**MAPÚA MALAYAN COLLEGES MINDANAO**

**IT101-2L COMPUTER PROGRAMMING CONCEPT 2**
**(LABORATORY)**

# TRANSACTIONAL PROCESSING SYSTEM: PRODUCT ORDER TRACKING SYSTEM (JAVA PROTOTYPING)

In Partial Fulfillment of the Requirements in

IT101-2 – Computer Programming Concepts 2

Mayo, Jonathan Lance S.

Villegas, Allyza Fe E.

**Bachelor of Science in Information Systems**

# INTRODUCTION

In recent years, the move to adopt a more digitalized business model is becoming more of a practical approach. This move led to major changes in the business industry. The introduction of innovative Business Models led to new product and service offerings while extending company relationships with customers and employees (Rachinger et.al., 2018). The transformation can also have a large effect as the digital adoption of Micro, Small, and Medium Enterprises (MSMEs) largely increased the viability of the existing business and increased its output on the economy in terms of business operations, market operations, and product distributions (PwC Philippines). However, not all transition processes are due to the drive to perform better but rather a necessity, like how businesses implemented Cashless payment at the start of the pandemic but kept after it died down due to its potential benefits.

## *PROBLEM SCENARIO*

The chosen business in question is a private business named: “Dorothy’s Cakes and Pastries” which is a business that provides baked products such as cakes, pastries, and desserts. Furthermore, it provides catering services for various events and has customized orders based on customer preferences. The business runs on a 4-person team, with the owner being the head baker/pastry chef while the co-operator usually handles the logistical side and also aids in high-demand operations. Based on the interviews, the team has found several deficiencies with the current business model and operations which hinder operating capacity and efficiency. The problems found can be easily fixed by implementing various technologies made using basic Java applications.

## *SUSTAINABLE DEVELOPMENT GOALS (SDGs)*

As the world pushes for more sustainable solutions to addressing various problems. The team addresses the key issues by solving related problems in existing solutions. These solutions include:

***SDG 12**:* The project in question addresses the problem of decent work and economic growth. By implementing a digital ordering-tracking system, it reduces the dependencies on paper and ink which consumes natural resources that can be avoided through transitioning transactions digitally resulting in a reduced carbon footprint (Team, 2024).

***SDG 9**:* While the project addresses more on small and medium-sized enterprises (SMEs), digitalization improves overall productivity, efficiency, and overall contribution to society, reducing waste and increasing economic contribution (Yip, 2023).

## *PROJECT OBJECTIVES*

The project objectives are based on the given criteria and problems provided by the client as to the current nature of the business' operations. This project aims to provide:

-***Digital Order Tracking System*.** This solution aims to create a platform that the business operator can use to record, keep, and track all business transactions which includes sales, inventory, profits/revenue, and lastly, can provide a datasheet that the owner can use to modify business practices.

-**ease of order collection.** The project aims to answer the problem of order collection which often causes problems when using traditional means (I.e. Physical data/paper written).

**-easy order manipulation.** The project aims to provide a platform that the user can manipulate existing orders based on customer changes and feedback.

### *TARGET MARKET*

The project aims to provide Small to medium Enterprises and Businesses (SMEs & SMBs) with a lightweight portable dedicated order tracking system that contains information with regards to the product in mind. The basic implementation of various features allows for a more open-source modification to the program which can be modified to cater to specific conditions based on the business applied. The project can be implemented in various businesses such as cakes and pastries, catering services, custom job orders, and many more.

### *SIMILAR APPLICATION*

This application is a product in-between a proper Point-Of-Sales (POS) such as those used in various establishments such as fast food and stores, and of a calendar application with tracking such as Google calendars. The basic implementation of the system places all of the functionalities in a single area which eases complexity and is easier to manage but at the cost of features. It contains a built-in database that saves orders on a Txt file, however, this is shared with the display which contains the current orders. The specifications of the application is based on various conditions stated by the client which results in a hybrid system with various functionalities borrowed from various systems catered towards the client.

## *CONCEPTUAL FRAMEWORK*

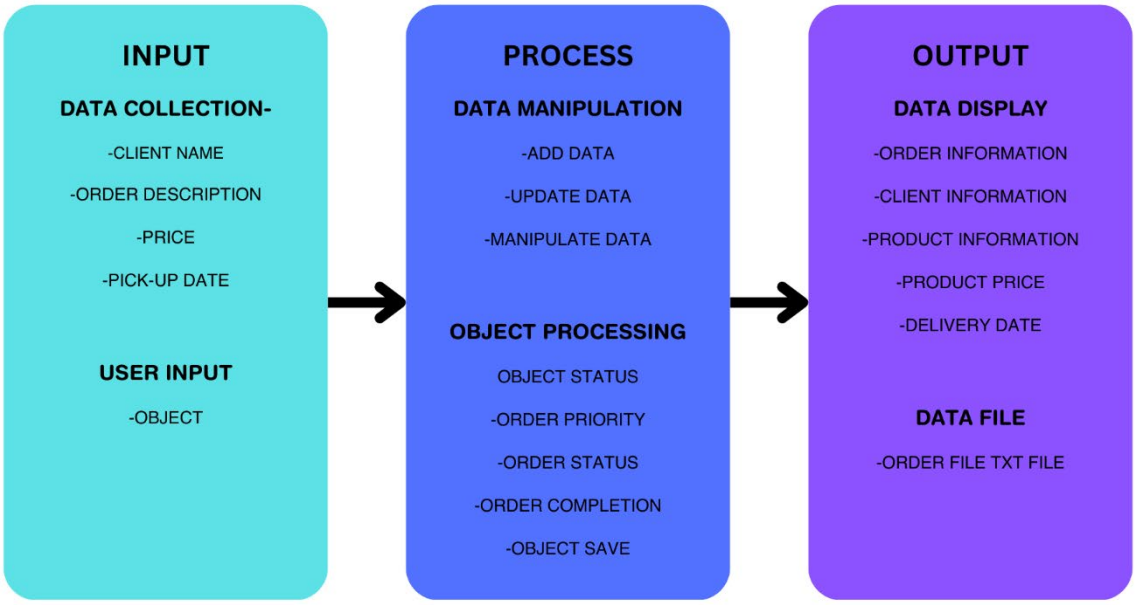


*FIGURE 1.1. CONCEPTUAL FRAMEWORK OF PROGRAM.*

The project contains three functionalities as detailed by the Conceptual framework presented in Figure 1.1. The program follows a Data input, Data process, and Data Output style of manipulation which consists of the user inputting order information which is manipulated through various objects. Buttons included allow for data Process which data input is processed through various parts, then lastly the data is displayed in the program, and saved on a Txt file for future uses.

