Financial Planner Website

Jasmin Gilmore
Term 2 - Software Engineering Stage 6
GitHub Repository: https://github.com/jasg99/Financial-Planner
Site URL: https://jasg99.github.io/Financial-Planner/

Financial Planner Website **1**
1. Identifying and Defining 1
2. Research and Planning 2
3. System Design 4
4. Producing and Implementing 7
5. Testing and Evaluation 11
6. (Client) Feedback and Reflection 14
7. Appendices 15

---

1. Identifying and Defining

1.1 Problem Statement

Many people struggle with financial planning, including planning to save while allocating themselves spending money, and working towards goals consistently. The program allows for people to visualise and recognise their goals.

1.2 Project Purpose and Boundaries

The financial planner allows users to calculate the number of days until their savings goal is reached, and a structured way to view their finances. I intend to include the ability to input expenses, income and goals, a calculation for the number of days until their goal is reached, a signup and a login feature, and secure coding to prevent XSS and malicious injection.

1.3 Stakeholder Requirements
The stakeholders are the intended audience, primarily young people, aged between 14 and 25. From this software they require a feasible and user-friendly interface that allows them to gather saving calculations quickly while also encouraging them to continue saving.

1.4 Functional Requirements
The requirements of the program to allow functionality are: the ability to input expenses, income, a goal, a calculation function for days until the goal is reached, the signup and login features, and coding securely to prevent XSS and malicious injection.

1.5 Non-functional Requirements
The non-functional requirements to ensure feasibility and user satisfaction are: an aesthetic and simple user interface.

1.6 Constraints
Due to this project's status as a school project, time constraints and limited knowledge base implicate many features. Increased security including encryption of user information should be planned for, however due to the short 10 week timeline, this is not possible. The project is limited to free resources, therefore APIs and advanced programs to improve the functionality are not viable.

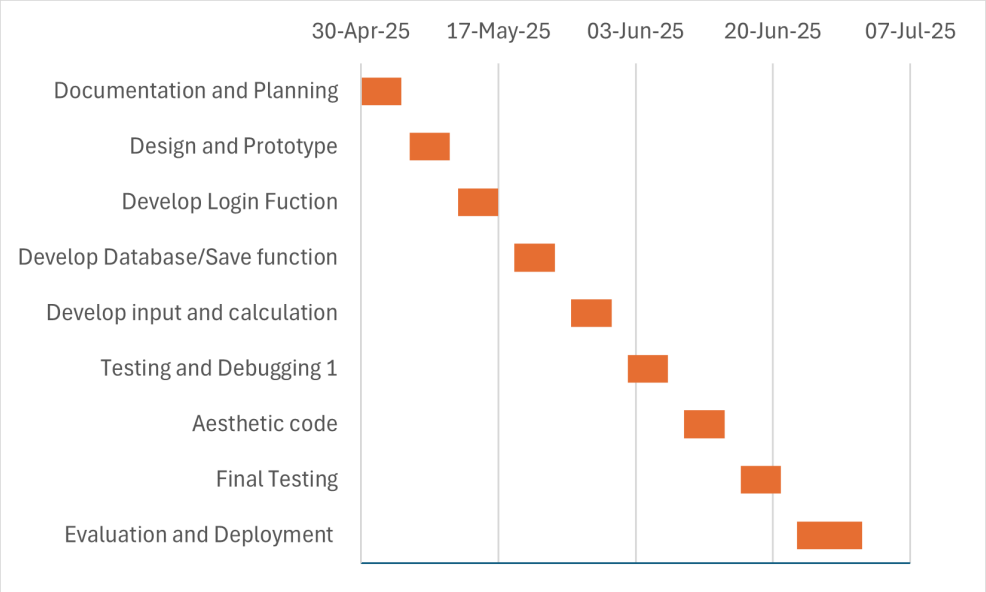
---

2. Research and Planning

2.1 Development Methodology
The Agile SDLC will be employed within the project to ensure rapid delivery of functional, secure software by breaking down the project into parts. Requirements gathering and Analysis will be implemented during the design and planning phase to ensure the target audience and the project guidelines are satisfied. Design will also be incorporated into this stage to ensure the project is achievable. During the development stage incremental coding and feature integration will be implemented to ensure the code is rapidly deployed. Throughout the project frequent testing will occur, to ensure that it is functional, secure and issues are resolved quickly.

2.2 Tools and Technology
To construct the program, a web-based stack incorporating Javascript, HTML, and CSS will be employed. The development of code, testing and debugging, will be achieved through utilizing Visual Studio Code as an IDE as it is a convenient, commonly used platform. Additionally, Github will be utilized to save the project and its version history, due to its feasibility and common use among developers.

2.3 Gantt Chart / Timeline



2.4 Communication Plan

I will conduct a small survey on a group between the ages of 15 and 21, to determine the usefulness of this project. The survey will consist of 5 questions. The questionnaire will include:

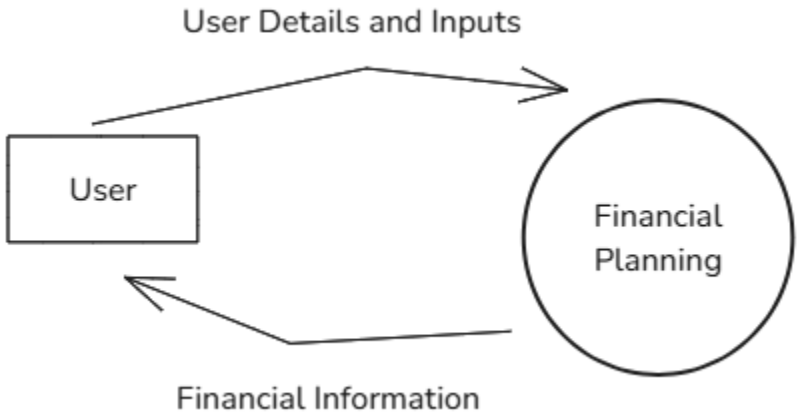
- Is the page aesthetically pleasing to you?
- Is the signup page easy to navigate?
- Is the login page easy to navigate?
- Is the calculation function easy to understand and navigate?
- Is this site useful to you?

The individual will answer on a scale of 1-5, 1 being very poor, 5 being very satisfactory

---

3. System Design

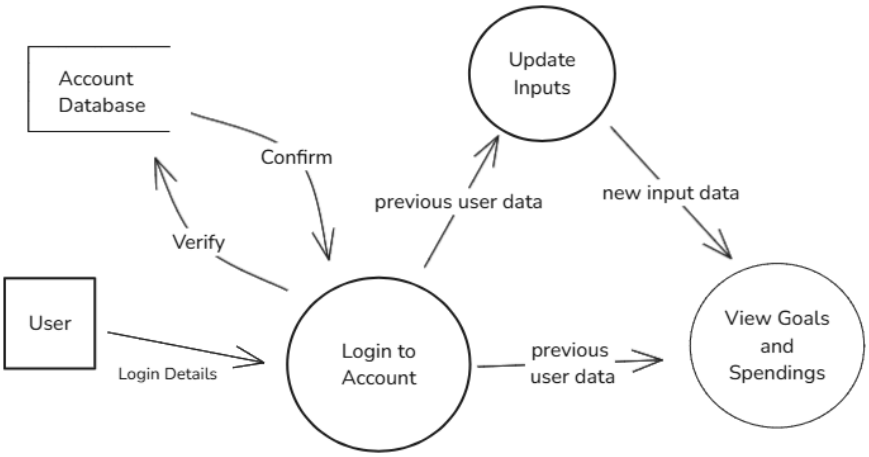
3.1 Context Diagram



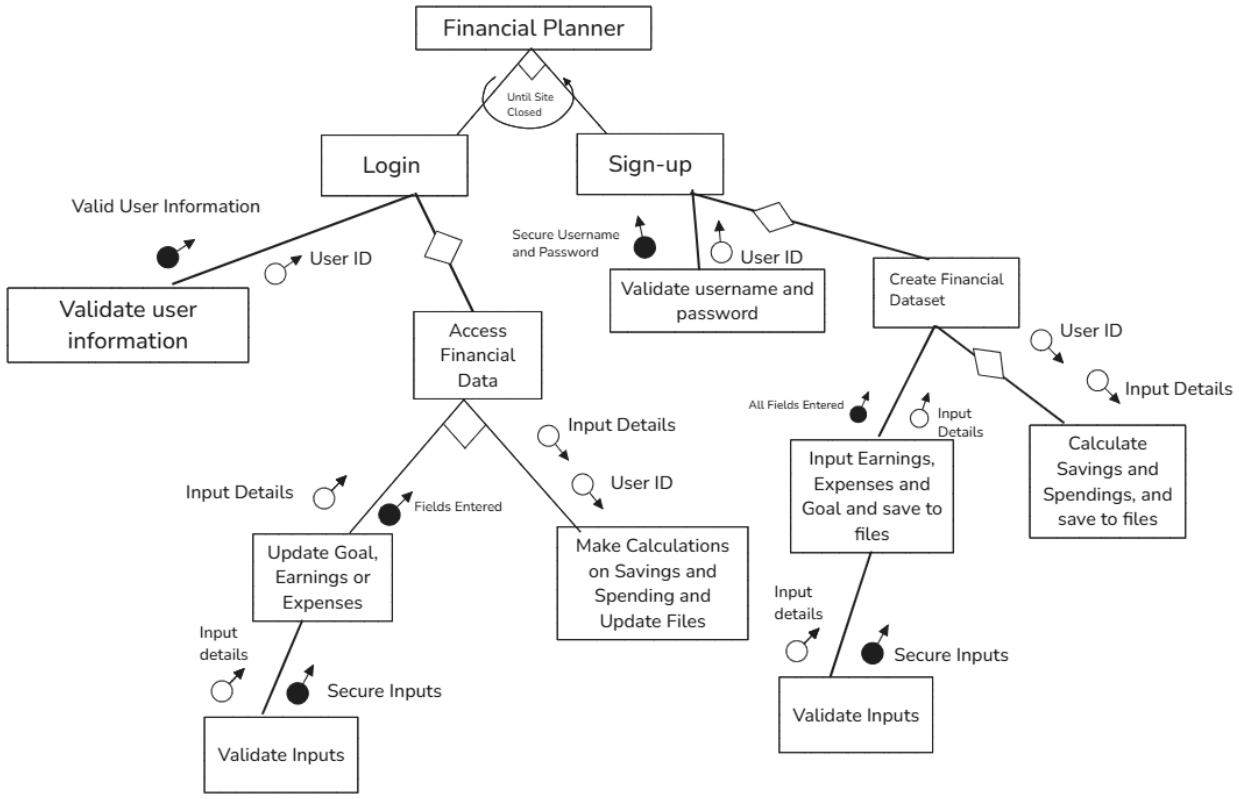
3.2 Data Flow Diagrams (Level 0 and 1)

