1. **INTRODUCTION**

A website is the face of any reputed organization. It is highly advisable to have a website running with the latest technological trends. Robustness of technology isn’t the only thing that makes a good website, having a pleasing user interface is a must have. Functionality, ease of use and site administration should be hassle free.

**1.1 Existing system**

The existing website of PVPSIT is based on one of the best available designs. But the problems that we are currently facing are related to attendance posting and the content management system. This leads to great difficulty to maintain and the existing system leads to consumption of more time.

**1.2 Proposed System and Advantages**

The main aim of the project is to overcome the difficulties by modifying the existing system in the areas like: editing the things from the front end, avoiding multiple submissions, there by providing an efficient working module of the student attendance.

**MODULES:**

1. **User Interface Module:** User interface is one of the most important parts of any program because it determines how easily you can make the program do what you want. A User Interface is a medium between user and computer through which user communicates with the program.
2. **Admin Module:** With the help of this module the administrator can add, delete and modify the news and the notices that are visible in our home page.
3. **Page Editor Module:** This module helps in the creation of new pages whenever we want to create a page. Here this module provides a User Interface for the creation of new pages.
4. **Attendance Module:** In this module, the problems already present are rectified by disabling the submit button after submitting just once.

**2. REQUIREMENTS SPECIFICATIONS**

**2.1 Software Requirements:**

Operating System : Windows or Linux

Web Server : Apache Tomcat

Web Browser : Modern web browsers

Front-End : HTML

Client side Script : JavaScript

Server side Script : PHP

Back-End : MySQL

**2.1.1. HTML:**

HTML, an acronym of Hypertext Markup Language, is the predominant markup language for web pages. It provides a means to describe the structure of text-based information in a document — by denoting certain text as headings, paragraphs, lists, and so on — and to supplement that text with interactive forms, embedded images, and other objects. HTML is written in the form of labels (known as tags), surrounded by angle brackets. HTML can also describe, to some degree, the appearance and semantics of a document, and can include embedded scripting language code which can affect the behavior of web browsers and other HTML processors.

Hypertext Markup Language (HTML), the languages of the World Wide Web (WWW), allows users to produces Web pages that include text, graphics and pointer to other Web pages (Hyperlinks).

HTML is not a programming language but it is an application of ISO Standard 8879, SGML (Standard Generalized Markup Language), but specialized to hypertext and adapted to the Web. The idea behind Hypertext is that instead of reading text in rigid linear structure, we can easily jump from one point to another point. We can navigate through the information based on our interest and preference. A markup language is simply a series of elements, each delimited with special characters that define how text or other items enclosed within the elements should be displayed. Hyperlinks are underlined or emphasized works that load to other documents or some portions of the same document.

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

HTML provides tags (special codes) to make the document look attractive. HTML tags are not case-sensitive. Using graphics, fonts, different sizes, color, etc., can enhance the presentation of the document. Anything that is not a tag is part of the document itself.

**Basic HTML Tags:**

<! -- --> specifies comments

<A>……….</A> Creates hypertext links

<B>……….</B> Formats text as bold

<BODY>…</BODY> Contains all tags and text in the HTML doc

<CENTER>...</CENTER> Creates text

<DD>…</DD> Definition of a term

<DL>...</DL> Creates definition list

<FONT>…</FONT> Formats text with a particular font

<FORM>...</FORM> Encloses a fill-out form

<FRAME>...</FRAME> Defines a particular frame in a set of frames

<H#>…</H#> Creates headings of different levels (1 – 6)

<HEAD>...</HEAD> Contains tags that specify information

<HR>...</HR> Creates a horizontal rule

<HTML>…</HTML> Contains all other HTML tags

<META>...</META> Provides meta-information about document

<SCRIPT>…</SCRIPT> Contains client-side or server-side script

<TABLE>…</TABLE> Creates a table

<TD>…</TD> Indicates table data in a table

<TR>…</TR> Designates a table row

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

**Attributes:**

The attributes of an element are name-value pairs, separated by "=", and written within the start label of an element, after the element's name. The value should be enclosed in single or double quotes, although values consisting of certain characters can be left unquoted in HTML (but not XHTML).Leaving attribute values unquoted is considered unsafe.

