Collaborative Online Forum for Filipino Experts Everywhere

Emil John Lopez
De La Salle University
Manila, Philippines
emil_lopez@dlsu.edu.ph

***Abstract*— The number of sites Filipinos can ask other Filipino experts about very specific fields is low. The Collaborative Online Forum for Filipino Experts Everywhere website or COFFEE aims to be a bridge between curious-minded Filipinos and local experts that can answer their questions about Filipino-specific things. This can include topics such as Filipino history, agribusiness, Filipino literature, and much more. The site will be divided into categories that encompass these large fields. People who wish to answer and have the expertise to do so can register their credentials along with evidence that proves it (URL or file upload) and they will be branded as experts when they answer in that category. The system mainly uses PHP as its scripting language of choice and MySQL as its database system. It primarily uses three databases that contain a total of twelve (12) tables. The first database is for storing information about user expertise credentials, the second is for keeping track of the users and moderators in the site, and the third is for storing the contents of the site itself such as posts and votes. The PHP-based website interacts with the website to create different functionalities such as posting, answering questions, adding new categories/topics/posts, registering credentials, and moderating the site by assessing credentials and deleting bad answers. There are still many aspects of the website that needs improvement overall. The developer suggests improving the user interface, adding more ways of moderating content such as a report post system, adding a feed that recommends based on interest, and adding a notification system. Overall, COFFEE does many of its basic tasks accurately and it can be used by Filipinos to share knowledge in their expertise.**

Keywords—PHP, SQL, forum, expert, credential, vote

I. Introduction

Unlike in most other countries, the number of places Filipinos can ask local experts about their highly technical questions is very sparse. Although some sites like Quora have Filipino experts answering in their site, questions and discussions about more obscure or specific fields of interest like Filipino history, the Filipino language, navigating the Filipino bureaucracy, and agribusiness are often non-existent except in specific threads on social media sites. The *Collaborative Online Forum for Filipino Experts Everywhere* or *COFFEE* website aims to connect the general populace to these Filipino experts.

To concentrate categories of information into sections, this website will be designed like an online forum, where people can go to their selected category, choose a topic, then ask their question about that specific topic. Users of the website can set their credentials on a specific field or category by selecting or creating a credential that corresponds to that field and uploading evidence that proves their credential. When a person is a certified expert in a category, it will show up in their answers. Moderators assigned to specific categories will oversee the approval of these credentials and the sanitation of their respective categories by deleting inappropriate posts. There will also be a voting system where people can upvote or downvote an answer to show their approval or disdain toward the content of the post. The original asker of the question, like in other Q&A forums, can mark a post as the answer they were looking for.

Just like any online forum, COFFEE will strive to be a place for trading ideas, information, tricks, and tips [1]. Most information will be traded via BB code-formatted text. Unlike other contemporaries such as Pinoy Exchange that focuses more on general information and pop culture, this website wants to strike a more subtle demographic of curious, like-minded Filipino enthusiasts, academics, and experts. In essence, it aims to be a Quora-like website that focuses on topic areas members are experts in and attracts real names whose credentials hold verifiable weight to answer [2].

II. Related Literature

Internet forums are websites that allow people to trade information about specific fields or topics. It gives its users a way to communicate their questions and give answers to people who need it. They generally have some form of moderation to keep the website appropriate. Overall, they are functionally mini-portals on specific topics of interest [3]. COFFEE aims to be one such site for Filipino experts that do not have a medium to share their knowledge and expertise to the Filipino populace. Forums can be classified as anonymous or registration-based [3]. Since COFFEE relies on highly trained experts that have a lot of credibility, the website itself cannot be anonymous and it should display as much information about the people answering to give some accountability to people when answering so that they will do it responsibly. COFFEE takes a lot of inspiration from various forums, and picks some of the important elements they possess.

COFFEE's interface is largely derived from the Filipino-based forum PinoyExchange, where they express opinions regarding local and international issues, updates on showbiz, and other pop culture topics. [4] Coincidentally, PinoyExchange organizes its topics in the same structure as COFFEE, but this is largely coincidental. However, since they

already lined up in terms of content, it was decided by the developer that it is possible to replicate the interface format.

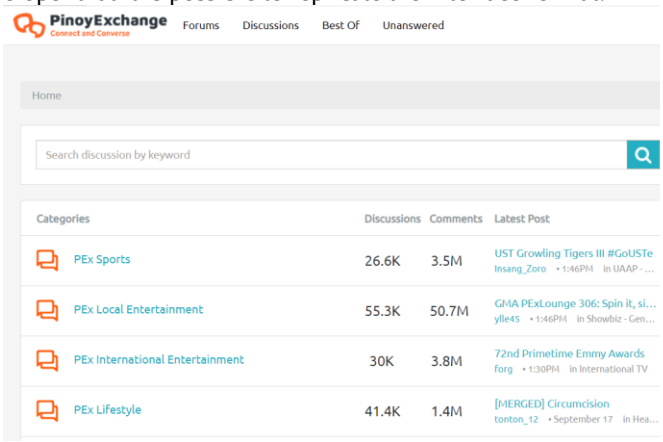


Figure 1. PinoyExchange UI

As can be seen in Figure 1, Pinoy Exchange divides the forum into categories, where each covers a specific topic about popular culture ranging from sports to international news. The categories themselves are arranged in table format with the first column corresponding to categories, the second to number of discussions, the third to the number of comments, and the fourth to the latest post. Inside the categories, there are subcategories essentially display in the same way. COFFEE takes this basic structure and makes subtle changes to it to make it more native to the site.

The concepts of credibility on the site is largely drawn from Quora. It is a question-and-answer site that aims to be a databank of knowledge by using real bigshot names from every field to get answers to a variety of questions. The website has many mechanisms in place to ensure the reliability of the answers it gets from its user base. For example, Quora has a real name policy where its users are required to register with their actual name. [5] This ensures that there is some level of social accountability in the site. COFFEE will derive this feature and at the same time, give the users the freedom to have their own username for certain parts of the site. Quora also has a mechanism for requesting answers from people who are suited to answer any specific question and a voting system to grasp the quality of the answers on the site [6].

III. Methodology

To achieve a rich set of features, forums typically need some of the more relational design patterns in MySQL databases. After all, most of the features of any typical forum will involve some one user-to-many *objects* mapping, where *objects* may refer to posts, categories they moderate, and in the case of an expert-oriented forum like COFFEE, the credentials they hold.

The first section of this methodology will provide exposition for each database and their tables. Because of the importance of the relationships between the tables, it will also be dedicated to exploring how they relate to one another via their primary and foreign keys and why it was done this way. The second section will show all the mechanisms of the website itself and how they were developed with the aid of PHP.

A. *MySQL Databases and Their Tables*

COFFEE has three primary databases for storing information. This section will discuss each one and the tables contained within them.

1) *COFFEE User Database*

The first database is *coffee_user_db*, which stores some of the information directly related to each user such as their account information, current sessions, and categories they moderate. They are made up of three tables that all share a strong relationship with one another.

a) user

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
Username	varchar(30)	NO		NULL	
LastName	varchar(255)	NO		NULL	
FirstName	varchar(255)	NO		NULL	
Password	varchar(255)	NO		NULL	
Birthday	date	YES		NULL	
Email	varchar(255)	NO		NULL	
Status	enum('Admin','User')	NO		User	

The *user* table contains all the basic information one would expect from their user such as their username, last name, first name, password, birthday, and email address. These are all self-explanatory so there is no need to elaborate on them one-by-one. All there is to be said about them is that there are front-end mechanisms that verify their contents (e.g. the email field is checked for valid emails). Additionally, they are all *varchar*s with mostly a size of 255 characters, an arbitrary choice to accommodate as many possible inputs as possible.

