EXPENSE TRACKER SYSTEM

“Software Engineering: Theory and Practice”

Submitted By
Balaji MK 20C009
Dharsana V 20C019
N Gopal Sankar 20C023

Table of Contents

Table of Contents

❖ Introduction

➢ Purpose

To develop a software to manage a person's routine and circumstancial expenses and determine whether they are spending as per budget. The user can also cut-down all unrequired expenses.

➢ Document Conventions

Font used in writing this document is Times New Roman and the font size - 14.
All topics, sub headings and key-terms have been highlighted or *Italicized*.

➢ Intended Audience and Reading Suggestions

The document is intended to be read by project managers,developers, testers, marketing staff, users and document writers. The document is organized into five parts :
1.Introduction
2.Overall description
3.System features
4.External Interface Requirements
5.Quality attribute Requirements.
All the parts are independent however reading the document sequentially helps the reader understand the expense tracking system better.

➢ Product Scope

Project Objectives:

- ❖ To keep track of daily expenses and budgeting.
- ❖ To save money for pre-defined expenses which will help planning on our future investments.

Project Scope:

- ❖ This application can have a good market as it is usable by anyone who is willing to manage their expenses and is aiming to save for future investments.There aren't any restrictions criteria.

➢ References

https://myfik.unisza.edu.my/www/fyp/fyp19sem2/report/046376.pdf
https://nevonprojects.com/daily-expense-tracker-system/
https://www.slideshare.net/RashnaMaharjan2/daily-expense-tracker
https://youtu.be/XuFDcZABiDQ

❖ Overall Description

➢ Product Perspective

This project is developed for the indviduals who likely to manage their daily expenses in a efficient way. The intent is for these individuals to use this web-based product for filing their expenses, and also manage their daily,monthly and yearly expenses. This product will become a one-stop solution for easy and faster approval of expenses. This product should also be user-friendly, quick to learn and reliable for the above purpose. This project is intended to be a stand-alone product and should not depend on the availability of any other software. It should run on all platform and should be independent of web browser.

➢ Product Functions

Managing finances is always challenging, and for this, expense tracking plays a key role. Since every business has different financial management and expense tracking needs, it is always advisable to build an expense tracking app with custom features that suit your specific requirements.

What are the key features you need to consider for expense tracker app development? What features are must-haves for such an app, and what can be regarded as advanced ones. Let's list out all these features.

- ❖ Expense and revenue tracking
- ❖ Managing transaction receipts and records
- ❖ Paying taxes in time
- ❖ Processing payments and invoices
- ❖ Create in-depth reports

➢ User Classes and Characteristics

❖ The Expense tracker project is meant to offer a shared expenses solution that is faster, easier, and more convenient than manually calculating and handling debts. Consequently, the application will have little or no learning curve, and the user interface will be as intuitive as possible. Thus, technical expertise should not be an issue. Instead, anticipated users can be defined by how they will use the product in a particular situation. The following list

❖ 1. Long-term recurring expenses (e.g. rent, groceries, utilities)

❖ Key functions: Keep track of expenses Notify users when debts are incurred Record who has paid and who still owes

❖ Requirements: Method for inputting expenses into the web based application . Support for automated notifications Database containing current debts, past transactions, etc.

❖ 2. Short-term recurring expenses (e.g. travel costs – gas, food, hotel)

❖ Key functions: Add new expenses (quickly and easily) Record who is paying and what he/she is paying for Update member balances on the fly o

❖ Requirements: Simple and intuitive user interface for adding expenses Expense-tracking system Automated, background algorithm for calculating debts

❖ 3. Single expense (e.g., splitting a bill at dinner)

❖ o Key functions: Create a group (quickly and easily) Add non-registered individuals to the group Quickly calculate each member's balance

❖ o Requirements: Simple and intuitive user interface for adding groups Support for group members without the Expense tracker application Option to calculate all balances on the spot These groups are not meant to separate or categorize users, just the different situations in which Expense tracker is likely to be used. In fact, a user may utilize the application for all of these scenarios simultaneously. This is another defining feature of the Expense tracker system: support for multiple groups. This functionality allows a user to track expenses pertaining to several unrelated groups at the same time.

➢ Operating Environment

- ★ Microsoft Windows Server 2003 (32 and 64 bit) Standard Edition or Enterprise Edition
- ★ Microsoft Windows Server 2008 Microsoft Windows Server 2008 R2 (32 and 64 bit)
- ★ Microsoft Windows Server 2012 (32 and 64 bit)
- ★ Microsoft Windows Server 2016 R2 (32 and 64 bit)
- ★ Microsoft Windows Server 2018 (32 and 64 bit)

Processor Requirements:

- ★ X86 and X64 at 1,0 Ghz
- ★ X86 or X64 at 1,5 Ghz or more

Internal memory:

- ★ 1 GB
- ★ 2 Gb RAM or more

➢ Design and Implementation Constraints

The primary design constraint is the mobile platform. Since the application is designated for web based application, limited screen size and resolution will be a major design consideration. Creating a user interface which is both effective and easily navigable will pose a difficult challenge. Expense tracker is meant to be quick and responsive, even when dealing with large groups and transactions, so each feature must be designed and implemented with efficiency in mind.

