Exp No: 1 Reg No:312217205059

Date: Name: Niranjana S

**SOFTWARE REQUIREMENT AND SPECIFICATION**

**For Finance Management System**

1. **Introduction**
   1. **Purpose**

The purpose of finance management is to provide effective and efficient planning , organising , directing and controlling of financial activities of any organisation . It is used to keep track of activity of business partners , share holders , loaners and also about profit and losses in various transactions .

* 1. **Document Conventions**

IEEE (Institute of Electrical and Electronics Engineers) Std 830-1998 convention is followed when writing this SRS documentation. IEEE Recommended Practice for Software Requirements Specifications. IEEE Computer Society, 1998. The font-style followed here is Times New Roman with size 12.

* 1. **Product Scope**

The proposed system involves gathering information about funds acquisition from different sources , interest for various loan activities ,profit or loss details of every transaction and generates a statistical report on financial condition of the organisation. The admin manages and authorises all the users of an organisation.

* 1. **References**

The implementation of this Software refers to the finance management and the user interface and system requirements refers to various techniques involved in managing finance of company.

<https://ieeexplore.ieee.org/document/5432562>

https://www.incometax filing.com/

1. **Overall Description**
   1. **Product Perspective**

The product is useful for many organisations and companies. The organisations can register , enter the details of share holders ,business partners and business transactions taking place every day and they can manage and update the transactions .It also enables users to view the financial condition of their organisation.

* 1. **Product Functions**

The major product functions include

* The user first registers by giving essential details.
* The user submits details regarding the business partners ,share holders and clients.
* The user can upload details regarding the business transactions.
* The authenticity of the user is verified by the admin.
* The day book is updated by the accountant.
* Income tax return filing is also carried out by the accountant.
* The accountant processes the transaction details and generates a statistical report on financial position of the organisation.
  1. **User Classes and Characteristics**

The various user classes of this application include the main end users i.e.the employees of any organisation , share holders , accountant, administrator.

The finance management system can be used by any organisation who want to keep track of their profit or loss ,manage finance records and can also view financial status of their company.

Accountants play an important role in maintaining the daybook ,processing of transactions and generate finance report.

The system is also used by an admin who authorises the users and the transactions.

**Operating Environment**

This system easily coexists with many bulk software that already pre-exists in the device without any compromise to the speed of operation. The operating environment in which the system peacefully operates well include

1. Devices with the ability to connect to the internet.
2. Windows/Linux OS provided PC/Laptop.
3. Android (version > 6) supported mobile phones.
4. **Functional Requirements**

3.1 Signup: The software contains a form for organisation to register.

3.2 Database: It is used to store the details of business partners ,shareholders and transactions.

3.3 View report: This module is shall be used to view the transaction details and financial position of the company.

3.4Update Daybook: This shall be used to update the business transactions.

3.5 Statistical summary: This module shall be used to generate a monthly finance report.

|  |  |
| --- | --- |
| Requirement ID | REQ-1 |
| Requirement Statement | Sign up/Login enables the new user to register and existing user to login. |
| Author | Niranjana |
| Revision | 1.1 |
| Release Date | January 3 2020. |
| Keyword | Login |
| Scenario Description | When a user of organisation tries to access with username and password. |
| Design Assumption | The sign up and login details of the users of organisation are stored in the database. Only registered users are allowed to access the content. |

|  |  |
| --- | --- |
| Requirement ID | REQ-2 |
| Requirement Statement | It enables the users to enter the transaction details. |
| Author | Niranjana |
| Revision | 1.1 |
| Release Date | January 3, 2020. |
| Keyword | Enter |
| Scenario Description | When the user logs into the application they could enter the transaction details including transaction id ,date of transaction and amount of money. |
| Design Assumption | Once the transaction details are entered ,they are stored in a database. |

|  |  |
| --- | --- |
| Requirement ID | REQ-3 |
| Requirement Statement | Database helps to store user details as well as transaction details. |
| Author | Niranjana |
| Revision | 1.1 |
| Release Date | January 3 2020. |
| Keyword | Database |
| Scenario Description | Once the transaction details are entered by patient, admin updates the patient database with their current transactions. This database also plays an important in storing the organisation and user details. |
| Design Assumption | The database is created in MySQL server where there would be tables user details, transaction details. |

