**TABLE OF CONTENTS**

**File page**

1. Admin ………………………………………………………………19
2. Volunteer……………………………………………………………22

**Technologies used**

1. HTML x CSS
2. jQuery
3. PHP X MySQL

**Software Requirement**

You guys know you need XAMPP Server so I won’t.

I have implemented two roles WITH LOGIN system:

1. Volunteer
2. Admin

Admin role has full access to the system; they are responsible for the following:

1. Approve a volunteer.
2. Admin removes volunteer from the system.
3. Admin removes answer provided by a volunteer.
4. Manage learners’ lesson effectively.
5. Ensure that learners question answers are based on their questions.
6. Admin removes question asked by the user.

**DATABASE DESIGN**

ADMINISTRATOR LOGIN TABLE:

| **Field Name** | **Field Type** | **Field Length** | **Description** |
| --- | --- | --- | --- |
| Username | Varchar | (50) |  |
| Password | Varchar | (256) |  |

Table above named admin in the database, serves as a repository for administrator details that will be able to have access to the entire details on the application. It is a table that stores the administrator login details.

VOLUNTEER LOGIN TABLE:

| **Field Name** | **Field Type** | **Field Length** | **Description** |
| --- | --- | --- | --- |
| id | Test | () |  |
| Fullname | Varchar | (30) |  |
| Status | Varchar | (15) | Approved or not |
| Special | Test | () | Field of specialization |
| Contact | Varchar | (100) |  |
| note | Test | () | Short note on about. |
| Password | Varchar | (50) |  |
| Username | Varchar | (50) |  |

Table above named volunteers in the database, serves as a repository for volunteers’ details.

QUESTIONS TABLE:

| **Field Name** | **Field Type** | **Field Length** | **Description** |
| --- | --- | --- | --- |
| id | Test | () |  |
| questions | Test | () |  |
| contact | Test | () | Contact provided by user |

Table above named questions in the database, serves as a repository for questions asked by the users.

QUESTIONS AND ANSWERS TABLE:

| **Field Name** | **Field Type** | **Field Length** | **Description** |
| --- | --- | --- | --- |
| volunteers | Test | () | Volunteers name |
| id | Test | () |  |
| question | Test | () | questions asked by the users |
| answers | Test | () |  |

Table above named qanda in the database, serves as a repository for questions asked by the users and answers provided by volunteers.

**Pages/Files Description**

I will explain the p of the websites and their algorithm. I have seven different interfaces.

1. Index page
2. Admin login page
3. Admin page
4. Volunteer Register Page
5. Volunteer login page
6. Volunteer page
7. Questions and answers page

