**ATM APP(console application)**

**DEEPIGA.V**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **TOPICS** | **PAGE.NO** |
| **1** | **Aim of the Project** | **3** |
| **2** | **Business problem or Problem statement** | **3** |
| **3** | **Project Description** | **3** |
| **4** | **Functionalities** | **4** |
| **5** | **Input versatility with Error Handling and Exception Handling** | **6** |
| **6** | **Code Implementation** | **6** |
| **7** | **Results and Outcome** | **9** |
| **8** | **Conclusion** | **9** |

**ATM APP(Console Application)**

**Project Description**

**1. Aim of the project:**

The aim of the project is to develop a Python program that emulates an ATM, allowing users to withdraw cash and check their account balance, similar to the functionality provided by an actual ATM machine .

The main goal of this project are:

* Develop a console-based ATM application in Python
* Simulate real-world ATM functionalities such as cash withdrawals, deposits, balance inquiries.
* Create an user-friendly interface for seamless interaction.
* To Enhance understanding of Python programming principles, through the project.
* Top of Form

**2.Business problem or problem statement:**

Enhancements are necessary to address these issues and ensure the application meets the security and usability requirements of modern banking standards. In today's fast-paced world, providing users with efficient, secure, and easy access to their financial resources is paramount. Banks and financial institutions are continually seeking ways to enhance the customer experience while ensuring high levels of security and reliability. One of the critical touchpoints for customers is the Automated Teller Machine (ATM), which provides essential banking services such as balance inquiries, deposits, and withdrawals. This project addresses the ATM functionality by developing a secure and user-friendly ATM application in python.

**3. Project Description:**

The ATM Console Application project aims to develop a user-friendly and functional ATM simulation program in Python. The application will provide users with a console-based interface to perform basic banking operations such as withdrawing cash, depositing funds, checking account balances. The project will focus on user interaction to replicate the experience of using a real ATM. It also includes security features such as PIN authentication to ensure the privacy and security of users' accounts.

* User authentication using a PIN.
* Account balance inquiry.
* Cash withdrawal and deposit options.
* Real-time updates to the available balance.
* User-friendly command-line interface.

**Methodologies used**

1. Object - oriented programming(OOP)
2. Iterative development
3. Conditional statements
4. **Functionalities:**

This Python code defines a simple ATM (Automated Teller Machine) class and a basic user interface for interacting with the ATM functionalities.

**ATM Class:**

1. **Initialization (\_\_init\_\_ method):**
   * Initializes the ATM with a default balance of $10,000 (or any other specified balance).
   * balance attribute holds the current balance of the ATM.
2. **check\_balance method:**
   * Returns a string containing the current account balance.

* **deposit method:**
  + Takes an amount as input and adds it to the account balance.
  + Performs input validation to ensure that the deposit amount is a positive integer.
  + If successful, updates the balance and returns a message confirming the deposit and showing the new balance.
  + If an error occurs during the deposit process, it catches the exception and returns an error message.
* **withdraw method:**
  + Takes an amount as input and subtracts it from the account balance.
  + Performs input validation to ensure that the withdrawal amount is a positive integer.
  + Checks if the account balance is sufficient for the withdrawal.
  + If successful, updates the balance and returns a message confirming the withdrawal and showing the new balance.
  + If there are insufficient funds or an error occurs, it returns an appropriate error message.

**3. User PIN Handling:**

* + The program starts by prompting the user to create a new 4-digit PIN and confirming it.
  + It verifies that the entered PINs match and are of the correct format (4 digits).
  + If the PIN creation is successful, it stores the PIN for further use.
  + If not, it prompts the user to enter a valid PIN again.

**4. Loop:**

* + Once the PIN is set, the program enters a loop for user interaction.
  + It prompts the user to enter their PIN to access the ATM functionalities.
  + If the entered PIN matches the stored PIN, it enters another loop for ATM operations.
  + Inside the operations loop, the user can choose from options:
    - Check Balance
    - Deposit
    - Withdraw
    - Exit
  + Depending on the user's choice, it calls the respective methods of the ATM class and displays the results.
  + If the user chooses to exit, the program terminates.