|  |  |
| --- | --- |
| Requirement ID | REQ-4 |
| Requirement Statement | View the statistical report |
| Author | Niranjana |
| Revision | 1.1 |
| Release Date | January 3 2020. |
| Keyword | View |
| Scenario Description | This module allows the registered users to view the statistical report and thus user will be able to know the financial position of the company. |
| Design Assumption | Once the transactions are processed by the accountant ,the database is update and a statistical report is generated. |

|  |  |
| --- | --- |
| Requirement ID | REQ-5 |
| Requirement Statement | Update the Daybook by accountant |
| Author | Niranjana |
| Revision | 1.1 |
| Release Date | January 2 2020. |
| Keyword | Daybook |
| Scenario Description | Once the transaction details are entered by the user ,the accountant updates the daybook. |
| Design Assumption | After the validation of transaction details, the daybook is updated by the accountant. |

1. **Non-Functional Requirements**
   1. **Performance**:

This system shall allows several organisations to register and use at the same time without downgrading the performance.

* 1. **Security:**

Each user is provided a unique username and password and also a unique organisation ID is assigned to them. This system shall make it impossible for unauthorized access.

* 1. **Usability:**

This system shall be easy to learn and use by all the users and is suitable for all types of users.

* 1. **Availability:**

This system shall be available to all users and can be accessed anywhere.

* 1. **Software Quality Attributes:**

This system provides other non-functional requirements such as adaptability and portability. This system is adaptable to any operating environment and can be operated without any performance degradation in other platforms too.

1. **External Interface Requirements**

**5.1 Hardware Requirements**

Standard PC/Laptop with the ability to connect to the internet

**5.2 Software Requirements:**

HTML

CSS

JAVASCRIPT

PHP

MySQL SERVER

**Result:**

Thus, the software requirements specification for the Finance Management system has been developed.

Exp No: 2 Reg No:312217205059

Date: Name: Niranjana S

**USE CASE MODEL**

**Aim:**

To identify use cases and develop the Use Case Model.

**Use Case Diagram:**

A use case diagram is a dynamic or behaviour diagram in [UML](https://www.smartdraw.com/uml-diagram/). Use case diagrams model the functionality of a system using actors and use cases. Use cases are a set of actions, services, and functions that the system needs to perform. In this context, a "system" is something being developed or operated, such as a web site. The "actors" are people or entities operating under defined roles within the system.

