1. INTRODUCTION

This faculty profile management system is a simple solution to view, update and download profile details. The change/update of data is immediately reflected on the webpage and is available for viewing and download. This system provides a simple platform to maintain the faculty profile data upto date with ease.

1.1 OVERALL DESCRIPTION:

As mentioned earlier, this Profile Management System is a lucid solution for all the hassle in maintaining profile data upto date. This system’s access is granted only to the Administrator and Faculty and provides the users a very friendly user interface.

Administrator’s functionalities are to register new faculty, modify faculty profile data, delete profiles and modify the database.

Faculty’s functionalities are to register, fill his/her details in the registration form and re-edit or add data to the form when necessary .

And also this is used to generate excel with specific fields for all the registered faculties in the department.

Other users who are not part of admin and faculty users can only have feasibility of accessing the profile details present in this system. They don't have any feasibility of modifying the data present in the system.

This System is a replacement to the existing Profile Management System, so as to reduce the complication associated with updating the data on a static webpage and to keep the data upto date. This system can be accessed by all the intended users, all the time based on the Network Availability through a desktop computer or personal mobile device.

This system ensures both security and maintainability. This System has scope to make changes easily, because it has been developed by using the feature of Modularity.

2.SYSTEM ANALYSIS

System Analysis is the description of a system into its component pieces to study how thecomponent pieces are study and work.

2.1 Software Requirement Specifications:

Software Requirement Specification is the starting point of software developing activity. As a system grew more complex it became evident that the goal of the entire system cannot be easily comprehended. Hence the needs for the requirements phase are use. The software project is initiated by the client needs, the SRS is the means of translating the ideas of the minds ofclients (he input) into a formal document. The purpose of the software requirement specification is to reduce the communication gap between the clients and developers. Software Requirement Specification is the medium through which the client and user needs are accurately specified. It forms the basis of software development. A good SRS should satisfy all the parties of involved in the system.

2.1.1 Purpose:

Faculty profile system's main purpose is to maintain the details online in order to avoid time delay and loss of the records of the teaching staff .

2.1.2 Scope:

The scope of this system is to make management of faculty data easier and available to the students or other faculty members . Internet Connection is necessary to access this System.

2.1.3 Objective:

The main objective of the system is to provide easy way for accessing and modifying the faculty data with easy user interface which enables them to do so with much ease , this system also allows one to view the faculty information.

2.1.4 Existing System:

Existing system is static and each time a new faculty joins the web page has to be updated making it a tedious task to do each time a new faculty joins , this system doesn't allow one to download the data or view data categorically.

2.1.5Proposed System:

The Proposed System is a replacement for the existing ols system, which updates the page dynamically each time a new faculty is being recruited , this system also allows faculty to add/modify their data.

2.1.6 Functional Requirements:

- Administrator or Faculty has the ability to Login.
- Faculty can register themselves by using their EMAIL ID.
- Admin can Import faculty Details.
- Admin has the ability to update the Database.
- Either Admin or Faculty can view faculty Details.
- Admin and Faculty have the ability to Change Passwords.

2.1.7 Non Functional Requirements:

- User Interface should be compatible to load HTML, CSS, Javascript pages in Front End.
- Software Interface should support the OS and Database of the User.
- Communication Interface can support all Web Browsers except Internet Explorer.
- Availability: The application is available to all the intended users, all the time based on the Network Availability.
- Maintainability: Issues that have been solved can be deleted from the database so as to maintain less complexity.
- Implementation: This System can be easily implemented and has scope for making future changes easily, since the system is developed by using the feature of Modularity.

2.1.8 Software Requirements:

- HTML.
- CSS.
- JAVASCRIPT.
- MY SQL.
- Any browser except internet explorer.

2.1.9 Hardware Requirements:

• Desktop Computers and Personal Mobile Devices
• Keyboard.
• Mouse.
• Minimum 4GB RAM.
• Pentium Processor and above.
• Minimum 256GB Hard Disk.

3.SYSTEM DESIGN

Object Oriented Design is concerned with developing an object oriented model of a software system to implement the identified requirements. It is the process of defining the components, interfaces, objects, classes, Attributes and operations that will satisfy the requirements.

The designer's goal is how the outputs to be produced and in what format samples of output are also presented. The processing phases are handled through the program construction and testing.

The importance of software design can be stated in a single word "QUALITY". Design provides us with representations of software that can be accessed for quality. Design is the only way that can be able to accurately translate a customer's requirements into a finished software product or system without design risk.

Object oriented design can yield the following benefits:

- **MAINTAINABILITY:** Through simplified mapping to the problem domain, which provides for less analysis effort, less complexity in system design, easier verification by the user.
- **REUSABILITY:** Of the design artifacts, which saves time and cost
- **PRODUCTIVITY:** Gains through direct mapping of features of Object Oriented Programming Languages.

3.1.1 PROBLEM ARCHITECTURE:

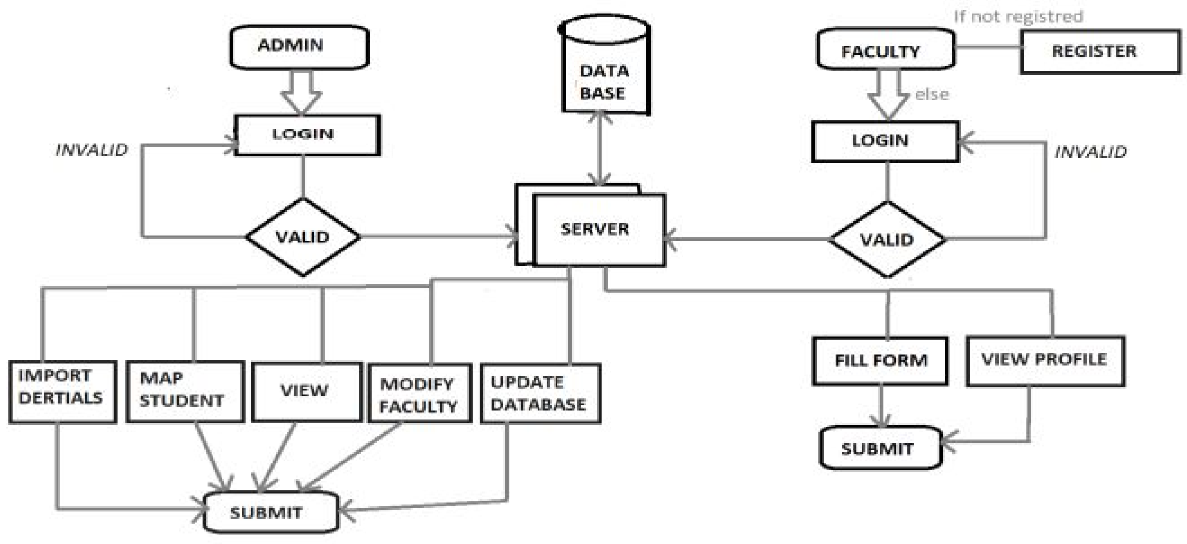


Fig 3.1.1.1 Problem Architecture for Faculty Profile management System

3.2 UML DESIGN:

3.2.1 DATA FLOW DIAGRAM:

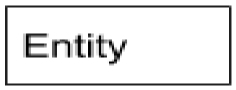
The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of the input data to the system, various processing carried out on these data, and the output data is generated by the system. It maps out the flow of the information for any process or system, how data is processed in terms of inputs and outputs. It uses defined symbols like rectangles, circles and arrows to show data inputs, outputs, storage points and the routes between each destination. They can be used to analyze an existing system or model a new one. A DFD can often visually "say" things that would be hard to explain in words and they work for both technical and non-technical.

There are four components in DFD: 1. External Entity
2. Process
3. Data Flow
4. Data Store

1. External Entity:

It is an outside system that sends or receives data, communicating with the system. They are the sources destinations of the information entering and leaving the system. They might be an outside organization or person, a computer system or a business system. They are known as terminators, sources and sinks or actors. They are typically drawn on the edges of the diagram. These are sources and destinations of the system's input and output.

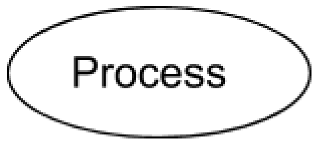
Representation:



2. Process:

It is just like a function that changes the data, producing an output. It might perform computations or sort data based on logic or direct the dataflow based on business rules.

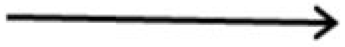
Representation:



3.Data Flow:

A dataflow represents a package of information flowing between two objects in the data-flow diagram. Data flows are used to model the flow of information into the system, out of the system and between the elements within the system.

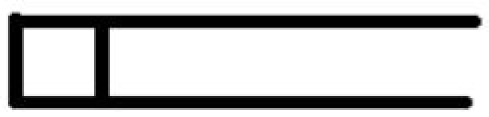
Representation:



4. Data Store:

These are the files or repositories that hold information for later use, such as a database table or a membership form. Each data store receives a simple label.

Representation:



DFD Levels:

A data flow diagram can drive into progressively more detail by using levels. DFD levels are numbered as 0, 1 or 2 and occasionally go to even level 3 or beyond. The necessary level of the detail depends on the scope of the task.

DFD Level 0:

It is also called as context diagram. It's a basic overview of the whole system or process being analyzed or modeled. It's designed to be an at-a-glance view, showing the system as a single high-level process, with its relationship to external entities. It should be easily understood.

DFD Level 1:

It provides a more detailed breakout of pieces of the Context Level Diagram. The main functions carried out by the system, break-down of the high-level process of the context diagram into its sub-process.

DFD Level 2:

This goes one step deeper into parts of level 1. It may require more text to reach the necessary level of detail about the system's functioning.

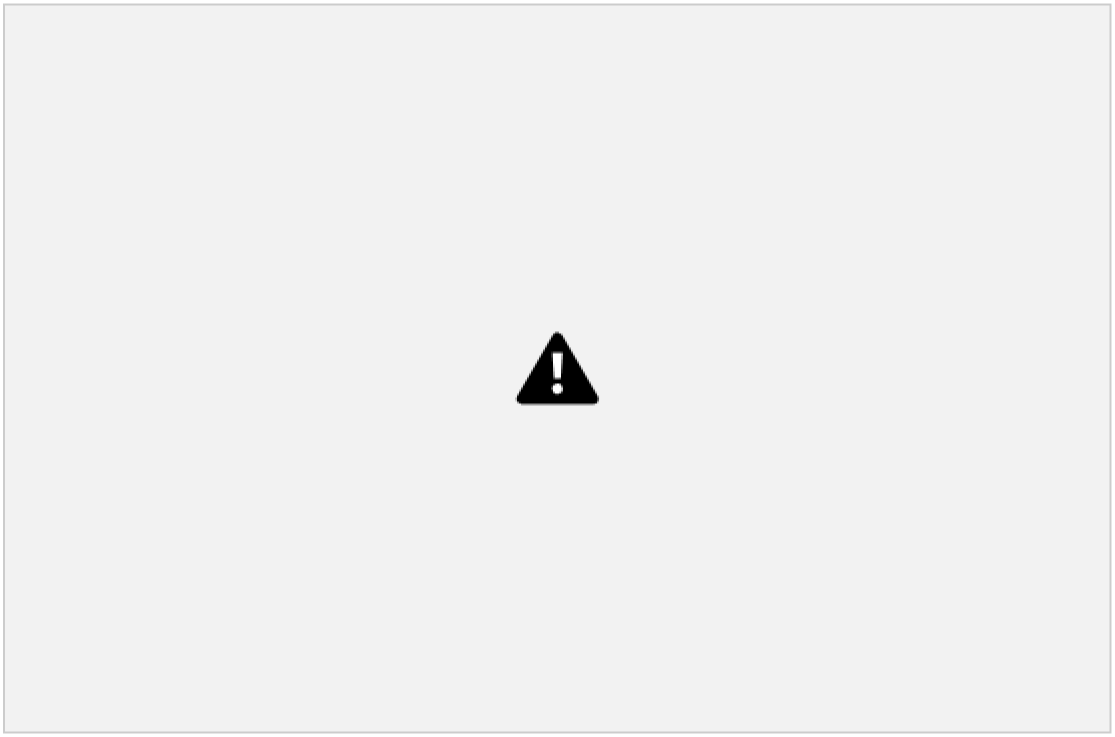


Fig 3.2.1.1 Data Flow Diagram for Proctoring System

3.2.2 Use Case Diagram:

Use Cases are used during Requirement Elicitation and Analysis Phase to represent the functionality of the system. The different roles that the people can fill, when they interact with a system are known as Actors. Use Case describes a function provided by the system that yields a visible result for an Actor.

The identification of Actor and Use Case result in the definition of the boundary of the system, i.e., differentiating the tasks accomplished by the system and the tasks accomplished by its environment. The Actors are outside the boundary of the system, where as the Use Cases are inside the boundary of the system. Use Cases describe the behavior of the system, as seen from Actor's point of view.

Use Case Diagram for Project:

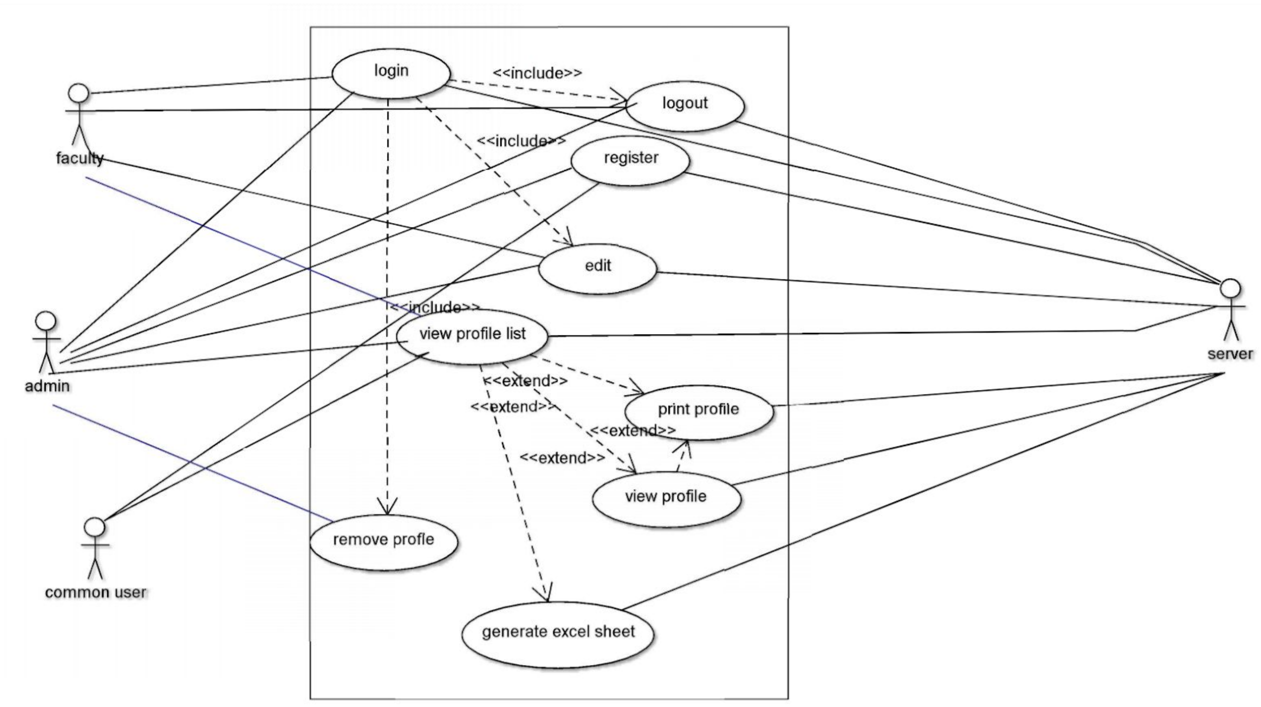


Fig 3.2.2.1 Use Case Diagram for Project

3.2.3 CLASS DIAGRAM:

Class diagram model class structure and contents using design elements such as classes, packages and objects. Class diagram describes 3 perspectives when designing a system-Conceptual, Specification, Implementation. Classes are composed of three things: name, attributes and operations. Class diagrams also display relations such as containment, inheritance, associations etc. The association relationship is most common relationship in a class diagram. The association shows the relationship between instances of classes.

