

Java Project

EXPENSE TRACKER

(A Personal expense tracking system)

1. Introduction

The Expense Tracker Application is a console-based Java application designed to help users manage their financial transactions, categorize expenses, and set and track budgets. The application provides functionality to add, update, delete, and view transactions, along with managing user accounts and budget limits.

2. Objective

This application aims to help users manage their finances by tracking income and expenses, setting and monitoring budgets, and providing insights into spending habits. It automates financial tracking, alerts users of potential overspending, and ensures data security and privacy. The ultimate goal is to empower users to make informed financial decisions and achieve their financial goals.

3. Features

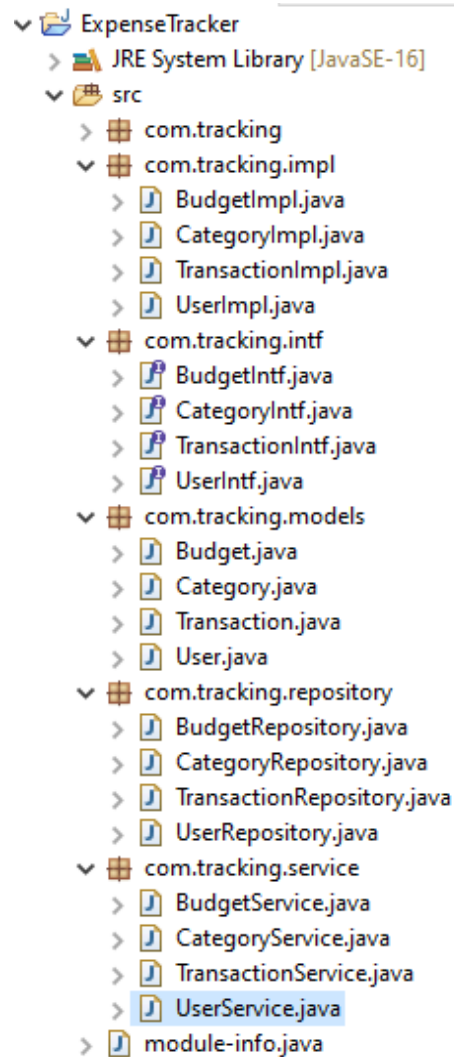
- **User Authentication:** Secure login for users.
- **Category Management:** Users can create, update, and delete categories for transactions.
- **Transaction Management:** Add, update, and delete transactions with income or expense types.
- **Budget Management:** Set and track budgets for different categories. Alerts users when expenses approach budget limits.
- **Expense Tracking:** Calculate monthly expenses for each category.
- **Reset Budgets:** Automatically reset budget expenses at the beginning of each month.

4.System Architecture

The application follows a layered architecture:

- **Model Layer:** Contains the data classes representing Users, Categories, Transactions, and Budgets.
- **Service Layer:** Contains the business logic for managing users, categories, transactions, and budgets.
- **Repository Layer:** Provides data storage and retrieval functionality.
- **UI Layer:** Handles user input through the console interface.(Not implemented)

5.Project Structure



6.Modules

6.1 User Management

Features:

- Login functionality.
- Update user details.
- Logout functionality.

Usage Examples:

- **User Login:** Existing users can log in using their credentials to access their personalized expense tracker dashboard.
- **Edit User Details:** Users can update their profile information, such as changing their password or updating their email address.

6.2 Category Management

Features:

- Create new categories.
- View, update, and delete existing categories.

Usage Examples:

- **Add Category:** Users can create custom categories like "Groceries," "Transportation," or "Entertainment" to better organize their expenses.
- **View Categories:** Users can view a list of all categories they have created.
- **Delete Category:** Users can delete a category they no longer use, provided there are no associated transactions.

6.3 Transaction Management

Features:

- Add transactions with details like amount, type (income/expense), and date.
- Update and delete transactions.

- Track transactions by date and category.

Usage Examples:

- **Add Transaction:** Users can record expenses or income by selecting a category, entering the amount, and providing a description.
- **View Transactions by Month:** Users can filter and view all transactions for a specific month to analyze their spending patterns.
- **Update/Delete Transaction:** Users can update or delete existing transactions to correct errors or remove unnecessary records.

6.4 Budget Management

Features:

- Set budgets for different categories.
- Automatically reset budgets at the start of each month.
- Alert user if expenses approach or exceed the budget limit.

Usage Examples:

- **Set Budget:** Users can allocate a budget to a specific category, such as setting a Rs.5000 limit for "Groceries" for the month.
- **Track Budget Usage:** The application automatically tracks expenses against the set budget and provides updates.
- **Reset Budgets:** At the start of each month, the budget expenses are reset to zero if no transactions were made in that category.

7.Code

7.1 User Module

7.1.1 User.java - Model file

```
package com.tracking.models;

public class User {
    private int userID;
    private String userName;
    private String email;
    private String password;
    private String created_at;
    private String updated_at;

    // Getters and Setters
    public int getUserID() { return userID; }
    public void setUserID(int userID) { this.userID = userID; }
    public String getUserName() { return userName; }
    public void setUserName(String userName) { this.userName = userName; }
    public String getEmail() { return email; }
    public void setEmail(String email) { this.email = email; }
    public String getPassword() { return password; }
    public void setPassword(String password) { this.password = password; }
    public String getCreated_at() { return created_at; }
    public void setCreated_at(String created_at) { this.created_at = created_at; }
    public String getUpdated_at() { return updated_at; }
    public void setUpdated_at(String updated_at) { this.updated_at = updated_at; }
}
```

7.1.2 UserRepository.java - Repository file

```
package com.tracking.repository;

import com.tracking.models.User;
import java.util.HashMap;
import java.util.Map;
```

```

public class UserRepository {
    private Map<Integer, User> userMap = new HashMap<>();
    private static int userIDCounter = 1;

    public UserRepository() {
        // Initialize default users
        for (int i = 1; i <= 10; i++) {
            User user = new User();
            user.setUserID(userIDCounter++);
            user.setUserName("User" + i);
            user.setEmail("user" + i + "@gmail.com");
            user.setPassword("password" + i);
            user.setCreated_at("2024-08-12");
            user.setUpdated_at("2024-08-12");
            addUser(user);
        }
    }

    // CRUD Operations
    public void addUser(User user) { userMap.put(user.getUserID(), user); }
    public User getUser(int userID) { return userMap.get(userID); }
    public void updateUser(User user) { userMap.put(user.getUserID(), user); }

    // Authentication
    public User authenticate(String email, String password) {
        for (User user : userMap.values()) {
            if (user.getEmail().equals(email) && user.getPassword().equals(password)) {
                return user;
            }
        }
        return null;
    }
}

