## Task 2 – Developing the solution

**The solution**

**Indicative content and marker guidance**

Students demonstrate use of code to produce a solution that includes front-end and back-end processes to meet the needs of **GibJohn Tutoring**.

**Functionality:**

- Must demonstrate the use of two different languages, e.g. JS and PHP or SQL. Python and SQL etc.
- Use of logic and programming structure should be appropriate for the chosen language such as:
  - Modularisation
  - Appropriate interrelation of arts pf the program e.g. modules pass the correct data between each other, front-end and back-end processes have sufficient separation.
  - Suitable use of programming structures (sequence, selection and iteration) to solve problems in an efficient manner.
  - complex data models in database interlinking more than one table
  - Web service APIs and parsing JSON/XML to service a complex client- server model
- Examples of efficiency may include:
  - Avoidance of very linear structures to more logically and efficient approach e.g. the use of function and procedures
  - OOP design features (classes, inheritance etc.)
  - Recursive algorithms – reminder that if used in PHP here must be a mechanism (IF statement, etc.) that stops the recursion after the desired result has been found
  - Loading data only when necessary

Note – data normalisation may not be required depending on the identified/proposed solution

The program should ensure correct outcomes to meet the needs of **GibJohn tutoring** for example:

- Handing user access e.g. Setting up an account, logging-in, changing password
- Collection and processing of data e.g. registration, tracking student marks, student progress through tasks
- Communication/data exchange between different platforms or between front-end and back-end systems e.g. analysing students marks to make recommendations, pulling in assessment data to personalise quizzes
- Key calculations e.g. Student log-in/study time, cost of tutoring.
- Data filtering and visualization e.g. Student and tutor performance, usage stats

**Code organisation:**

- Students should avoid multiple pages of nested if clauses and unnecessary repeated code that could more effectively be implemented as a called function
- Clear and meaningful indentation
- Code should consist of pieces of logic, classes, or objects, with proper structure.
- Comments should be used wherever possible to help explain the logic
- Good use of local variables and minimal use of global variables
- Use of constants
- consistent style throughout

**User Experience considerations:**

- Inputs form the user are handled appropriately such as:
  - invalidation and sanitization
  - ease of user input e.g. GUI features, input masks
  - well-designed interface including UI/UX feaures:
    - Useful – does it meet the needs of the user? Are outputs etc. accurate
    - Usable – is it easy to use / intuitive?
    - Desirable – is it aesthetically pleasing? Does it represent a brand? Consistency.
    - Findable – is it easy to navigate around and find the required
- user guidance and error messages ensure that the product is easy to use and helpful which may include:
  - Short and meaning full error messages (avoid jargon)
  - Provides information about solution/corrective action that can be taken by the user
  - Help/instructions and/or meaningful feature labels to reduce errors
- A robust solution which may include:
  - good exception handling that can deal with unexpected or incorrect outputs
  - "fall back" code to deal with different systems and platforms without crashing
  - Appropriate use of validation

**Legal and regulatory guidelines and Standards**

- Accessibility has been considered and there should be evidence that their design takes different users in to account such as:
  - Selectable accessibility features (e.g. adjustable fonts, colours etc.)
  - Sensible font and colour base colour selection
- Compatibility with different platforms has been considered (where appropriate) which may include:
  - Creating a standalone/self-contained program
  - Implementation of web-standards / program is accessible through a web browser
  - "fall back" code to deal with different systems and platforms
- legal and ethical considerations considered as appropriate to their solution which may include:
  - Cookie consent notice
  - GDPR considerations
  - Privacy and data policy/informing the user
  - Intellectual property consideration (see asset log)
  - Terms and conditions

| Assessment focus | Band 0 | Band 1 | Band 2 | Band 3 | Band 4 |
|---|---|---|---|---|---|
| | 0 | 1-2 | 3-4 | 5-6 | 7-8 |
| Functionality | No rewardable material | The prototype implements code in a single language with some functionality but the code lacks efficiency and some major errors persist.<br>Uses some precise logic and programming structures which would result in some correct outcomes | The prototype implements code with some functionality in at least two different languages but the code lacks efficiency and some major errors persist.<br>Uses sufficient precise logic and programming structures which would result in adequate correct outcomes. | The prototype implements mostly efficient functional code in at least two different languages, but some minor errors still persist.<br>Uses mostly precise logic and programming structures which would result in mostly correct outcomes. | The prototype implements consistently efficient functional code in at least two different languages.<br>Uses precise logic and programming structures throughout which would result in consistently correct outcomes. |
| | | 1-2 | 3-4 | 5-6 | 7-8 |
| Code organisation | | Code is maintainable by a third party, but would present significant difficulties through the use of:<br>• inconsistent naming conventions.<br>• limited logical organisation<br>• limited informative commenting. | Code is maintainable by a third party, but would present some difficulties through the use of:<br>• somewhat appropriate naming conventions.<br>• Some logical organisation<br>• some informative commenting. | Code is maintainable by a third party, and would present only a few minor difficulties through the use of:<br>• mostly appropriate naming conventions.<br>• mostly logical organisation<br>• mostly informative commenting. | Code is easily maintainable by a third party through the use of consistently appropriate:<br>• naming conventions.<br>• logical organisation<br>• informative commenting. |
| | | 1-2 | 3-4 | 5-6 | 7-8 |
| User experience | | Basic user experience is provided through limited effective use of:<br>• input handling<br>• user guidance and error messages<br>• outputs<br>The solution is partially robust and effectively handles some common errors | Adequate user experience is provided through somewhat effective use of:<br>• input handling<br>• user guidance and error messages<br>• outputs<br>The solution is adequately robust and effectively handles sufficient common and unexpected errors | Good user experience is provided through mostly effective use of:<br>• input handling<br>• user guidance and error messages<br>• outputs<br>The solution is largely robust and effectively handles most common and unexpected errors | Excellent user experience is provided through consistently effective use of:<br>• input handling<br>• user guidance and error messages<br>• outputs<br>The solution is fully robust and effectively handles common and unexpected errors |

| | | 1-2 | 3-4 | 5-6 | [redacted] |
|---|---|---|---|---|---|
| **Legal and regulatory guidelines and Standards** | | Some effective application of standards and guidelines in relation to:<br>● accessibility<br>● compatibility<br>● legal and ethical considerations<br>Some effective application of procedures and security controls to ensure confidentiality, integrity and availability. | Mostly effective application of standards and guidelines in relation to:<br>● accessibility<br>● compatibility<br>● legal and ethical considerations<br>A mostly effective application of procedures and security controls to ensure confidentiality, integrity and availability. | Consistent and effective application of standards and guidelines in relation to:<br>● accessibility<br>● compatibility<br>● legal and ethical considerations<br>Thoroughly effective application of procedures and security controls to ensure confidentiality, integrity and availability. | [redacted] |

## Task 2 – Developing the solution

### Testing

**Indicative content and marker guidance**

**Test data**

- Suitable test data should be demonstrated (as appropriate to the student’s solution) which may include:
  - **test-specific data**: influence the system behaviour and reveal the case specifics under the test
  - **test-reference data**: have little influence on the test performance
  - **application reference** data: irrelevant to the behaviour under test, and are needed to start the application
  - **valid test data** does the system functions are in compliance with the requirements, does the system processes and stores the data as intended
  - **invalid test data**: check to see if the software correctly processes invalid values, shows the relevant messages, and notifies the user that the data are improper
  - **boundary test data**: help to reveal the defects connected with processing boundary values
  - **wrong data**: entering the data of inappropriate format, whether it shows the correct error messages thus showing the use of validation if appropriate
  - **absent data**: should check that the solution handles entering a blank field

**Use of testing to inform the iterative development process**

- Learners should demonstrate how testing was used throughout the development to test and refine the solution. This may include reference to:
  - Runtime errors that occurred when implementing code
  - Fixing of syntax errors
  - How logical errors were detected through the use of appropriate test data
  - Descriptions of how issues were fixed
  - Refinements made in the code to mitigate issues
  - Details of regression testing
  - Comparison of manual calculations against program calculations to ensure logic and outputs are correct
  - Testing around boundaries to ensure logic of validation is correct

| Assessment focus | Band 0 | Band 1 | Band 2 | Band 3 |
|---|---|---|---|---|
| | 0 | 1-2 | 3-4 | 5-6 |
| **Suitability of test data** | No rewardable material | Tests selected show a basic understanding of how to effectively test inputs, calculations, validation and processes using test data which makes limited use of:<br>• normal data<br>• erroneous data<br>• extreme data | Tests selected show a good understanding of how to effectively test inputs, calculations, validation and processes using test data which includes:<br>• normal data<br>• erroneous data<br>• extreme data | Tests selected show a thorough and detailed understanding of how to effectively test inputs, calculations, validation and processes using test data which includes:<br>• normal data<br>• erroneous data<br>• extreme data |
| | | 1-2 | 3-4 | 5-6 |
| **Use of testing to inform the iterative development process** | | Comments show a basic understanding of how errors/problems were identified and how they were rectified (as appropriate) for:<br>• inputs<br>• calculations<br>• validation and processes<br>Testing shows evidence of a basic iterative development process | Comments show a good understanding of how errors/problems were identified and how they were rectified (as appropriate) for:<br>• inputs<br>• calculations<br>• validation and processes<br>Testing shows evidence of a good iterative development process | Comments show a comprehensive understanding of how errors/problems were identified and how they were rectified (as appropriate) for:<br>• inputs<br>• calculations<br>• validation and processes<br>Testing shows evidence of an effective iterative development process |