SOLENT UNIVERSITY
Department of Science and Engineering

MSc Computer Engineering

Software Design and Development - COM714

Academic Year 2023-2024

Travel Management System

Q102144251

Report-Assignment 1

Tutor: Dr. Taiwo Ayodele **[08/05/2024]**

Table of Contents

The Travel Management System Project

The Travel Management System project provided a rich learning experience, offering valuable insights into various aspects of software development. This comprehensive breakdown delves deeper into the concepts explored, highlighting key takeaways and considerations for future endeavors. Here, we'll focus on the provided code snippet, analyzing its functionalities and demonstrating core programming principles in action.

Object-Oriented Programming (OOP) in Action

The project served as a practical playground for applying OOP principles. You created core classes like Trip and Traveller, representing real-world entities with their attributes and functionalities. Defining the relationship between these classes (one Trip can have many Travellers) solidified the concept of modeling real-world interactions in code.

Understanding Data Encapsulation and Abstraction:

By creating classes, you likely utilized data encapsulation, restricting direct access to internal data members. Instead, methods were defined within the class to manipulate and access data appropriately. This promotes data integrity and security while making the code more maintainable. Abstraction further complemented encapsulation by focusing on what the class can do (methods) rather than how it does it (implementation details hidden within the class). This improves code readability and reusability.

Building a User Interface with tkinter

The project involved designing a user-friendly interface using tkinter, a Python library for creating graphical user interfaces (GUIs). You likely used various widgets like labels, entry fields, buttons, and listboxes to allow users to interact with the system. Handling user interactions through events (e.g., button clicks) provided practical experience in making the application responsive.

Exploring the Limits of tkinter:

While tkinter offered a solid foundation for building a basic GUI, you might have encountered limitations for complex layouts or visually appealing interfaces. This highlights the need to consider alternative GUI frameworks like PyQt or Kivy for projects requiring more advanced user interface features.

Data Persistence: Stepping Stone to Robust Solutions

The project explored data persistence with CSV (Comma-Separated Values) files. You likely used CSV files to store trip and traveler data, allowing information to be retrieved and saved beyond the application's runtime. This experience serves as a stepping stone to understanding more robust database solutions like SQL or NoSQL databases. These databases are better equipped to handle complex data structures, large amounts of data, and frequent updates, making them ideal for larger-scale applications.

Planning and Design: The Power of Foresight

The importance of planning and design was evident in the use of UML diagrams (Class Diagrams and Use Case Diagrams). By visually representing the system's components and user interactions before coding, you likely identified potential issues early on, saving time and effort during development. This emphasizes the value of planning and design to ensure a well-structured and maintainable application.

Version Control with Git: A Safety Net for Development

The project showcased the benefits of version control with Git. Git allows you to track changes made to your code over time, revert to previous versions if needed, and collaborate effectively with others. This project likely involved committing your code regularly to a Git repository, creating a valuable history of the development process. This safety net ensures you can always backtrack if issues arise and provides a collaborative platform for team-based development.

Following Best Practices: The Mark of Professionalism

Adhering to the specified libraries like tkinter and csv, along with following recommended coding conventions, likely improved the overall quality of your code. Using appropriate libraries ensures you leverage existing functionalities rather than reinventing the wheel. Additionally, consistent coding style makes your code more readable and maintainable, not only for yourself but also for others who might collaborate on the project in the future.

Project Scope Management: Prioritization and Focus

The project likely required managing the scope to deliver core functionalities within a specific timeframe. This might have involved leaving out advanced features like online payments or itinerary generation. This experience emphasizes the importance of prioritizing essential features and ensuring the project remains focused to achieve its core objectives within the allocated resources.

Testing and Debugging: The Importance of Continuous Improvement

Testing and debugging became crucial aspects of the development process. As you built the Travel Management System, you likely encountered unexpected behavior or errors in your code. The process of identifying, understanding, and fixing these issues (debugging) is essential for delivering a functional and reliable application. Continuous testing throughout development helps catch issues early on, preventing them from snowballing into larger problems later.