```

7.1.3 UserIntf.java - Interface file

```

package com.tracking.intf;

import com.tracking.models.User;

```

```
public interface UserIntf {  
    void addUser(User user);  
    User getUser(int userID);  
    void updateUser(User user);  
    User authenticate(String email, String password);  
}
```

7.1.4 UserImpl.java - Implementation file

```
package com.tracking.impl;  
  
import com.tracking.intf.UserIntf;  
import com.tracking.models.User;  
import com.tracking.repository.UserRepository;  
  
public class UserImpl implements UserIntf {  
    private UserRepository userRepository;  
  
    public UserImpl(UserRepository userRepository) {  
        this.userRepository = userRepository;  
    }  
  
    @Override  
    public void addUser(User user) { userRepository.addUser(user); }  
  
    @Override  
    public User getUser(int userID) { return userRepository.getUser(userID); }  
  
    @Override  
    public void updateUser(User user) { userRepository.updateUser(user); }  
  
    @Override  
    public User authenticate(String email, String password) {  
  
        return userRepository.authenticate(email, password);  
    }  
}
```

7.1.5 UserService.java - Service file

```
package com.tracking.service;

import java.util.Scanner;

import com.tracking.SessionManager;
import com.tracking.impl.UserImpl;

import com.tracking.models.User;

public class UserService {

    private UserImpl userImpl;

    public UserService(UserImpl userImpl) {
        this.userImpl = userImpl;
    }

    public void viewUserDetails() {
        User currentUser = SessionManager.getCurrentUser();
        System.out.println("User Details:");
        System.out.println("ID: " + currentUser.getUserID());
        System.out.println("Username: " + currentUser.getUserName());
        System.out.println("Email: " + currentUser.getEmail());
        System.out.println("Created At: " + currentUser.getCreated_at());
        System.out.println("Updated At: " + currentUser.getUpdated_at());
    }

    public void updateUserDetails(Scanner scanner) {
        User currentUser = SessionManager.getCurrentUser();
        System.out.println("Update User Details:");
```



```

        System.out.print("New Username: ");
        String newUsername = scanner.nextLine();
        System.out.print("New Email: ");
        String newEmail = scanner.nextLine();
        System.out.print("New Password: ");
        String newPassword = scanner.nextLine();

        currentUser.setUserName(newUsername);
        currentUser.setEmail(newEmail);
        currentUser.setPassword(newPassword);
        currentUser.setUpdated_at("2024-08-12");

        userImpl.updateUser(currentUser);
        System.out.println("User details updated successfully.");
    }
}

```

7.2 Category Module

7.2.1 Category.java - Model file

```

package com.tracking.models;

public class Category {
    private int categoryID;
    private int userID;
    private String categoryName;
    private String description;

    // Getters and Setters
    public int getCategoryID() { return categoryID; }
    public void setCategoryID(int categoryID) { this.categoryID = categoryID; }
    public int getUserID() { return userID; }
}

```

```

public void setUserID(int userID) { this.userID = userID; }
public String getCategoryName() { return categoryName; }
public void setCategoryName(String categoryName) { this.categoryName = categoryName; }
public String getDescription() { return description; }
public void setDescription(String description) { this.description = description; }

@Override
public String toString() {
    return "Category{" +
        "id=" + categoryID +
        ", name=" + categoryName + "\" +
        ", description=" + description + "\" +
        ", userID=" + userID +
        "'";
}
}

```

7.2.2 CategoryRepository.java - Repository file

```

package com.tracking.repository;

import com.tracking.models.Category;
import java.util.HashMap;
import java.util.Map;

public class CategoryRepository {
    private Map<Integer, Category> categoryMap = new HashMap<>();
    private static int categoryIDCounter = 1;

    public CategoryRepository() {
        // Initialize with a default category for each user
        for (int userID = 1; userID <= 10; userID++) {
            Category defaultCategory = new Category();
            defaultCategory.setCategoryID(categoryIDCounter);
            defaultCategory.setUserID(userID);
            defaultCategory.setCategoryName("Groceries");
            defaultCategory.setDescription("Spending on food and everyday essentials");
            addCategory(defaultCategory);
        }
    }
}

```

```

// CRUD Operations
public void addCategory(Category category) {
    category.setCategoryID(categoryIDCounter++);
    categoryMap.put(category.getCategoryID(), category);
}

public Category getCategory(int categoryID) {
    return categoryMap.get(categoryID);
}

public void updateCategory(Category category) {
    categoryMap.put(category.getCategoryID(), category);
}

public void deleteCategory(int categoryID) {
    categoryMap.remove(categoryID);
}

public Map<Integer, Category> getAllCategories() {
    return categoryMap;
}

}

```

7.2.3 CategoryIntf.java - Interface file

```

package com.tracking.intf;

import com.tracking.models.Category;
import java.util.Map;

public interface CategoryIntf {
    void addCategory(Category category);
    Category getCategory(int categoryID);
    void updateCategory(Category category);
    void deleteCategory(int categoryID);
    Map<Integer, Category> getAllCategories();
}

```

7.2.4 CategoryImpl.java - Implementation file

```
package com.tracking.impl;

import com.tracking.intf.CategoryIntf;
import com.tracking.models.Category;
import com.tracking.repository.CategoryRepository;
import java.util.Map;

public class CategoryImpl implements CategoryIntf {
    private CategoryRepository categoryRepository;

    public CategoryImpl(CategoryRepository categoryRepository) {
        this.categoryRepository = categoryRepository;
    }

    @Override
    public void addCategory(Category category) {
        categoryRepository.addCategory(category);
    }

    @Override
    public Category getCategory(int categoryID) {
        return categoryRepository.getCategory(categoryID);
    }

    @Override
    public void updateCategory(Category category) {
        categoryRepository.updateCategory(category);
    }

    @Override
    public void deleteCategory(int categoryID) {
        categoryRepository.deleteCategory(categoryID);
    }

    @Override
    public Map<Integer, Category> getAllCategories() {
        return categoryRepository.getAllCategories();
    }
}
```

```
}  
}
```

