TRAVEL EXPENSE PLANNER AND BUDGET PLANNING SYSTEM

A Project Report Submitted in the partial fulfillment of the requirements of the course titled

"Problem Solving Through Programming (JAVA)"

BACHELOR OF TECHNOLOGY

In

DEPARTMENT OF FRESHMAN ENGINEERING

By

J.Geya Reddy	**2520090123**
Kamtam Priyanka	**2520090001**
S. Tejitha	**2520040008**
Tej Sahit	**2520030606**
AMPS Sreepad	**2520030152**
Karthik	**2520080006**

Under the Esteemed Guidance of

Guide Name
Dr. Jayaram Boga
(Assistant Professor)
Department of Freshman Engineering

Koneru Lakshmaiah Education Foundation
(Deemed to be University estd. u/s. 3 of the UGC Act, 1956)
Off-Campus: Bachupally-Gandimaisamma Road, Bowrampet, Hyderabad, Telangana - 500 043.
Phone No: 7815926816, www.klh.edu.in

K L (Deemed to be) University

DEPARTMENT OF FRESHMAN ENGINEERING

Declaration

The Project Report entitled **"TRAVEL EXPENSE PLANNER AND BUDGET PLANNING SYSTEM"** is a record of Bonafide work of **J. Geya Reddy 2520090123, Kamtam Priyanka 2520090001, S.Tejitha 2520040008, TejSahit 2520030606, AVSM Sreepad 2520030152, Karthik 2520080006** submitted in partial fulfillment of the requirements of the course titled "Problem Solving Through Programming (JAVA)" under the B.Tech Ist Year Trimester - I program in Department of Freshman Engineering at K L University. The results presented in this report have not been copied from any other department, university, or institute.

J.Geya Reddy-2520090123

Kamtam Priyanka -2520090001

S. Tejitha-2520040008

Tej Sahit-2520030606

AMPS Sreepad-2520030152

Karthik-2520080006

K L (Deemed to be) University

DEPARTMENT OF FRESHMAN ENGINEERING

CERTIFICATE

This is certify that the java project based report entitled **"TRAVEL EXPENSE PLANNER AND BUDGET PLANNING SYSTEM"** is a bonafide work done and submitted by **J.Geya Reddy 2520090123, Kamtam Priyanka 2520090001, S. Tejitha 2520040008, Tej Sahit 2520030606, AVSM Sreepad 2520030152, Karthik-2520080006** in partial fulfillment of the requirements of the course titled "Problem Solving Through Programming (JAVA)" under the B.Tech Ist Year Trimester - I program in Department of Freshman Engineering, K L (Deemed to be University), during the academic year **2025-2026.**

Signature of the Guide

Signature of the Course Coordinator **Signature of the HOD**

ACKNOWLEDGEMENT

The success in this project would not have been possible but for the timely help and guidance rendered by many people. Our wish to express my sincere thanks to all those who have assisted us in one way or the other for the completion of my project.

Our greatest appreciation to my Course Coordinator **Dr Y Ashok**, and my guide **Dr. Jayaram Boga,** Department of Freshman Engineering which cannot be expressed in words for his/her tremendous support, encouragement and guidance for this project.

We express our gratitude to **Dr. N. Chaitanya Kumar**, Head of the ***Department for Freshman Engineering*** for providing us with adequate facilities, ways and means by which we are able to complete this project-based Lab.

We thank all the members of teaching and non-teaching staff members, and also who have assisted me directly or indirectly for successful completion of this project. Finally, I sincerely thank my parents, friends and classmates for their kind help and cooperation during my work.

J.Geya Reddy-2520090123

Kamtam Priyanka -2520090001

S. Tejitha-2520040008

Tej Sahit-2520030606

AMPS Sreepad-2520030152

Karthik-2520080006

ABSTRACT

The primary aim of this project is to provide an intuitive tool that allows travelers to estimate trip costs, plan a budget, and continuously track their spending throughout the journey. Users can input their initial budget and update their expenditures in real time, enabling the system to display the remaining balance and guide them toward better financial decisions. The aim of our project is to assist users in forecasting trip expenses and maintaining control over their spending, and to implement a cost-tracking mechanism that automatically calculates and updates the remaining travel budget.

The advantages are that it improves financial discipline during travel, provides easy-to-use expense tracking with real-time updates, and requires minimal system resources due to its lightweight Java console design. The disadvantages are limited to single-user offline mode, no support for direct online information, bookings, or automated expense categorization, and console interface may be less user-friendly than graphical or mobile applications. Future Scope includes, supporting multi-user access over the internet, development of a graphical user interface (GUI) or mobile app for improved usability, secure authentication for user accounts and online transaction capabilities, and integration with travel services such as booking to expand into a full e-commerce travel platform.

Index

CHAPTER -1 INTRODUCTION

1.1Background of the Project

Traveling has become an integral part of modern life, whether for business, education, or leisure purposes. With travel comes the need to manage a variety of expenses such as transportation, accommodation, meals, and miscellaneous costs. Managing these expenses efficiently can often be challenging, as Travelers frequently struggle to keep track of where their money is going. Without proper management, it is easy to overspend or lose control of the budget, which can lead to financial stress and inconvenience. A Travel Expense Calculator addresses this problem by providing a simple, automated way to plan, monitor, and manage travel-related expenses. By using such a system, travelers can gain a comprehensive view of their financial commitments during a trip, make informed decisions, and optimize their spending to avoid financial surprises.

The primary purpose of a Travel Expense Calculator is to automate and streamline the process of expense management. Traditionally, Travelers relied on manual methods, such as notebooks, journals, or spreadsheets, to record and calculate expenses. These methods are prone to human error, time-consuming, and often inconvenient, particularly during multi-day or international trips. With modern technology, it is now possible to design solutions that are faster, more accurate, and user-friendly. A well-designed calculator enables users to estimate the costs of transportation, accommodation, food, and other essentials before their journey begins. During travel, the system allows Travelers to record their actual expenses and compare them against the planned budget. This feature not only helps in staying within budget but also provides valuable insights into spending habits for future trips.