**Use Case Diagram Notations:**

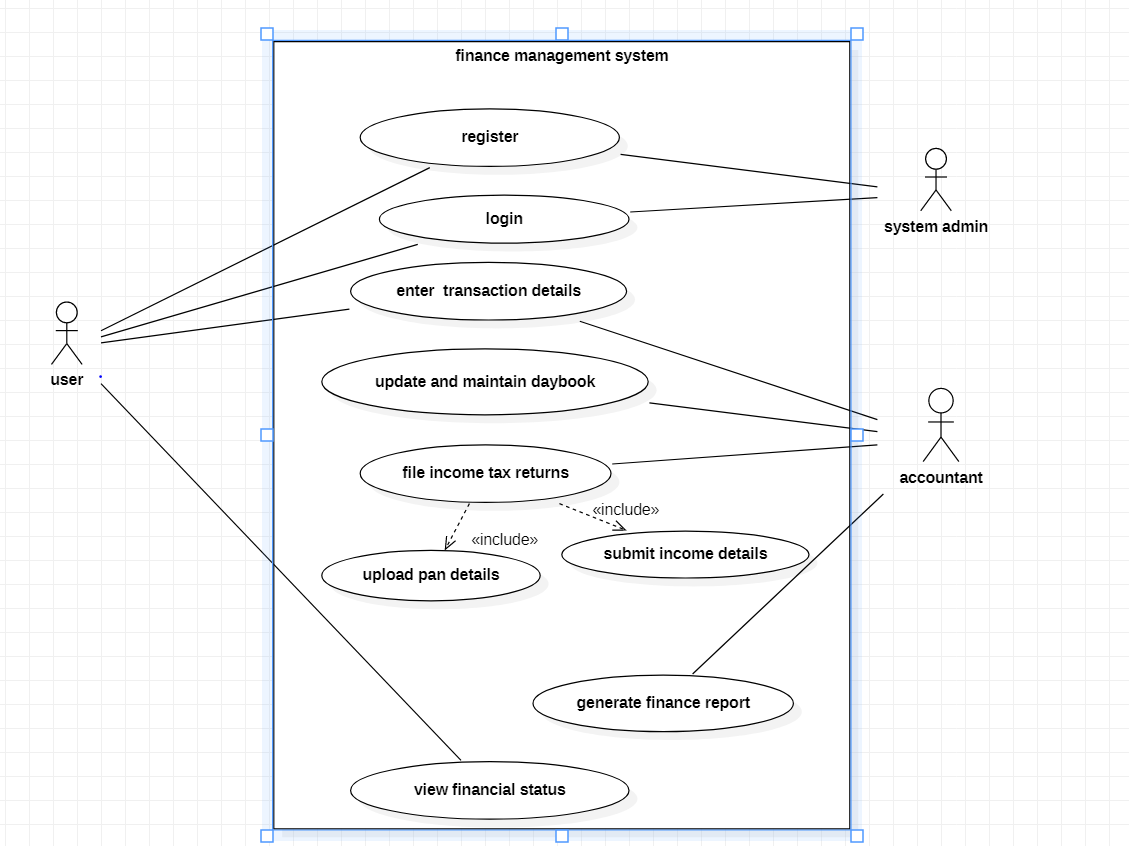
* **Use cases:** Horizontally shaped ovals that represent the different uses that a user might have.
* **Actors:** Stick figures that represent the people actually employing the use cases.
* **Associations:** A line between actors and use cases. In complex diagrams, it is important to know which actors are associated with which use cases.
* **System boundary boxes:** A box that sets a system scope to use cases. All use cases outside the box would be considered outside the scope of that system. For example, Psycho Killer is outside the scope of occupations in the chainsaw example found below.
* **Packages:** A UML shape that allows you to put different elements into groups. Just as with component diagrams, these groupings are represented as file folders.

**Use Case Relationships:**

| **Relationship** | **Symbol** | **Meaning** |
| --- | --- | --- |
| Communicates | https://www.w3computing.com/systemsanalysis/wp-content/uploads/2014/08/2.13.1.jpg | An actor is connected to a use case using a line with no arrowheads. |
| Includes | https://www.w3computing.com/systemsanalysis/wp-content/uploads/2014/08/2.13.2.jpg | A use case contains a behaviour that is common to more than one other use case. The arrow points to the common use case. |
| Extends | https://www.w3computing.com/systemsanalysis/wp-content/uploads/2014/08/2.13.3.jpg | A different use case handles exception from the basic use case. The arrow points from the extended to the basic use case. |
| Generalizes | https://www.w3computing.com/systemsanalysis/wp-content/uploads/2014/08/2.13.4.jpg | One UML "thing" is more general than another "thing." The arrow points to the general "thing." |

**Use Case Diagram:**

Finance Management System



**Use Case Text:**

The purpose of this system is to assist the users to register and maintain the financial records of their organisation. They can also file income tax returns and view the financial position of their organisation.

**Actors:**

User - Primary Actor

Admin - Primary Actor

Accountant – Secondary actor

**Basic Flow:**

1. The User registers an account.
2. The User logins to their profile.
3. The Admin validates his/her credentials.
4. The User enters the transaction details.
5. The details are verified and stored in the database.
6. The Accountant processes all the transactions.
7. The daybook is updated.
8. The Accountant prepares a financial report.
9. The Accountant also files the income tax returns based on the income of the company.
10. User logs out.

**Alternate Flow:**

1. The User logs in to their profile.
2. The admin updates the report.
3. The User can view the finance report.

**Use case Specification:**

Register – This module is used to register the profile for a new user.

