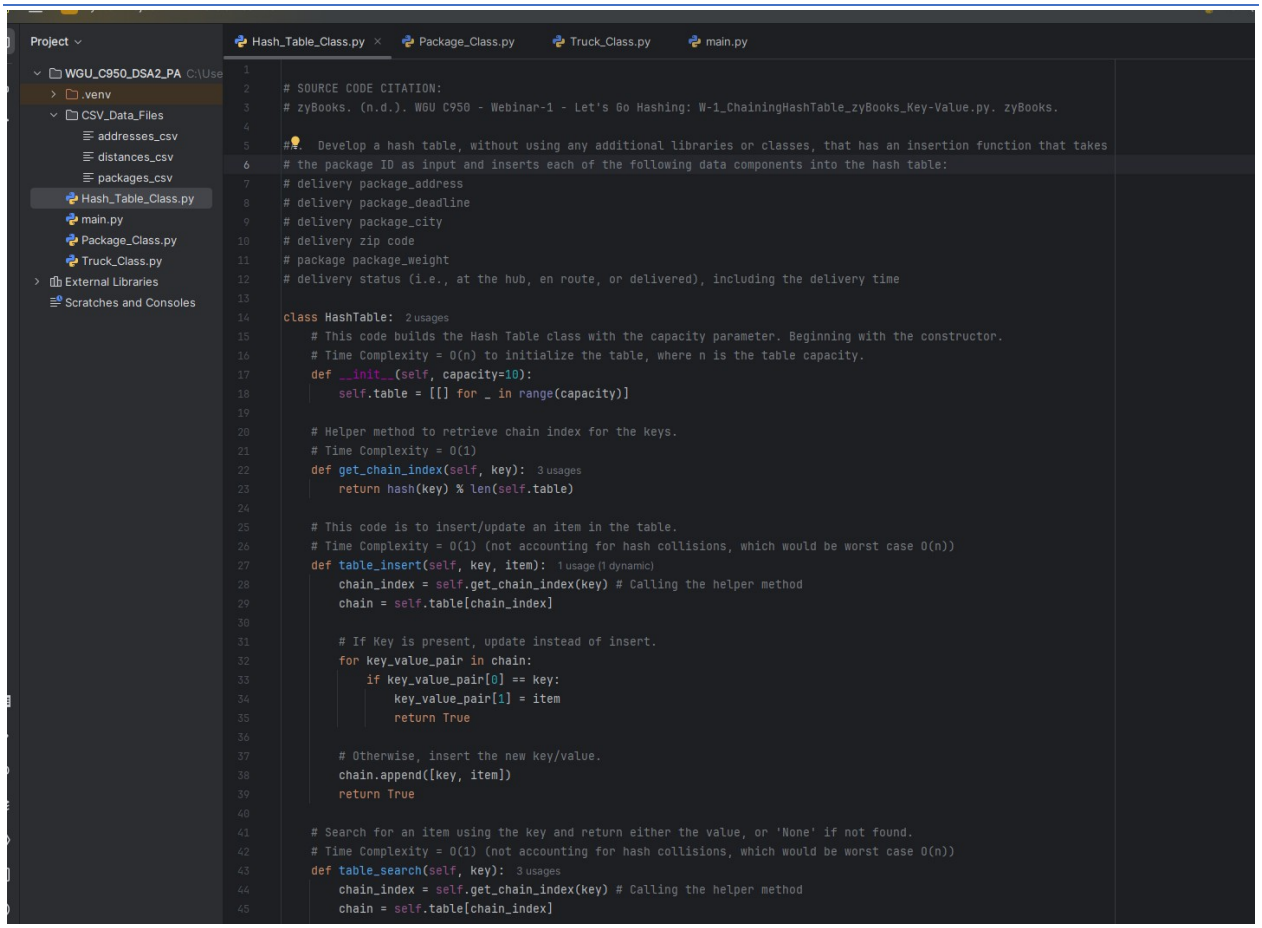
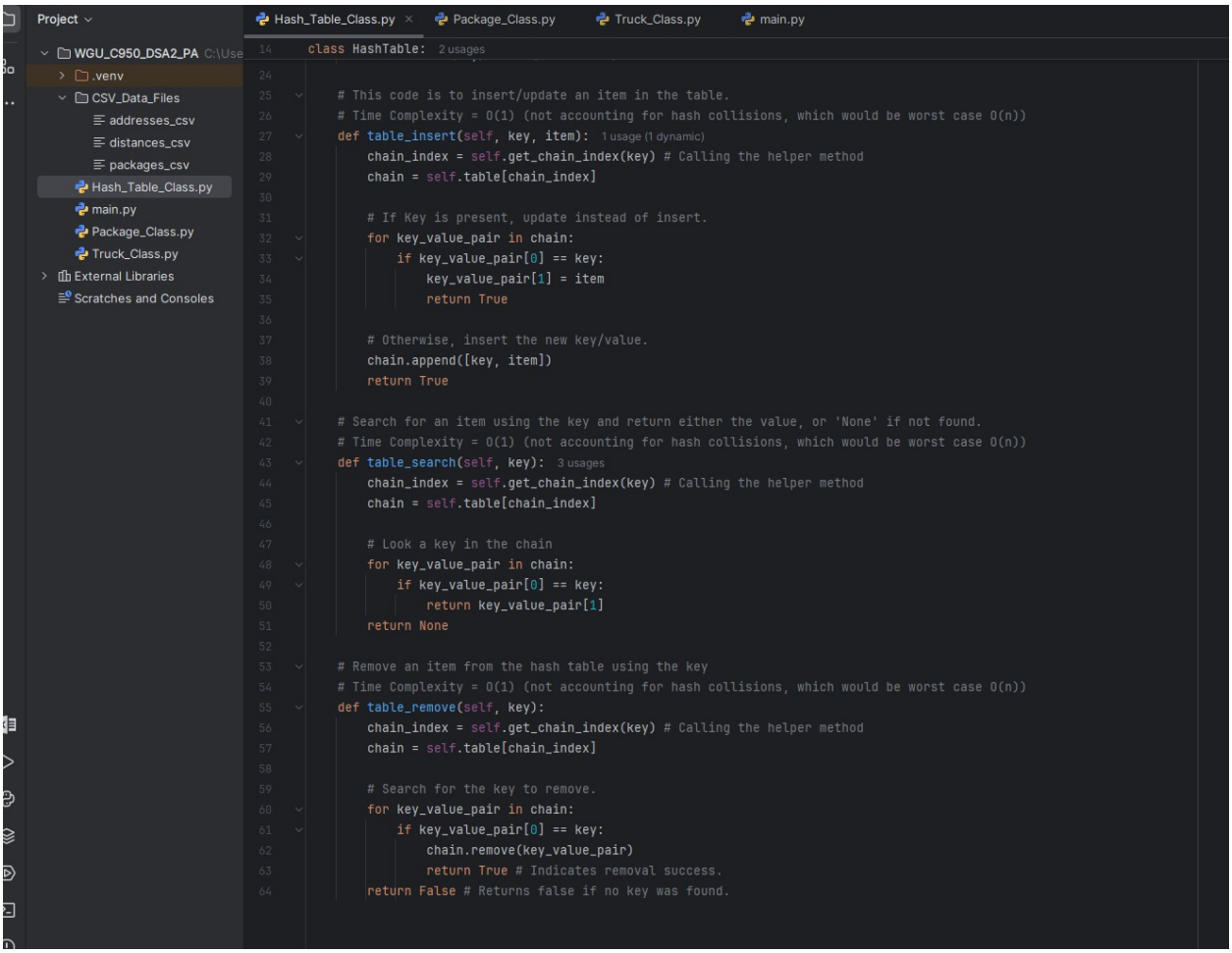
WGUPS Write-Up

John McGinnes

02/24/25

A. Hash Table

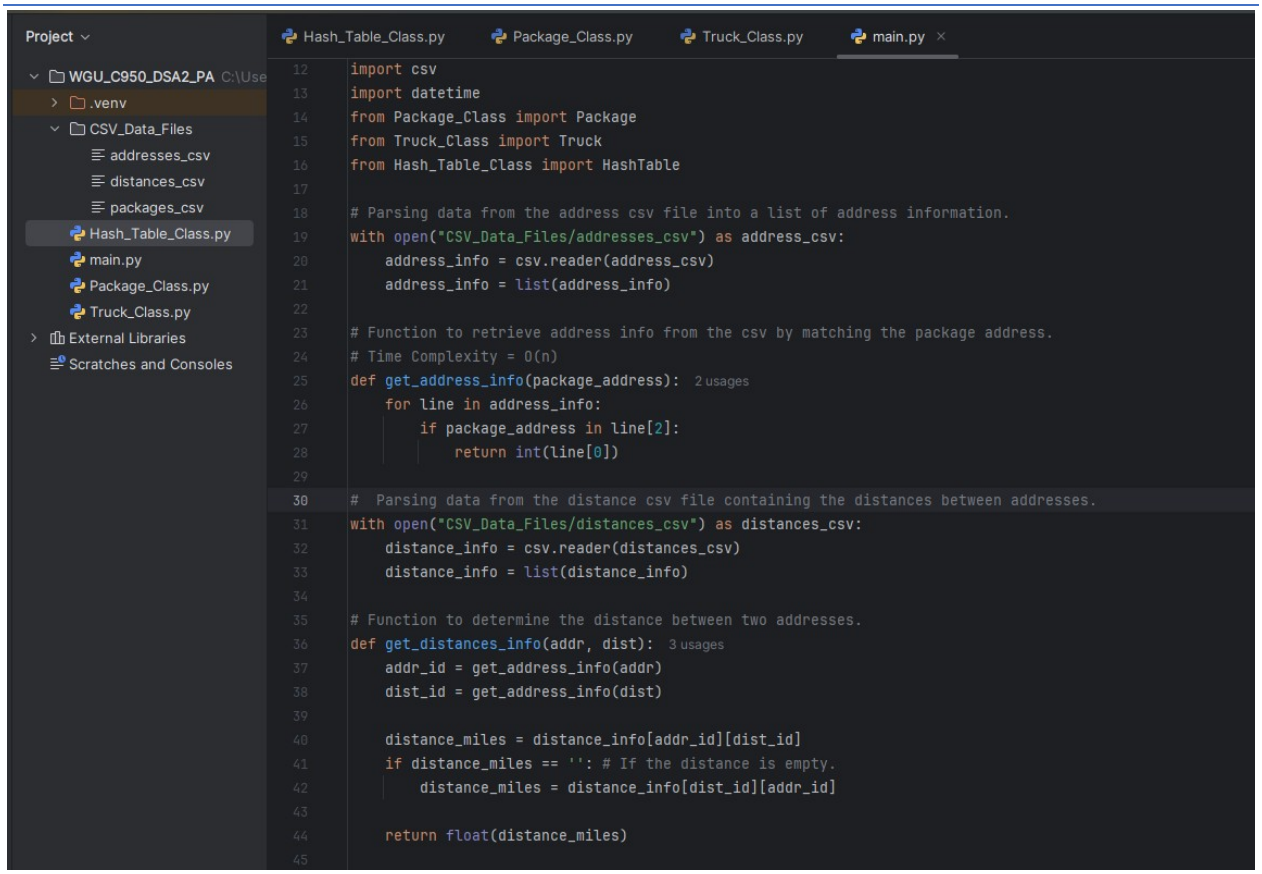

Project
WGU_C950_DSA2_PA C:\Use
.venv
CSV_Data_Files
addresses_csv
distances_csv
packages_csv
Hash_Table_Class.py
main.py
Package_Class.py
Truck_Class.py
External Libraries
Scratches and Consoles
Hash_Table_Class.py
Package_Class.py
Truck_Class.py
main.py
SOURCE CODE CITATION:
zyBooks. (n.d.). WGU C950 - Webinar-1 - Let's Go Hashing: W-1_ChainingHashTable_zyBooks_Key-Value.py. zyBooks.
Develop a hash table, without using any additional libraries or classes, that has an insertion function that takes
the package ID as input and inserts each of the following data components into the hash table:
delivery package_address
delivery package_deadline
delivery package_city
delivery zip code
package package_weight
delivery status (i.e., at the hub, en route, or delivered), including the delivery time
class HashTable: 2 usages
# This code builds the Hash Table class with the capacity parameter. Beginning with the constructor.
# Time Complexity = O(n) to initialize the table, where n is the table capacity.
def __init__(self, capacity=10):
self.table = [[] for _ in range(capacity)]
# Helper method to retrieve chain index for the keys.
# Time Complexity = O(1)
def get_chain_index(self, key): 3 usages
return hash(key) % len(self.table)
This code is to insert/update an item in the table.
Time Complexity = O(1) (not accounting for hash collisions, which would be worst case O(n))
def table_insert(self, key, item): 1 usage (1 dynamic)
chain_index = self.get_chain_index(key) # Calling the helper method
chain = self.table[chain_index]
# If Key is present, update instead of insert.
for key_value_pair in chain:
if key_value_pair[0] == key:
key_value_pair[1] = item
return True
# Otherwise, insert the new key/value.
chain.append([key, item])
return True
Search for an item using the key and return either the value, or 'None' if not found.
Time Complexity = O(1) (not accounting for hash collisions, which would be worst case O(n))
def table_search(self, key): 3 usages
chain_index = self.get_chain_index(key) # Calling the helper method
chain = self.table[chain_index]


