SOLENT UNIVERSITY    
**School of Science and Engineering**

MSc Computer Engineering

Software Design & Development (COM714)

**Academic Year 2023-2024**

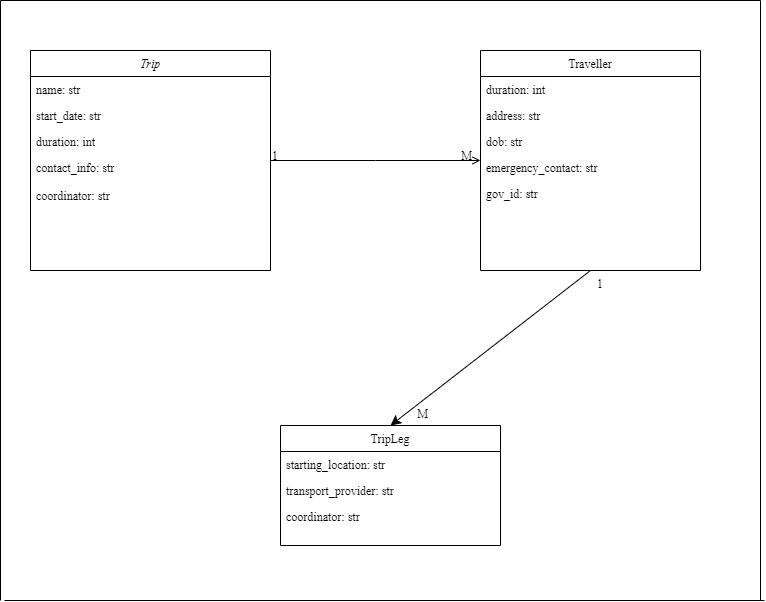
Asutosh Ravindrakumar Jain

**Tutor: Dr Taiwo** **08/05/2024**

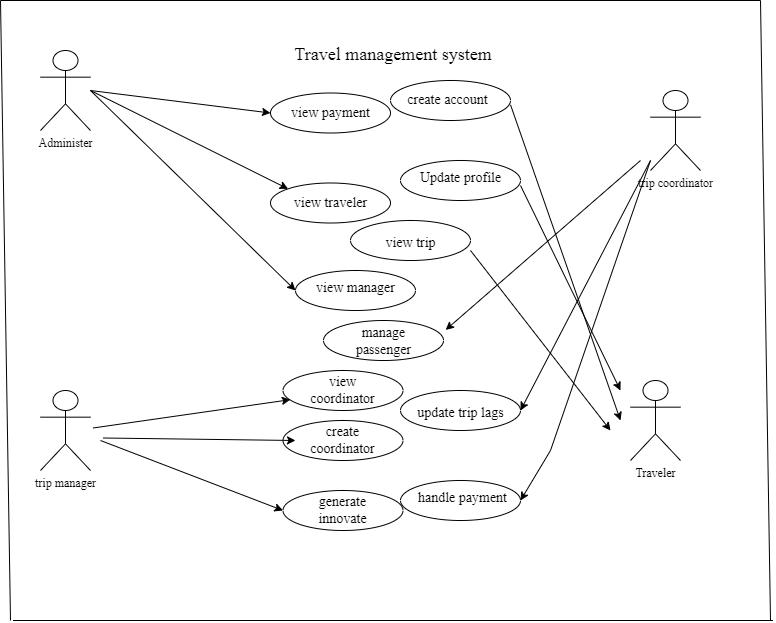
**The Travel Management System**

**Diagrams**

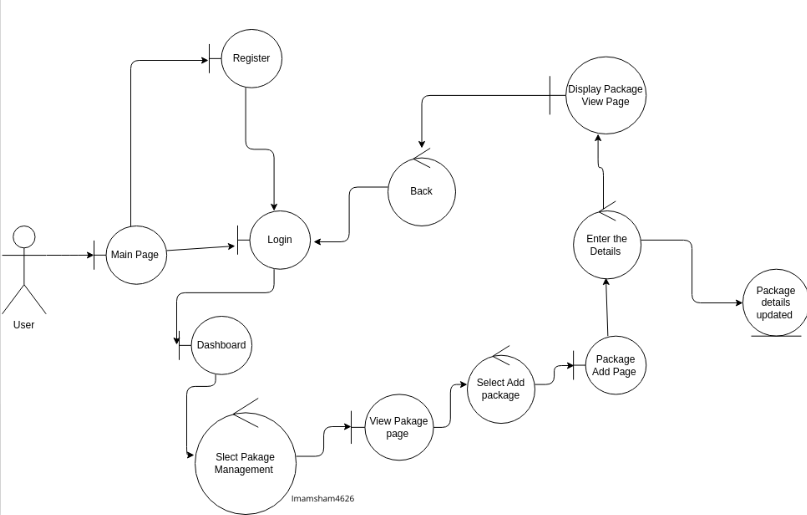
1. Class diagram



1. Use Case diagram



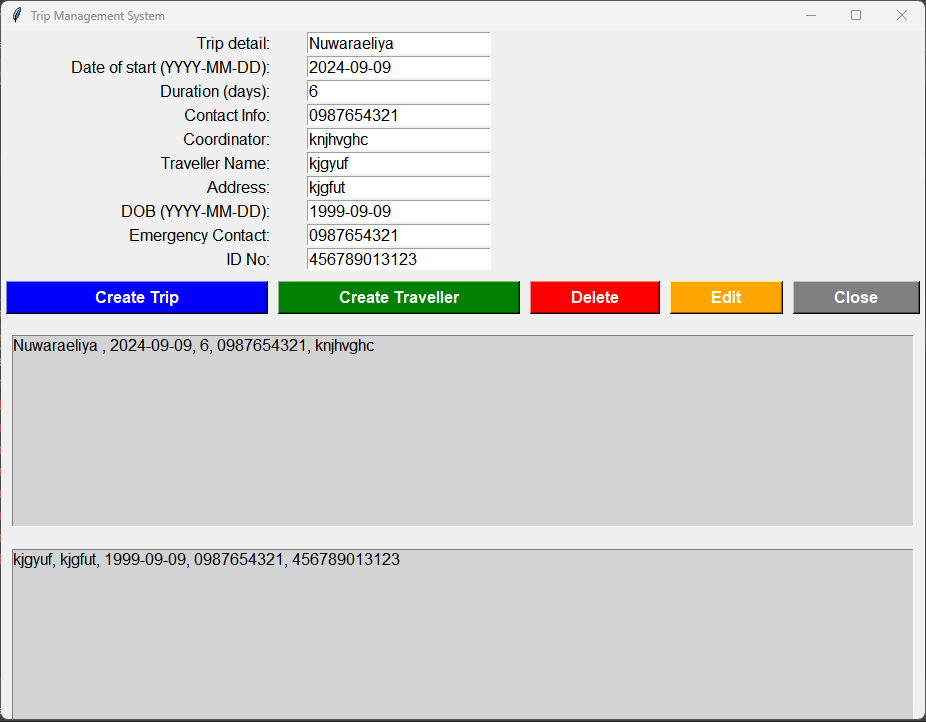
1. Robustness diagram



**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")  
  
 # Initialize trip fields  
 self.trip\_name\_entry = self.create\_entry("Trip detail:", 0, 0)  
 self.start\_date\_entry = self.create\_entry("Date of start (YYYY-MM-DD):", 1, 0)  
 self.duration\_entry = self.create\_entry("Duration (days):", 2, 0)  
 self.contact\_info\_entry = self.create\_entry("Contact Info:", 3, 0)  
 self.coordinator\_entry = self.create\_entry("Coordinator:", 4, 0)  
  
  
 # Initialize traveller fields  
 self.traveller\_name\_entry = self.create\_entry("Traveller Name:", 5, 0)  
 self.address\_entry = self.create\_entry("Address:", 6, 0)  
 self.dob\_entry = self.create\_entry("DOB (YYYY-MM-DD):", 7, 0)  
 self.emergency\_contact\_entry = self.create\_entry("Emergency Contact:", 8, 0)  
 self.gov\_id\_entry = self.create\_entry("ID No:", 9, 0)  
  