## *SCOPE AND LIMITATIONS*

The project uses Java which is an easy-to-use and proven language to use in applications, especially in a client-specific program. However, much more complicated UI systems require additional libraries which can add up complexities and maintainability. As a prototype, this project utilizes basic Java libraries to function.

***PROJECT DEFINITION***

The project developed is a Graphical-User-Interface (GUI) based application used in taking orders and providing useful data to the client in managing various transactions. The application uses Java as its programming language which is a widely used Object-Oriented Programming Language and Software platform that can run on most devices due to its portability, ease of code, and common syntax based on C and C++ (What Is Java? | IBM, n.d.). It contains various packages which part of the Java Library that allows for various functions.

**Java.AWT.* -** Java AWT (Abstract Window Toolkit) is a package that represents an application programming interface (API) that facilitates Graphical User Interfaces (GUI) functionality and Windows-based applications in the Java programming language.

**Java.time.*** - Java time is a package that contains the API for date and time-related functionalities.

**Java.util.* -** Java Utilities is a package that contains collection frameworks, legacy functions, and other functionalities intended for complicated processes.

**javax.swing**.* - Java Swing is a package that contains platform-independent and lightweight components for various functionalities used in GUIs.

# PROJECT SOLUTION

## *PROJECT/SYSTEM PROTOTYPING*

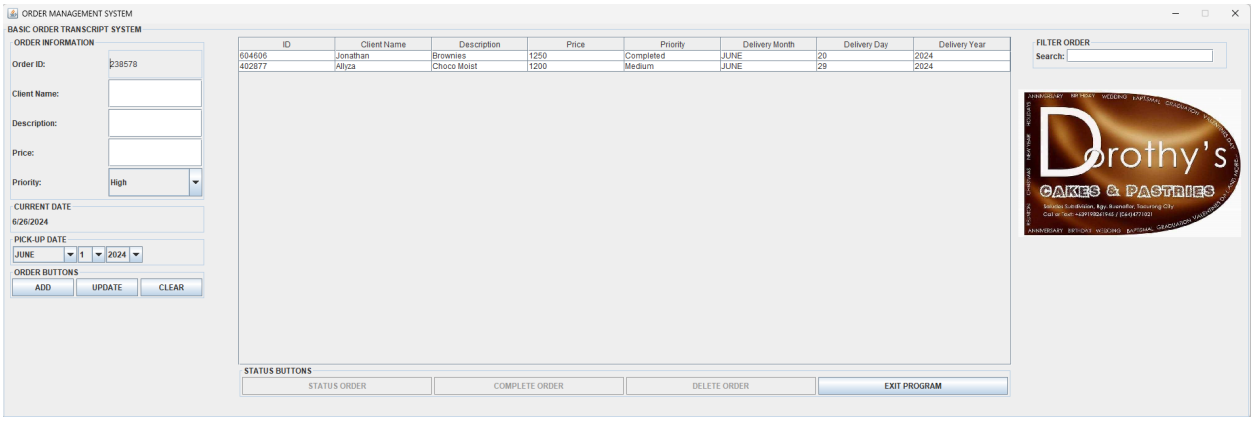


*Figure 2.1. Application Menu/Start.*

## *SAMPLE INPUT/OUTPUT*

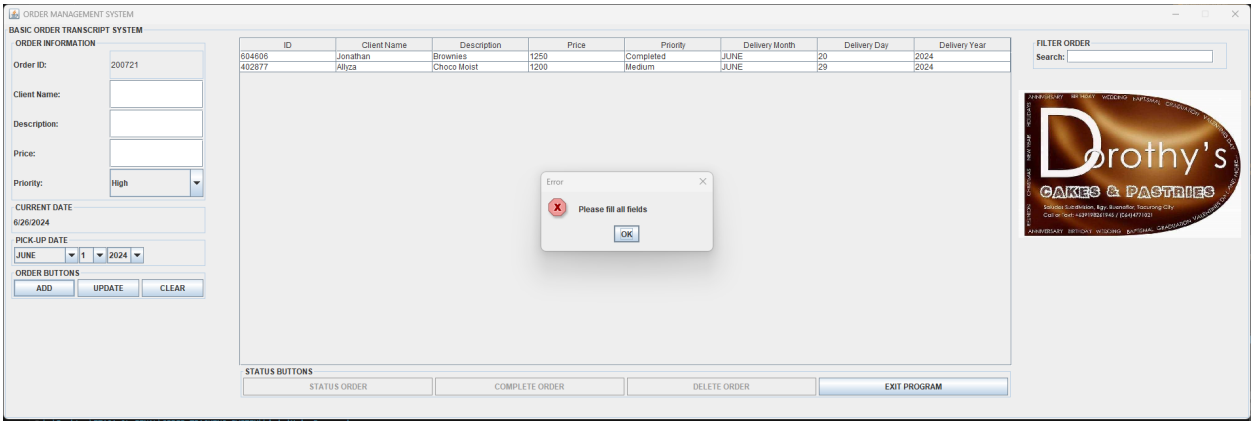


*Figure 3.1. INPUT (No DATA)*

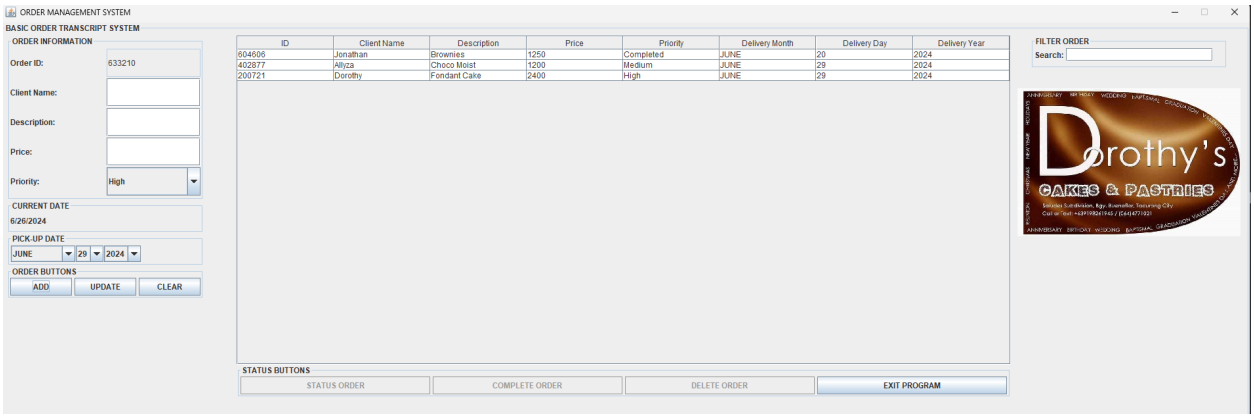


*Figure 3.2. INPUT (DATA ADD)*

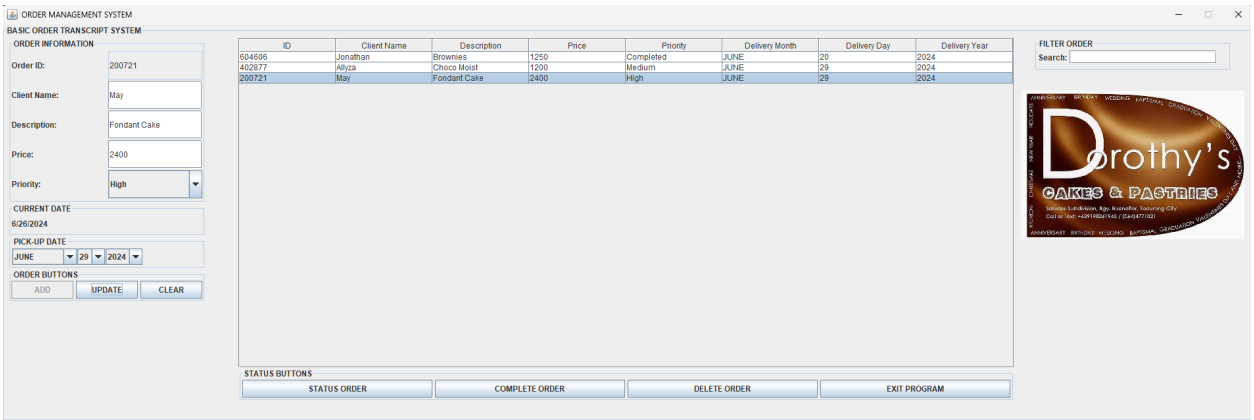


*Figure 3.3. INPUT (DATA UPDATE)*

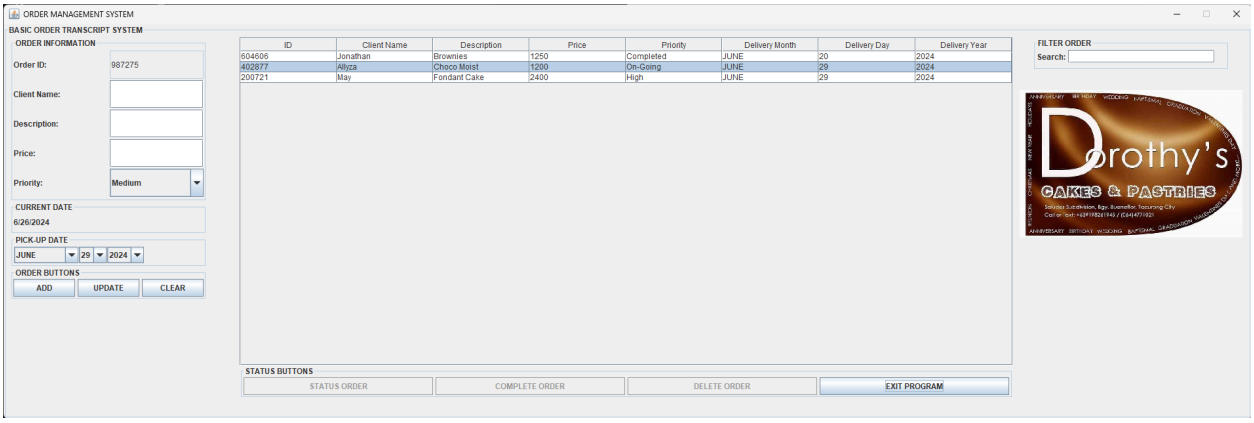


*Figure 3.4. OUTUT (CHANGE STATUS/PRIORITY)*

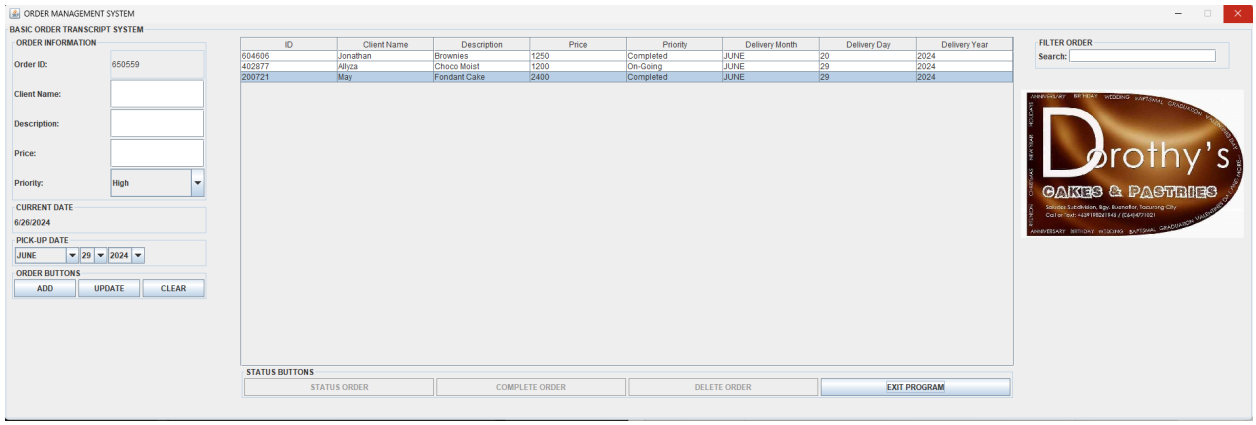


*Figure 3.5. OUTPUT (CHANGE TO COMPLETE)*

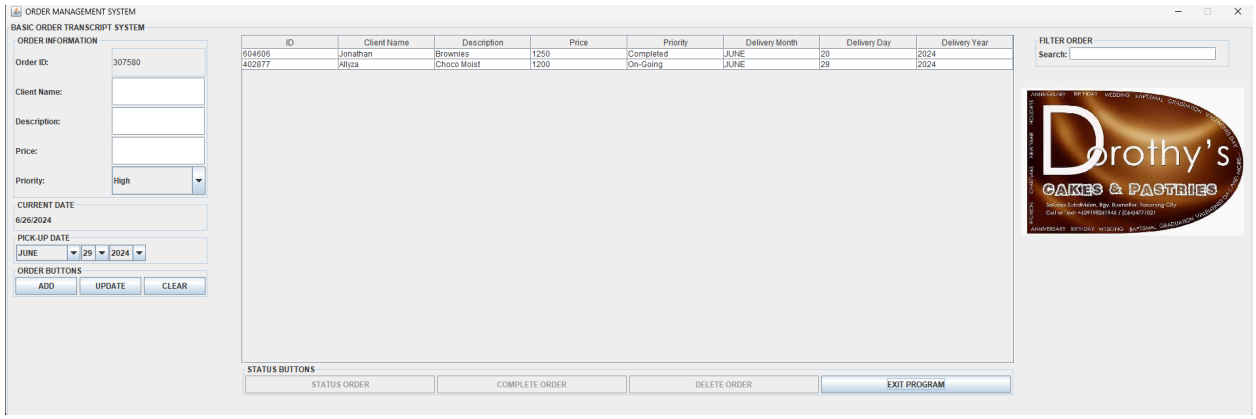


*FIGURE 3.6. ORDER DELETED.*

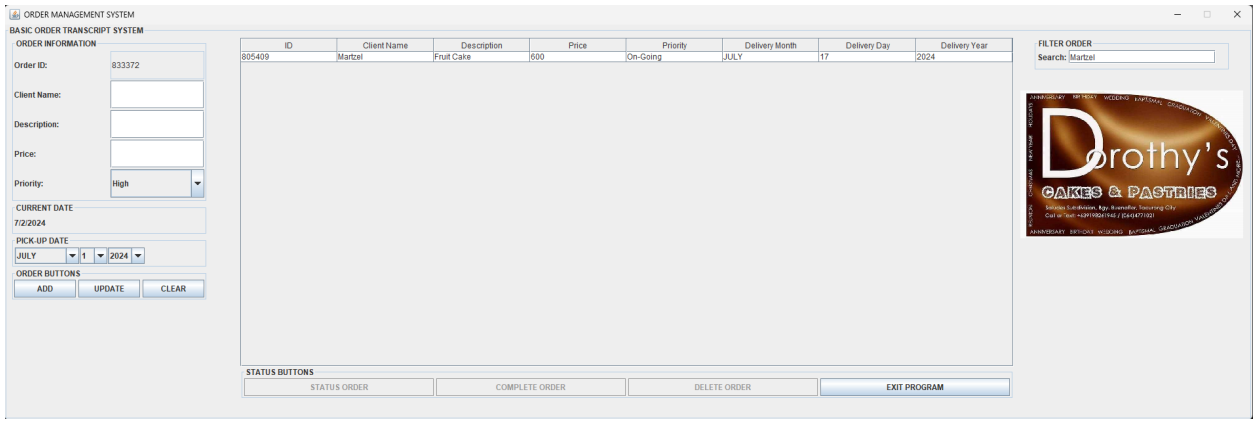


*Figure 3.7. ORDER FILTER.*

***SOURCE CODE***

**Main_Program.java**

```
/**
 * The Main_Program class represents the main program of the Order Tracking System.
 * It provides functionality for managing orders, including adding, updating, deleting, and
completing orders.
 * The program uses a graphical user interface (GUI) built with Swing components.
 *
 * The class extends the UI_Config class and implements the ActionListener, KeyListener,
and MouseListener interfaces.
 * It contains various instance variables for storing and manipulating order data, as well as
GUI components.
 *
 * The program initializes the GUI components, sets up event listeners, and loads order data
from a file.
 * It also provides methods for generating random order IDs, displaying order information,
and managing order dates.
 *
 * To use the program, simply run the Main_Program class.
 * The GUI will be displayed, allowing you to interact with the order management system.
 *
 * Note: This code is just a selection and may not compile or run on its own.
 */
import java.awt.BorderLayout;
import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.Image;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.time.Month;
import java.time.Year;
import java.util.Random;
import java.util.Vector;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableRowSorter;
```

```java
public class Main_Program extends UI_Config implements ActionListener, KeyListener,
MouseListener {
    // VARIABLES LIST
    private Storage st = new Storage("Order_list.txt");
    private JTable order_list;
    private DefaultTableModel order_model;
    private Vector<String> columns, rows;
    private TableRowSorter<DefaultTableModel> order_sort;

    private JLabel filterSearch;
    private JTextField filterText;

    private JComboBox<Month> month_delivery;
    private JComboBox<Integer> day_delivery;
    private JComboBox<Integer> year_delivery;
    private int current_year = Year.now().getValue();
    private int current_month = java.time.LocalDate.now().getMonthValue();
    private int current_day = java.time.LocalDate.now().getDayOfMonth();

