

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

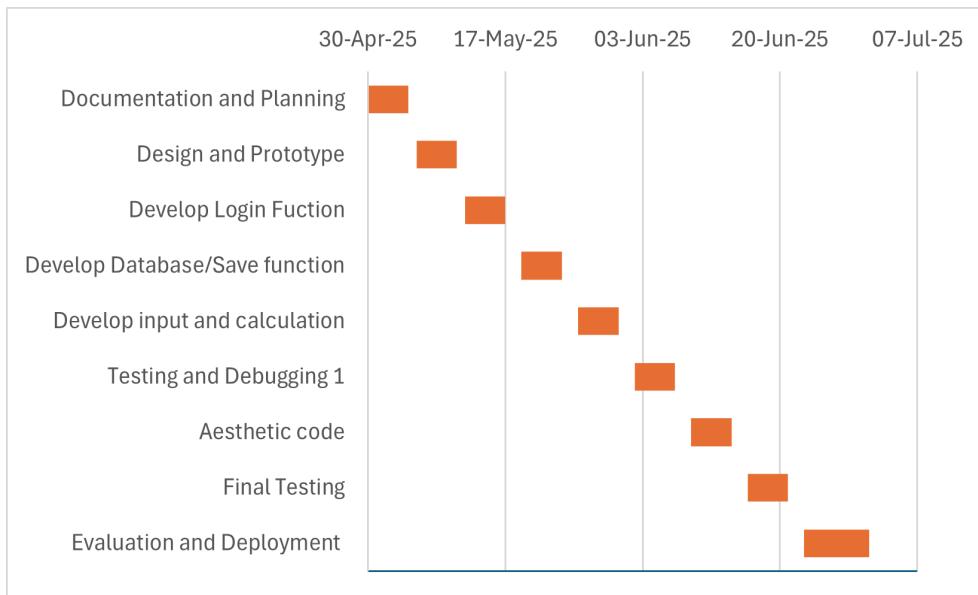
2.1 Development Methodology

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

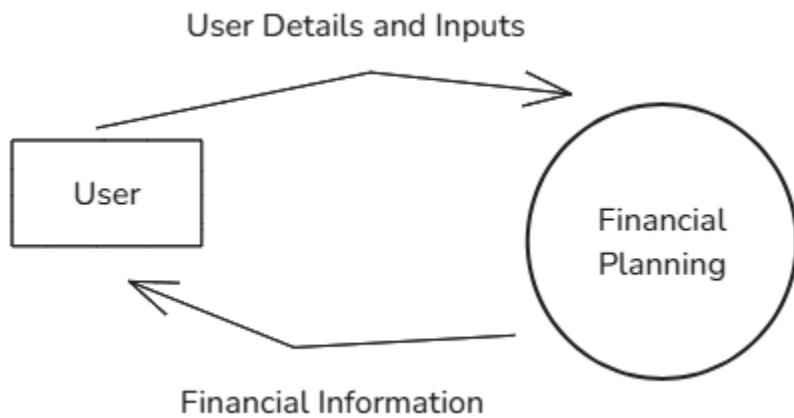
I will conduct a small survey on a group between the ages of 15 and 21, to determine the usefulness of this project. The survey will consist of 5 questions. The questionnaire will include:

- 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

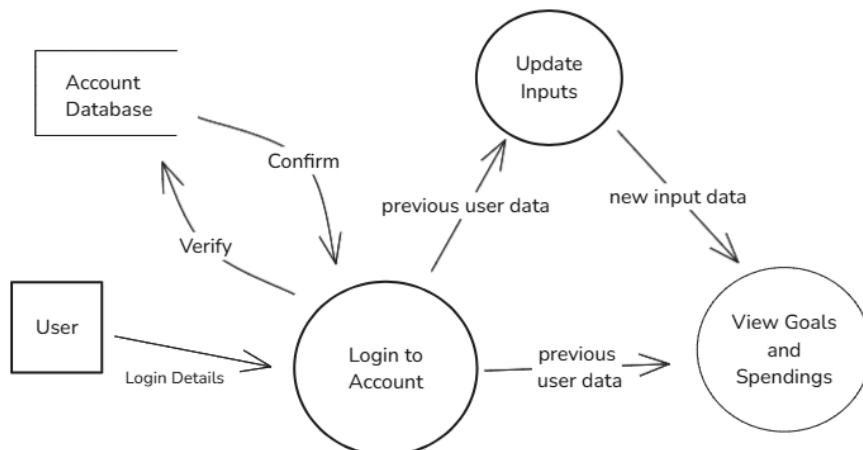


3.2 Data Flow Diagrams (Level 0 and 1)

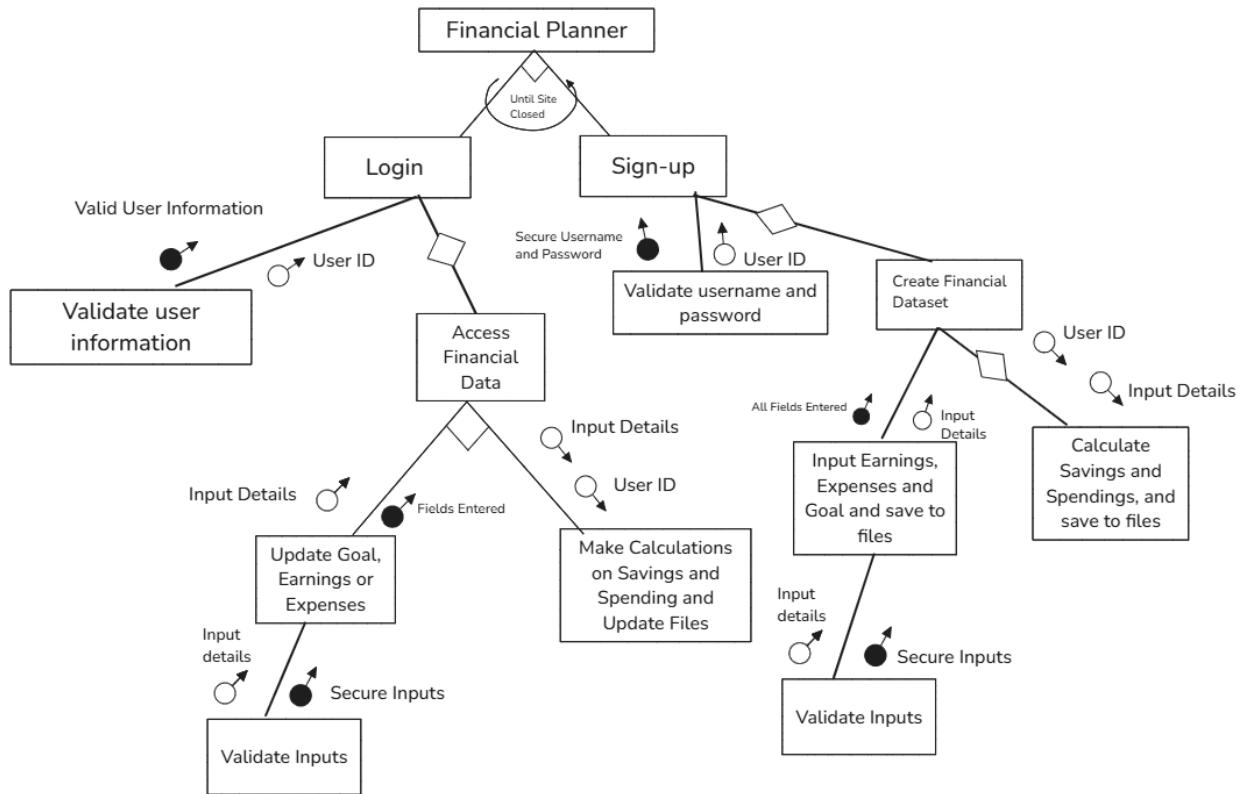
Level 0



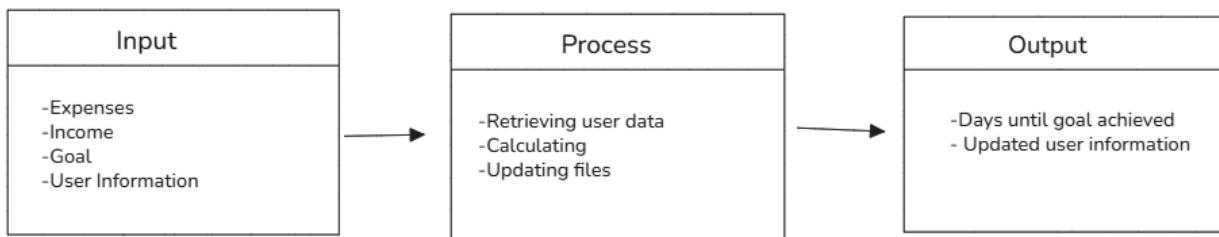
Level 1



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".

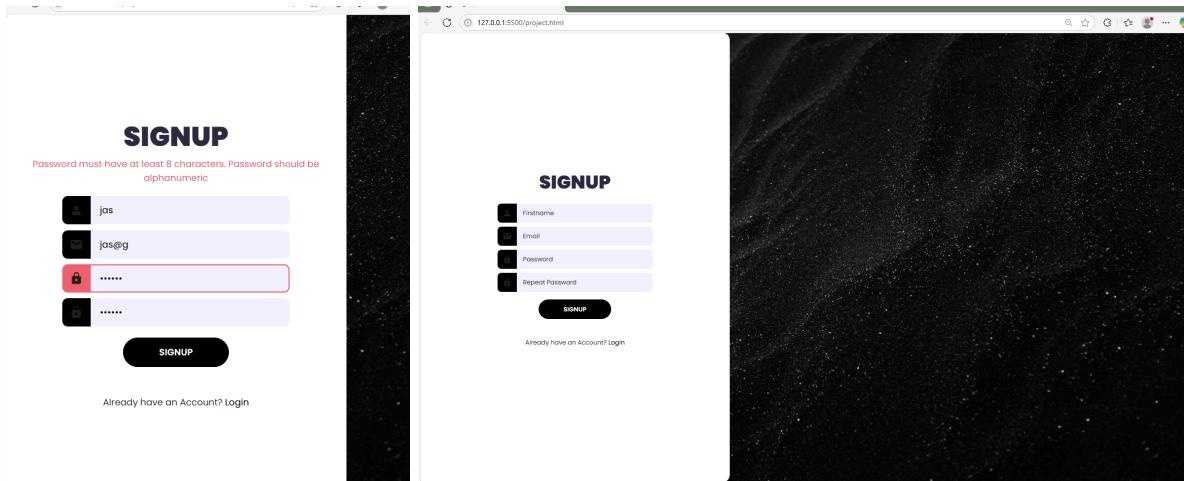
4.2 Key Features Developed

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.

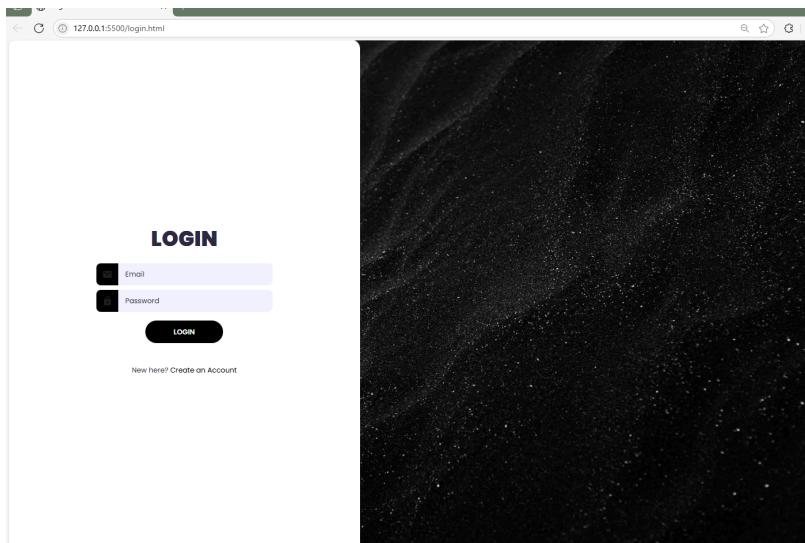
The calculation feature uses the inputs on the planner page to calculate the number of days until the user's goal is reached. It employs the javascript function 'performCalculation' (fig 1.4) when the 'calculate' button is clicked. Additionally, if the calculation results in a negative number or is invalid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

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.

Financial Planner

calculated.

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.

Commits on Jul 1, 2025				
Update README.md		9b815eb		
(jasg99 authored 47 minutes ago · 3 / 3)				
Update README.md		852b9e0		
(jasg99 authored 48 minutes ago · 1 / 3)				
file sorting		59c3cba		
(jasg99 committed 50 minutes ago · 3 / 3)				
readme		2d2c7ab		
(jasg99 committed 54 minutes ago · 3 / 3)				
readme		867c5fb		
(jasg99 committed 1 hour ago · 0 / 3)				
folder name		6f0fa00		
(jasg99 committed 1 hour ago · 3 / 3)				
Merge branch 'main' of https://github.com/jasg99/project		1d680e6		
(jasg99 committed 1 hour ago · 3 / 3)				
name change		13ceab5		
(jasg99 committed 1 hour ago)				
Update README.md		9779c0b		
(jasg99 authored 1 hour ago · 1 / 3)				
delete uncessesary		4b9aced		
(jasg99 committed 1 hour ago)				
finishing touches		3fca9a6		
(jasg99 committed 1 hour ago)				
finishing touches		4f19bcc		
(jasg99 committed 1 hour ago)				
Commits on Jun 30, 2025				
Merge branch 'main' of https://github.com/jasg99/project		b2df582		
(jasg99 committed yesterday)				
attempt to get.js working		5c29b73		
(jasg99 committed yesterday)				
Commits on Jun 29, 2025				
Delete .vscode/launch.json		8afefbf		
(jasg99 authored 2 days ago)				
Merge branch 'main' of https://github.com/jasg99/project		b3d1ef		
(jasg99 committed 2 days ago)				
changing login/signup system		129ae86		
(jasg99 committed 2 days ago)				
Commits on Jun 20, 2025				
Create README.md		7ae2872		
(jasg99 authored 2 weeks ago)				
Commits on Jun 17, 2025				