Most elements take any of several common attributes: id, class, style and title. Most also take language-related attributes: Lang and dir.

The id attribute provides a document-wide unique identifier for an element. This can be used by style sheets to provide presentational properties, by browsers to focus attention on the specific element or by scripts to alter the contents or presentation of an element. The class attribute provides a way of classifying similar elements for presentation purposes. For example, an HTML document (or a set of documents) may use the designation class="notation" to indicate that all elements with this class value are all subordinate to the main text of the document (or documents). Such notation classes of elements might be gathered together and presented as footnotes on a page, rather than appearing in the place where they appear in the source HTML.

An author may use the style non-attribute codes presentational properties to a particular element. It is considered better practice to use an element’s son- id page and select the element with a style sheet, though sometimes this can be too cumbersome for a simple ad hoc application of styled properties. The title is used to attach sub textual explanation to an element. In most browsers this title attribute is displayed as what is often referred to as a tool tip. The generic inline span element can be used to demonstrate these various non-attributes.

The preceding displays as HTML (pointing the cursor at the abbreviation should display the title text in most browsers).

**Advantages:**

* A HTML document is small and hence easy to send over the net.
* It is small because it does not include formatted information.
* HTML is platform independent.
* HTML tags are not case-sensitive.

**2.1.2 PHP**

PHP (Hypertext Preprocessor) is a widely-used open source general-purpose scripting language that is especially suited for web development and can be embedded into HTML. Instead of lot of commands to output, HTML (as seen in C or Perl), PHP pages contain HTML with embedded code that does "something". The PHP code is enclosed in special [start and end processing instructions <? And?>](http://www.php.net/manual/en/language.basic-syntax.phpmode.php) that allow you to jump into and out of "PHP mode. “The best things in using PHP are that it is extremely simple for a newcomer, but offers many advanced features for a professional programmer. Although PHP's development is focused on server-side scripting, we can do much more with it. It provides security to the files and it is much more flexible to the users. PHP performs system functions i.e. from files on a system it can create, open, read, write and close them. Using PHP you can restrict users to access some pages of your website.

**Capabilities of PHP:**

PHP is mainly focused on server-side scripting, so you can do anything any other CGI program can do, such as collect form data, generate dynamic page content, or send and receive cookies. But PHP can do much more.

There are three main areas where PHP scripts are used.

* **Server-side scripting**: This is the most traditional and main target field for PHP. You need three things to make this work. The PHP parser (CGI or server module), a web server and a web browser. You need to run the web server, with a connected PHP installation. You can access the PHP program output with a web browser, viewing the PHP page through the server. All these can run on your home machine if you are just experimenting with PHP programming. See the [installation instructions](http://www.php.net/manual/en/install.php) section for more information.
* **Command line scripting**: You can make a PHP script to run it without any server or browser. You only need the PHP parser to use it this way. This type of usage is ideal for scripts regularly executed Task Scheduler (on Windows). These scripts can also be used for simple text processing tasks.

PHP can be [used](http://www.php.net/manual/en/install.php) on all major operating systems, including Linux, many UNIX variants (including HP-UX, Solaris and OpenBSD), Microsoft Windows, Mac OS X, RISC OS, and probably others. PHP has also support for most of the web servers today..

So with PHP, you have the freedom of choosing an operating system and a web server. Furthermore, you also have the choice of using procedural programming or object oriented programming (OOP), or a mixture of them both.

With PHP you are not limited to output HTML. PHP's abilities include outputting images, PDF files and even Flash movies (using libswf and Ming) generated on the fly. PHP can auto generate these files, and save them in the file system, instead of printing it out, forming a server-side cache for your dynamic content.

One of the strongest and most significant features in PHP is its support for a [wide range of databases](http://www.php.net/manual/en/refs.database.php). Writing a database-enabled web page is incredibly simple using one of the database specific extensions (e.g., for [MySQL](http://www.php.net/manual/en/book.mysqli.php)), or using an abstraction layer like [PDO](http://www.php.net/manual/en/book.pdo.php), or connect to any database supporting the Open Database Connection standard via the [ODBC](http://www.php.net/manual/en/book.uodbc.php) extension. And many other interesting extensions exist, which are categorized both [alphabetically](http://www.php.net/manual/en/extensions.php) and by [category](http://www.php.net/manual/en/funcref.php). And there are additional PECL extensions that may or may not be documented within the PHP manual itself, like [» Debug](http://xdebug.org/).

