1. Title Page

Project Title: Smart Daily Expense Tracker
Student Name: Raghav Gupta
Instructor: Dr. Tanu Singh
SAP ID: 590025486
Batch Number: 38

---

2. Abstract

This project presents a simple and modular C program designed to help users track their daily expenses. The program accepts multiple expense entries consisting of category names and amounts, calculates the total spending, and compares it with a daily budget limit provided by the user. It then generates a clear report summarizing the expenses and indicating whether the user is under or over their budget. The project demonstrates key C programming concepts including structures, modular design through header and source files, and standard input processing. The implementation follows the exact GitHub repository and documentation structure prescribed by the UPES Major Project Guidelines.

---

3. Problem Definition

People commonly lose track of how much money they spend on small, everyday purchases. This leads to unplanned overspending and difficulty in maintaining a budget. A simple tool to record daily expenses can help users understand their spending habits and stay within financial limits.

The objective of this project is to:

- Read a list of daily expenses
- Store the expenses in a structured format
- Calculate the total amount spent
- Compare it with the daily budget
- Display a clear, readable report

The solution must be lightweight, modular, and easily testable using standard input redirection.

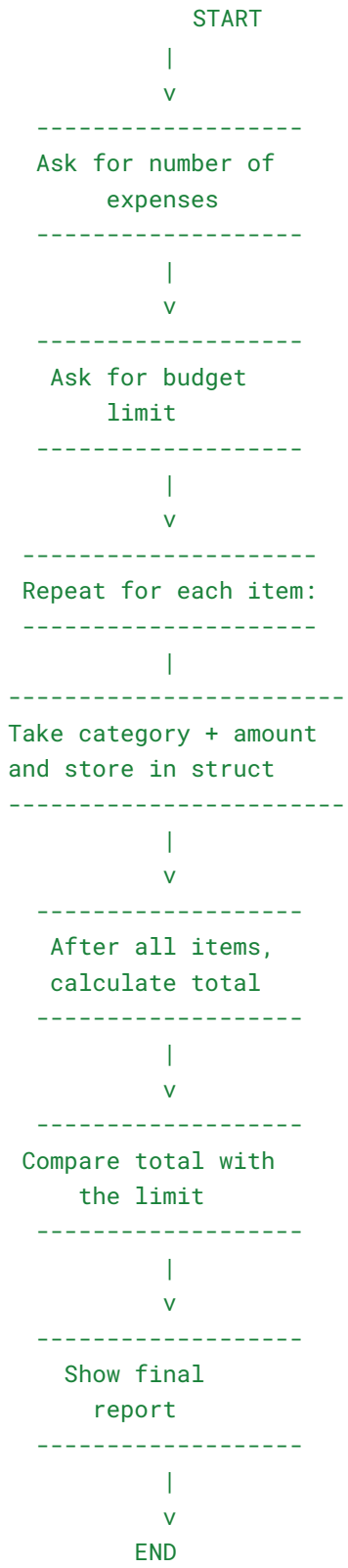
---

4. System Design

4.1 Algorithm

1. Start
2. Read the number of expenses
3. Read the daily budget limit
4. For each expense:
    - Read category
    - Read amount
    - Store in a structure
5. Calculate total spending
6. Compare total with budget
7. Display formatted report
8. End

4.2 Flowchart



5. Implementation Details (with snippets)

The project is implemented using a modular structure with separate header and source files.

5.1 Expense Structure

```
typedef struct {
    char category[20];
    float amount;
} Expense;
```

This structure stores the name and amount of each daily expense.

5.2 Reading Input

```
void inputExpenses(Expense expenses[], int *count, float *limit) {
    scanf("%d", count);
    scanf("%f", limit);

    for (int i = 0; i < *count; i++) {
        scanf("%19s %f", expenses[i].category, &expenses[i].amount);
    }
}
```

This function reads all expenses through standard input.

5.3 Calculating Total

```
float calculateTotal(const Expense expenses[], int count) {
    float total = 0;
    for (int i = 0; i < count; i++)
        total += expenses[i].amount;
    return total;
}
```

5.4 Printing the Report

```
void printReport(const Expense expenses[], int count, float total,
float limit) {
    printf("----------- DAILY REPORT -----------\n");
    for (int i = 0; i < count; i++) {
        printf("%s\t%.2f\n", expenses[i].category,
expenses[i].amount);
    }
    printf("Total Spending: %.2f\n", total);
    printf("Budget Limit: %.2f\n", limit);
}
```

5.5 Main Function

```
int main() {
    Expense expenses[50];
    int count;
    float limit;

    inputExpenses(expenses, &count, &limit);
    float total = calculateTotal(expenses, count);
    printReport(expenses, count, total, limit);

    return 0;
}
```

---

6. Testing & Results

The program was tested with various inputs such as:

- Few expenses
- Many expenses
- Budget higher than total
- Budget equal to total
- Budget lower than total

Across all tests, the program:

- Correctly read the number of expenses
- Calculated totals accurately
- Displayed clear output
- Correctly indicated when the budget was exceeded
- Produced no errors or crashes

The output format remained clean and readable.

---

7. Conclusion & Future Work

Conclusion

The Smart Daily Expense Tracker successfully performs its intended task of recording expenses and comparing them against a daily budget. It demonstrates modular programming concepts clearly and meets all requirements of the UPES C Programming Major Project Guidelines. The project is simple, efficient, and useful as a basic financial tool.

Future Work

This project can be expanded with features such as:

- Saving and loading expenses using files
- Weekly or monthly summaries
- Sorting expenses
- Category-wise totals
- A menu-driven interactive interface
- Exporting reports to files

---

8. References

- Course material and class notes
- Standard C library documentation
- UPES Major Project Guidelines
- GCC Compiler Manual