Documentation: The Key to Understanding

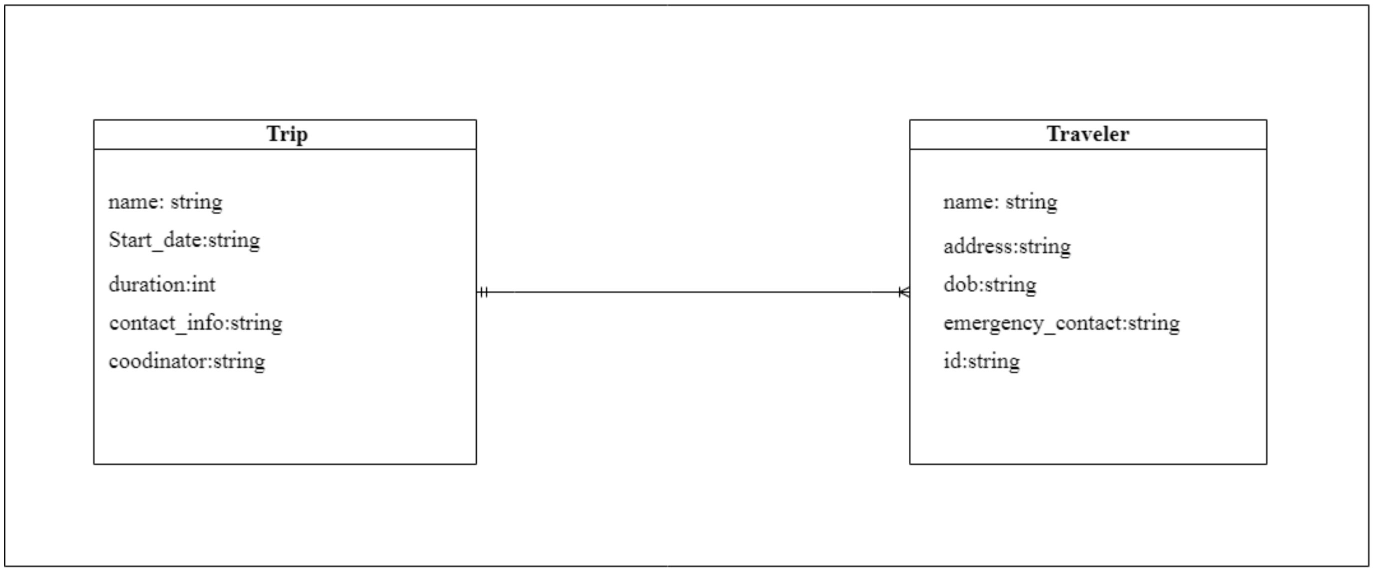
The project solidified the value of good documentation. Writing comments within your code to explain specific functionalities and logic likely improved code readability and maintainability. Additionally, creating UML diagrams visually represented the system's architecture and interactions, further enhancing understanding for yourself and others who might interact with the codebase in the future.