7.2.5 CategoryService.java - Service file

```
package com.tracking.service;  
  
import com.tracking.impl.CategoryImpl;  
import com.tracking.models.Category;  
  
import java.util.Scanner;  
  
public class CategoryService {  
    private CategoryImpl categoryImpl;  
  
    public CategoryService(CategoryImpl categoryImpl) {  
        this.categoryImpl = categoryImpl;  
    }  
  
    public void addCategory(Scanner scanner, int loggedInUserID) {  
        Category category = new Category();  
        category.setUserID(loggedInUserID);  
  
        System.out.print("Enter category name: ");  
        category.setCategoryName(scanner.nextLine());  
  
        System.out.print("Enter category description: ");  
        category.setDescription(scanner.nextLine());  
  
        categoryImpl.addCategory(category);  
        System.out.println("Category added successfully.");  
    }  
  
    public void updateCategory(Scanner scanner, int loggedInUserID) {  
        System.out.print("Enter category ID to update: ");  
        int categoryID = scanner.nextInt();  
        scanner.nextLine();  
  
        Category category = categoryImpl.getCategory(categoryID);
```

```

if (category != null && category.getUserID() == loggedInUserID) {
    System.out.print("Enter new category name (leave blank to keep current): ");
    String nameInput = scanner.nextLine();
    if (!nameInput.isEmpty()) {
        category.setCategoryName(nameInput);
    }

    System.out.print("Enter new description (leave blank to keep current): ");
    String description = scanner.nextLine();
    if (!description.isEmpty()) {
        category.setDescription(description);
    }

    categoryImpl.updateCategory(category);
    System.out.println("Category updated successfully.");
} else {
    System.out.println("Category not found or you don't have permission to update it.");
}
}

public void deleteCategory(Scanner scanner, int loggedInUserID) {
    System.out.print("Enter category ID to delete: ");
    int categoryID = scanner.nextInt();
    scanner.nextLine();

    Category category = categoryImpl.getCategory(categoryID);
    if (category != null && category.getUserID() == loggedInUserID) {
        categoryImpl.deleteCategory(categoryID);
        System.out.println("Category deleted successfully.");
    } else {
        System.out.println("Category not found or you don't have permission to delete it.");
    }
}

public void viewCategories(int loggedInUserID) {
    System.out.println("Your Categories:");
    categoryImpl.getAllCategories().values().stream()
        .filter(c -> c.getUserID() == loggedInUserID)
        .forEach(System.out::println);
}

```

```
}  
}
```

7.3 Transaction Module

7.3.1 Transaction.java - Model file

```
package com.tracking.models;
```

```
import java.util.Date;
```

```
public class Transaction {  
    private int transactionID;  
    private int userID;  
    private int categoryID;  
    private double amount;  
    private Date transactionDate;  
    private String description;  
    private String type;  
  
    // Getters and Setters  
    public int getTransactionID() { return transactionID; }  
    public void setTransactionID(int transactionID) { this.transactionID = transactionID; }  
    public int getUserID() { return userID; }  
    public void setUserID(int userID) { this.userID = userID; }  
    public int getCategoryID() { return categoryID; }  
    public void setCategoryID(int categoryID) { this.categoryID = categoryID; }  
    public double getAmount() { return amount; }  
    public void setAmount(double amount) { this.amount = amount; }  
    public Date getTransactionDate() { return transactionDate; }  
    public void setTransactionDate(Date transactionDate) { this.transactionDate =  
transactionDate; }  
    public String getDescription() { return description; }  
    public void setDescription(String description) { this.description = description; }  
    public String getType() { return type; }  
    public void setType(String type) { this.type = type; }  
  
    @Override  
    public String toString() {  
        return "Transaction{" +
```

```

        "id=" + transactionID +
        ", userID=" + userID +
        ", categoryID=" + categoryID +
        ", amount=" + amount +
        ", transactionDate=" + transactionDate +
        ", description=" + description + "\" +
        ", type=" + type +
        '}'
    }
}

```

7.3.2 TransactionRepository.java - Repository file

```

package com.tracking.repository;

import com.tracking.models.Transaction;

import java.time.LocalDate;
import java.util.*;
import java.time.ZoneId;
import java.util.Date;

public class TransactionRepository {
    private Map<Integer, Transaction> transactionMap = new HashMap<>();
    private static int transactionIDCounter = 1;

    public void addTransaction(Transaction transaction) {
        transaction.setTransactionID(transactionIDCounter++);
        transactionMap.put(transaction.getTransactionID(), transaction);
    }

    public Transaction getTransaction(int transactionID) {
        return transactionMap.get(transactionID);
    }

    public void updateTransaction(Transaction transaction) {
        transactionMap.put(transaction.getTransactionID(), transaction);
    }

    public void deleteTransaction(int transactionID) {

```



```

        transactionMap.remove(transactionID);
    }

    public List<Transaction> getTransactionsByDate(LocalDate date) {
        List<Transaction> transactions = new ArrayList<>();

        for (Transaction transaction : transactionMap.values()) {
            Date date_date = transaction.getTransactionDate();
            LocalDate transactionDate =
date_date.toInstant().atZone(ZoneId.systemDefault()).toLocalDate();
            if (transactionDate.equals(date)) {
                transactions.add(transaction);
            }
        }
        return transactions;
    }

    public List<Transaction> getTransactionsByMonth(LocalDate date) {
        List<Transaction> transactions = new ArrayList<>();
        for (Transaction transaction : transactionMap.values()) {
            Date date_date = transaction.getTransactionDate();
            LocalDate transactionDate =
date_date.toInstant().atZone(ZoneId.systemDefault()).toLocalDate();

            if (transactionDate.getYear() == date.getYear() && transactionDate.getMonth() ==
date.getMonth()) {
                transactions.add(transaction);
            }
        }
        return transactions;
    }

    public List<Transaction> getTransactionsByCategoryAndMonth(int categoryID) {
        List<Transaction> transactions = new ArrayList<>();
        LocalDate currentDate = LocalDate.now();
        for (Transaction transaction : transactionMap.values()) {
            Date date_date = transaction.getTransactionDate();
            int transactionCategoryID = transaction.getCategoryID();
            LocalDate transactionDate =
date_date.toInstant().atZone(ZoneId.systemDefault()).toLocalDate();

```

```

        if (transactionDate.getYear() == currentDate.getYear() &&
            transactionDate.getMonth() == currentDate.getMonth() &&
            transactionCategoryID == categoryID &&
            "Expense".equalsIgnoreCase(transaction.getType())) {

            transactions.add(transaction);
        }
    }
    return transactions;
}
}

```

7.3.3 TransactionIntf.java - Interface file

```

package com.tracking.intf;

import com.tracking.models.Transaction;

import java.time.LocalDate;
import java.util.List;

public interface TransactionIntf {
    void addTransaction(Transaction transaction);
    Transaction getTransaction(int transactionID);
    void updateTransaction(Transaction transaction);
    void deleteTransaction(int transactionID);
    List<Transaction> getTransactionsByDate(LocalDate date);
    List<Transaction> getTransactionsByMonth(LocalDate date);
    List<Transaction> getTransactionsByCategoryAndMonth(int categoryID);
}

