University of Hertfordshire UH

School of Computer of Science

ASSIGNMENT BRIEFING SHEET (2018/19 Academic Year) – ANONYMOUS MARKING

Assignment Title	Assignment 1: Interactive Painting Program	**Submission Date**	5th November 2018
Module Title	Programming	**Module Code**	4COM1037
Tutor	Watkins, M., Marczyk, O., Morris, P., Koay, K. & Hunt, S.	**GROUP or INDIVIDUAL Assignment**	Individual

FOR INDIVIDUAL ASSIGNMENTS – *STUDENT TO COMPLETE*

By completing **BOX A** below, I certify that the submitted work is entirely mine and that any material derived or quoted from the published or unpublished work of other persons has been duly acknowledged. **[ref. UPR AS12, section 7 and UPR AS14 (Appendix III)].)].** I also certify, that any work with human participants has been carried out under an approved ethics protocol in accordance with UPR RE01.
*Please **ONLY** provide your ID (srn) number as this assignment will be anonymously marked*

BOX A

Student ID Number (SRN)

FOR GROUP ASSIGNMENTS - *STUDENTS TO COMPLETE*

Group Name/Number (if allocated by module team)

By completing **BOX B** below, we certify that the submission is entirely ours and that any material derived or quoted from the published or unpublished work of other persons has been duly acknowledged. **[ref. UPR AS/C/6.1, section 7 and UPR AS/C/5 (Appendix III)].)].** I also certify, that any work with human participants has been carried out under an approved ethics protocol in accordance with UPR RE01.
*Please **ONLY** provide your ID (srn) number as this assignment will be anonymously marked and actual time spent on the assignment. By **BOX B** below you certify that this work represents equal contributions from all team members. If this is not the case, the module leader **<u>must</u>** be informed before submission.*

BOX B

Student ID Number (SRN)	**Actual Time Spent by each Student (hours)**

This sheet must be submitted with the assignment, and either BOX A or B filled in.
LATE SUBMISSION WILL ATTRACT A STANDARD LATENESS PENALTY.

1. For undergraduate modules, a score of 40% or above represents a pass mark.
2. For postgraduate modules, a score of 50% or above represents a pass mark.
3. For work submitted up to 5 working days late marked is capped to a bare pass (40% for undergraduate and 50% for postgraduate).
4. For work submitted more than 5 working days a mark of zero will be awarded for the assignment.

Regulations governing assessment offences including Plagiarism and Collusion are available from:
http://sitem.herts.ac.uk/secreg/upr/pdf/AS14-Apx3-Assessment%20Offences-v10.0.pdf

School of Computer of Science

ASSIGNMENT BRIEFING SHEET (2018/19 Academic Year) – ANONYMOUS MARKING

THE ASSIGNMENT TASK:

Use the “cw1_studentid.html” assignment template (available via the module’s pages on Studynet or via the module’s GitHub repository) as a starting point on which to design & further develop a working web-based painting style program. Please note, although you are free to improve and/or add to the aesthetic, layout, functionality (e.g. the visual look, adding buttons, images, etc.) of the template failure to use the template or changing pre-set HTML element ID’s or existing JavaScript function or variable names will result in a penalty of 50% being applied to your mark for this assignment.

Your submitted program will be tested using a standard version of Google’s Chrome browser (as installed in the Computer Science labs) and will be reviewed using a standard text editor (such as Notepad ++). Although your program is expected to work without errors and provide the requested user-based features and functions - marks will also be awarded based on your work’s overall quality, attention to detail and technical depth. If your program contains errors and/or is not easily testable marks will be deducted (see section 6, “Solution Failure Penalty” under the “Marks Awarded” part of this document for more details).

You are required to design and develop an interactive painting program that will run as a single, standard HTML web page running in a Chrome browser. Your painting program must not use any third-party libraries (such as jQuery, etc.) or plugins. Your submission must be entirely self-contained (consist of only 1 HTML document) and incorporate ALL the used and needed programming code as well as any supportive and/or needed HTML or CSS elements.

Your program is required to allow the user to (via the mouse and in freehand) paint pictures using multiple colours, multiple brush sizes and be supported with options to clear the current picture, set the background colour and erase previously painted areas. Please review the 6 sections outlined in the Marks Awarded part of this document for a full breakdown of the features, functions and programming requirements as well as the potential penalties associated with any incorrect or constrained aspects of this assignment.

General Behaviours:

Your program should render a suitable painting canvas (e.g. area size: 800px x 800px) within which the user can “paint” colours. When the mouse is moved over the painting area a suitable stylus is used to replace the mouse pointer; the stylus will track any mouse movements made over the canvas and so will indicate where the painting (or erasing) will occur.