**Index page**

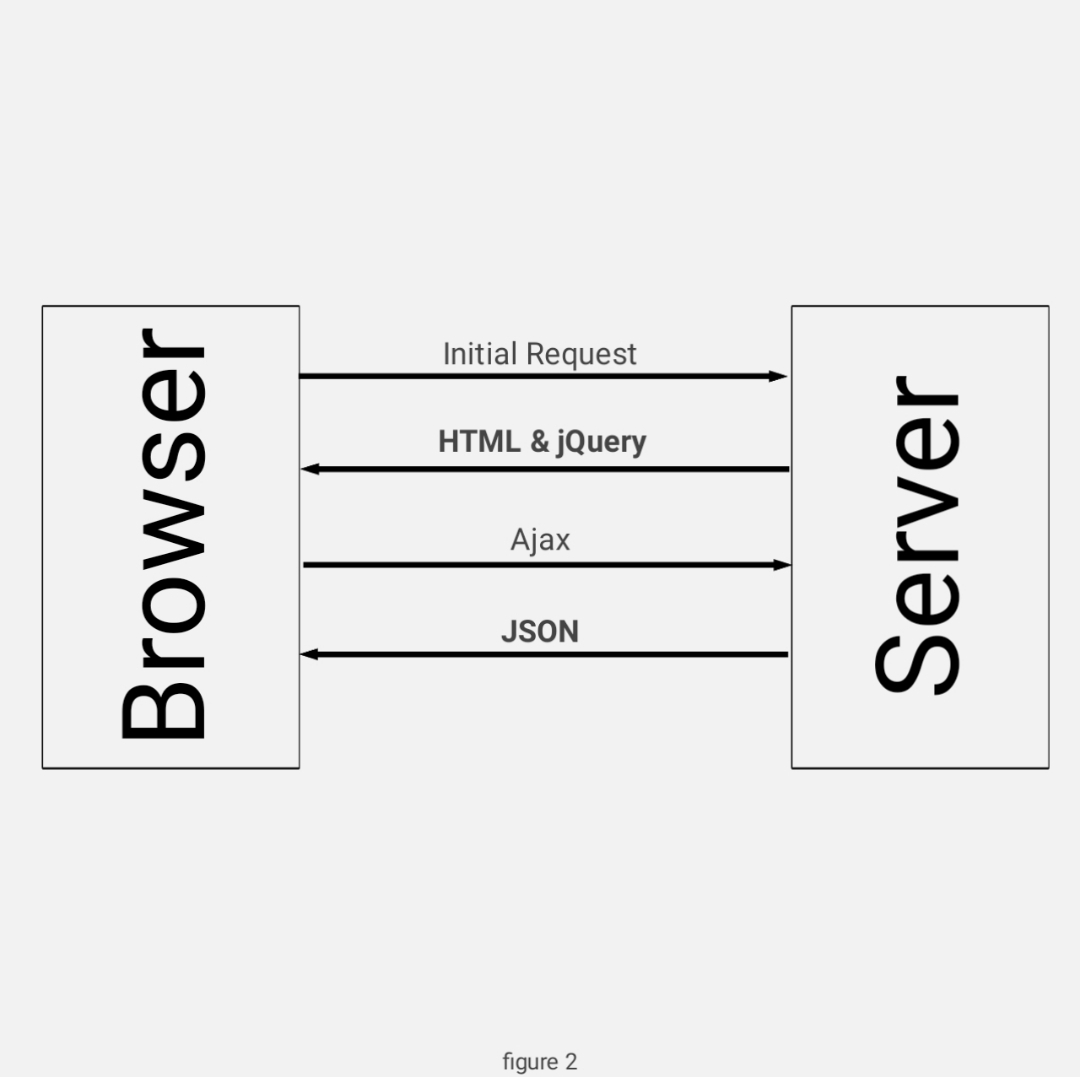
This works as a single page application and contains the lessons and all what user demands, users can access to read and ask questions through the provided ask-question-textbox at the footer section of the page.

Additionally, volunteer has to access to the login page through index page, they can click the user icon located in the upper level of the page to navigate their login page.

**SPA oriented**

This is a single-page application and performs a single request to the server at the initial download and saves all the data it gets, so user can use the received data to work offline if needed, which makes it more convenient for users as it enables them to consume fewer data resources.

When users access the website the entire page is loaded after the initial request they see all necessary content in the browser, including the navigation bar where the lessons links are. When users start interacting with the page, they dynamically rewrite or update only the portion of the page and there is no need to refresh the page. See below.



What SPA ideas:

1. Performance
2. Improved user experience
3. Data caching
4. Development speed
5. Ease of debugging

**Data caching**

After the first request to the server, all the necessary local data is stored in the cache, and that provides users with the possibility to stay in an offline mode.

