Student Name: Dang Khoa Le
Student ID: 103844421
Tutorial Session: Wednesday 8.30 AM

Table of Contents

Education Website Report

1. Introduction

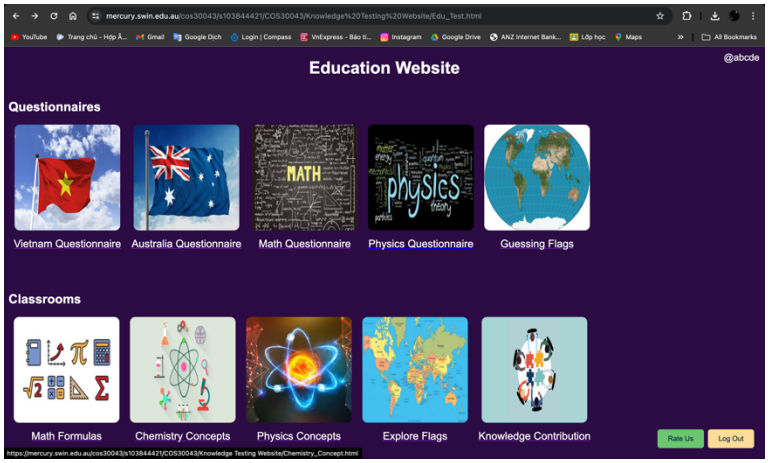
The Educational Website is a comprehensive online platform designed to facilitate learning across various subjects such as Mathematics, Physics, Chemistry, Geography, and History. Leveraging my understanding of HTML, JavaScript, CSS, and constructors such as Vue.js and Bootstrap, the site features an intuitive design that incorporates interactive elements and algorithms to enhance user engagement. Key implementations include:

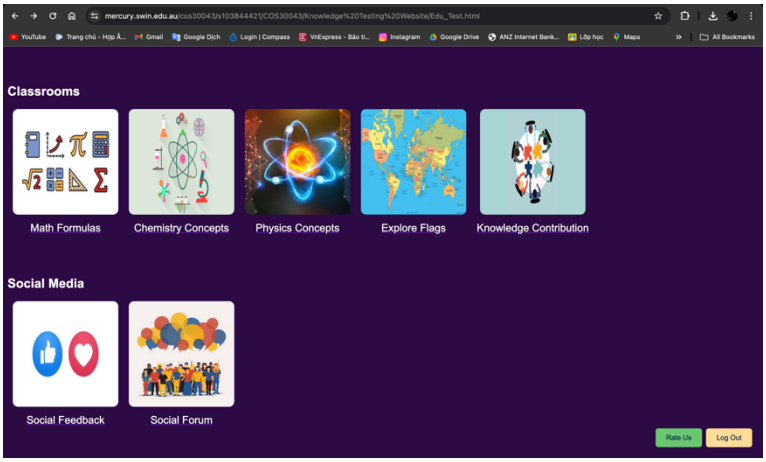
- Utilization of Vue for dynamic content rendering and seamless user interactions.
- Implementation of tables to organize and present information effectively, aiding in the visualization of complex data.
- Filter and Lookup algorithms assorting out data.
- Fetch and binding data files to construct webpage table.
- Integration of credential verification for secure user authentication, enabling seamless login and logout processes.
- Incorporation of APIs to fetch and display relevant content, enriching the user experience with real-time data.
- Quizzes using multiple choices selection.
- Append data array for functional operation.
- Utilising MariaDB SQL database to store and update data as table.
- Utilising Single Page Application (SPA) design with Vue app, router and Vuetify.

2. Interface

a. Menu page:

This page mainly server as a Menu / User Interface page to direct users towards different sites in this project. Which satisfy the requirement of excellently demonstrating usage API.
After processed through the credential step, user is redirected to the Menu page, where different sites are provided from Questionnaires, Classrooms to Social Media pages, users can click on any of those to direct them to the requested page. An image illustrating different sites is also provided for a better visual effect.



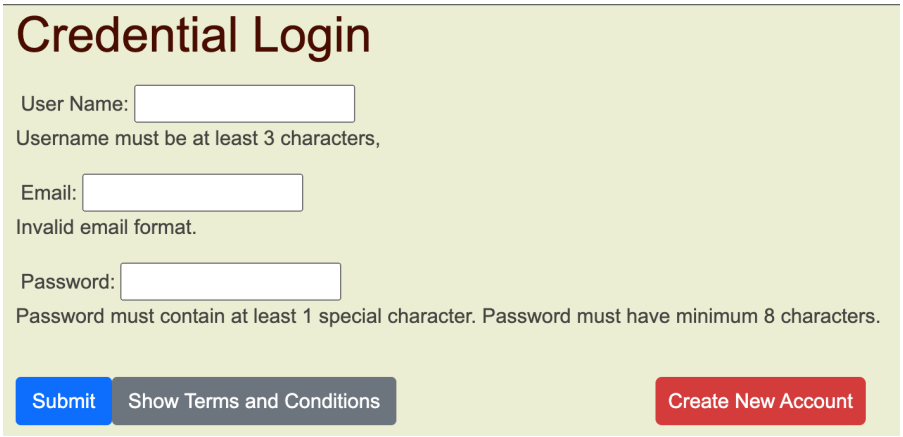


User can also log out when clicked to the “Log Out” button on the bottom right side of the interface and leaves rating on the “Rate Us” button.
The username (or userid variable) is also extracted and displayed on the top right side of this page, corresponding to the user currently using the webpage (shown as “@username”).

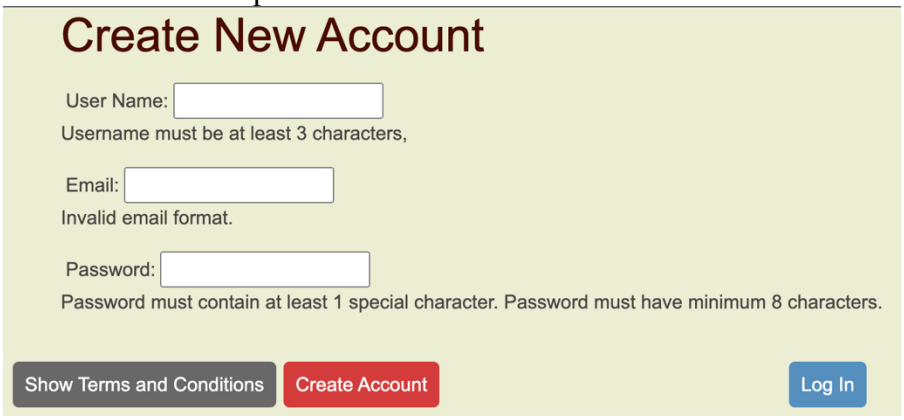
When logging out, the current user data (username, useremail and userpassword) per this login will be removed but not deleted. Hence, user can log in again with a different account without any error existence.

b. Credential:

Credential login functionality where users can Login to their account or Create New Account. User's account will be pushed, sorted, and stored on localStorage, which is saved in an SQLite file in a subfolder in the user's profile.



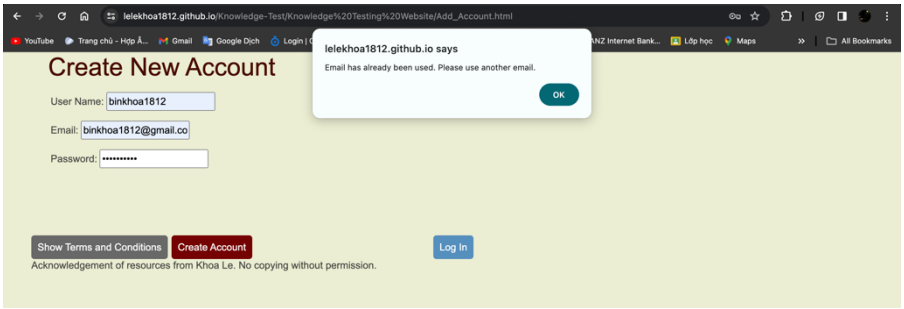
An account consists of username, email and password components. Username must be at least 3 characters, email must contain '@gmail.com' as the domain and password must be at least 8 characters with 1 special character.



Once an account is created, its username will be used to filtrate the corresponding email and password for later attempts to logged in this account, hence, performing accurately all credential activities.

"Terms and Conditions" can also be shown upon clicking.

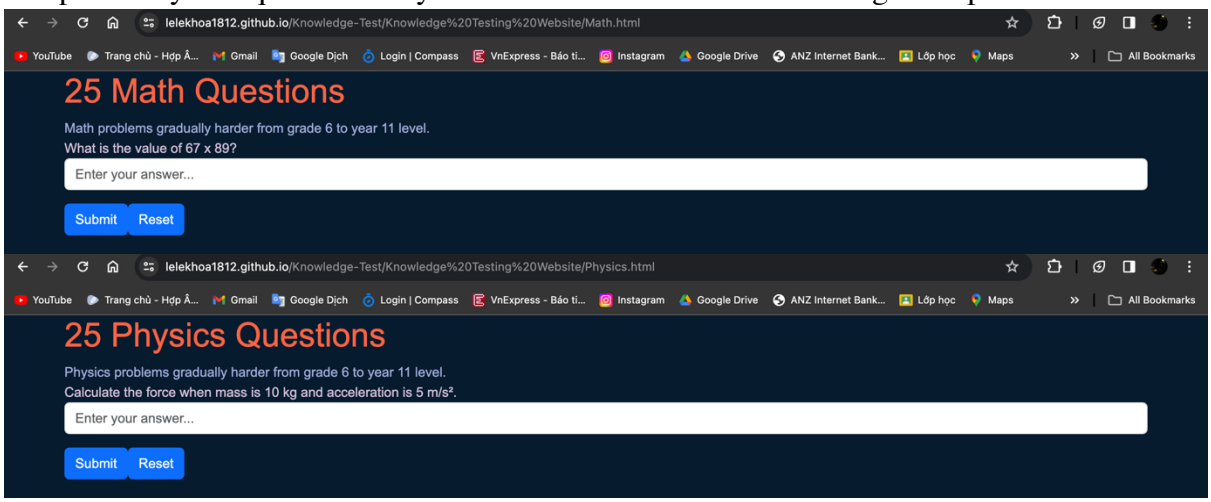
Additionally, an error message will be announced to the user if their account created used an existing username or email (matching any record data of username and email), hence, ensure seamless operation and avoid error.



