# CODELAB II

## ASSESSMENT 2: Data Driven Application

| | |
|---|---|
| Contribution towards overall module mark | 60% |
| Date set | January 10, 2026 |
| Marked work returned by | February 10, 2026 |
| DEADLINE DATE | January 15, 2024 |

# Assessment 2: Data Driven Application

**The Brief**

For this assignment you are tasked with developing an application that makes use of data retrieved from an API. Your application should demonstrate the use functions, and pass arguments between these as appropriate. Where possible, the application should also implement object oriented programming techniques.

The final application should be delivered via a functioning interactive GUI built using Tkinter. This GUI should allow the user to interact via mouse and/or keyboard input.

You are required to use one of the APIs listed below:

- The Cocktail Database
- The Sports Database
- The Audio Database
- The Meal Database
- The Movie Database

How you use the chosen API and the functionality you provide within your application is up to you. Making creative use of the API and/or providing the user with a range of options to query the API will likely afford higher marks. You should explore the documentation for each API to see what is possible in order to help select the API you wish to work with.
As noted above the application should be built using the Tkinter library module.

**Assignment setup**

You should use the GitHub assignment repository to keep track of your code as you develop your solution and submit your work to GitHub.

**Deliverables**

The deliverables for the Data Driven Application are as follows:

The Application

The Python code and any supplementary assets (images, sounds, fonts) required to run your data driven application.

The Development Document

Your data driven application must be accompanied by a Development Document of 1000 words. This development document should consist of the following elements:

- *Abstract:* A short description of your application including which API you have selected. You should also provide a link to your Github repository.
- *Project Plan:* Project plan that breaks the project down into key milestones (e.g planning, design, development & testing) and tasks. This plan should allocate the estimated time to complete each of the listed tasks.
- *Evidence of design:* This may include but is not limited to: specification list of your program requirements, Flowchart, Pseudo-Code, Wireframes, UML Data Structure Diagram. A minimum expectation for this section is the inclusion of a wireframe for the design of the GUI.

- *Technical Description & Walkthrough:* In this section you should provide a link to a video which includes a walkthrough of your program running as well as a technical breakdown of your code. This technical breakdown should explain how the key features of your program have been implemented via code. The video technical description & walkthrough contributes to the overall word count. The anticipated length of the video is 3-5 minutes.
- *Testing:* Test plan and evidence of testing (e.g. screenshots and/or videos). The minimum expectation for this section is a testing table complete with test cases covering the core functionality of your app.
- *Critical Reflection:* An open and detailed evaluation of your application that notes what is compelling about the work, what could be improved, and what you need to learn to make these improvements.
- *Appendix:* A copy of your code should be included in an appendix at the end of your documentation. To provide this simply copy and paste the code from each of your project files (do not provide the code in screenshots).

**Submission**

Please follow the submission instructions below. Work that is submitted incorrectly may not be accepted or could incur a points penalty.

Before submitting have you…

- Spell-checked and grammar-checked your work? Please make an appointment with the Writing and Learning Centre or speak to your tutor if you are experiencing challenges in this area.
- Formatted your written work to the specification below?

- Referenced all sources of information accurately? Please refer to www.citethemrightonline.com for guidance.

Your development document must be submitted via Turnitin and code submitted via your GitHub repository. Please adhere to the following method:

- Check your Data Driven Application is working as expected.
- Commit and push your final version to your unique Github repository for this assignment. Ensure you push all the files required to run your application. Only code committed to your GitHub assignment repository will be marked.
- Ensure the link for your GitHub assignment repository is included within the abstract of the development document.
- Check you have completed all the required sections of the development document.
- Save your development document as a Word document.
- Log into Minerva, go to the Assessment tab and submit your finalised document via the appropriate Turnitin Link.

**Format**

All written work must conform to university styling and submission guidelines. They must:

- Contain appropriate in-text citation that supplies an accurate list of references.
- Be accurate in referencing. See Bath Spa guidelines and the Harvard system described at www.citethemrightonline.com.
- Be accurate in spelling and paragraphing.