    private JButton add_order, delete_order, update_order, clear_order, status_order,
complete_order, exit_program;

    private JLabel nameID, nameClient, nameDesc, namePrice, namePrio;
    private JTextField textID, textClient, textDesc, textPrice;
    private JComboBox<String> textPrio;

    private JPanel orderPanel, current_datePanel, pick_up_panel, orderButtonPanel,
statusButtonPanel, filterPanel, listPanel;

    /**
     * Constructor for the Main_Program class.
     */
    public Main_Program() {
        startup();

        orderInfo();
        add(orderPanel).setBounds(10, 20, 300, 250);

        current_date();
        add(current_datePanel).setBounds(10, 270, 300, 50);

        pick_up_date();
        add(pick_up_panel).setBounds(10, 320, 300, 50);

        orderButton();
```

```
        add(orderButtonPanel).setBounds(10, 370, 300, 50);

        statusButton();
        add(statusButtonPanel).setBounds(360, 520, 1180, 50);

        add(filterSearch()).setBounds(1570, 20, 300, 50);
        add(listTable()).setBounds(360, 20, 1180, 500);
        add(frame()).setBounds(20, 20, 1920, 1080);

        textID.setText(randomID());

        resetData();

        // BUTTON LISTENERS
        add_order.addActionListener(this);
        update_order.addActionListener(this);
        clear_order.addActionListener(this);
        status_order.addActionListener(this);
        complete_order.addActionListener(this);
        delete_order.addActionListener(this);
        exit_program.addActionListener(this);

        order_list.addMouseListener(this);
        filterText.addKeyListener(this);

        // FILE UTILIZATION
        st = new Storage("Order_list.txt");
        st.displayRecords(order_model);

        // UI RESOLUTION
        setMyFrame("ORDER MANAGEMENT SYSTEM", 1920, 640, true,
DISPOSE_ON_CLOSE, false);
        setLocationRelativeTo(null);
    }

    // OBJECT INITIALIZATION
    public void startup() {
        nameID = new JLabel("Order ID: ");
        nameClient = new JLabel("Client Name: ");
        nameDesc = new JLabel("Description: ");
        namePrice = new JLabel("Price: ");
        namePrio = new JLabel("Priority: ");

        textID = new JTextField(20);
        textID.setEditable(false);
```

```
        textClient = new JTextField(20);
        textDesc = new JTextField(20);
        textPrice = new JTextField(20);
        textPrio = new JComboBox<>();

        textPrio.addItem("High");
        textPrio.addItem("Medium");
        textPrio.addItem("Low");

        JLabel logo = new JLabel(new ImageIcon("LOGO\\Logo.jpg"));
        ImageIcon imageIcon = new ImageIcon(((ImageIcon)
logo.getIcon()).getImage().getScaledInstance(340, 225, Image.SCALE_DEFAULT));
        logo.setIcon(imageIcon);
        add(logo).setBounds(1550, 100, 340, 225);
    }

    // RANDOM ID NUMBERS
    public String randomID() {
        Random random = new Random();
        int randomNumber = random.nextInt(900000) + 100000;
        return String.valueOf(randomNumber);
    }

    // ORDER INFORMATION
    public void orderInfo() {
        orderPanel = new JPanel();

        orderPanel.setLayout(new GridLayout(5, 2, 4, 2));
        orderPanel.setBorder(BorderFactory.createTitledBorder("ORDER INFORMATION"));

        orderPanel.add(nameID);
        orderPanel.add(textID);

        orderPanel.add(nameClient);
        orderPanel.add(textClient);

        orderPanel.add(nameDesc);
        orderPanel.add(textDesc);

        orderPanel.add(namePrice);
        orderPanel.add(textPrice);

        orderPanel.add(namePrio);
        orderPanel.add(textPrio);
    }
```

```java
    // DISPLAYS CURRENT DATE
    public void current_date() {
        current_datePanel = new JPanel();
        current_datePanel.setLayout(new GridLayout(1, 3, 4, 2));
        current_datePanel.setBorder(BorderFactory.createTitledBorder("CURRENT DATE"));

        JLabel current_date = new JLabel(current_month + "/" + current_day + "/" +
current_year);
        current_datePanel.add(current_date);
    }

    // CREATES A 3 COMBO BOX LAYOUT FOR DISPLAY DATE
    public void pick_up_date() {
        pick_up_panel = new JPanel();
        pick_up_panel.setLayout(new GridLayout(1, 5, 4, 2));
        pick_up_panel.setBorder(BorderFactory.createTitledBorder("PICK-UP DATE"));

        month_delivery = new JComboBox<>();
        day_delivery = new JComboBox<>();
