ECE 6400 NTTF Presentation

Core Testing & Integration Team

Kody Abbott
Yahia Aly

Agenda

- Team Purpose
- Technology Discussion
- Assignment 1 Discussion
- Assignment 2 Discussion
- Assignment 3 Discussion
- Live Demo
- Future Improvements

Purpose and Goals

Setting Up Automation and Monitoring

Integrating Separate Code Modules

Overseeing Integration/System Testing

Ensuring Code Functionality

Key Technologies

Jenkins

Integration Pipelining and Module Status

GitHub

General Purpose Code Repository

Docker

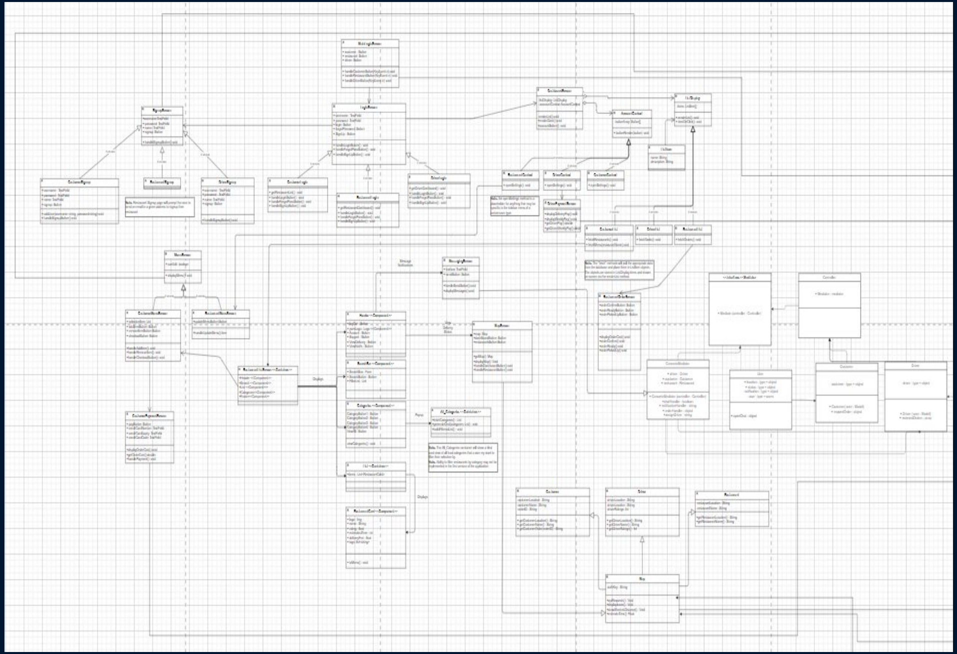
Deployment System Capable of Executing Programs

