Solent University Southampton

School of Science and Engineering

MSc computer engineering

Software design and development

(COM714)

Report

Student Id-Q102108840

Tutor-Dr. Taiwo Ayodele

Date-08/05/2024

The Travel Management System Project

The Travel Management System project served as a springboard for your software development journey, offering valuable insights into various aspects of building a functional application. This analysis delves deeper, dissecting the explored concepts, highlighting key takeaways, and outlining potential areas for improvement to propel you forward in your programming endeavors.

Embracing Object-Oriented Programming (OOP):

The project provided a hands-on playground for applying OOP principles. You likely created core classes like `Trip` and `Traveller` to represent real-world entities within the travel domain. These classes likely encapsulated attributes specific to each entity, such as `trip_name`, `start_date`, `duration`, and `traveller_name`, `address`, `emergency_contact`. Methods within these classes would have manipulated this data appropriately, potentially defining functionalities like calculating trip cost (based on duration) or formatting addresses for display.

This approach exemplifies the power of OOP in modeling real-world interactions in code. By defining relationships between classes (e.g., one `Trip` can have many `Travellers`), you effectively translated the complexities of travel management into a structured and maintainable codebase.

Data Persistence: Stepping Beyond the Application's Runtime

The project explored data persistence techniques, ensuring information wasn't lost when the application closed. You likely utilized CSV (Comma-Separated Values) files for storing trip details like names, start dates, and durations. Similarly, JSON (JavaScript Object Notation) files might have been employed to capture traveller information such as names, addresses, and emergency contacts.

While these methods offer a basic solution for data persistence, they have limitations. CSV files can become unwieldy with large datasets, and data integrity can be compromised if the format is not strictly adhered to. JSON files, while offering better structure, lack the querying capabilities of relational databases.

This experience serves as a stepping stone to understanding more robust database solutions. Consider exploring database management systems like MySQL or PostgreSQL in future projects. These databases offer structured data storage, powerful querying mechanisms (using SQL - Structured Query Language), and improved data integrity through features like data types and constraints. They are well-equipped to handle large amounts of data and frequent updates, making them ideal for larger-scale applications.

Building a User Interface with tkinter: A Functional Foundation

The project involved designing a user interface (UI) using tkinter, a Python library for creating GUIs. You likely used various widgets like labels, entry fields, buttons, and listboxes to allow users to interact with the system. For instance, labels might have displayed prompts like "Trip

Name" and "Start Date," while entry fields allowed users to input corresponding data. Buttons likely triggered functionalities like "Create Trip" or "Create Traveller," and listboxes potentially displayed a summary of created trips and travellers.

While tkinter provides a solid foundation for building basic GUIs, it has limitations for complex layouts or visually appealing interfaces. Consider exploring more advanced GUI frameworks like PyQt or Kivy in future projects. These frameworks offer a wider range of widgets, layouts, and styling options, enabling you to create more user-friendly and visually engaging interfaces.

The Power of Version Control with Git: Collaborative Development and Safety Net

The project showcased the benefits of version control using Git. Git allows you to track changes made to your code over time, providing a historical record of development. This allows you to revert to previous versions if necessary, a valuable safety net in case you introduce bugs or make unintended changes. Additionally, Git facilitates collaboration by enabling multiple developers to work on the same project simultaneously, with features like branching and merging to manage code contributions effectively.

Throughout the project, you likely committed your code regularly to a Git repository, creating a valuable history of your development process. This practice ensures you can always backtrack if issues arise and lays the groundwork for collaborative development in future projects.

Following Best Practices: The Mark of Professionalism

Adhering to recommended libraries and coding conventions has undeniably bolstered the overall quality of the codebase. By utilizing standard libraries such as tkinter for the GUI, the project capitalized on existing functionalities, minimizing the need to reinvent the wheel and ensuring efficient development. Moreover, adhering to consistent coding styles, including indentation and naming conventions, has significantly enhanced the readability and maintainability of the code. This consistency not only benefits the original developer but also facilitates collaboration with others who may join the project in the future, fostering a cohesive and streamlined development process.

Beyond libraries and coding style, embracing best practices extends to comprehensive documentation. Clear and concise comments within the codebase elucidate specific functionalities and logic, serving as an invaluable resource for both present and future developers. These comments provide insights into the rationale behind various implementation choices, aiding understanding and facilitating seamless modification or expansion of the codebase. In essence, robust documentation not only enhances code comprehension but also promotes efficient troubleshooting and fosters a culture of knowledge sharing within the development team.