Level 0

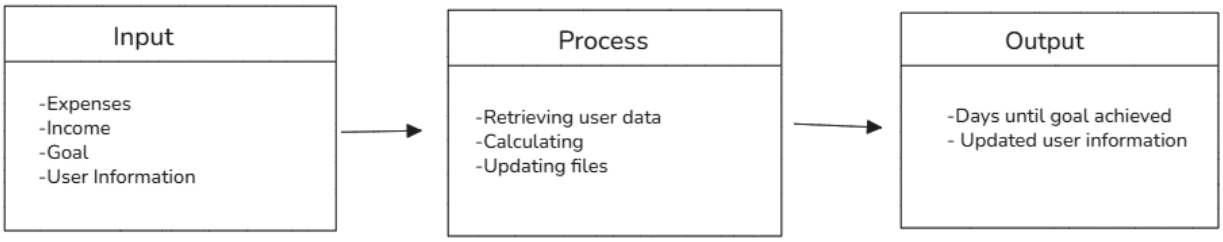
Level 1



3.3 Structure Chart



3.4 IPO Chart



3.5 Data Dictionary

Field Name	Data Type	Description	Example
User ID	Integer	User ID to identify and validate user and access user data	12345
Username	String	Username to identify user attempting to access program	JohnDoe1
Password	String	Password to grant user access to program and validate ID	Duck123
Expenses	Float	Money user has to spend during a month	$210.50
Goal	Float	Goal user is attempting to reach by the end of the month	$500
Income	Float	Users Weekly Income	$330
Days	Float	Number of Days until the user saves up completely	12.434

---

4. Producing and Implementing

4.1 Development Process

The development process was approached by initially planning for a signup page, a login page and a planner page. The signup html, css and javascript files were created following an online tutorial. Security was considered in this process to prevent malicious injection by stopping form actions if the input fields requirements were not met. This process was repeated to create the login html file, while utilizing the preexisting css and javascript files to improve aesthetic and feasibility.

The calculation page was then created using the same stylistic features from the login and signup page in the planner's css file for aesthetic consistency across the site. The calculation function was made within Javascript. Initially, if a non-numeric character was input as an expense, the function would return an error, or if the calculation returned a negative value, it would appear in the variable anyways. This error was fixed by using javascript to prompt the user to recheck their inputs, and preventing the site from displaying an invalid value.

Once the planner was completed, the security surrounding the login and signup page was reconsidered. Multiple attempts were made to save user information and allow log in, however due to the timeframe this did not succeed. Therefore, the site was made to allow users to access the planner once submitting their signup details, or using the correct login "Admin@gmail" with the password "zebra1234".

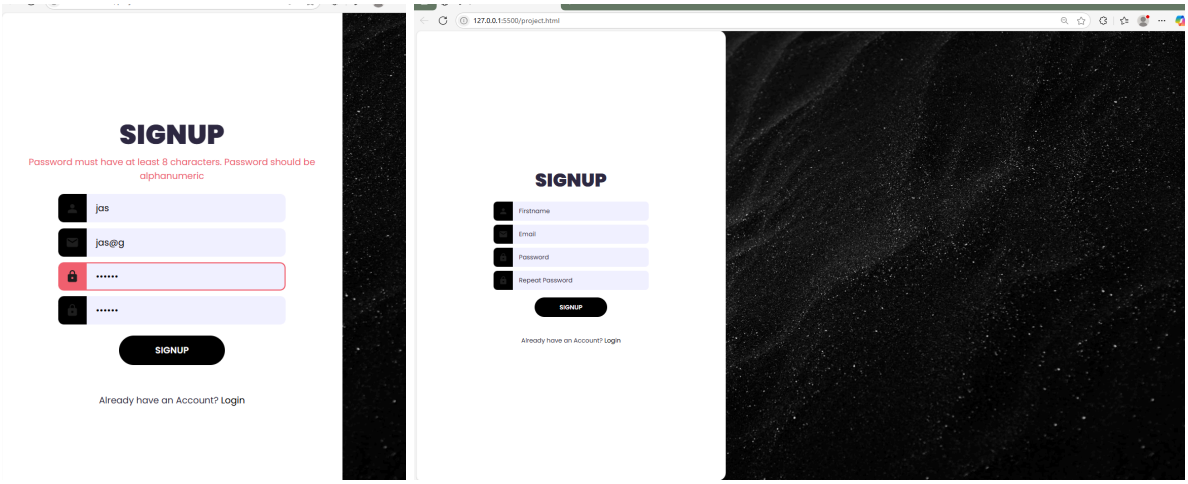
4.2 Key Features Developed

The key features are the security implementation, and the calculation feature. Within the signup page feature, security has been incorporated to prevent malicious injection or XSS. If the user enters invalid information, i.e. a password that is not alphanumeric, or a repeated password that does not match the initial one, the javascript event listener (fig 1.1) is employed on clicking the submit button to prevent the form from refreshing, prompting an error message and asking users to alter their details, utilizing the 'GetSignupFormErrors' function (fig1.2). This secure coding practice is also implemented in the login page, also using the event listener, however employing the function 'GetLoginFormErrors' (fig1.3) to prevent access if the user has invalid login details.

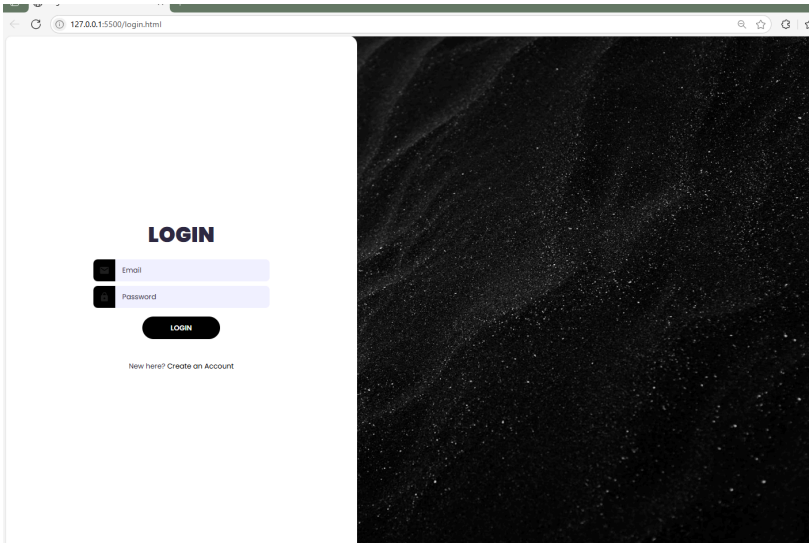
The calculation feature uses the inputs on the planner page to calculate the number of days until the user's goal is reached. It employs the javascript function 'performCalculation' (fig 1.4) when the 'calculate' button is clicked. Additionally, if the calculation results in a negative number or is invalid due to non-numeric inputs, an error message will appear.

4.3 Screenshots of Interface

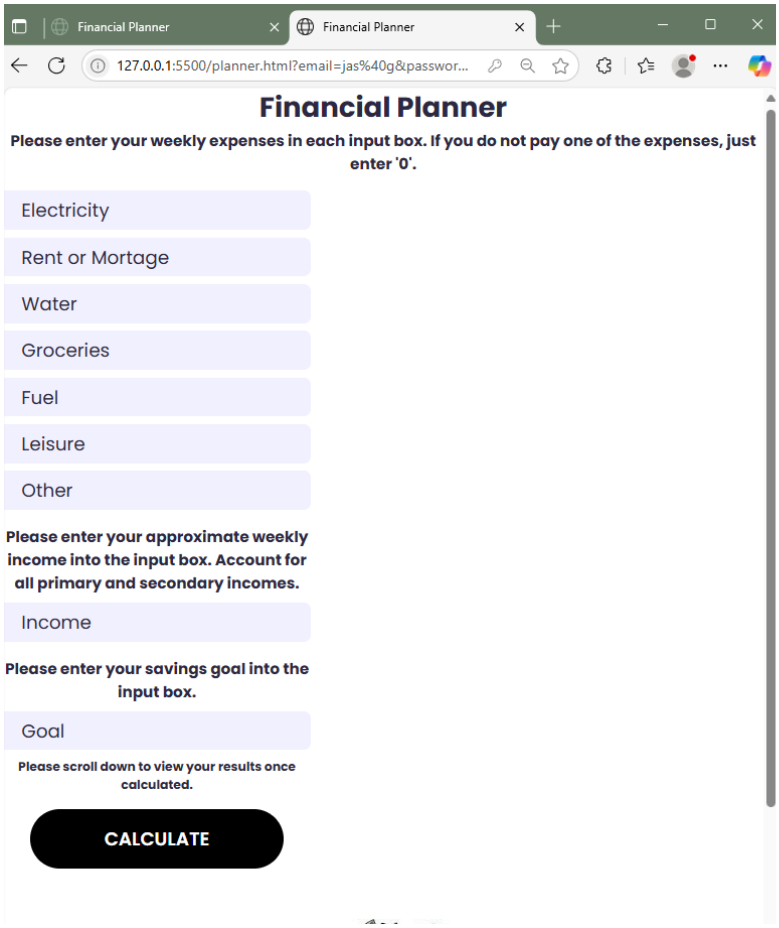
The Signup Page allows users to enter their email, name and create a password. If the user's inputs are invalid, an error message will appear. The Signup button links to the planner page.