**Development speed**

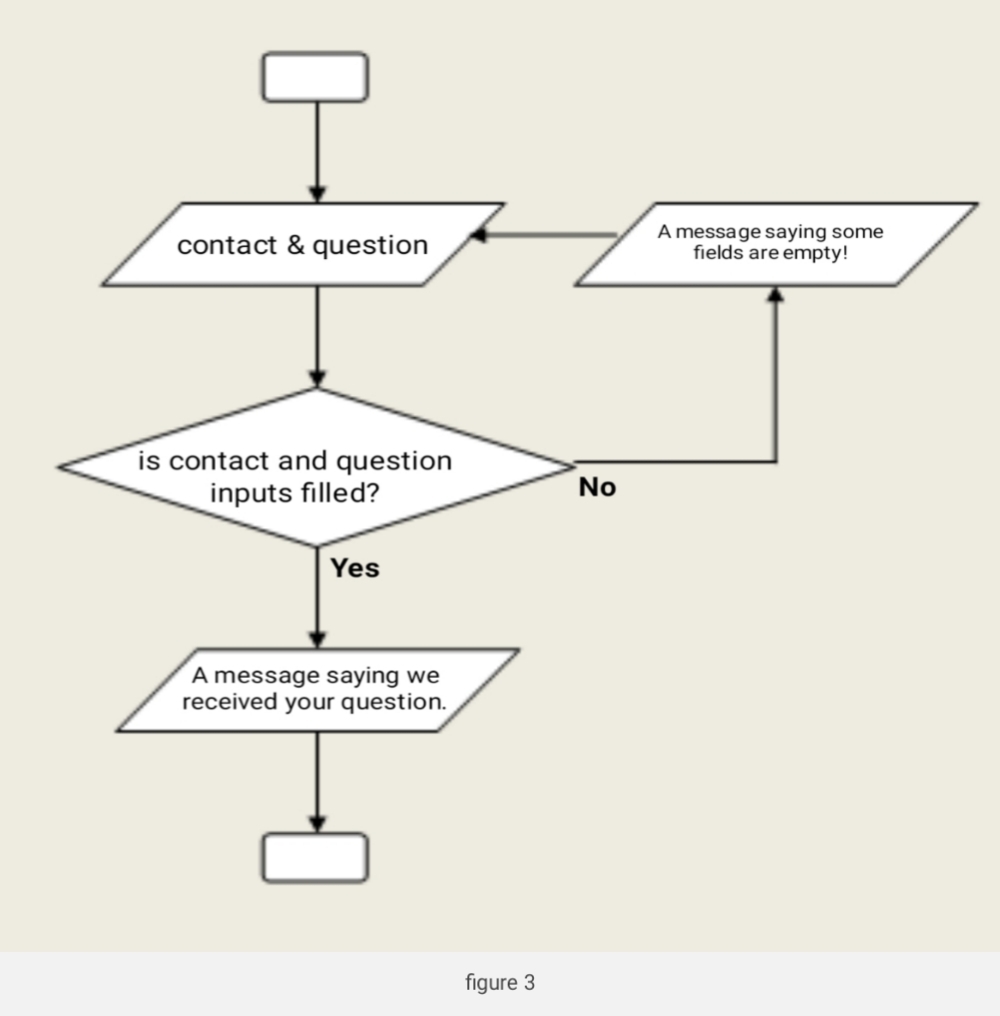
I don’t have to develop the whole page or run a script when adding more lessons in the application, Ajax got me covered.