**2.1.3 Introduction to MYSQL:**

MYSQL is a fast, easy-to-use RDBMS used for databases on many web sites. Speed was the developer’s main focus from the beginning. In the interest of speed, they made the decision to offer fewer features than their major competitors (for instance, Oracle and Sybase). However, even though MYSQL is less full featured than its commercial competitors, it has all the features needed by the large majority of database developers. It’s easier to install and use than its commercial competitors.

**2.1.4 Apache Tomcat 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 Java Server 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.

**3. SYSTEM ANALYSIS**

**3.1 Study of the System**

* User Interface module deals with providing rich client interface for the user which helps in the easy navigation in the website
* Admin module deals with the Addition, Deletion, Modification of the News and the Events that are present without getting connected to the back end.
* Page Editor module deals with the addition of new pages to the website with ease for the user.
* Attendance module rectifies the problem already present in the existing system such as resubmission of the attendance by just disabling the submit button after once the attendance has been submitted.

**Stages in SDLC:**

* Requirement Gathering
* Analysis
* Designing
* Coding
* Testing
* Maintenance

**Requirements Gathering Stage:**

The requirements gathering process takes as its input the goals identified in the high-level requirements section of the project plan. Each goal will be refined into a set of one or more requirements. These requirements define the major functions of the intended application, define operational data areas and reference data areas, and define the initial data entities. Major functions include critical processes to be managed, as well as mission critical inputs, outputs and reports. A user class hierarchy is developed and associated with these major functions, data areas, and data entities. Each of these definitions is termed a Requirement. Requirements are identified by unique requirement identifiers and, at minimum, contain a requirement title and description.

**Analysis Stage:**

The planning stage establishes a bird's eye view of the intended software product, and users this to establish the basic project structure, evaluate feasibility and risks associated with the project, and describe appropriate management and technical approaches. The minimum information for each goal consists of a title and textual description, although additional information and references to external documents may be included. The outputs of the project planning stage are the configuration management plan, the quality assurance plan, and the project plan and schedule, with a detailed listing of scheduled activities for the upcoming requirements stage, and high level estimates of effort for the out stages.

**Designing Stage:**

The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and prototype efforts. Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input.

**Development (Coding) Stage:**

The development stage takes as its primary input the design elements described in the approved design document. For each design element, a set of one or more software artifacts will be produced. Software artifacts include but are not limited to menus, dialogs, and data management forms, data reporting formats, and specialized procedures and functions. Appropriate test cases will be developed for each set of functionally related software artifacts, and an online help system will be developed to guide users in their interactions with the software.

**Integration and Test Stage:**

During the integration and test stage, the software artifacts, online help, and test data are migrated from the development environment to a separate test environment. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite confirms a robust and complete migration capability. During this stage, reference data is finalized for production use and production users are identified and linked to their appropriate roles. The final reference data (or links to reference data source files) and production user list are compiled into the Production Initiation Plan.

**Installation and Acceptance Test:**

During the installation and acceptance stage, the software artifacts, online help, and initial production data are loaded onto the production server. At this point, at least cases are run to verify the correctness and completeness of the software. Successful execution of the test suite is a prerequisite to acceptance of the software by the customer.

**Maintenance:**

Outer rectangle represents maintenance of a project, Maintenance team will start with requirement study, understanding of documentation later employees will be assigned work and they will undergo training on that particular assigned category.

**4. SYSTEM DESIGN**

**4.1 Introduction to UML:**

The Unified Modeling Language (UML) is a graphical language for visualizing,specifying, constructing, and documenting the artifacts of a software-intensive system. The UML offers a standard way to write a system's blueprints, including conceptual things such as business processes and system functions as well as concrete suchas programming language statements, database schemas, and reusable softwarecomponents.