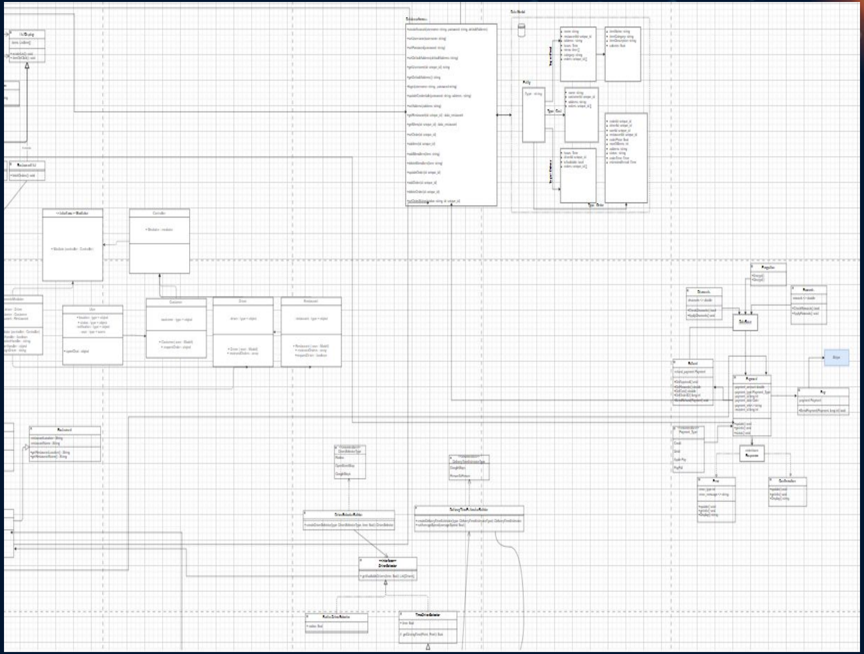
Jest/Pytest

Testing Frameworks for Main Languages

Assignment 1 Discussion

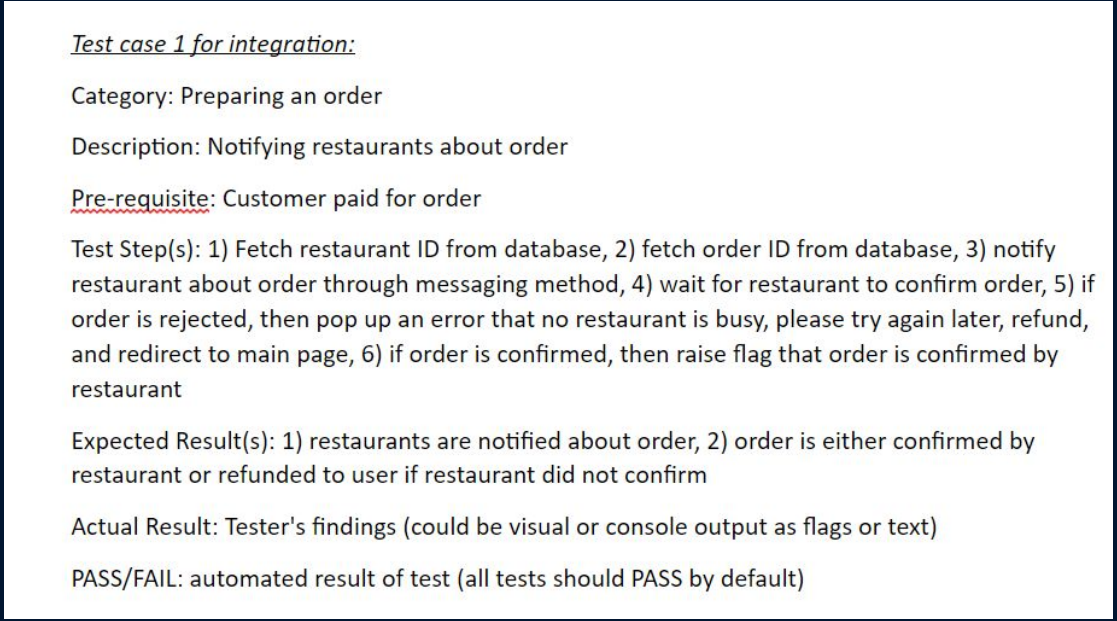
Huge UML diagram for integration!





Assignment 1 Discussion Cont'd

Unit test cases for Integration



Test case 1 for integration:

Category: Preparing an order

Description: Notifying restaurants about order

Pre-requisite: Customer paid for order

Test Step(s): 1) Fetch restaurant ID from database, 2) fetch order ID from database, 3) notify restaurant about order through messaging method, 4) wait for restaurant to confirm order, 5) if order is rejected, then pop up an error that no restaurant is busy, please try again later, refund, and redirect to main page, 6) if order is confirmed, then raise flag that order is confirmed by restaurant

Expected Result(s): 1) restaurants are notified about order, 2) order is either confirmed by restaurant or refunded to user if restaurant did not confirm

Actual Result: Tester's findings (could be visual or console output as flags or text)

