“Dynamic Aviation Management System”
First year Mini Project Report

Submitted in partial fulfillment of the requirements of the degree

BACHELOR OF ENGINEERING IN COMPUTER ENGINEERING

By

Niraj Chaudhari(11)

Yash Israni(19)

Tanmay Jadhav(20)

Sahil Kachare(24)

Supervisor

Prof. Richard Joseph

Department of Computer Engineering

Vivekanand Education Society’s Institute of Technology

HAMC, Collector’s Colony, Chembur,

Mumbai-400074

University of Mumbai

(AY 2023-24)

CERTIFICATE

This is to certify that the Mini Project entitled **“ Dynamic Aviation Management System ”** is a bonafide work of **Niraj Chaudhari(11) ,Yash Israni(19),Tanmay Jadhav(20), Sahil Kachare(24)** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **“Bachelor of Engineering”** in **“Computer Engineering” .**

(Prof. Richard Joseph **)**
Supervisor

(Prof._____) **(Prof._____)**

Head of Department Principal

ABSTRACT

The project titled "Dynamic Aviation Management System" aims to modernize the management of airports and airlines operations by leveraging data-driven insights and innovative technological solutions. The primary objectives of the project include enhancing the efficiency, accuracy, and agility of airport and airline management processes, thereby improving the overall travel experience for passengers and stakeholders.

To achieve these objectives, the project employs a comprehensive methodology that integrates advanced concepts such as Model-View-Controller (MVC) architecture, user-defined exceptions, JSON data handling, inheritance, polymorphism, encapsulation, abstract interfaces, and AWT/Swing for UI development. These concepts are meticulously implemented to establish a robust and scalable framework for managing airport facilities, airline routes, passenger enrollments, and flight schedules.

Key findings from the project encompass the successful implementation of CRUD (Create, Read, Update, Delete) operations for managing airport and airline data, coupled with the seamless integration of UI components to deliver an intuitive user interface for system interaction. Furthermore, the project showcases effective error handling mechanisms through the utilization of user-defined exceptions, ensuring smooth operation under diverse scenarios.

The implications of the project extend beyond its immediate scope, offering insights into the potential for leveraging technology to transform conventional airport and airline management practices. By embracing a data-driven approach and harnessing advanced technologies, the "Dynamic Travel Management System" sets a precedent for future advancements in the aviation industry, facilitating enhanced efficiency, innovation, and customer satisfaction.

In conclusion, the project represents a significant advancement in the realm of travel management, demonstrating the transformative power of technology to revolutionize traditional practices and deliver superior experiences for passengers, airlines, and airport authorities alike.

Acknowledgements

We would like to express our deepest gratitude to all those who contributed to the realization of the "Dynamic Aviation Management" project.

First and foremost, we extend our heartfelt thanks to our dedicated team members whose unwavering commitment, hard work, and collaborative efforts have been the driving force behind this project's success. Each member's unique skills and perspectives have enriched the development process and helped us overcome various challenges.

We are immensely thankful to Mr. Richard Joseph for his invaluable guidance, mentorship, and expertise throughout the project's lifecycle. His insightful feedback, constructive criticism, and unwavering support have been pivotal in shaping the project and expanding our knowledge of Java programming.

Special appreciation goes to our fellow batchmates,Niraj Chaudhari , Tanmay Jadhav,Yash Israni, Sahil Kachare for their continuous support, encouragement, and willingness to lend a helping hand whenever needed. Their contributions have significantly contributed to the project's success.

Furthermore, we acknowledge the invaluable resources and assistance provided by online platforms such as GeeksforGeeks, JavaTpoint, Stack Overflow, and W3Schools. These platforms have served as invaluable sources of knowledge, offering solutions to complex problems and facilitating our learning process.

Last but not least, we would like to express our gratitude to our friends, family, and well-wishers for their encouragement and support throughout this journey. Their belief in our abilities has been a constant source of motivation and inspiration.

In conclusion, we are deeply grateful to everyone who played a part in this project's success, no matter how big or small. Your contributions have made a significant impact, and we are truly appreciative of your support.

Introduction
"Aviation Management System"

The "Aviation Management System" project endeavors to streamline and optimize the management of aviation-related activities, catering to the diverse needs of airport authorities, airline operators, and passengers alike. Rooted in the principles of efficiency, accessibility, and innovation, this project aims to revolutionize the way airports and airlines function, ultimately enhancing the overall travel experience and operational efficiency within the aviation industry.

At its core, the project seeks to address key challenges faced by airports and airlines, including but not limited to passenger management, flight scheduling, resource allocation, and data management. By leveraging cutting-edge technologies and employing robust software solutions, the system aims to automate and streamline various processes, leading to improved service delivery, cost-effectiveness, and operational transparency.

With a comprehensive suite of features and functionalities, the "Airports and Airlines Management System" offers a holistic approach to managing aviation operations. From passenger check-in and baggage handling to flight scheduling and maintenance tracking, the system provides end-to-end solutions that cater to the diverse needs of aviation stakeholders.

Furthermore, the project places a strong emphasis on user experience and accessibility, ensuring that the system is intuitive, user-friendly, and accessible to all users, regardless of their technical expertise. Through the implementation of intuitive interfaces, interactive dashboards, and seamless navigation, the system aims to empower users to efficiently manage and monitor aviation operations with ease.

In addition to enhancing operational efficiency, the project also aims to improve data accuracy, reliability, and security. By centralizing and standardizing data management processes, the system minimizes the risk of errors, redundancies, and data breaches, thereby safeguarding sensitive information and ensuring compliance with regulatory standards.

Overall, the "Airports and Airlines Management System" represents a significant step forward in the modernization of aviation management practices. By harnessing the power of technology and innovation, the project aims to drive positive change within the aviation industry, ushering in a new era of efficiency, reliability, and customer satisfaction.

MOTIVATION

1. Enhanced Operational Efficiency: Traditional airport and airline management systems often rely on manual processes, leading to inefficiencies, delays, and errors. The project aims to automate key processes, such as passenger check-in, baggage handling, and flight scheduling, to improve operational efficiency and reduce turnaround times.

2. **Improved Passenger Experience**: Travelers today expect seamless and hassle-free experiences when navigating airports and boarding flights. By implementing user-friendly interfaces, self-service kiosks, and real-time updates, the system aims to enhance the overall passenger experience and satisfaction.

3. Optimized Resource Allocation: Efficient resource allocation is crucial for maximizing the utilization of airport facilities and airline resources. The project seeks to optimize resource allocation by providing insights into passenger traffic, flight demand, and resource availability, allowing airports and airlines to make data-driven decisions.

4. **Streamlined Regulatory Compliance:** The aviation industry is subject to stringent regulatory requirements and safety standards. The project aims to streamline regulatory compliance by centralizing data management, ensuring accurate reporting, and facilitating audits and inspections.

5. Scalability and Adaptability: As the aviation industry evolves and grows, there is a need for scalable and adaptable management systems that can accommodate changing requirements and emerging trends. The project is designed to be scalable and adaptable, allowing for future enhancements and customization as needed.

6. **Competitive Advantage:** By adopting modern management systems and technologies, airports and airlines can gain a competitive edge in the market. The project aims to provide innovative solutions that differentiate airports and airlines from their competitors and attract more passengers and business opportunities.

7. Industry Innovation: Ultimately, the project is driven by a desire to innovate and modernize the aviation industry, positioning airports and airlines for success in the digital age. By embracing technological advancements and best practices, the project aims to shape the future of aviation management and set new standards for excellence.

Problem Statement & Objectives

Problem Statement:

The aviation industry faces significant challenges in managing airports and airlines efficiently and effectively. Current systems often rely on outdated technology, manual processes, and disjointed systems, leading to operational inefficiencies, passenger dissatisfaction, and increased costs. Key issues include suboptimal resource allocation, lengthy check-in procedures, inadequate passenger information management, and regulatory compliance challenges. There is a pressing need for a modernized and integrated management solution that addresses these issues and enhances the overall airport and airline experience.

Objectives:

1. Streamline Operations: Develop a comprehensive management system that streamlines airport and airline operations, including passenger check-in, baggage handling, flight scheduling, crew management, and regulatory compliance processes. Integrate all functions into a centralized platform to improve efficiency and reduce delays.

2. Enhance Passenger Experience: Implement user-friendly interfaces, self-service kiosks, mobile applications, and real-time information systems to enhance the passenger experience from check-in to boarding. Provide personalized services, timely updates, and seamless travel experiences to improve satisfaction and loyalty.

3. **Optimize Resource Allocation**: Utilize data analytics, predictive modeling, and optimization algorithms to optimize resource allocation, including gate assignments, aircraft utilization, crew scheduling, and facility management. Minimize waste, reduce costs, and maximize operational efficiency.

4. Ensure Regulatory Compliance: Develop tools and features to ensure compliance with aviation regulations, safety standards, and security protocols. Implement real-time monitoring, reporting, and auditing capabilities to facilitate regulatory compliance and oversight.

5. Enhance Security: Implement advanced security measures, biometric authentication systems, and risk management protocols to enhance airport and airline security. Ensure the safety and well-being of passengers, staff, and assets while maintaining compliance with international security standards.

6. Drive Innovation: Foster a culture of innovation and continuous improvement within the aviation industry by introducing cutting-edge technologies, best practices, and industry standards. Encourage experimentation, creativity, and collaboration to drive positive change and maintain competitiveness in the global aviation market.

Proposed System

"Dynamic Aviation Management"

"Dynamic Aviation Management" represents a paradigm shift in the aviation industry, poised to redefine air travel experiences through cutting-edge technology and data-driven insights. This innovative system offers a holistic solution tailored to efficiently manage every facet of airport and airline operations, from passenger handling to fleet management and scheduling.

At its essence, this platform is engineered to streamline the complexities inherent in airport and airline management, providing intuitive interfaces and robust backend functionality. By granting airlines, airports, and ground service providers seamless access to vital information, it aims to optimize resource allocation, enhance operational efficiency, and elevate the overall passenger experience.

With "Dynamic Aviation Management," stakeholders can anticipate a transformative evolution in how air travel operations are orchestrated and experienced. By harnessing advanced technologies such as artificial intelligence, predictive analytics, and automation, this system empowers industry professionals to make data-driven decisions, mitigate risks, and ensure smooth and secure travel experiences for passengers worldwide.

Architecture/Framework

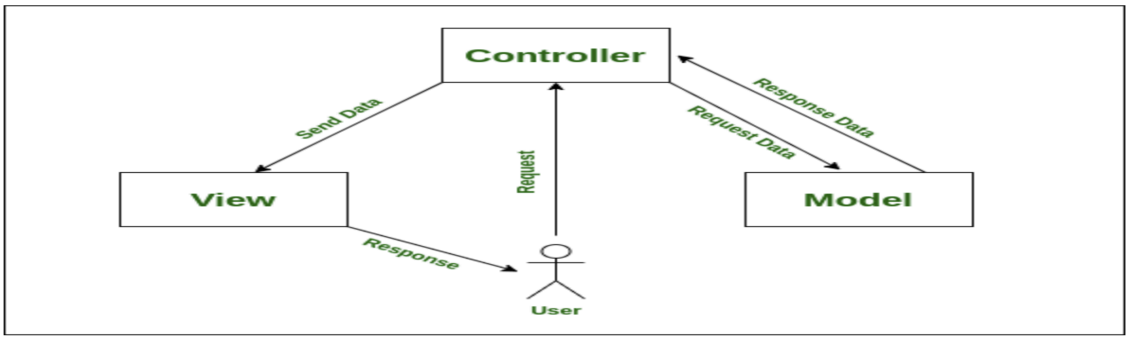
1. Model-View-Controller (MVC) Architecture:

The "Dynamic Aviation Management" project follows the Model-View-Controller (MVC) architecture, providing a structured approach to developing a Java application for efficient management of airport and airline operations. The project comprises three main components:

1. Model (M):
- The Model class encapsulates the application's data and business logic, managing the system's state and interactions with data sources such as databases.
- It includes classes like Airports, Airlines, and Trips, which represent entities and operations related to airports, airlines, and flight trips.
- Additionally, classes responsible for data persistence, such as DBHandler and FileHandler, handle database interactions and file operations, abstracting away implementation details from other components.

2. View (V):
- The View class is responsible for presenting data to users in a user-friendly format through graphical user interfaces (GUIs).
- It includes classes like ManageAirportsFrame, ManageAirlinesFrame, and ManageTripsFrame, which create GUIs for managing airports, airlines, and trips, respectively.
- These view classes provide interactive components for users to perform actions such as adding, updating, deleting, and viewing information related to airports, airlines, and trips.

3. Controller (C):
- The Controller layer acts as an intermediary between the Model and View layers, processing user input, interacting with the model to retrieve or update data, and updating the view accordingly.
- The Controller class coordinates actions between the model and view components, responding to user interactions and updating the model as needed
- It ensures proper communication between the user interface and the underlying data model, maintaining separation of concerns and facilitating modularity.



4. Database Connectivity:
- Utilizes JDBC (Java Database Connectivity) to connect to a relational database for data storage and retrieval.
- Establishes connections with the database server and executes SQL queries to interact with tables related to airlines, passengers, and trips.

5. JSON Data Handling:
- Incorporates JSON (JavaScript Object Notation) for data serialization and deserialization.
- Facilitates efficient storage and exchange of data between the application and external systems.
- JSON files may be utilized to store configurations, user preferences, or transient data.

6. Inheritance and Polymorphism:
- Leveraged to establish relationships and promote code reusability.
- Inheritance enables subclasses to inherit properties and behaviors from superclasses, while polymorphism allows objects of different classes to be treated interchangeably.

7. Exception Handling:
- Implements robust exception handling mechanisms to gracefully manage errors and exceptional situations.
- Custom exception classes may be defined to handle specific error scenarios encountered during runtime.

8. User Interface (UI):
- Develops intuitive and user-friendly interfaces for interacting with airline, passenger, and trip-related functionalities.
- Utilizes Swing and AWT components to create GUI elements such as buttons, text fields, and panels.

9. Concurrency: - Ensures concurrency management to handle multiple user interactions and background tasks simultaneously.
- Although explicit multithreading may not be extensively utilized, Swing GUI applications inherently involve concurrency management to maintain responsiveness.

10. CRUD Operations:
- Implements CRUD (Create, Read, Update, Delete) operations for managing airline, passenger, and trip data effectively.
- Users can create, retrieve, update, and delete records as necessary, ensuring data accuracy and integrity.

