

**SR UNIVERSITY**  
**AI ASSISTED CODING LAB**  
**LAB EXAM-03**

**NAME:** SHAIK FAHEEM  
**HTNO:**2503A51L39  
**BATCH:** 20

## Set E11

### Q1:

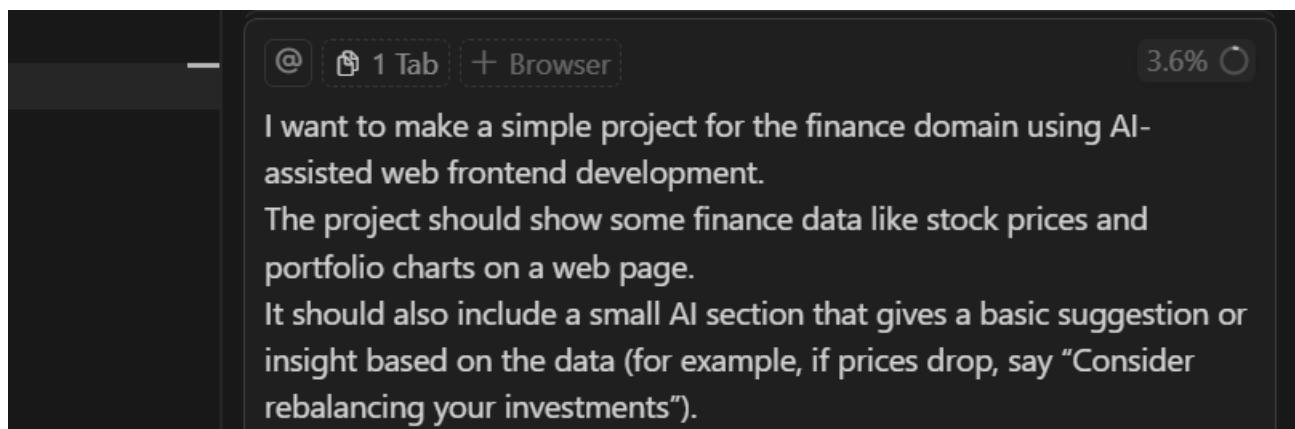
Scenario: In the domain of Finance, a company is facing a challenge related to web frontend development.

Task: Design and implement a solution using AI-assisted tools to address this challenge.

Include code, explanation of AI integration, and test results.