Looking Forward: Building on the Foundation

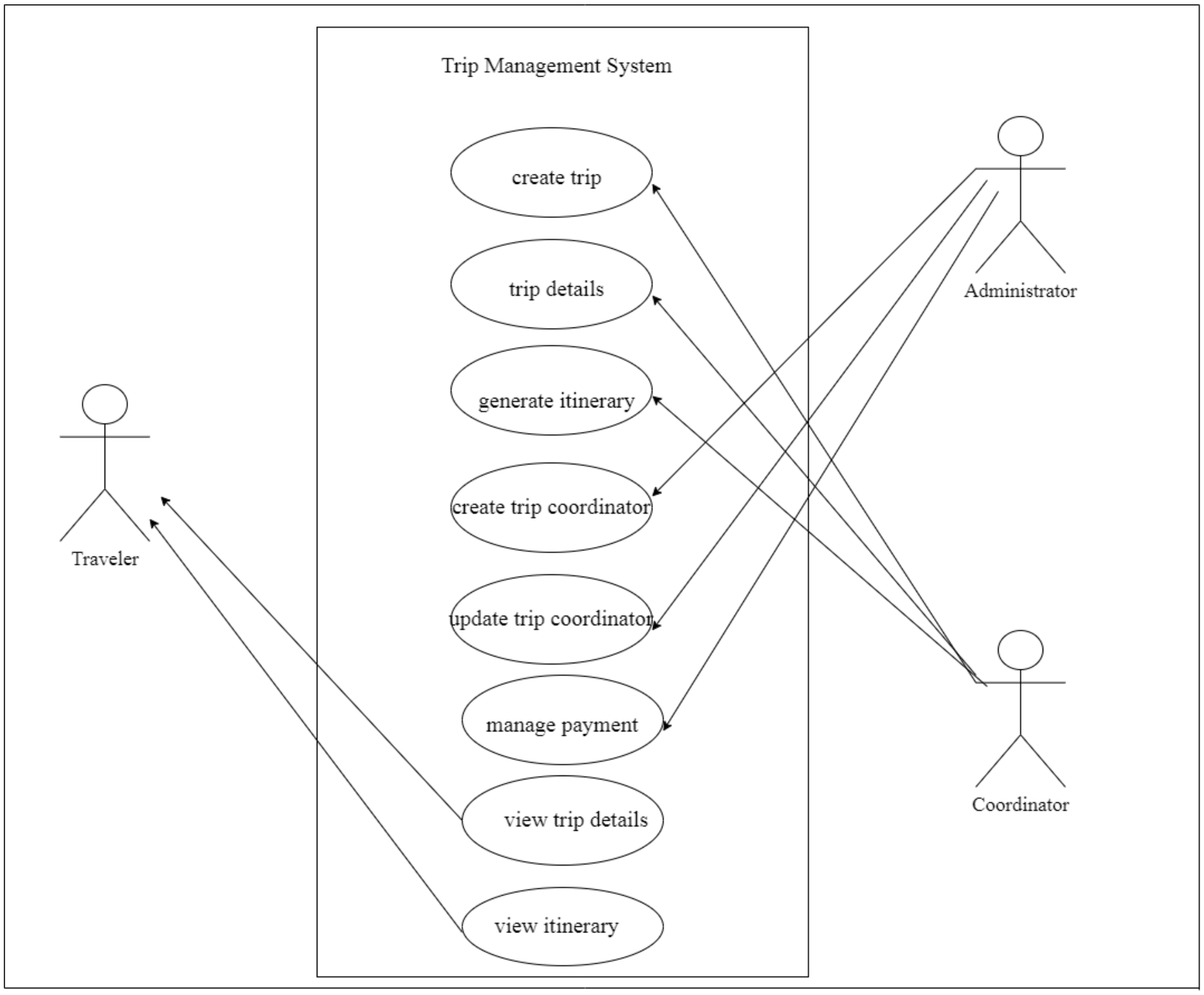
The Travel Management System project provided a well-rounded introduction to core software development concepts. You gained practical experience in areas like OOP, GUI development, data persistence, project management, and version control. This foundation prepares you for more advanced software development endeavors. Consider exploring the following areas to further broaden your skillset:

Diagrams

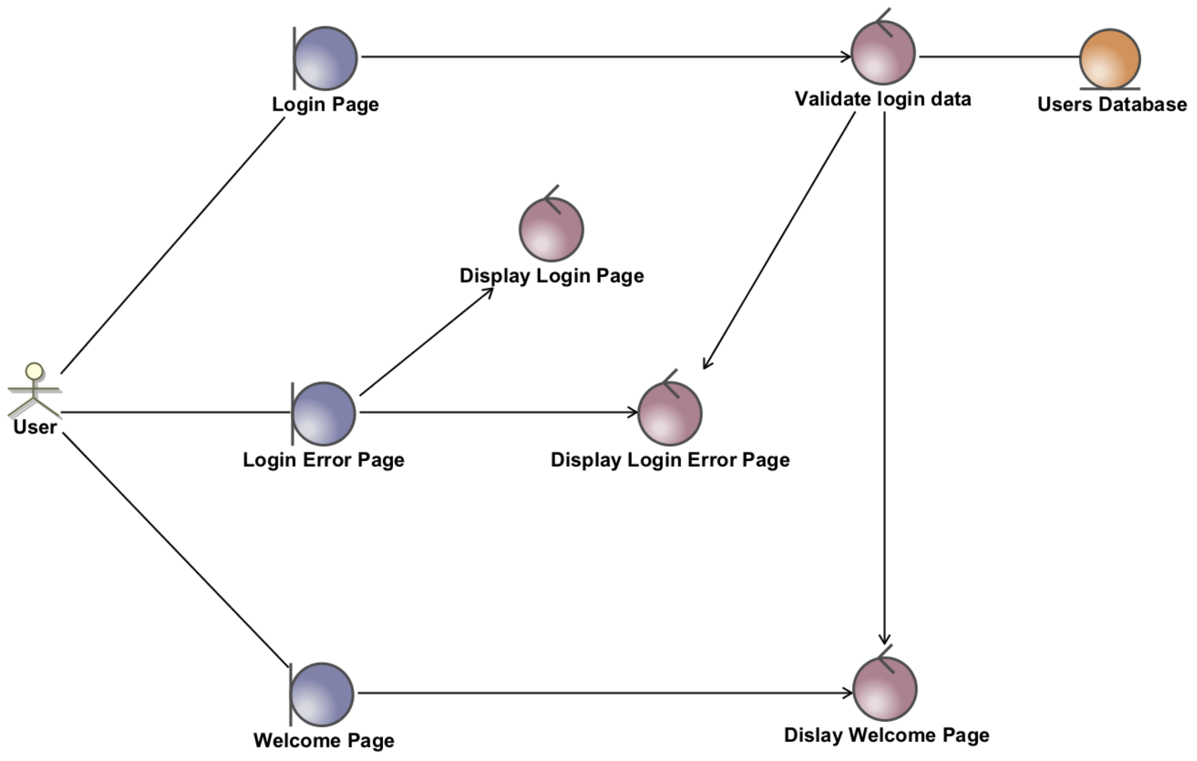
1. Class diagram



2. Use Case diagram



3. Robustness diagram



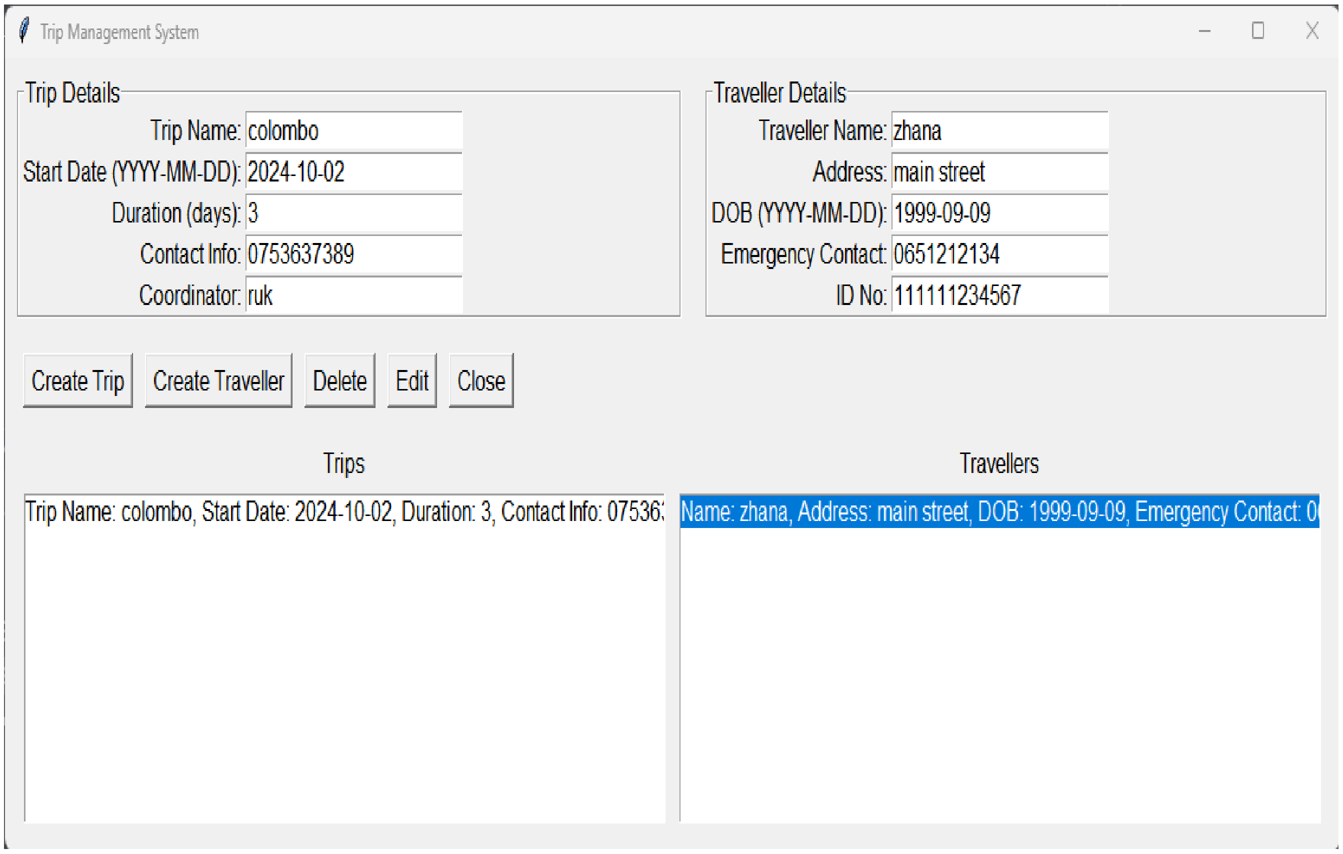
Code

```
import tkinter as tk
from tkinter import messagebox
import csv
import json
import datetime

class TripManagementApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Trip Management System")

        self.create_trip_fields()
        self.create_traveller_fields()
        self.create_buttons()
        self.create_listboxes()

    def create_trip_fields(self):
        trip_frame = tk.LabelFrame(self.root, text="Trip Details",
font=("Arial", 12))
        trip_frame.grid(row=0, column=0, padx=10, pady=10, sticky="nsew")

        tk.Label(trip_frame, text="Trip Name:", font=("Arial",
12)).grid(row=0, column=0, sticky="e")
        self.trip_name_entry = tk.Entry(trip_frame, font=("Arial", 12))
        self.trip_name_entry.grid(row=0, column=1)

        tk.Label(trip_frame, text="Start Date (YYYY-MM-DD):", font=("Arial",
12)).grid(row=1, column=0, sticky="e")
        self.start_date_entry = tk.Entry(trip_frame, font=("Arial", 12))
        self.start_date_entry.grid(row=1, column=1)

        tk.Label(trip_frame, text="Duration (days):", font=("Arial",
12)).grid(row=2, column=0, sticky="e")
        self.duration_entry = tk.Entry(trip_frame, font=("Arial", 12))
        self.duration_entry.grid(row=2, column=1)

        tk.Label(trip_frame, text="Contact Info:", font=("Arial",
12)).grid(row=3, column=0, sticky="e")
        self.contact_info_entry = tk.Entry(trip_frame, font=("Arial", 12))
        self.contact_info_entry.grid(row=3, column=1)

        tk.Label(trip_frame, text="Coordinator:", font=("Arial",
12)).grid(row=4, column=0, sticky="e")
        self.coordinator_entry = tk.Entry(trip_frame, font=("Arial", 12))
        self.coordinator_entry.grid(row=4, column=1)

    def create_traveller_fields(self):
        traveller_frame = tk.LabelFrame(self.root, text="Traveller Details",
font=("Arial", 12))
        traveller_frame.grid(row=0, column=1, padx=10, pady=10,
sticky="nsew")
```

```python
        tk.Label(traveller_frame, text="Traveller Name:", font=("Arial",
12)).grid(row=0, column=0, sticky="e")
        self.traveller_name_entry = tk.Entry(traveller_frame, font=("Arial",
12))