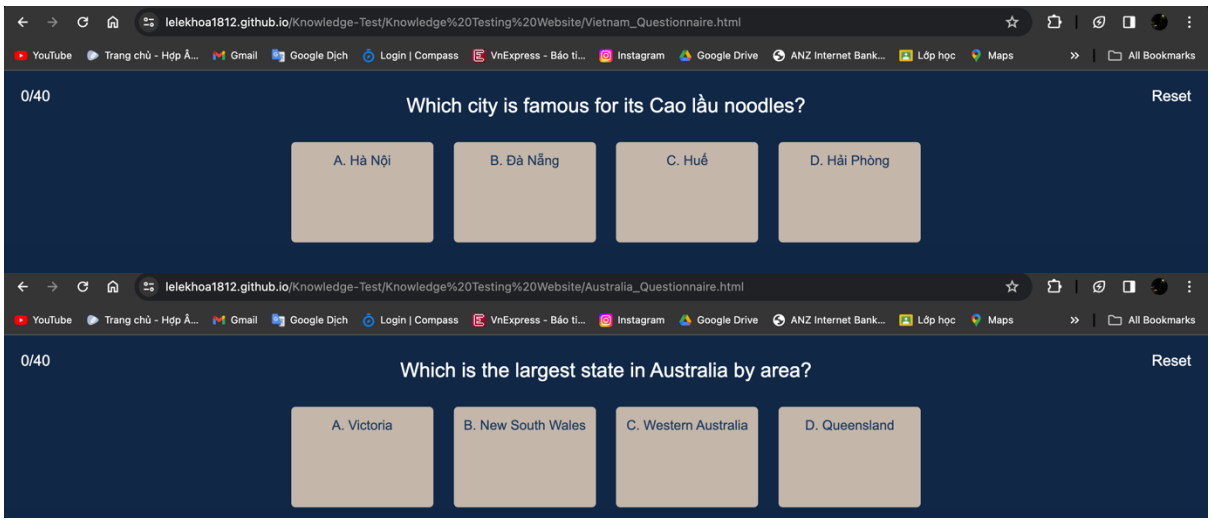
c. Questionnaires:

Questionnaire sites provide an environment for users to test their knowledge on different topics, their point will be calculated and provided to them at the end, reflecting their knowledge base on that given topic.

Multiple scientific-topics Questionnaires are implemented such as Math Questionnaire and Physics Questionnaire, which contains 25 questions from grade 6 to 11. All answers are saved as in the code file as array, while user's answer will be used to check if it matches the answer component by that question array. User can reset their current testing attempt as well.



Country knowledge topics such as Vietnam Questionnaire and Australia Questionnaire present 40 multiple choices tests with topics related to Food, History, Landscape and Geography. These test' order is scrambled so that they will not being displayed on a fixed order sequence. User can reset their current testing attempt as well.



All questions data locate inside the html code file to that webpage as arrays. User's point is incremented and calculated on the top-left side, which point will be used to execute corresponding message to the score, alerting to the user.

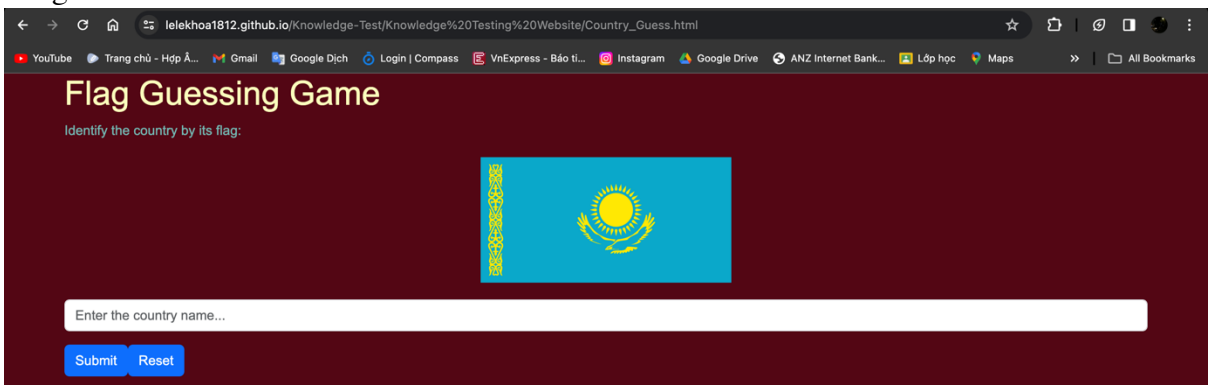
Upon finishing these scientific and countries topic questionnaires, user can also check their answer. Whilst answers are saved in ‘questions’ arrays, these will be forwarded to answer_sheet.html code, which contain algorithms to fetch and construct the question arrays as a table to show user the correct answer to the test they have just done.

@null

Answer Sheet

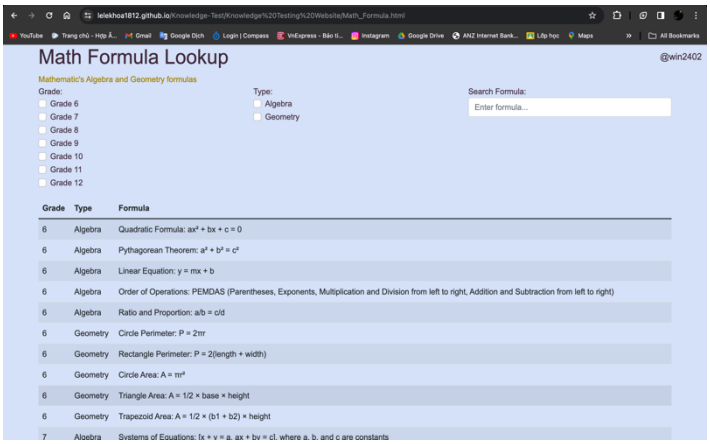
Question	Correct Answer
What is the name of the largest sand island in the world, located off the coast of Queensland, Australia?	Fraser Island
What is a traditional Australian meat pie often served with?	Tomato sauce
Which Australian city is known for its iconic beaches such as Bondi Beach and Manly Beach?	Sydney
Which is the largest island in Australia?	Great Barrier reef
What is the name of the mountain range that runs along the eastern coast of Australia?	Great Dividing Rg
What is the name of the Australian pastry filled with minced meat, onions, and gravy?	Sausage Roll
Which Australian state or territory is home to the Blue Mountains?	NSW
Which is the largest state in Australia by area?	Western Australia
What is the name of the mass protest movement by Aboriginal Australians for land rights and equality?	Tent Embassy
What is the name of the policy that restricted non-European immigration to Australia from 1901 to 1973?	White Australia Policy
When did Australia become a federation and gain independence from the United Kingdom?	1901
What is the name of the dish consisting of grilled steak typically served with fried onions, mushrooms, and a fried egg?	Steak Diane
Who was the Australian Prime Minister during the Sydney 2000 Olympic Games?	John Howard
What is the name of the traditional Australian dessert consisting of meringue, whipped cream, and fresh fruit?	Pavlova
Which Australian state or territory is known as the 'Top End'?	Northern Territory
Who was the first Prime Minister of Australia?	Edmund Barton
Which foreign cuisine is most popular in Australia?	Italian
What is the name of the famous rock formation in the Northern Territory known for its unique shapes and colors?	Kata Tjuta
What event marked the beginning of British colonization of Australia?	Arrival of the First Fleet

A questionnaire “Guessing Flags” is also introduced to test user’s knowledge on distinguishing countries’ flag around the world. This questionnaire used the quite similar website constructions to the Math Questionnaire and Physics Questionnaire, while also importing images data from the server.

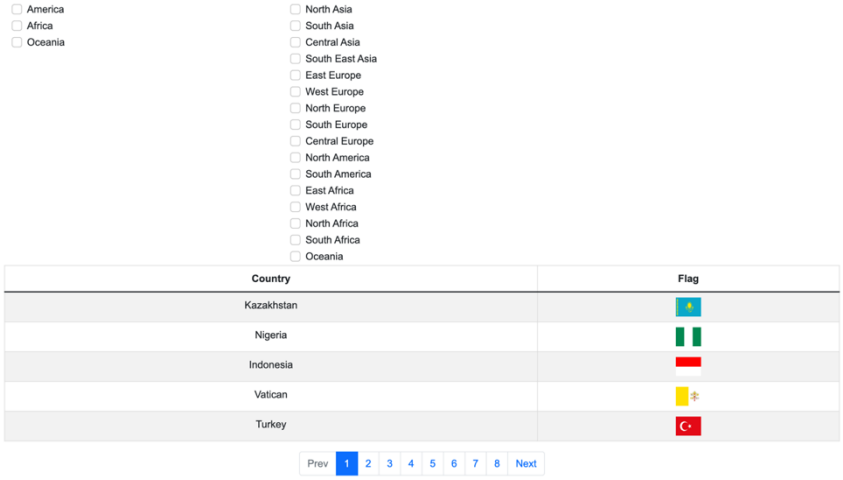


d. Classrooms:

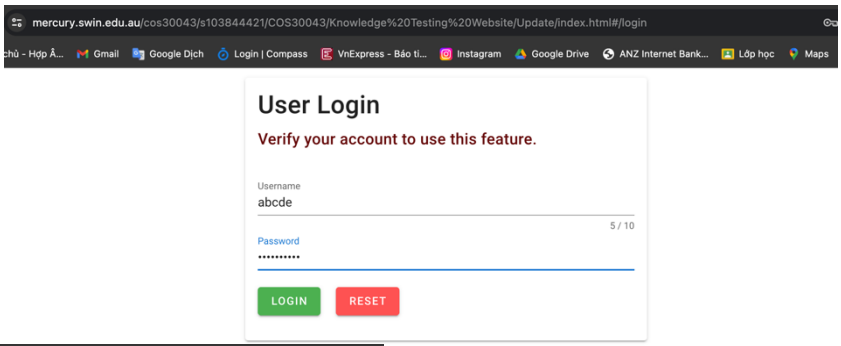
Introducing innovative features, the Classrooms workspace, including Math Formulas (Algebra and Calculus formula from grade 6 to 12), Physics Concepts and Chemistry Concepts (Theory and Formula concepts from grade 6 to 12). All databases are JSON file to be read from the server (i.e. my GitHub account). These data will be formatted by using Vue construction to create table with proper implementation of rows and columns, legends, and lookup filter (search bar and tickbox) to search for any specific data on request.

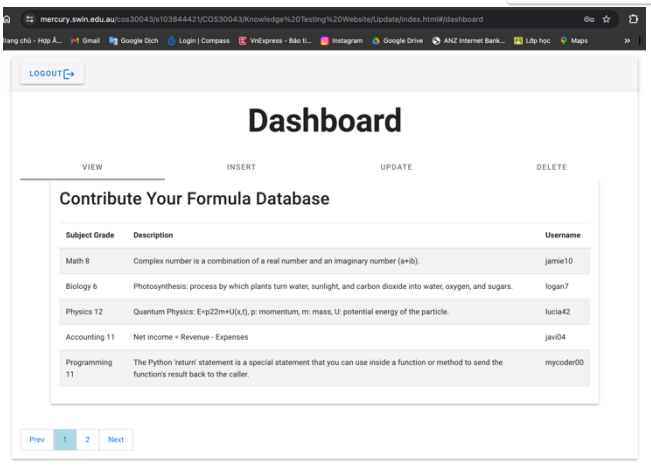


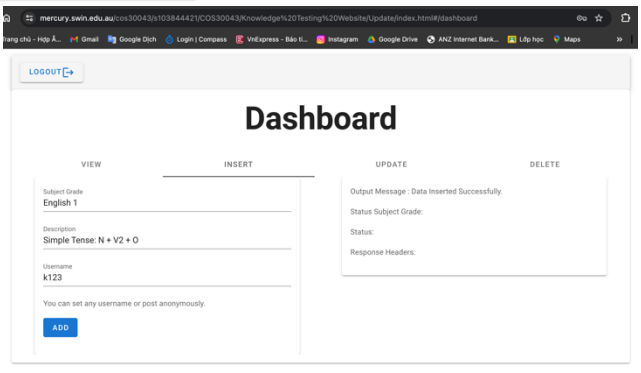
Next, we have Explore Flag site, where user can look up for up to 40 flags from different countries around the world. Flags data is saved on server (my GitHub account) and constructed as a table with that Country's name and the flag image.
This webpage implements pagination feature.



