Tenant Management

System

**System Design Document – Version 1.0.0**

**Team Members**

**Muhammad Fateh Tariq**

**Muhammad Osama Syed**

**Zia Haider**

Date Created: **04/08/2019**

Table of Contents

[**Introduction** 3](#_Toc15845315)

[Purpose 3](#_Toc15845316)

[Scope 3](#_Toc15845317)

[Overview 3](#_Toc15845318)

[System Architecture 4](#_Toc15845319)

[Architectural Design 4](#_Toc15845320)

[Decomposition Description 5](#_Toc15845321)

[Design Rationale 5](#_Toc15845322)

[System Design 6](#_Toc15845323)

[Class Diagram 6](#_Toc15845324)

[Sequence Diagrams 7](#_Toc15845325)

[Component Design 12](#_Toc15845326)

[Requirements Matrix 14](#_Toc15845327)

# **Introduction**

## Purpose

This software design document describes the architecture and system design of **Tenant Management System** which is to be made for Umer Associates. The Design Document is intended for the Developers on the Team, the Testers and Designers.

## Scope

The main aim of the Software is to allow Effective Communication between the Tenants and the concerned personal of the Apartment Rental Organization and to make day to day tasks for the Organization and Tenants easier and more effective. The Software will help in Automation of the Organizations current system which will indefinitely help in solving the common problems the Organization and the Tenants faced every day.

## Overview

This Document contains the Purpose of this Document and Scope of this Project. It contains the Architectural Model that is needed to be adopted, the reasons why it is being chosen and some of the disadvantages that arise. The Document also contains the Sequence Diagrams which reflects upon how the sub systems within the Main System itself interact with each other to achieve a goal. There is a class Diagram which contains all the Attributes and Operations which work together to achieve a common goal. The Document also contains how each functional requirement is supposed to work by providing concise pseudo codes. Moreover, the Document contains a comparison of the functional requirements within the SRS and groups them into their respective Classes and Operations.

# **System Architecture**

## Architectural Design

The Architecture that is chosen is a 3 Tier Architecture, which will be developed based upon the Spiral Model.



Business Logic is divided into 8 sub-systems which work parallel to achieve their desired function. The Users interact with the Presentation Layer, which is the Interactive User interface. The Presentation layer then interacts with the various sub systems and the sub systems interact with the Database which stores all relevant information.

## Decomposition Description



The Sub Systems are responsible for managing various key aspects of the Software. Apartment Management manages the Apartments, Notification System is responsible for sending messages and files to different personal, Billing System manages Rent and Utility Bill collection and processing, Login/Registration System manages the login and Registration. Complaint Management manages complaints, while Maintenance and Inspection System manages everything related to scheduling Maintenance and Inspection. Security System manages Security Footages and Incident Reports, while the Database stores all relevant information and the Security Repository stores all cctv footages.

## 

## Design Rationale

The three tier Architecture gives you the ability to update the technology stack of one tier, without impacting other areas of the application. It allows for different development teams to each work on their own areas of expertise. Today’s developers are more likely to have deep competency in one area, like coding the front end of an application, instead of working on the full stack. You can scale the application up and out. A separate back-end tier, for example, allows you to deploy to a variety of databases instead of being locked into one technology. It also allows you to scale up by adding multiple web servers. It adds reliability and more independence of the underlying servers or services. It provides an ease of maintenance of the code base, managing presentation code and business logic separately, so that a change to business logic, for example, does not impact the presentation layer. However, the Three-tier architecture is not inherently flawed; it is just out of date. It was designed for software product development before smart phones and other mobile devices existed, when applications only needed to interact with one kind of entity at the presentation tier (the web browser running on the desktop). Its drawbacks derive not from the number of layers nor how data processing is distributed across them, but from the fact that the application is written as single, unified code base. The main drawbacks are that the Three-tier nature makes it difficult for developers to change an application with the agility and flexibility they need to keep pace with the expectations of mobile users, and for operations teams to scale the application up and down to match demand. A Three-tier design hampers agility at several phases of the application development process. Even if application functionality is distributed in a modular fashion, a change to any module requires rebuilding and testing the entire application. This can be quite time consuming. Nowadays, you need to roll out incremental improvements quickly and often to keep up with users’ thirst for ever-better performance and the latest cool new feature. The flexibility to choose from an array of solutions isn’t available with a 3-tier design, where solutions are typically built from a set of highly interdependent coupled components. However, since the organizational requirements don’t require fast incremental updates, it is much better to adopt this architecture.

# **System Design**

## Class Diagram



## Sequence Diagrams













# **Component Design**

**Tenant**

Check Apartment Availability ():

1. Request for list of available apartments in DB
2. Database Return Available Apartment List from DB
3. List is Displayed on Screen.

Request Lease Information ():

1. Gets apartment Lease number from user
2. Request the Apartment List from DB
3. DB Returns the Apartment list.
4. Matches the apartment Lease number with List of apartment Lease Information in DB.

