

|  |  |
| --- | --- |
| Instrument of Assessment Internally Verified by: | Date: |

|  |  |  |
| --- | --- | --- |
| Assessor: | Date: | |
| IV Sampled by: | Date: | |
|  | Result |  |

# CQA1.1.9: Assessment Material Cover Sheet

Unit Number and Title: H173 34 Developing Software: Introduction

Verification Group: 357

Outcome(s): ALL Version: C

Student Name:

## Where the assessment has not taken place under controlled conditions the learner should sign and date to indicate agreement with the statement below.

“I declare that this submission is entirely my own work, and any content by other authors has been clearly acknowledged and referenced. I understand that disciplinary action will be taken by the college if this is not the case.”

## Student Signature: Artur Kyshner Date: 7-06-2023

# Information

1. You will have until date given by your lecturer to complete this assessment.
2. This is an open book assessment
3. This assessment will be graded either Pass or Fail
4. To achieve a Pass you must complete all tasks correctly

If your work is graded as “Fail” you will be given a re-sit. If your work is graded as “Fail” on your second attempt, you may be given a third and final assessment **but only if agreed by the members of your staff course team**. For detailed information on the college’s assessment policy please refer to the college intranet policies area or ask your Lecturer.

If you are not satisfied with the grade awarded by your Lecturer, you should speak to your lecturer in the first instance. If you are not satisfied with this response then the Lecturer or Curriculum & Quality Leader will refer you to the college’s appeals procedure.

Impartial advice about the college appeals procedure is available from your student advisor.

**Outcomes covered**

H173 34 Developing Software: Introduction

1. Implement and test code to carry out tasks following a given design
2. Prepare technical documentation in line with good practice

## 

## Assessment Instructions

This assessment is open-book and you are encouraged to refer to study notes and previously completed code.

1. Read the complete assessment before starting
2. Completed the detailed design table with appropriate names for all variables, arrays and functions on page 8
3. Write programs which provide a solution to the set tasks, making sure you demonstrate all the coding evidence listed on the following page (p4)
4. Once the program(s) are complete and working, thoroughly test using the testing document provided and complete the test log on pages 9 and 10
5. Complete the learner checklist on page 4
6. Complete the short report on page 11
7. Sign the declaration on the front page to confirm that the completed submission is all your own work

### Upload this completed document and a zipped folder containing your programs to Moodle

**Coding Checklist**

The two programs together must demonstrate all of the following evidence requirements

|  |  |  |
| --- | --- | --- |
| **Evidence** | **Learner Check**  Add in program and/or line numbers where each is demonstrated | **Assessor Check and comments** |
| Appropriate naming conventions for variables and functions | 36. startQuiz  25. genRandomAvatar |  |
| Code indentation to aid readability | Line 84 -114 (quiz)  Line 24 – 38 (avatar) |  |
| Internal comments to aid maintainability | Line 107 (quiz)  Line 24 (avatar) |  |
| Appropriate use of at least 2 variables of different data types | Avatar  7. nameElement  14. username |  |
| Selection (if or case) | Line 117- 141 (quize) |  |
| Iteration/Loop (for or while) | Validation.js file  Task 1 |  |
| User defined function that includes parameter passing | Line 87 chechAnswer() |  |
| Inbuilt functions | Line 89 use  userAnswer.trim() |  |
| Operators   * arithmetic ( + - / \*) * comparison (==, !=, >,< etc.) | validateInput()  in validation file |  |

**Assessment Tasks**

Read the full assessment thoroughly before starting.

You will be provided with a template HTML and CSS file that can be used

Write well documented programs to carry out the following tasks:

**Task 1**

Design a general knowledge quiz for college students with a minimum of **3** questions.

The questions should be appropriate and simple with one correct one-word answer.

*For example: What is the common name for a zucchini? Answer - courgette*

1. When the index quiz page launches collect the players name and display it in the **#ID name** provided within the quiz front panel.
2. The quiz should then start by clicking the ***‘Start Quiz’*** button which displays the first question
3. All user text answers should be converted to lowercase and number answers should be parsed as numbers
4. A players score should be tracked throughout the quiz as well as guess attempts taken for each question.   
   The user will be allowed **3** attempts at each question and if they do not supply the correct answer they should be forced to proceed to the next question
5. Appropriate feedback should be displayed for a correct or incorrect answer or too many attempts made
6. Validation should be implemented where necessary ***i.e.*** *no empty fields allowed or numbers outside a given range should be validated*

|  |
| --- |
| ***Note: The following function must demonstrate parameter passing*** |

1. After completing the questions the players score should be tested using a custom function and an appropriate message displayed with their name and score – an alert box can be used or the results can be sent to browser

|  |
| --- |
| *Note: You may alter the CSS to style your programs but it is not a requirement* |

**Task 2**

Players are asked to select an **Avatar** as they are setting up their profile page:

You have been supplied with the ‘*choose an avatar’* stage of the set up.

