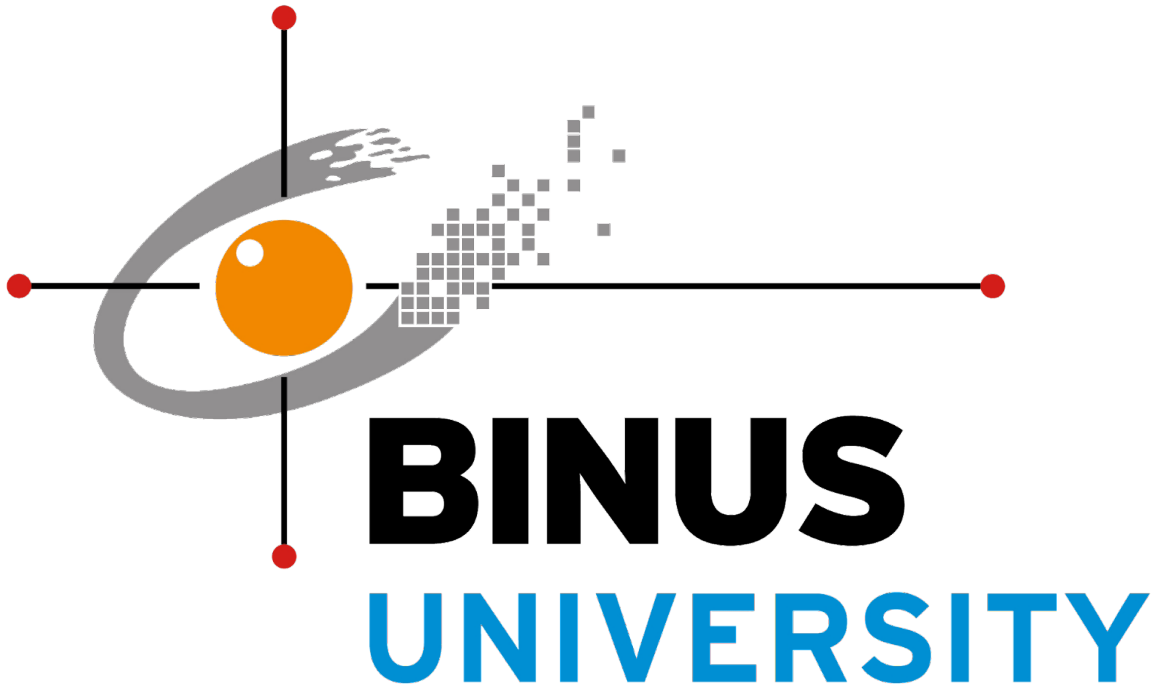


**Algorithm - ODD 2023**  
**The Written Report Finale project**  
**Flazz Card Usage History**



**Made By:**

Alrismany Abigail Sudjarwo  
L1BC | 2702414841

# CHAPTER 1

## Project Specification

### 1.Introduction:

- Provide an overview of the project, introducing the Flazz Card Transaction Management System.
- Mention the objective: to create a user-friendly application for managing financial transactions using Flazz cards.

### 2. Problem Statement:

- Identify the challenges faced by individuals in tracking and managing their Flazz card transactions.
- Discuss the need for a dedicated system to streamline financial record-keeping.

### 3. Project Scope:

- Specify the functionalities of the Flazz Card Transaction Management System, including:
  - User registration and login.
  - Real-time transaction recording (deposits and withdrawals).
  - Categorization of transactions (public transportation, supermarket, etc.).
  - Visualization of transaction history through charts and graphs.
  - Exporting transaction history to Excel and PDF formats.
  - Dark and light mode options for user interface customization.

### 4. import Used:

- Provide a list of technologies and libraries utilized in the project:
  - Python (Tkinter for GUI).
  - Pandas for data handling.
  - Matplotlib for data visualization.
  - ReportLab for EXCEL and PDF generation.