➢ User Documentation

The primary goal of expense tracker is to facilitate the process of managing expenses. Consequently, the web based application will be designed to be as simple to use as possible. Nonetheless, users may still require some supplementary information about each component of the expense tracker. The application will contain two features that offer this: the expense tracker document and the Help menu.

➢ Assumptions and Dependencies

❖ As mentioned previously, the features of Expense tracker are divided into two groups: core features and additional features. Core features are crucial to the basic functionality of the Expense tracker application. These features must all be implemented in order for the application to be useful. Optional features, however, are not critical to the function of the application. They are usability improvements and convenience enhancements that may be added after the application has been developed. Thus, the implementation of these features is entirely dependent upon the time spent designing and implementing the core features. The final decision on whether or not to implement these features will be made during the later stages of the design phase.

❖ **EXTERNAL DEPENDENCIES** Several of the features presented in this document rely on the existence and maintained operation of several APIs. A non-exhaustive list follows.

❖ **EMAIL NOTIFICATIONS** : The Android platform is not suited for sending mass emails. Thus, the central server will be responsible for this feature of this web based application. The smartphone client will notify the server when messages need to be sent using a custom API that is to be created. This API will use standard HTTP messaging to facilitate client-server communications. The API will be implemented using PHP.

❖ **SMS NOTIFICATIONS** : This feasibility of this feature is yet to be determined. If implemented, this feature would allow offline users without an Android smartphone to receive notifications of outstanding debt and other information via text messages. A suitable and free text messaging API that can be called from the server has yet to be found. The possibility of sending text messages from the Android smartphone client itself is also being reviewed.

❖ External Interface Requirements

➢ User Interfaces

Starting from the teeny-tiny icon and ending with a full-screen image background, the user interface includes everything that can be found on the web page.Depending on the project, it may require different elements. However, there are still some good tips that help to create a web application design that rocks. Here are some of them:

- Keep it simple. Whatever grandiose idea you have in mind, leave it. When it comes to the web application sphere, things should be straightforward. Remember, all fancy ideas will “eat” your performance, to say nothing about creating possible issues with unobstructed interaction with the interface.
- Eliminate all the compatibility issues. This is a big one. Your interface should look, work, and feel the same on all devices regardless of the screen size, operating system, browser, etc.
- Make navigation flawless. As we have said earlier, keep things simple, especially with the navigation. Users should navigate between products and

functionality without a hitch. It is the heart and soul of every good successful website out there. No handy navigation – no good user experience.

- Focus on delivering messages. Avoid busy UI. The web application design should clearly explain the purpose of the product. Structure everything using such time-proven components as tables, grids, carousels, etc.
- Stick to cleaner interface with expressive fonts and supportive visual graphics. Create a beautiful look and feel. Cluttered solutions do not solve the issue. The design should be pleasant to use.

The user interface is the first layer of your presentation that potential customers deal with. Therefore, everything should be impeccable; otherwise, users will leave even without reaching their destination, aka your product.

➢ Hardware Interfaces

This application works on Android, IOS mobile devices and tablets. No other hardware is required.

➢ Software Interfaces

Since this is a web application, it will need only the minimum software requirements required to display and use an ordinary web page. It could be *Android 4.0* or newer versions. In case of windows, the minimum OS requirement required is *Windows 7* or latest versions.

➢ Communications Interfaces

- ❖ This application uses Wi-Fi Direct to communicate Android devices, Multipeer connectivity to communicate IOS devices.
- ❖ We will use the Internet to communicate between IOS and Android devices since Bluetooth and Wi-Fi Direct communication are not supported between these two different platforms.
- ❖ External communication interfaces-RS232 and RS485, USB, infrared, Bluetooth, Wi-Fi, ZigBee, GPRS, GSM.

❖ System Features

❖ System Feature 1

4.1.1 Description and Priority

1. Continuous compliance
2. Compatible with any accounting software
3. Multi-currency functionality
4. Automated workflows
5. Automatic categorization of expenses and allocation of cost center data
6. Reporting functionality

4.1.2 Stimulus/Response Sequences :

- **Stimulus: User requests to analyze a new expense.**

  Response: System creates a new expense chart and displays it.
- **Stimulus: User tries to input a new expense list to the database.**

  Response: A standard open dialog is invoked. The system opens the database at the requested location.
- **Stimulus: User requests to save an expense or reminder for future expense.**

  Response: A standard save dialog is invoked. The system saves the expense and reminder request as a file at the requested location.

- **Stimulus: The user attempts to find where the expense for the month is high.**

  Response: A standard dialogue box is invoked. The system retrieves the expense data from its repository and finds where the expense is high over a particular period of time.

- **Stimulus: The user requests to view the previous expenses.**

  Response: The system will access the data for the particular date and will selectively retrieve information from the database.

