MASENO UNIVERSITY

SCHOOL OF COMPUTING AND INFORMATICS

DEPATMENT OF COMPUTER SCIENCE

CCS 406: COMPUTER SCIENCE PROJECT II

PROJECT TITLE: RENTAL HOUSE MANAGEMENT SYSTEM

REGISTRATION NUMBER	NAME
CI/00013/014	SOLOMON ALEKA

A project report submitted in partial fulfillment of the requirement for the Bachelor of Science Degree (BSc.) in Computer Science.

DECLARATION

I **Solomon Aleka** do hereby declare that this project proposal is my original work and to the best of my knowledge, it has not been presented to any other examination body.

Sign: ……………………… Date: ……………………

This project proposal is hereby presented for examination with the approval of the project supervisor.

Name: Mr. Michael Adongo

Signature: ………………………. Date: ………………………….

ABSTRACT

Nowadays, there is a lot of reliance on computerized applications which are used to solve a variety of problems encountered on a daily basis. One of these issues is that of managing rental houses. With the current paradigm shift in technological field, there is an urgent need to embrace and appreciate the power of technology. Housing sector remains vigilant to face the challenges of change caused by deployment of a new strategy that facilitates easy management of rental houses. There is great need to develop a RHMS that can simplify work for agencies that manage rental houses so that all the work that has to be done is done in a more efficient manner thereby reducing the frequently encountered errors and faults. In the process of acquiring information on how rental houses are currently managed, I prepared questionnaires and submitted them to a number of RHMs and from the feedbacks that I gathered, I realized that a lot of paper work was involved in managing the rental houses. Papers can easily get damaged or lost thereby leading to loss of data. It is also relatively expensive to keep on buying files to store your records. A lot of files not only make a place untidy but also consume space. Locating the appropriate file from many files becomes a tedious task. In consideration of the aforementioned problems associated with manual management of rental houses, I am proposing the Rental House Management System as the solution. The system is expected to simplify the operations of RHs management agencies.

ACKNOWLEDGEMENT

First and foremost, I would like to thank my project supervisor Mr. Michael Adongo. His input towards completion of the project is remarkable. He was very keen on mistakes and promptly corrected me when I did something wrongly or failed to comply by any of the set standards.

Secondly, I would like to thank my parents. They were key in handling the financial aspect to see that the project is completed on time. Moreover, they provided their support and conducive environment apt for serious working and learning.

Thirdly, I would also like to thank the experts who provided their advice on various aspects related to management of rental houses. I remain greatly indebted to them. Their contribution in the project proposal has been of great value to me.

Last but not least, I would also like to thank the faculty of Computing and Informatics for providing a conducive environment for learning as well as availing wireless internet connectivity that greatly aided me during the research process.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE: INTRODUCTION

1.1 Background of the study

Housing has a central importance to quality of life with considerable economic, social, cultural and personal significance. Though a country's national prosperity is usually measured in economic terms, increasing wealth is of diminished value unless all can share its benefits and if the growing wealth is not used to address growing social deficiencies one of which is housing, then prosperity will not have been fully attained. Housing plays a huge role in revitalizing economic growth in any country, with shelter being among key indicators of development.

The universal declaration of human rights gives one of the basic human rights as the right to a decent standard of living central to which is the access to adequate housing. The focus of this research project is basically managing housing for low, medium and high-income households. Most families choose to rent houses based on their income and family situations. Unfortunately, there may not be enough good quality rental housing for those families that purpose to pay less.

Housing is a major problem in Kenya especially in Nairobi city. Millions of people are living in sprawling slams and also in other informal settlements around Nairobi. This explains why many people have shifted their focus to developing rental houses in Nairobi and other parts of the country to address the shortage. The demand for rental houses is extremely high and more rental houses need to be put in place. Developing rental houses comes with many advantages especially to the landlords who are able to generate revenue through rent paid by the tenants. However, with increasing number of tenants, management becomes difficult for landlords who are no longer able to efficiently able to manage the tenants and therefore they end up resorting to the services of RH managing agencies. The RH managing agencies have to manage several rental houses and this calls for adoption of a system that ensures all operations are done effectively and efficiently thus the need for automation of the whole process.

1.2 Problem statement

Over the years, landlords and property managers have been facing impediments in the process of managing their tenants and processing their rental houses related requests. Some of the issues that have made manual management of rental houses difficult are:

a. Rapid data growth. Data increases on a daily basis and it therefore becomes difficult to locate specific information from the files due to their voluminous nature. This is not only time consuming but the files consume a lot of space.

b. Lack of proper record keeping. Due to manual file keeping, the files are susceptible to errors which when not keenly monitored can lead to devastating consequences.

c. Uncertainty on data security. Unauthorized personnel can access records stored in files and manipulate them thus making them incoherent and not a true reflection of the operations that took place. This lack of data confidentiality is due to lack of mechanisms to ensure that access to the files is well controlled and that instances of unauthorized access are minimum.

d. Lack of a centralized storage location. All those who wished to have a look at the records have to physically travel to the location where the files are stored and this further contributed to time wastage.

e. Lack of a proper channel to advertise vacant houses. The channels that are used in advertising vacant houses are not effective in that not many people are reachable and this consequently increases the length of time taken to locate a prospect tenant and ultimately settling him/her.

The above impediments justify the need to automate management of rental houses.

1.3 Purpose of the study

The purpose of the study is to devise a web-based application that will integrate key operations in RH management into a unified powerful software tool. The proposed application aims at reducing the time taken to locate a prospect tenant as well as the paperwork involved in formalizing tenancy thus relieving both the prospect tenants and the managing agency the burden of taking too long to settle a tenancy. The proposed application will be accessible by prospect tenants, tenants, landlords as well as the staff and the managers from the managing agencies that will deploy the system. A prospect tenant will be able to have a glimpse of some of the services offered by the application.

For instance, he/she will be in a position to know the vacant houses. A tenant will be able to perform a number of operations. For instance, he/she may make a room change request or request maintenance services. The managing agency on the other hand will be in a position to perform a wide range of operations remotely. The staff under the managing agency will be able to grant or revoke services depending on the prevailing circumstances. In short, the proposed system upon completion will automate almost all rental house related operations without either time or distance been a hindrance.

1.4 Objectives

The objectives of the proposed application will be broadly categorized into two:

The general/main objective:

To develop an application that will automate most of the operations involved in initiating, settling, maintaining and ultimately terminating a tenancy.

Some of the specific objectives of the proposed system will be:

a. To economize on space by doing away with the bulky file storages.

b. To reduce the amount of time taken to locate a prospect tenant.

c. To provide a centralized storage for all the data pertaining to management of the rental houses thus ensuring access from any location with internet connection.

d. To improve on data security by introducing different login modules bearing different levels of privileges thus ensuring crucial data can only be accessed by authorized personnel thereby ensuring both confidentiality and integrity of data in the database is upheld.

e. To increase efficiency in processing requests by reducing the amount of time taken by the tenants to air their issues to the management.

f. To effectively control rent payment.

g. To reduce the error rate associated with manual file storage mechanisms

1.5 Significance of the study

Impacts that will be realized upon implementation of the proposed application are:

a. The amount of time taken in locating a prospect tenant will be greatly reduced. This will be made possible in that the staff within the MA will be charged with the task of frequently updating the application with details of vacant rooms so as to attract prospect tenants from whichever location.

b. The amount of time taken for a tenant to make his/her petition or complaint known to the management will be reduced. From whichever geographical location, as long as the tenant can access internet, he/she will be able to fill the necessary form and submit to the managing agency which will in turn respond promptly.

c. Data integrity and confidentiality will be improved to a large extent. Unlike the manual records stored in files which could be easily manipulated, data stored in the database of the proposed system will be password protected thus ensuring only authorized users are allowed to access it. Users will be granted different access levels with a prospect tenant having the lowest access level while the manager (system administrator) will be granted the highest access level.

d. Generation of reports will also be simplified. The trained staff in the MA will only have to inject an appropriate query and the desired results are automatically generated from the database. Moreover, the speed involved in the generation of the reports will increase as the task won't be done manually- a method susceptible to errors and mistakes.

e. There will be reduction in the amount of paper work handled by the MA. Records that would have taken several files to be stored will be conveniently stored in the system database taking very little space. In short, availability of space for data storage is guaranteed as the number of houses under a given managing agency grows.

1.6 Scope and Limitation of the study

1.6.1 In scope

RHMS will include prospect tenants and tenants as well as their requests and obligations. Apart from that, RHMS will include provision of maintenance services such as gardening of flowers, roof improvement and central heating. Also, RHMS will incorporate contract management as new tenant contract, current tenant renewals. It also includes requests and reports from the managers to the central administration and service contract from the central administration to the managers. Payment will be accomplished through a payment API.

1.6.2 Out scope

Rental House Management System (RHMS) will not include a central accounting system. Financial reports will be periodically collected from the responsible bank.

1.7 Basic assumptions

Some of the assumptions that will be made while designing the proposed system are:

a. Rent is due on the 5th day of every month. Tenants who will fail to comply will be surcharged as agreed.

b. Inspection of the rental houses will be done prior to the tenant vacating or as scheduled by the management. Inspections will be performed by a third party referred to as the inspector.

c. After an inspection has been conducted, the rental houses must be found to be up to code and if that is not the case then the landlord and the tenant in question must discuss who will be liable for the repair costs.

d. Regular maintenance will be performed on the rental houses upon request from the tenant or when need arises.

e. That the application will be deployed for use by an agency managing rental houses but not an individual landlord because the system will be designed in a manner such that critical managerial operations are entrusted to the managing agency.

