Principle of Software Engineering

[CEN 4010]

**GymBro**

**Group: Team 9**

Sharon Nir (Team Lead)

**Team Members:**

Sharon Nir – Scrum Master/ Front-End Developer

Romeo Francois – Back-End Developer

Luis Timana - Front-End Developer

Christopher Anastasis - Front-End Developer

Juan Reyes Pinzon - Front-End Developer

Milestone 1

Date: April 10, 2023

**Product Summary**

Welcome to GymBro, a workout companion for those who are currently exercising or seek to do so without the need for subscription fees.

In this application, you will find a Gym Locator. Just enter your location, and the map will show you nearby gyms. Our goal in adding this feature is to make it easier for customers to search for gyms in the same app that allows them to plan their fitness.

Our application also provides a BMI calculator. This allows customers to insert their weight and height in order to calculate their BMI score. This is helpful for figuring out if the customer is overweight, underweight, or has a normal weight. With that knowledge, the customer is able to adjust what diet they eat, and the intensity of their daily exercising.

In the center of the application, we have created a workout planner and workout goals. Customers will be able to create two lists: one for their workout goals, and the other for planning out what they are going to do, whether workout plans or diet plans. This allows the user to motivate themselves by creating goals and planning out how they are going to do it.

Our site also provides links to recipes that the customer can make themselves in order to eat healthier. The goal in adding this feature is to encourage the customer to avoid junk food and put a links right in front of them in order to incentivize clicking them and making a healthier meal.

Below the recipes, we have provided three example diet plans for each type of weight category. For those who are underweight, there is a link that provides a more calory heavy diet. For those who are normal weight, there is a link for a balanced diet in order to maintain that weight. And finally, for those who are overweight, there is a link that provides a diet plan that is low on calories. Although, just like with everything, the diet plan won’t be enough without exercising along with it.

Our main goal was and still is, the creation of a workout companion that is accessible to those who may not be able to afford a monthly/yearly subscription fee. We have included all the basics that a fitness app needs.

Website Link: <http://lamp.cse.fau.edu/~cen4010-sp23-g09/Homepage.html>

**Usability test**

**Test objectives:** The usability test is going to be focusing on using the BMI calculator to find their current BMI. The BMI calculator takes the users height and weight to calculate their BMI. This task should take less than a minute if the users already have this information on hand. The user needs to have a very basic understanding of computers to use the calculator. All they need to know is how to operate a mouse. Users include people at anypoint in their fitness journey. This includes people just getting into fitness and people who are well into their fitness journey. The steps to operate the calculator include moving two sliders to reflect your current height and weight. The only difficulty with this current system is that if the user has an unsteady hand or if they have difficulties operating a mouse then they won’t be able to move the slider to their correct measurements.

**Test Plan:** In order to start the test the user must navigate to the demo page where they can find the BMI calculator. The BMI calculator sliders start in their default position. The user is now tasked with moving the sliders to their correct measurements. After this is done the calculator will generate their BMI and tell the user their current fitness level. After this the task is complete. The user can revisit the calculator at a later date to see if any changes have been made to their BMI.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Strongly Disagree | Disagree | Neutral | Agree | Strongly Agree |
| Were you able to accurately calculate your BMI and get an idea of your current fitness level? |  |  |  |  |  |
| Was it easy for you to calculate your BMI? |  |  |  |  |  |
| Would you recommend the BMI calculator to friends or family? |  |  |  |  |  |

**QA test plan**

**Test objectives:** The goal of this test is to determine whether or not the BMI calculator functions as intended on different browsers. Meaning the user must be able to navigate to the demo page and be able to accurately manipulate the sliders to get their correct BMI utilizing browsers. This test is to be conducted on two browsers, google chrome and Microsoft edge. The test will be considered a success if the BMI calculator works on both browsers and the users calculate their BMI. It will be considered a failure otherwise.

**Hardware and software:** The hardware setup for this test will be my PC. The software setup for the test will include the use of brackets, google chrome, and Microsoft edge.

**Feature to be tested:** The BMI calculator will be tested on functionality and usability.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | BMI calculator functions on google chrome | BMI calculator functions on Microsoft edge | BMI calculator slider’s function | BMI calculator calculates BMI and gives fitness level |
| Test #1 | PASS | PASS | PASS | PASS |
| Test #2 | PASS | PASS | PASS | FAIL |
| Test #3 | PASS | PASS | PASS | PASS |

**Test summary:** Overall the test was largely successful. The webpage was able to be loaded up into both google chrome and Microsoft edge without issue. The sliders used to calculate the BMI functioned as intended for each test. 2 participants were able to calculate their BMI correctly without issue in a timely manner. 1 participant struggled to move the sliders to their correct positions because they are not too familiar with operating a touchpad.