Fig 3.2.3.1 Class Diagram for Faculty profile management System

3.2.4 SEQUENCE DIAGRAM:

Sequence Diagrams: Sequence Diagrams display the time sequence of the objects participating in the interaction. This consists of the vertical dimension (time) and horizontal dimension (different objects).
Object: Object can be viewed as an entity at a particular point in time with a specific value and as a holder of identity that has different values over time.

Actor: An Actor represents a coherent set of roles that users of a system play when interacting with the use cases of the system.
Message: A Message is sending of a signal from one sender object to other receiver objects.

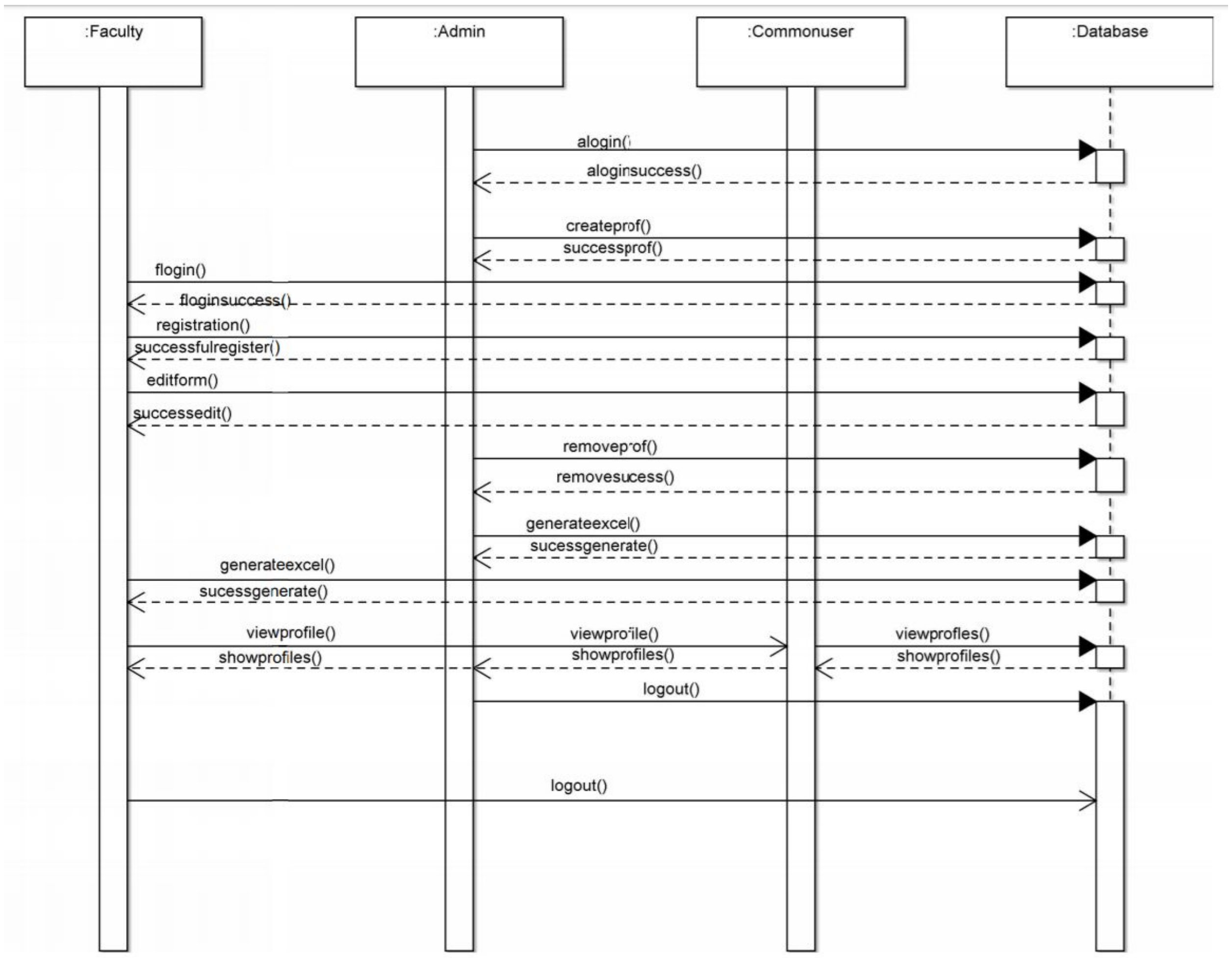


Fig 3.2.4.1 Sequence Diagram

3.2.5 COLLABORATION DIAGRAM:

Collaboration Diagram displays an interaction organized around the objects and their links to one another. Numbers are used to show the sequence of messages. Collaboration diagram is the dynamic behavior of the objects in addition to sequence diagrams.

The transformation form of a sequence diagram into a collaboration diagram is a bi-directional function. The difference between the sequence and collaboration diagrams is that the collaboration diagram emphasizes more on the structure than the sequence of interactions.

Collaboration diagram have two features:

- There is a path to indicate how one object is linked to another.
- There is a sequence number to indicate the time order of messages.

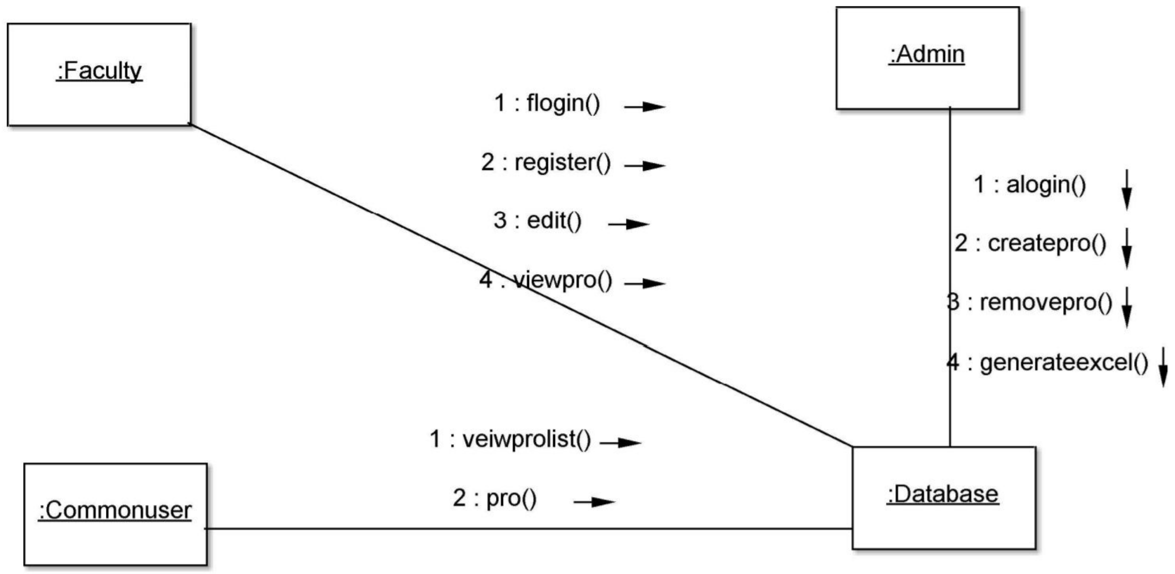


Fig 3.2.5.1 Collaboration Diagram

3.2.6 State Chart Diagram:

A state chart diagram describes a state machine which shows the behavior of classes. It shows the actual changes in state not processes or commands that create those changes and is the dynamic behavior of objects overtime by modeling the lifecycle of objects of each class.

It describes how an object is changing from one state to another state. There are mainly two states in state chart diagram:

- Initial state
- Final state

Some of the components of State Chart Diagram are:
State: It is a condition or situation in lifecycle of an object during which it satisfies same condition or performs some activity or waits for some event.
Transition: It is a relationship between two states indicating that object in first state performs some action and enters into the nextstate.
Event: An event is specification of significant occurrence that has a location in tome and space.

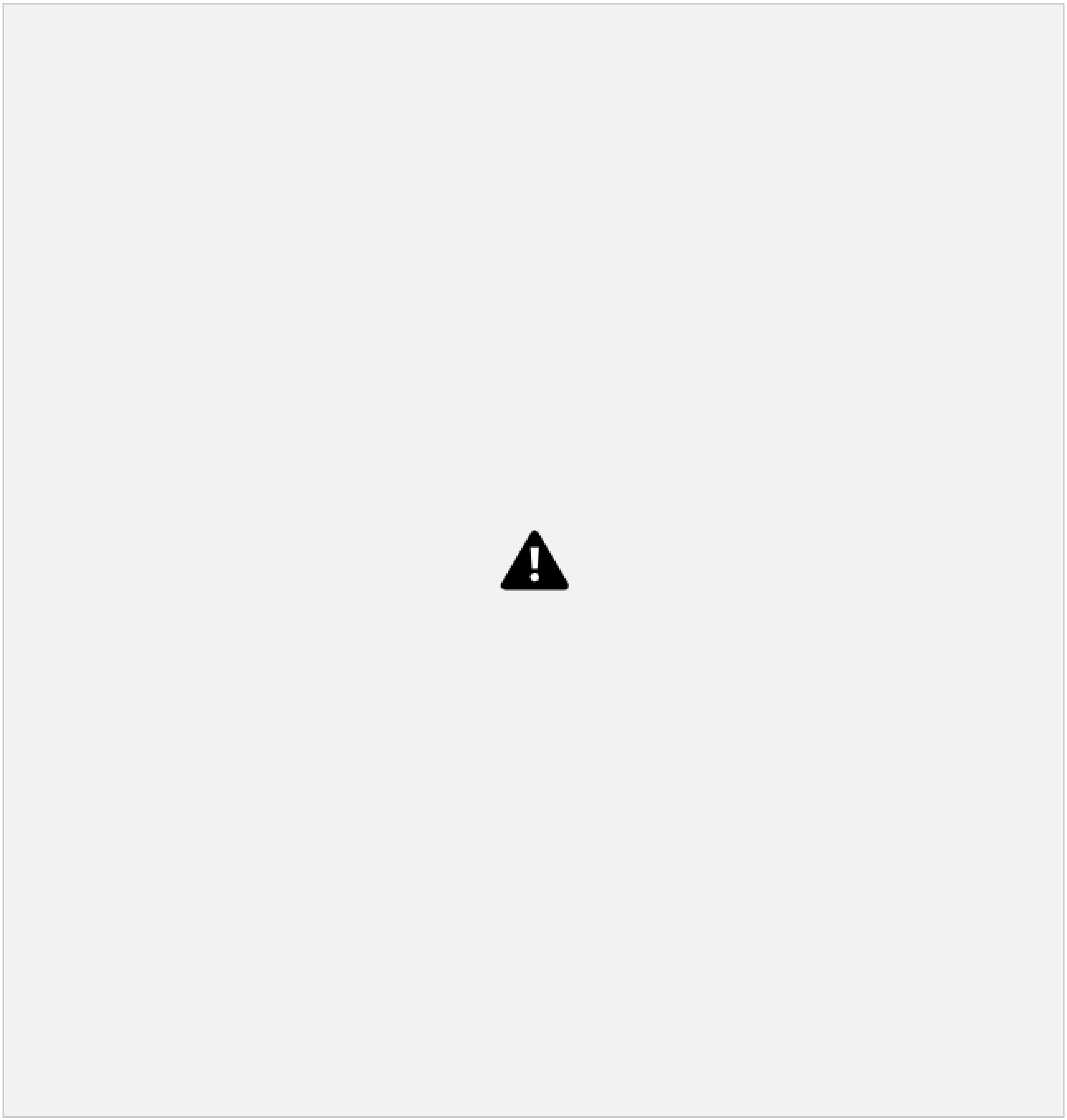


Fig 3.2.6.1 State Chart Diagram for Mapping and Importing Details

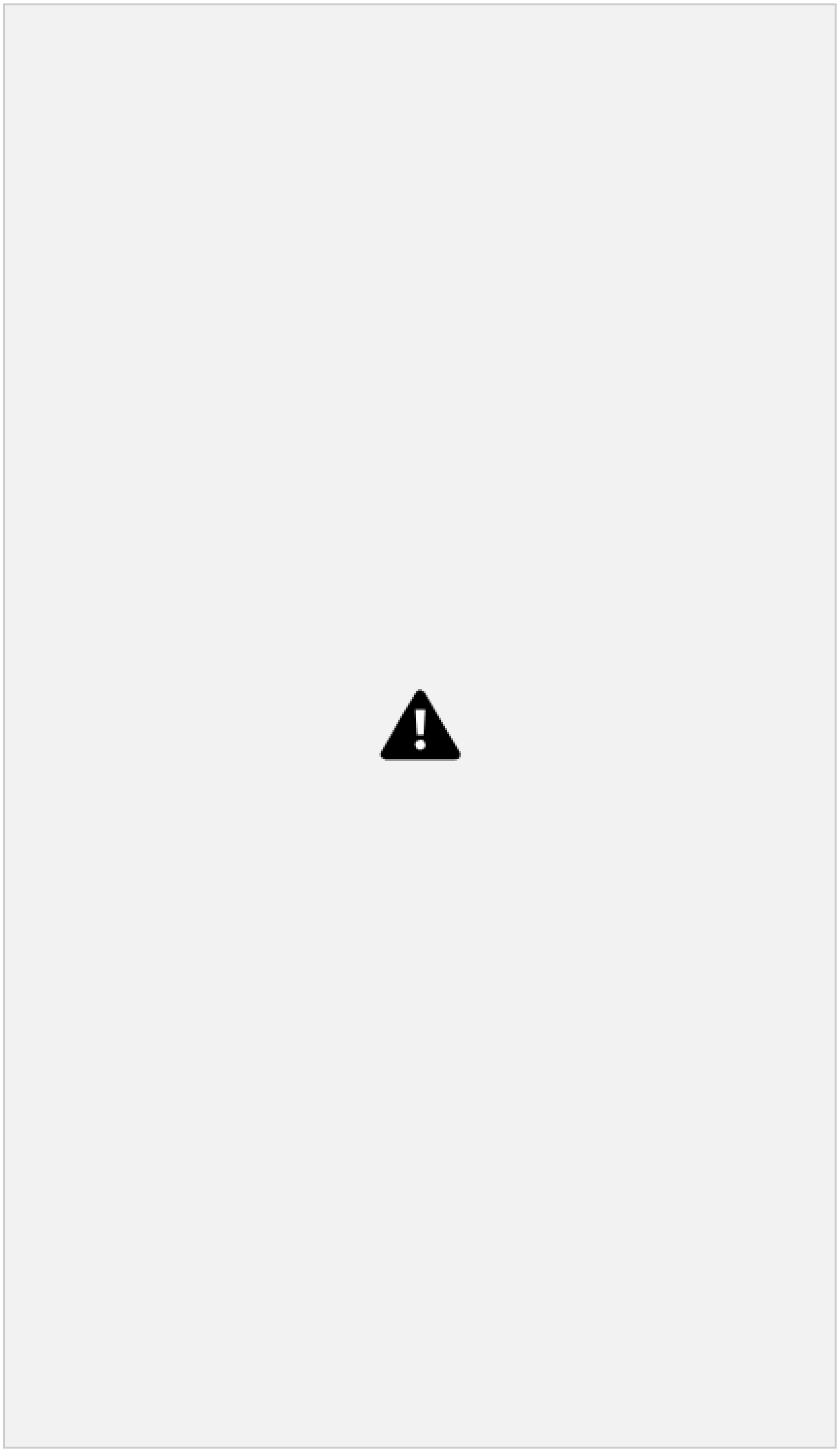


Fig 3.2.6.2 State Chart Diagram for Modifying and Viewing details

Fig 3.2.6.3 State Chart Diagram for Faculty's Functionalities

3.2.7 Activity Diagram:

An activity diagram shows the flow from activity to activity. An activity is a going non-atomic execution within a state machine. An activity results in some action, results in a change of state or return of a value.