This framework provides a solid foundation for developing your project on airlines, passengers, and trips, emphasizing modularity, maintainability, and user experience.

Algorithm and Process Design

1. Requirement Analysis:
- Gather and analyze requirements from stakeholders, including functional and non-functional requirements.
- Identify user needs, business objectives, and system constraints to ensure the final product meets expectations.

2. System Architecture Design:
- Define the overall architecture, such as the Model-View-Controller (MVC) pattern, to determine component interactions.
- Establish components, responsibilities, and data flow to provide a blueprint for development.

3. GUI Design:
- Design intuitive, visually appealing GUI using Swing and AWT components.
- Consider usability, accessibility, and responsiveness to ensure a positive user experience.

4. Data Model Design:
- Identify entities like airlines, passengers, and trips.
- Define attributes and relationships to organize data effectively and accurately represent the domain.

5. Controller Implementation:
- Handle user inputs, coordinate actions, and control data flow.
- Interpret user requests, invoke operations on the model, and update the view accordingly.

6. View Implementation:
- Present information to users and capture interactions through the GUI.
- Render updates from the model and provide a user-friendly, responsive interface.

7. Model Implementation:

- Manage data storage, perform CRUD operations, enforce business rules, and maintain data integrity.
- Interact with databases or other data sources to store and retrieve information.

8. Error Handling:
- Implement robust error handling mechanisms to detect and handle exceptions gracefully.
- Define custom exception classes and perform validation checks to ensure data integrity and correctness.

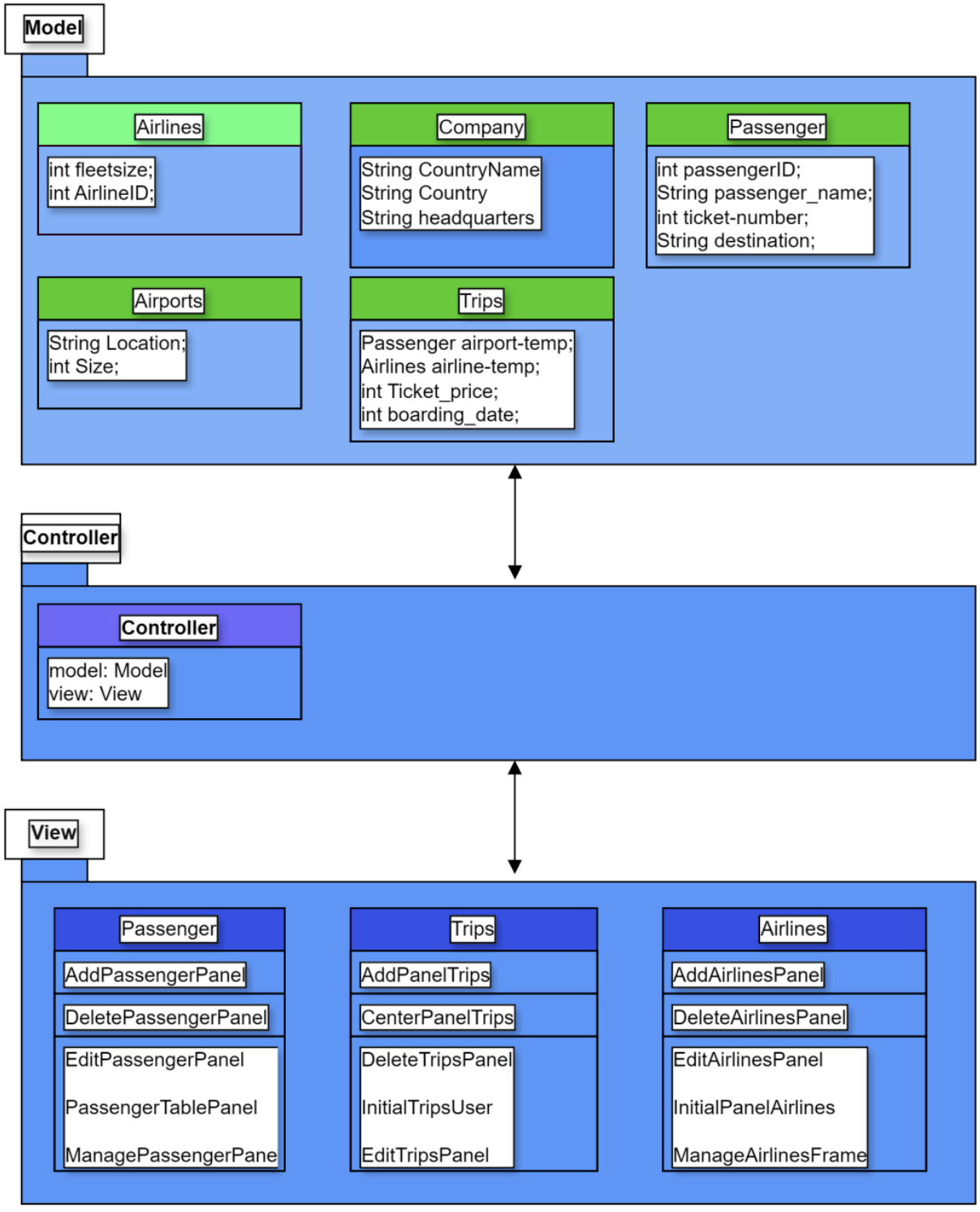
9. Integration:
- Integrate different components to work together seamlessly.
- Define communication protocols, data formats, and APIs for data exchange between components.

10. Testing and Debugging:
- Conduct rigorous testing, including unit testing, integration testing, and user acceptance testing.
- Identify and rectify defects, errors, and inconsistencies to ensure system stability and quality.

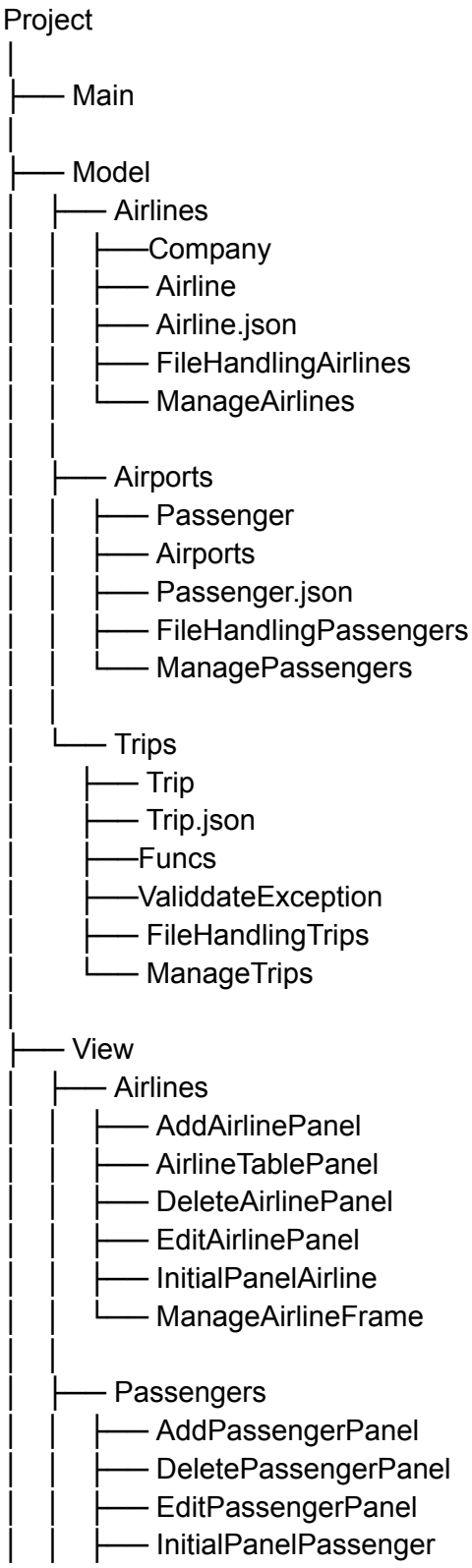
11. Documentation:
- Create comprehensive documentation, including design documents, user manuals, API documentation, and technical guides.
- Provide insights into the design, implementation, and usage of the system to aid developers, administrators, and end-users.

12. Optimization and Refinement:
- Continuously optimize and refine the system based on feedback, performance metrics, and evolving requirements.
- Address performance bottlenecks, implement usability enhancements, and add new features to keep the system competitive.

This algorithm and process design ensures a systematic and structured approach to building a robust and user-friendly Airlines, Passengers, and Trips Management System. By following these steps, the project aims to deliver a comprehensive solution for efficient management and optimization of airline operations, passenger bookings, and trip planning.


Model
Airlines
int fleetsize;
int AirlineID;
Company
String CountryName
String Country
String headquarters
Passenger
int passengerID;
String passenger_name;
int ticket-number;
String destination;
Airports
String Location;
int Size;
Trips
Passenger airport-temp;
Airlines airline-temp;
int Ticket_price;
int boarding_date;
Controller
Controller
model: Model
view: View
View
Passenger
AddPassengerPanel
DeletePassengerPanel
EditPassengerPanel
PassengerTablePanel
ManagePassengerPane
Trips
AddPanelTrips
CenterPanelTrips
DeleteTripsPanel
InitialTripsUser
EditTripsPanel
Airlines
AddAirlinesPanel
DeleteAirlinesPanel
EditAirlinesPanel
InitialPanelAirlines
ManageAirlinesFrame

Code


Project
Main
Model
Airlines
Company
Airline
Airline.json
FileHandlingAirlines
ManageAirlines
Airports
Passenger
Airports
Passenger.json
FileHandlingPassengers
ManagePassengers
Trips
Trip
Trip.json
Funcs
ValiddateException
FileHandlingTrips
ManageTrips
View
Airlines
AddAirlinePanel
AirlineTablePanel
DeleteAirlinePanel
EditAirlinePanel
InitialPanelAirline
ManageAirlineFrame
Passengers
AddPassengerPanel
DeletePassengerPanel
EditPassengerPanel
InitialPanelPassenger

```
│   │   └── ManagePassengerFrame
│   │
│   └── Trips
│       ├── AddTripPanel
│       ├── DeleteTripPanel
│       ├── EditTripPanel
│       ├── InitialPanelTrip
│       └── ManageTripFrame
│
└── Controller
    └── Controller
```

#INPUT

Main.Java

```
import Controller.Controller;
import Model.Model;
import View.View;

import java.io.IOException;

public class Main {
  public static void main(String[] args) throws IOException {
    View view = new View();
    Model model = new Model();
    Controller controller = new Controller(model, view);
  }
}
```