        self.traveller_name_entry.grid(row=0, column=1)

        tk.Label(traveller_frame, text="Address:", font=("Arial",
12)).grid(row=1, column=0, sticky="e")
        self.address_entry = tk.Entry(traveller_frame, font=("Arial", 12))
        self.address_entry.grid(row=1, column=1)

        tk.Label(traveller_frame, text="DOB (YYYY-MM-DD):", font=("Arial",
12)).grid(row=2, column=0, sticky="e")
        self.dob_entry = tk.Entry(traveller_frame, font=("Arial", 12))
        self.dob_entry.grid(row=2, column=1)

        tk.Label(traveller_frame, text="Emergency Contact:", font=("Arial",
12)).grid(row=3, column=0, sticky="e")
        self.emergency_contact_entry = tk.Entry(traveller_frame,
font=("Arial", 12))
        self.emergency_contact_entry.grid(row=3, column=1)

        tk.Label(traveller_frame, text="ID No:", font=("Arial",
12)).grid(row=4, column=0, sticky="e")
        self.gov_id_entry = tk.Entry(traveller_frame, font=("Arial", 12))
        self.gov_id_entry.grid(row=4, column=1)

    def create_buttons(self):
        button_frame = tk.Frame(self.root)
        button_frame.grid(row=1, column=0, columnspan=2, padx=10, pady=10,
sticky="nsew")

        self.create_trip_button = tk.Button(button_frame, text="Create Trip",
command=self.create_trip, font=("Arial", 12))
        self.create_trip_button.grid(row=0, column=0, padx=5)

        self.create_traveller_button = tk.Button(button_frame, text="Create
Traveller", command=self.create_traveller,
                                                 font=("Arial", 12))
        self.create_traveller_button.grid(row=0, column=1, padx=5)

        self.delete_button = tk.Button(button_frame, text="Delete",
command=self.delete_selected_item, font=("Arial", 12))
        self.delete_button.grid(row=0, column=2, padx=5)

        self.edit_button = tk.Button(button_frame, text="Edit",
command=self.edit_selected_item, font=("Arial", 12))
        self.edit_button.grid(row=0, column=3, padx=5)

        self.close_button = tk.Button(button_frame, text="Close",
command=self.root.quit, font=("Arial", 12))
        self.close_button.grid(row=0, column=4, padx=5)

    def create_listboxes(self):
        listbox_frame = tk.Frame(self.root)
        listbox_frame.grid(row=2, column=0, columnspan=2, padx=10, pady=10,
sticky="nsew")