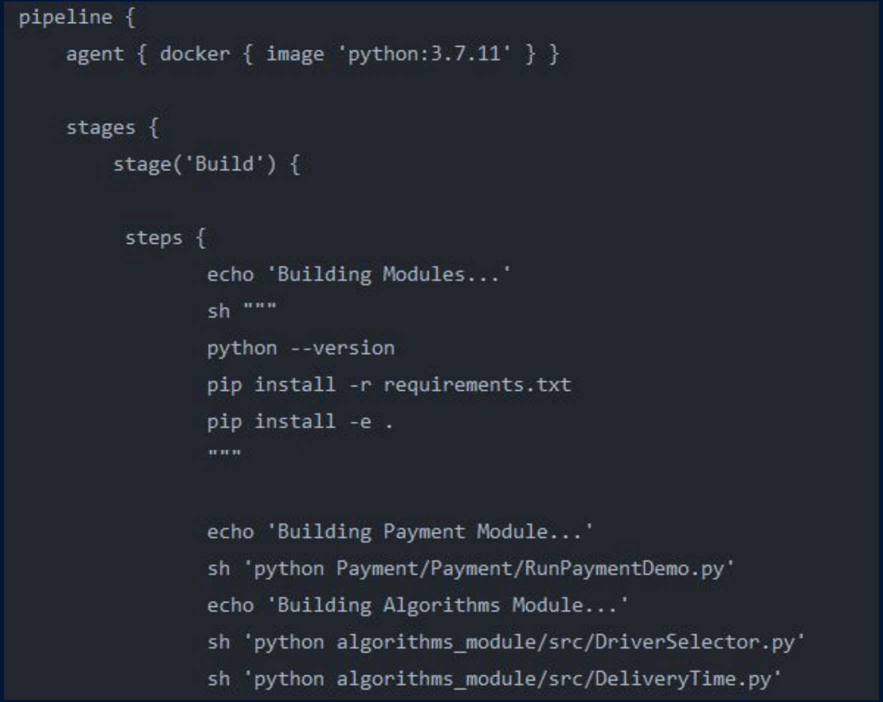
PASS/FAIL: automated result of test (all tests should PASS by default)

Other test cases examples

- Getting available drivers
- Getting free driver
- Getting time estimate for order
- Real-time mapping of order
- Customer received order
- Placing an Order
- Checking order status
- Logging in to user account

Assignment 2 Discussion

Minimal Viable Product



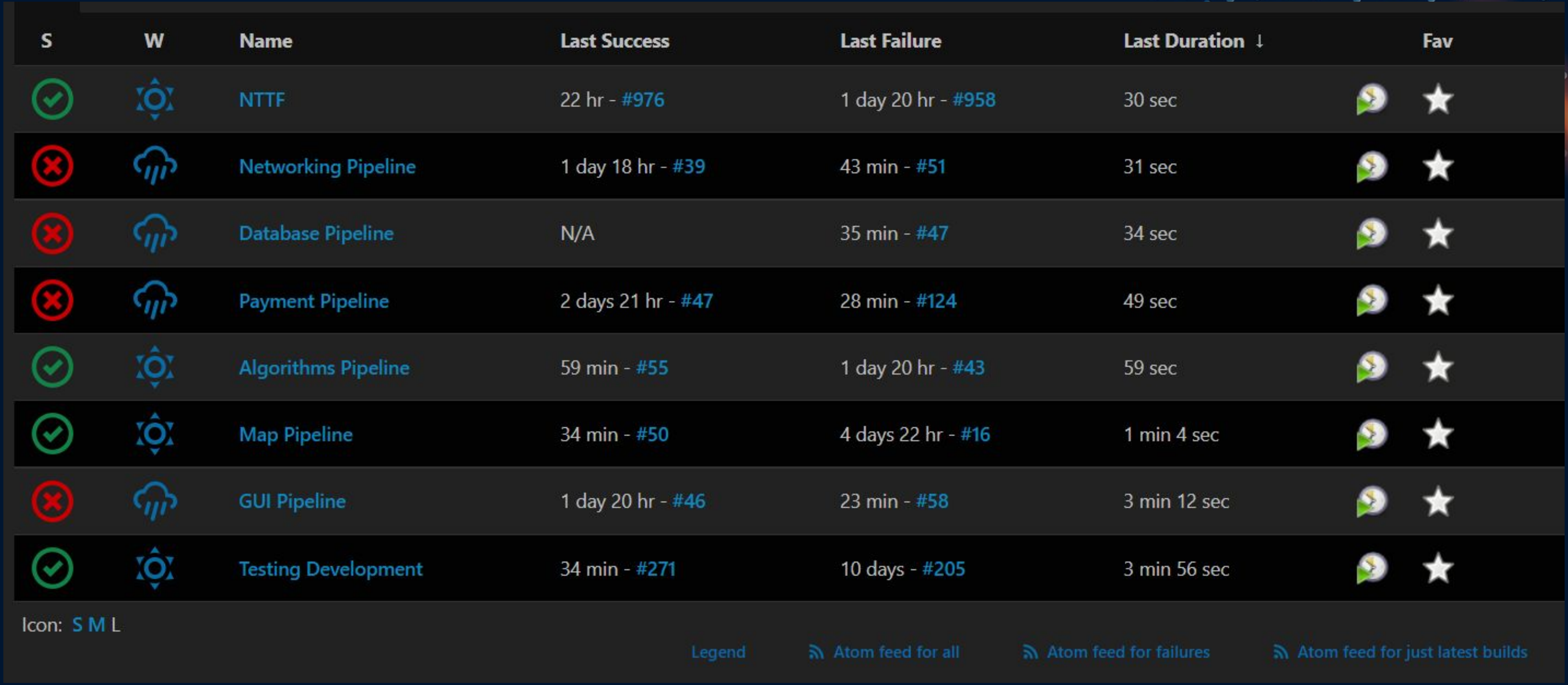
```
pipeline {
    agent { docker { image 'python:3.7.11' } }

    stages {
        stage('Build') {

         steps {
                echo 'Building Modules...'
                sh """
                python --version
                pip install -r requirements.txt
                pip install -e .
                """


                echo 'Building Payment Module...'
                sh 'python Payment/Payment/RunPaymentDemo.py'
                echo 'Building Algorithms Module...'
                sh 'python algorithms_module/src/DriverSelector.py'
                sh 'python algorithms_module/src/DeliveryTime.py'
```