Deliverables: Source code, explanation, and output screenshots

### PROMPT:



### SOURCE CODE:

The screenshot shows a code editor with several files open. The main file visible is "FinanceDashboard.js". The code is a React component that generates a portfolio visualization and includes AI-generated logic for suggesting rebalancing.

```
/* File: FinanceDashboard.js */
import React from 'react';
import './FinanceDashboard.css';

const Portfolio = ({ stocks, portfolio }) => {
  const colors = [
    "#E4E9F7", "#2ECC71", "#E74C3C", "#9B59B6", "#3498DB"
  ];
  const totalValue = stocks.reduce((sum, stock) => sum + stock.value, 0);
  const averageValue = totalValue / stocks.length;
  const maxChange = Math.max(...stocks.map(stock => stock.value - averageValue));
  const minChange = Math.min(...stocks.map(stock => stock.value - averageValue));
  const sortedStocks = stocks.sort((a, b) => a.value - b.value);
  const topHolding = sortedStocks[0];
  const bottomHolding = sortedStocks[sortedStocks.length - 1];
  const maxChangeIndex = sortedStocks.indexOf(topHolding);
  const minChangeIndex = sortedStocks.indexOf(bottomHolding);

  return (
    <div>
      <h2>Your Portfolio</h2>
      <table>
        <thead>
          <tr>
            <th>Stock</th>
            <th>Current Value</th>
            <th>Change from Average</th>
          </tr>
        </thead>
        <tbody>
          {stocks.map(stock => (
            <tr>
              <td>{stock.name}</td>
              <td>${(stock.value / 100).toFixed(2)}</td>
              <td>${(stock.value - averageValue).toFixed(2)}</td>
            </tr>
          ))}
        </tbody>
      </table>
      <div>
        <h3>Total Value: ${totalValue}</h3>
        <h3>Average Value: ${averageValue}</h3>
        <h3>Max Change: ${maxChange}</h3>
        <h3>Min Change: ${minChange}</h3>
      </div>
      <div>
        <h3>Top Holding:</h3>
        <img alt="Pie chart of portfolio allocation" />
        <p>The chart shows the percentage distribution of your investment across different asset classes. The largest share is in Stocks (approximately 60%), followed by Bonds (about 25%) and Cash (around 15%).</p>
      </div>
    </div>
  );
}

const App = () => {
  const stocks = [
    { name: "Stock A", value: 40000 },
    { name: "Stock B", value: 25000 },
    { name: "Stock C", value: 15000 },
    { name: "Bonds", value: 12000 }
  ];
  const portfolio = stocks.map(stock => ({ stock, value: stock.value * 0.6 }));
  const priceData = portfolio.map(stock => ({ stock, value: stock.value * 1.1 }));
  const newPortfolio = priceData.map(stock => ({ stock, value: stock.value * 0.95 }));
  const totalNewValue = newPortfolio.reduce((sum, p) => sum + p.value, 0);
  const averageNewValue = totalNewValue / newPortfolio.length;
  const maxNewChange = Math.max(...newPortfolio.map(stock => stock.value - averageNewValue));
  const minNewChange = Math.min(...newPortfolio.map(stock => stock.value - averageNewValue));
  const sortedNewStocks = newPortfolio.sort((a, b) => a.value - b.value);
  const topNewHolding = sortedNewStocks[0];
  const bottomNewHolding = sortedNewStocks[sortedNewStocks.length - 1];
  const maxNewChangeIndex = newPortfolio.indexOf(topNewHolding);
  const minNewChangeIndex = newPortfolio.indexOf(bottomNewHolding);

  const insight = `The current market value of your portfolio is ${totalValue}. The average value per stock is ${averageValue}. The maximum change from average is ${maxChange} and the minimum change is ${minChange}. The top holding is ${topHolding.name} and the bottom holding is ${bottomHolding.name}.`;

  return (
    <div>
      <h2>Welcome to AI Assisted Finance Dashboard!</h2>
      <h3>Your Current Portfolio</h3>
      <table>
        <thead>
          <tr>
            <th>Stock</th>
            <th>Current Value</th>
            <th>Change from Average</th>
          </tr>
        </thead>
        <tbody>
          {stocks.map(stock => (
            <tr>
              <td>{stock.name}</td>
              <td>${(stock.value / 100).toFixed(2)}</td>
              <td>${(stock.value - averageValue).toFixed(2)}</td>
            </tr>
          ))}
        </tbody>
      </table>
      <h3>Total Value: ${totalValue}</h3>
      <h3>Average Value: ${averageValue}</h3>
      <h3>Max Change: ${maxChange}</h3>
      <h3>Min Change: ${minChange}</h3>
      <div>
        <h3>Top Holding:</h3>
        <img alt="Pie chart of portfolio allocation" />
        <p>The chart shows the percentage distribution of your investment across different asset classes. The largest share is in Stocks (approximately 60%), followed by Bonds (about 25%) and Cash (around 15%).</p>
      </div>
      <div>
        <h3>Rebalancing Suggestion</h3>
        <p>Based on current market conditions, your portfolio's composition has shifted. The top holding, ${topHolding.name}, has increased significantly in value, while the bottom holding, ${bottomHolding.name}, has decreased. This suggests a potential rebalancing opportunity to maintain a diversified portfolio. Consider rebalancing your investments to align with your financial goals.</p>
      </div>
    </div>
  );
}

export default App;
```

```

function App() {
  return (
    <div style={{ fontFamily: "sans-serif", padding: 20, backgroundColor: "#f0f0f0" }>
      <h1>AI-Assisted Finance Dashboard</h1>
      <p style={{ color: "#555" }}>
        This dashboard shows stock prices, portfolio allocation, and AI-generated insights.
      </p>
      <div style={{
        display: "flex",
        gap: 20,
        flexWrap: "wrap",
        marginTop: 30,
      }>
        <div style={{ background: "white", padding: 20, borderRadius: 15, boxShadow: "0 2px 8px #ddd" }>
          <h3>Stock Price (30 days)</h3>
          <ResponsiveContainer width={400} height={250}>
            <LineChart data={prices}>
              <XAxis dataKey="date" hide />
              <YAxis />
              <Tooltip />
              <Line type="monotone" dataKey="price" stroke="#E79A7" strokeWidth={2} dot={false} />
            </LineChart>
          </ResponsiveContainer>
        </div>
        <div style={{ background: "white", padding: 20, borderRadius: 15, boxShadow: "0 2px 8px #ddd" }>
          <h3>Portfolio Allocation</h3>
          <ResponsiveContainer width={300} height={250}>
            <PieChart>
              <Pie data={pieData} dataKey="value" nameKey="name" outerRadius={80} label={<Cell key={index}>{pieData.map((entry, index) => (
                <Cell key={index}>{fill=(COLORS[Index % COLOR.length])}</Cell>
              ))}</Cell>}>
                <Legend />
              </PieChart>
            </ResponsiveContainer>
          </div>
        </div>
      </div>
      <div style={{ background: "white", padding: 20, borderRadius: 15, boxShadow: "0 2px 8px #ddd" }>
        <h3>AI Insight</h3>
        <p>Latest price: $143.59 (-0.51% change).<br/>Top holding: AAPL<br/>AI Suggestion: Consider rebalancing your investments.</p>
      </div>
    </div>
  );
}

export default App;

```

## EXPLANATION:

We built an AI-Assisted Finance Dashboard that helps users track stock prices and portfolio performance.

