

## BuffGrades\_Part\_6

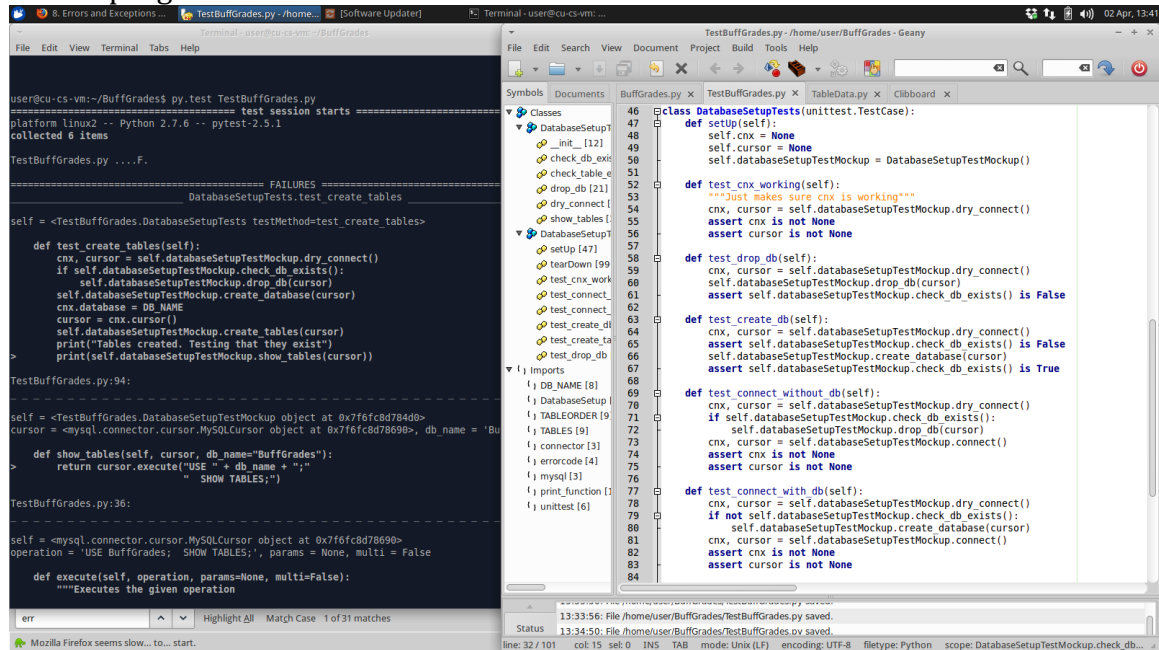
### Title: BuffGrades

**Vision:** To show students how well they are doing, and how well they need to do to be successful.

**Who:** Michael Asnes, Nicholas Riffel, Nishesh Shukla, David Becker, Jeremy Chrestionson.

### Automated Tests:

Test in progress:



The screenshot shows a development environment with a terminal window on the left and a code editor on the right. The terminal displays the output of running a test suite, showing a failure in the `test_create_tables` method. The code editor shows the implementation of the `DatabaseSetupTests` class, which includes methods for setting up the database, creating tables, and testing the database connection.

```
user@cu-cs-vm:~/BuffGrades$ py.test TestBuffGrades.py
===== test session starts =====
platform linux2 -- Python 2.7.6 -- pytest-2.5.1
collected 6 items

TestBuffGrades.py ....F.

===== FAILURES =====
DatabaseSetupTests.test_create_tables

self = <TestBuffGrades.DatabaseSetupTests testMethod=test_create_tables>

def test_create_tables(self):
    cnx, cursor = self.databaseSetupTestMockup.dry_connect()
    if self.databaseSetupTestMockup.check_db_exists():
        self.databaseSetupTestMockup.drop_db(cursor)
    self.databaseSetupTestMockup.create_database(cursor)
    cnx.database = DB_NAME
    cursor = cnx.cursor()
    self.databaseSetupTestMockup.create_tables(cursor)
    print("Tables created. Testing that they exist")
    print(self.databaseSetupTestMockup.show_tables(cursor))

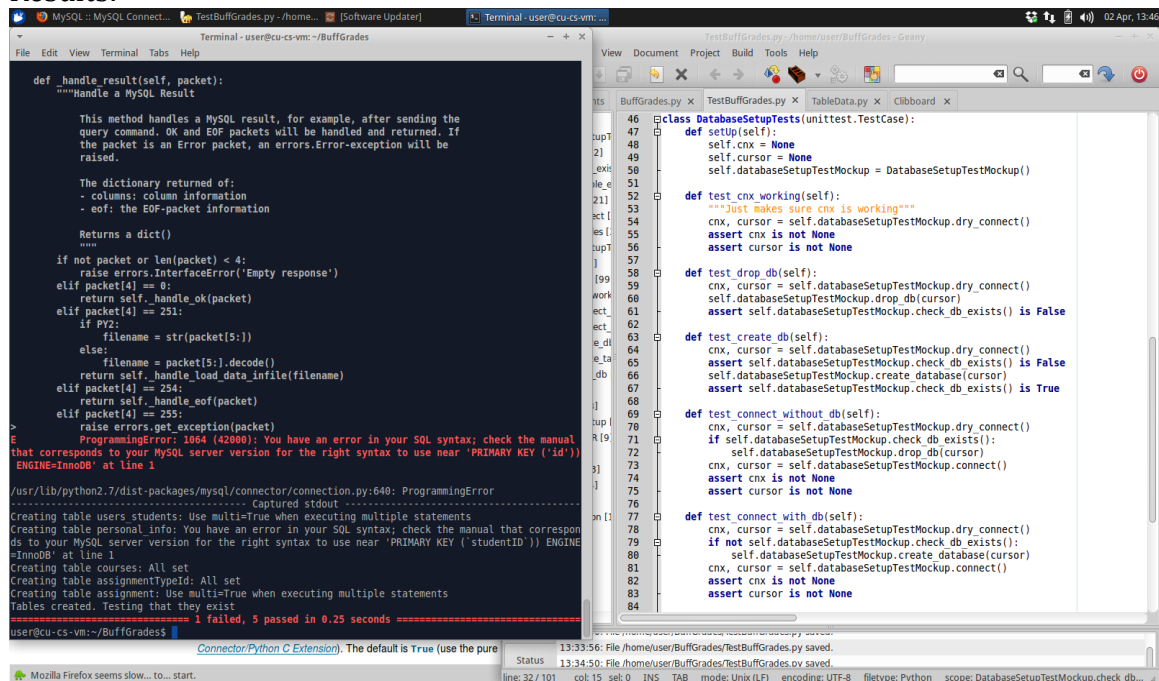
TestBuffGrades.py:94:
-----
self = <TestBuffGrades.DatabaseSetupTestMockup object at 0x7f6fc8d784d0>
cursor = <mysql.connector.cursor.MySQLCursor object at 0x7f6fc8d78690>, db_name = 'BuffGrades'

def show_tables(self, cursor, db_name="BuffGrades"):
    return cursor.execute("USE " + db_name + ";"
                          " SHOW TABLES;")

TestBuffGrades.py:36:
-----
self = <mysql.connector.cursor.MySQLCursor object at 0x7f6fc8d78690>
operation = 'USE BuffGrades; SHOW TABLES;', params = None, multi = False

def execute(self, operation, params=None, multi=False):
    """Executes the given operation
```

### Results:



The screenshot shows a development environment with a terminal window on the left and a code editor on the right. The terminal displays the output of running a test suite, showing a failure in the `test_create_tables` method. The code editor shows the implementation of the `DatabaseSetupTests` class, which includes methods for setting up the database, creating tables, and testing the database connection.

```
def handle_result(self, packet):
    """Handle a MySQL Result

    This method handles a MySQL result, for example, after sending the
    query command, OK and EOF packets will be handled and returned. If
    the packet is an Error packet, an errors.Error-exception will be
    raised.

    The dictionary returned of:
    - columns: column information
    - eof: the EOF-packet information

    Returns a dict()
    """
    if not packet or len(packet) < 4:
        raise errors.InterfaceError('Empty response')
    elif packet[4] == 0:
        return self.handle_ok(packet)
    elif packet[4] == 251:
        if PY2:
            filename = str(packet[5:])
        else:
            filename = packet[5:].decode()
        return self.handle_load_data_infile(filename)
    elif packet[4] == 254:
        return self.handle_eof(packet)
    elif packet[4] == 255:
        raise errors.get_exception(packet)
    ProgrammingError: 1064 (42000): You have an error in your SQL syntax; check the manual
    that corresponds to your MySQL server version for the right syntax to use near 'PRIMARY KEY ('id')'
    ENGINE=InnoDB' at line 1

/usr/lib/python2.7/dist-packages/mysql/connector/connection.py:640: ProgrammingError
===== 1 failed, 5 passed in 0.25 seconds =====

user@cu-cs-vm:~/BuffGrades$
```

We used Py.test as our automatic testing software, and placed our test code into TestBuffGrades.py . We have written tests that check that the database was setup properly and that we can connect to it without any issues. Simply run py.test testgrades.py after compiling and the tests should run.

### User Acceptance Tests:

Project Name: Buff Grades						
<b>User Acceptance Test Case 1</b>						
Test Case ID: BuffGrades_Test_001			Test Designed by: Nicholas Riffel, Nishesh Shukla, Michael Asnes, David Becker, Jeremy Chrestionson			
Test Priority (Low/Medium/High): High			Test Designed date: 03/31/15			
Module Name: BuffGrades User Check			Test Executed by:			
Test Title: User Check			Test Execution date:			
Description: Making sure user is a verified user and has already inputted information, else queries for their information regarding classes and grades.						
Pre-conditions: User has valid username						
Dependencies: User has, or has not actually inputted information.						
Step	Test Steps	Test Data	Expected Result	Actual Results	Status (Pass/Fail)	Notes
1	Open up GUI	Button Exists	It Opens GUI			
Post Conditions: User is recognized by the application/GUI.						

Project Name: BuffGrades	
<b>User Acceptance Test Case 2</b>	
Test Case ID: BuffGrades_Test_002	Test Designed by: Nicholas Riffel, Nishesh Shukla, Michael Asnes, David Becker, Jeremy Chrestionson
Test Priority (Low/Medium/High): High	Test Designed date: 03/31/15
Module Name: User Data Input	Test Executed by:
Test Title: Add A Course	Test Execution date:

## BuffGrades\_Part\_6

Description: User adds the name of a course that he/she is currently enrolled in.						
Pre-conditions: GUI opened with verified user						
Dependencies:						
Step	Test Steps	Test Data	Expected Result	Actual Results	Status (Pass/Fail)	Notes
1	Click on "Add a Course"		Entry box appears			
2	Enter the course name	"Test Course"	GUI displays "Test Course" under Courses			
Post Conditions: Course is now saved within database						

Project Name:BuffGrades						
User Acceptance Test Case 3						
Test Case ID: BuffGrades_Test_003				Test Designed by: Nicholas Riffel, Nishesh Shukla, Michael Asnes, David Becker, Jeremy Chrestionson		
Test Priority (Low/Medium/High): High				Test Designed date: 03/31/15		
Module Name: User Data Input				Test Executed by:		
Test Title: Add Assignment Type				Test Execution date:		
Description: User adds an assignment type.						
Pre-conditions: Test Course has been added						
Dependencies:						
Step	Test Steps	Test Data	Expected Result	Actual	Status	Notes

## BuffGrades\_Part\_6

				Results	(Pass/Fail)	
1	Click "Add Assignment Type"		Entry box appears			
2	Enter assignment type name	"Test Assignment Type"	GUI displays "Test Assignment Type" under Assignment Types.			
3	Change "Test Assignment Type" Weight	"1.00"	"Test Assignment Type" Displays Weight of "1.00"			
Post Conditions: GUI displays Test Assignment Type under Assignment Type.						

Project Name: BuffGrades						
User Acceptance Test Case 4						
Test Case ID: BuffGrades_Test_004				Test Designed by: Nicholas Riffel, Nishesh Shukla, Michael Asnes, David Becker, Jeremy Chrestionson		
Test Priority (Low/Medium/High): High				Test Designed date: 03/31/15		
Module Name: User Data Input				Test Executed by:		
Test Title: Add Assignment				Test Execution date:		
Description: User adds assignments and grade of each assignment						
Pre-conditions: User has Added Assignment Types						
Step	Test Steps	Test Data	Expected Result	Actual Results	Status (Pass/Fail)	Notes
1	Click on "Add Assignment"		Entry box appears			
2	Enter Assignment Name	"Test Assignment"	Test Assignment appears under Test Assignment Type			
3	Enter Test Assignment	"0.49"	"Test Assignment" Displays grade of			

## BuffGrades\_Part\_6

	Grade		"0.49". "Test Course" Displays grade of "0.49".			
Post Conditions: GUI displays Test Assingment						

Project Name: BuffGrades						
<b>User Acceptance Test Case 5</b>						
Test Case ID: BuffGrades_Test_005			Test Designed by: Nicholas Riffel, Nishesh Shukla, Michael Asnes, David Becker, Jeremy Chrestionson			
Test Priority (Low/Medium/High):High			Test Designed date: 03/31/15			
Module Name: Personal Overview			Test Executed by:			
Test Title: GPA Calculation			Test Execution date:			
Description: User determines the weight of the Assignment						
Pre-conditions: User has created and set a grade for assignment.						
Dependencies: GPA calculation algorithm						
Step	Test Steps	Test Data	Expected Result	Actual Results	Status (Pass/Fail)	Notes
1	Click on "Overview"		GUI Displays Home page and GPA.			
Post Conditions: User is able to ascertain what overall GPA is currently based on current classes entered and 4.0 scale.						

BuffGrades\_Part\_6

**VCS** (Link): <https://github.com/masnes/BuffGrades>