The user is able to select a “paint” tool or an “erase” tool (these can be buttons or icons). If the “paint” tool is selected, when the user presses (holds) the left mouse button the system will begin painting using the currently selected colour. If the “erase” tool is selected, the system will perform a similar function to that of “paint” but will erase any previously painted marks (in essence, this could simply be a repaint of the area using the canvas background colour – technically this is more like ‘overpainting’ rather than erasing but the effect should be identical). Regardless of the selected tool – while the left mouse button is depressed the painting stylus will track all mouse movements while on the canvas; e.g. as the user moves the mouse the stylus and painting effect will track it. Once the left mouse button is released the pain or erase effect should cease.

A Word of Advice:

Although this is likely a challenging assignment - as you consider solutions you will likely begin to see ways to start to reduce some of its potential complexities by using functions. Functions are reusable, named blocks of code (a block of code is simply a series of instructions that will be executed) that can be given information (data) to act on, use and/or changed. Functions can also return data; this data can be the same information it was given or be an updated value or completely different bit of information - you (the programmer) decides based on their needs or the solution’s requirements). Data returned from a function can be stored (using variables) and/or be passed back to the same or a different function so that a series of actions can be performed, or the data can be changed, modified, further acted upon or calculated; functions are incredibly powerful programming constructs and are expected to be evident, well organised and well commented in your program.

A Reminder About Solving Programming Problems:

- Consider and try to define the problem in simple terms.
- Review, think, test and seek to improve your understanding of the problem and any possible solutions.
- Create a (as simple as possible) flowchart-based solution.
- Test and refine the flowchart until you are happy that your solution logically works; remember, if you don't know how to solve the problem you can't solve the problem.
- Decide the best approach to build the solution in code (e.g. which bits can be done and tested first?).
- **Don't try to build the entire solution in one instance** – break it up in to bits first!!
- Focus on the easy to finish bits, for the more challenging bits: research and create small prototypes to test ideas and improve your understanding on possible solutions.
- Once you know how to solve a problem - update your program with the new capability or feature.
- Build and test each bit until you are happy it is DONE and works (a.k.a. testing).
- Move onto the next bit and repeat.
- Update and maintain your change and test logs as you progress.

In Addition:

- Save and test your code each time you make a change.
- Keep older copies of your code by renaming the source file with a version number (e.g. cw1_07654321_v1.html).
- Use old copies or prototype attempts as a source of reference (build your own library of simple, small working examples).

Higher-Level Enhanced Challenge; this is not a requirement:
Before beginning these higher-level enhancements, you must have fully completed your program as per the requirements described in the Marks Awarded section: 1, 2, 3 and 4; failure to address these requirements will result in your failing this assignment - please note: you do not need to complete this higher-level challenge to pass this assignment.

Okay, you've confirmed you have completed ALL the requirements defined in section: 1, 2, 3 & 4 and found them too easy and have a significant amount of time left before the assignment is due - meaning you would like to take this opportunity to truly test and demonstrate your programming prowess! Please note, these enhancements will present you with considerable technical, logical and design challenges - part of this challenge is that you do not introduce new errors or bugs into your previously working code or in any way impede or prevent your original program (the program that is currently, fully working) from being run and tested by a marker; don't forget - failure to successfully program the requirements will significantly impact your mark for this assignment. Before beginning any of these challenges you are strongly advised to make sure you have successfully created backup copies (more than one) of your currently, fully working program. This will ensure that (in a worse-case scenario) you are still able to submit a complete program that addresses all requirements defined in section: 1, 2, 3 & 4.

Higher-Level Enhancements *(please ensure you have read the Higher-Level Enhanced Challenge section first)*:

1. *Add an "spray" feature: once activated, this feature will seem to spray colour onto the canvas instead of painting with using a solid rectangle or circle etc.*
2. *Add an "undo" feature: once activated, this feature will restore the canvas to the previous state by removing the users last action; e.g. the user has mistakenly painted a yellow line across their work, after "clicking" undo the yellow line is removed and the original colour pattern is fully restored.*

University of Hertfordshire UH

MODULE LEARNING OUTCOMES ASSESSED BY THIS ASSIGNMENT:

No.	Learning Outcome	Assessed
1	Sufficient features of a high-level programming language to develop solutions to simple programming problems	✓
2	The concepts of data declaration and operations, control flow (sequence, selection, iteration, subroutine call) and modularization	✓
3	The terminology used in describing programs and programming.	✓
4	Design and implement solutions to simple programming problems in a given programming language	✓
5	Execute, test and de-bug programs	✓
6	Document programs to an agreed standard	✓

SUBMISSION REQUIREMENTS

This is assignment is to be submitted and marked anonymously. Students should ONLY use their student ID number to identify themselves on their work. Work submitted via StudyNet for anonymous marking will automatically have an anonymity number allocated to it.

