ddd;  
}
```

/\* Responsive layout - when the screen is less than 400px wide, make the navigation links stack on top of each other instead of next to each other \*/

@media screen and (max-width: 400px) {

  .navbar a {

    float: none;

    width: 100%;

  }

}

.logo img

{

  float:center;

  width:150px;

  height:auto;

}

td, th {

  border: 1px solid #ddd;

  padding: 8px;

background-color:#a8a49e;

}

```
tr:hover {background-color: #ddd;}
```

```
th {
```

```
    padding-top: 12px;
```

```
    padding-bottom: 12px;
```

```
    text-align: left;
```

```
    background-color:#af6b4c;
```

```
    color: white;
```

```
    border: 1px solid black;
```

```
    margin-top:"50%";
```

```
    margin-bottom: 50px;
```

```
    margin-right:50px;
```

```
    margin-left: 50px;
```

```
}
```

```
</style>
```

```
    <link rel="stylesheet" type="text/css" href="home.css">
```

```
</head>
```

```
<body>
```

```
<div class="header">
```

```
    <div class="logo">
```

```
        
```

```
    </div>
```

```
<h1>SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY</h1>
```

<p> CLASS WORK ADJUSTMENT</p>

</div>

<div class="navbar">

<a href="home.jsp">HOME</a>

<a href="register.jsp">REGISTER</a>

<a href="login.jsp">LOGIN</a>

<a href="home.jsp" class="right">LOGOUT</a>

</div>

<table class="login-page">

<tr>

<th>facultyname</th>

<th>day</th>

<th>period1</th>

<th>period2</th>

<th>period3</th>

<th>period4</th>

<th>period5</th>

<th>period6</th>

<th>period7</th>

<th>period8</th>

<th>period9</th>

</tr>

<%

```

int cnt=1;

while(rsSelectRecord.next())

