SDEV400 6381: Secure Programming in the Cloud

Homework 2: File Reading Application

Mark Kardash

Professor Nathan Braun

University of Maryland Global Campus

8 February 2024

In this document, I will be showing the creation and handling of tables for the Homework 2 assignment of my SDEV400 course. This will be done via AWS CLI.

The first thing I will need to do is create the “Sensors” table. For this, I will need to specify the table name, table description, and various keys needed to create it.

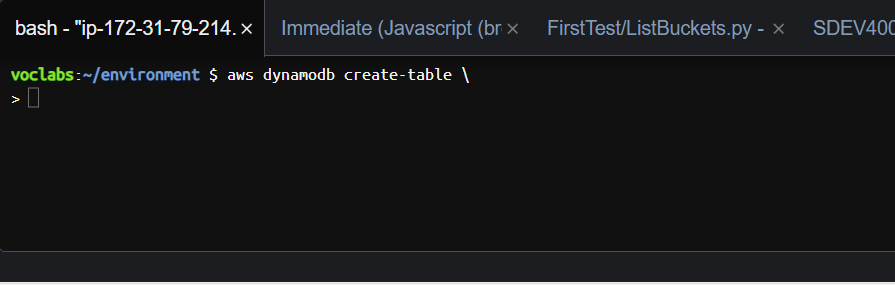


Figure 1: Initializing “create-table” command.

The first step in getting the program to work is initializing the aws “create-table” command. This will allow us to write in exactly what attributes and elements we need for our table.

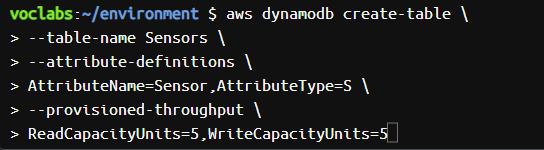


Figure 2: Code to Create “Sensors” table

Above, one can see the code used to create the “Sensors” table. In it, we name attributes, assign types to them, and set proportions, in the form of read and write capacity units. Per assignment instructions, we also set the KeyType to “HASH”.