One of the most important aspects of a Travel Expense Calculator is its ability to track expenses in real-time. For example, when a business traveler books a flight, checks into a hotel, or dines at a restaurant, the calculator can record these expenditures immediately, categorize them, and update the total budget automatically. This real-time

tracking ensures that users are constantly aware of their financial standing, preventing overspending and helping them make adjustments on the go. Additionally, advanced calculators can include features such as currency conversion for international travel, graphical representations of spending patterns, and the ability to export detailed reports for personal use or reimbursement purposes. Such features enhance the utility of the system and make it an indispensable tool for frequent Travelers.

The scope of the Travel Expense Calculator is broad, covering nearly every aspect of travel-related finances. Transportation expenses can include airfares, train and bus fares, fuel costs, taxi charges, and rideshare services. Accommodation expenses may consist of hotel bookings, service charges, or rental stays, while food and beverage expenses cover meals, snacks, and drinks during the trip. Miscellaneous expenses, often overlooked, can include entry tickets, souvenirs, emergency expenditures, and local transportation costs. By encompassing all these aspects, a Travel Expense Calculator provides a comprehensive solution that simplifies the travel experience. Users can plan budgets in advance, track spending in real-time, and analyze their expenses after the trip to identify areas of improvement. This structured approach ensures that travelers remain financially disciplined while enjoying their journey.

In today's fast-paced world, the importance of managing travel expenses efficiently cannot be overstated. As the cost of travel continues to rise, travelers are increasingly seeking tools that allow them to stay within budget without compromising on the quality of their experience. Modern Travelers are tech-savvy and rely on digital solutions to make informed decisions. The Travel Expense Calculator is an excellent example of such a solution, combining convenience, accuracy, and practicality. By leveraging technology, the calculator replaces outdated manual methods, providing transparency and accountability while reducing the stress associated with financial planning. For businesses, the calculator is equally valuable, enabling accurate tracking of employee expenses, simplifying reimbursement processes, and supporting auditing requirements.

The objectives of a Travel Expense Calculator are clear and focused. Firstly, it aims to provide accurate and real-time expense calculations, minimizing human error and saving time. Secondly, it simplifies the process of budgeting and expense tracking, allowing users to plan ahead and monitor costs effectively. Thirdly, the system provides detailed reporting, enabling Travelers and organizations to analyze spending patterns, identify unnecessary costs, and optimize future budgets. Fourthly, it offers flexibility to accommodate different types of travel, including domestic, international, short-term, and long-term trips. Finally, the system seeks to enhance financial awareness, empowering users to make informed decisions about their spending habits and improving overall financial discipline. By fulfilling these objectives, a Travel Expense Calculator serves as a practical and comprehensive tool for both individual Travelers and organizations.

In conclusion, the Travel Expense Calculator is an essential tool in modern travel management. It bridges the gap between financial planning and actual spending, providing a seamless, efficient, and user-friendly solution to a common problem faced by Travelers worldwide. By automating expense tracking, offering real-time insights, and generating detailed reports, the system empowers users to manage their travel budgets effectively. With features like expense categorization, currency conversion, and reporting capabilities, it not only simplifies travel but also promotes better financial decision-making. As travel becomes more frequent and diverse, tools like the Travel Expense Calculator will play a vital role in ensuring that journeys are enjoyable, stress-free, and financially well-managed.

The Features and Functionalities of a Travel Expense Calculator. This is more than just a simple arithmetic tool; it is a comprehensive system that helps Travelers plan, monitor, and evaluate expenses throughout their journey. Key features of an effective calculator include expense categorization, real-time tracking, automated calculations, and reporting tools. Expense categorization ensures that costs are neatly organized into groups such as transportation, accommodation, food, and miscellaneous items. This makes it easier for users to identify where most of their money is being spent and take corrective measures if necessary.

Real-time tracking allows Travelers to enter expenses as they occur, avoiding the hassle of recalling costs later. Automated calculations ensure that totals, subtotals, and remaining budget estimates are instantly updated whenever new data is entered. Some advanced calculators even integrate with online payment systems or banking apps to automatically import transactions, making the process seamless and highly efficient. Reporting tools are also vital, as they provide visual charts and summaries that give a clear overview of spending patterns. Reports can be exported in formats such as PDF or Excel, which is particularly useful for corporate travel or reimbursement purposes.

The Benefits to Individual Travelers, For individual Travelers, the Travel Expense Calculator offers several tangible benefits. First, it prevents overspending by providing a clear picture of available budget versus actual expenses. For instance, a backpacker on a limited budget can see how much money is left for meals or sightseeing and adjust plans accordingly. Second, it helps in planning future trips. By analyzing past travel data, users can identify trends in their spending habits and make more informed decisions for upcoming journeys. Third, it saves time and effort. Instead of manually tracking each expense and performing calculations, users can simply input data into the system and let the calculator do the work.

Another benefit is enhanced financial awareness. Many Travelers are unaware of how small, seemingly insignificant expenses add up during a trip. By consistently tracking costs, users develop a habit of mindful spending, which can lead to long-term financial discipline. Additionally, for international Travelers, the calculator's ability to handle currency conversion and fluctuations in exchange rates ensures accurate budgeting across different countries.

The Benefits to Businesses and Organizations, Travel Expense Calculators are equally valuable for organizations and businesses that require employees to travel for work. Managing employee expenses manually can be cumbersome and prone to errors, especially when dealing with multiple trips, currencies, and expense categories. By implementing a Travel Expense Calculator, businesses can streamline the reimbursement process, ensuring employees are reimbursed accurately and promptly.

The system also provides audit-ready reports, which help organizations maintain transparency and accountability. Detailed records of travel expenses can be reviewed by finance teams, minimizing errors and reducing the risk of fraudulent claims. Additionally, businesses can use the data to analyze spending patterns across departments, identify cost-saving opportunities, and optimize travel policies. For example, a company might notice that taxi expenses are consistently high in certain locations and decide to negotiate corporate rates with local transportation providers.

The Technological Considerations, Developing an effective Travel Expense Calculator requires careful consideration of technology and usability. The system must be user-friendly, allowing Travelers of all age groups and technical proficiency levels to navigate it with ease. Cross-platform compatibility is also important, as users may want to access the calculator via desktop computers, laptops, or mobile devices.

Integration with other digital tools, such as online banking, email receipts, and travel booking platforms, enhances the functionality of the calculator. For example, some systems can automatically import flight or hotel bookings, converting them into expense entries without manual input. Security is another critical factor, particularly when handling sensitive financial data. The system should ensure data encryption, secure login, and proper privacy protocols to protect user information.

