Personal Piggy Bank

Elena Georgescu

Detailed Project Description:

The project is a straightforward personal financial management application developed with Python, leveraging the Tkinter library for the graphical user interface (GUI). Users can log in, create profiles, and effectively manage their bank accounts. This application encompasses various features, including viewing user profiles, accessing a dashboard displaying cashflow and transfer forms, and presenting detailed account information.

Under the hood, the application utilizes CSV files to store data, ensuring a structured approach to managing user information. For more advanced financial functionalities, calculations for monthly totals are carried out using Pandas and NumPy libraries. This integration enhances the application's capability to provide comprehensive insights into cashflow, enabling users to make informed financial decisions.

GitHub: https://github.com/Elena5288/PersonalPiggyBank

Programming Concepts Implemented:

- *Modular Design (OOP)*: The code utilizes classes (LoginPage, LabeledEntry, NewProfile, UserPiggyBankPage) to structure the GUI and separate concerns.
- *GUI Development*: The GUI is created using the tkinter library for various widgets like labels, buttons, frames, and notebooks.
- *File Handling*: CSV files (user_info.csv, account_info.csv, cashflow.csv, account_monthly_totals.csv) are read and written for user and account information.
- *Data Analysis*: Monthly totals are calculated using Pandas and NumPy for financial data analysis.
- *Dynamic Widget Creation*: The dynamic creation of widgets for bank account details in the NewProfile class is a good feature.
- *Error Handling*: There is basic error handling for empty fields during profile creation.

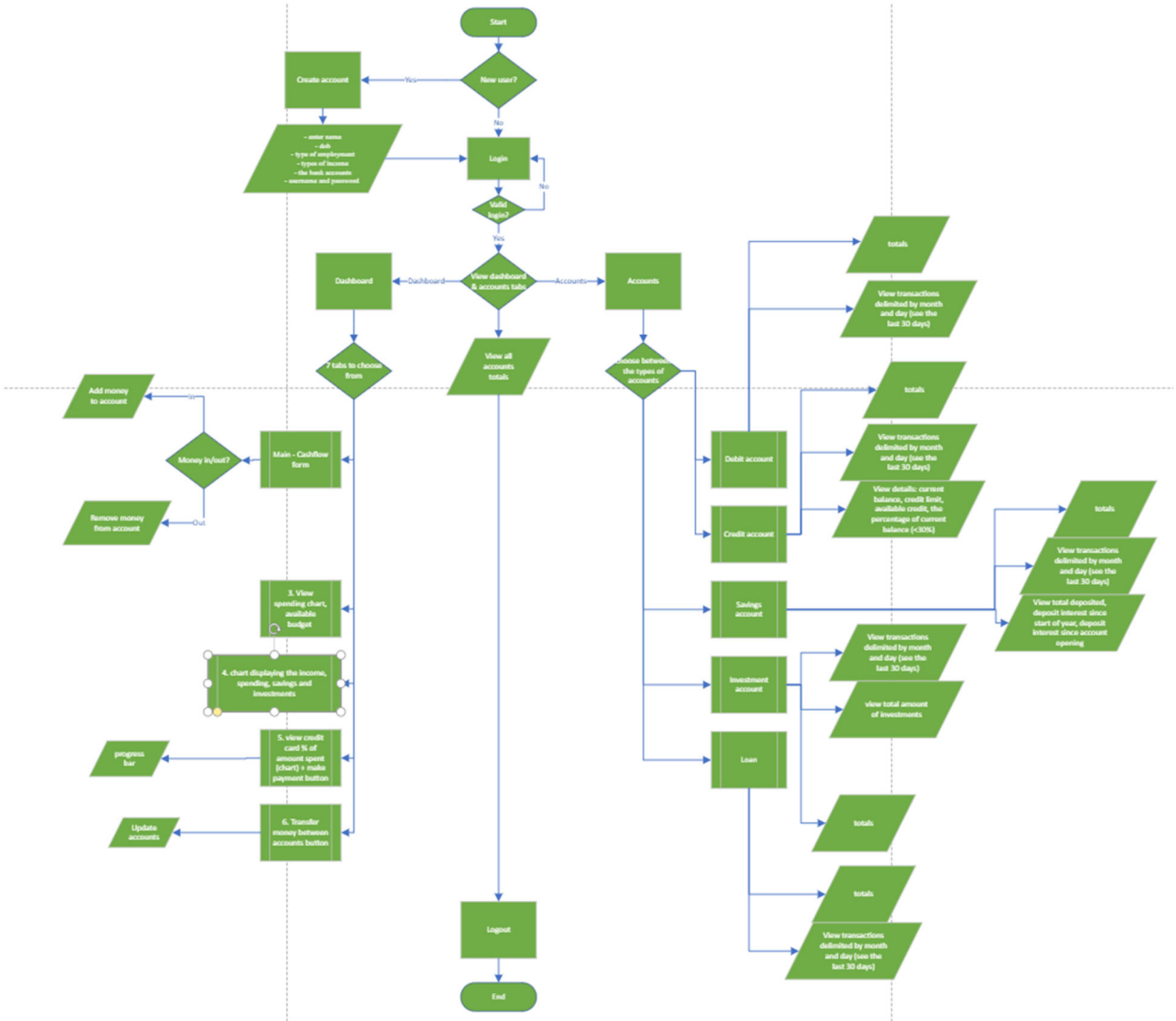
New Techniques Learned:

- *Dynamic Widget Creation*: In NewProfile.py, dynamic widgets are created to handle an arbitrary number of bank accounts based on user input. This allows flexibility in adding and managing multiple accounts without predefined widgets.
- *Focus Binding*: The use of self.bind in LabeledEntry class is a technique to handle the focus event when the user clicks on an entry field. This ensures that placeholder text is removed when the entry gains focus, enhancing user interaction.
- *Combobox*: The Combobox widget is utilized in NewProfile.py to provide a dropdown list for selecting the type of employment. It offers a user-friendly way to choose options from a predefined list.
- *Spinbox*: The Spinbox widget is employed in NewProfile.py for selecting day, month, and year in the date of birth entry. It allows users to increment or decrement values within a specified range, providing a convenient way to input date information.
- *Storing Usernames and Passwords with Login Verification*: Usernames and passwords are securely stored in a CSV file (user_info.csv). The program employs the verify_credentials method in LoginPage.py during the login process, cross-referencing entered credentials with the information stored in the CSV file.

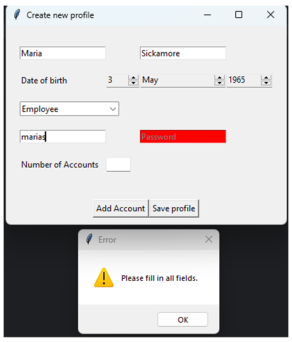
- *Opening New Windows*: The program opens a new window for the user's piggy bank (UserPiggyBankPage) upon successful login. This involves the creation of a new Tkinter Toplevel window, demonstrating how to manage multiple windows in a GUI application.
- *DateEntry*: widget from the tkcalendar module to create a date picker calendar in the cashflow form.
- *Menubar*: Implemented a menubar using the Menu widget to create a dropdown menu for different options like "Profile," "Dashboard," "Accounts," and "Logout" to provide a user-friendly interface.
- *Radio Button*: Incorporated the ttk.Radiobutton widget to create radio buttons for selecting "Spending" or "Deposit" in the cashflow form
- *Tabs and Notebook*: Employed the ttk.Notebook widget to create a tabbed interface for displaying different functionalities. Implemented tabs for "Accounts," "Dashboard," and "Profile" to organize and present information systematically.
- *Hasattr*: Used the hasattr function to check if an object has a certain attribute. Applied in dynamically recreating the dashboard frame if it already exists, ensuring a fresh display when switching tabs.
- *Pandas and NumPy*: Integrated Pandas and NumPy libraries for efficient data handling and analysis. Leveraged Pandas DataFrames to read and manipulate CSV files (user_info.csv, account_info.csv, cashflow.csv). Utilized NumPy for numerical operations, such as calculating monthly totals based on cashflow data.

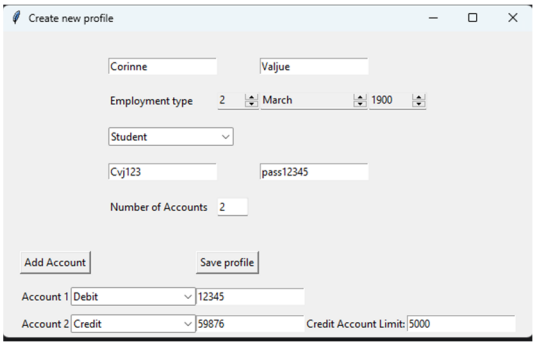
Screenshots of Sample Output:

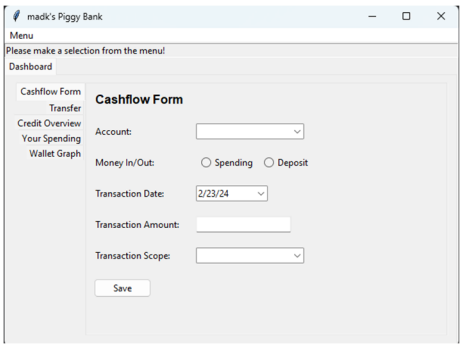
Flowchart:

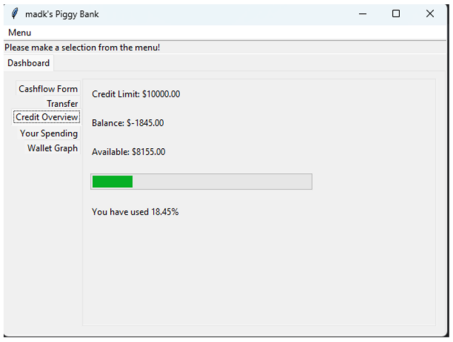


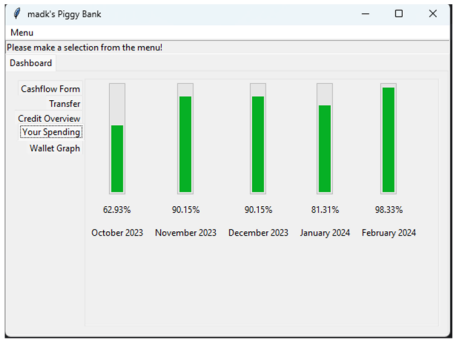
Key screenshots:

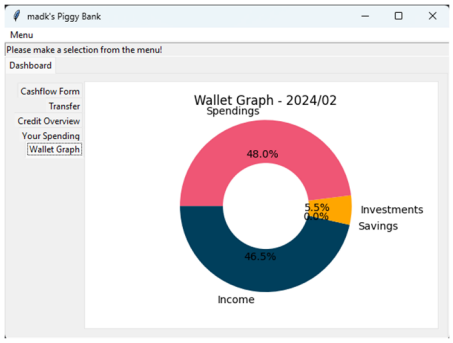


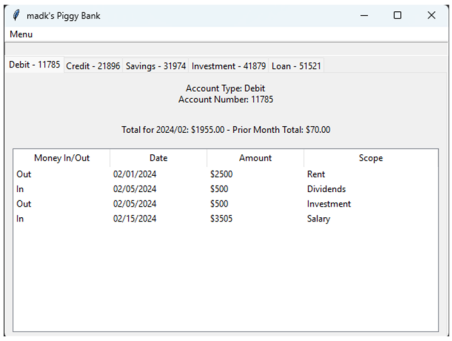


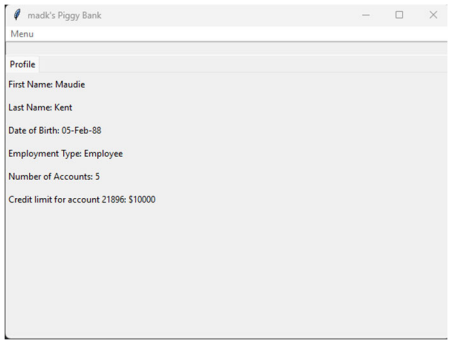












Overall Flow of the Program:

1. The program starts with Main.py, initializing the login page.
2. Users can log in or sign up.
3. The sign-up option opens the New Profile page.
4. Users can input personal information and add multiple bank accounts.
5. An error message appears if the personal information is not filled in and those fields are highlighted
6. Profile and account information are saved to CSV files.
7. Login credentials are verified against a CSV file.
8. Upon successful login, a message confirming this appears and the user's piggy bank window is opened.
9. On the main page the user can see the dashboard and it's first tab Cashflow Form where the user can input it's last transaction
10. Menu options allow the user to navigate between profile, dashboard, and accounts.
11. The profile contains information regarding the user: full name, date of birth, employment type, number of accounts, fetched from CSV files
12. Dashboard contains tabs for
    - Cashflow Form and Transfer Form to add new account transactions or transactions between accounts.

- Credit Overview to see how much the user spent from credit card and to keep an eye on the percentage which impacts its credit score,
- Your Spending to see the average spending in the last 5 months
- Wallet Graph to see the percentages of income, spendings, savings and investments in the last month

13. On save the cashflow and account_monthly_totals CSV files are updated as per user's input.
14. Accounts shows a tab for each of the user's accounts and the account information: type, number, totals and transactions from the last 30 days.
15. Monthly totals are calculated and displayed using Pandas and NumPy.
16. Logout the window is closed.

Names of Functions and Their Input/Output: *(some examples)*

get_user_id_from_csv (Function): Retrieves user ID from CSV data.

Input: self, username

Output: user_id

show_dashboard (Function): Displays the financial dashboard.

Input: self

Output: Dashboard information

save_cashflow_data (Function): Saves cashflow data to CSV.

Input: self

Output: None

read_account_monthly_totals (Function): Reads monthly totals for accounts

Input: self

Output: account_totals

load_accounts (Function): Loads user accounts.

Input: user_id

Output: Account details

get_account_totals_info (Function): Retrieves account totals information.

Input: account_number

Output: Totals information

calculate_monthly_totals (Function): Calculates and updates the totals in the CSV file

Input: None

Output: None

Structure of Classes

- NewProfile class:
  - Attributes: user_id, first_name, last_name, dob, employment, num_accounts.
  - Methods: __init__, add_account, save_profile.
- LabeledEntry class:
  - Inherits from Tkinter Entry.
  - Customized for labeled entry fields.
- UserPiggyBankPage Class:

- Variables: username, user_id, message_label, notebook, accounts_frame, dashboard_frame, profile_frame, profile_labels, cashflow_data.
- Methods: __init__, get_user_id_from_csv, show_profile, get_user_details_from_csv, show_dashboard, save_cashflow_data, save_transfer_data, show_accounts, read_account_monthly_totals, load_accounts, get_account_totals_info.