**Ease of debugging**

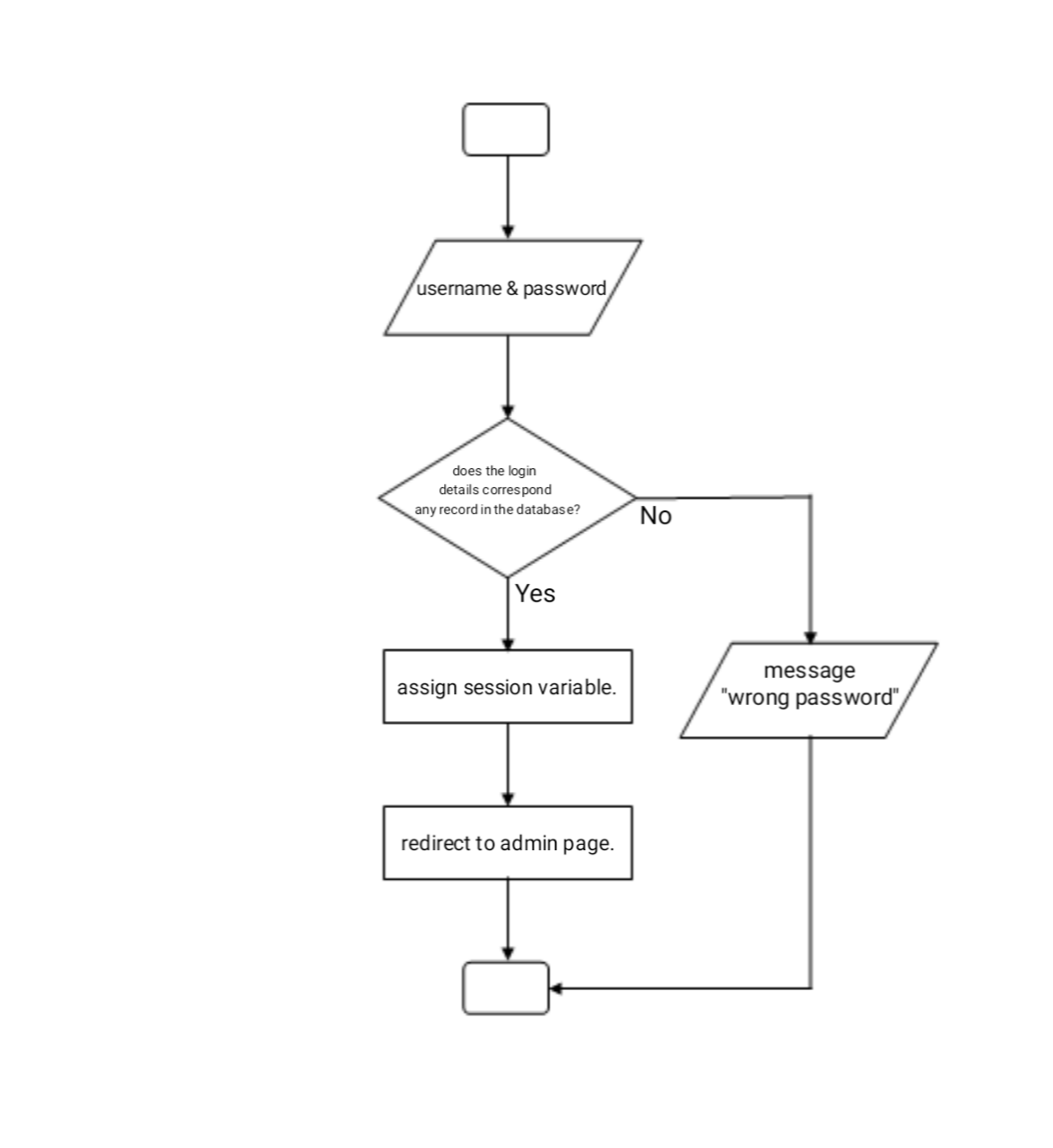
This application developed based on Ajax get method that offer debugging tools, this method returns promise and promise interface allows Ajax $get method to chain multiple .done(), .fail(), and .always() callbacks on a single request, so that will allow me to catch error.

**Index Page Validate User Entry**

Index page validates user entries in ask-me-question textbox and does not allow user to send an empty message to the database, so once user clicked to send a question, it validates the textboxes, a message saying some fields are empty displayed. See below.



