13/06/2021

Python BE SQL Checker

Harold Chandler - chahf004, Praneet Shrestha (shrpy006), Duy Anh Pham (phady026), Duane Stevenson (stedy026), Justin Branson (brajm008)
GROUP: 2021-SP2-26

Introduction:

This user manual is intended for use with Group 2021-SP2-26's Python BE SQL Checker system that was made as part of the University of South Australia's ICT Capstone Project course.

The overall system comprises a python-based application that accepts n number of test SQL queries and n number of student SQL queries that are associated with n number of assignment questions. The checker makes a comparison between each of the student queries and the corresponding test query (with regards to the retrieved data i.e., columns, rows, values, order of columns, order of rows, time to execute, joins and use of top). From the comparison the system generates feedback and marking data which it can structure via graphable and exportable student records. The application also comprises a student side API that allows students to individually test their query attempts and receive feedback on how they can improve their queries. The overall purpose/business justification for the application is to improve the overall 'student experience' for students undertaking the University of South Australia's Database Fundamentals course. The enhanced feedback provided by the system is intended to support students, allowing them to improve the quality of their assessment work, which is expected to flow through to better learning outcomes and ultimately better grades.

This user manual is split into two sections. The first is intended for the admin user (i.e., the course coordinator or teacher/s of the Database Fundamentals course) and the second is intended for students using the associated API component of the system.

System Requirements:

- ODBC Driver for SQL Server: https://docs.microsoft.com/en-us/sql/connect/odbc/download-odbc-driver-for-sql-server?view=sql-server-ver15
- pyodc
- os
- flask
- flask_restful
- apispec
- marshmallow
- flask_apispec
- tkinter
- csv
- matplotlib
- random
- string
- requests
- sys
- uuid
- time
- zipfile
- ntpath

Part One: Settings Page

Database/Server Connections:

To start up the core Python BE SQL Checker application run the mainGUI.py file in the FrontEnd folder of the project directory. After running this file, you will be presented with the main menu GUI screen as seen in Figure 1. From here you can access all the system's pages/functionalities, however, most of these functionalities won't work unless the database/server connections have first been established in the Settings page. To Navigate to the settings page, and configure these connections, click the rightmost button on the navigation bar at the top of the page or click the Settings button in the bottom right corner of the main menu GUI.

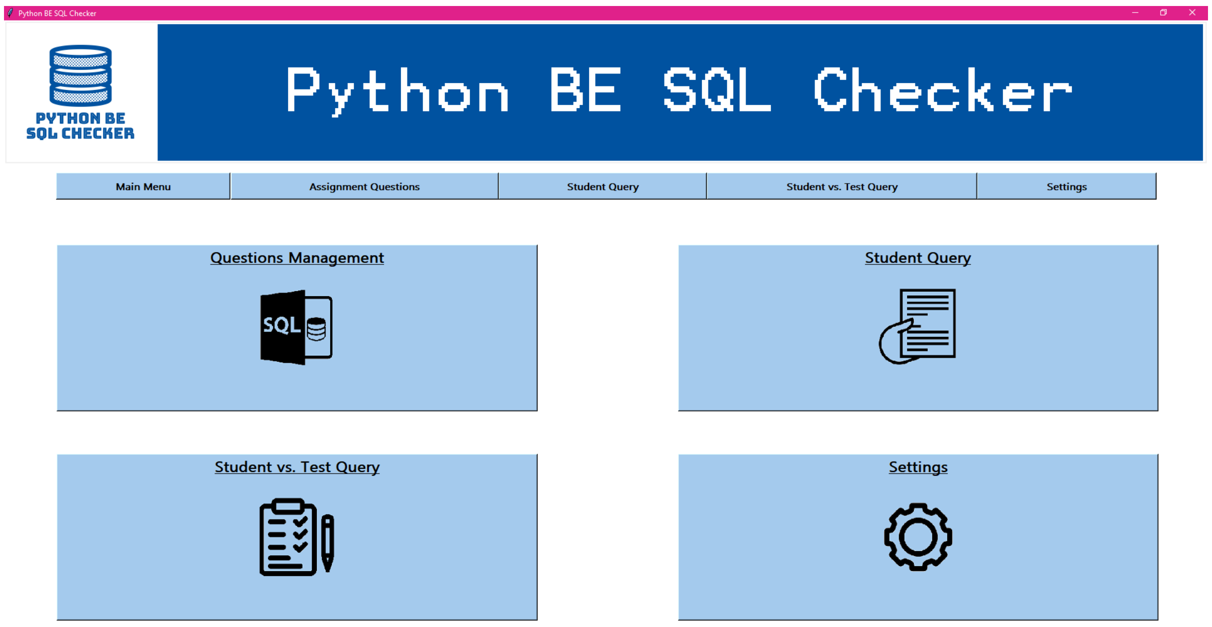
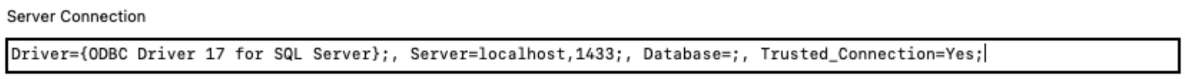


Figure 1: Main Menu GUI

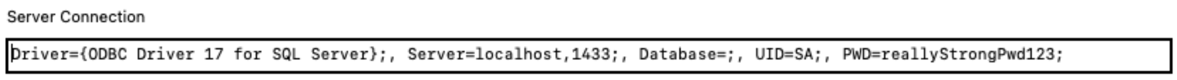
After clicking the settings button, you will be taken to the page seen in Figure 4. From here you will need to input the server connection string as well as the name of the two databases you wish to connect to on the server. From Figures 2 and 3 you can see that the server connection string takes 4 or 5 parameters depending on your device's operating system. In the case of Mac you will need to input the username and password of your desired server while on Windows you'll simply need to add a parameter called Trusted Connection, which needs to be set to Yes. Follow the format in Figures 2 and 3 to avoid any errors. NOTE: This is just the server connection string so you can leave the database parameter equal to nothing as it is in Figures 2 and 3. Also your device may not have the required driver to establish a connection between a python file and your desired server. In which case you'll need to download and install the ODBC Driver for SQL Server from the link below:

https://docs.microsoft.com/en-us/sql/connect/odbc/download-odbc-driver-for-sql-server?view=sql-server-ver15.