Login – This module is for the user to login to their profile.

Enter Transaction Details – This module allows the users to enter transaction details.

Maintain daybook– This module is used by accountant to update the daybook entries.

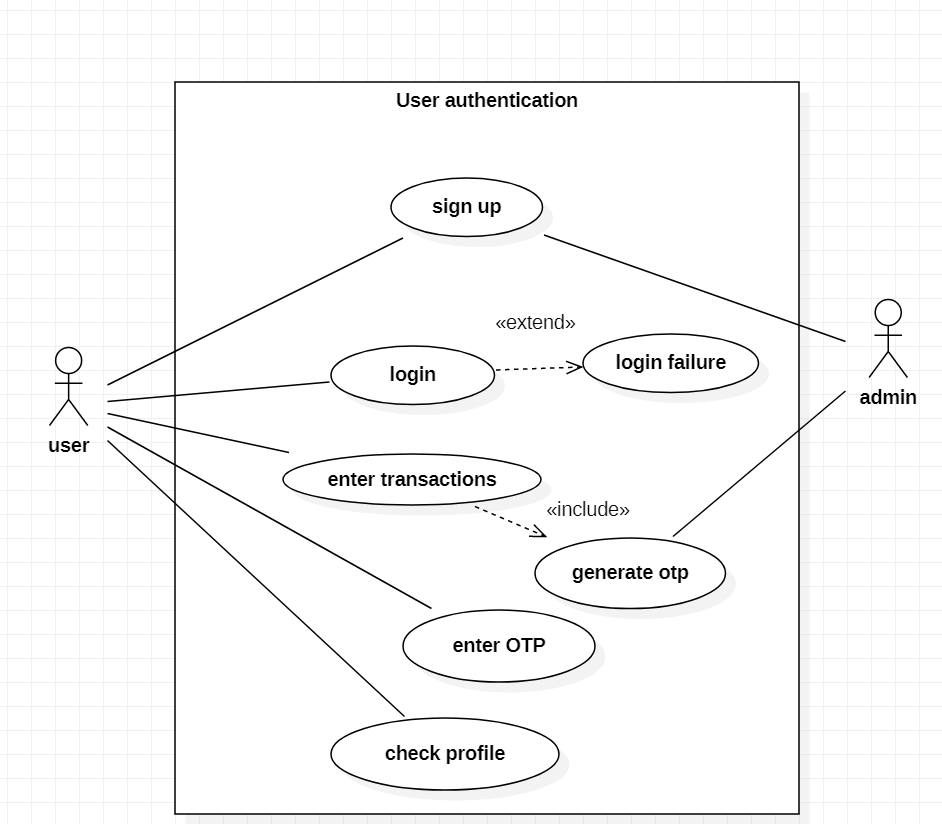
Process transactions – This module is used by accountants to process all the transactions.

File income tax returns – This module is used by the user to file income tax returns.

Generate finance report – This module is used by the Accountant to generate the finance report.

View Finance Report- This module enables users to view the financial condition of their company.

* 1. **User Authentication**



**Use Case Text:**

In this module the user is allowed to login to his account. If he is a new user then he needs to register. After verifying the credentials, the user is authenticated and by the generation of one time password. After entering OTP, user logins to the system.

**Actors:**

User - Primary Actor

Admin - Primary Actor

**Basic Flow:**

* 1. The user logs in to his account.
  2. If he is a new user then he registers.
  3. The user can enter the transaction details only after verification of OTP.
  4. The user details are got and inserted into the database.
  5. The user can check their profile.