4.1.3 Functional Requirements

Feature ID	Feature Name	Description
FR - 1ETS	**Tracks Receipts**	This helps store all receipts by only clicking their images in your expenses handling app.
FR - 2ETS	**Mitigates Human Errors**	With the help of an expense handling app, you can lower as well as prevent every mistake caused because of carelessness.
FR - 3ETS	**Analytics and Insights**	This feature helps you get ready-made reports that have comprehensible visuals, charts, and graphs. These reports can be utilized for assessing and building insights.
FR - 4ETS	**Creates Reports**	Money management software allows you to run and create reports of expenses, loss and profits.You can utilize this feature and produce the budgeting and inventory reports.

FR - 5ETS	**Budget vs. Actual Expenditure**	Many expense-handling software solutions showcase the difference between expenditures and income. Undoubtedly, this helps but you must see the full picture to receive detailed insights.Thereby, you should also have total details of the budget and how much will be invested in real-time. Furthermore, this helps follow the budget and make solutions like reducing unwanted costs.
FR - 6ETS	**Prediction Analysis**	Helps to know your finances, expense behavior, and predicts your further purchases. Also, it provides unique info about how to save money. From how much to spend to where to spend your money, you will learn all the factors that can lead to fluctuations in your expenditures.
FR - 7ETS	**Recurring Bills**	An expense tracking app also tracks particular transactions and bills that occur at frequent intervals. You can enable the app for taking care of recurring costs. This will automate and enable the seamless operation of recurring costs, checks, and invoices.
FR - 8ETS	**Secure Access**	Through this app, you can give secure access to your users. Moreover, you can give some specific access to some functionalities of the expense management app to reduce mistakes.

❖ Other Nonfunctional Requirements

➢ Performance Requirements

- ❖ The best starting point for setting performance expectations is an application's legal requirements. In particular, many software-as-a-service applications have a Service Level Agreement that specifies required uptime and other performance standards.
- ❖ These are the bare minimum requirements that an application must pass.
- ❖ Companies and individual developers can benefit from setting personal standards for all of these metrics. Because the Internet is always evolving, so is the study of website performance metrics. New research about users' browsing habits will undoubtedly give us more insight into how to better cater to everyone's preferences. In the meantime, consider analyzing the metrics above and make any adjustment necessary to help improve the performance of these metrics.

1. Time to title

The amount of time between the instant a visitor requests your website and the moment your site's title shows up in their browser tab is called the time to title.Time to title is determined by the speed of delivery from your origin server to the user's browser

2. Time to start render

The time elapsed between a user's request and the moment when content appears in their browser is called time to start render. This is also a very important metric to analyze as the sooner a visitor sees content appear, the more likely they will be to stay for the rest of the page to load.

3. Time to interact

The time between a request and the moment when a user can click on links, type in text fields or scroll the page is called time to interact. Some elements such as scripts and trackers may continue to load during this period.

4. DNS lookup time

The amount of time it takes for your DNS provider to translate a domain name into an IP address. Services such as Pingdom or WebPageTest can quickly calculate your website's DNS lookup times for each domain it must lookup.

5. Connection time

The time between a request and when a connection is established between the user's browser and your origin server is called the connection time.

Identifying challenges to connection time can be difficult because it depends on many factors. Too much server traffic, whether it be from users or bots, can cause connection times to spike. Users in different geographic regions are likely to experience longer connection times.

To ensure better connection time, you may need to upgrade your infrastructure. Alternatively, you could offload some assets onto a CDN or caching server.

6. Time to first byte

The time it takes for the very first byte of information to reach a user's browser after a connection to the server has been established is called time to first byte or TTFB. The order in which users receive information is important, and some slight alterations in your code can boost this website performance metric. This way, users will receive your content right away while waiting for slower personalized content to load. Rigorous metric monitoring and load testing can help developers identify time to first byte issues.

7. Time to last byte

When the user's browser finally receives each and every byte of your website, the last byte time is recorded. The quality of your code and database queries play a big role in this metric. Other factors that may be affecting your TTLB include a misconfigured web server, or if the origin server has reached its capacity.

8. Overall weight

The total number of bytes the user receives is referred to as the overall weight of your website. More important than the overall weight is the relationship between each asset since one heavy asset can slow down everything else. Furthermore, the growth of web page size has continued to expand year after year, therefore it's important to sometimes step back and see which assets are truly necessary.

By separating individual metrics such as JavaScript weight, CSS weight, image weight and total asset weight, you can pick out which categories are too heavy, and then you can perform a waterfall analysis to identify the asset that needs to be altered or removed.

9. Overall asset count

The number of assets you have obviously affects your website's overall weight, but it's important to differentiate between asset count and weight. Every asset, no matter how small or compressed, has the potential to add more time to the loading process. Your overall asset count includes your total JavaScript, CSS and image counts.

See how we improved WordPress performance and went from a 532 ms load time to 167 ms by reducing the amount of HTTP requests made.

10. Top pages

You can find out which of your pages draws the most traffic by simply checking under the Behavior section of Google Analytics. Knowing where users focus their attention can give you an idea of which content is helping you retain an audience. Keep in mind that the number of views a page receives isn't the only measure of its relevance; the number of shares a page receives via social media is also important.

