Republic of the Philippines
SURIGAO DEL NORTE STATE UNIVERSITY
Narciso Street, Surigao City 8400, Philippines

"For Nation's Greater

In Partial Fulfillment of the Requirements for the

CS 223 - Object-Oriented Programming

Mobile Marketing Analysis for Student

Presented to:

Dr. Unife O. Cagas
Professor

Prepared by:

Olang, Nilo Jr.
Student

BSCS-2A: Computer Science

May. 2024

Project Description:

Mobile devices have become an essential part of our daily lives, particularly among students who rely on them for communication, education, and entertainment. Understanding the student mobile device usage preferences and trends is crucial for mobile related businesses. This project proposal outlines an analysis for student mobile device preferences with the goal of providing valuable insights for marketing strategies.

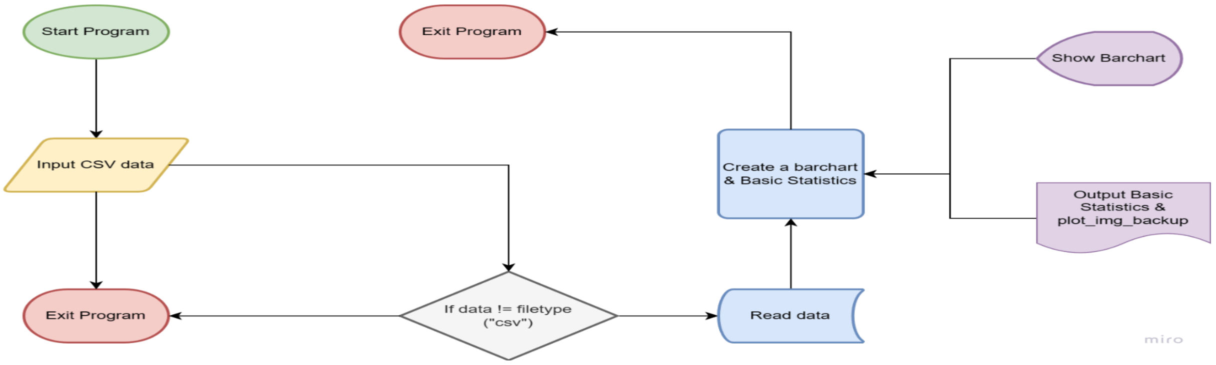
Objectives:

- To analyze survey data collected from students regarding their preferences in mobile devices.
- To identify popular brands, models, and hardware specifications among students.
- To generate reports and visualizations that provide actionable insights for mobile marketing strategies.
- To create a user-friendly interface for uploading survey data and viewing analysis results.

Importance and Contribution:

This project for student mobile device preferences serves for decision-making across various sectors for example in Business, Educational and Developers. By implementing this on consumer behavior and technological trends, it contributes to the advancement of mobile technology, educational strategies, and regulatory frameworks, making the digital landscape in a manner that benefits both students and society.

Flowchart of the Program:


Start Program
Exit Program
Show Barchart
Input CSV data
Create a barchart & Basic Statistics
Output Basic Statistics & plot_img_backup
Exit Program
If data != filetype ("csv")
Read data
miro

Hardware & Software Used:

Hardware:

- Computer or Laptop

Software:

- Python 3.11
- Vscode

Principles of Object Oriented Programming:

1. **Class and Object:**

```
# Concrete implementation of the survey class
class StudentSurvey(AbstractSurvey):  # Class (Inheritance)
    def __init__(self, dataset_path):
        self.dataset_path = dataset_path
        # Load survey data from a CSV file
        self.data = self.load_data()  # Object
```

StudentSurvey is a concrete implementation of the survey class. It inherits from AbstractSurvey and initializes its attributes in the constructor. Instances of StudentSurvey are created as objects.