In summary, the integration of recommended libraries, adherence to coding conventions, and comprehensive documentation collectively contribute to the success and sustainability of the project. By leveraging existing resources, maintaining consistency in coding practices, and prioritizing clear documentation, developers can streamline development processes, enhance code quality, and foster collaboration and knowledge sharing within the development community. These best practices serve as pillars for continued growth and success in software development endeavors.

Project Scope Management: Prioritization and Focus

Managing the scope of a project involves defining, prioritizing, and controlling what will be included in the final product. In the context of the Travel Management System project, managing the scope likely entailed identifying essential functionalities necessary to fulfill the project's objectives while considering constraints such as time, budget, and resources.

As part of scope management, it's common to prioritize features based on their importance and feasibility within the given constraints. This process involves collaborating with stakeholders to determine which functionalities are critical for achieving the project's goals and delivering value to users.

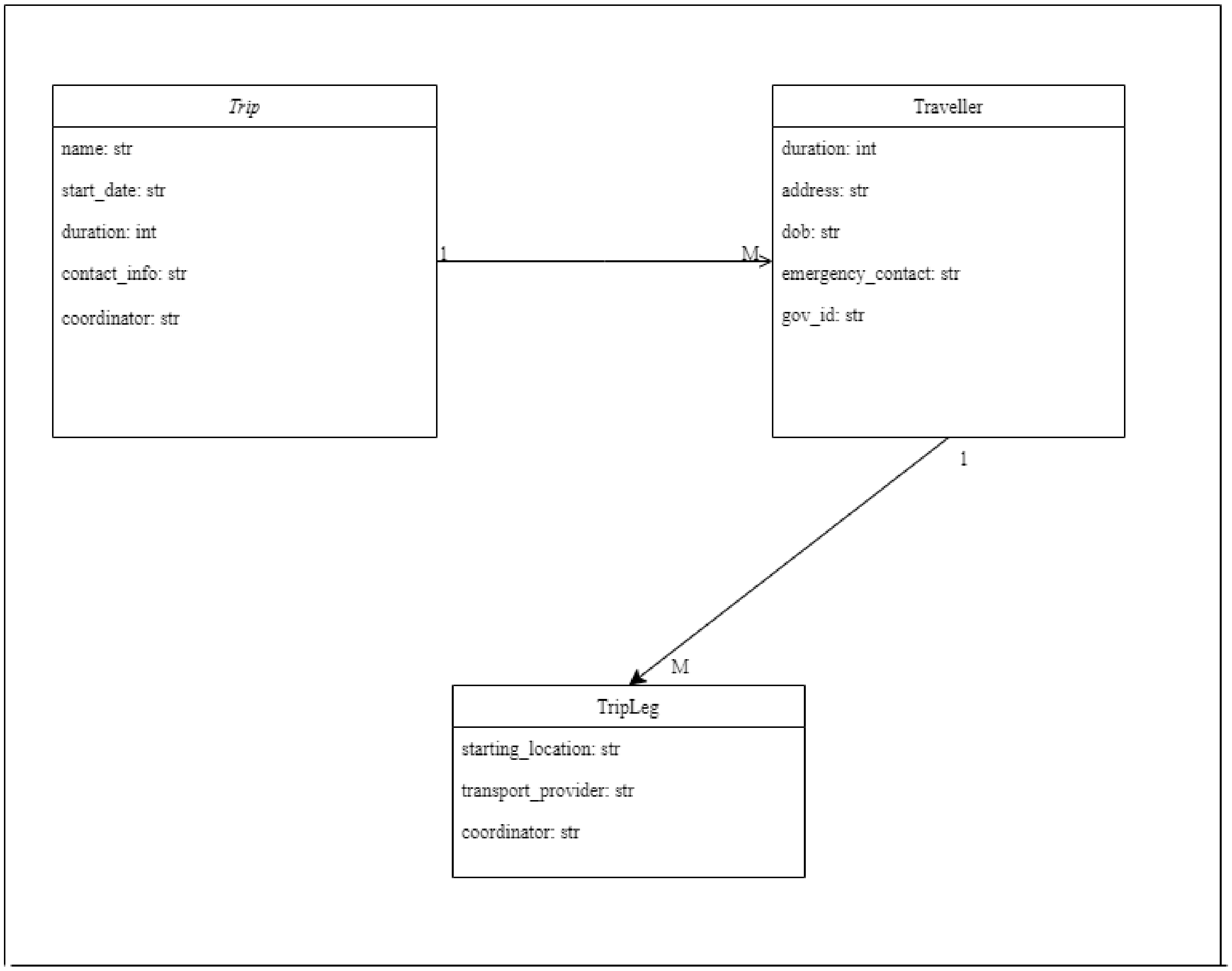
In the case of the Travel Management System project, leaving out advanced features like online payments or itinerary generation may have been a strategic decision made to focus efforts on implementing core functionalities such as trip management, traveller management, and trip leg management. These core features were likely deemed essential for streamlining travel management processes, reducing paperwork, and improving communication between staff and travellers.