1.8 Operation definition of terms

Definition of terms as used in the context of Rental House Management System:

1. SDLC: System Development Life Cycle.
2. DFD: Data Flow Diagram.
3. RHM: Rental House Manager
4. MA: Managing Agency
5. RHMS: Rental Houses Management System.
6. RH: Rental House.
7. TSS: Tenant Self Service
8. API: Application Programming Interface

CHAPTER TWO: LITERATURE REVIEW

2.1 Introduction to the Literature Review

Rental housing is a common phenomenon in most countries of the world whether developed or a developing. In most developing countries, demand for rental houses has increased over the past decades, but rental supply has not been able to keep pace with the demand. The objective of this chapter is to review literatures relevant to critical issues relating to rental housing from the international context in order to draw some lessons for rental housing system in Kenya. Some of the critical issues that will be handled under this chapter are summarized below:

a. The role of actors in the provision of rental housing.

b. Rent policy.

c. Rent level and rent setting.

d. Rent subsidy.

2.2 Literature Reviews on Topics Related to the Project

2.2.1 The Role of Actors in the Provision of Rental Housing

There are various actors involved in rental housing businesses. They are the landlords, tenants and sharers. The named actors and their roles are described in detail below.

Landlords

The supply side of most rental markets is characterized by the landlords. There are private landlords who are mostly small-scale investors and own one to a few structures. The other type of landlord is the institutional landlord which would include central government, local government, public enterprise, corporation, pension funds, investment bank or insurance company. All landlords let property to obtain an income but the reasons they do so and the strategies that they follow in generating that income are highly diverse. Some landlords do it as a commercial exercise and we can call them professional. The others are not letting for commercial purposes and we therefore call them non-professional landlords. For professional landlords, the rate of return or the amount of profit realized is the main objective. The small scale ones not only think of rate of return but also want to take any incentives that may be on offer from the government and minimize their taxable income. For non-professionals, they will certainly not be able to say whether their housing investment matches the potential returns from other activities.

Tenants

Tenants are usually families who are staying in a dwelling unit with a condition of paying a certain amount of money regularly. The characteristics of the tenant population depend fundamentally upon the nature of the local housing market. In certain cities one kind of household may own whereas in others the same kind of households may rent. A household's decision rests on the relative costs of renting versus owning and upon the relative size of the rental housing stock.

2.2.2 Rent Policy

It has been a goal of many governments to keep housing affordable for people on low incomes and this goal could be achieved through drafting and enacting of rent policies. In Netherlands, rents and rent increases are subject to rules. This applies to both the private sector and the social sector. However, this is not applicable to many other countries. To describe in detail rent policy, it has been broadly categorized into rent level and setting and rent subsidy.

2.2.3 Rent Level and Rent Setting

As housing is a human right, with reference to the Universal Declaration of Human Rights and the UN International Covenant on Economic, Social and Cultural Rights, rents have to be set at an affordable level. Landlords always tend to argue that rents are too low, tenants say that they are too high. Both sides are likely to abuse the statistics. As such, it is rarely easy to establish the truth. Part of the problem about rent levels relates to what is meant by rents being 'too high'. Sometimes, rents make up a very high proportion of the tenants' income but are nonetheless still too low for landlords to make a profit or even pay for maintenance of the property. Rents should be in a reasonable proportion to the income.

2.2.4 Rent Subsidy

The rent subsidy is a housing benefit and intended for people who are not able to pay the housing costs for their rental dwelling. It is part of an enabling approach to housing in which the goal is to create a well-functioning housing sector that serves the needs of all key stakeholder groups. In many countries, in order to be considered for rent subsidy, regulations are used that have to do with income, age, personal means and the rental price.

2.3 Review on System Development Methods

2.3.1 Structured Analysis

Represents the system in terms of data and the processes that act upon the data. System development is organized into phases, with deliverables and milestones to measure progress. The SDLC waterfall model typically consists of five phases and iteration is possible among the phases. Data Flow Diagrams and business process modelling are the modelling tools used in this method. This is a traditional method which has been very popular over time. The method relies heavily on written documentation. Frequent phase iteration can provide flexibility comparable with other methods. Well-suited to project management tools and techniques. Changes can be costly especially in later phases. Requirements are defined early and can change during development. Users might not be able to describe their needs until they see examples of features and functions.

2.3.2 Object-Oriented Analysis

Views the system in terms of objects that combine data and processes. The objects represent actual people, things, transactions and events. Compared to structured analysis, Object-Oriented phases tend to be more interactive. Can use the waterfall model or the model that stresses greater iteration. Various object-oriented diagrams depict system actors, methods and messages. Like structured analysis, it uses business process modelling. Integrates easily with object-oriented programming languages. Code is modular and reusable which can reduce cost and development time. Easy to maintain and expand as new objects can be cloned using inherited properties. Interactions of objects and classes can be complex in larger systems.

2.3.3 Agile/Adaptive Methods

Stresses intense team-based effort. Breaks development process into cycles or iterations that add functionality. Each iteration is designed, built and tested in an ongoing process. Attempts to reduce major risks by incremental steps in short time intervals. Typically uses the spiral model. Uses modelling tools that enhance communication such as collaborative software, brainstorming and whiteboards. Business process modelling also works well with agile methods. Very flexible and efficient in dealing with change.

Stresses team interaction and reflects a set of community-based values. Frequent deliverables constantly validate the project and reduce risks. Team members need a high level of technical and communication skills. Lack of structure and documentation can introduce risk factors. Overall project might be subject to scope change as user requirements change.

2.4 Review on Several Software Methodologies

A software development methodology is a collection of procedures, techniques, tools, and documentation aids which will help the systems developers in their efforts to implement a new information system. There are a number of software development methodologies each of which is adopted based on a number of factors relating to the project for example, time, cost, incorporation of requirement changes during the development process, system complexity, communication between customers and developers, software criticality, size of the development team. These generic models are not definitive descriptions of software processes but they are abstractions of the process that can be used to explain different approaches to software development. You can think of them as process frameworks that may be extended and adapted to create more specific software engineering processes. Below are a selected number of models:

2.4.1 The Waterfall Model

The waterfall model is a sequential design process, often used in software development processes. It takes the fundamental process activities of specification, development, validation, and evolution and represents them as separate process phases such as requirements specification, software design, implementation, testing, and so on.

2.4.2 Incremental Development

This approach interleaves the activities of specification, development, and validation. The system is developed as a series of versions (increments), with each version adding functionality to the previous version.

2.4.3 Reuse-Oriented Methodology

This approach is based on the existence of a significant number of reusable components. The system development process focuses on integrating these components into a system rather than developing them from scratch.

CHAPTER THREE: METHODOLOGY

3.1 Introduction to methodology

The Rental House Management System was designed using several development tools. They include Hypertext Markup Language (HTML5), Cascading Style Sheets (CSS), Hypertext Preprocessor (PHP), a server-side scripting language and very powerful for making dynamic and interactive web pages, MySQL (Structured Query Language), the most popular Open Source Relational SQL database management system, JavaScript and Ajax

3.2 Development Tools

This part of the chapter is essentially an account of some of the technologies that I used in implementing the system

3.2.1 Front-End Technology

Front-end is a term used to characterize program interfaces and services relative to the initial user of these interface and services. It usually refers to the client side of an application. A front-end application is one that users interact with directly. Front-end can also be defined as the portion of an e-seller's portal, electronic catalogs, a shopping cart, a search engine or a payment gateway.

HTML

Hypertext Markup Language (HTML) is a computer language devised to allow website creation. These websites can then be viewed by anyone else connected to the Internet. It is relatively easy to learn, with the basics being accessible to most people in one sitting; and quite powerful in what it allows you to create. Having the basic knowledge of HTML, could help make or develop m-commerce websites and will also prove to be handy especially for editing and modifying web pages.

JavaScript

JavaScript is a scripting language that is browser based and was developed by Netscape to enable web masters or web authors to add interactivity and enhance behavior of web pages. Some of the dynamic behavior that can be generated by JavaScript is validating form, performing specific actions after a mouse click, adding timestamps etc. JavaScript is an open language and anyone can use it. It also shares many features and structures of the Java programming language, though it is not really related to Java. It was developed independently.

CSS

CSS is a style sheet language used to describe presentation and layout of HTML tags. CSS is used to enable separation of document content from document presentation. This refers to the document presentation aspects such as colors, layouts and fonts from the actual document content.

Ajax

AJAX, is a web development technique for creating interactive web applications. AJAX stands for **A**synchronous **Ja**vaScript and **X**ML. AJAX is a new technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS, and Java Script. Ajax uses XHTML for content, CSS for presentation, along with Document Object Model and JavaScript for dynamic content display. Conventional web applications transmit information to and from the server using synchronous requests. It means you fill out a form, hit submit, and get directed to a new page with new information from the server.

jQuery

jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way millions of people write JavaScript.

3.2.2 Back-End Technology

PHP

PHP is an HTML embedded scripting language. Much of its syntax is borrowed from C, Java and Perl with a couple of unique PHP-specific features thrown in. The goal of the language is to allow web developers to write dynamically generated pages quickly." This is generally a good definition of PHP. However, it does contain a lot of terms you may not be used to. Another way to think of PHP is a powerful, behind the scenes scripting language that your visitors won't see.

MySQL

MySQL stands for My Structured Query Language. It is the world's most popular open source relational DBMS. MySQL is available for free under the GNU General Public License for open source benefits/reasons related to development. Initially MySQL was free and some versions of it are still free though if you desire to use MySQL for commercial purposes you will need to purchase a license. It is non-proprietary, easily extensible and platform independent. Its downside is that it lacks a graphical user interface and therefore you need to know how the database works to make the most efficient use of it. I have decided to use MySQL embedded on the WampServer. The properties of the WampServer are:

- Apache Version: 2.4.27
- PHP Version: 7.19
- MySQL Version: 5.7.19

CHAPTER FOUR: ANALYSIS AND DESIGN

4.1 Introduction to Analysis and Design

This chapter gives a detailed outline of the software development methodology that will be used in implementing the proposed project after close consideration of the various existing software development methodology discussed in literature review. The strengths and weaknesses of the chosen methodology have been outlined. Further, the user, the functional and non-functional requirements of the system are explained in detail and a variety of UML diagrams are also discussed. One such example is the use cases which are a list of steps, typically defining interactions between a role and a system, to achieve a goal.

