AIRPORT MANAGEMENT SYSTEM

A PROJECT REPORT

Submitted by

NISARG [RA2311003030593]
AKANSHA [RA2211003030587]
RAGINI [RA2211003030546]
DAKSH [RA2211003030595]
TANISHKA [RA2311003030589]

Under the guidance of

Ms. Neetu Bansla

(Assistant Professor, Department of Computer Science & Engineering)

SRM
INSTITUTE OF SCIENCE AND TECHNOLOGY
DELHI-NCR CAMPUS, GHAZIABAD (U.P)

SRM INSTITUTE OF SCIENCE & TECHNOLOGY, NCR CAMPUS

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified to be the bonafide record of work done by ***Nisarg,Ragini,Daksh,Akansha & Tansihka*** *of 3th semester 2nd year* ***B.TECH*** *degree course in* ***SRM INSTITUTE OF SCIENCE AND TECHNOLOGY, NCR Campus*** *of Department of Computer Science & Engineering in Database Management Systems Lab, during the academic year 2023-2024.*

SIGNATURE

Ms. Neetu Bansla
Assistant Professor
Computer Science & Engineering

SIGNATURE

Dr. Avneesh Vashistha
Head of the Department
Computer Science & Engineering

Acknowledgement

I would like to express my special thanks of gratitude to my teacher/mentor, Ms Neetu Bansal , for their invaluable guidance and support throughout the project. I am also grateful to my classmates and friends for their assistance in developing this project. Without their help and encouragement, the completion of this project would not have been possible.

Abstract

The Airport Management System project in Java is designed to provide a computerized and user-friendly system for managing the operations at an airport. It automates various processes like flight scheduling, passenger handling, ticket booking, and employee management. The system offers an efficient way to manage flight information, schedule management, and data about passengers and staff, thus helping airport authorities streamline their work and offer better service to passengers.

Introduction

The Airports are critical hubs that connect cities, countries, and continents. Managing airport operations is a complex task that includes handling flight schedules, passengers, ticketing, and other logistics. The Airport Management System is a Java-based solution that streamlines these tasks. This system is aimed at automating most airport management functions to improve accuracy, speed, and reliability. The system is built using Java due to its platform independence, scalability, and strong community support.

Objectives

- The main objectives of the Airport Management System are:
- To automate the management of flight schedules and ticket bookings.
- To handle passenger information efficiently.
- To maintain airport employee records.
- To ensure a seamless and efficient operation for both passengers and airport staff.
- To improve the overall experience of airport management through automation.

Hardware Requirements

Processor: Intel i3 or higher

RAM: 4 GB or more

Hard Disk: 500 GB or more

Monitor: 15-inch or above

Input Devices: Keyboard and mouse

Network: LAN or Wi-Fi connection for server-based implementation

Software Requirements

Operating System: Windows/Linux/MacOS

Programming Language: Java (JDK 8 or higher)

Database: MySQL or any relational database management system (RDBMS)

IDE: Eclipse/NetBeans/IntelliJ IDEA

Web Server: Apache Tomcat (if building a web application)

Frameworks: JavaFX or Swing for GUI, JDBC for database connectivity

Other Tools: MySQL Workbench (for database management), Git (for version control)

Advantages

Efficiency: Automation speeds up various airport operations like ticket booking, flight scheduling, and employee management.

Accuracy: Reduces human errors in flight schedules, passenger details, and other important data.

Data Management: Proper database management ensures secure and well-organized data handling.

Cost-Effective: Reduces operational costs by minimizing the need for manual work.

User-Friendly: Provides a graphical user interface (GUI) for easy navigation and use by both staff and passengers.

Disadvantages

Initial Cost: Setting up the system involves hardware and software expenses.

Maintenance: Regular maintenance and updates are required to keep the system functioning smoothly.

Dependency on Technology: In case of system failure or downtime, the airport may face operational disruptions.

Training: Staff might need training to use the system efficiently, which could lead to additional costs

Procedure

Step 1: System Design:

Define the various modules such as flight management, passenger management, ticketing system, and staff management.