        tk.Label(listbox_frame, text="Trips", font=("Arial", 12)).grid(row=0,
column=0)
        self.trip_listbox = tk.Listbox(listbox_frame, width=60, height=10,
font=("Arial", 12))
        self.trip_listbox.grid(row=1, column=0, padx=5, pady=5)

        tk.Label(listbox_frame, text="Travellers", font=("Arial",
12)).grid(row=0, column=1)
        self.traveller_listbox = tk.Listbox(listbox_frame, width=60,
height=10, font=("Arial", 12))
        self.traveller_listbox.grid(row=1, column=1, padx=5, pady=5)

    def create_trip(self):
        trip_name = self.trip_name_entry.get()
        start_date = self.start_date_entry.get()
        duration = self.duration_entry.get()
        contact_info = self.contact_info_entry.get()
        coordinator = self.coordinator_entry.get()

        if not self.validate_trip_fields(duration, contact_info, start_date):
            return

        trip_details = f"Trip Name: {trip_name}, Start Date: {start_date},
Duration: {duration}, " \
                       f"Contact Info: {contact_info}, Coordinator:
{coordinator}"
        self.trip_listbox.insert(tk.END, trip_details)

        with open('trips.csv', 'a', newline='') as csvfile:
            writer = csv.writer(csvfile)
            writer.writerow([trip_name, start_date, duration, contact_info,
coordinator])

    def create_traveller(self):
        traveller_name = self.traveller_name_entry.get()
        address = self.address_entry.get()
        dob = self.dob_entry.get()
        emergency_contact = self.emergency_contact_entry.get()
        gov_id = self.gov_id_entry.get()

        if not self.validate_traveller_fields(dob, emergency_contact,
gov_id):
            return

        traveller_details = f"Name: {traveller_name}, Address: {address},
DOB: {dob}, " \
                            f"Emergency Contact: {emergency_contact}, Gov ID:
{gov_id}"
        self.traveller_listbox.insert(tk.END, traveller_details)

        traveller_data = {
            'Name': traveller_name,
            'Address': address,
            'DOB': dob,
            'Emergency Contact': emergency_contact,
            'Gov ID': gov_id
        }
        with open('travellers.json', 'a') as jsonfile:
            json.dump(traveller_data, jsonfile, indent=4)

    def validate_trip_fields(self, duration, contact_info, start_date):