➢ Safety Requirements

The basic web application requirements are:

1. **Secure the web environment (prevent web server bugs)**

A web application is only as secure as the environment it operates in. This means that the security of all the application's dependencies must be ensured. Common dependencies of web applications include:

- the web application framework it is based on
- the web server and modules used by it
- the underlying operating systems
- network components on the way between the user and the application
- the storage layer used by the application
- middleware systems

In addition, other factors such as systems running near the application (management hosts or other web servers, for example) or other web applications that are hosted on the same server, may influence and affect the security of the application provided to Google. The security program and operating procedures used around an application may also have a big impact on its security.

2. **Validate user input (prevent XSS and injection attacks)**

Anything that is transmitted from the browser to the application can potentially be manipulated by a malicious actor. As such the application should always assume that any user input is, in fact, malicious. It is a common misconception that input received from cookies, hidden form fields or drop down boxes cannot be changed by an attacker.

Everything in an HTTP request can be modified, thus stringent checks of all input are required (see the section on serialization issues for further issues to consider).

Furthermore anything an application obtains from outside its trust boundary may also be malicious. For example, information written to a database by a different application may not have gone through the same stringent sanitization routines that the application itself applies. Therefore it is necessary to validate all input an application receives from any system outside its trust boundary.

The following section describes some common vulnerabilities that are caused by insufficient input validation. This list should by no means be deemed exhaustive. Developers must ensure that input cannot change the way data or code is interpreted, regardless of context.

3. Avoid third-party scripts and CSS

Loading content from other sites is dangerous under certain circumstances since a security issue in a third-party site might also affect your application.

To avoid this problem it is not permissible to load scripts or style sheets (e.g. via <link rel="stylesheet" href=…> or <script src=…>) from any third-party site that is not Google owned and operated. By inserting malicious code into the script or style sheets, whoever has control over the servers from where those resources are loaded will also have full control over your site.

When the use of third-party libraries is unavoidable, these resources must be sourced locally and not loaded from an any third-party site that is not Google owned and operated. Care should also be taken to ensure that the third-party libraries are :

the latest stable version available

actively supported (not deprecated or relying on deprecated functionality that may cause issues with ongoing support)

In instances where the application requires to communicate with another service or API to retrieve/send data, these communications must be completed over an HTTPS connection (including suitable validation of the SSL certificate). The use of unencrypted HTTP communications or unvalidated HTTPS connections (e.g. with services delivering self-signed or untrusted certificates), even for public data, is not permissible.

4. Use encryption protect data, prevent mixed content bugs

For an attacker it is often extremely easy to listen in on the packets as they are transmitted between a user and the web application (for example when the user is on a public WiFi network). In order to avoid having sensitive data read by an attacker while it is in transit, any application that allows users to log in, or contains anything but public data, must be available solely over HTTPS. Applications developed specifically for Google must **always** be SSL-only (including any inter-server communications that occur on the backend). A web server listening on port 80 (plain HTTP) and redirecting users to the SSL version of the application is fine, and can make it easier for users to access the application.

5. Use the right authentication

In many cases, the application or the data in the application should not be public. In order to control access many applications ask the user to log in. The requirements in this section apply to all applications that require users to prove their identity to the application.

OAuth2 Login Implementation Requirements

When supporting authentication through OAuth2 Login, the application must make sure that Google is the only IdentityProvider (IdP) that is authoritative for @google.com, @gmail.com and @googlemail.com email addresses.

The only scopes that should be requested from the user are the ones necessary to uniquely identify them.

Typically, those are:https://www.googleapis.com/auth/userinfo.email

Cross Site Request Forgery (XSRF or CSRF)

Cross Site Script Inclusion (XSSI), Clickjacking

6. Content Security Policy

If you develop software specifically for Google implementation of Content Security Policy is **required**. In all other cases it is highly recommended.

CSP is a defense-in-depth mechanism web applications can use to mitigate a broad class of content injection vulnerabilities, such as cross-site scripting (XSS). This goal is achieved through a declarative policy that lets the authors of a web application inform the client about the sources from which the application expects to load resources. Almost all major browsers support some version of CSP.

➢ Security Requirements

The expense management software holds a lot of valuable information. It is crucial to make sure that this information does not fall into the wrong hands. Dealing with a data breach can be a long and stressful ordeal.So to prevent this we need requirements like

- Encrypt the sensitive data
- Invest in a cybersecurity insurance
- Make use of a robust anti-malware software

Software Quality Attributes

1) Usability

It is described as how the user is utilizing a system effectively and the ease of which users can learn to operate or control the system. The well-known principle of usability is KISS (Keep It Simple Stupid). Software applications should be user-friendly.

2) Reliability
It is the ability of a system to continue to keep operating over time

3) Availability
It is the ratio of the available system time to the total working time it is required or expected to function.
4) Portability
It is the ability of a software application to run on numerous platforms such as data portability, hosting, viewing, etc.,

5) Testability

It shows how well the system or component facilitates to perform tests to determine whether the predefined test criteria have been met.

6) Scalability

It is the ability of a system to handle the demand for stress caused by increased usage without decreasing performance.

7) Flexibility

It is the ability of a system to adapt to future changes

8) Reusability

It is the use of existing software. We use more than one software with small or no change. It is a cost-efficient and time-saving quality attribute.