## CHAPTER 2

### Solution Design

#### Architecture Overview:

The Flazz Card Transaction Management System follows a modular and object-oriented architecture. The primary components include the GUI (Graphical User Interface) built using Tkinter in Python, the backend logic for transaction handling, and the integration with external libraries such as Matplotlib for data visualization and Pandas for efficient data management. The system operates on a client-server model, with the user interacting through the GUI while the backend processes transactions, updates account information, and generates visualizations.

#### Points:

- Modularity: The system is modular, with distinct components handling GUI, backend logic, and data visualization.
- Object-Oriented Design: Classes such as SetupWindow, Transaction, Account, and Bank encapsulate functionality, promoting code reusability and maintainability.
- Client-Server Model: The user interacts with the GUI (client), which communicates with the backend (server) to perform transactions and update account information.

#### Algorithms:

##### 1. Transaction Handling Algorithm:

- The system employs a straightforward algorithm for handling transactions. When a user initiates a deposit or withdrawal, the corresponding methods in the Account class execute the transaction, updating the account balance and creating a Transaction object with relevant details such as amount, timestamp, and category.

##### 2. Chart Update Algorithm:

- To update charts and visualizations dynamically, the system relies on Matplotlib. The `update_charts` method iterates through transaction history, categorizes data, and plots charts using Matplotlib's plotting functions.

Points:

- Efficient Transaction Processing: Transaction handling is efficient, leveraging Python's object-oriented capabilities for seamless data manipulation.
- Dynamic Chart Updates: The algorithm for updating charts ensures real-time representation of transaction patterns.

## Data Structures:

### 1. Account Class:

- The Account class stores essential information such as account number, account holder name, current balance, and a list of transactions. The use of a list facilitates efficient management of transaction history.

### 2. Transaction Class:

- Instances of the Transaction class encapsulate details about individual transactions, including amount, timestamp, and category. This class ensures a structured representation of transaction data.

### 3. Bank Class:

- The Bank class maintains a dictionary where each account number maps to an Account object. This data structure allows quick retrieval and management of user accounts.

Points:

- List for Transaction History: The use of a list in the Account class enables easy access and modification of transaction records.
- Dictionary for User Accounts: The dictionary in the Bank class ensures fast retrieval and management of user accounts.

## CHAPTER 3

### Implementation Details

#### 1. User Registration and Login:

- The user registration and login functionality is implemented in the SetupWindow class using Tkinter. Upon entering the Binusian ID Flazz number and full name, the system validates the input, creates an Account object in the Bank class, and allows users to log in.

#### 2. Transaction Handling:

- The Account class handles deposits and withdrawals. When a user adds or uses money, the corresponding methods update the account balance, create a Transaction object, and append it to the list of transactions.

#### 3. GUI Elements:

- Tkinter is extensively used to create the graphical user interface. Entry widgets, labels, buttons, and combo boxes are strategically placed to ensure a user-friendly experience. Dark and light mode themes are toggled dynamically, providing customization options.

#### 4. Data Visualization:

- Matplotlib is employed to generate various charts representing spending patterns, deposit history, spending history, and transaction statistics. The update\_charts method utilizes Matplotlib's plotting functions to visualize data dynamically.

#### 5. Data Export:

- The system enables users to export transaction history to Excel and PDF formats. The Pandas library is utilized to convert transaction data into DataFrames, and these DataFrames are exported to Excel and PDF files using the to\_excel and reportlab libraries, respectively.

#### 6. Security Measures:

- While the system is relatively simple, basic security measures are implemented. The Tkinter Entry widgets use validation functions to ensure that only valid inputs are accepted during user registration. Passwords and sensitive information are not handled in this system.

## 7. Logout Functionality:

- The logout button triggers a simple confirmation dialog, and if the user confirms, the current window is destroyed. Upon logout, the user is redirected to the initial setup window.

## 8. Chart Saving:

- Users can save the generated charts as JPEG files. The `save_chart_as_jpeg` method uses Matplotlib's `savefig` function to export the charts as JPEG images.