Project
WGU_C950_DSA2_PA C:\Use
.venv
CSV_Data_Files
addresses_csv
distances_csv
packages_csv
Hash_Table_Class.py
main.py
Package_Class.py
Truck_Class.py
External Libraries
Scratches and Consoles
Hash_Table_Class.py
Package_Class.py
Truck_Class.py
main.py
class HashTable: 2 usages
This code is to insert/update an item in the table.
Time Complexity = O(1) (not accounting for hash collisions, which would be worst case O(n))
def table_insert(self, key, item): 1 usage (1 dynamic)
chain_index = self.get_chain_index(key) # Calling the helper method
chain = self.table[chain_index]
# If Key is present, update instead of insert.
for key_value_pair in chain:
if key_value_pair[0] == key:
key_value_pair[1] = item
return True
# Otherwise, insert the new key/value.
chain.append([key, item])
return True
Search for an item using the key and return either the value, or 'None' if not found.
Time Complexity = O(1) (not accounting for hash collisions, which would be worst case O(n))
def table_search(self, key): 3 usages
chain_index = self.get_chain_index(key) # Calling the helper method
chain = self.table[chain_index]
# Look a key in the chain
for key_value_pair in chain:
if key_value_pair[0] == key:
return key_value_pair[1]
return None
# Remove an item from the hash table using the key
# Time Complexity = O(1) (not accounting for hash collisions, which would be worst case O(n))
def table_remove(self, key):
chain_index = self.get_chain_index(key) # Calling the helper method
chain = self.table[chain_index]
# Search for the key to remove.
for key_value_pair in chain:
if key_value_pair[0] == key:
chain.remove(key_value_pair)
return True # Indicates removal success.
return False # Returns false if no key was found.

B. Look-Up Functions

```python

# Objective B:
# Develop a look-up function that takes the following components...

# Function to find a package by the ID number and print the current status.
def package_search(package_id, current_time=datetime.timedelta(hours=17)):  7 usages
    package = package_hash_table.table_search(package_id) # Time Complexity = O(1)
    if package is None: # Time Complexity = O(1)
        print(f"ERROR PACKAGE ID {package_id} INVALID.")
        return

        # For package #9, check the time and set the correct address
    if package.pkg_id == 9:
        # Time comparison: Package address is incorrect before 10:20
        time_threshold = datetime.timedelta(hours=10, minutes=20)
        if current_time < time_threshold:
            package.pkg_address = "300 State St"  # Incorrect address before 10:20
            package.pkg_zip = "84103"  # Incorrect ZIP code
        else:
            package.pkg_address = "410 S State St"  # Correct address after 10:20
            package.pkg_zip = "84111"  # Correct ZIP code

    package.set_pkg_status(current_time) # Time Complexity = O(1)
    print(str(package)) # Time Complexity = O(1)

```

C. Original Code

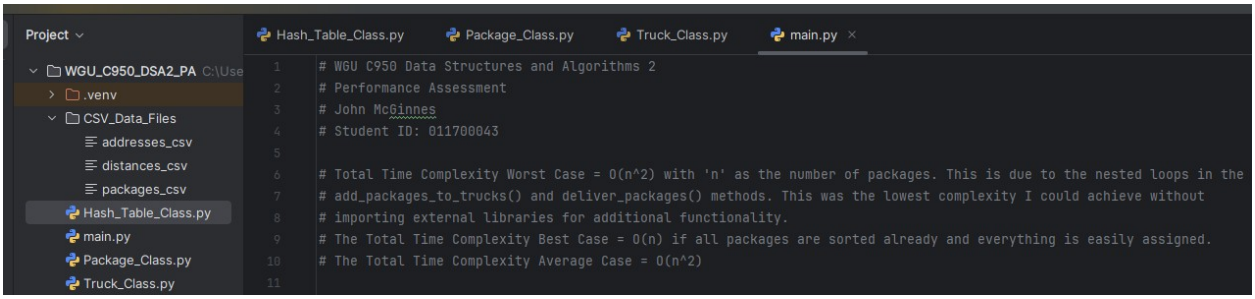


```python
# Objective A:
# Develop a hash table, without using any additional libraries or classes...

# Function to add packages to the hash table using the packages csv data.
def add_packages(pkg_hash_table):  1 usage
    with open("CSV_Data_Files/packages_csv") as package_csv:
        packages_info = csv.reader(package_csv)

        # Looping through each package and adding to the hash table.
        # Time Complexity = O(n) (n = number of packages)
        for package_row in packages_info:
            package_id = int(package_row[0])
            package_address = package_row[1]
            package_city = package_row[2]
            package_state = package_row[3]
            package_zip = package_row[4]
            package_deadline = package_row[5]
            package_weight = package_row[6]
            package_notes = package_row[7]
            package_status = "At The WGUPS Hub"
            package_departure_time = ""
            package_arrival_time = ""

            package = Package(package_id, package_address, package_city, package_state, package_zip, package_deadline,
                              package_weight, package_notes, package_status, package_departure_time, package_arrival_time)

            pkg_hash_table.table_insert(package_id, package) # Time Complexity = O(1)

# Initialize the hash table and add packages.
package_hash_table = HashTable()
add_packages(package_hash_table)
```

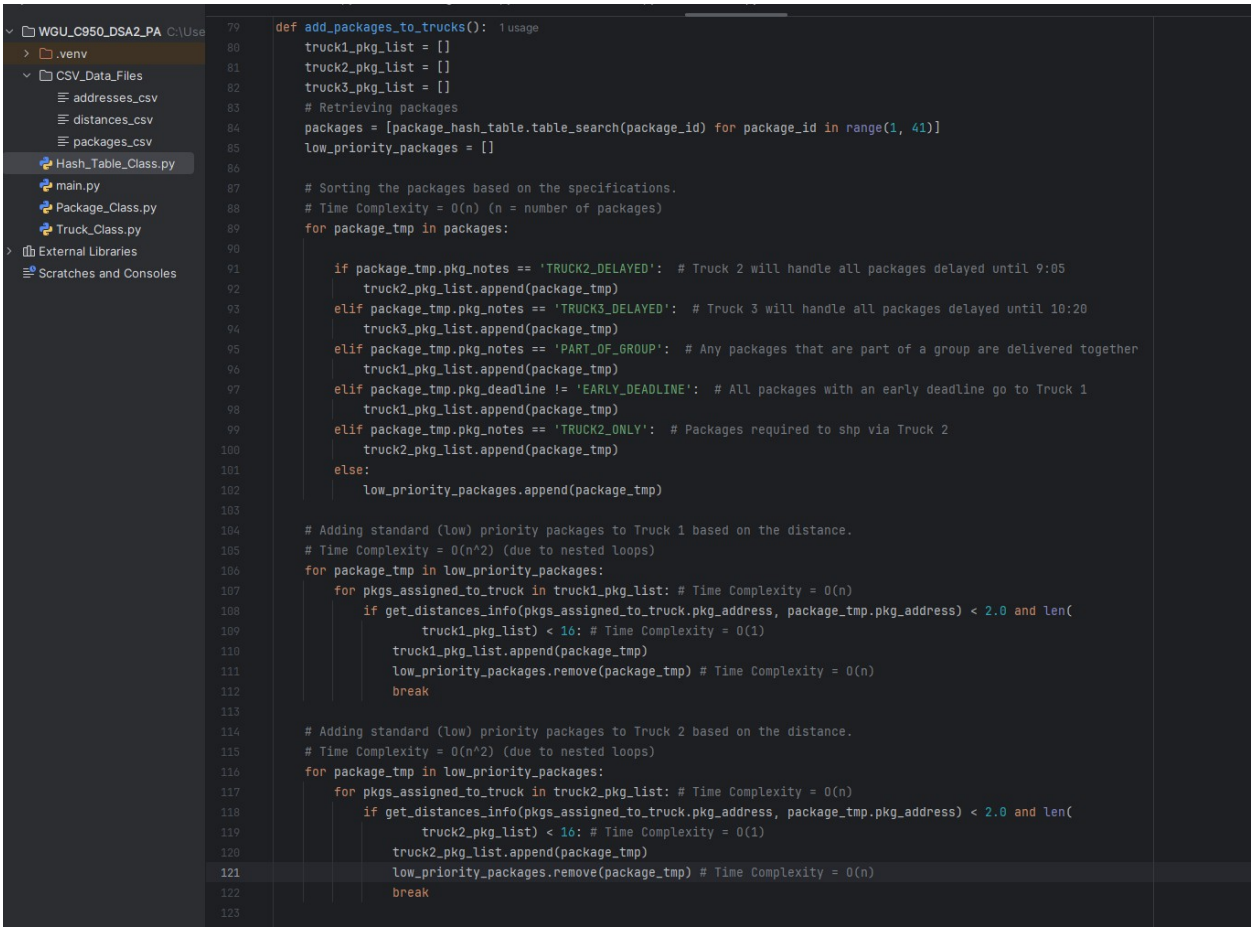
```python
# Function to add packages to trucks based on the package notes.
def add_packages_to_trucks():  1 usage
    truck1_pkg_list = []
    truck2_pkg_list = []
    truck3_pkg_list = []
    # Retrieving packages
    packages = [package_hash_table.table_search(package_id) for package_id in range(1, 41)]
    low_priority_packages = []

    # Sorting the packages based on the specifications.
    # Time Complexity = O(n) (n = number of packages)
    for package_tmp in packages:

        if package_tmp.pkg_notes == 'TRUCK2_DELAYED':  # Truck 2 will handle all packages delayed until 9:05
            truck2_pkg_list.append(package_tmp)
        elif package_tmp.pkg_notes == 'TRUCK3_DELAYED':  # Truck 3 will handle all packages delayed until 10:20
            truck3_pkg_list.append(package_tmp)
        elif package_tmp.pkg_notes == 'PART_OF_GROUP':  # Any packages that are part of a group are delivered together
            truck1_pkg_list.append(package_tmp)
        elif package_tmp.pkg_deadline != 'EARLY_DEADLINE':  # All packages with an early deadline go to Truck 1
            truck1_pkg_list.append(package_tmp)
        elif package_tmp.pkg_notes == 'TRUCK2_ONLY':  # Packages required to shp via Truck 2
            truck2_pkg_list.append(package_tmp)
        else:
            low_priority_packages.append(package_tmp)

    # Adding standard (low) priority packages to Truck 1 based on the distance.
    # Time Complexity = O(n^2) (due to nested loops)
    for package_tmp in low_priority_packages:
        for pkgs_assigned_to_truck in truck1_pkg_list: # Time Complexity = O(n)
            if get_distances_info(pkgs_assigned_to_truck.pkg_address, package_tmp.pkg_address) < 2.0 and len(
                    truck1_pkg_list) < 16: # Time Complexity = O(1)
                truck1_pkg_list.append(package_tmp)
                low_priority_packages.remove(package_tmp) # Time Complexity = O(n)
                break

    # Adding standard (low) priority packages to Truck 2 based on the distance.
    # Time Complexity = O(n^2) (due to nested loops)
    for package_tmp in low_priority_packages:
        for pkgs_assigned_to_truck in truck2_pkg_list: # Time Complexity = O(n)
            if get_distances_info(pkgs_assigned_to_truck.pkg_address, package_tmp.pkg_address) < 2.0 and len(
                    truck2_pkg_list) < 16: # Time Complexity = O(1)
                truck2_pkg_list.append(package_tmp)
                low_priority_packages.remove(package_tmp) # Time Complexity = O(n)
                break