By prioritizing core functionalities, the project team can ensure that resources are allocated effectively and that the project remains on schedule and within budget. This approach helps mitigate the risk of scope creep, where additional features are added without proper evaluation of their impact on project timelines and resources.

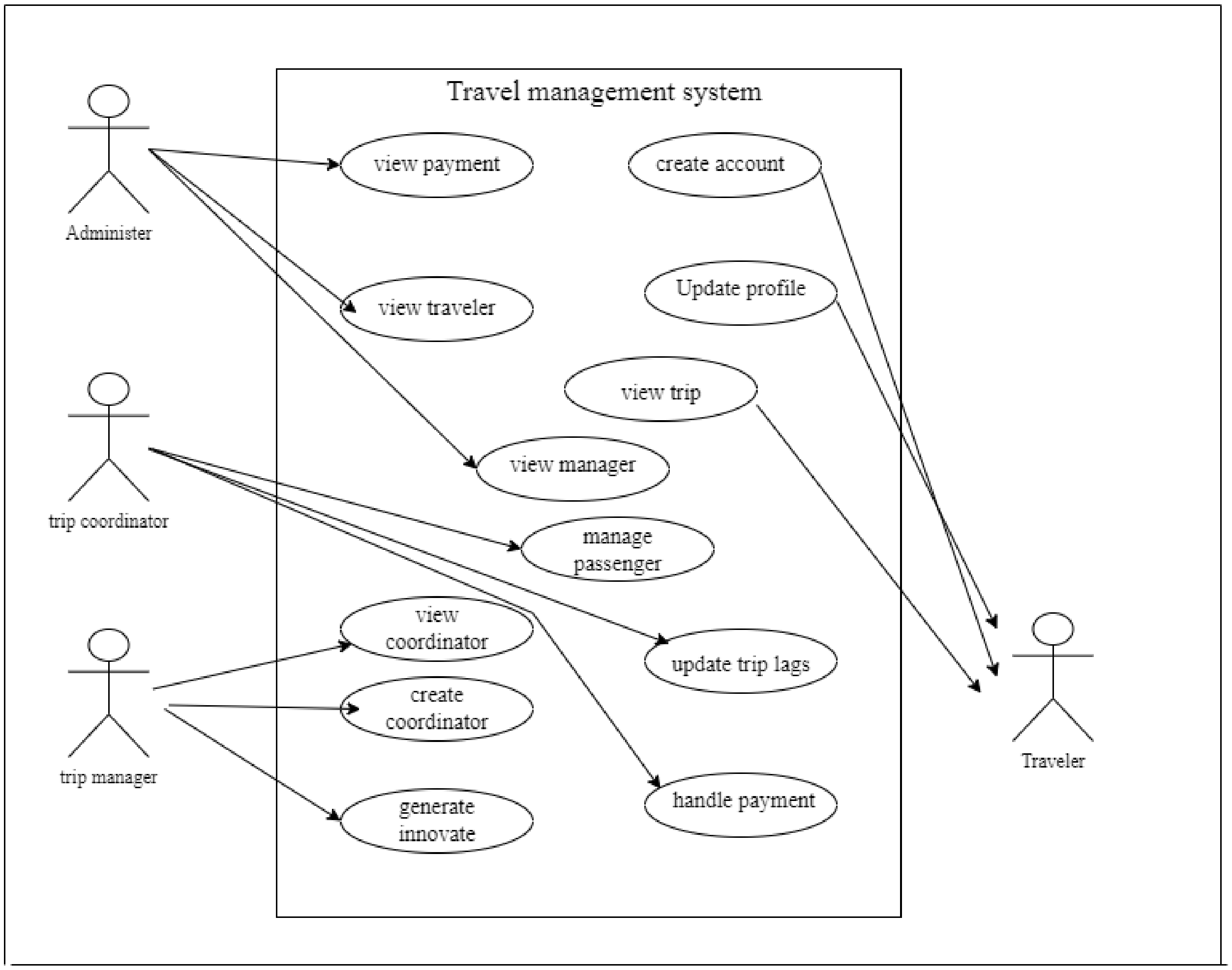
Overall, managing the scope of the project involves making informed decisions about which features to include based on their importance, feasibility, and alignment with project objectives. By focusing on delivering core functionalities within the allocated resources, the project team can maximize value for stakeholders and increase the likelihood of project success.