The Future Scope and Enhancements, The future of Travel Expense Calculators is promising, with possibilities for further automation, artificial intelligence, and data analytics integration. For instance, AI could analyze past spending patterns and suggest optimal budgets for future trips. It could also provide personalized recommendations on cost-effective accommodations, dining options, or transportation methods based on historical data.

Integration with mobile applications can make the calculator even more convenient. Features such as push notifications for budget alerts, offline data entry for areas with limited internet access, and cloud synchronization for accessing data across multiple devices can significantly enhance user experience.

1.2 Problem Statement

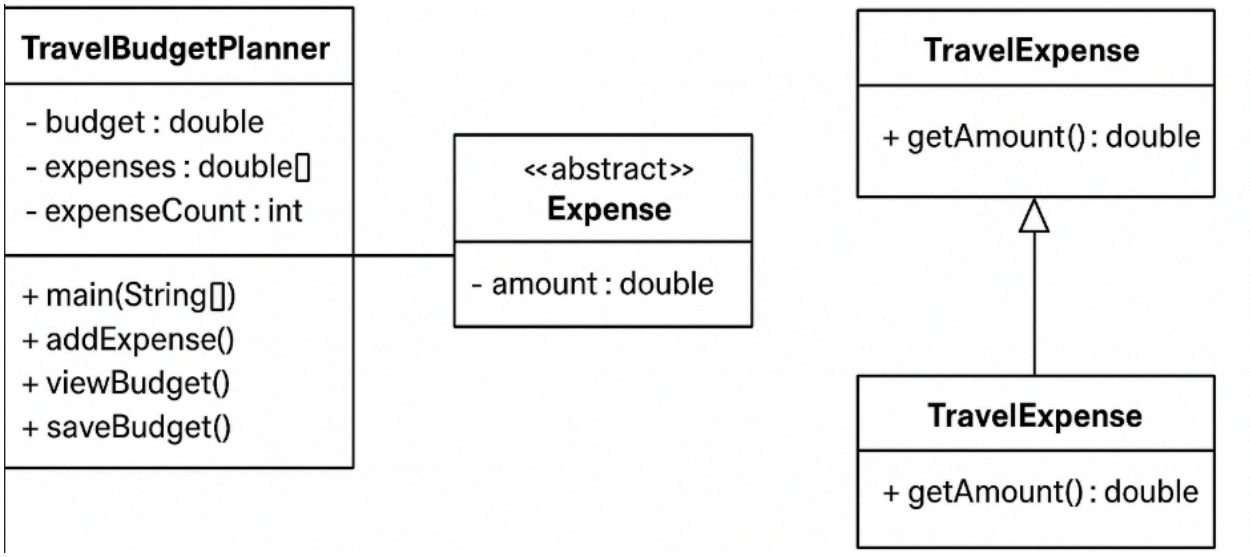
Traveling involves multiple expenses such as transportation, accommodation, food, and miscellaneous costs. Many individuals and families face difficulties in managing these expenses effectively due to the lack of a simple and structured budgeting system. Expenses are often recorded manually or remembered mentally, which can lead to budget overruns, poor financial planning, and stress during travel.

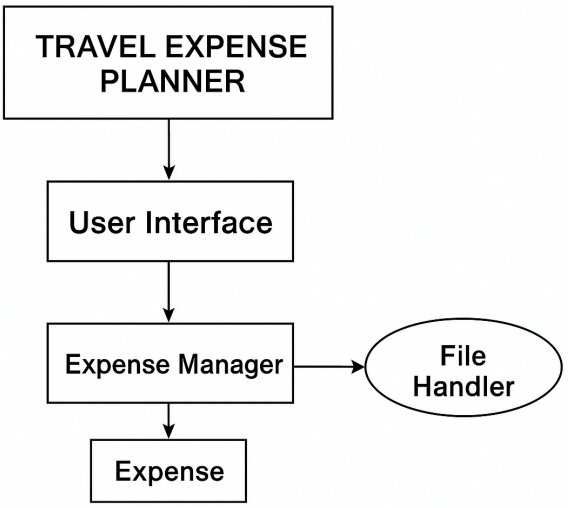
Existing expense management applications are often complex, require constant internet access, or involve advanced interfaces that may not be convenient for all users. There is a need for a lightweight, easy-to-use, and offline solution that helps users plan their travel budget, track expenses, and monitor the remaining balance in real time.

The problem is to design and implement a **Java console-based Travel Budget Planner** that allows users to set an initial travel budget, record expenses as they occur, calculate total expenditure, and display the remaining budget. The system should also provide data persistence through file handling so that budget information can be saved and retrieved. The application must be user-friendly, efficient, and reliable, while demonstrating core programming and object-oriented concepts.

CHAPTER -2 SYSTEM ARCHITECTURE

2.1 High-level architecture diagram





The class diagram represents the static structure of the *Travel Budget Planner* system and illustrates the relationship between the various classes involved in the application. It shows how data members and member functions are organized and how object-oriented principles are applied to achieve modularity, reusability, and maintainability.

The Expense class is an abstract base class that defines the common structure for all types

of expenses in the system. It contains a protected data member amount and an abstract method getAmount(). Since this class is abstract, it cannot be instantiated directly. It provides a template that enforces uniform behavior for all expense-related subclasses, thereby demonstrating abstraction.

The TravelExpense class is a concrete subclass of the Expense class and represents a specific type of expense incurred during travel. It inherits the amount attribute from the Expense class and provides an implementation of the abstract method getAmount(). This class also contains additional data such as the date on which the expense was recorded. The relationship between Expense and TravelExpense is an inheritance relationship, demonstrating polymorphism through method overriding.

The TravelBudgetPlanner class acts as the main controller class of the system. It contains the main() method, which serves as the entry point of the application. This class manages user interaction, stores expense data, performs calculations, and handles file operations. It maintains collections such as arrays and lists to store expense information efficiently. The **class also** includes multiple methods such as addExpense(), analyzeExpenses(), saveBudget(), and loadBudget() to ensure proper separation of responsibilities.

