COMP9120 Database Management Systems   **Semester 1 2020**

Assignment 3: Database Application Development

Group Assignment (15%)

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 2 group. You should inform the teaching assistant (Iwan Budiman) 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 Canvas.

Submission Details

The final submission of your database application is due at 11pm on the 22nd May. You should submit the *items for submission* (detailed below) via Canvas.

Items for submission

Please submit your solution to Assignment 3 in the 'Assignment' section of the unit's Canvas site by the deadline, including EXACTLY TWO files:

- A SQL file (`IssueTrackerSchema.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.

Section 1: Introducing the Issue Tracking System

In this assignment, you will be working with an Issue Tracking System ("Issue Tracker") which is currently under development. The system still requires work in numerous areas, including the 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 Issue Tracker should include from a UI perspective, and then discuss where the majority of your database code needs to be implemented.

Logging In

The first page a user is presented with when starting Issue Tracker is Login, as shown in Figure 1. This feature is still under development and currently only requires that a user enters their UserName to log into the system (secure logins will be implemented at a later stage and are not part of this assignment). Once logged in, a user is taken to the Issues page to see their associated issues.

Figure 1 – Login form

Viewing Issue List

Once a user has logged in, they are shown a list of all their <u>associated</u> issues, and their details,, as shown below in Figure 2. This issue list must be ordered by Title. Each issue has a creator, and can also have a resolver and/or verifier and/or a description. An issue is <u>associated with a user</u> if they are recorded as its creator or resolver or verifier.
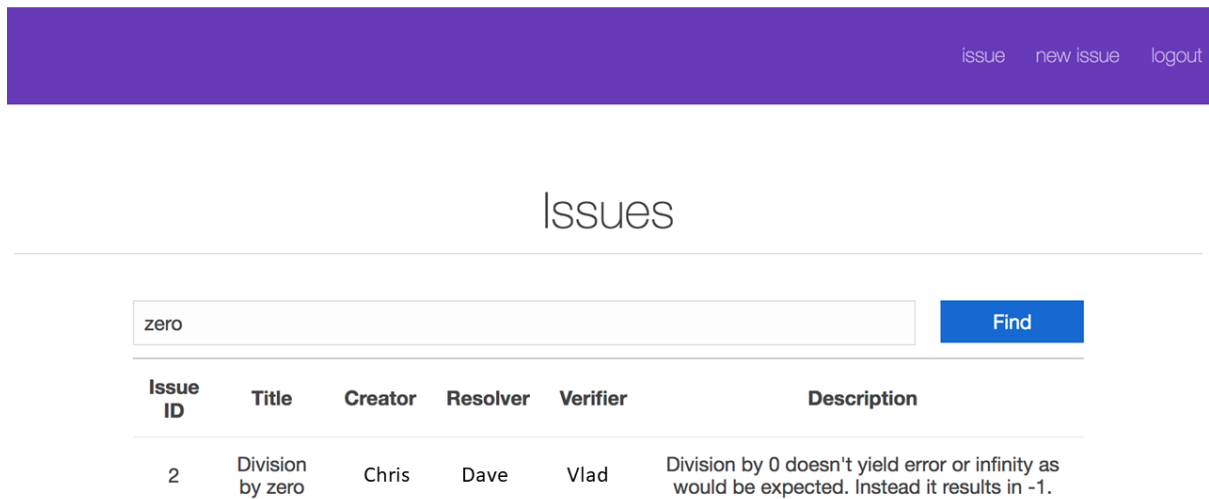