```

```python
# Time Complexity = O(N^2) due to the nested loop required to search for the nearest package.
def deliver_packages(truck):  3 usages
    tmp_pkg_queue = []

    # Initializing the package queue.
    # Time Complexity = O(n) (n = number of packages)
    for package in truck.packages:
        tmp_pkg_queue.append(package)

    truck.packages.clear() # Time Complexity = O(1)

    # Nearest-Neighbor algorithm.
    # Time Complexity = O(n^2) (n = number of packages)
    while len(tmp_pkg_queue) > 0: # Time Complexity = O(n)
        nearest_address = 999
        nearest_package = None

        for package in tmp_pkg_queue: # Time Complexity = O(n) (n = number of packages)
            package_distance = get_distances_info(truck.current_address, package.pkg_address) # Time Complexity = O(n) (n = number of packages)

            if package_distance <= nearest_address: # Time Complexity = O(1)
                nearest_address = package_distance
                nearest_package = package

        truck.packages.append(nearest_package.pkg_id) # Time Complexity = O(1)
        tmp_pkg_queue.remove(nearest_package) # Time Complexity = O(n) (n = number of packages)

        nearest_package.depart = truck.last_departure

        truck.current_address = nearest_package.pkg_address
        truck.total_distance += nearest_address # Time Complexity = O(1)

        truck.last_departure += datetime.timedelta(hours=nearest_address / 18) # Time Complexity = O(1)
        nearest_package.arrive = truck.last_departure
```