{

%>

<tr>

    <td><%=rsSelectRecord.getString("facultyname")%>&nbsp;</td>

    <td><%=rsSelectRecord.getString("day")%>&nbsp;</td>

    <td><%=rsSelectRecord.getString("period1")%>&nbsp;</td>

    <td><%=rsSelectRecord.getString("period2")%>&nbsp;</td>

    <td><%=rsSelectRecord.getString("period3")%>&nbsp;</td>

    <td><%=rsSelectRecord.getString("period4")%>&nbsp;</td>

    <td><%=rsSelectRecord.getString("period5")%>&nbsp;</td>

    <td><%=rsSelectRecord.getString("period6")%>&nbsp;</td>

    <td><%=rsSelectRecord.getString("period7")%>&nbsp;</td>

    <td><%=rsSelectRecord.getString("period8")%>&nbsp;</td>

    <td><%=rsSelectRecord.getString("period9")%>&nbsp;</td>

</tr>

<%

cnt++; // increment of counter

} // End of while loop

%>

</table>

<div class="login-page">

```

```
<div class="form">

    <form class="register-form" action="faculty.jsp">

        <input type="text" placeholder="enter day" name="day">


        <input type="text" placeholder="enter period" name="period">

        <input type="text" placeholder="enter section" name="section">

        <input type="text" placeholder="enter year" name="year">


        <input type="submit" value="display">


    </form>

</div>

</div>

</div>

</header>

<div class="footer">

    <h4> Copyright &copy; 2018 SNIST. All Rights Reserved. </h4>

</div>


</body>

</html>
```

## **CHAPTER-6**

### **OUTPUT SCREENS**


← → ↻ localhost:8888/classwork/home.jsp 🔍 ☆ 🏠 🌐



**SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY**  
CLASS WORK ADJUSTMENT

HOME REGISTER LOGIN LOGOUT

**Dr. Aruna Varanasi**  
Professor & HOD C SE Dept.




Dr. Aruna Varanasi is a hardworking and enthusiastic researcher. The Department of Computer Science and Engineering has taken leaps and bounds of progress under her headship. She is a persistent pursuer to get things done by all the faculty and staff in the department and possess very remarkable HR skills. She has got a number of research papers published in leading journals. She has rendered her services in Government sector for six years after which she shifted to teaching. She now has 14 years of teaching experience in addition to her industrial experience. She has great passion in devising solutions to security issues in Information Technology and has been doing lot of research work in Cryptography, Image cryptography, Information Security and Bioinformatics. She has 19 International Journal publications to her credit. She has also been a resource person for several other institutions for the benefit of students of those institutions. Dr. Aruna Varanasi is a self motivated and self driven leader and has taken several initiatives for the development of the department. Dr. Aruna Varanasi has received "Best Teacher in Computer Science and Engineering Department, AP & TS" award by ISTE, during December, 2017.