1. When the page loads collect a username the player would like displayed above their avatar – convert the username to UPPER case and display in the **<span>** tag with an **#ID** = **name**
2. Create an array and populate it with the avatar images supplied – when the player clicks the **’GENERATE AVATAR**’ button a random *avatar/image* should be displayed to the page in the **<div>** tag with an **#ID** = **avatar**

*Optional*

1. There is a button provided with text ‘**SET AVATAR’** that when clicked should set the chosen avatar to the **<div>** tag with the **#ID = setavatar**

|  |
| --- |
| ***Note: ALL code must be commented and variables, functions & arrays named appropriately*** |

**Appendices:**

1. Detailed Design – to be completed

2. Test Log – to be completed

3. Short report to be completed

## 1. Detailed Design

### Complete this table. You must document ALL (*Variables, Arrays and Functions*) with their type, scope and a description of what they do in the program – REMOVE THE RED EXAMPLES BEFORE SUBMITTING

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Scope** | **Description** |
|  |  |  |  |
| totalScore | integer | Global name variable | Variable to store the total score of a player |
| showQuestion() | function | global | Function for displaying a question |
| minRange  maxRange | integer | Global name variable | Set min and max range use number |
| validationInput | Function | Global function | Validation input(inputValue, minRange, maxRange) |

## 2. Test log Sheets

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **QUIZ LOG SHEET** | |  |  |  |  |
| **Subroutine name** | **Test case** | **Test data** | **Expected result** | **Actual result** | **Comments** |
| ON PAGE LOAD | Enter a user name in the format required | Your name | Logon name converted to uppercase and displayed in the front of the panel interface | DONE |  |
| startQuiz() | Click the START QUIZ button |  | App should launch and quiz will begin with first question | DONE |  |
| checkAnswer() |  | Correct answer to Q1 – Q3 | User should see an alert box with an appropriate message - Exit the loop and taken to the next question – score variable updated | DONE | this function does validation on input data |
| checkAnswer() |  | Incorrect answer to Q1 – Q3 | User should see an alert box with an appropriate message - Remain the loop and guess variable updated | DONE | checks for the correctness of the answer and starts the counter if the answer is not correct |
| checkAnswer() |  | Empty string  (exceptional) | Returns user to enter their answer again | DONE | and gives the next attempt if they are left if not, then proceeds to the next question |
| checkScore()  I don't have such function but inserts  I got  endQuiz() | Value passed from the users score | Score variable | Alert box displaying the corresponding message dependent on the value passed | DONE | in this variable, there is a part of the code that displays the points scored and then writes them to the sheet and all this is done using DOM |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **AVATAR LOG SHEET** | |  |  |  |  |
| **Subroutine name** | **Test case** | **Test data** | **Expected result** | **Actual result** | **Comments** |
| ON PAGE LOAD | Enter a user name in the format required | Your name | collect name and convert to uppercase then display in the front of the panel interface | Done | Requesting a username via promt and save in “username” |
| selectAvatar()  only I have this function called differently  genRandomAvatar() | Click the GENERATE AVATAR button |  | random image should be displayed each time the button is clicked and displayed in the #id named avatar | DONE |  |
| setAvatar() | Click the SET AVATAR button |  | The chosen avatar should be set to the tag with an #id setavatar | No pass | I couldn't set my avatar. I tried to implement it through a list of avatars, but the wrong one is set on which you click. |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

1. **Short Report**

|  |
| --- |
| Write a short report outlining the reasons (pros and/or cons) of using JavaScript for this task.  JavaScript is a widely used programming language that offers a number of advantages and reasons for using it in similar tasks:  1. interactivity: JavaScript allows you to create interactive web pages where users can interact with elements and see instantaneous changes without reloading the page. This is particularly useful when you want to select a particular element or adjust settings.  2. Dynamic update: JavaScript allows you to update page content dynamically without having to reload the page. For example, text, images or element styles can be changed depending on the user's choices.  3. Data validation: JavaScript provides the ability to verify and validate the data entered by the user before it is sent to the server. This helps to prevent errors and improve data quality.  4. event handling: JavaScript makes it easy to manage events such as clicks, hovering, form submissions, and other user actions. This allows you to react to events and perform appropriate actions programmatically.  5. DOM manipulation: JavaScript provides access to the Document Object Model (DOM), which allows you to programmatically create, modify, and delete elements on a page. This is useful for dynamically creating and changing page content to suit the task.  Arguments AGAINST using JavaScript in such tasks may include:  1. Client-side dependency: JavaScript runs on the client side (in the browser), so it depends on the user side to support and enable JavaScript. If JavaScript is disabled or unsupported, functionality may not be available.  2) Security: JavaScript may be used to execute malicious code, so measures should be taken to ensure security and protect against possible attacks.  3. alternative approaches: Depending on the task, there may be alternative programming languages or tools that may be more suitable or convenient to perform the task without the need to use JavaScript.  In general, using JavaScript to perform such tasks has many |