International Journal of Scientific Research & Engineering Trends
Volume 12, Issue 2, Mar-Apr-2026, ISSN (Online): 2395-566X

Fairshare System: Bill Splitting and Expense Managing Assistant

Ishita Shinde[1], Tanushri Jadhav[2], Mrs. Patil P.M.[3]
AISSMS Institute Of Information Technology. Pune, India

Abstract- Managing the shared financial transactions has become a very difficult process with the rapid growth of group-oriented activities like shared accommodation, travel, events, and projects. Calculations and transparency issues, financial management problems, and interpersonal relationship issues result from the calculations performed for the splitting of bills and expense management in the traditional manner. Because of these constraints, a methodical and technologically simple way of managing the shared expense is necessary. Bill Splitting System - This is a simple application used to divide the bill or expense among a group of people. By providing an automated, methodical, and user-centered system that aims to simplify and enhance the management of group expenses, the proposed FairShare System: Bill Splitting and Expense Managing Assistant aims to resolve the problem. In current financial environments, FairShare System is designed to make group expense management simpler and quicker. It splits the bill among all the peoples or with friends. It is a very useful apporoch to avoid misunderstandings amoung group of peoples. It addresses common issues like inconsistent data, wrong settlements, duplicates, and the difficulty in tracking balances. It ensures fairness and transparency in the process using optimized algorithms and minimizes the total number of transactions required for settling debts.This system provides a reliable,scalable,and user friendly way of managing groupexpense with a well-oragnized backend that ensures the security and accuracy of the data.

Keywords- Object-Oriented Programming, Expense Splitting, Abstraction, Code Reusability, Maintainability.

I. INTRODUCTION

Background of the Study

The motivation behind creating this app comes from the general problem that people face in their lives. This problem arises when people want to split their expenses with others. In recent times, many people are living in groups, either due to travel or working on projects together. There were different methods used before to maintain the expenses like making use of sticky note by normal people, professional people uses spreadsheet to record expenses or by using ledger to maintain huge amounts. There are still problems in area like data consistency, there are chances were critical inputs can be missed and the manual errors my creep in. The Data recorders are not always handy and it could be hectic process to have overall view of those expenses[1]

The FairShare System: Bill Splitting and Expense Managing Assistant helps ease the pain of split bills and expense. It's designed to be smart, simple, and actually useful way for bill splitting. Users can apply customizable splitting techniques, record expenses, create groups, and produce optimized settlement summaries with the system. The project turns something that typically causes frustration into a seamless and stress-free experience by streamlining the process of splitting expenses. This solution provides clarity, equity, and peace of mind to shared financial responsibilities, whether it's traveling as a group, managing household bills, or going on a casual outing with friends [2].

Statement of the Problem

Traditionally used methods of dealing with shared expenses reveal significant shortcomings as group-based financial activities increase in complexity. Errors in calculation, duplicate entries, ambiguous tracking of balance, delays in reimbursement, and misunderstandings among members of groups are often due to traditional methods of keeping track of these expenses using spreadsheets and even messaging apps. With an increase in the number of persons involved in these transactions, these methods prove to be ineffective. The need of the hour is to have an organized and automated expense management system to overcome these challenges. In the context of contemporary collaboratives environments, the FairShare System: Bill Splitting and Expense Managing Assistant can achieve the goal of efficiency ,trust and long term usability with precise expense tracking and balance tracker.

Limitations of non-structured system design

In procedural programming, the data and the function that operates on that works on the are generally related to each other. Most of the time, both are written together in the same file. Because of this, making changes or adding new features can become difficult as the program grows. debuagging and maintaining the system is also hard.In spite of having some data protection ,the non-modular structure often leads to code duplication. As the application becomes larger and more complex, performance and efficiency can also decrease.In the

absence of proper data and function organization,the progam may slow down,become unstable and even fail to scale.[8]

II. LITERATURE REVIEW

Author / Source	Method / Tools	Key Objective	Contribution	Limitation
IJCESR (Troindia)	Mobile-based expense tracking & bill splitting app	Develop a system with expense recording and settlement features	Reduces manual calculation errors; provides graphical reports	Limited scalability and optimization techniques
IJRTI – SettleUp Web App	Django + SQLite; graph-based settlement logic	Automate group expense tracking and debt settlement	Optimized transaction minimization; transparent ledger system	No real payment gateway integration
Java Mini Project – Expense Splitter	JavaFX + SQLite; group & split management	Create a user-friendly expense splitter system	Simple UI with modular desktop features	Desktop-only; lacks real-time sync & advanced optimization

III. METHODOLOGY

Research Design: Analytical and Comparative Study

The type of research methodology used in this study is qualitative, analytical, and comparative; it is a qualitative non-experimental study that uses existing software engineering principles as well as available theoretical literature in Computer Science (i.e. research papers, book chapters) as its basis. The purpose of this type of methodology is to identify the causes of software complexity based on a systematic deconstruction of each of those causes and then to analytically demonstrate Why object-oriented programming (OOP) behaves like it does and why the core principles of OOP provide solutions for the structure and concept of software complexity.

