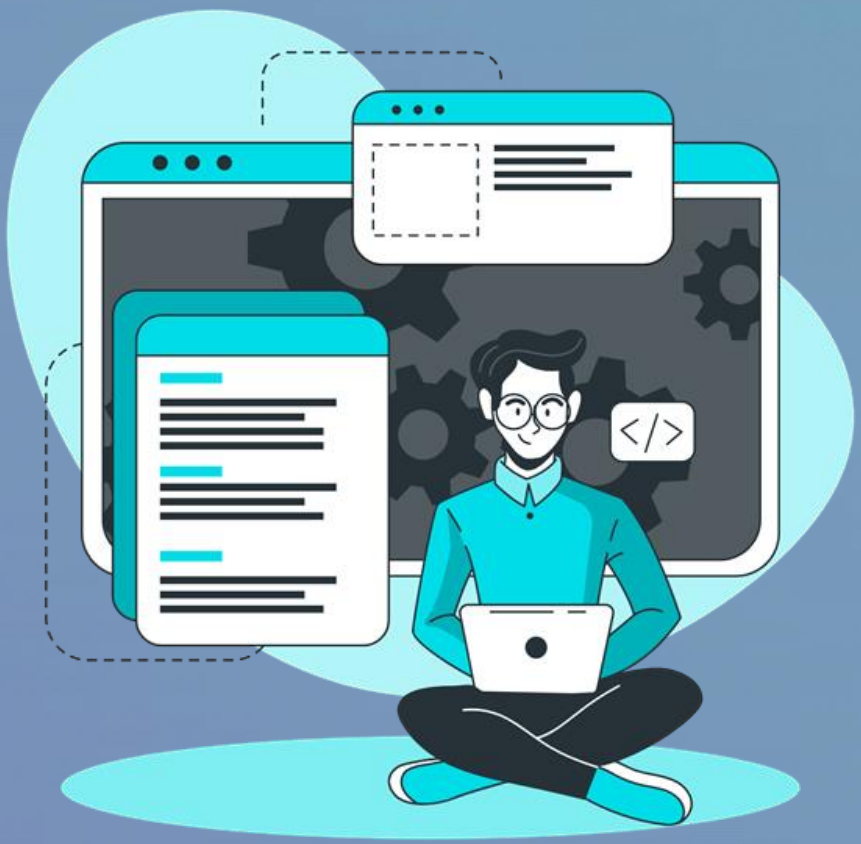
مکتب شریف

اولین بوتکمپ آموزشی - استخدامی ایران

PYTHON

BOOTCAMP

Project1-2



Introduction

In this document, we'll outline the design and implementation of a bank system using Object-Oriented Programming (OOP) principles in Python. The system will allow users to register, login, create accounts, deposit, and withdraw and transfer funds. PostgreSQL will be used as the database in backend, managed by a DBManager class utilizing Python's context managers. Additionally, logging will be employed to record transaction processes (use Decorators).

To facilitate collaborative development and version control, developers must use Git and GitHub effectively. The project repository should be organized following a Git branching model, such as Gitflow.

Phase 2

User Interface: Terminal Menu (OOP)

To provide a user-friendly interface for interacting with the bank system, developers must implement a terminal menu using Object-Oriented Programming (OOP) principles. The terminal menu will present users with options such as registering, logging in, creating accounts, depositing, and withdrawing funds.

Menu Features:

1. **Modular Design:** Each menu option will be encapsulated within a separate class, allowing for easy addition, removal, and modification of menu items.
2. **Navigation:** Users will navigate through the menu options using intuitive prompts and inputs.
3. **Error Handling:** The menu will handle user inputs gracefully, providing informative error messages for invalid selections or inputs.
4. **Integration with Bank System:** The menu will interact seamlessly with the bank system's underlying functionality, invoking appropriate methods based on user selections.