Controller.java

```
package Controller;

import Model.Airlines.Airlines;
import Model.Model;
import Model.Airports.*;
import View.View;
import java.awt.event.*;
import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.IOException;


public class Controller {
  Model model;
  View view;

  public Controller(Model m, View v) {
    model = m;
    view = v;

view.getFf().getManageAirlineBtn().addActionListener(new
ActionListener() {
      @Override
      public void actionPerformed(ActionEvent e) {
        System.out.println("Course Button Clicked");
        view.getFf().setVisible(false);
        view.getMcf().setVisible(true);
      }
    });


    view.getMcf().addWindowListener(new
java.awt.event.WindowAdapter() {
      @Override
      public void
windowClosing(java.awt.event.WindowEvent windowEvent) {
        view.getFf().setVisible(true);
      }
    });

    model.getMa().setLinesBeingDisplayed(20);

view.centerInitialSetupAirline(model.getMa().getLinesBeingD
isplayed(), model.getMa().getHeaders().size());

    model.getMa().setFirstLineToDisplay(0);

view.centerUpdateAirline(model.getMa().getLines(model.get
Ma().getFirstLineToDisplay(),
model.getMa().getLastLineToDisplay()),
model.getMa().getHeaders());

view.getMcf().getairline_ip().getCtp().addMouseWheelListen
er(new MouseWheelListener() {
      @Override
      public void mouseWheelMoved(MouseWheelEvent e)
{
        int units = e.getUnitsToScroll();
```

```
            System.out.println(units);
            int current_first_line =
model.getMa().getFirstLineToDisplay();
            int current_last_line =
model.getMa().getLastLineToDisplay();
            int no_of_airline = model.getMa().getTable().size();
            int no_of_display_lines =
model.getMa().getLinesBeingDisplayed();
            if(units <= 0 && current_first_line == 0)
            {
               model.getMa().setFirstLineToDisplay(0);
            }
            else if(units <= 0 && current_first_line > 0)
            {
               int new_first_line = current_first_line + units;
               if(new_first_line <= 0)
               {
                  model.getMa().setFirstLineToDisplay(0);
               }
               else
               {

model.getMa().setFirstLineToDisplay(new_first_line);
               }
            }
            else if(units > 0 && current_last_line ==
no_of_airline-1)
            {

model.getMa().setFirstLineToDisplay(current_first_line);
            }
            else if (units > 0 && current_last_line <
no_of_airline-1)
            {
               int new_first_line = current_first_line + units;
               if(new_first_line > no_of_airline -
no_of_display_lines)
               {
                  new_first_line =
no_of_airline-no_of_display_lines;

model.getMa().setFirstLineToDisplay(new_first_line);
               }
               else
               {

model.getMa().setFirstLineToDisplay(new_first_line);
               }
            }


view.centerUpdateAirline(model.getMa().getLines(model.get
Ma().getFirstLineToDisplay(),
model.getMa().getLastLineToDisplay()),
model.getMa().getHeaders());
         }
      });
```

```
view.getFf().getManagePassBtn().addActionListener(new
ActionListener() {
         @Override
         public void actionPerformed(ActionEvent e) {
            view.getFf().setVisible(false);
            view.getMsf().setVisible(true);
         }
      });
      view.getMsf().addWindowListener(new
java.awt.event.WindowAdapter() {
         @Override
         public void
windowClosing(java.awt.event.WindowEvent windowEvent) {
            view.getFf().setVisible(true);
         }
      });
      model.getMp().setLinesBeingDisplayed(20);

view.centerInitialSetupPassenger(model.getMp().getLinesBei
ngDisplayed(), model.getMp().getHeaders().size());


      model.getMp().setFirstLineToDisplay(0);

view.centerUpdatePassenger(model.getMp().getLines(model
.getMp().getFirstLineToDisplay(),
model.getMp().getLastLineToDisplay()),
model.getMp().getHeaders());

view.getMsf().getIp().getCp().addMouseWheelListener(new
MouseWheelListener() {
         @Override
         public void mouseWheelMoved(MouseWheelEvent e)
{
            int units = e.getUnitsToScroll();
            System.out.println(units);
            int current_first_line =
model.getMp().getFirstLineToDisplay();
            int current_last_line =
model.getMp().getLastLineToDisplay();
            int no_of_pass = model.getMp().getTable().size();
            int no_of_display_lines =
model.getMp().getLinesBeingDisplayed();
            if(units <= 0 && current_first_line == 0)
            {
               model.getMp().setFirstLineToDisplay(0);
            }
            else if(units <= 0 && current_first_line > 0)
            {
               int new_first_line = current_first_line + units;
               if(new_first_line <= 0)
               {
                  model.getMp().setFirstLineToDisplay(0);
               }
               else
               {

model.getMp().setFirstLineToDisplay(new_first_line);
               }
            }
```

```
            else if(units > 0 && current_last_line ==
no_of_pass-1)
            {

model.getMp().setFirstLineToDisplay(current_first_line);
            }
            else if (units > 0 && current_last_line <
no_of_pass-1)
            {
                int new_first_line = current_first_line + units;
                if(new_first_line > no_of_pass -
no_of_display_lines)
                {
                    new_first_line =
no_of_pass-no_of_display_lines;

model.getMp().setFirstLineToDisplay(new_first_line);
                }
                else
                {

model.getMp().setFirstLineToDisplay(new_first_line);
                }
            }


view.centerUpdatePassenger(model.getMp().getLines(model
.getMp().getFirstLineToDisplay(),
model.getMp().getLastLineToDisplay()),
model.getMp().getHeaders());

        }
    });


view.getFf().getManageTripsBtn().addActionListener(new
ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            System.out.println("Trips Button Clicked");
            view.getFf().setVisible(false);
            view.getMef().setVisible(true);
        }
    });

    view.getMef().addWindowListener(new
java.awt.event.WindowAdapter() {
        @Override
        public void
windowClosing(java.awt.event.WindowEvent windowEvent) {
            view.getFf().setVisible(true);
        }
    });
    model.getMe().setLinesBeingDisplayed(20);

view.centerInitialSetupTrips(model.getMe().getLinesBeingDis
played(), model.getMe().getHeaders().size());

    model.getMe().setFirstLineToDisplay(0);

view.centerUpdateTrips(model.getMe().getLines(model.getM
e().getFirstLineToDisplay(),
model.getMe().getLastLineToDisplay()),
model.getMe().getHeaders());

view.getMef().gettrip_ip().getCtp().addMouseWheelListener(
new MouseWheelListener() {
        @Override
        public void mouseWheelMoved(MouseWheelEvent e)
{
            int units = e.getUnitsToScroll();
            System.out.println(units);
            int current_first_line =
model.getMe().getFirstLineToDisplay();
            int current_last_line =
model.getMe().getLastLineToDisplay();
            int no_of_trips = model.getMe().getTable().size();
            int no_of_display_lines =
model.getMe().getLinesBeingDisplayed();
            if(units <= 0 && current_first_line == 0)
            {
                model.getMe().setFirstLineToDisplay(0);
            }
            else if(units <= 0 && current_first_line > 0)
            {
                int new_first_line = current_first_line + units;
                if(new_first_line <= 0)
                {
                    model.getMe().setFirstLineToDisplay(0);
                }
                else
                {

model.getMe().setFirstLineToDisplay(new_first_line);
                }
            }
            else if(units > 0 && current_last_line ==
no_of_trips-1)
            {

model.getMe().setFirstLineToDisplay(current_first_line);
            }
            else if (units > 0 && current_last_line <
no_of_trips-1)
            {
                int new_first_line = current_first_line + units;
                if(new_first_line > no_of_trips -
no_of_display_lines)
                {
                    new_first_line =
no_of_trips-no_of_display_lines;

model.getMe().setFirstLineToDisplay(new_first_line);
                }
                else
                {

model.getMe().setFirstLineToDisplay(new_first_line);
                }
            }
```

```
view.centerUpdateTrips(model.getMe().getLines(model.getMe().getFirstLineToDisplay(),
model.getMe().getLastLineToDisplay()),
model.getMe().getHeaders());

        }
    });



    view.getMsf().addWindowListener(new
java.awt.event.WindowAdapter() {
        @Override
        public void
windowClosing(java.awt.event.WindowEvent windowEvent) {
            view.getFf().setVisible(true);
        }
    });

view.getMcf().getairline_ip().getAcp().getAddAirlineBtn().add
ActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            String txt_airline_name =
view.getMcf().getairline_ip().getAcp().getTxt_Airline_name().
getText();
            String txt_country =
view.getMcf().getairline_ip().getAcp().getTxt_Country().getTe
xt();
            String txt_headquarters =
view.getMcf().getairline_ip().getAcp().getTxt_Headquarters().
getText();
            String txt_fleetsize =
view.getMcf().getairline_ip().getAcp().getTxt_FleetSize().getT
ext();
            try {

model.getMa().addNewAirline(txt_airline_name,txt_country,tx
t_headquarters,Integer.valueOf(txt_fleetsize));
            } catch (IOException ex) {
                throw new RuntimeException(ex);
            }
        }
    });

view.getMsf().getIp().getAps().getAddPassBtn().addActionLis
tener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            String txt_name =
view.getMsf().getIp().getAps().getTxt_lname().getText();
            String txt_tic_no =
view.getMsf().getIp().getAps().getTxt_tic_no().getText();
            String txt_des =
view.getMsf().getIp().getAps().getTxt_des().getText();
            String txt_size =
view.getMsf().getIp().getAps().getTxt_size().getText();
            String txt_loc =
view.getMsf().getIp().getAps().getTxt_loc().getText();
            try {
model.getMp().addNewPassenger(txt_name,Integer.valueOf(
txt_tic_no),txt_des,Integer.valueOf(txt_size),txt_loc);
            } catch (IOException ex) {
                throw new RuntimeException(ex);
            }
        }
    });
    view.getMsf().addWindowListener(new
java.awt.event.WindowAdapter() {
        @Override
        public void
windowClosing(java.awt.event.WindowEvent windowEvent) {
            view.getFf().setVisible(true);
        }
    });

view.getMef().gettrip_ip().getAcp().getAddTripsBtn().addActi
onListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            String txt_airline_name =
view.getMef().gettrip_ip().getAcp().getTxt_pass_id().getText()
;
            String txt_country =
view.getMef().gettrip_ip().getAcp().getTxt_Airline_id().getText
();
            String txt_headquarters =
view.getMef().gettrip_ip().getAcp().getTxt_Boarding_date().g
etText();
            String txt_fleetsize =
view.getMef().gettrip_ip().getAcp().getTxt_Ticket_price().getT
ext();

model.getMe().addNewTrip(Integer.valueOf(txt_airline_name
),Integer.valueOf(txt_country),txt_headquarters,Integer.value
Of(txt_fleetsize));
        }
    });

view.getMcf().getairline_ip().getEap().getEditAirlineBtn().add
ActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            System.out.println("Edit Airline Button Clicked");
            String txt_course_idx =
view.getMcf().getairline_ip().getEap().getTxt_get_airline_idx(
).getText();
            String txt_course_id =
view.getMcf().getairline_ip().getEap().getTxt_Airline_Name().
getText();
            String txt_course_name =
view.getMcf().getairline_ip().getEap().getTxt_country().getTe
xt();
            String txt_course_duration =
view.getMcf().getairline_ip().getEap().getTxt_headquarters().
getText();
            String txt_course_credits =
view.getMcf().getairline_ip().getEap().getTxt_Fleetsize().getT
ext();
            try {
```

```
model.getMa().editAirline(Integer.valueOf(txt_course_idx),txt
_course_id,txt_course_name,txt_course_duration,Integer.val
ueOf(txt_course_credits));
            } catch (IOException ex) {
                throw new RuntimeException(ex);
            }
        }
    });

view.getMsf().getIp().getEpp().getEditpassBtn().addActionLis
tener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            System.out.println("Edit Passenger Button
Clicked");
            String txt_pass_idx =
view.getMsf().getIp().getEpp().getTxt_get_pass_idx().getText
();
            String txt_pass_id =
view.getMsf().getIp().getEpp().getTxt_pass_name().getText()
;
            String txt_pass_name =
view.getMsf().getIp().getEpp().getTxt_tic_no().getText();
            String txt_pass_duration =
view.getMsf().getIp().getEpp().getTxt_des().getText();
            String txt_pass_credits =
view.getMsf().getIp().getEpp().getTxt_size().getText();
            String txt_pass_loc =
view.getMsf().getIp().getEpp().getTxt_loc().getText();
            try {

model.getMp().editPass(Integer.valueOf(txt_pass_idx),txt_pa
ss_id,Integer.valueOf(txt_pass_name),txt_pass_duration,Inte
ger.valueOf(txt_pass_credits),txt_pass_loc);
            } catch (IOException ex) {
                throw new RuntimeException(ex);
            }
        }
    });

view.getMcf().getairline_ip().getDap().getDeletAirlinebtn().ad
dActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            System.out.println("Delete Airline Button Clicked");
            String txt_course_idx =
view.getMcf().getairline_ip().getDap().getTxt_airline_id().getT
ext();

model.getMa().deleteairline(Integer.valueOf(txt_course_idx));
        }
    });

view.getMsf().getIp().getDpp().getDeletePassBtn().addAction
Listener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            String txt_airline_idx =
view.getMsf().getIp().getDpp().getTxt_pass_id().getText();

model.getMp().deletePass(Integer.valueOf(txt_airline_idx));
        }
    });

view.getMef().gettrip_ip().getDep().getDeletAirlinebtn().addA
ctionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            String txt_airl_idx =
view.getMef().gettrip_ip().getDep().getTxt_airline_id().getText
();
            String txt_pass_idx =
view.getMef().gettrip_ip().getDep().getTxt_pass_id().getText(
);

model.getMe().deleteEnroll(Integer.valueOf(txt_airl_idx),Inte
ger.valueOf(txt_pass_idx));
        }
    });

  }
}
```

View.java

```
package View;

import View.Airlines.ManageAirlineFrame;
import View.Trips.ManageTripsFrame;
import View.Passenger.ManagePassengerFrame;

import java.awt.*;
import java.util.ArrayList;

public class View {
  FirstFrame ff;
  ManagePassengerFrame msf;

  ManageAirlineFrame mcf;

  ManageTripsFrame mef;



  public View()
  {
    ff = new FirstFrame();
    msf = new ManagePassengerFrame();
    mcf = new ManageAirlineFrame();
    mef = new ManageTripsFrame();
  }

  public void centerInitialSetupPassenger(int
linesBeingDisplayed, int size) {
    msf.getIp().getCp().setLayout(new
GridLayout(linesBeingDisplayed+1,size));

msf.getIp().getCp().createButtons((linesBeingDisplayed+1) *
size);
```

```
  }

  public void centerInitialSetupAirline(int
linesBeingDisplayed, int size) {
      mcf.getairline_ip().getCtp().setLayout(new
GridLayout(linesBeingDisplayed+1,size));

mcf.getairline_ip().getCtp().createButtons((linesBeingDisplay
ed+1) * size);
  }

  public void centerInitialSetupTrips(int linesBeingDisplayed,
int size) {
      mef.gettrip_ip().getCtp().setLayout(new
GridLayout(linesBeingDisplayed+1,size));

mef.gettrip_ip().getCtp().createButtons((linesBeingDisplayed
+1) * size);
  }

  public void
centerUpdateAirline(ArrayList<ArrayList<String>> lines,
ArrayList<String> headers) {
      for (int i = 0; i < headers.size(); i++)
      {

mcf.getairline_ip().getCtp().getAllButtons().get(i).setText(hea
ders.get(i));
      }

      for (int row_no = 0; row_no < lines.size(); row_no++)
      {
         for (int col_no = 0; col_no < headers.size(); col_no++)
         {
            int button_no = row_no * headers.size() +
headers.size() + col_no;
            String button_txt = lines.get(row_no).get(col_no);


mcf.getairline_ip().getCtp().getAllButtons().get(button_no).se
tText(button_txt);
         }
      }
  }


  public void
centerUpdatePassenger(ArrayList<ArrayList<String>> lines,
ArrayList<String> headers) {
      for (int i = 0; i < headers.size(); i++)
      {

msf.getIp().getCp().getAllButtons().get(i).setText(headers.get
(i));
      }

      for (int row_no = 0; row_no < lines.size(); row_no++)
      {
         for (int col_no = 0; col_no < headers.size(); col_no++)
         {
            int button_no = row_no * headers.size() +
headers.size() + col_no;
            String button_txt = lines.get(row_no).get(col_no);


msf.getIp().getCp().getAllButtons().get(button_no).setText(bu
tton_txt);
         }
      }
  }
  public void
centerUpdateTrips(ArrayList<ArrayList<String>> lines,
ArrayList<String> headers) {
      for (int i = 0; i < headers.size(); i++)
      {

mef.gettrip_ip().getCtp().getAllButtons().get(i).setText(header
s.get(i));
      }

      for (int row_no = 0; row_no < lines.size(); row_no++)
      {
         for (int col_no = 0; col_no < headers.size(); col_no++)
         {
            int button_no = row_no * headers.size() +
headers.size() +col_no;
            String button_txt = lines.get(row_no).get(col_no);


mef.gettrip_ip().getCtp().getAllButtons().get(button_no).setTe
xt(button_txt);
         }
      }
  }


  public void setFf(FirstFrame ff) {
      this.ff = ff;
  }

  public FirstFrame getFf() {
      return ff;
  }

  public void setMsf(ManagePassengerFrame msf) {
      this.msf = msf;
  }

  public ManagePassengerFrame getMsf() {
      return msf;
  }

  public void setMcf(ManageAirlineFrame mcf) {
      this.mcf = mcf;
  }

  public ManageAirlineFrame getMcf() {
      return mcf;
  }

  public void setMef(ManageTripsFrame mef) {
```

```
        this.mef = mef;
    }
    public ManageTripsFrame getMef() {
        return mef;
    }
}
```