Please program, test and submit a SINGLE HTML document that ensures the following:

- The name of the HTML document uses the convention: "cw1_**studentid**.html", where "**studentid**" is replaced with your actual student ID.

  e.g.
  cw1_07654321.html
  cw1_01092910.html

- Your single HTML document must use this assignment's coursework template; the template (cw1_studentid.html) is available for download via the Studynet module or the module's GitHub repository. Failure to use the template or maintain any pre-set/existing HTML elements and/or JavaScript variables & function names will result in a 50% penalty being applied to your mark.

- Your single HTML document must include clear evidence for each of the requirements defined in the Marks Awarded section: 1, 2, 3 & 4 of this document; failure to complete ALL the requirements will reduce your mark.

You will be required and expected to submit **a single, fully checked and tested HTML document that contains ALL necessary HTML elements, JavaScript program code and if needed CSS styling** via the assignment submission facility in Studynet's Programming (4COM1037) module before the assignment's published deadline.

University of Hertfordshire UH

FEEDBACK FROM THIS ASSIGNMENT

Each student will receive feedback in the form of an individual mark (out of 100) supported with a range of comments via Studynet; marks will be broken down against each of the overall sections outlined below.

MARKS AWARDED FOR:

Section	Detail	Mark
1. Non-Use of Template You are required to use, expand and fully develop your program based on the template provided. You are strongly advised to use, expand and further develop (but not alter) any existing function names, variable names, HTML elements or HTML IDs. You are expected to complete and add any needed or missing instructions, functions, variables and or elements etc.	This section of requirements relates to whether you have used and maintained the standards and pre-set elements defined in the coursework template. 1 — Template Not Used 2 — Template HTML Elements NOT Used 3 — Template Naming Convention NOT Used 4 — Template Functions NOT Used *You are required to use the associated template for this assignment, failure to use the template, or excessive modifications to any predefined elements, structure or functions etc. within the template will also cause this penalty to apply.*	Fixed Penalty **-50%**
2. Features & Functions Your submission will be expected to provide the user with a number of programmed features – each feature will be expected to be straightforward, usable, visible, work intuitively and without error. It is suggested that you think about prioritising the importance and complexity of which feature or function to develop first based on its ease and possible time needed.	This section of requirements relates to the completion of the expected features of your program. **Features available to the user:** 1 — Select and use 8 different, pre-set colours 2 — Select and use a custom colour blend tool (mix #RRGGBB) 3 — Select and use an 'eraser' to remove any painted mistakes 4 — Select and use 4 different, pre-set brush sizes 5 — Select and use a 'clear all' option that resets (clears) the entire canvas *This list covers each of the features your program will be expected to implement. Partially complete functions may still garner marks.*	Up to **20 Marks**
3. Use & Operation Your program will be expected to operate in a specific manner. These operations will need to support your program's features and should be easily identifiable within the code as being planned, commented and well tested.	This section of requirements relates to your program's design, implementation, operation and use. **Your program should support the following uses and operations:** 1 — Paint mode: once selected if the user depresses the left mouse button the system tracks the mouse movement leaving a painted line in its path; the colour will be based on the active colour selected. The paint effect must be located directly beneath the mouse position and be based on the currently selected brush size. This mode should be somehow indicated to the user. 2 — Erase mode: once selected if the user depresses the left mouse button the system tracks the mouse movement and removes any existing paint from the canvas; the erase effect must be located directly beneath the mouse position and be based on the currently selected brush size. This mode should be somehow indicated to the user. 3 — Four, organised and labelled tools (displayed on the browser, these could be buttons or icons) are used to set the size of the paint brush; the currently selected size should be somehow indicated to the user. 4 — Eight, organised and labelled tools (displayed on the browser) are used to set (or change) the selected paint colour; the currently selected/active colour should be somehow indicated to the user. 5 — A blend tool (created by you and displayed on the browser) that can be used to blend different RRGGBB values to create and select a "unique" colour (not available as a pre-set). If selected, the blended "unique" colour should be somehow indicated to the user. 6 — A clear tool (shown on the browser) that when activated clears the canvas by setting its entire background to the currently selected/active	Up to **24 Marks**

