ADVANCED ATTENDANCE SYSTEM

Winter Training Project (2013-2014, NSIT)

PROJECT BY :

AKANSHI GUPTA (216/CO/12) | CHITRASOMA SINGH (254/CO/12) | DIVJOT SINGH (262/CO/12)

COE – 1, (2012-2016)

Contents

Acknowledgment

We would like to acknowledge with deep appreciation and gratitude the invaluable help of following persons:

- Dr. (Ms) Ritu Sibal, Assistant Professor, Division of Computer Engineering / Information Technology, NSIT
- Ms. Veenu , Assistant Professor, Division of Computer Engineering / Information Technology, NSIT

We are highly indebted to the above for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

Our thanks and appreciations also go to our colleagues in developing the project and people who have willingly helped us out with their abilities.

Introduction

1 ABSTRACT

The project is meant to create an advanced attendance taking system to help teachers, students and college administration by automating the entire process. It is designed keeping scalability and code-reusability in mind, which means that the same project can be altered by changing few variables to get desired results. Furthermore, there's scope of adding more functionality without disturbing any of the existing one. This is made possible by using Object Oriented Programming, a Modular for designing web service, and following large parts of MVC model. This allows us to extend the project to mobile applications and other environments. The web service is built on Linux Operating System with

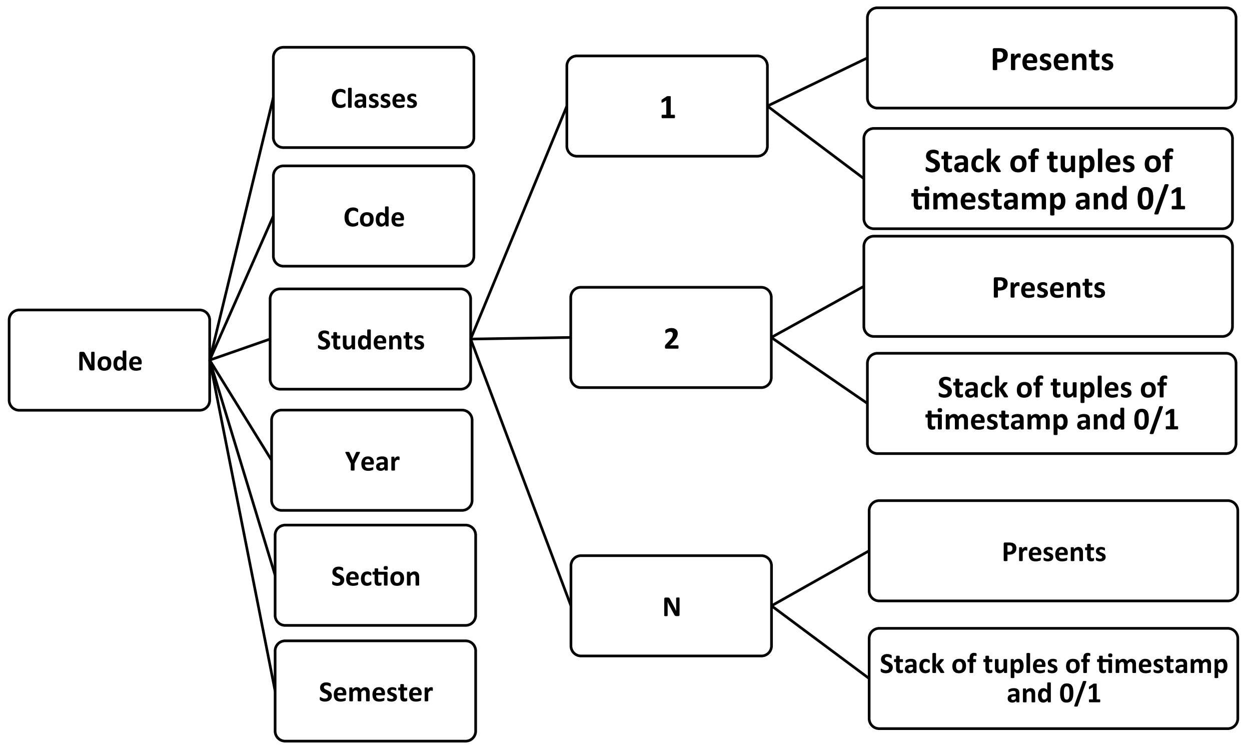
- Apache as web server
- MySQL as database management language
- PHP myAdmin as database manager
- PHP as backend programming language
- JavaScript as frontend programming language
- BootStrap as frontend framework
- jQuery as JavaScript framework for better functionality (AJAX, DOM Traversal)