User can filtrate the flag they need to look up as tick on the box to sort them out by Region or Continent. The first 10 flags show the answer to the Guessing Flags questionnaire site as well.
The Knowledge Contribution utilise framework from 9.2C task, allowing user to CRUD (create, read, update, and delete) or View – Insert – Update – Delete a set of Knowledge data base on this web application. This is the environment where user can see and contribute their knowledge regarding any grade level of any subject. This webpage utilises effectively SPA (Single Page Application) design with data binding and parsing functionalities from external server database (MariaDB).
In order to use this webpage services, you have to re-login with the username and password used to logged in this time.







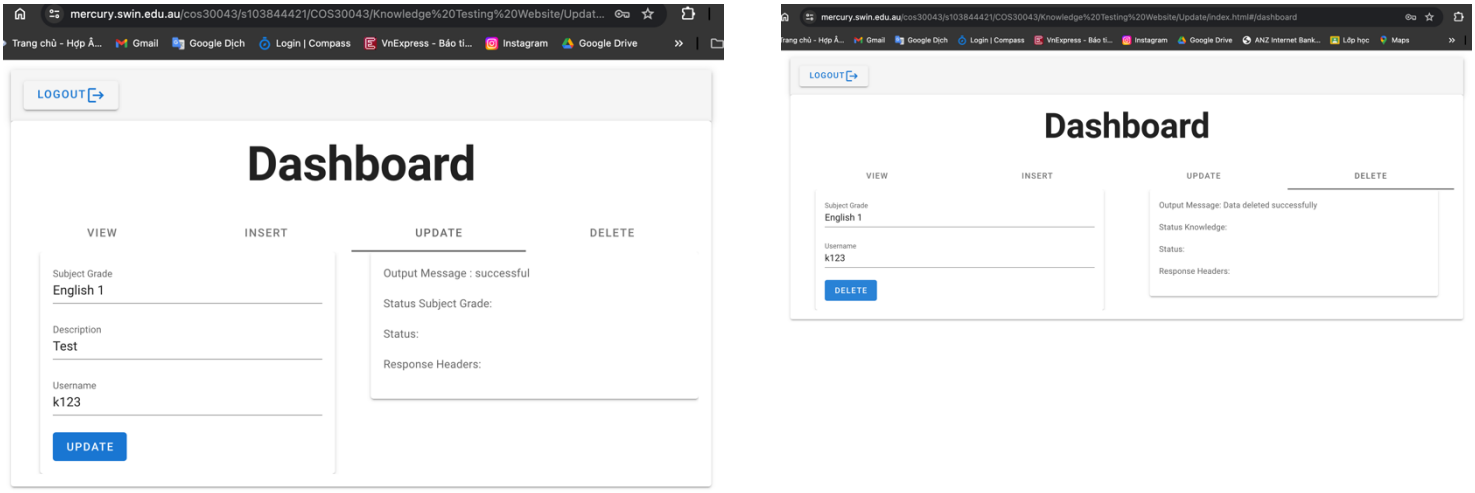
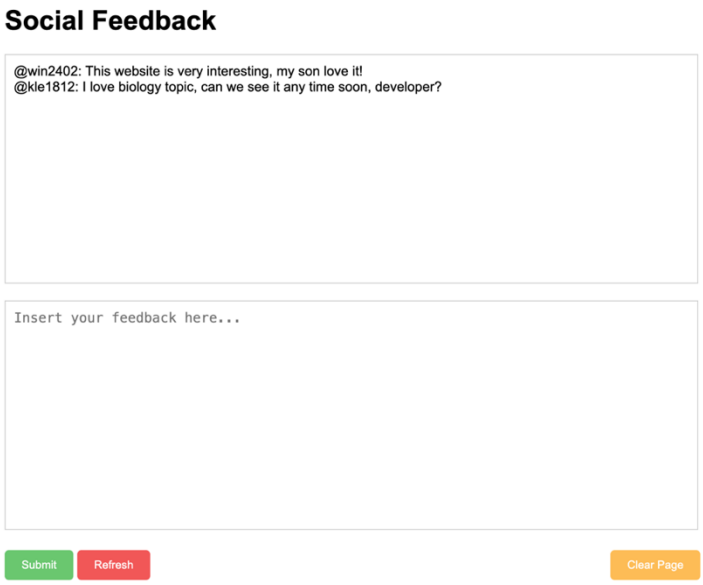


Figure. Login - View - Insert - Update - Delete pages.

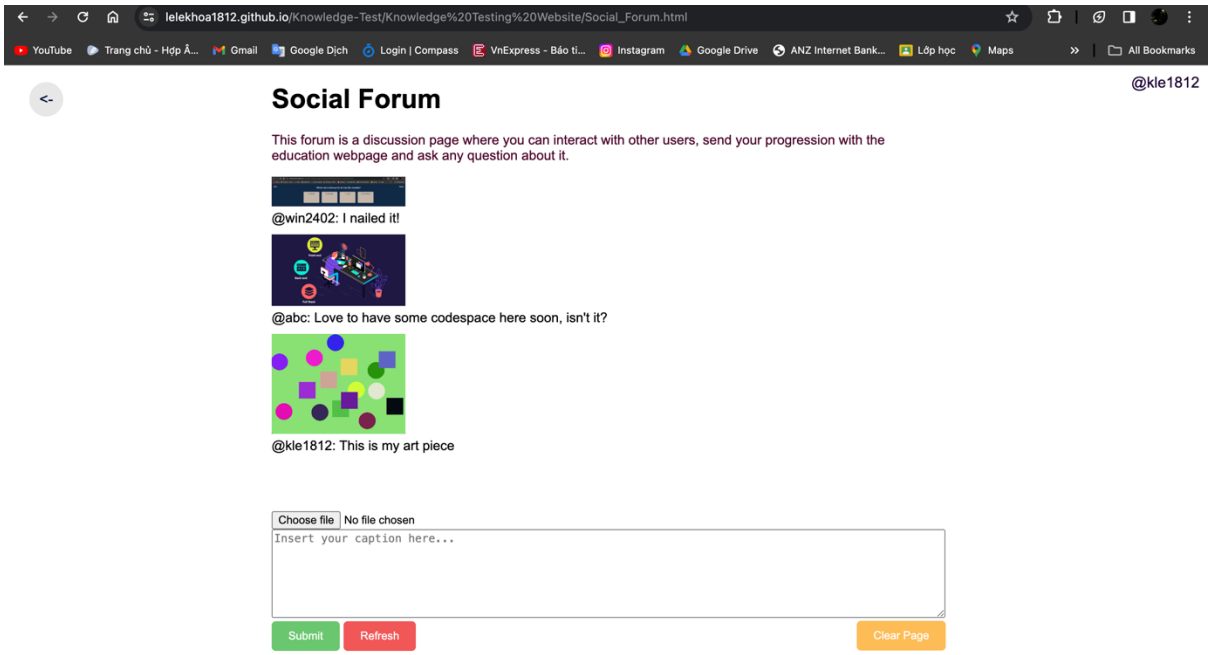
Notice: This webpage can only function properly when accessing via the Mercury server link provided. GitHub page link cannot operate properly this feature.
While using this classroom environment, users can access educational lessons and materials, to expand and acknowledge themselves. These sections utilize Vue construction and MariaDB for efficient data management and storage.

e. Social Media:

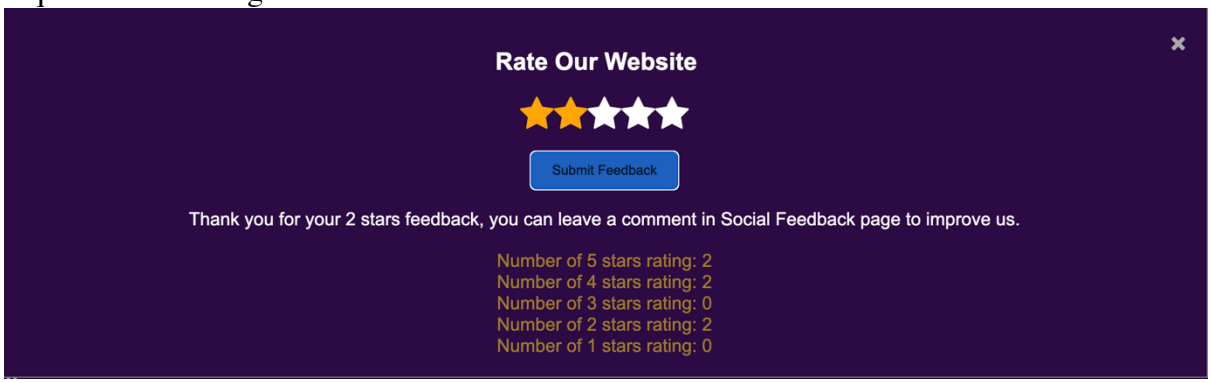
Addition of social networking elements such as Social Feedback and Social Forum, empowering users to provide feedback, communicate with developers, and share their experiences with others, thus fostering a vibrant online community dedicated to learning and knowledge exchange.
Social Feedback webpage allow users to leave feedback as text-messages, enable them to request the developers improving the Website, raising user's interaction.



Meanwhile Social Forum allow users to leave both text-message and upload images dually, allow an open-space for user to discuss and collaborate on their acknowledgement, or share their achievement while using the website.



Upon clicking on the “Rate Us” button from the main menu, a modal box will be displayed to request user’s rating to the website.



Rating functionality also implemented as users can leave their rating from 1 to 5 stars to the developer of the website, the number of feedback-star provided can be save and incremented by type from 1 star to 5 star each time user leaves their feedback about this website. There will be a customised message reflect to the user’s rating.

2. Instruction

When login, user will access the Credential file, and they can either log in to the account they create or use this default account information as below (note that they are only used for testing and development purpose.
Username: liam2003
Email: liam@gmail.com
Password: Kho@le2003.

a. Data storage:

1. Where is localStorage stored? In Google Chrome, web storage data is saved in an SQLite file in a subfolder in the user's profile.