Activity Diagram commonly contains:

- Activity states and action states
- Transitions
- Objects, it may contain nodes and constraint

Activity states and action states: An executable atomic computation is called action state, which cannot be decomposed. Activity state is non-atomic, decomposable and takes some duration to execute.

Transition: It is a path from one state to the next state, represented as simple directed line.

Branching: When an alternate path exists, branching arises which is represented by open diamond. It has a incoming transition, two or more outgoing transitions.

Forking and Joining: The synchronization bar when split one flow into two or more flows is called fork. When two or more flows are combined at synchronization bar, the bar is called join.

Swim Lanes: Group work flow is called swim lanes. All groups are portioned by vertical solid lines. Each swim lane specifies locus of activities and has a unique name. Each swim lane is implemented by one or more classes. Transition may occur between objects across swim lanes.

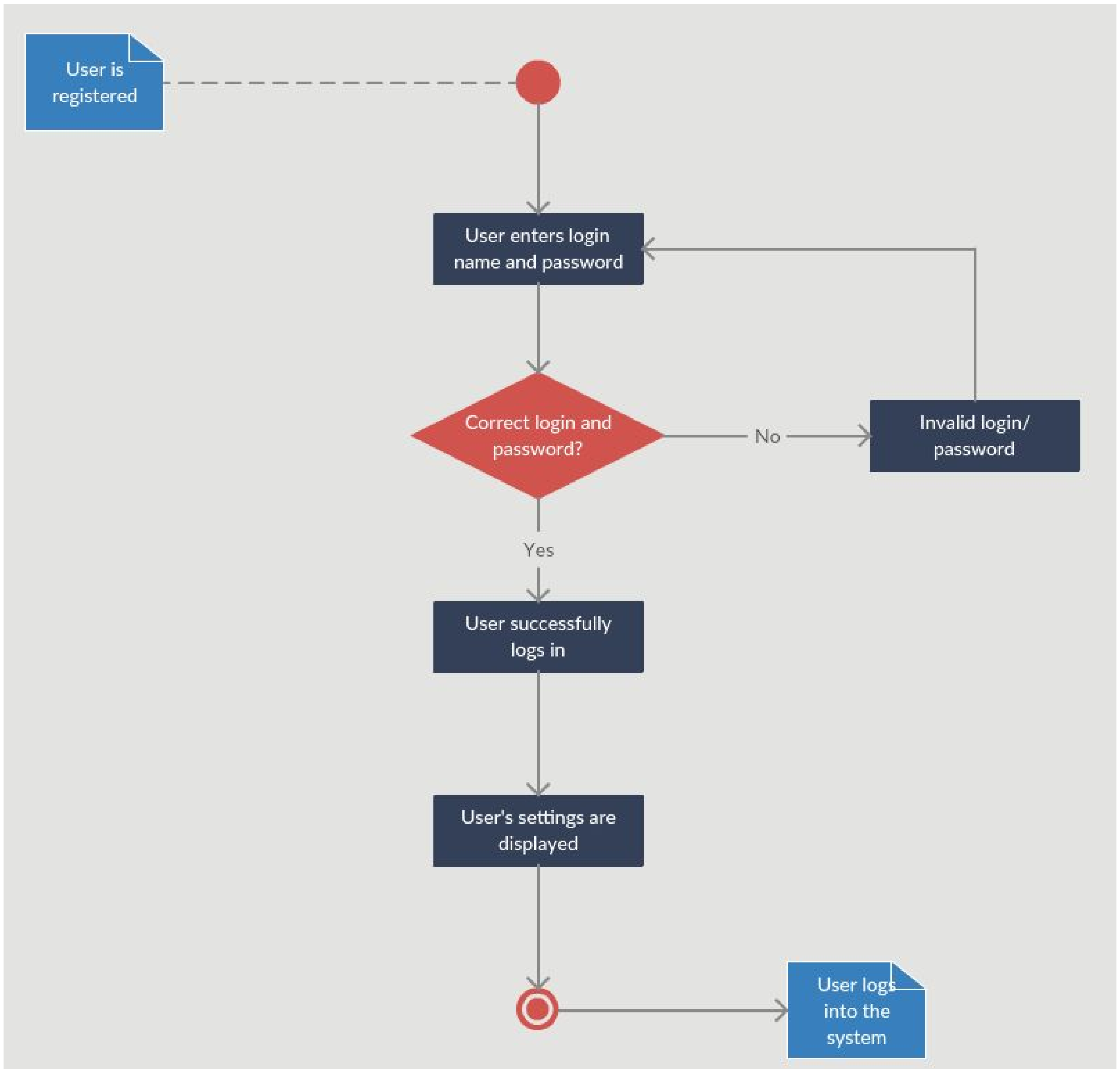

User is registered
User enters login name and password
Correct login and password?
No
Invalid login/ password
Yes
User successfully logs in
User's settings are displayed
User logs into the system

Fig 3.2.7.1 Activity Diagram

3.2.8 COMPONENT DIAGRAM:

Component Diagram describes the organization and wiring of the Physical Components in a system. It can be presented to key project state holders and implementation staff.

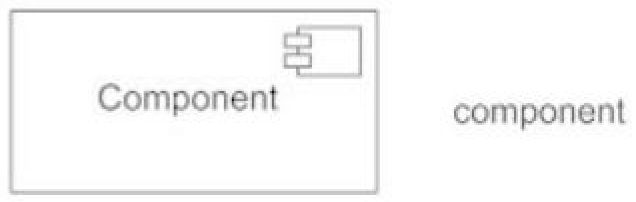
Component Diagram can be used:

- To model the components of a system.
- To model the database schema.
- To model the executability of an application.
- To model the system source code.

Component Diagram Symbols and Notations are as follows:

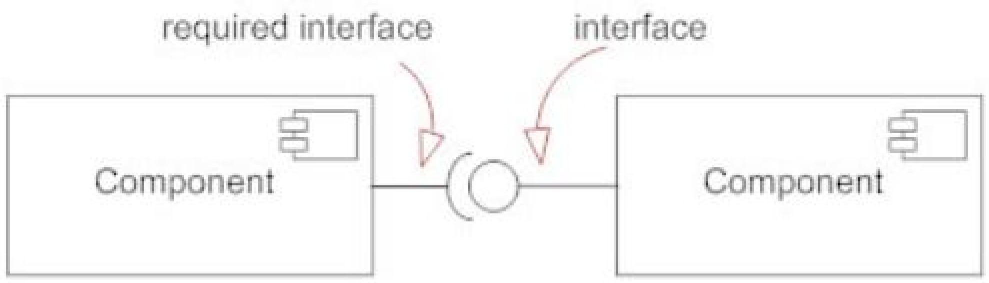
Component:

A component is a logical unit block of the system, a slightly higher abstraction than classes. It is represented as a rectangle with a smaller rectangle in the upper right corner with tabs or the word written above the name of the component to help distinguish it from a class.



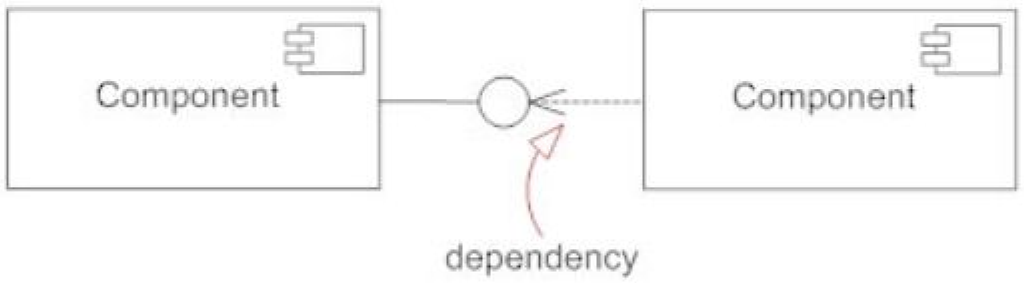
Interface:

An interface (small circle or semi-circle on a stick) describes a group of operations used (required) or created (provided) by components. A full circle represents an interface created or provided by the component. A semi-circle represents a required interface, like a person's input.



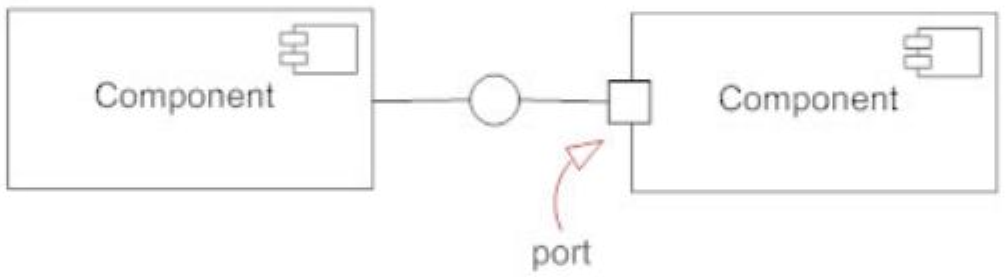
Dependencies:

Draw dependencies among components using dashed arrows.



Port:

Ports are represented using a square along the edge of the system or a component. A port is often used to help expose required and provided interfaces of a component.



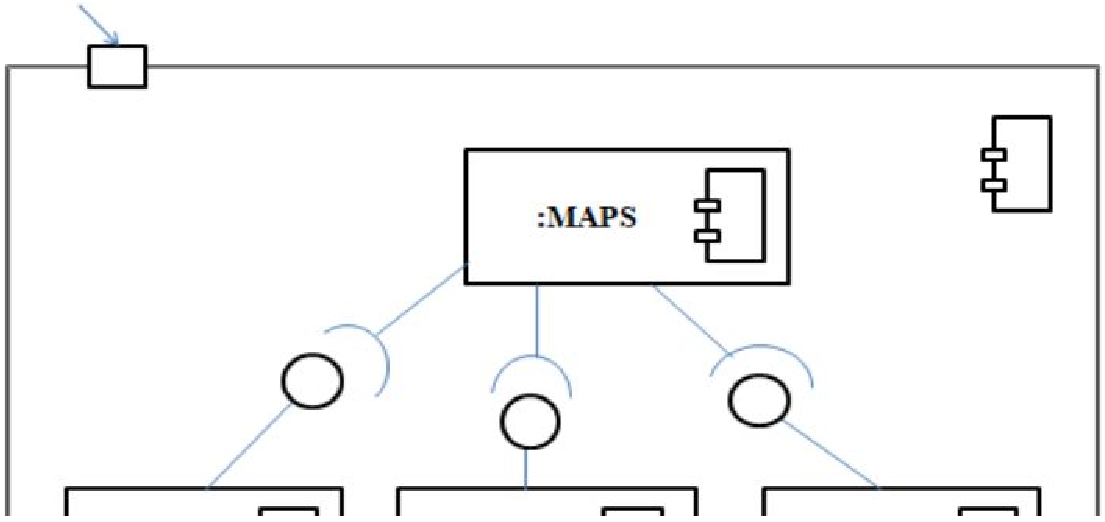


Fig 3.2.8.1 Component Diagram for Mapping Students to Faculty

3.2.9 DEPLOYMENT DIAGRAM:

Deployment Diagram is used to visualize the Topology of the physical component of a system, where the software components are deployed.

A Deployment Diagram consists of Nodes and their relationships. Deployment Diagram is used to show how they are deployed in hardware.
These are useful for System Engineers. An efficient Deployment Diagram is necessary as it controls the performance, maintainability and portability.

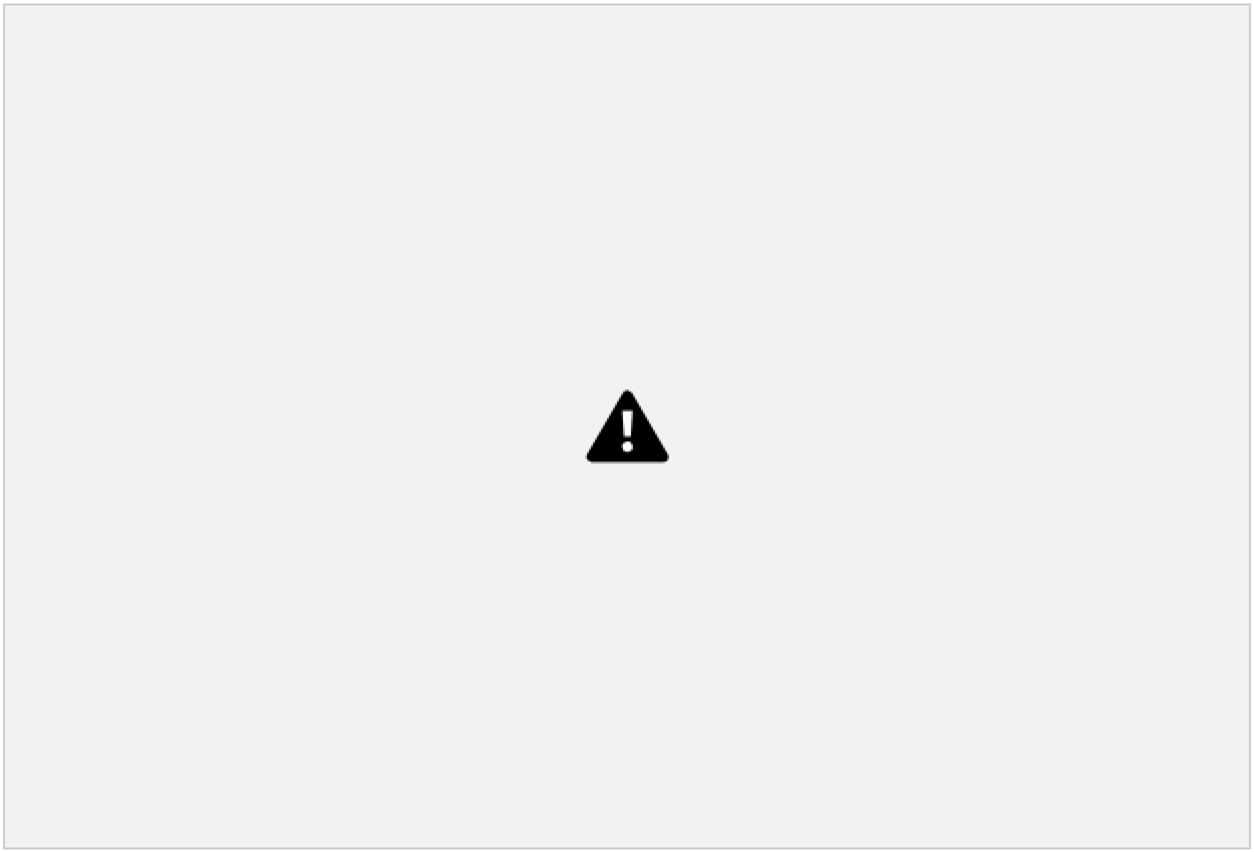


Fig 3.2.9.1 Deployment Diagram for Faculty Profile Management

4. DATABASE DESIGN

4.1 Database:

A database is an organized collection of data, generally stored and accessed electronically from a computer system. Where databases are more complex they are often developed using formal design and modeling techniques.