University of Hertfordshire UH

	colour; the current background colour should be somehow indicated to the user. *This list covers each of the operations your program will be expected to provide. Partially complete operations may still garner some marks.*	
4. **Programming Standards and Documentation** Your program will be expected to be internally well documented using both HTML and JavaScript comments. In addition, you will be required to include a documented "change" and "test" log. Your "change log" (as illustrated in the template) should be located close to the top of your program and is expected to (chronologically) list any notable changes, revisions and updates you have made to your code, design, implementation etc. For your ease, it is strongly suggested that you maintain this "as you go" rather than trying to retrospectively reengineer or "create" it. You are expected to (as you develop your program) update and maintain an overall "test log" (shown close to the bottom of your program, see template). Instead of "logging" individual tests on specific lines of code log any errors at the end of each development session; e.g. at the end of a tutorial or assignment workshop or at the end of a couple of hours of individual work (perhaps at home, a lab or in the LRC) - this will reduce the number of entries while also providing more suitable evidence on how you developed your solution and what problems you experienced.	This section of requirements relates to your use of programming standards and documentation (e.g. use of indentation, naming conventions, coding style, programming quality, use of functions, commenting consistency, technical merit, program planning, design, organisation and testing). **Your program should be internally documented and contain the following:** 1 – A maintained Change Log (located at the beginning of your code) that identifies your program's development history, your log should follow the structure illustrated in the template and provide a dated (dd/mm/yyyy) entry that defines a completed feature, function or update; completed means, designed, programmed and tested. You are also expected to log instances where you have improved or refactored a previously completed feature, function or capability or attempted/completed any of the higher-level enhancements 2 – A well commented test log (located at the end of your code) that confirms the final tested state (e.g. **PASSED** or **FAILED**) of each of the features, functions and operations (identified in section 2 and 3); ensure you include clear reasons why a test failed (e.g. what you think the issue is together with why and a possible solution that could resolve it next time). 3 – Comprehensive commenting on all significant, important and unique lines of code; comments must include details on what this line does, how it does it and why. Don't forget the HTML comments look like this: **<!-- some comment -->** and JavaScript comments look like this: **//some comment** 4 – Comprehensive variable names – each variable must be appropriately and consistently named (naming convention) and indicate the type or nature of data being stored; e.g. (i)INT, (s)STRING, (b)BIT, (c)CHAR, (f)FLOAT – e.g. **sActiveColour** = "#FF0000"; the prefixed 's' indicates the type of data being stored. 5 – Comprehensive function names – each function must be appropriately and consistently named (your naming convention must include and underscore '_' followed by your initials in uppercase as the last part of its name) and include a clear and concise comment (either directly above or directly below its declaration), if the function expects or uses arguments or parameters these should also follow a consistent naming convention; e.g. function **fnPaintAt_MW**(iXPos, iYPos). *This list indicates each of the areas that will be assessed as part of your program's review.*	Up to **44 Marks**
5. **Overall Solution** In addition to each of the highly structured sections above, there is an additional mark (up to a maximum of 12) that will be awarded based on your program's overall technical quality, standard, detail, depth and effort. This could also include evidence of code refactoring (reworking previous code in an attempt to minimize or remove unused and/or erroneous code etc) or your attempt to complete the **higher-level enhanced challenge**.	This section is setup to recognise and reward any additional or perceived additional time and/or effort you invested in the development and completion of your program and/or the implementation of at least some higher-level enhancement. **Additional marks can be gained based on the following:** 1 – Overall Quality 2 – Attention to Detail 3 – Technical Depth 4 – Higher-Level Enhancement *This list indicates areas of potential reward and recognition with regards to your programming effort.*	Up to **12 Marks**
6. **Solution Failure Penalty** Your program is expected to work; you will be issued with a penalty (of up to -10 marks) If (for any reason) your submission cannot be immediately run and/or tested. The extent of this penalty will depend on the time, type and effort needed to	This section is setup to identify issues when attempting to run your program. It also provides markers with an opportunity to try to "fix" any minor errors that seem to prevent your solution from being run and/or tested. 1 – HTML Related Problems 2 – JavaScript Related Problems 3 – File/Document Related Problem	Up to **-10 Marks**

correct and test your code. To avoid this penalty, make sure **YOU** have tested and fixed any errors that may prevent your program from running before you submit it.	4 — Syntax and/or Logical Errors 5 — Operations or User Interactivity Errors *This list indicates potential areas that could cause problems when attempting to test your program. To avoid this possible penalty – fully ensure that **you** test and **fix** your own work!*	

DEADLINES AND ASSIGNMENT WEIGHTINGS

1 This assignment is worth *%26* of the **overall assessment** for this module.

2 You are expected to spend about *20* Hours to complete this assignment to a satisfactory standard

3 Date assignment set	8th October 2018	Date completed assignment to be handed in	5th November 2018
4 Target date for return of marked assignment	3rd December 2018		

INTERNAL MODERATION

This assignment has been internally moderated. I confirm: • That the assignment set, meets the requirements of the module and that the brief provides appropriate content for students to successfully complete the assignment. • That the assessment is at an appropriate level and matches QAA level descriptors and is an appropriate form of assessment within the total range of assessments for this module • That the marking scheme is attached and that students can determine how marks are allocated. • That this assessment can be completed **and** marked within University timeframes, and provides detailed feedback (more than just a grade) that supports learning.	***Moderator name, signature and date***