9) Maintainability

It is the ability of a software application to maintain easily and support changes cost-effectively.

10) Supportability

It is the ability of a system that satisfies necessary requirements and needs to identifying and solving problems.

11) Interoperability

It is the ability of two or more systems to communicate or exchange data easily and to use the data that has been exchanged.

12) Performance

It is the ability of a system in the form of responsiveness to various actions within a certain period of time

➢ Business Rules

The system must have at least a Super-User role and a User role defined for accessing and interacting with the system. Additional roles may be defined for the system as long as the business rules for the administrator and user roles are satisfied.

At a minimum, the User role must account for the data explorer account-type requirements .
The following business rules must apply to the administrator and user roles.

Super-User Role Business Rules Super

UserRole/001 Maintains all VDS and QDR back-end system configurations
SuperUserRole/002 Maintains all VDS and QDR schemas and templates
SuperUserRole/003 Maintains all user groups and user accounts
SuperUserRole/004 Maintains all QDR predefined queries

User Role Business Rules
UserRole/001 Connects to and retrieves and reads data from VDS and QDR in conformance with the maintained VDS and QDR schemas and templates

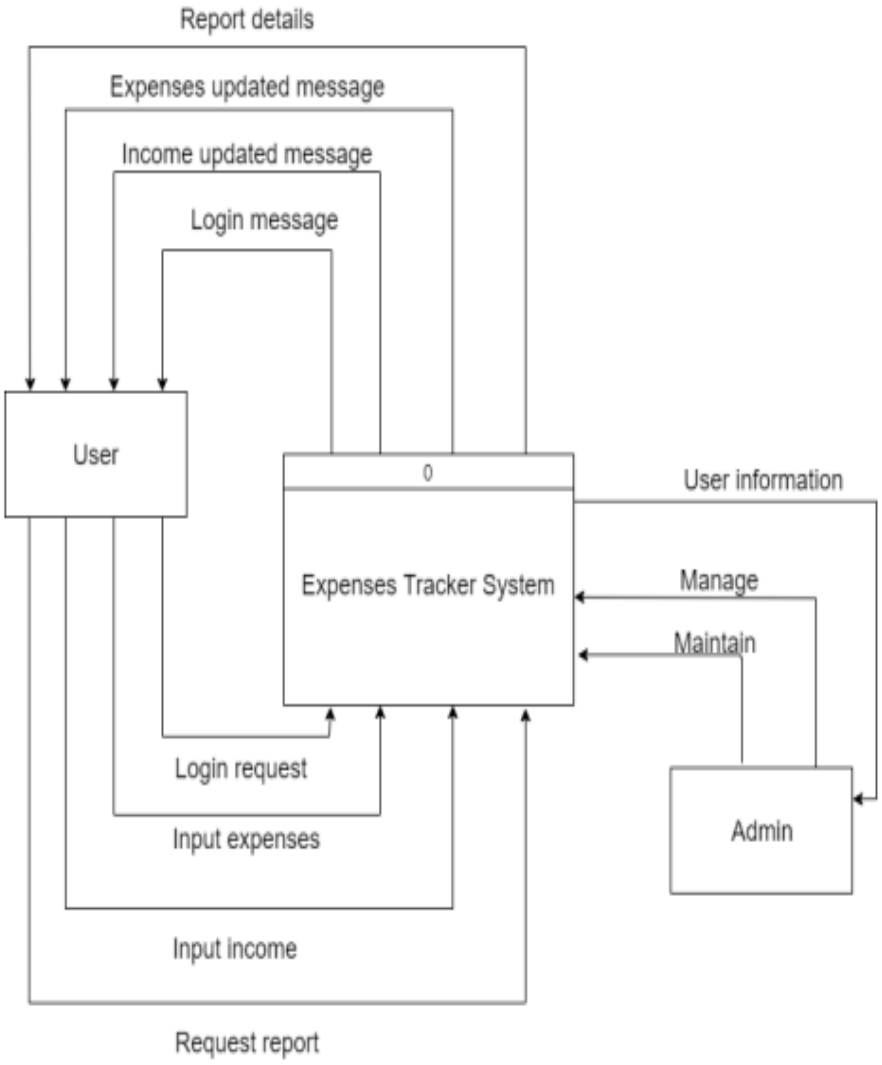
❖ Other Requirements

A database for Expense tracker calls for a server side implementation that holds information for the users, groups, bills, transactions, as well as all the relationships involved. The database will be using MySQL.
The following provides an example of information that may be stored in the database:
• Users: ID, Name, phone number, email,Username
• Groups: ID, Name, Members
• Bills: ID, Name, Owner, Group, Cost, Receipt, Date, Time
• Transactions: ID, Members Involved, Group, Date, Time, Amount
The server will be configured on a windows platform, and through use of PHP will allow interaction and processing in conjunction with the database. Processes to be done on the server include: pushing/pulling data, updating data, and generating notifications.