Server Connection

```
Driver={ODBC Driver 17 for SQL Server};, Server=localhost,1433;, Database=;, Trusted_Connection=Yes;
```

Figure 2: Windows Server Connection


Server Connection

```
Driver={ODBC Driver 17 for SQL Server};, Server=localhost,1433;, Database=;, UID=SA;, PWD=reallyStrongPwd123;
```

Figure 3: Mac Server Connection

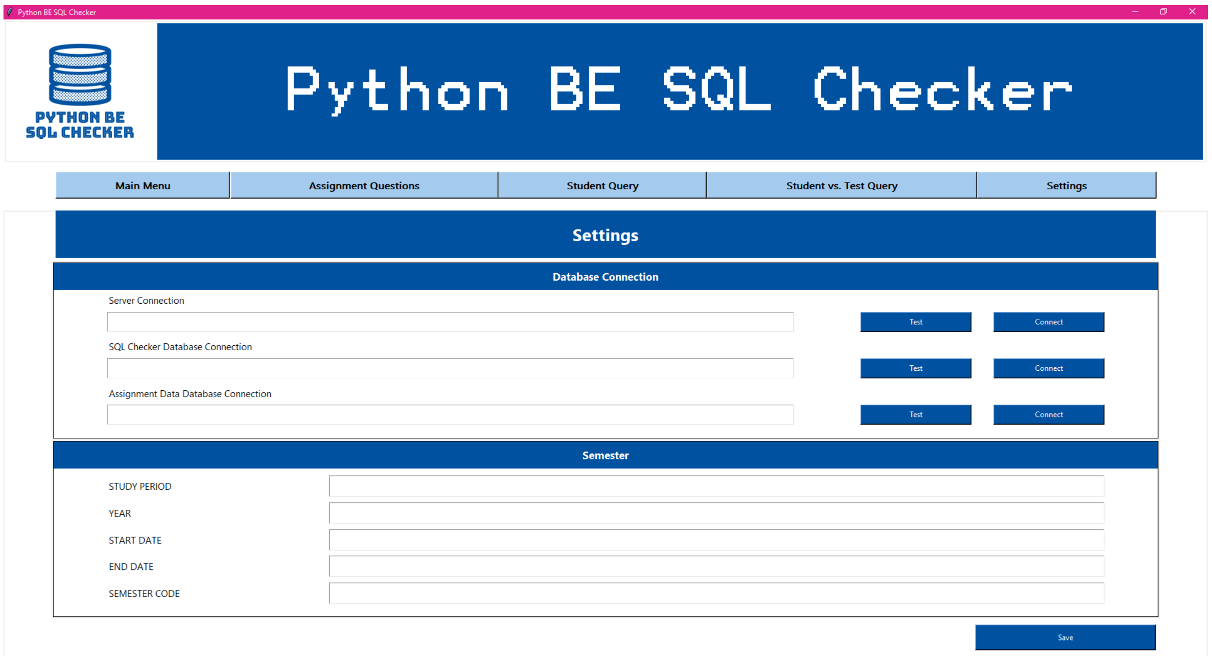
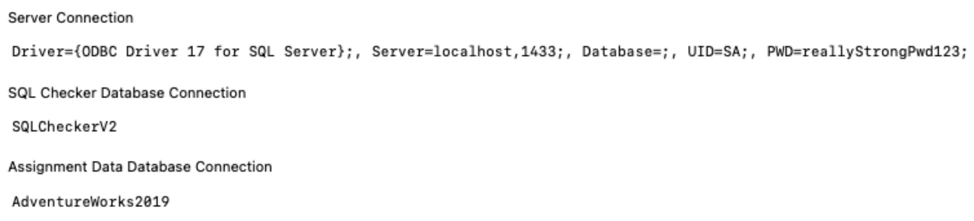


Figure 4: Settings Page GUI

After you've input the server connection string you can test its validity by clicking the test button on the right of the entry box. You can then establish connection to the server by clicking the connect button although this will return an error message pop up if the connection string is invalid or connection cannot be established for any reason.

The next step is to input the name of the database (on the server) that has/will store the SQL Checker data i.e., the assignment questions, test queries, students, marks etc. See Figure 5 for an example. In this case the database name is SQLCheckerV2 (The .bak file of this database is provided inside the Project directory so that you can restore it on your machine. The disconnected database can also be found in the project deliverables folder submitted via the learn online). You can test this database connection by clicking the test button to the right of the entry and establish the database connection by clicking the connect button.

Lastly repeat the previous step except with the assignment data database name. By this we mean the database that has the data that the student queries and test queries will search through. See Figure 5 for an example. In this case the database name is AdventureWorks2019. Again, you can test this database connection by clicking the test button on the right of the entry and establish the database connection via the connect button.



Server Connection

```
Driver={ODBC Driver 17 for SQL Server};, Server=localhost,1433;, Database=;, UID=SA;, PWD=reallyStrongPwd123;
```

SQL Checker Database Connection

```
SQLCheckerV2
```

Assignment Data Database Connection

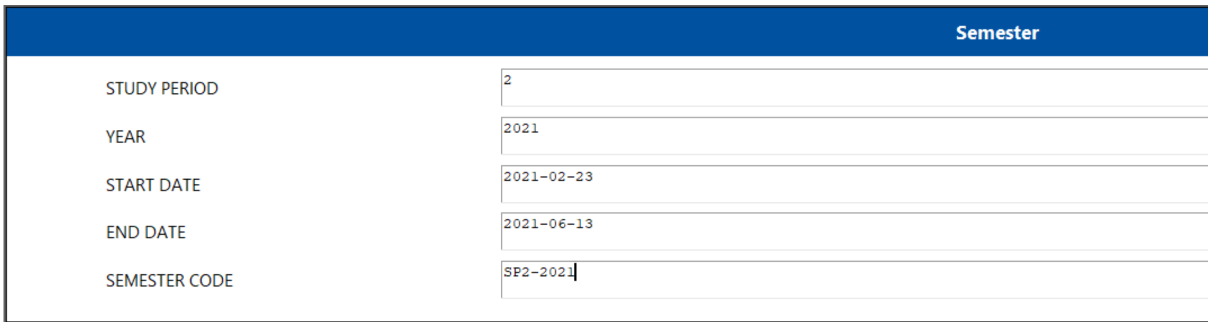
```
AdventureWorks2019
```

Figure 5: Completed Server Connection and Database Connections

Semester:

The last functionality provided by the Settings page is the ability to set the current Semester i.e., the semester that all the input assignment questions will be associated with. This portion of the page is relatively straight forward, however, it does have some specific input formatting requirements.

