DICT Leave Credit Computation

Technical Documentation

Tyrel Justin A. Dogup

Paul Jeremiah D. Naraval

Overview

This project was made to keep track of employees' leave records and compute their leave credit balance on a given date. Additionally, this project includes a calendar wherein the user can set holidays and suspended work days which are both taken into account for the computation of leave credits balance. This project is a website requiring Apache and MariaDB/MySQL.

Frameworks and Libraries Used

PHP

- CodeIgniter 3.1.9
- CodeIgniter Ion Auth 2

HTML and CSS

- BootStrap 4.1.1
- FontAwesome 5.0.13

JavaScript

- AngularJS 1.7.0
- Moment.js 2.22.2
- Chart.js 2.7.2
- Angular-Chart.js 1.1.1
- JQuery 3.3.1

This website uses the CodeIgniter PHP framework for its backend logic and AngularJS JavaScript framework for its frontend logic.

Configuration

- Before accessing the website, the database connection information must be set at /application/config/database.php
- The URL of the site can be changed at /application/config/config.php using the base_url configuration item.
- Each of the database tables generated by the website is prefixed by the value of the DB_PREFIX constant located in /application/config/constants.php.
- If the employee database table is taken from an external source, by default the user can't change Employee information. To enable this, go to /application/config/config.php and change the value of employee_editable_if_external.

Program Implementation

For additional information regarding the classes and function described below, refer to the comments found in the source code.

CodeIgniter

CodeIgniter is based on the MVC (Model-View-Controller) architectural pattern wherein the 'model' component is in charge of storing data, the 'view' for user interface, and the 'controller' for acting as an intermediary between the two as well as displaying 'views' to the user. The models, views, and controllers of this project are stored within /application/ in their respective folders. Every controller extends the *MY_Controller* class and every model extends the *MY_Model* class, which are both located in /application/core/.

Core Classes (/application/core/):

- *MY_Controller.php*
  - Loads the authentication plugin (IonAuth) of the website.
  - Also loads the *angular_communication_helper* found in /application/helpers (refer to Communication Between CodeIgniter and AngularJS)
  - Functions:
    a. *checkLogin* - checks if the user is logged-in and is the admin of the website. (Non-admin users were not implemented.)
    b. *index* - displays the header and footer views around the result of the 'body' function.
    c. *body* - meant to be overridden. Child classes should display the body of their respective pages in here.
    d. *html* - used for displaying subpages. (<base_website_url>/<controller-name>/<sub-page>: e.g given http://www.localhost.com/calendar/manageCollisions, 'manageCollisions' is a subpage of calendar)
- *MY_Model.php*
  - Loads the database libraries of CodeIgniter
  - Functions:
    a. *checkFields*($checker,$checkee) - *$checkee* is filtered using data described by *$checker* (refer to the private variables defined in

application/models/employee_leaves_model.php). This is used to filter unwanted data contained in *$checkee*.

Controllers: (/application/controllers/)

A controller is called when a user adds the controller's name as a segment to the base URL of the site. (E.g. given a base URL of http://www.localhost.com/, Employee.php is called when the user accesses http://www.localhost.com/employee. The functions of Employee.php are called using http://www.localhost.com/employee/functionName). The controller's *index* function is called when nothing is supplied. The default controller is *Main.php*.

- *Main.php*
  - Controls the default page of the website. Checks first if the user is logged in. If so, it displays the homepage, otherwise it displays the login page.
- *Calendar.php*
  - Manages the calendar components of the website.
- *DB.php*
  - This controller is in charge of initializing the database of the website.
- *Employee.php*
  - Contains the employee-related logic of the website.
  - In charge of addition, deletion, and editing of employee information as well as the management of leave records.

Models: (/application/models/)

Each database table generated by this site is prefixed with the DB_PREFIX constant found in /application/config/constants.php

- *Employee_Leaves_Model.php*
  - Creates and manages the database tables for the employees, employee-leave connector, and leave records.

    - The employee table is used for storing the employees' personal info, while the employee-leave connector table is used for storing an employee's info that are relevant to the computation of his leave credits.
    - The 'employee_table_meta' table is for storing the metadata of the employee table. This is needed because the user can define an external employee table at initialization.
- *Calendar_Model.php*
    - Creates and manages the database tables for calendar events and calendar collisions (Calendar collisions are suspended work days that overlapped with an employee's leave)

Views: (/application/views/)

Most of the views are backed by AngularJS. While the backend is in charge of storing data, AngularJS does the computations and formatting of these data.

- *header.php* (ng-app directive is set here; uses the 'initializer' and 'login' controller), *footer.php*, *breadcrumbs.php*
    - Respectively displays the header, footer, and breadcrumbs of the website.
- *welcome_message.php*
    - Displays the homepage of the website.
- *login.php* (uses the 'login' controller)
    - Login page. Gets displayed if the user is not logged-in.
- *permission_error.php*
    - Displays a permission error message. This is unused because the only registered user of the site is the admin.
- *db_init.php* (uses the 'init_db' controller)
    - Displays the database initialization form. Here the user can set the metadata of an external employee table, or initialize the internal employee table.
- admin.php (uses the 'admin' controller)
    - In here, the admin can change his credentials.
- */employee/navigation.php* (uses the 'employee_nav' controller)

    - Default subpage for the Employees page. Displays the list of employees as well as a search bar.
- */employee/add.php* (uses the 'employee_add' controller)
    - Displays the form for adding employee records. Inaccessible by default if the employee table is external. (Refer to Configuration section)
- */employee/display.php* (uses the 'employee_display' and 'employee_statistics' controllers)
    - Displays a selected employee. Contains the main component of the website. In here, the admin can edit the employee's information and manage and print his leave records
- */employee/leave_records.php* (uses the 'employee_leave_records' controller)
    - Requires to be called within *display.php*. Contains the form for the addition, editing, and deletion of leave records.
- */calendar/base.php* (uses the 'calendar_display' controller)
    - Displays the calendar. The user can add, edit, or delete calendar events here.
- */calendar/collisions.php* (uses the 'calendar_collisions' controller)
    - Displays the leave records that overlapped with suspended work days. In here the user can manage these collisions.
- */calendar/holiday.php* (uses the 'event_display' controller)
    - In here, the user can manage holiday events. In contrast to */calendar/collisions.php*, this view displays holiday events as a list.
- */calendar/suspend.php* (uses the 'event_display' controller)
    - In here, the user can manage suspended work days. In contrast to */calendar/collisions.php*, this view displays suspended work days as a list.

AngularJS

It is recommended that the reader knows the basics of AngularJS before reading this part.

- /js/app.js
    - Contains the initialization of the site's module.
    - Initialization: ('app.run(function($rootScope,$http,$httpParamSerializer,$timeout){...}')
        - In this block of code, all global variables and functions are stored in the root scope.

- Global Variables:
    - *dateFormat* - refers to the date format used for displaying date data.
    - *busy* - refers to whether the website is loading or handling some computation.
    - *customModalData* - A global modal is defined in the application for displaying dialog to the user. *customModalData* is the object that contains the data of this modal.
    - *minuteCreditTable* and *hourCreditTable* - stores time-credit equivalence as defined by Omnibus. Minute to credit equivalence cannot be computed because it sometimes does not equal the Omnibus-defined equivalence. Hour to credit equivalence table is defined for consistency.
    - *moment* - A global variable for Moment.js. This is defined so that Moment.js functions can be used in HTML.
- Global Functions
    - showCustomModal($header, $body, $onConfirm, $onClose, $confirmName, $closeName)
        - When called, this function sets the global *customModal* data. $header is for the header of the modal and $body for the body. $*onConfirm* is a function that is called when the modal is confirmed, $*onClose* is a function that is called when the modal is closed. $*confirmName* and $*closeName* respectively defines the button names for confirming and closing the modal.
    - post($url,$inputData,$onSuccess,$onFailure)
        - A wrapper function for posting data to the server.
        - $*url* defines the URL for posting, $*inputData* is the data to be posted, $*onSuccess* is a function that is called if the posting returns no errors, and $*onFailure* is a function that is called if there are errors. (Refer to Communication Between CodeIgniter and AngularJS)
    - -longComputation($scope,$field,$valueGiver,$args)
        - This function is used for computations that take a significant time to complete. It asynchronously assigns the result of a function to a given variable. While the computation is being done, this function displays the loading screen.
        - Since JavaScript doesn't allow pass-by-reference in a strict sense, this function requires the scope ($scope) in which the variable ($field) is

defined. The variable's value is given by the function $*valueGiver* and its parameters $*args*.

- -creditsToTime($credits)
  - Converts leave credits ($*credits*) to time. Returns an object containing the fields 'hours' and 'minutes'
- -timeToCredits($hours,$minutes)
  - Converts $*hours* and $*minutes* to leave credits.

- Controllers:
  - *initializer* - adds the base URL of the site to the root scope, as well as the CSRF token name and hash. (The CSRF token name and hash are used for the Cross-Site Request Forgery protection of the site)
  - *login* – handles the logging in and out of the user.
  - *admin* – handles the changing of the admin's credentials.
  - *init_db* – handles the initialization of the sites database.

- /js/app_employee.js
  - Controllers:
    - *employee_nav* – manages the displaying of the Employees homepage.
    - *employee_add* – handles the addition of new employee records.
    - *employee_display* – contains the algorithm for calculating leave credits balance. Also handles the displaying of an employee's information.
    - *employee_statistics* – child controller of *employee_display* that manages the graphing of an employee's leave credits balance over time.
    - *employee_leave_records* – handles the addition, editing, and deletion of leave records.
- /js/app_calendar.js
  - Controllers:
    - *calendar_display* – handles the displaying of the Calendar homepage as well as the management of calendar events.
    - *event_display* – another controller that handles the management of calendar events.
    - *calendar_collisions* – handles the displaying and resolution of calendar collisions.

Communication between CodeIgniter and AngularJS.

Angular to CodeIgniter

Input by the user to the website's forms are stored as JavaScript objects in AngularJS. In this project, the CSRF token name and hash (initialized by the 'initializer' controller) and the user input data are wrapped in an object. The CSRF hash is included in order to get past CodeIgniter's security against Cross-Site Request Forgery attacks. This object is then encoded as form data for posting to CodeIgniter. Once the input data is received in the CodeIgniter, it is converted to an associative array using the function *parse_custom_post* from /application/helpers/angular_communication_helper.php/.

CodeIgniter to Angular

When the website loads a page, it passes relevant data to the view and then to AngularJS using the 'ng-init' directive. Some of these data are first encoded as JSON objects before it is sent to AngularJS. When AngularJS posts data to CodeIgniter, the function *custom_response* from /application/helpers/angular_communication_helper.php/ is used to send a response back to AngularJS.

Future Recommendations

- The project was only designed to accommodate one user which is the admin. If this project is to be scaled for larger businesses, it might be imperative to have more users and have multiple privilege tiers. This way, an intern, for example, can help with the management of leave records while not having permission to edit employee information.
- If more users are allowed to access the website, query caching might be required in order the speed-up its performance.
- An auto-backup mechanism is also recommended so as to mitigate data loss.