Example Menu Options:

- **Register:** Allow users to create a new account by providing a username and password.
- **Login:** Authenticate users with their credentials to access their accounts.
- **Create Account:** Create a new bank account associated with the logged-in user.
- **Deposit:** Add funds to an existing bank account.
- **Withdraw:** Withdraw funds from an existing bank account.
- **Transfer:** Transfer funds from one account to another.
- **View Balance:** Display the current balance of a bank account.

- **Logout:** Terminate the current session and return to the login screen.

Integrating Log File Data into the Database

To enhance the bank system's functionality, developers can integrate log file data into the database and implement a method to search for transactions involving transfers and deposits above $500. Here's how to achieve this:

Storing Log File Data in the Database:

1. **Database Schema Modification:**

   Extend the existing schema to include table with fields for log file data, such as log entry ID, timestamp, message, transaction type, amount, etc.

2. **Log File Parsing and Database Insertion:**

   Modify the DBManager class to include methods for inserting log file data into the log table.

3. **Searching Transactions with Transfer** and **Deposit Above $500:**
   - Add a new method to the DBManager class to search for transactions involving transfers and deposits above $500.
   - Construct SQL queries to filter transactions based on criteria such as transaction type (transfer, deposit), amount, and timestamp.

Extra Points (Optional):

Regular and Professional Accounts

To further enrich the bank system, developers can introduce two types of accounts: Regular and Professional. Professional accounts offer enhanced functionality, including profit calculation based on the number of transactions and higher transfer limits. Here's how to implement these features:

Regular and Professional Account Types:

1. **Account Class Modification:**

   Extend the existing BankAccount class to include properties and methods specific to Regular and Professional accounts.

   Define attributes such as account type, transaction count, and profit calculation logic for Professional accounts.

2. **Transaction Profit Calculation:**

   Implement a method to calculate transaction profit for Professional accounts based on the number of transactions performed.

Define a formula or algorithm to determine the profit amount, considering factors such as transaction volume and account balance.

3. **Enhanced Transfer Limits:**

   Set higher transfer limits for Professional accounts compared to Regular accounts to accommodate larger transactions.

   Apply validation checks to ensure that transfer amounts adhere to the specified limits for each account type.

نکات

- مهلت ارسال تمرین تا **پایان ساعت 9 روز پنج شنبه** می‌باشد.
- زین پس تمامی تحویل تمارین تنها و تنها از طریق گیتهاب (**Github**)صورت می‌پذیرد.
- بدین منظور لازم است یک مخزن(**repository**) بصورت (**private**) ساخته شود.
- آدرس ریپازیتوری را در کارتابل گروهی خود اعلام کرده و تیم تدریس را بعنوان **collaborator** بیفزایید:
  - **غزاله رنجبران**
  - رضا غلامی
  - زهرا متین فر
  - سعید کرمانشاهی
  - سینا مبارز
  - سید محمد رضوی

- ملاک و معیار ارزیابی تاریخ آخرین **commit** شما می‌باشد . (بصورت استاندارد و اصولی کامیت انجام شود.)
- **توصیه دوستانه.** از مواجهه با هیچ سوالی نترسید. به هر میزانی که در حل سوالات پیشروی کرده باشید نمره بخش مورد نظر را دریافت می‌کنید. بنابراین بیش از آنکه رسیدن به خروجی نهایی مهم باشد، تلاش شما ارزشمندتر است.
- قطعا هدف از تمارین صرفا رسیدن به جواب نهایی نیست و تمیز بودن کد و خلاقیتی که در انجام آن به خرج می دهید از اهمیت و امتیاز بالایی برخوردار است. ارائه راه حل کلی و عمومی برای یک مسئله که حالت های مختلف آن را در نظر بگیرد و فراتر از خواسته‌ی مسئله است. (خواسته‌ی مسئله گسترش داده شود یا حالت های خاص مسئله را پوشش دهد. قطعا مشمول امتیاز بیشتری خواهد شد.)
- سوالات امتیازی شامل مواردی است که نیازمند سرچ بیشتر شما عزیزان می‌باشد. بنابراین حل این سوالات نمره امتیازی دارد.

موفق باشید