Lease Information ():

1. Returns the Lease Information from database if matched otherwise display Error.

Request Lease Termination ():

1. Gets apartment Lease number from user
2. Request the Apartment List from DB
3. DB Returns the Apartment list.
4. Matches the apartment Lease number with List of apartment Lease Information in DB.
5. Calls the Lease termination Method if matched otherwise display Error.

**Staff**

Edit Apartment Info ():

1. Gets apartment number from user
2. Request the Apartment list from DB
3. DB Returns the Apartment list.
4. Matches the apartment number with List of apartments Information in DB.
5. Display Information of Respective apartment with Edit Information and Save Information Options.
6. Calls UpdateApartmentInfo() method to Save the edited Apartment information.

Process Apartment Change Request ():

1. Calls apartment change method to complete the request of tenant.

Process Tenant Registration ():

1. Gets All Necessary All Necessary Tenant Information.
2. Calls the Tenant Registration Method which stores Information in DB.

**Landlord**

listAvailableApartments ():

1. Request for list of available apartments in DB
2. Database Return Available Apartment List from DB
3. List is Displayed on Screen.

**Manager**

Send Eviction Notice ():

1. For specific tenant it will send notification using notification class
2. It will retrieve lease object and calls lease termination for that object.

Store Occupancy Verification ():

1. For specific tenant object it stores occupancy verification Certificate in DB.

# **Requirements Matrix**

|  |  |  |  |
| --- | --- | --- | --- |
| **S. R** | **Functional Requirement** | **Class** | **Method** |
| 1. | Schedule Visit | ProspectTenant | ScheduleVisit() |
| 2. | Send Rental Application | ProspectTenant | SendRentalApplication() |
| 3. | Check Room Availability | Apartment | getAvailableApartments() |
| 4. | Review incident Report | Security | getIncidentReport() |
| 5. | Check Payment Status | Check\_Payment | getStatus() |
| 6. | Review Security Footage | Security | getAllFootage() |
| 7. | Remove Security Footage | Security | removeFootage() |
| 8. | Request Maintainance | Apartment | addMaintainance() |
| 9. | Request Inspection | Apartment | addInspection() |
| 10. | Submit Feedback Form | Apartment | setFeedback() |
| 11. | Request Lease Information | Tenant | getCurrentLease() |
| 12. | Request Lease Termination | Tenant | setTermination() |
| 13. | Pay Rent | Payment | getRent() |
| 14. | View Notifications | Notification | getNotificationList() |
| 15. | Report Emergency | Security | reportEmergency() |
| 16. | Through Credit Card | Bill | setPayment() -> getRent() |
| 17. | Through Challan | Bill | setPayment() -> getUtilityBill() |
| 18. | View Complaints | Apartment | getComplaintList() |
| 19. | Add Complaint | Apartment | addComplaint() |
| 20. | Remove Complaint | Apartment | removeComplaint() |
| 21. | View Personal Information | Lease | getTenant() |
| 22. | Add Information | Lease | setTenant() |
| 23. | Pay Utility Bill | Bill | getUtilityBill() |
| 24. | Through Credit Card | Bill | setPayment() -> getUtilityBill() |
| 25. | Edit Apartment Info | Building | setApartment() |
| 26. | Process  Tenant Registration | Lease | setTenant() -> addRental() |
| 27. | Process Lease Termination | Lease | setTermination() |
| 28. | Process Apartment Change  Requests | Lease | setApartmentChange() |
| 29. | View Tenant Billing Info | Tenant | getBill() |
| 30. | View Rent Info | Bill | getRent() |
| 31. | View Utility Bill Info | Bill | getUtilityBill() |
| 32. | Send Notifications  to Tenants | Notification | sendNotification() |
| 33. | Add Custom Notifications | Notification | addNotification() |
| 34. | Renew Lease | Lease | addRenewal() |
| 35. | Record Regular  Maintenance | Apartment | addMaintenance() |
| 36. | Send Renewal Notice | Notification | sendNotification() |
| 37. | Schedule Apartment  Inspection | Apartment | addInspection() |
| 38. | Enter Inspection Results | Apartment | setInspectionResult() |
| 39. | Add Inspector | Apartment | addInspector() |
| 40. | Process Tenant Bills | Bills | setRent(), setUtilityBill() |
| 41. | Process Tenant Rent | Bills | setRent() |
| 42. | Process Tenant Utility  Bills | Bills | setUtilityBill() |
| 43. | Process Incident Report to  Manager | Security, Notification | addIncidentReport(), sendNotification() |
| 44. | Store  Occupancy Verification | Tenant | setOccupancyVerification() |
| 45. | Send Eviction Notice | Lease, Notification | setTermination(), sendNotification() |
| 46. | View Incident Report | Security | getIncidentReport() |
| 47. | Process Incident Report to  LandLord | Security, Notification | setIncidentReport(), sendNotification() |
| 48. | View Security Footage | Security | getFootage() |
| 49. | Enter Incident Report | Security | setIncidentReport() |
| 50. | Emergency Hotline | Security | reportEmergency() |
| 51. | Send Alert Notification  To Tenant | Notification | sendNotifcation() |