**SIMPLE DRAWING APP**

**Abstract**

The Simple Drawing App is a user-friendly digital platform designed to facilitate effortless sketching and note-taking for individuals of all ages. Emphasizing minimalism and ease of use, the app provides a clean interface that allows users to focus solely on their creativity without the distraction of complex tools or menus. Whether it's jotting down quick ideas, illustrating concepts, or engaging in casual doodling, the app serves as a versatile canvas for spontaneous expression.Key features include a variety of brush sizes, color selection options, and an intuitive undo/redo functionality, ensuring a seamless drawing experience.

**Introduction**

Airline reservation systems play a crucial role in modern air travel, enabling airlines to manage seat bookings, cancellations, and waiting lists efficiently. With the increase in air travel demand, manual reservation methods have become impractical and error-prone. A computerized airline reservation system automates the process, providing convenience to both passengers and airline staff.

This project implements a simple airline reservation system in C, demonstrating the fundamental concepts of data structures such as linked lists and queues. The system allows users to book tickets, cancel bookings, and manage waiting lists when the flight reaches full capacity. Booked passengers are stored in a linked list, while passengers waiting for a seat are maintained in a queue, ensuring a first-come, first-served approach.

Key features of the system include:

1. Ticket Booking: Allows passengers to reserve seats. If seats are available, the booking is confirmed; otherwise, the passenger is added to the waiting list.

2. Ticket Cancellation: Passengers can cancel their tickets, which automatically moves the first passenger from the waiting list to the booked list.

3. Display Options: The system can display all booked passengers and the current waiting list, providing a clear overview of the flight occupancy.

This project demonstrates how dynamic data structures can be utilized to simulate real-world systems efficiently. By managing seat allocation and waiting lists programmatically, the system reduces human errors and improves overall operational efficiency. It also serves as a practical example for students learning data structures and basic programming concepts in C.

**Existing System**

In traditional airline reservation systems, bookings are often managed manually through counters or over the phone. This manual approach has several limitations:

1. Time-Consuming: Staff must manually record passenger details, check seat availability, and issue tickets, which increases waiting time.

2. Error-Prone: Manual entry can lead to mistakes in passenger information, duplicate bookings, or seat allocation errors.

3. Limited Accessibility: Passengers need to physically visit the airline office or contact a reservation agent, making the process inconvenient.

4. No Automated Waiting List: In the event of full flights, handling waiting lists manually is difficult, and the first-come, first-served rule may not be accurately followed.

5. Data Management Issues: Storing, searching, and updating passenger information is cumbersome without a computerized system.

Due to these inefficiencies, airlines face challenges in maintaining accurate booking records and providing a smooth experience for passengers.

**Proposed System**

The proposed airline reservation system is a computerized, automated solution that addresses the shortcomings of the existing system. Using the C programming language and data structures like linked lists and queues, the system efficiently manages bookings and waiting lists.

Key improvements include:

1. Automated Booking: Passengers can book tickets quickly, and the system checks seat availability in real-time.

2. Waiting List Management: If the flight is full, the system automatically adds passengers to a waiting list, ensuring fairness and first-come, first-served allocation.

3. Efficient Cancellation Handling: When a passenger cancels a booking, the first passenger from the waiting list is automatically moved to the booked list.

4. Data Accuracy: Storing passenger information in linked lists reduces errors and enables efficient searching, updating, and deletion.

5. User-Friendly Interface: The menu-driven interface allows airline staff to perform all operations easily, including booking, canceling, and viewing passenger details.

The proposed system streamlines the airline booking process, reduces manual errors, and enhances passenger satisfaction. It also provides a practical example of how data structures can be applied to real-world problems.

**Software Requirements**

The software requirements define the environment needed to develop and run the airline reservation system effectively. The proposed system requires minimal but essential software tools:

1. Operating System: Windows, Linux, or macOS – any OS capable of running a C compiler.

2. Programming Language: C language, used for implementing core functionality such as booking, cancellation, and waiting list management.