The database management system(DBMS) is the software that interacts with end users, applications, and the database itself to capture and analyze the data. The DBMS software additionally encompasses the core facilities provided to administer the database. The sum total of the database, the DBMS and the associated applications can be referred to as a "database system". Often the term "database" is also used to loosely refer to any of the DBMS, the database system or an application associated with the database.

4.2 ER Diagram:

An Entity Relationship Diagram (ERD) is a visual representation of different entities within a system and how they relate to each other.ER-modeling is a data modeling technique used in software engineering to produce a conceptual data model of an information system. Diagrams created using this ER-modeling technique are called Entity-Relationship Diagrams, or ER diagrams or ERDs. So you can say that Entity Relationship Diagrams illustrate the logical structure of databases.

ERDs show entities in a database and relationships between tables within that database. It is essential to have ER-Diagrams if you want to create a good database design. The diagrams help focus on how the database actually works.

ER modeling is one of the most cited papers in the computer software field. Currently the ER model serves as the foundation of many system analysis and design methodologies, computer-aided software engineering (CASE) tools, and repository system.

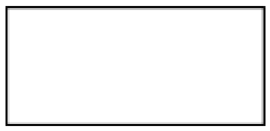
4.2.1 Elements in ER diagram:

Entity relationship diagrams are used in software engineering during the planning stages of the software project. They help to identify different system elements and their relationships with each other. It is often used as the basis for data flow diagrams or DFD's as they are commonly known.

The basic elements in ER-Diagrams:

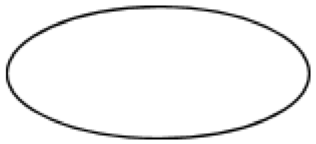
- **Entity:**

Entities are the "things" for which we want to store information. An entity is a person, place, thing or event. Entity can be represented with Rectangles.



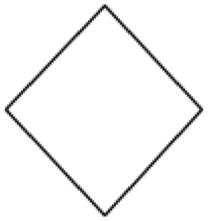
- **Attributes:**

Attributes are the data we want to collect for an entity. An attribute is a property, trait, or characteristic of an entity, relationship, or another attribute. Attributes are represented by Oval shapes.

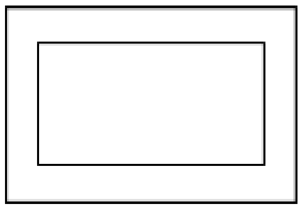


- **Relationships:**

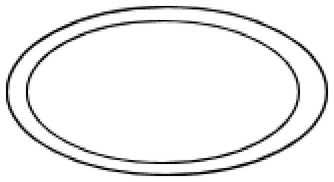
Relationships describe the relations between the entities. ERDs show entities in a database and relationships between tables within that database. It is essential to have ER-Diagrams if you want to create a good database design. The diagrams help focus on how the database actually works.



- **Weak Entity:** A weak entity is an entity that depends on the existence of another entity. In more technical terms it can be defined as an entity that cannot be identified by its own attributes. It uses a foreign key combined with its attribute to form the primary key.



- **Multi-valued Attribute:** If an attribute can have more than one value it is called a multi-valued attribute. It is important to note that this is different from an attribute having its own attributes.



4.2.2 How to Draw ER Diagrams:

Below points show how to go about creating an ER diagram.

1. Identify all the entities in the system. An entity should appear only once in a particular diagram. Create rectangles for all entities and name them properly.
2. Identify relationships between entities. Connect them using a line and add a diamond in the middle describing the relationship.
3. Add attributes for entities. Give meaningful attribute names so they can be understood easily.

4.2.3 ER Diagram for Project:

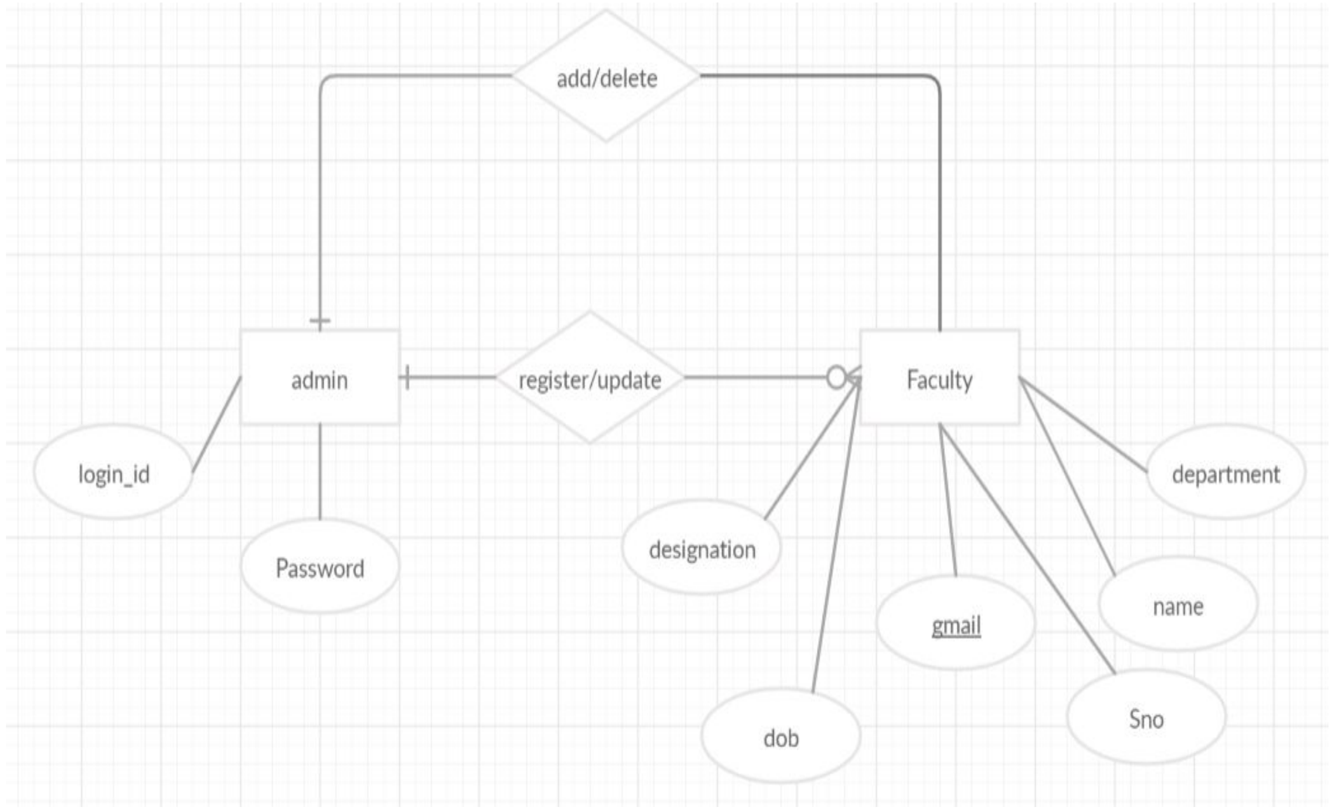


Fig 4.2.3.1 ER Diagram for Faculty Profile Management

4.3 DATABASE TABLES:

- **achievements Table:**

Field	**Type**	**Null**	**Key**	**Default**	**Extra**
gmail	varchar(100)	NO	PRI	NULL	
year_start	varchar(100)	NO		NULL	
date_start	date	NO		NULL	
achievement_name	varchar(100)	NO		NULL	
type	varchar(100)	NO		NULL	
category	varchar(100)	NO		NULL	
staff_name	varchar(100)	NO		NULL	

Table 4.3.1

- **experience Table:**

Field	**Type**	**Null**	**Key**	**Default**	**Extra**
gmail	varchar(50)	No	PRI	NULL	
teaching_exp	int(11)	Yes		0	
research_exp	int(11)	Yes		0	
industry_exp	int(11)	Yes		0	
other_exp	varchar(300)	Yes		NULL	

Table 4.3.2

- **journals Table:**

Field	**Type**	**Null**	**Key**	**Default**	**Extra**
gmail	varchar(255)	NO	PRI	NULL	
internationaljour	int(11)	NO		NULL	
nationaljour	int(11)	NO		NULL	
internationalconf	int(11)	NO		NULL	
nationalconf	int(11)	NO		NULL	
undergraduatestu	int(11)	NO		NULL	
postgraduatestu	int(11)	NO		NULL	
phdstu	int(11)	NO		NULL	
projects	int(11)	NO		NULL	
patents	int(11)	NO		NULL	
noofbooks	int(11)	NO		NULL	
techtransfer	int(11)	NO		NULL	

Table 4.3.3

- **login table:**

Field	**Type**	**Null**	**Key**	**Default**	**Extra**
userid	varchar(20)	NO	PRI	NULL	
pwd	varchar(200)	NO		NULL	

Table 4.3.4

- **membership Table:**

Field	**Type**	**Null**	**Key**	**Default**	**Extra**
gmail	varchar(50)	NO	PRI	NULL	
field_of_membership	varchar(100)	YES		NULL	

Table 4.3.5

- **profile Table:**

Field	**Type**	**Null**	**Key**	**Default**	**Extra**
gmail	varchar(50)	NO	PRI	NULL	
sno	int(11)	NO		NULL	
name	varchar(100)	YES		NULL	
designation	varchar(100)	YES		NULL	
department	varchar(100)	YES		NULL	
dob	date	YES		NULL	
qualification	varchar(300)	YES		NULL	
doj	date	YES		NULL	
phno	varchar(15)	YES		NULL	

Table 4.3.6

- **publications Table:**

Field	**Type**	**Null**	**Key**	**Default**	**Extra**
gmail	varchar(255)	NO	PRI	NULL	
title	varchar(255)	NO		NULL	
conferencename	text	NO		NULL	
category	varchar(20)	NO		NULL	
academic	varchar(10)	NO		NULL	

Table 4.3.7

- **role Table:**

Field	**Type**	**Null**	**Key**	**Default**	**Extra**
gmail	varchar(50)	NO	PRI	NULL	
role_of_faculty	varchar(50)	YES		NULL	
duration	int(11)	YES		NULL	

Table 4.3.8

- **specialization Table:**

Field	**Type**	**Null**	**Key**	**Default**	**Extra**

gmail	varchar(50)	NO	PRI	NULL	
area_of_specialization	varchar(300)	YES		NULL	
ug	varchar(300)	YES		NULL	
pg	varchar(300)	YES		NULL	

Table 4.3.9

- **workshops Table:**

Field	**Type**	**Null**	**Key**	**Default**	**Extra**
gmail	varchar(255)	NO	PRI	NULL	
title	varchar(255)	NO		NULL	
s/w/c	varchar(255)	NO		NULL	
category	varchar(255)	NO		NULL	
place	varchar(20)	NO		NULL	
date	date	NO		NULL	
staff_name	varchar(20)	NO		NULL	
department	varchar(20)	NO		NULL	

Table 4.3.10

- IMPLEMENTATION

Implementation is the stage where the theoretical design is turned into a working system. The most crucial stage in achieving a new successful system and in giving confidence on the system for the users that will work efficiently and effectively. The system will be implemented only after through testing and if its found to work according to the specification.

5.1 Overview of Software Used:

HTML:

Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications. With Cascading Style Sheets (CSS) and JavaScript, it forms a triad of cornerstone technologies for the World Wide Web.

Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items.

HTML elements are delineated by tags, written using angle brackets. Tags such as <img/> and <input/> directly introduce content into the page. Other tags such as <p> surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags, but use them to interpret the content of the page.

HTML can embed programs written in a scripting language such as JavaScript, which affects the behavior and content of web pages. Inclusion of CSS defines the look and layout of content.

CSS:

Stands for "Cascading Style Sheet". Cascading style sheets are used to format the layout of Web pages. They can be used to define text styles, table sizes, and other aspects of Web pages that previously could only be defined in a page's HTML.

CSS helps Web developers create a uniform look across several pages of a Web site. Instead of defining the style of each table and each block of text within a page's HTML, commonly used styles need to be defined only once in a CSS document. Once the style is defined in cascading style sheet, it can be used by any page that references the CSS file. Plus, CSS makes it easy to change styles across several pages at once. For example, a Web developer may want to increase the default text size from 10pt to 12pt for fifty pages of a Web site. If the pages all reference the same style sheet, the text size only needs to be changed on the style sheet and all the pages will show the larger text.

While CSS is great for creating text styles, it is helpful for formatting other aspects of Web page layout as well. For example, CSS can be used to define the cell padding of table cells, the style, thickness, and color of a table's border, and the padding around images or other objects. CSS gives Web developers more exact control over how Web pages will look than HTML does. This is why most Web pages today incorporate cascading style sheets.

Java Script:

JavaScript is a programming language commonly used in web development. It was originally developed by Netscape as a means to add dynamic and interactive elements to websites. While JavaScript is influenced by Java, the syntax is more similar to C and was developed by James Gosling of Sun Microsystems.

JavaScript is a client-side scripting language, which means the source code is processed by the client's web browser rather than on the web server. This means JavaScript functions can run after a webpage has loaded without communicating with the server. For example, a

JavaScript function may check a web form before it is submitted to make sure all the required fields have been filled out. The JavaScript code can produce an error message before any information is actually transmitted to the server.