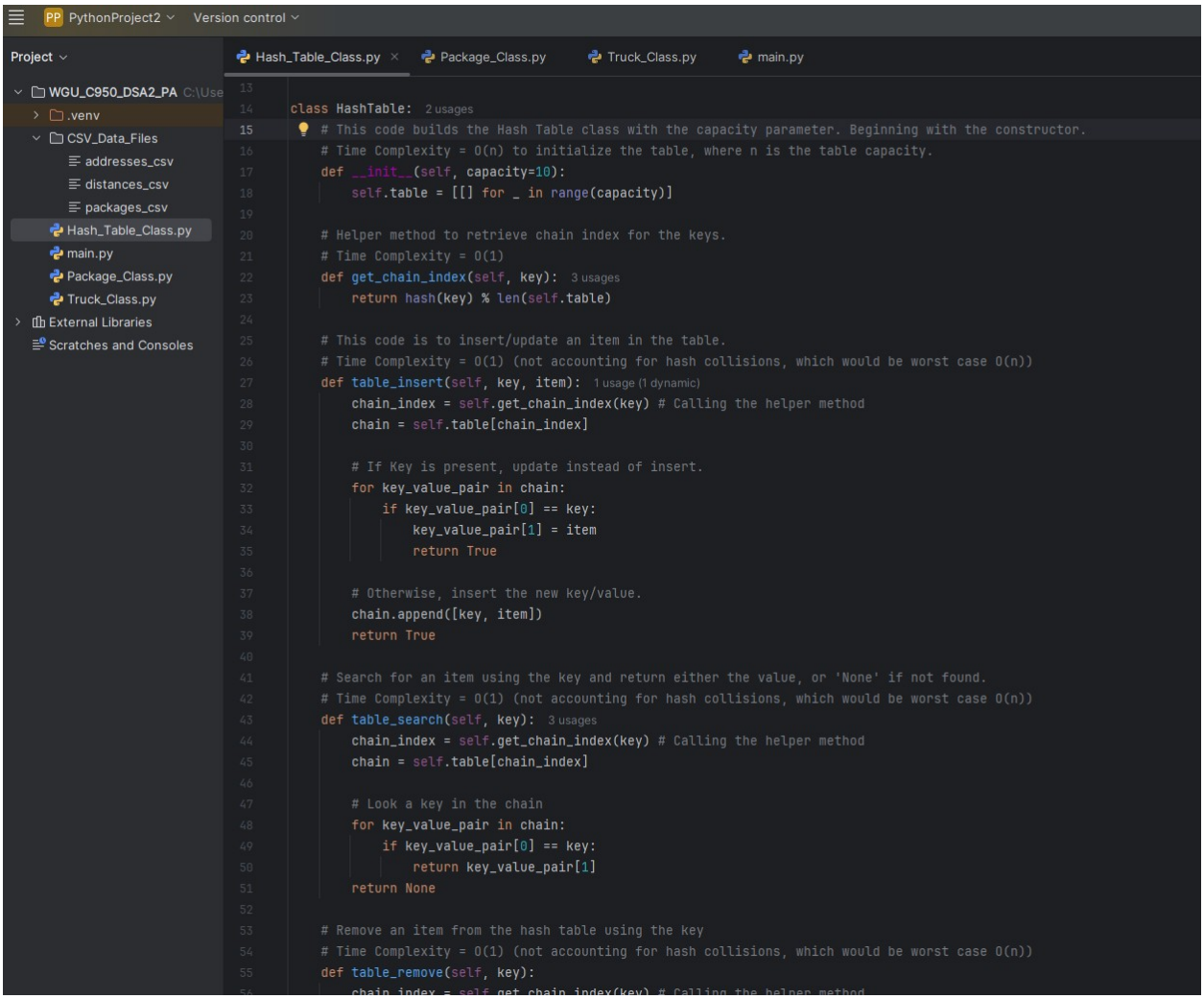
C1. Identification Information



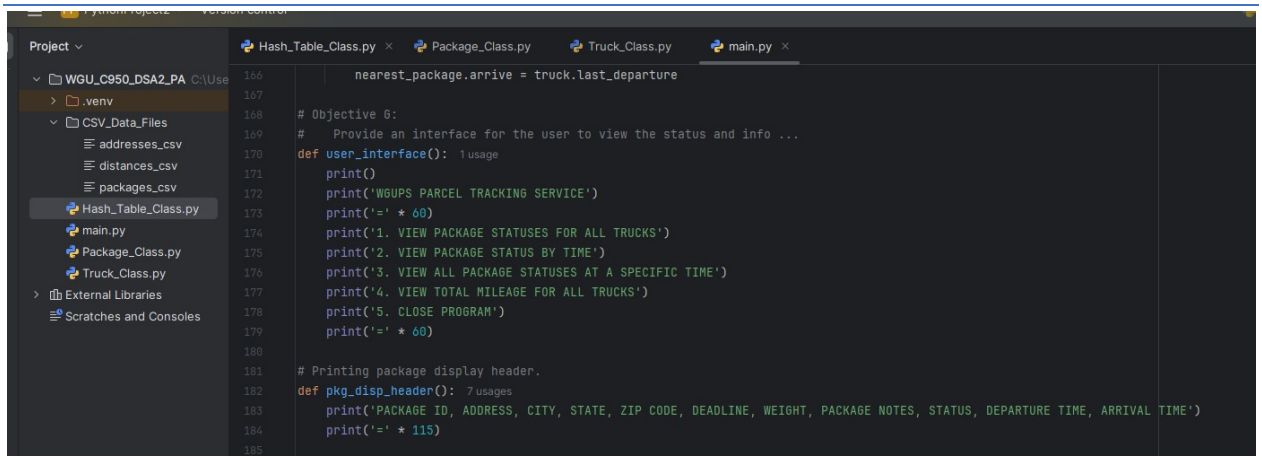
C2. Process and Flow Comments

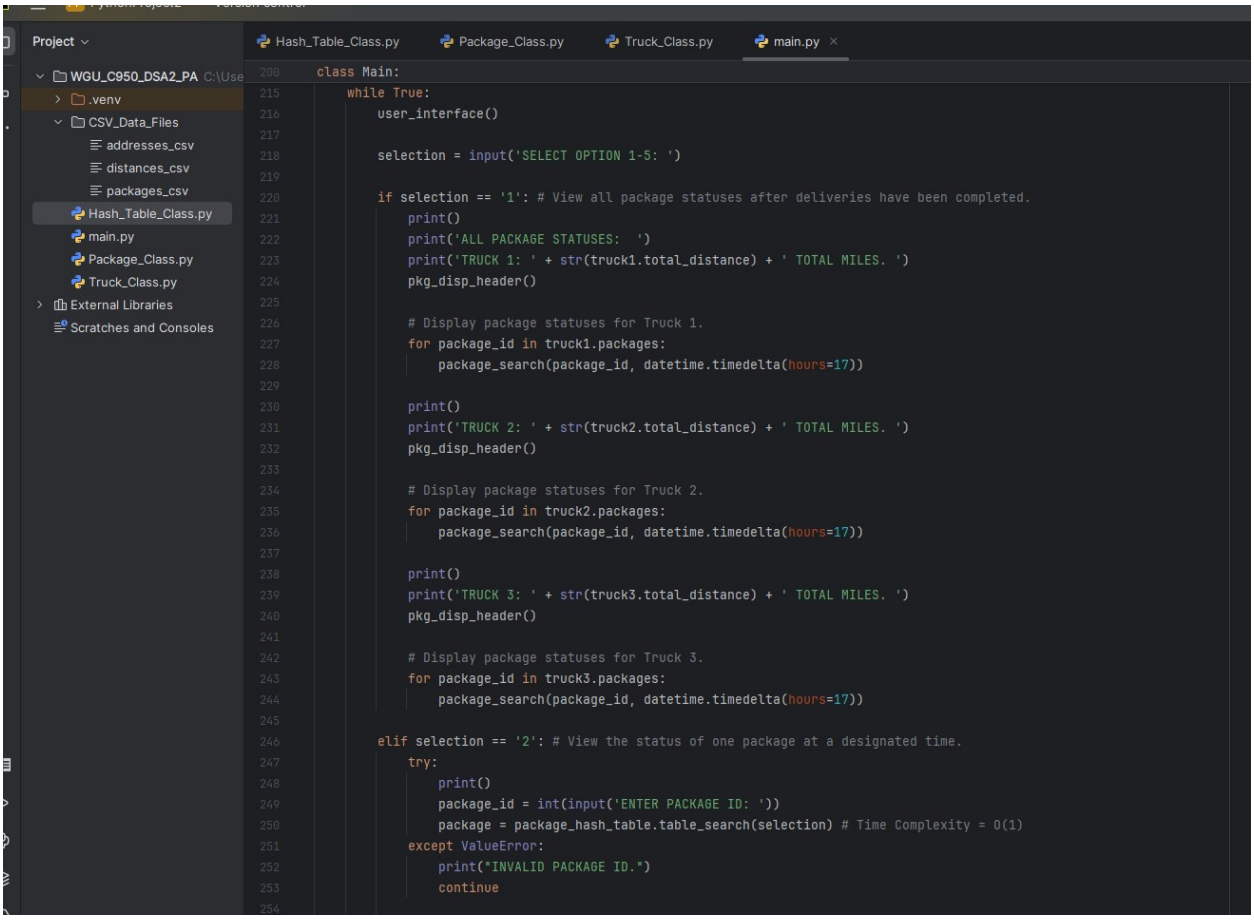


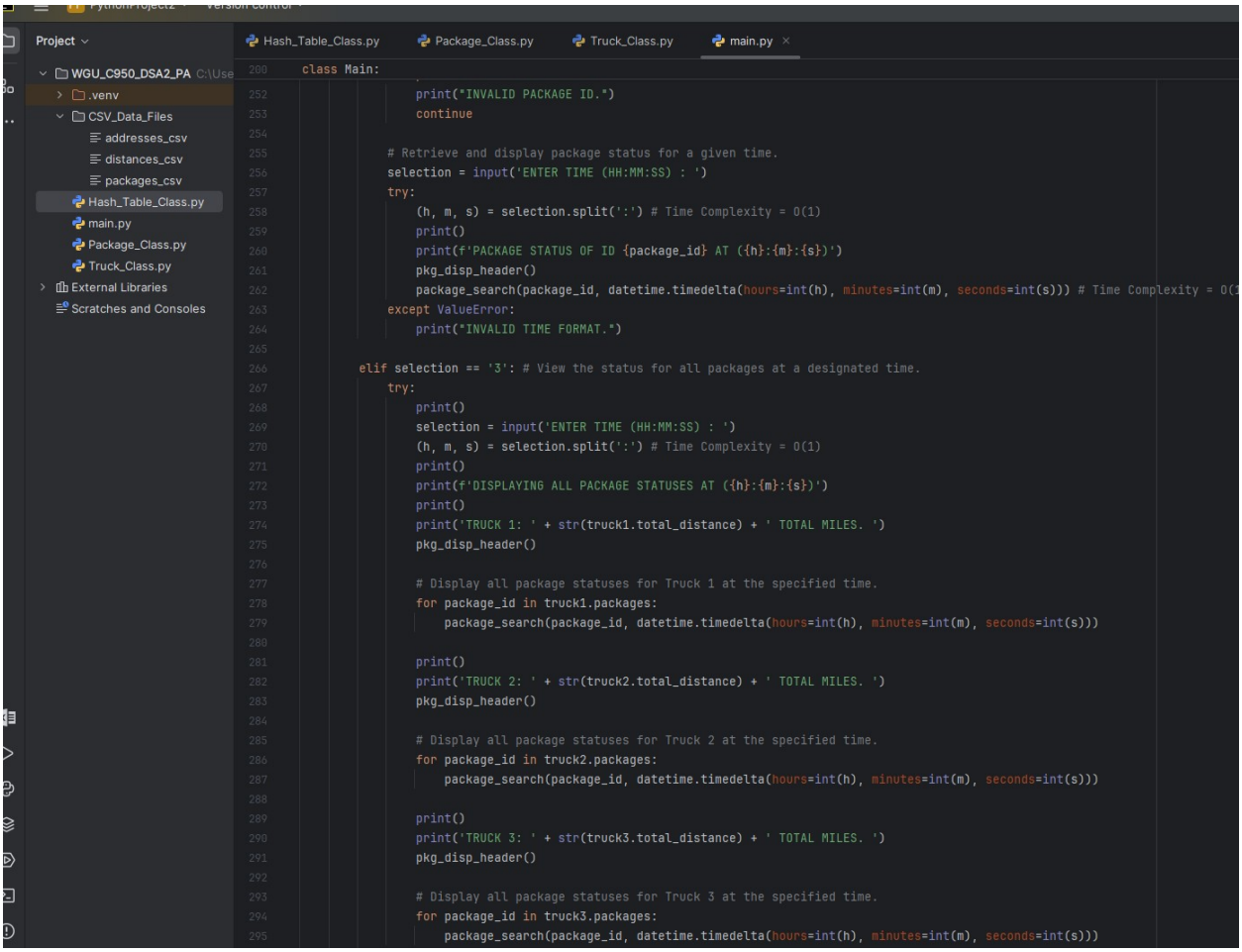

```python
def add_packages_to_trucks():  1 usage
    truck1_pkg_list = []
    truck2_pkg_list = []
    truck3_pkg_list = []
    # Retrieving packages
    packages = [package_hash_table.table_search(package_id) for package_id in range(1, 41)]
    low_priority_packages = []

    # Sorting the packages based on the specifications.
    # Time Complexity = O(n) (n = number of packages)
    for package_tmp in packages:

        if package_tmp.pkg_notes == 'TRUCK2_DELAYED':  # Truck 2 will handle all packages delayed until 9:05
            truck2_pkg_list.append(package_tmp)
        elif package_tmp.pkg_notes == 'TRUCK3_DELAYED':  # Truck 3 will handle all packages delayed until 10:20
            truck3_pkg_list.append(package_tmp)
        elif package_tmp.pkg_notes == 'PART_OF_GROUP':  # Any packages that are part of a group are delivered together
            truck1_pkg_list.append(package_tmp)
        elif package_tmp.pkg_deadline != 'EARLY_DEADLINE':  # All packages with an early deadline go to Truck 1
            truck1_pkg_list.append(package_tmp)
        elif package_tmp.pkg_notes == 'TRUCK2_ONLY':  # Packages required to shp via Truck 2
            truck2_pkg_list.append(package_tmp)
        else:
            low_priority_packages.append(package_tmp)

    # Adding standard (low) priority packages to Truck 1 based on the distance.
    # Time Complexity = O(n^2) (due to nested loops)
    for package_tmp in low_priority_packages:
        for pkgs_assigned_to_truck in truck1_pkg_list: # Time Complexity = O(n)
            if get_distances_info(pkgs_assigned_to_truck.pkg_address, package_tmp.pkg_address) < 2.0 and len(
                    truck1_pkg_list) < 16: # Time Complexity = O(1)
                truck1_pkg_list.append(package_tmp)
                low_priority_packages.remove(package_tmp) # Time Complexity = O(n)
                break

    # Adding standard (low) priority packages to Truck 2 based on the distance.
    # Time Complexity = O(n^2) (due to nested loops)
    for package_tmp in low_priority_packages:
        for pkgs_assigned_to_truck in truck2_pkg_list: # Time Complexity = O(n)
            if get_distances_info(pkgs_assigned_to_truck.pkg_address, package_tmp.pkg_address) < 2.0 and len(
                    truck2_pkg_list) < 16: # Time Complexity = O(1)
                truck2_pkg_list.append(package_tmp)
                low_priority_packages.remove(package_tmp) # Time Complexity = O(n)
                break
```

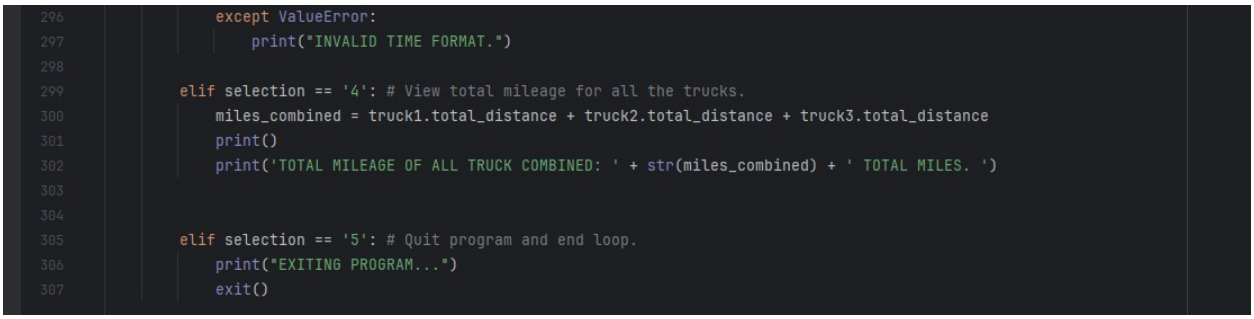

PythonProject2
Version control
Project
Hash_Table_Class.py
Package_Class.py
Truck_Class.py
main.py
WGU_C950_DSA2_PA C:\Use
.venv
CSV_Data_Files
addresses_csv
distances_csv
packages_csv
Hash_Table_Class.py
main.py
Package_Class.py
Truck_Class.py
External Libraries
Scratches and Consoles
class HashTable: 2 usages
# This code builds the Hash Table class with the capacity parameter. Beginning with the constructor.
# Time Complexity = O(n) to initialize the table, where n is the table capacity.
def __init__(self, capacity=10):
self.table = [[] for _ in range(capacity)]
# Helper method to retrieve chain index for the keys.
# Time Complexity = O(1)
def get_chain_index(self, key): 3 usages
return hash(key) % len(self.table)
This code is to insert/update an item in the table.