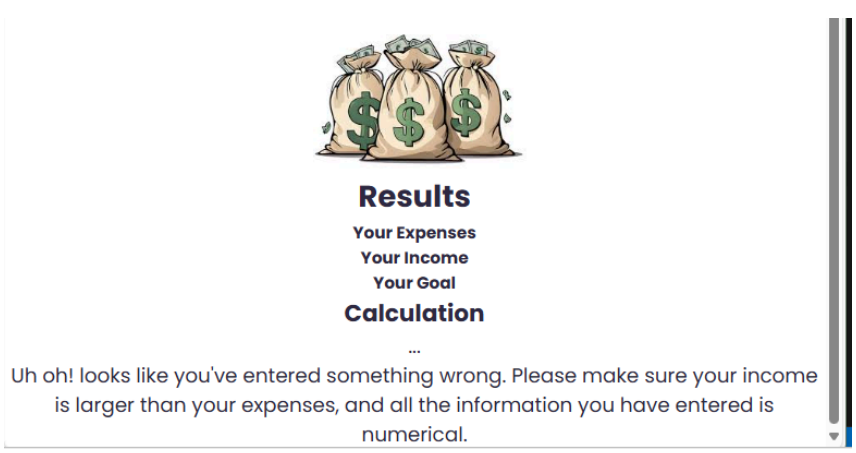


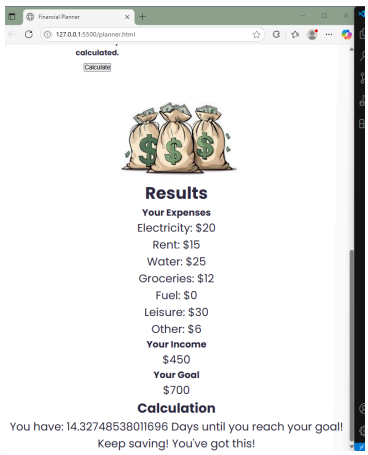
The Login Page allows users to enter their email and password information to grant access to the planner page of the website.



The Calculation Page allows users to input their expenses, income and savings goal information, and displays the number of days they have until they reach their goal. If an invalid value is entered, an error message will encourage the user to try again. Otherwise, a summary of their expenses, income and goal, the number of days until they reach their goal and an encouraging message will appear.

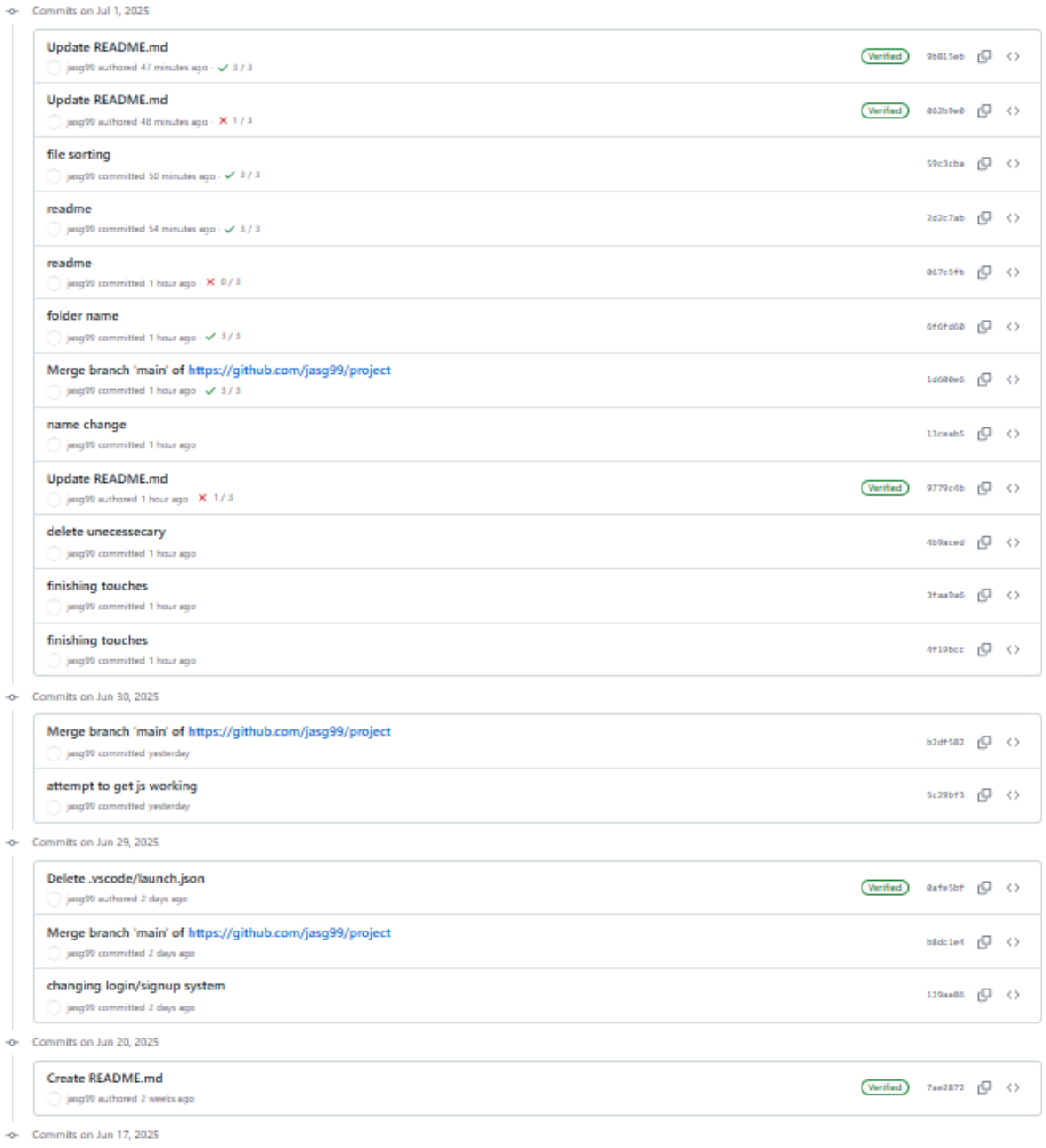

Financial Planner
Financial Planner
127.0.0.1:5500/planner.html?email=jas%40g&passwor...
Financial Planner
Please enter your weekly expenses in each input box. If you do not pay one of the expenses, just enter '0'.
Electricity
Rent or Mortage
Water
Groceries
Fuel
Leisure
Other
Please enter your approximate weekly income into the input box. Account for all primary and secondary incomes.
Income
Please enter your savings goal into the input box.
Goal
Please scroll down to view your results once calculated.
CALCULATE

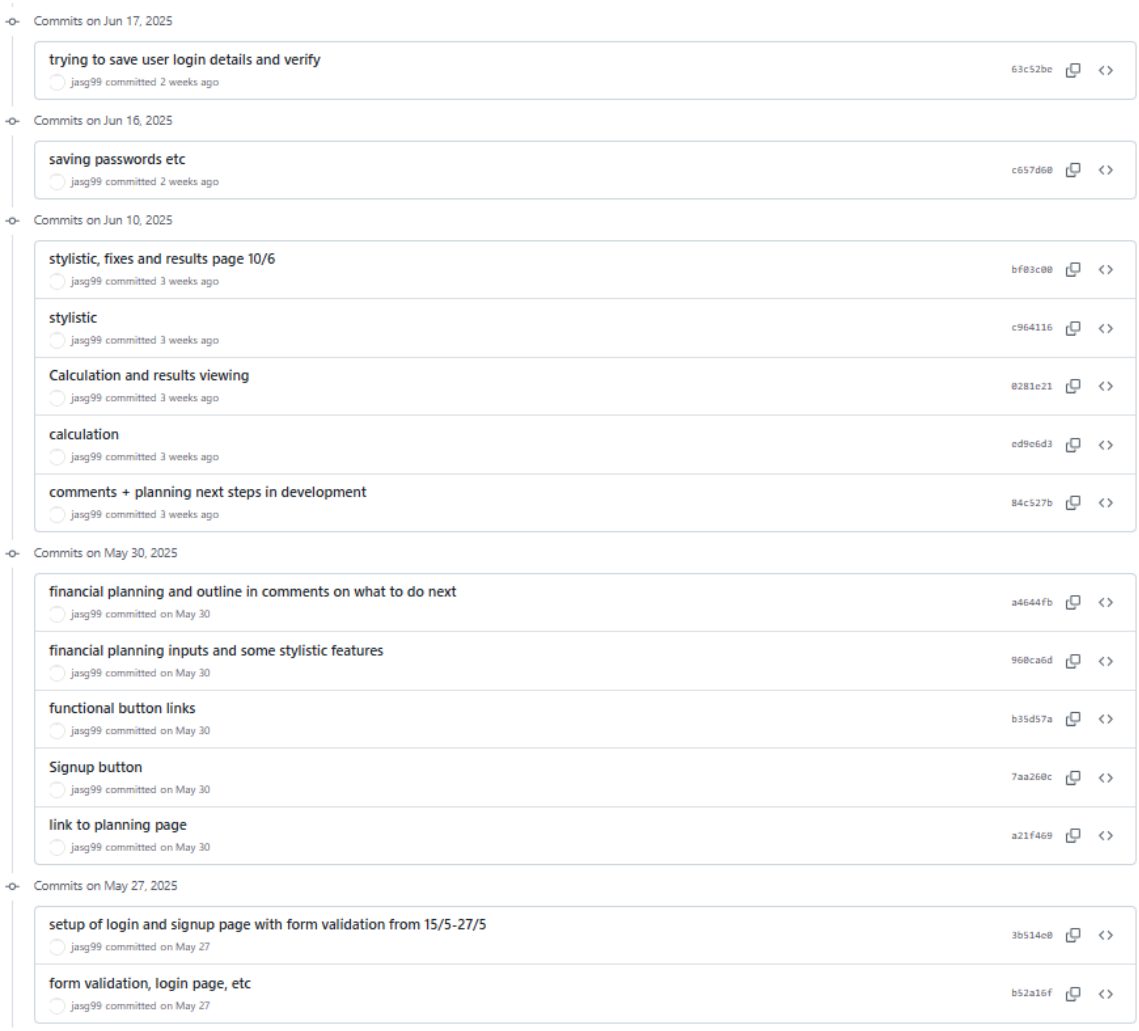

Results
Your Expenses
Your Income
Your Goal
Calculation
...
Uh oh! looks like you've entered something wrong. Please make sure your income is larger than your expenses, and all the information you have entered is numerical.