Of interest are the *id* and *Status* fields, which contain consequential information that will affect their interaction with the site. The *status* field can be one of two values: ‘Admin’ and ‘User’. The former is given absolute authority in all aspects of the website and will be given access to all controls while the latter may have control of specific *categories*, giving them the authority to possibly delete inappropriate or misleading posts, to approve or decline credential requests, and to assign other experts as moderators of the category. The *id* field will be used to refer to a specific user in all other MySQL tables to minimize the amount of redundant information about them in other aspects. Joins will be used to map the users to any piece of information outside all the basic ones laid out here.

b) sessions

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	
userID	int(11)	NO		NULL	

The *sessions* table stores all the logins in the site by mapping an *id*, which identifies a section to the *userID* of the user that session corresponds to. The id is a 6-digit integer generated in PHP that will be stored in the user’s browser’s cookies to maintain their login throughout the site. Whenever the user interacts with the site in some way that necessitates their user info be used, this session ID will be used to find their

user info. For example, when they post something, this will be used to retrieve the user ID and this user ID will be attached to the post.

c) categories_moderators

Field	Type	Null	Key	Default	Extra
userID	int(11)	NO	PRI	NULL	
categoryID	int(11)	NO	PRI	NULL	

The *categories_moderators* stores information about which users will serve as the moderators for each category. This is done by mapping the primary key IDs *userID* and *categoryID*. When a user is a moderator of a certain category (e.g. Agribusiness), his ID will be mapped to the ID of that category (see *categories* table in *COFFEE Main Database*). Being a moderator entails that the user is authorized to approve or decline expertise requests and regulate content in their own respective category. Only people that has a credential that corresponds to a category can serve as moderators. This is all controlled by the PHP mechanisms in place.

2) COFFEE Credential Database

The credential database *coffee_cred_db* contains all the information that pertains to all information relevant to the expert credential trust system that COFFEE has in place to ensure that the sources of information in the site are actual experts. This is not to be confused with the user database, which is used for *site* credentials.

a) Credential_master_list

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
name	varchar(255)	NO		NULL	

The credential master list table contains all possible credentials that can possibly be used for the website. Each credential will have an associated name and id and another table, *credential_category_connection*, will be used to identify which categories the credential can possibly be used for. The name field is basically a descriptor of the credential, which the user can see when trying to register a credential. For example, a degree in BS Bioengineering will have "BS Bioengineering" as its name and it will be assigned some ID.

b) Credential_category_connection

Field	Type	Null	Key	Default	Extra
credentialID	int(11)	NO	PRI	NULL	
categoryID	int(11)	NO	PRI	NULL	

The *credential_category_connection* contains the mapping of credentials in the master list to categories they can be used for via their primary key IDs. In particular, when some credential, say "BS History" with ID 2, can be used as a credential for the category "Philippine History" with ID 1, both will be placed on the same column. Note that both are primary keys, which ensures that no connections are redundant. Furthermore, when the user adds a custom credential and associates it with a category, it will be added here and to the *credential_master_list.*

c) User_credentials

Field	Type	Null	Key	Default	Extra
userID	int(11)	NO	PRI	NULL	
credentialID	int(11)	NO	PRI	NULL	
evidence_file_dir	text	YES		NULL	
status	enum('PENDING','APPROVED','REJECTED')	YES		PENDING	

This table contains the actual credentials attributed to each user. When a user ID requests that a particular credential in the *credential_master_list* be added to their account, their user ID will be mapped to the ID of that credential. The status is an enumerable that signifies the current state of the credential (it is "PENDING" by default). When it is "PENDING", the credential will be evaluated by the moderators and they will decide if the credential is "APPROVED" or "REJECTED". The evidence_file_dir (evidence file directory) serves as the evidence of the said credential. It can contain a relative link (if the user uploaded the credential unto the site) or an absolute link (if the user inputted an external URL as their evidence instead).

3) COFFEE Main Database

COFFEE's main database, *coffee_db*, is what the developer would dub as the flesh and blood of the site. It contains the actual content that users engage in such as posts, categories, and answer requests. In essence, it contains some of the most important tables on the website. They are purposely arranged to maintain extensibility to easily accommodate for new features in the future.

a) Categories and Topics

```
MariaDB [coffee_db]> show columns from topics;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| id          | int(11)      | NO   | PRI | NULL    | auto_increment |
| name        | varchar(255) | NO   |     | NULL    |                |
| description | text         | NO   |     | NULL    |                |
| categoryID  | int(11)      | NO   |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
4 rows in set (0.013 sec)

MariaDB [coffee_db]> show columns from categories;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| id          | int(11)      | NO   | PRI | NULL    | auto_increment |
| name        | varchar(255) | NO   |     | NULL    |                |
| description | text         | NO   |     | NULL    |                |
| creatorID   | int(11)      | NO   |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
4 rows in set (0.010 sec)
```

The posts in COFFEE are classified by their topics and this is then classified by their categories. Both the topics and categories table contain the similar columns because they represent very similar things. However, it is important to differentiate the two here.

Categories represent different Filipino-related fields such as Philippine History, Agribusiness, and Construction. The name of the category is specified through the *name* column and a brief description that states what it is all about is stored in the *description* field. Each category is assigned its own ID, to allow other tables to refer to it. Finally, the creatorID shows the user ID of the person who created the category. The categories play a different functional role from topics in that moderators

handle entire categories instead of specific topics. Furthermore, credentials (see the COFFEE Credential Database section) can be used throughout every post in a category it is valid for.

Topics are basically sub-fields or specific subject matters within the category. For example, a topic might discuss about "The Philippines Under WWII" under the "Philippine History" category. It essentially contains the same field as the categories database except for the *categoryID*, which represents the ID in which the topic is under.

b) Posts and Responses

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
title	varchar(255)	NO		NULL	
datetimePosted	timestamp	NO		current_timestamp()	
topicID	int(11)	NO		NULL	
userID	int(11)	NO		NULL	
content	text	NO		NULL	

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
content	text	NO		NULL	
datetimePosted	timestamp	NO		current_timestamp()	
upvotes	int(11)	NO		0	
downvotes	int(11)	NO		0	
postID	int(11)	NO		NULL	
userID	int(11)	NO		NULL	
parentResponseID	int(11)	YES		NULL	

The posts section contains *all* the posts within the website itself regardless of topic or category. Each row represents a post (essentially a question or a thread).They are all placed on the same database to allow separation or combination when necessary. The *title*, *datetimePosted* and *content* fields are self-explanatory. The *datetimePosted* is defaulted to current_timestamp() to minimize the need for time manipulation in the PHP scripts. This means that the date and time a post was published will be based on the SQL's time. The *topicID* represents the ID of the topic the post is under while the *userID* represents the ID of the person who posted it.

The responses table contains every response or answer to a post in the website. It only differs to the post table in that it contains the *postID* instead of the *topicID* to show that it was a reply to a particular post. The *parentResponseID* is reseved for when a response is a reply to another response within the post, in which case the ID of that response will be shown.

Although similar, they are functionally different in the eyes of PHP. Only posts can be directly linked into the website and answer requests (whose table will be discussed later) can only be done with posts. Further differences will be discussed in the next section of the methodology.

c) Vote_master_list

Field	Type	Null	Key	Default	Extra
voterUserID	int(11)	NO		NULL	
postID	int(11)	NO		NULL	
vote	enum('UP','DOWN')	NO		NULL	
responseID	int(11)	YES		NULL	

This table is used to store all the upvotes and downvotes made on the website. An upvote as discussed earlier is used to signify that a user liked a response to a question while a downvote signifies that they disliked or disagreed with the answer. When a user votes on a particular response, their vote will be added to the table. The *voterUserID* is used to refer to the person who casted it. This is to ensure that no duplicate votes are made. The *postID* and *responseID* both identify which response in particular the vote was for. The vote field is used to identify whether it was upvote or a downvote.

The votes are used to signify how much other people liked or appreciated a particular response. It is also used to rank the best answers on the website.

d) Answer_requests

Field	Type	Null	Key	Default	Extra
requestorUserID	int(11)	NO		NULL	
requestedUserID	int(11)	NO		NULL	
postID	int(11)	NO		NULL	

This table stores the answer requests on the website. In particular, any user can ask an expert in that field to answer the question in a specific post (identified with the postID). The *requestorUserID* and *requestedUserID* refers to the ID of the user that made the request and the expert requested for.

