Ofqual QAN	**610/6138/4**
Quartz ID	**2332**
Qualification	**Level 2 Diploma in Computing**

Unit ID	**2334**
Assessment	**Computational Problem Solving**
Assessment Type	**Global Assignment**
Assessment Cycle	***Sample Assignment***

Assignment title: Lost Pet Finder App

Marking Scheme

Markers are advised that, unless a task specifies that an answer be provided in a particular form, then an answer that is correct (factually or in practical terms) **must** be given the available marks. If there is doubt as to the correctness of an answer, the relevant NCC Education materials should be the first authority.

This marking scheme has been prepared as a **guide only** to markers and there will frequently be many alternative responses which will provide a valid answer.

Each candidate's script must be fully annotated with the marker's comments (where applicable) and the marks allocated for each part of the tasks.

Throughout the marking, please credit any valid alternative point.

Where markers award half marks in any part of a task, they should ensure that the total mark recorded for the task is rounded up to a whole mark.

Marker's comments:

Moderator's comments:

Mark:	**Moderated mark:**	**Final mark:**

Penalties applied for academic malpractice:

Introduction

You should read through the information given and each task section of the assignment carefully before you begin.

You must work independently and should not share your work with other learners. All work produced must be your own.

Any sources of information, text, images created by others that you include in your work must be clearly identified and referenced. The use of work of others as your own without permission or without proper acknowledgement is considered plagiarism and may result in disqualification from the assessment.

You may ask your teacher for support if you have questions about the requirements of each task, however, they cannot give guidance with solutions for tasks or provide you with feedback about how to improve your work.

Scenario

In the vibrant city of Bristol, pets are a beloved part of many households. However, as a busy city with its mix of urban streets, green spaces, pets can easily go missing. Whether it is a dog slipping out of a garden gate, a cat wandering too far, or a rabbit escaping, pet owners often rely on posters or social media to find their animals—methods that can be slow and unreliable.

To help address this issue, Bristol City Council is partnering with local animal shelters and tech students to develop a **Lost Pet Finder App**. This app will allow residents to report lost pets, submit sightings, and receive alerts when a potential match is found. You have been selected as part of a student tech team to design and prototype a basic version of the app's logic using computational thinking, programming, and ethical awareness.

Aims:

You need to:

1. Apply computational thinking to design the app's logic.
2. Write a simple program that allows users to:
    - Report a lost pet.
    - Submit a pet sighting.
    - Match sightings with lost pet reports based on basic criteria (e.g., type, colour, location).
3. Test your program with sample data.
4. Evaluate your solution by reflecting on its strengths, limitations, and possible improvements.
5. Consider ethical issues such as:
    - How user data (e.g., location, contact info) is collected and stored.
    - How to prevent false reports or misuse of the app.
    - How to ensure fairness and accessibility for all users.

Task 1: Computational Thinking (LO1) – 50 Marks

Produce a report with **NINE HUNDRED (900)** words.

Design the **core logic and structure** of the Lost Pet Finder App for use in Bristol by applying the **FOUR (4)** principles of computational thinking:

a) Problem Decomposition **10 Marks**

Break the problem into smaller parts:

- What features should the app include (e.g., report lost pet, submit sighting, send alerts)?
- What are the steps involved in matching a sighting to a missing pet?

0 - 2 marks	3 marks	4 - 5 marks	6 marks	7 -10 marks
Does not effectively break down the problem. Fails to identify necessary components.	Attempts to break down the problem but lacks detail. Identifies few necessary components.	Breaks down the problem into manageable parts. Identifies some necessary components.	Breaks down the problem into detailed and manageable parts. Identifies most necessary components.	Breaks down the problem into highly detailed and manageable parts. Clearly identifies all necessary components.

b) Pattern Recognition **10 Marks**

Identify patterns in pet sightings and reports:

- What traits help match a sighting to a missing pet (e.g., breed, colour, location)?
- Are there common areas in Bristol where pets are often lost or found (e.g., parks, neighbourhoods)?