- Creating and Configuring Integration Pipeline
- Ensuring Modules could Build and Test Successfully
- Debugging Jenkins and Jenkinsfile
- Resolving Design Conflicts with Other Teams

Assignment 3 Discussion

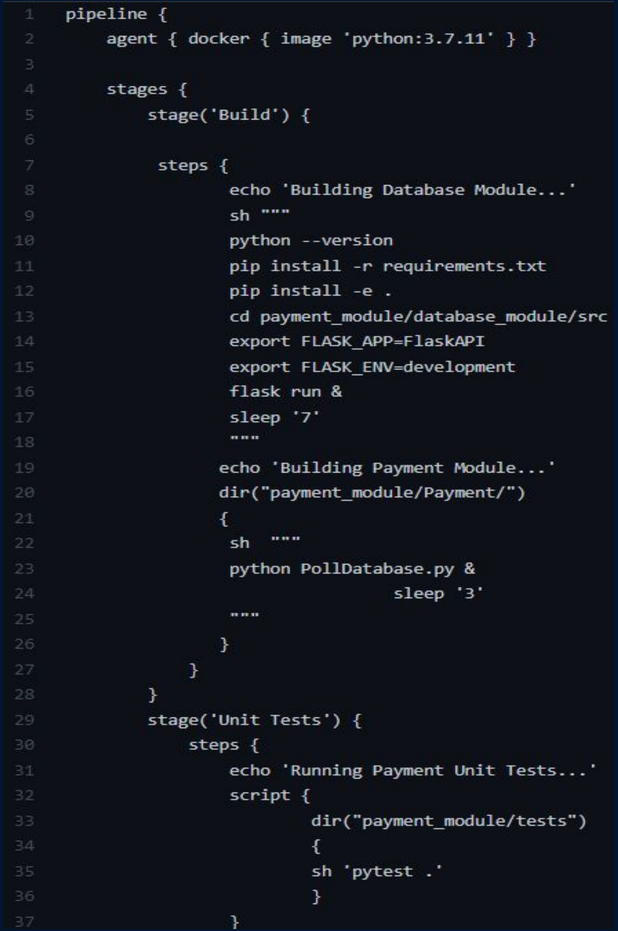
- Jenkins pipelines for each team’s build and unit testing
- Integration tests for each team with the database module
- Systems Testing
- Deployment research
- Several challenges in unit and integration testing => difficulty implementing system tests