2 CODE STRUCTURE

Entire project is divided in 4 main parts:

1. Node_class (**Model** , located at */php/node_class.php*)
2. Helper PHP Files (**Controller** , located in */php/*)
3. JavaScript Files (**For Frontend Programming** , located in */js/*)
4. PHP Files (**Views** , located in root of project folder */*)

3 DATA STRUCTURE


Node
Classes
Code
Students
Year
Section
Semester
1
2
N
Presents
Stack of tuples of timestamp and 0/1
Presents
Stack of tuples of timestamp and 0/1
Presents
Stack of tuples of timestamp and 0/1

This is the data structure used in **node_class.php.** It is a complex data structure making use of hash maps, stacks and linked lists. It helps us to save the data in a much efficient manner, allowing faster operations of search , modify and access.

Model (*/php/node_class.php*)

1 PROPERTIES

1. **rollNumberRegex -** Regex constant used to deal with roll numbers

2. **teacher –** Teacher’s unique ID from ‘teacher’ table in Database

3. **subjectCode –** Subject Code of a particular class

4. **section –** Section of a particular class

5. **year –** Year of a particular class

6. **semester –** Subject Code of a particular class

7. **numberOfDays –** Number of days class is conducted up till now

8. **records** – Data Structure holding all the data regarding attendance

2 METHODS

Constructor

__construct7($code,$teacher_uid,$year,$semester,$section,$start,$end)

Takes required arguments to create a new object of the class. Processes them and finally saves into the database.

Helper Functions

public function retrieveObject($code,$section,$year)

Takes SubjectCode , Section , Year. Returns a new Object of Node class based on code , section and year of a class. Throws false for any kind of error.

public function retrieveObjecti($class_id,$teacher_uid)

Takes Class Unique ID from objects table , Teacher Unique ID from teacher table. Returns a new Object of Node class based on code , section and year of a class. Throws false for any kind of error.

public function initRecords($start,$end)

Takes starting and ending roll number. Initializes the records. It will update the records property based on starting and ending roll number.Throws false for any kind of error.

public function deleteNode()

Deletes the node from data base.Returns true if deleted , false in case of any error

public function saveNode()

Saves the node into data base , it also inserts if the object isn't present in database.Returns true if saved , false in case of any error

public function isPresent($rollNumber,$newPresents)

Takes Roll number and new number of present days. Tells if that student was present or not. Returns 1 if present , 0 if absent , false if not found

public function deleteRoll($rollNumber)

Takes Roll number. Deletes a roll number. Returns true if deleted , false if not found

Getters

public function getTeacherID()

Returns teacher unique ID

public function getTeacherName()

Returns teacher name by searching in data base

public function getCode()

Returns class code

public function getYear()

Returns year of class

public function getSemester()

Returns semester of class

public function getSection()

Returns section of class

public function getDays()

Returns days conducted by teacher for the class

public function getPercent($rollNumber)

Takes roll number. Returns Computers the percentage and returns it. Throws False if not found

public function getTimeline($rollNumber)

Takes roll number. Returns the timeline for that roll number. Throws False if not found.

public function getRecords()

Returns the records for entire object

Setters

public function setTeacher($val)

Sets the teacher unique ID

public function setCode($val)

Sets the subject code of the class

public function setYear($val)

Sets the year of the class

public function setSemester($val)

Sets the semester of the class

public function setSection($val)

Sets the section of the class

public function setDays($val)

Sets the number of days the classes is conducted

public function setPresence($rollNumber,$newPresents,$timestamp)

Takes Roll Number, New Presents, Timestamp. Returns False in case of error

Controllers (*/php*)

1 ADD_CLASS.PHP

Add a class with the data provided to the database.

2 DELETE_CLASS.PHP

Deletes a class on the basis of given parameters.

3 DELETE_ROLL.PHP

Deletes a roll number of a class using methods of node_class.

4 UPDATE_PROFILE.PHP

Validates the details and helps in updating profile of a user.

5 UPDATE_CLASS.PHP

Updates a class' details using the methods of node_class.