4.2 Requirements Analysis

Requirement analysis will involve defining customer needs and objectives in the context of planned customer use, environments and identified system characteristics so as to determine requirements for system functions.

4.2.1 User Requirements

It will entail user involvement and statements of facts and assumptions that define the expectations of the system in terms of mission objectives, environment, constraints and measures of effectiveness and suitability. Basically, the users will expect:

i. A system that improves the efficiency of information storage and retrieval.

ii. A system that is easy to learn and use.

iii. A system that is fast in processing instructions.

iv. A system that is flexible, safe and convenient.

4.2.2 Functional Requirements

Statement of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations. May state what the system should not do. They include the following:

i. **Room management.** A prospect tenant will be able to make a request so as to get details of a given rental house before renting out a room. A tenant will be able to make a request of apartment change. After submission of the apartment change request, either the manager or any staff member can process and consequently accept or reject the request.

ii. **Filing complaints.** A tenant will be able to file a complaint as a way of illustrating dissatisfaction or breach of a policy on the management part.

iii. **Lease management.** A tenant will be able to request lease termination before tenancy duration elapses depending on the prevailing circumstances for example, need to vacate to another residential place. Apart from that, the tenant will be able to renew his/her lease even after the tenancy elapses so as to extend the stay in that residential place. On the other hand, the manager will be able to terminate lease of tenants once the lease period expires or before expiring of the lease duration. Also, the manager will approve lease renewal requests from tenants based on the available history pertaining to that particular tenant.

iv. **Adding a new tenant.** Either the manager or any member of the staff will be able to add a new tenant into the system and thereby manage his/her contract.

v. **Adding a new rental house.** Either the manager or any member of the staff will be able to add a new tenant into the system and thereby manage his/her contract. Apart from addition of a new rental house, the management can also edit information of the available rental houses.

vi. **Notice management.** The manager or any staff member will be able to send various notices to tenants. For instance, informing them of an upcoming maintenance or inspection operation. However, only the manager will be able to send an eviction notice to a tenant. Under normal conditions this will precede termination of the lease even if the lease period has not expired.

vii. **Updating site.** The manager or any staff member will be in a position to post any relevant new information. For example, new services, updates details pertaining to a given rental house, tax rates, etc.

viii. **Inspection management.** The manager or any staff member will be able to schedule an inspection and inform tenants promptly. The tenant will also be able to request an inspection probably before he/she vacates the room(s). Only the manager will bear the privilege of hiring a new inspector. At the end of an inspection operation the inspector will be expected to prepare an inspection report.

ix. **Maintenance management.** A tenant will be able to request maintenance operation on a given section of the rental house probably after noting reluctance from the management or after an accident that causes wreckage. However, in cases where the need for maintenance is necessitated due to damage caused by the tenant maliciously, the tenant will incur the repairing expenses. A maintenance operation will also be scheduled d by either the manager or any staff member as a way of repairing worn out structures or replacing them.

4.2.3 Non-functional Requirements

These are requirements on usability, reliability, performance, supportability, security, recovery, interface, implementation, operation, and legal. They include the following:

i. The proposed system will bear both the front-end and the back-end aspects. The front-end will essentially consist of a set of interfaces (GUI) while the back-end will constitute the RDBMS.

ii. Generally, the interfaces will bear some uniformity menus will be organized in a hierarchical manner (usability).

iii. The proposed system will be password-protected (security).

The Rental Houses Management System will be backed up daily. This will aid in data recovery in case of data loss. (Back up).

iv. There will be no limit on the number of users that can be added to the system.

v. The application will be accessed as long as there is internet connectivity.

4.3 Software Development Methodology Selected

After close consideration of the various software methodologies discussed in chapter 2, I have settled on using the Incremental Software Development Methodology. The reasons for my selection are listed below:

i. It will allow for development of high risk or major functions first. Customers will be able to use and gain value of the product earlier compared to using a waterfall model.

ii. Each release will be delivering an operational step.

iii. Customer will be able to respond to each build. It will be easier to get customer feedback on the development work that has been done because customers will be able to comment on the demonstrations of the application and see what has been implemented.

iv. The methodology will use “divide and conquer” breakdown of tasks.

v. The initial delivery cost will be lowered. Moreover, the initial product delivery will be faster.

vi. Customers or the end-users will be in a position to get important functionality faster.

vii. The risks involved while changing requirements will be significantly reduced.