### Points:

#### - Python Libraries:

The implementation heavily relies on Python libraries such as Tkinter for GUI, Matplotlib for data visualization, Pandas for data handling, and reportlab for PDF generation.

#### - Dynamic Updates:

The system ensures dynamic updates of charts, transaction history, and balance information in response to user interactions.

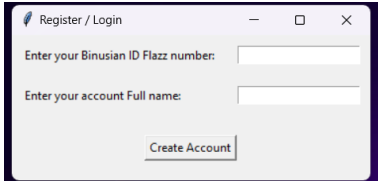
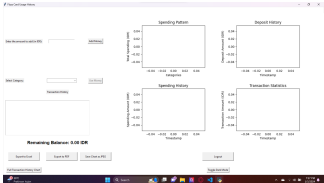
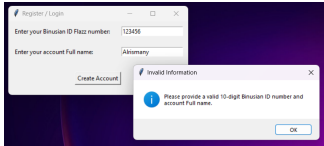

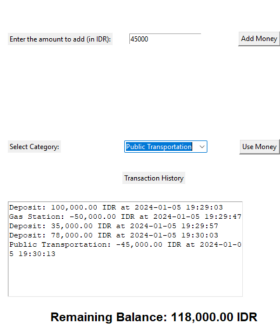
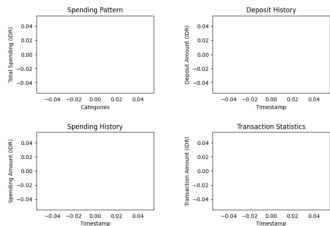
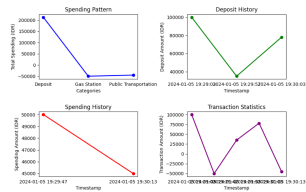
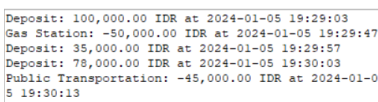

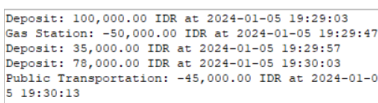

#### - User-Friendly Interface:

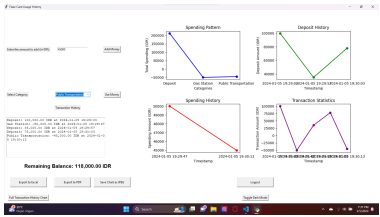
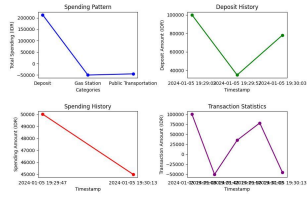
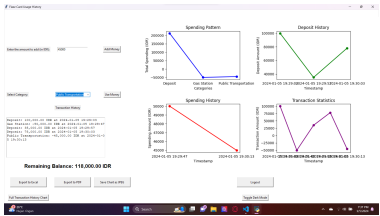
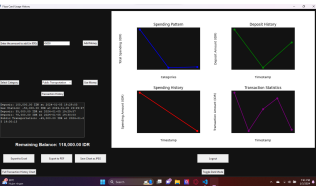
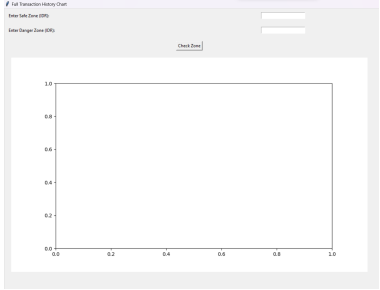
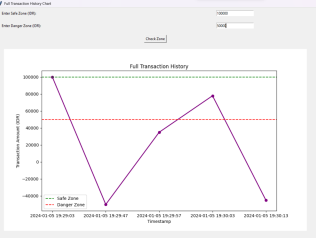
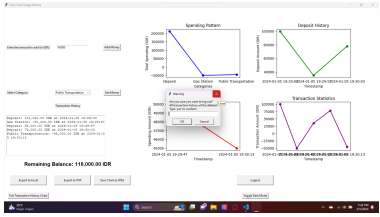
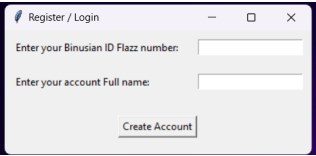
The graphical user interface is designed to be intuitive, with clear labels, entry fields, and buttons for seamless user interaction.

