|  |
| --- |
|  |
| Apartment Management System |
| Object Oriented Analysis and Design (CS6110) |

|  |
| --- |
| Ganesh Kumaar S.  2018503025  MN Batch |

**Experiment 1: Application**

**Application**: Apartment Management System

**Description**: A management system that can be used to track and monitor the accounts of a small apartment complex and generate reports.

Experiment 2: SRS

# Software Requirements Specification (SRS)

# for

# Apartment Management System

Ganesh Kumaar S.

Regno: 2018503025

Batch: MN

Table of Contents

Table of Contents 3

Revision History 3

1. Introduction 4

1.1 Purpose 4

1.2 Document Conventions 4

1.3 Intended Audience and Reading Suggestions 4

1.4 Product Scope 5

2. Overall Description 6

2.1 Product Perspective 6

2.2 Product function 6

2.3 User Classes and Characteristics 6

2.4 Operating Environment 6

2.5 Design and Implementation Constraints 6

Constraints  • The user require a computer to run this application • The user has to be careful while submitting the information. Much care is required. 7

2.6 Assumptions and Dependencies 7

3. External Interface Requirements 8

3.1 User Interfaces 8

3.2 Hardware Interfaces 8

3.3 Software Interfaces 8

3.4 Communications Interfaces 8

4. System function 9

4.1 Setup apartment details (Apt#, area, monthly maintenance dues, contact details) 9

4.2 Enter receipt of monthly maintenance dues 9

4.3 Enter details of expenses incurred in running the appartment complex 9

4.4 Generate report of defaulting residents of monthly maintenance dues 9

4.5 Generate monthly report of receipts and expenses 9

5. 5.Other Nonfunctional Requirements 10

5.1 Performance Requirements 10

Appendix A: Glossary 11

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason for Changes** | **Version** |
| Ganesh Kumaar S. | 27/8/20 | Initial Version | 1.0 |
|  |  |  |  |

# Introduction

## Purpose

Apartment Management System is an application that can be used by smaller apartment complexes to manage their books. It aims at improving the efficiency in managing monthly subscription fee collection, providing overdue notices, monthly reports, etc.

## Document Conventions

  Font                             Style               Size  
  
Heading                           Georgia     Bold                 18  
  
Sub-Heading                   Georgia Bold               14  
  
Other’s                             Georgia     Regular           11

## Intended Audience and Reading Suggestions

This SRS is mainly developed for the project development team. In this team there are the project treasurer, developer, coder, tester and documentation writer and the user of the project also.  
  
**User (Customer)**  
    This document is intended to user and customer to make ensure that the document satisfies the needs of the customer.  
  
**Project Treasurer**  
    This SRS document is also very important for the project treasurer as it helps in cost estimation which can be performed by referring to the SRS document and it contains all the information that is required for the project plan.  
  
**Project Developer**  
      The project developer will refer to the SRS document to ensure that the product developed is as per the needs of the customer.  
  
**Tester**  
        The tester reads the SRS document to ensure that the requirements are understandable based on the     
functionality specified so that he can test the software and validates its working.  
  
**Document Writer**  
        The document writer reads the SRS document to ensure that they understand the document well enough and write user manuals based on the SRS document.

**Maintenance**

The SRS document helps the maintenance engineers to understand functionality of the system. A clear knowledge of the functionality is needed to design and code

## Product Scope

The System provides a standalone application that can track the monthly maintenance dues, provide a summary of defaulting residents, track the expenses of the apartment complex and to generate monthly reports.

• Currently the books are managed manually in ledgers. This system would help to track the details in computer and generate reports.

• The clerk or general user can input the receipt of monthly maintenance dues, and also any expenses incurred in maintaining the apartment complex.

• The treasurer / Apartment Treasurer and can run the MIS (Management Information System) reports to find out defaulting residents, monthly statements, etc.

**1.5Definitions, Acronyms and the Abbreviations**

• **Administrator** - Refers to the super user who is the Central Authority who has been vested with the privilege to manage the entire system.