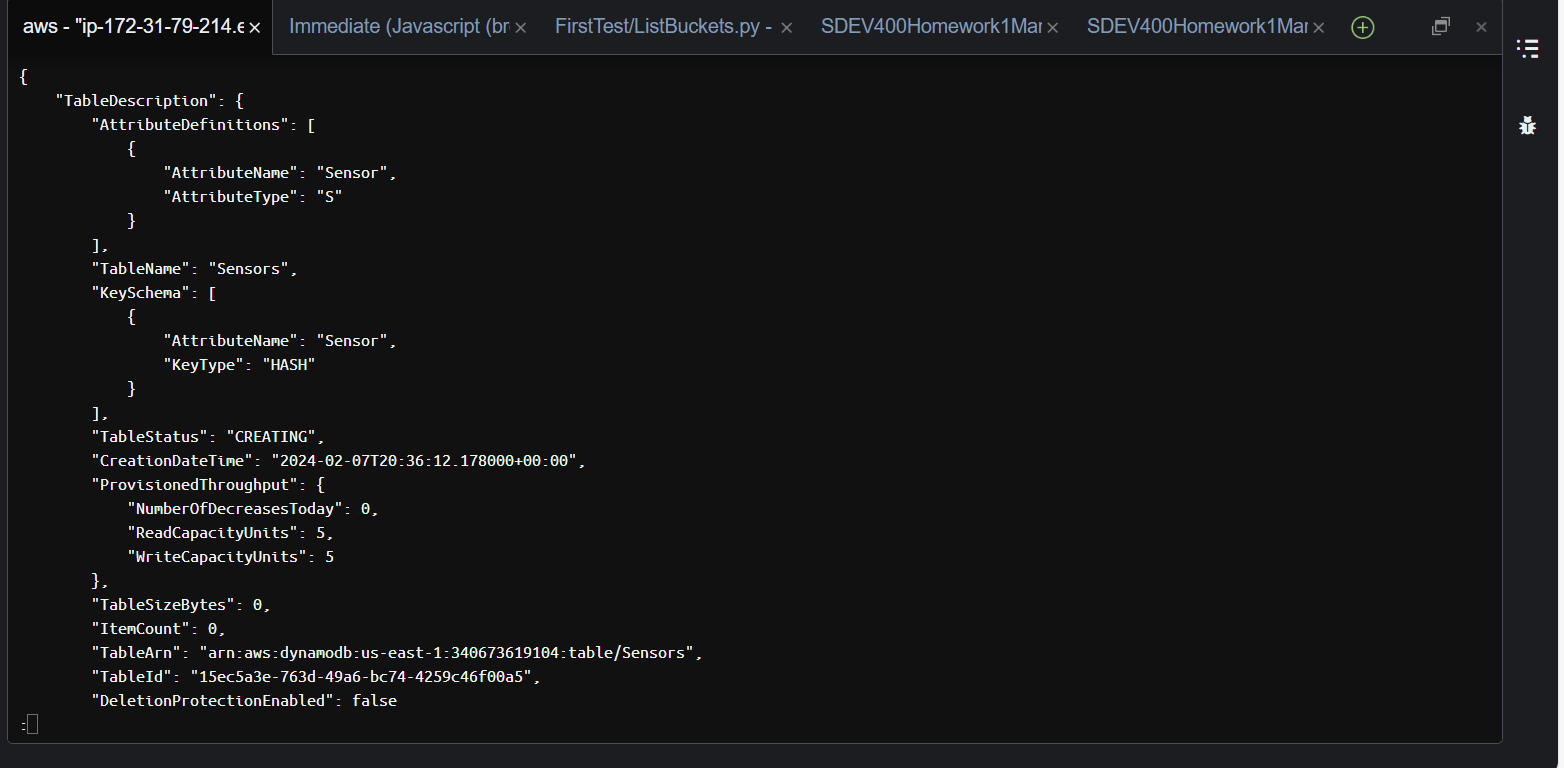


Figure 3: “Sensors” table created.

As can be seen in the above screenshot, the “Sensors” table has now been successfully created.



Figure 4: Command to put all sensors into their assigned table.

The above command will be used to place all the sensors we created in the “json” file in their appropriate table. At first, executing this command gave me a “No Such file or Directory” error.



Figure 5: “Invalid type for parameter” error.

However, I eventually realized that I had simply not placed the file in the same directory as I currently was in. As soon as I did, I corrected that mistake. Only re-running the command again gave me yet another error. This time, stating that the “type” of my item was incorrect.

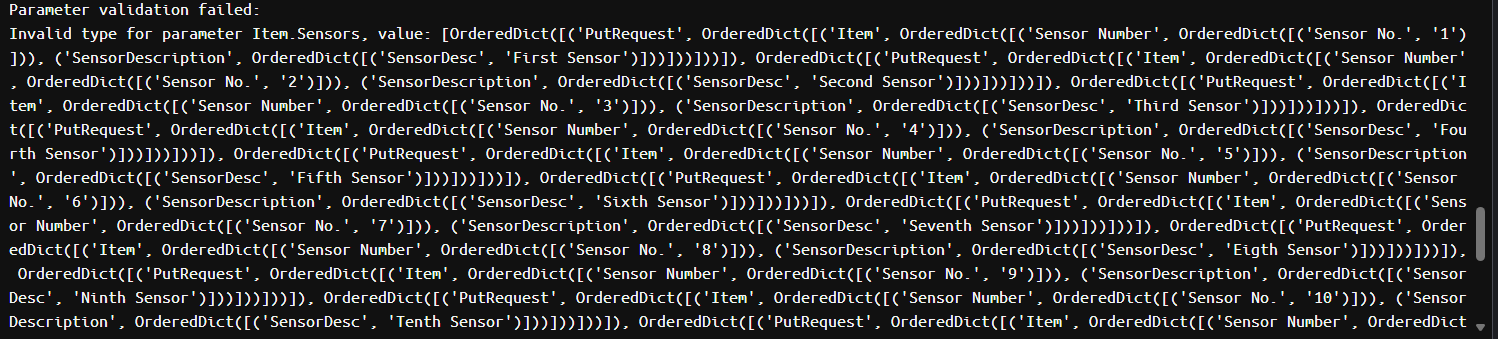


Figure 6: Invalid Type for Parameter Error

The above error seemed to indicate that there was something wrong with my “.json” file as a whole. However, after a long and exhaustive time, I still was not able to figure out what exactly the issue was, and was forced to move to the next step.

The next step was actually very simple, though it was quite uncertain without the previous step being properly completed. We needed to make the system print the contents of the Sensors table. Since I did not manage to load the sensors into it correctly, I expected the system to open an empty table.



Figure 7: “Scan” command to print data

To see the contents of the “Sensors” table, we use a command known as “scan”, which will get the contents printed out.

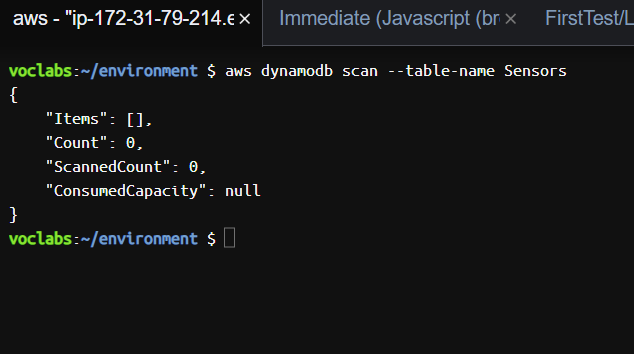


Figure 8: Empty Table

As expected, we see an empty table will a count of 0, making this test a failure.