As you can see from Figure 6 the Study Period entry just requires a single integer input (accepts 1-7 as a valid input as these are the only study periods offered by the University of South Australia). The year entry accepts a plain 4-digit year input (just entering the final 2 digits of the year will work but is not recommended). The start date and end date entries rely on a very specific input format due to the way MSSQL stores dates. The dates must be entered as seen in Figure 6 i.e., a 4-digit year followed by a hyphen, then a 2-digit month followed by a hyphen, followed by a 2-digit day. Lastly the Semester code should be entered and should comprise of the study period and the year separated by a hyphen (e.g. SP2-2021).



	Semester
STUDY PERIOD	2
YEAR	2021
START DATE	2021-02-23
END DATE	2021-06-13
SEMESTER CODE	SP2-2021

Figure 6: Semester Input Formatting Requirements

Part Two: Question Management

Input Question/Test Query:

Now that you've established connection to both databases and set the current semester you can start inputting questions and test queries. To do this click the Assignment Questions button in the navigation bar at the top of the screen or click the Questions Management button from the main menu GUI. From here you should be displayed with the option to either Input Questions/Test Queries or Select Assignment Questions. Begin by clicking the Input Question/Test Queries option and you should be taken to the page seen in Figure 7.

To input a question/test query simply place the desired question text/test query text in the corresponding entry box (on the same line **THIS IS IMPORTANT**), select the difficulty (can be either 001, 002, 003, 004 or 005) you wish to associate the question/query with and click Submit. Note: when you input a question it automatically gets populated with a set of default deductions based on the 8 tests (columns, rows, values, order of columns, order of rows, time to execute, joins, use of top). The values associated with these deductions can be individually modified in the Assignment Question Selection page. Note: typically, when you attempt to store an SQL query string in a database, any single quotes present in the query will break the POST string and throw an error when attempting to store it. The python BE SQL checker utilises a makeStorable function which adds a second single quote to any existing single quotes in the SQL query. This voids the first single quote making it storable in the MSSQL database. Furthermore, when these queries are being read back from the database, for the comparison functionality, the system utilises a makeExecutable method to remove the second single quote, returning the query back to an executable format.

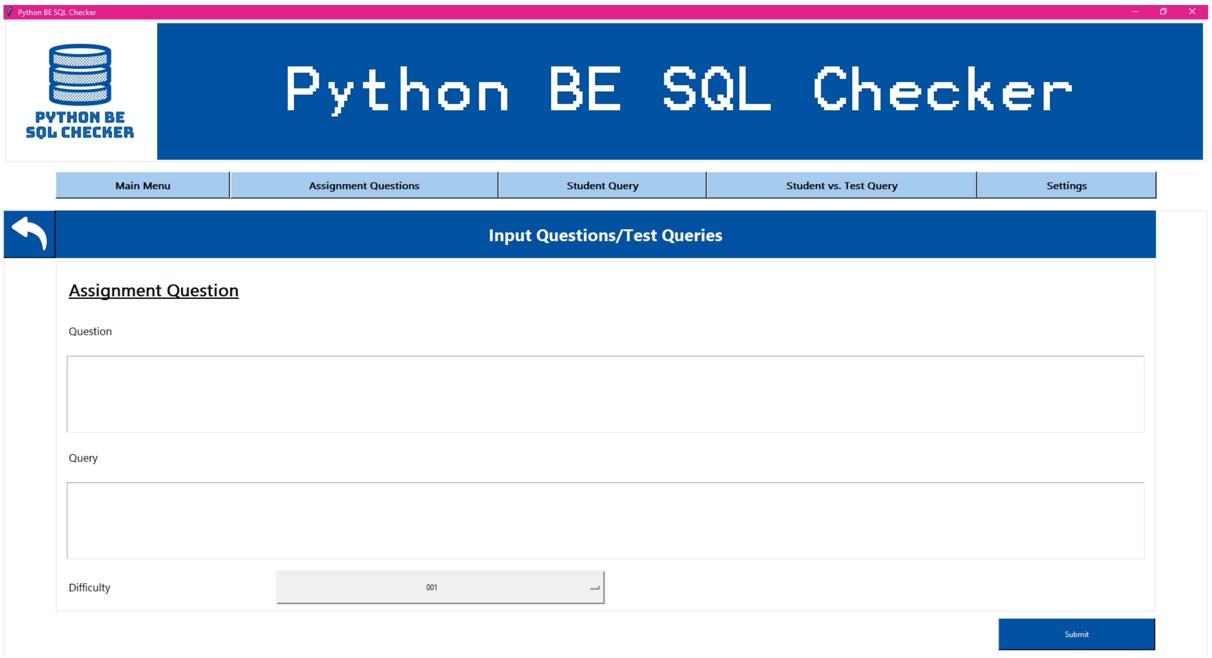


Figure 7: Question/Test Query Input Page

Assignment Question: Selection

After you've input a question and test query, the system doesn't automatically associate the question with the current semester. It's up to you to choose which questions (from the ones you've input) that you would like to be considered as active assignment questions i.e., the questions that students will be assessed on in the current semester. To do this you need to navigate back to the Question Management page and click the Select Assignment Questions button. From here you should be presented with the page seen in Figure 8.

To operate this page, you click the drop down and select which difficulty of question you would like to choose from and then click the Find button. A table will appear in the bottom half of the page displaying all the questions associated with the chosen difficulty. To select a question and subsequently convert it into an active assignment question simply click the associated tick box in the rightmost column of the table. Alternatively, to deselect a currently active assignment question simply deselect the tick box. In the 4th column of each row there is a button that says deductions. When clicked a pop-up page appears allowing you to modify the deductions (and each deductions value) associated with a specific question. See Figure 9 for an example of this deduction page. To save any changes to the deduction values for a specific question simply click the save button at the bottom of the page.