**Use Case Specifications:**

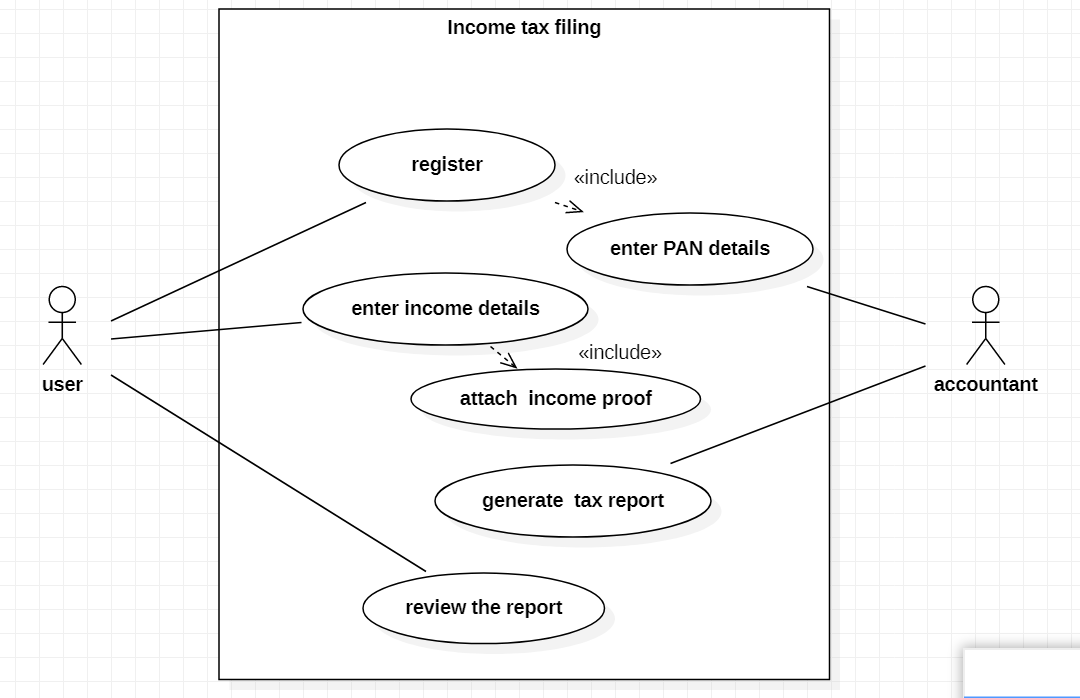
Register – This module is used to register the profile for a new user.

Login – This module is for the user to login to their profile.

Enter Transactions – This module allows the user to enter the transaction details only after the verification of OTP and the details are then stored in the database.

Check profile – This module is used by the user to check profile.

* 1. **Filing of income tax returns**



**Use Case Text:**

In this module the user can file income tax returns for their organisation.

**Actors:**

User - Primary Actor

Accountant - Primary Actor

**Basic Flow:**

* 1. The user logs in to his account.
  2. The user can register by entering the PAN details.
  3. The user enters the income details.
  4. The accountant files the income tax returns.
  5. The user can view the income tax returns.