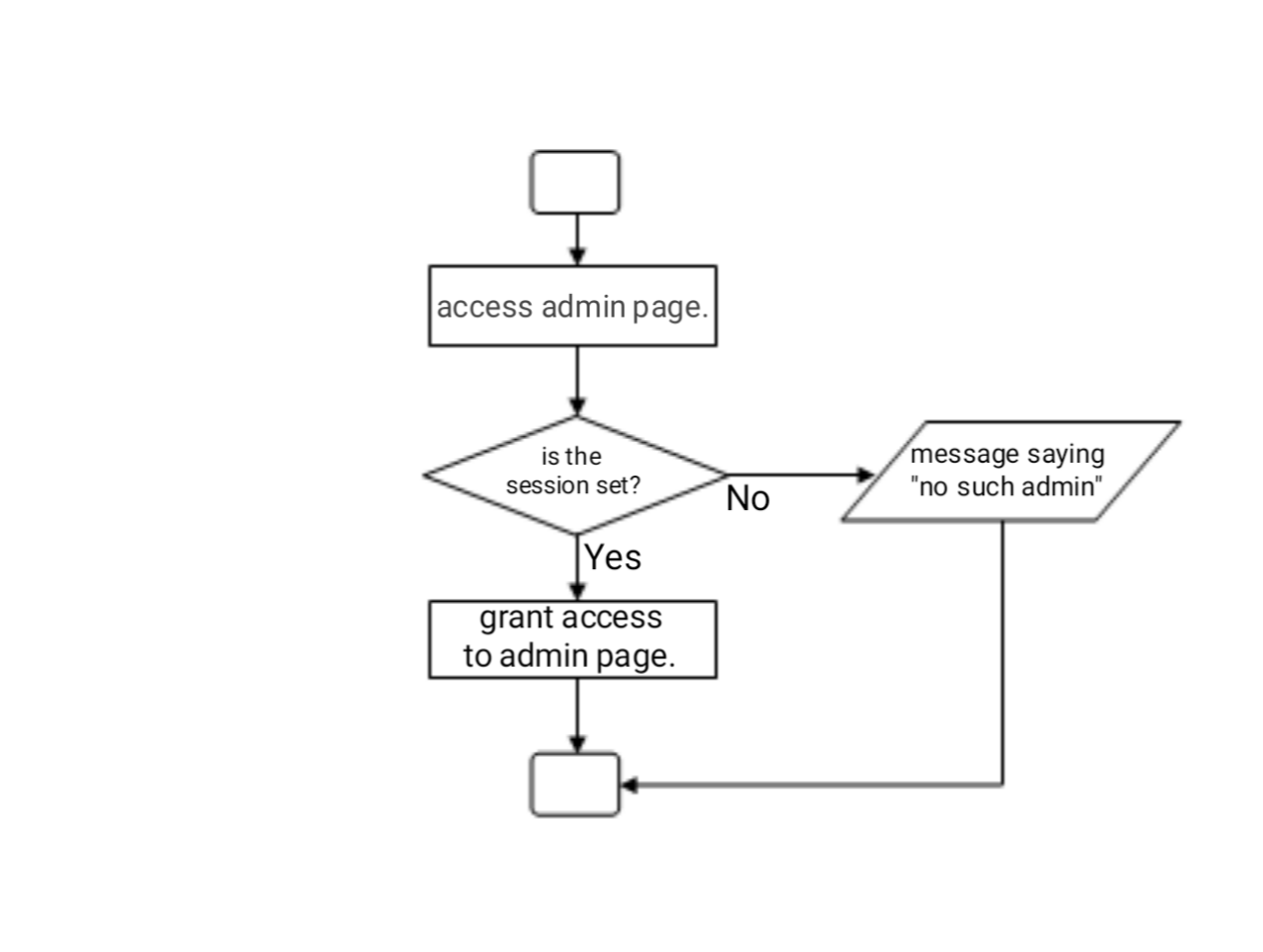
**Admin log in page**  
This page collects admin login details, once the login details entered, it checks the database if the entries correspond any record in the database, then assigns a session variable and redirects the user to admin page; a wrong password message is displayed if the entries are not the same with any record in the database. See below.



**Admin page**

This is the most important page in the system, admin page has access to delete and update the database.

The page contains initial security check using session. In log in page I explained that it assigns a session variable, the session is checked here to allow access to admin page. See below.