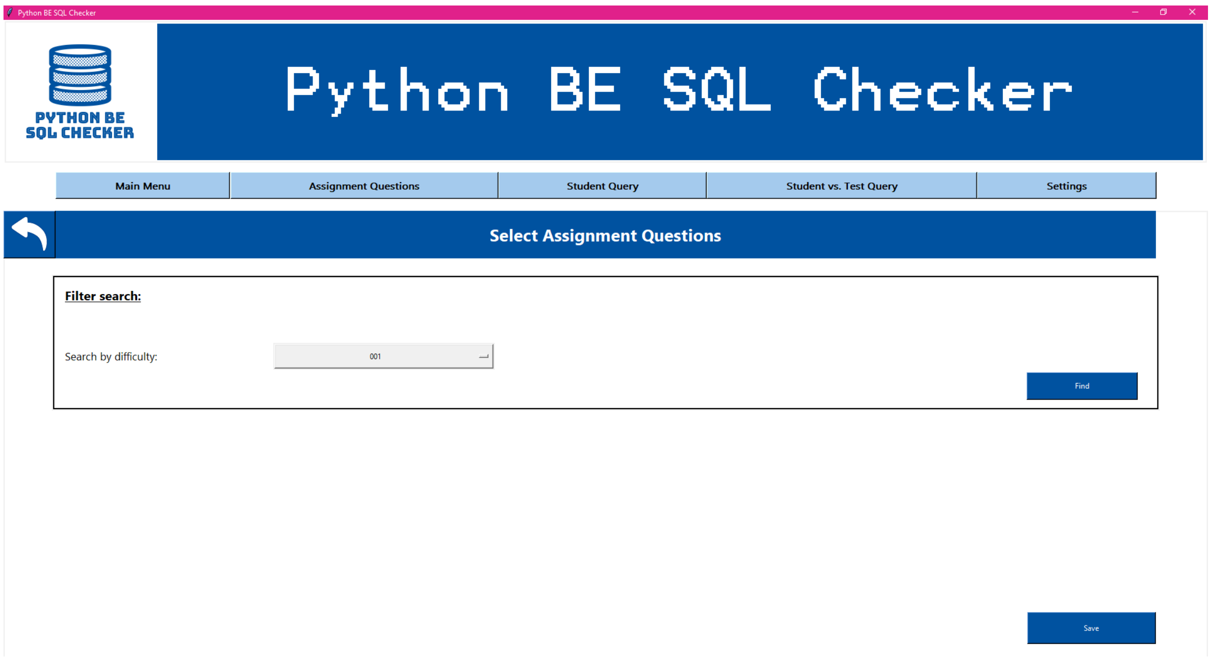
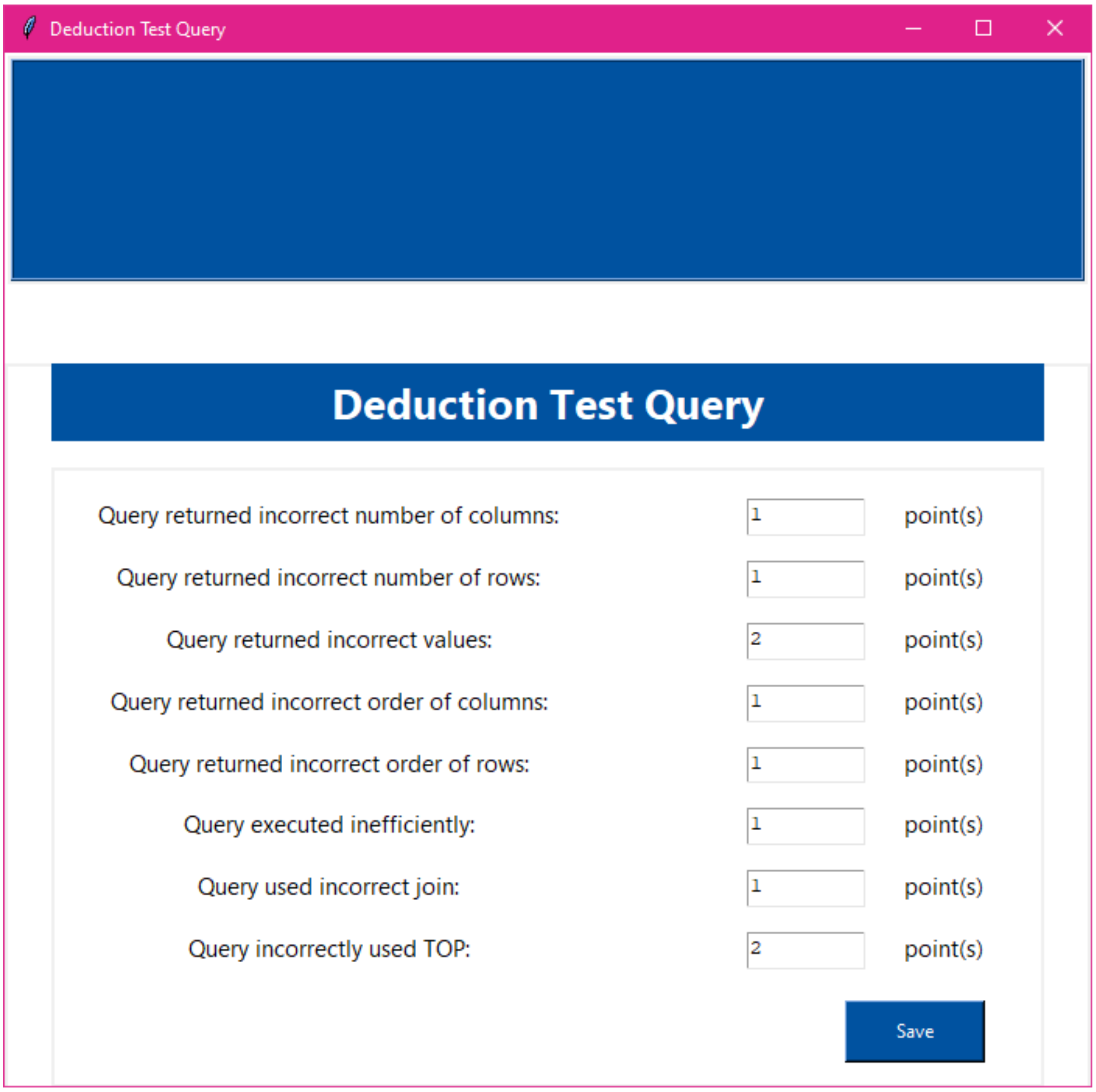


Figure 8: Assignment Question Selection Page



Deduction Test Query

Query returned incorrect number of columns:	1	point(s)
Query returned incorrect number of rows:	1	point(s)
Query returned incorrect values:	2	point(s)
Query returned incorrect order of columns:	1	point(s)
Query returned incorrect order of rows:	1	point(s)
Query executed inefficiently:	1	point(s)
Query used incorrect join:	1	point(s)
Query incorrectly used TOP:	2	point(s)