3. Compiler/IDE:

Code::Blocks or Dev-C++ for Windows.

GCC Compiler for Linux.

Visual Studio Code with C extensions for cross-platform development.

4. Text Editor: Notepad++, Sublime Text, or any standard text editor to write and edit source code.

5. Command Line Interface (CLI): Required to compile and execute C programs if an IDE is not used.

The software setup ensures the program runs efficiently and supports menu-driven interactions for airline staff.

**Hardware Requirements**

The hardware requirements specify the minimum system configuration needed to run the airline reservation system smoothly:

1. Processor: Intel Pentium or equivalent (minimum 1 GHz).

2. RAM: 1 GB or more, sufficient for compiling and running a simple C program.

3. Storage: At least 100 MB free disk space to store compiler, IDE, and project files.

4. Display: Monitor with 800×600 resolution or higher for clear visualization of the console interface.

5. Input Devices: Keyboard and mouse for interacting with the program.

6. Power Supply: Reliable source to ensure uninterrupted system operation during booking management.

These requirements are minimal because the system is console-based and does not require high-performance resources.

**Aim:**

The aim of this project is to build a Java-based desktop application to AWT/Swings,oop,util package,exception handling . It also provides freehand drawing using mouse,color and stroke size selection

**Sourcecode:**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

import java.util.ArrayList;

public class SimpleDrawingApp extends JFrame {

private Color currentColor = Color.BLACK;

private int currentStroke = 2;

private final DrawArea drawArea = new DrawArea();

public SimpleDrawingApp() {

setTitle("Simple Drawing App");

setSize(800, 600);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLocationRelativeTo(null);

JPanel topPanel = new JPanel();

// Color chooser

JButton colorButton = new JButton("Choose Color");

colorButton.addActionListener(e -> {

Color selectedColor = JColorChooser.showDialog(null, "Choose a color", currentColor);

if (selectedColor != null) {

currentColor = selectedColor;

drawArea.setColor(currentColor);

}

});

// Stroke size selector

JLabel strokeLabel = new JLabel("Stroke:");

String[] strokeOptions = {"1", "2", "4", "6", "8", "10"};

JComboBox<String> strokeCombo = new JComboBox<>(strokeOptions);

strokeCombo.setSelectedItem("2");

strokeCombo.addActionListener(e -> {

try {

currentStroke = Integer.parseInt((String) strokeCombo.getSelectedItem());

drawArea.setStroke(currentStroke);

} catch (NumberFormatException ex) {

JOptionPane.showMessageDialog(this, "Invalid stroke size.", "Error", JOptionPane.ERROR\_MESSAGE);

}

});

topPanel.add(colorButton);

topPanel.add(strokeLabel);

topPanel.add(strokeCombo);

add(topPanel, BorderLayout.NORTH);

add(drawArea, BorderLayout.CENTER);

}

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> new SimpleDrawingApp().setVisible(true));

}

// Custom panel for drawing

static class DrawArea extends JPanel {

private final ArrayList<Line> lines = new ArrayList<>();

private Point startPoint = null;

private Color color = Color.BLACK;

private int stroke = 2;

public DrawArea() {

setBackground(Color.WHITE);

MouseAdapter mouseAdapter = new MouseAdapter() {

public void mousePressed(MouseEvent e) {

startPoint = e.getPoint();

}

public void mouseDragged(MouseEvent e) {

Point endPoint = e.getPoint();

if (startPoint != null) {

lines.add(new Line(startPoint, endPoint, color, stroke));

startPoint = endPoint;

repaint();

}

}

public void mouseReleased(MouseEvent e) {

startPoint = null;

}

};

addMouseListener(mouseAdapter);

addMouseMotionListener(mouseAdapter);

}

public void setColor(Color c) {

this.color = c;

}

public void setStroke(int s) {

this.stroke = s;

}

protected void paintComponent(Graphics g) {

super.paintComponent(g);