**Code Review**

The entire project was reviewed, but only the "login" code block is analyzed here.

Coding Style:

This code is written in JavaScript and follows a functional programming style with the use of functions to encapsulate logic and separate concerns. It also makes use of event listeners to handle user interactions with the DOM. The code includes comments to provide clarity on what each block of code does. The code is written using modern JavaScript syntax and makes use of arrow functions, template literals, and ES6 modules. Additionally, it makes use of the localStorage API to store user data.

What's good:

1. The code is well-structured and follows best practices for JavaScript coding, with clear separation of concerns and reusable functions.
2. The code is modular and makes use of ES6 modules to keep the code organized and maintainable.
3. The code uses modern JavaScript syntax, including arrow functions and template literals, which makes it easier to read and understand.
4. The code makes use of the localStorage API to store user data, which is a good choice for small-scale applications that do not require a server-side database.
5. The code includes comments to provide clarity on what each block of code does, which makes it easier for other developers to understand and maintain the code.
6. The code makes use of event listeners to handle user interactions with the DOM, which is a good practice for building interactive web applications.

What's bad:

1. The code does not handle errors very well. For example, if the localStorage API fails to retrieve user data, the code simply returns -1, which is not very informative for the user or other developers who are trying to debug the code.
2. The code includes some duplicate logic, such as the code to validate passwords, which is used in multiple places. This could be refactored into a reusable function to simplify the code.
3. The code could benefit from more robust error handling and validation, such as checking for valid email addresses or enforcing password strength requirements.
4. The code does not make use of any front-end frameworks or libraries, which could make it easier to build and maintain complex user interfaces.

Peer Review:

The code appears to be functional and accomplishes the desired tasks of creating a sign-up and login feature. It is organized and uses clear variable names, making it easy to read and understand. The code also includes comments to explain the purpose of each function and block of code, which is helpful.

However, there are a few potential issues with the code. Firstly, the password validation function validatePassword() only alerts the user that the passwords do not match, without any further instructions or prompts. This could cause confusion or frustration for the user. Secondly, the addUser() function does not check if the email entered by the user is already associated with an existing account, potentially allowing for duplicate accounts to be created.

Overall, the code is functional and well-organized, but could benefit from additional error handling and user prompts.

Commented Code:

// Select the necessary HTML elements

const forms = document.querySelector('.forms');

const links = document.querySelectorAll('.link');

const form = document.getElementById('form');

const loginForm = document.getElementById('login-form');

const firstName = document.getElementById('name');

const lastName = document.getElementById('lastName');

const email = document.getElementById('email');

const password = document.querySelector('.password');

const passwordTwo = document.querySelector('.confirmPass');

const errors = document.getElementById('errors');

// Toggle the visibility of the login/signup forms when the user clicks the relevant link

links.forEach((link) => {

link.addEventListener('click', (e) => {

e.preventDefault();

forms.classList.toggle('show-login');

});

});

// Handle form submission when user signs up

form.addEventListener('submit', (event) => {

// Check that all required fields are filled in and the passwords match

const inputPass = checkInputs(firstName, lastName, email, password, errors);

// Check if account already exists in local storage

const foundUser = findUser(email.value);

// If all validation passes and the user is not already registered, create their account and notify them

if (inputPass == 1 && foundUser != 1){

addUser();

alert("Your account was created!");

event.preventDefault();

}

// If the user is already registered, notify them

else if (foundUser == 1){

alert("Already a Subscriber")

}

// Prevent form submission if there are errors

else{

event.preventDefault();

}

});

// Handle form submission when user logs in

loginForm.addEventListener('submit', (event) => {

event.preventDefault();

// Get the user's email and password from the login form

const email = document.getElementById('loginEmail');

const password = document.getElementById('loginPassword');

// Check if the user's email and password match a registered account

const foundUser = grantAccess(email.value, password.value);

// If a matching account is found, notify the user

if (foundUser == 1) {

alert('Login successful');

} else {

// If the user's email and password do not match a registered account, notify the user

alert('Email or Password is invalid');

}

});

// This function adds a new user to the local storage. It retrieves the user data from the HTML form

// and stores it as an object in the "accounts" array in local storage.

function addUser() {

const user = {

userEmail: email.value,

userName: firstName.value,

userLastName: lastName.value,

userPassword: passwordTwo.value

};

const accountsJSON = localStorage.getItem('accounts');

let accounts = [];

// If there are existing accounts in the local storage, they are retrieved and parsed into an array

if (accountsJSON) {

accounts = JSON.parse(accountsJSON);

}

// The new user is pushed into the "accounts" array and then stored back in the local storage

accounts.push(user);

localStorage.setItem('accounts', JSON.stringify(accounts));

}

// This function searches for a user in the local storage, given their email address.