Results
Your Expenses
Electricity: $20
Rent: $15
Water: $25
Groceries: $12
Fuel: $0
Leisure: $30
Other: $6
Your Income
$450
Your Goal
$700
Calculation
You have: 14.32748538011696 Days until you reach your goal!
Keep saving! You've got this!

4.4 Version Control Summary (optional)

List of commits from Github Repository.





---

5. Testing and Evaluation

5.1 Testing Methods Used

Testing will be employed continuously throughout the development stage to prevent major bugs or errors, and prior to deployment to ensure the program is complete and satisfies all requirements. Test data are inputs used to ensure each feature that requires inputs can validate and handle various data types. Validation testing is integrated within the program, preventing malicious text inputs into usernames, emails, passwords or text inputs. Unit testing tests individual functions.

5.2 Test Cases and Results

Test ID	Description	Expected Result	Actual Result	Pass/Fail
TC01	Unit test - Signup page basics, icons and input boxes.	Signup page title appears, and each input box appears on the page and is able to be typed in, and has the correct icon next to each one.	Signup page title, each input box appears and is able to be typed in, and has the correct icon next to each one.	Pass
TC02	Unit test - CSS background and style test	Background image	Background image	Pass
TC03	Integration Test - Combine CSS and Signup form	Graphics and style	Graphics and style	Pass
TC04	Unit Test - Signup button	Signup button functional - link to planner	Signup button functional - link to planner	Pass
TC05	System Test - Complete Signup page with Login Page Link	Login page link functional and full page completed	Login page link functional and full page completed	Pass
TC06	TC06 - Acceptance Test - Testing security of inputs.	Error message and prevent page action if incorrect inputs, i.e. passwords do not match, all fields not entered, non-alphanumeric password.	Error message and prevent page action if incorrect inputs, i.e. passwords do not match, all fields not entered, non-alphanumeric password.	Pass
TC07	System Test, Integration Test - Complete Login Page	Login Page functional, input boxes can be typed in, CSS intact, button links to planner	Login Page functional, input boxes can be typed in, CSS intact, button links	Pass

		page.	to planner page.	
TC08	Acceptance Test - Testing functionality and clear error messages if incorrect inputs.	Error message if no fields entered, or invalid email.	Error message if no fields entered, or invalid email.	Pass
TC09	Unit test - Testing Input Boxes and Page Setup	Page appears as anticipated. Input boxes are usable.	Page appears as anticipated. Input boxes are usable.	Pass
TC10	Unit test - Testing Calculate Button and Style	The Calculate button is on the page. The style sheet adds to the aesthetic of the page.	The Calculate button is on the page. The style sheet adds to the aesthetic of the page.	Pass
TC11	Unit Test - Testing initial Javascript Calculation	Calculate button outputs numeric value in the correct format.	Calculate button outputs numeric value in the correct format.	Pass
TC12	System Test - Testing Correct Inputs	Calculate button outputs numeric value and restates inputs.	Calculate button outputs numeric value and restates inputs.	Pass
TC13	System Test - Testing Incorrect Inputs	Calculate button outputs error message, and no value	Function attempted to calculate letters, leading to error being displayed.	Fail
TC14	System Test - Testing Incorrect Inputs	Calculate button outputs error message, and no value	Calculate button outputs error message, and no value	Pass
TC15	testing saving user details through localServer	User is prevented from logging in without signup/ registered account first	User was granted access with invalid login	Fail

TC16	saving user details through JSON	User is prevented from logging in without signup/ registered account first	User was granted access with invalid login	Fail
TC17	Admin login (one password and user allowed)	Test Data “*Admin@gmail” “zebra1234”* Grants entry to site.	Test Data “*Admin@gmail” “zebra1234”* Grants entry to site.	Pass
TC18	Acceptance test	Complete website functionality	Complete website functionality	Pass

5.3 Evaluation Against Requirements

The functional requirements surrounding the calculation site were exceeded, with a feasible user interface and functionality, however it does not adequately meet the security requirements. This was primarily due to the limited time constraints and resources, limiting time to develop a user registration saving function, as well as encryption of user information.

5.4 Improvements and Future Work

Considering an extended timeframe, additional features could include advanced security, for instance, saving user details within a JSON file using hashing and encryption, allowing users to login and access their previous information on the site. Additional graphics, such as a progress bar, could be implemented with an extended timeframe.

---

6. (Client) Feedback and Reflection

6.1 Summary of Client Feedback

Feedback within the interviews highlighted that the project has an aesthetically pleasing and easy to navigate interface on both the signup and login pages. However some found the planner page less simple to navigate, due to extended instructions and requirements. Some of the younger age range did not find the site useful, however this can be attributed to unemployment. Overall, the site met most of the user-based requirements, however, increasing the feasibility of the planner page would be beneficial.

6.2 Personal Reflection

From this project, I advanced my knowledge of HTML, Javascript and CSS, allowing me to create a well-developed website. Additionally, I also learnt how to navigate Github, which I had

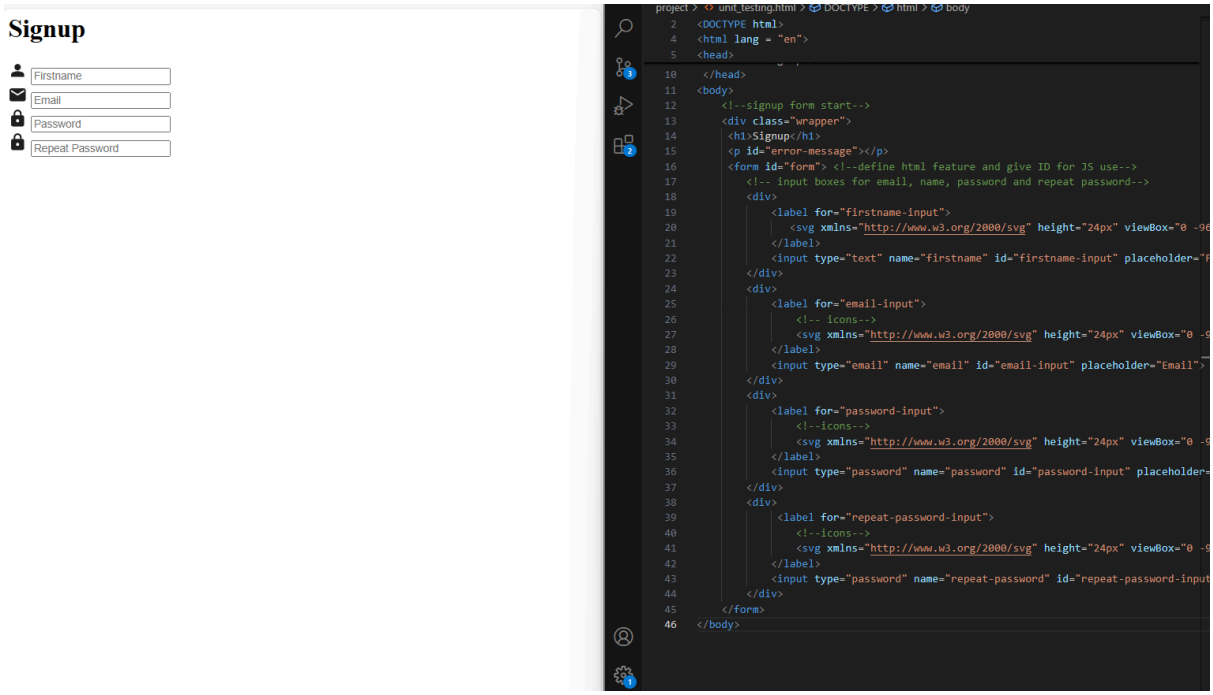
previously not used. Finally, I discovered the importance of time management and the project timeline, as it prevented me from feeling overwhelmed, and gave me additional time to deal with errors or mistakes.

---

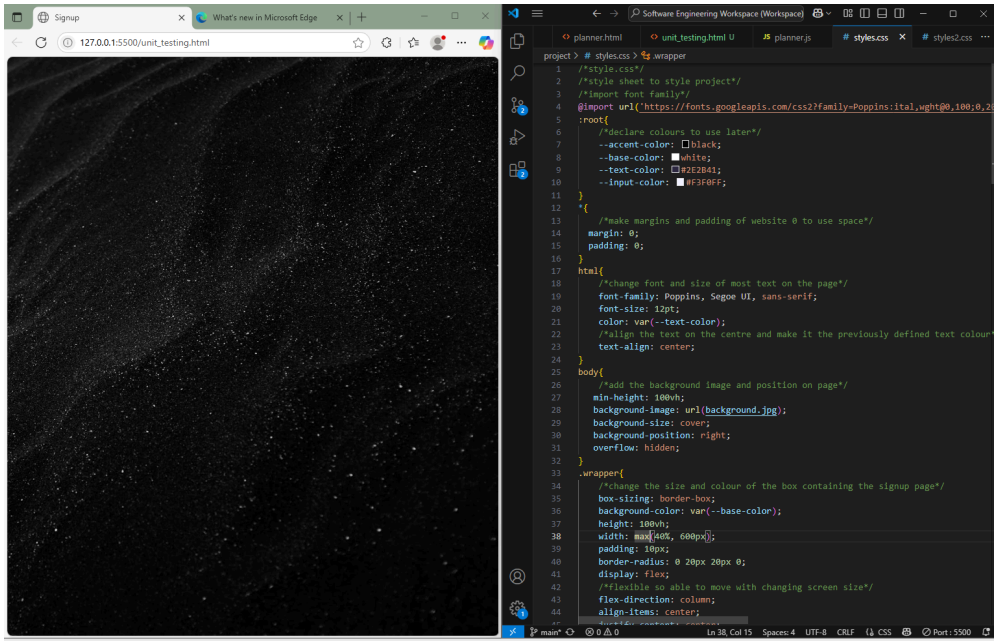
7. Appendices

- Full Test Logs

TC01 - Unit Test - Signup page form, icons and input boxes.