• **Clerk** – the user who can enter the receipt of monthly maintenance dues and any expenses incurred

• **Treasurer** **/Treasurer** – the user who can review the records, generate monthly reports

**1.6 References**

* <https://en.wikipedia.org/wiki/Accounting>
* <https://docs.python.org/3/reference/>
* <https://docs.python.org/3/library/tk.html>
* <https://docs.python.org/2/library/tkinter.html>
* <https://docs.python.org/3/library/sqlite3.html>

# Overall Description

## Product Perspective

The Apartment Management System helps smaller apartment complexes to manage their books in a system thereby helping them to manage the tracking of monthly maintenance receipts, and expenses incurred in the running the apartment complex. This system helps in moving from manual ledger entries to computerized tracking receipts and expenses and in generating timely reports.

## Product function

• Setup apartment details (Apt#, area, monthly maintenance dues, contact details, etc.)  
• Setup users (Clerk and Treasurer)

• Develop data entry screen for entering monthly maintenance receipts

• Develop data entry screen for entering expenses

• Generate report of defaulting residents (overdue maintenance dues)

• Generate report of monthly receipts and expenses

## User Classes and Characteristics

• **Clerk** – The person who is going to enter the receipts (monthly maintenance dues) and expense details in the system

• **Treasurer** – The person who can review the data entered by clerk and generate reports

## Operating Environment

|  |  |
| --- | --- |
| Particulars | Client System |
| Operating System | Windows |
| Processor | Intel or AMD |
| Hard Disk | 250 GB |
| RAM | 4 GB |

## Design and Implementation Constraints

• The apartment needs a computer to run this application.

• The users need to be trained on the application to enter details, review and print reports.

## Constraints

## • The user requires a computer to run this application • The user has to be careful while submitting the information. Much care is required.

## Assumptions and Dependencies

* The users must have basic knowledge of computers and English Language.
* Each User must have a UserID and password
* There must be an Administrator

# External Interface Requirements

## User Interfaces

• **Clerk** – This is the user who is going to enter the transactional details into system such as recording the receipts of monthly maintenance dues, enter details of expenses incurred in maintaining the apartment complex

• **Treasurer** – This user can review the details entered by the clerk, run reports such as defaulting residents and monthly reports

## Hardware Interfaces

This is a standalone application running on a Windows desktop.

## Software Interfaces

• Front End screens are developed in Python which are used to setup the application, enter data for receipts and expenses

• Back End – Sql lite database.

## Communications Interfaces

This application doesn’t require any online connectivity.

# System function

## Setup apartment details (Apt#, area, monthly maintenance dues, contact details)

This function will do the initial setup of the application at the apartment complex. This will include setting up users, setting up the number of apartments and their details such as apartment number, area, monthly maintenance dues, contact details.

## Enter receipt of monthly maintenance dues

As residents pay the monthly maintenance dues, clerk will enter those details in the system.

## Enter details of expenses incurred in running the apartment complex

This function will allow the clerk to enter the details of the expenses incurred for running the apartment complex such as staff salary, electricity dues, diesel for genset and other routine maintenance and repairs.

## Generate report of defaulting residents of monthly maintenance dues

After the cutoff date (5th of the month), the treasurer will be able to use this function to generate the defaulting residents who have not paid their monthly maintenance dues.

## Generate monthly report of receipts and expenses

At any point, the treasurer would be able to generate the monthly statement for any of the previous months for review.

# Other Nonfunctional Requirements

## 5.1 Performance Requirements

Performance requirements define acceptable response times for system functionality.

* The total time for user interface screens will take no longer than a few seconds.
* The login information shall be checked within the seconds.
* Reports should be generated within a few seconds.

## 5.2 Security Requirements

The software should be only accessible to users with valid credentials and users must have the right or privilege to perform a certain task within the software to make any changes. The software should also maintain the data in a secure and encrypted way to prevent loss of data.

## Software Quality Attributes

* **Design Constraints**The software shall be a standard system running in a windows environment.
* **Reliability**Given a certain set of data the result should always be the same.
* **Availability**The system should have an availability of 99.99%.
* **Portability**

The system should be extremely and the system shall be easy to migrate or

backed up.

* **Maintainability**

The system shall utilize multiple backups plugins. The system shall be easily updated for

fixes and patches.

## 5.4 Business Rules

Certain actions maybe be taken when a resident has been a defaulter and has not paid their fees over along period of time. Also, when a user is not available for various reasons, only a user with equal or higher role can make changes instead of them.