```

7.3.4 TransactionImpl.java - Implementation file

```

package com.tracking.impl;

```

```
import com.tracking.intf.TransactionIntf;
import com.tracking.models.Transaction;
import com.tracking.repository.TransactionRepository;

import java.time.LocalDate;
import java.util.List;

public class TransactionImpl implements TransactionIntf {
    private TransactionRepository transactionRepository;

    public TransactionImpl(TransactionRepository transactionRepository) {
        this.transactionRepository = transactionRepository;
    }

    @Override
    public void addTransaction(Transaction transaction) {
        transactionRepository.addTransaction(transaction);
    }

    @Override
    public Transaction getTransaction(int transactionID) {
        return transactionRepository.getTransaction(transactionID);
    }

    @Override
    public void updateTransaction(Transaction transaction) {
        transactionRepository.updateTransaction(transaction);
    }

    @Override
    public void deleteTransaction(int transactionID) {
        transactionRepository.deleteTransaction(transactionID);
    }

    @Override
    public List<Transaction> getTransactionsByDate(LocalDate date) {
        return transactionRepository.getTransactionsByDate(date);
    }
}
```

```

@Override
public List<Transaction> getTransactionsByMonth(LocalDate date) {
    return transactionRepository.getTransactionsByMonth(date);
}

@Override
public List<Transaction> getTransactionsByCategoryAndMonth(int categoryID){
    return transactionRepository.getTransactionsByCategoryAndMonth(categoryID);
}
}

```

7.3.5 TransactionService.java - Service file

```

package com.tracking.service;

import com.tracking.impl.TransactionImpl;
import com.tracking.impl.BudgetImpl;

import com.tracking.impl.CategoryImpl;
import com.tracking.models.Budget;
import com.tracking.models.Transaction;

import java.time.LocalDate;
import java.util.Scanner;
import java.time.ZoneId;
import java.util.Date;
import java.util.List;
import java.util.stream.Collectors;

public class TransactionService {
    private TransactionImpl transactionImpl;
    private CategoryImpl categoryService;
    private BudgetImpl budgetService;

    public TransactionService(TransactionImpl transactionImpl, CategoryImpl categoryService,
        BudgetImpl budgetService) {
        this.transactionImpl = transactionImpl;
        this.categoryService = categoryService;
        this.budgetService = budgetService;
    }
}

```

```

public void addTransactionWithCategory(Scanner scanner, int loggedInUserID) {
    Transaction transaction = new Transaction();
    transaction.setUserID(loggedInUserID);

    System.out.println("Select a category:");
    categoryService.getAllCategories().values().stream()
        .filter(c -> c.getUserID() == loggedInUserID)
        .forEach(c ->
            System.out.println(c.getCategoryID() + ": " + c.getCategoryName()));

    System.out.print("Enter category ID: ");
    transaction.setCategoryID(scanner.nextInt());
    scanner.nextLine();

    System.out.print("Enter amount: ");
    transaction.setAmount(scanner.nextDouble());
    scanner.nextLine();

    System.out.print("Enter description: ");
    transaction.setDescription(scanner.nextLine());

    System.out.print("Enter type (Income/Expense): ");
    transaction.setType(scanner.nextLine());

    System.out.print("Enter date (YYYY-MM-DD): ");
    LocalDate localDate = LocalDate.parse(scanner.nextLine());
    Date date = Date.from(localDate.atStartOfDay(ZoneId.systemDefault()).toInstant());

    transaction.setTransactionDate(date);

    transactionImpl.addTransaction(transaction);
    System.out.println("Transaction added successfully.");

    if (transaction.getType().equalsIgnoreCase("Expense")) {
        budgetService.updateExpense(transaction.getCategoryID(), transaction.getAmount(),
            loggedInUserID);

        List<Budget> budgets = budgetService.getBudgets().stream()
            .filter(b -> b.getUserID() == loggedInUserID)

```

```

        .filter(b -> b.getCategoryID() == transaction.getCategoryID())
        .collect(Collectors.toList());

    if (!budgets.isEmpty()) {
        for (Budget budget : budgets) {
            double currentExpense = budget.getExpense();
            double budgetAmount = budget.getAmount();
            double expensePercentage = (currentExpense / budgetAmount) * 100;

            if (expensePercentage >= 90) {
                System.out.println("Alert: You have used " + expensePercentage + "% of your
budget for this category.");
            }
        }
    }
}

}

public void updateTransaction(Scanner scanner, int loggedInUserID) {
    System.out.print("Enter transaction ID to update: ");
    int transactionID = scanner.nextInt();
    scanner.nextLine();

    Transaction transaction = transactionImpl.getTransaction(transactionID);
    if (transaction != null && transaction.getUserID() == loggedInUserID) {
        double originalAmount = transaction.getAmount();
        String originalType = transaction.getType();
        System.out.print("Enter new amount (leave blank to keep current): ");
        String amountInput = scanner.nextLine();
        if (!amountInput.isEmpty()) {
            transaction.setAmount(Double.parseDouble(amountInput));
        }

        System.out.print("Enter new description (leave blank to keep current): ");
        String description = scanner.nextLine();
        if (!description.isEmpty()) {
            transaction.setDescription(description);
        }
    }
}

```

```

        System.out.print("Enter new type (Income/Expense, leave blank to keep current): ");
        String type = scanner.nextLine();
        if (!type.isEmpty()) {
            transaction.setType(type);
        }

        System.out.print("Enter new date (YYYY-MM-DD, leave blank to keep current): ");
        String dateInput = scanner.nextLine();
        if (!dateInput.isEmpty()) {
            LocalDate localDate = LocalDate.parse(scanner.nextLine());
            Date date = Date.from(localDate.atStartOfDay(ZoneId.systemDefault()).toInstant());

            transaction.setTransactionDate(date);
        }

        transactionImpl.updateTransaction(transaction);
        System.out.println("Transaction updated successfully.");

        if ("Expense".equalsIgnoreCase(originalType) &&
            "Expense".equalsIgnoreCase(transaction.getType())) {
            double difference = transaction.getAmount() - originalAmount;
            budgetService.updateExpense(transaction.getCategoryID(), difference,
loggedInUserID);
        } else if ("Expense".equalsIgnoreCase(originalType)) {
            budgetService.updateExpense(transaction.getCategoryID(), -originalAmount,
loggedInUserID);
        } else if ("Expense".equalsIgnoreCase(transaction.getType())) {
            budgetService.updateExpense(transaction.getCategoryID(), transaction.getAmount(),
loggedInUserID);
        }
    } else {
        System.out.println("Transaction not found or you don't have permission to update it.");
    }
}

public void deleteTransaction(Scanner scanner, int loggedInUserID) {
    System.out.print("Enter transaction ID to delete: ");
    int transactionID = scanner.nextInt();
    scanner.nextLine();
    Transaction transaction = transactionImpl.getTransaction(transactionID);