Design the database schema for storing flight, passenger, and employee information.

Step 2: Development:

Implement the application using Java.

Use JavaFX or Swing for the GUI.

Use JDBC for connecting the Java application with the MySQL database.

Design forms for ticket booking, employee registration, and flight scheduling.

Step 3: Database Implementation:

Create a database using MySQL or any RDBMS.

Define tables for passengers, flights, and employees.

Write queries to fetch, insert, update, and delete information as needed.

Step 4: Testing:

Perform unit testing for each module to ensure they work independently.

Perform integration testing to ensure the modules work together seamlessly.

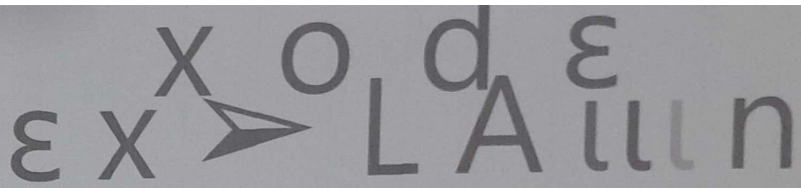
Ensure the application can handle multiple users simultaneously.

Step 5: Deployment :

Deploy the system on a server (if it's a web-based system).

Train the airport staff to use the system effectively.

PORTING LIBRARIES:
AVA
PORT JAVAX.SWING.*;
PORT JAVA.AWT.*;
PORT JAVA.AWT.EVENT.ACTIONEVENT;
PORT JAVA.AWT.EVENT.ACTIONLISTENER;



AVAX.SWING.*`: THIS PACKAGE PROVIDES CLASSES FOR CREATING GRAPHICAL USER INTERFACES.
`JAVA.AWT.*`: THIS PACKAGE PROVIDES CLASSES FOR CREATING WINDOW-BASED APPLICATIONS, INCLUDING
PHICAL USER INTERFACES.
`JAVA.AWT.EVENT.ACTIONEVENT` AND `JAVA.AWT.EVENT.ACTIONLISTENER`: THESE CLASSES ARE USED FOR
DLING EVENTS LIKE BUTTON CLICKS.

ASS DEFINITION:
AVA
BLIC CLASS AIRPORTTICKETBOOKINGSYSTEM {
/ ...

HIS DEFINES A CLASS NAMED `AIRPORTTICKETBOOKINGSYSTEM`.

TANCE VARIABLES:
.VA
/ATE JFRAME FRAME;
/ATE JCOMBOBOX<STRING> FROMCOMBOBOX;
/ATE JCOMBOBOX<STRING> TOCOMBOBOX;
/ATE JTEXTFIELD NAMETEXTFIELD;
/ATE JCOMBOBOX<STRING> GENDERCOMBOBOX;
/ATE JCOMBOBOX<STRING> AGECOMBOBOX;
/ATE JCOMBOBOX<STRING> DEPARTUREDATECOMBOBOX;
/ATE JCOMBOBOX<STRING> DEPARTUREMONTHCOMBOBOX;
/ATE JCOMBOBOX<STRING> DEPARTURETIMECOMBOBOX;
/ATE JBUTTON BOOKBUTTON;

ESE ARE INSTANCE VARIABLES (FIELDS) THAT WILL BE USED TO STORE COMPONENTS OF THE GUI LIKE BUTTONS,
FIELDS, AND COMBO BOXES.

NSTRUCTOR:
.VA
LIC AIRPORTTICKETBOOKINGSYSTEM() {
...