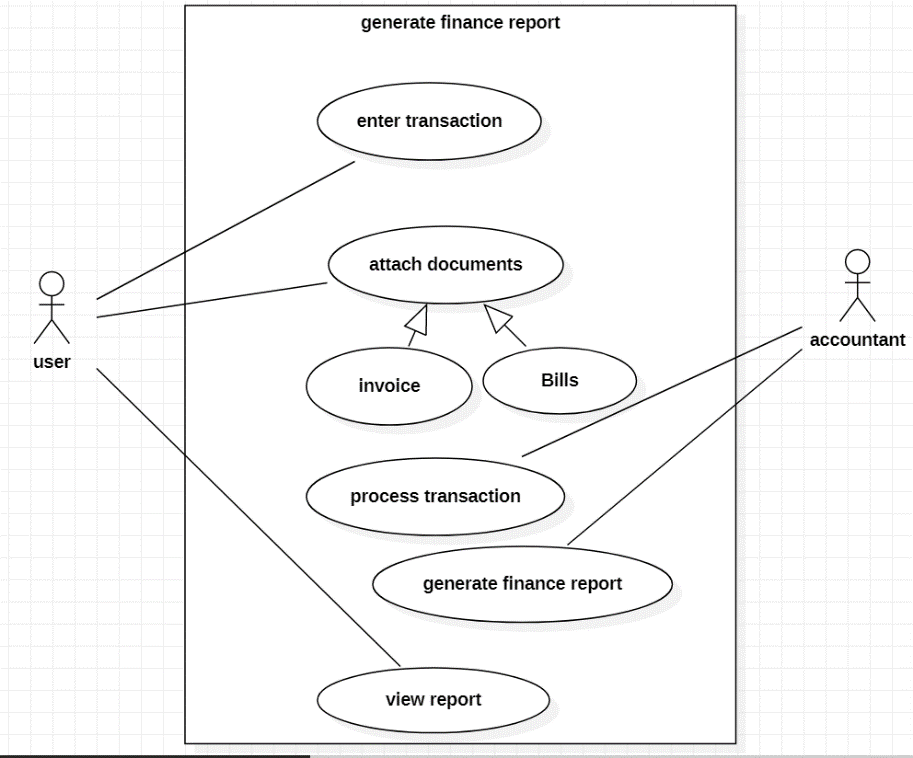
**Use Case Specifications:**

Register– This module is used to register into the system.

Income details – This module is used for entering the income details.

Generate income tax report- The account generates the income tax report.

* 1. **Generate Finance Report**



**Use Case Text:**

The user enters the transaction after attaching the essential documents such as invoice and bills. The accountant after authenticating the transaction details, processes the transaction and generates a finance report. The users can view the finance report.

**Actors:**

User -Primary Actor

Accountant – Primary Actor

**Basic Flow:**

* 1. The user registers in the system.
  2. The user enters the transaction details only after submitting the essentils documents such as bills and invoice.
  3. The accountant processes and analyses the transactions.
  4. The finance report is generated.
  5. Finally, the user can view the finance report.

**Use Case Specifications:**

Enter Transactions– This module allows the users to enter the transaction details.

Generation of finance report – This module is used by the Accountant to generate finance report.

**Result:**

Thus, the use cases have been identified and the Use Case Model has been developed.

Exp No: 3a Reg No:312217205059

Date: Name: Niranjana S

**DOMAIN MODEL**

**Aim:**

To identify the conceptual classes and develop a domain model.

**Domain Model:**

A domain model is a visual representation of conceptual classes or real-world objects in a domain of interest. The Domain Model is your organized and structured knowledge of the problem. The Domain Model should represent the vocabulary and key concepts of the problem domain and it should identify the relationships among all of the entities within the scope of the domain.

Most of the time a domain model is illustrated with a set of class diagrams which may show:

* domain objects or conceptual classes
* associations between conceptual classes
* attributes of conceptual classes

Identifying Conceptual Classes

1. Modify or reuse an existing model.
2. Use a conceptual class category list.
3. Identify noun phrases.

Heuristics for Identifying Associations

Some heuristics for identifying associations are:

* examine verb phrases,
* ensure that the roles and association names are clear,
* only add an association if it improves the understanding of the domain,
* wait until the list of associations are stable before considering the multiplicity,

Identifying Attributes

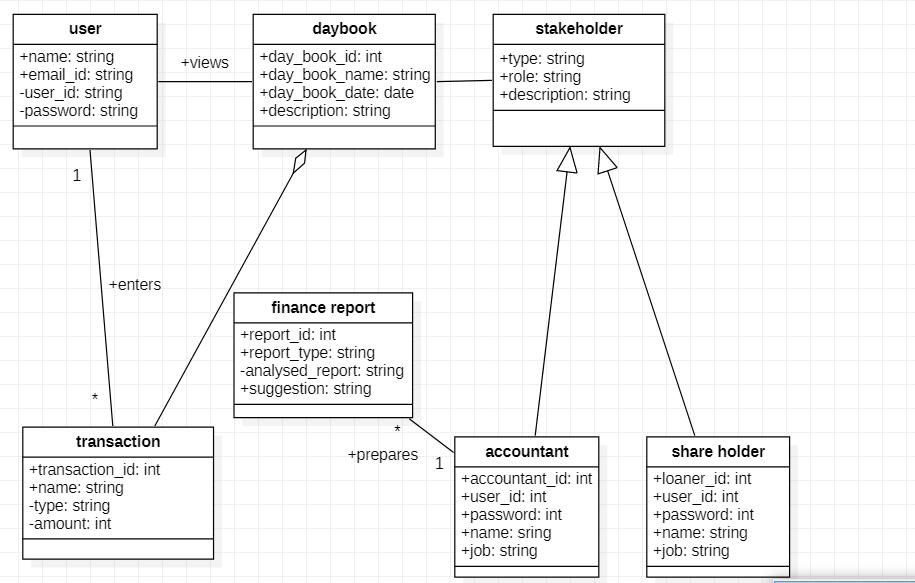
An attribute is a logical data (property) of an object (e.g., my eyes are hazel). Most attributes can be represented by simple data types (which are?).