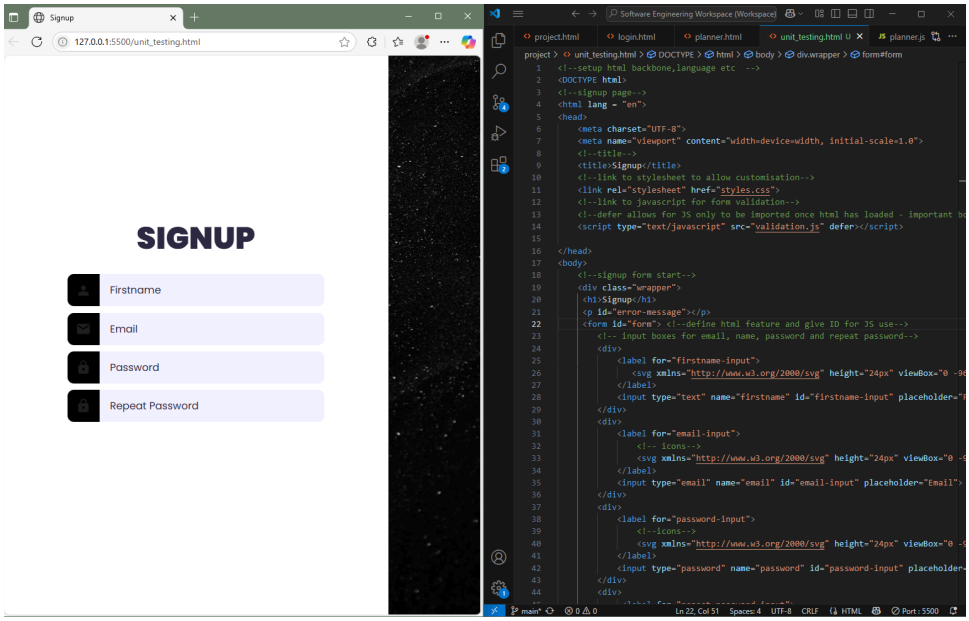


TC02 - Unit Test - CSS background and style test



```
/*style.css*/
/*style sheet to style project*/
/*import font family*/
@import url('https://fonts.googleapis.com/css2?family=Poppins:ital,wght@0,100;0,2
:root{
    /*declare colours to use later*/
    --accent-color: black;
    --base-color: white;
    --text-color: #2E2B41;
    --input-color: #F3F0FF;
}
*{
    /*make margins and padding of website 0 to use space*/
   margin: 0;
   padding: 0;
}
html{
    /*change font and size of most text on the page*/
    font-family: Poppins, Segoe UI, sans-serif;
    font-size: 12pt;
    color: var(--text-color);
    /*align the text on the centre and make it the previously defined text colour
    text-align: center;
}
body{
    /*add the background image and position on page*/
    min-height: 100vh;
    background-image: url(background.jpg);
    background-size: cover;
    background-position: right;
    overflow: hidden;
}
.wrapper{
    /*change the size and colour of the box containing the signup page*/
    box-sizing: border-box;
    background-color: var(--base-color);
    height: 100vh;
    width: max(40%, 600px);
    padding: 10px;
    border-radius: 0 20px 20px 0;
    display: flex;
    /*flexible so able to move with changing screen size*/
    flex-direction: column;
    align-items: center;
```

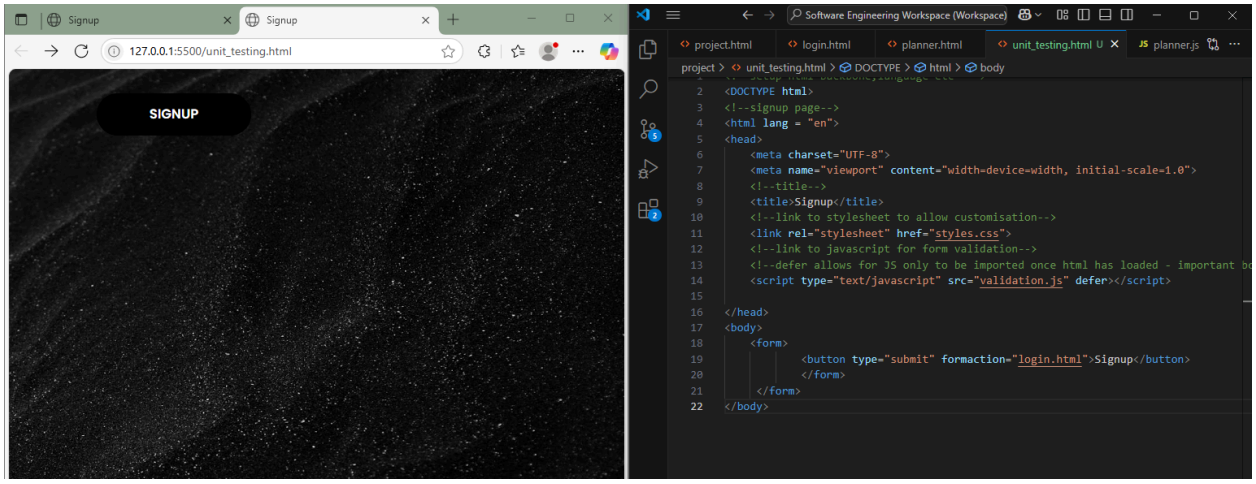
TC03 - Integration Test - Combine CSS and Signup form



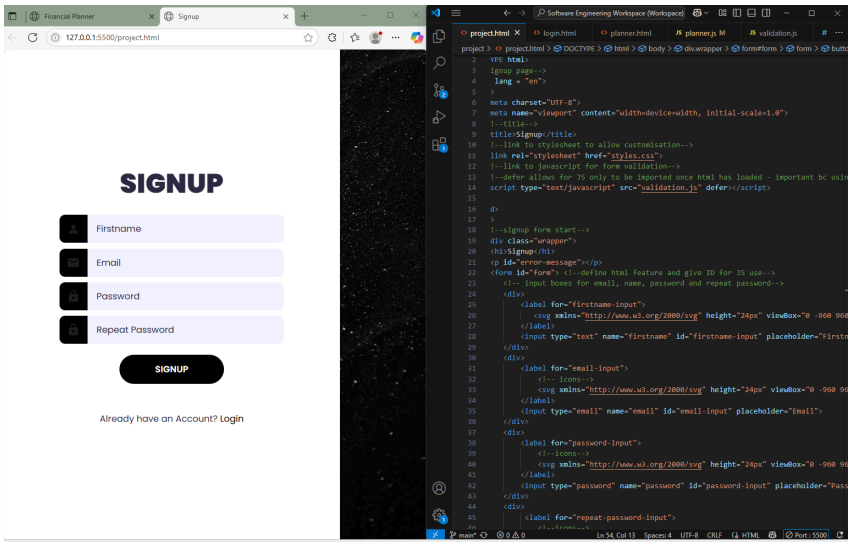

```
<!--setup html backbone,language etc  -->
<DOCTYPE html>
<!--signup page-->
<html lang = "en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <!--title-->
    <title>Signup</title>
    <!--link to stylesheet to allow customisation-->
    <link rel="stylesheet" href="styles.css">
    <!--link to javascript for form validation-->
    <!--defer allows for JS only to be imported once html has loaded - important
    <script type="text/javascript" src="validation.js" defer></script>

</head>
<body>
    <!--signup form start-->
    <div class="wrapper">
      <h1>Signup</h1>
      <p id="error-message"></p>
      <form id="form"> <!--define html feature and give ID for JS use-->
        <!-- input boxes for email, name, password and repeat password-->
        <div>
            <label for="firstname-input">
                <svg xmlns="http://www.w3.org/2000/svg" height="24px" viewBox="0 -96
            </label>
            <input type="text" name="firstname" id="firstname-input" placeholder="F
        </div>
        <div>
            <label for="email-input">
                <!-- icons-->
                <svg xmlns="http://www.w3.org/2000/svg" height="24px" viewBox="0 -9
            </label>
            <input type="email" name="email" id="email-input" placeholder="Email">
        </div>
        <div>
            <label for="password-input">
                <!--icons-->
                <svg xmlns="http://www.w3.org/2000/svg" height="24px" viewBox="0 -9
            </label>
            <input type="password" name="password" id="password-input" placeholder=
        </div>
        <div>
```

