Department of Computer Science
American International University-Bangladesh
Mid Term Report

Course Name: Programming in Python

"A report on Transport Management System"

Supervised By:

Dr. Abdus Salam

Assistant Professor, Computer Science-AIUB

Submitted By:

Rasel Mahmud

ID: 20-43867-2

Section: A

Mohammad Shafin

ID: 20-43736-2

Section: A

Submission Date: November 11, 2023.

Project Description:

The Transportation Management System (TMS) I am developing for Dhaka City and Chittagong City in Bangladesh is a comprehensive solution addressing both intracity and district transportation needs. This system facilitates users in planning their journeys by allowing them to input their source and destination, generating a list of available buses along with corresponding fares. The platform is designed to streamline the user experience, offering a seamless interface for bus selection and seat booking. Whether commuting within the city or traveling to different districts, users can access a database of transportation services, making informed decisions based on availability and cost. Additionally, the system provides information on local transportation options within each city. With the goal of enhancing efficiency and convenience, our project not only connects users with transportation services but also contributes to the overall improvement of public transit accessibility in Dhaka and Chittagong.

Project Features:

1.Registration:

1. __init__(self)
    - Initializes a new instance of the Registration class.
    - Creates an empty dictionary (self.users) to store user information.
2. register(self, username, email, name, password, phone)
    - Registers a new user with the provided information.
    - Checks if the username, email, and phone number are unique; if not, it prints corresponding messages.
    - Validates the phone number, email, password, and name using helper methods.
    - If all validations pass, creates a user information dictionary and adds it to the self.users dictionary with the username as the key.
    - Prints a success message if the registration is successful.
3. is_valid_password(self, password)
    - Checks if a password is valid.
    - Verifies that the password is at least 6 characters long and contains at least 2 digits and 2 special characters.
    - Returns True if the password is valid, otherwise False.

4. is_valid_phone(self, phone)
    - Uses a regular expression to check if a phone number is valid.
    - Ensures that the phone number is 11 digits long and starts with '01' followed by a digit from 3 to 9.
    - Returns True if the phone number is valid, otherwise False.
5. is_valid_email(self, email)
    - Uses a regular expression to check if an email address is valid.
    - Validates the email format.
    - Returns True if the email is valid, otherwise False.

2. Login View Profile and Edit Profile

This code defines a class named Login that inherits from the Registration class. The Login class adds functionality related to user login, viewing and editing user profiles, and accessing transportation services. Additionally, it utilizes two hypothetical classes, Dhaka City and Chittagong City, which are used to handle transportation services for the respective cities.

1. __init__(self)
    - Initializes a new instance of the Login class, calling the constructor of the parent class (Registration) using super().__init__().
2. login(self, username, password)
    - Validates user login by checking if the provided username exists and if the associated password matches.
    - If the login is successful, it prints a welcome message and calls the show_options method.
    - If the login fails, it prints an error message.
3. show_options(self, username)
    - Displays a menu of options for the logged-in user.
    - Allows the user to view their profile, take transportation services, or go back to the main menu.
4. view_profile(self, username)

- Displays the user's profile information.
- Provides options to edit the user's profile, including name, email, phone, and password.

5. transportation_service(self)
    - Presents options for transportation services, allowing the user to choose between Dhaka, Chittagong, or go back to the main menu.
    - If the user chooses Dhaka or Chittagong, it instantiates the corresponding hypothetical city classes (Dhaka City or Chittagong City) and calls their run methods.

3. Dhaka City

Local:

1. __init__(self)
    - Initializes a new instance of the DhakaCity class.
    - Creates an empty dictionary (self.seat_availability) to keep track of seat availability for different bus operators.
2. run(self)
    - Runs the main loop for the Dhaka City Transportation Services.
    - Displays a menu of options for intra-city and district transportation.
    - Calls corresponding methods based on the user's choice.
3. load_bus_routes_from_drive(self, file_path)
    - Loads bus routes from a JSON file located at the specified file_path.
    - Returns the loaded bus routes.
4. calculate_fare(self, distance)
    - Calculates the fare based on the given distance.
    - Uses a base fare and a fare-per-kilometer rate.
5. find_available_buses(self, source, destination, bus_routes)
    - Finds available buses based on the provided source and destination.
    - Returns a list of available buses.

