📄 Project Report: Capital Investment Advisor System (CIAS)

1. Introduction

The **Capital Investment Advisor System (CIAS)** is a custom-built software solution designed to streamline and standardize the capital budgeting process. The system automates the calculation of critical financial metrics and incorporates **Monte Carlo Simulation** for probabilistic risk assessment.

2. Problem Statement

Manual capital expenditure (CapEx) decision-making is prone to human error, cognitive biases, and time delays due to complex spreadsheet calculations. Businesses require a systematic, data-driven tool to efficiently evaluate long-term capital projects and quantify associated risks.

3. Functional Requirements

- **FR1: Data Input:** Allow users to input financial data, including initial outlay, project life, cash flow means, standard deviations (for simulation), and discount rate.
- **FR2: Deterministic Analysis:** Automatically calculate **Net Present Value (NPV)**, **Internal Rate of Return (IRR)**, **Payback Period**, and **Profitability Index (PI)** using mean cash flow inputs.
- **FR3: Monte Carlo Simulation:** Execute probabilistic risk assessment using standard deviations over a defined number of runs.
- **FR4: Recommendation:** Generate a final recommendation (**ACCEPT/REJECT**) based on both deterministic NPV and the probability of a positive NPV ($P(NPV>0)$) from the simulation.
- **FR5: Reporting:** Generate detailed output reports summarizing all inputs and results.

4. Non-functional Requirements

- **Accuracy:** Calculation engine must be mathematically validated using established financial libraries (**NumPy Financial**).
- **Performance:** Monte Carlo runs must be executed efficiently (leveraging Python's speed in computation).
- **Maintainability:** Use of a single-language stack (**Python**) ensures simplified debugging and future maintenance.

⚙️ 5. System Architecture

The CIAS is built upon a **three-tier, modular architecture** implemented entirely with a unified Python stack, ensuring maintainability, scalability, and rapid development.

5.1 Unified Python Technology Stack (Architectural Layers)

Component	Technology	Architectural Layer	Role in CIAS
Backend/ Core Logic	**Python (Flask)**	Application/Logic	Manages API endpoints, user sessions, and routing requests.
Calculation Engine	**NumPy & Pandas**	Application/Logic	Core module for highly accurate financial and statistical computations.
Frontend/ UI	**Plotly Dash / Streamlit**	Presentation	Interactive web interface for forms, dashboards, and visualizations.
Database	**SQLAlchemy**	Data/ Persistence	Object-Relational Mapper (ORM) for data storage.

5.2 Architectural Components and Workflow

5.2.1 Data Modeling (`models.py`)

Defines the database schema using **SQLAlchemy** for entities like `User`, `Project`, and `CashFlow`.

5.2.2 Calculation Engine (`calculation_engine.py`)

Utilizes **NumPy Financial** for core metrics and implements the Monte Carlo logic by sampling cash flows based on mean and standard deviation to determine $P(NPV>0)$.

5.2.3 Backend API and Routing (`app.py`)

Implemented using **Flask**, handles communication between the UI and the Calculation/Data layers.

5.2.4 User Interface (`ui_dashboard.py`)

Uses **Plotly Dash/Streamlit** components for data entry and display, with Callbacks connecting user actions to the Flask API.

6. Design Decisions & Rationale

Unified Python Stack: Decided to use Python exclusively to minimize technological overhead, accelerate development, and leverage Python's speed in numerical computing essential for the Monte Carlo analysis.

Monte Carlo Integration: Included to move beyond simple deterministic analysis, offering decision-makers a quantifiable measure of risk ($P(NPV>0)$).

7. Implementation Details

Detailed implementation focused on ensuring the **Calculation Engine** correctly handles variable cash flows and executes the high volume of calculations required for the Monte Carlo simulation quickly using vectorized operations from **NumPy**. The backend Flask API acts as a thin layer to serve the results efficiently.