The relationship between TravelBudgetPlanner and the Expense class is a dependency and aggregation relationship, as the planner class creates and manages multiple Expense objects without owning their lifecycle permanently. This design improves flexibility and allows future expansion to include different types of expenses.

Overall, the class diagram clearly demonstrates the use of object-oriented programming principles, including abstraction, inheritance, encapsulation, and polymorphism. The structured interaction between classes ensures scalability, ease of maintenance, and better code organization, making the system suitable for real-world applications

CHAPTER -3 CO's ATTAINMENT

3.1 CO1 Attainment

CO1 Syllabus	CO1 Concepts Included in Project
Apply fundamental programming constructs such as data types, operators, conditional and iterative statements in Java to develop logic-based solutions for basic computational problems.	Primitive data types (int, double, String), Arithmetic and relational operators, Conditional statements (if, if-else, switch), Iterative statements (for loop, while loop)

3.1.1 Scenarios for CO1 implementation.

In the Travel Expense Planner system, CO1 is attained through basic logic implementation. The application takes user input such as initial travel budget, expense amount, and expense category. Arithmetic operators are used to calculate the total expenses and remaining balance. Conditional statements verify whether expenses exceed the budget and display suitable warning messages. Iterative statements allow the user to continuously add multiple expenses during a single trip session. These constructs collectively form the foundation of the application logic.

3.1.2 CO1 code screen shot.

```java
// CO1: Iteration
do {
    displayMenu();
    choice = sc.nextInt();

    switch (choice) {

        case 1:
            addExpense();
            break;

        case 2:
            viewRemainingBudget();
            break;

        case 3:
            analyzeExpenses();
            break;

        case 4:
            displayExpenseList();
            break;

        case 5:
            saveBudget(budget);
            break;

        case 6:
            loadBudget();
            break;

        case 7:
            System.out.println("Exiting application.");
            break;

        default:
```

3.2 CO2 Attainment

CO2 Syllabus	**CO2 Concepts Included in Project**
Design, trace, and optimize algorithms using one-dimensional and two-dimensional arrays to solve mathematical, quantitative, and real-world problems efficiently through search, sort, and matrix manipulation techniques.	One-dimensional arrays, Array traversal, Storage of expense values, Index-based access and update**.**

3.2.1 Scenarios for CO2 implementation.

The project uses one-dimensional arrays to store expense amounts, expense categories, and dates. These arrays allow systematic storage and retrieval of multiple expense records. Iteration over arrays is used to calculate total expenditure and generate summaries. The use of arrays helps in organizing data efficiently and enables easy analysis of expenses across different categories.

3.2.2 CO2 code screen shot.

```
// CO2: One-dimensional array
static double[] expenses = new double[100];
static int expenseCount = 0;
```

3.3 CO3 Attainment

CO3 Syllabus	CO3 Concepts Included in Project
Construct and evaluate advanced problem-solving logic using strings, recursion, and bitwise operations for solving complex mathematical, pattern-based, and combinatorial problems relevant to competitive coding platforms.	String handling, Input validation, String comparison and formatting.

3.3.1 Scenarios for CO3 implementation.

String manipulation is used for handling expense categories, dates, and user commands. String comparison methods help identify expense types such as food, transport, or accommodation. Input validation using strings ensures correct user interaction and prevents invalid entries. Although recursion and bitwise operations are not central to this application, structured string operations help in user-friendly interaction and display of reports.

3.3.2 CO3 code screen shot.

```
// CO3: Recursive method
static double totalExpense(int n) {
    if (n == 0) {
        return 0;
    }
    return expenses[n - 1] + totalExpense(n - 1);
}

static void analyzeExpenses() {  1 usage
    // CO3: Recursive computation
    double total = totalExpense(expenseCount);
    System.out.println("Total Expenses: " + total);
}
```

3.4 CO4 Attainment

CO4 Syllabus	CO4 Concepts Included in Project
Develop structured and modular programs by applying object-oriented	Classes and objects, Methods and constructors, Encapsulation using access

programming principles such as encapsulation, abstraction, and modularization using Java classes, methods, and constructors.	specifiers, Modular program structure.

3.4.1 Scenarios for CO4 implementation.

The project is designed using multiple classes such as Expense, TravelExpense, and TravelBudgetPlanner. Data members are encapsulated using access specifiers, and operations are performed through methods. Constructors initialize objects with required values. This modular structure improves readability, reusability, and maintainability of the code.

3.4.2 CO4 code screen shot.

```
// CO4 & CO5: Abstract class (Abstraction, Inheritance)
abstract class Expense {
    protected double amount;

    Expense(double amount) {
        this.amount = amount;
    }

    abstract double getAmount();   // Polymorphism
}
```

```java
// ======================== BUDGET METHODS ========================

static double getInitialBudget() {  2 usages
    System.out.print("Enter initial travel budget: ");
    double value = sc.nextDouble();

    if (value <= 0) {
        System.out.println("Budget must be positive.");
        return getInitialBudget();
    }
    return value;
}

static void viewRemainingBudget() {  1 usage
    System.out.println("Remaining Budget: " + budget);
}

// ======================== EXPENSE METHODS ========================

static void addExpense() {  1 usage

    System.out.print("Enter expense amount: ");
    double amt = sc.nextDouble();

    if (amt <= 0) {
        System.out.println("Expense must be positive.");
        return;
    }

    if (amt > budget) {
        System.out.println("Expense exceeds available budget.");
        return;
    }
```

```
// ====================== DISPLAY METHODS ======================

static void displayWelcome() {  1 usage
    System.out.println("====================================");
    System.out.println("   TRAVEL BUDGET PLANNER SYSTEM");
    System.out.println("====================================");
}

static void displayMenu() {  1 usage
    System.out.println("\n---------- MENU ----------");
    System.out.println("1. Add Expense");
    System.out.println("2. View Remaining Budget");
    System.out.println("3. Analyze Total Expense");
    System.out.println("4. Display All Expenses");
    System.out.println("5. Save Budget to File");
    System.out.println("6. Load Budget from File");
    System.out.println("7. Exit");
    System.out.print("Enter your choice: ");
}
```

3.5 CO5 Attainment

CO5 Syllabus	CO5 Concepts Included in Project
Design extensible and reusable Java programs employing inheritance, polymorphism, abstract classes, interfaces, and reflection API to solve domain-oriented problems with clarity and maintainability.	Abstract class, Inheritance, Method overriding, Polymorphism

3.5.1 Scenarios for CO5 implementation.

The Expense class is implemented as an abstract class that defines a common structure for all expense types. The TravelExpense class extends this abstract class and provides its own implementation of abstract methods. This demonstrates inheritance and polymorphism, allowing the system to be easily extended by adding new expense types in the future without modifying existing code.