Since we are now done with everything related to the “Sensors” table, we must now move on to the second part of the assignment: Creating the Courses table. The application developed for that can be seen below.



Figure 9: Building “Courses” table.

After importing all the necessary elements, we first build the “Courses” table itself, setting its attributes and types.



Figure 10: Designing the user interface

After adding an exception statement to the “Courses” table in case of an error, we begin designing the user interface. The UI is designed in such a way that it will continue prompting the user to enter details for a course, until they input invalid data (ex. Non-existing course code). If that happens, the program will catch an exception, displaying an error message.

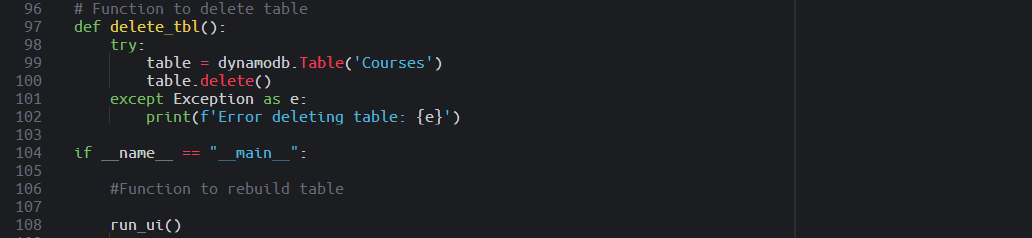


Figure 11: Functions to Delete and Rebuild Table

Finally, a function intended to delete the table, and one to rebuild it, just in case, have been created at the end of the UI.

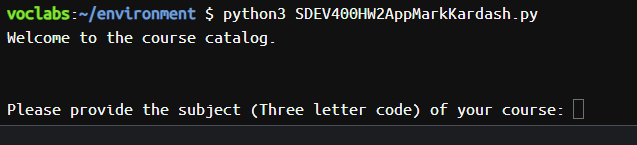


Figure 12: Successful Program Start

As we can see from the screenshot above, the program has started successfully. However, we still have not done one vital task, and that is loading the data from the .json file into the “Courses” table. That is exactly what will be done right now.



Figure 13: Command to fill courses table

To get the data from the json file with the courses, we use a similar command to the one we wrote in our failed attempt to get the sensors.