Airlines.java

```
package Model.Airlines;

import java.io.SyncFailedException;
import java.util.ArrayList;

public class Airlines extends Company {
    private static int airline_count=0;
    //Declaring variables
    public int AirlineID;

    public int fleetSize;
    public static int getAirline_count(){
        return airline_count;
    }

    public Airlines()
    {
        airline_count++;
        this.setAirlineID(airline_count);
    }
    public Airlines(String CompanyName , String Country
,String Headquarters , int FleetSize)
    {
        super(CompanyName, Country, Headquarters);
        airline_count++;
        this.setAirlineID(airline_count);
        this.setFleetSize(FleetSize);
    }
    public Airlines(int AirlineID, String CompanyName , String
Country ,String Headquarters , int FleetSize)
    {
        super(CompanyName, Country, Headquarters);
        airline_count++;
        this.setAirlineID(AirlineID);
        this.setFleetSize(FleetSize);
    }

    public void setAirlineID(int airlineID)
    {
        this.AirlineID = airlineID;
    }
    //Creating Setters

    public void setFleetSize(int fleetSize)
    {
        this.fleetSize = fleetSize;
    }

    //Creating Getters
    public int getAirlineID() {
        return AirlineID;
    }
    public int getFleetSize()
    {
        return fleetSize;
    }
    ///Display Airline data
    public void display(){
        super.display();
        System.out.println("Airline ID:"+getAirlineID() );
        System.out.println("Airline Fleet Size:"+getFleetSize() );
    }
//    Declaring Function bodies which are declared in
airlreview interface


}
```

Company.java

```
package Model.Airlines;

public class Company {
    private String CompanyName;
    private String Country;
    private String headquarters;
    public Company(){
        System.out.println("Creating a Company");
    }
    public Company(String CompanyName,String
Country,String Headquarters){
        this.setCompanyName(CompanyName);
        this.setCountry(Country);
        this.setHeadquarters(Headquarters);
    }
    public void setCompanyName(String CompanyName)
    {
        this.CompanyName = CompanyName;
    }
    public void setCountry(String Country)
    {
        this.Country = Country;
    }
    public void setHeadquarters(String headquarters)
    {
        this.headquarters = headquarters;
    }
    public String getCompanyName()
    {
        return this.CompanyName;
    }
    public String getCountry()
    {
        return this.Country;
    }
    public String getHeadquarters()
    {
        return this.headquarters;
    }
    public void display(){
        System.out.println("Airline Name:"+getCompanyName()
);
        System.out.println("Airline Country:"+getCountry() );
```

```
    System.out.println("Airline
Headquarters:"+getHeadquarters() );
  }
}
```

FileHandlingAirlines.java

```
package Model.Airlines;

import Model.Airports.Passenger;

import java.io.IOException;
import java.util.ArrayList;

public abstract class FileHandlingAirlines {
  protected abstract ArrayList<Airlines>
readAirlJsonFile(String file_path);
  protected abstract void writeAirlJsonFile(String file_path,
ArrayList<Airlines> airlines) throws IOException;
}
```

manageairl.java

```
package Model.Airlines;
import Model.Utils.Displayable;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import Model.Airports.FileHandelingAirports;
import java.io.File;
import java.io.IOException;
import java.nio.file.Paths;
import java.util.ArrayList;

public class manageairl extends FileHandlingAirlines
implements Displayable{
  ArrayList<Airlines> airlines = new ArrayList<Airlines>();

  ObjectMapper objectMapper = new ObjectMapper();

  private int linesBeingDisplayed;
  private int firstLineIndex;
  int lastLineIndex;
  int highlightedLine;

  public manageairl(){
    readAirlJsonFile("C:/Users/niraj/Downloads/exp__mp
(2)/exp__mp/src/Model/Airlines/Airline_Data.json");
  }
  public ArrayList<Airlines> readAirlJsonFile(String file_path)
  {
    try {
      JsonNode rootNode = objectMapper.readTree(new
File(file_path));

      // Iterate through JSON array
      airlines.clear();
      if (rootNode.isArray()) {
        for (JsonNode node : rootNode) {
          int airlineID = node.has("airlineID") ?
node.get("airlineID").asInt() : 0; // Default value is 0
          String companyName =
node.has("companyName") ?
node.get("companyName").asText() : ""; // Default value is
empty string
          int fleetSize = node.has("fleetSize") ?
node.get("fleetSize").asInt() : 0; // Default value is 0
          String country = node.has("country") ?
node.get("country").asText() : ""; // Default value is empty
string
          String headquarters = node.has("headquarters")
? node.get("headquarters").asText() : ""; // Default value is
empty string
          Airlines stud = new
Airlines(airlineID,companyName,country,headquarters,fleetSi
ze);
          airlines.add(stud);
        }
      }
    } catch (IOException e) {
      e.printStackTrace();
    }

    return airlines;
  }

  public void writeAirlJsonFile(String file_path,
ArrayList<Airlines> airlines) throws IOException {

//objectMapper.writeValue(Paths.get("src/Model/Students/stu
dents.json").toFile(), students);

objectMapper.writeValue(Paths.get(file_path).toFile(),airlines
);
  }

  public void setStudentsTable(ArrayList<Airlines> airlines) {
    this.airlines = airlines;
  }

  public ArrayList<String> getHeaders() {
    ArrayList<String> headers = new ArrayList<String>();
    headers.add("Airline Id");
    headers.add("Airline Name");
    headers.add("Country");
    headers.add("Headquarters");
    headers.add("FleetSize");

    return headers;
  }
  @Override
  public ArrayList<String> getLine(int line) {
    ArrayList<String> student_details = new
ArrayList<String>();


student_details.add(String.valueOf(airlines.get(line).getAirlin
eID()));

student_details.add(airlines.get(line).getCompanyName());
    student_details.add(airlines.get(line).getCountry());
    student_details.add(airlines.get(line).getHeadquarters());
```

```
student_details.add(String.valueOf(airlines.get(line).getFleet
Size()));

      return student_details;
  }

  @Override
  public ArrayList<ArrayList<String>> getLines(int firstLine,
int lastLine) {
      ArrayList<ArrayList<String>> students_subset = new
ArrayList<ArrayList<String>>();

      for (int i = firstLine; i <= lastLine; i++) {
          students_subset.add(getLine(i));
      }
      return students_subset;
  }

  @Override
  public int getFirstLineToDisplay() {
      return firstLineIndex;
  }

  @Override
  public int getLineToHighlight() {
      return highlightedLine;
  }

  @Override
  public int getLastLineToDisplay() {
      setLastLineToDisplay(getFirstLineToDisplay() +
getLinesBeingDisplayed() - 1);
      return lastLineIndex;
  }

  @Override
  public int getLinesBeingDisplayed() {
      return linesBeingDisplayed;
  }

  @Override
  public void setFirstLineToDisplay(int firstLine) {
      this.firstLineIndex = firstLine;
  }

  @Override
  public void setLineToHighlight(int highlightedLine) {
      this.highlightedLine = highlightedLine;
  }

  @Override
  public void setLastLineToDisplay(int lastLine) {
      this.lastLineIndex = lastLine;
  }

  @Override
  public void setLinesBeingDisplayed(int numberOfLines) {
      this.linesBeingDisplayed = numberOfLines;
  }

  public ArrayList getTable() {
      return airlines;
  }
  public void addNewAirline(String Airline_name, String
Country, String Headquarters, int Fleetsize) throws
IOException {

airlines=readAirlJsonFile("C:/Users/niraj/Downloads/exp__m
p (2)/exp__mp/src/Model/Airlines/Airline_Data.json");
      int x=airlines.size();
      Airlines temp_stud = new Airlines(Airline_name,Country,
Headquarters,Fleetsize);
      airlines.add(temp_stud);
      writeAirlJsonFile("C:/Users/niraj/Downloads/exp__mp
(2)/exp__mp/src/Model/Airlines/Airline_Data.json",airlines);

  }public void editAirline(int edit_course_idx,String
Airline_name, String Country, String Headquarters, int
Fleetsize) throws IOException {

airlines=readAirlJsonFile("C:/Users/niraj/Downloads/exp__m
p (2)/exp__mp/src/Model/Airlines/Airline_Data.json");

airlines.get(edit_course_idx).setCompanyName(Airline_nam
e);
      airlines.get(edit_course_idx).setCountry(Country);

airlines.get(edit_course_idx).setHeadquarters(Headquarters)
;
      airlines.get(edit_course_idx).setFleetSize(Fleetsize);
      writeAirlJsonFile("C:/Users/niraj/Downloads/exp__mp
(2)/exp__mp/src/Model/Airlines/Airline_Data.json",airlines);
  }
  public void deleteairline(int airline_idx){
      try {

airlines=readAirlJsonFile("C:/Users/niraj/Downloads/exp__m
p (2)/exp__mp/src/Model/Airlines/Airline_Data.json");
          airlines.remove(airline_idx);
          writeAirlJsonFile("C:/Users/niraj/Downloads/exp__mp
(2)/exp__mp/src/Model/Airlines/Airline_Data.json",airlines);
      } catch (IOException e) {
          e.printStackTrace();
      }
  }
}
```

Airports.java

```
package Model.Airports;


public class Airports {


  //Declaring Variables
  private  String Location;
  private int Size;


  public Airports()
```

```
  {
     System.out.println();
  }

  public Airports(int Size,String Location)
  {
     this.setSize(Size);
     this.setLocation(Location);
  }

  //Creating getters
  public String getLocation()
  {
     return this.Location;
  }
  public int getSize()
  {
     return this.Size;
  }




  //Creating Setters
  public void setLocation(String Location)
  {
     this.Location=Location;
  }
  public void setSize(int Size)
  {
     this.Size=Size;
  }




  //Display Airport data
  public void display(){
     System.out.println("Airport Country:"+getLocation() );
     System.out.println("Airport Size:"+getSize());
  }
}
```

FileHandlingAirports.java

```
package Model.Airports;

import java.io.IOException;
import java.util.ArrayList;

public abstract class FileHandelingAirports {
  protected abstract ArrayList<Passenger>
readPassJsonFile(String file_path);
  protected abstract void writePassJsonFile(String file_path,
ArrayList<Passenger> passengers) throws IOException;
}
```

Managepass.java

```
package Model.Airports;
import Model.Utils.Displayable;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import Model.Airports.FileHandelingAirports;
import java.io.File;
import java.io.IOException;
import java.nio.file.Paths;
import java.util.ArrayList;

public class managepass extends FileHandelingAirports
implements Displayable{
  ArrayList<Passenger> passengers = new
ArrayList<Passenger>();

  ObjectMapper objectMapper = new ObjectMapper();

  private int linesBeingDisplayed;
  private int firstLineIndex;
  int lastLineIndex;
  int highlightedLine;

  public managepass(){
     readPassJsonFile("C:/Users/niraj/Downloads/exp__mp
(2)/exp__mp/src/Model/Airports/passenger_details.json");
  }
  public ArrayList<Passenger> readPassJsonFile(String
file_path)
  {
     try {
        JsonNode rootNode = objectMapper.readTree(new
File(file_path));

        // Iterate through JSON array
        passengers.clear();
        if (rootNode.isArray()) {
           for (JsonNode node : rootNode) {
              int passenger_ID = node.has("passenger_ID") ?
node.get("passenger_ID").asInt() : 0; // Default value is 0
              String passenger_name =
node.has("passenger_name") ?
node.get("passenger_name").asText() : ""; // Default value is
empty string
              int ticket_number = node.has("ticket_number") ?
node.get("ticket_number").asInt() : 0; // Default value is 0
              String destination = node.has("destination") ?
node.get("destination").asText() : ""; // Default value is empty
string
              int Size = node.has("Size") ?
node.get("Size").asInt() : 0; // Default value is 0
              String Location = node.has("Location") ?
node.get("Location").asText() : ""; // Default value is empty
string

              Passenger stud = new
Passenger(passenger_ID, passenger_name, ticket_number,
destination,Size, Location);
              passengers.add(stud);
           }
        }
     } catch (IOException e) {
        e.printStackTrace();
     }
```

```
        return passengers;
    }

        public void writePassJsonFile(String file_path,
ArrayList<Passenger> passengers) throws IOException {

//objectMapper.writeValue(Paths.get("src/Model/Students/stu
dents.json").toFile(), students);

objectMapper.writeValue(Paths.get(file_path).toFile(),passen
gers);
    }

    public void setStudentsTable(ArrayList<Passenger>
passengers) {
        this.passengers = passengers;
    }

    public ArrayList<String> getHeaders() {
        ArrayList<String> headers = new ArrayList<String>();
        headers.add("Passenger Id");
        headers.add("Passenger Name");
        headers.add("Ticket Number");
        headers.add("Destination");
        headers.add("Size");
        headers.add("Location");

        return headers;
    }
    @Override
    public ArrayList<String> getLine(int line) {
        ArrayList<String> student_details = new
ArrayList<String>();

student_details.add(String.valueOf(passengers.get(line).get
Passenger_ID()));

student_details.add(passengers.get(line).getPassenger_nam
e());

student_details.add(String.valueOf(passengers.get(line).getT
icket_number()));

student_details.add(passengers.get(line).getDestination());

student_details.add(String.valueOf(passengers.get(line).get
Size()));
        student_details.add(passengers.get(line).getLocation());

        return student_details;
    }

    @Override
    public ArrayList<ArrayList<String>> getLines(int firstLine,
int lastLine) {
        ArrayList<ArrayList<String>> students_subset = new
ArrayList<ArrayList<String>>();

        for (int i = firstLine; i <= lastLine; i++) {
            students_subset.add(getLine(i));
        }
        return students_subset;
    }

    @Override
    public int getFirstLineToDisplay() {
        return firstLineIndex;
    }

    @Override
    public int getLineToHighlight() {
        return highlightedLine;
    }

    @Override
    public int getLastLineToDisplay() {
        setLastLineToDisplay(getFirstLineToDisplay() +
getLinesBeingDisplayed() - 1);
        return lastLineIndex;
    }

    @Override
    public int getLinesBeingDisplayed() {
        return linesBeingDisplayed;
    }

    @Override
    public void setFirstLineToDisplay(int firstLine) {
        this.firstLineIndex = firstLine;
    }

    @Override
    public void setLineToHighlight(int highlightedLine) {
        this.highlightedLine = highlightedLine;
    }

    @Override
    public void setLastLineToDisplay(int lastLine) {
        this.lastLineIndex = lastLine;
    }

    @Override
    public void setLinesBeingDisplayed(int numberOfLines) {
        this.linesBeingDisplayed = numberOfLines;
    }

    public ArrayList getTable() {
        return passengers;
    }
    public void addNewPassenger(String passenger_name, int
ticket_number, String destination, int Size, String Location)
throws IOException {
        readPassJsonFile("C:/Users/niraj/Downloads/exp__mp
(2)/exp__mp/src/Model/Airports/passenger_details.json");
        int x=passengers.size();
        Passenger temp_stud = new
Passenger(x+1,passenger_name, ticket_number,
destination,Size, Location);;
        passengers.add(temp_stud);
        writePassJsonFile("C:/Users/niraj/Downloads/exp__mp
(2)/exp__mp/src/Model/Airports/passenger_details.json",pas
sengers);
```

```
  }
  public void editPass(int edit_course_idx,String name,int
tic_no,String Des,int size,String loc) throws IOException {

passengers=readPassJsonFile("C:/Users/niraj/Downloads/ex
p__mp
(2)/exp__mp/src/Model/Airports/passenger_details.json");

passengers.get(edit_course_idx).setPassenger_name(name
);

passengers.get(edit_course_idx).setTicket_number(tic_no);
      passengers.get(edit_course_idx).setDestination(Des);
      passengers.get(edit_course_idx).setSize(size);
      passengers.get(edit_course_idx).setLocation(loc);
      writePassJsonFile("C:/Users/niraj/Downloads/exp__mp
(2)/exp__mp/src/Model/Airports/passenger_details.json",pas
sengers);
  }
  public void deletePass(int pass_idx){
     try {

passengers=readPassJsonFile("C:/Users/niraj/Downloads/ex
p__mp
(2)/exp__mp/src/Model/Airports/passenger_details.json");
         passengers.remove(pass_idx);

writePassJsonFile("C:/Users/niraj/Downloads/exp__mp
(2)/exp__mp/src/Model/Airports/passenger_details.json",pas
sengers);
     } catch (IOException e) {
        e.printStackTrace();
     }
  }
}
```