Figure 9: Deduction Page

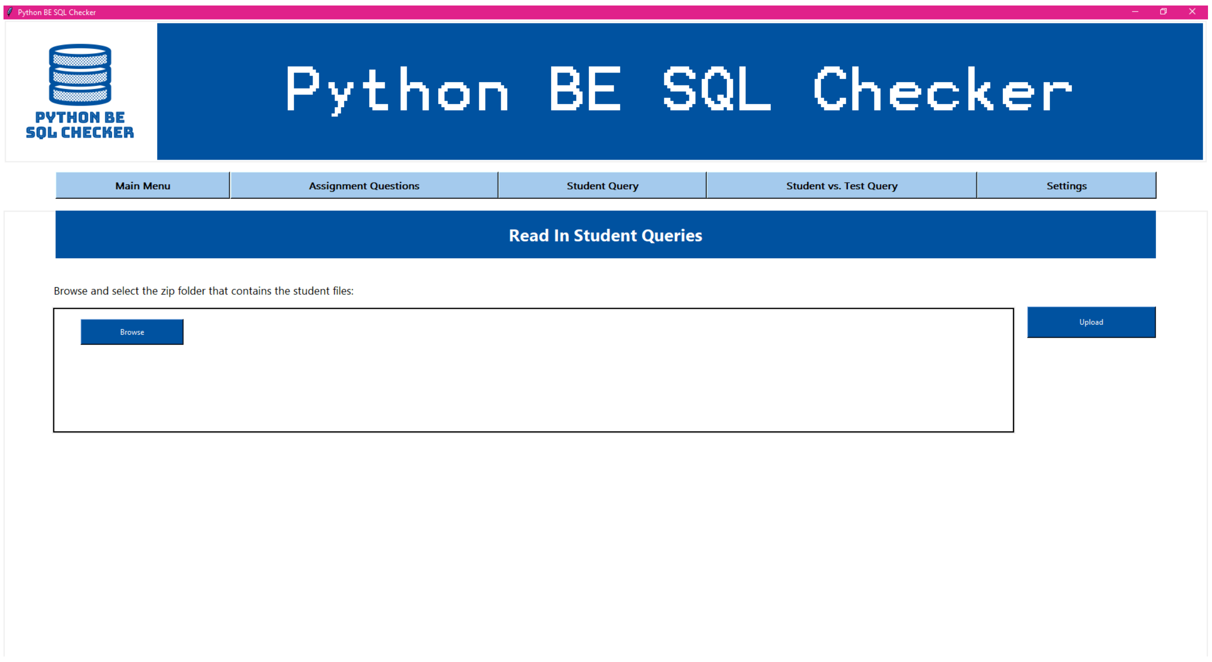
Part Three: Student File Read In

Once connection to the server and the required databases has been established and there is at least one selected assignment question, student query attempts can be read into the system.

NOTE: It is important that at least one assignment question has been selected as in order for student queries to be stored in the SQLCheckerV2 database they must reference an assignment question using its questionId and question number (questionNumber).

To begin reading in student queries navigate to the Student Query page using the navigation bar at the top of the page or click the Student Query button on the top right of the main menu

GUI. You will subsequently be navigated to the page seen in Figure 10. To operate this page, click the Browse button. This will bring up a directory navigation window allowing you to select the zip file that contains the student files. Upon selection, the files inside of the zip file get extracted to a temporary student file directory inside of the project directory. To read the student files from the temporary student file directory click the upload button on the right of the page. A table will subsequently appear in the bottom half of the page displaying all the student queries that have been read into the system (and stored in the SQLCheckerV2 database).



Browse and select the zip folder that contains the student files:

Figure 10: Student File Read In Page

Part Four: Student vs. Test Query Comparison

Once the Assignments questions have been selected and all the student query attempts have been read into the system navigate to the Student vs. Test Query page via the navigation bar at the top of the screen or the button on the bottom left of the main menu GUI.

As you can see from Figure 11 this page gives you the option to Generate Results which calls the systems core functionality, comparing each student query with the corresponding test query and generating a record of feedback and marks for each student for each assignment question. Once clicked this Generate Results button will show a tree view table containing each student and their mark and feedback for each assignment question. The information from this table can be exported to a csv file via the Export Results button at the bottom of the page and a variety of graphs can also be generated to visualise this data via the Graph Results button.

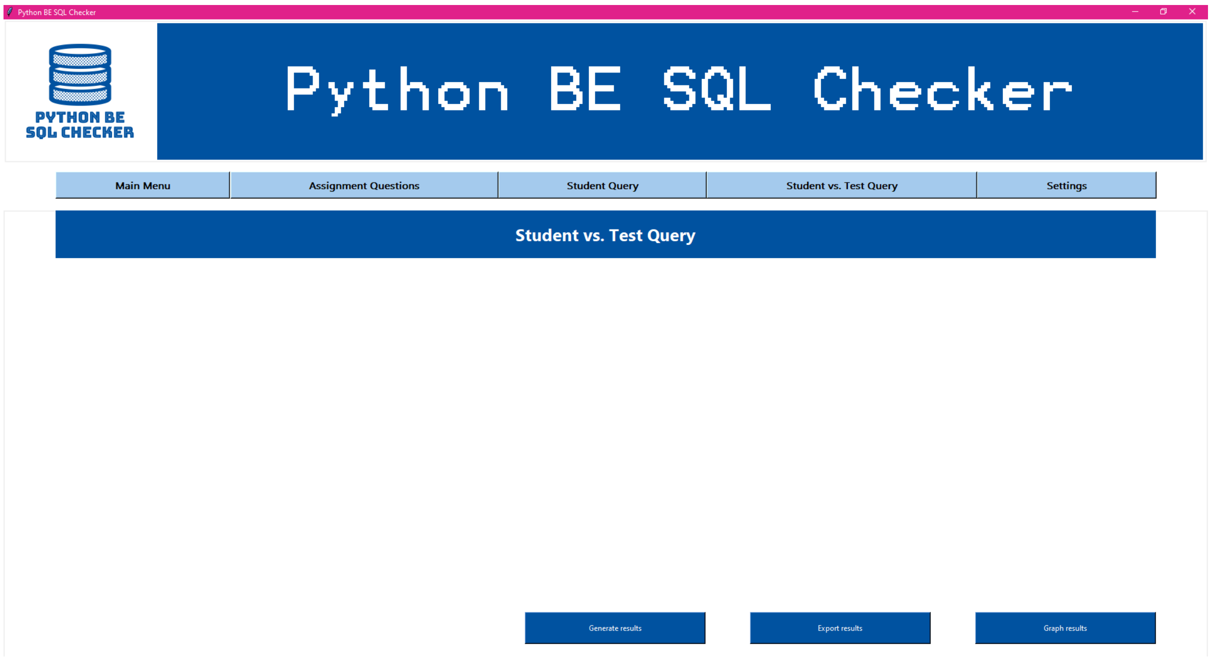
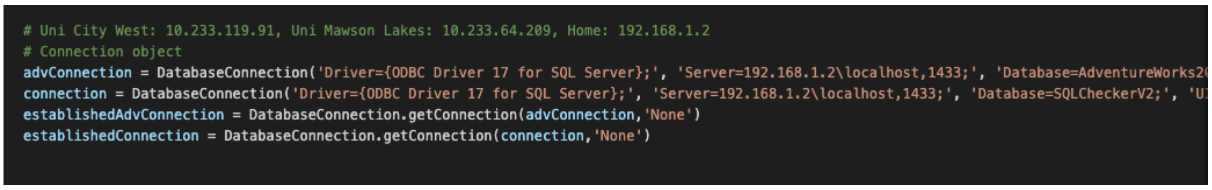