Diagrams

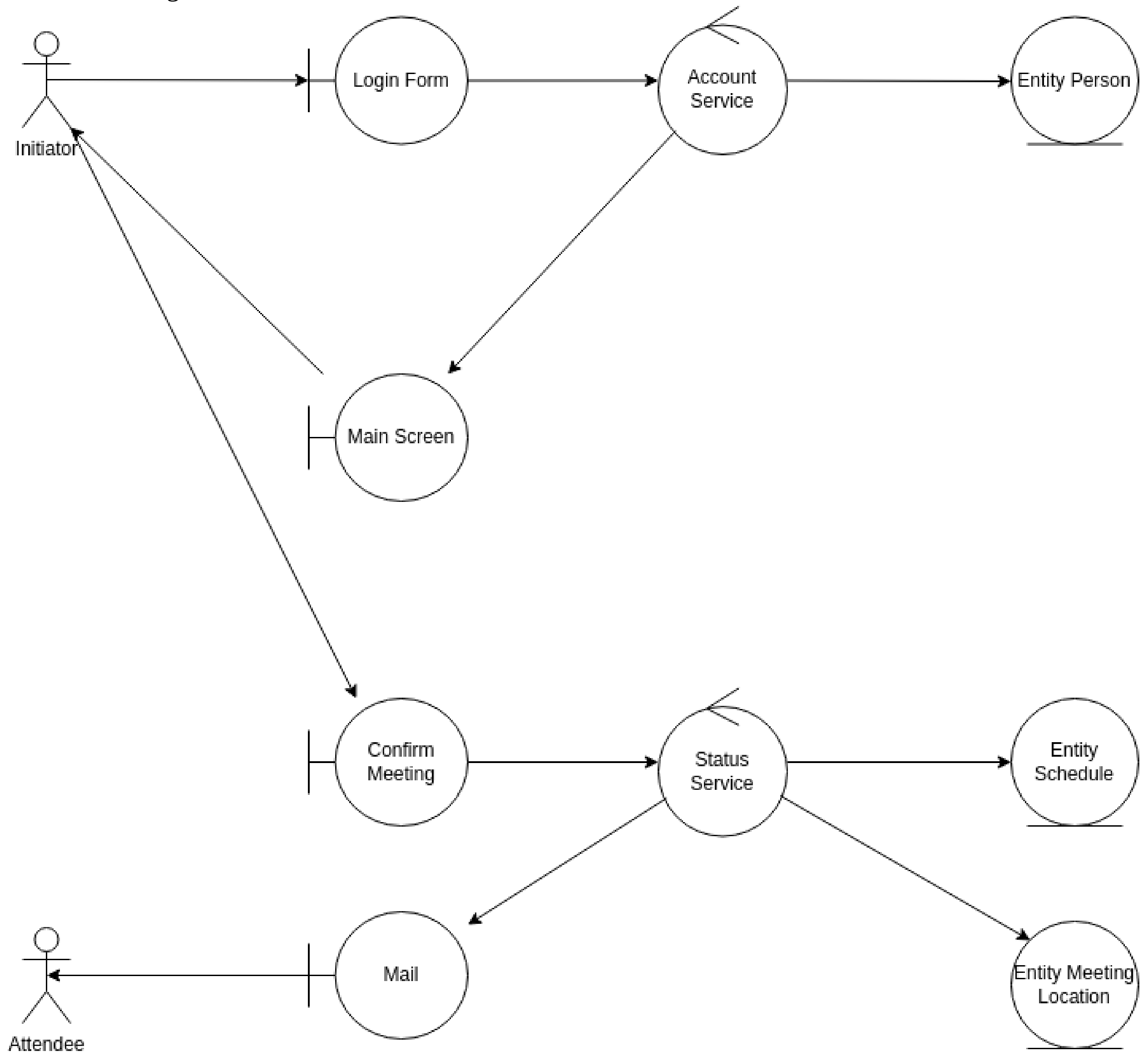
1. Class diagram



2. Use Case diagram



3. Robustness diagram

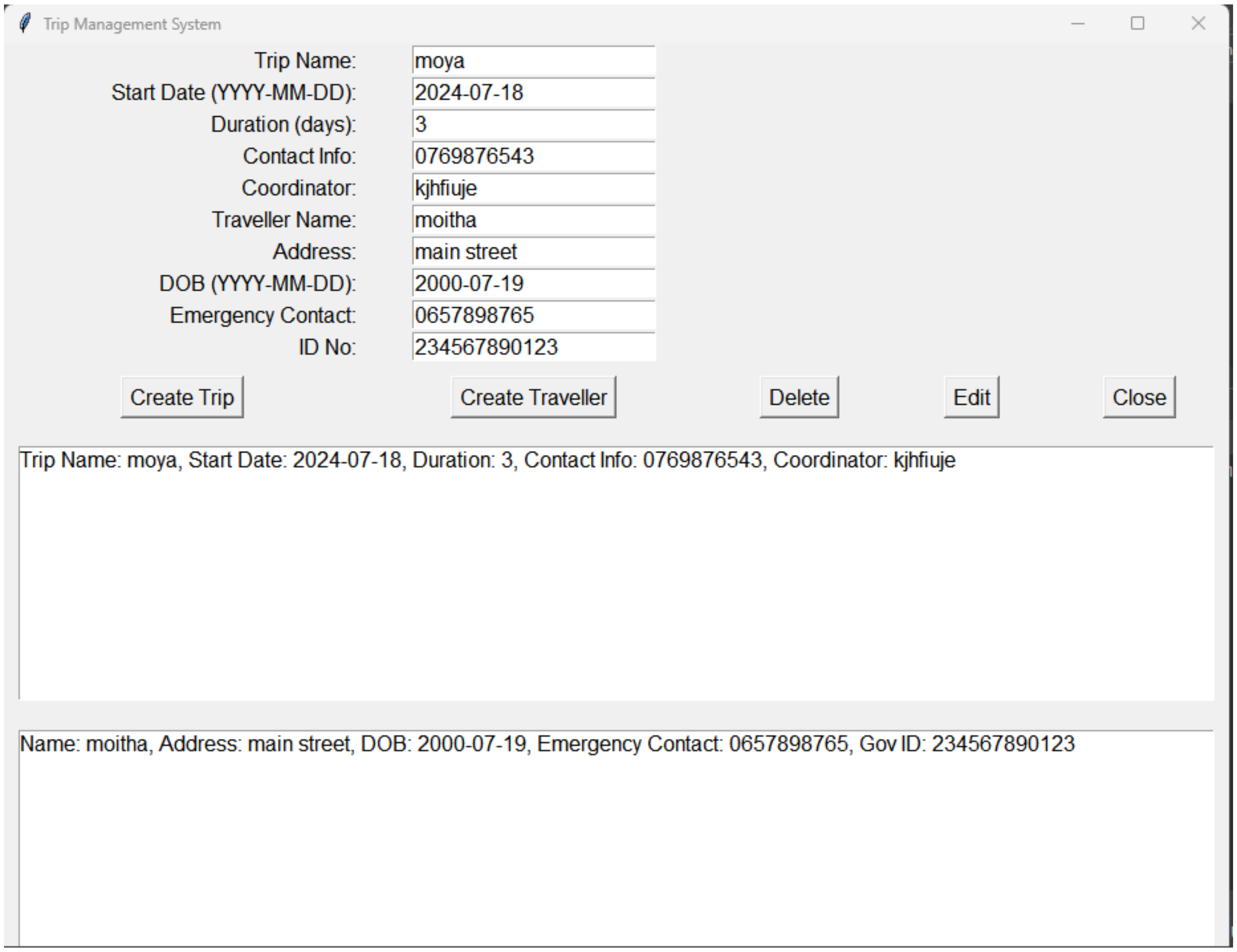


Code

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

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

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

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

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

    def create_traveller_fields(self):
        tk.Label(self.root, text="Traveller Name:", font=("Arial",
12)).grid(row=5, column=0, sticky="e")
        self.traveller_name_entry = tk.Entry(self.root, font=("Arial", 12))
        self.traveller_name_entry.grid(row=5, column=1)