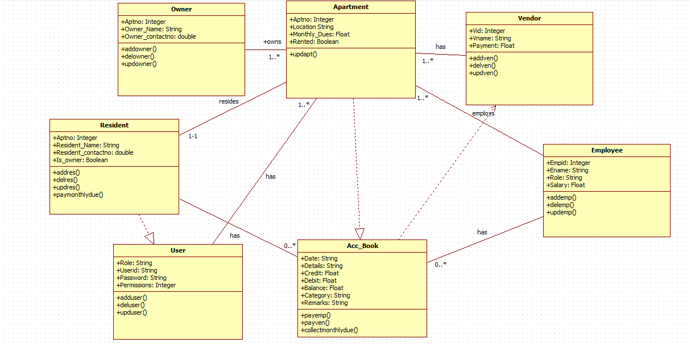
Appendix A: Glossary

Definitions, Acronyms and Abbreviation:

* SRS: Software Requirement Specification
* Client/User: The entity who will be using the system
* RAM: Random Access Memory
* mySQL: A relational database management system
* Username: Unique name given to each account of digital library
* Password: Unique word given to each user as a secret code

**Experiment 3: Class Diagram**

**Class Diagram:**



**List of Classes:**

* Apartment – An apartment class that contains basic details about an apartment such as apt no, location and monthly dues etc.
* Owner – A Owner class that contains the information about the owner of an apartment such as owner contact details and name etc.
* Resident – A Resident class that contains the information about the owner of an apartment such as owner contact details and name etc.
* User – A User class that contains details about the user such as the user’s login credentials.
* Vendor – A Vendor class that contains details such as the vendor name, the service that they provide and the cost of that service.
* Employee – An Employee class that contains details about an employee such as name and the salary of the employee
* Account Book - Account Book is a class that is used to maintain a record of all the transactions that are done from and to the apartment by the treasurer.

**Experiment 4: Inception**

**Apartment Management System Inception**

**Vision**

To develop an application for small apartment complexes to help maintain their accounts in a computerized system.

**Scope**

The goal of the apartment management system application is to improve and automate the current methods that are being used by small apartment complexes to maintain their accounts and monthly expenditures. Currently these small apartment complexes manually record all their transactions done by the Apartment Treasurer with the help of ledgers, in most cases without a backup. Many of the problems of manual book keeping can be solved by automating this system.

**Business Case**

The residents of the apartment complex will have to pay their monthly dues which needs to be collected and recorded as a receipt in the ledger books of the apartment. Also, when a resident has not paid their monthly dues for a month before the due date, they can be added to the defaulter list and a fine can be imposed on those residents in the defaulter list. The apartment must also pay its employees and the various vendors that provide services such as the watchman and water supplier respectively. These transactions must also be recorded in the ledger book as expenses. The system should also generate monthly statements, defaulters list, etc.

**Feasibility**

The development of the apartment management system application is feasible as it is not an overly complex application to develop. The application will require a database to store various data such as the residents, employees, vendors as well as the various transactions that are done by the apartment. The application will also require a user interface to interact with and basic training of the employees to use it. A backup should also be maintained.

**Buy or Build**

Commercial off the shelf applications are too expensive and cannot be afforded by small apartment complexes. Also, commercial applications will have a lot of features that are not required and expensive to customize. Hence, this application should be built rather then bought.