Figure 11: Student Vs. Test Query Comparison Page

Part Five: Setting up and using the API

In order for students to successfully use the API the lecturer will first need to go into the API.py file (located inside of the project directory) and configure their connection to both the SQL Checker Database and the assignment data database (See Figure 12 Below). In the example shown in Figure 12 the advConnection is the connection object for the assignment data database and connection is the connection object for the SQL Checker database. The teacher will need to adjust these connection objects to meet their server/database set up. Once both connections have been successfully adjusted the teacher will need to run the API.py file on their local network on which the databases are located. Once the API is running the teacher needs to obtain the IP address of their host computer which can be done by typing cmd into the search bar and selecting the command prompt as shown in the Figure 13.



```
# Uni City West: 10.233.119.91, Uni Mawson Lakes: 10.233.64.209, Home: 192.168.1.2
# Connection object
advConnection = DatabaseConnection('Driver={ODBC Driver 17 for SQL Server};', 'Server=192.168.1.2\localhost,1433;', 'Database=AdventureWorks2
connection = DatabaseConnection('Driver={ODBC Driver 17 for SQL Server};', 'Server=192.168.1.2\localhost,1433;', 'Database=SQLCheckerV2;', 'U
establishedAdvConnection = DatabaseConnection.getConnection(advConnection,'None')
establishedConnection = DatabaseConnection.getConnection(connection,'None')
```

Figure 12: API.py Connections (need to be changed by the admin user)

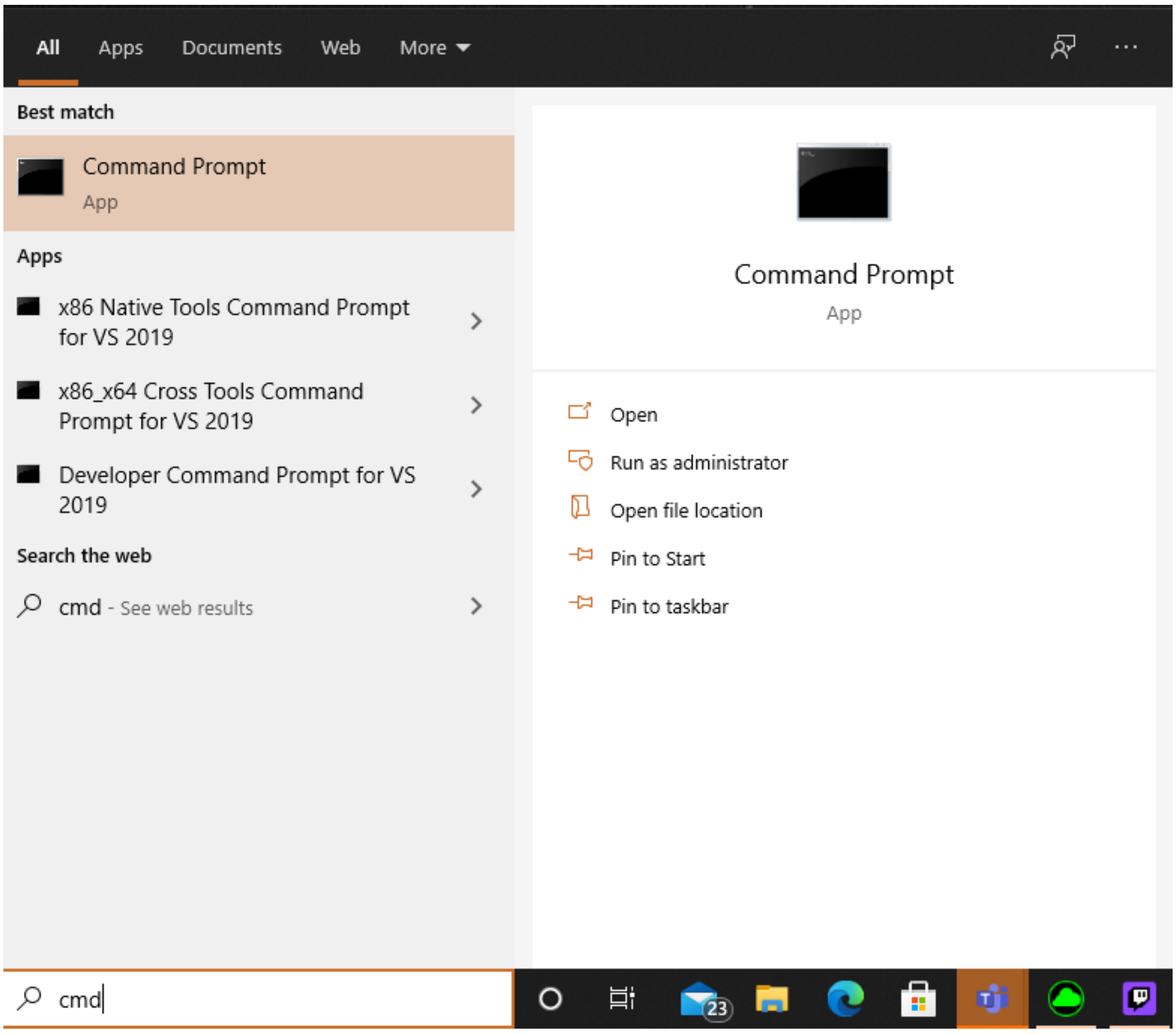
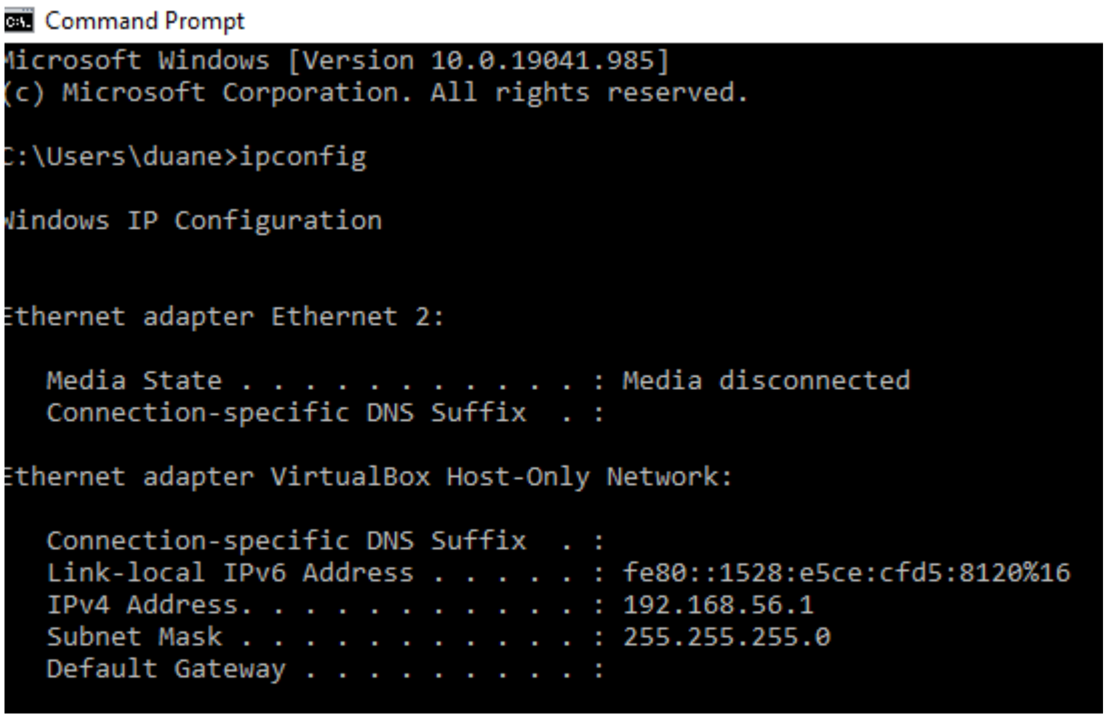