Figure 2 – Viewing Issue List

Finding Issues

A user can search through all issues 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 issues including this word or phrase in their title will be retrieved and shown in the list. For example, given the search keyword 'zero', Find will return all issues that include the word 'zero' in their title. Searching with a blank/empty keyword field will show all of the logged in user's associated issues. Any search results returned must be ordered by Title.
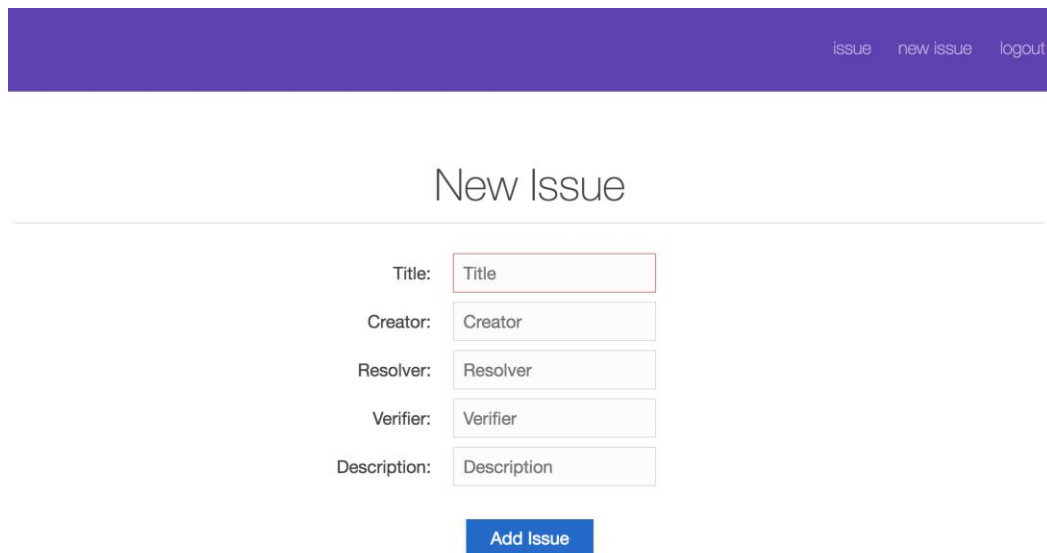
Figure 3 – Finding Issues

Adding an Issue

Users may also add a new issue by clicking on the *new issue* tab in the title bar, entering issue details in the popup dialog that appears, and then clicking on Add Issue. Roles that have not yet been assigned for an issue must be entered as a dash ('-').

Figure 4 – Adding a new Issue

Updating Issues

Users can also update an issue by modifying data in the issue details screen as shown in Figure 5, and clicking on the Update Issue Button. You can access this update screen by clicking on an issue from the list of issues in the Issues tab.

Figure 5 – Updating an Issue

Database Interaction Code

The files that are needed for the Python version of this assignment can be downloaded from the Canvas Modules page (in the Assignment section). The files you will need are:

1. **`IssueTrackerSchema.sql:`** a file which contains SQL statements you need to run to create and initialise the IssueTracker database.

2. **`Assignment3_PythonSkeleton.zip`**: a zip file containing the Python project for IssueTracker

You need to unzip the ZIP archive to create the folder **`Assignment3_PythonSkeleton.`** Ask your tutor for assistance if you experience any difficulties installing and exploring the project.

The skeleton code uses a number of Python modules to implement a simple browser-based GUI for the issue tracker. The main modules are the Flask framework for the GUI and the psycopg2 module for the PostgreSQL database access. Similar to tutorial 7, **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 7, 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.

Section 2: Your Task

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 (**IssueTrackerSchema.sql**). Specifically, you need to modify and complete these functions:

1. **checkUserCredentials** (for login)

2. **findUserIssues** (for viewing issue list)

3. **findIssueBasedOnExpressionSearchOnTitle** (for finding issue)

4. **addIssue** (for adding issue)

5. **updateIssue** (for updating issue)

Note that, for each function, the corresponding action should be implemented by issuing SQL queries to the database management system. Directly returning results without issuing SQL queries will be considered as cheating, and you will get zero points for the assignment.

Marking

This assignment is worth 15% of your final grade for the unit of study.

Group member participation

If members of your group do not contribute sufficiently you should alert your tutor 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 15 points.

	Part marks (~1.5)	Full marks (2.5)
Login	Can correctly login the user 'Dave' and get their userId.	All valid users can be logged in successfully, and unknown user logins are rejected.
View Issue List	Correctly list all issues associated with user 'Dave' in the correct order (see Figure 2).	Correctly list all issues associated with a given user, in the correct order, for all possible users.
Find Issue	Correctly list the issues for keyword "zero" (see Figure 3).	Correctly list issues for other keywords and phrases
Add Issue	Can correctly add the issue ('test', 'test', 'Chris', 'Dave', '-') to the database.	Can correctly add other valid issues to the database. Issues associated with unknown users are rejected.
Update Issue	Can correctly update the description of the issue as shown in Figure 5.	Can correctly update other issues, with checking that all associated users are valid.
Stored Procedure	A stored procedure is correctly created in the submitted SQL file	A stored procedure is correctly created in the submitted SQL file, and is correctly called in one of the functions specified above.