**Cost**

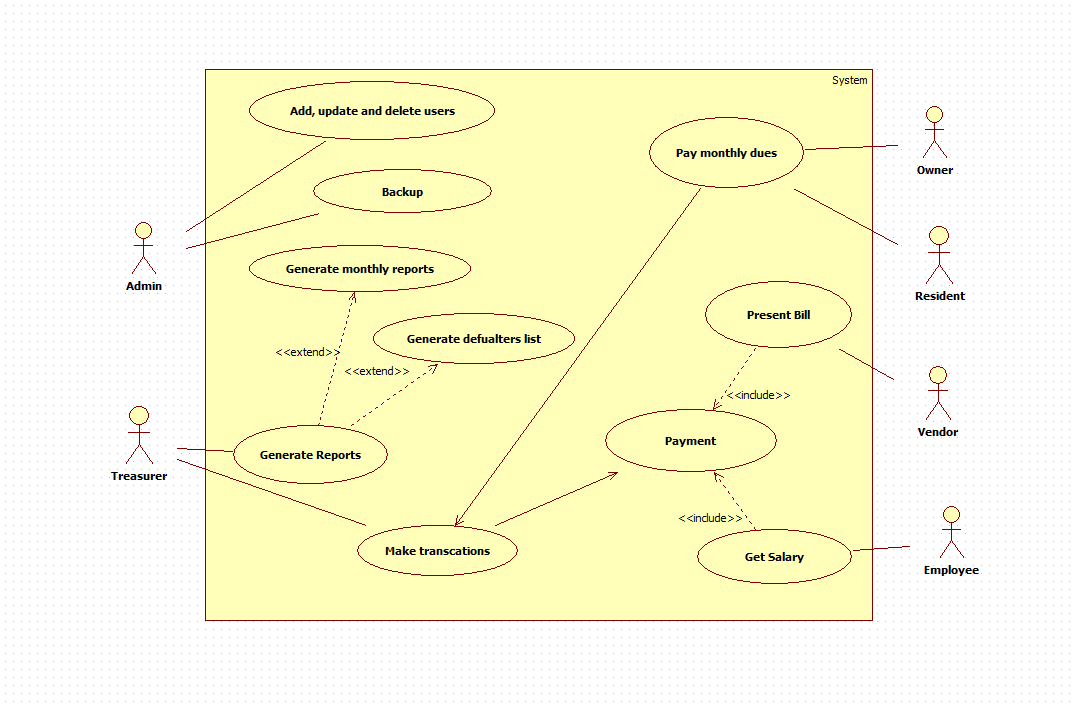
The cost of building and maintaining the application would also be minimal as it does not require much man power, a single person over the course of a month can finish building the application and the same person can help in maintaining the application over time.

**Should we proceed or stop?**

We should proceed with the development of the apartment management system as it is feasible and cost effective and also can make the booking keeping of an apartment much more efficient. Proceeding with this project can also generate a significant amount of revenue for the stake holders and investors.