```

```

        if (transaction != null && transaction.getUserID() == loggedInUserID) {
            transactionImpl.deleteTransaction(transactionID);
            System.out.println("Transaction deleted successfully.");

            if ("Expense".equalsIgnoreCase(transaction.getType())) {
                budgetService.updateExpense(transaction.getCategoryID(), -transaction.getAmount(),
loggedInUserID);
            }
        } else {
            System.out.println("Transaction not found or you don't have permission to delete it.");
        }
    }

    public void viewTransactionsByDate(Scanner scanner, int loggedInUserID) {
        System.out.print("Enter date (YYYY-MM-DD): ");
        LocalDate date = LocalDate.parse(scanner.nextLine());
        System.out.println(date);

        var transactions = transactionImpl.getTransactionsByDate(date).stream()
            .filter(c -> c.getUserID() == loggedInUserID)
            .collect(Collectors.toList());
        if (!transactions.isEmpty()) {
            System.out.println("Transactions on " + date + ":");
            transactions.forEach(System.out::println);
        } else {
            System.out.println("No transactions found on this date.");
        }
    }

    public void viewTransactionsByMonth(Scanner scanner, int loggedInUserID) {
        System.out.print("Enter month (YYYY-MM): ");
        String[] parts = scanner.nextLine().split("-");
        LocalDate date = LocalDate.of(Integer.parseInt(parts[0]), Integer.parseInt(parts[1]), 1);

        var transactions = transactionImpl.getTransactionsByMonth(date).stream()
            .filter(c -> c.getUserID() == loggedInUserID)
            .collect(Collectors.toList());
        if (!transactions.isEmpty()) {
            System.out.println("Transactions in " + date.getMonth() + ":");
        }
    }

```



```

        transactions.forEach(System.out::println);
    } else {
        System.out.println("No transactions found in this month.");
    }
}

public List<Transaction> getTransactionsByCategoryAndMonth(int loggedInUserID, int
categoryID) {

    List<Transaction> userTransactions =
transactionImpl.getTransactionsByCategoryAndMonth(categoryID);

    return userTransactions.stream()
        .filter(c -> c.getUserID() == loggedInUserID)

        .collect(Collectors.toList());
}
}

```

7.4 Budget Module

7.4.1 Budget.java - Model file

```

package com.tracking.models;

public class Budget {
    private int budgetID;
    private int userID;
    private int categoryID;
    private double amount;
    private String description;
    private double expense;

    // Getters and Setters
    public int getBudgetID() { return budgetID; }
    public void setBudgetID(int budgetID) { this.budgetID = budgetID; }
    public int getUserID() { return userID; }
    public void setUserID(int userID) { this.userID = userID; }
    public int getCategoryID() { return categoryID; }
    public void setCategoryID(int categoryID) { this.categoryID = categoryID; }
}

```

```

    public double getAmount() { return amount; }
    public void setAmount(double amount) { this.amount = amount; }
    public String getDescription() { return description; }
    public void setDescription(String description) { this.description = description; }
    public double getExpense() { return expense; }
    public void setExpense(double expense) { this.expense = expense; }

    @Override
    public String toString() {
        return "Budget{" +
            "id=" + budgetID +
            ", userID=" + userID +
            ", categoryID=" + categoryID + "\" +
            ", amount=" + amount +
            ", description=" + description + "\" +
            ", expense=" + expense +
            "'";
    }
}

```

7.4.2 BudgetRepository.java - Repository file

```

package com.tracking.repository;

import com.tracking.models.Budget;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class BudgetRepository {
    private Map<Integer, Budget> budgetMap = new HashMap<>();
    private static int budgetIDCounter = 1;

    // CRUD Operations
    public void addBudget(Budget budget) {
        budget.setBudgetID(budgetIDCounter++);
        budgetMap.put(budget.getBudgetID(), budget);
    }
}

```

```

public Budget getBudget(int budgetID) {
    return budgetMap.get(budgetID);
}

public void updateBudget(Budget budget) {
    budgetMap.put(budget.getBudgetID(), budget);
}

public void deleteBudget(int budgetID) {
    budgetMap.remove(budgetID);
}

public List<Budget> getBudgets() {
    List<Budget> budgets = new ArrayList<>();
    for (Budget budget : budgetMap.values()) {
        budgets.add(budget);
    }
    return budgets;
}

public void resetExpenses() {
    for (Budget budget : budgetMap.values()) {
        budget.setExpense(0.0);
    }
}

public void updateBudgetExpense(int categoryID, double amount, int loggedInUserID) {
    for (Budget budget : budgetMap.values()) {
        if (budget.getCategoryID() == categoryID && budget.getUserID() == loggedInUserID) {
            double newExpense = budget.getExpense() + amount;
            budget.setExpense(newExpense);
            updateBudget(budget);
        }
    }
}
}

```

7.4.3 BudgetIntf.java - Interface file

```

package com.tracking.intf;

import java.util.List;

```

```

import com.tracking.models.Budget;

public interface BudgetIntf {
    void addBudget(Budget budget);
    Budget getBudget(int budgetID);
    void updateBudget(Budget budget);
    void deleteBudget(int budgetID);
    List<Budget> getBudgets();
    void resetExpenses();
    void updateExpense(int categoryID, double amount, int loggedInUserID);
}

```

7.4.4 BudgetImpl.java - Implementation file

```

package com.tracking.impl;

import java.util.List;

import com.tracking.intf.BudgetIntf;
import com.tracking.models.Budget;
import com.tracking.repository.BudgetRepository;

public class BudgetImpl implements BudgetIntf {
    private BudgetRepository budgetRepository;

    public BudgetImpl(BudgetRepository budgetRepository) {
        this.budgetRepository = budgetRepository;
    }

    @Override
    public void addBudget(Budget budget) { budgetRepository.addBudget(budget); }

    @Override
    public Budget getBudget(int budgetID) { return budgetRepository.getBudget(budgetID); }

    @Override
    public void updateBudget(Budget budget) { budgetRepository.updateBudget(budget); }

    @Override

```

```

    public void deleteBudget(int budgetID) { budgetRepository.deleteBudget(budgetID); }

    @Override
    public List<Budget> getBudgets() {
        return budgetRepository.getBudgets();
    }

    @Override
    public void resetExpenses() { budgetRepository.resetExpenses(); }

    @Override
    public void updateExpense(int categoryID, double amount, int loggedInUserID) {
        budgetRepository.updateBudgetExpense
            (categoryID, amount, loggedInUserID); }

}

```

7.4.5 BudgetService.java - Service file

```

package com.tracking.service;

import com.tracking.impl.BudgetImpl;
import com.tracking.impl.CategoryImpl;
import com.tracking.models.Budget;
import com.tracking.models.Transaction;

import java.util.List;
import java.util.Scanner;
import java.util.stream.Collectors;
import java.time.LocalDate;

public class BudgetService {
    private BudgetImpl budgetImpl;
    private CategoryImpl categoryImpl;
}