0 - 2 marks	3 marks	4 - 5 marks	6 marks	7 -10 marks
Struggles to identify patterns in data, with significant gaps in understanding or detail. Provides few or no examples.	Identifies some patterns in data, but with gaps in understanding or detail. May recognise obvious patterns but miss more complex ones.	Identifies relevant patterns in data, demonstrating a good understanding of how patterns can be used to solve the problem. For example, recognises that certain pet types are more likely to be found in	Identifies detailed patterns in data, demonstrating a strong understanding of how patterns can be used to solve the problem. Provides good examples and explanations.	Identifies complex and subtle patterns in data, demonstrating an excellent understanding of how patterns can be used to solve the problem. Provides clear examples and explanations.

		specific areas and uses these patterns to inform the algorithm. May miss more subtle patterns or not fully explain their impact.		

c) Abstraction **10 Marks**

Focus on essential information:

- What data is necessary (e.g., pet type, last seen location, time and distinguishing features)?
- What can be ignored to simplify the process?

0 - 2 marks	3 marks	4 - 5 marks	6 marks	7 -10 marks
Struggles to focus on relevant information, including many irrelevant details or missing essential information. Demonstrates a limited understanding of abstraction.	Focuses on some relevant information, but may include irrelevant details or miss some essential information. Demonstrates a basic understanding of abstraction.	Focuses on relevant information, ignoring some irrelevant details, and demonstrates a good understanding of abstraction. Provides examples.	Focuses on essential information, ignoring most irrelevant details, and demonstrates a strong understanding of abstraction. Provides good examples.	Focuses on the most essential information, ignoring irrelevant details, and demonstrates an excellent understanding of abstraction. Provides clear examples.

d) Algorithm Design **20 Marks**

Create a flowchart or pseudocode to represent the logic of the Lost Pet Finder App for how the app will match sightings with missing pet:

- What happens when a user submits a sighting?
- How does the app notify the pet owner?

0 - 6 marks	7 marks	8 - 12 marks	13 marks	14 - 20 marks
Struggles to create a logical flowchart or pseudocode, with significant gaps in understanding or detail. Demonstrates little to no understanding of algorithm design. Limited or unclear representation of the algorithm with many errors.	Creates a flowchart or pseudocode that outlines the algorithm, but with some gaps in logic or detail. Shows some understanding of algorithm design but with noticeable gaps. Basic representation of the algorithm with significant errors.	Creates a logical flowchart or pseudocode that outlines the algorithm. Demonstrates a working understanding of algorithm design. Adequate representation of the algorithm with some errors.	Creates a detailed and logical flowchart or pseudocode that outlines the algorithm. Shows a solid understanding of algorithm design. Mostly logical and clear representation of the algorithm with minor errors.	Creates a highly detailed and logical flowchart or pseudocode that clearly outlines the algorithm. Demonstrates a deep and accurate understanding of algorithm design principles. Logical and clear representation of the algorithm with no errors.

Task 2: Programming Task (LO 2) – 20 Marks

Using a language (e.g. Python), write a simple program that includes:

- A way to store lost pet reports and sightings (e.g., using lists or dictionaries).
- A function to compare sightings with lost pets.
- A basic interface (text-based or graphical) for users to enter data.

0 - 6 marks	**7 marks**	**8 - 12 marks**	**13 marks**	**14 - 20 marks**
Limited or unclear code using inappropriate programming concepts with many errors. Code is poorly organised and inefficient. Uses inappropriate data structures and algorithms. Lacks comments and documentation.	Basic code using appropriate programming concepts with significant errors. Code is somewhat organised but may be inefficient. Uses basic data structures and algorithms. Lacks comments and documentation.	Adequate code using appropriate programming concepts with some errors. Code is organised and mostly efficient. Uses appropriate data structures and algorithms. Includes minimal comments and documentation.	Mostly functional, well-structured code using appropriate programming concepts with minor errors. Code is well-organised and efficient. Uses appropriate data structures and algorithms. Includes some comments and documentation.	Functional, well-structured code using appropriate programming concepts with no errors. Code is exceptionally well-organised, efficient, and easy to understand. Uses appropriate data structures and algorithms. Includes comments and documentation.

Task 3: Testing and Evaluation (LO 2) – 20 Marks

a) Testing — 10 Marks

- Test your program with **THREE (3)** lost pet reports and **THREE (3)** sightings and complete the testing by documenting it in the Test Log.
- Show the results of the matching process.
- Include screenshots or output logs.

0 - 2 marks	3 marks	4 - 5 marks	6 marks	7 -10 marks
Minimal or no testing. Test cases do not pass successfully. Lacks test results and explanations.	Limited testing with a few test cases. Few test cases pass successfully. Lacks detailed test results and explanations.	Adequate testing with several test cases. Some test cases pass successfully. Includes basic test results and explanations.	Thorough testing with multiple test cases. Most test cases pass successfully. Includes test results and explanations.	Extensive testing with a variety of test cases. All test cases pass successfully. Includes detailed test results and explanations.

b) Evaluation — 10 Marks

Reflect on your solution with **THREE HUNDRED (300)** words on :

- What worked well?
- What challenges did you face?
- How could the app be improved in future versions?

0 - 2 marks	3 marks	4 - 5 marks	6 marks	7 -10 marks
Minimal or no evaluation of the program. Does not identify strengths or weaknesses. No reflection on the development process.	Basic evaluation of the program. Identifies few strengths and weaknesses. Minimal reflection on the development process.	Adequate evaluation of the program. Identifies some strengths and weaknesses. Limited reflection on the development process.	Comprehensive evaluation of the program. Identifies strengths and weaknesses. Reflects on the development process.	Insightful and thorough evaluation of the program. Identifies strengths, weaknesses, and potential improvements. Reflects on the development process.

Task 4: Ethical Considerations (LO 6) – 10 Marks

Write a short reflection with **THREE HUNDRED (300)** words on:

- What ethical issues might arise from using this app?
- How would you protect user privacy and data?
- How could you prevent misuse or false reporting?
- How would you ensure the app is inclusive and accessible?

0 - 2 marks	3 marks	4 - 5 marks	6 marks	7 -10 marks
Limited or unclear reflection on privacy, fairness, and responsible use. Provides inadequate or no strategies for protecting user data. Provides inadequate or no measures to prevent misuse. Demonstrates limited understanding of inclusivity. Provides inadequate or no efforts to ensure the app is accessible.	Basic reflection on privacy, fairness, and responsible use with significant gaps. Offers minimal strategies for protecting user data. Offers minimal measures to prevent misuse. Shows basic understanding of inclusivity. Makes minimal efforts to ensure the app is accessible.	Adequate reflection on privacy, fairness, and responsible use with some gaps. Provides some strategies for protecting user data, though may lack detail. Provides some measures to prevent misuse, though may lack detail. Demonstrates good understanding of inclusivity. Attempts to ensure the app is accessible, though may have minor limitations.	Good reflection on privacy, fairness, and responsible use with minor gaps. Offers clear strategies for protecting user data and ensuring confidentiality Offers clear measures to prevent misuse and false reporting. Shows a strong understanding of inclusivity. Makes efforts to ensure the app is accessible and usable by most users.	Thoughtful and thorough reflection on privacy, fairness, and responsible use. Provides detailed strategies for protecting user data and ensuring confidentiality. Provides detailed measures to prevent misuse and false reporting. Exhibits a thorough understanding of inclusivity. Ensures the app is accessible and usable by all users, including those with disabilities.

Learning Outcomes Matrix

Task	Learning Outcomes assessed	Marker can differentiate between varying levels of achievement
1	LO 1	Yes
2	LO 2	Yes
3	LO 2	Yes
4	LO 6	Yes

Learning Outcome	Fail	Pass	Merit	Distinction
Describe and use the principles of computational thinking, including problem decomposition, pattern recognition, abstraction, and algorithm design.	Demonstrates li ttle or no understanding of computational thinking principles. Unable to correctly define or describe key concepts such as decomposition, pattern recognition, abstraction, or algorithm design.	Demonstrates a basic understanding of computational thinking principles. Define and describe decom position, pattern recognition, abstraction, and algorithm design with some accuracy.	Demonstrates a good understanding of computational thinking principles. Accurately describes and explains the role of decomposition, pattern recognition, abstraction, and algorithm design.	Demonstrates a comprehensiv e and insightful understanding of computational thinking principles. Provides detaile d and precise explanations of decomposition, pattern recognition, abstraction, and algorithm design.
Design, write, test and evaluate programs for well-defined and straightforward problems.	Minimal or no ability to design, write, test, or evaluate programs. Code is largely incomplete, incorrect, or fails to address the problem.	Design and write basic programs that address straightforward problems. Code is mostly functional with some errors or inefficiencies.	Design and write well-structured programs that solve the given problems effectively. Code is mostly correct, efficient, and demonstrates g ood use of programming techniques.	Design, write, test, and evaluate programs with a high level of proficiency. Code is robust, efficient, and well-documented, showing a deep understanding of programming principles.

Understand and apply basic data structures in problem solving.	Demonstrates little or no understanding of basic data structures. Unable to select or use appropriate data structures in problem-solving contexts.	Demonstrates a basic understanding of common data structures. Identify and use simple data structures to solve problems.	Demonstrates a good understanding of basic data structures and their purposes. Selects and applies appropriate data structures (e.g., lists, dictionaries) to solve problems effectively.	Demonstrates a comprehensive understanding of basic data structures and their applications. Selects and applies data structures efficiently and appropriately to solve a range of problems.
Understand basic algorithms and evaluate their efficiency using time and space complexity.	Demonstrates little or no understanding of basic algorithms. Unable to explain or apply the concepts of time and space complexity.	Demonstrates a basic understanding of common algorithms and their purposes. Able to describe time and space complexity in simple terms, though explanations may lack depth or accuracy.	Demonstrates a good understanding of basic algorithms and their efficiency. Accurately explains time and space complexity using Big O notation for standard algorithms.	Demonstrates a comprehensive and analytical understanding of basic algorithms and computational efficiency. Explains and applies time and space complexity concepts with confidence and precision.
Develop logical reasoning and understand of Boolean logic.	Demonstrates little or no understanding of Boolean logic or logical reasoning. Unable to correctly use logical operators (AND, OR, NOT) or truth tables.	Demonstrates a basic understanding of Boolean logic and logical reasoning. Can use simple logical operators and construct basic truth tables.	Demonstrates a good understanding of Boolean logic and logical reasoning. Accurately uses logical operators and constructs truth tables for common expressions.	Demonstrates a comprehensive and confident understanding of Boolean logic and logical reasoning. Uses logical operators, truth tables, and logic gates with precision and fluency.

Understand and apply ethical considerations in computational problem solving.	Demonstrates li ttle or no awareness of ethical issues in computational problem solving. Unable to identify or explain key ethical concerns such as privacy, bias, or data security.	Demonstrates a basic understanding of ethical considerations in computing. Can identify commo n ethical issues (e.g., data privacy, intellectual property, digital wellbeing).	Demonstrates a good understanding of ethical principles in computational contexts. Identifies and explains a range of ethical issues relevant to specific problems.	Demonstrates a comprehensiv e and insightful understanding of ethical considerations in computing. Critically evaluates ethical issues such as bias in algorithms, data misuse, and societal impact.