NAME: GAYATHRI K

COURSE: JAVA PROGRAMMING

PHONE NO: 9148530832

EMAIL : pinkgayathri588@gmail.com

WEEK-3 (PROJECT: Expense Tracker)

Motioncut

The week-3 project The Expense Tracker project aims to create a robust and user-friendly application that allows users to manage and monitor their expenses efficiently.

JAVA PROGRAMMING PROJECT WEEK -3

Here's a basic implementation of an Expense Tracker in Java. This implementation focuses on console-based interactions and uses file handling for persistence.

PROGRAM:

import java.io.\*;

import java.util.\*;

class User {

private String username;

private String password;

public User(String username, String password) {

this.username = username;

this.password = password;

}

public String getUsername() {

return username;

}

public String getPassword() {

return password;

}

}

class Expense {

private Date date;

private String category;

private double amount;

public Expense(Date date, String category, double amount) {

this.date = date;

this.category = category;

this.amount = amount;

}

public Date getDate() {

return date;

}

public String getCategory() {

return category;

}

public double getAmount() {

return amount;

}

}

class ExpenseTracker {

private List<User> users;

private List<Expense> expenses;

public ExpenseTracker() {

users = new ArrayList<>();

expenses = new ArrayList<>();

}

public void addUser(String username, String password) {

users.add(new User(username, password));

}

public boolean authenticate(String username, String password) {

for (User user : users) {

if (user.getUsername().equals(username) && user.getPassword().equals(password)) {

return true;

}

}

return false;

}

public void addExpense(Date date, String category, double amount) {

expenses.add(new Expense(date, category, amount));

}

public void listExpenses() {

for (Expense expense : expenses) {

System.out.println("Date: " + expense.getDate() + ", Category: " + expense.getCategory() + ", Amount: " + expense.getAmount());

}

}

public void calculateCategorySum() {

Map<String, Double> categorySum = new HashMap<>();

for (Expense expense : expenses) {

String category = expense.getCategory();

double amount = expense.getAmount();

categorySum.put(category, categorySum.getOrDefault(category, 0.0) + amount);

}

for (Map.Entry<String, Double> entry : categorySum.entrySet()) {

System.out.println("Category: " + entry.getKey() + ", Total Expenses: " + entry.getValue());

}

}

public void saveExpensesToFile(String filename) {

try (ObjectOutputStream outputStream = new ObjectOutputStream(new FileOutputStream(filename))) {

outputStream.writeObject(expenses);

System.out.println("Expenses saved successfully!");

} catch (IOException e) {

e.printStackTrace();

}

}

public void loadExpensesFromFile(String filename) {

try (ObjectInputStream inputStream = new ObjectInputStream(new FileInputStream(filename))) {

expenses = (List<Expense>) inputStream.readObject();

System.out.println("Expenses loaded successfully!");

} catch (IOException | ClassNotFoundException e) {

e.printStackTrace();

}

}

}

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

ExpenseTracker expenseTracker = new ExpenseTracker();

// Sample user registration

expenseTracker.addUser("user1", "password1");

// Sample expenses

expenseTracker.addExpense(new Date(), "Food", 50.0);

expenseTracker.addExpense(new Date(), "Transportation", 30.0);

expenseTracker.addExpense(new Date(), "Food", 20.0);

boolean loggedIn = false;

while (!loggedIn) {

System.out.print("Enter username: ");

String username = scanner.nextLine();

System.out.print("Enter password: ");

String password = scanner.nextLine();

loggedIn = expenseTracker.authenticate(username, password);

if (!loggedIn) {

System.out.println("Invalid username or password. Please try again.");

}

}

boolean running = true;

while (running) {

System.out.println("\n1. Add Expense\n2. List Expenses\n3. Calculate Category-wise Summation\n4. Save Expenses to File\n5. Load Expenses from File\n6. Exit");

System.out.print("Enter your choice: ");

int choice = scanner.nextInt();

scanner.nextLine(); // Consume newline character

switch (choice) {

case 1:

System.out.print("Enter expense date (yyyy-MM-dd): ");

String dateString = scanner.nextLine();

Date date = null;

try {

date = new SimpleDateFormat("yyyy-MM-dd").parse(dateString);

} catch (ParseException e) {

System.out.println("Invalid date format. Please enter date in yyyy-MM-dd format.");

break;

}

System.out.print("Enter category: ");

String category = scanner.nextLine();

System.out.print("Enter amount: ");

double amount = scanner.nextDouble();

expenseTracker.addExpense(date, category, amount);

System.out.println("Expense added successfully!");

break;

case 2:

System.out.println("List of Expenses:");

expenseTracker.listExpenses();

break;

case 3:

System.out.println("Category-wise Summation:");

expenseTracker.calculateCategorySum();

break;

case 4:

expenseTracker.saveExpensesToFile("expenses.dat");

break;

case 5:

expenseTracker.loadExpensesFromFile("expenses.dat");

break;

case 6:

running = false;

break;

default:

System.out.println("Invalid choice. Please enter a number between 1 and 6.");

}

}

scanner.close();

}

}

OUTPUT :

Enter username: user1

Enter password: password1

1. Add Expense

2. List Expenses

3. Calculate Category-wise Summation

4. Save Expenses to File

5. Load Expenses from File

6. Exit

Enter your choice: 1

Enter expense date (yyyy-MM-dd): 2024-04-05

Enter category: Food

Enter amount: 50

Expense added successfully!

1. Add Expense

2. List Expenses

3. Calculate Category-wise Summation

4. Save Expenses to File

5. Load Expenses from File

6. Exit

Enter your choice: 2

List of Expenses:

Date: Mon Apr 05 00:00:00 UTC 2024, Category: Food, Amount: 50.0

1. Add Expense

2. List Expenses

3. Calculate Category-wise Summation

4. Save Expenses to File

5. Load Expenses from File

6. Exit

Enter your choice: 3

Category-wise Summation:

Category: Food, Total Expenses: 50.0

1. Add Expense

2. List Expenses

3. Calculate Category-wise Summation

4. Save Expenses to File

5. Load Expenses from File

6. Exit

Enter your choice: 4

Expenses saved successfully!

1. Add Expense

2. List Expenses

3. Calculate Category-wise Summation

4. Save Expenses to File

5. Load Expenses from File

6. Exit

Enter your choice: 5

Expenses loaded successfully!

1. Add Expense

2. List Expenses

3. Calculate Category-wise Summation

4. Save Expenses to File

5. Load Expenses from File

6. Exit

Enter your choice: 2

List of Expenses:

Date: Mon Apr 05 00:00:00 UTC 2024, Category: Food, Amount: 50.0

1. Add Expense

2. List Expenses

3. Calculate Category-wise Summation

4. Save Expenses to File

5. Load Expenses from File

6. Exit

Enter your choice: 6