```

```

private TransactionService transactionService;

public BudgetService(BudgetImpl budgetImpl, CategoryImpl categoryImpl,
TransactionService transactionService) {
    this.budgetImpl = budgetImpl;
    this.categoryImpl = categoryImpl;
    this.transactionService = transactionService;
}

public void addBudget(Scanner scanner, int loggedInUserID) {
    Budget budget = new Budget();
    budget.setUserID(loggedInUserID);

    System.out.println("Select a category:");
    categoryImpl.getAllCategories().values().stream()
        .filter(c -> c.getUserID() == loggedInUserID)
        .forEach(c ->
            System.out.println(c.getCategoryID() + ": " + c.getCategoryName()));

    System.out.print("Enter category ID: ");
    budget.setCategoryID(scanner.nextInt());
    scanner.nextLine();

    List<Transaction> transactions =
transactionService.getTransactionsByCategoryAndMonth(loggedInUserID,
budget.getCategoryID());

    double initialExpense = transactions.stream()
        .mapToDouble(Transaction::getAmount)
        .sum();

```

```
budget.setExpense(initialExpense);
```

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

```
budget.setAmount(scanner.nextDouble());
```

```
scanner.nextLine();
```

```
System.out.print("Enter budget description: ");
```

```
budget.setDescription(scanner.nextLine());
```

```
budgetImpl.addBudget(budget);
```

```
System.out.println("Budget added successfully.");
```

```
}
```

```
public void updateBudget(Scanner scanner, int loggedInUserID) {
```

```
    System.out.print("Enter budget ID to update: ");
```

```
    int budgetId = scanner.nextInt();
```

```
    scanner.nextLine();
```

```
    Budget budget = budgetImpl.getBudget(budgetId);
```

```
    if (budget != null && budget.getUserID() == loggedInUserID) {
```

```
        System.out.print("Enter new amount (leave blank to keep current): ");
```

```
        String amountStr = scanner.nextLine();
```

```
        if (!amountStr.isEmpty()) {
```

```
            budget.setAmount(Double.parseDouble(amountStr));
```

```
        }
```

```
        System.out.print("Enter new description (leave blank to keep current): ");
```

```
        String description = scanner.nextLine();
```

```
        if (!description.isEmpty()) {
```

```
            budget.setDescription(description);
```

```
        }
```

```

        budgetImpl.updateBudget(budget);
        System.out.println("Budget updated successfully.");
    } else {
        System.out.println("Budget not found or you don't have permission to update it.");
    }
}

```

```

public void deleteBudget(Scanner scanner, int loggedInUserID) {
    System.out.print("Enter budget ID to delete: ");
    int budgetId = scanner.nextInt();
    scanner.nextLine();
    Budget budget = budgetImpl.getBudget(budgetId);
    if (budget != null && budget.getUserID() == loggedInUserID) {
        budgetImpl.deleteBudget(budgetId);
        System.out.println("Budget deleted successfully.");
    } else {
        System.out.println("Budget not found or you don't have permission to delete it.");
    }
}

```

```

public List<Budget> viewBudgets(int loggedInUserID) {
    List<Budget> budgets = budgetImpl.getBudgets();
    System.out.println("Budgets for User ID " + loggedInUserID + ":");

    budgets.stream()
        .filter(c -> c.getUserID() == loggedInUserID)
        .collect(Collectors.toList());
    if (!budgets.isEmpty()) {
        budgets.forEach(System.out::println);
    }
}

```



```

    } else {
        System.out.println("No budgets found.");
    }

    return budgets;
}

public void resetExpenses() {
    List<Budget> budgets = budgetImpl.getBudgets();
    LocalDate today = LocalDate.now();
    if (today.getDayOfMonth() == 1) {
        for (Budget budget : budgets) {
            budget.setExpense(0.0);
            budgetImpl.updateBudget(budget);
        }
    }
}

public void updateExpense(int categoryID, double amount, int loggedInUserID) {
    List<Budget> budgets = budgetImpl.getBudgets();
    for (Budget budget : budgets) {
        if (budget.getCategoryID() == categoryID && budget.getUserID() == loggedInUserID) {
            double newExpense = budget.getExpense() + amount;
            budget.setExpense(newExpense);
            budgetImpl.updateBudget(budget);
        }
    }
}
}

```

7.5 ExpenseTrackerApplication.java - Main file

```
package com.tracking;

import com.tracking.impl.UserImpl;
import com.tracking.impl.CategoryImpl;
import com.tracking.impl.TransactionImpl;
import com.tracking.impl.BudgetImpl;
import com.tracking.repository.UserRepository;
import com.tracking.repository.CategoryRepository;
import com.tracking.repository.TransactionRepository;
import com.tracking.repository.BudgetRepository;
import com.tracking.models.User;
import com.tracking.service.TransactionService;
import com.tracking.service.UserService;
import com.tracking.service.CategoryService;
import com.tracking.service.BudgetService;

import java.util.Scanner;

public class ExpenseTrackerApplication {

    public static void main(String[] args) {

        // Repositories
        UserRepository userRepository = new UserRepository();
        CategoryRepository categoryRepository = new CategoryRepository();
        TransactionRepository transactionRepository = new TransactionRepository();
        BudgetRepository budgetRepository = new BudgetRepository();

        // Implementations
        UserImpl userImpl = new UserImpl(userRepository);
        CategoryImpl categoryImpl = new CategoryImpl(categoryRepository);
        TransactionImpl transactionImpl = new TransactionImpl(transactionRepository);
        BudgetImpl budgetImpl = new BudgetImpl(budgetRepository);

        // Services
        UserService userService = new UserService(userImpl);
        CategoryService categoryService = new CategoryService(categoryImpl);
```

```

    TransactionService transactionService = new TransactionService(transactionImpl,
categoryImpl, budgetImpl);
    BudgetService budgetService = new BudgetService(budgetImpl, categoryImpl,
transactionService);
    budgetService.resetExpenses();

    Scanner scanner = new Scanner(System.in);

    while (true) {
        if (!SessionManager.isLoggedIn()) {
            // Handle user login
            System.out.println("Login:");
            System.out.print("Email: ");
            String email = scanner.nextLine();
            System.out.print("Password: ");
            String password = scanner.nextLine();

            User loggedInUser = userImpl.authenticate(email, password);
            if (loggedInUser != null) {
                SessionManager.login(loggedInUser);
                System.out.println("Login successful! Welcome " + loggedInUser.getUserName());
            } else {
                System.out.println("Invalid credentials!");
                continue;
            }
        }
    }

    // Main menu
    System.out.println("\nMain Menu:");
    System.out.println("1. Manage User");
    System.out.println("2. Manage Transactions");
    System.out.println("3. Manage Categories");
    System.out.println("4. Manage Budgets");
    System.out.println("5. Logout");
    System.out.print("Choose an option (1-5): ");

    int choice = scanner.nextInt();
    scanner.nextLine();

    switch (choice) {