Address  
Sreenidhi Institute of Science & Technology  
Yamnapet, Ghatkesar  
Hyderabad - 501 301, Telangana

Corporate Office  
#1-2-288/23/1  
Domaiguda, Hyderabad - 500 029  
Telangana, India


## REGISTER



**SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY**  
CLASS WORK ADJUSTMENT

HOME REGISTER LOGIN LOGOUT





# SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY

CLASS WORK ADJUSTMENT

HOME REGISTER LOGIN
LOGOUT

subba reddy

9912056672

cse

ALGORITHMS


subject2

subject3

.....

C/18376

## Timetable



# SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY

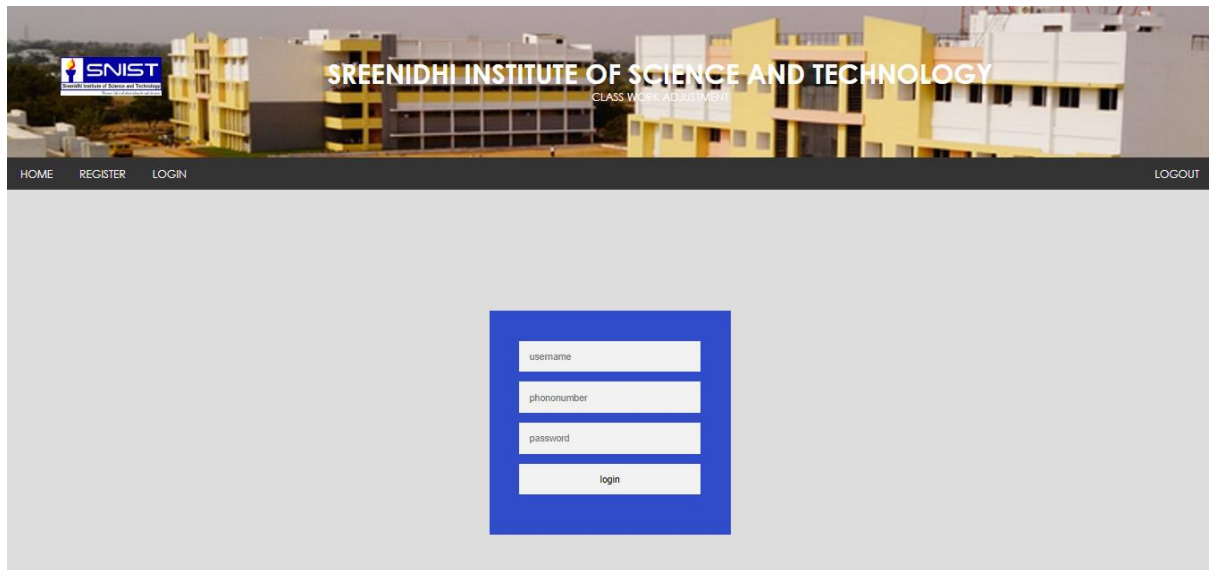
CLASS WORK ADJUSTMENT

HOME REGISTER LOGIN
LOGOUT

123456

facultyname	day	period1	period2	period3	period4	period5	period6	period7	period8	period9
Mr. N. V. Subba Reddy	thursday	II CSE-D WT LAB (2214-A)	II CSE-D WT LAB (2214-A)	II CSE-D WT LAB (2214-A)	II CSE-D WT LAB (2214-A)	null	null	null	II CSE-B OOPJ	null
Mr. N. V. Subba Reddy	wednesday	null	null	null	null	null	null	null	null	II CSE-B OOPJ
Mr. N. V. Subba Reddy	tuesday	II CSE- AOOPJLAB	II CSE- AOOPJLAB	II CSE- AOOPJLAB	II CSE- AOOPJLAB	null	null	null	II CSE-B OOPJ(T)	null
Mr. N. V. Subba Reddy	saturday	null	null	null	null	null	null	null	null	null
Mr. N. V. Subba Reddy	monday	II CSE- BOOPJLAB	II CSE- BOOPJLAB	II CSE- BOOPJLAB	II CSE- BOOPJLAB	null	null	II CSE-B OOPJ	null	null
Mr. N. V. Subba Reddy	friday	null	II CSE-B OOPJ	null	null	null	null	II CSE-B OOPJ	II CSE-B TECHSEM	II CSE-B TECHSEM

## LOGIN



SNIST  
SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY  
CLASS WORK ADJUSTMENT

HOME REGISTER LOGIN LOGOUT

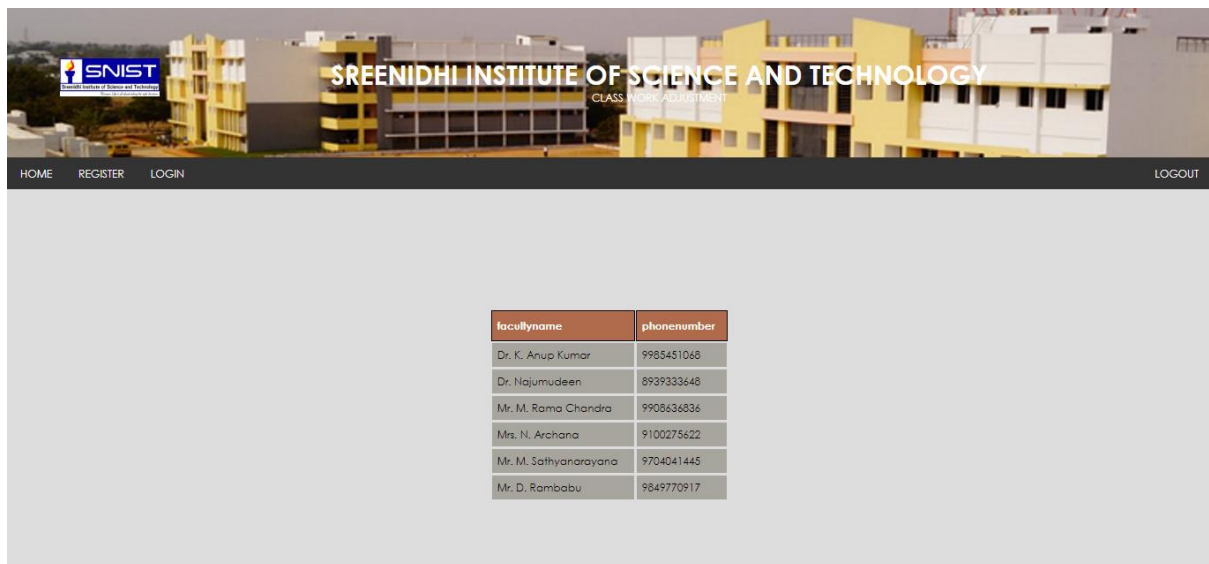
username

phononumber

password

login

## FREEFACULTIES:



SNIST  
SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY  
CLASS WORK ADJUSTMENT

HOME REGISTER LOGIN LOGOUT

facultyname	phononumber
Dr. K. Anup Kumar	9985451068
Dr. Najumdeen	8939333648
Mr. M. Rama Chandra	9908636836
Mrs. N. Archana	9100275622
Mr. M. Sathyanarayana	9704041445
Mr. D. Rambabu	9849770917

**CHAPTER-7**  
**CONCLUSION**

**CONCLUSION:**

Faculty need not apply brute force mechanism to find who are in a particular slot to adjust his/her class. They can simply login to the website and adjust their slots. This approach is very helpful for the faculty who are willing to take leave and adjust their slots to another faculty. This process is more effective than the manual work.