**Project Structure**

1. **User Registration**
   * Create a class User to handle user details.
   * Implement methods for user registration and login.
   * Securely store user credentials (consider hashing passwords).
2. **Expense Entry**
   * Create a class Expense to handle expense details (date, category, amount).
   * Implement methods to add new expenses for a user.
3. **Expense Listing**
   * Store expenses in a data structure (e.g., a list or a database).
   * Implement methods to list expenses, and provide sorting and filtering options.
4. **Category-wise Summation**
   * Implement a method to calculate and display total expenses by category.
5. **Persistence**
   * Use file I/O to save and load expense data (consider using JSON or another format).
   * Ensure data is saved for each user and can be reloaded upon login.

**Basic Class Design(code):**

class User {

private String username;

private String passwordHash; // Store hashed password

private List<Expense> expenses;

// Constructors, getters, and setters

// Methods for registration, login, and managing expenses

}

class Expense {

private LocalDate date;

private String category;

private double amount;

// Constructors, getters, and setters

}

class ExpenseTracker {

private Map<String, User> users; // Store users by username

// Methods for user registration, login, and managing expenses

// Methods for saving and loading data

}

**Implementation Steps**

1. **User Registration and Login**
   * Implement a registration method to create a new User object.
   * Implement a login method to authenticate users.
2. **Adding and Listing Expenses**
   * Implement methods in the User class to add expenses.
   * Implement methods to list all expenses, with options to sort and filter.
3. **Category-wise Summation**
   * Implement a method to iterate over expenses and calculate the sum for each category.
4. **Data Persistence**
   * Implement methods to save user data to a file.
   * Implement methods to load user data from a file at startup.

### Example Code Snippet

Here's a simple example to get you started with user registration and adding expenses:

import java.util.\*;

import java.time.LocalDate;

class User {

private String username;

private String passwordHash;

private List<Expense> expenses;

public User(String username, String passwordHash) {

this.username = username;

this.passwordHash = passwordHash;

this.expenses = new ArrayList<>();

}

public String getUsername() {

return username;

}

public boolean checkPassword(String passwordHash) {

return this.passwordHash.equals(passwordHash);

}

public void addExpense(Expense expense) {

expenses.add(expense);

}

public List<Expense> getExpenses() {

return expenses;

}

}

class Expense {

private LocalDate date;

private String category;

private double amount;

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

this.date = date;

this.category = category;

this.amount = amount;

}

// Getters

}

class ExpenseTracker {

private Map<String, User> users;

public ExpenseTracker() {

users = new HashMap<>();

}

public void registerUser(String username, String passwordHash) {

if (!users.containsKey(username)) {

users.put(username, new User(username, passwordHash));

} else {

System.out.println("User already exists!");

}

}

public User loginUser(String username, String passwordHash) {

User user = users.get(username);

if (user != null && user.checkPassword(passwordHash)) {

return user;

} else {

System.out.println("Invalid credentials!");

return null;

}

}

// Methods for adding expenses, listing expenses, and calculating category-wise summation

}

public class Main {

public static void main(String[] args) {

ExpenseTracker tracker = new ExpenseTracker();

tracker.registerUser("testUser", "testHash");

User user = tracker.loginUser("testUser", "testHash");

if (user != null) {

user.addExpense(new Expense(LocalDate.now(), "Food", 12.50));

user.addExpense(new Expense(LocalDate.now(), "Transport", 7.00));

for (Expense expense : user.getExpenses()) {

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

}

}

}

}

This code provides a basic framework. You can expand it by adding more functionality such as sorting, filtering, and persistence.