DIY Electrical Quiz board yr8.docx

AURORA COLLEGE

MATHEMATICS & SDD

ASSESSMENT TASK 2 NOTIFICATION

SUBJECT	Software Design and Development
YEAR	Year 11
TYPE OF TASK	Major Project
WEIGHTING OF TASK	40%
DATE TASK ISSUED	Friday, 14 May 2021
METHOD OF ISSUE	Email from Head Teacher to all Students, Parents and Aurora Coordinators
DATE TASK DUE	Monday, 9 August 2021
METHOD OF SUBMISSION	Upload to GitHub

Outcomes assessed

P1.2	describes and uses appropriate data types
P3.1	identifies the issues relating to the use of software solutions
P4.1	analyses a given problem in order to generate a computer-based solution
P4.3	uses a variety of development approaches to generate software solutions and distinguishes between these approaches
P5.1	uses and justifies the need for appropriate project management techniques
P5.2	uses and develops documentation to communicate software solutions to others
P6.2	communicates with appropriate personnel throughout the software development process
P6.3	designs and constructs software solutions with appropriate interfaces

Task Description

Overview

Your task is to design and develop a Graphical Hangman Game using *pygame* for a client (Dr G).
You will be working in groups of two students.
You will use the agile approach with 3 sprints. You will meet with the client at the beginning and end of each sprint to discuss the 'problem' and requirements, and you will develop a brief based on these meetings. At the end of each sprint you will meet with the client to present source code and documentation and demonstrate the current version of the game.

Milestones

17 May: Initial 'client consultation' - you will be briefed about the client's requirements.
4 Jun: Code freeze sprint 1
7 Jun: Client meeting sprint 1
16 Jul: Code freeze sprint 2
19 Jul: Client meeting sprint 2
6 Aug: Code freeze sprint 3
9 Aug: Final submission and presentation to client.

Submission

- All work is to be submitted in the team GitHub repository.
- All documents listed under *other project components* (below) should be in the **`docs`** folder.
- All images and/or audio that the program uses at runtime should be in the **`assets`** folder.
- All tests and test data should be in the **`test`** folder.

Project Management

You will:

- use the project management tools in GitHub to collaborate on the project;
- create a separate fork for each sprint and then create a pull request which will allow you to start a discussion with your team;
- allocate tasks by creating *issues* in the planner;
- make regular *commits* which include a detailed description of the changes you made, the reason for those changes, and any new *issues* that need to be raised. The comments associated with the issues and commits will constitute your logbook;
- make a merge request at the end of each sprint and merge the fork back to the master repository;
- label all files clearly with a name that indicates what the file is, for example 'data_dictionary.pdf', 'storyboard.png' etc.

You may:

- create your documentation in a wiki associated with the project.

Marking Guide

Code (50 marks)

In each category below, higher marks will be awarded to teams that demonstrate appropriate use of comments including docstrings, descriptive variable names, a modular approach and efficient code.

General coding and program marking rubric (/20)

This marking rubric is used regardless of what sprint your solution is on. Note that if you only get marks in this section, it means that you have not met requirements for any of the sprints of this project.

	9 – 10	7 – 8	5 – 6	3 – 4	0 – 2
Coding	Writes valid code which employs a modular approach including passing and returning values, and demonstrates: appropriate use of comments including docstrings; descriptive variable names.	Writes valid code which employs a modular approach, and demonstrates: appropriate use of comments including docstrings; descriptive variable names.	Writes valid code which employs a modular approach and demonstrates some of the following: appropriate use of comments including docstrings; descriptive variable names.	Writes valid code which demonstrates some of the following: appropriate use of comments including docstrings; descriptive variable names.	Writes some valid code
Working program				Creates a program which performs some functions.	Creates a program which runs.
Testing			Writes a comprehensive suite of unit tests as appropriate for the program	Writes some unit tests	Writes a unit tests

Marking rubrics for each sprint (/30)