2. Data also saved as SQL table via MariaDB (Account s103844421).

b. Accessibility URLs:

1. Mercury server accessibility:
https://mercury.swin.edu.au/cos30043/s103844421/COS30043/Knowledge%20Testing%20Website/Credential.html
Important Notice: The Contribution feature can ONLY being used from using Mercury server link. If you would like to access them, logged to this Mercury credential information upon request:
Username: s103844421
Password: Kho@le2003

2. GitHub page accessibility:
(https://lelekhoa1812.github.io/Knowledge-Test/Knowledge%20Testing%20Website/Credential.html)

3. YouTube video presentation of this work can be accessed via:
https://youtube.com/watch?v=Nl9Xi0jPzBU

3. Functional Analysis

a. Add_Account.html:

This html code file handle the functional development to enable user to create new account with accuracy and data consistency.
Firstly, I use Vue to construct a text input form that web-user can input their username, email and password from their own. I define these entries with rules such as username must be at least 3 characters using 'minlength="3" required', email must contain '@gmail.com', and password having pattern="(?=.*[@$%&^*])(?=.*[A-Za-z0-9]).{8,}" that the password can be a special character, upper and lower cases, digit and must have minimum of 8 characters. I also define v-show to check for each requirements if the entries may not match those, a span message corresponded to will be shown to suggest user that their credential information is not satisfied. The template for this form input is taken from task 6.1C.

```
<h1>Create New Account</h1>
    <fieldset style="margin-bottom: 20px;">
        <legend></legend>
        <div class="mb-3">
            <label class="mx-1" for="username">User Name:</label>
            <input type="text" name="username" v-model="username" minlength="3" required><br>
            <span v-show="username.length < 3">Username must be at least 3 characters,</span> <!-- username length can't be < 3 -->
        </div>
        <div class="mb-3">
            <label class="mx-1" for="email">Email:</label>
            <input type="email" name="email" v-model="email" required><br>
            <span v-show="!email.match(/@gmail.com/)">Invalid email format.</span> <!-- Email format containing @gmail.com -->
        </div>
        <div class="mb-3">
            <label class="mx-1" for="password">Password:</label>
            <input type="password" name="password" v-model="password" pattern="(?=.*[@$%&^*])(?=.*[A-Za-z0-9]).{8,}" required><br>
            <span v-show="!password.match(/[$%&@^*]/)">Password must contain at least 1 special character.</span> <!-- Special characters such as [$%&@^*]
            <span v-show="password.length < 8"> Password must have minimum 8 characters.</span> <!-- password length can't be < 8 -->
        </div>
    </fieldset>
    <button type="button" class="term-button" @click="toggleTerms">Show Terms and Conditions</button>
    <button type="button" class="newAccount-btn" @click="createAccount">Create Account</button>
    <p class="terms">Acknowledgement of resources from Khoa Le. No copying without permission.</p>
    <button type="button" class="login-btn" @click="loginAccount">Log In</button>
</form>
```

In this web design, all user credential information will be set and saved to localStorage under the item "users".
Then, we create a Vue app with data of username, email and password. The web will get (JSON.parse method) all 'users' items data (or return as an empty string if no 'users' item has

been created). It will check to see if any username and email under the item ‘users’ may match the corresponding entry field from the user’s form input. If there are, the web will alert error and return, therefore, not allowing same username and email information, which can create data inconsistency.
If there are no error, then the username, email and password from the user form input will be push to the ‘users’ item, which will be set (saved) to localStorage using setItem method. Afterwards, the webpage will redirect user back to the Credential page for log in using window.location.replace method.

```
Vue.createApp({
    data() {
        return {
            username: "",
            email: "",
            password: "",
        }
    },
    methods: {
        createAccount() {
            // Get existing user data or initialize an empty array (in case no existing user data).
            // JSON stringtify references from https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON/stringify
            let users = JSON.parse(localStorage.getItem('users')) || [];

            // Check if username or email already exists
            if (users.some(user => user.username === this.username)) {
                alert("Username has already been used. Please choose another username.");
                return;
            }
            if (users.some(user => user.email === this.email)) {
                alert("Email has already been used. Please use another email.");
                return;
            }

            // Push new user data array to the user item
            users.push({
                username: this.username,
                email: this.email,
                password: this.password
            });

            // Store and set new user data back to localStorage
            localStorage.setItem("users", JSON.stringify(users));

            // Redirect to the Credential page
            window.location.replace("Credential.html");
        },
```

We also have the Log In button to redirect user to the Credential.html page in case they have already got an account.

b. Credential.html:

This webpage handle credential information for login validation. User can only use the website’s features and services if they login.
At the first stage, we also create the similar form entry to the Add_Account.html file with similar rules.
Then, we also create a Vue app, with data of username, email and password. Similarly, we will parse all ‘users’ item or return with an empty string if none has been created. After that, we will set conditional equation that whether the username, email and password from the form match with the 2 default account (testing usage) or create a boolean constant ‘match’ that if any ‘user’ data from ‘users’ item may match with the form entries, then the web swill set items of userid (as the username), useremail (as the email) and userpassword (as the password) from the form entries to localStorage using localStorage.setItem method, these 3 items will be used later when the web-user experience the webpage. Afterwards, if it is validated, we redirect user to the Menu page (Edu_Test.html) or return with an error message.

```
Vue.createApp({
    data() {
        return {
        username: "",
        email: "",
        password: "",
        }
    },
    methods: {
        validation() {
            // Get existing user data or initialize an empty array (in case no existing user data).
            // JSON stringtify references from https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON/stringify
            let users = JSON.parse(localStorage.getItem('users')) || [];

            // Default accounts (testing, debug, maintainance and update purposes usage only)
            if (
                (this.username === "kle1812" && this.email === "kle@gmail.com" && this.password === "Kho@le2003") ||
                (this.username === "win2402" && this.email === "win@gmail.com" && this.password === "Kho@le2003")
            ) {
                // Store user's info to localStorage for this log in attempt temporarily
                localStorage.setItem("userid", this.username);
                localStorage.setItem("useremail", this.username);
                localStorage.setItem("userpassword", this.username);
                // Redirect to Menu page
                window.location.replace("Edu_Test.html");
                return;
            }
            else {
                // Check if user credentials match any stored user data, create boolean constant
                const match = users.some(user => user.username === this.username && user.email === this.email && user.password === this.passwor
                if (match) {
                // Store user's info to localStorage for this log in attempt temporarily
                    localStorage.setItem("userid", this.username);
                    localStorage.setItem("useremail", this.email);
                    localStorage.setItem("userpassword", this.password);
                    // Redirect to Meu page
                    window.location.replace("Edu_Test.html");
                } else {
                    alert("Wrong Username, Email, or Password!");
                }
            }
        },
```

We also have the Create New Account button to redirect user to the Add_Account.html page.

c. Edu_Test.html:

This is the webpage serving as a Menu where user can access different features in this web-design, including Questionnaires, Classrooms and Social Media, alongside with the Rating and Log Out function.

Each webpage in this web-design will have a lightbulb icon, using this code line.

```
<link rel="website icon" type="png" href="lb.png" >  <!-- show website icon -->
```

The top right side of this page will display the userid (username) of the user, fetched from the localStorage, and then append as a text content to user-id container with a '@' symbol at beginning.

```
const userid = localStorage.getItem('userid'); // Extract userid (username) component saved temporarily
document.querySelector('.user-id').textContent = '@' + userid; // userid shown after @ symbol
```

Then, we have a set of containers, contain the image (from source), description text and corresponding destination link to the webpage feature, redirect web-user upon where they clicking to (using href method). Each of these containers will also being sorted by row to their type, Questionnaires, Classrooms and Social Media with a header to that type on top.

```html
<div class="container">  <!-- List of menu components (image, description, redirect) -->
    <!-- Row 1: Questionnaires -->
    <div class="row">
        <h2>Questionnaires</h2>
        <div class="image-container">
            <a href="Vietnam_Questionnaire.html">
                <img src="Questionnaire/vn.jpeg" alt="Vietnam Questionnaire">
                <div class="image-text">Vietnam Questionnaire</div>
            </a>
        </div>
        <div class="image-container">
            <a href="Australia_Questionnaire.html">
                <img src="Questionnaire/au.jpeg" alt="Australia Questionnaire">
                <div class="image-text">Australia Questionnaire</div>
            </a>
        </div>
        <div class="image-container">
            <a href="Math.html">
                <img src="Questionnaire/math.jpeg" alt="Math Questionnaire">
                <div class="image-text">Math Questionnaire</div>
            </a>
        </div>
        <div class="image-container">
            <a href="Physics.html">
                <img src="Questionnaire/phy.jpeg" alt="Physics Questionnaire">
                <div class="image-text">Physics Questionnaire</div>
            </a>
        </div>
        <div class="image-container">
            <a href="Country_Guess.html">
                <img src="Questionnaire/world.jpeg" alt="Guessing Flags">
                <div class="image-text">Guessing Flags</div>
            </a>
        </div>
    </div>
    <!-- Row 2: Classrooms -->
    <div class="row">
        <h2>Classrooms</h2>
        <div class="image-container">
            <a href="Math_Formula.html">
                <img src="Classroom/math_formula.jpeg" alt="Math Formulas">
                <div class="image-text">Math Formulas</div>
            </a>
```

Next, we have the rating feature. Upon clicking to the rating-button (Rate Us), a modal box will be displayed (hidden at default) with fall back color and opacity, z-index order and position as below CSS, this box shows the star rating request that once user clicked on the corresponding star, they will be display and saved to localStorage.

```css
.rating-button {
    position: fixed;
    bottom: 20px;
    right: 115px;
    background-color: #6ac66f;
    color: rgb(0, 34, 70);
    padding: 10px 20px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s ease;
}
.rating-button:hover {
    background-color: #0d450f;
}
/* Modal box for feedback */
.modal {
    display: none; /* Hidden by default */
    position: fixed; /* Fixed position */
    z-index: 1; /* On top of everything else */
    left: 0;
    top: 0;
    width: 100%;
    height: 100%;
    overflow: auto; /* Enable scroll */
    background-color: rgb(0,0,0); /* Fallback color */
    background-color: rgba(0,0,0,0.4); /* Black with opacity */
}
.modal-content {
    background-color: #2c0b44;
    margin: 15% auto; /* 15% from top, centered */
    padding: 20px;
    border: 1px solid #888;
    width: 80%;
}
.close {
    color: #aaa;
    float: right;
    font-size: 28px;
    font-weight: bold;
}
```

So firstly, I create span classes displaying the star (fa-star) with toggle feature (light up or down) with onmouseover checking user's cursor position on mouse position. Then, I have the function to set 'stars' constant on index value, for instance, if cursor is on the second star, star with index < 2 (star 1 and 2 on the left side) and 'fa-star' will be checked and therefore lighten up, the rest not being lighten.

Then, once user click the “Submit Feedback” button, it call the acquireFeedback function. The ‘stars’ constant will be acquired by checking the ‘fa-star’ with value checked, set them by length. The difference at the ‘stars’ value will reflect a different customised message to the web-user.