B. Website Components and Functionalities

This section conceptually explains in some detail how each major function and component of COFFEE were implemented and how the webpages interact with the SQL database to achieve the desired results. Since many of the queries used can be inferred through the explanations, they will not be included here. If it cannot be inferred, the author will explain the query used in some detail.

1) The Header Page and General Functions

The header page *header.php* is a PHP script that will serve as the banner of the website on every page. Therefore, it will be placed on all accessible pages using the *include* function in PHP. In *header.php*, another PHP script called *general_functions.php* is included. This contains most of the essential functions that will be used throughout the site, including most SQL queries used.

Inside the general functions script is the main connection to the SQL databases, which can be accessed using the *get_sql_conn()* function. It also a function for cleansing all the contents of the $_POST variable and a function for getting the user info of the currently logged in user (*get_session_user*). Most important of all is the *do_query* function which streamlines sending queries to the database by other parts of the site to perform a query safely without the risk of potentially crashing unexpectedly. It does so by performing the check if the passed query string returns false after being executed. This removes the need to check if the query was successful on every occasion. It will instead be checked from this single point. It only removes the need to respecify the database connection every **mysqli_query** call. All function calls that involve some query will take the form query_xxx_xxx($identifiers,…). Most of them will return a query result object that can be used by the caller.

2) Login and Registration

The login and registration are handled by the scripts located in the *user* directory. For the registration system, the user will be asked to input their last name, first name, username, password, and email. The validation of the value of the email is done at the front end (by using the input type "email"). After submission, the PHP script will submit using a POST request the contents of the registration unto itself and it check if there are any empty fields or if the passwords do not match and it will display the appropriate error. Once this is done, the values of their submission are inserted into the *users* database using the *user_register* function with an INSERT query.

The login system takes the username and password of the user via a POST request for security purposes and then it calls the function inside *user_functions.php* that uses a query to return to login the user by first using a SELECT query that uses the username and password in a WHERE condition. If the number of rows returned is 1, a valid user is assumed to have been detected and the ID of that user is added to the *sessions* table where it is paired with a randomly generated 6-digit number that designates the session. This session number is given by PHP to the user via a cookie, where it can be accessed from anywhere in the site. This access is cleanly implemented inside *general_functions* as *get_session_user*, which returns an associated array of the info of the currently logged in user, which includes their ID.

3) Categories (Index Page)

The index PHP script is responsible for displaying to the user all categories, their descriptions, and their creators. It does so by using the *query_categories* function that simply loads everything from the *categories* table. Since all users have special access to this page, this page does not retrieve the user's session ID. Once it has queried all categories, it simply displays all of them into a table. A hyperlink leading to the topics page (*category.php)* will be attached to the name of each category with its ID as the parameter. It also has a link that leads to a page for creating new categories. This will be discussed in a later section. This will take the form **category.php?id=*id***.

4) Topics and Posts Page (category.php and topic.php)

The URLs in the Categories page (*index.php)* directly lead to the topics page. Note that it has passed the category ID as a parameter. This means that it can be treated as a GET request and the ID is retrievable using $_GET["id"]. Using this, the *query_topics* function can use this in a WHERE condition to filter all the topics within the category using the categoryID field. The page *category.php* also retrieves information about the last post made in each category listed (since this is standard format for an internet forum) and it does so by joining the *users* and *posts* tables to simultaneously get information on the post and the user that posted it and it orders the results by the datetimePosted field in descending order. The query also limits the result to 1 since only the latest post is needed and it would be a waste of resources to load all of them. If the last post query is empty, it simply returns "No posts yet."

The posts list page (*topic.php*) has a similar mechanism. The topic names in the Topics page have hyperlinks with the topic ID as a parameter. The posts page simply takes this and uses it to filter all the posts that are under the selected topic. Unlike the Topics page, this also shows the name of the *poster's* username, which it can directly retrieve from the result of *query_post*. It also takes information from the last reply by doing something similar to the Topics page, except the *responses* table are joined to the *posts* and *users* table to find the latest response in a particular post by filtering for the postID and ordering by responses.datetimePosted.

5) Single Post Page (post.php and response_template.php) and Voting (vote_script.php)

Similar to the previous pages, this takes the id parameter passed on from the links to the posts from *topic.php*. It begins by querying the post with the ID as a filter to retrieve the single post being viewed. Note that the ID has no relationship with which topic the post is under, so adjacent IDs could be in in different posts. To verify if the post with the given ID truly exists, the number of rows returned is checked. If it is 0, the page simply returns an error and dies. It displays the contents of the asker's post by simply taking the contents of the associative array from the post query.

To display the responses to the post, the *query_response_ids_from* function takes the post ID and returns all response IDs from the *responses* table with that post ID foreign key. Another external script *response_template.php* is then included iteratively through each response ID, which it uses to display that specific response.

The response template script query joins the *responses* table and *users* table to retrieve the reply with the given response ID. Specifically, it takes the content, date and time posted, user ID of the responder, and the post ID belongs to. It also uses join queries to retrieve any credential that user has that pertains to the category the post was under and formats his response appropriately to reflect this.

It also uses the response ID to count the total number of upvotes and downvotes for that response to display. It then subtracts them to get the total cumulative vote for that response. It also retrieves what the current logged in user voted for in that response and colors the vote buttons appropriately.

To register votes once the user has clicked on it, the vote buttons link to an external script with the response ID, post ID and the vote as parameters. This page *vote_script.php* takes the current user and inserts his ID under the voter user ID field in the *votes_master_list.* All of the GET parameters are inserted in the appropriate fields in the said table. After the vote has been registered, the user is redirected back to the Single Post page.

There is a form at the bottom of the page for getting user responses and answers to the post. It is submitted to the same page via a POST request. Upon submission, the page checks if a POST request was done and if so, the current logged in user ID is taken and the post is reformatted to ensure that there is no cross-site scripting on the response. The reformatting will also support the use of BB codes ([b], [i], etc.) to format the text. It

will convert them into HTML tags when inserted into the *response* table along with the responder's user ID.

6) Credential Handling (credentials.php, credential_script.php)

The credential page allows the user to submit their credentials so that their posts will have some weight in the eyes of the reader. However, this does not mean that they can just arbitrarily put anything in their credentials. Instead, moderators will regulate and check their submitted evidence to ensure the legitimacy of the credential. This is all done from *credentials.php*.

At the top of the page, there is a table that shows the user's credential in every field. This is done via a simple query that loops through every category and checks if the user has any credential for that category. This is done by passing each category ID to *credentials_table_template.php*. This template script is included in the main credentials page and it queries for the user's credentials in the specified category using a long series of JOINs. When retrieved, it prints the name and status (pending, approved, rejected) of the credential, and inserts a hyperlink to the evidence.

At the bottom is a form for submitting new credentials to the site. Here, the user is given some flexibility for submission. They can choose to pick an already existing credential position for a category, say "Historian" for "Philippine History" or if they cannot find it, they can create a custom credential (as long as they choose its category). Upon submission, this will be added to the *credentials_master_list* table for future use. They can also select between uploading their evidence or linking to it (e.g. a Linkedin profile). When they upload, PHP takes the file and places it unto the credential_evidence folder. *Credential_script.php*, the PHP script responsible for submissions, then inserts a relative link to the *user_credentials* database. If the user inputs an external link, the entire link is stored.

7) Moderator System (moderator.php, moderator/.php, delete.php)*

The moderator system allows moderators of a specific category to add other moderators, evaluate submitted credentials, and monitor posts and responses.

At the top of the moderator page, an iframe linking to another page, *add_moderator.php?categoryID=$id* can be seen. This allows any moderator to look up any expert on the category and assign them as moderators. This is done by searching for them. The add moderator script submits this search into itself via a GET request, with the search query as a parameter. A LIKE condition is used to look up the username of closely matching *experts.* Note that this means these users are filtered by both their username and their credential - the credential tables are joined to enable the script to filter for experts on the category.