4.4 System Design

4.4.1 Data Flow Diagram (DFD)

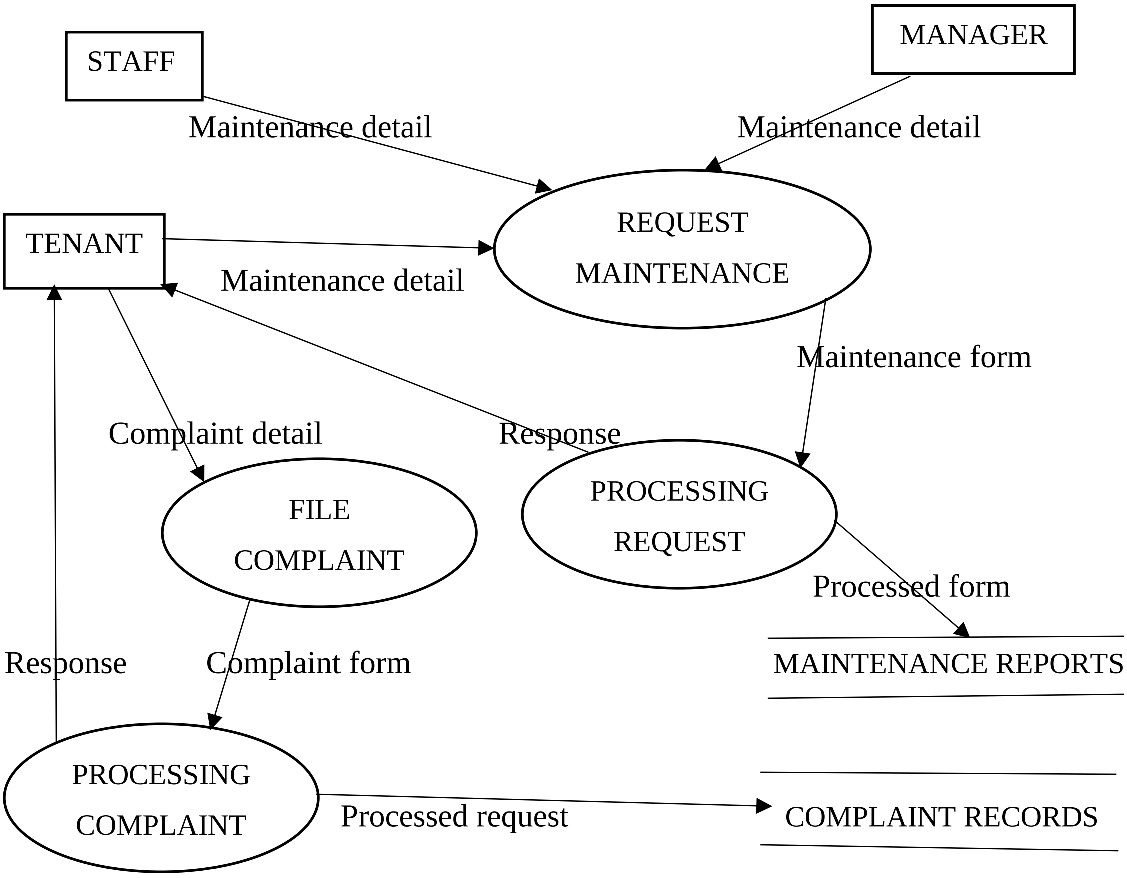


Figure 1: Complaint and Maintenance Processing DFD

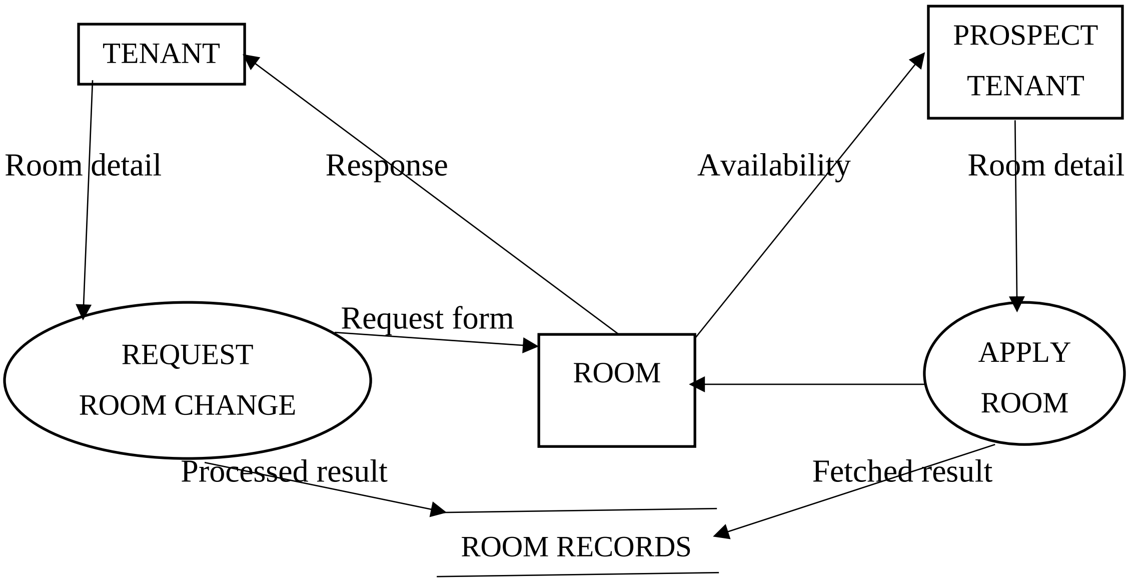


Figure 2: Room Change and Room Application Processing DFD

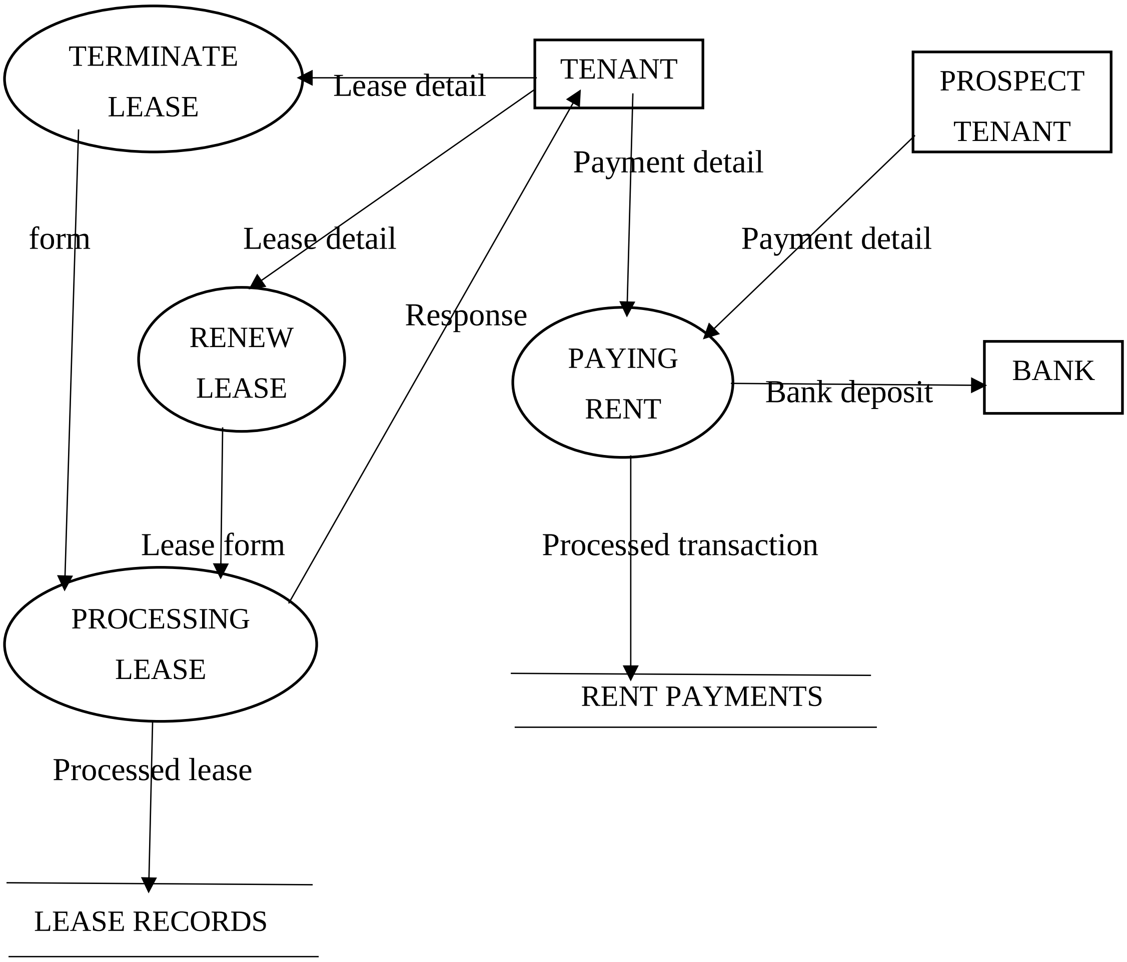


Figure 3: Lease Processing and Rent Processing DFD

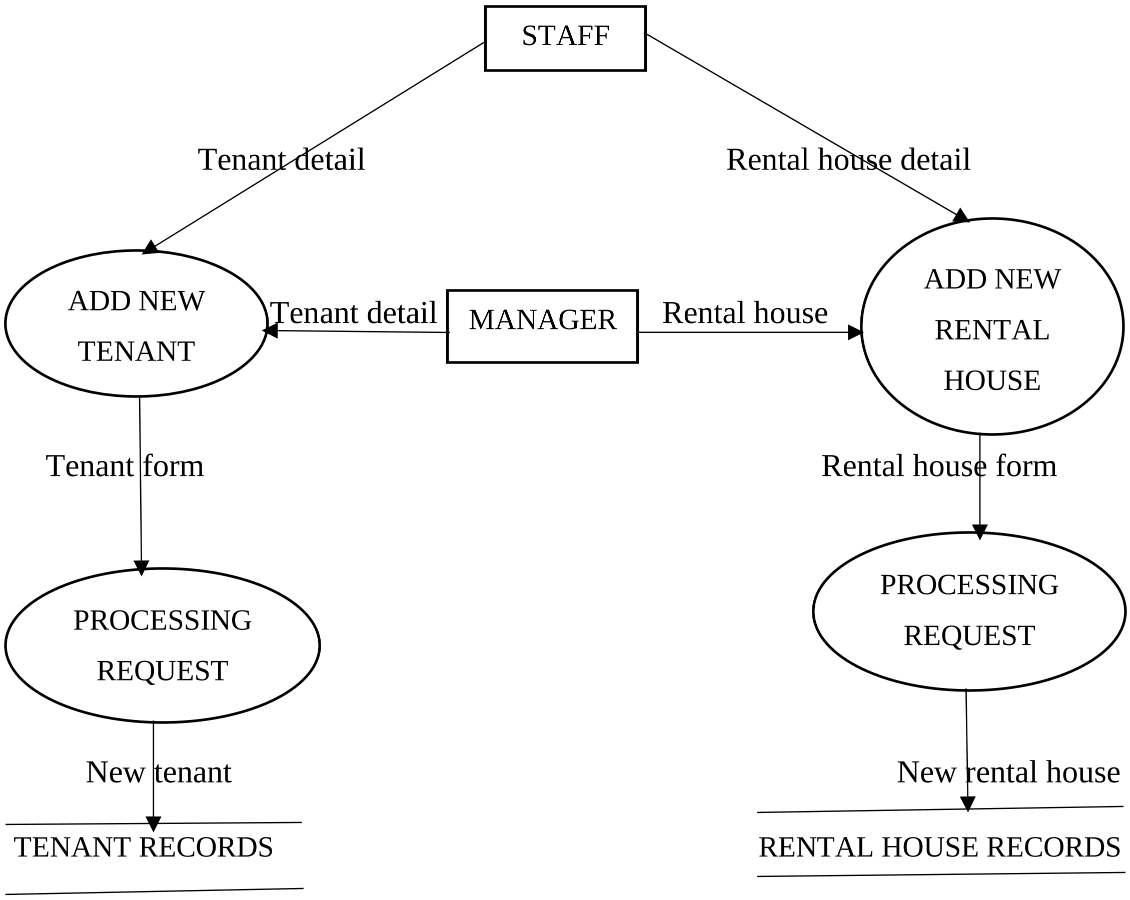


Figure 4: Adding Tenant and Adding Rental House DFD

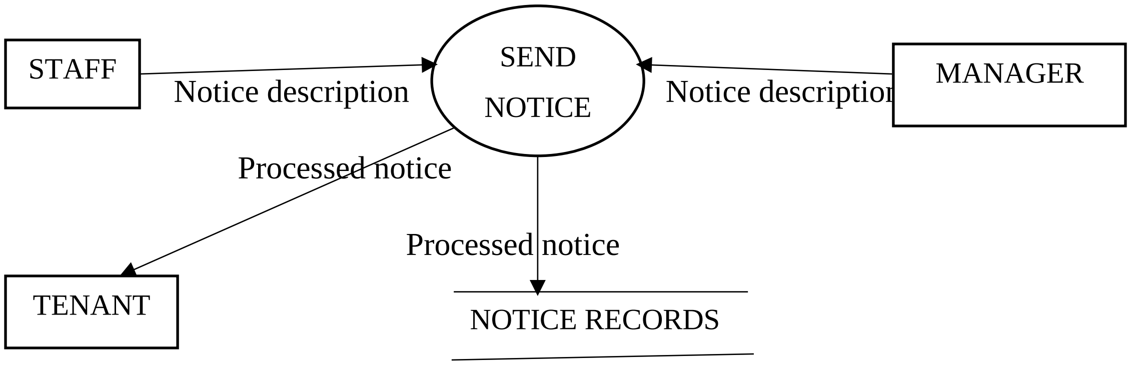