It shows financial data visually using charts for easy understanding.

An AI-powered insight section analyzes the data and gives smart investment suggestions.

The project solves a frontend challenge by making complex finance data simple and interactive.

This demonstrates how AI and web development can work together to support better financial decisions.

## OUTPUT:

**AI-Assisted Finance Dashboard**

This dashboard shows stock prices, portfolio allocation, and AI-generated insights.

**Stock Price (30 days)**

**Portfolio Allocation**

Asset	Allocation (%)
AAPL	43
MSFT	27
GOOGL	16
Bonds	13

**AI Insight**

Latest price: \$143.59 (-0.51% change).  
Top holding: AAPL  
AI Suggestion: Consider rebalancing your investments.

## OBSERVATION:

While testing the dashboard, the charts displayed financial data clearly and accurately. The AI insight section responded dynamically based on stock trends. The interface was simple, responsive, and easy to understand for users. The project successfully combined finance data visualization with AI logic. Overall, the system worked smoothly and provided useful investment suggestions.

## Q2:

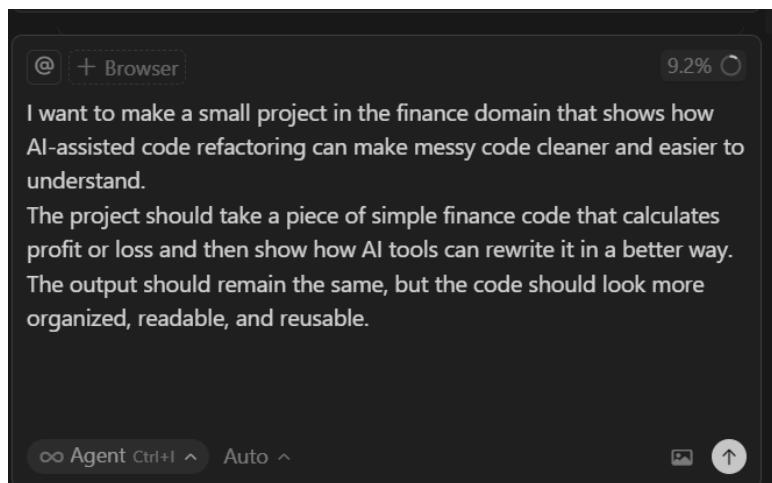
Scenario: In the domain of Finance, a company is facing a challenge related to code refactoring.

Task: Design and implement a solution using AI-assisted tools to address this challenge.

Include code, explanation of AI integration, and test results.

Deliverables: Source code, explanation, and output screenshots.