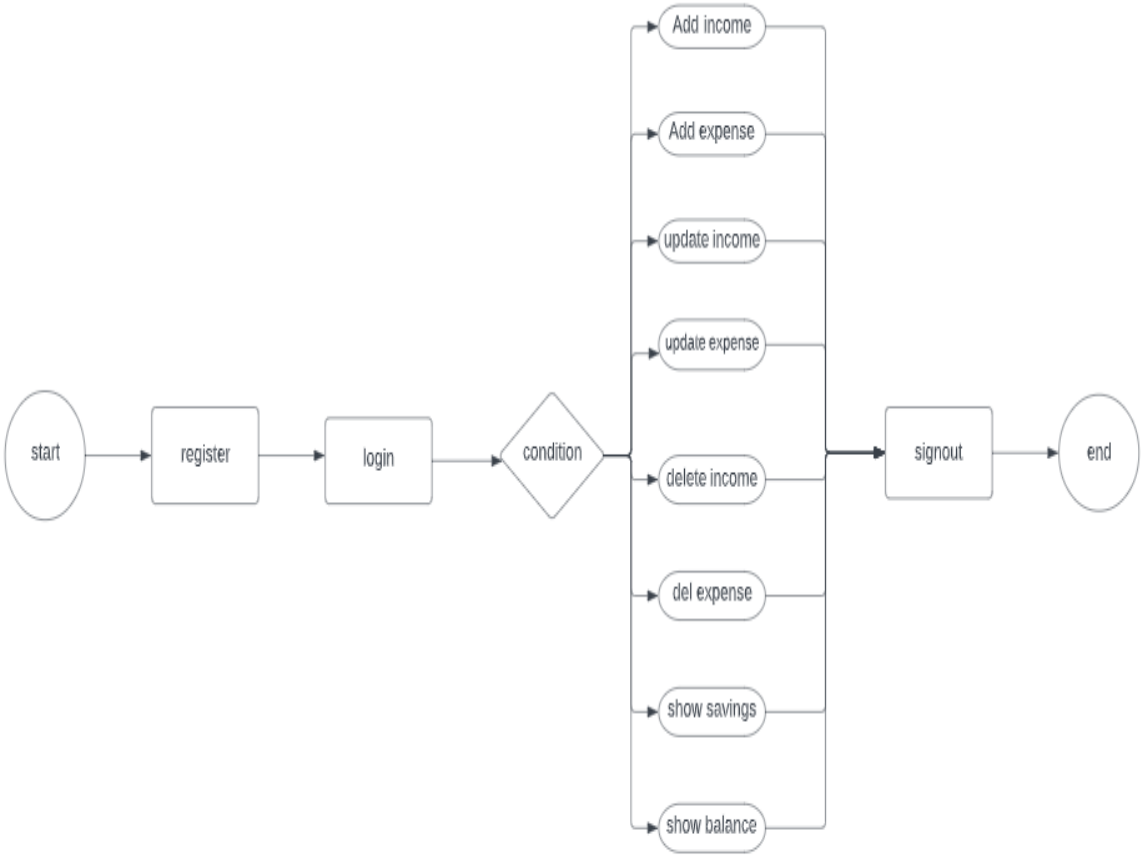
Appendix A: Glossary

API Application programming interface	A system of tools and resources (e.g., subroutine definitions, protocols) in an operating system that enables developers to create software applications.
XML Extensible Markup Language	A markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable in accordance with the World Wide Web Consortium (W3C) specification.
MTTF Mean Time to Failures	The predicted average operation time for a system replaced after a failure
QIF Quality Information Framework	A unified XML framework and ASNI-accredited standard for enabling interoperability between computer-aided quality measurement systems.
SSL Secure Sockets Layer	An early standard security technology, later replaced by Transport Layer Security (TLS), for establishing an encrypted link between a web server and a browser.
TLS Transport Layer Security	A cryptographic protocol that provides communications security over a computer network in accordance with RFC 6176.
Administrator	A person whose job is to organize or manage a system.

CONTEXT DIAGRAM :

