**CS 340 README**

**About the Project/Project Title**

To interface with databases, CRUD functionality is necessary. To enable build and read capability for the database, I had to construct a Python module in a PY file using object-oriented programming techniques. The Python code has to be imported as a module by other Python scripts in order to facilitate code reuse. In order to call and test the create and read instances of CRUD capability, I had to write a Python testing script in Jupyter Notebooks that imports the CRUD Python module.

**Motivation**

The backend is powered by MongoDB, a NoSQL database application with quick performance and direct data access. The documents in this program are stored using JSON-based formatting, which can manage high-volume loads and enable the application to scale both vertically and horizontally. Through our data-driven approach, Dash, an open-source Python framework frequently used for analytical web applications, enables the simplicity of a front-end interface. We utilize Python to connect everything, providing simple dynamic typing capabilities and a quick way to link various parts of a web application.

**Getting Started**

**Installation**

You must first install a local copy of the complete project to begin working on it. You must make sure that the most recent Python version is installed on your machine, as well as Mongo DB. Once your project is properly formatted, we may start by setting up database access. To build a new Mongo database and import the CSV file containing the corporate data, you should use your current terminal. Last but not least, make sure you create a user with read and write permissions to this database.

**Usage**

The project at hand can handle all four basic CRUD operations on the database. With MongoDB, we can create documents, read from the database, and update an individual or multiple documents, while also having the ability to delete a single or multiple sets of documents at once. The update and delete methods in the module can take in a third parameter known as “count” which can determine if a single or multiple amounts of documents are needing to be operated on.

**Code Example-** This is part of the code for the setup and the photo. You would have to edit the username and password to use this. Also to make it more personal you would have to adjust the name “Jasmine Sadler” with your own name.

###########################

# Data Manipulation / Model

###########################

# FIX ME change for your username and password and CRUD Python module name

username = "aacuser"

password = "jasmine2023"

shelter = AnimalShelter(username, password)

# class read method must support return of cursor object

df = pd.DataFrame.from\_records(shelter.read({}))

#########################

# Dashboard Layout / View

#########################

app = JupyterDash('Jasmine Sadler SNHU')

#FIX ME Add in Grazioso Salvares logo

image\_filename = 'Grazioso Salvare Logo.png' # replace with your own image

encoded\_image = base64.b64encode(open(image\_filename, 'rb').read())

app.layout = html.Div([

html.Div(id='hidden-div', style={'display':'none'}),

html.Center(html.Img(src='data:image/png;base64,{}'.format(encoded\_image.decode()))),

html.Center(html.B(html.H1('Jasmine Sadler - CS-340 Dashboard'))),

html.Center(html.P('Select up to five options from the table for the map')),

html.Hr(),

html.Div(

#Radio Items to select the rescue filter options

dcc.RadioItems(

id='filter-type',

# create the labels based on the Grazioso requirements

options=[

{'label': 'Water Rescue', 'value': 'WR'},

{'label': 'Mountain/Wilderness Rescue', 'value': 'MWR'},

{'label': 'Disaster Rescue/Individual Tracking', 'value': 'DRIT'},

{'label': 'Reset - returns unfiltered state', 'value': 'RESET'}

],

value='RESET',

labelStyle={'display': 'inline-block'}

)

),

html.Hr(),

dt.DataTable(

id='datatable-id',

columns=[

{"name": i, "id": i, "deletable": False, "selectable": True} for i in df.columns

],

data=df.to\_dict('records'),

**Tests**  - This was the test to make sure my CRUD function worked correctly.

from animalShelter import AnimalShelter

a = AnimalShelter("aacuser", "jasmine2023")

testData = {

'1': 4,

'age\_upon\_outcome': '10 years',

'animal\_id': 'ABCD',

'animal\_type': 'testType',

'breed': 'testBreed',

'color': 'testColor',

'date\_of\_birth': 'testDOB',

'datetime': '2021-09-25 12:04:01',

'monthyear': '2021-09-25 12:04:02',

'name': 'testName',

'outcome\_subtype': 'testSubType',

'sex\_upon\_outcome': 'testOutcome',

'location\_lat': 25.55,

'location\_long': -25.55,

'age\_upon\_outcome\_in\_weeks': '323.433'