Then we have the obtainFeedback function running after that. It set the ‘stars’ constant’s length, then parse the ‘website-ratings’ array item, which is an index with 5 variable (or return as an array of five 0s that if no one has ever rated the webpage. Next, the ‘website-ratings’ item will increment the corresponding array index based on the corresponding ‘stars’ length using ‘websiteRatings[stars - 1]++;’ (it minus 1 as the index is from 0 to 4 but ‘stars’ value is from 1 to 5). Afterwards, it filtrate the ‘website-ratings’ array by index (0 to 4) and display the message showing the total time rated per each stars type (1 to 5 stars).

```
<!-- Rating content, references from https://www.w3schools.com/howto/tryit.asp?filename=tryhow_css
<!-- Modal box, references from https://www.w3schools.com/howto/howto_css_modals.asp -->
<div id="ratingModal" class="modal">
    <div class="modal-content">
        <span class="close" onclick="closeRatingModal()">&times;</span>  <!-- close modal box upon
        <h2>Rate Our Website</h2>
        <div class="stars">  <!-- display star and set mouseover position on star -->
            <span class="fa fa-star" onmouseover="toggleStars(1)"></span>
            <span class="fa fa-star" onmouseover="toggleStars(2)"></span>
            <span class="fa fa-star" onmouseover="toggleStars(3)"></span>
            <span class="fa fa-star" onmouseover="toggleStars(4)"></span>
            <span class="fa fa-star" onmouseover="toggleStars(5)"></span>
        </div>
        <button class="submitFeedback-button" onclick="acquireFeedback()">Submit Feedback</button>
        <div id="feedbackMessage" class="feedback-message"></div>
        <div id="feedbackRating" class="rating-message"></div>
    </div>
</div>
// Function to toggle star colors based on mouseover
function toggleStars(num) {
    const stars = document.querySelectorAll('.fa-star');
    for (let i = 0; i < stars.length; i++) { // Per mouseover, it set all star to the left toggled
        if (i < num) {
            stars[i].classList.add('checked');
        } else {
            stars[i].classList.remove('checked');
        }
    }
}
```

```
// Function to send message accquiring feedback
function acquireFeedback() {
    const stars = document.querySelectorAll('.fa-star.checked').length; // Checking number of star acquired, therefore set feedback message upon ratting type
    const feedbackMessage = document.getElementById('feedbackMessage');
    if (stars >= 3) {
        feedbackMessage.textContent = "Thank you, we are happy with your " + stars + " stars feedback!";
    } else {
        feedbackMessage.textContent = "Thank you for your " + stars + " stars feedback, you can leave a comment in Social Feedback page to improve us.";
    }
    obtainFeedback();
}

// Function to saved ratting type per attempt and send message obtaining feedback count (with increment)
function obtainFeedback() {
    const stars = document.querySelectorAll('.fa-star.checked').length;
    // Array to store counts for each star rating, reference from https://www.geeksforgeeks.org/what-is-the-most-efficient-way-to-create-a-zero-filled-array-
    // Get current website ratings from localStorage
    let websiteRatings = JSON.parse(localStorage.getItem('website_ratings')) || [0, 0, 0, 0, 0];
    // Increment the count for the selected star rating
    websiteRatings[stars - 1]++;
    // Update website ratings in localStorage
    localStorage.setItem('website_ratings', JSON.stringify(websiteRatings));

    // Display the feedback counts
    const feedbackRating = document.getElementById('feedbackRating');
    feedbackRating.innerHTML = `
        Number of 5 stars rating: ${websiteRatings[4]}<br>
        Number of 4 stars rating: ${websiteRatings[3]}<br>
        Number of 3 stars rating: ${websiteRatings[2]}<br>
        Number of 2 stars rating: ${websiteRatings[1]}<br>
        Number of 1 stars rating: ${websiteRatings[0]}<br>
    `;
}
```

Finally, the Menu page has the Log Out button that once clicked onto, it remove the userid, useremail and userpassword items that has been set to localStorage when log in. This is to ensure data consistency and allowing web-user to log in with a different account each time.

```
function logout() {
    //localStorage.removeItem('users'); // Remove all users data in local storage, please commend this line, only for testing purpose.
    // Remove the temporary - current logged in user info per log out
    localStorage.removeItem('userid');
    localStorage.removeItem('useremail');
    localStorage.removeItem('userpassword');
    window.location.href = 'Credential.html';
}
```

d. Vietnamese_Questionnaires.html and Australia_Questionnaires.html:

These 2 webpages are multiple choices quizzes testing user’s understanding about the 2 countries. Both webpage has similar code construction, only having the different s’s arrays.

We started with creating the answer-box containers, which show the available answer choices to the user, which they can click onto to lock in their answer. Then we have the score displaying on the left side, the Return button which redirect users to the Menu and Check Answer button to show the correct answer sheet after finishing (the 2 button initially hidden or display as none).

```
<div id="score">0/40</div>
<div id="reset" onclick="resetGame()">Reset</div>
<div class="container">
    <div class="question" id="question"></div>
    <div class="answer-box" id="answerA" onclick="checkAnswer('A')"></div> <!-- Set onclick position for user choice -->
    <div class="answer-box" id="answerB" onclick="checkAnswer('B')"></div>
    <div class="answer-box" id="answerC" onclick="checkAnswer('C')"></div>
    <div class="answer-box" id="answerD" onclick="checkAnswer('D')"></div>
</div>
<div id="returnButton" class="container" style="display: none;">
    <button onclick="returnSite()" class="btn btn-primary btn-lg mt-3">Return</button> <!-- Return button shown after finish the game -->
</div>
<div id="checkAnswerButton" class="container" style="display: none;">
    <button onclick="showAnswerSheet()" class="btn btn-primary btn-lg mt-3">Check Answer</button> <!-- Check Answer button shown after finish the game -->
</div>
```

Then, we create the ‘questions’ array with the question, 4 available answer choices and the correct answer to the matching question.

```
// Define question, available answers and correct answer as array
const questions = [
    // Geography
    { question: "What is the capital city of Australia?", answers: ["A. Sydney", "B. Melbourne", "C. Canberra", "D. Perth"], correctAnswer: "C" },
    { question: "Which desert covers most of central Australia?", answers: ["A. Sahara Desert", "B. Mojave Desert", "C. Gobi Desert", "D. Simpson Desert"], c
    { question: "Which is the largest state in Australia by area?", answers: ["A. Victoria", "B. New South Wales", "C. Western Australia", "D. Queensland"],
    { question: "Which river is the longest river in Australia?", answers: ["A. Yarra River", "B. Darling River", "C. Lachlan River", "D. Murray-Darling"], c
    { question: "Which is the largest island in Australia?", answers: ["A. Tasmania", "B. Fraser Island", "C. Kangaroo Island", "D. Great Barrier reef"], cor
    { question: "What is the tallest mountain in Australia?", answers: ["A. Mount Kosciuszko", "B. Mount Bogong", "C. Mount Ossa", "D. Mount Townsend"], corr
    { question: "What is the largest coral reef system in the world, located off the coast of Queensland, Australia?", answers: ["A. Great Barrier Reef", "B.
    { question: "Which Australian state or territory is known as the 'Top End'?", answers: ["A. Western Australia", "B. Northern Territory", "C. South Austra
    { question: "What is the name of the largest sand island in the world, located off the coast of Queensland, Australia?", answers: ["A. Fraser Island", "B
    { question: "Which Australian state or territory is known for its wine regions such as Barossa Valley and McLaren Vale?", answers: ["A. New South Wales",
    // Food
    { question: "What is the name of the spread that is popular in Australia, made from yeast extract?", answers: ["A. Vegemite", "B. Marmite", "C. Nutella",
    { question: "What is a traditional Australian meat pie often served with?", answers: ["A. Tomato sauce", "B. Mustard", "C. Gravy", "D. Ketchup"], correct
    { question: "Which fruit is most consumned in Australia?", answers: ["A. Mango", "B. Banana", "C. Grape", "D. Apple"], correctAnswer: "B" },
    { question: "What is a popular Australian dessert made from a sponge cake filled with jam and cream, and dusted with powdered sugar?", answers: ["A. Anza
    { question: "What is the name of the dish consisting of grilled steak typically served with fried onions, mushrooms, and a fried egg?", answers: ["A. Fis
    { question: "What is the name of the Australian pastry filled with minced meat, onions, and gravy?", answers: ["A. Empanada", "B. Samosa", "C. Pastie", "
    { question: "Which foreign cuisine is most popular in Australia?", answers: ["A. Thai", "B. Greek", "C. Italian", "D. Vietnamese"], correctAnswer: "C" },
    { question: "Which ice cream is most famous in Australia?", answers: ["A. Splice", "B. Cornetto", "C. Golden Gaytime", "D. Maxibon"], correctAnswer: "C"
    { question: "What is the name of the traditional Australian dessert consisting of meringue, whipped cream, and fresh fruit?", answers: ["A. Lemon Tart",
    { question: "Which Australian city is known for its coffee culture and laneway cafes?", answers: ["A. Sydney", "B. Melbourne", "C. Brisbane", "D. Perth"]
    // History
    { question: "Who is considered the first European to discover Australia?", answers: ["A. James Cook (British)", "B. Abel Tasman (Dutch)", "C. Chirstopher
    { question: "When did Australia become a federation and gain independence from the United Kingdom?", answers: ["A. 1776", "B. 1901", "C. 1945", "D. 2000"
    { question: "What is the name of the policy that restricted non-European immigration to Australia from 1901 to 1973?", answers: ["A. White Australia Poli
    { question: "Who was the first Prime Minister of Australia?", answers: ["A. John Howard", "B. Robert Menzies", "C. Edmund Barton", "D. Alfred Deakin"], c
    { question: "What event marked the beginning of British colonization of Australia?", answers: ["A. Arrival of the First Fleet", "B. Australian Gold Rushe
    { question: "Who was the Indigenous Australian leader known for his resistance against British settlement in Tasmania?", answers: ["A. Truganini", "B. Ja
    { question: "What is the name of the Indigenous Australian art form characterized by dot painting?", answers: ["A. Dreamtime Art", "B. X-Ray Art", "C. Po
    { question: "Which Australian state or territory was the first to grant women the right to vote?", answers: ["A. Victoria", "B. New South Wales", "C. Sou
    { question: "What is the name of the mass protest movement by Aboriginal Australians for land rights and equality?", answers: ["A. Freedom Ride", "B. Ten
    { question: "Who was the Australian Prime Minister during the Sydney 2000 Olympic Games?", answers: ["A. Paul Keating", "B. John Howard", "C. Bob Hawke",
```

Then, I create the currentQuestionIndex, which index present your current question order over 40 and the score variable as your total correct answer out of 40. We have the displayQuestion function, firstly create the current currentQuestion constant, which show the ‘question’ row from ‘questions’ array corresponded to the currentQuestionIndex which show the 4 answer-box A, B, C, D filtrated by the answers (available answer choices) from the ‘questions’ array, corresponded to the currentQuestion constant. So therefore, the text content by each available answer will be displayed per each answer-box.

```
let currentQuestionIndex = 0;
let score = 0;

// Display initial question with matching data
displayQuestion();

function displayQuestion() {
    const currentQuestion = questions[currentQuestionIndex];
    document.getElementById("question").textContent = currentQuestion.question;
    document.getElementById("answerA").textContent = currentQuestion.answers[0];
    document.getElementById("answerB").textContent = currentQuestion.answers[1];
    document.getElementById("answerC").textContent = currentQuestion.answers[2];
    document.getElementById("answerD").textContent = currentQuestion.answers[3];
}

// If answer correct, increment score
function checkAnswer(selectedAnswer) {
    const currentQuestion = questions[currentQuestionIndex];
    if (selectedAnswer === currentQuestion.correctAnswer) {
        score++;
    }
    currentQuestionIndex++; // Move to next question
    if (currentQuestionIndex < questions.length) { // If remain question, display, else gameover
        displayQuestion();
    } else { // Show customized message upon score achieved
        alert("Game Over! Your score: " + score + "/40");
        if (score < 10) {
            alert("You need to practice more");
        } else if (score >= 10 && score < 20) {
            alert("Good try but still need practice.");
        } else if (score >= 20 && score < 30) {
            alert("Good attempt, keep learning.");
        } else if (score >= 30 && score < 40) {
            alert("That's great, almost there.");
        } else if (score === 40) {
            alert("Awesome, you beat it!");
        }
        if (currentQuestionIndex === questions.length) { // Return and Check Answer button shown when game over
        document.getElementById("returnButton").style.display = "block";
        document.getElementById("checkAnswerButton").style.display = "block";
        }
    }
    document.getElementById("score").textContent = score + "/40"; // Show current score over 40
}
```