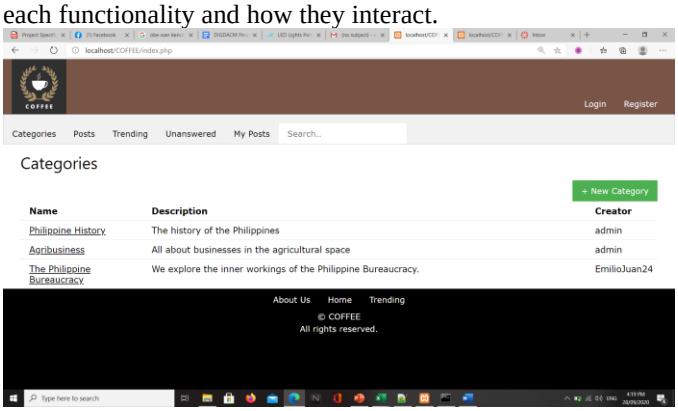
To add the user as a moderator, *add_moderator_script.php*, another PHP script, takes the user ID as a parameter and inserts it into the *moderator* table.

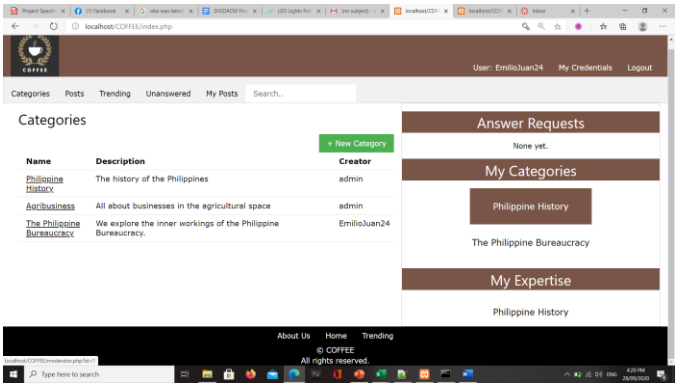
At the bottom of moderator page, there exists a table for pending credentials in that category. The script lists all credentials that are both pending and from that category by using an SQL query in the *query_pending_credentials_from* function. This is then displayed in table form with a link for rejecting and accepting, and the evidence URL. The hyperlink passes the action to be done on the credential via the parameter *action* and it also passes the ID of the credential in question, the owner of the credential, and the category. The *moderator_credential_action.php* script then updates the status of the credential in question using another query.

8) Answer Request System (a2a.php)

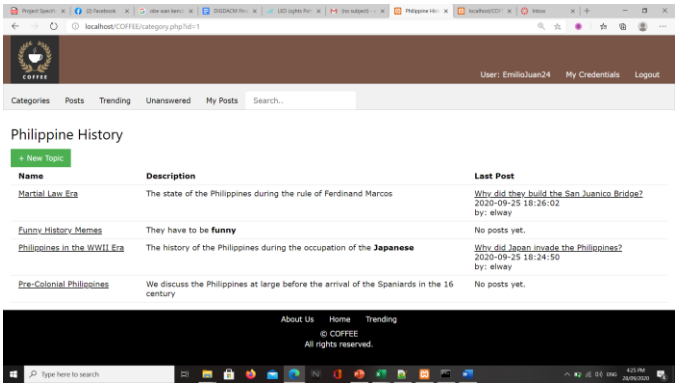
IV. Results And Disucssion

This section contains a discussion of all pages that were successfully integrated into the website. It will include a discussion of the logged in and logged out views for each page and how they differ. The later parts of the results will discuss each functionality and how they interact.

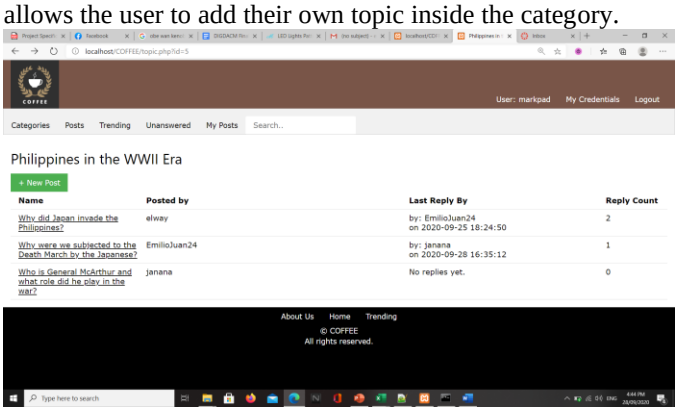




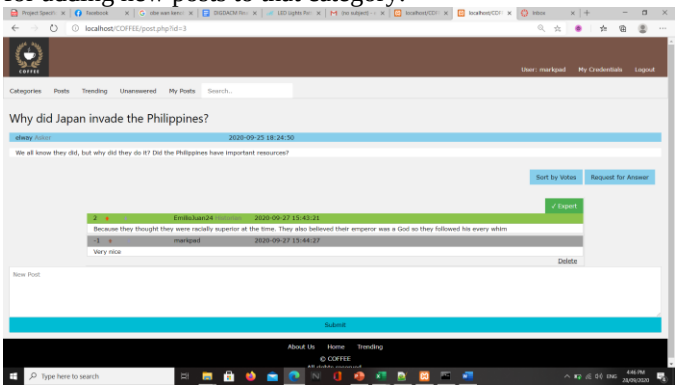
The home page of the website includes a complete list of all the categories on the website and a button for adding new categories. In the right side of the page, all the logged in user's answer requests, moderated categories, and categories they have an expert in for is shown. On the top is a hyperlinked logo that leads back to the main page and on the right of the header is the user section where there are URLs for logging in, logging out, and viewing credentials. Below the logo is a bar that leads to some of the top-level pages in the website such as trending and unanswered posts.



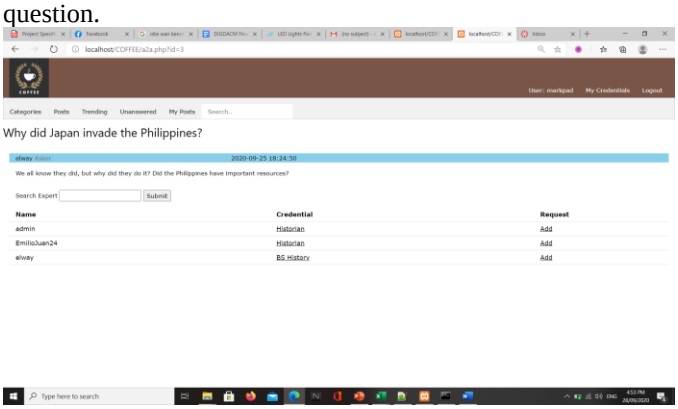
Upon clicking a category, the user is brought to a list of all the topics inside that category, including its description and details about the last post. There is a button on the top that allows the user to add their own topic inside the category.



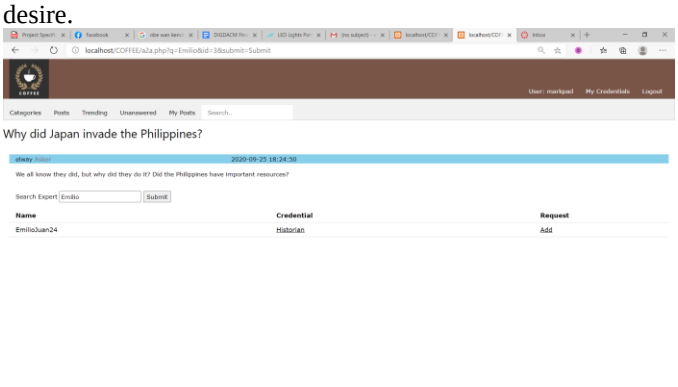
Once the user clicks on a topic, they will be brought to a page that lists all the posts related to that topic. This will include information such as the name, the original poster, the last reply, and the number of replies. There will also be a button for adding new posts to that category.