-o- Commits on Jun 17, 2025
trying to save user login details and verify (jasg99 committed 2 weeks ago) 63c52be   
-o- Commits on Jun 16, 2025
saving passwords etc (jasg99 committed 2 weeks ago) c657d60   
-o- Commits on Jun 10, 2025
stylistic, fixes and results page 10/6 (jasg99 committed 3 weeks ago) bf03c00   
stylistic (jasg99 committed 3 weeks ago) c964116   
Calculation and results viewing (jasg99 committed 3 weeks ago) 0281e21   
calculation (jasg99 committed 3 weeks ago) ed9e6d3   
comments + planning next steps in development (jasg99 committed 3 weeks ago) 84c527b   
-o- Commits on May 30, 2025
financial planning and outline in comments on what to do next (jasg99 committed on May 30) a4644fb   
financial planning inputs and some stylistic features (jasg99 committed on May 30) 968ca6d   
functional button links (jasg99 committed on May 30) b35ds7a   
Signup button (jasg99 committed on May 30) 7aa260c   
link to planning page (jasg99 committed on May 30) a21f469   
-o- Commits on May 27, 2025
setup of login and signup page with form validation from 15/5-27/5 (jasg99 committed on May 27) 3b514c0   
form validation, login page, etc (jasg99 committed on May 27) b52a16f   

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

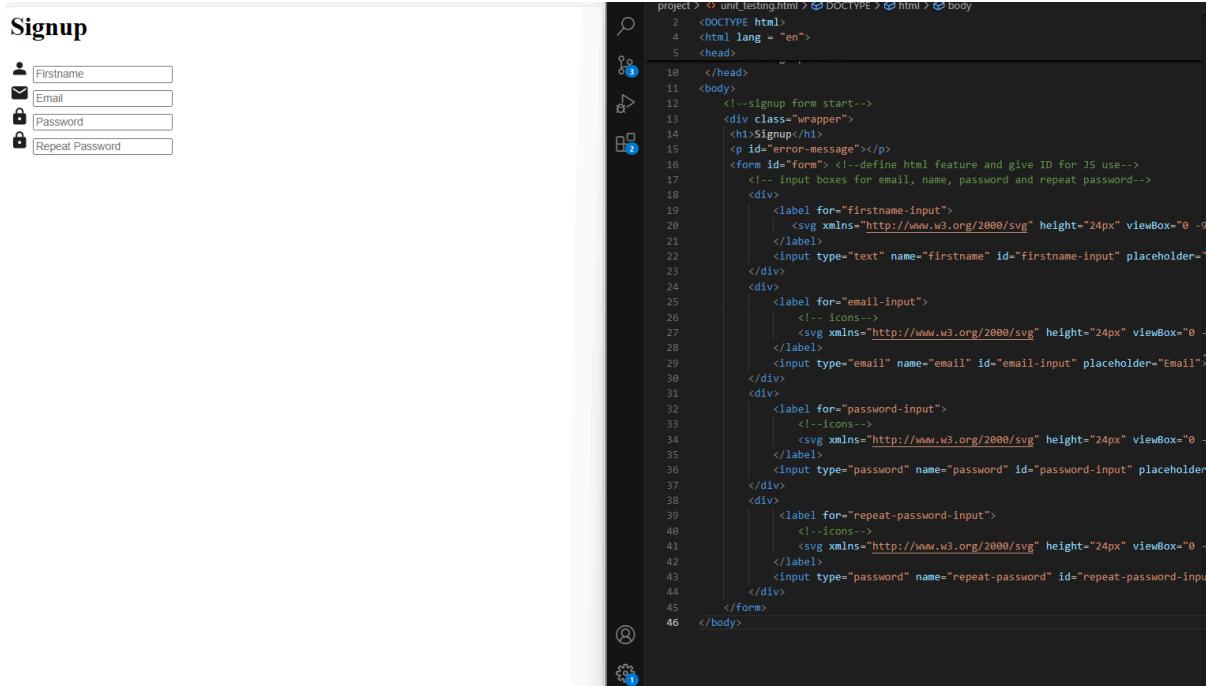
From this project, I advanced my knowledge of HTML, Javascript and CSS, allowing me to create a well-developed website. Additionally, I also learnt how to navigate Github, which I had