Next, we have the checkAnswer function, which check if the selectedAnswer (A, B, C, D) that user has selected match the correctAnswer from the ‘questions’ array, corresponded to the current ‘currentQuestion’. If the answer picked is matched then the ‘score’ variable will be incremented by 1. For each time user select an answer, currentQuestionIndex variable will be incremented by 1. If the currentQuestionIndex is smaller than the length of the ‘questions’ array (rows), then the displayQuestion will keep being called, else, we reflect the customise message (alert) corresponding to the score that user obtained. Also, when the currentQuestionIndex reaches the ‘questions’ array length (40), the Return and Check Answer button will be shown (display = block). Also, the ‘score’ variable will be updated gradually showing the current ‘score’ over (/) the total of 40.
Then next, we have the resetGame function resetting the currentQuestionIndex and the ‘score’ variable, then shuffle the question’s order and display again, upon clicking to Reset button.
We have the showAnswerSheet function called upon clicking to Show Answer button. This function split out the question and correctAnswer component from the ‘questions’ array and set each of them as the answerData constant. Each of these answerData constant will be set under the item answerData to localStorage and send to the answer_sheet.html file to show the web-user the correct answer to that given quiz.
Lastly, we have the suffleArray method to be called each time before displaying the question, that use the Fisher-Yates algorithm scrambling the array order. Firstly, we set each 'question row from the ‘questions’ array with ‘i’ value from 0 to 39. Then, we create a random set of ‘j’

```
// Reset everything upon request
function resetGame() {
    currentQuestionIndex = 0;
    score = 0;
    shuffleArray(questions);
    displayQuestion();
    document.getElementById("score").textContent = score + "/40";
}

function showAnswerSheet() {
    // Open the answer sheet page in a new tab, initialise 'questions' array data as answerData constant
    const answerData = questions.map(question => ({
        question: question.question,
        correctAnswer: question.answers[question.correctAnswer.charCodeAt(0) - 65].split('. ')[1]
    }));

    // Store the answerData in localStorage
    localStorage.setItem("answerData", JSON.stringify(answerData));

    // Open the answer sheet page in a new tab
    window.open("answer_sheet.html", "_blank");
}

// Function to shuffle array (Fisher-Yates algorithm)
// Change the questions' constructive order i to the random order j
function shuffleArray(array) {
    for (let i = array.length - 1; i > 0; i--) {
        const j = Math.floor(Math.random() * (i + 1));
        [array[i], array[j]] = [array[j], array[i]];
    }
}
```

variables, notice that this is also from 0 to 39 but not in order. Therefore, upon switching the ‘questions’ array’s row order I to j, the question displayed will be shuffled and not displaying on a correct structural order.

e. Math.html, Physics.html, Country_Guess.html:

These 3 files showing quizzes to math and physics problem, alongside with a guessing flag game. The Math and Physics questionnaires have a similar structure using Vue.js, while the guessing flag game add in a picture, fetched from local.

We started with the Vue construction showing the form field input for user's answer to the question if 'gameOver' boolean state hasn't been reached. Else if 'gameOver' state reaches, the 'score' will be displayed with reflection message to the user's test score, alongside with the Return and Check Answer button displayed.

```
div class="container">
   <h1>25 Math Questions</h1>
   <div id="app">
      <div v-if="!gameOver"> <!-- If game over state has not reached -->
         <p1>Math problems gradually harder from grade 6 to year 11 level.</p1>
         <div class="question">{{ currentQuestion }}</div>
         <input type="text" v-model="userAnswer" placeholder="Enter your answer..." class="form-control"> <!-- Form to input answer -->
         <button v-on:click="submit()" class="btn btn-primary mt-3">Submit</button>
         <button v-on:click="resetGame()" class="btn btn-primary mt-3">Reset</button>
      </div>
      <div v-else>
         <h2>Game Over!</h2>
         <p>Your score: {{ score }}/25</p> <!-- Show achieved score when game over -->
         <p v-if="score < 5">You need to practice more</p> <!-- Display customized message upon score achived -->
         <p v-else-if="score >= 5 && score < 10">Good try but still need practice.</p>
         <p v-else-if="score >= 10 && score < 15">Good attempt, keep learning.</p>
         <p v-else-if="score >= 15 && score < 20">That's great, almost there.</p>
         <p v-else-if="score >= 20 && score < 25">Fantastic, so close.</p>
         <p v-else-if="score == 25">Awesome, you beat it!</p>
         <button v-on:click="redirect()" class="btn btn-primary mt-3">Return</button>
         <button v-on:click="showAnswerSheet()" class="btn btn-primary mt-3">Check Answer</button>
      </div>
   </div>
/div>
```

Then, we create the Vue app with data of 'questions' array with 'question' and 'correctAnswer' components, the currentQuestionIndex, 'userAnswer' obtained from the form entry, 'score' variable and the 'gameOver' boolean state.

```
const app = Vue.createApp({
    data() {
        return {
            // questions array with the question and correct answer data
            questions: [
                // Grade 6 Math
                { question: "What is the value of 67 x 89?", correctAnswer: "5963" },
                { question: "Calculate: 24 ÷ 6.", correctAnswer: "4" },
                { question: "Find the area of a rectangle with length 10 cm and width 5 cm.", correctAnswer: "50" },
                { question: "What is the square root of 49?", correctAnswer: "7" },
                // Grade 7 Math
                { question: "Simplify: 3x + 2x - 5x.", correctAnswer: "0" },
                { question: "If the ratio of boys to girls in a class is 3:2 and there are 25 students in total, how many boys are there?", correctAnswer: "15"
                { question: "What is the volume of a cube with side length 4 cm?", correctAnswer: "64" },
                { question: "Find the circumference of a circle with radius 5 cm. (Use π = 3.14)", correctAnswer: "31.4" },
                // Grade 8 Math
                { question: "Solve for x: 2(x - 3) = 10.", correctAnswer: "8" },
                { question: "What is the value of (2/3) + (3/4)? [Answer in fraction].", correctAnswer: "8/9" },
                { question: "If a car travels at a speed of 60 km/h for 3 hours, how far does it travel?", correctAnswer: "180" },
                { question: "What is the slope of the line passing through the points (2, 3) and (5, 9)?", correctAnswer: "2" },
                // Grade 9 Math
                { question: "Factorize: x² - 4. [Avoid spaces].", correctAnswer: "(x-2)(x+2)" },
                { question: "Find the value of log₂(8).", correctAnswer: "3" },
                { question: "If sin θ = 0.5, what is the value of θ in degrees?", correctAnswer: "30" },
                { question: "Solve for x: 2x² + 5x - 3 = 0. [Answer as a,b format, smaller number first].", correctAnswer: "-1.5,1"},
                // Grade 10 Math
                { question: "What is the discriminant of the quadratic equation 3x² - 4x + 2 = 0?", correctAnswer: "-32" },
                { question: "Find the value of cos(60°).", correctAnswer: "0.5" },
                { question: "If f(x) = 2x³ - 5x² + 3x - 7, what is f'(x) (the derivative)? [Avoid spaces].", correctAnswer: "6x^2-10x+3"},
                { question: "Solve for x: log₃(x + 1) = 2.", correctAnswer: "8" },
                // Grade 11 Math
                { question: "Solve the equation for x: 2x + 5 = 3x - 1.", correctAnswer: "6" },
                { question: "Calculate the determinant of the matrix: [[3, 2], [4, 5]].", correctAnswer: "7" },
                { question: "Find the derivative of the function f(x) = x^3 - 4x^2 + 2x - 7. [Avoid spaces].", correctAnswer: "3x^2-8x+2" },
                { question: "Solve the trigonometric equation: sin(x) = 0.5 for 0 ≤ x ≤ 2π. [Answer in degree].", correctAnswer: "30" },
                { question: "Find the equation of the tangent line to the curve y = x^2 + 2x - 3 at the point (1, 0). [Avoid spaces].", correctAnswer: "y=3x-3"
            ],
            currentQuestionIndex: 0,
            userAnswer: null,
            score: 0,
            gameOver: false
        };
    }
```

Next, we have the function checkAnswer function (called upon clicking Submit button) which check if the userAnswer match the correctAnswer from the 'questions' array, if it is, increment 'score' and increment currentQuestionIndex per each answer attempt. Once the currentQuestionIndex match the total question number, it return 'gameOver' state as true and clear the entry field in the form (we also clear it per each answer attempt using nextround method). Then, we have the resetGame, refreshing everything and the showAnswerSheet using similar algorithm to the one from Vietnam and Australia Questionnaires.

```
methods: {
    submit() { // Submit answer checking for correct answer and move to next question
        this.checkAnswer();
        this.nextround();
    },
    checkAnswer() { // Validate answer, if correct increment score, then increment current question index to move to next round
        const correctAnswer = this.questions[this.currentQuestionIndex].correctAnswer;
        if (this.userAnswer === correctAnswer) {
            this.score++;
            this.userAnswer = null;
        }
        this.currentQuestionIndex++;
        if (this.currentQuestionIndex === this.questions.length) { // No more question, return game over state
            this.gameOver = true;
            this.userAnswer = null;
        }
    },
    showAnswerSheet() {
        // Initialise data for the answer sheet
        const answerData = this.questions.map(question => ({
            question: question.question,
            correctAnswer: question.correctAnswer
        }));

        // Store the answer data in localStorage
        localStorage.setItem("answerData", JSON.stringify(answerData));

        // Open answer sheet
        window.location.href = "answer_sheet.html";
    },
    nextround() {
        this.userAnswer = null;
    },
    // Reset everything
    resetGame() {
        this.currentQuestionIndex = 0;
        this.userAnswer = null;
        this.score = 0;
        this.gameOver = false;
    },
    // Redirect to Menu
    redirect() {
        window.location.replace("Edu_Test.html");
    }
```

Key implementations on the Country_Guess.html file is that it upload the flag image from a folder "Flag". Then the 'flags' array contain a set of flag image files with constant countryNameMap defining the flag file with the country's name. The user's answer also being set to lower case.

```
<div class="text-center">
    <img :src="'Flag/' + currentFlag" alt="Flag Image" class="flag-image">
</div>
```

```
flags: [
    "kazakhstan.png",
    "nigeria.png",
    "indonesia.svg",
    "vatican.png",
    "turkey.png",
    "egypt.png",
    "suriname.png",
    "thai.jpg",
    "taiwan.svg",
    "iran.png",
    "pakistan.png",
    "ukraine.png",
    "phillipines.png",
    "kenya.webp",
    "malaysia.png",
    "argentina.webp",
    "ethiopia.png",
    "iraq.png",
    "afghanistan.svg",
    "colombia.svg"
],
```

```
const countryNameMap = {
    "kazakhstan.png": "Kazakhstan",
    "nigeria.png": "Nigeria",
    "indonesia.svg": "Indonesia",
    "vatican.png": "Vatican",
    "turkey.png": "Turkey",
    "egypt.png": "Egypt",
    "suriname.png": "Suriname",
    "thai.jpg": "Thailand",
    "taiwan.svg": "Taiwan",
    "iran.png": "Iran",
    "pakistan.png": "Pakistan",
    "ukraine.png": "Ukraine",
    "phillipines.png": "Philippines",
    "kenya.webp": "Kenya",
    "malaysia.png": "Malaysia",
    "argentina.webp": "Argentina",
    "ethiopia.png": "Ethiopia",
    "iraq.png": "Iraq",
    "afghanistan.svg": "Afghanistan",
    "colombia.svg": "Colombia"
};
```

e. Math_Formulas.html, Physics_Concepts.html and Chemistry_Concepts.html:

These files showing references for Math, Physics and Chemistry subject with table displaying the knowledge by each row, alongside with Look Up functions. These 3 files utilise the same construction. These 3 files will fetch data from external JSON file. These designs use the template taken from 3.2P.