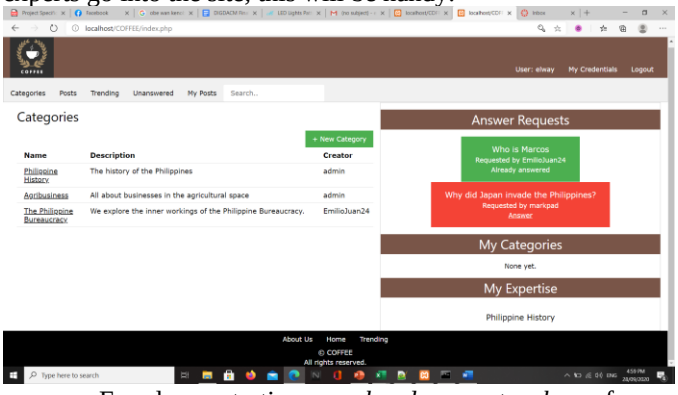
Once a post has been selected, the user is brought to a page that shows all the replies to the said post. Here, the user has the ability through the new response bar at the bottom of the page to submit their own answer to the originally posed question. Also, note that some users' answers have a green header bar and a tag in the right saying they are an expert. There is also a title next to their name that indicates what their credential is for and a timestamp for when the response was made. On the left hand side of a post, there are also two arrows for voting up and down based on whether the user liked the answer. The number to the left of the arrows indicate the total cumulative votes. The arrow that indicates the current logged in user's vote is highlighted and the other is grayed out. There are also additional buttons to the right. The Sort by Vote allows you to sort the answers by the number of votes. Of course, the expert tag will still be present if you doubt the validity of their claims. The request to answer button allows the current user to request any expert on the category to give their answer to the question.



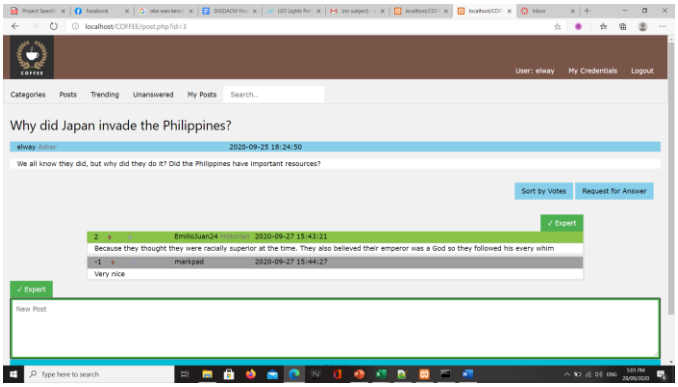
When the user presses the Ask to Answer button, they are brought to an A2A page that lists all the experts in that field. The user can search for a particular expert they want if they so desire.



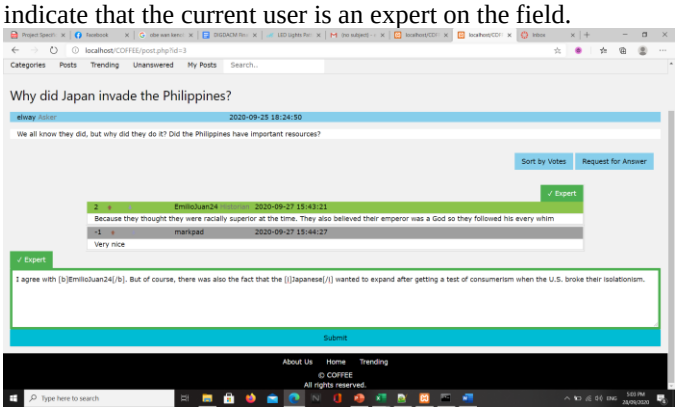
Of course, here, this is not very useful but once many experts go into the site, this will be handy.



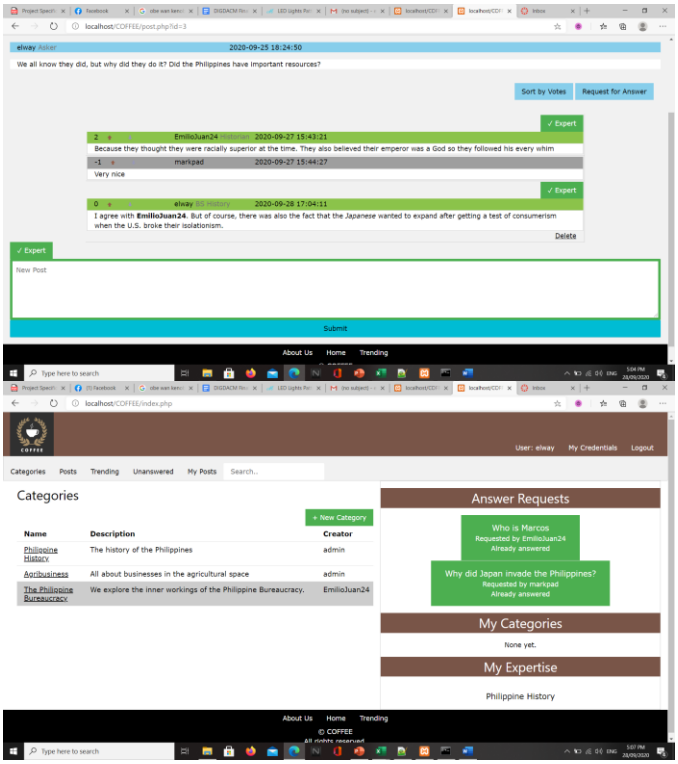
For demonstration, *markpad* requests *elway* for an answer to the question regarding Japan. On *elway's* home page, the request is shown in red, indicating that he has not answered the request yet. Navigating with the answer button, he will be led to the question



The answer text area will be highlighted in green to indicate that the current user is an expert on the field.

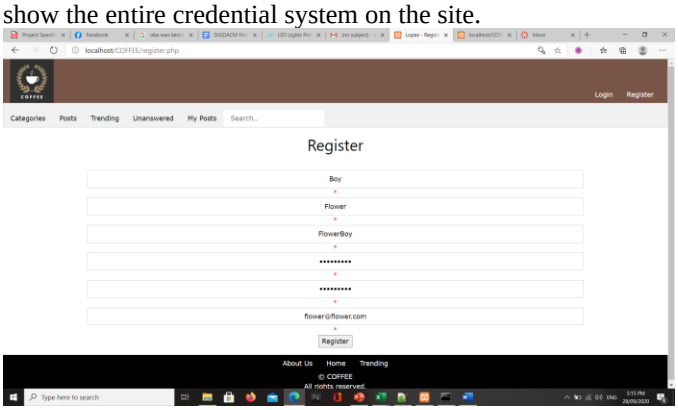


The website supports BB code markups, meaning users can put HTML-like tags to format their posts' typeface. When posted, the figure below shows the result. Again, since *elway* is a major in History, the site registers him as an expert. He also has the ability to delete his post since it is owned by him.

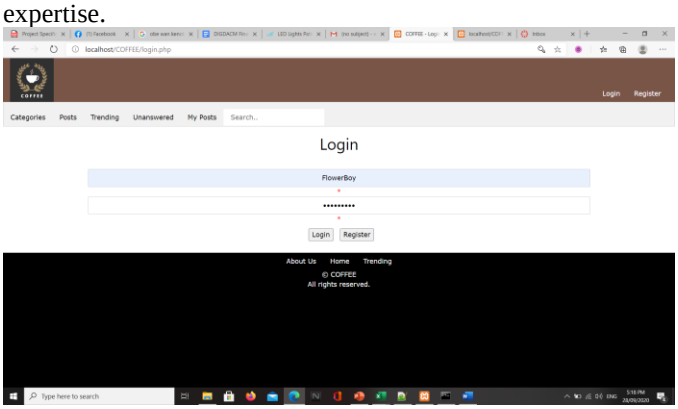


Upon answering, it will reflect on *elway*'s account that he has already fulfilled the request. Here, it can be seen that the post he answered was already answered by him.

