Table of Contents

[**Chapter 2: Analysis** 2](#_Toc15332703)

[**2.1** **Introduction** 2](#_Toc15332704)

[**2.1.1** **Analysis methodology, diagrams associated with methodology** 2](#_Toc15332705)

[**2.2** **Information gathering technique** 3](#_Toc15332706)

[**2.3** **Feasibility study** 4](#_Toc15332707)

[**2.4** **SRS (System Requirement Specification)** 5](#_Toc15332708)

[**2.4.1** **Functional requirement** 5](#_Toc15332709)

[**2.4.2** **Non-functional requirement** 7](#_Toc15332710)

[**2.4.3** **Prioritization** 8](#_Toc15332711)

[**2.4.4** **Requirement specification** 9](#_Toc15332712)

[**2.5** **Use case** 10](#_Toc15332713)

[**2.6** **System architecture** 12](#_Toc15332714)

[**2.6.1** **Natural Language Analysis (NLA)** 13](#_Toc15332715)

[**2.6.2** **Initial class diagram** 14](#_Toc15332716)

# **Chapter 2: Analysis**

## **Introduction**

The term ‘Analysis’ means examining something in detailed. Similarly, in software development the process of examining the system requirement, data and document.

## **Analysis methodology, diagrams associated with methodology**

Analysis methodology is an organized way of carrying out analysis. There are different methodologies that is suited for solving different types of issues. Among these methodology I have chosen OOM (Object Oriented Methodology).

OOM is a methodology which does not concentrate solely on processes or data but rather views the system as group of objects which works together to solve a task. Objects are the representation of items in the real world. The structure is divided into different steps which consists of tasks which is further divided into sub-tasks. To solve a problem different model are created like object model, dynamic model and functional model. The object model represents the static structure of the system whereas the dynamic model represents the dynamic structure that is flow of control and events. The functional model represents the internal processes.

OOM is fit for this project because of the re-usability of the analysis, design, objects and code. It is easy to understand. It is also more flexible to change as it is easier to make update in response to requirement change. The system can also be developed rapidly.

The diagram related to OOM like activity diagram, event-trace diagram, sequence diagram and class diagram will be drawn in the Design phase.

## **Information gathering technique**

Information gathering is the process of collecting information from different stakeholders that will be used to produce requirement definition (SRS). However, this task is not an easy one as just asking stakeholders will not always solve the problem as we may not get what they are trying to explain or we may get something else than they are trying to explain. There are various methods to gather information like interview, questionnaires, survey, observation, etc.

For this project I have chosen questionnaires and interview.

* **Questionnaires**

Questionnaires is a method in which a group of questions are presented to a group of people. The questions are generally of three types: fixed-alternatives, scale and open ended. This method can be used to explore qualitative data. It provides large amount of data for relatively low cost.

The list of question that can be asked are listed below:

1. What is your age?
2. Do you know how much money you have to spend each week/month?
3. Do you know how much your fixed costs (rent, utilities, etc) total each week/month?
4. Do you put a little bit of money aside regularly in case of emergencies in the future?
5. Do you avoid carrying a balance on a credit card to avoid paying interest?
6. Do you compare the cost of items (groceries, supplies, etc) at many shops before buying?
7. Do you pay mainly in cash for your purchases?
8. Do you plan inconsistent purchases (haircut, clothing, etc) so they do not fall at once?
9. Do you look for offers and deals prior to purchasing items?
10. When budgeting, do you over-estimate purchases to ensure you have enough money?
11. Do you build a little discretionary spending (for treats) into your budget?
12. Would you be willing to ask for help with budgeting if you find out you are not good at it?

* **Interview**

Interview is a one on one conversation between the interviewer and interviewee where questions are asked and answered. Interview will be a good fit to know what the problems are and how they hope them to be solved. However, this is not an easy task.

## **Feasibility study**

Feasibility study is an analysis that takes all project’s relevant factors like technical, economic, scheduling, legal and social are taken into account to determine the success of the project. It is important as it helps to find if the project is feasible or not. It identifies the pros and cons of the project to find whether the project is worth the investment of money and time.

