Financial frontier

Financial planning and investment guidance

Curated by

241001076-HARISH AKSHAY H
241001065-GOPIKA S
241001081-HARISH S

ABSTRACT

Input Handling :

Accept user's monthly salary And currency (INR, USD ,EUR , or JPY) and Convert the salary to Indian rupees (INR) using predefined conversion rates

Investment personalization :

The program tailors investment suggestion based on the user's financial goals and risk appetite – GOLD SELECTION : Short-Term or long term – RISK APPETITE : Low , Moderate, or high risk.

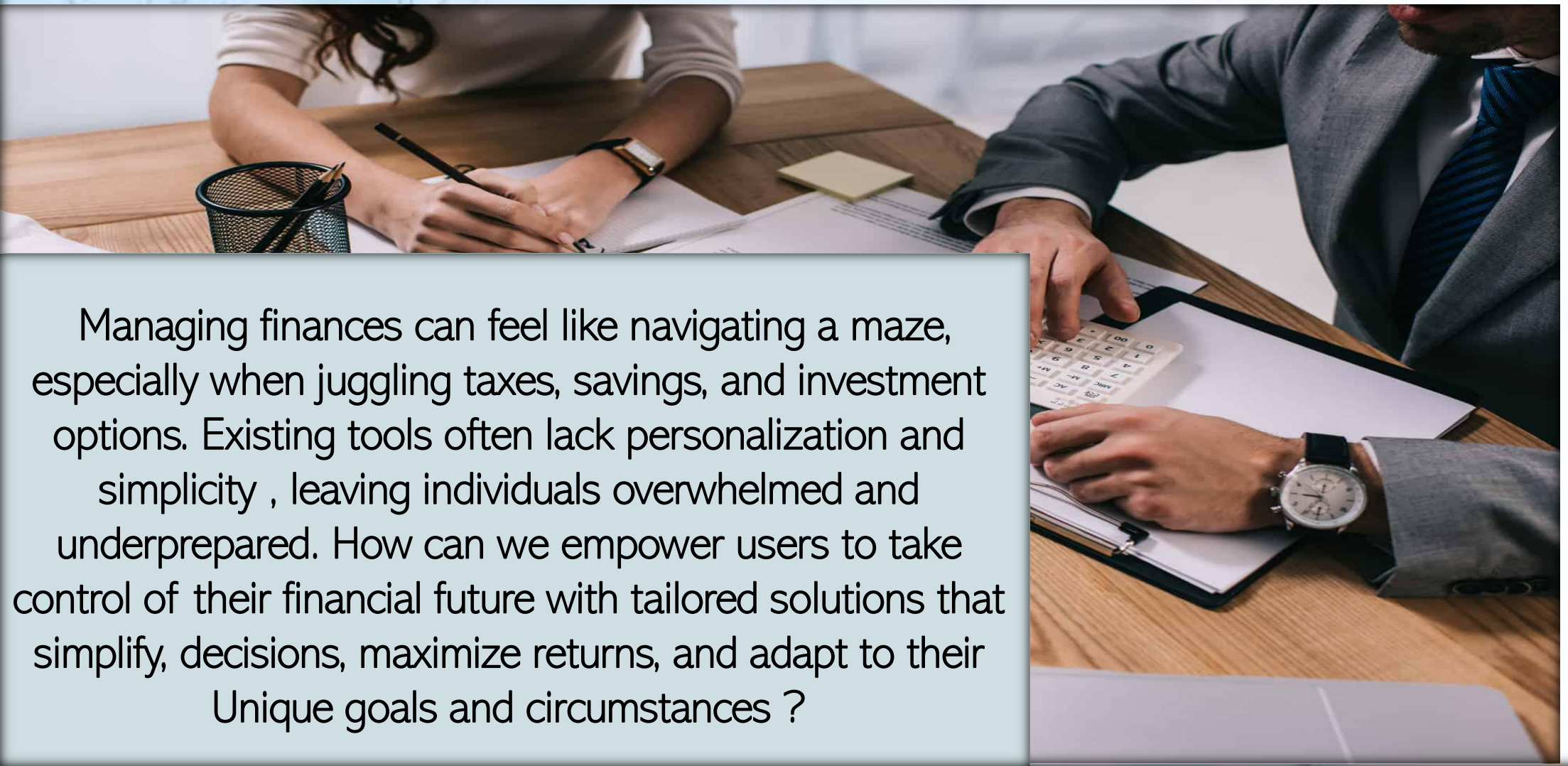
Tax and Investment calculation :

Applies India's Progressive tax system to calculate the total tax liability and determines the user's Savings as 20% of the post-tax income . Extract 7% as the investable amount .