2. **Inheritance:**

```
# Concrete implementation of the survey class
class StudentSurvey(AbstractSurvey):  # Class (Inheritance)
    def __init__(self, dataset_path):
        # Implementation details...
```

The StudentSurvey class inherits from the AbstractSurvey class. This demonstrates inheritance, where StudentSurvey is a subclass of AbstractSurvey, inheriting its methods and properties.

3. Abstract:

```
# AbstractSurvey is an abstract class, and StudentSurvey is its concrete implementation.
class AbstractSurvey(ABC):  # Class (Abstract class)
    @abstractmethod
    def analyze_data(self):  # Abstract method
        pass
```

AbstractSurvey is an abstract class defining the structure of survey objects. It contains an abstract method analyze_data() which must be implemented by its subclasses. By making analyze_data() abstract, AbstractSurvey defines a contract that all subclasses must fulfill, ensuring that they provide functionality for analyzing survey data.

4. Encapsulation:

```
# Function to exit the program
def ext_program():
    return exit()
```

The ext_program() function is one of the example of encapsulates its a functionality of exiting the program. Encapsulation refers to bundling the data (attributes) and methods (functions) that operate on the data into a single unit, which in this case is a class ext_program()

5. Polymorphism:

```
def analyze_data(self):
  if self.data is None:
    return

    # Print basic statistics about the survey data
    print("Basic Statistics:")
```

```
# And the rest of the code…
```

analyze_data() method demonstrates polymorphism that depend on the specific implementation in StudentSurvey. This method can be overridden in subclasses to provide different implementations while maintaining the same interface. The implementation of analyze_data() in StudentSurvey allows instances of StudentSurvey to respond to calls to analyze_data() in a way that's specific to the behavior defined in StudentSurvey.

Code:

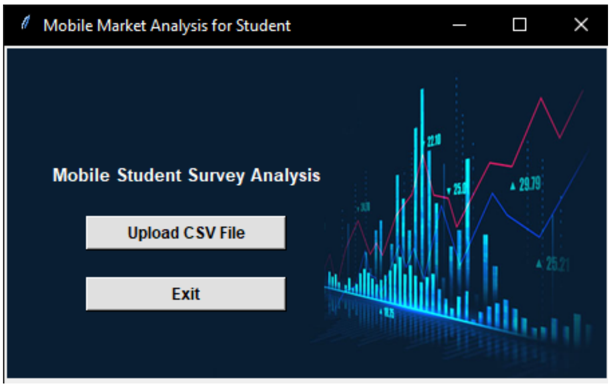
```
import tkinter as tk

import pandas as pd

import matplotlib.pyplot as plt

from tkinter import filedialog, messagebox

from PIL import Image, ImageTk

from abc import ABC, abstractmethod


# Abstract class defining the structure of survey objects

# The AbstractSurvey and StudentSurvey classes represent the concept of surveys in the code.

# AbstractSurvey is an abstract class, and StudentSurvey is its concrete implementation.

class AbstractSurvey(ABC):  # Class (Abstract class)

    @abstractmethod

    def analyze_data(self):  # Abstract method

        pass


# Concrete implementation of the survey class

# StudentSurvey inherits from AbstractSurvey, demonstrating inheritance.

# The AbstractSurvey class contains an abstract method analyze_data, which must be implemented by its subclasses.

class StudentSurvey(AbstractSurvey):  # Class (Inheritance)

    def __init__(self, dataset_path):

        self.dataset_path = dataset_path

        # Load survey data from a CSV file

        self.data = self.load_data()  # Object
    # Method to load survey data from a CSV file
    def load_data(self):
        try:
            # Read CSV file into a DataFrame
            data = pd.read_csv(self.dataset_path)
            return data
        except FileNotFoundError:
            print(f"Error: File '{self.dataset_path}' not found.")
            return None

    # Method to analyze the survey data and generate visualizations
    # The analyze_data method is polymorphic, as its behavior varies depending on the
specific implementation in StudentSurvey.
    def analyze_data(self):
        if self.data is None:
            return

        # Print basic statistics about the survey data
        print("Basic Statistics:")
        # Create subplots for visualization
        fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(5, 8))

        # Plot count of students for each existing brand
        existing_brand_count = self.data['Exst_brand'].value_counts()
        print("\nExisting Brand:")
        print(existing_brand_count)
        existing_brand_count.plot(kind='bar', ax=ax1)
        ax1.set_title('Existing Brand')
        ax1.set_xlabel('Brand')
        ax1.set_ylabel('Count')
        ax1.tick_params(axis='x', labelrotation=0)

        # Plot count of students for each preferred brand
        prefer_brand_count = self.data['Prefer_brand'].value_counts()
        print("\nPreferred Brand:")
        print(prefer_brand_count)
        prefer_brand_count.plot(kind='bar', ax=ax2)
        ax2.set_title('Preferred Brand')
        ax2.set_xlabel('Brand')
        ax2.set_ylabel('Count')
        ax2.tick_params(axis='x', labelrotation=0)

        # Adjust layout to prevent overlap
        plt.tight_layout()

        # Save plots as image files
        self.save_plot(ax2, 'brand_count_plot.png')

        # Show the plots
        plt.show()

    # Method to save a plot as an image file
    def save_plot(self, ax, filename):
        fig = ax.get_figure()
        fig.savefig(filename)

    # Method to generate a report and save it as a text file
    def generate_report(self):
        if self.data is None:
            return

        # Generate a report and save it as a text file
        report_text_filename = "survey_report.txt"
        with open(report_text_filename, 'w') as report_file:
            report_file.write("Survey Report\n\n")

            report_file.write("Basic Statistics:\n")
            report_file.write(str(self.data.describe().round(3)) + '\n\n')

            report_file.write("Count of Students for Each Existing Brand:\n")
            existing_brand_count = self.data['Exst_brand'].value_counts()
            report_file.write(str(existing_brand_count.reset_index(drop=True)) + '\n\n')

            report_file.write("Count of Students for Each Preferred Brand:\n")
            prefer_brand_count = self.data['Prefer_brand'].value_counts()
            report_file.write(str(prefer_brand_count.reset_index(drop=True)) + '\n\n')

            report_file.write("Plot of Preferred Brand Count saved as 'brand_count_plot.png'\n")

        print(f"Report generated. Text saved as '{report_text_filename}'.")
```

```
# Function to browse and upload a CSV file for analysis
# Objects of the StudentSurvey class are created in the browse_file function.
def browse_file():
    file_path = filedialog.askopenfilename(filetypes=[("CSV files", "*.csv")])
    if file_path:
        survey = StudentSurvey(file_path)  # Object
        survey.analyze_data()
        survey.generate_report()

# Function to exit the program
def ext_program():  # The ext_program function encapsulates the functionality of exiting the program.
    return exit()

# Main function to create the GUI window and handle user interaction
def main():
    width, height = 450, 250
    root_window = tk.Tk()
    root_window.title("Mobile Market Analysis for Student")
    root_window.minsize(width, height)
    root_window.configure(bg='white')  # Set the background color of the window to white

    # Load image
    image = Image.open("background.jpg")
    image = image.resize((width, height), Image.LANCZOS)
    bg_image = ImageTk.PhotoImage(image)
    # Create a canvas that fills the window
    canvas = tk.Canvas(root_window, width=width, height=height)
    canvas.pack(fill="both", expand=True)



    # Add the image to the canvas
    canvas.create_image(0, 0, image=bg_image, anchor="nw")

    # Frame to position widgets within the image
    main_windows = tk.Frame(canvas, bg='#091E33', width=10, height=10)
    main_windows.place(relx=0.3, rely=0.3, anchor='n')  # Adjust positioning as needed

    # Welcome message (placed within the frame)
    label = tk.Label(main_windows, text="Mobile Student Survey Analysis", font=("Arial", 10, "bold"), bg='#091E33', fg='white')
    label.pack(pady=10)

    # Button styling
    button_style = {'font': ("Arial", 9,"bold"), 'bg': '#e0e0e0', 'activebackground': '#cccccc'}

    # Upload CSV File button
    upload_button = tk.Button(main_windows, text="Upload CSV File", command=browse_file, width=20, **button_style)
    upload_button.pack(pady=10)

    # Exit button
    ext_button = tk.Button(main_windows, text="Exit", command=ext_program, width=20, **button_style)
    ext_button.pack(pady=10)

    root_window.mainloop()

if __name__ == "__main__":
    print("Noted:")
    print("CSV Format: Name, Age, Gender, Exst_brand, Exst_model, Prefer_brand, Prefer_model")
    messagebox.showinfo("NOTE!", "CSV Format: \n\nName, Age, Gender, Exst_brand, Exst_model, Prefer_brand, Prefer_model")
    main()
```