3.5.2 CO5 code screen shot.

```
// CO4 & CO5: Abstract class (Abstraction, Inheritance)
abstract class Expense {
    protected double amount;

    Expense(double amount) {
        this.amount = amount;
    }

    abstract double getAmount();   // Polymorphism
}
```

```
// CO5: Inheritance + Polymorphism
class TravelExpense extends Expense {
    TravelExpense(double amount) {
        super(amount);
    }
```

3.6 CO6 Attainment

CO6 Syllabus	CO6 Concepts Included in Project
Implement robust, scalable, and generic Java applications integrating exception handling, file I/O, generics, and collections framework, along with functional programming constructs to handle real-world data-driven tasks.	Exception handling using try-catch, File input and output operations, Collections such as ArrayList, Data persistence

3.6.1 Scenarios for CO6 implementation.

Exception handling is used to manage invalid inputs and runtime errors during execution. File I/O operations help store and retrieve budget and expense data, ensuring persistence between program runs. The ArrayList collection is used for dynamic storage of expenses, allowing scalability and efficient data handling. These features collectively make the application robust and suitable for real-world use.

3.6.2 CO6 code screen shot.

```
// CO6: Collection (ArrayList with Generics)
static List<Expense> expenseList = new ArrayList<>();
```

```java
// CO6: File I/O + Exception Handling

static void saveBudget(double budget) {  1 usage
    try (FileWriter fw = new FileWriter( fileName: "budget.txt")) {

        fw.write( str: budget + "\n");
        fw.write( str: "Total Expenses: " + totalExpense(expenseCount));
        fw.write( str: "\nExpense Count: " + expenseCount);

        System.out.println("Budget saved to file.");

    } catch (IOException e) {
        System.out.println("Error saving file.");
    }
}

static void loadBudget() {  1 usage

    try (Scanner fileScanner = new Scanner(new File( pathname: "budget.txt")))

        if (fileScanner.hasNextDouble()) {
            budget = fileScanner.nextDouble();
            System.out.println("Budget loaded successfully.");
            System.out.println("Current Budget: " + budget);
        }

    } catch (FileNotFoundException e) {
        System.out.println("No saved budget file found.");
    }
}
```

CHAPTER-4 SCREEN SHOTS

4.1 Screen Shots

```java
import java.util.*;        // CO6: Collections, Generics
import java.io.*;          // CO6: File I/O

// ======================= EXPENSE ABSTRACTION =======================
// CO4 & CO5: Abstract class (Abstraction + Inheritance)
abstract class Expense {  4 usages  1 inheritor

    protected double amount;  2 usages

    // Constructor
    Expense(double amount) {  2 usages
        this.amount = amount;
    }

    // Abstract method (Polymorphism)
    abstract double getAmount();  1 usage  1 implementation
}

// ======================= TRAVEL EXPENSE ============================
// CO5: Inheritance + Polymorphism
class TravelExpense extends Expense {  1 usage

    private Date dateRecorded;  2 usages

    TravelExpense(double amount) {  1 usage
        super(amount);
        this.dateRecorded = new Date();
    }

    @Override  1 usage
    double getAmount() {
        return amount;
    }

```

```
35      public Date getDateRecorded() {  no usages
36          return dateRecorded;
37      }
38  }
39
40  // ====================== MAIN PLANNER CLASS ======================
41  public class TravelBudgetPlanner {
42
43      // ====================== GLOBAL VARIABLES ======================
44      // CO1: Fundamental programming constructs
45      static Scanner sc = new Scanner(System.in);  4 usages
46
47      // CO2: One-dimensional array
48      static double[] expenses = new double[200];  2 usages
49      static int expenseCount = 0;  4 usages
50
51      // CO6: Collections with Generics
52      static List<Expense> expenseList = new ArrayList<>();  3 usages
53
54      static double budget = 0.0;  7 usages
55
56      // ====================== MAIN METHOD ============================
57      public static void main(String[] args) {
58
59          displayWelcome();
60
61          budget = getInitialBudget();
62
63          int choice;
64
65          // CO1: Iteration
66          do {
67              displayMenu();
68              choice = sc.nextInt();
```

```
70          switch (choice) {
71
72              case 1:
73                  addExpense();
74                  break;
75
76              case 2:
77                  viewRemainingBudget();
78                  break;
79
80              case 3:
81                  analyzeExpenses();
82                  break;
83
84              case 4:
85                  displayExpenseList();
86                  break;
87
88              case 5:
89                  saveBudget(budget);
90                  break;
91
92              case 6:
93                  loadBudget();
94                  break;
95
96              case 7:
97                  System.out.println("Exiting application.");
98                  break;
99
100             default:
101                 System.out.println("Invalid choice. Try again.");
102         }
103
```

```
104         } while (choice != 7);
105
106         sc.close();
107     }
108
109     // ====================== DISPLAY METHODS ======================
110
111     static void displayWelcome() {  1 usage
112         System.out.println("====================================");
113         System.out.println("   TRAVEL BUDGET PLANNER SYSTEM");
114         System.out.println("====================================");
115     }
116
117     static void displayMenu() {  1 usage
118         System.out.println("\n---------- MENU ----------");
119         System.out.println("1. Add Expense");
120         System.out.println("2. View Remaining Budget");
121         System.out.println("3. Analyze Total Expense");
122         System.out.println("4. Display All Expenses");
123         System.out.println("5. Save Budget to File");
124         System.out.println("6. Load Budget from File");
125         System.out.println("7. Exit");
126         System.out.print("Enter your choice: ");
127     }
128
129     // ====================== BUDGET METHODS ======================
130
131     static double getInitialBudget() {  2 usages
132         System.out.print("Enter initial travel budget: ");
133         double value = sc.nextDouble();
134
135         if (value <= 0) {
136             System.out.println("Budget must be positive.");
```

```
137             return getInitialBudget();
138         }
139         return value;
140     }
141
142     static void viewRemainingBudget() {  1 usage
143         System.out.println("Remaining Budget: " + budget);
144     }
145
146     // ====================== EXPENSE METHODS ======================
147
148     static void addExpense() {  1 usage
149
150         System.out.print("Enter expense amount: ");
151         double amt = sc.nextDouble();
152
153         if (amt <= 0) {
154             System.out.println("Expense must be positive.");
155             return;
156         }
157
158         if (amt > budget) {
159             System.out.println("Expense exceeds available budget.");
160             return;
161         }
162
163         // CO2: Array usage
164         expenses[expenseCount++] = amt;
165
166         // CO5: Polymorphic object
167         expenseList.add(new TravelExpense(amt));
168
169         budget -= amt;
```

```
171         System.out.println("Expense added successfully.");
172     }
173
174     static void analyzeExpenses() {  1 usage
175         // CO3: Recursive computation
176         double total = totalExpense(expenseCount);
177         System.out.println("Total Expenses: " + total);
178     }
179
180     static void displayExpenseList() {  1 usage
181
182         if (expenseList.isEmpty()) {
183             System.out.println("No expenses recorded.");
184             return;
185         }
186
187         System.out.println("\n--- Expense History ---");
188
189         int index = 1;
190
191         // CO6: Enhanced for-loop with collection
192         for (Expense e : expenseList) {
193             System.out.println(index + ". Amount: " + e.getAmount());
194             index++;
195         }
196     }
197
198     // ====================== RECURSION ============================
199     // CO3: Recursive function
200     static double totalExpense(int n) {  3 usages
201         if (n == 0) {
202             return 0;
203         }
```

```
204         return expenses[n - 1] + totalExpense( n: n - 1);
205     }
206     // ====================== FILE HANDLING ========================
207     // CO6: File I/O + Exception Handling
208
209     static void saveBudget(double budget) {  1 usage
210         try (FileWriter fw = new FileWriter( fileName: "budget.txt")) {
211
212             fw.write( str: budget + "\n");
213             fw.write( str: "Total Expenses: " + totalExpense(expenseCount));
214             fw.write( str: "\nExpense Count: " + expenseCount);
215
216             System.out.println("Budget saved to file.");
217
218         } catch (IOException e) {
219             System.out.println("Error saving file.");
220         }
221     }
222
223     static void loadBudget() {  1 usage
224
225         try (Scanner fileScanner = new Scanner(new File( pathname: "budget.txt"))) {
226
227             if (fileScanner.hasNextDouble()) {
228                 budget = fileScanner.nextDouble();
229                 System.out.println("Budget loaded successfully.");
230                 System.out.println("Current Budget: " + budget);
231             }
232
233         } catch (FileNotFoundException e) {
234             System.out.println("No saved budget file found.");
235         }
236     }
237 }
```

Output:

```
====================================
   TRAVEL BUDGET PLANNER SYSTEM
====================================
Enter initial travel budget: 50000

----------- MENU -----------
1. Add Expense
2. View Remaining Budget
3. Analyze Total Expense
4. Display All Expenses
5. Save Budget to File
6. Load Budget from File
7. Exit
Enter your choice: 1
Enter expense amount: 7000
Expense added successfully.

----------- MENU -----------
1. Add Expense
2. View Remaining Budget
3. Analyze Total Expense
4. Display All Expenses
5. Save Budget to File
6. Load Budget from File
7. Exit
Enter your choice: 1
Enter expense amount: 4000
Expense added successfully.
```

```
---------- MENU ----------
1. Add Expense
2. View Remaining Budget
3. Analyze Total Expense
4. Display All Expenses
5. Save Budget to File
6. Load Budget from File
7. Exit
Enter your choice: 1
Enter expense amount: 10000
Expense added successfully.

---------- MENU ----------
1. Add Expense
2. View Remaining Budget
3. Analyze Total Expense
4. Display All Expenses
5. Save Budget to File
6. Load Budget from File
7. Exit
Enter your choice: 3
Total Expenses: 21000.0
```

```
---------- MENU ----------
1. Add Expense
2. View Remaining Budget
3. Analyze Total Expense
4. Display All Expenses
5. Save Budget to File
6. Load Budget from File
7. Exit
Enter your choice: 4

--- Expense History ---
1. Amount: 7000.0
2. Amount: 4000.0
3. Amount: 10000.0
```

```
---------- MENU ----------
1. Add Expense
2. View Remaining Budget
3. Analyze Total Expense
4. Display All Expenses
5. Save Budget to File
6. Load Budget from File
7. Exit
Enter your choice: 5
Budget saved to file.
```

```
29000.0
Total Expenses: 21000.0
Expense Count: 4
```

CHAPTER -5 TESTING

5.1 Test Cases and Results

Test case 1:

```
====================================
   TRAVEL BUDGET PLANNER SYSTEM
====================================
Enter initial travel budget: 50000

---------- MENU ----------
1. Add Expense
2. View Remaining Budget
3. Analyze Total Expense
4. Display All Expenses
5. Save Budget to File
6. Load Budget from File
7. Exit
Enter your choice: 1
Enter expense amount: 3000
Expense added successfully.
```

Test case 2:

```
---------- MENU ----------
1. Add Expense
2. View Remaining Budget
3. Analyze Total Expense
4. Display All Expenses
5. Save Budget to File
6. Load Budget from File
7. Exit
Enter your choice: 2
Remaining Budget: 47000.0
```

Test case 3:

```
---------- MENU ----------
1. Add Expense
2. View Remaining Budget
3. Analyze Total Expense
4. Display All Expenses
5. Save Budget to File
6. Load Budget from File
7. Exit
Enter your choice: 1
Enter expense amount: 500
Expense added successfully.
```

Test case 4:

```
---------- MENU ----------
1. Add Expense
2. View Remaining Budget
3. Analyze Total Expense
4. Display All Expenses
5. Save Budget to File
6. Load Budget from File
7. Exit
Enter your choice: 3
Total Expenses: 3500.0
```

Test Case 5:

```
---------- MENU ----------
1. Add Expense
2. View Remaining Budget
3. Analyze Total Expense
4. Display All Expenses
5. Save Budget to File
6. Load Budget from File
7. Exit
Enter your choice: 4

--- Expense History ---
1. Amount: 3000.0
2. Amount: 500.0
```

