Approach Document

Design Principles

The application is built with the following principles:

- **Modularity**: Classes like **Expense** and **ExpenseTracker** are designed to be modular, code reusability and maintainability.
- **User Interaction**: A simple CLI ensures ease of use for users, focusing on functionality without unnecessary complexity.
- **Data Integrity**: Input validation and error handling are implemented to maintain data integrity and ensure the application's robustness.

Testing Strategy

Unit Test : Unit tests focus on testing individual components (classes and methods) in isolation to verify their correctness.

- **Expense Class**: Tests include creation from/to dictionary methods (to_dict, from_dict).
- **ExpenseTracker Class**:
    - Tests for adding, editing, and deleting expenses.
    - Tests for listing expenses and verifying correct outputs.
    - Tests for saving to and loading from a file.

Integration Tests : Integration tests ensure that different components of the application work together seamlessly.

- **Method Interactions**: Tests scenarios where multiple methods of **ExpenseTracker** are used in sequence.
- **File Operations**: Tests saving expenses to a file and loading them back to ensure data integrity.

Edge Case : Special attention is given to edge cases to ensure robustness and error handling:

- Adding/editing/deleting expenses at boundary conditions (first, last, middle indexes).
- Handling empty expense lists gracefully.
- File I/O errors (invalid JSON format).

Challenges Faced and Solutions Implemented

Challenge: File Handling

Solution: Implemented robust file handling using os.path for existence checks and json module for serialisation/deserialization.

Challenge: Data Validation

Solution: Implemented input validation in methods to ensure valid data types and ranges, preventing invalid data from corrupting the application state.

Challenge: Testing Isolation

Solution: Utilised **pytest.fixture** to manage test dependencies, ensuring tests run in isolation.

Assumptions

- **Single User Environment**: Assumes the application is used in a single-user context without concurrent access concerns.
- **Date Format**: Uses `%Y-%m-%d` format for dates consistently throughout the application.

Additional Features

- **Date Handling**: Implemented automatic date assignment when editing expenses to maintain consistency.
- **Verbose Listing**: Enhanced list_expenses method to provide detailed output of expenses including index, amount, category, description, and date.