Figure 13: Opening the Command Prompt

Once the command prompt is open type ipconfig and press enter. This will provide you with your ipv4 address which will need to be shared with the students for them to connect to the API. Shown below is the expected output when the ipconfig command is successful.



```
Command Prompt
Microsoft Windows [Version 10.0.19041.985]
(c) Microsoft Corporation. All rights reserved.

C:\Users\duane>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet 2:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Ethernet adapter VirtualBox Host-Only Network:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::1528:e5ce:cfd5:8120%16
   IPv4 Address. . . . . . . . . . . : 192.168.56.1
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :
```

Figure 14: IP Config Output

Once the students have been provided with the correct IP address (this will always be the ipv4 address) they will need to input the IP address into their web browser of choice and hit enter. The example below is using Google Chrome.

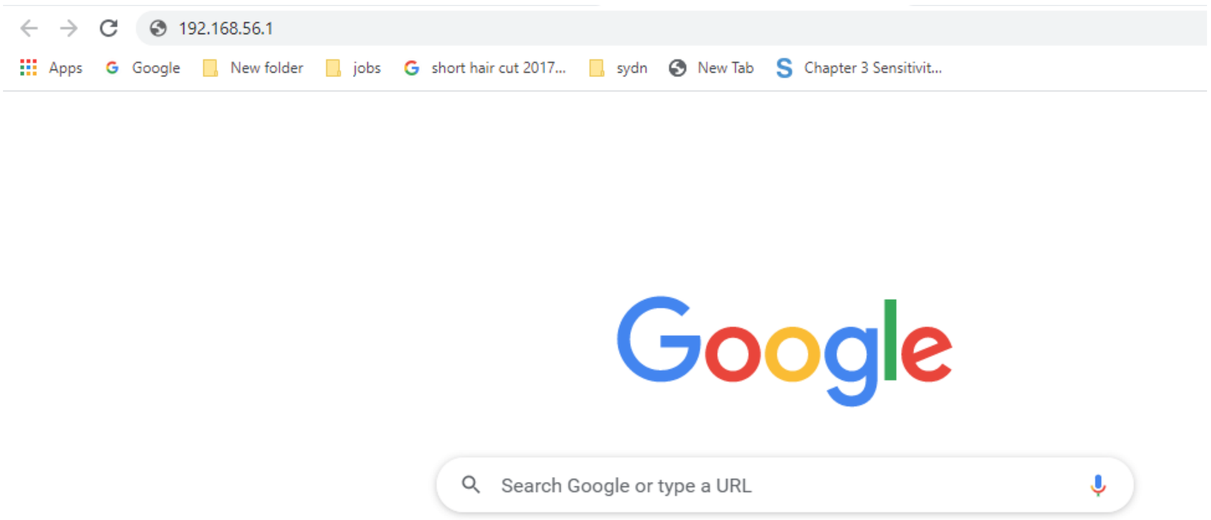
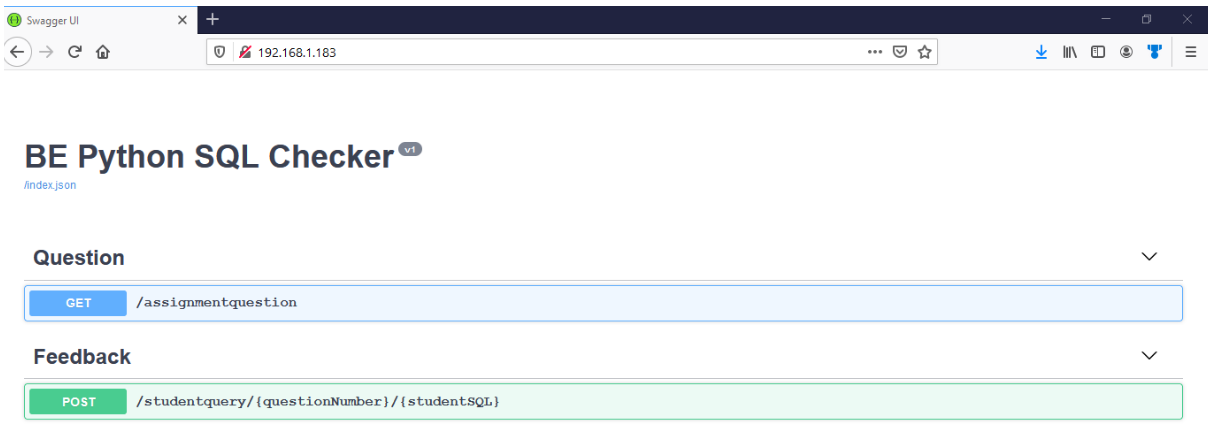


Figure 15: Input IP Address Into Web Browser

Upon navigating to the IP address provided, they will reach the following page.