Time Complexity = O(1) (not accounting for hash collisions, which would be worst case O(n))
def table_insert(self, key, item): 1 usage (1 dynamic)
chain_index = self.get_chain_index(key) # Calling the helper method
chain = self.table[chain_index]
# If Key is present, update instead of insert.
for key_value_pair in chain:
if key_value_pair[0] == key:
key_value_pair[1] = item
return True
# Otherwise, insert the new key/value.
chain.append([key, item])
return True
Search for an item using the key and return either the value, or 'None' if not found.
Time Complexity = O(1) (not accounting for hash collisions, which would be worst case O(n))
def table_search(self, key): 3 usages
chain_index = self.get_chain_index(key) # Calling the helper method
chain = self.table[chain_index]
# Look a key in the chain
for key_value_pair in chain:
if key_value_pair[0] == key:
return key_value_pair[1]
return None
# Remove an item from the hash table using the key
# Time Complexity = O(1) (not accounting for hash collisions, which would be worst case O(n))
def table_remove(self, key):

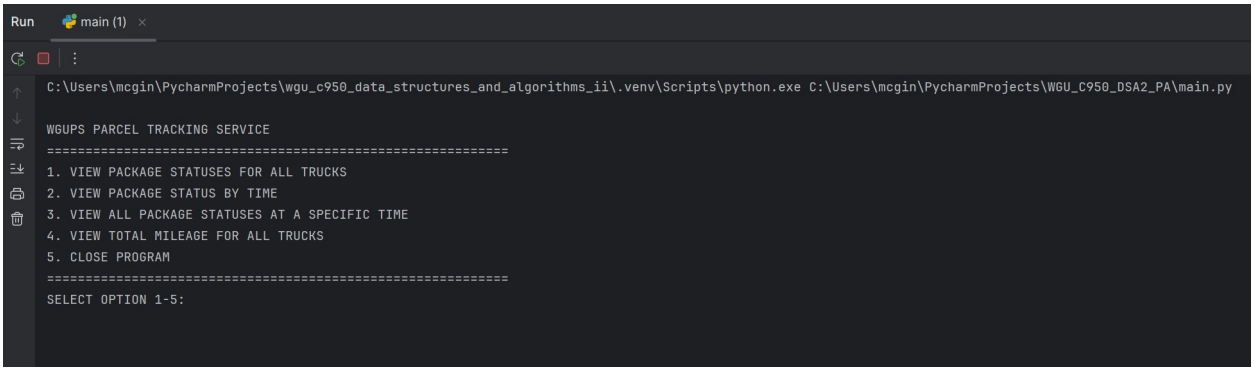
D. Interface


Project
Hash_Table_Class.py
Package_Class.py
Truck_Class.py
main.py
WGU_C950_DSA2_PA C:\Use
.venv
CSV_Data_Files
addresses_csv
distances_csv
packages_csv
Hash_Table_Class.py
main.py
Package_Class.py
Truck_Class.py
External Libraries
Scratches and Consoles
nearest_package.arrive = truck.last_departure
# Objective 6:
# Provide an interface for the user to view the status and info ...
def user_interface(): 1 usage
print()
print('WGUPS PARCEL TRACKING SERVICE')
print('=' * 60)
print('1. VIEW PACKAGE STATUSES FOR ALL TRUCKS')
print('2. VIEW PACKAGE STATUS BY TIME')
print('3. VIEW ALL PACKAGE STATUSES AT A SPECIFIC TIME')
print('4. VIEW TOTAL MILEAGE FOR ALL TRUCKS')
print('5. CLOSE PROGRAM')
print('=' * 60)
# Printing package display header.
def pkg_disp_header(): 7 usages
print('PACKAGE ID, ADDRESS, CITY, STATE, ZIP CODE, DEADLINE, WEIGHT, PACKAGE NOTES, STATUS, DEPARTURE TIME, ARRIVAL TIME')
print('=' * 115)

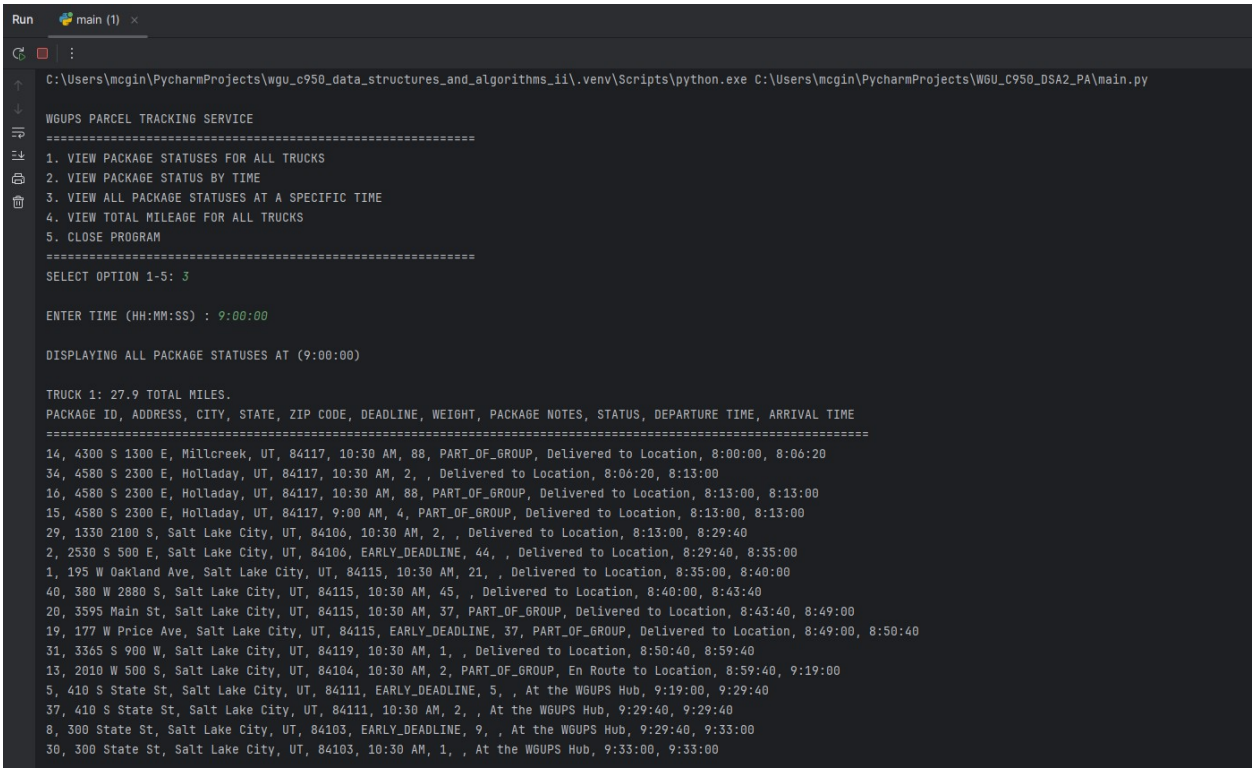

Project
WGU_C950_DSA2_PA C:\Use
.venv
CSV_Data_Files
addresses_csv
distances_csv
packages_csv
Hash_Table_Class.py
main.py
Package_Class.py
Truck_Class.py
External Libraries
Scratches and Consoles
Hash_Table_Class.py
Package_Class.py
Truck_Class.py
main.py
class Main:
while True:
user_interface()
selection = input('SELECT OPTION 1-5: ')
if selection == '1': # View all package statuses after deliveries have been completed.
print()
print('ALL PACKAGE STATUSES: ')
print('TRUCK 1: ' + str(truck1.total_distance) + ' TOTAL MILES. ')
pkg_disp_header()
# Display package statuses for Truck 1.
for package_id in truck1.packages:
package_search(package_id, datetime.timedelta(hours=17))
print()
print('TRUCK 2: ' + str(truck2.total_distance) + ' TOTAL MILES. ')
pkg_disp_header()
# Display package statuses for Truck 2.
for package_id in truck2.packages:
package_search(package_id, datetime.timedelta(hours=17))
print()
print('TRUCK 3: ' + str(truck3.total_distance) + ' TOTAL MILES. ')
pkg_disp_header()
# Display package statuses for Truck 3.
for package_id in truck3.packages:
package_search(package_id, datetime.timedelta(hours=17))
elif selection == '2': # View the status of one package at a designated time.
try:
print()
package_id = int(input('ENTER PACKAGE ID: '))
package = package_hash_table.table_search(selection) # Time Complexity = O(1)
except ValueError:
print("INVALID PACKAGE ID.")
continue