```

```

        case 1:
            manageUsers(scanner, userService);
            break;
        case 2:
            manageTransactions(scanner, transactionService,
SessionManager.getCurrentUser().getUserID());
            break;
        case 3:
            manageCategories(scanner, categoryService,
SessionManager.getCurrentUser().getUserID());
            break;
        case 4:
            manageBudgets(scanner, budgetService,
SessionManager.getCurrentUser().getUserID());
            break;
        case 5:
            SessionManager.logout();
            System.out.println("Logged out successfully.");
            break;
        default:
            System.out.println("Invalid choice, please select a valid option.");
    }
}
}
}

```

```

private static void manageUsers(Scanner scanner, UserService userService) {
    while (true) {
        System.out.println("\nUser Management:");
        System.out.println("1. View User Details");
        System.out.println("2. Update User Details");
        System.out.println("3. Back to Main Menu");
        System.out.print("Choose an option (1-3): ");

        int choice = scanner.nextInt();
        scanner.nextLine();

        switch (choice) {
            case 1:
                userService.viewUserDetails();
                break;

```

```

        case 2:
            userService.updateUserDetails(scanner);
            break;
        case 3:
            return;
        default:
            System.out.println("Invalid choice, please select a valid option.");
    }
}
}

```

```

private static void manageTransactions(Scanner scanner, TransactionService
transactionService, int loggedInUserID) {
    boolean transactionRunning = true;
    while (transactionRunning) {
        System.out.println("\n--- Transaction Menu ---");
        System.out.println("1. Add Transaction");
        System.out.println("2. Update Transaction");
        System.out.println("3. Delete Transaction");
        System.out.println("4. View Transactions by Date");
        System.out.println("5. View Transactions by Month");
        System.out.println("6. Return to Main Menu");
        System.out.print("Select an option: ");
        int choice = scanner.nextInt();
        scanner.nextLine();

        switch (choice) {
            case 1:
                transactionService.addTransactionWithCategory(scanner, loggedInUserID);
                break;
            case 2:
                transactionService.updateTransaction(scanner, loggedInUserID);
                break;
            case 3:
                transactionService.deleteTransaction(scanner, loggedInUserID);
                break;
            case 4:
                transactionService.viewTransactionsByDate(scanner, loggedInUserID);

```

```

        break;
    case 5:
        transactionService.viewTransactionsByMonth(scanner, loggedInUserID);
        break;
    case 6:
        transactionRunning = false;
        break;
    default:
        System.out.println("Invalid option. Please try again.");
    }
}
}
}

```

```

private static void manageCategories(Scanner scanner, CategoryService categoryService, int
loggedInUserID) {

```

```

    boolean categoryRunning = true;
    while (categoryRunning) {
        System.out.println("\n--- Category Menu ---");
        System.out.println("1. View Categories");
        System.out.println("2. Add Category");
        System.out.println("3. Update Category");
        System.out.println("4. Delete Category");
        System.out.println("5. Return to Main Menu");
        System.out.print("Select an option: ");
        int choice = scanner.nextInt();
        scanner.nextLine();

        switch (choice) {
            case 1:
                categoryService.viewCategories(loggedInUserID);
                break;
            case 2:
                categoryService.addCategory(scanner, loggedInUserID);
                break;
            case 3:
                categoryService.updateCategory(scanner, loggedInUserID);
                break;
            case 4:
                categoryService.deleteCategory(scanner, loggedInUserID);
                break;

```

```

        case 5:
            categoryRunning = false;
            break;
        default:
            System.out.println("Invalid option. Please try again.");
    }
}
}

```

```

private static void manageBudgets(Scanner scanner, BudgetService budgetService, int
loggedInUserID) {
    while (true) {
        System.out.println("\nBudget Management:");
        System.out.println("1. View Budgets");
        System.out.println("2. Add Budget");
        System.out.println("3. Update Budget");
        System.out.println("4. Delete Budget");
        System.out.println("5. Return to Main Menu");
        System.out.print("Choose an option (1-5): ");

        int choice = scanner.nextInt();
        scanner.nextLine();

        switch (choice) {
            case 1:
                budgetService.viewBudgets(loggedInUserID);
                break;
            case 2:
                budgetService.addBudget(scanner, loggedInUserID);
                break;
            case 3:
                budgetService.updateBudget(scanner, loggedInUserID);
            case 4:
                budgetService.deleteBudget(scanner, loggedInUserID);
            case 5:
                return;

            default:
                System.out.println("Invalid choice, please select a valid option.");
        }
    }
}

```

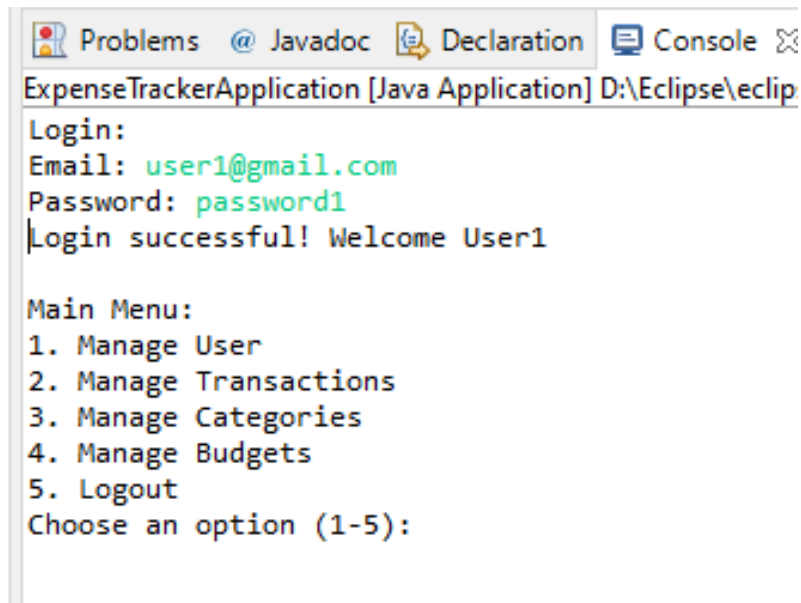
```
    }  
  }  
}
```

7.6 SessionManager.java - Session file

```
package com.tracking;  
  
import com.tracking.models.User;  
  
public class SessionManager {  
    private static User currentUser;  
  
    public static void login(User user) {  
        currentUser = user;  
    }  
  
    public static void logout() {  
        currentUser = null;  
    }  
  
    public static User getCurrentUser() {  
        return currentUser;  
    }  
  
    public static boolean isLoggedIn() {  
        return currentUser != null;  
    }  
  
}
```


