**BIT 23015: SOFTWARE PROJECT MANAGEMENT ASSIGNMENT DOCUMENTATION.**

GROUP MEMBERS

DOK MIYOM CHOL SCT221-0448/2021

CHARO ISHMAEL SCT221-0838/2021

HUSSEIN FARAH SCT221-0403/2021

**Finance Calculator App Documentation**

# Overview

The Finance Calculator App is a React-based application designed to evaluate and compare financial metrics of different projects. The app calculates key financial indicators such as Payback Period, Return on Investment (ROI), Net Present Value (NPV), and Internal Rate of Return (IRR) for multiple projects over a specified number of years. The app also determines which project has the highest ROI, making it the best investment option among the evaluated projects.

# Features

- Multiple Projects Evaluation: Users can input the number of projects and specify the cash flows for each project over a chosen number of years.  
- Dynamic Input: The app dynamically adjusts input fields based on the number of projects and years selected.  
- Key Financial Metrics Calculation:  
 - Payback Period: The time it takes for cumulative cash flows to turn positive.  
 - Return on Investment (ROI): The average annual return on the total investment.  
 - Net Present Value (NPV): The present value of future cash flows discounted at a specified rate (default is 10%).  
 - Internal Rate of Return (IRR): The discount rate that makes the NPV of cash flows zero.  
- Best Project Identification: The app identifies and displays the project with the highest ROI.

# User Interface

## Input Fields

1. Number of Projects: Allows the user to specify how many projects they want to evaluate.  
2. Number of Years: Specifies the time horizon (in years) over which the cash flows will be evaluated.  
3. Cash Flow Inputs: For each project, users input cash flows for each year.

## Buttons

- Calculate: Computes the financial metrics based on the provided inputs.

## Output Fields

- Results for Each Project: Displays Payback Period, ROI, NPV, and IRR for each project.  
- Best Project: Highlights the project with the highest ROI.

# Code Structure

## Components

- FinanceCalculator: The main component that manages the state and handles the logic for calculating financial metrics.  
 - State Variables:  
 - `numProjects`: Number of projects.  
 - `numYears`: Number of years.  
 - `cashFlows`: Array storing cash flows for each project.  
 - `results`: Array storing calculated results for each project.  
 - `bestProjectIndex`: Index of the project with the highest ROI.  
 - Handlers:  
 - `handleNumProjectsChange`: Updates the number of projects and adjusts the cash flow input fields.  
 - `handleNumYearsChange`: Updates the number of years and adjusts the cash flow input fields.  
 - `handleInputChange`: Updates the cash flow values for each project and year.  
 - `calculateResults`: Calculates the Payback Period, ROI, NPV, and IRR for each project and determines the best project.

## Utility Functions

- `calculatePayback`: Calculates the payback period for a project.  
- `calculateROI`: Calculates the ROI using the net profit and total expenses over the non-zero cash flow years.  
- `calculateNPV`: Calculates the NPV using the provided cash flows and a discount rate.  
- `calculateIRR`: Estimates the IRR using the Newton-Raphson method.

## Example Usage

```javascript  
<FinanceCalculator />  
```

# Calculation Methods

## Payback Period

The Payback Period is calculated by summing the cash flows until the cumulative cash flow becomes positive.

## Return on Investment (ROI)

ROI is calculated as the net profit divided by the total expenses, averaged over the years with non-zero cash flows.

## Net Present Value (NPV)

NPV is the sum of cash flows discounted at a specified rate. The discount rate is assumed to be 10% in the default implementation.

## Internal Rate of Return (IRR)

IRR is calculated using a numerical method (e.g., Newton-Raphson) to find the discount rate that sets the NPV to zero.

**PRACTICAL 2**

While working in teams of 3, use 4-matrices and SWOT analysis to determine the viability of the software project in practical one.

**4-Matrix Analysis**

1. Impact and Effort.
   * Automating calculations in a spreadsheet has high impact but low effort.
   * A simple user-friendly application has high impact and requires high effort.
   * Basic manual spreadsheet setup has low impact with low effort.
   * Building a complex application with features that are rarely used has low impact and needs high effort.
2. Risk and Reward
   * Spreadsheets are widely used and familiar have high reward with low risk.
   * Developing a new application can offer significant benefits has more uncertainty hence has high reward with high risks too.
   * Manual calculations can have low reward with low risk.
   * Developing an application without a clear user base may have low reward but high risk.
3. Urgency and Importance
   * Automating financial methods for quick and accurate analysis has high urgency and high importance.
   * Addressing immediate but less critical automation needs has high urgency but low importance.
   * Long-term application development for comprehensive financial analysis has low urgency but high importance.
   * Enhancing non-critical features in the application has low urgency and low importance.
4. Complexity and Value
   * Spreadsheet automation is straightforward and highly beneficial hence has high value but low complexity.
   * Advanced application development has high value and high complexity.
   * Spreadsheet has simple enhancements hence low value, low complexity.
   * Developing an application with unnecessary features has low value and high complexity.

**SWOT Analysis**

Strengths

Spreadsheet Automation:

* + Most users are familiar with spreadsheet.
  + Simple to implement and use.
  + No need for extensive development costs.

Application Development:

* + Tailored to specific user needs and can be scaled with additional features.
  + More polished and professional than spreadsheets.
  + Can integrate with other software and databases.

Weaknesses

Spreadsheet Automation:

* + Less flexible and harder to scale for more complex calculations.
  + Prone to errors in manual data entry and formula setup.

Application Development:

* + Requires significant time and resources to develop.
  + Needs ongoing maintenance and support.

Opportunities

Spreadsheet Automation:

* + Opportunity to offer training on financial analysis using spreadsheets.
  + Create and distribute templates for wider use.

Application Development:

* + Potential to reach a larger audience that require advanced financial tools.
  + Can continuously add features based on user feedback and technological advancements.

Threats

Spreadsheet Automation:

* + Many free templates and tutorials are available online.
  + Dependence on spreadsheet software capabilities and updates.

Application Development:

* Many financial analysis tools are already available.
* Rapid changes in technology may render the application obsolete.

For immediate and simple needs, spreadsheet automation is highly viable. For long-term and advanced requirements, developing a custom application is more beneficial.