Graphics2D g2 = (Graphics2D) g;

for (Line line : lines) {

g2.setColor(line.color);

g2.setStroke(new BasicStroke(line.stroke));

g2.drawLine(line.start.x, line.start.y, line.end.x, line.end.y);

}

}

static class Line {

Point start, end;

Color color;

int stroke;

Line(Point s, Point e, Color c, int stroke) {

this.start = s;

this.end = e;

this.color = c;

this.stroke = stroke;

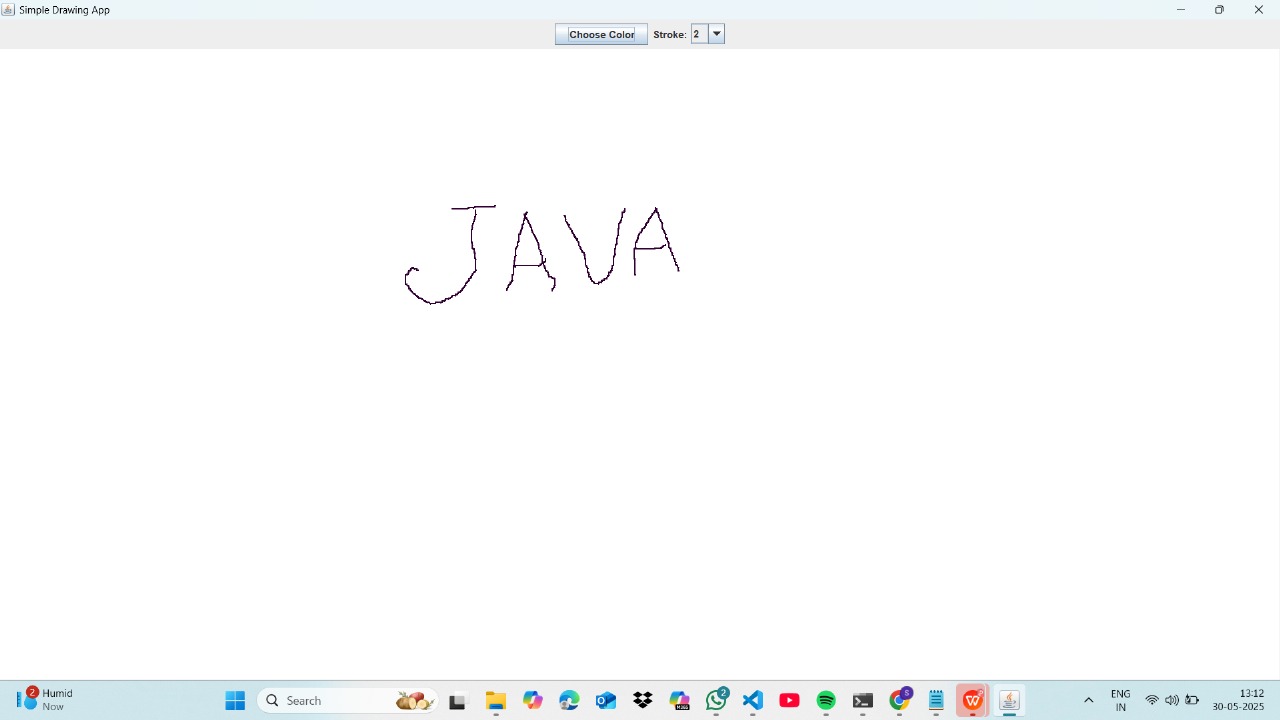
}

}

}

}

**Output:**



**Conclusion:**

The Simple Drawing App offers an intuitive platform for users to create sketches and doodles effortlessly. With features like customizable brush sizes, vibrant color palettes, and support for various formats such as PNG, JPG, and SVG, it caters to both beginners and seasoned artists. The app functions seamlessly offline, ensuring creativity isn't hindered by connectivity issues. Its user-friendly interface and lightweight design make it an ideal choice for spontaneous artistic expression anytime, anywhere.