6. find_route(self, source, destination, bus_routes)
    - Finds the route information for a given source and destination.
    - Returns the operator and total fare for the route.
7. Intra_City_Transportation(self)
    - Handles the process of intra-city transportation within Dhaka.
    - Allows the user to input source and destination.
    - Finds and displays available buses, allowing the user to select an operator and book seats.
8. check_seat_availability(self, operator, num_seats)
    - Checks seat availability for a given bus operator and number of seats.
    - If the operator is not in the seat availability dictionary, initializes it with a default value.
9. update_seat_availability(self, operator, num_seats)
    - Updates seat availability after a successful booking.
    - Reduces the number of available seats for the specified operator.
10. District_Transportation(self)
    - Handles district transportation from Dhaka to a user-specified destination.
    - As this method references a class DhakaDistrictTransportation.

District:

1. __init__(self)
    - Initializes a new instance of the DhakaDistrictTransportation class.
    - Creates an empty list (self.bus_routes) to store bus route information.
2. load_bus_routes_from_drive(self, file_path)
    - Loads bus routes from a JSON file located at the specified file_path.
    - Populates the self.bus_routes list with the loaded data.
3. find_available_buses(self, source, destination)
    - Finds available buses based on the provided source and destination.
    - Displays available buses, their operators, duration, distance, and fare information.

- Allows the user to book a bus ticket or go back to the main menu.

4. book_bus_ticket(self, available_buses, route)
    - Handles the process of booking a bus ticket.
    - Prompts the user to choose an operator, specify the number of seats, and select available seats.
    - Calculates the total fare based on the distance and the number of seats.
    - Asks for the payment method and confirms the booking details.
    - Optionally allows the user to delete the ticket, releasing the selected seats if desired.

4.Chittagong City

The ChittagongCity class handles transportation services in Chittagong City. Similar to the explanation for the DhakaCity class, here's an explanation of the functions in the ChittagongCity class:

Local:

1. __init__(self)
    - Initializes a new instance of the ChittagongCity class.
    - Creates an empty dictionary (self.seat_availability) to keep track of seat availability for different bus operators.
2. run(self)
    - Runs the main loop for Chittagong City Transportation Services.
    - Displays a menu of options for intra-city and district transportation.
    - Calls corresponding methods based on the user's choice.
3. load_bus_routes_from_drive(self, file_path)
    - Loads bus routes from a JSON file located at the specified file_path.
    - Returns the loaded bus routes.
4. find_available_buses(self, source, destination, bus_routes)
    - Finds available buses based on the provided source and destination.
    - Returns a list of available buses.
5. find_route(self, source, destination, bus_routes)

- Finds the route information for a given source and destination.
- Returns the operator and total fare for the route.

6. calculate_fare(self, distance)
    - Calculates the fare based on the given distance.
    - Uses a base fare and a fare-per-kilometer rate.
7. check_seat_availability(self, operator, num_seats)
    - Checks seat availability for a given bus operator and number of seats.
    - If the operator is not in the seat availability dictionary, initializes it with a default value.
8. update_seat_availability(self, operator, num_seats)
    - Updates seat availability after a successful booking.
    - Reduces the number of available seats for the specified operator.
9. Intra_City_Transportation(self)
    - Handles the process of intra-city transportation within Chittagong.
    - Allows the user to input source and destination.
    - Finds and displays available buses, allowing the user to select an operator and book seats.
10. District_Transportation(self)
    - Handles district transportation from Chittagong to a user-specified destination.
    - As this method references a class ChittagongDistrictTransportation.

District:

1. __init__(self)
    - Initializes a new instance of the ChittagongDistrictTransportation class.
    - Creates an empty list (self.bus_routes) to store bus route information.
2. load_bus_routes_from_drive(self, file_path)
    - Loads bus routes from a JSON file located at the specified file_path.
    - Populates the self.bus_routes list with the loaded data.
3. find_available_buses(self, source, destination)

- Finds available buses based on the provided source and destination.
- Displays available buses, their operators, duration, distance, and fare information.
- Allows the user to book a bus ticket or go back to the main menu.

4. book_bus_ticket(self, available_buses, route)
    - Handles the process of booking a bus ticket.
    - Prompts the user to choose an operator, specify the number of seats, and select available seats.
    - Calculates the total fare based on the distance and the number of seats.
    - Asks for the payment method and confirms the booking details.
    - Optionally allows the user to delete the ticket, releasing the selected seats if desired.