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

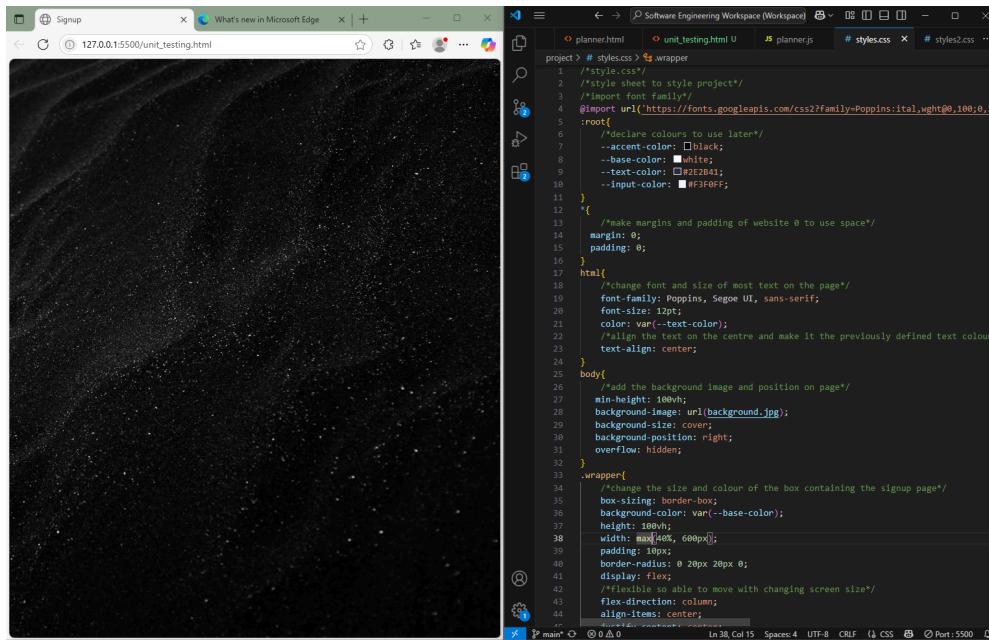
- Full Test Logs

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



```
project > unit_testing.html > DOCTYPE > html > body
1 <!DOCTYPE html>
2 <html lang = "en">
3 <head>
4   </head>
5 <body>
6   <!--signup form start-->
7   <div class="wrapper">
8     <h1>Signup</h1>
9     <p id="error-message"></p>
10    <form id="form"> <!--define html feature and give ID for JS use-->
11      <!-- input boxes for email, name, password and repeat password-->
12      <div>
13        <label for="firstname-input">
14          <input type="text" name="firstname" id="firstname-input" placeholder="First Name" />
15          <!-- icons-->
16          <svg xmlns="http://www.w3.org/2000/svg" height="24px" viewBox="0 0 24 24">
17            <path d="M12 2a10 10 0 1 1 20 0A10 10 0 0 1 2 2z M12 12a8 8 0 1 0 0 16A8 8 0 0 0 12 12z" />
18        </label>
19      </div>
20      <div>
21        <label for="email-input">
22          <input type="email" name="email" id="email-input" placeholder="Email" />
23          <!-- icons-->
24          <svg xmlns="http://www.w3.org/2000/svg" height="24px" viewBox="0 0 24 24">
25            <path d="M12 2a10 10 0 1 1 20 0A10 10 0 0 1 2 2z M12 12a8 8 0 1 0 0 16A8 8 0 0 0 12 12z" />
26        </label>
27      </div>
28      <div>
29        <label for="password-input">
30          <input type="password" name="password" id="password-input" placeholder="Password" />
31          <!-- icons-->
32          <svg xmlns="http://www.w3.org/2000/svg" height="24px" viewBox="0 0 24 24">
33            <path d="M12 2a10 10 0 1 1 20 0A10 10 0 0 1 2 2z M12 12a8 8 0 1 0 0 16A8 8 0 0 0 12 12z" />
34        </label>
35      </div>
36      <div>
37        <label for="repeat-password-input">
38          <input type="password" name="repeat-password" id="repeat-password-input" placeholder="Repeat Password" />
39          <!-- icons-->
40          <svg xmlns="http://www.w3.org/2000/svg" height="24px" viewBox="0 0 24 24">
41            <path d="M12 2a10 10 0 1 1 20 0A10 10 0 0 1 2 2z M12 12a8 8 0 1 0 0 16A8 8 0 0 0 12 12z" />
42        </label>
43      </div>
44    </form>
45  </div>
46 </body>
```

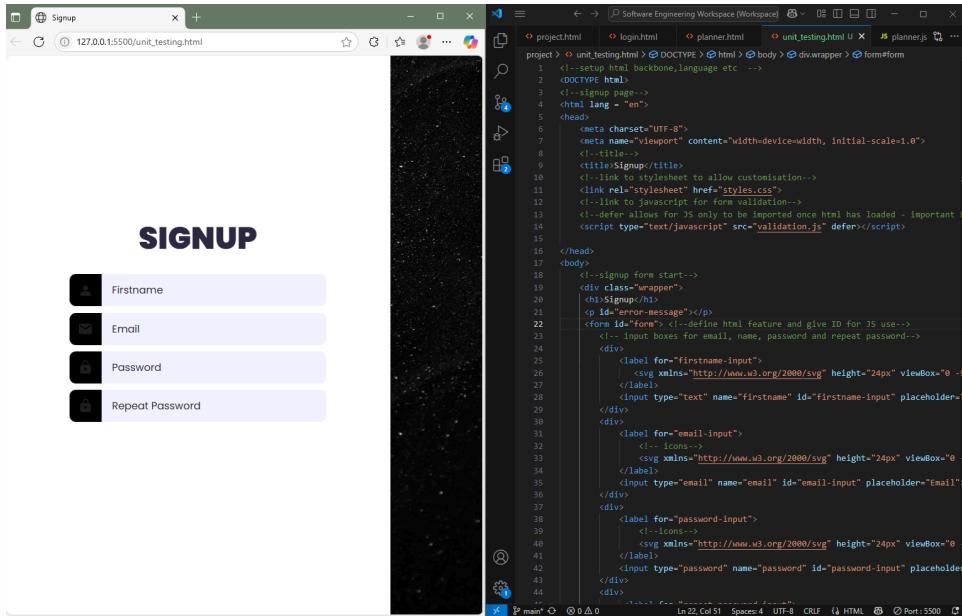
TC02 - Unit Test - CSS background and style test



The screenshot shows a Microsoft Edge browser window with a dark-themed website titled "Signup". The page has a background image of a landscape. A code editor on the right displays the CSS file "styles.css" with the following content:

```
project > # styles.css > # wrapper
1  /*style.css*/
2  /*style sheet to style project*/
3  /*font family*/
4  @import url('https://fonts.googleapis.com/css?family=Poppins:ital,wght@0,100;0,200');
5  root{
6    /*declare colours to use later*/
7    --accent-color: #black;
8    --base-color: #white;
9    --text-color: #2E2B41;
10   --input-color: #3F5FFF;
11 }
12 {
13   /*make margins and padding of website 0 to use space*/
14   margin: 0;
15   padding: 0;
16 }
17 html{
18   /*change font and size of most text on the page*/
19   font-family: Poppins, Segoe UI, sans-serif;
20   font-size: 12pt;
21   color: var(--text-color);
22   /*align the text on the centre and make it the previously defined text colour*/
23   text-align: center;
24 }
25 body{
26   /*add the background image and position on page*/
27   min-height: 100vh;
28   background-image: url(background.jpg);
29   background-size: cover;
30   background-position: right;
31   overflow: hidden;
32 }
33 .wrapper{
34   /*change the size and colour of the box containing the signup page*/
35   box-sizing: border-box;
36   background-color: var(--base-color);
37   height: 100vh;
38   width: max(40%, 600px);
39   padding: 10px;
40   border-radius: 0 20px 20px 0;
41   display: flex;
42   /*flexible able to move with changing screen size*/
43   flex-direction: column;
44   align-items: center;
```

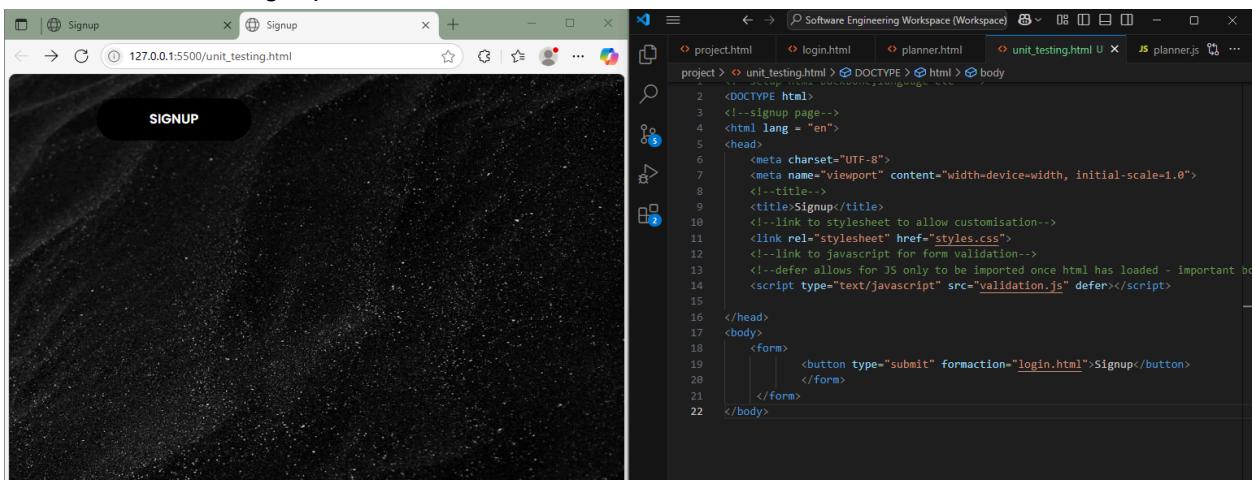
TC03 - Integration Test - Combine CSS and Signup form