}

sampleTarget = {'animal\_type': 'testType'}

sampleUpdate = {'$set': {'name': '22'}}

sampleUpdateTwo = {'$set': {

'name': '21',

'animal\_type': 'Dog'

}}

sampleDelete = {

'name': '21'

}

a.create(testData)

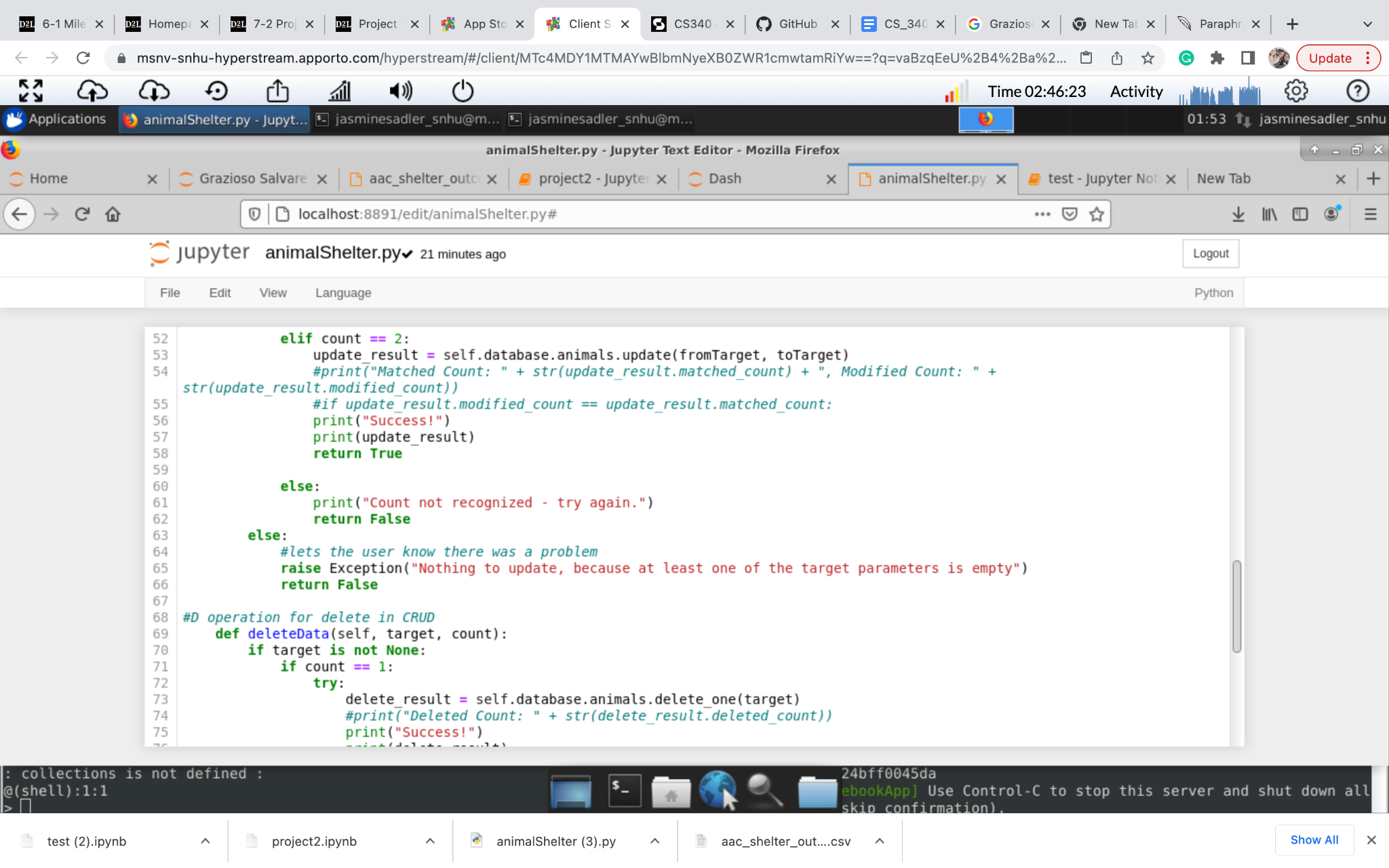