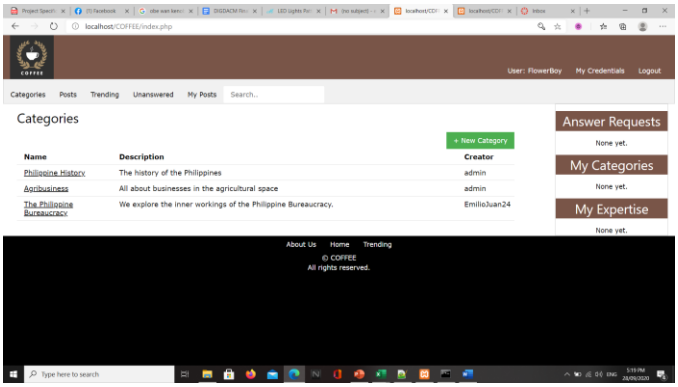
Hypothetically, the user might want to create an account with their expertise registered. To serve as a demonstration, we will assume a hypothetical user *FlowerBoy* wants to register as an expert on agribusiness. This will help show the entire credential system on the site.



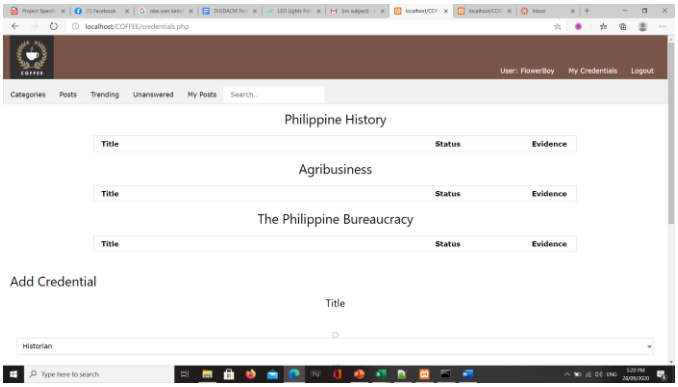
The registration system asks the user to input their last name, first name, username, password, and email. Upon completion, they will be registered to the site without any expertise.



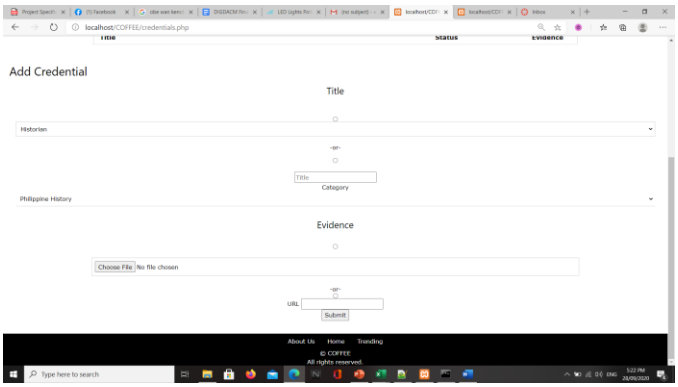
The login system simply asks for their username and password. If it matches any user on the database, the user is given a cookie that tracks their login by linking their session ID to their user ID in the database.



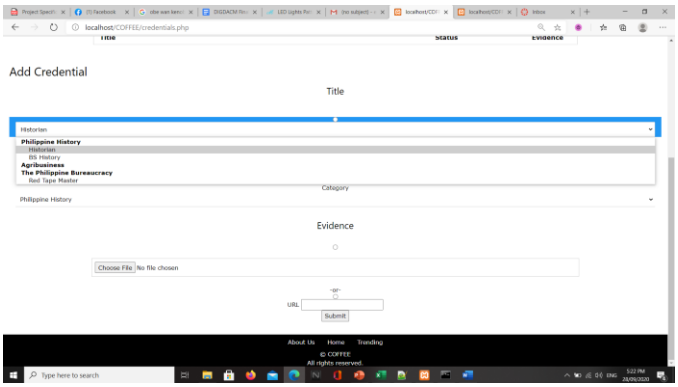
The login succeeds because the user has been created. It can be seen here that the user has no expertise yet. To register them, they must go to the My Credentials page in the link above.

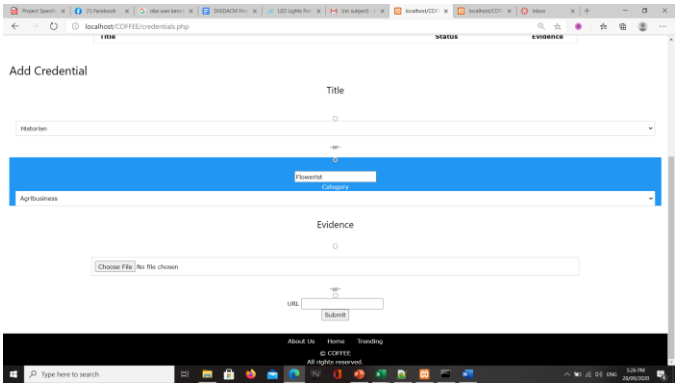


The page contains tables that contain their expertise in every category. At the moment, the hypothetical *FlowerBoy* user has no credentials. To register for one, they use the form below.

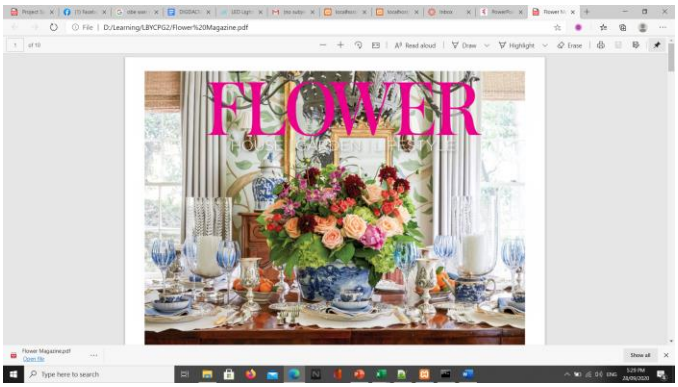


To specify their title (e.g. Florist, BS Biologist), they use the Title section of the registration. They have two options to specify their title: first they can choose any of the predefined ones in the site using the dropdown list and second, they can define their own. For *FlowerBoy*, there is nothing he can choose under *Agribusiness* because other users have not registered credentials under the category. To circumvent this, he can manually insert one himself, which can be used by future users.

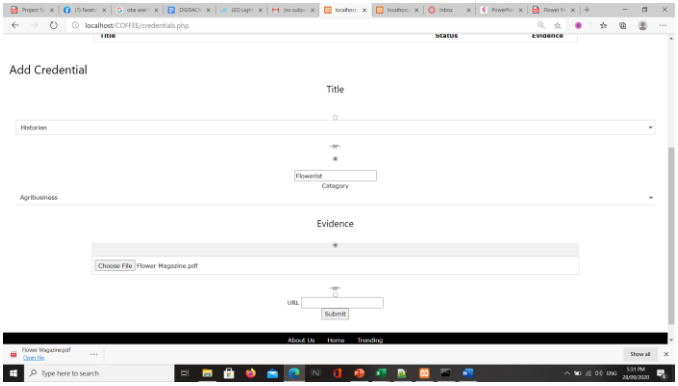




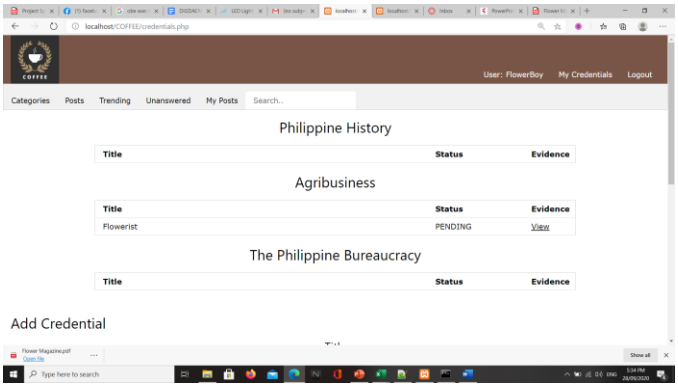
The user also must submit evidence that they do indeed have the specified title. They have two options for doing this: manually uploading a file that proves their credential or a URL that shows their work.