The screenshot shows a Microsoft Edge browser window with a dark-themed "Signup" form. The form includes fields for Firstname, Email, Password, and Repeat Password, each with a small icon next to the input field. A code editor on the right displays the combined HTML and CSS code:

```
project > unit_testing.html > DOCTYPE > <!DOCTYPE html> > <html lang="en">
1   <!--setup html backbone,language etc -->
2   <DOCTYPE html>
3   <!--signup page-->
4   <html lang = "en">
5     <head>
6       <meta charset="UTF-8">
7       <meta name="viewport" content="width=device-width, initial-scale=1.0">
8       <title>-->
9       <title>Signup</title>
10      <!--link to stylesheet to allow customisation-->
11      <link rel="stylesheet" href="styles.css">
12      <!--link to JavaScript for form validation-->
13      <!--defer allows for JS only to be imported once html has loaded-- important b
14      <script type="text/javascript" src="validation.js" defer></script>
15
16    </head>
17    <body>
18      <!--signup form start-->
19      <div class="wrapper">
20        <h1>Signup</h1>
21        <div id="error-message"></p>
22        <form id="form" ><!--define html feature and give ID for JS use-->
23          <!-- Input boxes for email, name, password and repeat password-->
24          <div>
25            <label for="firstname-input">
26              <img alt="User icon" data-bbox="158 598 178 618" />
27              <input type="text" name="firstname" id="firstname-input" placeholder="Firstname" />
28            </label>
29            <input type="text" name="firstname" id="firstname-input" placeholder="Firstname" />
30          </div>
31          <div>
32            <label for="email-input">
33              <img alt="Email icon" data-bbox="158 628 178 648" />
34              <input type="email" name="email" id="email-input" placeholder="Email" />
35            </label>
36            <input type="email" name="email" id="email-input" placeholder="Email" />
37          </div>
38          <div>
39            <label for="password-input">
40              <img alt="Lock icon" data-bbox="158 658 178 678" />
41              <input type="password" name="password" id="password-input" placeholder="Password" />
42            </label>
43            <input type="password" name="password" id="password-input" placeholder="Password" />
44          </div>
45        </form>
46      </div>
47    </body>
48  </html>
```

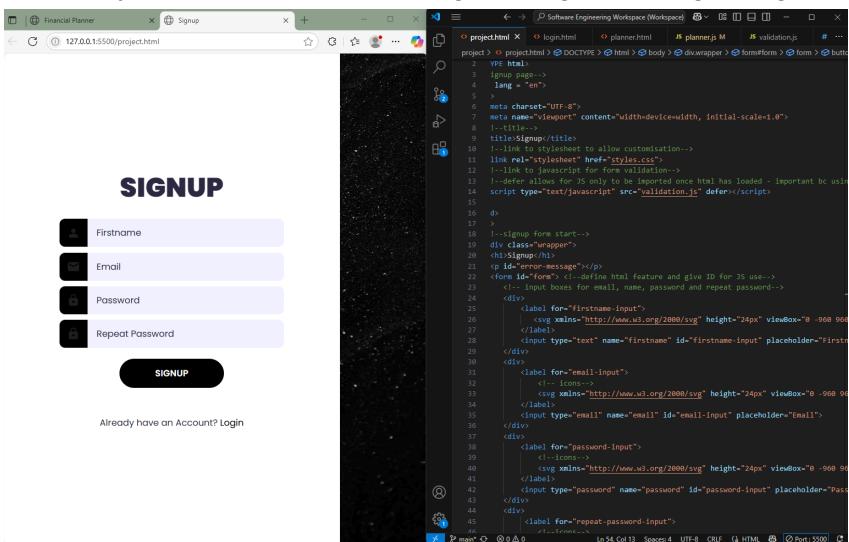
TC04 - Unit Test - Signup button



The screenshot shows a browser window titled "Signup" with the URL "127.0.0.1:5500/unit_testing.html". The page has a dark background and a central "SIGNUP" button. To the right is a code editor showing the HTML code for this page:

```
<!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 bc using validation.js-->
    <script type="text/javascript" src="validation.js" defer></script>
</head>
<body>
    <form>
        <button type="submit" formaction="login.html">Signup</button>
    </form>
</body>
```

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



The screenshot shows a browser window titled "Financial Planner" with the URL "127.0.0.1:5500/project.html". The page displays a "SIGNUP" form with four input fields: "Firstname", "Email", "Password", and "Repeat Password", each with a small icon to its left. Below the inputs is a "SIGNUP" button. At the bottom of the page, there is a link "Already have an Account? Login". To the right is a code editor showing the full HTML code for this page:

```
<!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 bc using validation.js-->
    <script type="text/javascript" src="validation.js" defer></script>
</head>
<body>
    <div class="sign-up">
        <h2>SIGNUP</h2>
        <form id="sign-up-form">
            <div>
                <label>Firstname</label>
                <input type="text" name="firstname" id="firstname-input" placeholder="First Name" />
            </div>
            <div>
                <label>Email</label>
                <input type="email" name="email" id="email-input" placeholder="Email" />
            </div>
            <div>
                <label>Password</label>
                <input type="password" name="password" id="password-input" placeholder="Password" />
            </div>
            <div>
                <label>Repeat Password</label>
                <input type="password" name="repeat-password" id="repeat-password-input" placeholder="Repeat Password" />
            </div>
            <div>
                <button type="submit" formaction="login.html">Signup</button>
            </div>
        </form>
        <small>Already have an Account? <a href="login.html">Login</a></small>
    </div>
</body>
```

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

<input type="text"/> jas
<input type="text"/> jas@g
<input type="password"/> (highlighted in red)
<input type="password"/> (highlighted in red)

SIGNUP

Already have an Account? [Login](#)

Firstname is required. Firstname should be alphanumeric . Email is required. Password is required. Password must have at least 8 characters. Password should be alphanumeric

<input type="text"/> Firstname
<input type="text"/> Email
<input type="password"/> Password
<input type="password"/> Repeat Password

SIGNUP

Already have an Account? [Login](#)

SIGNUP

Password does not match repeated password

<input type="text"/> jasmin
<input type="text"/> jas@g
<input type="password"/> (highlighted in red)
<input type="password"/> (highlighted in red)

SIGNUP

Already have an Account? [Login](#)

Firstname is required. Firstname should be alphanumeric . Email is required. Password is required. Password must have at least 8 characters. Password should be alphanumeric

