COMP9120 Database Management Systems

Assignment 2: Database Application Development

Group Assignment (12%)

Introduction

The objectives of this assignment are to gain practical experience in interacting with a relational database management system using an Application Programming Interface (API) (Python DB-API). This assignment additionally provides an opportunity to use more advanced features of a database such as functions.

This is a group assignment for teams of about 3 members, and it is assumed that you will continue in your Assignment 1 group. You should inform the unit coordinator as soon as possible if you wish to change groups.

Please also keep an eye on your email and any announcements that may be made on Ed.

Submission Details

The final submission of your database application is due at 11:59pm on Friday 20th May. You should submit the *items for submission* (detailed below) via Canvas.

Items for submission

Please submit your solution to Assignment 2 in the 'Assignment' section of the unit's Canvas site by the deadline, including EXACTLY THREE files:

- An assignment coversheet as a PDF document (.pdf file suffix) which is available for download from this link on Canvas.
- A SQL file (`WSASchema.sql`) containing the SQL statements necessary to generate the database schema and sample data. This should contain the original schema and insert SQL statements, and any changes or additions you may have made.
- A Python file (`database.py`) containing the Python code you have written to access the database.

Task 1: The Western Sydney Airport (WSA) System

In this assignment, you will be working with the WSA system which is currently under development. The system still requires work in numerous areas, including interaction with the database. Your main task in this assignment is to handle requests for reads and writes to the database coming from the user interface (UI). We first describe the main features that the WSA system should include from a UI perspective, and then discuss where the majority of your database code needs to be implemented.

Logging In

The first form a user is presented with when starting the WSA system is Login, as shown in Figure 1. This feature is still under development and currently requires that an employee (technician or test engineer) enters his/her username and password to be validated prior to successfully log in to the system. Security features such as password encryption/hashing will be implemented at a later stage (and is out of scope for this assignment). Once logged in, the user is taken to the Civil Aviation Safety Authority (CASA) Test Events List

page to view their associated tests for the various aircrafts.

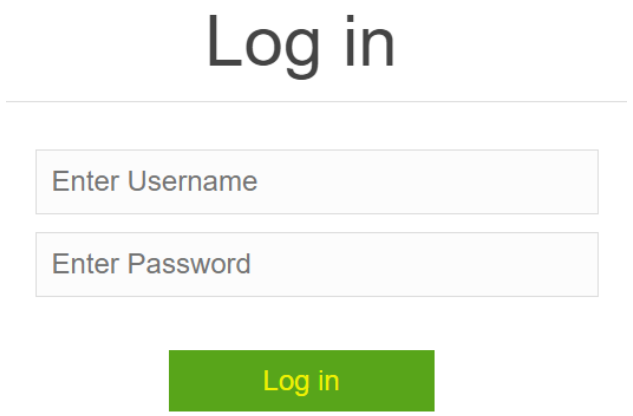

Figure 1 – Login form

Viewing Test Events List

Once a user has logged in, they are shown a list of all their associated test events, as shown below in Figure 2. This list must be ordered such that aircrafts needing test to be performed appear at the top. The list is also sorted by test date in ascending order. Each test event has a test date, aircraft registration number, status, assigned technician, and test engineer. A test event is associated with an employee if the appropriate employee is allocated as either the technician or test engineer to conduct the CASA compliance test.