**5. Error Handling:**

* + The program handles errors such as invalid PIN formats, incorrect PIN entries, invalid menu choices, and errors during deposit/withdrawal operations.
  + It provides appropriate error messages to guide the user.

This code provides a basic framework for an ATM system. Depending on your project requirements, you may want to enhance it with features like user authentication, transaction history tracking, error handling, and integration with a database for storing account information.

**5. Input Versatility with Error Handling and Exception Handling:**

Modified the deposit and withdraw methods to handle various types of input, including numeric values and strings.Implemented error handling to handle invalid inputs gracefully, such as non-numeric values or negative amounts for deposit/withdrawal.If an invalid input is detected, appropriate error messages are displayed to guide the user on correcting their input.These enhancements improve user-friendliness of the ATM system by ensuring that it can handle a wider range of input scenarios and provide informative feedback to the user in case of errors.

This approach ensures that all errors related to invalid transaction amounts are handled consistently and that users receive clear feedback for both successful and unsuccessful operations.

**Pin Creation and Validation**:

* The create\_pin function ensures that the PIN is exactly 4 digits and numeric. It also confirms that the PIN is entered correctly twice.
* The enter\_pin function checks the entered PIN against the stored PIN and provides feedback if the PIN is incorrect.

**Transaction Amount Validation**:

* In both deposit and withdraw methods, the code checks if the input is a valid positive integer.
* Provides appropriate error messages for invalid inputs or insufficient funds.

**Menu Option Selection**:

* The main\_menu function ensures that only valid menu options (1-4) are accepted and provides feedback for invalid selections.

**Error Messaging**:

* Clear and user-friendly error messages guide the user through the process, improving the overall user experience.

Top of Form

**6. Code implementation:**

To implement the project, we utilize basic Python programming concepts to create a modular and maintainable codebase. We leverage key algorithms and data structures to e efficiently manage data processing tasks. The code is organized into modules to ensure modularity and readability, with extensive documentation provided for clarity and future development.

**Description**

In this project, we implement various modules using basic Python programming concepts. Each module is designed to handle specific functionalities of the ATM console application. This Python code implements a simple ATM (Automated Teller Machine) simulator using object-oriented programming principles.

* program creates an instance of the ATM class.
* It runs an infinite loop to continuously display the menu of options until the user chooses to exit.
* Inside the loop, it displays the menu options: check balance, deposit, withdraw, and exit.
* Based on the user's choice, it calls the corresponding method of the ATM class (check\_balance, deposit, withdraw) and prints the result.
* If the user chooses to exit (option 4), the program breaks out of the loop and displays a thank-you message.



**7. Results and Outcome:**

The code creates a basic simulation of an ATM (Automated Teller Machine) system. It prompts the user to enter a 4-digit PIN. If the entered PIN is recognized, the user is presented with a menu of options:

* Check Balance
* Deposit
* Withdraw
* Exit

Based on the user's selection, the system performs the corresponding operation: checking balance, depositing funds, withdrawing funds, or exiting the program. The user is guided through the process with prompts.

**Outcome:**

**Account management:**

* Users can check their account balance, deposit funds, and withdraw funds.
* The account balance is updated accordingly after each transaction.

**Security Measures:**

* Users are required to enter a 4-digit PIN for authentication.
* If the entered PIN is not recognized, users can create a new PIN for account access.

**User Interaction:**

* The program provides a simple interface for users to interact with.
* Users can easily navigate through the menu options to perform desired transactions.

**Error Handling:**

* The code includes basic error handling to handle scenarios such as invalid PIN format and mismatched PIN confirmation.
* Users receive appropriate error messages and prompts to correct their input.

**Termination:**

* Users can exit the process at any time by selecting the exit option from the menu.

**8. conclusion:**

In conclusion, the ATM simulation project demonstrates the implementation of basic banking functionalities in Python,within a class-based structure. The program allows users to interact with the ATM by providing options to check balance, deposit funds, withdraw funds, and exit the system.