Passenger.java

```
package Model.Airports;


import java.util.ArrayList;


public class Passenger extends Airports {

  private static int passenger_count = 0;
  int passenger_ID;
  String passenger_name;
  int ticket_number;
  String destination;

  public static int getPassenger_count() {
     return passenger_count;
  }

  public Passenger() {
     passenger_count++;
     this.setPassenger_ID(passenger_count);
  }

  public Passenger(String passenger_name, int
ticket_number, String destination,  int Size, String Location) {
     super(Size,Location);
     passenger_count++;
     this.setPassenger_ID(passenger_count);
     this.setPassenger_name(passenger_name);
     this.setTicket_number(ticket_number);
     this.setDestination(destination);
  }

  public Passenger(int passenger_ID, String
passenger_name, int ticket_number, String destination, int
Size, String Location) {
     super(Size,Location);
     passenger_count++;
     this.setPassenger_ID(passenger_ID);
     this.setPassenger_name(passenger_name);
     this.setTicket_number(ticket_number);
     this.setDestination(destination);

  }


  public void setPassenger_ID(int passenger_ID) {
     this.passenger_ID = passenger_ID;
  }

  public void setPassenger_name(String passenger_name) {
     this.passenger_name = passenger_name;
  }

  public void setTicket_number(int ticket_number) {
     this.ticket_number = ticket_number;
  }

  public void setDestination(String destination) {
     this.destination = destination;
  }


  public int getPassenger_ID() {
     return passenger_ID;
  }

  public String getPassenger_name() {
     return passenger_name;
  }

  public int getTicket_number() {
     return ticket_number;
  }

  public String getDestination() {
     return destination;
  }

  public void display()
  {
     System.out.println("Passenger ID;"+getPassenger_ID());
```

```
      System.out.println("Passenger
Name:"+getPassenger_name());
      System.out.println("Ticket
Number:"+getTicket_number());
      System.out.println("Enter Destination"+getDestination());
      super.display();
  }

}
```

Trips.java

```
package Model.Trips;
import Model.Airports.*;
import Model.Airlines.*;

public class Trips {
  private Passenger airport_temp;
  private String Boarding_date;
  private int Ticket_Price;
  private Airlines airline_temp;

  public Trips(Passenger p,Airlines a, String date, int
tic_price){
      setAirport_temp(p);
      setBoarding_date(date);
      setTicket_Price(tic_price);
      setAirline_temp(a);
  }

  public void setAirport_temp(Passenger airport_temp) {
      this.airport_temp = airport_temp;
  }
  public void setAirline_temp(Airlines airline_temp) {
      this.airline_temp = airline_temp;
  }


  public void setBoarding_date(String Boarding_date) {
      this.Boarding_date = Boarding_date;
  }

  public void setTicket_Price(int ticket_Price) {
      this.Ticket_Price = ticket_Price;
  }

  public Passenger getAirport_temp() {
      return airport_temp;
  }
  public Airlines getAirline_temp() {
      return airline_temp;
  }

  public String getBoarding_date() {
      return Boarding_date;
  }

  public int getTicket_Price() {
      return Ticket_Price;
  }

}
```

manageTrips.java

```
package Model.Trips;

import Model.Airports.*;
import Model.Utils.Displayable;
import Model.Airlines.*;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;

import java.io.File;
import java.io.IOException;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;

public class manageTrips extends FileHandlingTrips
implements Displayable {
  ArrayList<Passenger> passengers = new
ArrayList<Passenger>();
  ArrayList<Airlines> airlines = new ArrayList<Airlines>();
  ArrayList<Trips> Trips_data = new ArrayList<Trips>();

  ObjectMapper objectMapper = new ObjectMapper();

  private int linesBeingDisplayed;
  private int firstLineIndex;
  int lastLineIndex;
  int highlightedLine;

  public manageTrips() {
      managepass mp1 = new managepass();
      passengers = mp1.getTable();
      manageairl ma1 = new manageairl();
      airlines = ma1.getTable();


      readTripsJsonFile("C:/Users/niraj/Downloads/exp__mp
(2)/exp__mp/src/Model/Trips/Enrollment_Data.json");
  }

  @Override
  public ArrayList<Trips> readTripsJsonFile(String file_path)
{
      try {
          JsonNode rootNode = objectMapper.readTree(new
File(file_path));

          // Iterate through JSON array
          if (rootNode.isArray()) {
              for (JsonNode node : rootNode) {
                  int airport_temp =
node.get("airport_temp").asInt();
                  int airline_temp =
node.get("airline_temp").asInt();
                  String Boarding_date =
node.get("Boarding_date").asText();
                  int Ticket_Price =
node.get("Ticket_Price").asInt();
```

```
                Passenger p_temp_obj = null;
                Airlines a_temp_obj = null;

                for (int i = 0; i < passengers.size(); i++) {
                    if (airport_temp ==
passengers.get(i).getPassenger_ID()) {
                        p_temp_obj = passengers.get(i);
                    }
                }

                for (int i = 0; i < airlines.size(); i++) {
                    if (airline_temp == airlines.get(i).getAirlineID())
{
                        a_temp_obj = airlines.get(i);
                    }
                }

                Trips e_temp = new Trips(p_temp_obj,
a_temp_obj, Boarding_date, Ticket_Price);
                Trips_data.add(e_temp);
            }
        }
    } catch (IOException e) {
        e.printStackTrace();
    }

    return Trips_data;
  }

  @Override
  public void writeTripsJsonFile(String file_path,
ArrayList<Trips> enroll_al) throws IOException {
    ArrayList<Map<String, Object>> enroll_to_be_written =
new ArrayList<>();

    for (int i = 0; i < enroll_al.size(); i++) {
        HashMap<String, Object> data = new HashMap<>();
        data.put("airport_temp",
enroll_al.get(i).getAirport_temp().getPassenger_ID());
        data.put("airline_temp",
enroll_al.get(i).getAirline_temp().getAirlineID());
        data.put("Boarding_date",
enroll_al.get(i).getBoarding_date());
        data.put("Ticket_Price",
enroll_al.get(i).getTicket_Price());

        enroll_to_be_written.add(data);
    }

    objectMapper.writeValue(Paths.get(file_path).toFile(),
enroll_to_be_written);
  }

  public ArrayList<String> getHeaders() {
    ArrayList<String> headers = new ArrayList<String>();
    headers.add("Passenger Name:");
    headers.add("Airline Name:");
    headers.add("Boarding Date:");
    headers.add("Ticket Price:");
    return headers;
  }

  @Override
  public ArrayList<String> getLine(int line) {
    ArrayList<String> enroll_details = new
ArrayList<String>();

    // Check if enrollment_data is empty or if line is out of
bounds
    if (!Trips_data.isEmpty() && line < Trips_data.size()) {

enroll_details.add(Trips_data.get(line).getAirport_temp().get
Passenger_name());

enroll_details.add(Trips_data.get(line).getAirline_temp().get
CompanyName());

enroll_details.add(Trips_data.get(line).getBoarding_date());

enroll_details.add(String.valueOf(Trips_data.get(line).getTick
et_Price()));
    }

    return enroll_details;
  }

  @Override
  public ArrayList<ArrayList<String>> getLines(int firstLine,
int lastLine) {
    ArrayList<ArrayList<String>> Airline_subset = new
ArrayList<ArrayList<String>>();

    // Check if enrollment_data is empty
    if (!Trips_data.isEmpty()) {
        for (int i = firstLine; i <= lastLine && i <
Trips_data.size(); i++) {
            Airline_subset.add(getLine(i));
        }
    }

    return Airline_subset;
  }

  @Override
  public int getFirstLineToDisplay() {
    return this.firstLineIndex;
  }

  @Override
  public int getLineToHighlight() {
    return 0;
  }

  @Override
  public int getLastLineToDisplay() {
    setLastLineToDisplay(this.firstLineIndex +
this.linesBeingDisplayed - 1);
    return this.lastLineIndex;
  }
```

```
  @Override
  public int getLinesBeingDisplayed() {
     return this.linesBeingDisplayed;
  }

  @Override
  public void setFirstLineToDisplay(int firstLine) {
     this.firstLineIndex = firstLine;
  }

  @Override
  public void setLineToHighlight(int highlightedLine) {

  }

  @Override
  public void setLastLineToDisplay(int lastLine) {
     this.lastLineIndex = lastLine;
  }

  @Override
  public void setLinesBeingDisplayed(int numberOfLines) {
     this.linesBeingDisplayed = numberOfLines;
  }

  public ArrayList<Trips> getTable() {
     readTripsJsonFile("C:/Users/niraj/Downloads/exp__mp
(2)/exp__mp/src/Model/Trips/Enrollment_Data.json");
     return Trips_data;
  }
  public void addNewTrip(int pass,int air,String date,int
price){
     try {

Trips_data=readTripsJsonFile("C:/Users/niraj/Downloads/ex
p__mp (2)/exp__mp/src/Model/Trips/Enrollment_Data.json");
        managepass mp1 = new managepass();
        passengers = mp1.getTable();
        manageairl ma1 = new manageairl();
        airlines = ma1.getTable();
        Passenger passa=passengers.get(pass);
        Airlines airl=airlines.get(air);
        Trips en=new Trips(passa,airl,date,price);
        Trips_data.add(en);

writeTripsJsonFile("C:/Users/niraj/Downloads/exp__mp
(2)/exp__mp/src/Model/Trips/Enrollment_Data.json",Trips_da
ta);
     } catch (IOException e) {
        e.printStackTrace();
     }
  }
  public void deleteEnroll(int a,int b) {
     Trips_data =
readTripsJsonFile("CC:/Users/niraj/Downloads/exp__mp
(2)/exp__mp/src/Model/Trips/Enrollment_Data.json");
     managepass mp1 = new managepass();
     passengers = mp1.getTable();
     manageairl ma1 = new manageairl();
     airlines = ma1.getTable();
     for (int i = 0; i < Trips_data.size(); i++) {
        if(Trips_data.get(i).getAirline_temp().getAirlineID()==a
&&
Trips_data.get(i).getAirport_temp().getPassenger_ID()==b){
           Trips_data.remove(i);
        }
     }
     }
  }
```

FileHandlingTrips.java

```
package Model.Trips;

import java.io.IOException;
import java.util.ArrayList;

public abstract class FileHandlingTrips {
  protected abstract ArrayList<Trips>
readTripsJsonFile(String file_path);
  protected abstract void writeTripsJsonFile(String file_path,
ArrayList<Trips> enrolls) throws IOException;
}
```

Funcs.java

```
package Model.Trips;

import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.time.format.DateTimeParseException;

public class Funcs {

  public static boolean isValidDate(String dateStr) {
     DateTimeFormatter formatter =
DateTimeFormatter.ofPattern("dd/MM/yyyy");
     try {
        LocalDate.parse(dateStr, formatter);
        return true; // Valid date
     } catch (DateTimeParseException e) {
        return false; // Invalid date
     }
  }
}
```

ValidDateException.java

```
package Model.Trips;

public class ValidDateException extends Exception{
  public ValidDateException(String message){
     super(message);
  }
}
```

Displayable.java

```
package Model.Utils;


import java.util.ArrayList;

public interface Displayable
{   ArrayList<String> getLine(int line);
```

```
  ArrayList<ArrayList<String>> getLines(int firstLine, int
lastLine);

  public int getFirstLineToDisplay();
  public int getLineToHighlight();
  public int getLastLineToDisplay();
  public int getLinesBeingDisplayed();

  public void setFirstLineToDisplay(int firstLine);
  public void setLineToHighlight(int highlightedLine);
  public void setLastLineToDisplay(int lastLine);
  public void setLinesBeingDisplayed(int numberOfLines);

}
```

Model.java

```
package Model;

import Model.Airports.managepass;
import Model.Airlines.manageairl;
import Model.Trips.manageTrips;

public class Model {
  managepass mp;
  manageairl ma;
  manageTrips me;

  public Model() {
     mp = new managepass();
     ma = new manageairl();
     me = new manageTrips();
  }

  public managepass getMp() {
     return mp;
  }

  public manageairl getMa() {
     return ma;
  }

  public manageTrips getMe() {
     return me;
  }

  public void setMp(managepass mp) {
     this.mp = mp;
  }

  public void setMa(manageairl ma) {
     this.ma = ma;
  }

  public void setMe(manageTrips me) {
     this.me = me;
  }
}
```