        year_delivery = new JComboBox<>();
        year_delivery.addItem(2024);

        pick_up_panel.setLayout(new FlowLayout(FlowLayout.LEFT, 1, 1));
        pick_up_panel.setBorder(BorderFactory.createTitledBorder("PICK-UP DATE"));

        pick_up_panel.add(month_delivery);
        pick_up_panel.add(day_delivery);
        pick_up_panel.add(year_delivery);

        for (int i = 1; i <= 31; i++) {
            day_delivery.addItem(i);
        }
        for (int i = current_month; i <= 12; i++) {
            month_delivery.addItem(Month.of(i));
        }
    }

    // CREATES A PANEL FOR ORDER MANIPULATION
    public void orderButton() {
        orderButtonPanel = new JPanel();
        orderButtonPanel.setLayout(new GridLayout(1, 3, 4, 2));
        orderButtonPanel.setBorder(BorderFactory.createTitledBorder("ORDER BUTTONS"));

        add_order = new JButton("ADD");
```

```
    update_order = new JButton("UPDATE");
    clear_order = new JButton("CLEAR");

    orderButtonPanel.add(add_order);
    orderButtonPanel.add(update_order);
    orderButtonPanel.add(clear_order);
}

// CREATES A SECONDARY PANEL FOR DATA MANIPULATION
public void statusButton() {
    statusButtonPanel = new JPanel();
    statusButtonPanel.setLayout(new GridLayout(1, 3, 4, 2));
    statusButtonPanel.setBorder(BorderFactory.createTitledBorder("STATUS BUTTONS"));

    status_order = new JButton("STATUS ORDER");
    complete_order = new JButton("COMPLETE ORDER");
    delete_order = new JButton("DELETE ORDER");
    exit_program = new JButton("EXIT PROGRAM");

    statusButtonPanel.add(status_order);
    statusButtonPanel.add(complete_order);
    statusButtonPanel.add(delete_order);
    statusButtonPanel.add(exit_program);

    status_order.setEnabled(false);
    complete_order.setEnabled(false);
    delete_order.setEnabled(false);
}

// CREATES A PANEL FOR FILTERING ORDERS
public JPanel filterSearch() {
    filterPanel = new JPanel();
    filterSearch = new JLabel("Search: ");
    filterPanel.setLayout(new FlowLayout(FlowLayout.LEFT, 1, 1));
    filterPanel.setBorder(BorderFactory.createTitledBorder("FILTER ORDER"));

    filterText = new JTextField(20);

    filterPanel.add(filterSearch);
    filterPanel.add(filterText);

    return filterPanel;
}

// CREATES A PANEL FOR FRAME BORDER
```

```
    public JPanel frame() {
        JPanel frame = new JPanel();
        frame.setLayout(new BorderLayout());
        frame.setBorder(BorderFactory.createTitledBorder("BASIC ORDER TRANSCRIPT
SYSTEM"));

        return frame;
    }

    // CREATES A PANEL FOR DISPLAYING DATA ON A TABLE
    public JPanel listTable() {
        listPanel = new JPanel();
        order_list = new JTable();
        order_model = new DefaultTableModel();

        listPanel.setLayout(new BorderLayout());
        listPanel.add(new JScrollPane(order_list), BorderLayout.CENTER);

        String column_header[] = { "ID", "Client Name", "Description", "Price", "Priority",
"Delivery Month", "Delivery Day", "Delivery Year" };

        columns = new Vector<>();

        for (String val : column_header) {
            columns.add(val);
        }

        order_model.setColumnIdentifiers(columns);
        order_list.setModel(order_model);

        return listPanel;
    }

    // RETRIEVES DATA
    public void getData() {
        rows = new Vector<>();
        rows.add(textID.getText());
        rows.add(textClient.getText());
        rows.add(textDesc.getText());
        rows.add(textPrice.getText());
        rows.add(textPrio.getSelectedItem().toString());
        rows.add(month_delivery.getSelectedItem().toString());
        rows.add(day_delivery.getSelectedItem().toString());
        rows.add(year_delivery.getSelectedItem().toString());
    }
```

```java
// START OF PROGRAM
public static void main(String[] args) {
    new Main_Program();
}

// RESETS BUTTON AND DATA VALUES
public void resetData() {
    textID.setText(randomID());
    textClient.setText("");
    textDesc.setText("");
    textPrice.setText("");

    add_order.setEnabled(true);
    status_order.setEnabled(false);
    complete_order.setEnabled(false);
    delete_order.setEnabled(false);
}

// ENABLES AND DISABLES BUTTONS
public void order_click() {
    add_order.setEnabled(false);
    update_order.setEnabled(true);
    delete_order.setEnabled(true);
    status_order.setEnabled(true);