Assignment 3 Discussion Cont'd

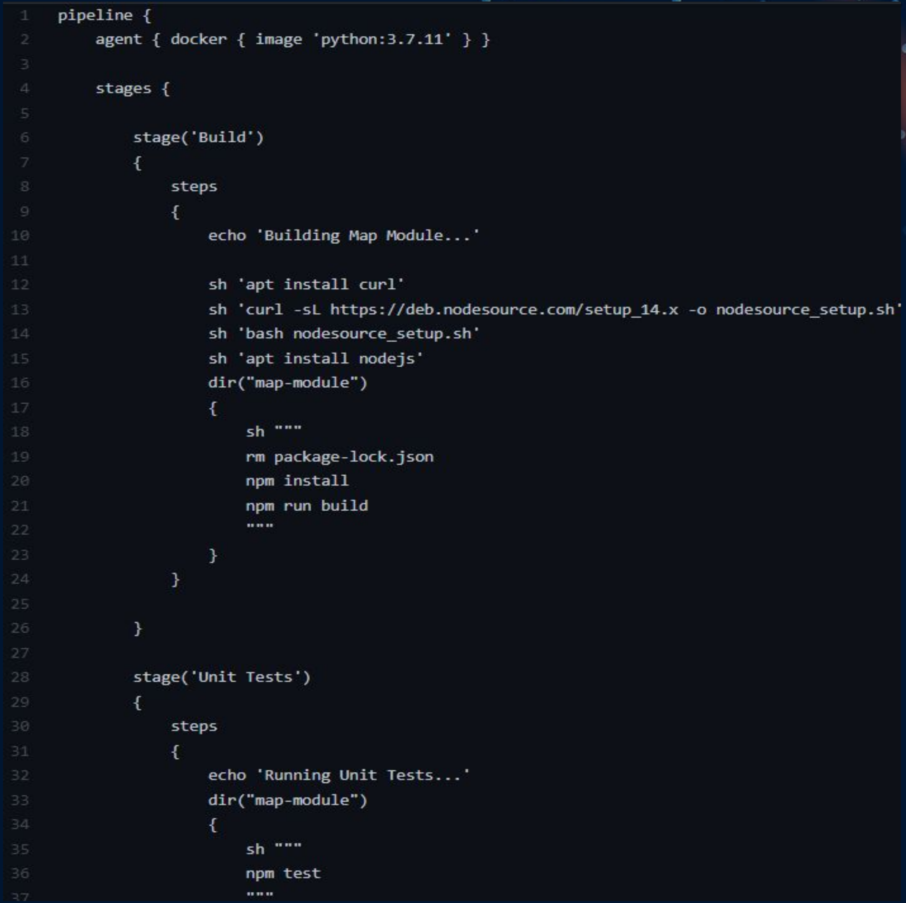


S	W	Name	Last Success	Last Failure	Last Duration ↓	Fav
✓	☀	NTTF	22 hr - #976	1 day 20 hr - #958	30 sec	★
✗	🌧	Networking Pipeline	1 day 18 hr - #39	43 min - #51	31 sec	★
✗	🌧	Database Pipeline	N/A	35 min - #47	34 sec	★
✗	🌧	Payment Pipeline	2 days 21 hr - #47	28 min - #124	49 sec	★
✓	☀	Algorithms Pipeline	59 min - #55	1 day 20 hr - #43	59 sec	★
✓	☀	Map Pipeline	34 min - #50	4 days 22 hr - #16	1 min 4 sec	★
✗	🌧	GUI Pipeline	1 day 20 hr - #46	23 min - #58	3 min 12 sec	★
✓	☀	Testing Development	34 min - #271	10 days - #205	3 min 56 sec	★