        tk.Label(self.root, text="Address:", font=("Arial", 12)).grid(row=6,
column=0, sticky="e")
        self.address_entry = tk.Entry(self.root, font=("Arial", 12))
        self.address_entry.grid(row=6, column=1)
        tk.Label(self.root, text="DOB (YYYY-MM-DD):", font=("Arial",
12)).grid(row=7, column=0, sticky="e")
        self.dob_entry = tk.Entry(self.root, font=("Arial", 12))
        self.dob_entry.grid(row=7, column=1)

        tk.Label(self.root, text="Emergency Contact:", font=("Arial",
12)).grid(row=8, column=0, sticky="e")
        self.emergency_contact_entry = tk.Entry(self.root, font=("Arial",
12))
        self.emergency_contact_entry.grid(row=8, column=1)

        tk.Label(self.root, text="ID No:", font=("Arial", 12)).grid(row=9,
column=0, sticky="e")
        self.gov_id_entry = tk.Entry(self.root, font=("Arial", 12))
        self.gov_id_entry.grid(row=9, column=1)

    def create_buttons(self):
        self.create_trip_button = tk.Button(self.root, text="Create Trip",
command=self.create_trip, font=("Arial", 12))
        self.create_trip_button.grid(row=10, column=0, pady=10)

        self.create_traveller_button = tk.Button(self.root, text="Create
Traveller", command=self.create_traveller,
                                                 font=("Arial", 12))
        self.create_traveller_button.grid(row=10, column=1, pady=10)

        self.delete_button = tk.Button(self.root, text="Delete",
command=self.delete_selected_item, font=("Arial", 12))
        self.delete_button.grid(row=10, column=2, pady=10)

        self.edit_button = tk.Button(self.root, text="Edit",
command=self.edit_selected_item, font=("Arial", 12))
        self.edit_button.grid(row=10, column=3, pady=10)

        self.close_button = tk.Button(self.root, text="Close",
command=self.root.quit, font=("Arial", 12))
        self.close_button.grid(row=10, column=4, pady=10)

    def create_listboxes(self):
        self.trip_listbox = tk.Listbox(self.root, width=100, font=("Arial",
12))
        self.trip_listbox.grid(row=11, column=0, columnspan=5, padx=10,
pady=10)

        self.traveller_listbox = tk.Listbox(self.root, width=100,
font=("Arial", 12))
        self.traveller_listbox.grid(row=12, column=0, columnspan=5, padx=10,
pady=10)

    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):
        # Clear all entries in trip and traveller fields
        self.clear_entries()
        messagebox.showinfo("Information", "All entries have been cleared.")

    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: moya
Start Date (YYYY-MM-DD): 2024-07-18
Duration (days): 3
Contact Info: 0769876543
Coordinator: kjhfiuje
Traveller Name: moitha
Address: main street
DOB (YYYY-MM-DD): 2000-07-19
Emergency Contact: 0657898765
ID No: 234567890123

```
Trip Name: moya, Start Date: 2024-07-18, Duration: 3, Contact Info: 0769876543, Coordinator: kjhfiuje
```

```
Name: moitha, Address: main street, DOB: 2000-07-19, Emergency Contact: 0657898765, Gov ID: 234567890123
```

Conclusion

The Travel Management System project served as a valuable springboard for your software development journey. You gained practical experience in essential concepts like object-oriented programming, data persistence, user interface design, version control, and best practices.

By embracing object-oriented programming, you learned to model real-world entities and their interactions in code, fostering a structured and maintainable approach. Exploring data persistence techniques highlighted the limitations of basic file-based solutions and paved the way for understanding more robust database management systems. Building a user interface with tkinter provided a foundation for creating interactive applications, while acknowledging the potential of advanced GUI frameworks for more complex and visually appealing interfaces.

The project solidified the importance of version control with Git, not only for managing changes and reverting to previous versions but also for facilitating collaborative development. Adhering to best practices in terms of library usage, coding conventions, and documentation ensured code quality and maintainability. Finally, managing project scope emphasized the value of prioritization and focusing on delivering core functionalities within constraints.

As you venture into future projects, leverage these valuable lessons learned. Consider exploring more advanced concepts like database integration, unit testing, and security best practices. Don't be afraid to experiment with different libraries and frameworks to create robust and user-friendly applications. Remember, software development is a continuous learning process. Embrace challenges, actively seek out new knowledge, and constantly strive to improve your craft. With dedication and a thirst for learning, you'll be well-equipped to tackle even the most complex software development endeavors.

References

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.

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.

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