**Experiment 5: Use Case Diagram**

**Use Case Diagram:**



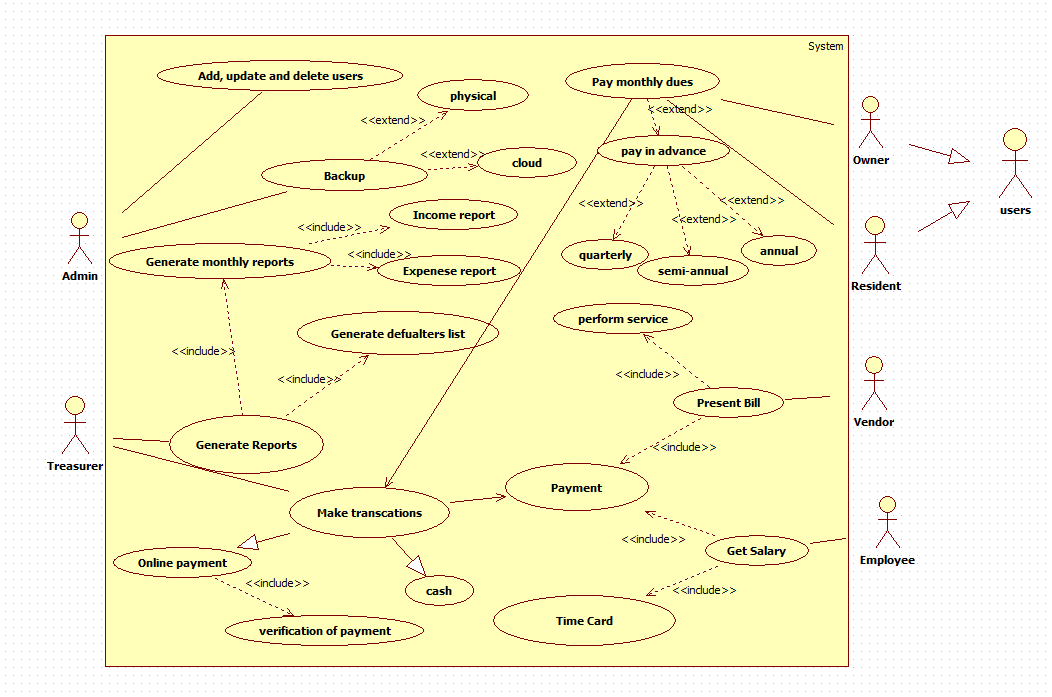
**Description:**

|  |  |
| --- | --- |
| **Use case UC1** | Generate monthly reports |
| **Scope** | Apartment Management System |
| **Level** | User Goal |
| **Goal** | To generate a report of all of the transactions carried out during the month in the maintenance of the apartment. |
| **Actors** | **Primary Actors**:  **Treasurer**: Generate report |
| **Secondary Actors**:  **Vendor**: Gets payment  **Employee**: Gets salary  **Residents**: Pays monthly dues |
| **Flow** | 1. Residents pay monthly fees 2. Vendors and employees receive payment 3. All transactions are recorded by the system 4. Treasurer generates a monthly report. 5. A monthly report of all these transactions are generated by the system. |

|  |  |
| --- | --- |
| **Use case UC2** | Add, delete and update users |
| **Scope** | Apartment Management System |
| **Level** | User Goal |
| **Goal** | To add delete or update a user to the system |
| **Actors** | **Primary Actors**:  Admin: Add, delete and update user |
| **Secondary Actors**:  **Resident**: Add, delete & update  **Vendor**: Add, delete & update  **Employee**: Add, delete & update |
| **Flow** | 1. Gather the necessary information about the user 2. Add, update or delete the necessary field of the particular user. 3. Save the changes 4. The user has been updated, added or deleted from the system. |

**Experiment 6: Refined Use Case Diagram**

**Refined Use Case Diagram:**

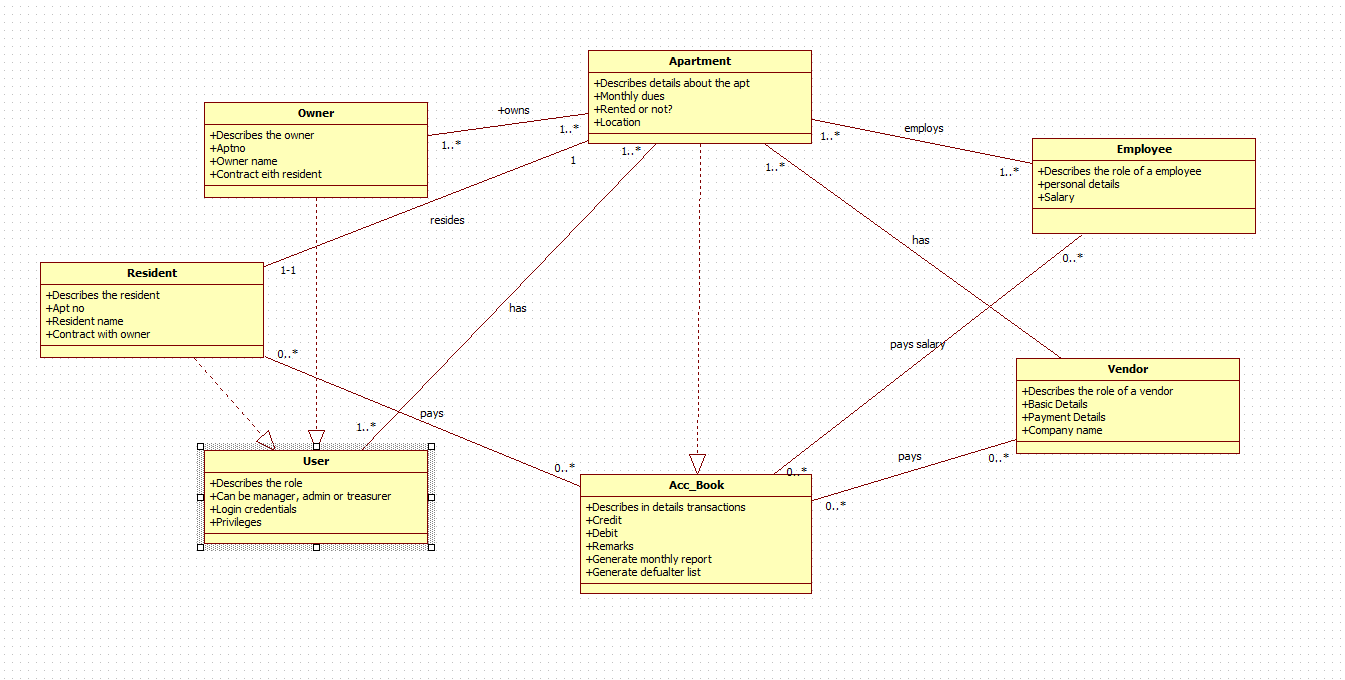
****

**Added use cases:**

* Backup extends physical backup
* Backup extends cloud backup
* Generate reports includes income report
* Generate reports includes expense report
* Make transactions generalizes online payment
* Make transactions generalizes cash
* Online payment includes verification of payment
* Get salary includes time card
* Present bill includes perform service
* Pay monthly dues extends pay in advance
* Pay in advance extends quarterly
* Pay in advance extends semi- annual
* Pay in advance extends annual

