Mock Student Responses

Computer Science Assessment 3 Set S of 2026

Synthetic examples for teaching and marking practice | completed work

The materials accompany the EcoQuest Learning sample coursework brief and staff mark scheme, revised on 14 October 2026. All three tasks are addressed: Python game 70 marks, database 15 marks and spreadsheet 15 marks.

The examples are synthetic materials for teaching and standardisation. Test records describe execution of the supplied generated examples; they do not represent observed learner activity. No learner signature or supervision record is supplied.

Three independent responses to the same EcoQuest coursework are provided. The response identifiers are SYN01, SYN02 and SYN03. The examples contain different levels of completeness and accuracy. They are intended to support discussion of evidence and assessment criteria, and are not prescribed answers.

The associated Staff documents contain proposed marking and teacher guidance. Those documents should be kept separate when this file is used for an unmarked exercise.

Response SYN01

Synthetic coursework report and evidence record

The project addresses IQM3010 Computer Science Assessment 3, Set S of 2026, for EcoQuest Learning. The client requires a text-based environmental terminology game for GCSE pupils, a relational database and an Excel analysis of supplied practice results. The three products are treated as separate deliverables that share a scenario; no technical connection between them is claimed.

1. Analysis

EcoQuest Learning develops educational games for schools. The proposed prototype is a console game in which a pupil receives a clue and guesses one letter at a time to reveal an environmental term. The database stores the same 12 term-and-clue records together with difficulty and subject values, while a Games table is prepared to associate a game record with a clue. The Excel workbook uses the supplied 12-record practice dataset to identify words whose observed success and rating may warrant review. These products are independent: no game history is written to SQLite and no Python output feeds Excel.

Success is defined by checkable behaviour. The game must use the supplied matching pairs, select randomly without reuse in a session, begin each round with six incorrect guesses, accept one letter after trimming surrounding spaces and ignoring case, reject blank, numeric and multi-character input without penalty, reveal every occurrence, preserve state after a repeated guess, show the guessed letters and remaining attempts, end on a win or loss and offer another round or exit. For the database, the five Clues fields must preserve all 12 supplied rows, use unique ClueID values, require meaningful values, constrain Difficulty to integer values 1–5 and Subject to the three supplied labels, and connect Games.ClueID to Clues.ClueID. For Excel, every word needs a formula and one-decimal percentage; the totals and unweighted average must be calculated; rows must be sorted ascending without separating fields; recommendations and one contextual modelling limitation must be recorded.

The user is a GCSE pupil revising environmental science. Immediate letter-level feedback can make a short practice round easier to follow, but a text-only interface may be less accessible to a pupil who relies on visual adjustment, speech input or a non-text interface. The program therefore avoids time pressure and uses a neutral remaining-attempts counter. The work is limited to a local console, supplied fictional data, standard Python and approximately 20 supervised hours. A staged approach is suitable: establish term/clue data and round logic first, test boundary behaviour, then complete the independent database and spreadsheet tasks. No consultation with actual pupils or client representatives is represented.

2. Design

The game is decomposed into a data module, a GameRound class and an EcoQuestGame session manager. The round object holds the selected term and clue, guessed letters and remaining attempts. The session manager owns the unused-pair list and random choice. The round method validates input before state changes, detects repeats, updates guesses or attempts, then returns a status and display state. The console loop displays the clue, progress, guessed letters and neutral counter, then offers replay or exit. If all supplied terms have been used, the session ends with a clear message.

Pseudocode: copy the supplied term/clue pairs into an unused list; while the list is not empty, choose a random unused pair and remove it; create round state with six attempts and no guesses; while the word is incomplete and attempts remain, display clue, masked word, guesses and attempts; read and normalise one input; if it is blank, non-letter or longer than one character, report invalid input and leave state unchanged; if it has already been guessed, report a repeat and leave attempts unchanged; otherwise record it, reveal every matching position if correct, or subtract one attempt if incorrect; report win or loss and the correct term; ask whether to continue; if the list is empty, end the session. The class attributes and method responsibilities are stated explicitly, and the pair remains associated as one data object.

The test plan covers initial state, case/space handling, repeated letters in a word, repeated correct and incorrect guesses, invalid input, a new incorrect guess, the sixth miss, a completed word, replay, session exhaustion and uniqueness of selected terms. Database tests cover valid linked rows, duplicate keys, missing foreign keys, invalid and boundary difficulty values, a blank required field and unknown subject. Spreadsheet checks cover every per-word percentage, totals, mean, ordering, recommendations and the limitation. Expected results are stated before the generated checks are run.

3. Python implementation

The native package contains game.py and terms_data.py. GameRound owns one round's state and performs input validation, guess processing, letter reveal and outcome checks. EcoQuestGame selects only from remaining supplied pairs, and the console loop handles display and replay. The implementation uses the Python standard library and keeps the game independent of the database and workbook. The source data remain separate so that a clue cannot be detached from its term during selection.

The selected logic is equivalent to letter = raw.strip().casefold() followed by a length/alphabetic check. When the letter has already been guessed, the method returns a repeated-guess status before changing attempts. Otherwise it records the letter, reveals all matching positions through a membership check over the full term, and deducts one attempt only when a new letter is absent. A final status identifies a completed word or zero attempts. The program is a synthetic example written for standardisation; the generated test harness is not candidate activity.

4. Recorded testing

The generated Python checks recorded 15/15 expected outcomes as passing. They include the initial six-attempt state, normalised input, all repeated letters in habitat, repeated correct and incorrect guesses without penalty, invalid input without state change, a new miss, the sixth-miss loss, a win, session non-reuse and clean exhaustion. The SQLite checks recorded 10/10 outcomes as passing: the imported count and values, field names and declared capacities, Games keys, a valid link, duplicate GameID rejection, missing ClueID rejection, Difficulty values 0, 6 and 2.5 rejection, blank Word rejection and unknown Subject rejection. The workbook check confirmed 12 row formulas, percentage formatting, totals 435 and 273, the unweighted 61.6362…% mean, ascending full-row order and a zero-result formula-error scan. These are executions of generated artifacts under defined synthetic inputs; they are not evidence of a learner's supervised activity. No defect was observed in the recorded test set, so no correction or retest is claimed.

The following expected and actual results are recorded from the supplied synthetic implementation. Complete machine-readable records and test scripts accompany the native files.

Python test	Expected outcome	Recorded outcome
G01 | Round starts with six attempts and no revealed letters	attempts: 6; word: _ _ _ _ _ _ _	As expected [PASS]

Python test	Expected outcome	Recorded outcome
G02 | Case and surrounding spaces are normalized; all matching letters reveal	status: correct; attempts: 6; word: _ a _ _ _ a _	As expected [PASS]
G03 | Repeated correct guess gives feedback without penalty	status: repeated_correct; attempts: 6	As expected [PASS]
G04 | A new incorrect guess uses one attempt	status: incorrect; attempts: 5	As expected [PASS]
G05 | Repeated incorrect guess does not use another attempt	status: repeated_incorrect; attempts: 5	As expected [PASS]
G060 | Invalid input '' leaves attempts unchanged	status: invalid; attempts: 5	As expected [PASS]
G061 | Invalid input '7' leaves attempts unchanged	status: invalid; attempts: 5	As expected [PASS]
G062 | Invalid input 'aa' leaves attempts unchanged	status: invalid; attempts: 5	As expected [PASS]
G07 | Completing a term ends the round as a win	status: win; word: p r e y	As expected [PASS]
G08 | Sixth distinct miss ends the round as a loss	status: loss; attempts: 0	As expected [PASS]
G09 | All 12 supplied pairs are selected once without term reuse	count: 12; unique_terms: 12; matching_pairs: true	As expected [PASS]
G10 | After all supplied terms are used the session ends cleanly	Session ends	As expected [PASS]
G11 | CLI completes rounds through replay and exits on n while another round remains	completed_rounds: 3; exit_without_exhaustion: true	As expected [PASS]
G13 | CLI completes habitat after each distinct letter and reveals both a positions	completed: true; all_matching_positions_visible: true; input_exhausted: false	As expected [PASS]
G12 | Headerless five-field CSV preserves the same 12 term/clue pairs	rows: 12; fields_each: [5]; pairs_match: true	As expected [PASS]

5. Python evaluation

The game evidence supports the stated success criteria for the generated version. All 15 recorded checks produced the expected result, including initial state, normalised input, repeated-letter reveal, repeated guesses, invalid entries, a new miss, the sixth-miss loss, a completed-word win, round replay, command-line completion of habitat after each distinct letter and consistency between the 12-row CSV and Python term data. The bounded session-selection check also selected every pair once, while a separate console-loop check completed three rounds and exited when requested. These results establish the behaviour of the included artifacts for the specified inputs; they do not establish that every possible user action has been tested.

The round class maintains a clear state boundary: the target term and clue, guessed letters and remaining attempts are kept together. Input is trimmed and case-folded before validation, so variations such as a surrounding space or uppercase letter do not create separate guesses. Invalid input returns before the guess set or attempt count changes. A repeated letter returns a repeat status without altering the counter. For a new correct guess, every matching position is rendered from the full term; a new incorrect guess reduces the counter by one. The recorded sixth distinct miss reaches zero and the completed-word test returns a win status. The console presents the clue, mask, guessed letters and counter, followed by a neutral outcome message.

The session manager keeps the term and clue as a pair and removes each selected pair from the unused list. In the fixed-order check, all 12 different terms were returned with their supplied clues and no pair was selected twice. The separate command-line exercise confirms that the user can choose another round and can exit while an unused round remains. The session ends after its supplied set is depleted. Because those tests use deterministic selection, they check association and non-reuse rather than the statistical quality of random distribution. No actual pupil session or supervised performance is represented.

The interface is suitable for a short text-based practice activity for a pupil who can read and enter text. No time limit is imposed, and the attempt count is presented neutrally. Normalizing letter input reduces avoidable entry differences. The interface has no adjustable text, speech output, graphical controls or assessed assistive-technology behaviour. Those omissions may limit access for some users; the present evidence cannot establish accessibility or learning impact. The included terms and clues are supplied practice data, not evidence that every pupil will know the vocabulary.

Two maintenance changes would be appropriate if the prototype were extended. First, the console could accept a documented command to repeat the clue without changing attempts; implementation would add a separate command branch before letter validation, and regression checks would confirm that the clue is redisplayed while the guessed set and counter remain unchanged. Second, a configurable session length could allow supervised practice without exhausting all 12 terms; implementation would select a random sample from the unused pairs, and tests would verify that no pair repeats, that the chosen number of rounds is completed and that the existing exit path still works. These changes are proposals only and have not been implemented or tested.

6. Relational database

A relational database stores related records in tables and uses keys to identify records and connect tables. Clues contains ClueID, Word, Clue, Difficulty and Subject. ClueID is the primary key because it uniquely identifies each clue row and provides a stable link even if a word or its wording changes. Games contains exactly GameID and ClueID. GameID is its primary key; ClueID is a foreign key to Clues.ClueID. One clue may be associated with several game records, while each game row refers to one existing clue. The relationship is enforced by SQLite foreign-key rules. The database contains the 12 supplied clue rows, and Games remains empty until game records are available.

The schema makes each field required, rejects whitespace-only text, restricts Difficulty to integer values 1–5 and Subject to Ecology, Resources or Energy, and rejects duplicate primary keys. All source clue texts are stored without truncation. The executable SQLite file and SQL schema are provided for staff review. Checks are run against fresh copies of the synthetic database so that each rejected insertion leaves the seeded 12 rows intact.

Database test	Expected outcome	Recorded outcome
D01 | All 12 source rows, field names and Games key structure are present	12 exact source rows; Clues has the five required fields; Games has GameID and ClueID, with GameID as primary key.	As expected [PASS]
D08 | Suitable text/numeric types preserve every supplied value	Suitable text and integer types; all supplied values preserved without truncation.	As expected [PASS]
D02 | A game linked to an existing clue is accepted	ACCEPTED	As expected [PASS]
D03 | Duplicate GameID is rejected	REJECTED	As expected [PASS]
D04 | Game referring to missing ClueID is rejected	REJECTED	As expected [PASS]
D05-0 | Out-of-range or non-integer Difficulty 0 is rejected	REJECTED	As expected [PASS]
D05-6 | Out-of-range or non-integer Difficulty 6 is rejected	REJECTED	As expected [PASS]
D05-2.5 | Out-of-range or non-integer Difficulty 2.5 is rejected	REJECTED	As expected [PASS]
D06 | Blank required Word is rejected	REJECTED	As expected [PASS]
D07 | Unknown Subject is rejected	REJECTED	As expected [PASS]

7. Spreadsheet analysis

The workbook keeps the supplied Clue ID, Word, Subject, Difficulty, Games played and Games won fields and adds a Win percentage formula in each of the 12 rows. Each formula divides wins by games played and displays one decimal place. The complete rows are sorted from the lowest rate to the highest. Total games played is 435 and total games won is 273. The workbook uses AVERAGE over the 12 unrounded per-word formula results; it returns 61.6362…%, displayed as 61.6%. It does not substitute the overall wins-to-games ratio.

The identified review group is habitat (25.0%, difficulty 1; 10 wins from 40 games), turbine (30.0%, difficulty 3; 12/40), predator (80.0%, difficulty 4; 24/30) and renewable (66.7%, difficulty 5; 20/30). The recommendation is to compare the clue and rating and seek further learner feedback before changing either. These rates are signals for investigation rather than proof that difficulty or clue wording caused a result. The workbook's limitation explains that aggregate simulated outcomes do not record prior knowledge, exposure, learning time or guessing strategy, so they cannot identify cause or predict every GCSE pupil's performance.

8. Accompanying evidence

Candidate/Supporting artifacts/SYN01/ contains the Python program and supplied term data, the SQL and SQLite database, the completed analysis workbook, the source data and the recorded checks. Paths are listed in the README file for the response. The editable files form part of this response and should be reviewed alongside the report.

IQM3010 A3 Candidate Coursework Brief, Set S of 2026, supplied with the sample assessment.

IQM3010 A3 supplied EcoQuest term, clue and test-data resources.

Generated Python source and test records included with this response.

Generated SQLite schema and validation records included with this response.

Synthetic authorship declaration: This report, code, database, workbook and test record were generated as synthetic staff practice material. They do not represent candidate-authored work, a learner's AI declaration, a signed submission, an observed supervised session or academic approval.

Response SYN02

Synthetic coursework report and evidence record

The project addresses IQM3010 Computer Science Assessment 3, Set S of 2026, for EcoQuest Learning. It includes a report, a Python environmental terminology game, an SQLite database and an Excel workbook. The three products are treated as separate outputs within the same scenario. Design, implementation and test evidence are presented for each product.

1. Analysis

EcoQuest Learning intends to provide GCSE pupils with a short environmental-science revision game. A clue is displayed with a hidden term, and a pupil enters letters until the term is completed or the six-attempt limit is reached. The twelve supplied term-and-clue pairs are also used in an SQLite database and a separate spreadsheet analysis.

The game accepts a single letter after case and surrounding spaces have been normalised. Blank, numeric and multi-character entries should leave the round unchanged. A new incorrect letter reduces the counter, while a repeated guess should not use another attempt. Correct letters are shown in the term, and the outcome and correct term are displayed when the round ends. A further round or exit is offered. The database stores ClueID, Word, Clue, Difficulty and Subject, with a Games table linked to a clue. Required values and the difficulty range are checked. The workbook records plays and wins, calculates percentages, sorts the terms and presents recommendations. An average of the results is included; it is treated as an average of outcomes without specifying whether each word or each game receives equal weight. The game, data storage and workbook are prepared and checked in sequence. The console provides immediate feedback, although text entry and reading may not suit every pupil.

2. Design

A GameRound object stores the target term, clue, guessed letters and remaining attempts. Its methods validate an entry, process a guess, update the displayed mask. An EcoQuestGame object holds the supplied pairs and selects a pair for a round. A console loop displays the clue, masked term, guessed letters and counter, and offers another round or exit.

Pseudocode:
Copy the supplied pairs into a list of unused pairs.
While the list is not empty:
  Select a random pair and remove it from the list.
  Create a round with six attempts and no guessed letters.
  While the term is incomplete and attempts remain:
    Display the clue, mask, guessed letters and attempts.
    Read one letter, trim surrounding spaces and ignore case.
    If the entry is blank, non-alphabetic or longer than one character, report invalid input.
    Else if the letter has already been guessed, report a repeat without reducing attempts.
    Else record the letter; reveal matching positions or reduce attempts for an incorrect guess.
  Display the win or loss and the correct term.
  Ask whether another round should begin.

The planned normal checks use habitat. A new correct a should reveal both a positions with six attempts remaining; a new incorrect z should leave five attempts. A blank entry, 7 and aa should each leave the state unchanged. A completed term should produce a win, six distinct misses should produce a loss and replay should initialise another round. Database checks cover required text and difficulty boundaries. Spreadsheet checks cover row percentages, totals and sorting.

3. Python implementation

The native game uses a purposeful GameRound class for term, clue, guesses and attempts, together with an EcoQuestGame session object. All 12 terms are kept associated with their clues, random selection removes a used pair, invalid strings are rejected without penalty, all matching letters are revealed and the replay route resets round state. Case and surrounding spaces are normalised. The defect occurs in the repeated-guess branch: when an incorrect letter is guessed again, the attempt count is reduced again. This conflicts with the stated requirement and is reproduced by test G05. No correction or retest is recorded.

The database stores the five required fields, all 12 records and a Games table with a unique GameID and foreign key to Clues. It rejects duplicate keys, orphan game rows, invalid Difficulty values and blank Word. However, the schema only requires Subject to be nonblank; it does not constrain values to the three allowed labels. The test with Air is accepted. The response does not claim that this defect was corrected.

4. Recorded testing

The generated Python checks recorded 14 of 15 expected outcomes as passing. Input normalisation, all-occurrence reveal, repeated correct guesses, new incorrect guesses, invalid input, win, loss, session non-reuse, exhaustion and CLI completion of habitat after each distinct letter passed. G05 failed: a repeated incorrect guess reduced attempts from five to four instead of leaving the count at five. The report lists this as an unresolved defect; no fix or retest is claimed.

The SQLite checks recorded 9 of 10 outcomes as passing. The single failure is D07: an unknown Subject value is accepted. The other checks confirm all 12 imported clue rows, valid game linkage, duplicate key rejection, missing clue rejection, invalid Difficulty rejection and blank Word rejection. Workbook checks confirm formulas in all 12 rows, one-decimal formatting, totals and ascending row order. The mean check fails because 62.8% is calculated from totals rather than the unweighted per-word rates. These executions are synthetic and do not represent learner activity.

The following expected and actual results are recorded from the supplied synthetic implementation. Complete machine-readable records and test scripts accompany the native files.

Python test	Expected outcome	Recorded outcome
G01 | Round starts with six attempts and no revealed letters	attempts: 6; word: _ _ _ _ _ _ _	As expected [PASS]
G02 | Case and surrounding spaces are normalized; all matching letters reveal	status: correct; attempts: 6; word: _ a _ _ _ a _	As expected [PASS]
G03 | Repeated correct guess gives feedback without penalty	status: repeated_correct; attempts: 6	As expected [PASS]
G04 | A new incorrect guess uses one attempt	status: incorrect; attempts: 5	As expected [PASS]
G05 | Repeated incorrect guess does not use another attempt	status: repeated_incorrect; attempts: 5	status: repeated_incorrect; attempts: 4 [FAIL]

Python test	Expected outcome	Recorded outcome
G060 | Invalid input '' leaves attempts unchanged	status: invalid; attempts: 4	As expected [PASS]
G061 | Invalid input '7' leaves attempts unchanged	status: invalid; attempts: 4	As expected [PASS]
G062 | Invalid input 'aa' leaves attempts unchanged	status: invalid; attempts: 4	As expected [PASS]
G07 | Completing a term ends the round as a win	status: win; word: p r e y	As expected [PASS]
G08 | Sixth distinct miss ends the round as a loss	status: loss; attempts: 0	As expected [PASS]
G09 | All 12 supplied pairs are selected once without term reuse	count: 12; unique_terms: 12; matching_pairs: true	As expected [PASS]
G10 | After all supplied terms are used the session ends cleanly	Session ends	As expected [PASS]
G11 | CLI completes rounds through replay and exits on n while another round remains	completed_rounds: 3; exit_without_exhaustion: true	As expected [PASS]
G13 | CLI completes habitat after each distinct letter and reveals both a positions	completed: true; all_matching_positions_visible: true; input_exhausted: false	As expected [PASS]
G12 | Headerless five-field CSV preserves the same 12 term/clue pairs	rows: 12; fields_each: [5]; pairs_match: true	As expected [PASS]

5. Python evaluation

The recorded Python evidence shows that most required game behaviour operates, while one repeat rule remains incorrect. Fourteen of the 15 checks pass. Input normalisation, all-occurrence reveal, repeated correct guesses, new incorrect guesses, invalid input, win and loss outcomes, non-reuse within the bounded session, command-line replay and exit, and consistency between the CSV and Python data all agree with their expected results. G05 fails: after a new incorrect guess reduces the counter to five, entering the same incorrect letter reduces it again to four. The failure is preserved in the test record; no correction or retest is claimed.

The implementation separates round state from session selection. The round stores the term, clue, guessed-letter set and attempt count. Input is stripped and case-folded before checking whether it is one alphabetic character. Blank, numeric and multi-character inputs return an invalid status without changing the state. For a new incorrect letter, one attempt is deducted; for a new correct letter, all matching positions are displayed. The six-distinct-miss test reaches a loss, and a completed term returns a win. These behaviours are supported by direct checks. The repeated-incorrect branch is the exception: it enters a separate branch but changes the attempt counter before returning, so the state no longer reflects the number of distinct incorrect letters.

The counter defect has a visible consequence. A pupil who repeats a wrong letter can lose an attempt without making a new guess, which may shorten the round and produce a loss earlier than the six-distinct-miss rule allows. The other repeated-guess check shows that a repeated correct letter does not change the counter. The source therefore contains enough evidence to localize the problem to the repeated-incorrect path. The defect is not hidden by the broader passing results, and the report does not claim that a passing test set proves all behaviour.

The session manager selects from an unused-pair collection and the recorded check confirms that all 12 pairs can be selected once without term reuse. The command-line test isolates the console interaction and shows that another round can be started and that the user can exit while a further round remains available. The corresponding session test uses a deterministic first-choice selector, so it verifies pair retention and removal, not whether random distribution is statistically balanced. No observed pupil performance is represented.

The console shows the clue, masked term, guessed letters and attempts, and uses neutral outcome language. Text-only use may remain difficult for pupils who require speech output, adjustable text or a different input method. No pupil trial or assistive-technology test is claimed. Two maintenance changes are proposed. First, the repeated-incorrect branch should return a repeat status without decrementing attempts; a regression test should record the count before and after repeating an incorrect letter at several remaining-attempt values, then rerun the full round suite. Second, the input prompt could display an example valid entry and clarify that invalid input does not use an attempt; implementation should change only prompt/help text, with checks confirming that blank, numeric and multi-character values remain penalty-free. Neither change is presented as completed.

6. Relational database

The report explains that a relational database separates clue details from game references. ClueID is selected to identify each clue and GameID to identify a game row. Games.ClueID refers to Clues.ClueID, so a game entry can be linked without copying the word and clue. SQLite is used with the supplied 12 records. This is a suitable two-table structure for the stated scope, although no queries or connection to the Python program are required.

The schema uses text values for identifiers and descriptions, an integer Difficulty and required fields. It does not enforce the allowed Subject values. The database test record shows that Ecology is accepted, but also that an unlisted value, Air, is accepted when rejection is expected. Duplicate GameID and missing ClueID are rejected. Difficulty values 0, 6 and 2.5 and a blank Word are rejected. The database remains a synthetic example and does not contain learner accounts or observed results.

Database test	Expected outcome	Recorded outcome
D01 | All 12 source rows, field names and Games key structure are present	12 exact source rows; Clues has the five required fields; Games has GameID and ClueID, with GameID as primary key.	As expected [PASS]
D08 | Suitable text/numeric types preserve every supplied value	Suitable text and integer types; all supplied values preserved without truncation.	As expected [PASS]
D02 | A game linked to an existing clue is accepted	ACCEPTED	As expected [PASS]
D03 | Duplicate GameID is rejected	REJECTED	As expected [PASS]

Database test	Expected outcome	Recorded outcome
D04 | Game referring to missing ClueID is rejected	REJECTED	As expected [PASS]
D05-0 | Out-of-range or non-integer Difficulty 0 is rejected	REJECTED	As expected [PASS]
D05-6 | Out-of-range or non-integer Difficulty 6 is rejected	REJECTED	As expected [PASS]
D05-2.5 | Out-of-range or non-integer Difficulty 2.5 is rejected	REJECTED	As expected [PASS]
D06 | Blank required Word is rejected	REJECTED	As expected [PASS]
D07 | Unknown Subject is rejected	REJECTED	ACCEPTED [FAIL]

7. Spreadsheet analysis

The workbook retains all supplied rows and includes a formula for each word's win percentage. The rates display to one decimal place and the complete rows are sorted ascending. The totals are calculated as 435 games played and 273 wins. The final summary formula is SUM(Games won)/SUM(Games played), which returns 62.7586…%, displayed as 62.8%. The required result is the unweighted average of the 12 individual percentages, 61.6362…%, displayed as 61.6%. The two values differ because words have different numbers of attempts. The response does not label the weighted result as a substitute or explain the mismatch, and the defect remains in the submitted workbook.

The review area identifies habitat, turbine, predator and renewable with their rates, counts and difficulty values, and recommends checking their clues before ratings are changed. The limitation explains that combined results cannot identify why one word is difficult. The recommendation and limitation are present, though the discussion does not fully connect the difference between weighted and unweighted calculation to the interpretation of the 12 terms.

8. Accompanying evidence

Candidate/Supporting artifacts/SYN02/ contains the Python program and supplied term data, the SQL and SQLite database, the completed analysis workbook, the source data and the recorded checks. Paths are listed in the README file for the response. The editable files form part of this response and should be reviewed alongside the report.

IQM3010 A3 Candidate Coursework Brief, Set S of 2026, supplied with the sample assessment.

IQM3010 A3 supplied EcoQuest term, clue and test-data resources.

Generated Python source and test records included with this synthetic response.

Generated SQLite schema and validation records included with this synthetic response.

Synthetic authorship declaration: This report, code, database, workbook and test record were generated as synthetic staff practice material. They do not represent candidate-authored work, a learner's AI declaration, a signed submission, an observed supervised session or academic approval.

Response SYN03

Synthetic coursework report and evidence record

The project addresses IQM3010 Computer Science Assessment 3, Set S of 2026, for EcoQuest Learning. The client requires a text-based environmental terminology game for GCSE pupils, a relational database and an Excel analysis of supplied practice results. The three products are considered separately, with a short report describing their design, implementation and testing.

1. Analysis

EcoQuest Learning requires a small environmental-science word game for GCSE pupils. A clue and hidden term are shown, and a pupil enters letters to reveal the answer. The game uses the supplied practice terms. A database stores clue and game information, while a spreadsheet shows percentages from test results; these are separate products.

The game accepts letters, reduces an attempt counter for incorrect answers and finishes when the term is guessed or the counter reaches zero. A clue, masked term and result are displayed in a console. The program is intended to provide quick practice, although text must be read and entered. The database keeps clue records and game records in separate tables. The spreadsheet adds a win percentage and helps identify terms with different results. The main success conditions are that the game can be played, information can be stored and percentages can be viewed. Development is constrained by the console format and available coursework time. The supplied results are treated as practice data rather than records of individual pupils.

2. Design

A GameRound class holds a word, clue and attempt counter. A session class holds the term list and starts a round. The console displays a clue and hidden word, accepts a letter, updates the display and continues until the word is guessed or no attempts remain. The result is shown and another round can be selected.

Pseudocode:
Choose a term and its clue.
Set the attempt counter to six.
Display the clue and hidden word.
While the word is not guessed and attempts remain:
  Read a letter.
  If the letter is in the word, show it.
  Otherwise reduce the attempt counter.
Display whether the word was guessed or attempts reached zero.
Ask whether another round should begin.

Basic checks are planned for one correct letter, one incorrect letter and replay.

3. Python implementation

The native code defines a GameRound class and a session manager. A single correct letter can be processed, and a new incorrect letter reduces the counter. Invalid inputs are rejected. Two important behaviours fail. The revealed method displays only one occurrence of a guessed letter, so guessing a in habitat produces one visible a rather than both. The session manager selects from the complete term list every time and does not remove or record used pairs. With the deterministic test choice, the same term starts repeatedly, and the session never reaches the all-terms-used exit. These failures are present in the generated files and are not described as corrected.

4. Recorded testing

The generated game checks recorded 11 of 15 expected outcomes as passing. Input normalisation, repeated correct guesses, new incorrect guesses, repeated incorrect guesses, invalid strings, a win and a loss passed. G02 failed because only one matching a is revealed in habitat. G09 failed because a used term was selected again. G10 failed because the session did not end after all 12 supplied terms had been consumed. G13 failed because the actual console loop did not complete habitat after each distinct letter and exhausted the bounded input. These observed results are shown without a claimed correction.

The database checks recorded 4 of 10 expected outcomes as passing. The full source rows, required field names, Games fields and GameID key were confirmed. SQLite TEXT fields preserve the supplied text without truncation. However, an orphan ClueID, Difficulty 0, 6 and 2.5, blank Word and unlisted Subject were accepted. The workbook checks found no formula in the 12 Win percentage cells, an incorrect ascending order, typed totals and a weighted mean. The recommendations and limitation are present but weakly supported. All test data and outputs are synthetic executions; no learner activity or observed marking event is represented.

The following expected and actual results are recorded from the supplied synthetic implementation. Complete machine-readable records and test scripts accompany the native files.

Python test	Expected outcome	Recorded outcome
G01 | Round starts with six attempts and no revealed letters	attempts: 6; word: _ _ _ _ _ _ _	As expected [PASS]
G02 | Case and surrounding spaces are normalized; all matching letters reveal	status: correct; attempts: 6; word: _ a _ _ _ a _	status: correct; attempts: 6; word: _ a _ _ _ _ _ [FAIL]
G03 | Repeated correct guess gives feedback without penalty	status: repeated_correct; attempts: 6	As expected [PASS]
G04 | A new incorrect guess uses one attempt	status: incorrect; attempts: 5	As expected [PASS]
G05 | Repeated incorrect guess does not use another attempt	status: repeated_incorrect; attempts: 5	As expected [PASS]
G060 | Invalid input '' leaves attempts unchanged	status: invalid; attempts: 5	As expected [PASS]
G061 | Invalid input '7' leaves attempts unchanged	status: invalid; attempts: 5	As expected [PASS]
G062 | Invalid input 'aa' leaves attempts unchanged	status: invalid; attempts: 5	As expected [PASS]
G07 | Completing a term ends the round as a win	status: win; word: p r e y	As expected [PASS]
G08 | Sixth distinct miss ends the round as a loss	status: loss; attempts: 0	As expected [PASS]
G09 | All 12 supplied pairs are selected once without term reuse	count: 12; unique_terms: 12; matching_pairs: true	count: 12; unique_terms: 1; matching_pairs: true [FAIL]
G10 | After all supplied terms are used the session ends cleanly	Session ends	habitat [FAIL]

Python test	Expected outcome	Recorded outcome
G11 | CLI completes rounds through replay and exits on n while another round remains	completed_rounds: 3; exit_without_exhaustion: true	As expected [PASS]
G13 | CLI completes habitat after each distinct letter and reveals both a positions	completed: true; all_matching_positions_visible: true; input_exhausted: false	completed: false; all_matching_positions_visible: false; input_exhausted: true [FAIL]
G12 | Headerless five-field CSV preserves the same 12 term/clue pairs	rows: 12; fields_each: [5]; pairs_match: true	As expected [PASS]

5. Python evaluation

The generated Python checks show that several basic round behaviours operate, while repeated-letter completion and session selection remain incomplete. Eleven of the 15 recorded checks pass. A normalised single-letter input is accepted, repeated correct and incorrect guesses do not reduce attempts, new incorrect guesses do, invalid entries leave the counter unchanged, and a six-miss loss and a completed non-repeating term win are recorded. Separate command-line checks confirm that replay and exit prompts can be used when the console is supplied with rounds. G02 and G13 fail: one a appears in habitat, and the console round does not complete after all distinct letters are entered. G09 fails because the same term is selected again instead of removing a used pair, and G10 consequently fails because the session cannot reach the empty-deck state. These outcomes remain unresolved.

The round class stores the target, clue, guesses and attempt count. Input is trimmed and case-folded, and invalid input is rejected before state changes. New correct and incorrect guesses update the state, and repeated guesses are recognized without penalty. The defect lies in the reveal routine: it removes a guessed letter from a temporary set after the first matching position. As a result, the visual mask does not reflect every matching letter in the term. The completion check compares the displayed mask with the complete term, so a repeated-letter term can remain unfinished even after every distinct letter has been guessed. The specific test evidence supports a deduction for reveal and completion behaviour, while the other tested input and attempt rules remain credited.

The session manager has an unused-list attribute, but its selection method chooses from the complete term list and does not remove the chosen pair. The bounded deterministic test therefore returns the same term repeatedly. The command-line replay/exit check uses a separate test double for round selection so that the console prompts can be assessed independently; it does not prove that the submitted manager supports replay without reuse. The session-selection and empty-deck failures are recorded separately. No additional deduction is made for exhaustion beyond the same underlying non-reuse defect. The CSV comparison confirms that the supplied 12 term/clue pairs are preserved in the headerless five-field data file.

The console provides a clue, mask, guessed letters and remaining attempts, and it reports a win or loss with the correct term for tested outcomes. A text-only format may not meet the needs of users who require speech output, larger text or alternative input. No user trial, supervised activity or accessibility validation is represented. The current artifact is therefore a partial practice prototype rather than an evaluated learning resource. The synthetic data do not support claims about attainment or pupil performance.

Two realistic maintenance changes are identified. First, the reveal method should test each term position against the complete guessed-letter set instead of consuming a letter after the first match; the regression check should guess each distinct letter in habitat, record the mask after each step and confirm that both a positions appear. Second, the session manager should choose and remove a pair from the unused list; a fixed-order check should confirm all 12 term/clue associations are returned once, that the next request reports no available round, and that a newly created session starts with all pairs available. These are proposed changes only; the code and test record do not claim they were applied.

6. Relational database

A relational database stores records in tables and can use identifiers to refer to related rows. The response uses Clues and Games and selects IDs as primary keys. The database contains all 12 clue rows. Games has the required GameID and ClueID fields and a GameID primary key, but Games.ClueID is only a text field; it is not linked to Clues.ClueID. Consequently, a game row can contain an identifier that does not exist. The validation record also shows that Difficulty values outside 1–5 and decimal values are accepted, blank Word values are accepted, and Subject is not controlled. The row count, valid game insertion and duplicate GameID checks pass; the separate console check covers replay and exit independently of the session manager. The database should therefore be treated as a partial implementation.

Database test	Expected outcome	Recorded outcome
D01 | All 12 source rows, field names and Games key structure are present	12 exact source rows; Clues has the five required fields; Games has GameID and ClueID, with GameID as primary key.	As expected [PASS]
D08 | Suitable text/numeric types preserve every supplied value	Suitable text and integer types; all supplied values preserved without truncation.	As expected [PASS]
D02 | A game linked to an existing clue is accepted	ACCEPTED	As expected [PASS]
D03 | Duplicate GameID is rejected	REJECTED	As expected [PASS]
D04 | Game referring to missing ClueID is rejected	REJECTED	ACCEPTED [FAIL]
D05-0 | Out-of-range or non-integer Difficulty 0 is rejected	REJECTED	ACCEPTED [FAIL]
D05-6 | Out-of-range or non-integer Difficulty 6 is rejected	REJECTED	ACCEPTED [FAIL]
D05-2.5 | Out-of-range or non-integer Difficulty 2.5 is rejected	REJECTED	ACCEPTED [FAIL]
D06 | Blank required Word is rejected	REJECTED	ACCEPTED [FAIL]
D07 | Unknown Subject is rejected	REJECTED	ACCEPTED [FAIL]

7. Spreadsheet analysis

The workbook contains all 12 supplied records and a Win percentage column. The percentages and summary totals were entered as values. The original row order was retained. Total games played is 435 and total games won is 273. The reported mean is 62.8%, calculated as the overall wins-to-games ratio. Two words are proposed for review: habitat has a 25.0% win rate and difficulty 1, and turbine has a 30.0% win rate and difficulty 3. The habitat note states “Could be easier”, and the turbine note states “Review the clue”. These notes are recorded in the workbook. The results combine several users and cannot show why one word is difficult; more pupil feedback would help.

8. Accompanying evidence

Candidate/Supporting artifacts/SYN03/ contains the Python program and supplied term data, the SQL and SQLite database, the completed analysis workbook, the source data and the recorded checks. Paths are listed in the README file for the response. The editable files form part of this response and should be reviewed alongside the report.

IQM3010 A3 Candidate Coursework Brief, Set S of 2026, supplied with the sample assessment.

IQM3010 A3 supplied EcoQuest term, clue and test-data resources.

Generated Python source and test records included with this synthetic response.

Generated SQLite schema and validation records included with this synthetic response.

Synthetic authorship declaration: This report, code, database, workbook and test record were generated as synthetic staff practice material. They do not represent candidate-authored work, a learner’s AI declaration, a signed submission, an observed supervised session or academic approval.