        try:
            int(duration)
        except ValueError:
            messagebox.showerror("Error", "Duration must be a number.")
            return False

        if len(contact_info)!= 10 or not contact_info.isdigit():
            messagebox.showerror("Error", "Contact Info must be a 10-digit
number.")
            return False

        try:
            datetime.datetime.strptime(start_date, '%Y-%m-%d')
        except ValueError:
            messagebox.showerror("Error", "Date of Start must be a valid
datetime.")
            return False

        return True

    def validate_traveller_fields(self, dob, emergency_contact, gov_id):
        try:
            datetime.datetime.strptime(dob, '%Y-%m-%d')
        except ValueError:
            messagebox.showerror("Error", "DOB must be a valid datetime.")
            return False

        if len(emergency_contact)!= 10 or not emergency_contact.isdigit():
            messagebox.showerror("Error", "Emergency Contact must be a 10-
digit number.")
            return False

        if not (len(gov_id) == 10 or len(gov_id) == 12) or not
gov_id.isdigit():
            messagebox.showerror("Error", "ID No must be a 10 or 12-digit
string.")
            return False

        return True

    def delete_selected_item(self):
        selected_index = self.trip_listbox.curselection()
        if selected_index:
            self.trip_listbox.delete(selected_index)
        else:
            messagebox.showinfo("Information", "Please select an item to
delete.")

    def edit_selected_item(self):
        self.clear_entries()
        messagebox.showinfo("Information", "All entries have been cleared.")
```

```python
    def clear_entries(self):
        self.trip_name_entry.delete(0, tk.END)
        self.start_date_entry.delete(0, tk.END)
        self.duration_entry.delete(0, tk.END)
        self.contact_info_entry.delete(0, tk.END)
        self.coordinator_entry.delete(0, tk.END)
        self.traveller_name_entry.delete(0, tk.END)
        self.address_entry.delete(0, tk.END)
        self.dob_entry.delete(0, tk.END)
        self.emergency_contact_entry.delete(0, tk.END)
        self.gov_id_entry.delete(0, tk.END)

if __name__ == "__main__":
    root = tk.Tk()
    app = TripManagementApp(root)
    root.mainloop()
```