8.Output

8.1 Login



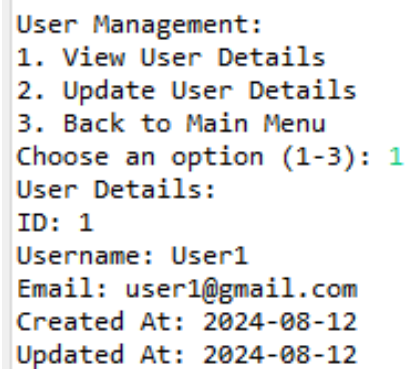
The screenshot shows the Eclipse IDE's console window. The title bar includes tabs for 'Problems', 'Javadoc', 'Declaration', and 'Console'. The console output for 'ExpenseTrackerApplication [Java Application] D:\Eclipse\eclip' is as follows:

```
Login:
Email: user1@gmail.com
Password: password1
Login successful! Welcome User1

Main Menu:
1. Manage User
2. Manage Transactions
3. Manage Categories
4. Manage Budgets
5. Logout
Choose an option (1-5):
```

8.2 User Module

8.2.1 View user



The screenshot shows the Eclipse IDE's console window with the following output for the 'User Management' module:

```
User Management:
1. View User Details
2. Update User Details
3. Back to Main Menu
Choose an option (1-3): 1
User Details:
ID: 1
Username: User1
Email: user1@gmail.com
Created At: 2024-08-12
Updated At: 2024-08-12
```

8.2.2 Update User

```
User Management:
1. View User Details
2. Update User Details
3. Back to Main Menu
Choose an option (1-3): 2
Update User Details:
New Username: sownthari
New Email: sownthariponnusamy@gmail.com
New Password: sownthari
User details updated successfully.

User Management:
1. View User Details
2. Update User Details
3. Back to Main Menu
Choose an option (1-3): 1
User Details:
ID: 1
Username: sownthari
Email: sownthariponnusamy@gmail.com
Created At: 2024-08-12
Updated At: 2024-08-12
```

8.3 Category Module

8.3.1 View Category

```
--- Category Menu ---
1. View Categories
2. Add Category
3. Update Category
4. Delete Category
5. Return to Main Menu
Select an option: 1
Your Categories:
Category{id=1, name='Groceries', description='Spending on food and everyday essentials', userID=1}
```

8.3.2 Add Category

```
--- Category Menu ---
1. View Categories
2. Add Category
3. Update Category
4. Delete Category
5. Return to Main Menu
Select an option: 2
Enter category name: Travel
Enter category description: travel related expenses
Category added successfully.
```

8.3.3 Edit Category

```
--- Category Menu ---
1. View Categories
2. Add Category
3. Update Category
4. Delete Category
5. Return to Main Menu
Select an option: 3
Enter category ID to update: 1
Enter new category name (leave blank to keep current): spending on fruits and other essentials
Enter new description (leave blank to keep current):
Category updated successfully.
```

8.3.4 Delete Category

```
--- Category Menu ---
1. View Categories
2. Add Category
3. Update Category
4. Delete Category
5. Return to Main Menu
Select an option: 4
Enter category ID to delete: 1
Category deleted successfully.
```

8.4 Transaction Module

8.4.1 Add Transaction

```
--- Transaction Menu ---
1. Add Transaction
2. Update Transaction
3. Delete Transaction
4. View Transactions by Date
5. View Transactions by Month
6. Return to Main Menu
Select an option: 1
Select a category:
11: Travel
Enter category ID: 11
Enter amount: 200
Enter description: went to office
Enter type (Income/Expense): Expense
Enter date (YYYY-MM-DD): 2024-08-13
Transaction added successfully.
```

8.4.2 Update Transaction

```
--- Transaction Menu ---
1. Add Transaction
2. Update Transaction
3. Delete Transaction
4. View Transactions by Date
5. View Transactions by Month
6. Return to Main Menu
Select an option: 2
Enter transaction ID to update: 1
Enter new amount (leave blank to keep current): 300
Enter new description (leave blank to keep current):
Enter new type (Income/Expense, leave blank to keep current):
Enter new date (YYYY-MM-DD, leave blank to keep current):
Transaction updated successfully.
```

8.4.3. Delete Transaction

```
--- Transaction Menu ---
1. Add Transaction
2. Update Transaction
3. Delete Transaction
4. View Transactions by Date
5. View Transactions by Month
6. Return to Main Menu
Select an option: 3
Enter transaction ID to delete: 1
Transaction deleted successfully.
```

8.4.4 View Transactions by Date

```
--- Transaction Menu ---
1. Add Transaction
2. Update Transaction
3. Delete Transaction
4. View Transactions by Date
5. View Transactions by Month
6. Return to Main Menu
Select an option: 4
Enter date (YYYY-MM-DD): 2024-08-13
2024-08-13
Transactions on 2024-08-13:
Transaction{id=1, userID=1, categoryID=11, amount=300.0, transactionDate=
```

8.4.5 View Transactions by Month

```
--- Transaction Menu ---
1. Add Transaction
2. Update Transaction
3. Delete Transaction
4. View Transactions by Date
5. View Transactions by Month
6. Return to Main Menu
Select an option: 5
Enter month (YYYY-MM): 2024-08
Transactions in AUGUST:
Transaction{id=1, userID=1, categoryID=11, amount=300.0, transactionDate
```

8.5 Budget Module

8.5.1 View Budgets

```
Budget Management:
1. View Budgets
2. Add Budget
3. Update Budget
4. Delete Budget
5. Return to Main Menu
Choose an option (1-5): 1
Budgets for User ID 1:
Budget{id=1, userID=1, categoryID='11', amount=1000.0, description='amount li
```

8.5.2 Add Budget

```
Budget Management:
1. View Budgets
2. Add Budget
3. Update Budget
4. Delete Budget
5. Return to Main Menu
Choose an option (1-5): 2
Select a category:
11: Travel
Enter category ID: 11
Enter budget amount: 1000
Enter budget description: amount limit for travel
Budget added successfully.
```

8.5.3 Update & Delete Budget

```
Budget Management:
1. View Budgets
2. Add Budget
3. Update Budget
4. Delete Budget
5. Return to Main Menu
Choose an option (1-5): 3
Enter budget ID to update: 1
Enter new amount (leave blank to keep current): 1500
Enter new description (leave blank to keep current):
Budget updated successfully.
Enter budget ID to delete: 1
Budget deleted successfully.
```

8.6 Logout

Main Menu:

1. Manage User
2. Manage Transactions
3. Manage Categories
4. Manage Budgets
5. Logout

Choose an option (1-5): 5

Logged out successfully.

Login:

Email: