383 Final Group Project

You will each select a partner and complete the following assignment as a group.

Register Your Group:

1. Go to the groups tab in the People Page.
2. Pick a group that is not already in use.
3. Register yourself in the group with your teammate.
4. Each group will be 2 people.
5. Instructors will randomly assign people who have not selected a group on Thursday Afternoon.

Implementation Details:

- Pick a reasonable name for your project, make sure the name is "appropriate"
- Names must not contain spaces or special characters
- One team member create a git repository
    - Make the 2nd team member a full owner
    - Make the following people developers
        - liangs9
        - campbest
- Each team member shall clone the project to ceclnx01.
- There are two assignments that you must submit:
    - There is a GROUP assignment which is the actual code and project.
        - You will submit ONCE for this assignment and both team members will be given a shared grade for this submission.
    - Individual Writeup
        - Each team member will independently submit a writeup directly into canvas that answers the following questions:
            - What was your primary contributions to this project?
            - Provide an annotated pseudo code for the REST.php program.
            - Write a short paragraph about the major obstacles you had in completing this assignment.
            - Percentage wise, what percent of this projects work was yours.
            - What grade would you give yourself (Very Good, Good, OK, Poor)
            - What grade would you give your team member (Very Good, Good, Ok, Poor)
            - Team Member A:

        - Provide a link to a video lasting no more than 2 minutes of you demonstrating error handling in your project.
    - Team Member B (2nd team member)
        - Provide a link to a video lasting no more than 2 minutes of you demonstrating proper operation of your project and its features.

Overview:

Create a single page web application that monitors a person's consumption of a set of items in a diary form. The list of items being monitored and the list of authorized users will both be fixed in a database. Only the diary entries will be dynamically added to the database.

The application flow shall be as follows:

1) User shall connect to the application/index.html.
    a) ALL html code shall be in this file
    b) all javascript shall be in a separate file.
    c) All css shall also be in a separate file
2) User shall be prompted to authenticate.
    a) I have created user entries in the user table for each class member. All password are the same "test"
    b) Using JSON the users credentials shall be sent to the rest api to obtain a token
3) Once properly authenticated the user shall be shown
    a) A summary of their Diary
    b) The last 20 entries of their diary
    c) A well formatted set of buttons allowing them to indicate they consumed one of the tracked items.
        - The items list is obtained from making an api call
        - upon clicking a button, a JSON call shall be made to api which will update their diary with the item and the date/time.
        - The page will then update its entries via javascript
4) misc:
    a) you must always use prepare statements where appropriate
5) REST API
    a) Get Token
        - Given user and password will get a token validating the user. If no user is present or the password does not match will return status will == "FAIL"
        - passwords are hashed using the php password_hash function
        - url: rest.php/v1/user
        - method: post
        - json_in:
            - user

        - password
    - json_out
        - status: "OK" or "FAIL"
        - msg:
        - token: string
    - Test:
        - curl -X 'POST' -d '{"user":"test","password":"test"}' https://ceclnx01.cec.miamioh.edu/~campbest/cse383/finalProject/restFinal.php/v1/user

b) Get list of items
    - Return the set of items we are tracking and their key
    - rest.php/v1/items
    - method: get
    - json_in: none
    - json_out:
        - status
        - msg
        - items[]
            - pk
            - item
    - test:
        - https://ceclnx01.cec.miamioh.edu/~campbest/cse383/finalProject/restFinal.php/v1/items
        - curl https://ceclnx01.cec.miamioh.edu/~campbest/cse383/finalProject/restFinal.php/v1/items

c) Get Items user consumed
    - rest.php/items/token
    - Call gets the tracked items for a given user
    - limit to last 30 items
    - Method: GET
    - JSON Response:
        - status: OK or AUTH_FAIL or FAIL
        - msg: text
        - items[]
            - pk
            - item
            - timestamp
    - test
        - https://ceclnx01.cec.miamioh.edu/~campbest/cse383/finalProject/restFinal.php/v1/items/1db4342013a7c7793edd72c249893a6a095bca71

d) Get Summary of Items

        - rest.php/v1/itemsSummary/token
        - method: GET
        - json_in: none
        - json_out
            - status
            - msg
            - items[]
                - item
                - count
        - test
            - https://ceclnx01.cec.miamioh.edu/~campbest/cse383/finalProject/restFinal.php/v1/itemsSummary/1db4342013a7c7793edd72c249893a6a095bca71
    e) Update Items Consumed
        - rest.php/v1/items
        - Updates item as being consumed.
        - method: post
        - JSON IN
            - token: string token
            - ItemFK: <key>
        - JSON OUT
            - status: OK or AUTH_FAIL or FAIL
            - msg: text
        - test
            - curl -X 'POST' -d '{"token":"1db4342013a7c7793edd72c249893a6a095bca71","itemFK":2}' https://ceclnx01.cec.miamioh.edu/~campbest/cse383/finalProject/restFinal.php/v1/items

Database:

- users: user table
    - pk
    - user
    - password
    - timestamp
- diary: Item Entries
    - pk
    - userFK -> foreign Key to user - not the user but the pk of the user
    - itemFK -> foreign Key to item. Not the item but the PK of the item
    - timestamp
- diaryItems: list of items

    - pk: int
    - item: tinytext
- tokens
    - pk
    - use - actual user string
    - token - token string created randomly
    - timestamp

You must use a php based datamodel file separate from your rest code.

feel free to explore my restpapi at
http://ceclnx01.cec.miamioh.edu/~campbest/cse383/finalProject/restFinal.php

Notes:

SQL Statements

I used two more advanced sql statements in my solution:

1st: Get List of items for user
to get a list of items for a user, I used a join to combine the diary table and the diaryItems table.

select diaryItems.item,timestamp from diaryItems left join diary on diaryItems.pk=diary.itemFK where userFK=? order by timestamp desc"

This statement combines the two tables, doing the "join" on diaryItems.pk = diary.item. Then I was able to get the name of the item and the timestamp in one call instead of having to do two calls.

2nd:
To get the summary of items, I used a "group by"
select diaryItems.item,count(timestamp) as count from diaryItems left join diary on diaryItems.pk=diary.itemFK where userFK=? group by diaryItems.item

This provides a list of items and the count of each item.

Button "trick"

In my buttons I used the following code to create the buttons:
<button class="itemButton" pk=4 onclick='recordItem(this)'>Milk</button>

I created similar button code like that above for each button, replacing the pk and name for each button.
Then in my code that handles the onclick event I looked up the PK so I could have it to make the ajax call.

```
function record(whichButton) {
  var itemPK = $(whichButton).attr('pk');
```

$(whichButton) will hold the button that is clicked.
$(whichButton).attr('pk') will get the value of the attriube called pk for the current button.
Now we know which button was clicked and can send this to the rest-server in our ajax code.

Of course I created the buttons dynamically from the list of items retrieved from the rest server.

Tests

- The following tests should all pass
- Copy this to a file on ceclnx01 - http://ceclnx01.cec.miamioh.edu/~campbest/cse383/finalProject/final-tests.txt
- Edit the field UNIQUEID
- Make it executable (chmod a+x final-tests.txt)
- run it "./final-tests.txt"
- Look at each line - tells you what output should look like