	9 – 10	7 – 8	5 – 6	3 – 4	0 – 2
First sprint requirements (Text-based game)	Working game which **exceeds** the requirements specified in the brief for the first sprint	Working game which meets **all** requirements specified in the brief for the first sprint	Working game which meets **most** requirements specified in the brief for the first sprint	Working game which meets **some** requirements specified in the brief for the first sprint	Working game which meets **one** requirement specified in the brief for the first sprint
Second sprint requirements (Graphics rendering)	Game with basic graphics which **exceeds** the requirements specified in the brief for the first sprint	Game with basic graphics which meets **all** requirements specified in the brief for the first sprint	Game with basic graphics which meets **most** requirements specified in the brief for the first sprint	Game with basic graphics which meets **some** requirements specified in the brief for the first sprint	Game with basic graphics which meets **one** requirement specified in the brief for the first sprint
Third sprint (Graphical game)	Graphical game which **exceeds** the requirements specified in the brief for the first sprint	Graphical game which meets **all** requirements specified in the brief for the first sprint	Graphical game which meets **most** requirements specified in the brief for the first sprint	Graphical game which meets **some** requirements specified in the brief for the first sprint	Graphical game which meets **one** requirement specified in the brief for the first sprint

Project Components (50 marks)

	9 – 10	7 – 8	5 – 6	3 – 4	0 – 2
Project Planning	**Thorough** evidence of planning of your project using tools in GitHub e.g. detailed planning and task allocation, regular commits and raising of issues with and **informative** annotations, peer review of code.	**Substantial** evidence of planning of your project using tools in GitHub e.g. detailed planning and task allocation, regular comments and generation of issues with **descriptive** annotation**.**	**Basic** evidence of planning of your project using tools in GitHub e.g. records of planning and allocating most tasks, irregular commits and generation of issues with basic annotation.	**Incomplete** evidence of planning of your project using tools in GitHub e.g. some planning and task allocation and **some** comments on issues.	Some use of planning tools or evidence of task allocation.
	5	4	3	2	0 – 1
Design brief	Students **identify** the problem and give a **description** of their proposed solution including **explicit,** verifiable outcomes	Students **state** the problem AND give a **definition** of their proposed solution including outcomes.	Students **state** the problem AND give a **definition** of their proposed solution including **some** outcomes	Students **state** the problem OR give a **definition** of their proposed solution including **some** outcomes	Students identify some relevant issues
Storyboard	'Screenshots' of the game at **each** important phase of play clearly described and in logical order	'Screenshots' of the game at **some** important phases of play but lacking clear explanations or logical order	'Screenshots' of the game at **some** important phases of play with descriptions	'Screenshots' of the game at **some** important phases of play	A screenshot of the game
Data flow diagram	**Explicitly** show **all** required processes, data flows, external entities and data stores	**Clearly** show **most** required processes, data flows, external entities and data stores	Show **some** processes, data flows, external entities and data stores	Creates a diagram with some correct symbols	Creates a diagram.
	13 – 15	10 – 12	7 – 9	4 – 6	0 – 3
Algorithm design (Flowchart or pseudocode)	Includes **all** terminators, processes, inputs and outputs, mainline, subprograms and decisions using correct symbols (flowchart) or syntax (pseudocode)	Includes **most** terminators, processes, inputs and outputs, main line, subprograms and decisions using correct symbols (flowchart) or syntax (pseudocode)	Includes **some** terminators, processes, inputs and outputs, main line, subprograms and decisions using correct symbols (flowchart) or syntax (pseudocode)	Writes pseudocode or creates flowchart which provides partially correct description of main line.	Writes some correct pseudocode or creates a flowchart with some correct symbols

	5	4	3	2	0 – 1
Data dictionary	Data dictionaries give **correct** descriptions of **all** variables used in the program.	Data dictionaries give descriptions of **most** variables used in the program.	Data dictionaries give descriptions of **some** variables used in the program.	Data dictionaries give partial descriptions of **some** variables used in the program, (missing important columns)	Table with some properties of a data dictionary
User manual	**Describes** the purpose of the game and give detailed and logical instructions for playing. User manual **includes** system requirements and licencing	**Defines** the purpose of the game and give detailed and logical instructions for playing. User manual **states some** system requirements and licencing	**States** the purpose of the game and give detailed and logical instructions for playing.	States the purpose of the game but gives incomplete instructions.	Describes some elements of the game