First, we construct table with ‘tr’ and ‘th’ component showing columns data of grade, type and concept (description), with each of these variable will be obtained with filtration from filteredConcepts function, and appended to the table by rows with ‘td’ function.

```
    <thead>
      <tr>
        <th>Grade</th>
        <th>Type</th>
        <th>Concept</th>
      </tr>
    </thead>
    <tbody>
      <tr v-for="concept in filteredConcepts">
        <td>{{ concept.grade }}</td>
        <td>{{ concept.type }}</td>
        <td>{{ concept.concept }}</td>
      </tr>
    </tbody>
  </table>
```

Then, we construct a set of tick-boxes options by Grade (6-12) and type (for instance Theory and Formula or Algebra and Calculus for Math). We record the selectedGrades v-model with corresponding ‘value’ and selectedTypes to check for user’s selected box(es).

```
<section class="container">
  <h1>Chemistry Concept Lookup</h1>
  <p1>Chemistry's theories and formulas lookup</p1>
  <div class="row">
    <div class="col-md-4">Grade:</div>
    <div class="col-md-4">Type:</div>
    <div class="col-md-4">Search Concept:</div>
  </div>
  <!-- Filter by Grade -->
  <div class="row">
    <div class="col-md-4">
      <div class="form-check">
        <input class="form-check-input" type="checkbox" v-model="selectedGrades" value="6" id="grade6">
        <label class="form-check-label" for="grade6">Grade 6</label>
      </div>
      <div class="form-check">
        <input class="form-check-input" type="checkbox" v-model="selectedGrades" value="7" id="grade7">
        <label class="form-check-label" for="grade7">Grade 7</label>
      </div>
      <div class="form-check">
        <input class="form-check-input" type="checkbox" v-model="selectedGrades" value="8" id="grade8">
        <label class="form-check-label" for="grade8">Grade 8</label>
      </div>
      <div class="form-check">
        <input class="form-check-input" type="checkbox" v-model="selectedGrades" value="9" id="grade9">
        <label class="form-check-label" for="grade9">Grade 9</label>
      </div>
      <div class="form-check">
        <input class="form-check-input" type="checkbox" v-model="selectedGrades" value="10" id="grade10">
        <label class="form-check-label" for="grade10">Grade 10</label>
      </div>
      <div class="form-check">
        <input class="form-check-input" type="checkbox" v-model="selectedGrades" value="11" id="grade11">
        <label class="form-check-label" for="grade11">Grade 11</label>
      </div>
      <div class="form-check">
        <input class="form-check-input" type="checkbox" v-model="selectedGrades" value="12" id="grade12">
        <label class="form-check-label" for="grade12">Grade 12</label>
      </div>
    </div>
    <!-- Filter by Type -->
    <div class="col-md-4">
      <div class="form-check">
        <input class="form-check-input" type="checkbox" v-model="selectedTypes" value="Theory" id="theory">
        <label class="form-check-label" for="theory">Theory</label>
```

We also allow user to look up explicitly the knowledge base when fill in the input form.
Then, we create a Vue app with an array of concept (or formula for Math), selectedGrades, selectedTypes and searchConcept 'value' (or searchFormula from Math).
We then call the filteredConcepts function, which filter the Grade and Type from tick-boxes and the form entry, which is set to lower case to be filtrated. We also trim off the empty string to allow better filtration accuracy. All filtrated requirement will set the 'filtrated' constant by the corresponding. The table displayed therefore will be filtrated to display knowledge base accordingly to the 'filtrated' constant.
Then, we run async fetchConcepts to fetch the array data from the corresponding JSON file, which data will be set to the 'concepts' (or 'formulas') constant.

```
const app = Vue.createApp({
    data() {
        return {
            concepts: [], // Array to store all concepts
            selectedGrades: [], // Array to store selected grades
            selectedTypes: [], // Array to store selected types
            searchConcept: "", // String to store search input
        };
    },
    computed: {
        // Computed property to filter concepts based on selected grades, types, and search input
        filteredConcepts() {
            let filtered = this.concepts;

            // Filter by selected grades
            if (this.selectedGrades.length > 0) {
                filtered = filtered.filter(concept => this.selectedGrades.includes(concept.grade));
            }
            // Filter by selected types
            if (this.selectedTypes.length > 0) {
                filtered = filtered.filter(concept => this.selectedTypes.includes(concept.type));
            }
            // Filter by search concept
            if (this.searchConcept.trim() !== "") {
                filtered = filtered.filter(concept => concept.concept.toLowerCase().includes(this.searchConcept.toLowerCase()));
            }
            return filtered;
        }
    },
    mounted() {
        // Fetch concepts data from the server
        this.fetchConcepts();
    },
    methods: {
        async fetchConcepts() {
            // Fetch data from the python server or from the folder
            const response = await fetch("python HTTP server/chemistry_concept.json");
            // Parse the response JSON
            const data = await response.json();
            // Assign the fetched concepts to the data property
            this.concepts = data;
        }
    }
```

f. flag_answer.html:

This webpage shows the table of the countries' flag and their name. There are also Look Up function to filtrate the flag and country displayed by either Continent or Region tick-boxes. Similarly to the Math, Physics and Chemistry Classrooms previously introduced, we implement the set of tick-boxes to filtrate the data array from upon type filtration from user's request. Major difference is that we will also set the array of 'countries' with 'name', 'flag' (image source), 'continent' and 'region' values, hard-coded to the file.

Besides, this webpage also have the pagination function, which set each page to only show 5 rows (pageSize = 5) and create a currentPage variable to keep track on the current page that the user is in. The template for pagination function is taken from task 8.1P.

```
<nav aria-label="Pagination"> <!-- Pagination function -->
    <ul class="pagination justify-content-center">
        <li class="page-item" :class="{ 'disabled': currentPage === 1 }">
            <button class="page-link" @click="prevPage">Prev</button>
        </li>
        <li class="page-item" v-for="page in totalPages" :key="page" :class="{ 'active': currentPage === page }">
            <button class="page-link" @click="changePage(page)">{{ page }}</button>
        </li>
        <li class="page-item" :class="{ 'disabled': currentPage === totalPages }">
            <button class="page-link" @click="nextPage">Next</button>
```

Computed programs allows setting totalPages by ceiling the total number of filteredCountries (length) divided by the pageSize (5), then paginatedCountries function split and append filteredCountries by look up requirement to the corresponding pagination format. The function filteredCountries utilise same approach as filteredConcepts from previous design of Physics and Chemistry Classrooms.

Lastly, we have the methods programs handling the current page (as this), method to increment and decrement the currentPage when clicked to Next and Prev pagination button, which also ensure to keep track on the totalPages variable so the currentPage will not exceed the page limitation.

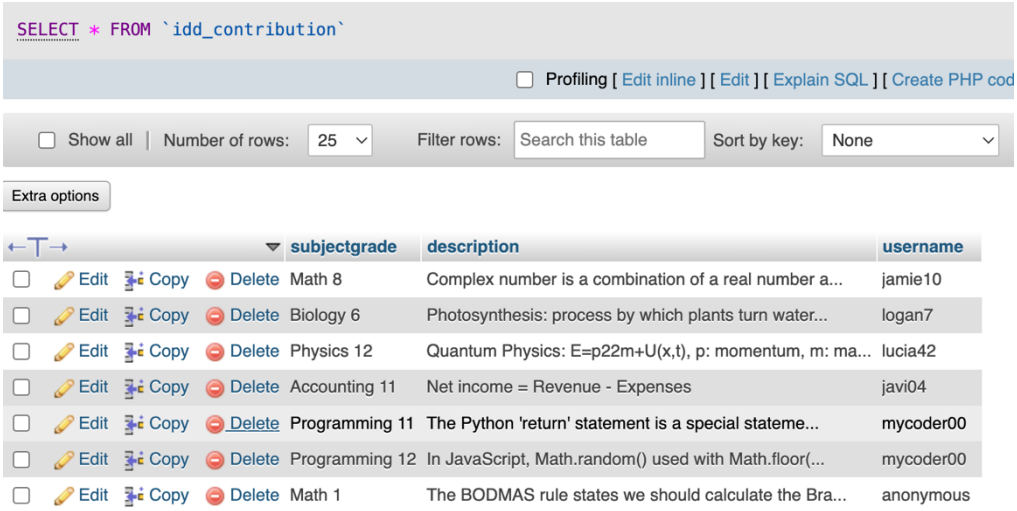
```
        pageSize: 5, // Each page 5 rows
        currentPage: 1
    }
},
computed: {
    totalPages() { // Set total page
        return Math.ceil(this.filteredCountries.length / this.pageSize);
    },
    paginatedCountries() {
        const start = (this.currentPage - 1) * this.pageSize;
        const end = start + this.pageSize;
        return this.filteredCountries.slice(start, end);
    },
    filteredCountries() { // Filter counries by region or continent
        if (this.selectRegion.length > 0) {
            return this.countries.filter(country => this.selectRegion.includes(country.region));
        }
        if (this.selectContinent.length > 0) {
            return this.countries.filter(country => this.selectContinent.includes(country.continent));
        }
        return this.countries;
    }
},
methods: { // Changin pagignation
    changePage(page) {
        this.currentPage = page;
    },
    prevPage() {
        if (this.currentPage > 1) {
            this.currentPage--;
        }
    },
    nextPage() {
        if (this.currentPage < this.totalPages) {
            this.currentPage++;
        }
    }
```

g. Knowledge Contribution section:

This section effectively utilise the template from task 9.2C to create a well-designed SPA program that works with Vue.js construction (vuetify, vue-paginate and vue-router).
All functional components (e.g. files) locate inside the Update folder.
Started with the index.html file, where the Menu page will redirect web-users to, this file will run script and sources file from other elements including the CSS (all styling), JS (vue-app components, router, paginate, vuetify etc) and resources (api php file reading parsing SQL table from MariaDB). The file is taken from week 9 learning material.
Next, major changes coming to the app-login page, where the rules has been changed to adapt with the webpage credential information format. Also, instead of parsing credential information from MariaDB as of week 9 material, this one parse the 'users' item directly from localStorage, creating match boolean constant and check with existing data (similar to Credential.html), then it will emit 'authenticated ' event to true and redirect to the dashboard app, with error message also printed out when credential not satisfied.

```
    usernameRules: [
      v => !!v || 'Username is required',
      v => (v && v.length >= 3) || 'Username must be more than 3 characters.',
    ],
    // defining username rules for validation
    passwordRules: [
      v => !!v || 'Password is required',
      v => (v && v.length >= 8) || 'Password must be at least 8 characters with a special character.',
    ],
  }
},
methods: {
  validation() {
    // Get existing user data from localStorage
    let users = JSON.parse(localStorage.getItem('users')) || [];
    // Check if user credentials match any stored user data
    const match = users.some(user => user.username === this.input.username && user.password === this.input.password);
    if (match) {
      this.$emit("authenticated", true); // Emit authentication event
      this.$router.replace({ name: "dashboard" }); // Redirect to dashboard
    } else {
      this.msg = "Username or password incorrect.";
    }
  },
  reset() {
    this.$refs.form.reset();
```

SQL database of the contribution is added to MariaDB (username s103844421, password 181203) with table idd_contribution. The table contains a 3 column ids subjectgrade, description and username. In resources, we have apis.php file to connect and parse this SQL table (taken from material). File apis.php also handle action command cases GET, PUT, POST and DELETE to perform accurately upon user's request too.