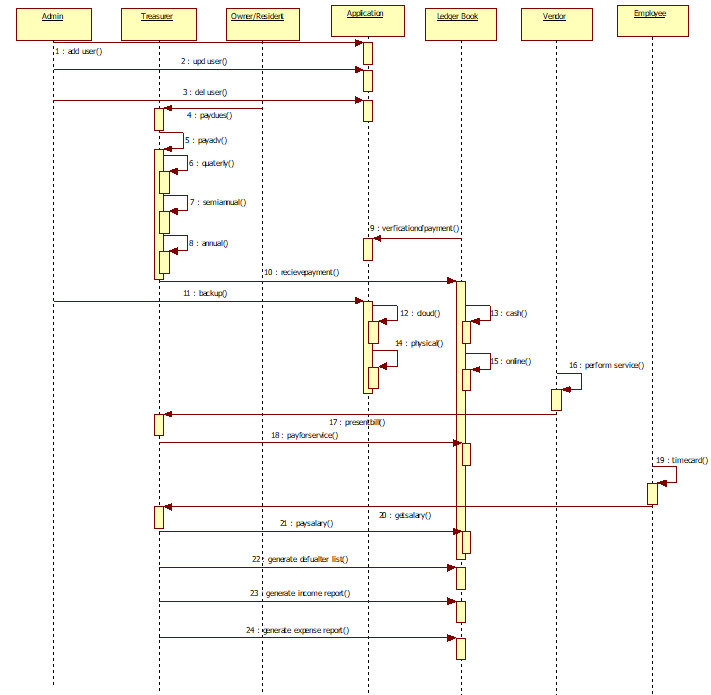
**Experiment 7: Domain Model**

**Domain Model:**

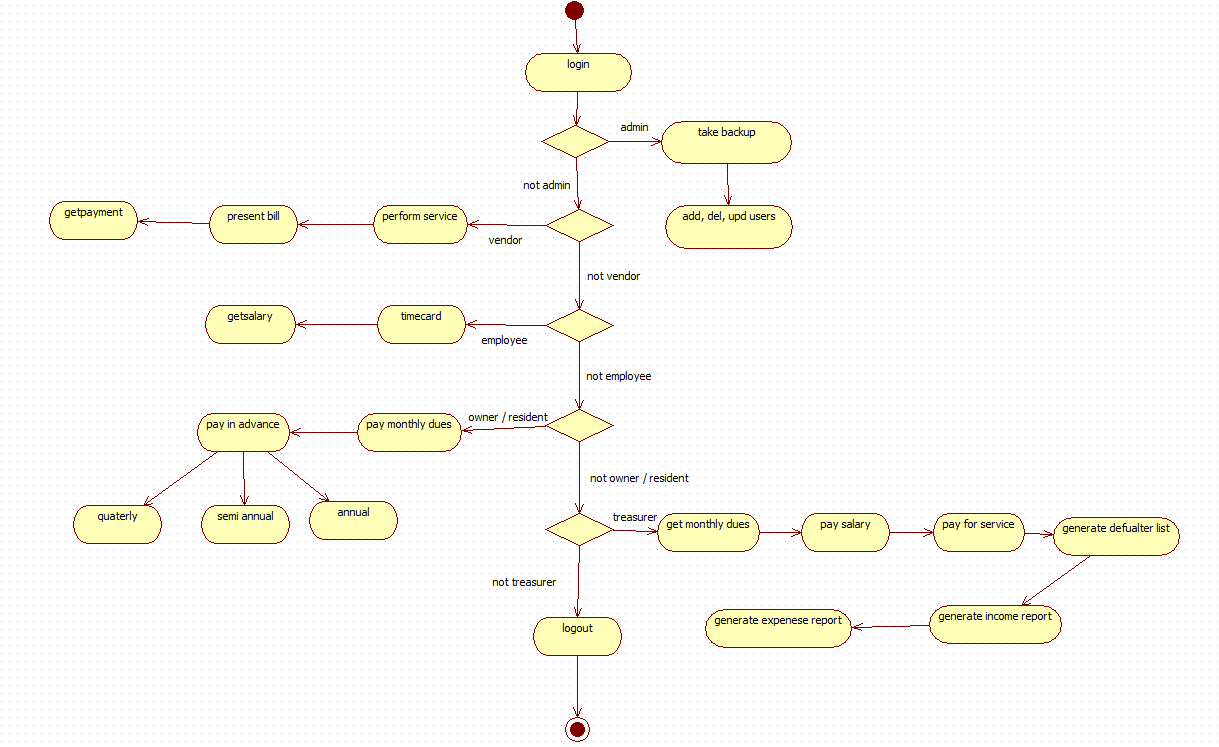
****

**Experiment 8: Complete Dynamic