Figure 5: Sending Notice DFD

4.4.2 Activity Diagrams

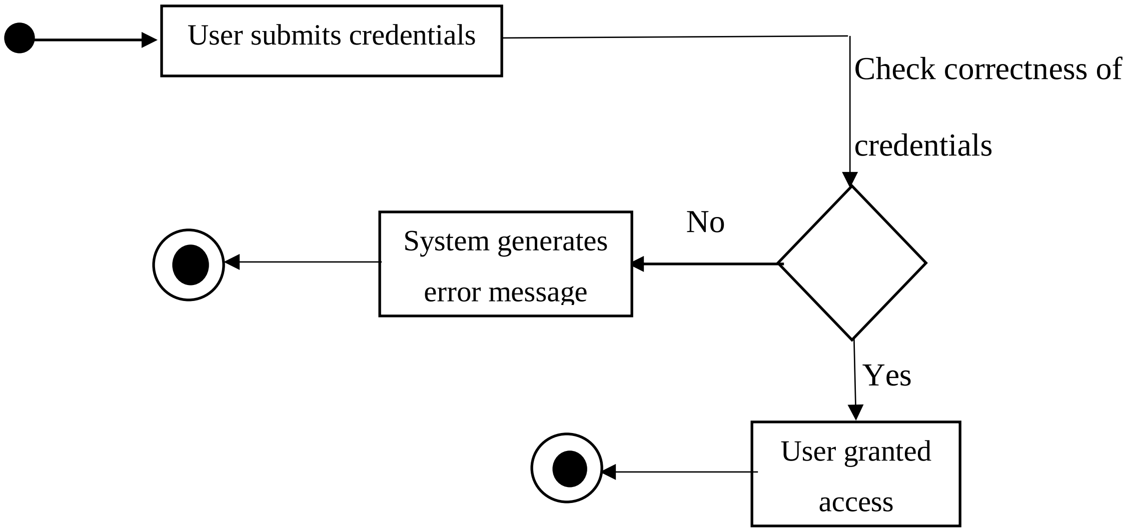


Figure 6: Login Activity Diagram

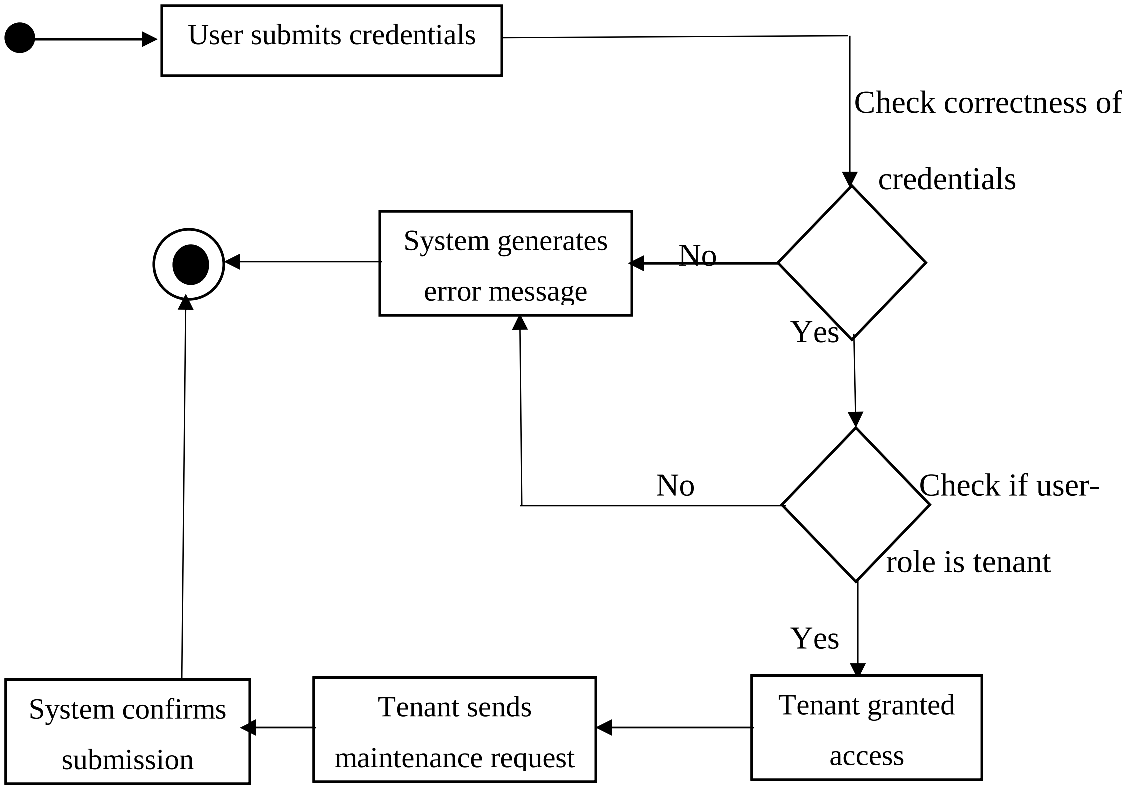


Figure 7: Maintenance Request Activity Diagram

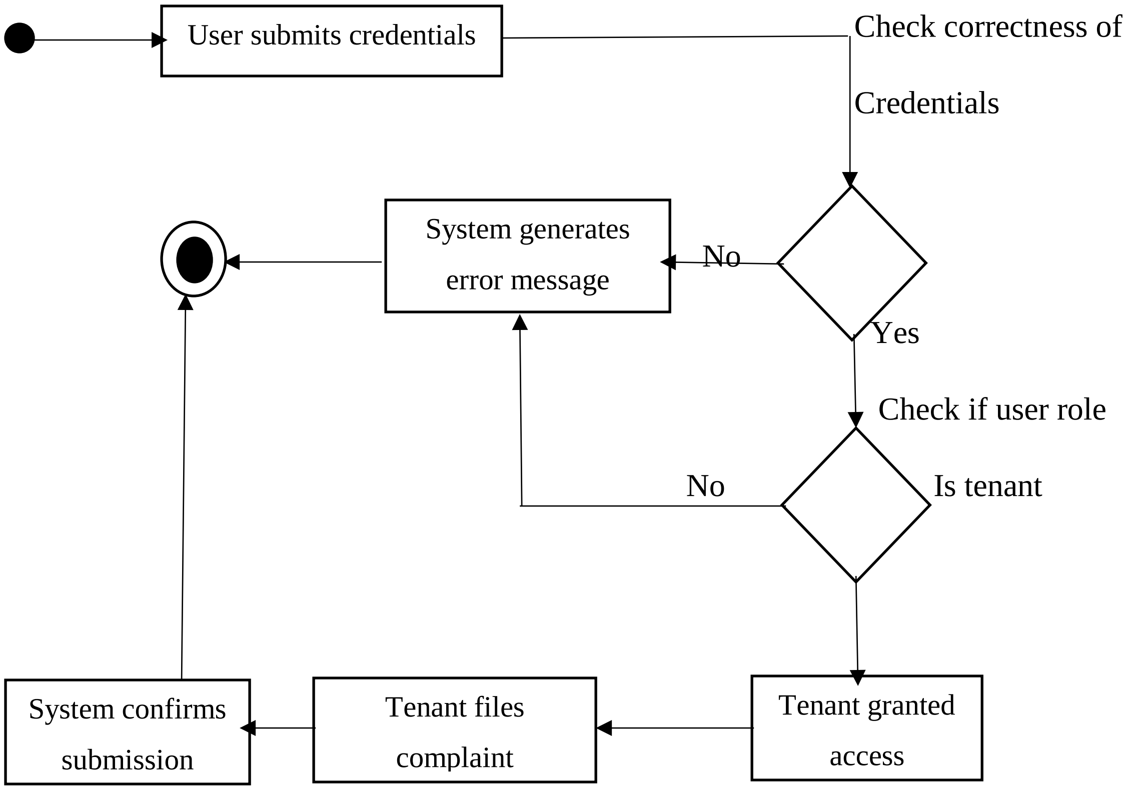


Figure 8: Complaint Submission Activity Diagram

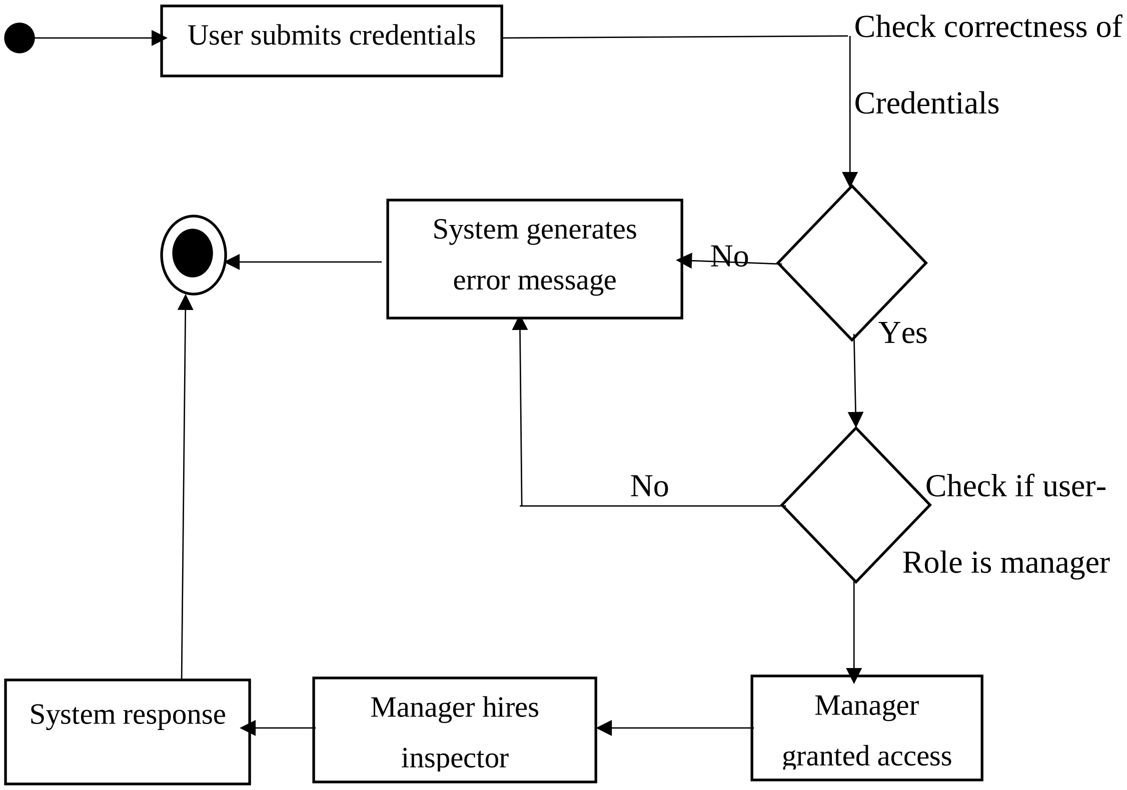


Figure 9: Inspector Hiring Activity Diagram

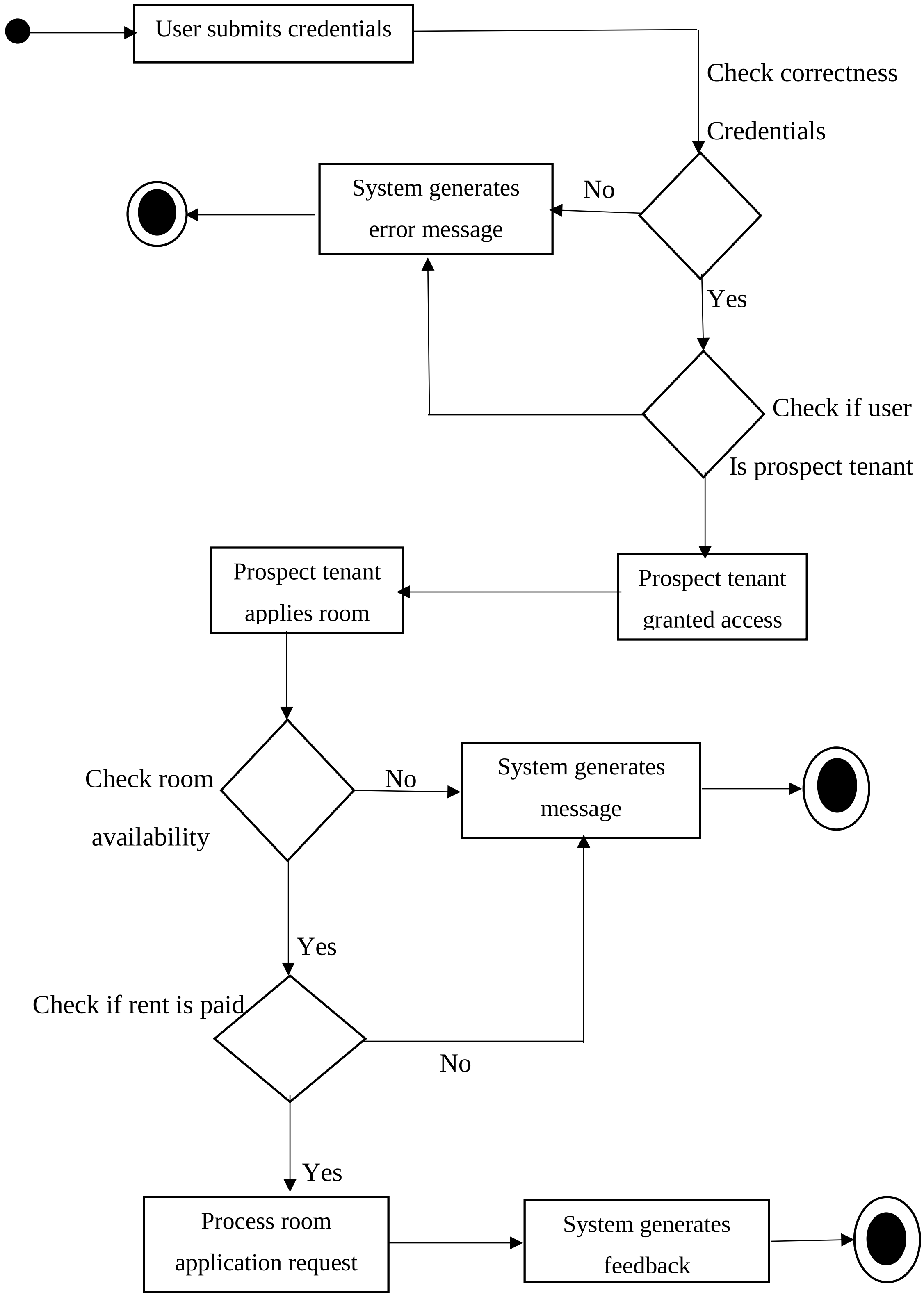