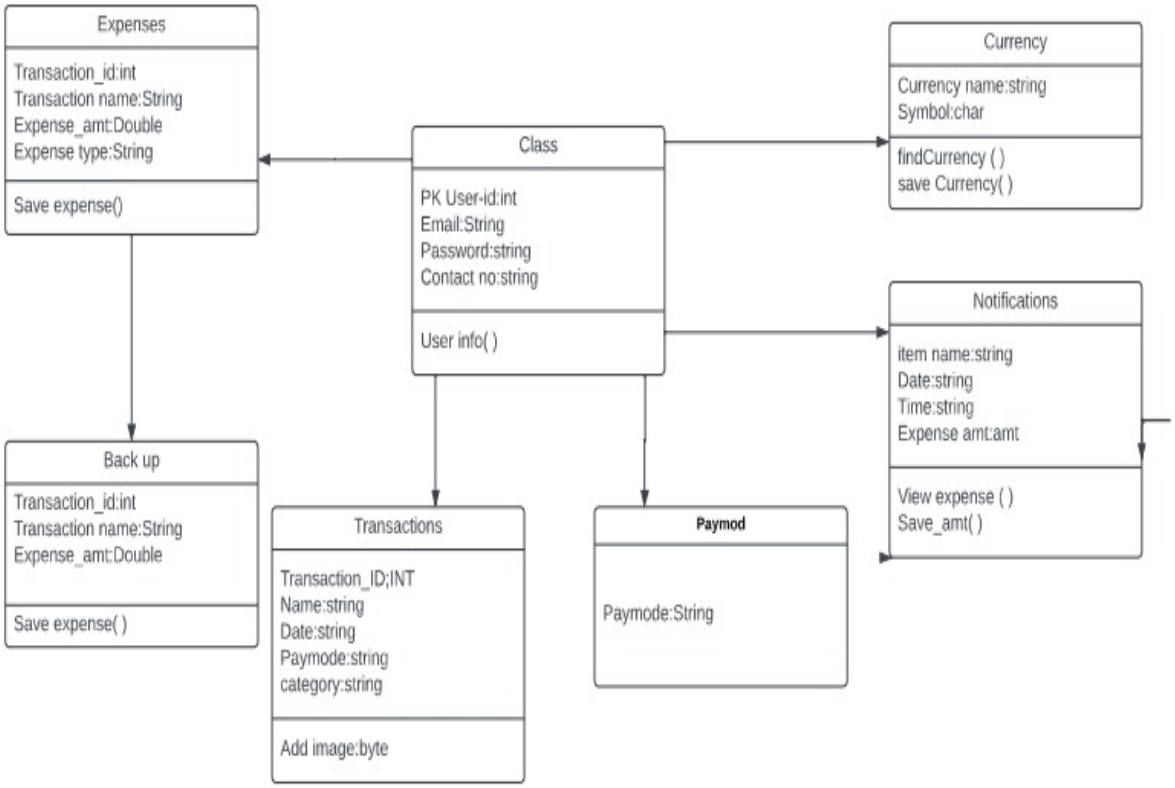


SOFTWARE ARCHITECTURE DIAGRAM :



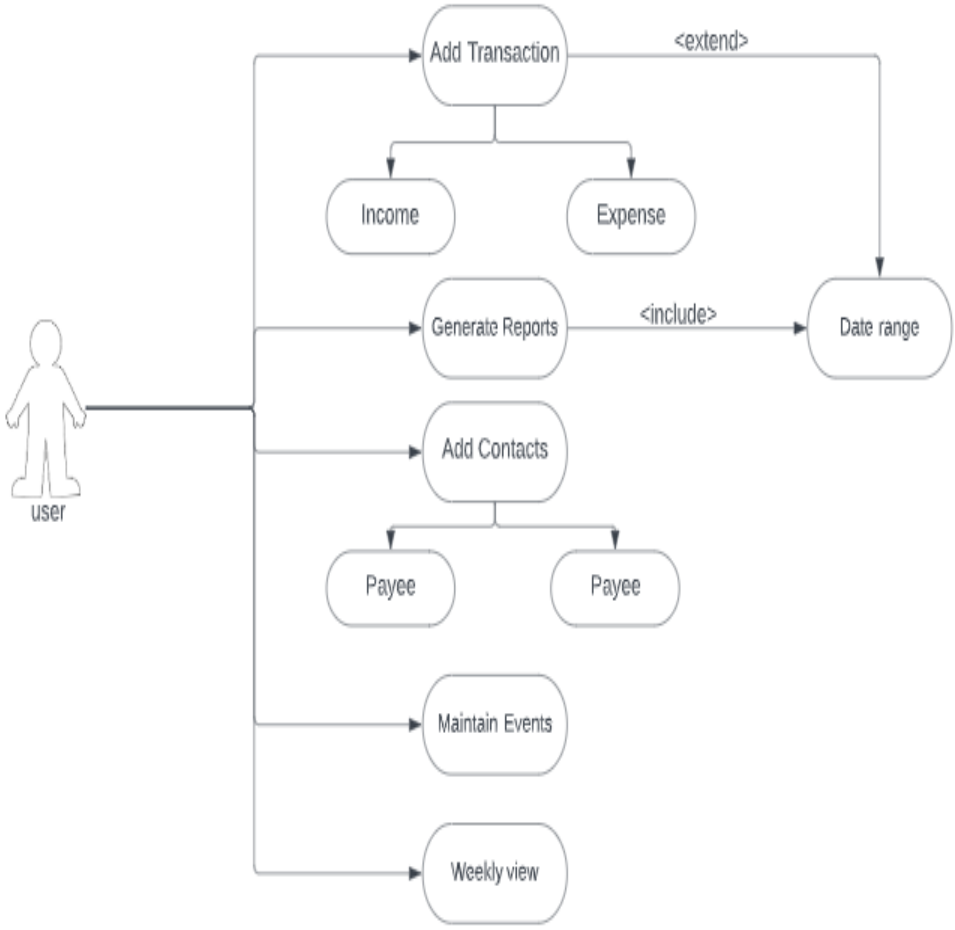
STRUCTURAL MODELS

CLASS DIAGRAM:



Ex.No: 5 (5.1) Interaction Diagrams - Use Case Diagram

USE CASE DIAGRAM :