# CHAPTER 4

## Evidence of Working Program

Below are screenshots of the Flazz Card Management System in action:

	system	before	after	Error input
1	Login / Account Creation Window			
2	Input transaction and history			
3	Charts			
4	Export history to Excel			
5	Export history to PDF			

6	Export Chart to JPEG			In forms of jpeg
7	Dark mode			
8	Safe zone and danger zone Chart			
9	Log out			



# Reference

- A. (2019, October 5). *Generate PDF files with Python and ReportLab - #1*. YouTube. <https://www.youtube.com/watch?v=ZDR7-iSuwkQ>
- Assets, P. (2022, February 12). *Create PDF Documents in Python With ReportLab*. Python Assets. <https://pythonassets.com/posts/create-pdf-documents-in-python-with-reportlab/>
- Assets, P. (2022, February 12). *Create PDF Documents in Python With ReportLab*. Python Assets. <https://pythonassets.com/posts/create-pdf-documents-in-python-with-reportlab/>
- C. (2019, March 27). *Open Files Dialog Box - Python Tkinter GUI Tutorial #15*. YouTube. [https://www.youtube.com/watch?v=Aim\\_7fC-inw](https://www.youtube.com/watch?v=Aim_7fC-inw)
- *datetime* — *Basic date and time types*. (n.d.). Python Documentation. <https://docs.python.org/3/library/datetime.html>
- F. (2019, November 19). *Tkinter Course - Create Graphic User Interfaces in Python Tutorial*. YouTube. <https://www.youtube.com/watch?v=YXPyB4XeYLA>
- *import pandas as pd – Bring Pandas to Python | Data Independent*. (2023, February 23). Data Independent. <https://dataindependent.com/pandas/import-pandas-as-pd-bring-pandas-to-python/>
- *Matplotlib Pyplot*. (n.d.). [https://www.w3schools.com/python/matplotlib\\_pyplot.asp](https://www.w3schools.com/python/matplotlib_pyplot.asp)
- N. (2021, December 6). *Analyzing Financial Statements in Python*. YouTube. <https://www.youtube.com/watch?v=ZAAuGEVJsH8>
- N. (2022, November 21). *Modern Graphical User Interfaces in Python*. YouTube. <https://www.youtube.com/watch?v=iM3kjbKbKHQU>
- N. (2023, January 14). *Professional Expense Tracker in Python*. YouTube. [https://www.youtube.com/watch?v=tMLsR0\\_2yIE](https://www.youtube.com/watch?v=tMLsR0_2yIE)
- N. (2023, July 1). *Matplotlib Full Python Course - Data Science Fundamentals*. YouTube. <https://www.youtube.com/watch?v=OZOOLe2imFo>
- *Python math.isnan() Method*. (n.d.). [https://www.w3schools.com/python/ref\\_math\\_isnan.asp](https://www.w3schools.com/python/ref_math_isnan.asp)
- Python, R. (2023, January 30). *Python GUI Programming With Tkinter*. <https://realpython.com/python-gui-tkinter/>
- *Tkinter canvas - Python Tutorial*. (n.d.). <https://pythonbasics.org/tkinter-canvas/>
- *Tkinter Open File Dialog*. (2021, January 6). Python Tutorial - Master Python Programming for Beginners From Scratch. <https://www.pythontutorial.net/tkinter/tkinter-open-file-dialog/>
- *tkinter* — *Python interface to Tcl/Tk*. (n.d.). Python Documentation. <https://docs.python.org/3/library/tkinter.html>