Package View

Airlines

AddAirlinesPanel.java

```
package View.Airlines;

import javax.swing.*;

public class AddAirlinesPanel extends JPanel {

  JTextField txt_Airline_name;
  JTextField txt_Country;
  JTextField txt_Headquarters;
  JTextField txt_FleetSize;
  JButton addAirlineBtn;

  public AddAirlinesPanel()
  {

     txt_Airline_name = new JTextField();
     txt_Country = new JTextField();
     txt_Headquarters = new JTextField();
     txt_FleetSize = new JTextField();
     addAirlineBtn = new JButton("Add Airline");


     txt_Airline_name.setText("txt_Airline_name");
     txt_Country.setText("txt_Country");
     txt_Headquarters.setText("txt_Headquarters");
     txt_FleetSize.setText("txt_FleetSize");

     add(txt_Airline_name);
     add(txt_Country);
     add(txt_Headquarters);
     add(txt_FleetSize);
     add(addAirlineBtn);
  }

  public JTextField getTxt_Airline_name() {
     return txt_Airline_name;
  }

  public JTextField getTxt_Country() {
     return txt_Country;
  }

  public JTextField getTxt_Headquarters() {
     return txt_Headquarters;
  }

  public JTextField getTxt_FleetSize() {
     return txt_FleetSize;
  }

  public JButton getAddAirlineBtn() {
     return addAirlineBtn;
  }

  public void setTxt_Airline_name(JTextField
txt_Airline_name) {
     this.txt_Airline_name = txt_Airline_name;
  }
```

```
  public void setTxt_Country(JTextField txt_Country) {
    this.txt_Country = txt_Country;
  }

  public void setTxt_Headquarters(JTextField
txt_Headquarters) {
    this.txt_Headquarters = txt_Headquarters;
  }

  public void setTxt_FleetSize(JTextField txt_FleetSize) {
    this.txt_FleetSize = txt_FleetSize;
  }

  public void setAddAirlineBtn(JButton addAirlineBtn) {
    this.addAirlineBtn = addAirlineBtn;
  }
}
```

AirlineTablePanel.java

```
package View.Airlines;

import javax.swing.*;
import java.awt.*;
import java.util.ArrayList;

public class AirlineTablePanel extends JPanel {

  ArrayList<JButton> airline_buttons = new ArrayList<>();

  public AirlineTablePanel()
  {
    super();
  }

  public void createButtons(int count)
  {
    for (int i = 1; i <= count; i++)
    {
      JButton b = new JButton();
      b.setBackground(Color.cyan);
      b.setSize(500,50);
      airline_buttons.add(b);
      this.add(b);
      validate();
      repaint();
    }

  }

  public void setButtonText(int button_no, String button_text)
{
    airline_buttons.get(button_no).setText(button_text);
  }

  public ArrayList<JButton> getAllButtons()
  {
    return airline_buttons;
  }
}
```

deletAirlinePanel.java

```
package View.Airlines;

import javax.swing.*;

public class deleteAirlinePanel extends JPanel {
  JTextField txt_airline_id;
  JButton deletAirlinebtn;
  public deleteAirlinePanel(){
    txt_airline_id=new JTextField();
    deletAirlinebtn=new JButton("Delete Airline");

    txt_airline_id.setText("txt_airline_id");

    add(txt_airline_id);
    add(deletAirlinebtn);
  }

  public void setDeletAirlinebtn(JButton deletAirlinebtn) {
    this.deletAirlinebtn = deletAirlinebtn;
  }

  public JButton getDeletAirlinebtn() {
    return deletAirlinebtn;
  }

  public void setTxt_airline_id(JTextField txt_airline_id) {
    this.txt_airline_id = txt_airline_id;
  }

  public JTextField getTxt_airline_id() {
    return txt_airline_id;
  }
}
```

editAirlinePanel

```
package View.Airlines;

import Model.Model;

import javax.swing.*;

public class editAirlinePanel extends JPanel {
  JTextField txt_get_airline_idx;
  JButton getAirlineBtn;
  JTextField txt_Airline_Name;
  JTextField txt_country;
  JTextField txt_headquarters;
  JTextField txt_Fleetsize;
  JButton editAirlineBtn;
  Model m1 =new Model();

  public editAirlinePanel()
  {

    txt_Airline_Name = new JTextField();
    txt_country = new JTextField();
    txt_headquarters = new JTextField();
    txt_Fleetsize = new JTextField();
```

```
        editAirlineBtn = new JButton("Edit Airline");
        txt_get_airline_idx = new JTextField();
        getAirlineBtn = new JButton("Get Airline to Edit");

        txt_Airline_Name.setText("txt_Airline_Name");
        txt_country.setText("txt_country");
        txt_headquarters.setText("txt_headquarters");
        txt_Fleetsize.setText("txt_Fleetsize");
        txt_get_airline_idx.setText("get airline id");

        add(txt_get_airline_idx);
        add(getAirlineBtn);
        add(txt_Airline_Name);
        add(txt_country);
        add(txt_headquarters);
        add(txt_Fleetsize);
        add(editAirlineBtn);
    }

    public JTextField getTxt_Airline_Name() {
        return txt_Airline_Name;
    }

    public JTextField getTxt_country() {
        return txt_country;
    }

    public JTextField getTxt_headquarters() {
        return txt_headquarters;
    }

    public JTextField getTxt_Fleetsize() {
        return txt_Fleetsize;
    }

    public JButton getEditAirlineBtn() {
        return editAirlineBtn;
    }

    public JTextField getTxt_get_airline_idx() {
        return txt_get_airline_idx;
    }
    public JButton getGetAirlineBtn() {
        return getAirlineBtn;
    }
    public void setTxt_Airline_Name(JTextField
txt_Airline_Name) {
        this.txt_Airline_Name = txt_Airline_Name;
    }

    public void setTxt_country(JTextField txt_country) {
        this.txt_country = txt_country;
    }

    public void setTxt_headquarters(JTextField
txt_headquarters) {
        this.txt_headquarters = txt_headquarters;
    }

    public void setTxt_Fleetsize(JTextField txt_Fleetsize) {
        this.txt_Fleetsize = txt_Fleetsize;
    }

    public void setEditAirlineBtn(JButton editAirlineBtn) {
        this.editAirlineBtn = editAirlineBtn;
    }

    public void setTxt_get_airline_idx(JTextField
txt_get_airline_idx) {
        this.txt_get_airline_idx = txt_get_airline_idx;
    }

    public void setGetAirlineBtn(JButton getAirlineBtn) {
        this.getAirlineBtn = getAirlineBtn;
    }
}
```

InitialPanelAirline.java

```
package View.Airlines;

import javax.swing.*;

public class InitialPanelAirline extends JPanel {

    private AirlineTablePanel ctp;
    private editAirlinePanel eap;
    private deleteAirlinePanel dap;
    private View.Airlines.AddAirlinesPanel acp;

    public InitialPanelAirline() {
        super();
        ctp = new AirlineTablePanel();
        add(ctp);
        acp = new View.Airlines.AddAirlinesPanel();
        add(acp);
        eap= new editAirlinePanel();
        add(eap);
        dap=new deleteAirlinePanel();
        add(dap);
    }

    public void setCtp(AirlineTablePanel ctp) {
        this.ctp = ctp;
    }

    public void setEap(editAirlinePanel eap) {
        this.eap = eap;
    }

    public void setDap(deleteAirlinePanel dap) {
        this.dap = dap;
    }

    public deleteAirlinePanel getDap() {
        return dap;
    }

    public editAirlinePanel getEap() {
        return eap;
    }
```

```
  public AirlineTablePanel getCtp() {
    return ctp;
  }


  public void setAcp(View.Airlines.AddAirlinesPanel acp) {
    this.acp = acp;
  }

  public View.Airlines.AddAirlinesPanel getAcp() {
    return acp;
  }
}
```

ManageAirlineFrame

```
package View.Airlines;

import javax.swing.*;

public class ManageAirlineFrame extends JFrame {
  InitialPanelAirline airline_ip;

  public ManageAirlineFrame()
  {
    super("Manage Airlines DashBoard");
    airline_ip = new InitialPanelAirline();
    add(airline_ip);
    pack();
    setSize(950, 800);
  }

  public void setairline_ip(InitialPanelAirline course_ip) {
    this.airline_ip = course_ip;
  }

  public InitialPanelAirline getairline_ip() {
    return airline_ip;
  }
}
```

Passengers

AddPanelPassenger

```
package View.Passenger;

import javax.swing.*;

public class AddPanelPassenger extends JPanel {
  JTextField txt_Passenger_name;
  JTextField txt_tic_no;
  JTextField txt_des;
  JTextField txt_loc;
  JTextField txt_size;
  JButton addPassBtn;

  public AddPanelPassenger()
  {
    txt_Passenger_name = new JTextField();
    txt_tic_no = new JTextField();
    txt_des = new JTextField();
    txt_loc = new JTextField();
    txt_size = new JTextField();
    addPassBtn = new JButton("Add Passenger");
;
    txt_Passenger_name.setText("Passenger_name");
    txt_tic_no.setText("tic_no");
    txt_des.setText("des");
    txt_loc.setText("loc");
    txt_size.setText("size");

    add(txt_Passenger_name);
    add(txt_tic_no);
    add(txt_des);
    add(txt_loc);
    add(txt_size);
    add(addPassBtn);
  }


  public JTextField getTxt_lname() {
    return txt_Passenger_name;
  }

  public JTextField getTxt_tic_no() {
    return txt_tic_no;
  }

  public JTextField getTxt_loc() {
    return txt_loc;
  }

  public JTextField getTxt_des() {
    return txt_des;
  }

  public JTextField getTxt_size() {
    return txt_size;
  }

  public JButton getAddPassBtn() {
    return addPassBtn;
  }

  public void setTxt_size(JTextField txt_size) {
    this.txt_size = txt_size;
  }

  public void setAddPassBtn(JButton addPassBtn) {
    this.addPassBtn = addPassBtn;
  }

  public void setTxt_des(JTextField txt_des) {
    this.txt_des = txt_des;
  }

  public void setTxt_Passenger_name(JTextField
txt_Passenger_name) {
    this.txt_Passenger_name = txt_Passenger_name;
  }

  public void setTxt_loc(JTextField txt_loc) {
    this.txt_loc = txt_loc;
  }
```

```
  public void setTxt_tic_no(JTextField txt_tic_no) {
    this.txt_tic_no = txt_tic_no;
  }

}
```

CenterPanelPasseneger.java

```
package View.Passenger;

import javax.swing.*;
import java.awt.*;
import java.util.ArrayList;

public class CenterPanelPassenger extends JPanel {

  ArrayList<JButton> buttons = new ArrayList<>();

  public CenterPanelPassenger()
  {
    super();
  }

  public void createButtons(int count)
  {
    for (int i = 1; i <= count; i++)
    {
      JButton b = new JButton();
      b.setBackground(Color.cyan);
      b.setSize(500,50);
      buttons.add(b);
      this.add(b);
      validate();
      repaint();
    }

  }

  public void setButtonText(int button_no, String button_text)
{
    buttons.get(button_no).setText(button_text);
  }

  public ArrayList<JButton> getAllButtons()
  {
    return buttons;
  }

}
```

deletePassPanel.java

```
package View.Passenger;

import javax.swing.*;

public class deletePassPanel extends JPanel {
  JTextField txt_pass_id;
  JButton deletePassBtn;
  public deletePassPanel(){
    txt_pass_id=new JTextField();
    deletePassBtn=new JButton("Delete Passenger");
    txt_pass_id.setText("txt_pass_id");

    add(txt_pass_id);
    add(deletePassBtn);
  }

  public void setDeletePassBtn(JButton deletePassBtn) {
    this.deletePassBtn = deletePassBtn;
  }

  public void setTxt_pass_id(JTextField txt_pass_id) {
    this.txt_pass_id = txt_pass_id;
  }

  public JButton getDeletePassBtn() {
    return deletePassBtn;
  }

  public JTextField getTxt_pass_id() {
    return txt_pass_id;
  }
}
```

editPasspanel.java

```
package View.Passenger;

import Model.Model;

import javax.swing.*;

public class editPasspanel extends JPanel{
  JTextField txt_get_pass_idx;
  JButton getpassBtn;
  JTextField txt_pass_name;
  JTextField txt_tic_no;
  JTextField txt_des;
  JTextField txt_size;
  JTextField txt_loc;
  JButton editpassBtn;
  Model m1 =new Model();

  public editPasspanel()
  {

    txt_pass_name = new JTextField();
    txt_tic_no = new JTextField();
    txt_des = new JTextField();
    txt_size = new JTextField();
    txt_loc=new JTextField();
    editpassBtn = new JButton("Edit Passenger");
    txt_get_pass_idx = new JTextField();
    getpassBtn = new JButton("Get Passenger to Edit");

    txt_pass_name.setText("txt_pass_name");
    txt_tic_no.setText("txt_tic_no");
    txt_des.setText("txt_des");
    txt_size.setText("txt_size");
    txt_loc.setText("txt_loc");
    txt_get_pass_idx.setText("get Passenger id");

    add(txt_get_pass_idx);
```

```
        add(getpassBtn);
        add(txt_pass_name);
        add(txt_tic_no);
        add(txt_des);
        add(txt_size);
        add(txt_loc);
        add(editpassBtn);
    }

    public JTextField getTxt_pass_name() {
        return txt_pass_name;
    }

    public JTextField getTxt_tic_no() {
        return txt_tic_no;
    }

    public JTextField getTxt_des() {
        return txt_des;
    }

    public JTextField getTxt_size() {
        return txt_size;
    }

    public JButton getEditpassBtn() {
        return editpassBtn;
    }

    public void setTxt_loc(JTextField txt_loc) {
        this.txt_loc = txt_loc;
    }

    public JTextField getTxt_loc() {
        return txt_loc;
    }

    public JTextField getTxt_get_pass_idx() {
        return txt_get_pass_idx;
    }
    public JButton getGetpassBtn() {
        return getpassBtn;
    }
    public void setTxt_pass_name(JTextField txt_pass_name)
{
        this.txt_pass_name = txt_pass_name;
    }

    public void setTxt_tic_no(JTextField txt_tic_no) {
        this.txt_tic_no = txt_tic_no;
    }

    public void setTxt_des(JTextField txt_des) {
        this.txt_des = txt_des;
    }

    public void setTxt_size(JTextField txt_size) {
        this.txt_size = txt_size;
    }

    public void setEditpassBtn(JButton editpassBtn) {
        this.editpassBtn = editpassBtn;
    }

    public void setTxt_get_pass_idx(JTextField
txt_get_pass_idx) {
        this.txt_get_pass_idx = txt_get_pass_idx;
    }

    public void setGetpassBtn(JButton getpassBtn) {
        this.getpassBtn = getpassBtn;
    }
}
```