Assignment 3 Discussion Cont'd



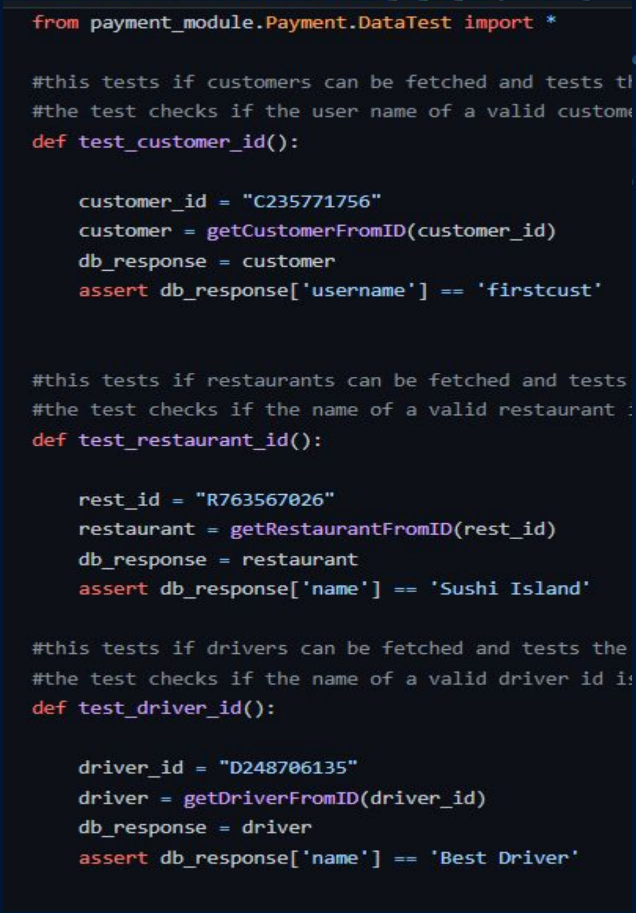
```
pipeline {
    agent { docker { image 'python:3.7.11' } }

    stages {
        stage('Build') {

         steps {
                echo 'Building Database Module...'
                sh """
                python --version
                pip install -r requirements.txt
                pip install -e .
                cd payment_module/database_module/src
                export FLASK_APP=FlaskAPI
                export FLASK_ENV=development
                flask run &
                sleep '7'
                """
               echo 'Building Payment Module...'
               dir("payment_module/Payment/")
               {
                sh  """
                python PollDatabase.py &
                                sleep '3'
                """
               }
            }
        }
        stage('Unit Tests') {
            steps {
                echo 'Running Payment Unit Tests...'
                script {
                        dir("payment_module/tests")
                        {
                        sh 'pytest .'
                        }
                }
```



```
pipeline {
    agent { docker { image 'python:3.7.11' } }

    stages {

        stage('Build')
        {
            steps
            {
                echo 'Building Map Module...'

                sh 'apt install curl'
                sh 'curl -sL https://deb.nodesource.com/setup_14.x -o nodesource_setup.sh'
                sh 'bash nodesource_setup.sh'
                sh 'apt install nodejs'
                dir("map-module")
                {
                    sh """
                    rm package-lock.json
                    npm install
                    npm run build
                    """
                }
            }

        }

        stage('Unit Tests')
        {
            steps
            {
                echo 'Running Unit Tests...'
                dir("map-module")
                {
                    sh """
                    npm test
                    """
```

Assignment 3 Discussion Cont’d

- Integration tests required modifying the import structure to suit the testing framework needs (Pytest)
- Not all modules were tested because of challenges such as builds failing, Github merging, error with DB interfacing - only the payment team was tested in terms of integration by assignment 3 submission



```
from payment_module.Payment.DataTest import *

#this tests if customers can be fetched and tests tl
#the test checks if the user name of a valid custome
def test_customer_id():

    customer_id = "C235771756"
    customer = getCustomerFromID(customer_id)
    db_response = customer
    assert db_response['username'] == 'firstcust'


#this tests if restaurants can be fetched and tests
#the test checks if the name of a valid restaurant i
def test_restaurant_id():

    rest_id = "R763567026"
    restaurant = getRestaurantFromID(rest_id)
    db_response = restaurant
    assert db_response['name'] == 'Sushi Island'

#this tests if drivers can be fetched and tests the
#the test checks if the name of a valid driver id is
def test_driver_id():

    driver_id = "D248706135"
    driver = getDriverFromID(driver_id)
    db_response = driver
    assert db_response['name'] == 'Best Driver'
```

Assignment 3 Discussion Cont'd

- Deployment recommendation is Docker container
- Individual container for each module
- Allows for further testing and examinations
- Easy learning curve and quicker than VMs
- Good for versioning

Live Demo Time!

“If you automate a mess, you get an automated mess.” - Rod Michael

Future Goals/Improvements

- Maintain communication with teams on a timely basis
- Ensure that unit tests pass in Jenkins for all modules
- Ensure teams have the freshest code in their module-specific branch
- Implement remaining integration tests
- Implement system-level tests
- Try deployment if possible

THANKS, any questions?

yhaly@mun.ca
kra646@mun.ca