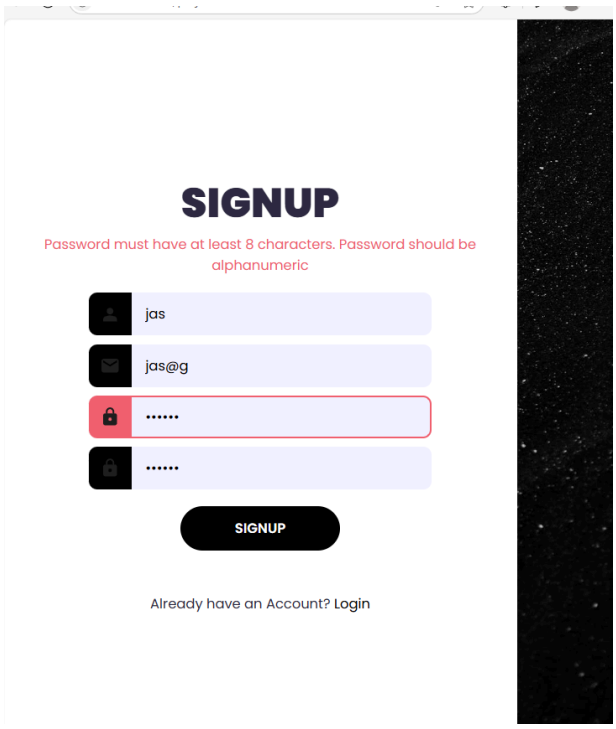
TC04 - Unit Test - Signup button

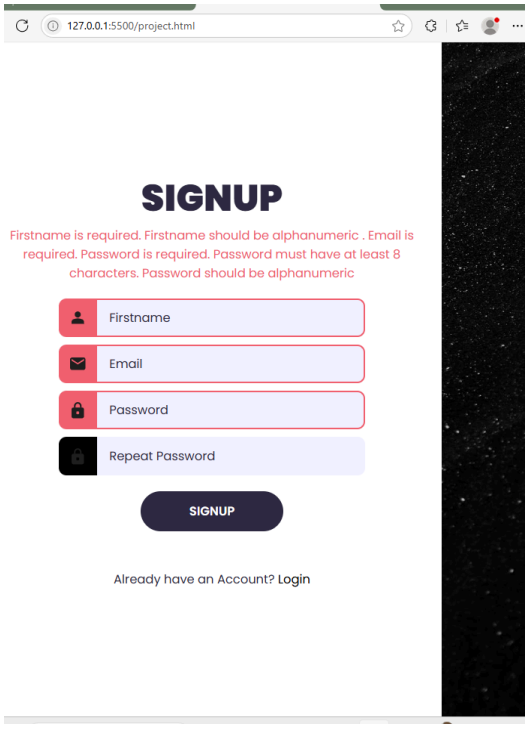


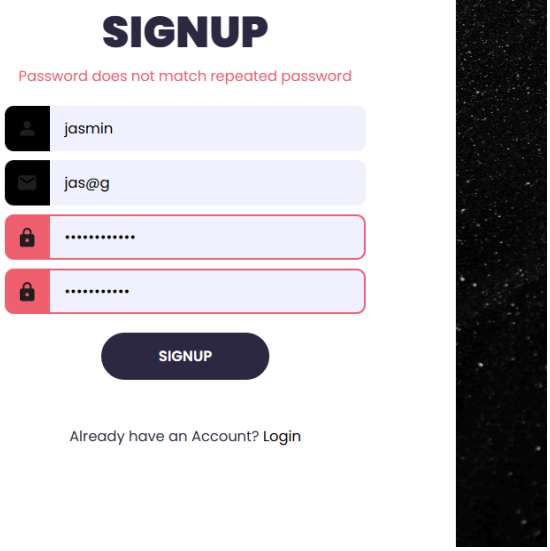
TC05 - System Test - Complete Signup page with Login Page Link

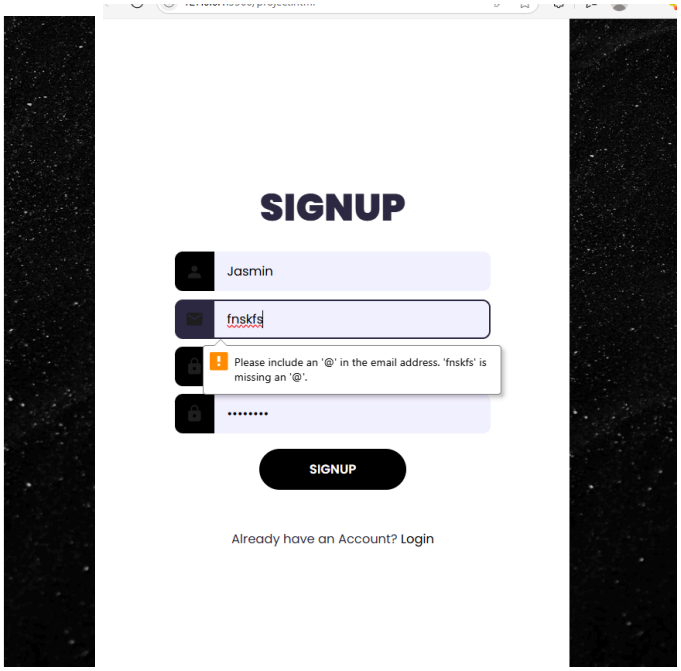


TC06 - Acceptance Test - Testing security of inputs. Ensuring actual email address, All fields entered, passwords match and passwords are alphanumeric to prevent XSS or injection. Test Data for alphanumeric is ‘12345!’

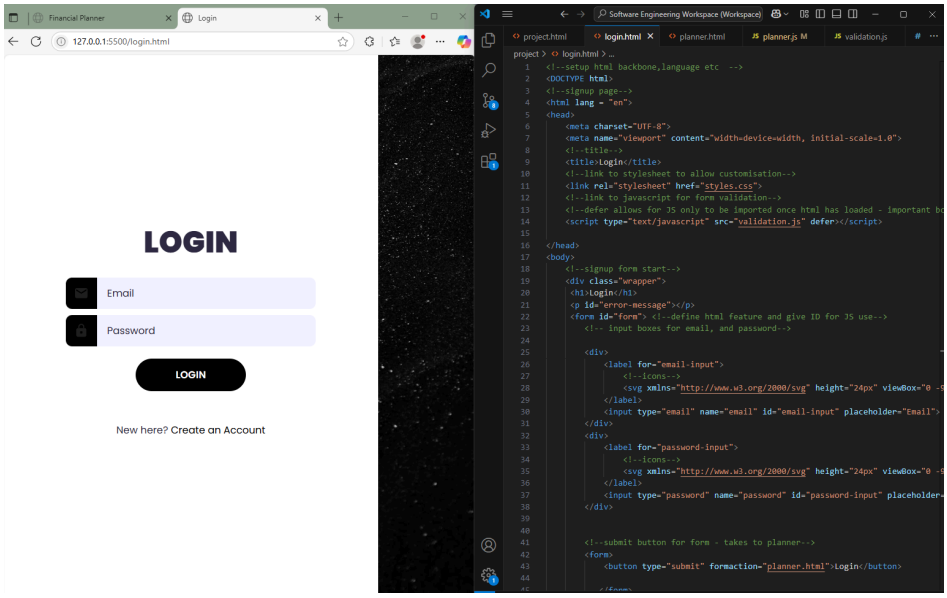

SIGNUP
Password must have at least 8 characters. Password should be alphanumeric
jas
jas@g
SIGNUP
Already have an Account? Login


127.0.0.1:5500/project.html
SIGNUP
Firstname is required. Firstname should be alphanumeric . Email is required. Password is required. Password must have at least 8 characters. Password should be alphanumeric
Firstname
Email
Password
Repeat Password
SIGNUP
Already have an Account? Login

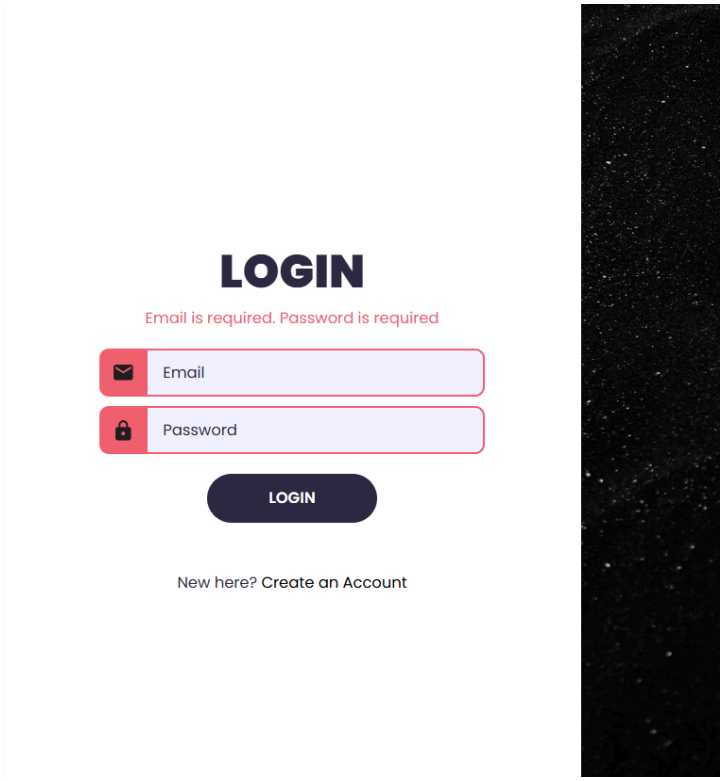

SIGNUP
Password does not match repeated password
jasmin
jas@g
SIGNUP
Already have an Account? Login


SIGNUP
Jasmin
fnskfs
Please include an '@' in the email address. 'fnskfs' is missing an '@'.
SIGNUP
Already have an Account? Login

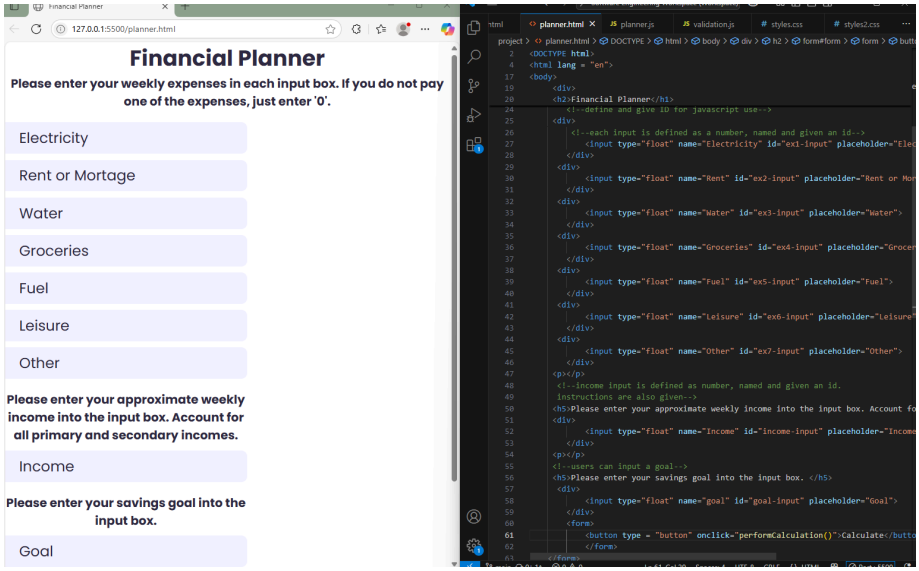
TC07 - System Test, Integration Test - Complete Login Page