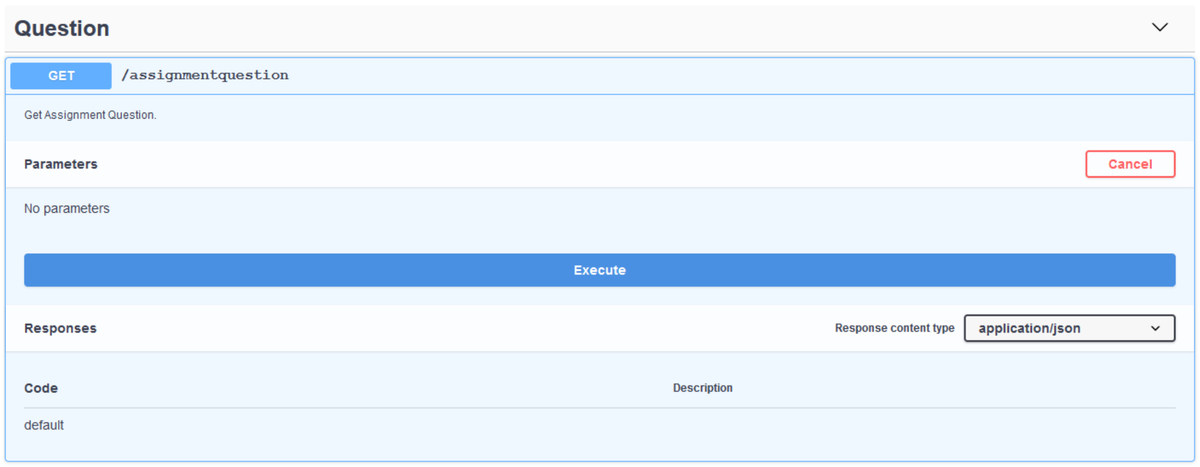
BE Python SQL Checker v1
/index.json

Question
GET /assignmentquestion

Feedback
POST /studentquery/{questionNumber}/{studentSQL}

Figure 16: Swagger API Interface

The endpoints are expandable and interactable by students. The first endpoint allows students to retrieve all the assignment questions from the SQL Checker database on the teacher's device. These questions have to be input into the system by the lecturer as shown above (See Test Query Input Page/Assignment Question Selection Page). Clicking execute will respond with the assignment questions.



Question
GET /assignmentquestion

Get Assignment Question.

Parameters

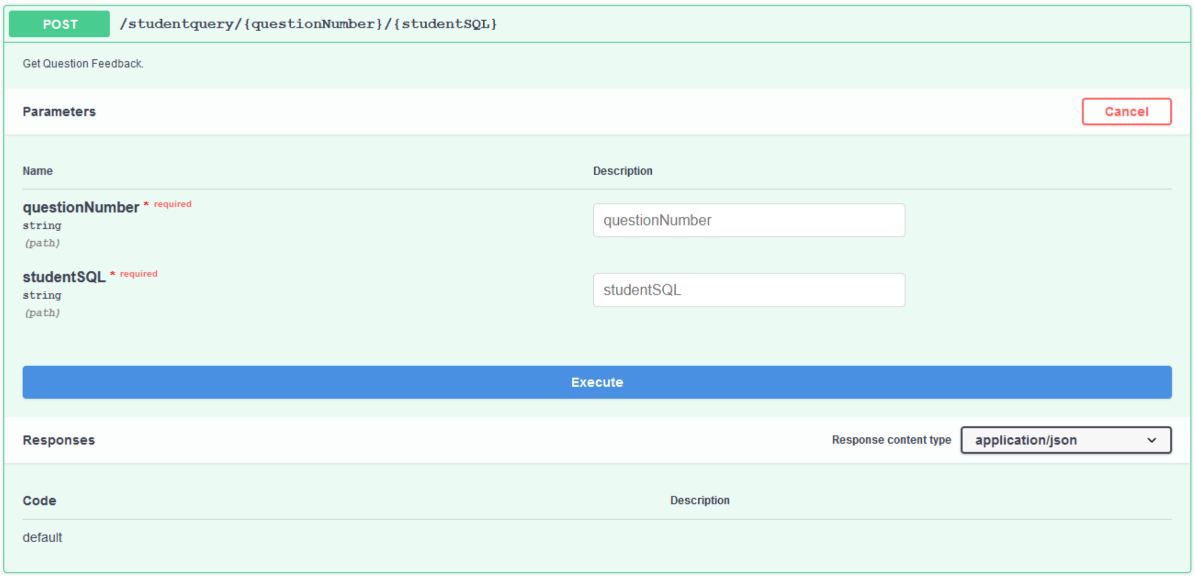
No parameters

Responses — Response content type: application/json

Code	Description
default	

Figure 17: Assignment Question Get Request

After the student retrieves the questions, they can interact with the second endpoint (See Figure 18).For this endpoint the student is required to pass in both the assignment question number (that they wish to answer) and their own SQL query attempt. Note: When passing in their SQL queries, students will have to change any forward slash symbols (/) into dollar signs ($) for the system to run correctly (this is due to the way that JSON is transported to the API, forward slashes are recognised as an extension of the endpoint and end up breaking the post request. These dollar signs will get converted back to forward slashes once the student query is successfully in the API via the unserialize function). After this the student can hit execute and they will receive a response providing feedback. This feedback will outline how their query compared to the selected test query with regards to columns, rows, time to execute, order of columns, order of rows, values, joins and the use of top. The student can then edit their query until they receive the desired feedback.


POST /studentquery/{questionNumber}/{studentSQL}

Get Question Feedback.

Parameters

Name	Description
questionNumber * required string (path)	questionNumber
studentSQL * required string (path)	studentSQL

Responses — Response content type: application/json

Code	Description
default	

Figure 18: Student Query Attempt Post Request