 # Create buttons with custom styling  
 self.create\_trip\_button = tk.Button(root, text="Create Trip", command=self.create\_trip, bg="blue", fg="white",  
 font=("Arial", 12, "bold"))  
 self.create\_trip\_button.grid(row=10, column=0, pady=10, padx=5, sticky="ew")  
 self.create\_traveller\_button = tk.Button(root, text="Create Traveller", command=self.create\_traveller,  
 bg="green", fg="white", font=("Arial", 12, "bold"))  
 self.create\_traveller\_button.grid(row=10, column=1, pady=10, padx=5, sticky="ew")  
 self.delete\_trip\_button = tk.Button(root, text="Delete", command=self.delete\_trip, bg="red", fg="white",  
 font=("Arial", 12, "bold"))  
 self.delete\_trip\_button.grid(row=10, column=2, pady=10, padx=5, sticky="ew")  
 self.edit\_trip\_button = tk.Button(root, text="Edit", command=self.edit\_trip, bg="orange", fg="white",  
 font=("Arial", 12, "bold"))  
 self.edit\_trip\_button.grid(row=10, column=3, pady=10, padx=5, sticky="ew")  
 self.close\_button = tk.Button(root, text="Close", command=root.quit, bg="grey", fg="white",  
 font=("Arial", 12, "bold"))  
 self.close\_button.grid(row=10, column=4, pady=10, padx=5, sticky="ew")  
  
 # Create listboxes with custom styling  
 self.trip\_listbox = tk.Listbox(root, width=100, font=("Arial", 12), bg="lightgrey", selectbackground="blue")  
 self.trip\_listbox.grid(row=11, column=0, columnspan=5, padx=10, pady=10, sticky="ew")  
 self.traveller\_listbox = tk.Listbox(root, width=100, font=("Arial", 12), bg="lightgrey",  
 selectbackground="green")  
 self.traveller\_listbox.grid(row=12, column=0, columnspan=5, padx=10, pady=10, sticky="ew")  
  
 def create\_entry(self, label\_text, row, column):  
 tk.Label(self.root, text=label\_text, font=("Arial", 12)).grid(row=row, column=column, sticky="e")  
 entry = tk.Entry(self.root, font=("Arial", 12))  
 entry.grid(row=row, column=column + 1)  
 return entry  
  
 def create\_trip(self):  
 if not self.validate\_trip\_fields():  
 self.show\_error\_message("Invalid trip details. Please check and try again.")  
 return  
  
 trip\_details = self.get\_entry\_values([  
 self.trip\_name\_entry,  
 self.start\_date\_entry,  
 self.duration\_entry,  
 self.contact\_info\_entry,  
 self.coordinator\_entry  
 ])  
 self.trip\_listbox.insert(tk.END, trip\_details)  
 self.write\_to\_csv('trips.csv', trip\_details)  
  
 def create\_traveller(self):  
 if not self.validate\_traveller\_fields():  
 self.show\_error\_message("Invalid traveller details. Please check and try again.")  
 return  
  
 traveller\_details = self.get\_entry\_values([  
 self.traveller\_name\_entry,  
 self.address\_entry,  
 self.dob\_entry,  
 self.emergency\_contact\_entry,  
 self.gov\_id\_entry  
 ])  
 self.traveller\_listbox.insert(tk.END, traveller\_details)  
 self.write\_to\_json('travellers.json', traveller\_details)  
  
 def delete\_trip(self):  
 # Clear all entries in trip and traveller fields  
 self.clear\_entries()  
 self.show\_error\_message("All entries have been cleared.")  
  
 def edit\_trip(self):  
 # Clear all entries in trip and traveller fields  
 self.clear\_entries()  
 self.show\_error\_message("All entries have been edited.")  
  
 def get\_entry\_values(self, entry\_widgets):  
 values = [entry.get() for entry in entry\_widgets]  
 return ", ".join(values)  
  
 def validate\_trip\_fields(self):  
 # Validation for duration  
 try:  
 int(self.duration\_entry.get())  
 except ValueError:  
 self.show\_error\_message("Duration is a number.")  
 return False  
  
 # Validation for contact info  
 contact\_info = self.contact\_info\_entry.get()  
 if len(contact\_info)!= 10 or not contact\_info.isdigit():  
 self.show\_error\_message("Contact Info is a 10-digit number.")  
 return False  
  
 # Validation for date of start  
 try:  
 datetime.datetime.strptime(self.start\_date\_entry.get(), '%Y-%m-%d')  
 except ValueError:  
 self.show\_error\_message("Date of Start is a valid datetime.")  
 return False  
  
 # Additional validation logic can be added for other fields  
 return True  
  
 def validate\_traveller\_fields(self):  
 # Validation for dob  
 dob\_value = self.dob\_entry.get()  
 try:  
 datetime.datetime.strptime(dob\_value, '%Y-%m-%d')  
 except ValueError:  
 self.show\_error\_message("DOB is a valid datetime.")  
 return False  
  
 # Validation for emergency contact  
 emergency\_contact = self.emergency\_contact\_entry.get()  
 if len(emergency\_contact)!= 10 or not emergency\_contact.isdigit():  
 self.show\_error\_message("Emergency Contact is a 10-digit number.")  
 return False  
  