Model**

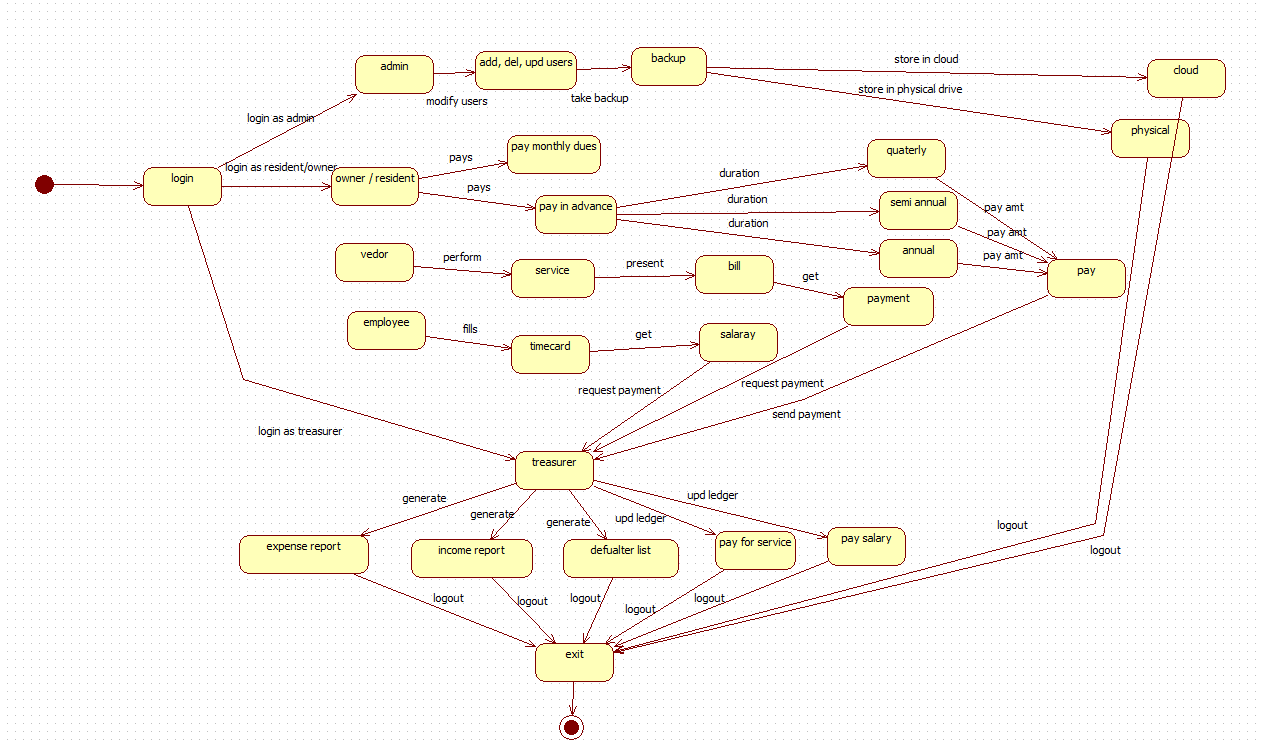
**Sequence Diagram:**

****

**Activity Diagram:**

****

**State Chart Diagram:**

****

**Experiment 9: Test cases**