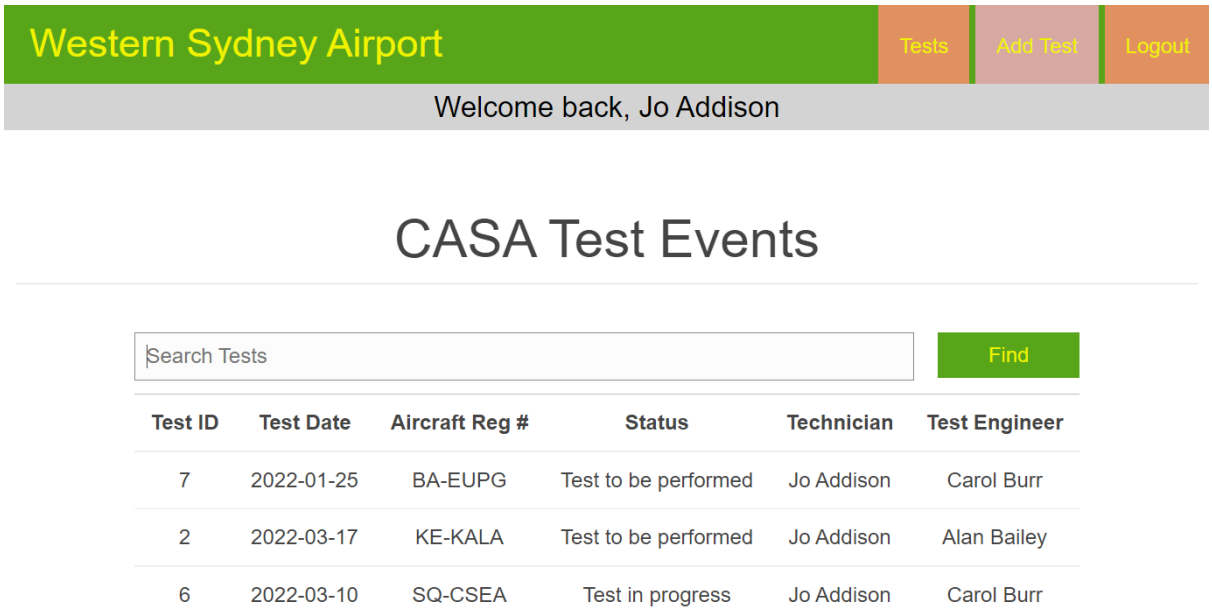

Figure 2 – Viewing Test Events List

Finding Test Events

A user can search through all test events by entering a word or phrase (a 'keyword') in the field next to the Find button, as shown in Figure 3, and then clicking on Find. When such a keyword is specified, then only test events including this word or phrase in their status description, aircraft registration number, or employee names will be retrieved and shown on the list. The search must ignore case sensitivity. For example, given the search keyword 'in progress', Find will return all test events that include the keyword 'in progress' in their

status description, aircraft registration number, or employee names. Searching with a blank/empty keyword field will show all of the logged in user's associated test events. Any search results returned must be ordered such that aircrafts needing test to be performed appear at the top, and by test date in ascending order.

Figure 3 – Finding Test Events

Adding a Test Event

Users may also add a new test event by clicking on the *Add Test* tab in the title bar, entering test details such as test date and aircraft registration number, in the popup dialog that appears, and then clicking on Add Test button, as shown in Figure 4.