Test case 6:

```
----------- MENU -----------
1. Add Expense
2. View Remaining Budget
3. Analyze Total Expense
4. Display All Expenses
5. Save Budget to File
6. Load Budget from File
7. Exit
Enter your choice: 5
Budget saved to file.
```

Test case 7:

```
----------- MENU -----------
1. Add Expense
2. View Remaining Budget
3. Analyze Total Expense
4. Display All Expenses
5. Save Budget to File
6. Load Budget from File
7. Exit
Enter your choice: 6
Budget loaded successfully.
Current Budget: 46500.0
```

Test case 8:

```
---------- MENU ----------
1. Add Expense
2. View Remaining Budget
3. Analyze Total Expense
4. Display All Expenses
5. Save Budget to File
6. Load Budget from File
7. Exit
Enter your choice: 7
Exiting application.
```

Test case 9:

```
Enter initial travel budget: 50000

---------- MENU ----------
1. Add Expense
2. View Remaining Budget
3. Analyze Total Expense
4. Display All Expenses
5. Save Budget to File
6. Load Budget from File
7. Exit
Enter your choice: 1
Enter expense amount: 60000
Expense exceeds available budget.
```

Test case 10:

```
---------- MENU ----------
1. Add Expense
2. View Remaining Budget
3. Analyze Total Expense
4. Display All Expenses
5. Save Budget to File
6. Load Budget from File
7. Exit
Enter your choice: 8
Invalid choice. Try again.
```

CHAPTER -6 FUTURE ENHANCEMENTS

6.1 Planned Features

The Travel Expense Calculator has strong potential for expansion, and several future enhancements can significantly improve its usability, accuracy, and overall efficiency. One of the major enhancements is the integration of advanced currency conversion capabilities, allowing the system to automatically fetch and update live exchange rates. This is especially beneficial for travelers who need accurate and up-to-date cost estimations in different currencies. Additionally, implementing receipt image upload with OCR will enable users to simply scan bills instead of entering details manually. This feature not only saves time but also ensures greater accuracy by extracting dates, amounts, and vendor names directly from the receipt.

Another important enhancement is the introduction of smart expense categorization, where the system intelligently assigns categories based on patterns, merchant names, or previous user behavior. This could be achieved through rule-based logic initially, and later upgraded with machine learning techniques to improve accuracy over time. To further support frequent travelers and corporate users, a comprehensive reporting and analytics dashboard can be included. This dashboard would display charts, category-wise spending, budget trends, and financial summaries, helping users understand their spending habits and make better decisions for future trips. Exporting reports in multiple formats such as PDF, Excel, or CSV will also make the tool more versatile and suitable for professional use.

Future versions of the system can also enhance convenience by offering multi-trip management and user profiles, allowing individuals to switch between personal and business trips easily. For companies, the system can be expanded to support a reimbursement and approval workflow, where employees submit expense reports and managers approve them digitally. This not only speeds up administrative processes but also provides transparency and reduces errors or fraudulent claims. Along with this, integrating the calculator with bank statements or travel booking platforms would allow automatic import of transactions and travel itineraries, reducing manual input even further.

From a technological perspective, the calculator can evolve into a mobile-friendly

application or a full-fledged mobile app with offline support. Travelers often face poor network connectivity, and offline data entry with automatic synchronization once the internet is available will significantly improve usability. Push notifications can also be introduced to alert users when they are nearing their budget limits or when new receipts are detected. Additionally, future versions could include AI-powered budget predictions, where the system analyzes past trips to recommend ideal budgets for upcoming journeys.

Security and reliability remain important aspects of future development. Implementing stronger authentication systems, data encryption, privacy controls, and audit logs will ensure that users' financial information remains secure. As the application grows, integrating cloud storage, caching mechanisms, and performance monitoring tools will help maintain smooth operation even with a large number of users. Overall, these enhancements will transform the Travel Expense Calculator from a simple budgeting tool into a smart, automated, and highly efficient travel management system capable of serving both individual travelers and large organizations.

CHAPTER -7 CONCLUSION

7.1 Summary of the Project

The *Travel Expense Planner and Budget Planning System* successfully demonstrates how technology can simplify and optimize financial management for travelers. By integrating expense tracking, budget forecasting, and real-time planning tools, the system empowers users to make informed decisions, avoid overspending, and enjoy stress-free travel experiences.

The key takeaways from the *Travel Expense Planner and Budget Planning System* highlight its overall effectiveness and impact. The system enhances efficiency by automating expense recording, thereby reducing manual effort and saving valuable time. It ensures accuracy through precise budget calculations and real-time updates, allowing travelers to make informed financial decisions during their trips. With its user-centric design, the platform provides intuitive interfaces that simplify planning and monitoring, making it accessible and easy to use. Most importantly, the system delivers a practical impact by helping travelers maintain financial discipline while still maximizing the enjoyment of their journeys, striking a balance between responsible budgeting and memorable travel experiences.

Ultimately, this project highlights the importance of combining financial management principles with smart technology solutions. It not only addresses common challenges faced during travel but also sets the foundation for future enhancements such as AI-driven recommendations, multi-currency support, and integration with booking platforms.

CHAPTER -8 REFERENCES

Books

Pressman, R. S., & Maxim, B. R. (2019). *Software Engineering: A Practitioner's Approach* (8th ed.). McGraw- Hill Education.

Freeman, E., & Freeman, E. (2020). *Head First HTML and CSS* (2nd ed.).

O'Reilly Media. Duckett, J. (2014). *HTML & CSS: Design and Build Websites*.

Duckett, J. (2015). *JavaScript and JQuery: Interactive Front-End Web*

Development. Wiley. Sommerville, I. (2016). *Software Engineering* (10th ed.).