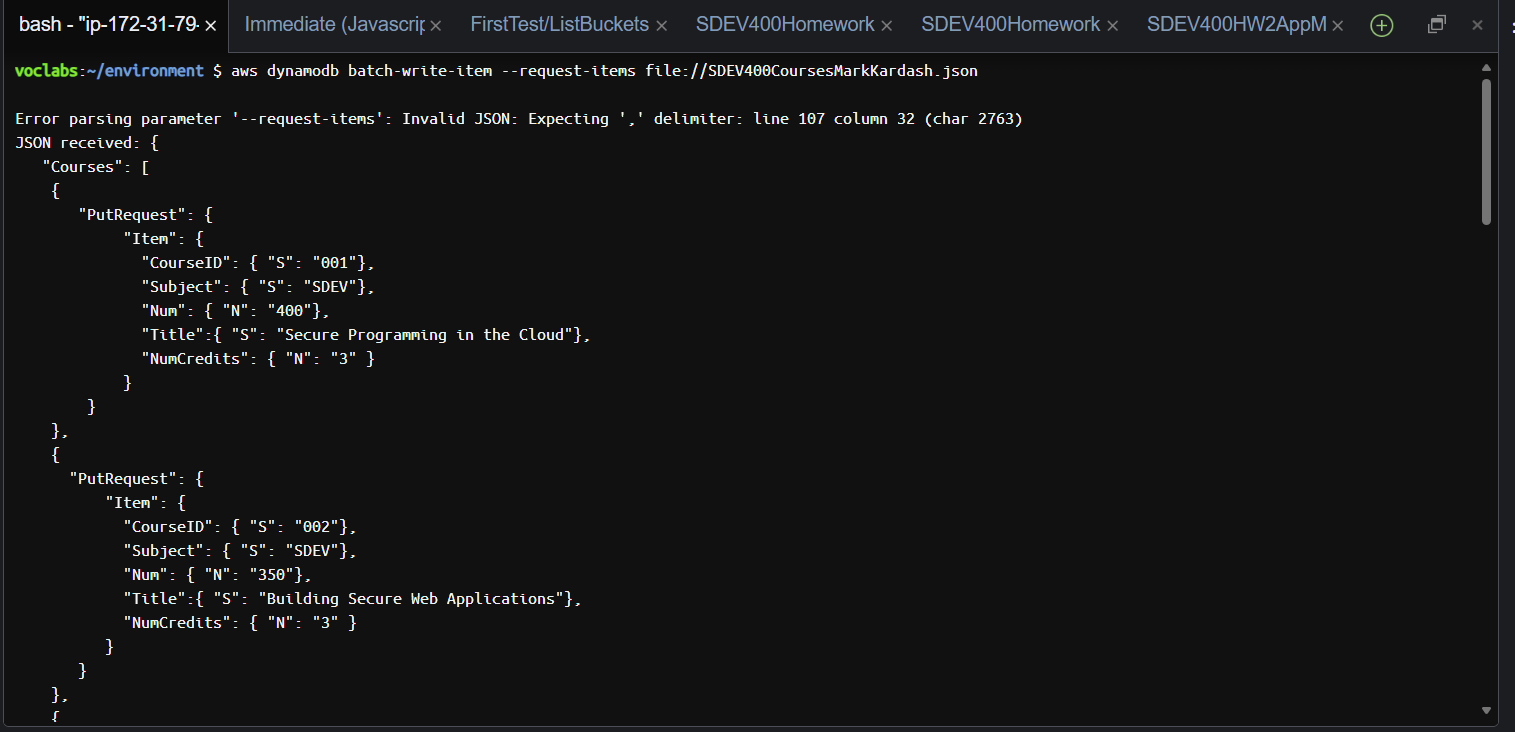


Figure 14: Courses file imported successfully

Surprisingly, this time, the system seems to have recognized the file, despite throwing a parsing error.

With all of this done, it is time to test the functionality of the app itself. We shall now run it, and enter our course selection.

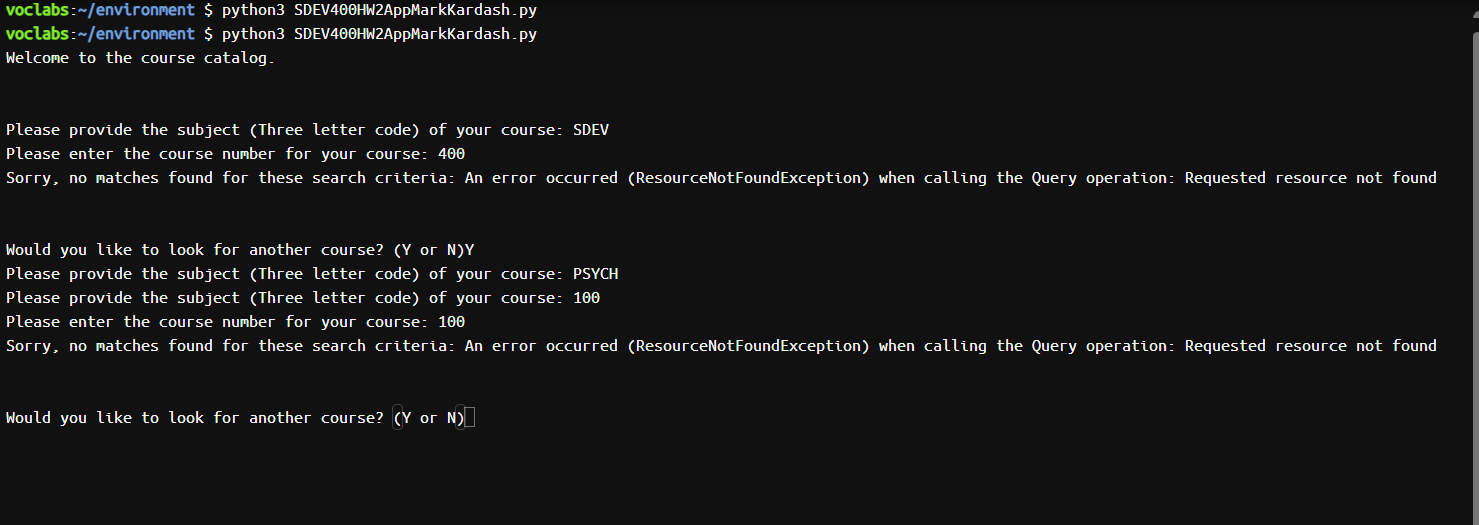


Figure 15: Partial Success of Run

After running the program, I spotted several errors in it. It seems that it did not recognize the “.json” file, after all, as I was given an error message even after entering courses that were definitely on file. After the second time searching for a course, it also repeated the subject prompt two times. Despite this, I consider this run partially successful, as the program showed some of the messages it was supposed to, and kept prompting the user to search for a course, just as expected.