* + 1. **Technical Feasibility**

Technical feasibility study is done to find whether is technical feasible like whether the computer can handle the processes or a new one is needed. For the current project the

* + 1. **Economic Feasibility**

In this study the economical part is taken into account and tested whether the project benefit overshadow the cost. As this project is done for educational purpose there is actually no budget. The basic requirements like a PC with specification are fulfilled as I already have a laptop with required specification so, budget will not be an issue.

* + 1. **Scheduling Feasibility**

Time is a major factor for the success of the project. Scheduling and strictly following the time schedule must be done to fulfil the requirements on time. We have about 3 month time to complete this project and I have divided the time into suitable periods to fulfil a specific task. If all the tasks are completed in the specified time then this project will be feasible in the given time.

* + 1. **Legal Feasibility**

Legal issues should not be taken for granted. It may be one of the major cause for the failure of the project. It can lead to a law suit if any laws are broken. There is no illegal issues in this project. It follow all the laws of the country so, it will not have legal issues.

* + 1. **Social Feasibility**

In a society there are many people with various cultures and traditions. Social feasibility study is done to find whether the current project can be properly implemented to the society. This project is a simple one and does not target any stereotype of people. So, this project is socially feasible.

## **SRS (System Requirement Specification)**

SRS is a part of documentation that describes the features and behavior requirements of the system. For this documentation, it contains both functional and non-functional requirements.

### **Functional requirement**

Functional requirements are the product’s functionalities like features, capabilities, usability and operations. The list of features

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Functional Requirements | Description | Rational | Dependency |
| F1 | Add Expense | User can add expense | To add expense to keep record | N/A |
| F2 | Add Income | User can add income | To add income to keep record | N/A |
| F3 | Update Expense | User can update expense | To edit the input expense | F1 |
| F4 | Update Income | User can update income | To edit the input income | F2 |
| F5 | Delete Expense | User can delete falsely entered expense | To delete the incorrect expenses | F1 |
| F6 | Delete Income | User can delete falsely entered expense | To delete the incorrect income | F2 |
| F7 | Generate graph | To produce a graph from the given data | It helps to visualize the data in graphs like bar or pie chart | F1,F2 |
| F8 | Generate report | To produce a report from the given data | It helps to generate valuable report | F1,F2 |
| F9 | Add Expense categories | To specifies the type of expense | It helps to differentiate expenses | N/A |
| F10 | Add income categories | To specifies the type income | It helps to differentiate income | N/A |
| F11 | Provide notification | Gives notification for remainder or any other purposes | Can be as a remainder or give status of the system | F1,F2 |
| F12 | Automate income | Automates repetitive tasks | Can help to ignore tedious work | F1,F2 |
| F13 | Automate expense | Automates repetitive tasks | Can help to ignore tedious work | F1,F2 |
| F14 | Export graph | Exports graph to other extensions | It helps to export valuable graph | F1,F2,F7 |
| F15 | Export report | Exports report to other extensions | It helps to report valuable graph | F1,F2,F8 |

### **Non-functional requirement**

The Requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors are non-functional requirement.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Title | Description | Dependency |
| N1 | Availability | The system should be available when it is needed. As this is an offline app it will be available even when there is no internet connection. | N/A |
| N2 | Integrity | The data should not change without intended to. Change in data may cause huge problems. | N/A |
| N3 | Documentation | This project will have proper documentation of the phases of development. It will also have an user manual. | N/A |
| N4 | Durability | The system should not have to undergo maintenance phase frequently. | N/A |
| N5 | Efficiency | The app should produce correct results. The output should not be inappropriate. | F1,F2,F7,F8 |
| N6 | User-friendly | The system should have a friendly user interface so that even user who have never used the system can easily use the system. A user manual will also be prepared to help the new user. | N/A |
| N7 | Security | Data should be always secure. It should not fall into the wrong hand. Data breach cause huge problems | N/A |
| N8 | Scalable | The project should be easy to add features or other updates. In the future there should not be problems to expand the system | N/A |
| N9 | Maintainability | The system should be easy to maintain. It should have easy fixes to bug and minor problems. | N/A |
| N10 | Reliability | The output generated should reliable. The expected results should be obtained without any failure in the system. | N/A |

### **Prioritization**

For the prioritization of requirements I have done MoSCoW prioritization in which the requirements are categorized into Must have, Should have, Could have and Won’t have.

