Assessment Task 2: Client's Briefing #1 Transcript

Hello, I'm from *QUT Entertainment Systems*. My twin brother at *QUT Data Services* told me about the sterling job you did on their data visualisation project, so I want to engage you to help with our new software development project.

With COVID-19 restrictions now easing people are starting to go out again and our firm wants to capitalise on this. We want you to help us develop a new application for selling tickets to live entertainment events. It must have a simple Graphical User Interface and a reliable back-end which extracts data about upcoming events from online sources.

We have five weeks to complete the project, but you need to start straight away because there's a lot to do. I'm aware that you've been attending our company's training courses and you already have the skills needed to complete the first part of the project which is due in two weeks from now (Sunday, May 15th, the end of Week 9).

In this initial stage we want you to focus on the user interface so that we have something to show to our investors. We need you to design and implement a suitable graphical user interface using Python 3's Tkinter GUI module. The program must be portable across multiple computing platforms, so you must use standard Python 3 modules only; extension modules that must be downloaded and installed separately are not allowed.

I've spoken to our back-room boffins and they've created a demo to illustrate the idea. They've chosen to call it "Hiram's Live Entertainment" and have included an appropriate logo for the app.

[*Runs the user interface demo*]

The GUI is intended to give its user access to three different venues for live events. Having selected one, the user then has four possible options. The first is to show the next event coming up at the chosen venue in the GUI. The event's name and date will appear in the text area at the bottom. The other options are to display the full details of the event, print a ticket for the event, or make a permanent record of the booking. I'll say more about all these requirements in my next briefing. For now we just want you to design the user interface.

You *don't* necessarily need to copy our boffins' example, but you must design and implement your own graphical user interface with equivalent capabilities, using Tkinter. Specifically:

- Your GUI's window must have a title which identifies your application.
- Within the window there must be an image and a name which clearly identify your application. The name and image can be separate widgets or can be combined into one, as per our demo.
- There must be one or more widgets that allow the user to select between three distinct live entertainment venues. Our boffins have used radio buttons for this purpose but there are other kinds of widget that could be suitable for the job, such as menus, lists or push buttons.
- There must be one or more widgets that allow the user to perform any of the four options for the currently-chosen venue:
    1. Showing the next event at the chosen venue in the GUI.

2. Displaying full details of the upcoming event.
3. Printing a ticket for the event.
4. Permanently saving details of the event.

Our boffins have used four push buttons for this purpose but there are other kinds of widget that could be suitable for the job, such as listboxes or checkbuttons. In particular, it is *not* necessary to have an explicit "Show event" option, because showing the event could be done automatically when the venue is chosen. We'll leave the precise mechanisms by which the user interacts with the app up to you, and we welcome innovative solutions.

- There must be one or more widgets suitable for displaying (at least) the name and date of the next event coming up at the chosen venue. In the demo our boffins have allowed space for these things in the big text area at the bottom. You should make it clear what this widget is for by putting in some text to indicate what will be displayed there in the final version.
- You're free to choose whatever names, graphics, labels and layout you like, as long as the GUI's operation is completely clear to the user, without the need for any separate instructions.
- Finally, your solution must be portable across different computing platforms (Windows, macOS and Linux) as far as possible. The boffins tell me that for this reason you should use Tkinter's "grid" geometry manager and *not* the "place" manager.

For now you don't need to implement any of the application's back-end functionality. None of the widgets in our boffins' demo actually do anything as yet.

[*Activates the various widgets in the GUI to show that they do nothing*]

Nevertheless, even though the GUI doesn't do anything as yet, you must make it completely clear how each part will work. In particular, note that we've included "greyed out" text in the text box down the bottom to explain what will appear there when the program is completed.

You don't need to make a final decision about which entertainment venues you will use as yet. Our boffins have chosen three venues which look promising because information about them can be found online easily but this may change if these web sites prove unsuitable to work with later on.

I'll provide full requirements for the live entertainment web sites and what you need to extract from them in my next briefing. However, if you want to make an early start while you're working on the GUI, you'll need to find three entirely separate sites, with regularly-updated details of upcoming events. For the latest upcoming event the sites must always provide three things:

1. The event's name
2. Its date (or dates)
3. A photo or image of the event

The sites must be ones that are updated regularly, preferably daily but certainly no less often than weekly. Sites which *may* be suitable for this task include the home pages of entertainment venues such as live music, comedy and drama theatres, sporting stadia, showgrounds, commercial or government entertainment centres, and so on. There are also many aggregation sites that list upcoming events in specific geographic areas that may be suitable.

To get you started we've provided you with a program template. For cyber security reasons, you must identify yourself at the beginning of the program.

[*Shows the template*]

This template contains very little, just some "import" statements and a single function definition. You don't need this function at all for your current task, so you can ignore it for now. All of your code must be developed at the very end of the file.

You must upload your submission to our corporate IT system before the deadline, which is Sunday, May 15th (the end of Week 9). Please upload multiple drafts as you develop your solution as insurance against system failures (at your end and ours).

You'll need to submit both a Python program and any image files you have used to support your GUI. Put the image files in the same folder as your Python code so that we don't have to worry about file path representations across different computing platforms. Put all the items needed to support your GUI in a "zip" archive and upload it as a single file. Do *not* use any other compression formats such as RAR or 7Z. To create a zip archive on a Mac simply right-click on the folder and select "Compress". Under Microsoft Windows select "Send to" a compressed archive.

So that it will work on any platform, your program must be portable and must run in a standard Python 3 environment. It must **not** rely on any modules that need to be downloaded and installed separately, such as "PIL" or "Pillow", because we won't be able to run your solution.

That's all for now. I'll provide you with the full commercial-in-confidence details of the necessary back-end functionality after you've delivered your user interface design.