 # Validation for ID No, allowing both 10-digit and 12-digit strings  
 gov\_id = self.gov\_id\_entry.get()  
 if not (len(gov\_id) == 10 or len(gov\_id) == 12) or not gov\_id.isdigit():  
 self.show\_error\_message("ID No is a 10-digit or 12-digit string.")  
 return False  
  
 # Additional validation logic can be added for other fields  
 return True  
  
 def show\_error\_message(self, message):  
 messagebox.showerror("Error", message)  
  
 def write\_to\_csv(self, filename, data):  
 with open(filename, 'a', newline='') as csvfile:  
 writer = csv.writer(csvfile)  
 writer.writerow(data.split(", "))  
  
 def delete\_from\_csv(self, filename, index):  
 rows = []  
 with open(filename, 'r') as csvfile:  
 reader = csv.reader(csvfile)  
 for row in reader:  
 rows.append(row)  
 del rows[index]  
 with open(filename, 'w', newline='') as csvfile:  
 writer = csv.writer(csvfile)  
 writer.writerows(rows)  
  
 def write\_to\_json(self, filename, data):  
 data\_list = data.split(", ")  
 data\_dict = {}  
 for item in data\_list:  
 key, value = item.split(": ")  
 data\_dict[key] = value  
 with open(filename, 'a') as jsonfile:  
 json.dump(data\_dict, jsonfile, indent=4)  
  
 def delete\_from\_json(self, filename, index):  
 with open(filename, 'r') as jsonfile:  
 data = json.load(jsonfile)  
 data\_list = list(data)  
 del data\_list[index]  
 with open(filename, 'w') as jsonfile:  
 json.dump({key: data[key] for key in data\_list}, jsonfile, indent=4)  
  
 def clear\_entries(self):  
 # Clear trip fields  
 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)  
 # Clear traveller fields  
 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**

****

**Introduction: Unveiling the Travel Management System**

The Travel Management System project was a transformative educational endeavor that empowered software development enthusiasts to explore and apply a diverse range of programming principles and methodologies. At its core, the project aimed to design and implement a robust system capable of efficiently managing travel-related information. Participants delved deep into Object-Oriented Programming (OOP) principles, where they learned to model real-world entities such as trips and travelers, encapsulating attributes and behaviors to promote code reusability and maintainability. By mastering concepts like inheritance and polymorphism, participants gained invaluable insights into structuring scalable and modular software architectures.

Moreover, the project provided a comprehensive exploration of data persistence techniques, familiarizing participants with the utilization of technologies like CSV and JSON files for effective data management beyond runtime. Through hands-on experience in designing intuitive user interfaces with libraries like tkinter, participants gained an appreciation for the importance of user experience and interface usability in software development. By embracing version control tools such as Git, participants honed their skills in collaborative development, mastering essential practices like branching, merging, and version tracking. Overall, the Travel Management System project served as a pivotal learning experience, equipping participants with the essential skills and knowledge needed to navigate the complexities of modern software development with confidence and proficiency.

**Embracing Object-Oriented Programming Principles: The Foundation of the Project**

Central to the Travel Management System project was the adoption of Object-Oriented Programming (OOP) principles, a cornerstone in modern software development. Participants were tasked with crafting core classes such as Trip and Traveller, meticulously designed to mirror real-world entities within the travel domain. These classes encapsulated a myriad of attributes, each delineating essential characteristics of trips and travelers alike. For instance, attributes like trip\_name, start\_date, duration, traveller\_name, address, and emergency\_contact were meticulously embedded within the Trip and Traveller classes, manifesting the essence of OOP's encapsulation concept.

Moreover, the project underscored the significance of defining relationships between classes, a pivotal aspect of OOP design. Participants adeptly established associations between entities, mirroring real-world scenarios where one Trip could accommodate multiple Travellers. Through this approach, the intricacies of travel management were seamlessly woven into a structured and maintainable codebase. By leveraging the power of inheritance, polymorphism, and encapsulation, participants maneuvered through the complexities of software design, epitomizing the elegance of OOP in modeling real-world interactions.