InitialPanelPassenger.java

```
package View.Passenger;

import javax.swing.*;

public class InitialPanelPassenger extends JPanel {

    private CenterPanelPassenger cp;
    private AddPanelPassenger aps;
    private editPasspanel epp;
    private deletePassPanel dpp;

    public InitialPanelPassenger() {
        super();
        cp = new CenterPanelPassenger();
        add(cp);
        aps = new AddPanelPassenger();
        add(aps);
        epp=new editPasspanel();
        add(epp);
        dpp=new deletePassPanel();
        add(dpp);
    }

    public CenterPanelPassenger getCp() {
        return cp;
    }

    public void setDpp(deletePassPanel dpp) {
        this.dpp = dpp;
    }

    public deletePassPanel getDpp() {
        return dpp;
    }

    public void setEpp(editPasspanel epp) {
        this.epp = epp;
    }

    public editPasspanel getEpp() {
        return epp;
    }


    public void setCp(CenterPanelPassenger cp) {
        this.cp = cp;
    }
```

```
  public void setAps(AddPanelPassenger aps) {
    this.aps = aps;
  }

  public AddPanelPassenger getAps() {
    return aps;
  }
}
```

ManagePassengerFrame.java

```
package View.Passenger;

import javax.swing.*;

public class ManagePassengerFrame extends JFrame {
  InitialPanelPassenger ip;

  public ManagePassengerFrame()
  {
    super("Manage Passenger DashBoard");
    ip = new InitialPanelPassenger();
    add(ip);
    pack();
    setSize(950, 800);
  }

  public void setIp(InitialPanelPassenger ip) {
    this.ip = ip;
  }

  public InitialPanelPassenger getIp() {
    return ip;
  }
}
```

Trips.java

```
package View.Trips;

import javax.swing.*;

public class AddTripsPanel extends JPanel {

  JTextField txt_Airline_id;
  JTextField txt_pass_id;
  JTextField txt_Boarding_date;
  JTextField txt_Ticket_price;
  JButton addTripsBtn;

  public AddTripsPanel()
  {

    txt_Airline_id = new JTextField();
    txt_pass_id = new JTextField();
    txt_Boarding_date = new JTextField();
    txt_Ticket_price = new JTextField();
    addTripsBtn = new JButton("Add Trips");


    txt_Airline_id.setText("Airline_id");
    txt_pass_id.setText("pass_id");
    txt_Boarding_date.setText("Boarding_date");
    txt_Ticket_price.setText("Ticket_price");

    add(txt_Airline_id);
    add(txt_pass_id);
    add(txt_Boarding_date);
    add(txt_Ticket_price);
    add(addTripsBtn);
  }

  public JTextField getTxt_Airline_id() {
    return txt_Airline_id;
  }

  public JTextField getTxt_pass_id() {
    return txt_pass_id;
  }

  public JTextField getTxt_Boarding_date() {
    return txt_Boarding_date;
  }

  public JTextField getTxt_Ticket_price() {
    return txt_Ticket_price;
  }

  public JButton getAddTripsBtn() {
    return addTripsBtn;
  }

  public void setTxt_Airline_id(JTextField txt_Airline_id) {
    this.txt_Airline_id = txt_Airline_id;
  }

  public void setTxt_pass_id(JTextField txt_pass_id) {
    this.txt_pass_id = txt_pass_id;
  }

  public void setTxt_Boarding_date(JTextField
txt_Boarding_date) {
    this.txt_Boarding_date = txt_Boarding_date;
  }

  public void setTxt_Ticket_price(JTextField txt_Ticket_price)
{
    this.txt_Ticket_price = txt_Ticket_price;
  }

  public void setAddTripsBtn(JButton addEnrollBtn) {
    this.addTripsBtn = addTripsBtn;
  }
}
```

deleteTripsPanel.java

```
package View.Trips;

import javax.swing.*;

public class deleteTripsPanel extends JPanel {
  JTextField txt_airline_id;
  JTextField txt_pass_id;
```

```
  JButton deletetripsbtn;
  public deleteTripsPanel(){
     txt_airline_id=new JTextField();
     txt_pass_id=new JTextField();
     deletetripsbtn=new JButton("Delete Trips");

     txt_airline_id.setText("txt_airline_id");
     txt_pass_id.setText("txt_pass_id");

     add(txt_airline_id);
     add(txt_pass_id);
     add(deletetripsbtn);
  }

  public void setDeletAirlinebtn(JButton deleteenrollbtn) {
     this.deletetripsbtn = deleteenrollbtn;
  }

  public JButton getDeletAirlinebtn() {
     return deletetripsbtn;
  }

  public void setTxt_airline_id(JTextField txt_airline_id) {
     this.txt_airline_id = txt_airline_id;
  }

  public void setTxt_pass_id(JTextField txt_pass_id) {
     this.txt_pass_id = txt_pass_id;
  }

  public JTextField getTxt_pass_id() {
     return txt_pass_id;
  }

  public JTextField getTxt_airline_id() {
     return txt_airline_id;
  }
}
```

InitialPanelTrips.java

```
package View.Trips;

import javax.swing.*;

public class InitialPanelTrips extends JPanel {

  private TripsTablePanel etp;
  private AddTripsPanel aep;
  private deleteTripsPanel dep;

  public InitialPanelTrips() {
     super();
     etp = new TripsTablePanel();
     add(etp);
     aep = new AddTripsPanel();
     add(aep);
     dep= new deleteTripsPanel();
     add(dep);
  }

  public void setCtp(TripsTablePanel etp) {
     this.etp = etp;
  }

  public TripsTablePanel getCtp() {
     return etp;
  }

  public void setDep(deleteTripsPanel dep) {
     this.dep = dep;
  }

  public deleteTripsPanel getDep() {
     return dep;
  }

  public void setAcp(AddTripsPanel aep) {
     this.aep = aep;
  }

  public AddTripsPanel getAcp() {
     return aep;
  }
}
```

ManageTripsFrame.java

```
package View.Trips;

import javax.swing.*;

public class ManageTripsFrame extends JFrame {
  InitialPanelTrips trip_ip;

  public ManageTripsFrame()
  {
     super("Manage Trips DashBoard");
     trip_ip = new InitialPanelTrips();
     add(trip_ip);
     pack();
     setSize(950, 800);
  }

  public void settrip_ip(InitialPanelTrips enroll_ip) {
     this.trip_ip = enroll_ip;
  }

  public InitialPanelTrips gettrip_ip() {
     return trip_ip;
  }
}
```

TripsTablePanel.java

```
package View.Trips;

import javax.swing.*;
import java.awt.*;
import java.util.ArrayList;

public class TripsTablePanel extends JPanel {

  ArrayList<JButton> trip_buttons = new ArrayList<>();
```

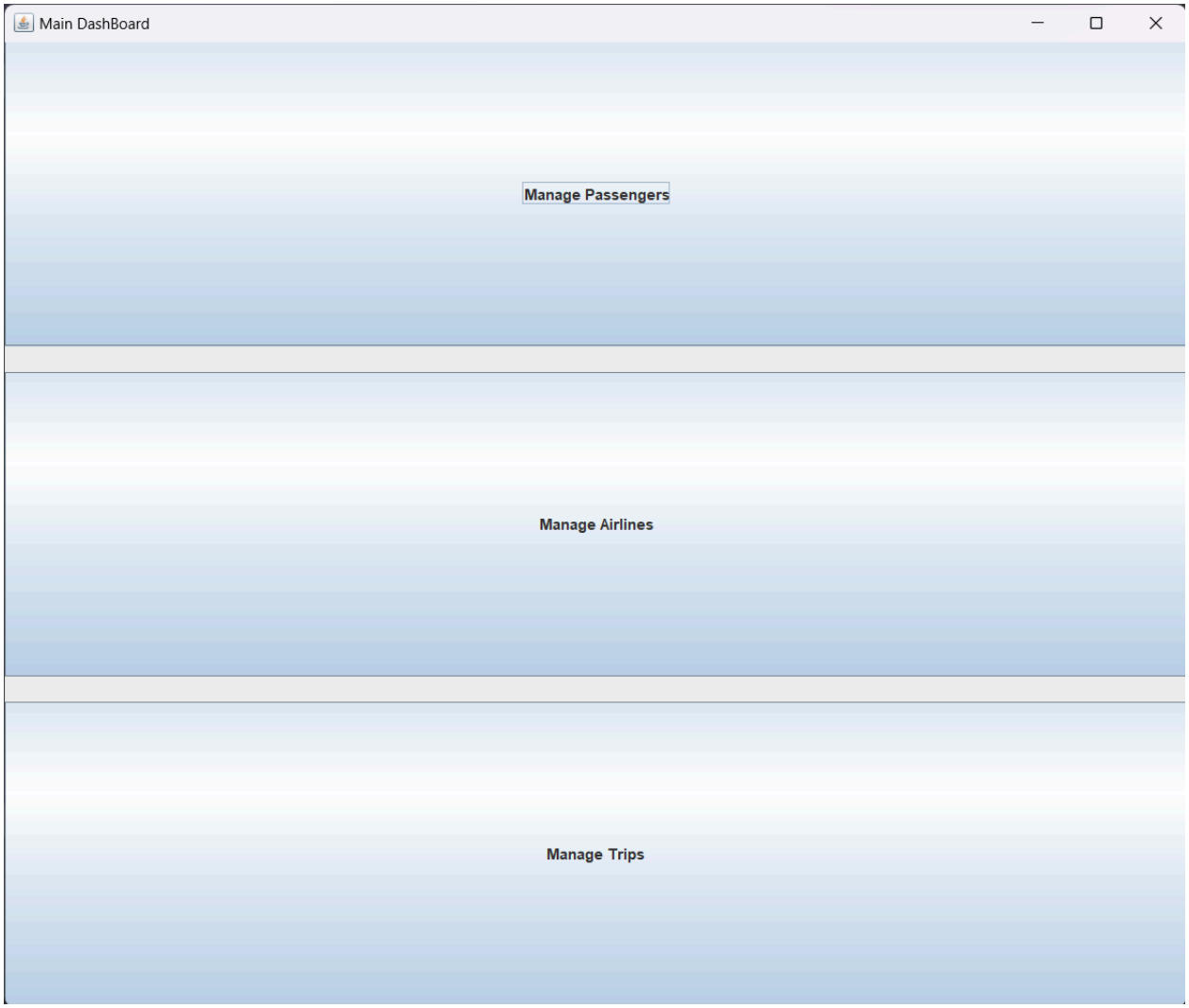
```
  public TripsTablePanel()
  {
     super();
  }

  public void createButtons(int count)
  {
     for (int i = 1; i <= count; i++)
     {
        JButton b = new JButton();
        b.setBackground(Color.cyan);
        b.setSize(500,50);
        trip_buttons.add(b);
        this.add(b);
        validate();
        repaint();
     }

  }

  public void setButtonText(int button_no, String button_text)
{
     trip_buttons.get(button_no).setText(button_text);
  }

  public ArrayList<JButton> getAllButtons()
  {
     return trip_buttons;
  }

}
```

FirstFrame.java

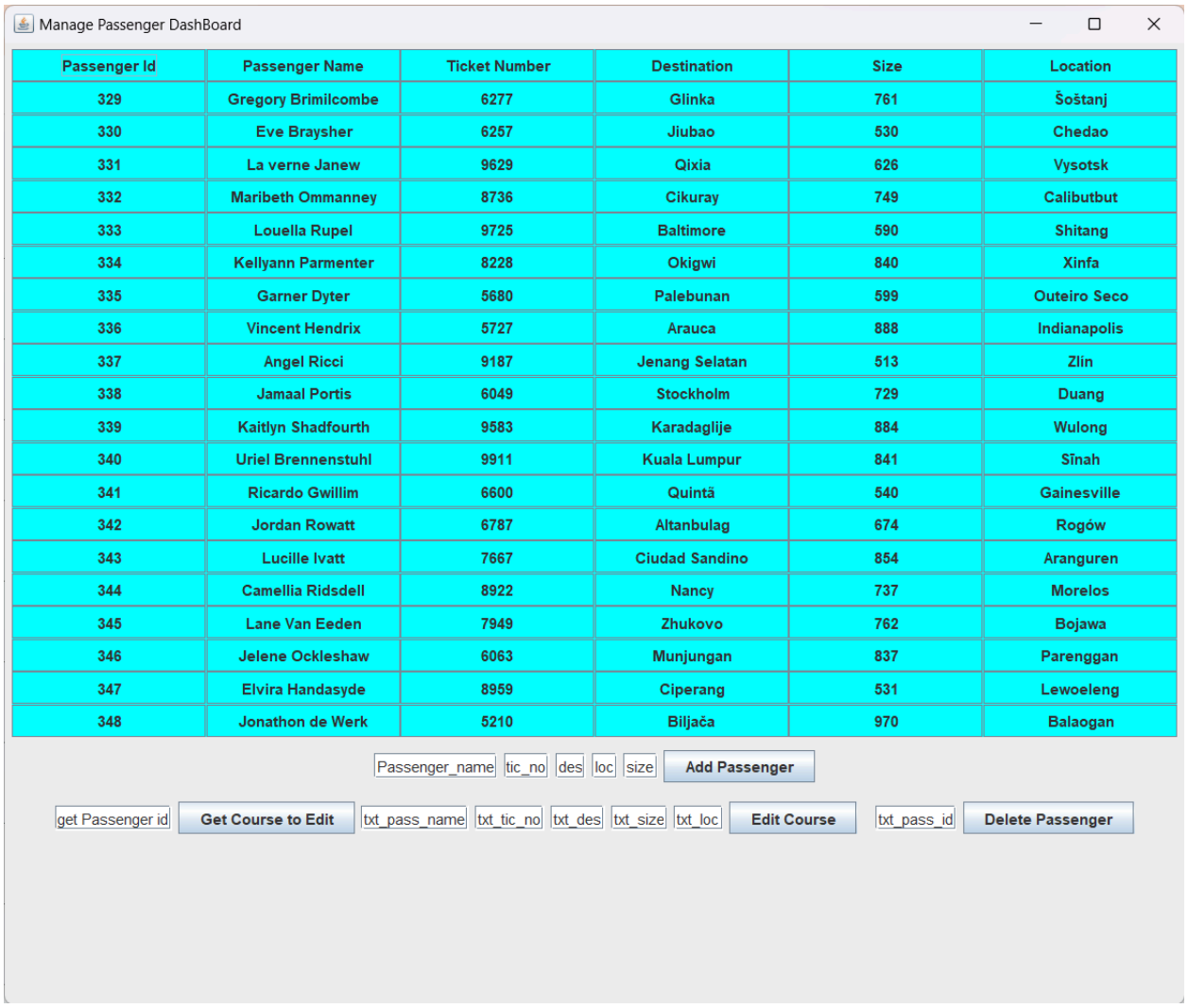
```
package View;

import javax.swing.*;
import java.awt.*;

public class FirstFrame extends JFrame {
  JButton managePassBtn;
  JButton manageAirlineBtn;
  JButton manageTripsBtn;
  JPanel firstPanel;


  FirstFrame()
  {
     super("Main DashBoard");
     managePassBtn = new JButton("Manage Passengers");
     manageAirlineBtn = new JButton("Manage Airlines");
     manageTripsBtn = new JButton("Manage Trips");

     firstPanel = new JPanel();
     firstPanel.setLayout(new GridLayout(3,1,20,20));
     firstPanel.add(managePassBtn);
     firstPanel.add(manageAirlineBtn);
     firstPanel.add(manageTripsBtn);

     add(firstPanel);

     pack();
     setSize(950, 800);
     setVisible(true);
  }

  public void setFirstPanel(JPanel firstPanel) {
     this.firstPanel = firstPanel;
  }

  public void setManageAirlineBtn(JButton
manageAirlineBtn) {
     this.manageAirlineBtn = manageAirlineBtn;
  }

  public void setManagePassBtn(JButton managePassBtn) {
     this.managePassBtn = managePassBtn;
  }

  public void setManageTripsBtn(JButton manageTripsBtn) {
     this.manageTripsBtn = manageTripsBtn;
  }

  public JButton getManageAirlineBtn() {
     return manageAirlineBtn;
  }

  public JButton getManagePassBtn() {
     return managePassBtn;
  }

  public JButton getManageTripsBtn() {
     return manageTripsBtn;
  }

  public JPanel getFirstPanel() {
     return firstPanel;
  }
}
```