    complete_order.setEnabled(true);
}

// PROCESS DATA INPUT
public void process_data() {
    String data = "";
    for (int i = 0; i < order_model.getRowCount(); i++) {
        for (int j = 0; j < order_model.getColumnCount(); j++) {
            data += order_model.getValueAt(i, j) + "#";
        }
        data += "\n";
    }
    st.storeToFile(data);
}

//
@Override
public void mouseClicked(MouseEvent e) {
    if (e.getSource().equals(order_list)) {
        int f = order_list.getSelectedRow();
```

```
            textID.setText(order_model.getValueAt(f, 0).toString());
            textClient.setText(order_model.getValueAt(f, 1).toString());
            textDesc.setText(order_model.getValueAt(f, 2).toString());
            textPrice.setText(order_model.getValueAt(f, 3).toString());
            textPrio.setSelectedItem(order_model.getValueAt(f, 4).toString());
            month_delivery.setSelectedItem(Month.valueOf(order_model.getValueAt(f,
5).toString()));
            day_delivery.setSelectedItem(Integer.parseInt(order_model.getValueAt(f,
6).toString()));
            year_delivery.setSelectedItem(Integer.parseInt(order_model.getValueAt(f,
7).toString()));

            order_click();
        }
    }

    @Override
    public void mousePressed(MouseEvent e) {
        // Unimplemented method
    }

    @Override
    public void mouseReleased(MouseEvent e) {
        // Unimplemented method
    }

    @Override
    public void mouseEntered(MouseEvent e) {
        // Unimplemented method
    }

    @Override
    public void mouseExited(MouseEvent e) {
        // Unimplemented method
    }

    @Override
    public void keyTyped(KeyEvent e) {
        if (e.getSource().equals(textPrice)) {
            if ((e.getKeyChar() >= 'a' && e.getKeyChar() <= 'z')) {
                e.consume();
            }
        } else if (e.getSource().equals(textClient) || e.getSource().equals(textDesc)) {
```

```
            if (!((e.getKeyChar() <= 'z' && e.getKeyChar() >= 'a') || (e.getKeyChar() <= 'Z' &&
e.getKeyChar() >= 'A'))) {
                e.consume();
            }
        }
    }

    @Override
    public void keyPressed(KeyEvent e) {
        // Unimplemented method
    }

    @Override
    public void keyReleased(KeyEvent e) {
        if (e.getSource().equals(filterText)) {
            String filter = filterText.getText();

            order_sort = new TableRowSorter<>(order_model);
            order_list.setRowSorter(order_sort);
            order_sort.setRowFilter(RowFilter.regexFilter(filter, 1));
        }
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (e.getSource().equals(add_order)) {
            if (textClient.getText().isEmpty() || textDesc.getText().isEmpty() ||
textPrice.getText().isEmpty()) {
                JOptionPane.showMessageDialog(this, "Please fill all fields", "Error",
JOptionPane.ERROR_MESSAGE);
                return;
            }
            getData();
            order_model.addRow(rows);
            process_data();
            resetData();
        } else if (e.getSource().equals(update_order)) {
            if (textClient.getText().isEmpty() || textDesc.getText().isEmpty() ||
textPrice.getText().isEmpty()) {
                JOptionPane.showMessageDialog(this, "Please fill all fields", "Error",
JOptionPane.ERROR_MESSAGE);
                return;
            }
            int f = order_list.getSelectedRow();
            getData();
```

```
            for (int col = 0; col < order_list.getColumnCount(); col++) {
                order_list.setValueAt(rows.get(col), f, col);
            }
        } else if (e.getSource().equals(clear_order)) {
            resetData();
        } else if (e.getSource().equals(status_order)) {
            int f = order_list.getSelectedRow();
            order_model.setValueAt("On-Going", f, 4);
            process_data();
            resetData();
        } else if (e.getSource().equals(complete_order)) {
            int f = order_list.getSelectedRow();
            order_model.setValueAt("Completed", f, 4);
            process_data();
            resetData();
        } else if (e.getSource().equals(delete_order)) {
            int f = order_list.getSelectedRow();
            order_model.removeRow(f);
            process_data();
            resetData();
        } else if (e.getSource().equals(exit_program)) {
            System.exit(0);
        }
    }
}
```