// It returns 1 if the user exists and -1 if the user doesn't exist.

function findUser(email) {

const accountsJSON = localStorage.getItem('accounts');

// If there are no existing accounts in the local storage, -1 is returned to indicate that the user does not exist

if (!accountsJSON) {

return -1;

}

const accounts = JSON.parse(accountsJSON);

const user = accounts.find(account => account.userEmail === email);

// If the user is found in the local storage, 1 is returned to indicate that the user exists

if (user) {

return 1;

} else {

return -1;

}

}

// This function checks if the input fields in the HTML form have been filled out correctly.

// It returns 1 if the inputs are valid and -1 if there are errors.

function checkInputs(firstName, lastName, email, password, errors) {

let message = [];

const passwordMatch = validatePassword(password, passwordTwo);

// The function checks if the first name, last name, and email fields have been filled out.

// If not, an error message is added to the "message" array

if (firstName.value == '' || firstName.value == null) {

message.push('Name is required');

}

if (lastName.value === '' || lastName.value == null) {

message.push('Last Name is required');

}

if (email.value === '' || email.value == null) {

message.push('Email is required');

}

// If there are errors in the input fields, an error message is displayed in the HTML page and the function returns -1

if (message.length > 0 || password.value == '') {

message.push('Password is required')

errors.innerText = message.join(', ');

errors.style.color = 'white';

errors.style.border = '2px red solid';

errors.style.borderRadius = '5px';

errors.style.fontSize = 'larger';

errors.style.padding = '5px';

event.preventDefault();

return -1;

}

// If there are no errors in the input fields, the function returns 1

if (message.length == 0 && passwordMatch == 1) {

event.preventDefault();

return 1;

}

}

// This function validates if the password fields in the HTML form match.

// It returns 1 if the passwords match and -1 if they do not match.

function validatePassword(password, passwordTwo) {

if (password.value === passwordTwo.value) {

return 1;

} else {

alert("Passwords don't match!!");

return -1;

}

}

**Self-check on best practices for security**

Some of the major assets we are protecting in this project are predominantly user related. We are protecting user info including their email, password, personal information, and location. Not to mention we are protecting our webpage’s design and everyone on the development team’s information as well. As of right now, we do not have an encrypted password in our database.

We plan on accomplishing this within the next Milestone. In regard to the beta launch, we tested input data for “Workout Goals”, “Planner”, “BMI” calculation, and “Sign Up/Log In” data. Everything works as expected. We could place restrictions in the near future if a user accidentally places incorrect data. For example, if the user inputs a number, and only a number, in the “Workout Goals” section and tries to submit. Here we can raise an error prompting the user to input correctly.

**List of Non-Functional Requirements**

1. **Performance**: ON TRACK  
   GymBro’s landing page will have 5 seconds or less response time under a max load of 5,000 users. Throughout the entire site, users will not have to worry about poor loading times. As the platform grows, we plan on expanding the max load while maintaining immediate response times.
2. **Usability**: ISSUE- As of right now, a personalized page for each user is not our main priority.  
   As a group we are focusing more on our page to be user friendly and not confusing.  
   Creating an account and receiving a personalized plan takes less than 5 minutes. After this is completed, users will find videos, diet plans, and information all immediately in their personalized page. Anytime a user has made progress in their health goal, the app will check off the workout/meal for users to keep note of what they have and haven’t done. The site will be gentle to the users’ eyes and not filled with unnecessary information.
3. **Accessibility**: DONE  
   Users from all around the world will be able to access their personalized page and track their progress through their account which can be logged in if they have an internet connection.
4. **Security requirements**: DONE  
   Our app will ask users to create an account with their own private email and a personalized password. Due to the app being completely free, there will be no additional security required such as payment processing gateways.
5. **Storage**: ISSUE - We have not tackled this objective and are putting all our focus on usability instead.  
   Any storage will be done through the users account and will not require users to create more space on their devices.