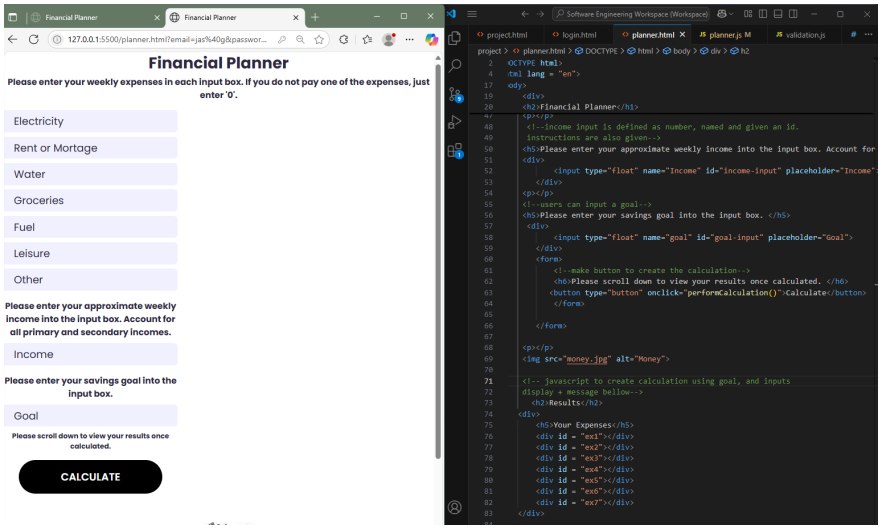
TC08 - Acceptance Test - Testing functionality and clear error messages if incorrect inputs.



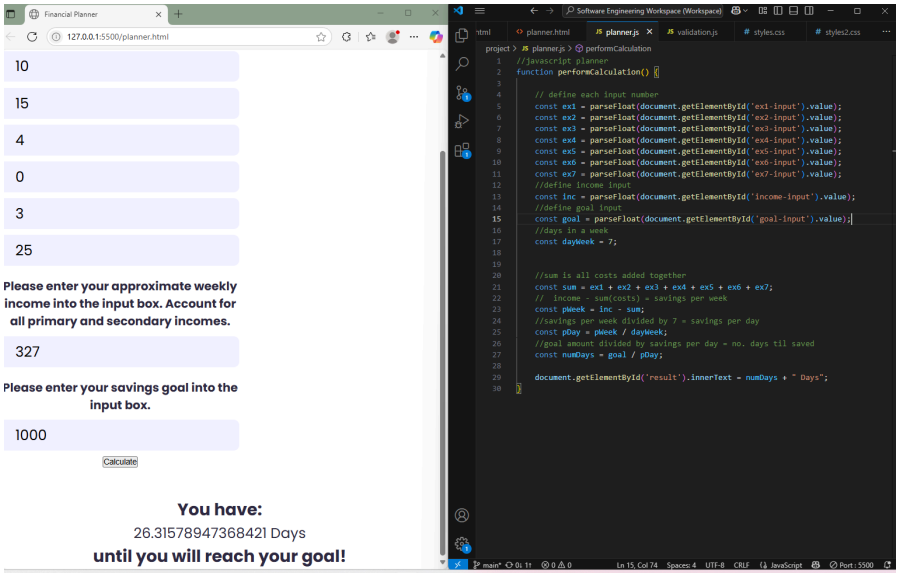
TC09 - Unit test - Testing Input Boxes and Page Setup



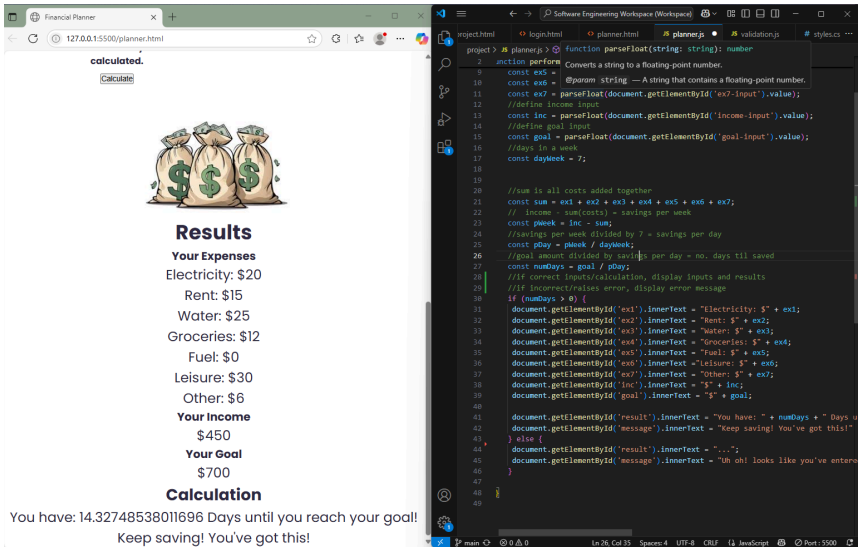
TC10 - Testing Calculate Button and Style



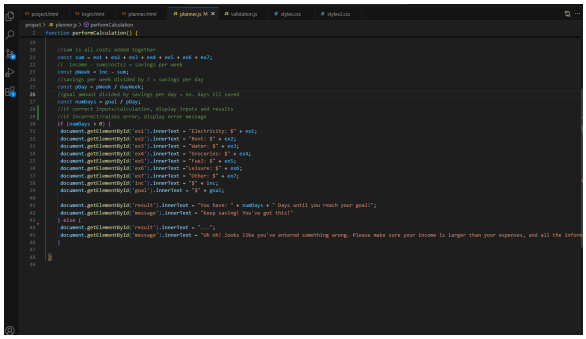
TC11 - Unit Test - Testing initial Javascript Calculation

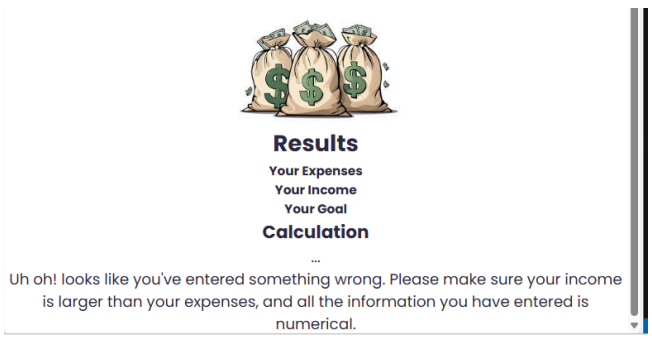


TC12 - System Test - Testing Correct Inputs

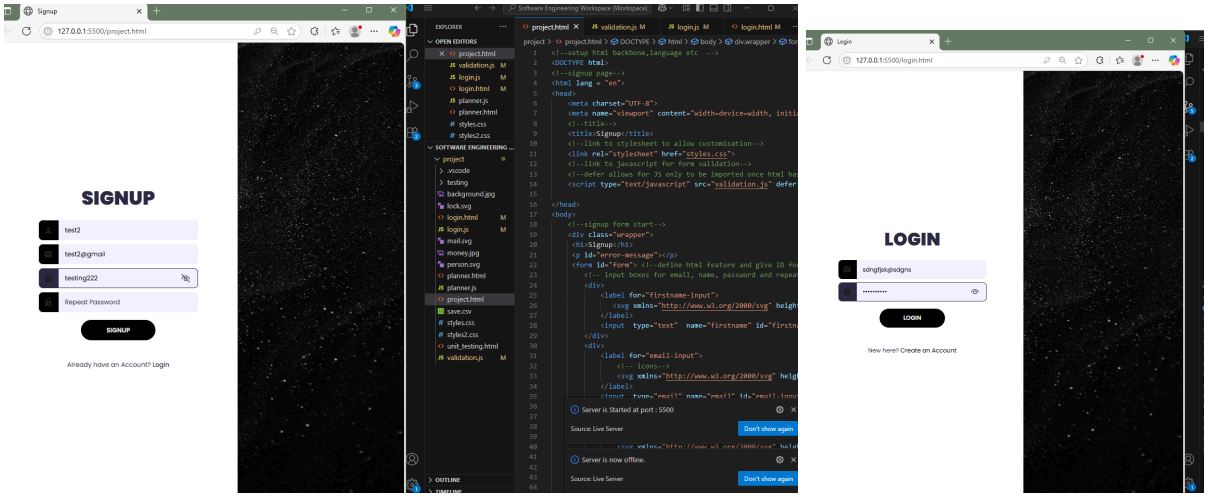


TC14 - System Test - Testing Incorrect Inputs

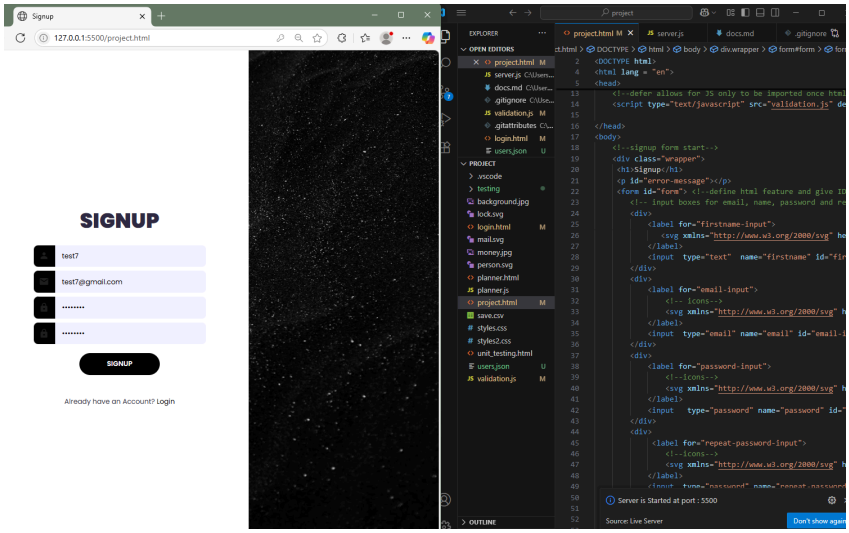


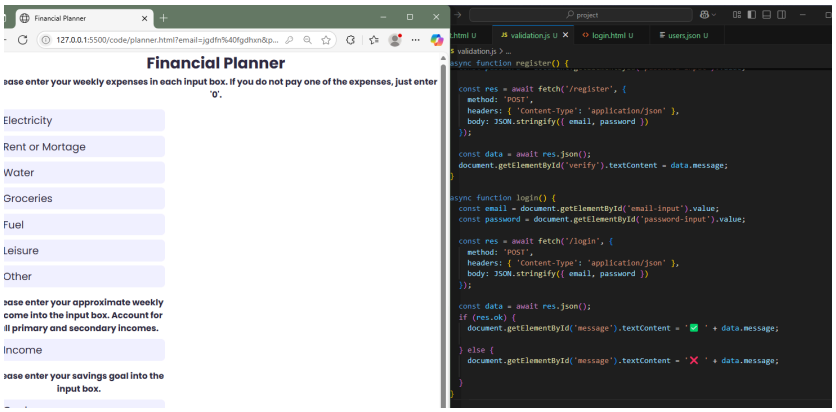


TC15 - testing saving user details through localServer



TC16- saving user details through JSON





- Raw Interview Notes/Feedback Forms

The individual will answer on a scale of 1-5, 1 being very poor, 5 being very satisfactory