**UI_Config.java**

```java
/**
 * The UI_Config class represents a custom JFrame with various methods to configure its properties.
 * It inherits from the JFrame class, extending its functionality.
 * The class provides methods to set the frame resolution, frame title, visibility, close operation, and resizability.
 * It also provides a method to set a background image for the frame.
 * The class demonstrates object-oriented programming concepts such as inheritance, encapsulation, polymorphism, and abstraction.
 */

import javax.swing.*;


public class UI_Config extends JFrame {
    private int H, W;

    // DEFAULT CONSTRUCTOR
    public UI_Config() {
        super();
        H = 1280;
        W = 720;
        SetResolution(H, W);
    }

    // PARAMETERIZED CONSTRUCTOR
    public UI_Config(String title, int width, int height, boolean visible) {
        super(title);
        H = height;
        W = width;
        SetResolution(H, W);
        setVisible(visible);
    }

    // SETS FRAME RESOLUTION AND OTHER BASIC PROPERTIES
    public void SetResolution(int height, int width) {
        setSize(width, height);
        setResizable(false);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    // SETS FRAME TITLE AND RESOLUTION
```

```java
    public void setMyFrame(String title, int width, int height) {
        setTitle(title);
        SetResolution(height, width);
    }

    // SETS FRAME TITLE, RESOLUTION, AND VISIBILITY
    public void setMyFrame(String title, int width, int height, boolean visible) {
        setMyFrame(title, width, height);
        setVisible(visible);
    }

    // SETS FRAME TITLE, RESOLUTION, VISIBILITY, AND CLOSE OPERATION
    public void setMyFrame(String title, int width, int height, boolean visible, int
close_operation) {
        setMyFrame(title, width, height, visible);
        setDefaultCloseOperation(close_operation);
    }

    // SETS FRAME TITLE, RESOLUTION, VISIBILITY, CLOSE OPERATION, AND
RESIZABILITY
    public void setMyFrame(String title, int width, int height, boolean visible, int
close_operation, boolean resize) {
        setMyFrame(title, width, height, visible, close_operation);
        setResizable(resize);
    }

    // SETS BACKGROUND IMAGE FOR THE FRAME
    public JPanel setBackgroundImage(String file) {
        JPanel panelBG = new JPanel();
        JLabel img = new JLabel(new ImageIcon(file));
        panelBG.add(img);
        return panelBG;
    }
}

// OBJECT-ORIENTED PROGRAMMING (OOP) IMPLEMENTATIONS:
// 1. INHERITANCE: THE UI_Config CLASS INHERITS FROM JFrame, EXTENDING ITS
FUNCTIONALITY.
// 2. ENCAPSULATION: THE PRIVATE VARIABLES H AND W ARE ENCAPSULATED
WITHIN THE CLASS, AND PUBLIC METHODS ARE PROVIDED TO INTERACT WITH
THEM.
// 3. POLYMORPHISM: METHOD OVERLOADING IS USED IN setMyFrame TO PROVIDE
MULTIPLE VERSIONS OF THE METHOD.
```

```
// 4. ABSTRACTION: THE CLASS PROVIDES SIMPLE INTERFACES (METHODS) FOR SETTING FRAME PROPERTIES AND BACKGROUND IMAGE, HIDING THE COMPLEXITY OF THE IMPLEMENTATION.
```