The MoSCoW prioritization for functional requirements are:

|  |  |  |
| --- | --- | --- |
| ID | Functional Requirements | MoSCoW |
| F1 | Add Expense | M |
| F2 | Add Income | M |
| F3 | Update Expense | M |
| F4 | Update Income | M |
| F5 | Delete Expense | M |
| F6 | Delete Income | M |
| F7 | Generate graph | S |
| F8 | Generate report | S |
| F9 | Add Expense categories | M |
| F10 | Add income categories | M |
| F11 | Provide notification | C |
| F12 | Automate income | C |
| F13 | Automate expense | C |
| F14 | Export graph | S |
| F15 | Export report | S |

The MoSCoW prioritization for non-functional requirements are:

|  |  |  |
| --- | --- | --- |
| ID | Title | MoSCoW |
| N1 | Availability | M |
| N2 | Integrity | M |
| N3 | Documentation | M |
| N4 | Durability | S |
| N5 | Efficiency | M |
| N6 | User-friendly | M |
| N7 | Security | S |
| N8 | Scalable | C |
| N9 | Maintainability | C |
| N10 | Reliability | M |

Where,

M – Must have

S – Should have

C – Could have

### **Requirement specification**

**Software Requirement:**

Programming Language: Java

Operating system: Windows (7 or higher) or Linux or macOS

Database: SQLite

Platform: Android

**Hardware Requirement:**

RAM: Minimum 3 GB, recommended 8 GB (plus 1 GB for the Android Emulator)

Memory: 2 GB, 4 GB recommended

Processor: 2.16 gigahertz Intel Core i3 **processor** (dual core)

Display: 1280\*800 minimum screen resolution

## **Use case**

Use case diagram represents the interaction amongst the system and entities. It is used to identify and organize the requirements. It shows the expected behavior rather than exact method it happens. A key concept of use case modeling is that it helps us design a system from end user's perspective. It is an effective technique for communicating system behavior in the user's terms by specifying all externally visible system behavior.



Figure 1: Use case diagram

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Use Case title** | **Summary** | **Actor** |
| C1 | Add expense | The user can add expenses | User |
| C2 | Add income | The user can add income | User |
| C3 | Update expense | User can update the entered expense | User |
| C4 | Update income | User can update the entered expense | User |
| C5 | Delete expense | User can delete expense | User |
| C6 | Delete income | User can delete income | User |
| C7 | View expense | The user can view the expense added and the system can use this data too. | User, System |
| C8 | View income | The user can view the income added and the system can use this data too. | User, System |
| C9 | Add expense categories | User can add new types of expense to categorize expense | User |
| C10 | Add income categories | User can add new types of income to categorize income | User |
| C11 | Generate graph | The system generate graph from the expenses or/and income. This graph can also be exported. | System |
| C12 | Generate report | The system generate report from the expenses or/and income. This report can also be exported. | System |
| C15 | Provide notification | The system can provides notification to remind the user. | System |

## **System architecture**

For the system architecture I have chosen 3 tier architecture because it divides the system into three parts. The user uses and sees only the presentation layer and the background work is done in application layer and data layer. Our system will be most suited in this architecture as the work will be divided and there will be no confusion on which tasks to do where. Similarly, the 3 layer can work in the following way:

**Presentation layer:** This is the only layer that the user can see so, this should be made user friendly. This is the layer where input are taken from the user.

**Application layer:** In this layer the overall tasks of the system is carried out like calculations and operations. It is the layer between the presentation layer and the data layer so, the communication of these layers are completed through this layer.

**Data layer:** It is the lowest layer of this system. The basic task of this layer is to handle data like store, retrieve and update. It is the layer where database are stored.

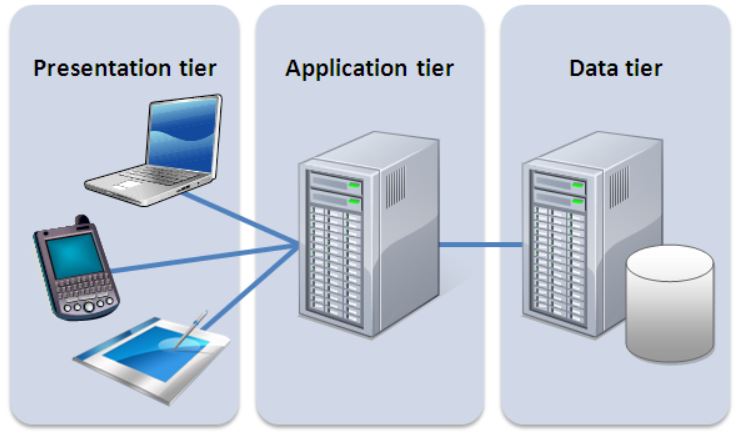


Figure 2: 3 tier architecture

### 

### **Natural Language Analysis (NLA)**

NLA is a method from which we filter out noun for classes, verbs for methods and adjectives for attributes by observation and discussion from a problem or a scenario. This method help us to identify the candidate and actual resources, attributes and properties. This method depend on each individual as there is no actual solution. So, there is no wrong or right in this process.

**Scenario**

The word "budget" has the power to make a lot of people panic, as it has a reputation for being both tedious and challenging. It really is important to keep track of how you are spending your money. It is the first step to understanding how you are managing your money, and to taking control of your finances. It’s one of the most important tools in building a successful financial future, because it helps you get the most out of your money. Tracking your expenses can help you stop feeling like you have little to show for your hard work.

Your task is to make an application which can help the users to track their transaction. The app should be able to store the income and outgoings of the user so that the user can view these records. The user should also be able to plan their budget in the future. These are the basic requirements for the app:

* Store and view the expenses
* Store and view the incomes
* Plan the budget for future

There should be categories of income (Example: salary, wages, tips, bonuses, commissions) and expenses (Example: food, transportation, entertainment). User should also be able to create their type of income or expenses. Then the user can visualize the categories in:

* Graphs. Graphical representation can help in better understanding the data. It helps to compare the expenses made with other expense or total income.
* Report. When transactions are classified into different categories or arranged in a specific order then it will be easy to view and understand.

It would be tedious to add the same expenses like food or transportation daily. To make the app interesting the app could be have automatic feature or semi- automation. For example if a user get a fixed salary every month than this could be done automatically.

The candidate class from the above scenario are:

|  |  |  |  |
| --- | --- | --- | --- |
| Budget | people | money | tools |
| expenses | User | Transaction | Income |
| Outgoing | Records | Future | App |
| Categories | Salary | Wages | Tips |
| Bonuses | Commission | Food | Transportation |
| Entertainment | Graph | Report | Automatic |

Actual class can be identified from the candidate classes listed above.

|  |  |  |
| --- | --- | --- |
| S.N. | Actual class | Description |
| 1 | Expenses | It can hold the expenses and categories of expenses |
| 2 | Income | It can hold the income and categories of income |
| 3 | Transaction | It could be the super class to the expenses and income |
| 4 | Records | It can be used to hold the records |
| 5 | Graph | It can be used to generate graph |
| 6 | Report | It can be used to generate report |
| 7 | Automatic | It will be used to do the automatic work |

### **Initial class diagram**

Looking at the above actual class diagrams the initial class diagram could look something like this:

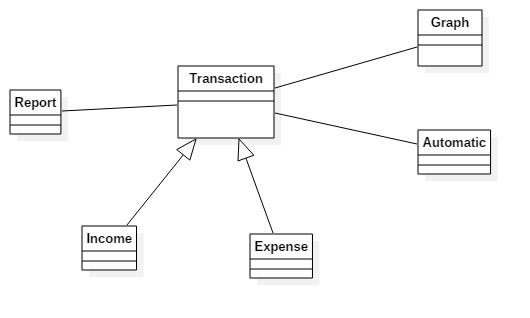


Figure 3: Initial class diagram