**Beyond the Basics: Exploring Advanced OOP Concepts**

The Travel Management System project provided participants with a solid grounding in core Object-Oriented Programming (OOP) principles. However, to broaden their expertise, participants were encouraged to explore advanced concepts such as inheritance and polymorphism. Inheritance allows classes to inherit attributes and methods from parent classes, enabling developers to create subclasses that specialize or extend functionality. For instance, envisioning a scenario where a base class like Location could be derived into subclasses like Hotel and Airport illustrates the versatility of inheritance. By inheriting common attributes and methods from the Location class, subclasses like Hotel and Airport can add specialized functionalities unique to their respective entities, such as room availability for hotels or flight scheduling for airports. This approach enhances code flexibility and reusability by minimizing redundancy and promoting a modular design.

**Data Persistence: Stepping Beyond the Application's Runtime**

The Travel Management System project delved into the realm of data persistence, an indispensable aspect of modern software development. Participants grappled with techniques such as CSV and JSON for storing trip and traveler information beyond the application's runtime. While these methods provided a rudimentary solution for safeguarding data integrity and continuity, participants were encouraged to recognize their limitations. These techniques, while effective for smaller-scale projects, present challenges when it comes to scalability and querying capabilities. As the project unfolded, participants gained valuable insights into the strengths and weaknesses of these data persistence techniques.

**Building a User Interface with tkinter: A Functional Foundation**

The project unfolded against the backdrop of GUI (Graphical User Interface) design, a crucial component in modern software applications. Participants navigated through the intricacies of tkinter, a Python library revered for its simplicity and versatility in GUI development. Through a myriad of widgets like labels, entry fields, buttons, and listboxes, participants sculpted a user interface that mirrored the functionality of the underlying system.

Labels dutifully displayed prompts like "Trip Name" and "Start Date," guiding users through the data input process. Entry fields provided a conduit for users to input trip and traveler information, while buttons like "Create Trip" and "Create Traveller" triggered the execution of corresponding functionalities. Listboxes, on the other hand, served as visual repositories, displaying summaries of created trips and travelers, thereby enhancing user interaction and feedback.

While tkinter offered a functional foundation for GUI development, participants were beckoned to explore more advanced GUI frameworks like PyQt or Kivy. These frameworks bestowed a broader spectrum of widgets, layouts, and styling options, empowering developers to craft visually captivating interfaces that resonate with end-users on a deeper level. By delving into the intricacies of advanced GUI frameworks, participants enriched their skill set and expanded their horizons in software design and development.

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

At the heart of the project lay the tenets of version control, epitomized by the ubiquitous Git. Participants traversed the terrain of collaborative development, leveraging Git's prowess to track changes, manage code contributions, and safeguard against inadvertent errors. Through a series of commits and branches, participants meticulously sculpted a repository that chronicled the evolution of the project, serving as a testament to their dedication and prowess.

Git's robust features, ranging from branching and merging to conflict resolution, empowered participants to navigate through the complexities of collaborative development with poise and finesse. By adhering to version control best practices, participants fortified their codebase against the vagaries of software development, ensuring resilience and agility in the face of adversities.

**Following Best Practices: The Mark of Professionalism**

Throughout the project, participants upheld a standard of professionalism and excellence by adhering to a range of best practices in software development. Their decision to leverage standard libraries like tkinter for GUI development highlighted their pragmatic approach, opting to utilize existing functionalities rather than reinventing solutions. This demonstrated efficiency in resource allocation and ensured the reliability and stability of the user interface components.

Consistency in coding conventions played a vital role in enhancing the project's overall success. Participants maintained uniformity across indentation, naming conventions, and code structure, enhancing the readability and maintainability of the codebase. This adherence to standardized coding practices created an environment where collaboration flourished, enabling seamless idea exchange among team members. With a shared understanding of coding conventions, developers could easily review, debug, and modify code, contributing to a cohesive and high-quality software product.

**Conclusion**

The Travel Management System project wasn't just a software development endeavor; it marked a significant milestone in your journey as a software developer. Throughout this immersive experience, you not only acquired practical skills in Object-Oriented Programming (OOP) but also honed your abilities in UI design, delved into data persistence intricacies, and navigated project management complexities. This project underscored the importance of meticulous planning, effective version control, rigorous testing, and comprehensive documentation.

Now armed with a solid foundation, you're poised to venture deeper into software development's vast landscape. Opportunities abound to explore advanced OOP concepts, dive into richer UI frameworks, and immerse yourself in robust database systems. The world of software development beckons with boundless possibilities for growth, innovation, and the creation of remarkable applications. Embrace this journey with enthusiasm, for it promises continuous learning, creativity, and the realization of your full potential.

## References

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>.

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

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>