**Storage.java**

```
/**
 * The Storage class provides functionality for storing data to a file and displaying records from the file.
 * It encapsulates the private variables file, fWrite, fRead, and scan, and provides public methods to interact with them.
 * The class supports both default and parameterized constructors for instantiation.
 * It also provides an error message dialog for displaying error messages.
 * The class can be easily reused in other parts of the program or in other projects due to its generic functionality.
 */
import java.io.*;
import java.util.*;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;

public class Storage {
    private File file = null;
    private FileWriter fWrite = null;
    private FileReader fRead = null;
    private Scanner scan = null;
    private Vector<String> row;

    // DEFAULT CONSTRUCTOR
    public Storage() {
    }

    // PARAMETERIZED CONSTRUCTOR
    public Storage(String filename) {
        file = new File(filename);
    }

    // SETS THE FILENAME FOR STORAGE OPERATIONS
    public void setFilename(String filename) {
        file = new File(filename);
    }

    // GETS THE FILENAME
    public String getFilename() {
```

```
        return file.getName();
    }

    // DISPLAYS AN ERROR MESSAGE DIALOG
    public void errorMessage(String message) {
        JOptionPane.showMessageDialog(null, message, "Error",
JOptionPane.ERROR_MESSAGE);
    }

    // STORES DATA TO THE SPECIFIED FILE
    public void storeToFile(String data) {
        try {
            fWrite = new FileWriter(file);
            fWrite.write(data);
            fWrite.flush();
            fWrite.close();
        } catch (Exception e) {
            errorMessage("Error 101: storeToFile\n" + e.getMessage());
        }
    }

    // DISPLAYS RECORDS FROM THE FILE TO THE PROVIDED TABLE MODEL
    public void displayRecords(DefaultTableModel model) {
        try {
            fRead = new FileReader(file);
            scan = new Scanner(fRead);

            String[] data;

            while (scan.hasNext()) {
                data = scan.nextLine().split("#");
                row = new Vector<>();
                for (int i = 0; i < model.getColumnCount(); i++) {
                    row.add(data[i]);
                }
                model.addRow(row);
            }
            fRead.close();
        } catch (Exception e) {
            errorMessage("Error 102: displayRecords\n" + e.getMessage());
        }
    }
}

// OBJECT-ORIENTED PROGRAMMING (OOP) IMPLEMENTATIONS:
```

```
// 1. ENCAPSULATION: THE PRIVATE VARIABLES file, fWrite, fRead, AND scan ARE
// ENCAPSULATED WITHIN THE CLASS, AND PUBLIC METHODS ARE PROVIDED TO INTERACT
// WITH THEM.
// 2. POLYMORPHISM: METHOD OVERLOADING IS USED IN THE CONSTRUCTORS TO PROVIDE
// MULTIPLE WAYS OF INSTANTIATING THE CLASS.
// 3. ABSTRACTION: THE CLASS PROVIDES SIMPLE INTERFACES (METHODS) FOR STORING
// DATA TO A FILE AND DISPLAYING RECORDS, HIDING THE COMPLEXITY OF THE
// IMPLEMENTATION.
// 4. REUSABILITY: THE CLASS CAN BE EASILY REUSED IN OTHER PARTS OF THE PROGRAM
// OR IN OTHER PROJECTS DUE TO ITS GENERIC FUNCTIONALITY.
```