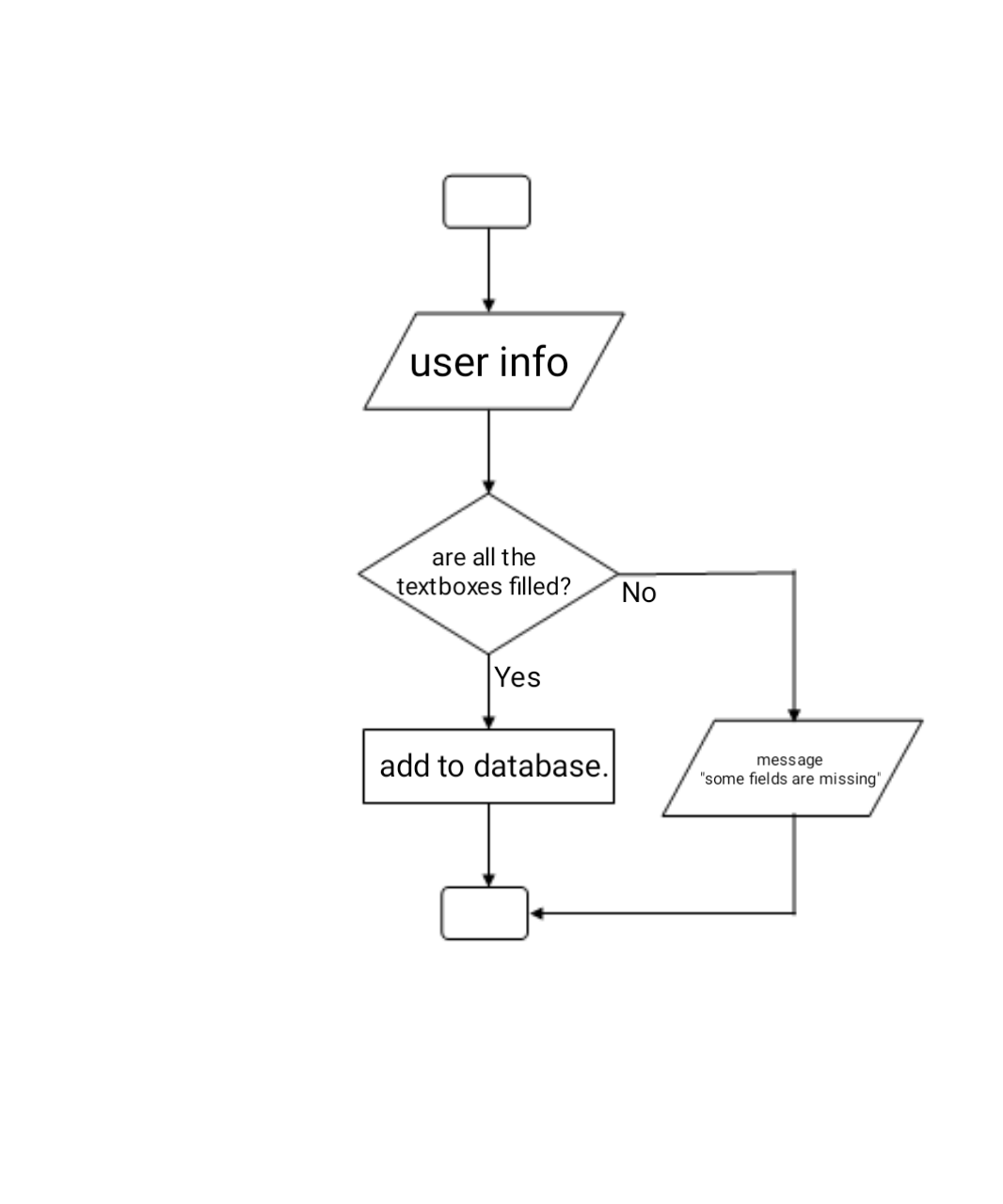


Admin page with all its functionalities share the same function. Admin role is to remove items from the database and approve volunteer using an id assigned for each record in the database.

I designed one function with multiple if condition to determine which task to be done; either remove an item or approve volunteer. I collect an id in the function to carry out the task.