Data and Information Sources

In this research, the sources of the data consist of both academic and industry sources. There are three primary areas of literature that comprise this set of sources:

Foundational Literature:

This area includes classic writings about relevant to both procedural and object-oriented programming; definitions of, and metrics for, software complexity; and quality attributes (e.g., maintainability, reusability, extensibility) associated with software engineering.

2. Comparative Case Studies:

This includes academic publications and white papers comparing code metrics (e.g., lines of code, cyclomatic complexity) developed using procedural versus object-oriented programming techniques, and comparing development outcomes of similar systems developed using these two different techniques.

3. Industry Best Practice:.

This includes documentation, design patterns, and case studies related to the use of object-oriented programming in large-scale enterprise systems as developed by major software vendors (e.g., Java, C++, or Python ecosystems).

Analytical Framework

The four stages of analysis are as follows:

1. Identify and isolate specific limitations of the procedural paradigm due to the increased size and longevity of software (ex. tightlycoupled architectures and/or duplication of code).
2. Systematically map each of the core Object-Oriented Programming (OOP) Principles to the problem that each was designed to solve (i.e. Encapsulation, Abstraction, Inheritance & Polymorphism).
3. Use deductive reasoning and existing literature/reference sources to compare modularity, management of change & isolation of faults in programs designed with an OOP-based architecture versus a procedural architecture.
4. Provide hypothetical examples or reference pieces of code (that were not used as part of any experimentation) to demonstrate increased clarity, reuse, & extensibility resulting from the implementation of OOP Principles.

Research Tools

With respect to the primary means of conducting this research, the tools consist of analysing a body of literature and evaluating and synthesising prior studies through reasoning and organising their content. This study evaluates and synthesises existing literature regarding the OOP paradigm to provide analytical and systematic proof of how imperative it is in terms of supporting modern software developed.

IV. RESULTS AND DISCUSSION

Core OOP Principles: Principle-to-Solution Mapping

Encapsulation

Encapsulation provides Global access to data and creates instability for your system.It wraps data and methods into a single unit. Data is hidden with access modifiers and accessed appropriately with methods.

Abstraction
It showing only important details and hiding implementation details that makes it too complex. Provide only the necessary interface while hiding the implementation details,It helps users focus on what an object does, not how it works.

Inheritance
There is a lot of repetition of common code, so inheritance is used to avoid such kind of problem. It is a concept of inherits the properties of parent class into child class. It helps in code reuse in a parent-child relationship through class hierarchy.

Polymorphism
Polymorphism means one thing can take many forms. It allows the same method name to behave differently in different situations. Method bindings are done at runtime allowing for flexibility in behavior.

Dynamic Binding
Dynamic Binding is the process where the method call is resolved at runtime rather than compile time. The method implementation of the actual object is executed, not the reference type.

Practical Implementation in Java
To demonstrate additional OOP constructs, we introduce a new entity: Book. This example incorporates:

- Class definition
- Constructors
- Method overloading
- this keyword
- Class variables
- Object as return type
- Object as argument

Code Implementation and Explanation in Java
Abstraction

```
abstract class Expense {
   protected double amount;
   protected User paidBy;
   public abstract void split(List<User> users);
```

Abstraction is a concept that allows a program to represent only the essential features while ignoring the complex internal details. In the given implementation, the Expense class is defined as an abstract class, which means it serves as a base structure for other classes.

It declares the method split() but does not provide its implementation. This approach ensures that the general behavior is defined at a higher level, while the specific implementation is left to the subclasses. As a result, the system becomes more organized, flexible, and easier to extend without modifying the core structure.

Constructor and This Keyword

```
public Expense(double amount, User paidBy) {
   this.amount = amount;
   this.paidBy = paidBy;
}
```

The Constructor is a special methos of class. It automatically invoked when object of class is created. In this code, the constructor is used to set the values of the variables 'amount' and 'paidBy.'

The 'this' keyword is used to refer to the current object, thus distinguishing between the variable and the parameter if the variable and the parameter have the same names.