Figure 10: Room Application Activity Diagram

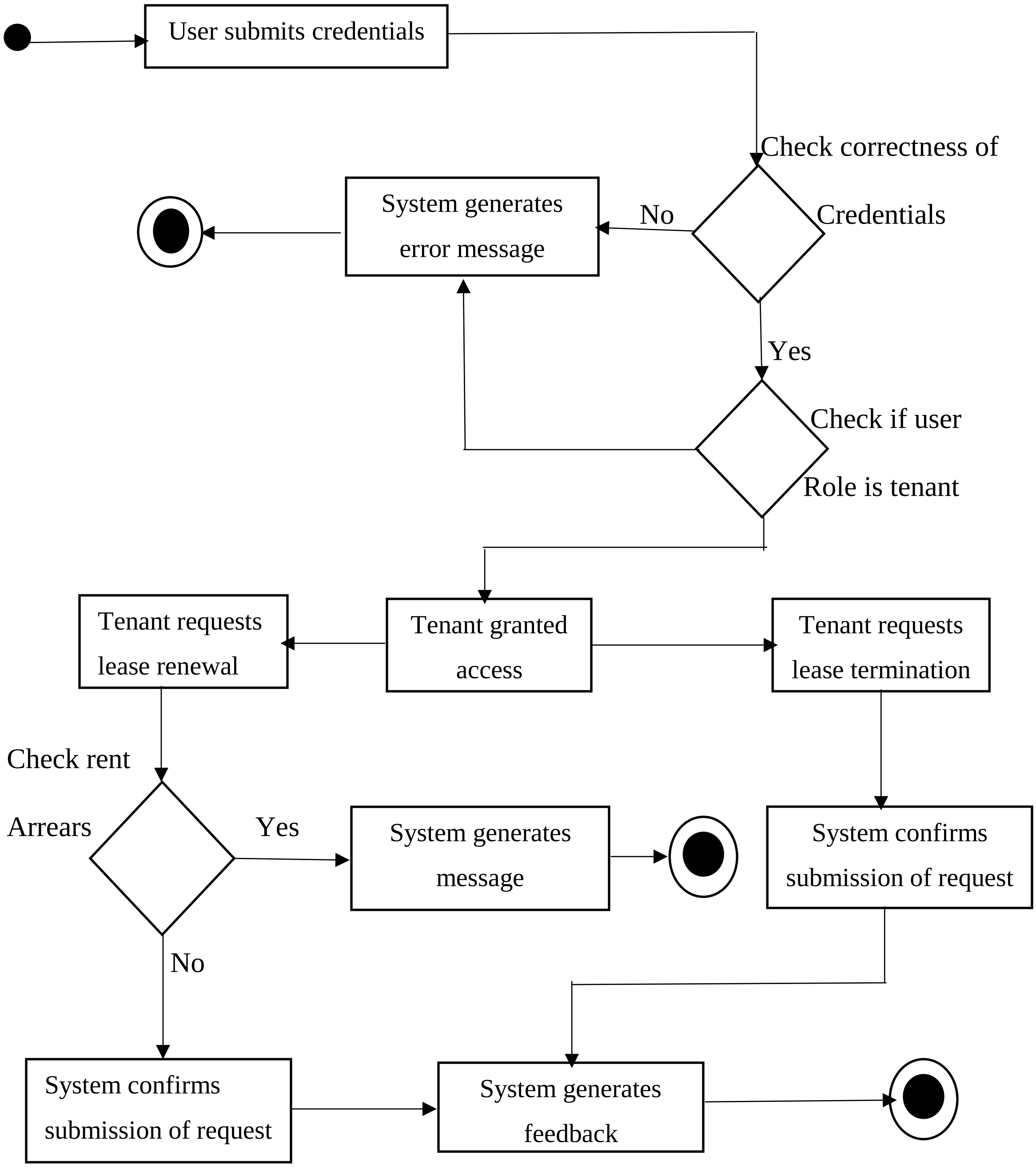

User submits credentials
Check correctness of
Credentials
No
System generates error message
Yes
Check if user
Role is tenant
Tenant requests lease renewal
Tenant granted access
Tenant requests lease termination
Check rent
Arrears
Yes
System generates message
System confirms submission of request
No
System confirms submission of request
System generates feedback