Some heuristics for identifying attributes:

* an attribute is part of the state of an object (a car's speed is 100 km/h, weight of a work item)
* attributes are required by the use case (i.e., ignore irrelevant attributes).

Judgment is required to separate attributes from associated classes.

**Domain Model for Finance Management System:**



**Result:**

Thus, the conceptual classes have been identified and the domain model has been developed.

Exp No: 3b Reg No:312217205059

Date: Name: Niranjana S

**CLASS DIAGRAM**

**Aim:**

To derive class diagram from the domain model.

**Class Diagram:**

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

A class notation consists of three parts:

1. Class Name
   * The name of the class appears in the first partition.
2. Class Attributes
   * Attributes are shown in the second partition.
   * The attribute type is shown after the colon.
   * Attributes map onto member variables (data members) in code.
3. Class Operations (Methods)
   * Operations are shown in the third partition. They are services the class provides.
   * The return type of a method is shown after the colon at the end of the method signature.
   * The return type of method parameters is shown after the colon following the parameter name.
   * Operations map onto class methods in code

**Relationships in class diagram:**

Generalization:

Generalization is a relationship between two classes: a general class and a special class:

https://sourcemaking.com/files/sm/images/uml/img_127.jpg

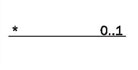
Association

An association represents a relationship between two classes:

An association indicates that objects of one class have a relationship with objects of another class, in which this connection has a specifically defined meaning.

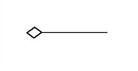
Multiplicity

A multiplicity allows for statements about the number of objects that are involved in an association:

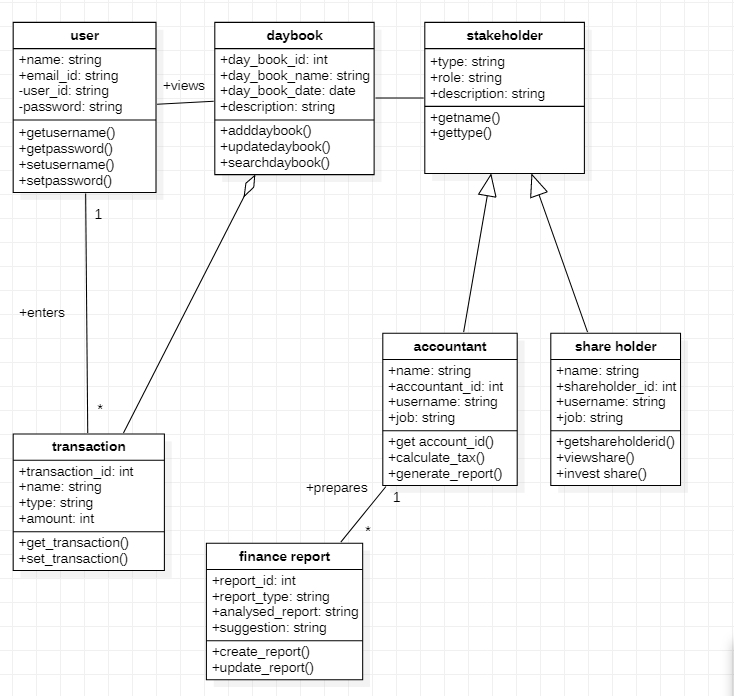


Aggregation

An aggregation is a special case of an association (see above) meaning “consists of”:



**Class Diagram for Finance Management System:**



**Result:**

Thus, the class diagram has been derived from the domain model.

