**THE EMIRATES NATIONAL SCHOOL, SHARJAH**

**GRADE 12**

**COMPUTER SCIENCE PROJECT WORK**

**PROJECT SYNOPSIS**

**1. Title of the Project**  
**Quantra** —A CLI Personal Finance Management System

**2.Members**

Akshay Anilkumar

Ibrahim Sarpanch

Jesroon Mammen Shoby

**3. Introduction / Overview**  
Quantra is a command-line interface (CLI) based, multi-user financial management system. It integrates core personal finance features—including banking, budgeting, loans, insurance, and stock market simulation—into a single, terminal-based application. The project addresses the need for a structured, lightweight, and educational tool to manage personal finances without the complexity of a graphical interface, serving as a central connection point (nexus) for all financial operations.

**4. Objectives of the Project**

* To provide a secure, multi-user authentication system for personalized finance management.
* To manage core banking functions: account creation, transactions, and balance tracking.
* To automate recurring financial operations such as bill payments and reminders.
* To simulate real-world financial products like loans (with interest calculation) and insurance policies.
* To implement a virtual stock trading module with simulated market fluctuations.
* To generate comprehensive textual financial reports and summaries for users.

**5. Scope of the Project**  
The project's scope encompasses user registration, authentication, and role-based access control. It includes modules for banking, loan management, insurance, and a simulated stock trading environment. All data is persisted in a relational database. The project is deliberately scoped to a CLI environment, meaning it will feature text-based input/output only and will not include a graphical user interface (GUI), web deployment, or real-time stock market data integration.

**6. Tools and Technologies Used**

* **Programming Language:** Python, SQL
* **IDE:** IDLE/Visual Studio Code
* **Database Management System:** MySQL
* **Key Python Libraries:**
  + sqlite3 for database interactions
  + hashlib for secure password hashing
  + datetime for handling dates and scheduling
  + os for system-level operations

**7. System Requirements**

* **Software Requirements:**
  + Python 3.12.x or later
  + MySQL
  + Operating System: Windows 10/11, Linux, or macOS
* **Hardware Requirements (Minimum):**
  + Dual-core Processor
  + 2 GB RAM
  + 500 MB of free storage space

**8. Methodology / Working of the Project**  
The system follows a structured, modular approach. The workflow begins with user authentication. Upon successful login, the user is presented with a menu-based interface corresponding to their role. Each menu option triggers a specific module (e.g., Banking, Loans). User actions are processed by Python logic, which queries or updates the central SQLite database. Automated scripts run in the background to handle scheduled tasks like interest accrual on loans or recurring payments. The system ensures data integrity and persistence through ACID-compliant database transactions.

**9. Modules Description**  
The project is divided into the following functional modules:

* **User Authentication & Role Management:** Handles user registration, login, and session management.
* **Banking System Core:** Manages the creation of accounts, deposits, withdrawals, and fund transfers.
* **Loan Management Module:** Handles loan applications, calculates interest, and processes repayments.
* **Insurance Management Module:** Allows users to enroll in and manage simulated insurance policies.
* **Stock Simulation Engine:** Provides a virtual stock market with fluctuating prices and allows users to buy/sell shares.
* **Automation & Scheduling System:** Manages all automated tasks like salary credits, bill payments, and reminders.
* **Reporting & Analytics Module:** Generates text-based summaries of transactions, portfolio performance, and financial health.

**10. Expected Output / Results**  
The final output will be a fully functional command-line application. Users will interact with text menus to perform all financial operations. The results will include:

* Successful user registration and login.
* Accurate updating of account balances after transactions.
* Generation of loan amortization schedules and insurance policy documents.
* A simulated stock portfolio that updates with market changes.
* Automated payment confirmations and reminder notifications.
* Detailed textual financial reports generated on demand.

**11. Conclusion**  
Quantra successfully demonstrates the integration of core computer science concepts—including database management, secure authentication, and modular software design—into a practical application. It provides a valuable, consolidated platform (a true "nexus") for understanding and managing personal finance through a terminal. The project serves as a strong foundation for understanding software development life cycles and complex system design.

**12. Limitations and Future Enhancements**

* **Limitations:**
  + The interface is purely text-based (CLI), which may not be as user-friendly as a GUI.
  + Stock market data is simulated and not connected to real-world live APIs.
  + Security is limited to password hashing; advanced encryption for data-at-rest is not implemented.
* **Future Enhancements:**
  + Development of a graphical user interface (GUI) using Tkinter or a web framework like Flask.
  + Integration with real financial APIs for live stock prices and currency exchange rates.
  + Implementation of advanced encryption standards (AES) for sensitive data.
  + Porting the application to mobile platforms (Android/iOS).
  + Adding machine learning algorithms for personalized spending insights and financial forecasting.

**13. Bibliography / References**

* Python Software Foundation. *Python 3.12.4 Documentation.*
* SQLite. *SQLite Documentation.*
* GeeksforGeeks. *"Python Programming Language"* tutorials and articles.
* W3Schools. *"Python Tutorial"* and *"SQL Tutorial".*
* Real Python. *Various articles on database management and CLI development*