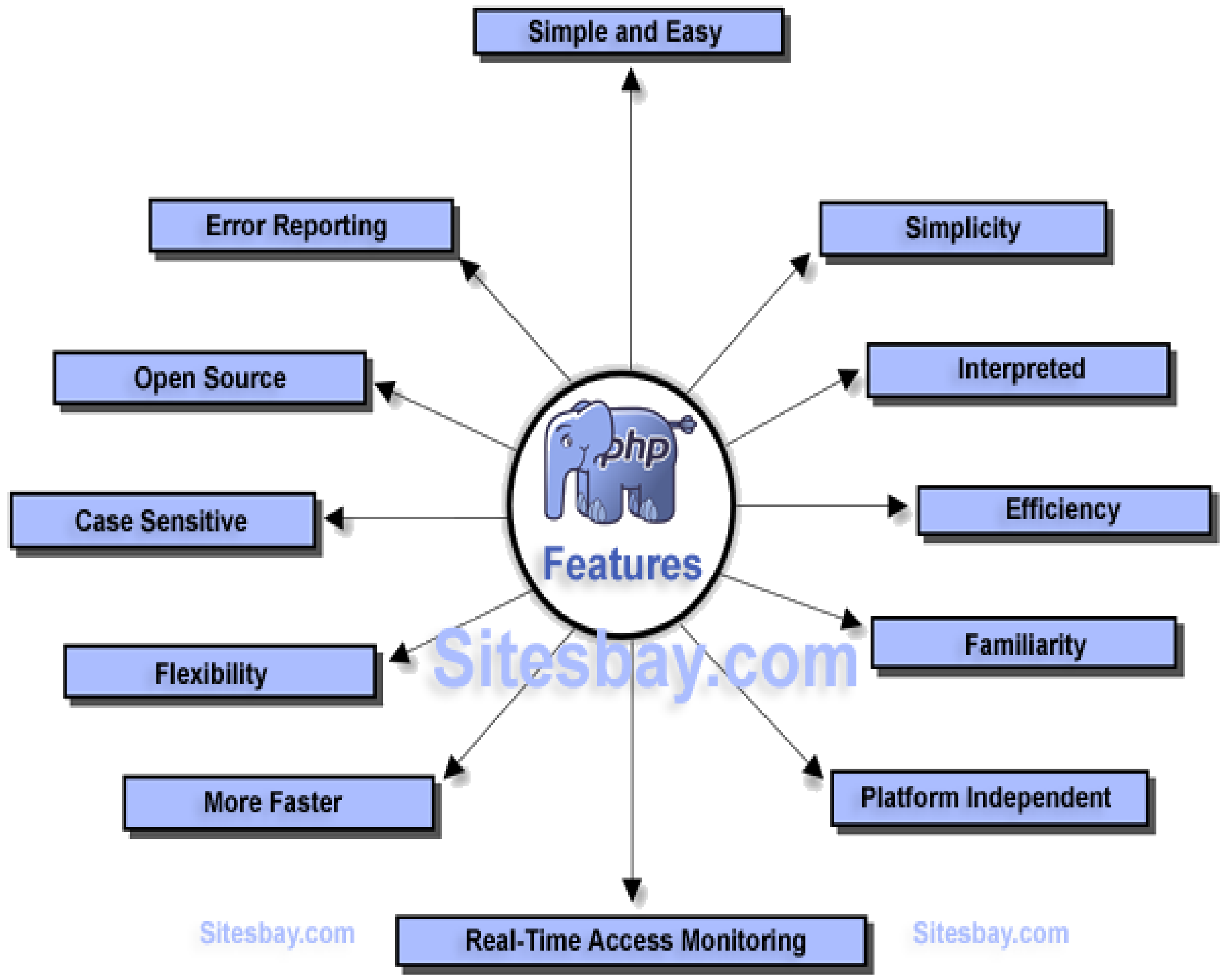
PHP:

PHP is a popular general-purpose scripting language that is especially suited to web development. It was originally created by Rasmus Lerdorf in 1994; the PHP reference implementation is now produced by The PHP Group. PHP originally stood for Personal Home Page, but it now stands for the recursive initialism PHP: Hypertext Preprocessor.

PHP code is usually processed on a web server by a PHP interpreter implemented as a module, a daemon or as a Common Gateway Interface (CGI) executable. On a web server, the result of the interpreted and executed PHP code – which may be any type of data, such as generated HTML or binary image data – would form the whole or part of a HTTP response. Various web template systems, web content management systems, and web frameworks exist which can be employed to orchestrate or facilitate the generation of that response. Additionally, PHP can be used for many programming tasks outside of the web context, such as standalone graphical applications and robotic drone control. Arbitrary PHP code can also be interpreted and executed via command line interface (CLI).

Features of PHP:

The main features of php is; it is open source scripting language so you can free download this and use. PHP is a server site scripting language. It is open source scripting language. It is widely used all over the world. It is faster than other scripting language. Some important features of php are given below.



Simple

It is very simple and easy to use, compare to other scripting language it is very simple and easy, this is widely used all over the world.

Interpreted

It is an interpreted language, i.e. there is no need for compilation.

Faster

It is faster than other scripting language e.g. asp and jsp.

Open Source

Open source means you no need to pay for use php, you can free download and use.

Platform Independent

PHP code will be run on every platform, Linux, Unix, Mac OS X, Windows.

Case Sensitive

PHP is case sensitive scripting language at time of variable declaration. In PHP, all keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are NOT case-sensitive.

Error Reporting

PHP have some predefined error reporting constants to generate a warning or error notice.

Real-Time Access Monitoring

PHP provides access logging by creating the summary of recent accesses for the user.

Loosely Typed Language

PHP supports variable usage without declaring its data type. It will be taken at the time of the execution based on the type of data it has on its value.

MYSQL:

MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. MySQL is developed, marketed and supported by MySQL AB, which is a Swedish company. MySQL is becoming so popular because of many good reasons −

- MySQL is released under an open-source license. So you have nothing to pay to use it.
- MySQL is a very powerful program in its own right. It handles a large subset of the functionality of the most expensive and powerful database packages.

- MySQL uses a standard form of the well-known SQL data language.
- MySQL works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA, etc.
- MySQL works very quickly and works well even with large data sets.
- MySQL is very friendly to PHP, the most appreciated language for web development.
- MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB).
- MySQL is customizable. The open-source GPL license allows programmers to modify the MySQL software to fit their own specific environments.

5.2 Coding:

5.2.1 Front End:

Index.html

```
<html xmlns="http://www.w3.org/1999/xhtml">

<head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title></title>
        <meta name="keywords" content="" />
        <meta name="description" content="" />
        <link            href="http://fonts.googleapis.com/css?family=Didact+Gothic"
rel="stylesheet" />
        <link href="default.css" rel="stylesheet" type="text/css" media="all" />
        <link href="fonts.css" rel="stylesheet" type="text/css" media="all" />

        <!-- [if IE 6]><link href="default_ie6.css" rel="stylesheet" type="text/css"
/><![endif] -->

        <style>
                /* Style The Dropdown Button */
                .dropbtn {
                        background-color: rgba(83, 83, 83, 0);
                        color: white;
                        font-family: 'Didact Gothic', sans-serif;
```

```
    /* padding: 16px; */
    /* font-size: 16px; */
    border: none;
    /* padding: 1em 1.5em; */
    letter-spacing: 1px;
    text-decoration: none;
    text-transform: uppercase;
    font-size: 0.8em;
    color: rgba(255, 255, 255, 0.80);
    cursor: pointer;
}

/* The container <div> - needed to position the dropdown content */
.dropdown {
    position: relative;
    display: inline-block;
    /* align-self: center; */

    border-radius: 25px;
}

/* Dropdown Content (Hidden by Default) */
.dropdown-content {
    text-align: center;
    display: none;
    position: absolute;
    background-color: #5f5f5f;
```

```
        min-width: 160px;
        box-shadow: 0px 8px 16px 0px rgba(0, 0, 0, 0.2);
        border-radius: 25px;
        z-index: 1;
}

/* Links inside the dropdown */
.dropdown-content a {
        color: black;
        padding: 12px 16px;
        text-decoration: none;
        display: block;
}

/* Change color of dropdown links on hover */
.dropdown-content a:hover {
        background-color: #a5a5a5;
        border-radius: 25px;

}

/* Show the dropdown menu on hover */
.dropdown:hover .dropdown-content {
        display: block;
}
```

```
                /* Change the background color of the dropdown button when the
dropdown content is shown */
                .dropdown:hover .dropbtn {
                        background-color: #49494900;
                }
        </style>



</head>


<body>
        <div id="header-wrapper">
                <div id="header" class="container" style="padding: 100px 0px;
width: 95%;">
                        <div id="logo">
                                <h1><a href="#">Faculty Profiles</a></h1>
                        </div>
                        <div id="menu" style="padding-right: 120px;">
                                <ul>
                                        <li class="active"><a href="#" accesskey="1"
title="">Homepage</a></li>
                                        <li><a href="ProfilesList.php" accesskey="2"
title="">Profiles</a></li>
                                        <li><a href="DataRetrieval.html" accesskey="3"
title="">Generate excel</a></li>
                                        <li><a href="Login.php" accesskey="3"
title="">Login</a></li>
```

```
                                        <!--     <li><a     href=""     accesskey="5"
title="">Login</a></li> -->
                                </ul>
                        </div>
                </div>
                <!-- <div id="banner" class="container">
                <div class="title">
                        <h2>Consectetuer adipiscing elit</h2>
                        <span     class="byline">Donec     pulvinar     ullamcorper
metus</span>
                </div>
                <ul class="actions">
                        <li><a href="#" class="button">Pulvinar mollis</a></li>
                </ul>
        </div> -->
        </div>
        <!-- <div id="wrapper">
        <div id="three-column" class="container">
                <div class="title">
                        <h2>Feugiat lorem ipsum dolor sed veroeros</h2>
                        <span class="byline">Donec leo, vivamus fermentum nibh in
augue praesent a lacus at urna congue</span>
                </div>
                <div class="boxA">
                        <p>Phasellus pellentesque, ante nec iaculis dapibus, eros justo
auctor lectus, a lobortis lorem mauris quis nunc. Praesent pellentesque facilisis elit.
Class aptent taciti sociosqu ad  torquent per conubia nostra.</p>
```

```
                        <a href="#" class="button button-alt">More Info</a>
                </div>
                <div class="boxB">
                        <p>Etiam neque. Vivamus consequat lorem at nisl. Nullam
wisi a sem semper eleifend. Donec mattis. Phasellus pellentesque, ante nec iaculis
dapibus, eros justo auctor lectus, a lobortis lorem mauris quis nunc.</p>
                        <a href="#" class="button button-alt">More Info</a>
                </div>
                <div class="boxC">
                        <p> Aenean lectus lorem, imperdiet at, ultrices eget, ornare et,
wisi. Pellentesque adipiscing purus. Phasellus pellentesque, ante nec iaculis
dapibus, eros justo auctor lectus, a lobortis lorem mauris quis nunc.</p>
                        <a href="#" class="button button-alt">More Info</a>
                </div>
        </div>
</div> -->
        <!-- <div id="welcome">
        <div class="container">
                <div class="title">
                        <h2>Fusce ultrices fringilla metus</h2>
                        <span class="byline">Donec leo, vivamus fermentum nibh in
augue praesent a lacus at urna congue</span>
                </div>
                <p>This is <strong>Assembly</strong>, a free, fully
standards-compliant CSS template designed by <a href="http://templated.co"
rel="nofollow">TEMPLATED</a>. The photos in this template are from <a
href="http://fotogrph.com/"> Fotogrph</a>. This free template is released under
the <a href="http://templated.co/license">Creative Commons Attribution</a>
license, so you're pretty much free to do whatever you want with it (even use it
commercially) provided you give us credit for it. Have fun :) </p>
```

```
				<ul class="actions">
					<li><a href="#" class="button">Etiam posuere</a></li>
				</ul>
			</div>
		</div> -->
			<div id="copyright" class="container">
				<!-- <p>&copy; Untitled. All rights reserved. | Photos by <a href="http://fotogrph.com/">Fotogrph</a> | Design by <a href="http://templated.co" rel="nofollow">TEMPLATED</a>.</p> -->
			</div>
</body>

</html>
```

Login.php

```
<html>
	<head> <title>Login-anits</title>
		<link rel="shortcut icon" href="logo.jpg"/>
		<link rel="stylesheet" type="text/css" href="Login.css">
	</head>
	<body>

		<div class="wrapper fadeIn first">
			<div id="formContent">
				<!-- Tabs Titles -->
```

```
                                <h2 class="active"> Sign In </h2>



                                <!-- Icon -->
                                <div class="fadeIn first">
                                        <form  method  =  "post"  id  =  "login"  action  =
"Login.php">
                                        <!-- Login Form -->
                                        <?php
                                                session_start();
                                                $host = "localhost";
                                                $uname = "root";
                                                $pass = "";
                                                $database = "fpm";
                                                $connection      =      mysqli_connect($host,
$uname, $pass);

                                                $selectdb  =  mysqli_select_db($connection,
$database) or
                                                die("Database could not be selected");
                                                $result    =    mysqli_select_db($connection,
$database)
                                                or die("database cannot be selected <br>");

                                                function redirect($url) {
                                                        ob_start();
                                                        header('Location: '.$url);
                                                        ob_end_flush();
                                                        die();
                                                }

                                                $login_placeholder = "login ID";
                                                $password_placeholder = "password";
                                                $login_val = "";

                                                function validate_input(){
                                                        global      $connection,      $login_val,
$login_placeholder, $password_placeholder;
                                                        $query  =  "select  userid  from  login
where userid = \"" . $_POST['login'] ."\";" ;
                                                        $result  =  mysqli_query($connection,
"$query");
                                                        if($row                              =
mysqli_fetch_array($result)){
                                                $login_val                   =
$_POST['login'];

if(isset($_POST['password']) and $_POST['password']!="")

$password_placeholder = "invalid password";
                                                else

$password_placeholder = "enter password";
                                                        }
                                                        else{
                                                $login_val = "";
                                                if($_POST['login'] != "")
                                                                $login_placeholder = "ID does not exist";
                                                                $password_placeholder = "password";
                                                        }
                                                }

                                                if(isset($POST['login']) != "" or isset($_POST['password']) != "")
                                                        validate_input();

                                                // $login_placeholder = "login ID";
                                                // // if(isempty($_POST["login"]))
                                                // if(isset($_POST['login']))
                                                // $login_placeholder = $_POST['login'];



                                                // $password_placeholder = "password";
                                                // // if(isempty($_POST["password"]))
                                                // if(isset($_POST['password']))
                                                //        $password_placeholder = $_POST['password'];

                                                echo "<input type=\"text\" id=\"login\" class=\"fadeIn second\" name=\"login\" value = \"". $login_val ."\" placeholder=\"" . $login_placeholder  . "\">";
                                                echo "<input type=\"password\" id=\"password\" class=\"fadeIn third\" name=\"password\" placeholder=\"" . $password_placeholder . "\">";
```

```
                    if(isset($_POST['login'])          and
isset($_POST['password'])){

                        $query = "select userid, pwd from
login where userid = \"" . $_POST['login'] ."\" and pwd = \"" .
$_POST['password'] . "\";" ;
                        // $query = "select * from login";
                        $result = mysqli_query($connection,
"$query");

                        //   echo   "<h1>".   $row['userid'].
$row['pwd'] . "</h1>";
                        if($row                            =
mysqli_fetch_array($result)){
                            $_SESSION['gmail']             =
$_POST['login'];
                            if($_POST['login'] == 'admin')

redirect('AdminOptions.php');
                            else{
                                $query = "select * from
profile where gmail = \"" . $_POST['login'] ."\" ;" ;
                                $result                    =
mysqli_query($connection, "$query");

                                if($row                    =
mysqli_fetch_array($result))
header('location:EditableForm.php');
                                                    else

header('location:ProfileDetails.html');

                                              }

                                         }
                                    }

                                    //                                                   echo
"<h1>".$_POST['password']."<h1>";

                               ?>

                               <input     type="submit"     class="fadeIn     fourth"
value="Log In">
                               </form>
                               <!-- Remind Passowrd -->
                          <div id="formFooter">
                          <a                                        class="underlineHover"
href="Forgotpassword.html">Forgot Password?</>
                     </div>
                </div>
           </body>
</html>
```

Publications.php

```
<!DOCTYPE html>



<html>

<head>

<title>publications by staff</title>

<link rel="stylesheet"  type="text/css" href="1.css">

</head>

<body>

<h1>Publications by Staff</h1>

        <div class="register">
        <form method="post" id="register" action="publicationsconnection.php">

        <h2>Enter your datails here</h2>

    <label>Title:</label><br>

        <input type="text" class="a" placeholder="" name="title"><br><br>

        <label>conference/Journal Name:</label><br>

        <input          type="text"          class="a"          placeholder=""
name="conferencename"><br><br>

   <label>category:</label><br>

        <input   type="text"   class="a"   placeholder="     National/international"
name="category"><br><br>
```

```
<!-- <label>Staff Name:</label><br>

<input type="text" class="a" placeholder="  Enter name of the Staff" name=""><br><br>

<label>Department</label><br>

<input type="text" class="a" placeholder=""><br><br> -->

<label>Academic Year</label><br>

<input type="text" class="a" placeholder="  YYYY-YYYY" name="academic"><br><br>

<label>E-Mail ID</label><br>

<input type="email" class="a" placeholder=" Enter email" name="gmail"><br><br>
```

```
        <input type="submit" class="submit" value="Submit"><br>

        </form></div>

        <button class="ach ach1"><a style="text-decoration: none;color:
white;"href="conferenceworkshops.php">Conferences/Workshops</a></button>

</body >
```

6. TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner.

Software testing is an important element of the software quality assurance and represents the ultimate review of specification, design and coding. The increasing feasibility of software as a system and the cost associated with the software failures are motivated forces for well planned through testing.

Testing Objectives:

These are several rules that can save as testing objectives they are:

- Testing is a process of executing program with the intent of finding an error.

- A good test case is one that has a high probability of finding an undiscovered error.

6.1 Types of Testing

In order to make sure that the system does not have errors, the different levels of testing strategies that are applied at differing phases of software development are:

6.1.1 Unit Testing

Unit Testing is done on individual modules as they are completed and become executable. It is confined only to the designer's requirements. Unit testing is different from and should be preceded by other techniques, including:

- Inform debugging
- Code debugging

Each module can be tested using the following two strategies:
Black Box Testing

In this strategy some test cases are generated as input conditions that fully execute all functional requirements for the program. This testing has been used to find error in the following categories:

- Incorrect or missing functions.
- Interface errors.
- Errors in data structure or external database access.
- Performance Error.
- Initialization and termination errors.
- In this testing only the output is checked for correctness.
- The logical flow of the data is not checked.

White Box Testing
In this the test cases are generated on the logic of each module by drawing flow graphs of that module and logical decisions are tested on all cases. It has been used to generate the test cases in the following cases:

- Guarantee that all independent paths have been executed.
- Execute all loops at their boundaries and within their operational bounds.
- Execute internal data structures to ensure their validity.

6.1.2 Integration Testing

Integration testing ensures that software and subsystems work together a whole. It tests the interface of all the modules to make sure that the modules behave properly when integrated together. It is typically performed by developers, especially at lower, module-to-module level. Testers become involved at the higher levels.

6.1.3 System Testing

Involves in house testing of entire system before delivery to the user. The aim is to satisfy the user, the meets all the requirements of the client's specifications. It is conducted by the testing organization if a company has one. Test data may range from hand generated to production.
Requires test scheduling to plan and organize:

- Inclusion of changes/fixes.
- Test data to use.

One common approach is graduated testing: as system testing progresses and (hopefully) fewer and fewer defects are found, the code is frozen for testing for increasingly longer time periods.

6.1.4 Acceptance Test

It is a pre-delivery testing in which entire system is tested at client's site on real world data to find errors.

User Acceptance Test (UAT)

"**Beta Testing**": Acceptance testing in the customer environment.

Requirements traceability:

- Match requirements to test cases.
- Every requirement ha to be cleared by at least one test case.
- Display in a matrix of requirements vs. test case

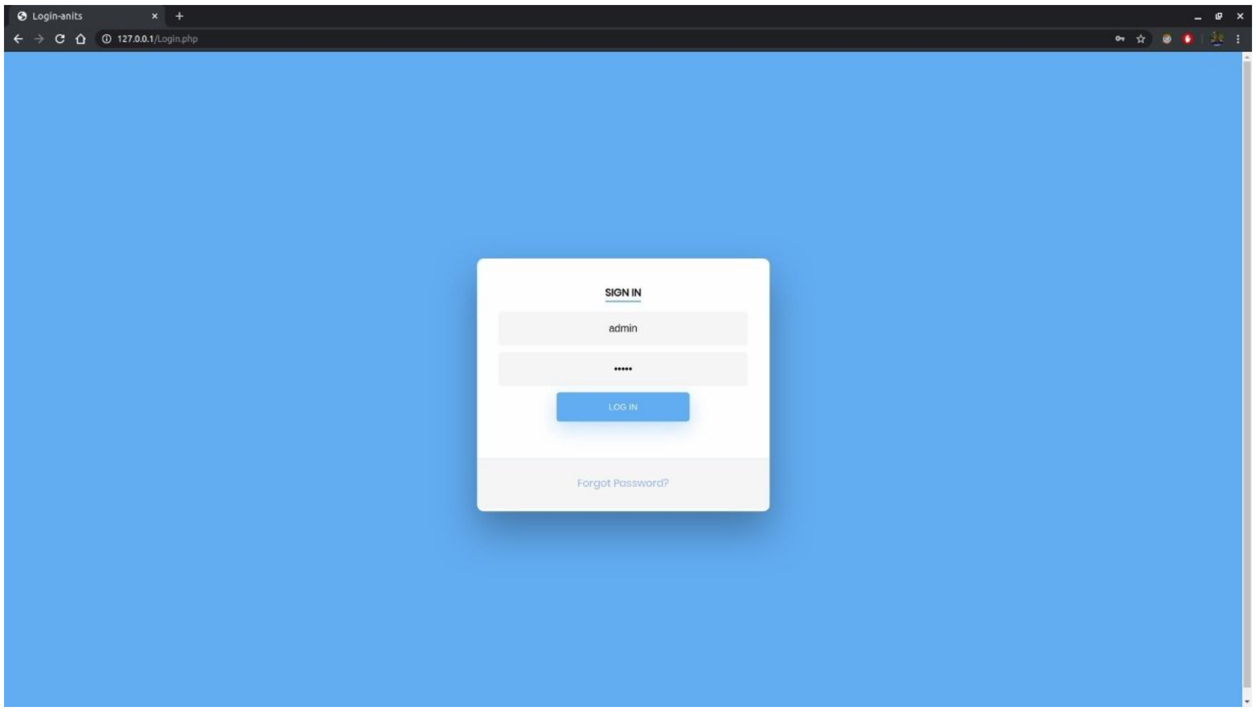
6.2 Test Cases

In general, a test case is a set of test data and test program and their expected results. A test case in software engineering normally consists of unique identifier, requirement references from a design specification, preconditions, events, a series of steps (also known as actions) to follow, input, output and it validates one or more system requirements and generates a pass or fail.

TEST CASES FOR PROJECT:

In general a test case is a set of test data and test programs and their expected results. A test case in software engineering normally consists of a unique identifier, requirement references from a design specification, pre conditions, events, a series of steps (also known as actions) to follow input, output and it validates one or more system requirements and generates a pass or fail.

6.2.1 FACULTY LOGIN



Test plan id**:** Pid1

Test case id: 101

Features to be tested: Pending Faculty Login

Pre conditions: 1. Website must be running

2. Valid userid

3. Check the details of userid must be from database

Test Script: 1.Verify Login Details

2.Check Login Credentials

3.Accept the Database if details are valid

Test Data: 1.Valid userid

2.Invalid userid

3.Valid credentials that are filled

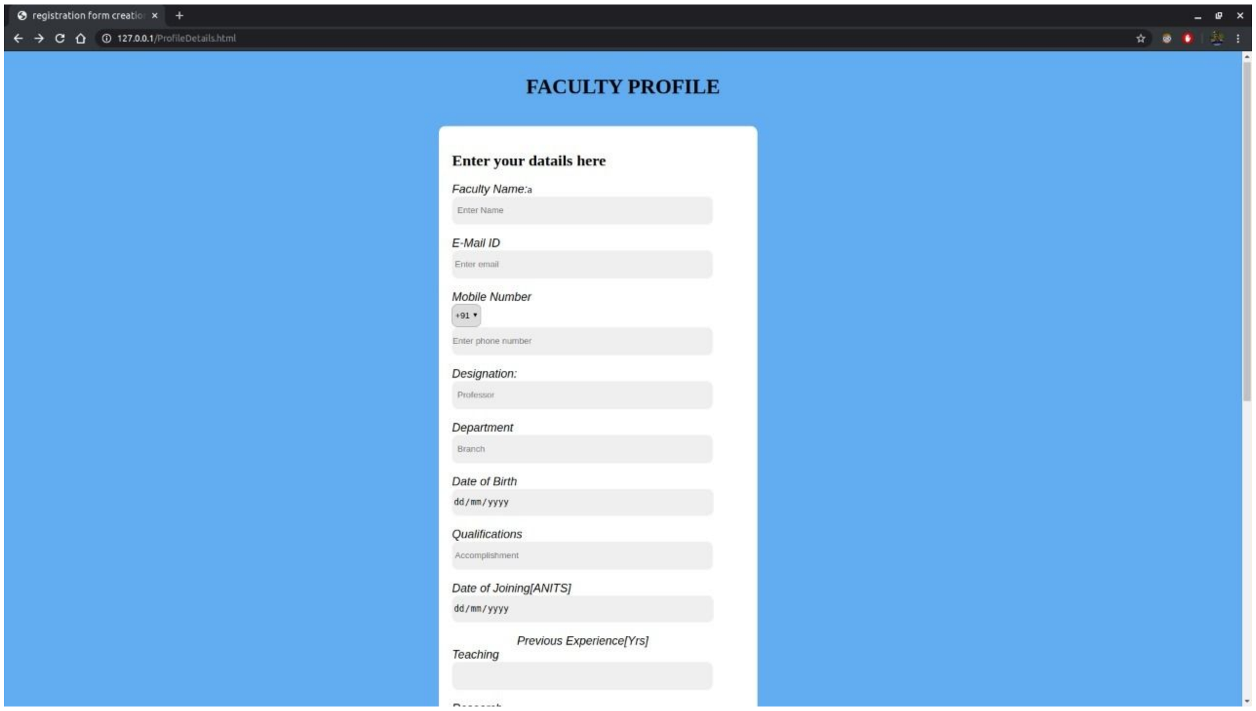
4.Invalid credentials that are filled

Expected Results: 1.Accept the faculty login request if details are valid

2.Decline the faculty login request if details are invalid

Test Status : Fail

6.2.2 FACULTY REGISTRATIONS



registration form creatio ×

127.0.0.1/ProfileDetails.html

FACULTY PROFILE

Enter your datails here

Faculty Name:a

Enter Name

E-Mail ID

Enter email

Mobile Number

+91

Enter phone number

Designation:

Professor

Department

Branch

Date of Birth

dd/mm/yyyy

Qualifications

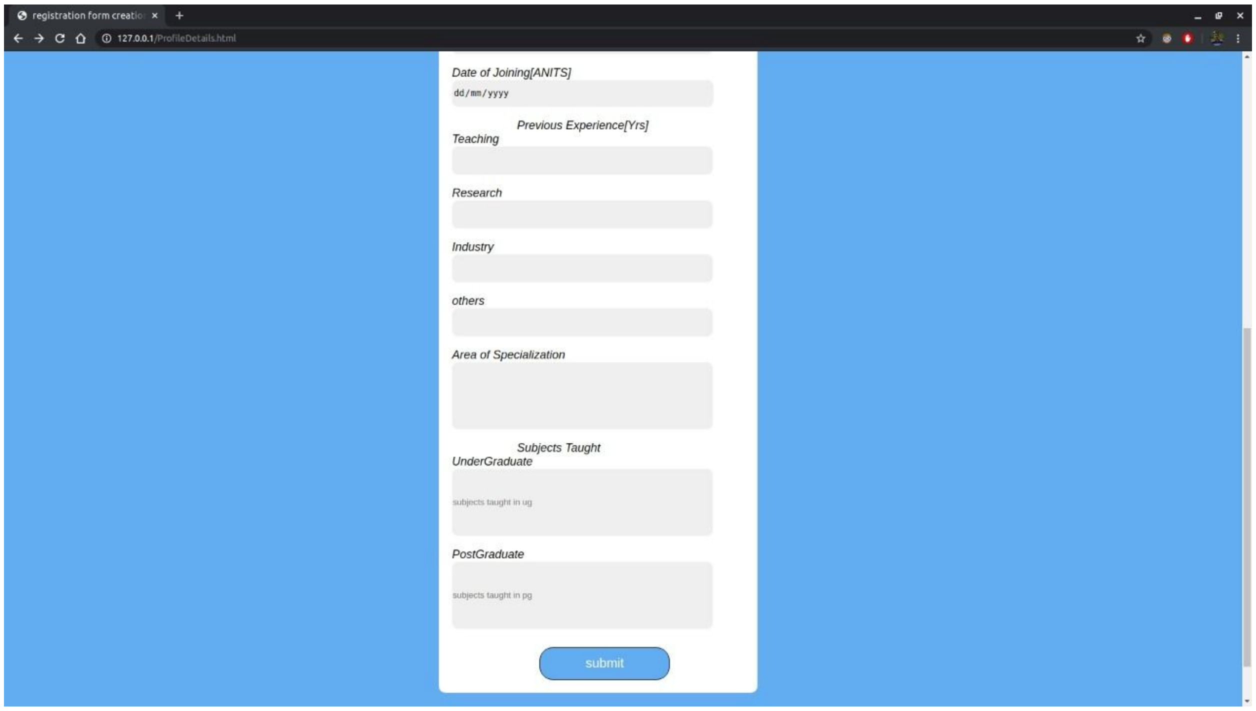
Accomplishment

Date of Joining[ANITS]

dd/mm/yyyy

Previous Experience[Yrs]

Teaching



registration form creatio ×

127.0.0.1/ProfileDetails.html

Date of Joining[ANITS]

dd/mm/yyyy

Previous Experience[Yrs]

Teaching

Research

Industry

others

Area of Specialization

Subjects Taught

UnderGraduate

subjects taught in ug

PostGraduate

subjects taught in pg

submit

Test plan id: Pid2

Test case id: 102

Features to be tested: Pending Faculty Registrations

Pre conditions: 1. Website must be running

2. Valid Login Details

Test Script: 1.Verify Login Details

2.Check faculty's Credentials

3.Accept and Update the Database if details are valid

Test Data: 1.Valid faculty id

2.Invalid faculty id

3.Valid credentials that are filled

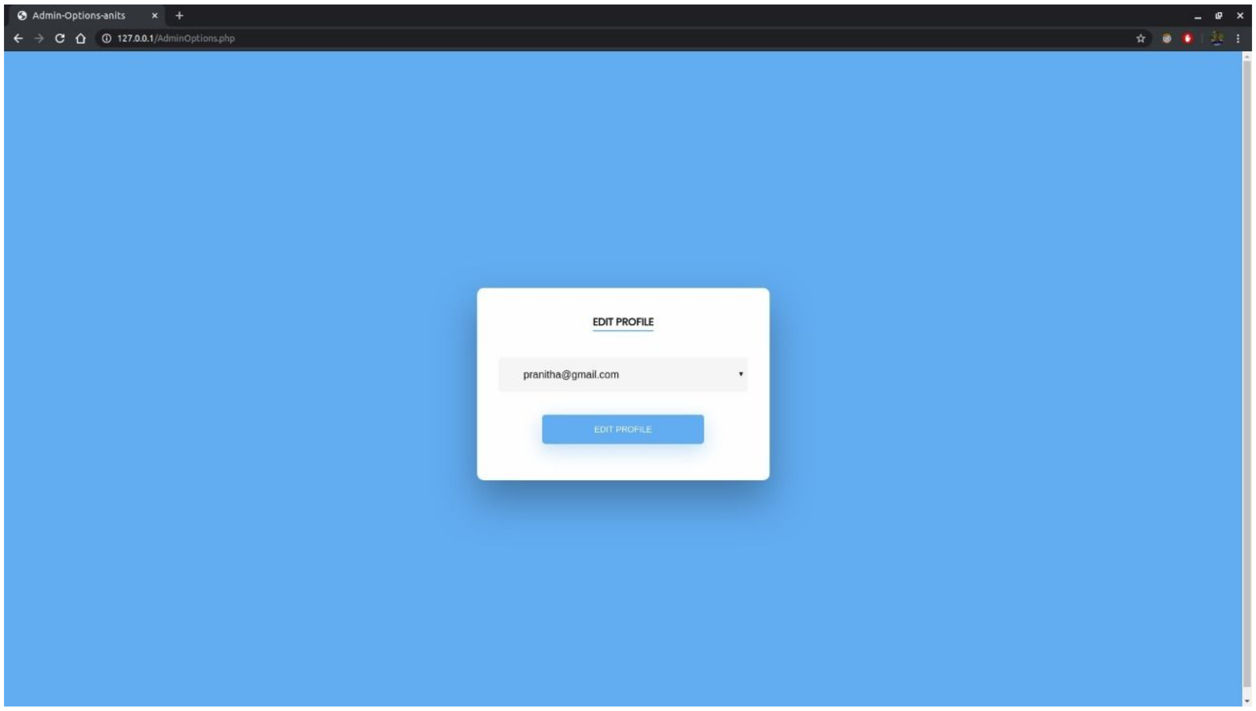
4.Invalid credentials that are filled

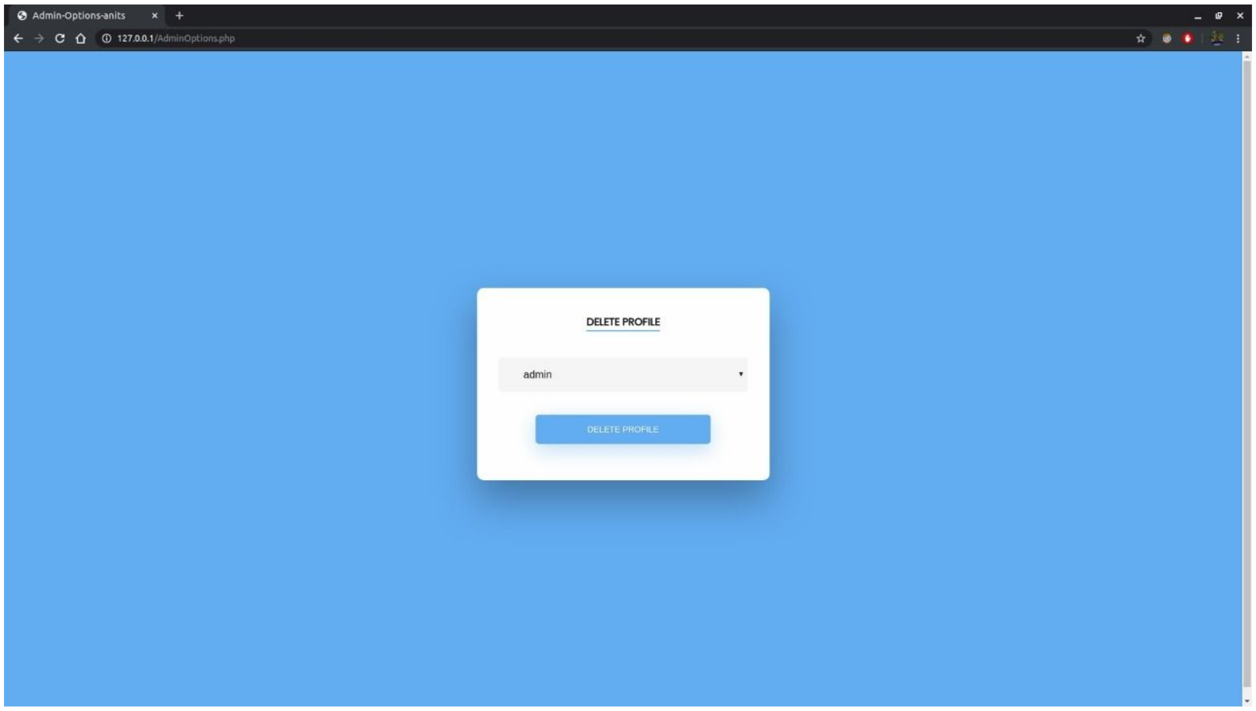
Expected Results: 1.Accept the faculty registration request if details are valid

2.Decline the faculty registration request if details are invalid

Test Status : Pass

6.2.3 ACCOUNT DELETION/ADDITION





Test plan id: Pid3

Test case id: 103

Features to be tested: Pending AccountAcoount deletion/addition

Pre conditions: 1. Website must be running

2. Valid Login Details of the Admin

3.Only admin can delete/add the account

Test Script: 1.Verify Login Details

2.Check Admin Credentials

3.Accept and Update the Database if details are valid

Test Data: 1.Valid faculty id

2.Invalid faculty id

3.Valid credentials that are filled

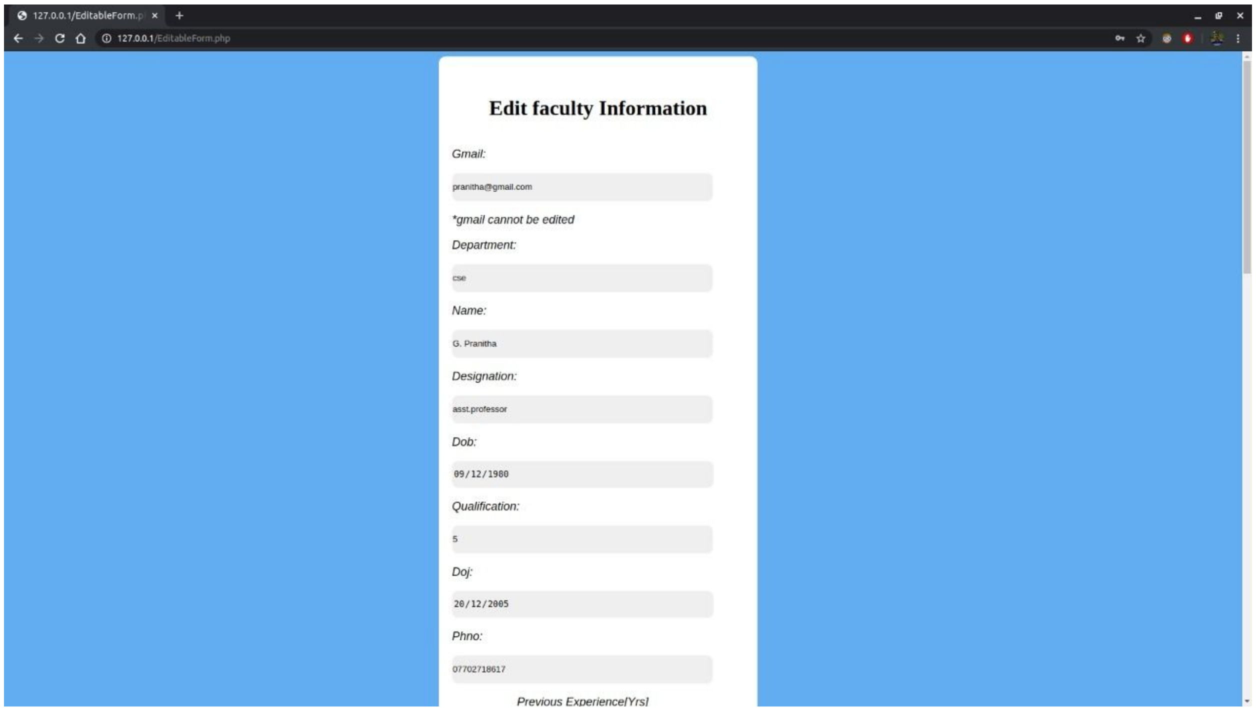
4.Invalid credentials that are filled

Expected Results: 1.Accept the account deletion/addition request if details are valid and must be updated in the database.

2.Decline the account deletion/addition request if details are invalid.

Test Status : Fail

6.2.4 EDITING THE DETAILS



Edit faculty Information

Gmail:

pranitha@gmail.com

*gmail cannot be edited

Department:

cse

Name:

G. Pranitha

Designation:

asst.professor

Dob:

09/12/1980

Qualification:

5

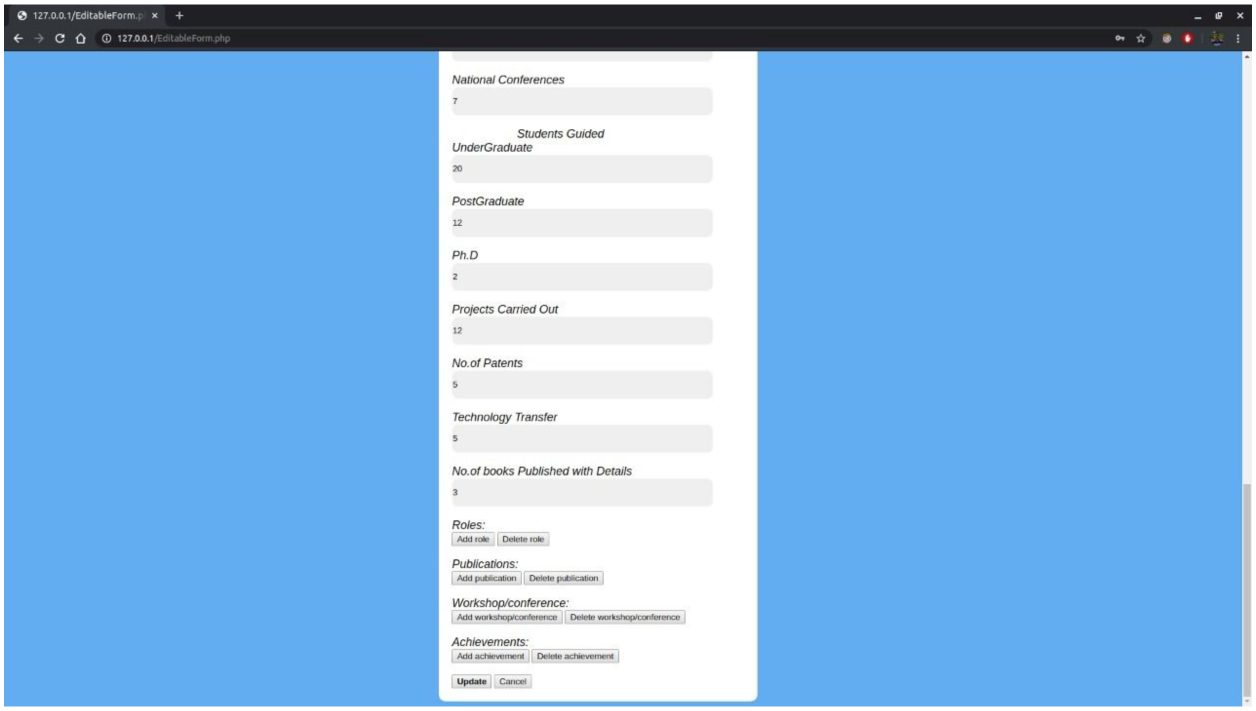
Doj:

20/12/2005

Phno:

07702718617

Previous Experience[Yrs]



National Conferences

7

Students Guided

UnderGraduate

20

PostGraduate

12

Ph.D

2

Projects Carried Out

12

No.of Patents

5

Technology Transfer

5

No.of books Published with Details

3

Roles:

Publications:

Workshop/conference:

Achievements:

Test plan id: Pid4

Test case id: 104

Features to be tested: Editing the details

Pre conditions: 1. Website must be running

2. Valid Login Details

3.Only admin or registered user can edit the details

Test Script: 1.Verify Login Details

2.Check faculty's or admin Credentials

3.Accept and Update the Database if details are valid

Test Data: 1.Valid faculty id

2.Invalid faculty id

3.Valid credentials that are filled

4.Invalid credentials that are filled

Expected Results: 1. Accept the details that are edited only by the registered user or admin and must be updated in the database

2. Decline the details that are edited by others and it must not be updated in the database

Test Status : Pass

6.3 ERROR REPORT:

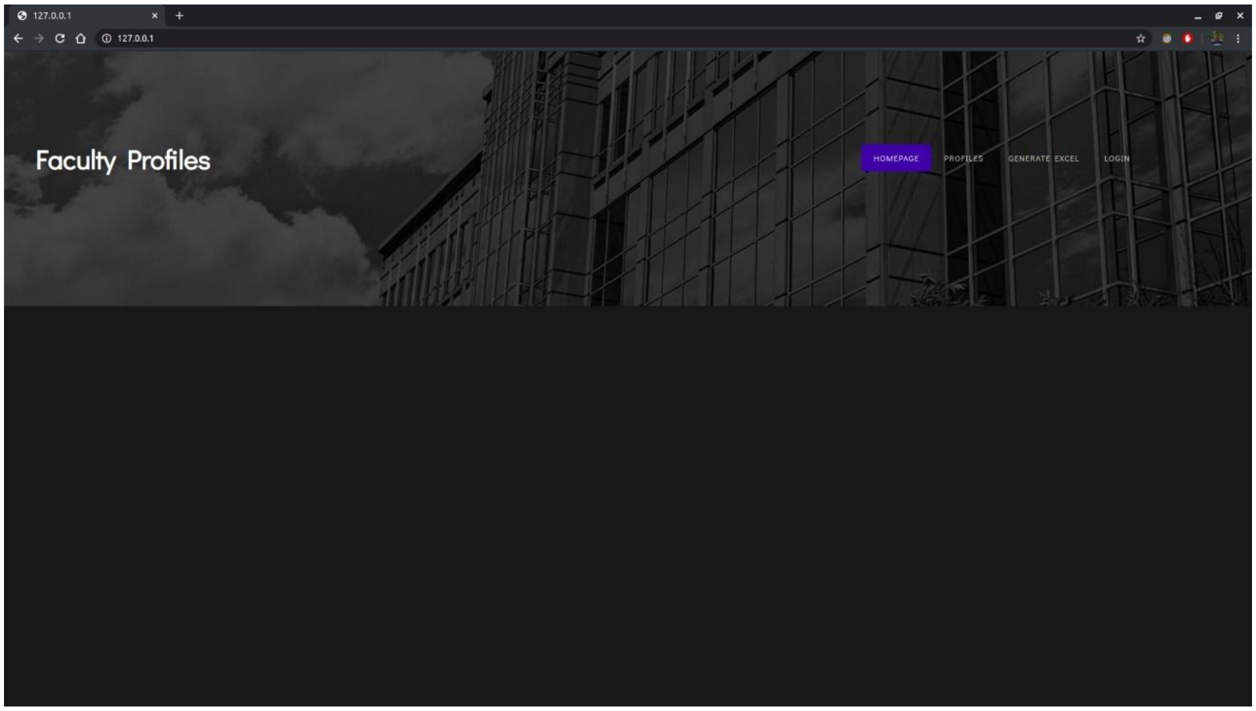
S.NO	` CASE CONDITION	INPUT	ERROR MESSAGE

1.	Invalid mail-id	Anitsmail@.com	Please type your e-mail address in the format : yourname@gmail.com
2.	Invalid Password	Anit	password you provided must have Atleast 8 characters
3.	Deleting Account	Admin Login	Only admin can delete the account.

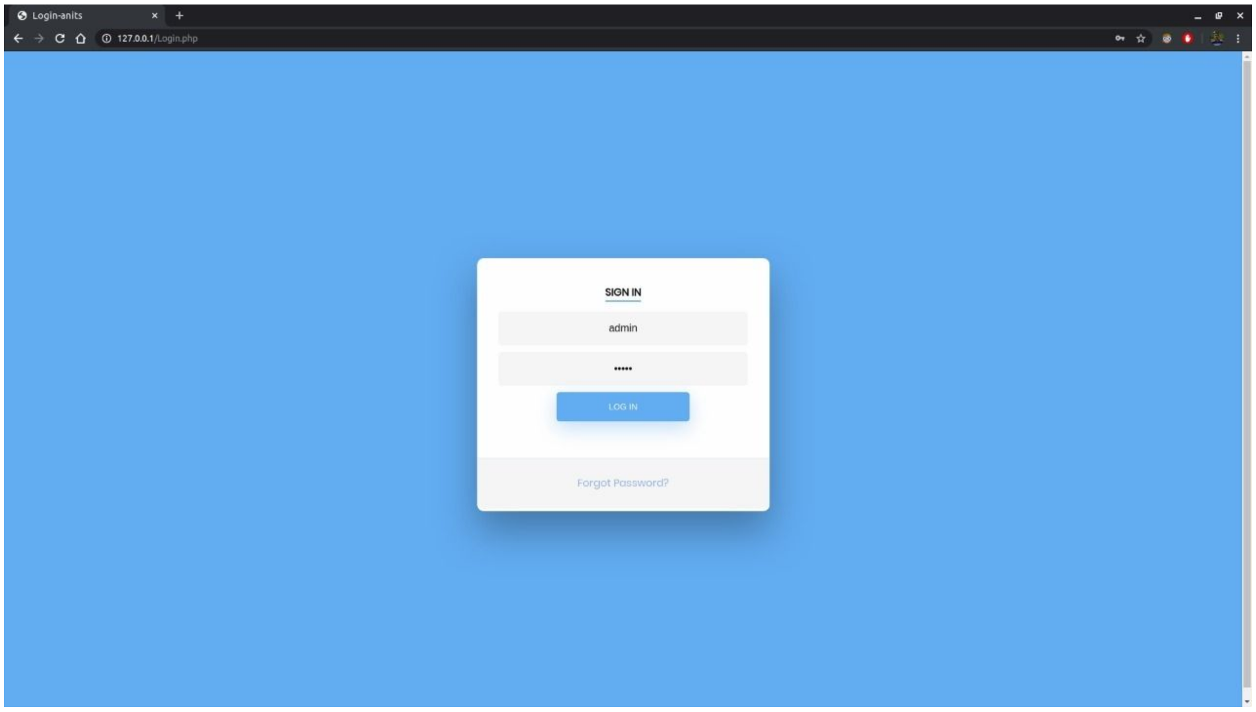
7. RESULTS

7.1 Input/Output Design:

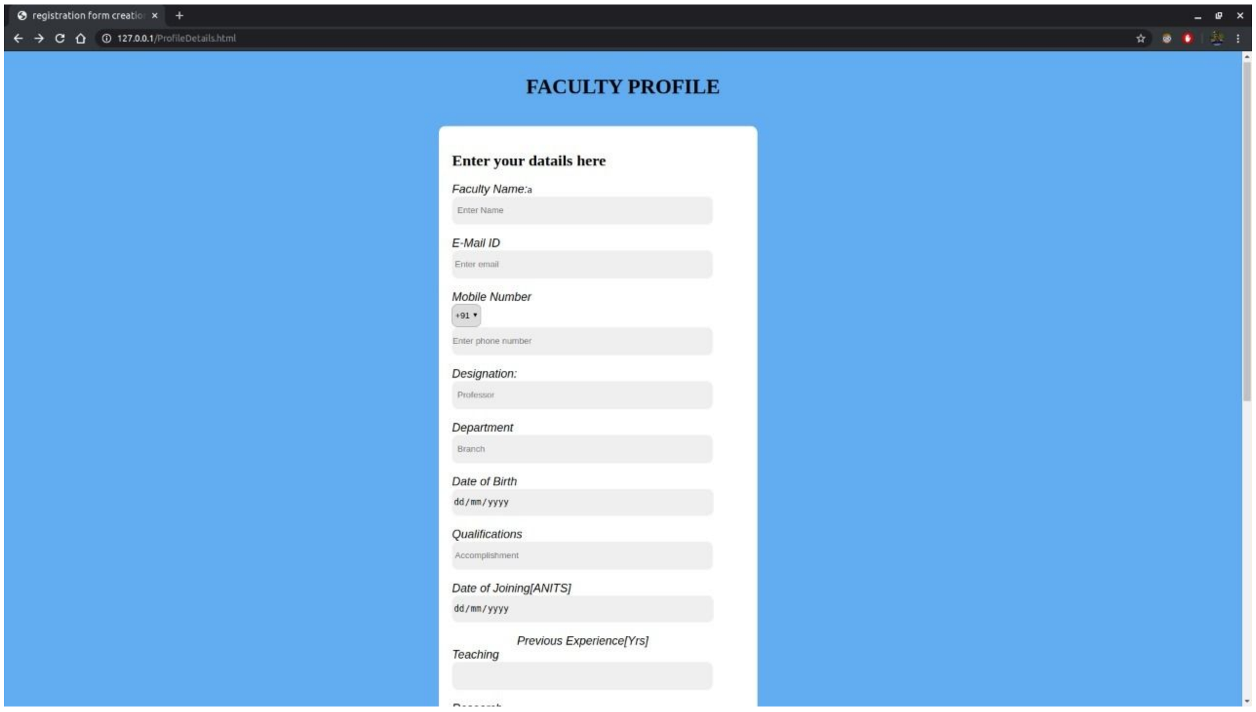
- **Home Page:**



- **Click on Admin/Faculty to Login:**



- **Registration Page for Faculty:**


registration form creatio
127.0.0.1/ProfileDetails.html

FACULTY PROFILE

Enter your datails here

*Faculty Name:*a
Enter Name

E-Mail ID
Enter email

Mobile Number
+91
Enter phone number

Designation:
Professor

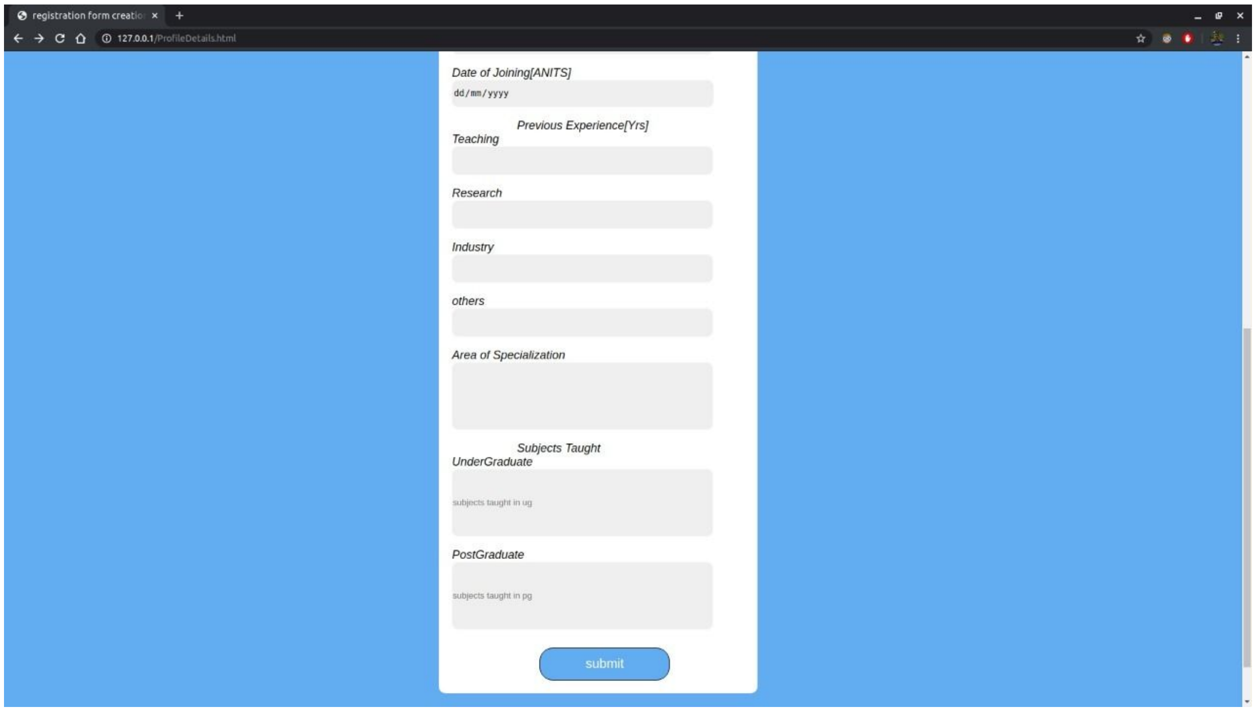
Department
Branch

Date of Birth
dd/mm/yyyy

Qualifications
Accomplishment

Date of Joining[ANITS]
dd/mm/yyyy

Previous Experience[Yrs]
Teaching


registration form creatio
127.0.0.1/ProfileDetails.html

Date of Joining[ANITS]
dd/mm/yyyy

Previous Experience[Yrs]
Teaching

Research

Industry

others

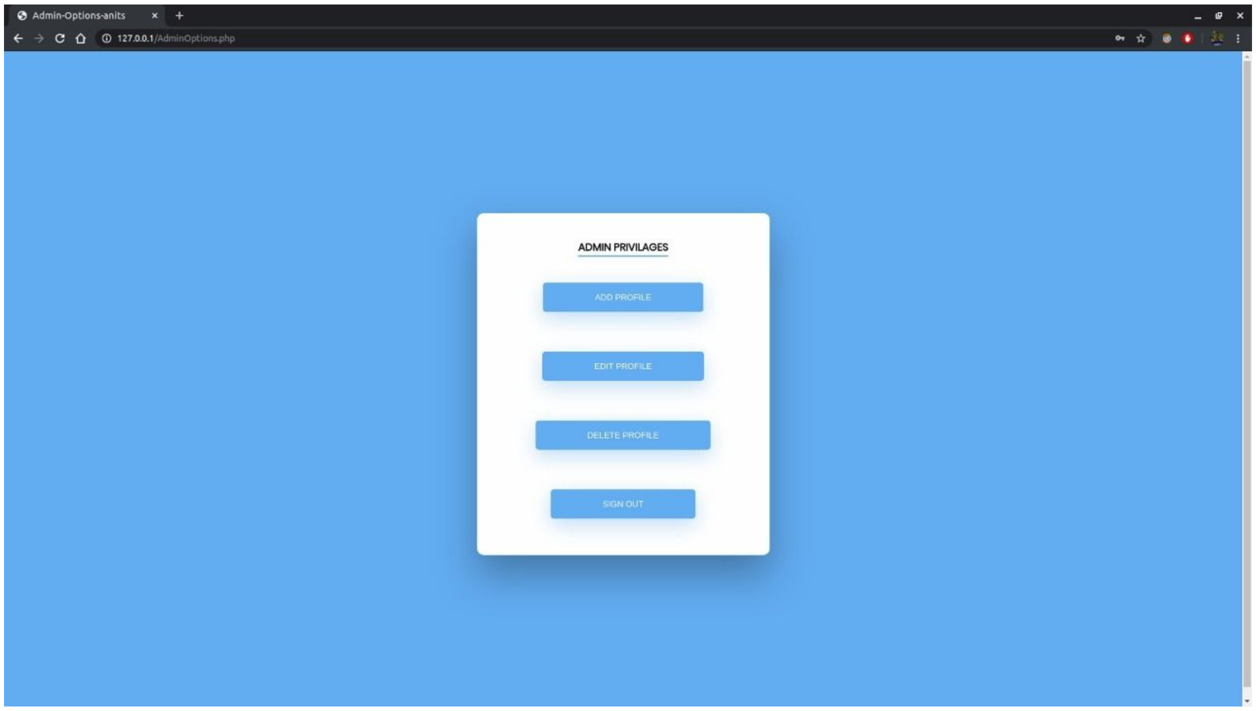
Area of Specialization

Subjects Taught
UnderGraduate
subjects taught in ug

PostGraduate
subjects taught in pg

submit

- **Admin's Functionalities Page:**



8. CONCLUSION AND FUTURE SCOPE

8.1 CONCLUSION:

The Proposed System of Faculty profile management System offers all functionalities of the Existing Faculty profile management System and it aims to reduce the paperwork. Information of a faculty can be retrieved whenever needed. This system provides Security as the access is granted to only Admin and Faculty and the faculty can view/edit his/her profile and cannot edit other profiles only the . When a new faculty registers, only the admin has the ability to create a new faculty profile in order to provide security. This system provides a user friendly and response interface.

8.2 FUTURE SCOPE:

In future, we would try to enhance the responsiveness of the system, while it is online profile management with security. Currently the proposed system is dedicated to only Computer Science Engineering Department of ANITS, and we would try to expand this system to all the Departments in ANITS in future.

9. REFERENCES

Web Sites:

1. https://www.w3schools.com/html/
2. https://www.tutorialspoint.com/javascript/
3. https://www.geeksforgeeks.org/web-technology/
4. https://www.javatpoint.com/php-json-example

Textbooks:

1. Steven Holzner, "HTML Black Book: The Programmer's Complete HTML Reference Book"
2. Robin Nixon, "Learning PHP, MySQL, and JavaScript", 4th edition

GitHub Link:

https://github.com/yogesh9d/Faculty-profile-management