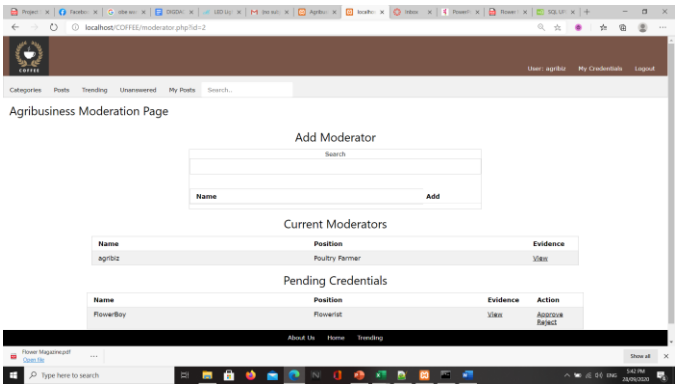
Here, we will have the user upload a Flower magazine they worked on to show their work on selling flowers to interested consumers.



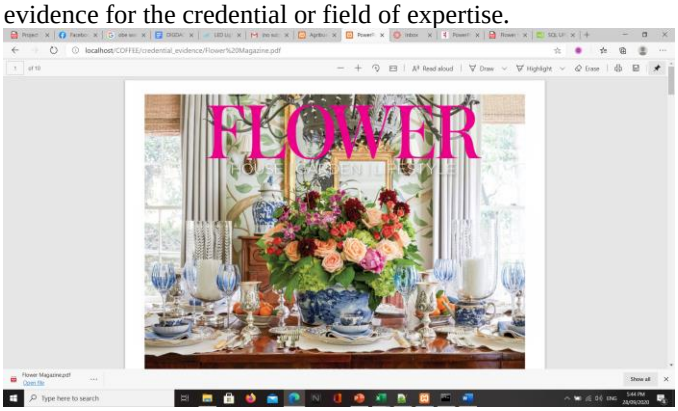
Once submitted, the new credential will appear as PENDING in his official credential list.



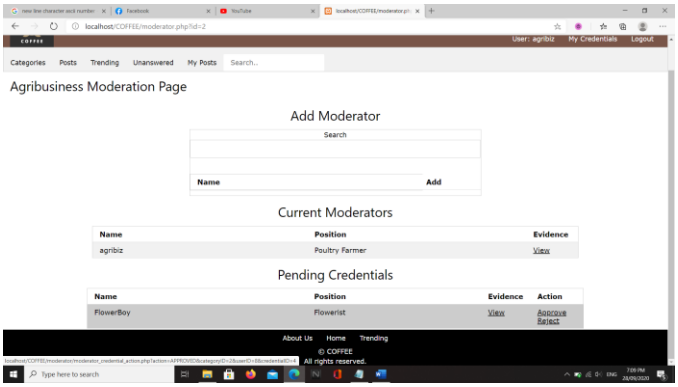
The uploaded magazine can be viewed through the View Evidence button. The user cannot use this yet to be marked as an expert in the Agribusiness category. It still has to be approved by a moderator for the *Agribusiness* category.



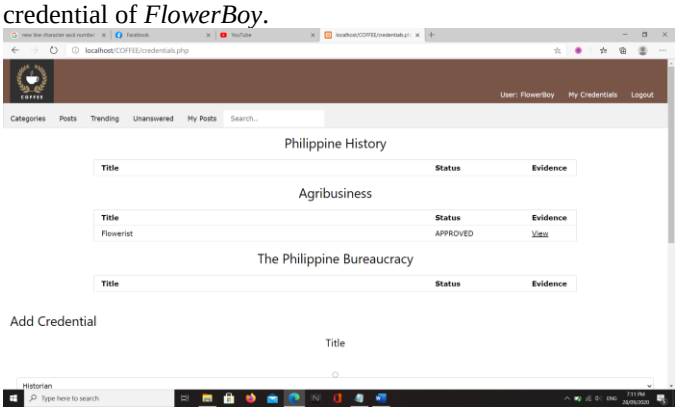
Currently, the only moderator in Agribusiness is a user named *agribiz*. If he goes to the moderator page for agribusiness, he will a list of all pending credentials which can approve or reject. He can do so by using the view link to go to or download the evidence for the credential or field of expertise.



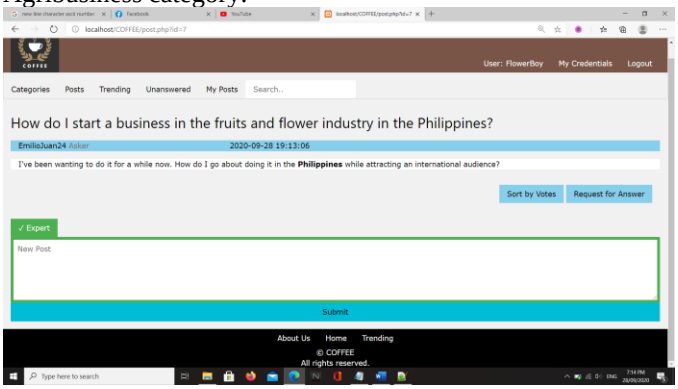
The moderator can look over the evidence uploaded on the site (see the URL) and find something that proves the user's claim.



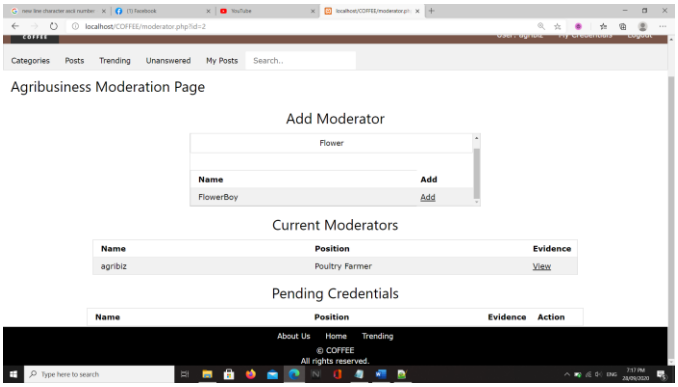
With this, the moderator *agribiz* can accept the credential of *FlowerBoy*.



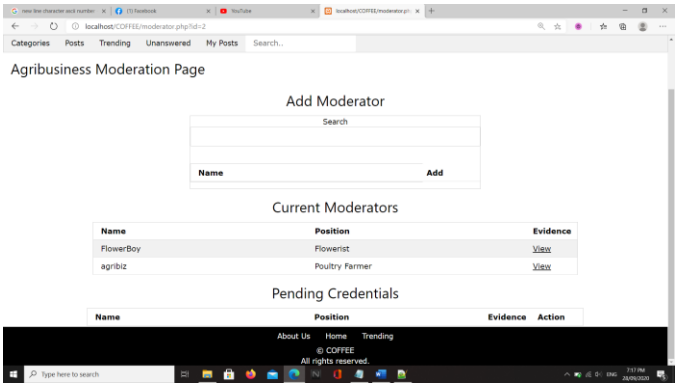
This now reflects in FlowerBoy's credential page. It also shows when he tries to post something under the Agribusiness category.



