Developer Documentation

Explanation of the Solution

The Bond Portfolio Calculator is a C program designed to manage a portfolio of bonds. It allows users to add, delete, and view bonds in the portfolio, perform scenario analysis, and save/load the portfolio to/from a file. The program uses dynamic memory allocation to handle the portfolio of bonds and provides various financial calculations such as bond price and yield to maturity (YTM).

Modules

1. **bond.h / bond.c**: Defines the `Bond` structure and functions for creating bonds, calculating bond prices, and calculating YTM.
2. **portfolio.h / portfolio.c**: Defines the `Portfolio` structure and functions for managing the portfolio, including adding, deleting, and viewing bonds.
3. **file_handler.h / file_handler.c**: Functions for saving and loading the portfolio to/from a text file.
4. **scenario.h / scenario.c**: Functions for performing scenario analysis on the portfolio.
5. **utils.h / utils.c**: Utility functions for input validation, waiting for user input, and clearing the screen.
6. **main.h / main.c**: The main program logic, including the user interface and menu handling.

Data Structures

- **Bond**: Represents a bond with attributes such as identifier, face value, coupon rate, years to maturity, frequency of payments, and discount rate.
- **Portfolio**: Represents a portfolio of bonds using a dynamic array to store the bonds and tracks the number of bonds and the capacity of the array.

Algorithms

- **Bond Price Calculation**: Calculates the present value of future cash flows (coupon payments and face value) discounted at the bond's discount rate.
- **Yield to Maturity (YTM) Calculation**: Uses the Newton-Raphson method to iteratively approximate the YTM of a bond.

Functions and Interfaces

bond.h / bond.c

- `Bond create_bond()` : Creates a new bond from user input.
- `double calculate_bond_price(const Bond *bond)` : Calculates the price of a bond.
- `double calculate_bond_ytm(const Bond *bond)` : Calculates the Yield to Maturity (YTM) of a bond.
- `void display_bond(const Bond *bond)` : Displays the details of a bond.

portfolio.h / portfolio.c

- `Portfolio create_portfolio()` : Initializes an empty portfolio.
- `void add_bond(Portfolio *portfolio, const Bond *bond)` : Adds a new bond to the portfolio.
- `void add_new_bond_to_portfolio(Portfolio *portfolio)` : Adds a new bond to the portfolio based on user input.
- `double calculate_portfolio_value(const Portfolio *portfolio)` : Calculates the total value of the portfolio.
- `void view_portfolio_summary(const Portfolio *portfolio)` : Displays a summary of the portfolio.
- `void delete_bond(Portfolio *portfolio, const char *identifier)` : Deletes a bond from the portfolio.
- `void free_portfolio(Portfolio *portfolio)` : Frees the memory allocated for the portfolio.

file_handler.h / file_handler.c

- `void save_portfolio(const Portfolio *portfolio)` : Saves the portfolio to a text file.
- `void load_portfolio(Portfolio *portfolio)` : Loads the portfolio from a text file.

scenario.h / scenario.c

- `void perform_scenario_analysis(const Portfolio *portfolio)` : Performs scenario analysis on the portfolio.
- `void perform_interest_rate_scenario_analysis(const Portfolio *portfolio, double interest_rate_change)` : Performs interest rate scenario analysis on the portfolio.

utils.h / utils.c

- `int validate_input(double value, double min, double max)` : Validates input within a specified range.
- `void wait_for_user()` : Waits for user input before continuing.
- `void clear_screen()` : Clears the terminal screen.

main.h / main.c

- `void display_main_menu()` : Displays the main menu.
- `int get_user_choice()` : Gets the user's menu choice.

Testing Documentation

The program has been tested with various scenarios to ensure correctness and robustness. The following tests were performed:

1. **Adding Bonds**: Tested adding multiple bonds to the portfolio and verified the details.
2. **Deleting Bonds**: Tested deleting bonds from the portfolio and verified the remaining bonds.
3. **Calculating Bond Price and YTM**: Tested the calculation of bond prices and YTM for different bonds.
4. **Scenario Analysis**: Tested the scenario analysis feature with different interest rate changes.

5. **Saving and Loading Portfolio**: Tested saving the portfolio to a file and loading it back.

All tests passed successfully, and the program behaves as expected.