## APPENDIX

### *REFERENCES*

Rachinger, M., Rauter, R., Müller, C., Vorraber, W., & Schirgi, E. (2019, December 9). Digitalization and its influence on business model innovation. *Journal of Manufacturing Technology Management*, *30*(8), 1143–1160. https://doi.org/10.1108/jmtm-01-2018-0020

*PwC Philippines (2020).Innovation and digital transformation: How are Philippine MSMEs performing?* https://www.pwc.com/ph/en/publications/ph-columns/business-unusual/2020/innovation-and-digital-transformation-how-are-philippine-msmes-performing.html

Team, D. (2024, May 27). How can going paperless reduce your carbon footprint? Document Management Software | Docsvault. https://www.docsvault.com/blog/how-can-going-paperless-reduce-your-carbon-footprint/

Yip, T. (2023, September 5). Benefits of Digital Transformation for SMEs: How to stay ahead of the competition. https://www.linkedin.com/pulse/benefits-digital-transformation-smes-how-stay-ahead-competition-yip/

What is Java? | IBM. (n.d.). https://www.ibm.com/topics/java

## PHOTO OP/DOCUMENTATION

### *CLIENT INTERVIEW/ ACCEPTANCE*

*Figure 2.1. Client Program Acceptance.*

### *POST-PRESENTATION*

# CURRICULUM VITAE

## PERSONAL INFORMATION

**NAME:** Mayo, Jonathan Lance S.
**BIRTHDATE:** May 16, 2004
**YEAR & SECTION:** 1st Year, IT101L.A124
**PROGRAM:** BSIS
**CURRENT ADDRESS:**
Unit 105, Casa Graciana, Camia Street Dao Corner, Juna Subdivision, Matina Crossing, Davao, Davao Del Sur

**CONTACT NO:** +63 920 569 6575
**EMAIL ADDRESS:** nathanmayo15@gmail.com

## EDUCATIONAL BACKGROUND

| | |
|---|---|
| COLLEGE | Mapua Malayan Colleges Mindanao<br>Gen. Douglas MacArthur Hwy, Talomo, Davao City, 8000 Davao del Sur (2023-Present) |
| SENIOR HIGH SCHOOL | Malayan Colleges Mindanao, a Mapua School<br>Gen. Douglas MacArthur Hwy, Talomo, Davao City, 8000 Davao del Sur (2020-2023) |
| HIGH SCHOOL | St. John Early Learning Center Inc.<br>08 Molave Street, Brgy. New Isabela, Tacurong City, Sultan Kudarat (2017-2020) |
| ELEMENTARY | St. John Learning Center Inc.<br>08 Molave Street, Brgy. New Isabela, Tacurong City, Sultan Kudarat (2011-2017) |

## AWARDS/ACHIEVEMENTS

List your awards/achievements in your extracurricular activities from elementary up to present in reverse chronological order (present to past).

| | |
|---|---|
| MALAYAN DEBATE CUP (PARTICIPANT) | February 2024 |
| Dean's Lister (MMCM 1st semester) | 2023-2024 |
| 2nd Place, Pasikatay Short Film | April 2023 |
| 2nd Place, E-Sports (Minecraft Mascot Building), Pasikatay 2022 | March 2022 |
| With High Honors (MMCM) | (SY 2021-2022) |
| 3rd Place, Division Level Technolympics (Technical Drafting) | October 2019 |
| With Honors/With High Honors (SJELCI) | March 2022 (SY 2007-2021) |

## SKILLS AND INTEREST

**Programming Knowledge:** Ren-py (VN creator based on Python), HPL (Hoi4-Programming Language based on C++), basic C++, Java, and Python knowledge.

**Skill/s:** Cognitive and Critical Thinking Skills.

**Talents:** Multi-lingualise, Mathematical Thinking, Strategic Thinking, Teaching, and Writing.

**Hobbies:** Chess, cooking, cycling, game modding (programming), fixing electronics, PC hobbyist, and Audiovisual Enthusiast.

# CURRICULUM VITAE

## PERSONAL INFORMATION

**NAME:** Villegas, Allyza Fe E.
**BIRTHDATE:** February 7, 2005
**YEAR & SECTION:** 1st Year, IT101L.A124
**PROGRAM:** BSIS
**CURRENT ADDRESS:**
Prk. 15, Upper Piedad, Bato, Toril, Davao City

**CONTACT NO:** +63 955 962 4784
**EMAIL ADDRESS:** villegasallyzafe@gmail.com

## EDUCATIONAL BACKGROUND

| | |
|---|---|
| COLLEGE | Mapua Malayan Colleges Mindanao<br>Gen. Douglas MacArthur Hwy, Talomo, Davao City, 8000 Davao del Sur (2023-Present) |
| SENIOR HIGH SCHOOL | Brokenshire College Toril<br>Lubogan Toril, Davao City, 8025 Davao del Sur, Philippines (2020-2023) |
| HIGH SCHOOL | YMCA INSTITUTE OF ARTS AND TECHNOLOGY, INC.<br>Purok 4, Bankas Heights, Toril, Davao City, 8025 Davao del Sur, Philippines (2017-2020) |
| ELEMENTARY | Brokenshire College Toril<br>Lubogan Toril, Davao City, 8025 Davao del Sur, Philippines (2011-2017) |

## AWARDS/ACHIEVEMENTS

List your awards/achievements in your extracurricular activities from elementary up to present in reverse chronological order (present to past).

| INSERT EVENTS/AWARDS | (MONTH & YEAR ACHIEVED) |
|---|---|
| Dean's Lister | 2023-2024 |
| With Honors/With High Honor | 2017-2018 |
| With Honors/With High Honor | 2011-2017 |

## SKILLS AND INTEREST

**Programming Knowledge:** basic C++, Java, and Python knowledge.

**Skill/s:** Communication, Creativity, and Content creation.

**Talents:** Designing, caligraphy, hiking, and negotiating.

**Hobbies:** Reading, drawing, writing, painting, cooking, and chess.