Inheritance

```
class EqualExpense extends Expense {
   public EqualExpense(double amount, User paidBy) {
      super(amount, paidBy);
   }
```

Inheritance is something that lets one class use things from another class. So the EqualExpense class is taking things from the Expense class. It does this with the "extends" keyword. This way the EqualExpense class can use things that're already in the Expense class. It does not have to make them.
The EqualExpense class is basically getting properties and methods from the

Expense class. This is really helpful because the EqualExpense class can just use the properties and methods, from the Expense class without having to do work.

Polymorphism+Dynamic binding

```
@Override
public void split(List<User> users) {
   double share = amount / users.size();

   for (User u : users) {
      if (!u.getName().equals(paidBy.getName())) {
         u.addDue(share);
      }
   }
```

The concept of polymorphism is really useful because it lets us do things in different situations using the same method. We use polymorphism here because we declare a method in the class but it is actually implemented in the EqualExpense class. When

we want to choose which method to use we use something called binding.
This is what polymorphism is about using the same method to do different things like, in the EqualExpense class.

Encapsulation

```
class User {
   private String name;
   private double totalDue;

   public String getName() {
      return name;
   }
 public void addDue(double amount) {
      totalDue += amount;
   }
```

EEncapsulation can be defined as the process of protecting the data by keeping it private and providing access to the data through methods. In this example, the variables name and totalDue are private. This means that these variables are not accessible directly. Instead, methods are used to interact with these variables. This provides safety to the data.

Method Overloading

```
public void addDue(double amount) {
   totalDue += amount;
}

public void addDue(double amount, String note) {
   totalDue += amount;
   System.out.println("Note: " + note);
}
```

Method Overloading is a feature that allows a class to have more than one method with the same name but different parameters.
In this case, the addDue method has been defined in two ways. It has one parameter and another parameter that includes a note. This has made the method more useful.

Main Class

```
public class ExpenseSplitter {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      List<User> users = new ArrayList<>();
   }
```

The main class has the method. The main class is where the program starts. The main class controls what the program will do. The main class gets the input from the user, makes objects, and calls the methods the program needs. The main class is where all the pieces of the program come together.
The main class does many things, such as getting the input from the user and making objects for the program. The main method is very important in the main class.

Creating Object

```
users.add(new User(name));
```

Object creation is the process of creating an instance of a class. In this example, a new User object is created using the new keyword and added to a list. Each object represents a real user in the system and stores individual data.

The above sequence shows how different Object-Oriented Programming concepts work together to create a well-organized and efficient system, improving both code clarity and reusability.

V. CONCLUSION

The system demonstrates the advantages that can be obtained by using bill splitting software based on object-oriented programming. It automates every step involved in the process. This means that the Expense Splitting project will effectively solve the problems that arise in the management of shared expenses. This will also minimize the amount of human effort needed to settle the shares. It will also minimize the involvement of humans in the splitting process to produce accurate results.

In modern software development frameworks, object-oriented programming has become the main programming approach due to its continued evolution. This is because it has the ability to control the complexity of the program and also to develop expanding applications.

REFERENCES

1. Rishabh Agrawal, Research scholar, “BILL SPLITTING AND EXPENSE MANAGING ASSISTANT”,Department of Information Technology, Oriental
Institute of Science & Technology, Thakral Nagar, Opp. Patel Nagar, Raisan Road, Bhopal.
2. Faiza Sayed, Sthairya Shetty and Hasan Phudinawala ,Department of Computer Science 1,2,3Royal College of Arts, Science and Commerce
(Autonomous), Mira Road (East), India

3. Nishant Choudhary, "Splitting Bills, Simplified: Developing a Splitwise-like App with Java and SOLID Principles", Noida, Uttar Pradesh, India
4. V. V. Mehtre and U. R. Verma, "Concepts Related to Object Oriented Programming(OOPs): Basics," Department of Electrical Engineering, Bharati Vidyapeeth Deemed University College of Engineering, Pune, India
5. V. V. Mehtre and U. R. Verma, "Concepts Related to Object Oriented Programming(OOPs): Basics," Department of Electrical Engineering, Bharati Vidyapeeth Deemed University College of Engineering, Pune, India
6. X. Zhang, J. D. Crabtree, M. G. Terwilliger and T. R. Redman, "Assessing Students' Object Oriented Programming Skills with Java: The 'Department-Employee' Project," Journal of Computer Information Systems.
7. R. S. Pressman and B. R. Maxim, Software Engineering: A Practitioner's
Approach, 8th ed. New York, NY, USA: McGraw-Hill Education, 2015.