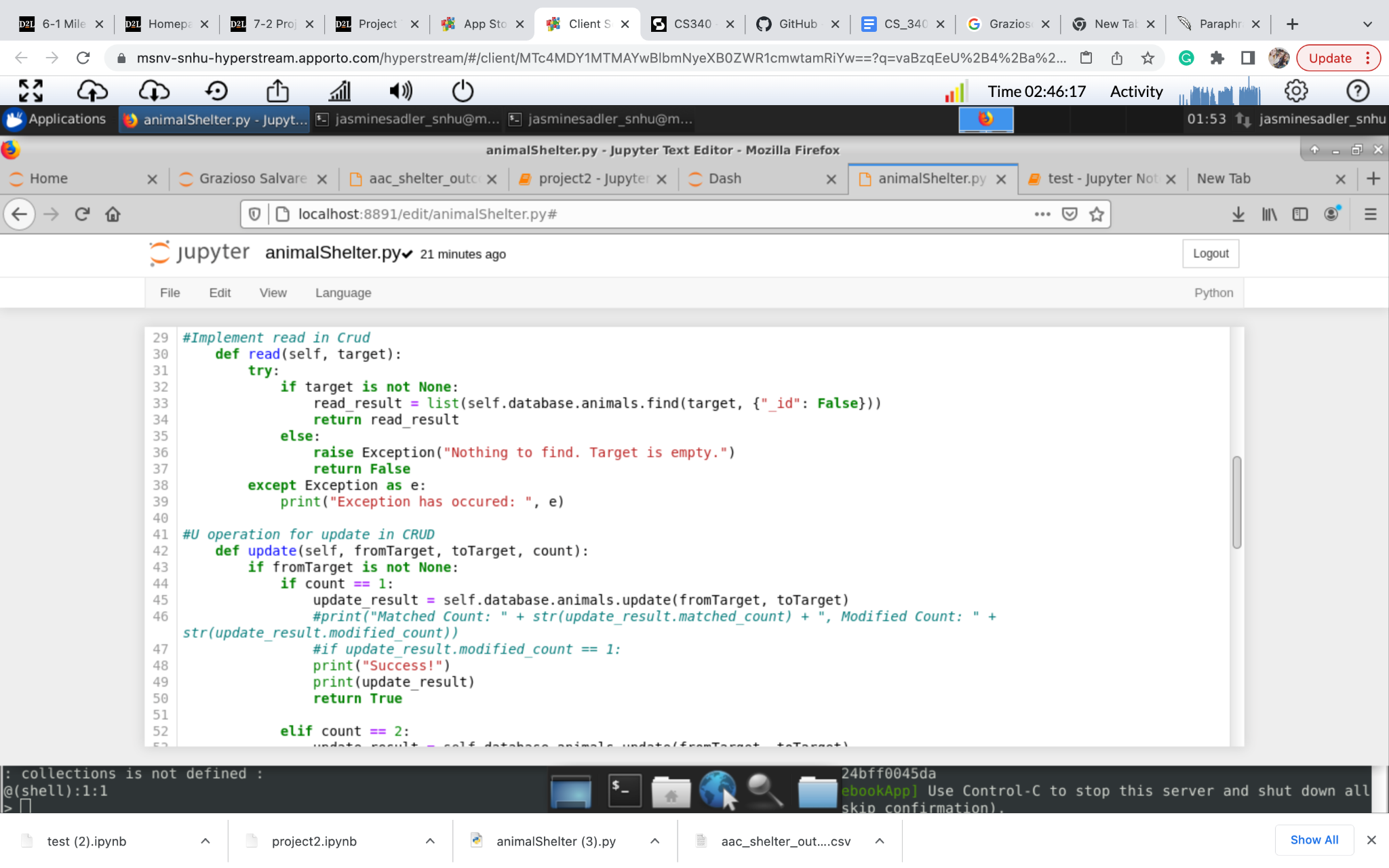
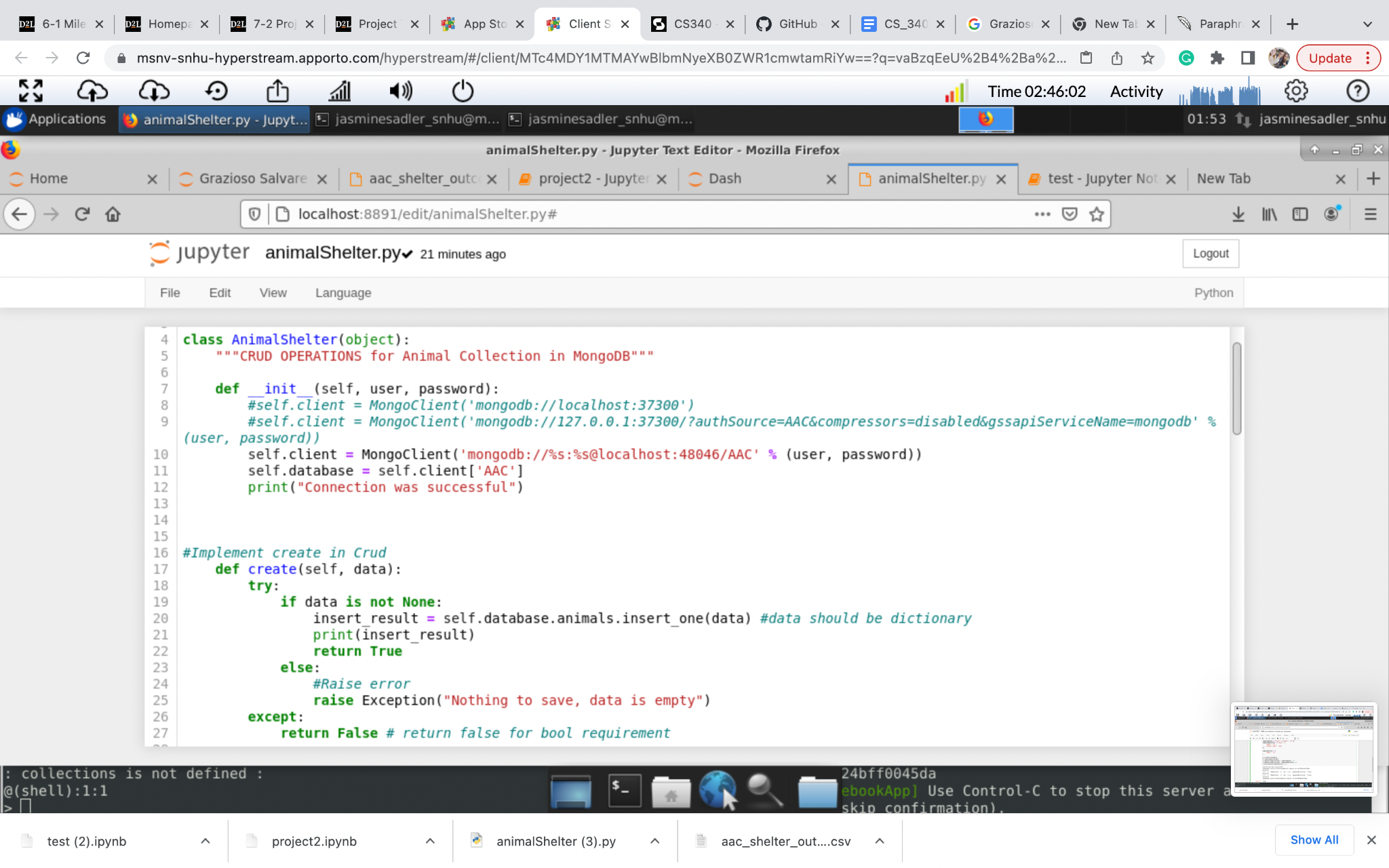
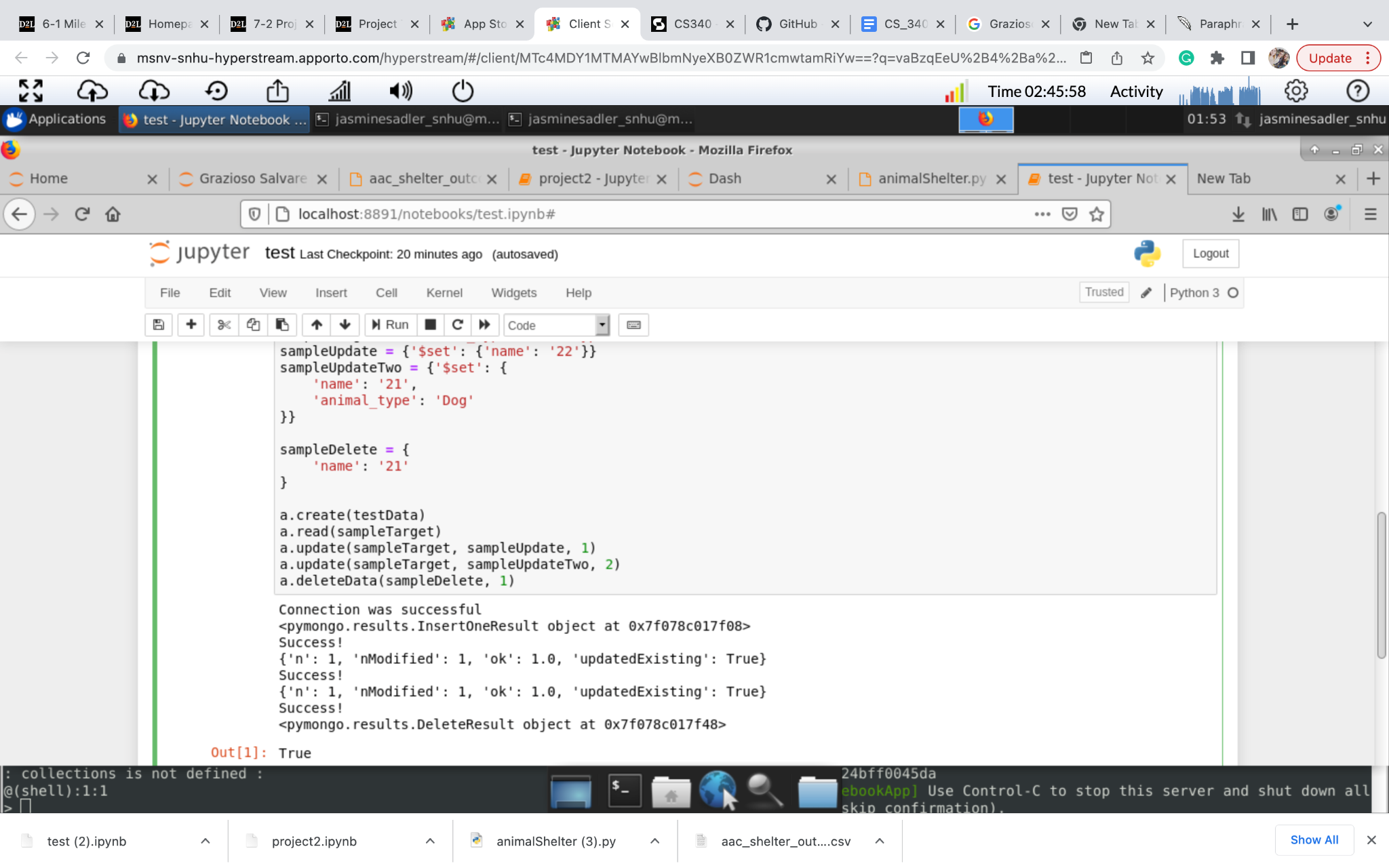
a.read(sampleTarget)

a.update(sampleTarget, sampleUpdate, 1)

a.update(sampleTarget, sampleUpdateTwo, 2)

a.deleteData(sampleDelete, 1)

**Screenshots:**

****