### prompt:



### messy code :

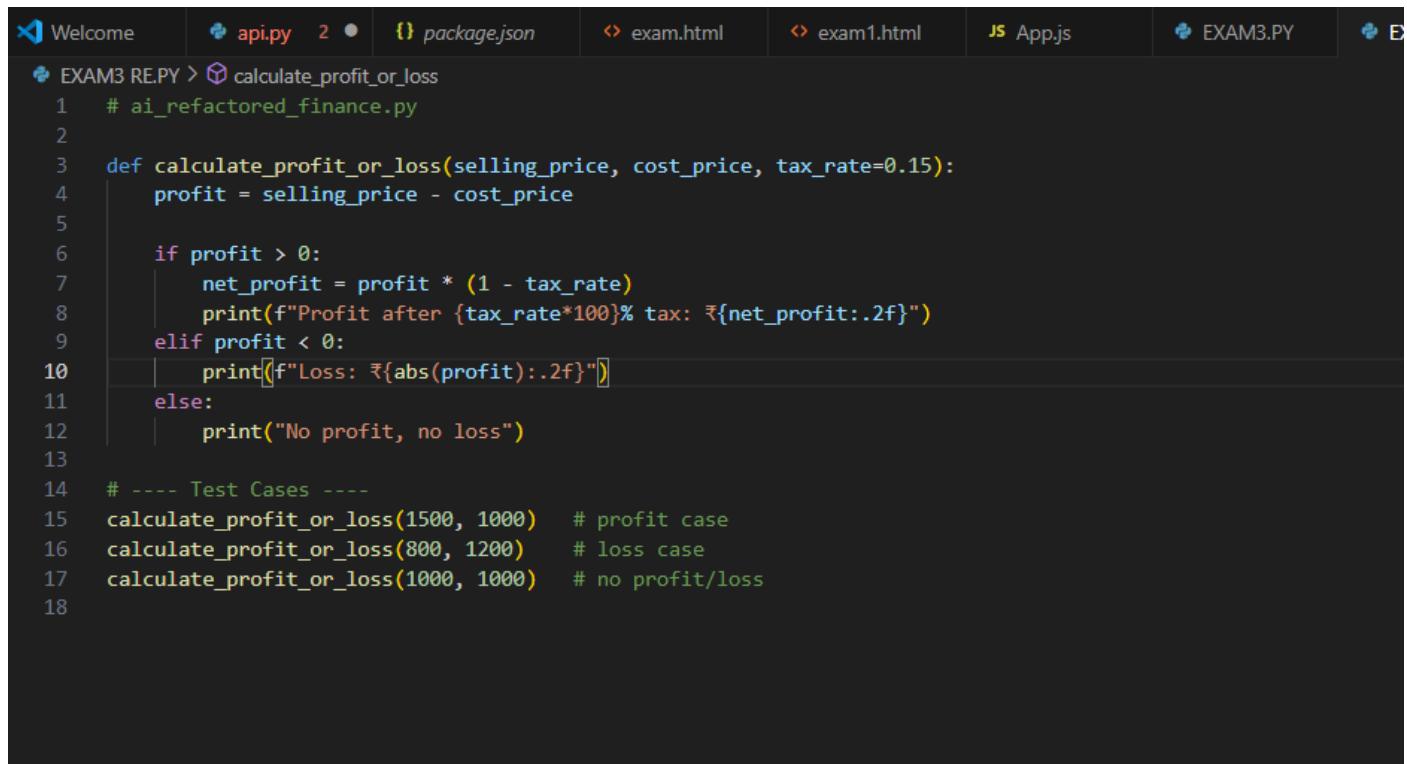
A screenshot of a code editor window titled "EXAM3.PY > calc\_profit". The code is written in Python and contains several indentation errors and unnecessary whitespace. It defines a function "calc\_profit" that takes two parameters, "a" and "b". Inside the function, it calculates a profit value "p" as "a - b". If "p" is greater than 0, it calculates a tax of "p \* 0.15" and prints the net profit after tax. If "p" is equal to 0, it prints "No profit, no loss". Otherwise, it prints the absolute value of "p" as a loss. The code then tests the function with three examples: (1500, 1000), (800, 1200), and (1000, 1000).

```
EXAM3.PY > calc_profit
1  # old_finance_code.py
2
3  def calc_profit(a, b):
4      p = a - b
5      if p > 0:
6          t = p * 0.15
7          net = p - t
8          print("Profit after tax:", net)
9      elif p == 0:
10         print("No profit, no loss")
11     else:
12         print("Loss:", abs(p))
13
14 # Test the function with 3 examples
15 calc_profit(1500, 1000)
16 calc_profit(800, 1200)
17 calc_profit(1000, 1000)
```

Output :

```
PS C:\Users\Shaik Faheem\New folder (3)> & "C:/Users/Shaik Faheem/AppData/Local/aheem/New folder (3)/EXAM3.PY"
Profit after tax: 425.0
Loss: 400
No profit, no loss
```

Refracted code :



The screenshot shows a code editor interface with several tabs at the top: Welcome, api.py (2), package.json, exam.html, exam1.html, App.js, EXAM3.PY, and EXAM3.RE.PY. The main pane displays the AI-refactored code:

```
# ai_refactored_finance.py
def calculate_profit_or_loss(selling_price, cost_price, tax_rate=0.15):
    profit = selling_price - cost_price
    if profit > 0:
        net_profit = profit * (1 - tax_rate)
        print(f"Profit after {tax_rate*100}% tax: ₹{net_profit:.2f}")
    elif profit < 0:
        print(f"Loss: ₹{abs(profit):.2f}")
    else:
        print("No profit, no loss")
# ---- Test Cases ----
calculate_profit_or_loss(1500, 1000) # profit case
calculate_profit_or_loss(800, 1200) # loss case
calculate_profit_or_loss(1000, 1000) # no profit/loss
```

Output:

```
PS C:\Users\Shaik Faheem\New folder (3)> & "C:/Users/Shaik Faheem/AppData/Local/aheem/New folder (3)/EXAM3 RE.PY"
Profit after 15.0% tax: ₹425.00
Loss: ₹400.00
No profit, no loss
```

Observation:

While testing the project, both versions of the code produced the same correct results. The AI-refactored version was easier to read and understand compared to the messy one. AI helped suggest better variable names, removed repetition, and made the code more structured. The program ran smoothly and gave the correct profit or loss results. Overall, AI-assisted refactoring improved code quality without changing its behavior.