Figure 11: Lease Processing Activity Diagram

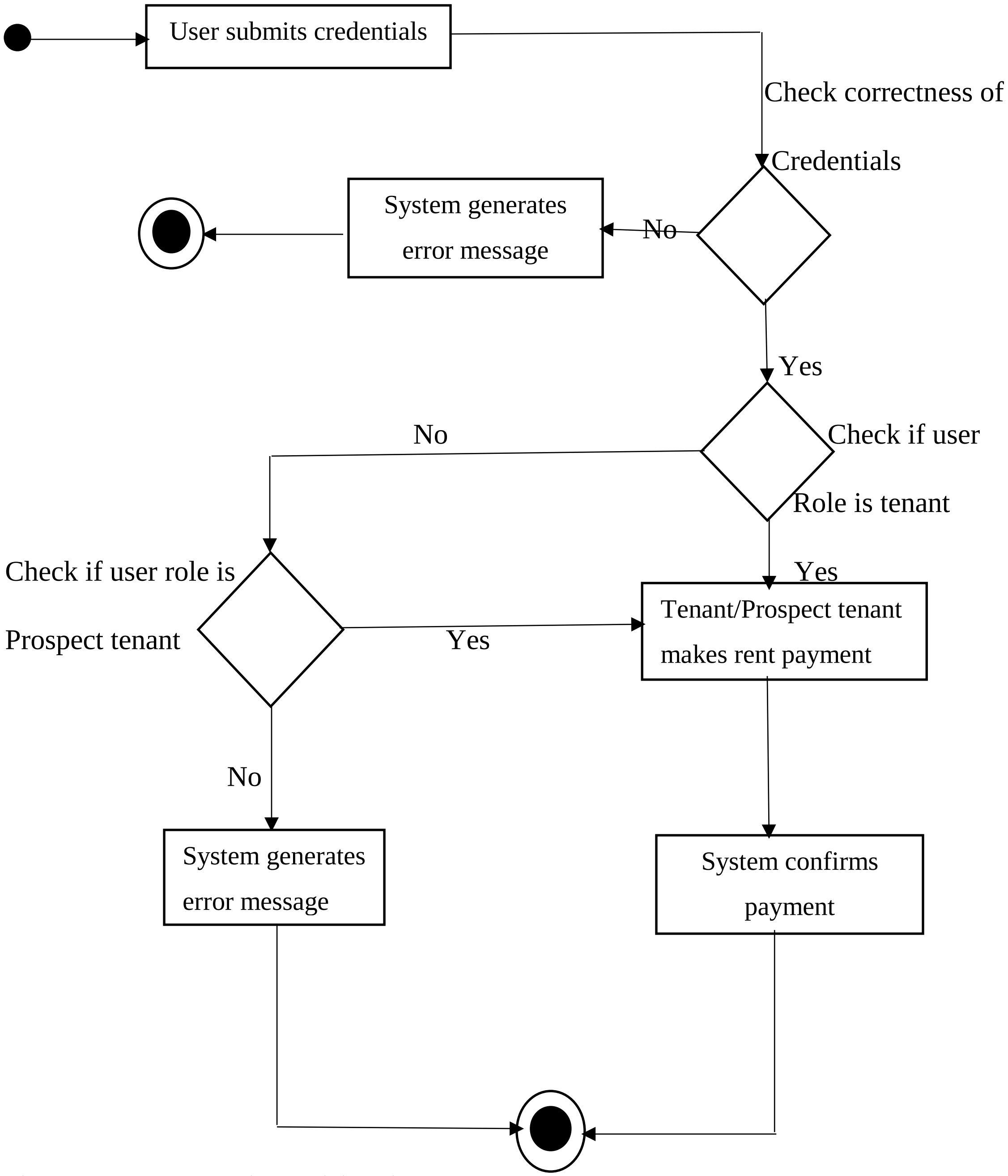


Figure 12: Rent Processing Activity Diagram

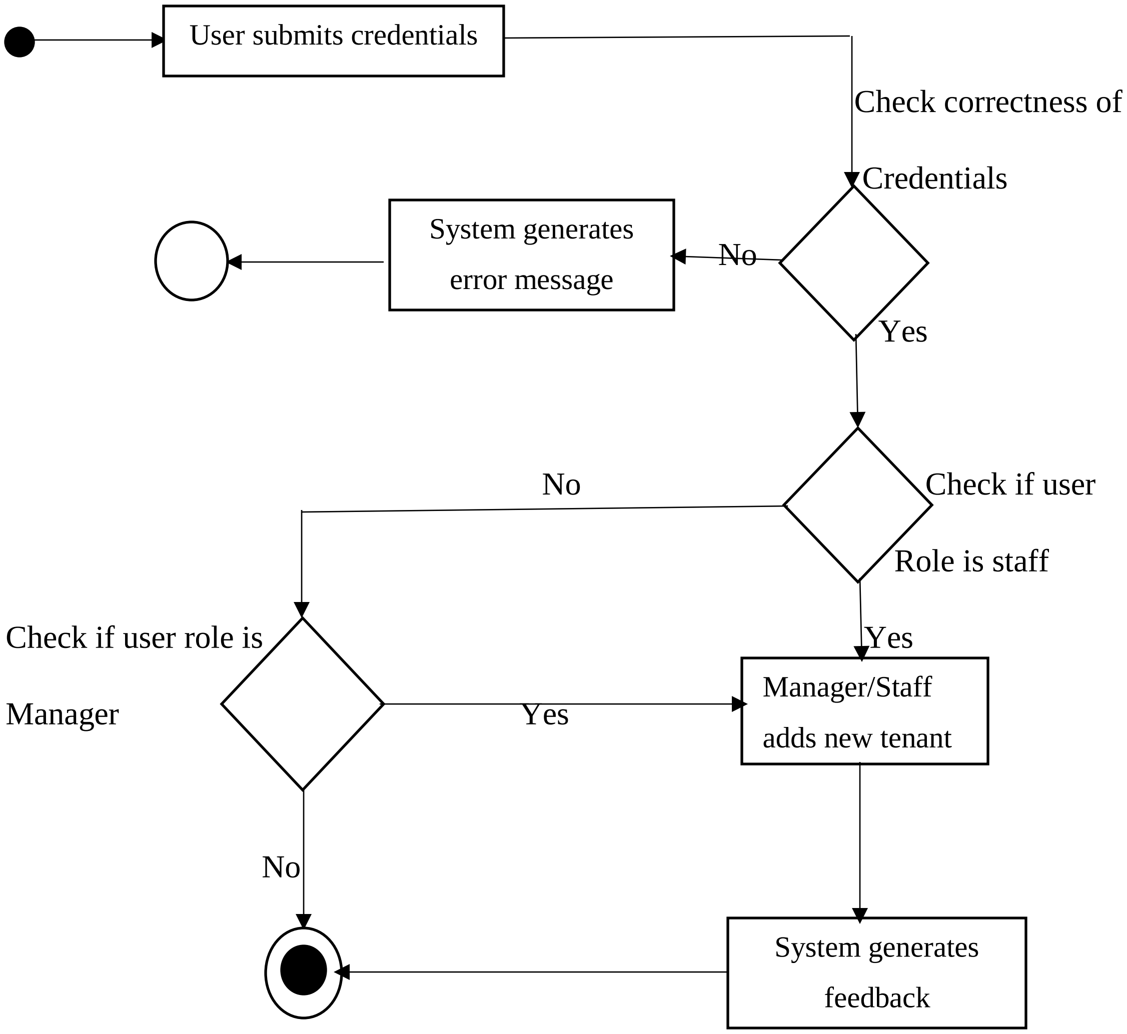


Figure 13: Tenant Addition Activity Diagram

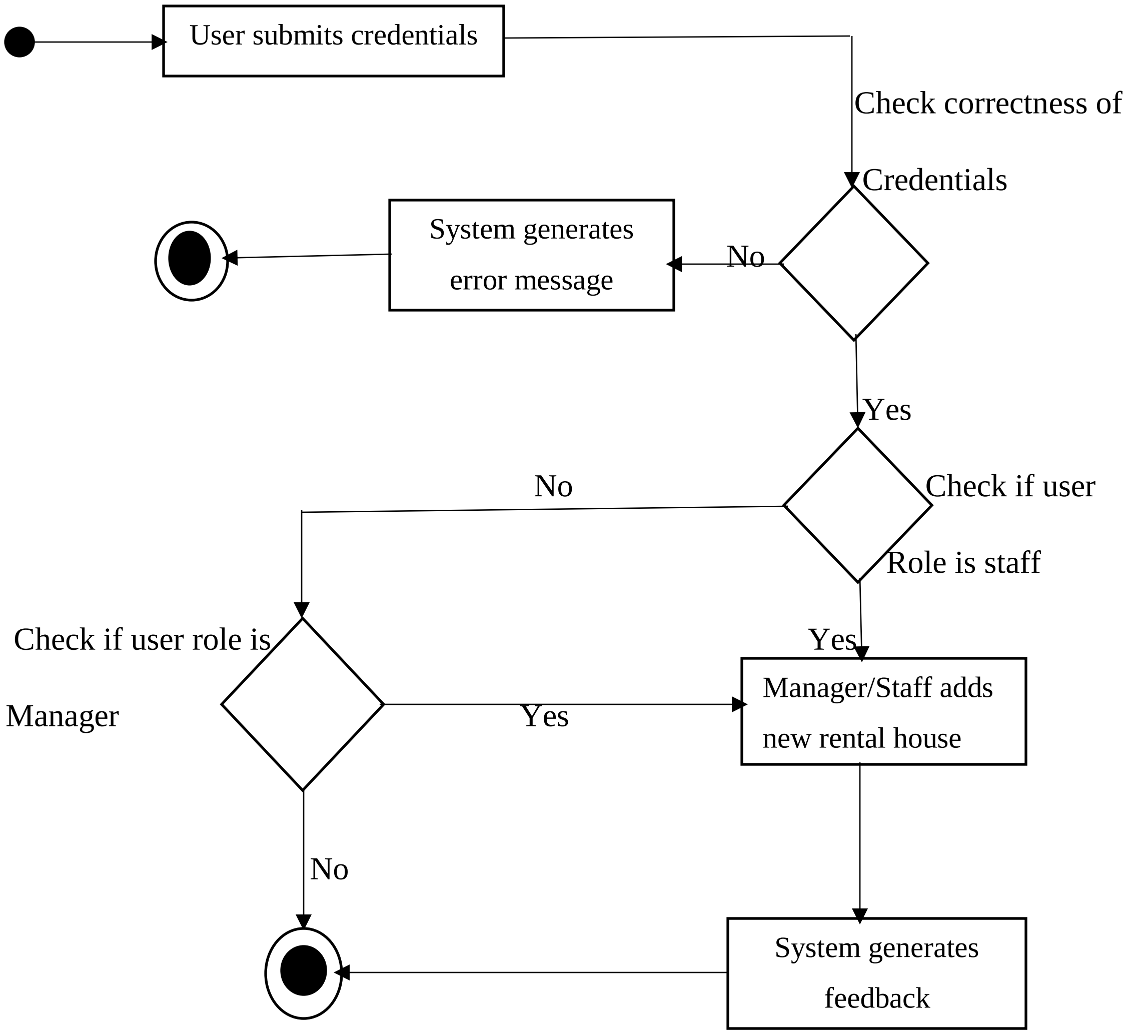


Figure 14: Rental House Addition Activity Diagram

4.5 Interface Design

The web application was created with the following design considerations in mind:

i. **Consistent**. The website should have a similar look and feel on every page. Every page should have the same header/logo, heading style, fonts, navigations etc.