#OUTPUT

MainFrame:

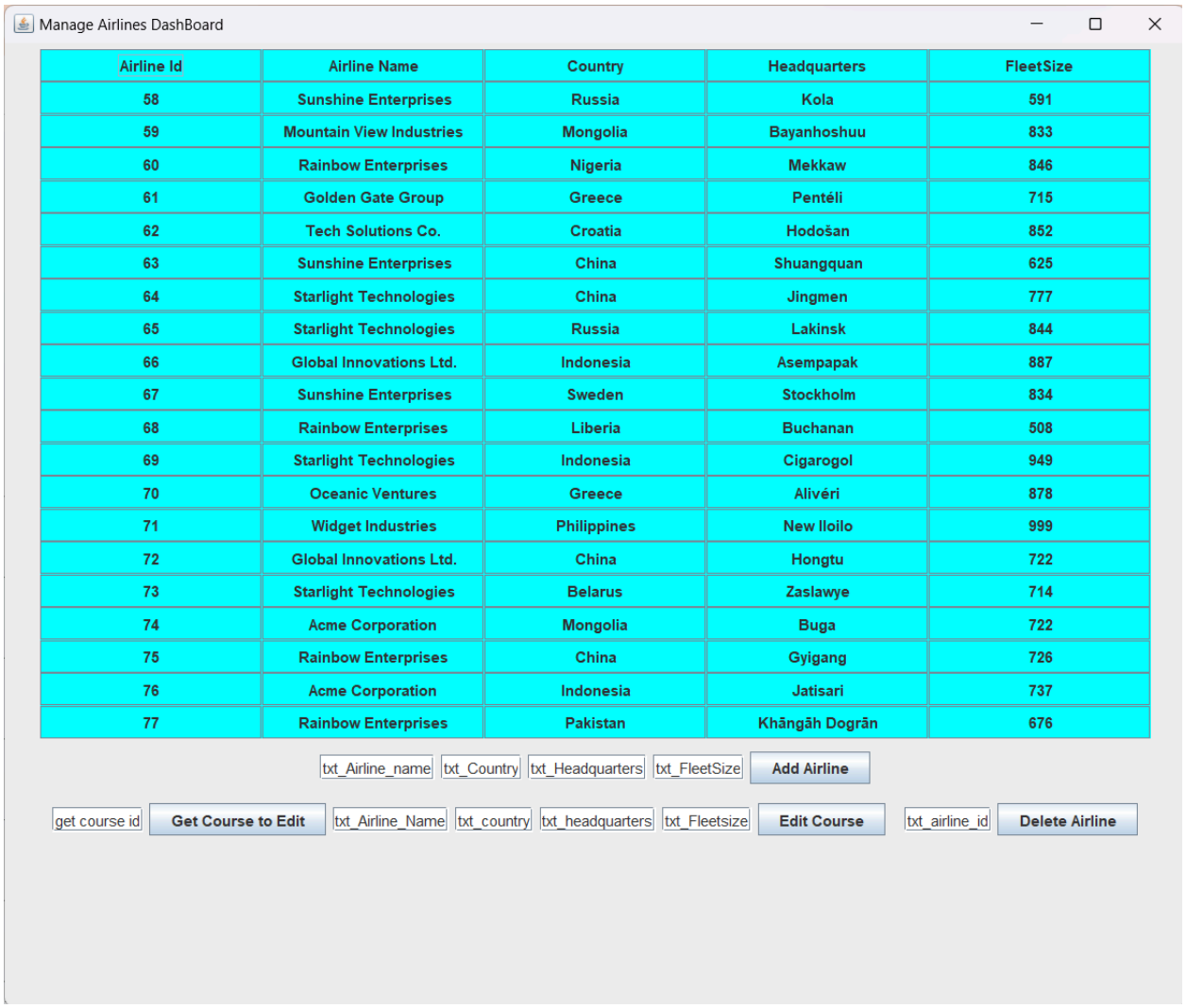


Manage Passenger

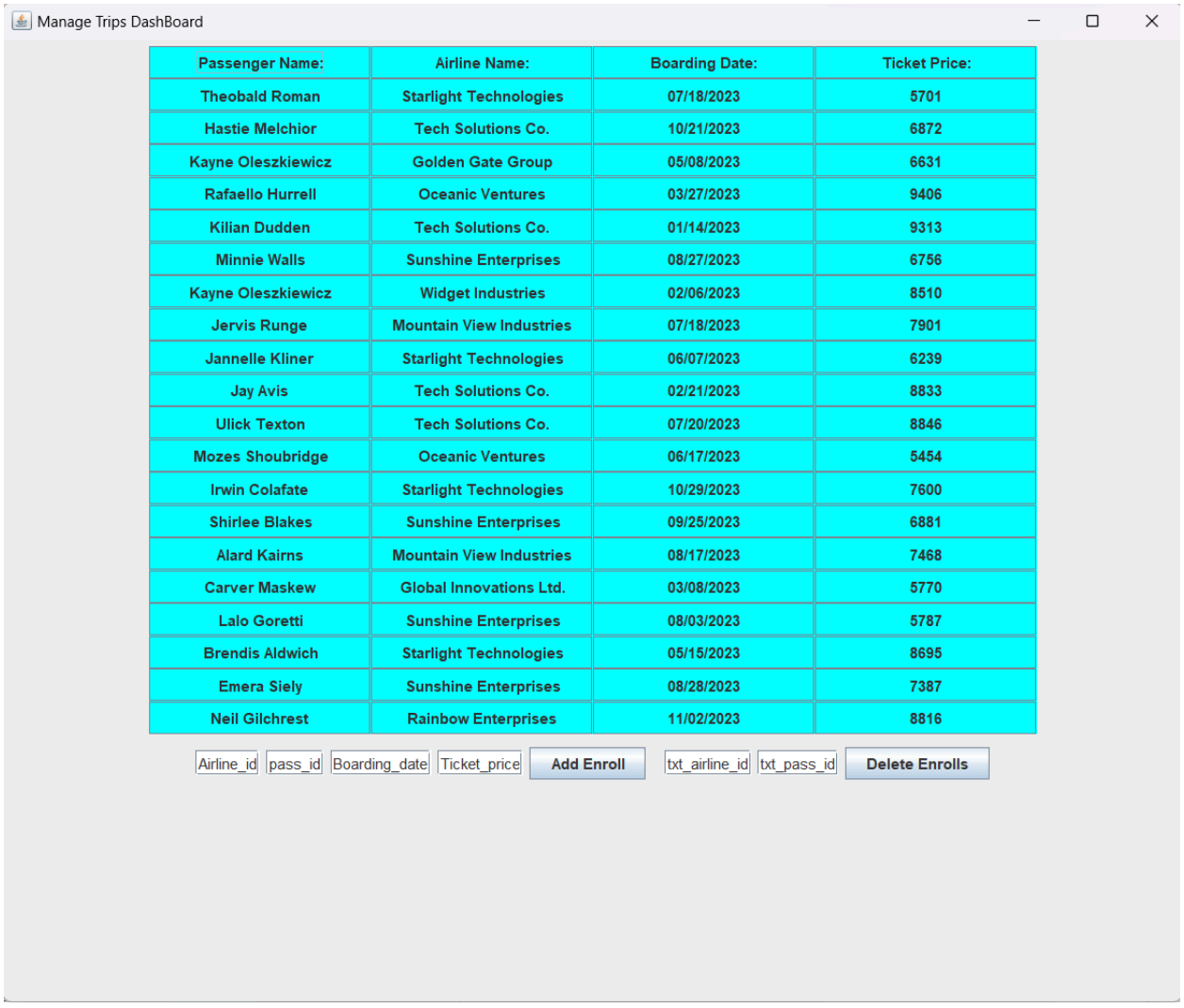


Passenger Id	Passenger Name	Ticket Number	Destination	Size	Location
329	Gregory Brimilcombe	6277	Glinka	761	Šoštanj
330	Eve Braysher	6257	Jiubao	530	Chedao
331	La verne Janew	9629	Qixia	626	Vysotsk
332	Maribeth Ommanney	8736	Cikuray	749	Calibutbut
333	Louella Rupel	9725	Baltimore	590	Shitang
334	Kellyann Parmenter	8228	Okigwi	840	Xinfa
335	Garner Dyter	5680	Palebunan	599	Outeiro Seco
336	Vincent Hendrix	5727	Arauca	888	Indianapolis
337	Angel Ricci	9187	Jenang Selatan	513	Zlín
338	Jamaal Portis	6049	Stockholm	729	Duang
339	Kaitlyn Shadfourth	9583	Karadaglije	884	Wulong
340	Uriel Brennenstuhl	9911	Kuala Lumpur	841	Sīnah
341	Ricardo Gwillim	6600	Quintã	540	Gainesville
342	Jordan Rowatt	6787	Altanbulag	674	Rogów
343	Lucille Ivatt	7667	Ciudad Sandino	854	Aranguren
344	Camellia Ridsdell	8922	Nancy	737	Morelos
345	Lane Van Eeden	7949	Zhukovo	762	Bojawa
346	Jelene Ockleshaw	6063	Munjungan	837	Parenggan
347	Elvira Handasyde	8959	Ciperang	531	Lewoeleng
348	Jonathon de Werk	5210	Biljača	970	Balaogan

Manage Airline:



Manage Trips:



Conclusion

In conclusion, our project has successfully implemented a robust MVC (Model-View-Controller) architecture to manage information related to airlines, passengers, and trips efficiently and systematically. The Model layer encapsulates the data management classes responsible for handling airline, passenger, and trip information, ensuring proper organization and management of data.

The View layer provides user interfaces tailored to display and interact with airline, passenger, and trip data, enhancing user experience and usability. Lastly, the Controller layer orchestrates the interactions between the Model and View layers, ensuring smooth communication and facilitating user actions on the data.

Throughout the development process, we have adhered to best practices in software design and coding standards, ensuring modularity, scalability, and maintainability of the project. By employing concepts such as encapsulation, inheritance, and polymorphism, we have created a flexible and extensible system capable of accommodating future enhancements and modifications with ease.

Furthermore, our project incorporates features such as file handling for data persistence, user input validation, and dynamic UI updates, enhancing its robustness and usability. We have also paid attention to error handling and exception management to ensure the reliability and stability of the application.

Overall, our project represents a successful implementation of MVC architecture in a real-world scenario, demonstrating effective separation of concerns, code organization, and collaboration between different components. It serves as a solid foundation for further development and customization, catering to the needs of diverse users and scenarios in the domain of airport and airline management.

Future Works

1. Enhanced Analytics: Integrate advanced analytics capabilities to provide stakeholders with deeper insights into travel patterns, passenger demographics, and route profitability, enabling better decision-making and planning.

2. Personalization: Implement personalized services and offerings based on passenger preferences, travel history, and loyalty status to enhance customer satisfaction and loyalty.

3. Mobile Compatibility: Develop mobile applications or responsive web interfaces to extend the system's reach and accessibility, allowing users to access features and functionalities on-the-go.

4. Integration with External Systems: Explore opportunities for integrating with external systems, such as reservation systems, airport databases, and flight tracking services, to provide seamless end-to-end travel experiences.

5. Social Media Integration: Integrate social media platforms to enable passengers to share their travel experiences, recommendations, and feedback, fostering community engagement and brand advocacy.

6. Geospatial Visualization: Incorporate geospatial visualization tools to provide interactive maps, route planning, and location-based services, enhancing the exploration and navigation of airports and travel destinations.

7. Machine Learning: Explore the use of machine learning algorithms for predictive analytics, demand forecasting, and anomaly detection, leveraging historical data and real-time inputs to optimize operations and improve service quality.

8. Feedback Mechanisms: Implement feedback mechanisms and sentiment analysis to gather passenger feedback, reviews, and ratings, enabling continuous improvement and refinement of services.

By pursuing these avenues for future work, the airport and airline management system can further enhance operational efficiency, customer satisfaction, and competitiveness in the aviation industry.

REFERENCE

CHAT GPT

JAVATPOINT