The building blocks of UML can be defined as:

* Things
* Relationships
* Diagrams

**1. Things:**

Things are the most important building blocks of UML. Things can be:

* + - * Structural
      * Behavioral
      * Grouping
      * Annotational

# Structural things:

The **Structural things** define the static part of the model. They represent physical and conceptual elements. Following are the brief descriptions of the structural things.

**Class:**

**Class** represents set of objects having similar responsibilities.

class

## Interface:

**Interface** defines a set of operations which specify the responsibility of a class.

Interface

## Collaboration:

**Collaboration** defines interaction between elements.



## Use case:

**Use case** represents a set of actions performed by a system for a specific goal.

Use case

## Component:

**Component** describes physical part of a system.

Component

## Node:

A **node** can be defined as a physical element that exists at run time.



# Behavioural things:

**A behavioural thing** consists of the dynamic parts of UML models. Following are the behavioural things:

## Interaction:

**Interaction** is defined as a behaviour that consists of a group of messages exchanged among elements to accomplish a specific task.

Interaction

## State machine:

**State machine** is useful when the state of an object in its life cycle is important. It defines the sequence of states an object goes through in response to events. Events are external factors responsible for state change.



# Grouping things:

**Grouping things** can be defined as a mechanism to group elements of a UML model together. There is only one grouping thing available.

## Package:

**Package** is the only one grouping thing available for gathering structural and behavioral things.



# Annotational things:

**Annotational things** can be defined as a mechanism to capture remarks, descriptions, and comments of UML model elements. **Note** is the only one Annotational thing available.

## Note:

A note is used to render comments, constraints etc of an UML element.

Note

**2. Relationship:**

**Relationship** is another most important building block of UML. It shows how elements are associated with each other and this association describes the functionality of an application.

There are four kinds of relationships available.

## Dependency:

Dependency is a relationship between two things in which change in one element also affects the other one.

Dependency

## Association:

Association is basically a set of links that connects elements of an UML model. It also describes how many objects are taking part in that relationship.

Association

## Generalization:

Generalization can be defined as a relationship which connects a specialized element with a generalized element. It basically describes inheritance relationship in the world of objects.

Generalization

## Realization:

Realization can be defined as a relationship in which two elements are connected. One element describes some responsibility which is not implemented and the other one implements them. This relationship exists in case of interfaces.

Realization

# 3. UML Diagrams:

There are two broad categories of diagrams

* Structural Diagrams
* Behavioural Diagrams

The structural diagrams represent the static aspect of the system. These static aspects represent those parts of a diagram which forms the main structure and therefore stable.

The four types of structural diagrams are:

* Class diagram
* Object diagram
* Component diagram
* Deployment diagram

Behavioral diagrams basically capture the dynamic aspect of a system. Dynamic aspect can be further described as the changing/moving parts of a system.

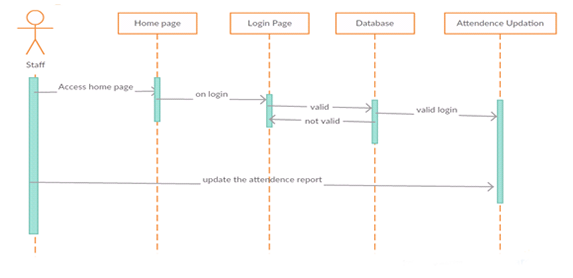
The five types of behavioral diagrams are:

* Use case diagram
* Sequence diagram
* Collaboration diagram
* State chart diagram
* Activity diagram

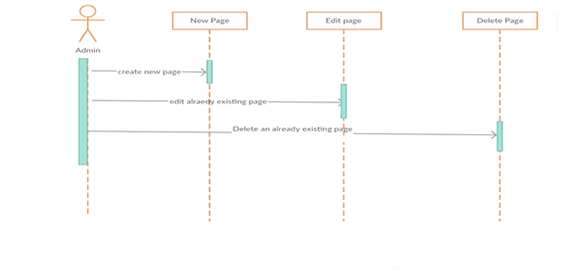
**4.2 Diagrams used in the Project**

**Sequence Diagram:**

A sequence diagram is an interaction diagram that shows how processes operate with one another and in what order. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams, event scenarios.

