SUMMARY

OUTLINE

Headings you add to the document will appear here.

The software's primary objective is to offer users a reliable and secure interface for managing their financial transactions effectively.

How you tend to solve this problem

The solution to this problem involves the isolation of the back-end and GUI logic into separate classes. Each screen in the program has its own dedicated class, resulting in a highly modular and flexible design. This approach not only ensures that the code is organized and easy to maintain, but also makes it easier to add new features or modify existing ones in the future.

A discussion of programming constructs I'll be using

The programming constructs used in this project include:

- Object-oriented programming (OOP) concepts such as inheritance, polymorphism, and encapsulation
- Graphical User Interface (GUI) components such as JFrame, JPanel, JButton, JLabel, JTextField, and JPasswordField
- Event-driven programming with ActionListener and ComponentListener interfaces
- Exception handling with try-catch blocks to handle errors and user input validation
- Collection framework with ArrayList to store and manipulate account information
- Control structures such as if-else statements and loops to control the flow of the program

What measures I've taken to detect and prevent errors.

To detect and prevent errors, I have implemented several validators such as input validator and authenticators to ensure stable functioning of the program. Furthermore, I have written extensive main functions to test individual modules as well as the program as a whole. By taking these measures, I can detect any errors and bugs that may arise during the testing phase and ensure a smooth user experience.

Evidence of Appropriate Testing

Please refer to main(String[] args) of classes present in the code. Every one of them includes test cases to validate functionality of individual modules as well as their coupling.