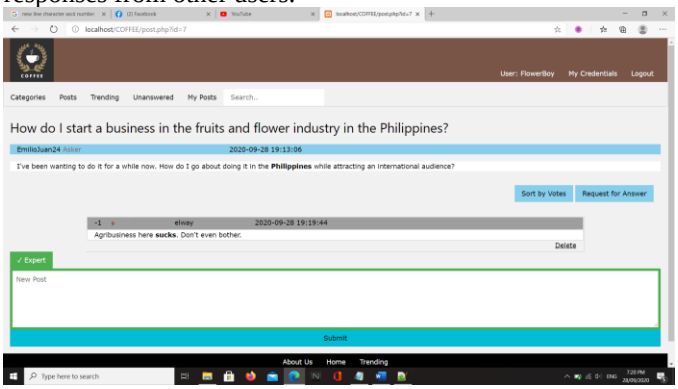
To add moderators, there is an Add Moderator section in the moderator page of any particular category. For example, say *agribiz* wanted to make *FlowerBoy* a moderator. He can do so by looking up FlowerBoy in the moderation page.



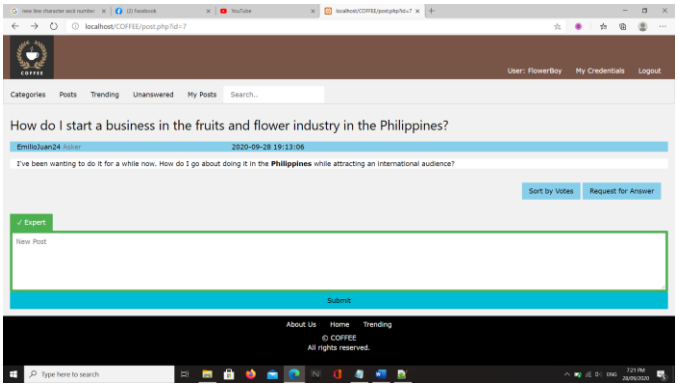
The current moderator list will immediately reflect this.



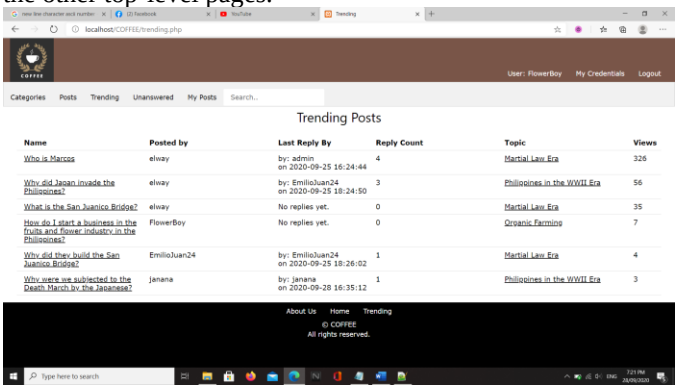
Moderators in any category has the power to: add other moderators, approve credentials, and delete inappropriate responses from other users.



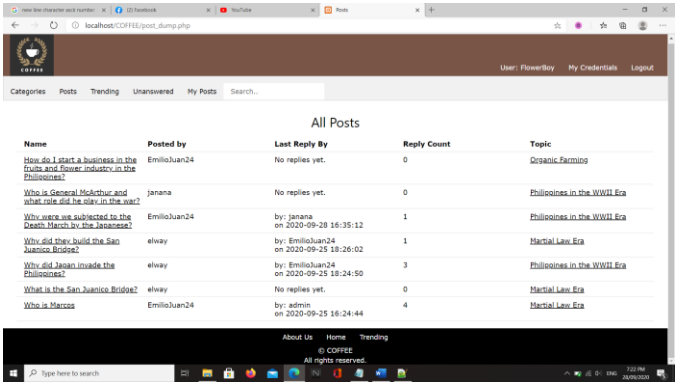
For example, user *FlowerBoy* sees a rude or offensive post in one of the questions. It is seen in the picture above that even though he did not make that post, he has the authority to delete it because he is a moderator. Deletion from a moderator is effective immediately.



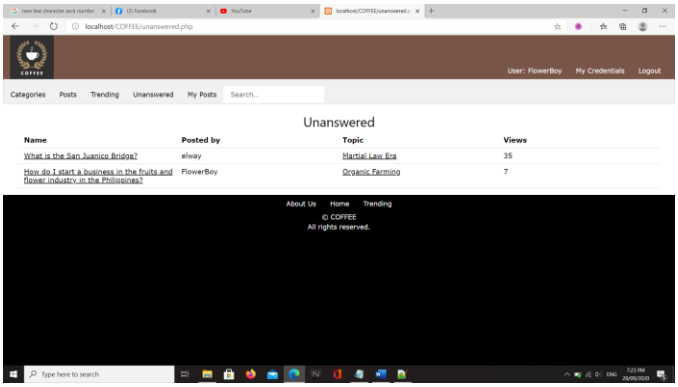
In the next parts, the developer will explain some of the other top-level pages.



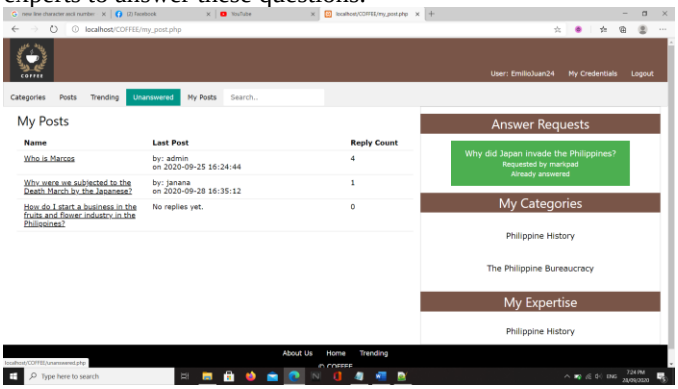
The trending post page shows some of the most viewed posts. The number of views can be seen on the right. Since this is not a unique view counter, this can exceed the number of users in the site.



The Posts page lists all the posts in the site and it also indicates the topic the post is from.



The Unanswered page shows all the questions in the site that have exactly zero replies. This encourages other experts to answer these questions.



The My Posts page shows all the posts by the current logged in user.

V. Conclusion

COFFEE strives to a place that wants to strike a more subtle demographic of curious, like-minded Filipino enthusiasts, academics, and experts. It seeks to encourage Filipino experts in their own field to share their knowledge by asking the real questions from inquisitive Filipinos. It aims to a reliable source by ensuring that its experts are credible and well-trained in their field. Although not perfect, it has fulfilled these goals to some extent through the implemented features. It has simple mechanisms in place to ensure that the experts are real by allowing moderators to inspect some evidence that shows what any alleged expert is saying is true. It also has a basic mechanism for ensuring that posts are not offensive. Finally, through answer requests, it will allow people who want to know something to feel like they are unheard by the experts.

VI. Recommendations

There are a variety of improvements that can be made to the system as it currently stands. The developer recommends the following changes be made in a future iteration of the site:

- Improving the user interface in general by making it more dynamic and making some of the more difficult to navigate parts like the Credentials page intuitive.
- Adding other features for content moderation such as a reporting system for bad, misleading or inappropriate answers and users.
- Creating a recommendation page based on the interest of the user and a follow feature that allows users to follow certain topics
- Putting in place a notification system in the site itself and in the user emails.
- Adding an email verification system to ensure the legitimacy of the addresses.

References

[1] "What is an "Internet forum"?," 5 June 2008. [Online]. Available: https://web.archive.org/web/20081011092536/http://www.videojug.com/expertanswer/internet-communities-and-forums-2/what-is-an-internet-forum.

[2] J. Wortham, "The New York Times," 12 March 2010. [Online]. Available: https://www.nytimes.com/2010/03/13/technology/13social.html.

[3] PC Mag Digital Group, "Definition of Internet forum," [Online]. Available: https://www.pcmag.com/encyclopedia/term/internet-forum.

[4] PinoyExchange, "PinoyExchange.com | About Us," [Online]. Available: https://info.pinoyexchange.com/about-us.html.

[5] Quora, "Terms of Service - Quora," [Online]. Available: https://www.quora.com/about/tos.

[6] L. Yang and X. Amatriain, "Recommending the World's Knowledge: Application of Recommender Systems at Quora," in *RecSys '16*, 2016.