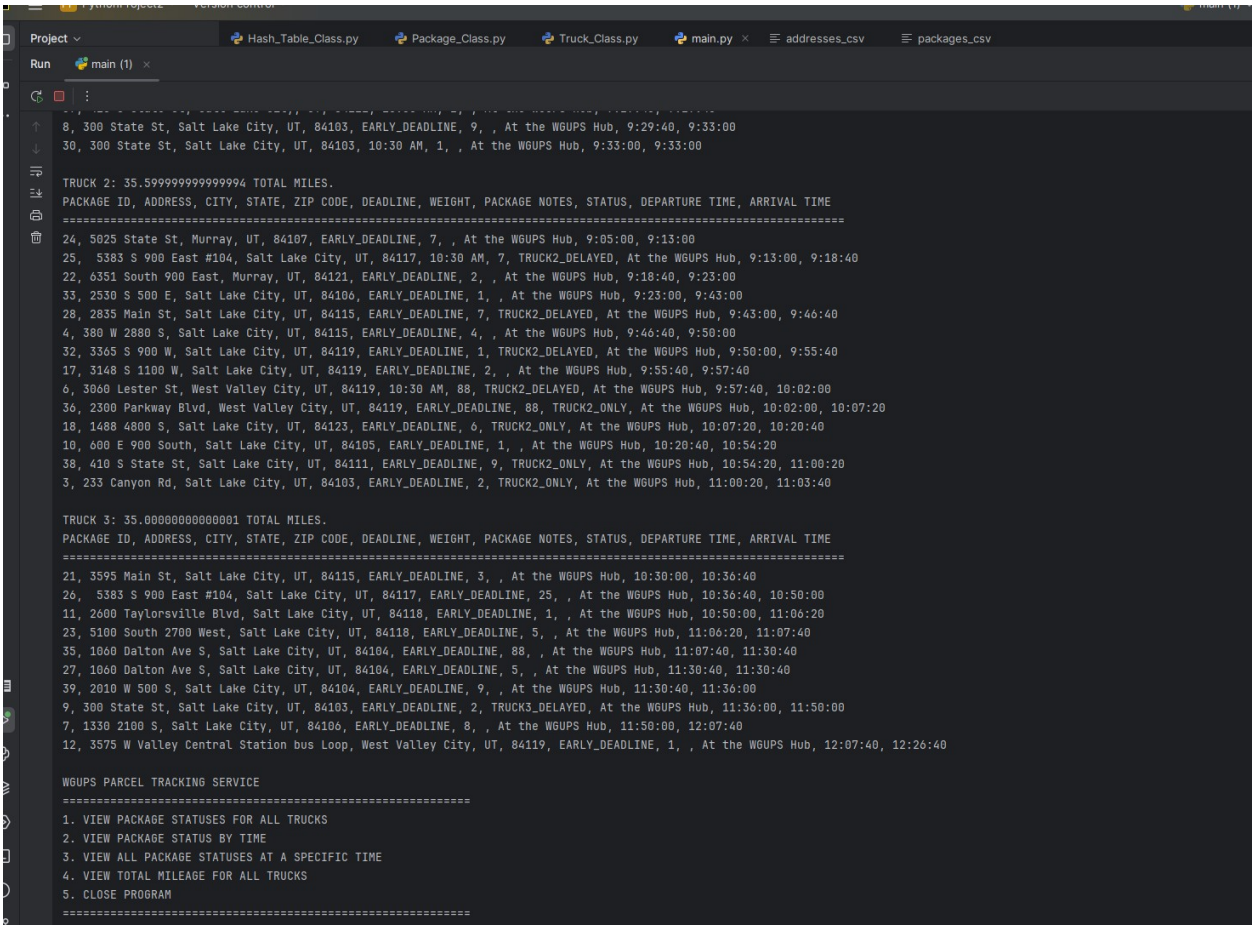

Project
WGU_C950_DSA2_PA C:\Use
.venv
CSV_Data_Files
addresses_csv
distances_csv
packages_csv
Hash_Table_Class.py
main.py
Package_Class.py
Truck_Class.py
External Libraries
Scratches and Consoles
Hash_Table_Class.py
Package_Class.py
Truck_Class.py
main.py
class Main:
print("INVALID PACKAGE ID.")
continue
# Retrieve and display package status for a given time.
selection = input('ENTER TIME (HH:MM:SS) : ')
try:
(h, m, s) = selection.split(':') # Time Complexity = O(1)
print()
print(f'PACKAGE STATUS OF ID {package_id} AT ({h}:{m}:{s})')
pkg_disp_header()
package_search(package_id, datetime.timedelta(hours=int(h), minutes=int(m), seconds=int(s))) # Time Complexity = O(1
except ValueError:
print("INVALID TIME FORMAT.")
elif selection == '3': # View the status for all packages at a designated time.
try:
print()
selection = input('ENTER TIME (HH:MM:SS) : ')
(h, m, s) = selection.split(':') # Time Complexity = O(1)
print()
print(f'DISPLAYING ALL PACKAGE STATUSES AT ({h}:{m}:{s})')
print()
print('TRUCK 1: ' + str(truck1.total_distance) + ' TOTAL MILES. ')
pkg_disp_header()
# Display all package statuses for Truck 1 at the specified time.
for package_id in truck1.packages:
package_search(package_id, datetime.timedelta(hours=int(h), minutes=int(m), seconds=int(s)))
print()
print('TRUCK 2: ' + str(truck2.total_distance) + ' TOTAL MILES. ')
pkg_disp_header()
# Display all package statuses for Truck 2 at the specified time.
for package_id in truck2.packages:
package_search(package_id, datetime.timedelta(hours=int(h), minutes=int(m), seconds=int(s)))
print()
print('TRUCK 3: ' + str(truck3.total_distance) + ' TOTAL MILES. ')
pkg_disp_header()
# Display all package statuses for Truck 3 at the specified time.
for package_id in truck3.packages:
package_search(package_id, datetime.timedelta(hours=int(h), minutes=int(m), seconds=int(s)))


except ValueError:
print("INVALID TIME FORMAT.")
elif selection == '4': # View total mileage for all the trucks.
miles_combined = truck1.total_distance + truck2.total_distance + truck3.total_distance
print()
print('TOTAL MILEAGE OF ALL TRUCK COMBINED: ' + str(miles_combined) + ' TOTAL MILES. ')
elif selection == '5': # Quit program and end loop.
print("EXITING PROGRAM...")
exit()

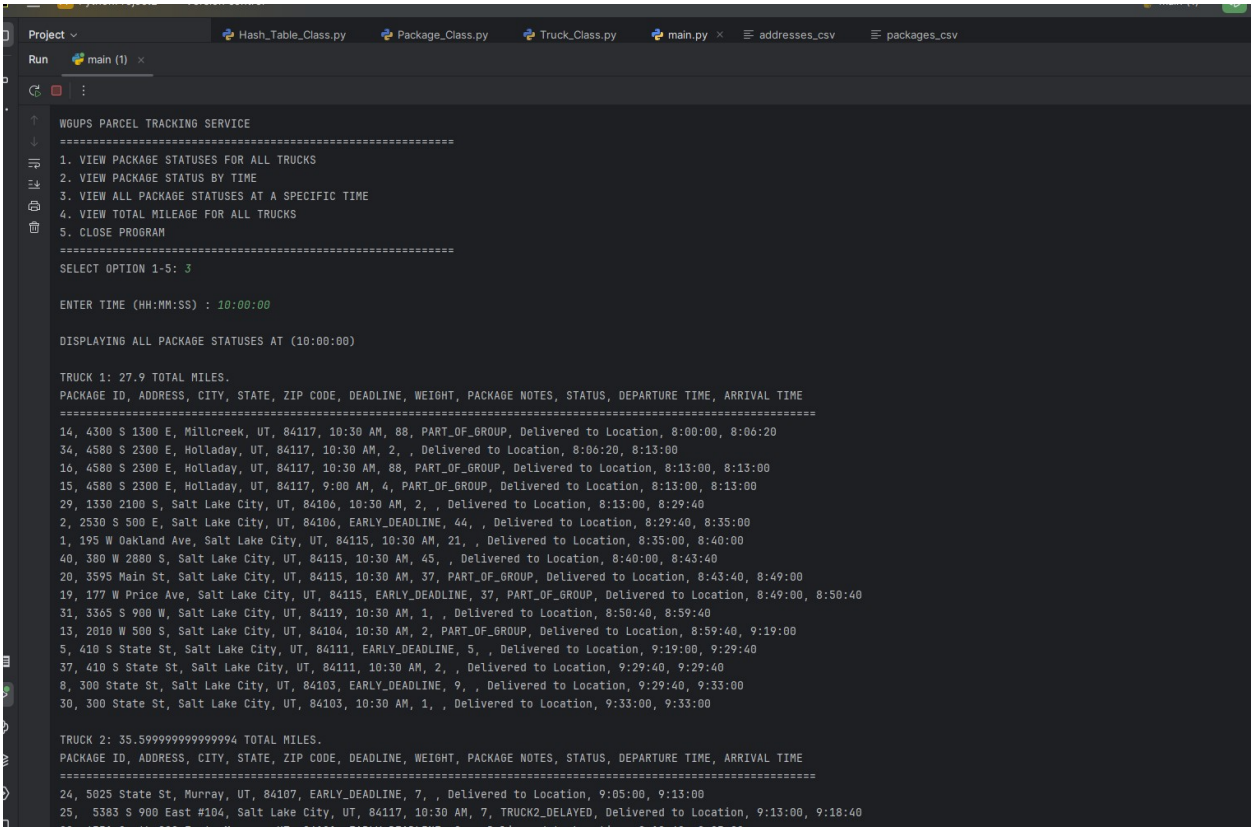

Run
main (1)
C:\Users\mcgin\PycharmProjects\wgu_c950_data_structures_and_algorithms_ii\.venv\Scripts\python.exe C:\Users\mcgin\PycharmProjects\WGU_C950_DSA2_PA\main.py
WGUPS PARCEL TRACKING SERVICE
1. VIEW PACKAGE STATUSES FOR ALL TRUCKS
2. VIEW PACKAGE STATUS BY TIME
3. VIEW ALL PACKAGE STATUSES AT A SPECIFIC TIME
4. VIEW TOTAL MILEAGE FOR ALL TRUCKS
5. CLOSE PROGRAM
SELECT OPTION 1-5:

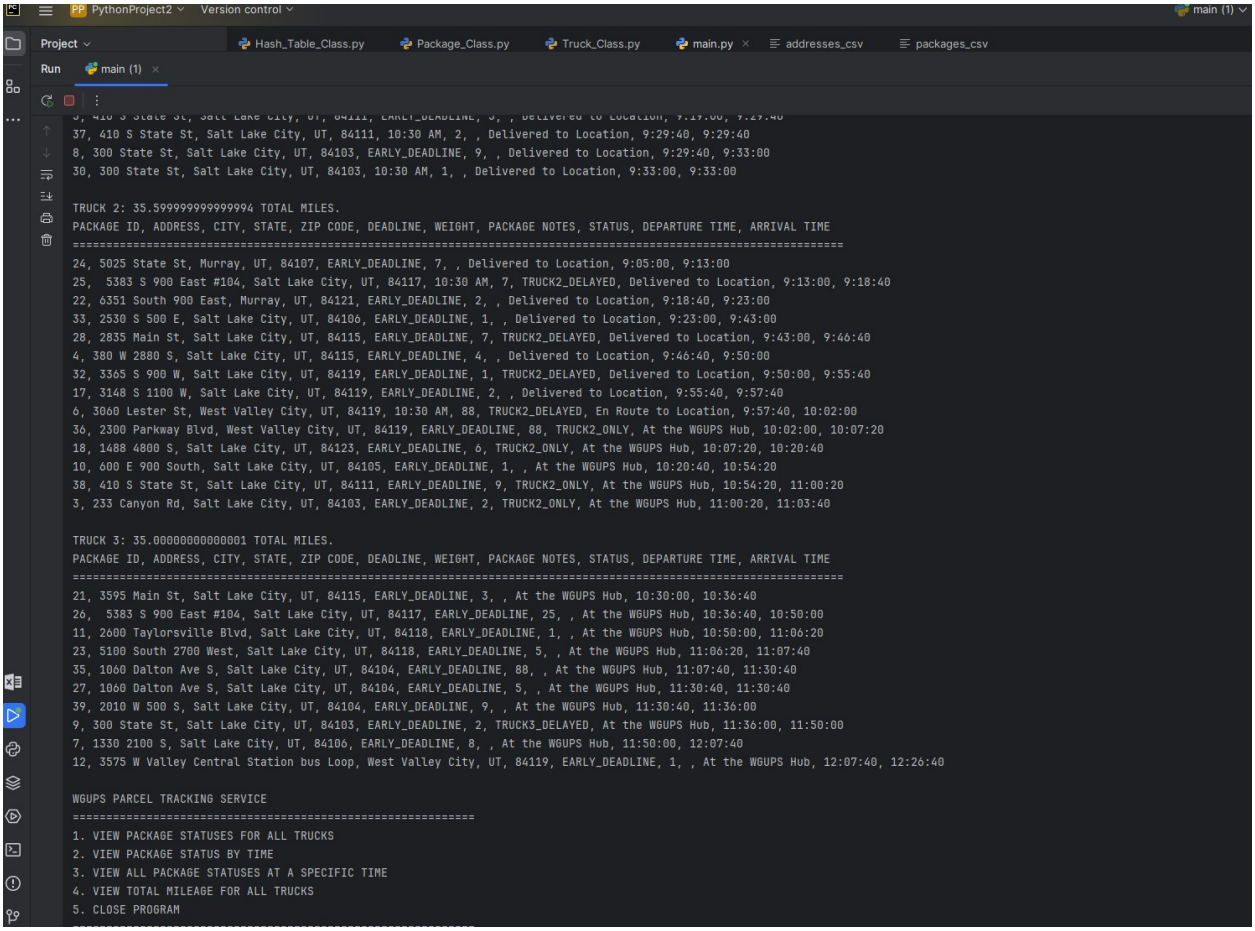
D1. First Status Check



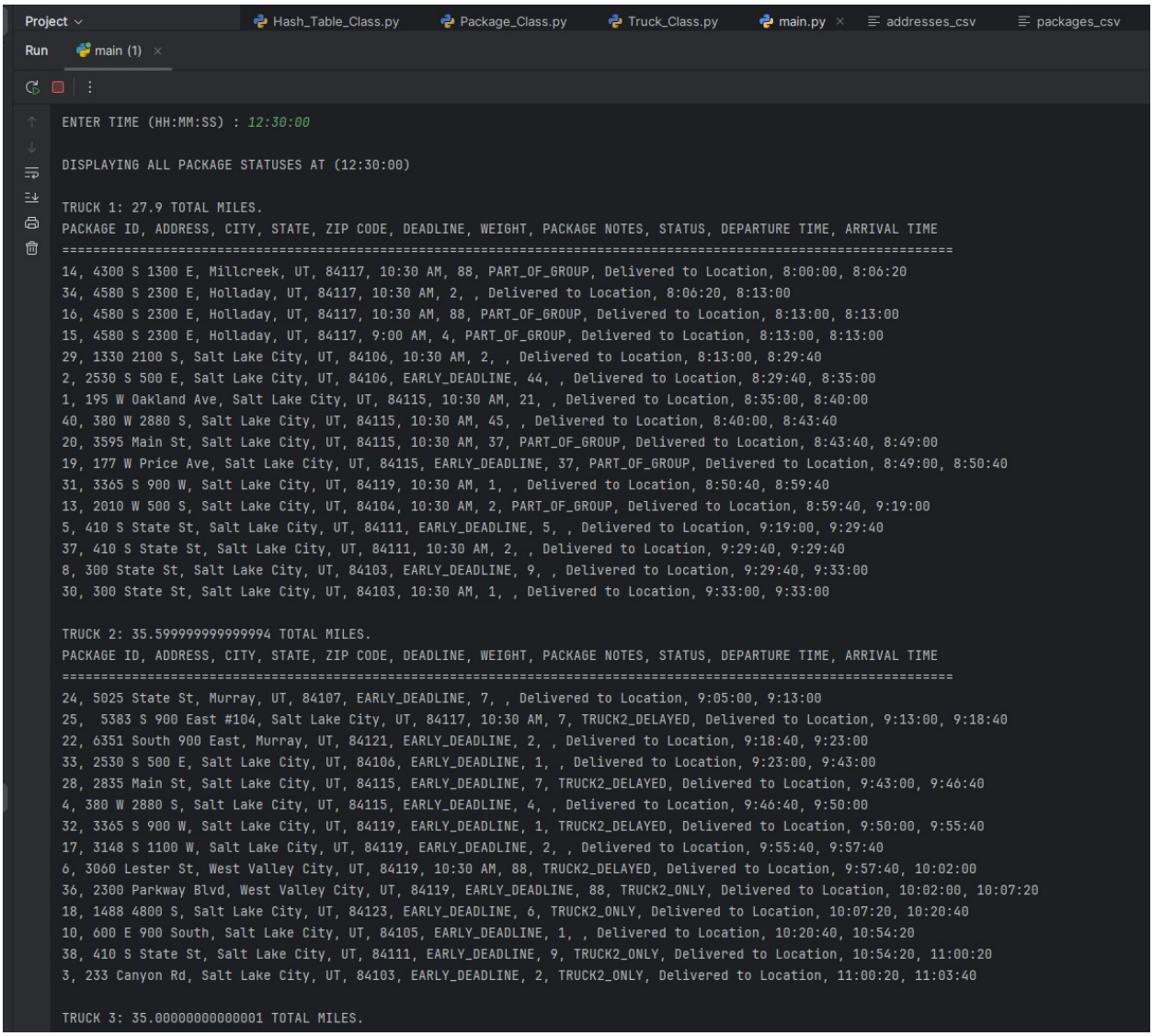


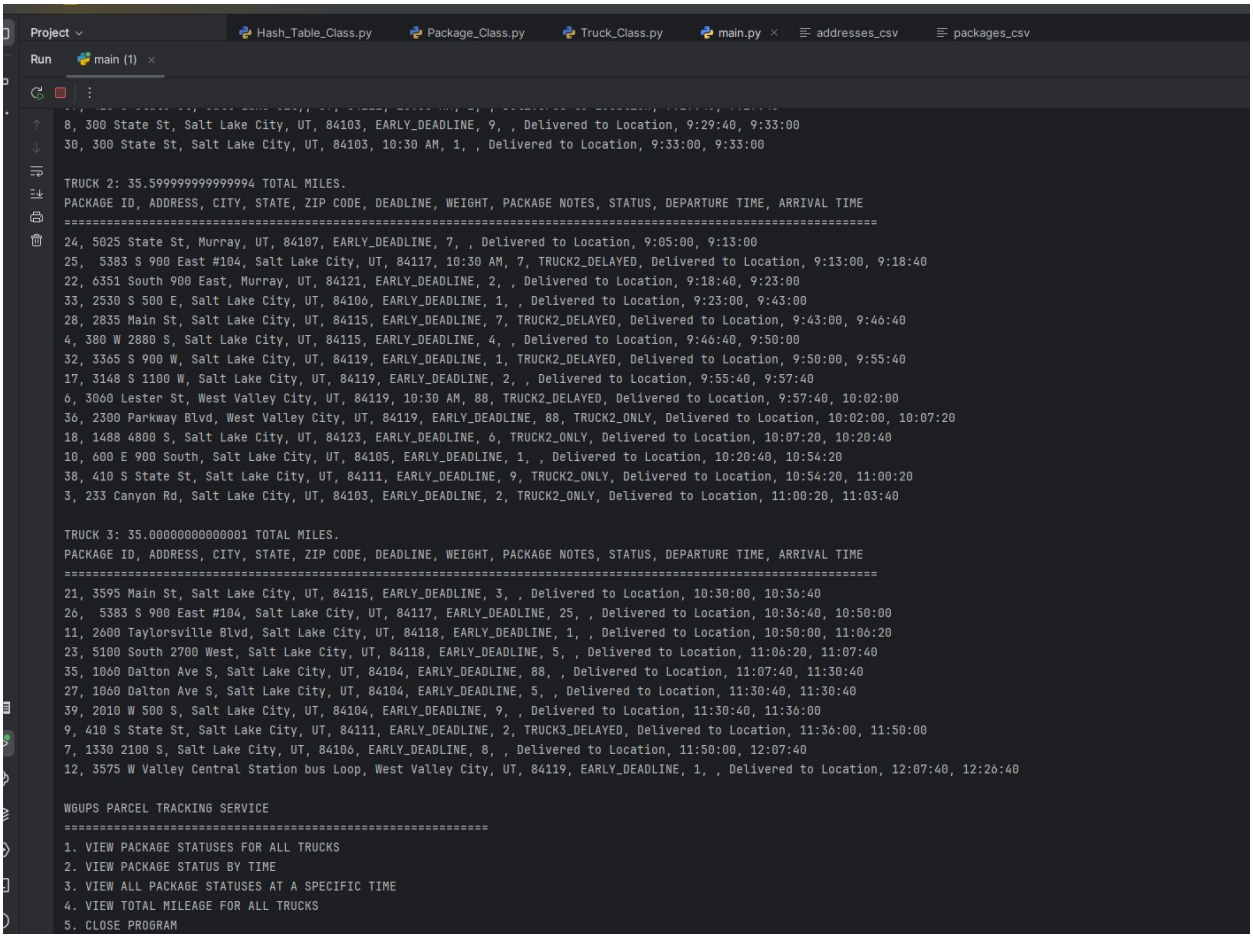
D2. Second Status Check


Project
Hash_Table_Class.py
Package_Class.py
Truck_Class.py
main.py
addresses_csv
packages_csv
Run
main (1)
WGUPS PARCEL TRACKING SERVICE
1. VIEW PACKAGE STATUSES FOR ALL TRUCKS
2. VIEW PACKAGE STATUS BY TIME
3. VIEW ALL PACKAGE STATUSES AT A SPECIFIC TIME
4. VIEW TOTAL MILEAGE FOR ALL TRUCKS
5. CLOSE PROGRAM
SELECT OPTION 1-5: 3
ENTER TIME (HH:MM:SS) : 10:00:00
DISPLAYING ALL PACKAGE STATUSES AT (10:00:00)
TRUCK 1: 27.9 TOTAL MILES.
PACKAGE ID, ADDRESS, CITY, STATE, ZIP CODE, DEADLINE, WEIGHT, PACKAGE NOTES, STATUS, DEPARTURE TIME, ARRIVAL TIME
14, 4300 S 1300 E, Millcreek, UT, 84117, 10:30 AM, 88, PART_OF_GROUP, Delivered to Location, 8:00:00, 8:06:20
34, 4580 S 2300 E, Holladay, UT, 84117, 10:30 AM, 2, , Delivered to Location, 8:06:20, 8:13:00
16, 4580 S 2300 E, Holladay, UT, 84117, 10:30 AM, 88, PART_OF_GROUP, Delivered to Location, 8:13:00, 8:13:00
15, 4580 S 2300 E, Holladay, UT, 84117, 9:00 AM, 4, PART_OF_GROUP, Delivered to Location, 8:13:00, 8:13:00
29, 1330 2100 S, Salt Lake City, UT, 84106, 10:30 AM, 2, , Delivered to Location, 8:13:00, 8:29:40
2, 2530 S 500 E, Salt Lake City, UT, 84106, EARLY_DEADLINE, 44, , Delivered to Location, 8:29:40, 8:35:00
1, 195 W Oakland Ave, Salt Lake City, UT, 84115, 10:30 AM, 21, , Delivered to Location, 8:35:00, 8:40:00
40, 380 W 2880 S, Salt Lake City, UT, 84115, 10:30 AM, 45, , Delivered to Location, 8:40:00, 8:43:40
20, 3595 Main St, Salt Lake City, UT, 84115, 10:30 AM, 37, PART_OF_GROUP, Delivered to Location, 8:43:40, 8:49:00
19, 177 W Price Ave, Salt Lake City, UT, 84115, EARLY_DEADLINE, 37, PART_OF_GROUP, Delivered to Location, 8:49:00, 8:50:40
31, 3365 S 900 W, Salt Lake City, UT, 84119, 10:30 AM, 1, , Delivered to Location, 8:50:40, 8:59:40
13, 2010 W 500 S, Salt Lake City, UT, 84104, 10:30 AM, 2, PART_OF_GROUP, Delivered to Location, 8:59:40, 9:19:00
5, 410 S State St, Salt Lake City, UT, 84111, EARLY_DEADLINE, 5, , Delivered to Location, 9:19:00, 9:29:40
37, 410 S State St, Salt Lake City, UT, 84111, 10:30 AM, 2, , Delivered to Location, 9:29:40, 9:29:40
8, 300 State St, Salt Lake City, UT, 84103, EARLY_DEADLINE, 9, , Delivered to Location, 9:29:40, 9:33:00
30, 300 State St, Salt Lake City, UT, 84103, 10:30 AM, 1, , Delivered to Location, 9:33:00, 9:33:00
TRUCK 2: 35.599999999999994 TOTAL MILES.
PACKAGE ID, ADDRESS, CITY, STATE, ZIP CODE, DEADLINE, WEIGHT, PACKAGE NOTES, STATUS, DEPARTURE TIME, ARRIVAL TIME
24, 5025 State St, Murray, UT, 84107, EARLY_DEADLINE, 7, , Delivered to Location, 9:05:00, 9:13:00
25, 5383 S 900 East #104, Salt Lake City, UT, 84117, 10:30 AM, 7, TRUCK2_DELAYED, Delivered to Location, 9:13:00, 9:18:40