User Guide:

Step 1:

- Start the program by running the Python script or executable file (Vscode).

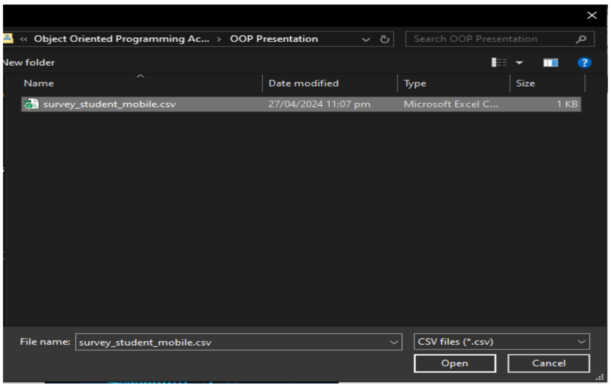


Mobile Market Analysis for Student

Mobile Student Survey Analysis

Step 2:

- Upload your csv dataset and ensure that your dataset is in CSV format.
- The dataset should contain the following attributes: "Name", "Age", "Gender", "Exst_brand", "Exst_model", "Prefer_brand", "Prefer_model".
- If the dataset does not meet the required format or attributes, the program may not function correctly.



« Object Oriented Programming Ac... › OOP Presentation

Search OOP Presentation

Name	Date modified	Type	Size
survey_student_mobile.csv	27/04/2024 11:07 pm	Microsoft Excel C...	1 KB