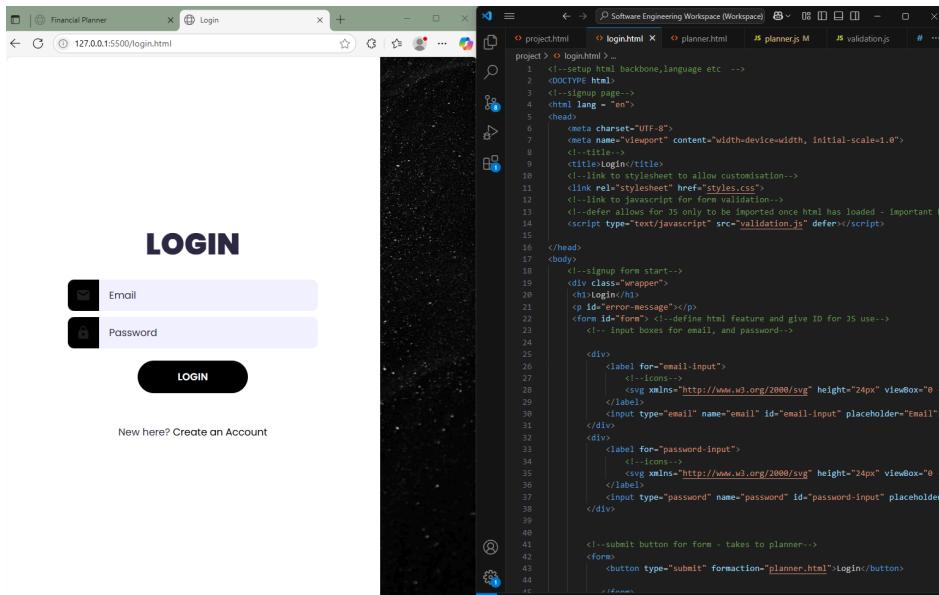
<input type="text"/> Jasmin
<input type="text"/> frnskfs (highlighted in red)
<input type="password"/> (highlighted in red)

SIGNUP

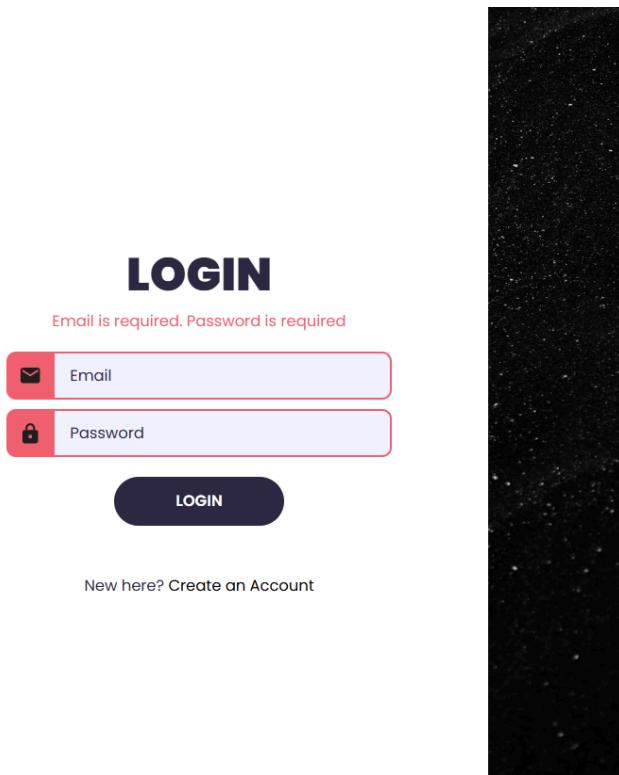
Already have an Account? [Login](#)

Please include an '@' in the email address. 'frnskfs' is missing an '@'.

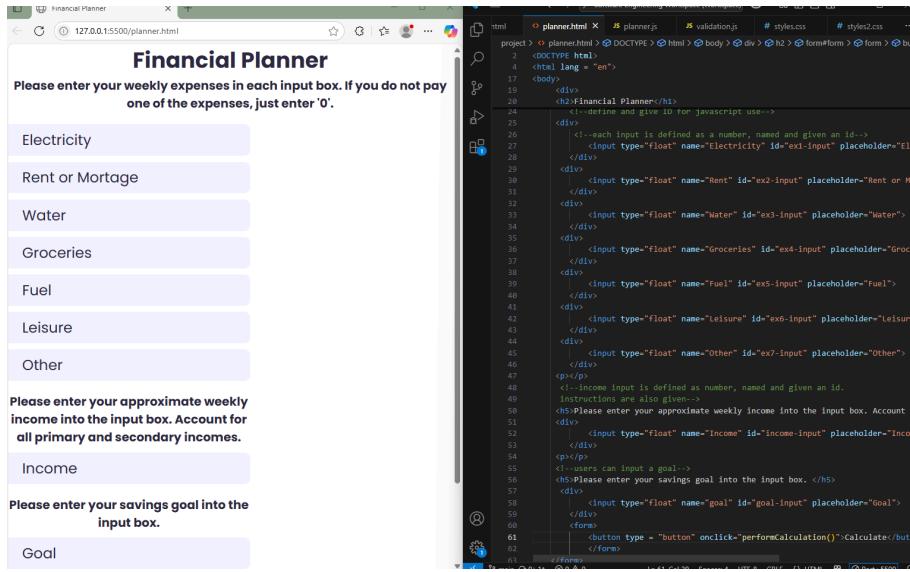
TC07 - System Test, Integration Test - Complete Login Page



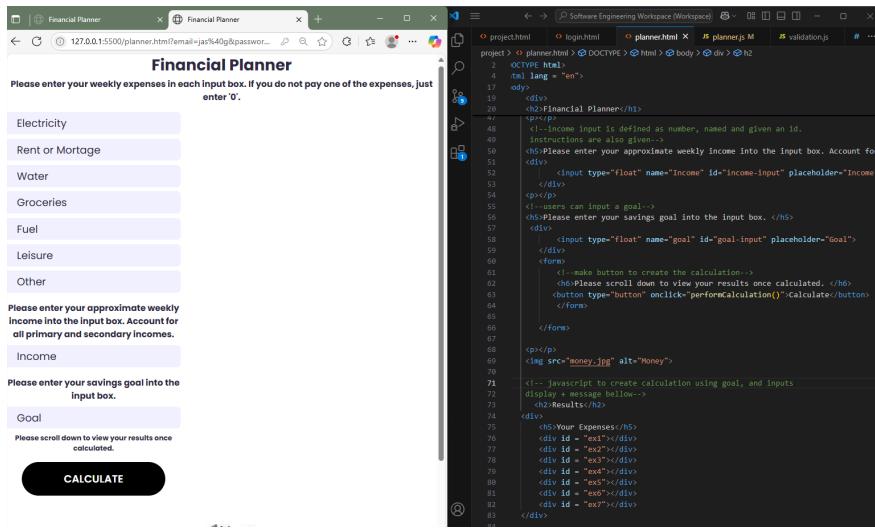
TC08 - Acceptance Test - Testing functionality and clear error messages if incorrect inputs.



