🏦 Renaissance Bank Management System - Project Presentation

Slide 1: Title Slide

- **Title:** 🏦 Renaissance Bank Management System
- **Subtitle:** A Python-based Object-Oriented Financial Simulation 🐍
- **Author:** PRANAY KOTHARI (25BCE10248)
- **Date:** [Current Date]

Slide 2: Project Overview & Objectives

- **Project Summary:** A console application simulating core banking operations (account creation, transactions) using Python. 💻
- **Core Goal:** Demonstrate practical application of Object-Oriented Programming (OOP) concepts in a real-world system design. ✨
- **Key Focus Areas:** Secure access control 🔒, data integrity, and robust input handling.

Slide 3: Key System Features

1. **Triple-Factor Authentication:** Requires Name, Account Number, and Password for secure access. 🔐
2. **Unique Account Generation:** Automated assignment of unique 6-digit account numbers upon registration. 💳
3. **Core Transaction Support:** Functions for secure Deposit, Withdrawal, and Balance Inquiry. 💸
4. **Overdraft Protection:** Built-in logic checks for sufficient funds during withdrawal operations. 🛡️
5. **Robust Input Validation:** Uses `try-except` blocks for graceful handling of non-numeric or invalid inputs. ✅

Slide 4: System Architecture (OOP Design)

The system operates based on two primary, interacting classes: 🏛️

1. `Bank` Class (The Controller)

- **Role:** Manages the entire client base and handles system-wide operations. 🧠
- **Key Functions:** Maintains the list of all `Client` objects and performs credential verification via the `authenticate()` method.

2. `Client` Class (The Model)

- **Role:** Encapsulates the state (data) and behavior of an individual user account. 👤
- **Key Data:** Stores name, balance, account number, and password.
- **Key Functions:** Exposes methods like `deposit()`, `withdraw()`, and `check()`.

Slide 5: Usage Example: Account Interaction

This demonstrates a user session from creating an account to performing a transaction: ⌨️

```
> System: What would you like to do: 1.) Create a new account | 2.) Inquire existing ac
> User: 1
> System: Enter name: Jane Smith
> System: Enter deposit amount: 7500
> System: Enter password: starfleet1
> System: Your account has been created, account number is: 790432 ➡️

> System: Press 1, 2, or 3: 2
> System: Enter name: Jane Smith
> System: Enter account number: 790432
> System: Enter password: starfleet1

> System: What would you like to do: 1.) Deposit | 2.) Withdraw | 3.) Check balance
> User: 2
> System: Enter withdrawal amount: 1200
> System: Withdrawal successful 🎉
```

Slide 6: Future Scope & Enhancements 🚀

- **Data Persistence:** Implement database integration (e.g., SQLite or JSON file handling) to prevent data loss upon application exit. 💾
- **Security Upgrade:** Implement cryptographic password hashing (e.g., using `bcrypt` or `hashlib`) for enhanced security. 🔑
- **Unit Testing:** Develop comprehensive unit tests (e.g., using `pytest`) for core business logic to ensure reliability and maintainability. 🧪
- **GUI Migration:** Transition from the console interface to a Graphical User Interface (GUI) using libraries like Tkinter or PyQt. 🖥️

Slide 7: Conclusion

- **Summary:** The project is a successful implementation of a secure, modular financial simulation using Python OOP. 🏆
- **Impact:** It provides a strong foundation for understanding object-oriented design patterns and secure coding practices in application development. ⭐
- **Thank You.** 🙏