IS IS THE CONSTRUCTOR OF THE CLASS. IT INITIALIZES THE GUI COMPONENTS AND SETS UP THE FRAME.

```
\ME INITIALIZATION:
.VA
ME = NEW JFRAME("AIRPORT TICKET BOOKING SYSTEM");
ME.SETDEFAULTCLOSEOPERATION(JFRAME.EXIT_ON_CLOSE);
ME.SETSIZE(500, 500);
ME.SETLAYOUT(NEW GRIDLAYOUT(15, 2));

IS CREATES A JFRAME (THE MAIN WINDOW) WITH A TITLE, SETS THE DEFAULT CLOSE OPERATION, AND SETS ITS
T ALSO SETS THE LAYOUT TO A GRID LAYOUT WITH 15 ROWS AND 2 COLUMNS.

GO LABEL:
.VA
GEICON ICON = NEW IMAGEICON("./SNAKE GAME/LOGO.PNG");
BEL LOGOLABEL = NEW JLABEL(ICON);
ME.ADD(LOGOLABEL);
ME.ADD(NEW JLABEL());

IS CODE ATTEMPTS TO LOAD AN IMAGE (LOGO) AND CREATE A LABEL TO DISPLAY IT. THE LOGO IS ADDED TO THE
E.
```

BELS AND COMBO BOXES:

```
IAVA
XAMPLE FOR 'FROM' LABEL AND COMBO BOX:
ABEL FROMLABEL = NEW JLABEL("FROM:");
OMCOMBOBOX = NEW JCOMBOBOX<>(NEW STRING[] { "NEW YORK", "LOS ANGELES", "CHICAGO", "SAN
NCISCO" });
AME.ADD(FROMLABEL);
AME.ADD(FROMCOMBOBOX);
```

HIS SECTION ADDS LABELS (E.G., "FROM:") AND CORRESPONDING COMBO BOXES (DROP-DOWN
IUS) FOR VARIOUS INPUTS LIKE DESTINATION, NAME, GENDER, ETC.

XT FIELD:

```
IAVA
ABEL NAMELABEL = NEW JLABEL("NAME:");
METEXTFIELD = NEW JTEXTFIELD();
AME.ADD(NAMELABEL);
AME.ADD(NAMETEXTFIELD);
```

HIS ADDS A LABEL FOR THE NAME AND A TEXT FIELD WHERE THE USER CAN ENTER THEIR NAME.

TONS AND ACTION LISTENERS:

```
VA
KBUTTON = NEW JBUTTON("BOOK FLIGHT");
(STYLING THE BUTTON)
ME.ADD(NEW  JLABEL());
ME.ADD(BOOKBUTTON);

KBUTTON.ADDACTIONLISTENER(NEW ACTIONLISTENER() {
OVERRIDE
IBLIC VOID ACTIONPERFORMED(ACTIONEVENT E) {
/ ...
```

S ADDS A BUTTON LABELED "BOOK FLIGHT" TO THE FRAME AND SETS UP AN ACTION LISTENER TO RESPOND WHEN UTTON IS CLICKED.

TION LISTENER FOR THE BOOK BUTTON:

```
\VA
)KBUTTON.ADDACTIONLISTENER(NEW ACTIONLISTENER() {
OVERRIDE
JBLIC VOID ACTIONPERFORMED(ACTIONEVENT E) {
// ...
```

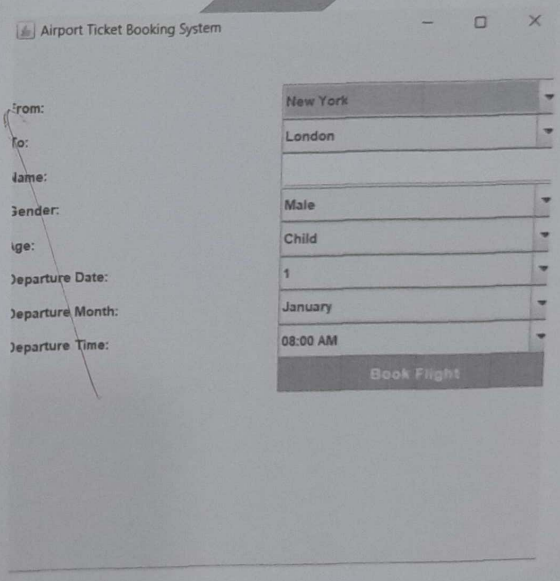
IS CREATES AN ANONYMOUS INNER CLASS THAT IMPLEMENTS THE `ACTIONLISTENER` INTERFACE. IT OVERRIDES THE `)NPERFORMED` METHOD, WHICH WILL BE CALLED WHEN THE BUTTON IS CLICKED.