Tutorials and Learning Resources

W3Schools. (2025). *HTML, CSS, and JavaScript Tutorials.* Retrieved from https://www.w3schools.com

MDN Web Docs. (2025). *Web Technologies Documentation.* Retrieved from https://developer.mozilla.org

FreeCodeCamp. (2025). *Responsive Web Design Certification.* Retrieved from https://www.freecodecamp.org

GeeksforGeeks. (2025). *Front-end Development and Web Technologies.* Retrieved from https://www.geeksforgeeks.org

Tutorialspoint. (2025). *HTML, CSS, JavaScript, and Web Development Tutorials.* Retrieved from https://www.tutorialspoint.com

APIs and Tools Used

Google Maps Platform. (2025). *Maps JavaScript API Documentation.* Retrieved from https://developers.google.com/maps

Unsplash API. (2025). *Free High-Resolution Image Access.* Retrieved from https://unsplash.com/developers

OpenWeatherMap. (2025). *Weather API for Travel Assistance.* Retrieved from https://openweathermap.org/api

RapidAPI Hub. (2025). *Travel and Tourism APIs.* Retrieved from https://rapidapi.com/hub

Firebase by Google. (2025). *Authentication and Database Services.* Retrieved from https://firebase.google.com

Documentation and Research References

World Tourism Organization (UNWTO). (2024). *Tourism Data Dashboard.* Retrieved from https://www.unwto.org

Booking.com. (2025). *Accommodation and Destination Insights.* Retrieved from https://www.booking.com

TripAdvisor. (2025). *Tourism and Destination Review Data.* Retrieved from https://www.tripadvisor.com

Travel + Leisure. (2025). *Top Global Destinations and Travel Trends.* Retrieved from https://www.travelandleisure.com

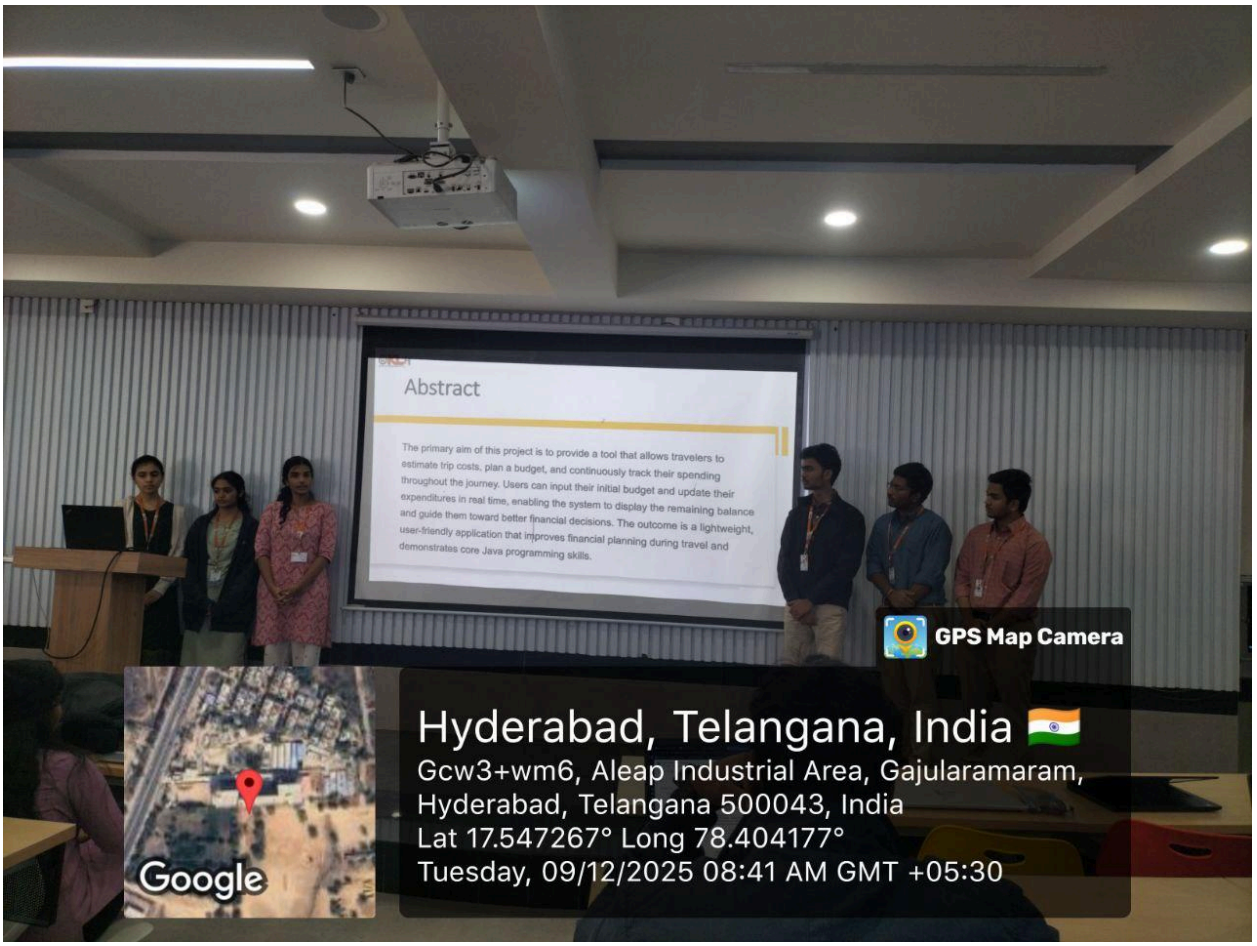
Tourism Review. (2025). *Digital Transformation in the Tourism Industry.* Retrieved from https://www.tourism- review.com

CHAPTER -9 APPENDICES

OCR - Optical Character Recognition
PDF- Portable Document Format
CSV- Comma Separated Values
AI- Artificial Intelligence
API- Application Programming Interface
UNWTO- United Nations World Tourism Organization

GitHub: https://github.com/geyaareddyj/Java-Project1
Youtube: https://youtu.be/HK0ja_0Agcs

Geo Tag photos with guide


Abstract
The primary aim of this project is to provide a tool that allows travelers to estimate trip costs, plan a budget, and continuously track their spending throughout the journey. Users can input their initial budget and update their expenditures in real time, enabling the system to display the remaining balance and guide them toward better financial decisions. The outcome is a lightweight, user-friendly application that improves financial planning during travel and demonstrates core Java programming skills.
GPS Map Camera
Google
Hyderabad, Telangana, India
Gcw3+wm6, Aleap Industrial Area, Gajularamaram, Hyderabad, Telangana 500043, India
Lat 17.547267° Long 78.404177°
Tuesday, 09/12/2025 08:41 AM GMT +05:30