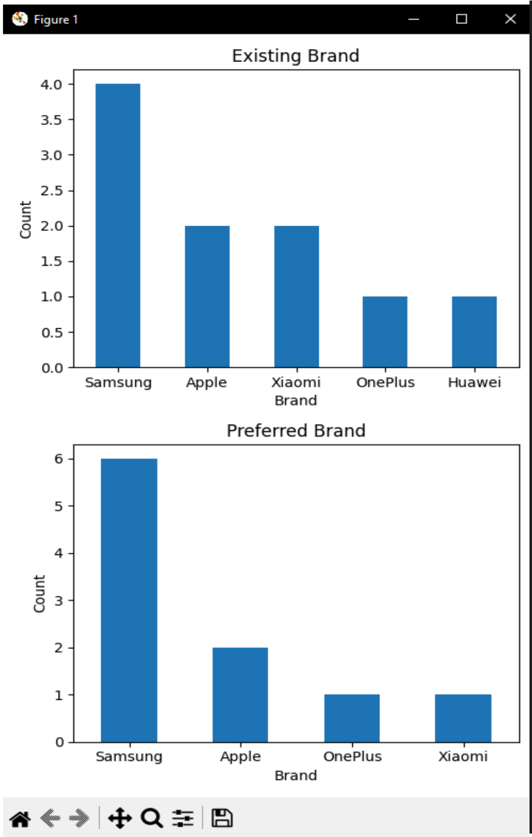
File name: survey_student_mobile.csv

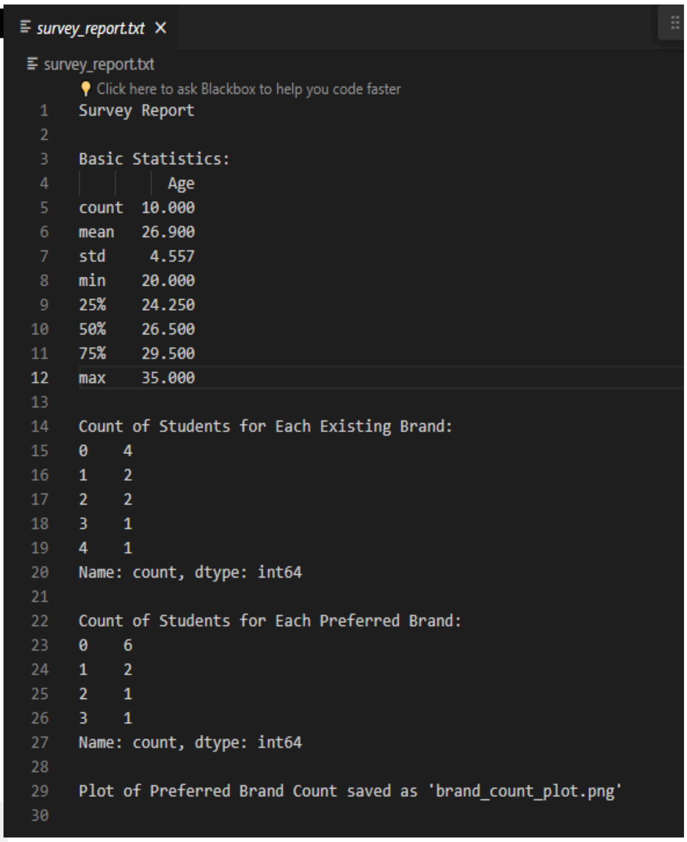
CSV files (*.csv)

Step 3:

- After uploading the dataset, the program will analyze the data and display basic statistics.
- These statistics will be saved in a text file named "survey_report.txt".
- The program will generate a bar chart showing the count of users for each existing brand.
- The bar chart will be saved as an image file named "brand_count_plot.png" for backup purposes.

Output:





```
Survey Report

Basic Statistics:
            Age
count  10.000
mean   26.900
std     4.557
min    20.000
25%    24.250
50%    26.500
75%    29.500
max    35.000

Count of Students for Each Existing Brand:
0    4
1    2
2    2
3    1
4    1
Name: count, dtype: int64

Count of Students for Each Preferred Brand:
0    6
1    2
2    1
3    1
Name: count, dtype: int64

Plot of Preferred Brand Count saved as 'brand_count_plot.png'
```

Description:

This program analyzes student preferences in mobile devices, presenting the results through a user-friendly GUI and generating insightful reports for business and also for education. Program concepts such as object-oriented design, graphical interface development, data analysis, and error handling is a valuable tool for understanding and interpreting survey data related to mobile technology preferences among students. The necessary libraries such as tkinter for GUI development, pandas for data manipulation, matplotlib for data visualization, filedialog for file handling, and PIL for image processing.

Conclusion:

The program can provide valuable insights into student preferences in mobile devices through data gathering and visualization. By understanding these preferences, business companies can develop more targeted and effective marketing strategies. I believe that this program has the potential to make a significant impact in the mobile industry to improve the student and also the people that are using their mobile phone.

Reference:

Reading Dataset	https://www.w3schools.com/python/pandas/default.asp
Plotting Dataset	https://www.w3schools.com/python/pandas/pandas_plotting.asp https://www.w3schools.com/python/matplotlib_intro.asp
GUI guide	https://www.pythontutorial.net/tkinter/
Abstract Sources	https://www.geeksforgeeks.org/abstract-classes-in-python/
Image Load	https://www.geeksforgeeks.org/loading-images-in-tkinter-using-pil/