I kept the app-dashboard the same with the week 9 material.
We have the View feature from app-read contain a table with column ids of subjectgrade, description and username with corresponding headers from idd_contribution database, and pagination function (taken from the material). Then, the Insert feature from app-post handling a form input for new knowledge base's Subject Grade (i.e. Math 10), Description and Username (set as user's choice), the Update feature from app-update with the same form template, and the Delete feature with only Subject Grade and Username entries.

```
<v-card-text>
 <div class= "table-responsive">
          <table class="table table-striped table-hover">
              <thead>
                  <tr>
                      <th scope="col" id="subjectgrade">Subject Grade</th>
                      <th scope="col" id="description">Description</th>
                      <th scope="col" id="username">Username</th>
                  </tr>
              </thead>
              <tbody>
                  <tr v-for="formula in getItems"  >
                      <td scope="row" headers="subjectgrade">{{formula.subjectgrade}}</td>
                      <td scope="row" headers="description">{{formula.description}}</td>
                      <td scope="row" headers="username">{{formula.username}}</td>
                  </tr>
              </tbody>
          </table>
       </div>
 </v-card-text>
 </v-card>
 </v-col>

 <v-col cols="12" md="12">
 <paginate
:page-count=getPageCount
:page-range="4"
:margin-pages="1"
:click-handler="clickCallback"
:prev-text=" 'Prev' "
:next-text="'Next'"
:container-class="'pagination'"
:page-class="'page-item'"
:active-class="'currentPage'">
 </paginate>
 </v-col>
 </v-row>
`,
```

```
<v-row>
  <v-col cols="12" md="6 " >

  <v-card
    class="mx-auto"
    max-width="90%"
    >

    <v-card-text>
  <v-form>
      <v-text-field label="Subject Grade" v-model="subjectgrade1" /></v-text-field>
      <v-text-field label="Description" v-model="description1" /></v-text-field>
      <v-text-field label="Username" v-model="username1" /></v-text-field>
      <p>You can set any username or post anonymously.<p>
      <v-btn
      depressed
      v-on:click="postData(subjectgrade1, description1, username1)"
      color="primary">
      Add
    </v-btn>

  </v-form>
  </v-card-text>

  </v-card>

  </v-col >
    <!-- Output -->
      <v-col cols="12"
          md="6">
  <v-card
    class="mx-auto"
    max-width="90%"
    >
    <v-card-text>
     <p>Output Message : {{ msg }}</p>
     <p>Status Subject Grade: {{statusVal}}</p>
     <p>Status: {{statusText}}</p>
     <p>Response Headers: {{headers}}</p>
  </v-card-text>
```

```
template: `
<!-- Updating mySQL Table With Name as Key -->
<v-row>
  <v-col cols="12" md="6 ">

    <v-card class="mx-auto" max-width="90%">
      <v-card-text>
        <!-- Input -->
        <v-form name="myForm2" class="form-horizontal">
          <v-text-field label="Subject Grade" v-model="subjectgrade2" />
          </v-text-field>
          <v-text-field label="Description" v-model="description2" />
          </v-text-field>
          <v-text-field label="Username" v-model="username2" />
          </v-text-field>


          <v-btn depressed v-on:click="putData(subjectgrade2, description2, username2)" color="primary">
            Update
          </v-btn>
        </v-form>
      </v-card-text>
    </v-card>

  </v-col>
  <!-- Output -->
  <v-col cols="12" md="6">
    <v-card>
      <v-card-text>
        <p>Output Message : {{msg}}</p>
        <p>Status Subject Grade: {{statusVal}}</p>
        <p>Status: {{statusText}}</p>
        <p>Response Headers: {{headers}}</p>
      </v-card-text>
    </v-card>
  </v-col>
```

```
<v-row>
    <v-col cols="12" md="6">
        <v-card class="mx-auto" max-width="90%">
            <v-card-text>
                <v-form>
                    <v-text-field label="Subject Grade" v-model="subjectgrade3" />
                    <v-text-field label="Username" v-model="username3" />
                    <v-btn depressed @click="delData(subjectgrade3, username3)" color="primary">Delete</v-btn>
                </v-form>
            </v-card-text>
        </v-card>
    </v-col>
    <!-- Output -->
    <v-col cols="12" md="6">
        <v-card class="mx-auto" max-width="90%">
            <v-card-text>
                <!-- Output -->
                <p>Output Message: {{ msg }}</p>
                <p>Status Knowledge: {{ statusVal }}</p>
                <p>Status: {{ statusText }}</p>
                <p>Response Headers: {{ headers }}</p>
            </v-card-text>
        </v-card>
    </v-col>
</v-row>
```

Figures. app-read, app-post, app-update and app-delete form input element.

In app-post, upon obtaining the entry inputs from the form, they will construct the JSON format body and send these data to the SQL table from apis.php via POST command. Response message will also print out to address if it is successful or exist errors.
Next, app-update will use the subjectgrade and username from the form to identify the SQL item's URL that would be modified, then construct the new body with the new description from user's request and set to MariaDB using PUT command. Similarly, app-delete will also set the requesting item from SQL table by the subjectgrade, and username variable provided to perform DELETE action, permanently remove the targeted item.

```
    subjectgrade1: '',
    msg: '',
    description1: '',
    username1: '',
    statusVal: '',
    statusText: '',
    headers: '',
    savingSuccessful: false
  }
},
methods: {
  postData: function(subjectgrade, description, username) {
    //define url for api
    var postSQLApiURL = 'resources/apis.php/'

    var self = this;
    // POST request using fetch with error handling
    const requestOptions = {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json'
      },
      body: JSON.stringify({
        subjectgrade: subjectgrade,
        description: description,
        username: username
      })
    };


  fetch(postSQLApiURL, requestOptions)
  .then( response =>{
    //turning the response into the usable data
    return response.json( );
  })
  .then( data =>{
    //This is the data you wanted to get from url
     self.msg = "Data Inserted Successfully."  ;
  })
  .catch(error => {
     self.msg = 'There was an error!' + error;
  });
```

```
putData: function(subjectgrade, description, username) {

  var putSQLApiURL = 'resources/apis.php/subjectgrade/' + subjectgrade + '/username/' + username;


  var self = this;
  // POST request using fetch with error handling
  const requestOptions = {
    method: 'PUT',
    headers: {
      'Content-Type': 'application/json'
    },
    body: JSON.stringify({
      subjectgrade: subjectgrade,
      description: description,
      username: username
    })
  };

fetch(putSQLApiURL, requestOptions)
.then( response =>{
  //turning the response into the usable data
  return response.json( );
})
.then( data =>{
  //This is the data you wanted to get from url
  self.msg="successful";
})
.catch(error => {
  self.err=error
});
```

Figures. post-app and update-app data handling functions.

h. Social_Feedback.html and Social_Forum.html:

This page allow web-users to leave feedback as test messages, which are to be saved to localStorage.
It started by fetching the storedFeedback (public item saving all feedbackNode to localStorage) item, or return as empty in case no feedback has been submitted. Then, we have displayFeedback constant, which are the feedbacks that are going to be displayed. We have the feedback constant to be formatted as @userid: $userfeedback, which are the userid fetched from localStorage of this user combined with the feedback user has just posted. Then, set feedbackNode as the text content of the constant feedback after formatted.
So once the page is loaded or user just opened the page, it will parse the storedFeedback item out and retrieved all feedbackNode from the storedFeedback, then display all of those feedbackNode,
Next, we have the submitFeedback function, where the userfeedback has just been posted is formatted properly, set as feedbackNode and display, then parsing the storedFeedback out, push this 'feedback' constant (after formatted) into the storedFeedback item and setItem to save it. Then, it automatically set the form input to empty after submission.

Additionally, we also have the resetFeedback function to set the form input to empty (in case user don't want to delete by each letter), and the clearFeedback in case user don't want to see those feedback anymore (remove item storedFeedback then set the display to empty).

```
// Retrieve feedback from localStorage when page loads
document.addEventListener('DOMContentLoaded', function() {
    const storedFeedback = JSON.parse(localStorage.getItem('storedFeedback')) || []; // Get stored feedback from localStorage (or empty string if there are no sa
    const displayFeedback = document.getElementById('displayFeedback'); // Feedback to be displayed

    storedFeedback.forEach(feedback => { // The stored feedback contain a set of children feedback nodes that each of them will be called
        const feedbackNode = document.createElement('div');
        feedbackNode.textContent = feedback;
        displayFeedback.appendChild(feedbackNode);
    });
});

// Submit feedback
function submitFeedback() {
    const userFeedback = document.getElementById('userFeedback').value;
    const displayFeedback = document.getElementById('displayFeedback'); // The feedback just being pushed are set to display

    // Construct format for a feedback
    const feedback = `@${userid}: ${userFeedback}`;

    const feedbackNode = document.createElement('div');
    feedbackNode.textContent = feedback; // Append feedback to a node
    displayFeedback.appendChild(feedbackNode); // The feedback displayed will contain a set of children node of the feedback node

    // Parse storedFeedback item from localStorage, push this feedback to 1 storedFeedback and then populate that storedFeedback to the storedFeedback item on lo
    const storedFeedback = JSON.parse(localStorage.getItem('storedFeedback')) || [];
    storedFeedback.push(feedback);
    localStorage.setItem('storedFeedback', JSON.stringify(storedFeedback));

    // Clear text input field after submission
    document.getElementById('userFeedback').value = '';
}

// Reset feedback entry (not submitted)
function refreshFeedback() {
    document.getElementById('userFeedback').value = ''
}

// Clear all feedbacks in the chatbox
function clearFeedback() {
    localStorage.removeItem('storedFeedback'); // Remove stored feedback from localStorage
    document.getElementById('displayFeedback').innerHTML = '';
}
```

Similarly, we have the Forum page, which user can post a caption of text and image content. This page has quite similar construction as the Feedback page, just that it will also have an image attached to the stored data item (storedCaption) set to the localStorage.
Starting with the form to enable image file upload, then, upon submission, we have the method to check if an image has been uploaded, if true, we upload that image using FileReader method.

```
<!-- Image upload input, reference from https://www.w3schools.com/howto/tryit.asp?filename=tryhow_html_file_upload_button -->
<input type="file" id="imageUpload" accept="image/*"><br>
```

```
// Check if an image is selected
if (fileInput.files.length > 0) {
    const file = fileInput.files[0];
    const reader = new FileReader();

    // Load image from local device
    reader.onload = function(event) {
        const imageSrc = event.target.result;

        // Construct format
        const caption = `@${userid}: ${userCaption}\n${imageSrc}`;
```

Alike Feedback page, this one will also fetch the storedCaption data to display all textNode (text caption like the 'feedback' constant with userid and user's input component), and the imageNode (the image uploaded).
Then, upon submission, it will set the 'caption' variable with a textNode (userid: userCaption) and an imageNode (scaled to 50px by 50px) then push to the storedCaption item and display. Submission also automatically clear the text and image form.
Refresh and Clear function are also operated similarly to the Feedback page.

3. Reference

Data saving by localStorage and Global Object:
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON/stringify
Star rating:
https://www.w3schools.com/howto/tryit.asp?filename=tryhow_css_star_rating
Modal box:
https://www.w3schools.com/howto/howto_css_modals.asp
Array filtration by index:
https://www.geeksforgeeks.org/what-is-the-most-efficient-way-to-create-a-zero-filled-array-in-javascript/
Image Upload:
https://www.w3schools.com/howto/tryit.asp?filename=tryhow_html_file_upload_button