Loan Fraud Detection and Response System Report

Objective

The goal of this project was to simulate common types of loan fraud and build a simple Python-based system capable of detecting and responding to them in real time. The system is designed to model basic financial security logic in a way that's understandable and expandable.

Tools and Technologies Used

- **Python 3.x** — Chosen for its simplicity and readability in prototyping logic-based systems.
- **Datetime module** — Used for tracking timestamps and detecting timing-based fraud patterns (e.g., loan stacking).
- **Standard Print Logging** — Simple logging to the console was used to simulate alerts and administrative responses.

Simulated Fraud Scenarios

Three common loan fraud patterns were implemented in the system:

1. **Account Takeover**
   This simulates a scenario where a user's account is accessed from an unusual IP address. The system compares the current login IP with the previously known one. If they differ, it flags the event as suspicious and locks the account.

2. **Fake Borrower Identity**
   This checks whether the borrower's identity is verified. If the system detects a mismatch or failure in identity verification, it assumes the possibility of a fake or stolen identity and locks the account.

3. **Loan Stacking**
   This scenario involves multiple loan requests submitted within a short period. If three or more requests are made in under one minute, the system identifies this as suspicious stacking behavior and triggers an alert.

Functions and Their Roles

- `detect_account_takeover(user_id, current_ip)`
  Compares login IPs to detect unauthorized access. If the IP changes unexpectedly, it alerts and locks the account.

- `detect_fake_identity(user_id, is_verified)`
  Checks a verification flag to confirm identity. A failed verification leads to the account being flagged and locked.

- `detect_loan_stacking(user_id)`
  Evaluates the timing of loan applications. If three or more are made within one minute, it locks the account to prevent abuse.

- `alert_admin(user_id, reason)`
  Sends a simulated alert (via console print) describing the suspected issue.

- `lock_account(user_id)`
  Marks the account as locked in the database to prevent further actions.

Challenges Encountered

- **Simplicity vs. Realism**: One of the challenges was keeping the system understandable while simulating meaningful fraud behavior. Real-world systems would involve much more data and complexity (e.g., behavioral analysis, external APIs, etc.).

- **Time-based Detection Logic**: Implementing the loan stacking check required precise use of timestamps to track submission windows.

- **Data Handling Without a Real Database**: Since this is a simulation, all data was stored in dictionaries. This limits the scalability but simplifies testing.

Conclusion

This project provided a basic but functional simulation of how a loan fraud detection system can operate. By combining simple condition checks with simulated data and user activity, the script

is able to catch and respond to three high-risk behaviors. While basic, the system lays a foundation for integrating real-time monitoring, secure data storage, and advanced analytics in future iterations.