****

**Figure No: 4.1: Sequence Diagram for Attendance module.**

Figure 4.1 shows the sequence of operations that are to be performed by the staff. The dotted lines represents the time lines these are used to represent the actions performed one after the other.

**Figure No: 4.2: Sequence Diagram for Admin module.**

Figure 4.2 shows sequence of operations that are to be performed by admin . The dotted lines represents time lines these are used to represent the actions performed one after the other.

**Deployment Diagram:**

**Deployment diagram** shows execution architecture of systems that represent the assignment of software [artifacts](http://www.uml-diagrams.org/deployment-diagrams.html#artifact) to deployment targets.

They could be connected through communication paths to create network systems of arbitrary complexity. [Artifacts](http://www.uml-diagrams.org/deployment-diagrams.html#artifact) represent concrete elements in the physical world that are the result of a development process and are deployed on nodes.



**Figure No: 4.3: Deployment diagram**

Figure 4.3 represents the Deployment diagram. In this end user interact with the system and retrieve the results from the server.

**5. SYSTEM TESTING**

**5.1 Introduction to Testing:**

Testing is a process, which reveals errors in the program. It is the major quality measure employed during software development, during testing, the program is executed with a set of test cases and the output of the program for the test cases is evaluated to determine if the program is performing as it is expected to perform.

**5.2 Testing Strategies**

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:

**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. 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 uses to find errors in the following categories:

* Incorrect or missing functions
* Interface errors
* Errors in data structure or external database access
* Performance errors
* 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 and logical decisions are tested on all the cases. It has been uses to generate the test cases.

* Guarantee that all independent paths have been executed.
* Execute all logical decisions on their true and false Sides.
* Execute all loops at their boundaries and within their operational bounds
* Execute internal data structures to ensure their validity.

**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.

**System Testing:**

Involves “in-house testing” of the entire system before delivering to the user. Its aim is to satisfy the user the system meets all requirements of the client’s specifications.

**Acceptance Testing:**

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

**Test Approach:**

Testing can be done in two ways:

* Bottom up approach
* Top down approach

**Bottom up Approach:**

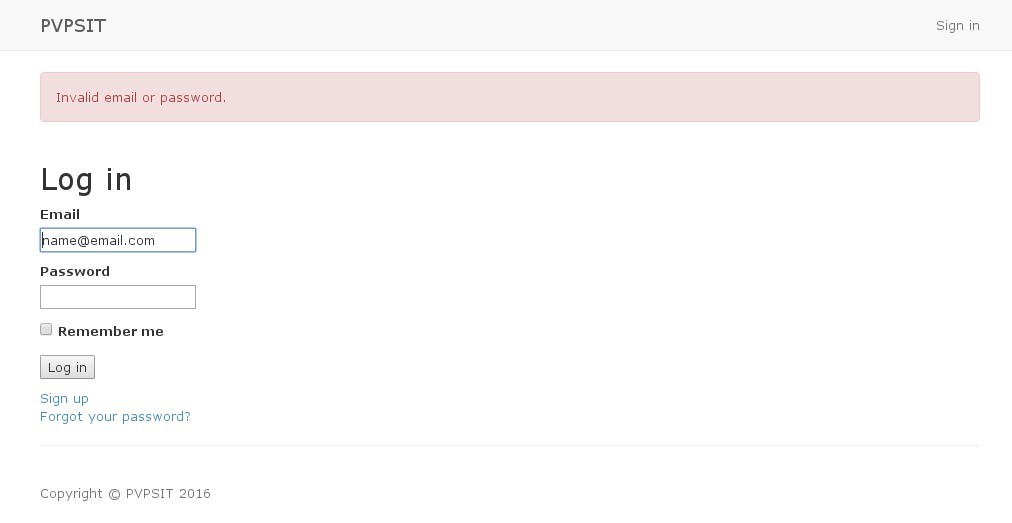
Testing can be performed starting from smallest and lowest level modules and proceeding one at a time. For each module in bottom up testing a short program executes the module and provides the needed data so that the module is asked to perform the way it will when embedded with in larger system. When bottom level modules are tested attention turns to those on the next level that use the lower level ones they are tested individually and then linked with the previously examined lower level modules.