TC09 - Unit test - Testing Input Boxes and Page Setup



TC10 - Testing Calculate Button and Style



TC11 - Unit Test - Testing initial Javascript Calculation

```

10
15
4
0
3
25

Please enter your approximate weekly income into the input box. Account for all primary and secondary incomes.

327

Please enter your savings goal into the input box.

1000

Calculate

You have:  

26.31578947368421 Days  

until you will reach your goal!

```

```

1 //javascript planner
2 function performCalculation() {
3
4     // define each input number
5     const ex1 = parseFloat(document.getElementById("ex1-input").value);
6     const ex2 = parseFloat(document.getElementById("ex2-input").value);
7     const ex3 = parseFloat(document.getElementById("ex3-input").value);
8     const ex4 = parseFloat(document.getElementById("ex4-input").value);
9     const ex5 = parseFloat(document.getElementById("ex5-input").value);
10    const ex6 = parseFloat(document.getElementById("ex6-input").value);
11    const ex7 = parseFloat(document.getElementById("ex7-input").value);
12
13    //define income input
14    const inc = parseFloat(document.getElementById("income-input").value);
15
16    const goal = parseFloat(document.getElementById("goal-input").value);
17
18    //days in a week
19    const dayweek = 7;
20
21    //sum of all costs added together
22    const sum = ex1 + ex2 + ex3 + ex4 + ex5 + ex6 + ex7;
23    // income - sum(costs) = savings per week
24    const pweek = inc - sum;
25    //savings per week divided by 7 = savings per day
26    const pday = pweek / dayweek;
27    //goal amount divided by savings per day = no. days till saved
28    const numdays = goal / pday;
29
30    document.getElementById("result").innerText = numdays + " Days";

```

TC12 - System Test - Testing Correct Inputs

```

calculated.
Calculate

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!

```

```

1 action perform Converts a string to a floating-point number.
2
3     const ex5 = /*param string — A string that contains a floating-point number.
4     const ex1 = /*param string — A string that contains a floating-point number.
5     const ex2 = /*param string — A string that contains a floating-point number.
6     const ex3 = /*param string — A string that contains a floating-point number.
7     const ex4 = /*param string — A string that contains a floating-point number.
8     const ex6 = /*param string — A string that contains a floating-point number.
9     const ex7 = /*param string — A string that contains a floating-point number.
10    const inc = /*param string — A string that contains a floating-point number.
11    const goal = /*param string — A string that contains a floating-point number.
12
13    //days in a week
14    const dayweek = 7;
15
16    //sum of all costs added together
17    const sum = ex1 + ex2 + ex3 + ex4 + ex5 + ex6 + ex7;
18    // income - sum(costs) = savings per week
19    const pweek = inc - sum;
20    //savings per week divided by 7 = savings per day
21    const pday = pweek / dayweek;
22    //goal amount divided by savings per day = no. days till saved
23    const numdays = goal / pday;
24
25    //if incorrect/raises error, display inputs and results
26    if (numdays > 0) {
27        document.getElementById("ex1").innerText = "Electricity: $" + ex1;
28        document.getElementById("ex2").innerText = "Rent: $" + ex2;
29        document.getElementById("ex3").innerText = "Water: $" + ex3;
30        document.getElementById("ex4").innerText = "Groceries: $" + ex4;
31        document.getElementById("ex5").innerText = "Fuel: $" + ex5;
32        document.getElementById("ex6").innerText = "Leisure: $" + ex6;
33        document.getElementById("ex7").innerText = "Other: $" + ex7;
34        document.getElementById("inc").innerText = "$" + inc;
35        document.getElementById("goal").innerText = "$" + goal;
36
37        document.getElementById("result").innerText = "You have: " + numdays + " Days until you reach your goal!";
38        document.getElementById("message").innerText = "Keep saving! You've got this!";
39    } else {
40        document.getElementById("result").innerText = "...";
41        document.getElementById("message").innerText = "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.";
42    }
43
44    //if incorrect/raises error, display inputs and results
45    if (numdays < 0) {
46        document.getElementById("ex1").innerText = "Electricity: $" + ex1;
47        document.getElementById("ex2").innerText = "Rent: $" + ex2;
48        document.getElementById("ex3").innerText = "Water: $" + ex3;
49        document.getElementById("ex4").innerText = "Groceries: $" + ex4;
50        document.getElementById("ex5").innerText = "Fuel: $" + ex5;
51        document.getElementById("ex6").innerText = "Leisure: $" + ex6;
52        document.getElementById("ex7").innerText = "Other: $" + ex7;
53        document.getElementById("inc").innerText = "$" + inc;
54        document.getElementById("goal").innerText = "$" + goal;
55
56        document.getElementById("result").innerText = "You have: " + numdays + " Days until you reach your goal!";
57        document.getElementById("message").innerText = "Keep saving! You've got this!";
58    }
59
60    //if incorrect/raises error, display inputs and results
61    if (numdays === 0) {
62        document.getElementById("ex1").innerText = "...";
63        document.getElementById("message").innerText = "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.";
64    }
65
66}

