Personal Finance Tools Python Module

Objective

You are tasked with developing a **Python module** called finance_tools.py that provides at least **10 distinct financial calculation functions**.
You must also create a **unit test file** test_finance_tools.py to thoroughly test these functions.

The focus is on:

- Writing clean, modular Python code.
- Correct and efficient algorithm design.
- Professional unit testing.
- Code readability, documentation, and robustness.

Functional Requirements

You must implement the following **10 financial calculators/tools** as independent functions inside finance_tools.py:

No	Tool Name	Description
1	**EMI Calculator**	Calculates monthly loan installments given principal, annual interest rate, and tenure.
2	**SIP Calculator**	Calculates maturity amount for Systematic Investment Plan (monthly investments).
3	**FD (Fixed Deposit) Calculator**	Calculates maturity amount for lump-sum fixed deposits over a period at a fixed rate.
4	**RD (Recurring Deposit) Calculator**	Calculates maturity value for recurring monthly deposits over time.
5	**Retirement Savings Estimator**	Projects future corpus based on current savings, monthly addition, and expected returns.
6	**Home Loan Eligibility Estimator**	Estimates max loan based on income and expenses.

No	Tool Name	Description
7	**Credit Card Interest Calculator**	Calculates outstanding balance if minimum payment is made each month.
8	**Taxable Income Calculator**	Computes taxable income after considering deductions (basic assumptions can be made).
9	**Simple Budget Planner**	Suggests saving/investing plans based on income vs expenses.
10	**Net Worth Calculator**	Calculates net worth based on assets and liabilities entered by user.

Technical Requirements

1. **Module File: finance_tools.py**
    - Each calculator must be a **separate Python function**.
    - Functions must:
        - Accept inputs as parameters.
        - Validate inputs (e.g., no negative principal or interest rates).
        - Return results (do **not** print inside the functions).
    - Use appropriate data types (float, int, etc.).
    - Write **docstrings** for each function describing its purpose, inputs, and outputs.
2. **Testing File: test_finance_tools.py**
    - Use the **unittest** framework.
    - Write at least **two test cases per function** (normal case + edge case).
    - Tests must check for correctness of output.
3. **Code Quality**
    - Follow **PEP8** standards for code formatting.
    - Handle potential input errors gracefully (e.g., negative values, wrong data types).
4. **Documentation**

- Provide a **README.md** explaining:
  - Purpose of the module
  - List of tools implemented
  - How to run and test the module

Suggested Function Naming

Function Name	Purpose
calculate_emi()	EMI for a loan
calculate_sip()	SIP maturity amount
calculate_fd()	FD maturity amount
calculate_rd()	RD maturity amount
estimate_retirement_corpus()	Future retirement savings
estimate_home_loan_eligibility()	Max loan eligibility
calculate_credit_card_balance()	Credit card interest buildup
calculate_taxable_income()	Post-deduction taxable income
plan_budget()	Budget recommendation
calculate_net_worth()	Net worth computation

Submission Requirements

- finance_tools.py (main module)
- test_finance_tools.py (unit test cases)
- README.md (documentation)
- GitHub repository (with clean commit history)
- (Bonus) JIRA Board screenshot showing task tracking

Online Banking System with Integrated Financial Tools

Objective

Develop a complete **web application** using **Python**, **Django**, and **basic Pandas** that allows users to:

- **Open a bank account** (registration).
- **Log in** securely to access their personal dashboard.
- **Perform basic banking transactions** (view balance, deposit, withdraw).
- **Access a suite of Personal Finance Tools** (EMI Calculator, SIP Calculator, FD Calculator, etc.) — *only after successful login*.

Additionally:

- Use **Git** and **GitHub** for version control.
- Track tasks using a **JIRA board**.
- Leverage **GitHub Copilot** responsibly to accelerate coding.
- Write **unit tests** for backend functionality and calculators.

Project Scope

🛠 Core Functionalities

1. **User Registration and Login System**
    - New users can register by providing basic details (name, email, password).
    - Existing users can securely log in.
    - Passwords must be encrypted (use Django's built-in authentication).
2. **Bank Account Management**
    - After login, users can:
        - View current balance.

        - Deposit money.
        - Withdraw money (validate sufficient balance).
    - Transaction history should be recorded (optional bonus).
3. **Access to Financial Tools**
    - Logged-in users can use 10 financial tools:
        - EMI Calculator
        - SIP Calculator
        - FD Calculator
        - RD Calculator
        - Retirement Savings Estimator
        - Home Loan Eligibility Estimator
        - Credit Card Interest Calculator
        - Taxable Income Calculator
        - Budget Planner
        - Net Worth Calculator
    - These tools should be accessible via a simple dashboard menu.
4. **Testing**
    - Develop **unit tests** for critical backend functions:
        - Registration
        - Login
        - Deposit/Withdraw logic
        - Financial tool functions
    - Minimum 80% code coverage.
5. **JIRA**
    - Create tasks/stories for:
        - User authentication
        - Bank operations
        - Financial tools integration

        - Unit testing
        - Deployment (if attempted)

6. **Git & GitHub**
    - Regular commits with meaningful messages.
    - Proper branch management if possible (feature branches).

Technical Requirements

Technology	Usage
Python	Core language
Django	Web framework
SQLite/PostgreSQL	Backend database
HTML/CSS/Bootstrap	Frontend (basic forms and pages)
Pandas	(Optional) Use if any tool needs data manipulation
Unittest or Pytest	Testing
Git + GitHub	Version control
JIRA	Agile task tracking
GitHub Copilot	For responsible code suggestion use

Submission Requirements

- Full Django project folder.
- test_banking.py (unit tests for core banking functions).
- JIRA task board screenshot (showing completed tasks).
- GitHub repository link.
- README.md (setup instructions + feature summary).

✅ **Example User Journey**

A user registers → logs in → deposits ₹10,000 → withdraws ₹3,000 → checks their balance → uses the EMI calculator to plan a loan → uses the SIP calculator to estimate returns on monthly savings.

Loan Amount Estimation Tool

Build a machine learning model that predicts the **maximum loan amount** a customer is eligible for, based on their personal and financial profile. This model will serve as a backend service for an online banking application, allowing users to input their details and receive an immediate loan estimate.

📄 **Inputs from the User:**

1. **Age** (years)
2. **Monthly Income** (in ₹)
3. **Credit Score** (300–850)
4. **Loan Tenure in Years**
5. **Existing Loan Amount** (if any)
6. **Number of Dependents**

🎯 **Output:**

- **Predicted Loan Amount** (in ₹): An estimated loan amount the bank could offer, computed by a trained regression model.

💡 **Use Case:**

This tool will be accessible to authenticated users of the Online Banking System. It helps assess potential borrowing capacity before a formal loan application is submitted. Users can interact via a web form, and the Django backend will return the predicted loan amount.

NOTE: Make sure you perform some EDA and generate some insights