**Volunteer Register Page**

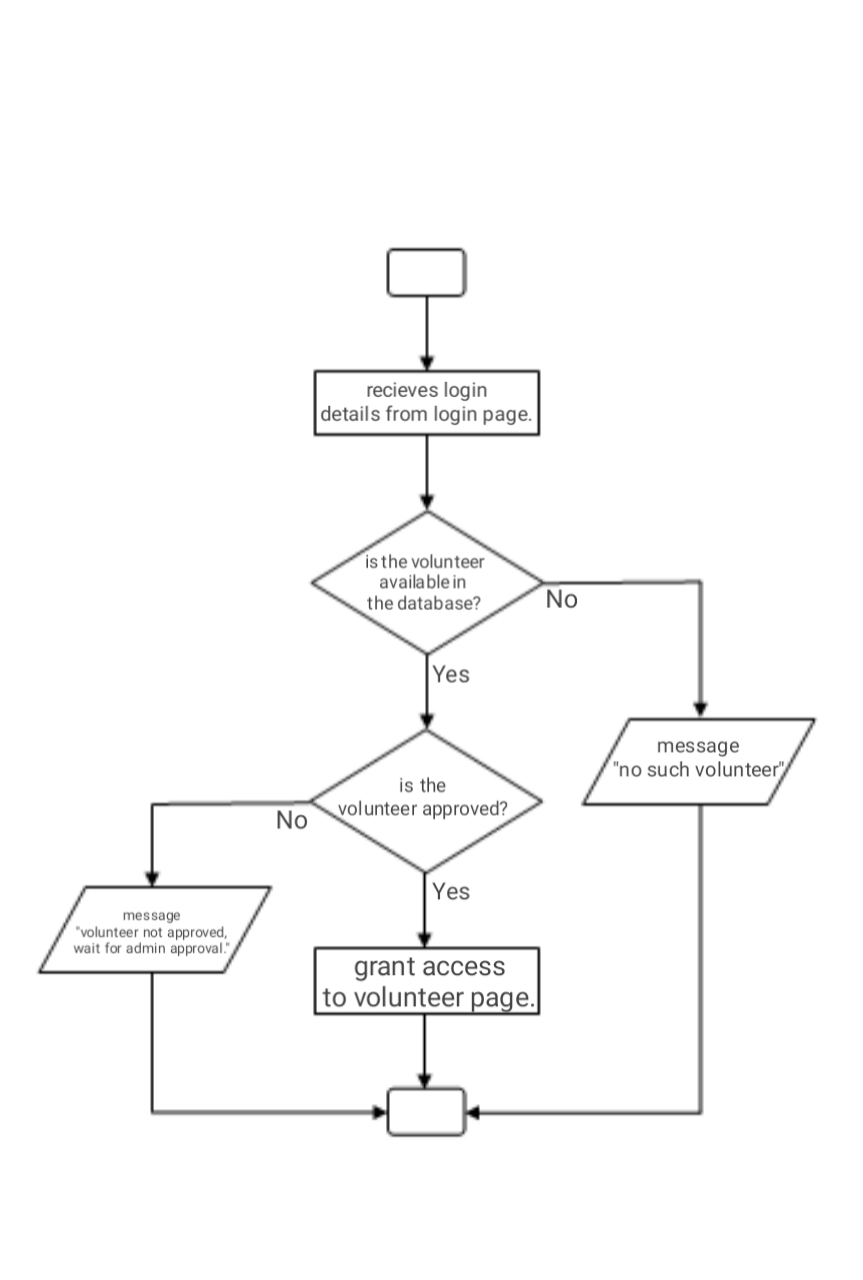
The page collects user data and send to the system database to be validated and approved by the admin. It allows user to send their information and checks if any textbox is empty, a message saying some fields are missing displayed if any of the textbox is empty. See below.



Volunteer login page

The page collects username and password and sends them to volunteer page for validation.

Only approved volunteer can access this page, the page validates user’s data received from login page before it allows access. It checks if the volunteer login detail correspond any record in the database and approved to grant access to the volunteer page, volunteers pending approval get a pending volunteer message as they attempt logging in as shown below.



Volunteer page uses one function as the role is to provide answer and upload it to the database; once the answer got uploaded an id is generated for it to be simply dealt with by the admin.

**Questions and Answers Page**

Once accessed, the page will retreat all the questions and answers and display on the user interface. The page performs a single request to the server.