INTERVIEW 1 - 17yr old female

Is the page aesthetically pleasing to you?
5
Is the signup page easy to navigate?
5
Is the login page easy to navigate?
5
Is the calculation function easy to understand and navigate?
5
Is this site useful to you?
5

INTERVIEW 2 - 19yr old male
Is the page aesthetically pleasing to you?
5
Is the signup page easy to navigate?
5
Is the login page easy to navigate?
5
Is the calculation function easy to understand and navigate?
4
Is this site useful to you?
4

INTERVIEW 3 - 16 yr old female
Is the page aesthetically pleasing to you?
5
Is the signup page easy to navigate?
5
Is the login page easy to navigate?
5
Is the calculation function easy to understand and navigate?
4
Is this site useful to you?
2

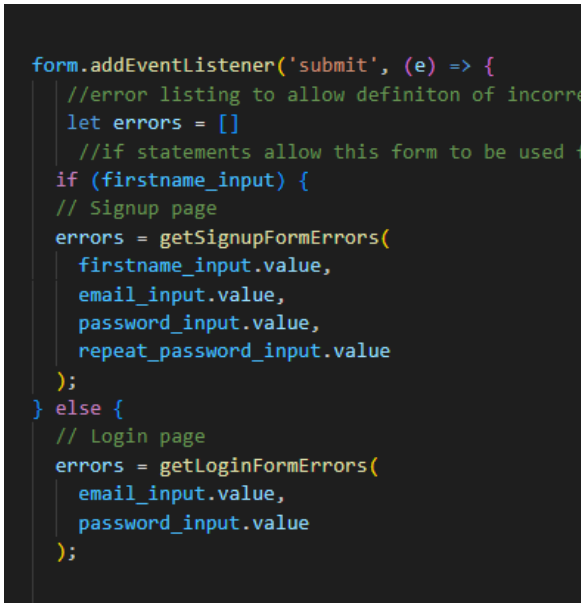
INTERVIEW 4 - 17yr old male
Is the page aesthetically pleasing to you?
4
Is the signup page easy to navigate?
5

Is the login page easy to navigate?
5
Is the calculation function easy to understand and navigate?
4
Is this site useful to you?
5

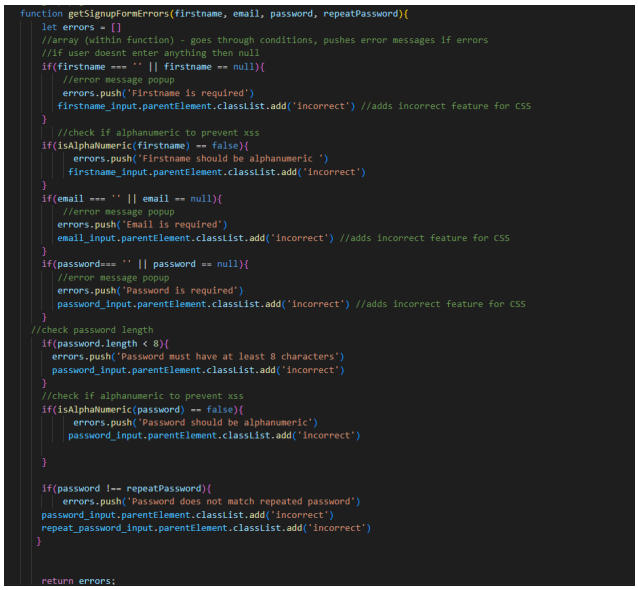
INTERVIEW 5 - 21 year old female
Is the page aesthetically pleasing to you?
5
Is the signup page easy to navigate?
5
Is the login page easy to navigate?
5
Is the calculation function easy to understand and navigate?
5
Is this site useful to you?
5

Exemplar Code Snippets

1.1 and 1.2



```
form.addEventListener('submit', (e) => {
  //error listing to allow definiton of incorr
  let errors = []
  //if statements allow this form to be used
  if (firstname_input) {
  // Signup page
  errors = getSignupFormErrors(
    firstname_input.value,
    email_input.value,
    password_input.value,
    repeat_password_input.value
  );
} else {
  // Login page
  errors = getLoginFormErrors(
    email_input.value,
    password_input.value
  );
```



```
function getSignupFormErrors(firstname, email, password, repeatPassword){
    let errors = []
    //array (within function) - goes through conditions, pushes error messages if errors
    //if user doesnt enter anything then null
    if(firstname === '' || firstname == null){
        //error message popup
        errors.push('Firstname is required')
        firstname_input.parentElement.classList.add('incorrect') //adds incorrect feature for CSS
    }
    //check if alphanumeric to prevent xss
    if(isAlphaNumeric(firstname) == false){
        errors.push('Firstname should be alphanumeric ')
        firstname_input.parentElement.classList.add('incorrect')
    }
    if(email === '' || email == null){
        //error message popup
        errors.push('Email is required')
        email_input.parentElement.classList.add('incorrect') //adds incorrect feature for CSS
    }
    if(password=== '' || password == null){
        //error message popup
        errors.push('Password is required')
        password_input.parentElement.classList.add('incorrect') //adds incorrect feature for CSS
    }
  //check password length
    if(password.length < 8){
        errors.push('Password must have at least 8 characters')
        password_input.parentElement.classList.add('incorrect')
    }
    //check if alphanumeric to prevent xss
    if(isAlphaNumeric(password) == false){
        errors.push('Password should be alphanumeric')
        password_input.parentElement.classList.add('incorrect')

    }

    if(password !== repeatPassword){
        errors.push('Password does not match repeated password')
    password_input.parentElement.classList.add('incorrect')
    repeat_password_input.parentElement.classList.add('incorrect')
    }


    return errors;
```

1.3 and 1.4

```
//login page validation
function getLoginFormErrors(email, password){
  let errors = []
  const trimmedPassword = password.trim()
  if (email === '' || email == null || email !== 'Admin@gmail') {
    errors.push('Email is invalid')
    email_input?.parentElement?.classList.add('incorrect')
  }
  if (trimmedPassword === '' || trimmedPassword !== 'zebra1234') {
    errors.push('Password is invalid')
    password_input?.parentElement?.classList.add('incorrect')
  }
  return errors
}

const allInputs = [ email_input, password_input].filter(input => input != null)
//undo red outline if fixed
allInputs.forEach(input => {
    input.addEventListener('input', () => {
        if(input.parentElement.classList.contains('incorrect')){
            input.parentElement.classList.remove('incorrect')
            error_message.innerText = ''
        }
    })
})
 // Function to check if a string is alphanumeric
function isAlphaNumeric(str) {
  // Using regular expression to check for alphanumeric characters
  return /^[a-zA-Z0-9]+$/.test(str);
}
```

```
function performCalculation() {

    // define each input number
    const ex1 = parseFloat(document.getElementById('ex1-input').value);
    const ex2 = parseFloat(document.getElementById('ex2-input').value);
    const ex3 = parseFloat(document.getElementById('ex3-input').value);
    const ex4 = parseFloat(document.getElementById('ex4-input').value);
    const ex5 = parseFloat(document.getElementById('ex5-input').value);
    const ex6 = parseFloat(document.getElementById('ex6-input').value);
    const ex7 = parseFloat(document.getElementById('ex7-input').value);
    //define income input
    const inc = parseFloat(document.getElementById('income-input').value);
    //define goal input
    const goal = parseFloat(document.getElementById('goal-input').value);
    //days in a week
    const dayWeek = 7;


    //sum is all costs added together
    const sum = ex1 + ex2 + ex3 + ex4 + ex5 + ex6 + ex7;
    //  income - sum(costs) = savings per week
    const pWeek = inc - sum;
    //savings per week divided by 7 = savings per day
    const pDay = pWeek / dayWeek;
    //goal amount divided by savings per day = no. days til saved
    const numDays = goal / pDay;
    //if correct inputs/calculation, display inputs and results
    //if incorrect/raises error, display error message
    if (numDays > 0) {
     document.getElementById('ex1').innerText = "Electricity: $" + ex1;
     document.getElementById('ex2').innerText = "Rent: $" + ex2;
     document.getElementById('ex3').innerText = "Water: $" + ex3;
     document.getElementById('ex4').innerText = "Groceries: $" + ex4;
     document.getElementById('ex5').innerText = "Fuel: $" + ex5;
     document.getElementById('ex6').innerText ="Leisure: $" + ex6;
     document.getElementById('ex7').innerText = "Other: $" + ex7;
     document.getElementById('inc').innerText = "$" + inc;
     document.getElementById('goal').innerText = "$" + goal;

     document.getElementById('result').innerText = "You have: " + numDays + " Days until you reach your goal!";
     document.getElementById('message').innerText = "Keep saving! You've got this!"
    } else {
     document.getElementById('result').innerText = "...";
     document.getElementById('message').innerText = "Uh oh! looks like you've entered something wrong. Please mak
```