**Top-Down Approach:**

This type of testing starts from upper level modules. Since the detailed activities usually performed in the lower level routines are not provided stubs are written. A stub is a module shell called by upper level module and that when reached properly will return a message to the calling module indicating that proper interaction occurred. No attempt is made to verify the correctness of the lower level module.

**Validation:**

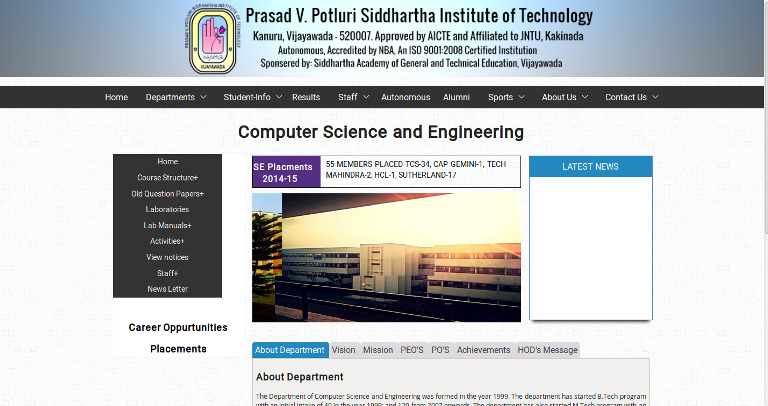
The system has been tested and implemented successfully and thus ensured that all the requirements specification are completely fulfilled. In case of erroneous input corresponding error messages are displayed.



**Figure No: 5.1 Validation in Login of the Admin Module**

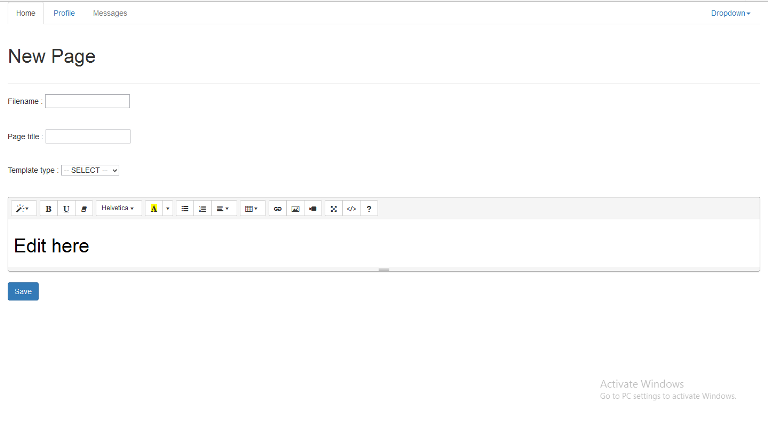
Figure No: 5.1 shows the Admin page login validation whether the password is correct or not. If the password is incorrect it gives an alert.

**6. OUTPUT SCREENSHOTS**

**  
Screenshot No. 6.1: Home Page that has been developed in the User Interface Module.**

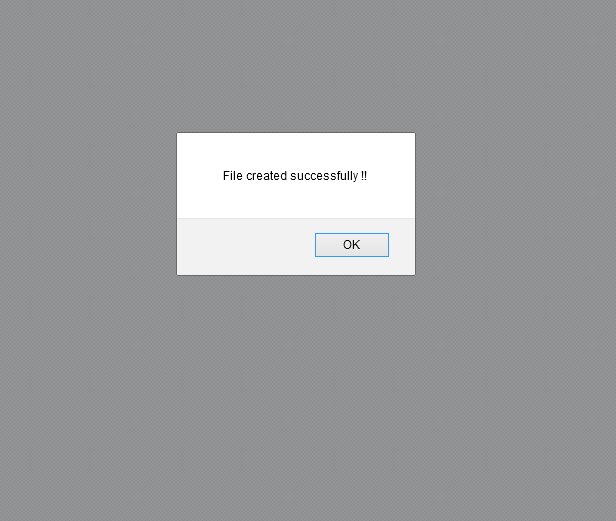
**Screenshot No.6.2: Dept. Home Page that has been developed in the User Interface Module.**

Screenshot 6.1 and 6.2 shows the home page and the department home page developed by the User Interface Module.