Exp No: 4a RegNo:312217205059

Date: Name: Niranjana S

**SEQUENCE DIAGRAM**

**Aim:**

To create sequence diagram based on the use cases from the use case diagram.

**Sequence Diagram:**

A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.

### **Sequence Diagram Notations –**

1. **Actors** – An actor in a UML diagram represents a type of role where it interacts with the system and its objects. It is important to note here that an actor is always outside the scope of the system we aim to model using the UML diagram.
2. **Lifelines –** A lifeline is a named element which depicts an individual participant in a sequence diagram. So each instance in a sequence diagram is represented by a lifeline. Lifeline elements are located at the top in a sequence diagram. The standard in UML for naming a lifeline follows the following format – Instance Name : Class Name
3. **Messages –** Communication between objects is depicted using messages. The messages appear in a sequential order on the lifeline. We represent messages using arrows. Lifelines and messages form the core of a sequence diagram.

Messages can be broadly classified into the following **categories:**

**Synchronous messages –** A synchronous message waits for a reply before the interaction can move forward. The sender waits until the receiver has completed the processing of the message.

**Asynchronous Messages –** An asynchronous message does not wait for a reply from the receiver. The interaction moves forward irrespective of the receiver processing the previous message or not. We use a lined arrow head to represent an asynchronous message.

**Create message –** We use a Create message to instantiate a new object in the sequence diagram. There are situations when a particular message call requires the creation of an object.

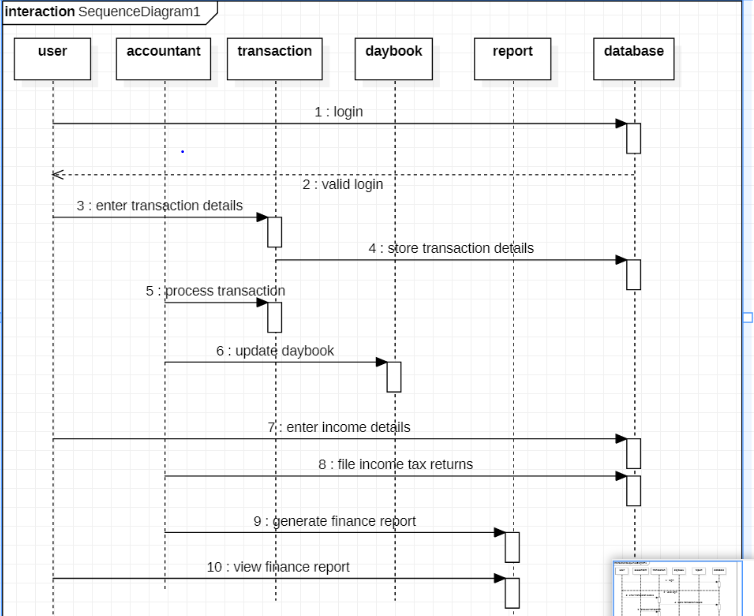
**Self-Message –** Certain scenarios might arise where the object needs to send a message to itself. Such messages are called Self Messages and are represented with a U-shaped arrow.

**Reply Message –** Reply messages are used to show the message being sent from the receiver to the sender. We represent a return/reply message using an open arrowhead with a dotted line.

**Found Message –** A Found message is used to represent a scenario where an unknown source sends the message. It is represented using an arrow directed towards a lifeline from an end point.

**Lost Message –** A Lost message is used to represent a scenario where the recipient is not known to the system. It is represented using an arrow directed towards an end point from a lifeline.

**Sequence Diagram for Finance Management System**



The above sequence diagram depicts the sequence:

1. The User registers an account.
2. The User logins to their profile.
3. The Admin validates his/her credentials.
4. The User enters the transaction details.
5. The transaction details are stored in a database.
6. The accountant processes the transactions.
7. The Accountant updates the daybook.
8. The user can enter the income details.
9. The accountant files income tax returns and also generates finance report.
10. The user can view finance report.
11. The user logs out.

**Result:**

Thus, the sequence diagram has been created based on the use case model.