Assessment Task 2: Client's Briefing #3 Transcript

Hello, it's your client again, back for one last briefing. When we last met I ran out of time to tell you about the final two features in the "live entertainment" application you're developing, but that's okay because they build upon the part you've already working on.

You'll recall that the app you're developing allows its user to see upcoming entertainment events in the graphical user interface. Let's have another look.

[*Shows the demo again, noting how the upcoming events have changed at all three venues*]

The most important feature of the application is that the data it displays is always up-to-date. It's only a short time since my last briefing, but already the events have changed. Despite this we didn't need to change our program code. Through the use of pattern matching it adapts automatically to updates in the source web pages.

Once you've completed this ability to extract data from the web pages we want you to do two different things with it, apart from showing it in the GUI. The most significant as far as our finance department is concerned is the ability to sell tickets to the user. That's what the "Print ticket" feature is for. After the user has selected a venue, they should be able to print a ticket for admittance to the chosen event. The tickets are actually created as small HTML documents. Let's see some examples.

[*Demonstrates printing "tickets" to the upcoming events and viewing them in a web browser*]

To make the tickets easy to find you must export them as a file named `your_ticket.html`. When viewed in a web browser, the ticket must have the following features:

1. Some kind of heading explaining what the document is.
2. The venue for the event.
3. A hyperlink to the website for the event.
4. The name of the selected entertainment event.
5. The date (or dates) of the selected event.
6. The photo or image associated with the event.

Naturally the information must all be correct with respect to the next upcoming event at the venue, and, of course, the ticket must be well-presented, with no visible HTML tags or mark-ups, and laid out neatly. Notice that the first three of these items don't depend on the down-loaded data, so you can still do much of this part of the task even if you're having trouble ex-tracting data from the web pages.

Also, the boffins tell me that to incorporate the photo, all you need to do is put an "`img`" tag in your HTML file with an appropriate URL, which you should be able to extract from the source web page. They also tell me that sometimes the URLs for photos appearing in web pages don't include the "`https`" part and the hostname, such as "www.thezoo.com.au". In

this case you simply need to prepend these strings to the URL before writing it into your HTML code.

(Of course, in the final application the users will need to supply a credit card number before they can print the ticket, but our finance department will add that feature to your program later. They're very keen on such things!)

Our final requirement is to allow the users to make a permanent record of their purchase. You'll recall that all data stored in your Python program's variables is lost when the user shuts down the application, so we need a way of storing it permanently (and potentially making it available to other applications). For this purpose we've provided you with an SQLite database called `bookings.db`. You can download a copy of it from our IT system. After the user has "bought" a ticket to an event, the application must allows them to store details of the event in the database. Let's see how it works.

[*Shows the database and demonstrates storing event data in it*]

Thus, each time the user "buys" a ticket your app must let them record the details of the event for later retrieval. For each such event the following information must be stored as a single row in table `tickets_bought`:

1. The event's name in field `event`.
2. The date or dates in field `date_or_dates`.
3. The venue in field `venue`.
4. The source website's URL in field `website`.

Once again the data stored must be correct and must not contain HTML tags or mark-ups. Also notice that the last two items don't rely on the downloaded data, so you can still do this part of the task even if you're having trouble extracting data from the web pages. Also be careful not to overwrite the user's previous bookings, if any. The database must keep a record of every booking the user chooses to save.

The event name and date stored in the database are exactly as they were shown in the GUI and printed on the ticket, so you don't have to do any more work to extract the information from the web pages. However the boffins tell me that you need to be careful of special characters in the text such as quote marks when inserting into the database. If these give you any trouble feel free to delete or replace them. They say that quote marks can be "escaped" in SQLite character strings by using two consecutive quotes.

Once again your program must not crash if anything goes wrong while trying to print the ticket or save a booking. Let's see what the boffins' demo does when there are problems with accessing the web sites.

[*Shows how the ticket appears and the database is updated when the Internet is disconnected*]

As usual please upload your solution to our IT system before the deadline which is Sunday, June 5th (the end of Week 12). We strongly urge you to submit multiple drafts before the deadline as insurance against IT problems on your side and ours.

For portability your program must run in a standard Python 3 environment with no extensions. Our evaluation team will be testing your program in such an environment. You must

not use any modules that have to be downloaded and installed separately such as "Beautiful Soup" or "Pillow". Also for portability, your Python program should assume that the SQLite database exists in the same folder as the Python code. Don't use hardwired file paths in your code.

Upload a single ".zip" archive containing only the files needed to support your application. Do **not** upload multiple separate files, and do **not** use any other compression formats such as ".rar" or ".7z". To compress your files into zip format on a Mac right-click on the folder and choose "Compress". Under Microsoft Windows right-click on the folder, choose "Send to" and then "Compressed (Zipped) Folder". Your zip archive should contain only three things: your Python program; any image files needed to support your GUI; and a copy of the database. (We don't need a copy of your ticket because we can generate that when we run your program.)

Our investors are looking forward to seeing your final product at your earliest convenience! Farewell.