```

TC14 - System Test - Testing Incorrect Inputs

```

calculated.
Calculate

Results  

Your Expenses  

Electricity: $20  

Rent: $15  

Water: $25  

Groceries: $12  

Fuel: $0  

Leisure: $30  

Other: $6  

Your Income  

$450  

Your Goal  

$700  

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.

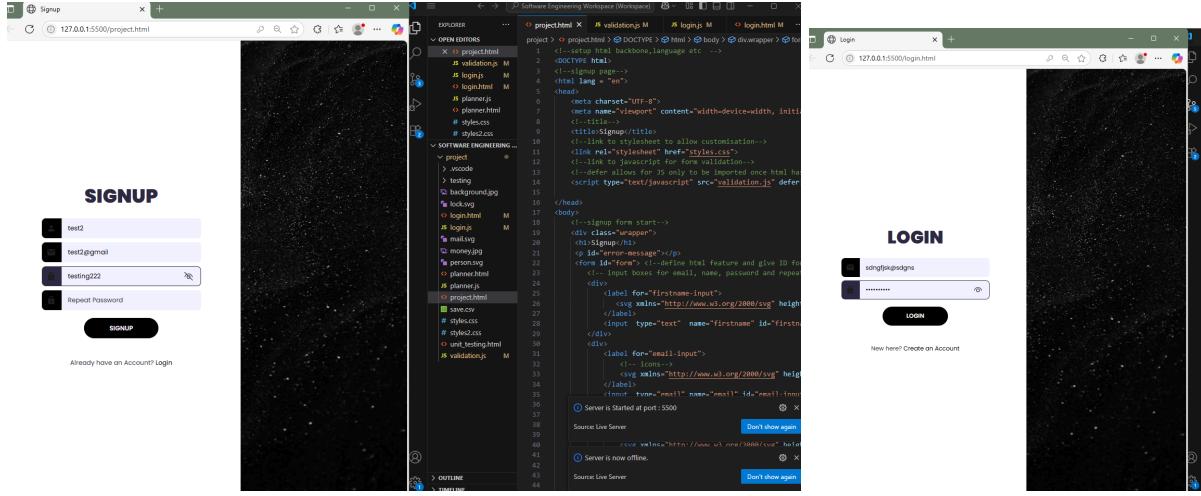
```

```

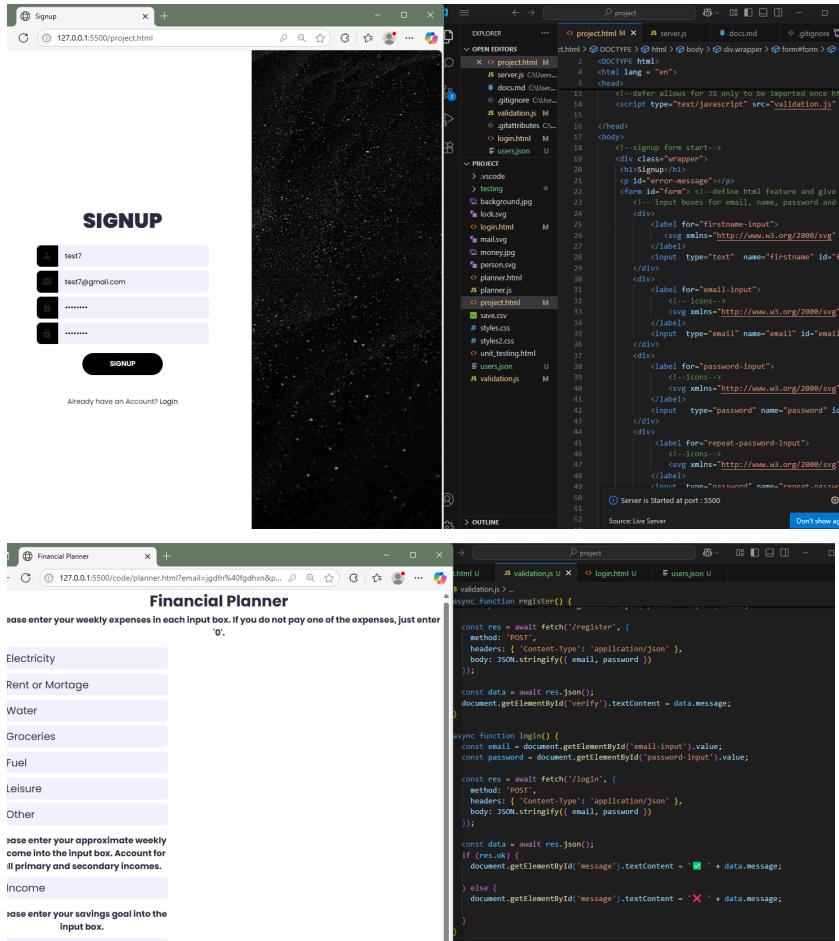
1 action perform Converts a string to a floating-point number.
2
3     const ex5 = /*param string — A string that contains a floating-point number.
4     const ex1 = /*param string — A string that contains a floating-point number.
5     const ex2 = /*param string — A string that contains a floating-point number.
6     const ex3 = /*param string — A string that contains a floating-point number.
7     const ex4 = /*param string — A string that contains a floating-point number.
8     const ex6 = /*param string — A string that contains a floating-point number.
9     const ex7 = /*param string — A string that contains a floating-point number.
10    const inc = /*param string — A string that contains a floating-point number.
11    const goal = /*param string — A string that contains a floating-point number.
12
13    //days in a week
14    const dayweek = 7;
15
16    //sum of all costs added together
17    const sum = ex1 + ex2 + ex3 + ex4 + ex5 + ex6 + ex7;
18    // income - sum(costs) = savings per week
19    const pweek = inc - sum;
20    //savings per week divided by 7 = savings per day
21    const pday = pweek / dayweek;
22    //goal amount divided by savings per day = no. days till saved
23    const numdays = goal / pday;
24
25    //if incorrect/raises error, display inputs and results
26    if (numdays > 0) {
27        document.getElementById("ex1").innerText = "Electricity: $" + ex1;
28        document.getElementById("ex2").innerText = "Rent: $" + ex2;
29        document.getElementById("ex3").innerText = "Water: $" + ex3;
30        document.getElementById("ex4").innerText = "Groceries: $" + ex4;
31        document.getElementById("ex5").innerText = "Fuel: $" + ex5;
32        document.getElementById("ex6").innerText = "Leisure: $" + ex6;
33        document.getElementById("ex7").innerText = "Other: $" + ex7;
34        document.getElementById("inc").innerText = "$" + inc;
35        document.getElementById("goal").innerText = "$" + goal;
36
37        document.getElementById("result").innerText = "You have: " + numdays + " Days until you reach your goal!";
38        document.getElementById("message").innerText = "Keep saving! You've got this!";
39    } else {
40        document.getElementById("result").innerText = "...";
41        document.getElementById("message").innerText = "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.";
42    }
43
44    //if incorrect/raises error, display inputs and results
45    if (numdays < 0) {
46        document.getElementById("ex1").innerText = "Electricity: $" + ex1;
47        document.getElementById("ex2").innerText = "Rent: $" + ex2;
48        document.getElementById("ex3").innerText = "Water: $" + ex3;
49        document.getElementById("ex4").innerText = "Groceries: $" + ex4;
50        document.getElementById("ex5").innerText = "Fuel: $" + ex5;
51        document.getElementById("ex6").innerText = "Leisure: $" + ex6;
52        document.getElementById("ex7").innerText = "Other: $" + ex7;
53        document.getElementById("inc").innerText = "$" + inc;
54        document.getElementById("goal").innerText = "$" + goal;
55
56        document.getElementById("result").innerText = "You have: " + numdays + " Days until you reach your goal!";
57        document.getElementById("message").innerText = "Keep saving! You've got this!";
58    }
59
60    //if incorrect/raises error, display inputs and results
61    if (numdays === 0) {
62        document.getElementById("ex1").innerText = "...";
63        document.getElementById("message").innerText = "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.";
64    }
65
66}

```

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 definition of incorrect
    let errors = []
    //if statements allow this form to be used for both
    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 = []
    //error listing function - goes through conditions, pushes error messages if errors
    //if user doesn't enter anything then null
    if(firstname === '' || firstname == null){
        //error message pop up
        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){
        //error message pop up
        errors.push('Firstname should be alphanumeric')
        firstname_input.parentElement.classList.add('incorrect')
    }
    if(email === '' || email == null){
        //error message pop up
        errors.push('Email is required')
        email_input.parentElement.classList.add('incorrect') //adds incorrect feature for CSS
    }
    if(password === '' || password == null){
        //error message pop up
        errors.push('Password is required')
        password_input.parentElement.classList.add('incorrect') //adds incorrect feature for CSS
    }
    //check password length
    if(password.length < 8){
        //error message pop up
        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){
        //error message pop up
        errors.push('Password should be alphanumeric')
        password_input.parentElement.classList.add('incorrect')
    }
    if(password != repeatPassword){
        //error message pop up
        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?.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 till 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 = "Gas: $" + 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 make sure all inputs are valid numbers.";
  }
}
```