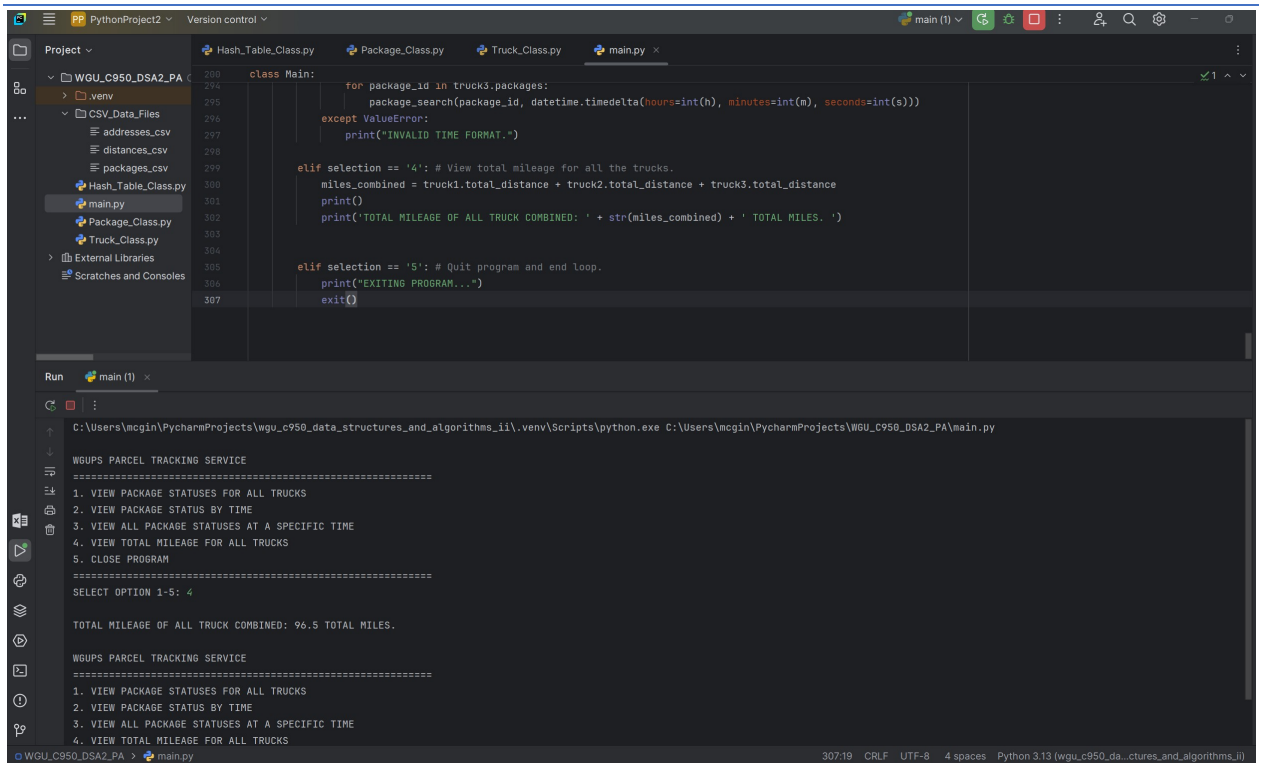


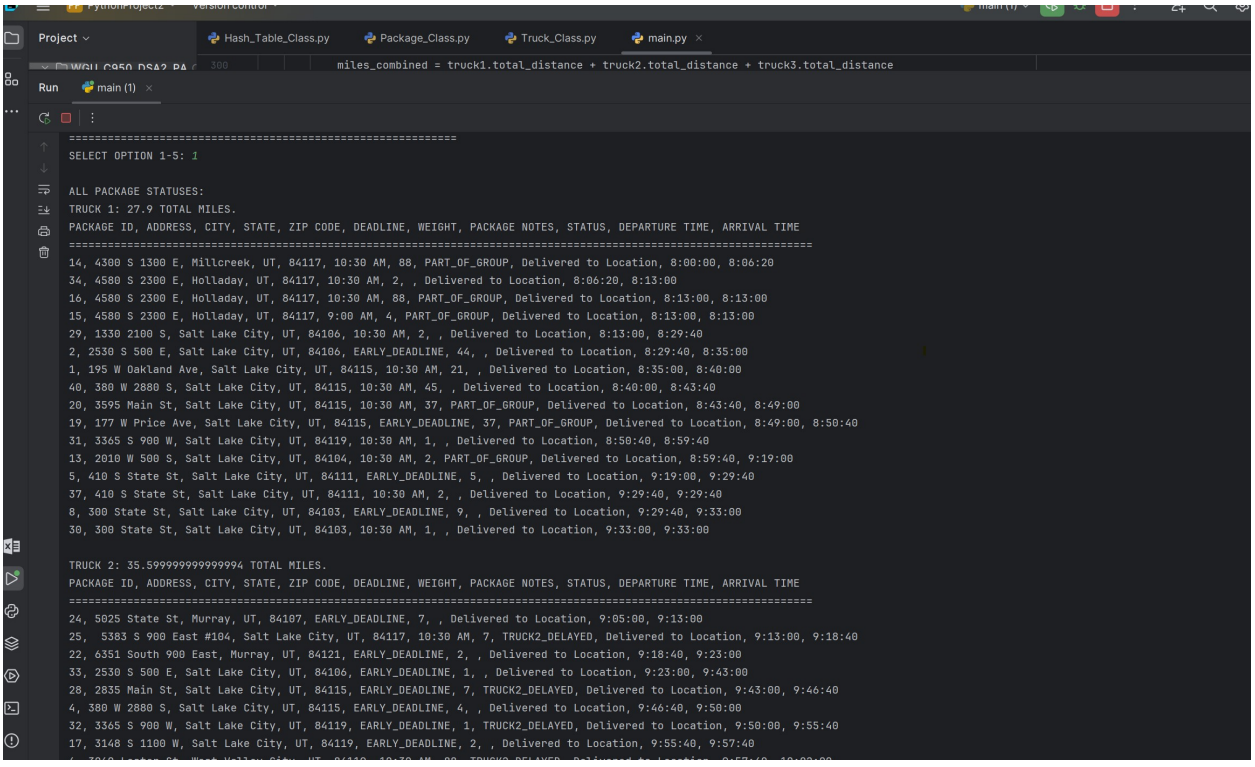
D3. Third Status Check

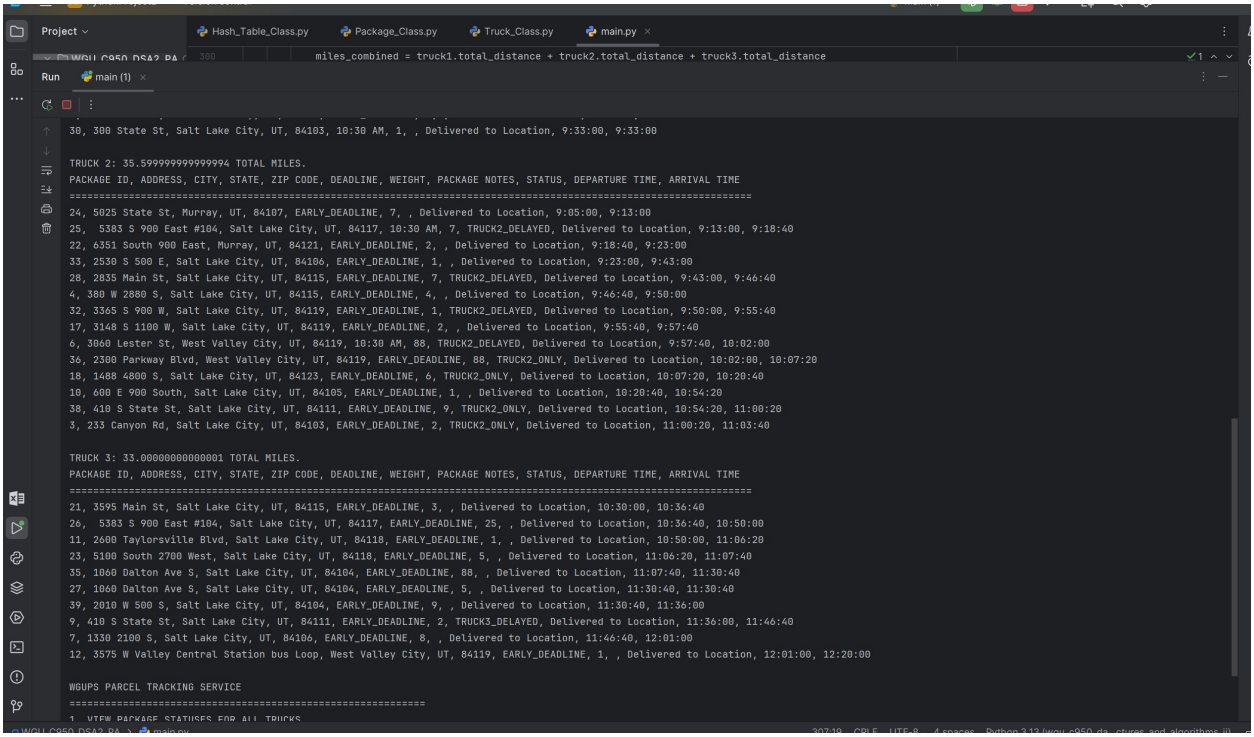

Project
Hash_Table_Class.py
Package_Class.py
Truck_Class.py
main.py
addresses_csv
packages_csv
Run
main (1)
ENTER TIME (HH:MM:SS) : 12:30:00
DISPLAYING ALL PACKAGE STATUSES AT (12:30:00)
TRUCK 1: 27.9 TOTAL MILES.
PACKAGE ID, ADDRESS, CITY, STATE, ZIP CODE, DEADLINE, WEIGHT, PACKAGE NOTES, STATUS, DEPARTURE TIME, ARRIVAL TIME
14, 4300 S 1300 E, Millcreek, UT, 84117, 10:30 AM, 88, PART_OF_GROUP, Delivered to Location, 8:00:00, 8:06:20
34, 4580 S 2300 E, Holladay, UT, 84117, 10:30 AM, 2, , Delivered to Location, 8:06:20, 8:13:00
16, 4580 S 2300 E, Holladay, UT, 84117, 10:30 AM, 88, PART_OF_GROUP, Delivered to Location, 8:13:00, 8:13:00
15, 4580 S 2300 E, Holladay, UT, 84117, 9:00 AM, 4, PART_OF_GROUP, Delivered to Location, 8:13:00, 8:13:00
29, 1330 2100 S, Salt Lake City, UT, 84106, 10:30 AM, 2, , Delivered to Location, 8:13:00, 8:29:40
2, 2530 S 500 E, Salt Lake City, UT, 84106, EARLY_DEADLINE, 44, , Delivered to Location, 8:29:40, 8:35:00
1, 195 W Oakland Ave, Salt Lake City, UT, 84115, 10:30 AM, 21, , Delivered to Location, 8:35:00, 8:40:00
40, 380 W 2880 S, Salt Lake City, UT, 84115, 10:30 AM, 45, , Delivered to Location, 8:40:00, 8:43:40
20, 3595 Main St, Salt Lake City, UT, 84115, 10:30 AM, 37, PART_OF_GROUP, Delivered to Location, 8:43:40, 8:49:00
19, 177 W Price Ave, Salt Lake City, UT, 84115, EARLY_DEADLINE, 37, PART_OF_GROUP, Delivered to Location, 8:49:00, 8:50:40
31, 3365 S 900 W, Salt Lake City, UT, 84119, 10:30 AM, 1, , Delivered to Location, 8:50:40, 8:59:40
13, 2010 W 500 S, Salt Lake City, UT, 84104, 10:30 AM, 2, PART_OF_GROUP, Delivered to Location, 8:59:40, 9:19:00
5, 410 S State St, Salt Lake City, UT, 84111, EARLY_DEADLINE, 5, , Delivered to Location, 9:19:00, 9:29:40
37, 410 S State St, Salt Lake City, UT, 84111, 10:30 AM, 2, , Delivered to Location, 9:29:40, 9:29:40
8, 300 State St, Salt Lake City, UT, 84103, EARLY_DEADLINE, 9, , Delivered to Location, 9:29:40, 9:33:00
30, 300 State St, Salt Lake City, UT, 84103, 10:30 AM, 1, , Delivered to Location, 9:33:00, 9:33:00
TRUCK 2: 35.599999999999994 TOTAL MILES.
PACKAGE ID, ADDRESS, CITY, STATE, ZIP CODE, DEADLINE, WEIGHT, PACKAGE NOTES, STATUS, DEPARTURE TIME, ARRIVAL TIME
24, 5025 State St, Murray, UT, 84107, EARLY_DEADLINE, 7, , Delivered to Location, 9:05:00, 9:13:00
25, 5383 S 900 East #104, Salt Lake City, UT, 84117, 10:30 AM, 7, TRUCK2_DELAYED, Delivered to Location, 9:13:00, 9:18:40
22, 6351 South 900 East, Murray, UT, 84121, EARLY_DEADLINE, 2, , Delivered to Location, 9:18:40, 9:23:00
33, 2530 S 500 E, Salt Lake City, UT, 84106, EARLY_DEADLINE, 1, , Delivered to Location, 9:23:00, 9:43:00
28, 2835 Main St, Salt Lake City, UT, 84115, EARLY_DEADLINE, 7, TRUCK2_DELAYED, Delivered to Location, 9:43:00, 9:46:40
4, 380 W 2880 S, Salt Lake City, UT, 84115, EARLY_DEADLINE, 4, , Delivered to Location, 9:46:40, 9:50:00
32, 3365 S 900 W, Salt Lake City, UT, 84119, EARLY_DEADLINE, 1, TRUCK2_DELAYED, Delivered to Location, 9:50:00, 9:55:40
17, 3148 S 1100 W, Salt Lake City, UT, 84119, EARLY_DEADLINE, 2, , Delivered to Location, 9:55:40, 9:57:40
6, 3060 Lester St, West Valley City, UT, 84119, 10:30 AM, 88, TRUCK2_DELAYED, Delivered to Location, 9:57:40, 10:02:00
36, 2300 Parkway Blvd, West Valley City, UT, 84119, EARLY_DEADLINE, 88, TRUCK2_ONLY, Delivered to Location, 10:02:00, 10:07:20
18, 1488 4800 S, Salt Lake City, UT, 84123, EARLY_DEADLINE, 6, TRUCK2_ONLY, Delivered to Location, 10:07:20, 10:20:40
10, 600 E 900 South, Salt Lake City, UT, 84105, EARLY_DEADLINE, 1, , Delivered to Location, 10:20:40, 10:54:20
38, 410 S State St, Salt Lake City, UT, 84111, EARLY_DEADLINE, 9, TRUCK2_ONLY, Delivered to Location, 10:54:20, 11:00:20
3, 233 Canyon Rd, Salt Lake City, UT, 84103, EARLY_DEADLINE, 2, TRUCK2_ONLY, Delivered to Location, 11:00:20, 11:03:40
TRUCK 3: 35.00000000000001 TOTAL MILES.


Project
Hash_Table_Class.py
Package_Class.py
Truck_Class.py
main.py
addresses_csv
packages_csv
Run
main (1)
8, 300 State St, Salt Lake City, UT, 84103, EARLY_DEADLINE, 9, , Delivered to Location, 9:29:40, 9:33:00
30, 300 State St, Salt Lake City, UT, 84103, 10:30 AM, 1, , Delivered to Location, 9:33:00, 9:33:00
TRUCK 2: 35.599999999999994 TOTAL MILES.
PACKAGE ID, ADDRESS, CITY, STATE, ZIP CODE, DEADLINE, WEIGHT, PACKAGE NOTES, STATUS, DEPARTURE TIME, ARRIVAL TIME
24, 5025 State St, Murray, UT, 84107, EARLY_DEADLINE, 7, , Delivered to Location, 9:05:00, 9:13:00
25, 5383 S 900 East #104, Salt Lake City, UT, 84117, 10:30 AM, 7, TRUCK2_DELAYED, Delivered to Location, 9:13:00, 9:18:40
22, 6351 South 900 East, Murray, UT, 84121, EARLY_DEADLINE, 2, , Delivered to Location, 9:18:40, 9:23:00
33, 2530 S 500 E, Salt Lake City, UT, 84106, EARLY_DEADLINE, 1, , Delivered to Location, 9:23:00, 9:43:00
28, 2835 Main St, Salt Lake City, UT, 84115, EARLY_DEADLINE, 7, TRUCK2_DELAYED, Delivered to Location, 9:43:00, 9:46:40
4, 380 W 2880 S, Salt Lake City, UT, 84115, EARLY_DEADLINE, 4, , Delivered to Location, 9:46:40, 9:50:00
32, 3365 S 900 W, Salt Lake City, UT, 84119, EARLY_DEADLINE, 1, TRUCK2_DELAYED, Delivered to Location, 9:50:00, 9:55:40
17, 3148 S 1100 W, Salt Lake City, UT, 84119, EARLY_DEADLINE, 2, , Delivered to Location, 9:55:40, 9:57:40
6, 3060 Lester St, West Valley City, UT, 84119, 10:30 AM, 88, TRUCK2_DELAYED, Delivered to Location, 9:57:40, 10:02:00
36, 2300 Parkway Blvd, West Valley City, UT, 84119, EARLY_DEADLINE, 88, TRUCK2_ONLY, Delivered to Location, 10:02:00, 10:07:20
18, 1488 4800 S, Salt Lake City, UT, 84123, EARLY_DEADLINE, 6, TRUCK2_ONLY, Delivered to Location, 10:07:20, 10:20:40
10, 600 E 900 South, Salt Lake City, UT, 84105, EARLY_DEADLINE, 1, , Delivered to Location, 10:20:40, 10:54:20
38, 410 S State St, Salt Lake City, UT, 84111, EARLY_DEADLINE, 9, TRUCK2_ONLY, Delivered to Location, 10:54:20, 11:00:20
3, 233 Canyon Rd, Salt Lake City, UT, 84103, EARLY_DEADLINE, 2, TRUCK2_ONLY, Delivered to Location, 11:00:20, 11:03:40
TRUCK 3: 35.00000000000001 TOTAL MILES.
PACKAGE ID, ADDRESS, CITY, STATE, ZIP CODE, DEADLINE, WEIGHT, PACKAGE NOTES, STATUS, DEPARTURE TIME, ARRIVAL TIME
21, 3595 Main St, Salt Lake City, UT, 84115, EARLY_DEADLINE, 3, , Delivered to Location, 10:30:00, 10:36:40
26, 5383 S 900 East #104, Salt Lake City, UT, 84117, EARLY_DEADLINE, 25, , Delivered to Location, 10:36:40, 10:50:00
11, 2600 Taylorsville Blvd, Salt Lake City, UT, 84118, EARLY_DEADLINE, 1, , Delivered to Location, 10:50:00, 11:06:20
23, 5100 South 2700 West, Salt Lake City, UT, 84118, EARLY_DEADLINE, 5, , Delivered to Location, 11:06:20, 11:07:40
35, 1060 Dalton Ave S, Salt Lake City, UT, 84104, EARLY_DEADLINE, 88, , Delivered to Location, 11:07:40, 11:30:40
27, 1060 Dalton Ave S, Salt Lake City, UT, 84104, EARLY_DEADLINE, 5, , Delivered to Location, 11:30:40, 11:30:40
39, 2010 W 500 S, Salt Lake City, UT, 84104, EARLY_DEADLINE, 9, , Delivered to Location, 11:30:40, 11:36:00
9, 410 S State St, Salt Lake City, UT, 84111, EARLY_DEADLINE, 2, TRUCK3_DELAYED, Delivered to Location, 11:36:00, 11:50:00
7, 1330 2100 S, Salt Lake City, UT, 84106, EARLY_DEADLINE, 8, , Delivered to Location, 11:50:00, 12:07:40
12, 3575 W Valley Central Station bus Loop, West Valley City, UT, 84119, EARLY_DEADLINE, 1, , Delivered to Location, 12:07:40, 12:26:40
WGUPS PARCEL TRACKING SERVICE
1. VIEW PACKAGE STATUSES FOR ALL TRUCKS
2. VIEW PACKAGE STATUS BY TIME
3. VIEW ALL PACKAGE STATUSES AT A SPECIFIC TIME
4. VIEW TOTAL MILEAGE FOR ALL TRUCKS
5. CLOSE PROGRAM

E. Screenshot of Code Execution







F1. Strengths of the Chosen Algorithm

One of the main strengths of the nearest neighbor algorithm is the ease of implementation. It is less complex than similar algorithms while still efficiently choosing the closest location. It also is a greedy algorithm, which uses the immediate location to find the next path. This algorithm works well for small datasets, though it may have time complexity issues for larger datasets.

F2. Verification of Algorithm

The Nearest Neighbor algorithm does meet all the requirements for this scenario by determining the closest location throughout the route of each truck. It is used to efficiently deliver packages with minimal issues, as this is a relatively small set of data.

F3. Other Possible Algorithms

Other possible algorithms would include Dijkstra's algorithm and the A* (A Star) algorithm.

F3a. Algorithm Differences

Dijkstra's Algorithm would be able to determine all possible routes to find the shortest path, where the Nearest Neighbor only uses the current location. Though it is more complex to implement, it would improve the time complexity to O(n log n), which would be beneficial for larger datasets.

A* algorithm is similar to Dijkstra's algorithm with the addition of heuristics to prioritize a certain path, such as priority. This would also improve performance with large datasets.

G. Different Approach

If I were to take an alternate approach to the project, I would have explored the option of implementing one of the algorithms listed above to optimize time complexity. Also, if the constraints were removed for using external libraries, I would be able to remove the nested loops used for sorting and distance calculation and handle those using the NumPy or pandas libraries.

I would also consider breaking down some of the logic into additional helper methods, particularly the truck routing or distance calculations, to make the code more modular.

H. Verification of Data Structure

The hash table data structure easily meets the requirements of the program. It provides O(1) time complexity for inserting, updating, and searching, which efficiently stores the data using the package ID as a key.

This data structure will also properly scale to accommodate larger sets of data while maintaining stability and speed of access.

H1. Other Data Structures

A Binary Search Tree or an Array would also be reasonable data structures to use in this scenario.

H1a. Data Structure Differences

A Binary Search Tree would use a tree-based structure and organize elements hierarchically, which would be O(log n) time complexity if the tree is balanced. This is less efficient than the hash table, however, which maps the keys to a specific memory location.

An array could be used in other circumstances. This would store the information sequentially in a fixed-sized structure, but it would not allow access based on keys as the hash table does. The time complexity for an array would be O(n), which would be less efficient than the hash table.

I. Sources

Lysecky, R., & Vahid, F. (2018, June). C950: Data Structures and Algorithms II. zyBooks. Retrieved February 24, 2025, from https://learn.zybooks.com/zybook/WGUC950Template2023