8. Screenshots / Results

This section presents the output of the CIAS, demonstrating both the standard **Deterministic Analysis** and the advanced **Monte Carlo Simulation** for two distinct scenarios.

8.1 Scenario 1: High-Return Project (ACCEPT Recommendation)

- **Inputs:** Initial Investment: 200,000; Lifetime: 10 years; Discount Rate: 0.7.
- **Deterministic Results:** NPV: **368,594.09**; IRR: **1.99996**; PI: **2.843**.
- **Monte Carlo Simulation Results:** Monte Carlo Mean NPV: **367,759.64**; P(NPV > 0): **1.0**.
- **Recommendation: ACCEPT**

8.2 Scenario 2: High-Risk, Low-Return Project (REJECT Recommendation)

- **Inputs:** Initial Investment: 300,000; Lifetime: 5 years; Discount Rate: 0.9.
- **Deterministic Results:** NPV: **-193,376.23**; IRR: **0.1985**; PI: **0.355**.
- **Monte Carlo Simulation Results:** Monte Carlo Mean NPV: **-193,359.79**; P(NPV > 0): **0.0**.

```
=== Capital Investment Advisor — interactive demo ===
Initial investment (positive number) [500000]: 200000
Lifetime (years) [7]: 10
Annual revenue (mean) [150000]: 500000
Annual cost (mean) [40000]: 100000
Discount rate (decimal, e.g. 0.10) [0.1]: 0.7
Annual revenue std dev [20000]: 56777
Annual cost std dev [5000]: 67899
Monte Carlo runs [3000]: 678

--- Deterministic results ---
NPV: 368594.0957324828
IRR: 1.9999661259997505
Payback (undiscounted): 0.5
Profitability Index: 2.843

--- Monte Carlo simulation ---
Monte Carlo mean NPV: 367759.63977522135
P(NPV>0): 1.0
Recommendation: ACCEPT
```

- **Recommendation: REJECT**

```
=== Capital Investment Advisor — interactive demo ===
Initial investment (positive number) [500000]: 300000
Lifetime (years) [7]: 5
Annual revenue (mean) [150000]: 200000
Annual cost (mean) [40000]: 100000
Discount rate (decimal, e.g. 0.10) [0.1]: 0.9
Annual revenue std dev [20000]: 2000
Annual cost std dev [5000]: 30000
Monte Carlo runs [3000]: 23444

--- Deterministic results ---
NPV: -193376.23414895768
IRR: 0.19857709787320127
Payback (undiscounted): 3.0
Profitability Index: 0.355

--- Monte Carlo simulation ---
Monte Carlo mean NPV: -193359.78518211693
P(NPV>0): 0.0
Recommendation: REJECT
```

9. Testing Approach

The project utilized the **Pytest** framework. **Unit Tests** were performed extensively on the `calculation_engine.py` module, validating the accuracy of the deterministic metrics and the statistical outputs of the Monte Carlo sampler. **Integration Testing** verified seamless data flow from the Dash/Streamlit front-end through the Flask API to the database and back.

10. Challenges Faced

The primary challenge was optimizing the performance of the **Monte Carlo simulation**. This required careful use of vectorized operations within **NumPy** to generate and process thousands of randomized cash flow arrays concurrently.

11. Learnings & Key Takeaways

The project successfully demonstrated that a complex financial advisory system, including advanced risk modeling, can be efficiently developed using a unified Python stack. The most significant takeaway is the value added by moving from point-estimate (deterministic) analysis to **probabilistic risk assessment** via Monte Carlo simulation.

12. Future Enhancements

- **Advanced Risk Distributions:** Allowing users to select alternative probability distributions (e.g., Triangular, Beta) for cash flow variables.
- **Portfolio Optimization:** Integrating optimization algorithms to recommend the best combination of accepted projects given a fixed total capital budget.

- **External Data Integration:** Fetching real-time market interest rates for dynamic adjustment of the discount rate.