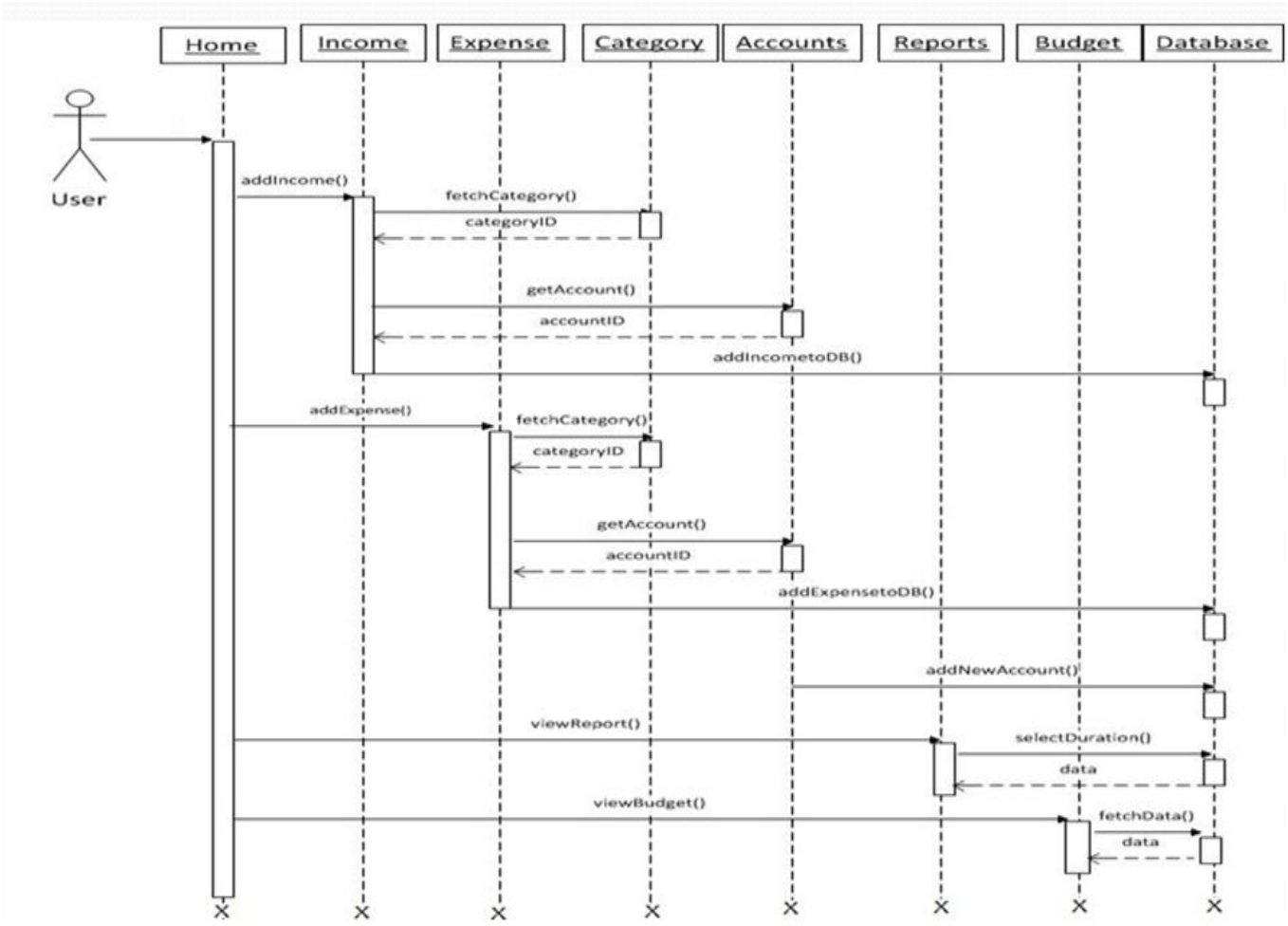


Conclusion

Thus the Interaction Diagram (Use Case Diagram) for the "Online e-commerce application" is successfully done

Ex.No: 5 (5.2) Interaction Diagrams - Sequence Diagram

SEQUENCE DIAGRAM:



Conclusion

Thus the Interaction Diagram (Use Case Diagram) for the "Online e-commerce application" is successfully done

Ex.No: 6 **"EXPENSE TRACKER SYSTEM"**

Test Cases:

1. public static void Compute(int sal, int expense)
   {
   void sal_remaining= sal - expense;
   System.out. println(sal_remaining);
   }
2. public static void createLogin()
   {
   Scanner ip = new Scanner(System.in);
   String first_name = ip.nextLine();
   String last_name = ip.nextLine();
   String username = ip.nextLine();
   String password = ip.nextLine()’
   }
3. boolean checkCredentials(String username, String pwd)
   {
   if(username == username_original)
   {
   return true;
   }
   else
   {

```
        return false;
    }
}
```

4.
```
public static void expenseFeedback(float expense)
{
    if (sal>expense)
    {
        System.out. println("Your have spent within your means.");
    }
    else
    {
        System.out. println("Your expenses have exceeded your income!");
    }
}
```

CONCLUSION

Thus the Test Driven Development with Unit Test for the **Expense Tracker System** is successfully done.