**Marking Criteria**

Assignment 2: Data Driven Application will be marked against the following criteria:

1. System Design & User Experience (15%)
2. Evidence of Testing (15%)
3. Documentation (20%)
4. Technical Implementation (50%)

| Criteria | Weighting | Mark Range Description | Mark Range |
|---|---|---|---|
| System Design & User Experience | 15% | A very limited design that may not be implementable. User experience is unclear and confusing to use. | 0 - 19 (Low Fail) |
| | | A poor system design that omits key features and the resulting application may not be fit for purpose. User experience is confusing and may not | 20 - 39 (Fail) |

| | | | |
|---|---|---|---|
| | | include all information required to navigate the application. | |
| | | A basic design with only limited options for querying the API. User experience is basic and some key information may be missing. Minimal attention given to user interface design. | 40 - 49<br>(Third) |
| | | A fair design with some variety in the queries used but these are limited in depth or creativity. User experience provides key information but attention to detail may be missing. A fair design that implements basic graphical elements. Look and feel is only loosely considered. | 50 - 59<br>(2:2) |
| | | A good design that includes a range of queries or filters for accessing the API. User experience is good with clear accurate information presented throughout. User interface is well designed though some refinement may be necessary. | 60 - 69<br>(2:1) |
| | | Very good system design that provides a wide range of queries for accessing the API some of which are creative in their approach. User experience is very good and provides clear and detailed information to the user. The user interface is visually appealing with very good use of graphical elements. | 70 - 79<br>(First) |

| | | | |
|---|---|---|---|
| | | Excellent system design that includes many highly original and sophisticated queries for accessing the API. User experience is compelling with intuitive and visually appealing user interface. | 80 - 89<br>(High First) |
| | | Beyond expectations for this level of study. | 90 - 100<br>(Outstanding) |
| Evidence of testing | 15% | No evidence of testing. | 0 - 19<br>(Low Fail) |
| | | Very limited evidence of testing that does not include a test plan or only provides minimal information on what has been tested in the application. | 20 - 39<br>(Fail) |
| | | Limited test plan that may omit the testing of key features. Evidence of testing may be missing (e.g. screenshots or videos). Testing has not been used to modify or improve the final solution. | 40 - 49<br>(Third) |
| | | A fair test plan with the core functionality tested and some evidence of testing is provided (e.g. screenshots or videos). There is a lack of detail on how the testing has been used to modify or improve the final solution. Independent user testing may have been completed yet is limited in scope. | 50 - 59<br>(2:2) |
| | | A good test plan that goes beyond just testing the core functionality. Evidence of testing is good (e.g. screenshots or videos). There is a | 60 - 69<br>(2:1) |

| | | | |
|---|---|---|---|
| | | good level of detail on how testing has been used to modify or improve the final solution. Some independent user testing has been conducted. | |
| | | Comprehensive test plan and test data with clear evidence of testing (e.g. screenshots or videos. Multiple iterations of testing may have been conducted resulting in a very good level of detail on how testing has been used to modify or improve the final solution. Several independent user tests have been conducted. | 70 - 79<br>(First) |
| | | Highly comprehensive test plan and test data with multiple iterations and clear evidence of testing (e.g. screenshots or videos. Detail on how testing has been used to affect the final solution is extensive. Several rigorous independent user tests have been conducted. | 80 - 89<br>(High First) |
| | | Beyond expectations for this level of study. | 90 - 100<br>(Outstanding) |
| Documentation | 20% | Very limited documentation that demonstrates little or no understanding of the design and build process. Some sections are missing and presentation is unacceptable | 0 - 19<br>(Low Fail) |
| | | A poor attempt that does not meet the requirements of the development document. Likely missing key sections or poorly presented. | 20 - 39<br>(Fail) |

| | | | |
|---|---|---|---|
| | | Basic documentation that offers only minimal information about the design direction and technical implementation. Correct use of technical terminology is lacking and some sections require greater depth. Presentation is may be lacking attention to detail | 40 - 49<br>(Third) |
| | | A fair attempt that provides only key information about the design and technical implementation. Limited in the clarity of the technical description (e.g may include some inaccuracies). Some sections may require further detail. Presentation is acceptable with few issues. | 50 - 59<br>(2:2) |
| | | A good to very good development document that provides a clear overview of the design direction and a sound description of the technical implementation. All sections have been completed to a sufficient level. Presentation is good although it may have some minor issues. | 60 - 69<br>(2:1) |
| | | Very good overview that provides detailed description of the concept design and technical implementation. All sections are complete to a very good level of detail and the presentation is without error. | 70 - 79<br>(First) |
| | | An excellent overview that provides a high level of insight into the design and technical implementation. | 80 - 89<br>(High First) |

| | | | |
|---|---|---|---|
| | | Reflection is highly critical and detailed. Presentation is flawless. | |
| | | Beyond expectations for this level of study. | 90 - 100<br>(Outstanding) |
| Technical Implementation | 50% | A very limited implementation that demonstrates little evidence of programming skill. | 0 - 19<br>(Low Fail) |
| | | A poor implementation that demonstrates a limited understanding of the brief, and may contain significant errors. | 20 - 39<br>(Fail) |
| | | A basic implementation that attempts to connect to the selected API, but may contain errors. Functionality may not be fully implemented via an openFrameworks GUI (e.g. returned data output on the console) Minimal commenting in the code. No / minimal use of functions | 40 - 49<br>(Third) |
| | | A fair implementation that successfully connects to the selected API. Functionality fully delivered via an interactive GUI although the overall functionality may be limited. Code has some commenting but this is limited in clarity. Programme uses functions but there is scope for improved structure and logic (e.g. greater use of parameters). | 50 - 59<br>(2:2) |
| | | A good implementation that successfully connects to the selected API. Functionality fully delivered via an | 60 - 69<br>(2:1) |

| | | | |
|---|---|---|---|
| | | interactive GUI with a range of functionality afforded by the program. Range of programming skills are showcased including good use of functions. OOP techniques beyond the standard openFrameworks classes may have been implemented, though there is scope for further extension of these techniques. Coding Conventions have been observed (comments, indentation, camelCase etc), with only minor errors present | |
| | | A very good implementation that successfully connects to the selected API. Program functions without error and demonstrates very good exploration and implementation of the selected API. Wide range of programming skills are showcased including very good use of functions to improve code reuse and efficiency. OOP techniques are implemented beyond the standard Openframeworks classes to a good standard. Additionally the implementation may include techniques that go beyond the class materials. Code is commented and structured to a high standard and coding conventions have been well observed. | 70 - 79<br>(First) |
| | | An excellent implementation that deploys techniques well beyond the scope of those demonstrated in class. Programme successfully integrates with the selected API with the user | 80 - 89<br>(High First) |

| | | | |
|---|---|---|---|
| | | given access to multiple potential queries that demonstrates a range of data available via the API. Commenting is detailed and coding conventions have been observed with precision. Wide range of programming skills are showcased including sophisticated use of functions to improve code reuse and efficiency. OOP techniques are implemented beyond the standard Openframeworks classes to a very good standard. | |
| | | Beyond Expectations for this level of study | 90-100 (Outstanding) |

**Intended Learning Outcomes (ILOs)**

| ILO | Assessed |
|---|---|
| The implementation and testing of a prototype system that is driven by object-oriented programming techniques. | ✓ |
| Application of the iterative design cycle of prototyping, testing, analysing and refinement. | ✓ |
| A recognition of personal knowledge limits, which is addressed through the identification of learning opportunities. | ✓ |
| An ability to critically review the key features and challenges of developing software for an end user, and evaluate their relevance to the field of creative computing. | |

Mark penalties may be applied to late submissions without prior approval of extension. Please ensure that you prepare and submit your work in good time to allow for any issues that may arise.