Output and suggestion :

Display a breakdown of the user's salary, taxes, savings, and the investable amount. Suggests a diversified allocation of the Investable amount, such as 50% in fixed Deposits, 30% in gold, and 20% in the stock market

PROBLEM STATEMENT



Managing finances can feel like navigating a maze, especially when juggling taxes, savings, and investment options. Existing tools often lack personalization and simplicity , leaving individuals overwhelmed and underprepared. How can we empower users to take control of their financial future with tailored solutions that simplify, decisions, maximize returns, and adapt to their Unique goals and circumstances ?

CODE :

Convert salary to INR

```c
// Function to convert salary to INR
double convertToINR(double salary, char currency[]) {
    if (strcmp(currency, "USD") == 0) {
        return salary * 82.5; // Example: 1 USD = 82.5 INR
    } else if (strcmp(currency, "EUR") == 0) {
        return salary * 90.0; // Example: 1 EUR = 90 INR
    } else if (strcmp(currency, "JPY") == 0) {
        return salary * 0.55; // Example: 1 JPY = 0.55 INR
    } else if (strcmp(currency, "INR") == 0) {
        return salary;
    } else {
        printf("Invalid currency input. Assuming salary is in INR.\n");
        return salary;
    }
}
```

Display investment options

```
// Function to calculate tax based on income slabs
double calculateTax(double annualIncome) {
    double tax = 0.0;

    if (annualIncome > 1500000) {
        tax += (annualIncome - 1500000) * 0.30;
        annualIncome = 1500000;
    }
    if (annualIncome > 1200000) {
        tax += (annualIncome - 1200000) * 0.20;
        annualIncome = 1200000;
    }
    if (annualIncome > 1000000) {
        tax += (annualIncome - 1000000) * 0.15;
        annualIncome = 1000000;
    }
    if (annualIncome > 700000) {
        tax += (annualIncome - 700000) * 0.10;
    }

    return tax;
}
```

Display investment options

```c
// Function to display investment options based on user selection
void displayInvestmentOptions(InvestmentOption options[], int size) {
    for (int i = 0; i < size; i++) {
        printf("%d. %s: %s\n", i + 1, options[i].option, options[i].description);
    }
}
```

Get valid input

```c
// Function to ask for valid input (ensures the input is within a range)
int getValidInput(int min, int max, const char *prompt) {
    int choice;
    while (1) {
        printf("%s", prompt);
        if (scanf("%d", &choice) != 1 || choice < min || choice > max) {
            // Clear the buffer if the input is invalid
            while(getchar() != '\n');
            printf("Invalid input. Please enter a valid choice between %d and %d.\n", min, max);
        } else {
            break;
        }
    }
    return choice;
}
```

Suggest investment scheme and allocate the investable amount from savings

```c
// Function to suggest investment schemes based on investable amount from savings
void suggestInvestmentSchemes(double investableFromSavings) {
    int goal, riskAppetite;

    // Get valid input for investment goal
    printf("\n--- Investment Personalization ---\n");
    goal = getValidInput(1, 2, "Select your investment goal:\n1. Short-term (1-3 years)\n2. Long-term (5+ years)\nEnter your choice (1 or 2): ");

    // Get valid input for risk appetite
    printf("\nSelect your risk appetite:\n");
    riskAppetite = getValidInput(1, 3, "1. Low Risk\n2. Moderate Risk\n3. High Risk\nEnter your choice (1, 2, or 3): ");

    printf("\n--- Personalized Investment Options ---\n");
```

```c
    // Suggest how to allocate the investable amount from savings
    printf("\n--- Suggested Allocation ---\n");
    printf("Consider allocating your investable amount across 2-3 options for diversification.\n");
    printf("Example: 50%% in Fixed Deposits, 30%% in Gold, and 20%% in Share market.\n");

    if (investableFromSavings < 1000) {
        printf("Consider saving more before investing to access better options.\n");
    } else {
        printf("Investable Amount from Savings: %.2lf\n", investableFromSavings);
        printf("Suggested Split:\n");
        printf("  - %.2lf in Fixed Deposits\n", investableFromSavings * 0.5);
        printf("  - %.2lf in Gold\n", investableFromSavings * 0.3);
        printf("  - %.2lf in Share Market\n", investableFromSavings * 0.2);
    }
}
```

Store investment options

```c
// Struct to store investment options
typedef struct {
    char option[100];
    char description[300];
} InvestmentOption;

// Array to store investment options
InvestmentOption shortTermLowRisk[] = {
    {"Fixed Deposits", "Guaranteed returns, no risk."},
    {"Savings Account", "Easy access to funds."}
};
InvestmentOption shortTermModerateRisk[] = {
    {"Debt Mutual Funds", "Lower risk than equities with better returns than FDs."},
    {"Recurring Deposits", "Steady growth with moderate returns."}
};
InvestmentOption shortTermHighRisk[] = {
    {"Stock Trading", "High risk but potential for quick gains."},
    {"Gold ETFs", "Hedge against inflation with moderate growth potential."}
};
InvestmentOption longTermLowRisk[] = {
    {"Government Bonds", "Safe with steady returns."},
    {"Public Provident Fund (PPF)", "Tax-saving long-term plan."}
};
InvestmentOption longTermModerateRisk[] = {
    {"Balanced Mutual Funds", "Diversified investments with moderate risk."},
    {"National Pension Scheme (NPS)", "Ideal for retirement planning."}
};
InvestmentOption longTermHighRisk[] = {
    {"Equity Mutual Funds", "High returns with long-term growth."},
    {"Direct Stock Investment", "High risk, but significant growth potential over time."}
};
```

output

```
Enter your monthly salary amount: 50000
Enter the currency (INR/USD/EUR/JPY): INR

--- Income Details ---
Monthly Salary in INR: 50000.00
Annual Income in INR: 600000.00

--- Salary Breakdown ---
Taxes: 0.00
Amount after reducing taxes: 600000.00
Savings : 120000.00
Amount to invest from savings : 8400.00
```

```
--- Investment Personalization ---
Select your investment goal:
1. Short-term (1-3 years)
2. Long-term (5+ years)
Enter your choice (1 or 2): 1

Select your risk appetite:
1. Low Risk
2. Moderate Risk
3. High Risk
Enter your choice (1, 2, or 3): 2

--- Personalized Investment Options ---
1. Debt Mutual Funds: Lower risk than
    equities with better returns than FDs.
2. Recurring Deposits: Steady growth with
    moderate returns.
```

```
--- Suggested Allocation ---
Consider allocating your investable amount
    across 2-3 options for diversification
    .
Example: 50% in Fixed Deposits, 30% in
    Gold, and 20% in Share market.
Investable Amount from Savings: 8400.00
Suggested Split:
  - 4200.00 in Fixed Deposits
  - 2520.00 in Gold
  - 1680.00 in Share Market


=== Code Execution Successful ===
```

FORMULAS

Converted Salary (INR)	Salary × 82.5
Annual income	Monthly salary × 12
Investable Amount	Annual income - taxes
Savings	20% of investable amount
Investable from savings	7% of savings

Suggested Investment allocation

Example split :

50% in Fixed Deposits

30% in Gold

20% in Stock Market

ADVANTAGES OF THE FINANCIAL FRONTIER PROJECT

1. **PERSONALIZED INVESTMENT SUGGESTIONS :**
Tailored recommendations based on user goals and risk Preferences for smarter financial planning.

2. **COMPREHENSIVE FINANCIAL ANALYSIS :**
Detailed breakdown of income, taxes, savings, and Investable amounts to empower decisions-making.

3. **GLOBAL USABILITY :**
Supports multi-currency input and conversion for diverse User bases

4. **SMART SAVINGS AND RISK ASSESSMENT :**
Promotes savings and aligns investment strategies With risk appetite scalable and future-ready

5. **MODULAR DESIGN :**
Allows easy integration of advanced features like live Exchange rates or AI- driven insights

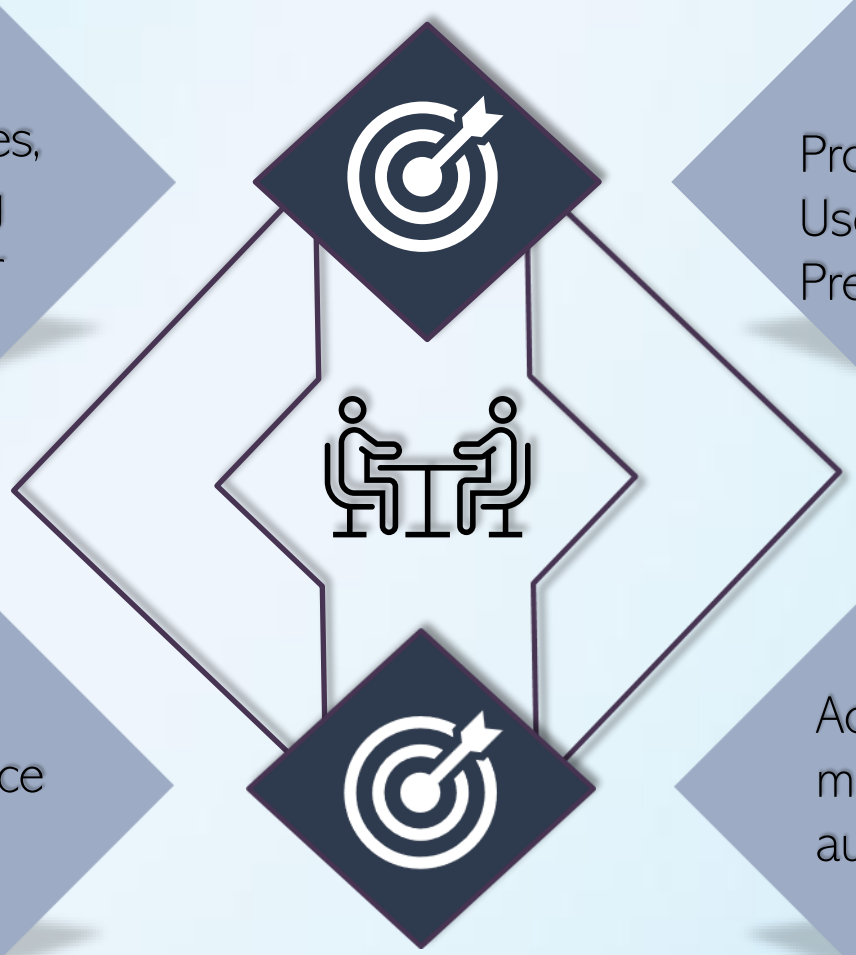
Advantages over existing solutions

SIMPLIFIED AND ACTIONABLE INSIGHTS

Offers a clear breakdown of taxes, Savings, and investment, making Financial planning easy and user friendly

PERSONALIZED INVESTMENT PLAN

Provides tailored suggestions based User-specific goals and risk Preferences , unlike generic tools



FUTURE-READY DESIGN

Modular structure enables seamless integration with advance features, keeping it adaptable and scalable

GLOBAL COMPATIBILITY

Accepts and converts salaries from multiple currencies, catering to a audience seamlessly

Usage of C language in my project :

USAGE OF DATA STRUCTURES - ARRAYS

MATHEMATICAL COMPUTATION

STRING HANDLING FUNCTIONS

CONTROL STRUCTURES

POINTER AND ERROR HANDLING

Conclusion

This program effectively introduces users to financial literacy concepts like risk profiling , tax computation , and investment diversification. It serves as a useful tool for individuals looking to plan their savings and investment more

THANK YOU