**Screenshot No .6.3: Creation of new page using Page editor Module**

The Screenshot 6.3 shows the user interface for the creation pages in which we give the page name, page title and the template to be showed for that page.



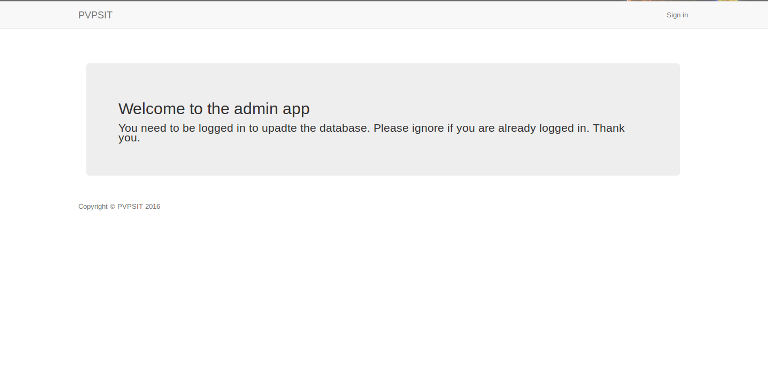
**Screenshot No .6.4: Successful Creation of new page using Page editor Module**

The Screenshot 6.4 shows the successful creation of the page using the page editor module.

****

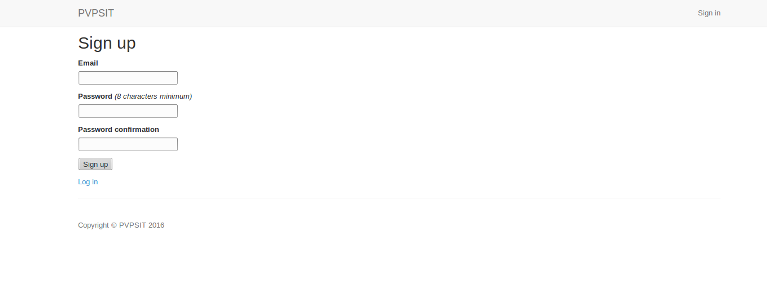
**Screenshot No: 6.5: Accessing the page that has been created by using Page Editor Module.**

Screenshot 6.5 shows the page that has been created by the Page Editor module. Here there will be two templates according to which one template has the navigation bar on the left side and the other which has no menus.

****

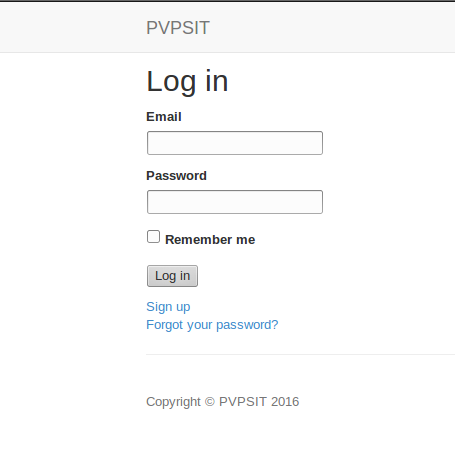
**Screenshot No: 6.6: Home Page for the creation of Admin Module.**

Screenshot 6.6 shows the main home page for the Admin module which is used for the creation of the News and Events, Editing them.



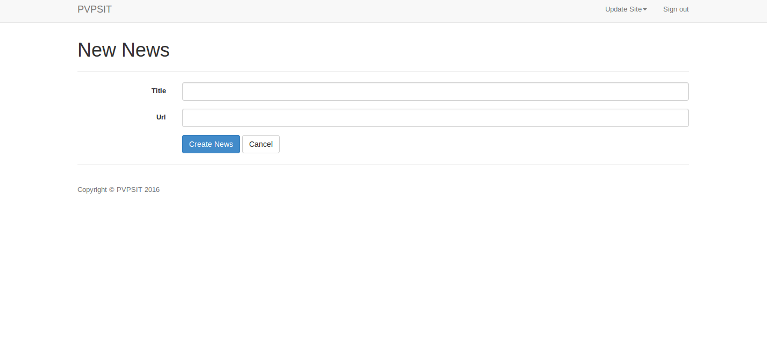
**Screenshot No: 6.7: Sign Up Page for the Admin Module.**

Screenshot 6.7 shows the Sign Up page for the Admin Module with the help of which the Admin present will register into the admin module.



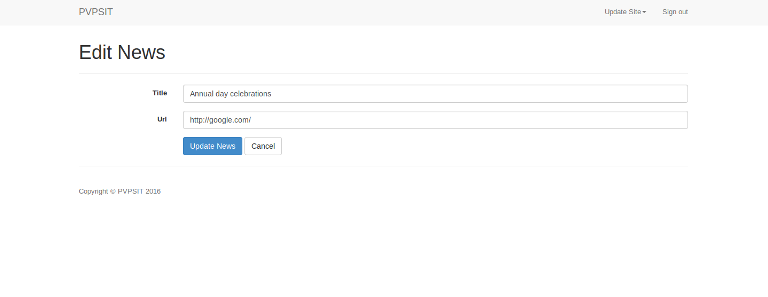
**Screenshot No: 6.8: Login Page for the Admin Module.**

Screenshot 6.8 shows the Login page for the Admin through which admin logins into the module.



**Screenshot No: 6.9: New News creation with the Admin module.**

Screenshot 6.9 shows the creation of new News and in this we give the news a title and the URL to be loaded when the title has been clicked.



**Screenshot No: 6.10: Editing News created with the Admin module.**

Screenshot 6.10 shows the Editing of the News that are already present. We can Edit the news by giving the title and the URL can be changed if we want to edit it.

**7. CONCLUSION & FUTURE SCOPE**

Finally, the project titled "Recuperation of PVPSIT website" has an Admin Module which is used by the Department Administrator for the updating the News and Events, Notices. A Page Editor Module is used for creating pages easily. The User Interface developed provides rich user interaction which is nothing but more user-friendly, when compared to that of in the existing system. Due to implementing the code in PHP, without using any template engine the website is swift when compared to the existing one. This project avoids errors that are occurring currently due to multiple submission of attendance.

**FUTURE SCOPE:**

In the field of Computers, there will be continuous evolution. One technology present today will become obsolete the next day. Hence the website can be updated according to the technologies that may offer a wide range of features, which can enhance the present system.

**8. REFERNCES**

[1]. Head First PHP & MySQL by Lynn Beighley and Michael Morrison.

[2]. PHP and MySQL Web Development (5th Edition) by Luke Welling and Laura Thomson.

[3]. MySQL Cookbook: Solutions for Database Developers and Administrators by Paul DuBois.

[4]. Web Application Design & Implementation: PHP5, MySQL, PHP, by Steven A. Gabarro.

[5]. Building Web Apps with WordPress by Brian Messenlehner and Jason Coleman.

[6]. PHP Script & JQuery: Interactive Front-End Web Development by Jon Duckett.

[7]. Advanced Guide to PHP on IBM i by Kevin Schroeder.

[8]. Beginning PHP and MySQL by [W. Jason Gilmore](http://it-ebooks.info/author/1446/).

[9]. MYSQL by Paul DuBois-4th edition.

[10]. Head First PHP & MySQL by Lynn Beighley and Michael Morrison.

[11]. PHP and MySQL Web Development by Luke Welling and Laura Thomson-5th Edition.

[12]. MySQL Cookbook: Solutions for Database Developers and Administrators by DuBois.

[13]. PHP With MySQL by Mr. Rishi Ahluvalia.

[14]. Beginning PHP and MySQL by W. Jason Gilmore.

[15]. Php 6 and MYSQL 5 for Dynamic Web Sites Paperback – 2008 by Larry Ullman

[16].php-5-power-programming by Stig Bakken

[17]. http://w3schools.com/

[18]. http://tutorialspoint.com/

[19]. www.phpreferencebook.com/

[20]. www.headfirstlabs.com/books/hfphp