Output



```
Trip Name: colombo
Start Date (YYYY-MM-DD): 2024-10-02
Duration (days): 3
Contact Info: 0753637389
Coordinator: ruk

Traveller Name: zhana
Address: main street
DOB (YYYY-MM-DD): 1999-09-09
Emergency Contact: 0651212134
ID No: 111111234567

Trip Name: colombo, Start Date: 2024-10-02, Duration: 3, Contact Info: 07536:
Name: zhana, Address: main street, DOB: 1999-09-09, Emergency Contact: 0
```

explain how it works

import libraries

import tkinter as tk

import csv

import datetime

import json

- tkinter as tk: Imports the tkinter library and assigns it the alias tk for easier use. This library is used to create the graphical user interface (GUI) of the application.
- csv: Imports the csv library for working with Comma-Separated Values files. This will be used to store trip details.
- datetime: Imports the datetime library for handling date and time functions. This is used to get the current date and time for reference.
- json: Imports the json library for working with JSON (JavaScript Object Notation) files. This will be used to store traveller details.

TripManagementApp Class:

```
class TripManagementApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Trip Management System")

# Get the current date and time
        current_datetime = datetime.datetime.now()

        # ... (rest of the code)
```

……………………………………………………………………………..

- This defines a class called TripManagementApp. This class will hold the functionalities for creating the application's interface and handling user interactions.
- The __init__ method (also known as the constructor) is called whenever a new instance of the TripManagementApp class is created. It takes the main window (root) as an argument and assigns it to the self.root variable for later use.
- It then sets the title of the window to "Trip Management System".
- Finally, it retrieves the current date and time using datetime.datetime.now() and stores it in the current_datetime variable (not used in the provided code snippet).

Creating GUI Elements:

```
# Trip fields
        tk.Label(root, text="Trip Name:").grid(row=0, column=0, sticky="e")
        self.trip_name_entry = tk.Entry(root)
        self.trip_name_entry.grid(row=0, column=1)

…………………………………………………………………….
```

- This section creates various GUI elements using the tkinter library.
- It uses tk.Label to create labels for each trip and traveller field (name, start date, etc.). The grid method is used to position these labels on the window.
- tk.Entry is used to create entry fields where users can input trip and traveller details. These are stored in variables like self.trip_name_entry for later access.
- Similarly, tk.Button creates buttons for "Create Trip" and "Create Traveller" actions. These buttons are linked to the self.create_trip and self.create_traveller methods using the command attribute.
- Finally, tk.Listbox creates two listboxes, one for displaying trip details and another for displaying traveller details.

Conclusion

The Travel Management System project wasn't just about developing software; it served as a pivotal moment in your journey as a software developer. Throughout this endeavor, you acquired practical skills in Object-Oriented Programming (OOP), honed your abilities in UI design, delved into the intricacies of data persistence, and navigated the complexities of project management. Moreover, it underscored the paramount importance of meticulous planning, effective version control, rigorous testing, and comprehensive documentation.

Now equipped with this solid foundation, you stand poised to delve deeper into the realm of software development. You have the opportunity to explore advanced OOP concepts, delve into richer UI frameworks, and immerse yourself in robust database systems. The world of software development beckons with boundless opportunities for growth, innovation, and the creation of even more remarkable applications. Embrace this journey with enthusiasm, for it is a journey marked by continuous learning, creativity, and the realization of your full potential.

References

Rutvik Baban, K., Mehra, P., Sanjay and Manoj, J. (n.d.). *TOURS AND TRAVEL MANAGEMENT SYSTEM*. [online] *International Research Journal of Modernization in Engineering Technology and Science*, Peer-Reviewed, Open Access, pp.2582–5208. Available at: https://www.irjmets.com/uploadedfiles/paper/issue_3_march_2022/20436/final/fin_irjmets1652809292.pdf.

Wang, S. and Chen, L. (2015). Application of Travel Management System Based on Route Inquiry. *International Journal of Smart Home*, 9(6), pp.133–140. https://doi.org/10.14257/ijsh.2015.9.6.15.

Rohitha, W.A. (2011). Travel management system. *dl.lib.uom.lk*. [online] Available at: http://dl.lib.uom.lk/handle/123/1787

Arminen, I. and Poikus, P. (2009). Diagnostic Reasoning in the Use of Travel Management System. *Computer Supported Cooperative Work (CSCW)*, 18(2-3), pp.251–276. https://doi.org/10.1007/s10606-008-9086-3

Shuo, Z. (2012). *Design and Implementation of a WEB-based Tourism Information Management System: Travel-SYS*. [online] *www.diva-portal.org*. Available at: https://www.diva-portal.org/smash/record.jsf?pid=diva2:490645