6 MARK_ATTENDANCE.PHP

Checks for the existence of the class id for that teacher's id and then saves the attendance.

7 GET_ATTENDANCE.PHP

Generates the timeline for the student and returns all the data based on given parameters.

8 PROCESS_SIGNUP.PHP

Validates data and helps in signup of a new user.

9 PROCESS_LOGIN.PHP

Matches the logged in information with the one present in the database, if match found then it starts the session with the credentials fetched from the database.

Miscellaneous PHP files

LOGOUT.PHP

Destroys the session to logout the user.

SESSION.PHP

A php file to debug and find what all is stored in session. Used only for testing purposes.

DEFINES.PHP

It contains various helper functions for easy operation. This is a part of modular programming used throughout the project.

function connectTo()

Connects to data base. Returns MySQLi Connection object.

function sqlReady($input)

Takes any string and returns the escaped form of it to prevent sql injection.

function hashPass($pass,$rounds = 9)

Takes any string and returns the hashed string using blow-fish algorithm.

function verifyPass($input,$pass)

Verifies if the a password (input) matches the hashed (pass) password. Returns true or false accordingly.

function respond($as,$what)

A helper function to return a JSON string using key and value pair of (as,what). While returning , it also *dies()* the process.

function updateSession($email)

Updates the session variable for a given email id with refreshed data from database

function verify($type,$input)

A helper function to perform various regex operations. Return true if the pattern is matched and false if not. (type can have any of following – EMAIL,PHONE,NAME,CODE,ROLL,NUMBER)

Views (/)

1 INDEX.PHP

The home page for the project. Allows a faculty to sign up or login. Redirects a student to student dashboard.

2 STUDENT.PHP

Student dashboard. Allows any student to check his/her attendance records for any given class.

3 TEACHER.PHP

Teacher dashboard. Allows teacher's to add more classes and to visit (redirects to take.php) any class to take attendance for the same. Teacher can also delete the present classes from this page.

4 PROFILE.PHP

Allows a teacher to edit profile details.

5 CLASS.PHP

Displays list of classes and allows teacher to edit its details accordingly.

6 STATISTICS.PHP

Shows a teacher average attendance per class and list of students with short attendance.

7 TAKE.PHP

This page allows a teacher to take attendance for a given class. He/she can also delete a student from that class (drop/year back and other cases).

JavaScript (*/js/*)

1 CLASS.JS

Manages all the user interactions made in **class.php** file , validates all the data entries and finally upon saving the details makes an AJAX request to **php/update_class.php** to do the same.

2 LOGIN.JS

Manages all the user interactions made in **index.php** file, validates all the data entries and makes AJAX requests to **php/process_login.php** in case of a login and **php/process_signup.php** in case of signup.

3 PROFILE.JS

Manages all the user interactions made in **profile.php** file, validates all the data entries and finally upon saving the details makes an AJAX request to **php/update_profile.php** to do the same.

4 STATISTICS.JS

Manages all the user interactions made in **statistics.php** file, and shows graphs and short attendance students' lists.

5 STUDENT.JS

Manages all the user interactions made in **student.php** file, validates all the data entries and finally uses the details to make an AJAX request to **php/get_attendance.php** to generate graphs and pie chart of his/her attendance records.

6 TAKE.JS

Manages all the user interactions made in **take.php** file, validates all the data entries and finally upon saving the details makes an AJAX request to **php/mark_attendance.php** to do the same or to **php/delete_roll.php** to delete a student from the class.

7 TEACHER.JS

Manages all the user interactions made in **teacher.php** file, validates all the data entries and finally upon saving the details makes an AJAX request to **php/add_class.php** to add a new class or **php/delete_class.php** to delete one.

Future Additions

1 Email teacher with monthly reports for each of his/her class

With the help of an **SMTP** server, we can further email the teacher with monthly reports.

2 Develop an assisting mobile application with offline support

Since the project is very modular in nature, we can extend its capabilities to mobile applications and allow a teacher to take the attendance without availability of internet and push his records whenever feasible.

3 The data can be used in several other ways

As the project is put into usage, a lot of data will be gathered allowing any student to find which teacher takes which subjects and how attendance records are in his/her classes. Similarly a student's records can tell us a lot about how attentive that student is and can take an action accordingly. Furthermore a general trend can be found with the data of average attendance and courses, showing which classes are of interest for students and/or which teacher in particular is of interest.