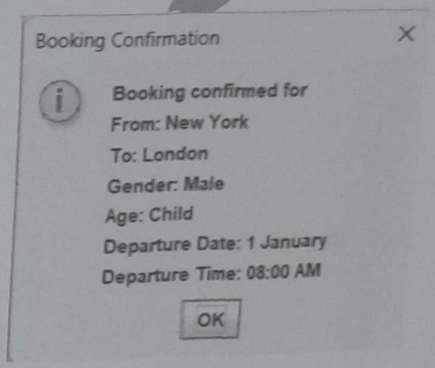
```
CTION PERFORMED:
JAVA
OVERRIDE
IBLIC VOID ACTIONPERFORMED(ACTIONEVENT E) {
STRING FROM = (STRING) FROMCOMBOBOX.GETSELECTEDITEM();
STRING TO = (STRING) TOCOMBOBOX.GETSELECTEDITEM();
STRING NAME = NAMETEXTFIELD.GETTEXT();
STRING GENDER = (STRING) GENDERCOMBOBOX.GETSELECTEDITEM();
STRING AGE = (STRING) AGECOMBOBOX.GETSELECTEDITEM();
STRING DEPARTUREDATE = (STRING) DEPARTUREDATECOMBOBOX.GETSELECTEDITEM();
STRING DEPARTUREMONTH = (STRING) DEPARTUREMONTHCOMBOBOX.GETSELECTEDITEM();
STRING DEPARTURETIME = (STRING) DEPARTURETIMECOMBOBOX.GETSELECTEDITEM();

STRING MESSAGE = "BOOKING CONFIRMED FOR " + NAME + "\NFROM: " + FROM + "\NTO: " + TO +
    "\NGENDER: " + GENDER + "\NAGE: " + AGE +
    "\NDEPARTURE DATE: " + DEPARTUREDATE + " " + DEPARTUREMONTH +
    "\NDEPARTURE TIME: " + DEPARTURETIME;
JOPTIONPANE.SHOWMESSAGEDIALOG(FRAME, MESSAGE, "BOOKING CONFIRMATION",
TIONPANE.INFORMATION_MESSAGE);


HIS METHOD IS CALLED WHEN THE "BOOK FLIGHT" BUTTON IS CLICKED. IT RETRIEVES THE SELECTED
JES FROM VARIOUS COMPONENTS AND CONSTRUCTS A MESSAGE. THE MESSAGE IS THEN DISPLAYED
DIALOG BOX.
```

. UTILITY METHODS:
```JAVA
PRIVATE STRING[] GETDAYSOFMONTH() { ... }
PRIVATE STRING[] GETMONTHS() { ... }
PRIVATE STRING[] GETDEPARTURETIMES() { ... }
```
- THESE METHODS ARE USED TO POPULATE THE COMBO BOXES WITH APPROPRIATE OPTIONS.

. MAIN METHOD:
```JAVA
PUBLIC STATIC VOID MAIN(STRING[] ARGS) {
    SWINGUTILITIES.INVOKELATER(NEW RUNNABLE() {
        @OVERRIDE
        PUBLIC VOID RUN() {
            NEW AIRPORTTICKETBOOKINGSYSTEM();
        }
    });
}
```

ROJECT
UTPUT





Conclusion

The Airport Management System is a useful tool for improving airport operations by automating many manual tasks. It enhances the experience for passengers, reduces human errors, and allows airport authorities to focus on better service delivery. By integrating flight scheduling, passenger handling, and staff management into one unified system, this project ensures a smooth and reliable operational process. With its user-friendly interface and strong backend capabilities, it is a valuable asset to any airport.

You can expand this project with real implementation details for the Java code, GUI design, database schema, and more depending on how in-depth you want to go.