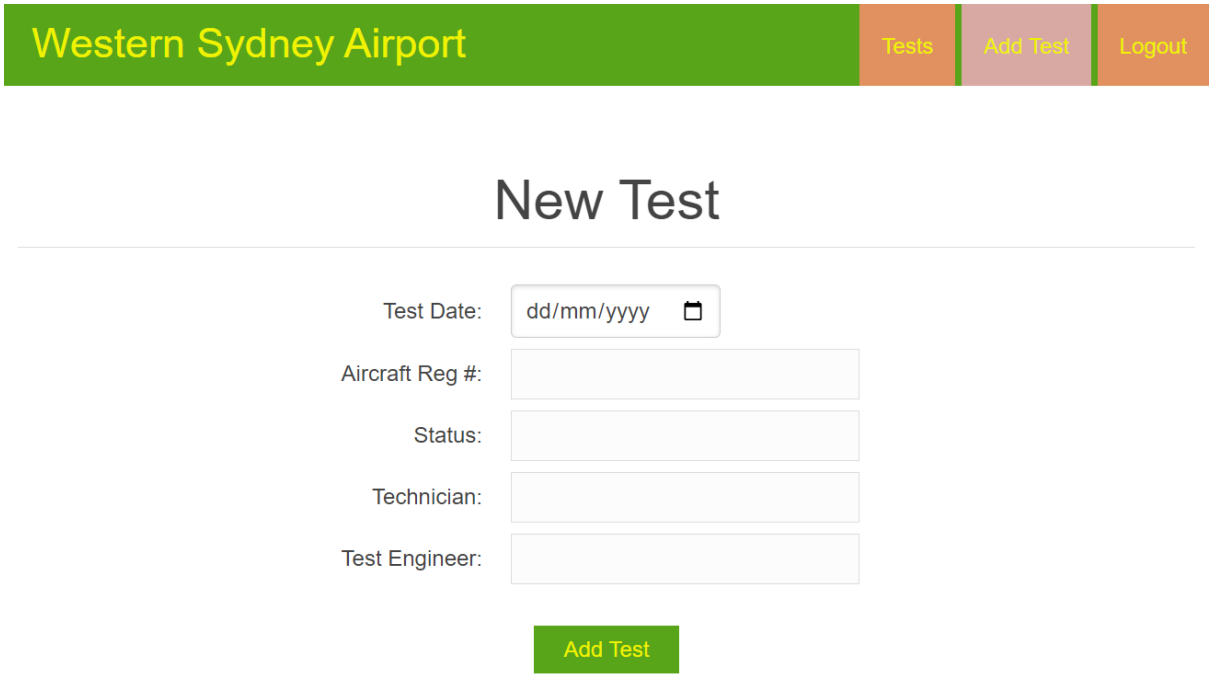

Figure 4 – Adding a Test Event

Updating a Test Event

Users can also update a test event by modifying data in the test event details screen as shown in Figure 5, and clicking on Update Test button. You can access this update screen by clicking on a test event from the list of test events in the Tests tab.

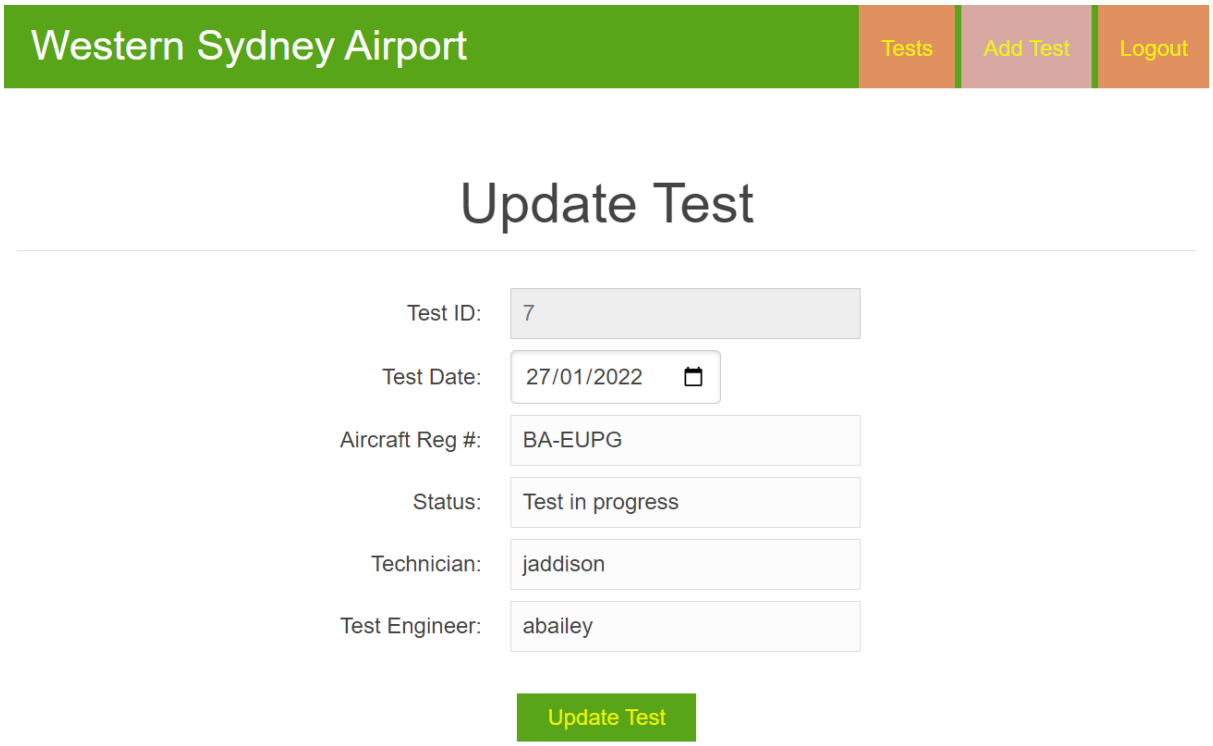

Figure 5 – Updating a Test Event

Database Interaction Code

The files that are needed for the Python version of assignment are as follows:

1. **`WSASchema.sql`:** a file which contains SQL statements you need to run to create and initialise the WSA system database, before starting the application https://canvas.sydney.edu.au/files/23473879/download?download_frd=1
2. **`Assignment2_PythonSkeleton.zip`**: a zip file encapsulating the Python project for the WSA system https://canvas.sydney.edu.au/files/23484341/download?download_frd=1

To inspect the WSA system code, you need to unzip the ZIP archive first, which will create a folder that includes the name **`Assignment2_PythonSkeleton.`** If you experience any difficulties installing and exploring the project, ask your tutor or lecturer for assistance.

The skeleton code uses a number of Python modules to implement a simple browser-based GUI for the WSA system. The main modules are the Flask framework for the GUI and the psycopg2 module for the PostgreSQL database access. Similar to tutorial 8, **you will need to install the Psycopg2 module and the Flask module**. The skeleton code follows the structure described below:

- The main program starts in the **`main.py`** file. You need to use the correct username/password details as specified in tutorial 8, and then implement the missing database access functions – including any necessary SQL code statements required – in the data layer **`database.py`**.
- The presentation layer is done via a simple HTML interface that can be accessed from a web browser. The corresponding page templates are located in the **`templates/`** subdirectory, their CSS style

file is **`static/css`**.

- The transition between the different GUI pages and the initialisation of the Flask framework is done in the **`routes.py`** file. It currently just invokes the pages, but there is no further business logic implemented yet.

You can run the code by running "**`python main.py`**". This starts a local web server and prints out some debug messages in the terminal; the GUI can then be accessed with any web browser on the same computer via the local URL `http://127.0.0.1:5000/` (If that doesn't work you can also try `http://0.0.0.0:5000/`). Please note that, to terminate the application, you will need to stop the local web server which is running in the background.

Task 2: Functions Implementation

Core Functionality

In this assignment, you are provided with a Python skeleton project that must serve as the starting point for your assignment. Your task is to provide a complete implementation for the file **`database.py`**, as well as make any modifications necessary to the database schema (i.e., **`WSASchema.sql`**). Specifically, you need to modify and complete these five functions:

1. **`checkEmpCredentials`** (for login)
2. **`findTestsByEmployee`** (for viewing test events list)
3. **`findTestsByCriteria`** (for finding test events)
4. **`addTest`** (for adding test event)
5. **`updateTest`** (for updating test event)

Note that, for each function, the corresponding action should be implemented by issuing SQL queries to the database management system. If you directly output and/or manipulate the result without issuing SQL queries, you are considered as cheating, and you will get zero point for the assignment.

Marking

This assignment is worth 12% of your final grade for the unit of study. Your group's submission will be marked in line with the rubric that follows.

Group member participation

If members of your group do not contribute sufficiently, you should alert the unit coordinator as soon as possible. The course instructor has the discretion to scale the group's mark for each member as follows:

Level of contribution	Proportion of final grade received
No participation.	0%
Full understanding of the submitted work.	50%
Minor contributor to the group's submission.	75%
Major contributor to the group's submission.	100%

Marking Rubric

Your submissions will be marked according to the following rubric, with a maximum possible score of 12 points.

	Part marks (0 – 1)	Full marks (1.5 – 2)
Login	Can correctly login the user 'jaddison' and validate her username and password.	All valid users can be logged in successfully, and unsuccessful user logins should be rejected.
View Test Events List	Correctly list all test events associated with user 'jaddison' in the correct order (see Figure 2).	Correctly list all test events associated with a user, in the correct order, for all possible username input from Figure 1.
Find Test Event	Correctly list test events for keyword "in progress" (see Figure 3).	Correctly list test events for all possible keywords.
Add Test Event	Can correctly add a test event to the database.	Can correctly add all valid test events to the database. Test events added with invalid details should be rejected.
Update Test Event	Can correctly update the status of a test event as shown in Figure 5.	Can correctly update details of all test events, ensuring the updated details for a test event are valid.
Stored Procedure	A couple of stored procedures (functions) are correctly created in the submitted SQL file.	A couple of stored procedures (functions) are correctly created in the submitted SQL file, and correctly called in two of the five specified functions.