ii. **Efficient and easy to maintain**. This refers to the fact that there is need to separate content from layout, so that you can easily change your page design without editing every page on the site.

iii. **Layout**. The layout of each page should have a good contrast between the text and background area. This helps considerably with visibility as it will be difficult to read the text if it is almost the same color as the background. Monitor size should also be taken into consideration.

iv. **Easy to navigate and use**. Users should not have a hard time trying to navigate the site. Navigation links should be consistent and clearly labeled. All navigation links should also be working properly and should point to the intended page/site.

v. **Browser compatible**. When designing the site consider different browser environments. Extensive testing should be done on each page in all the major browsers and the design changed appropriately to cater for all.

vi. **Visually appealing**. The use of color, text, fonts and graphics should be carefully considered and used to ensure that the site is visually appealing to its visitors.

vii. **Speed**. The performance of a website is mostly rated by its up -time and downtime. These terms refer to the amount of time it takes the site to respond to requests. Graphics should be kept to a minimum to allow the site to load faster. The pages on the site should load within an acceptable time e.g. under 10seconds.

CHAPTER FIVE: IMPLEMENTATION

5.1 Description of Developed System

The system encompasses different activities associated with management of rental houses. The main functionalities of the system are:

- Maintaining tenant profiles.
- Maintaining of landlord profiles.
- Maintenance of rental houses.
- Tenant Self Service (TSS).
- Payment.

All the above activities involve addition operations, updating operations and retrieval of records stored in the database.

5.2 Technical Details of Implemented System

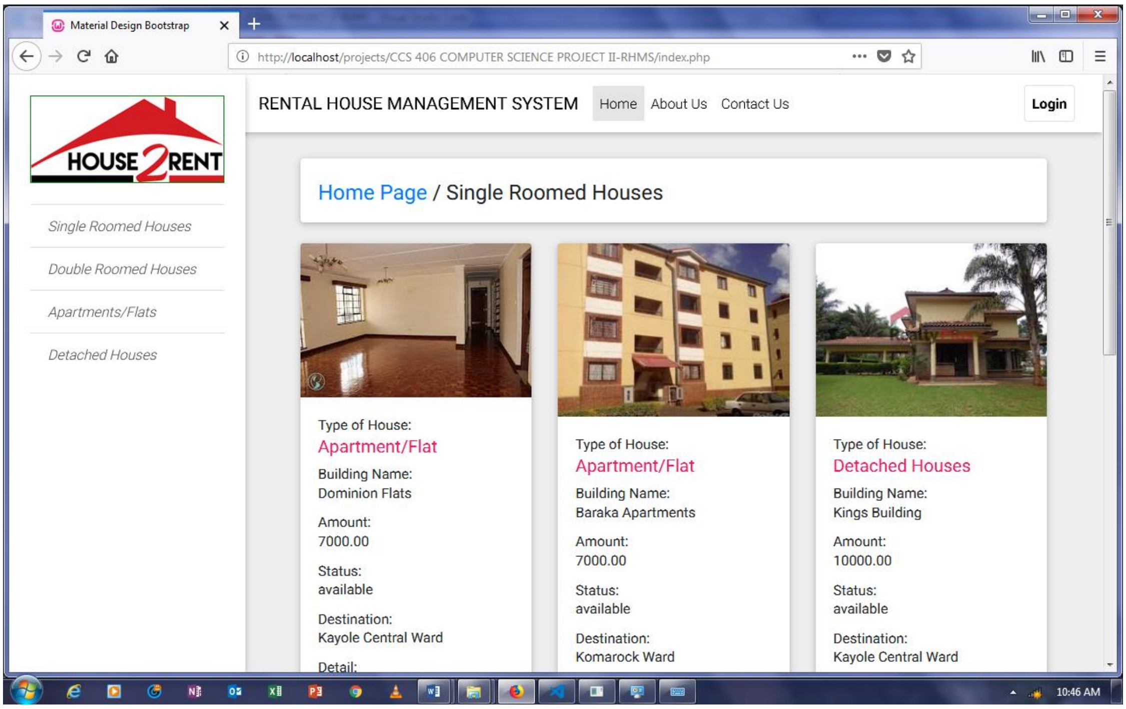
5.2.1 Algorithms

MD5 Encryption. MD5 algorithm was used for password encryption. MD5 stands for Message Digest algorithm 5 and is a widely used cryptographic hash function. The idea behind this algorithm is to take up a random data (text or binary) as an input and generate a fixed size “hash value” as the output. The input data can be of any size or length but the output “hash value” size is always fixed. As can be seen from the above example, whatever input size is given, the algorithm generates a fixed size (32-digit hex) MD5 hash.

5.2.2 System Installation

The system was developed and tested on a laptop computer running Windows 7 and the WAMP Server. In order for the Web application to be accessible via the internet it will have to be installed on a Web Server running Apache, PHP and MySQL. The suitable operating system for the web server will be Linux as it is more stable and less prone to virus but a windows-based platform will equally do the job just as well. A suitable domain name will have to be chosen and registered in order for the web application to be accessed via a URL and hosting and administration fees paid to the web hosting company of choice either annually or monthly depending on the package and terms agreed upon. The web application will be accessible via most of the popular web browsers on the market. A suitable web browser like Mozilla Firefox will have to be installed on the client machine wishing to access the web application.

5.3 Screenshots of the Developed System


Material Design Bootstrap
http://localhost/projects/CCS 406 COMPUTER SCIENCE PROJECT II-RHMS/index.php
HOUSE 2 RENT
RENTAL HOUSE MANAGEMENT SYSTEM
Home
About Us
Contact Us
Login
Single Roomed Houses
Double Roomed Houses
Apartments/Flats
Detached Houses
10:46 AM

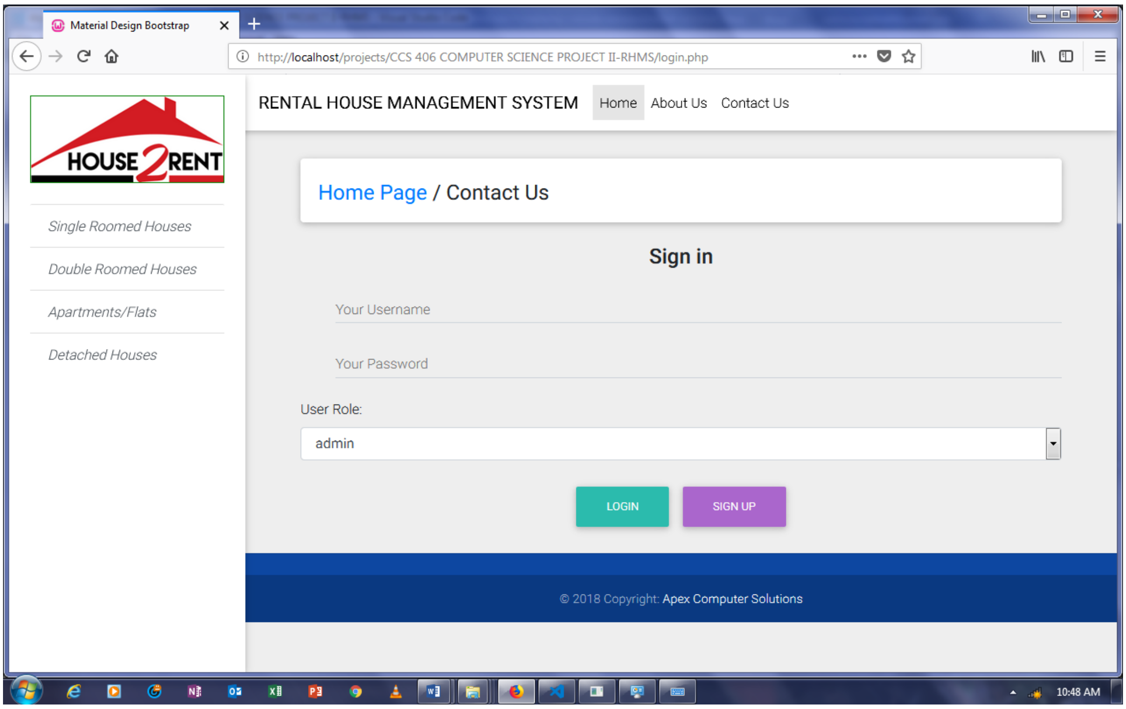
Home Page / Single Roomed Houses

Type of House:
Apartment/Flat
Building Name:
Dominion Flats
Amount:
7000.00
Status:
available
Destination:
Kayole Central Ward
Detail:

Type of House:
Apartment/Flat
Building Name:
Baraka Apartments
Amount:
7000.00
Status:
available
Destination:
Komarock Ward

Type of House:
Detached Houses
Building Name:
Kings Building
Amount:
10000.00
Status:
available
Destination:
Kayole Central Ward

Figure 15: Homepage


Material Design Bootstrap
http://localhost/projects/CCS 406 COMPUTER SCIENCE PROJECT II-RHMS/login.php
HOUSE 2 RENT
Single Roomed Houses
Double Roomed Houses
Apartments/Flats
Detached Houses
RENTAL HOUSE MANAGEMENT SYSTEM
Home
About Us
Contact Us
LOGIN
SIGN UP
10:48 AM

Home Page / Contact Us

Sign in

Your Username

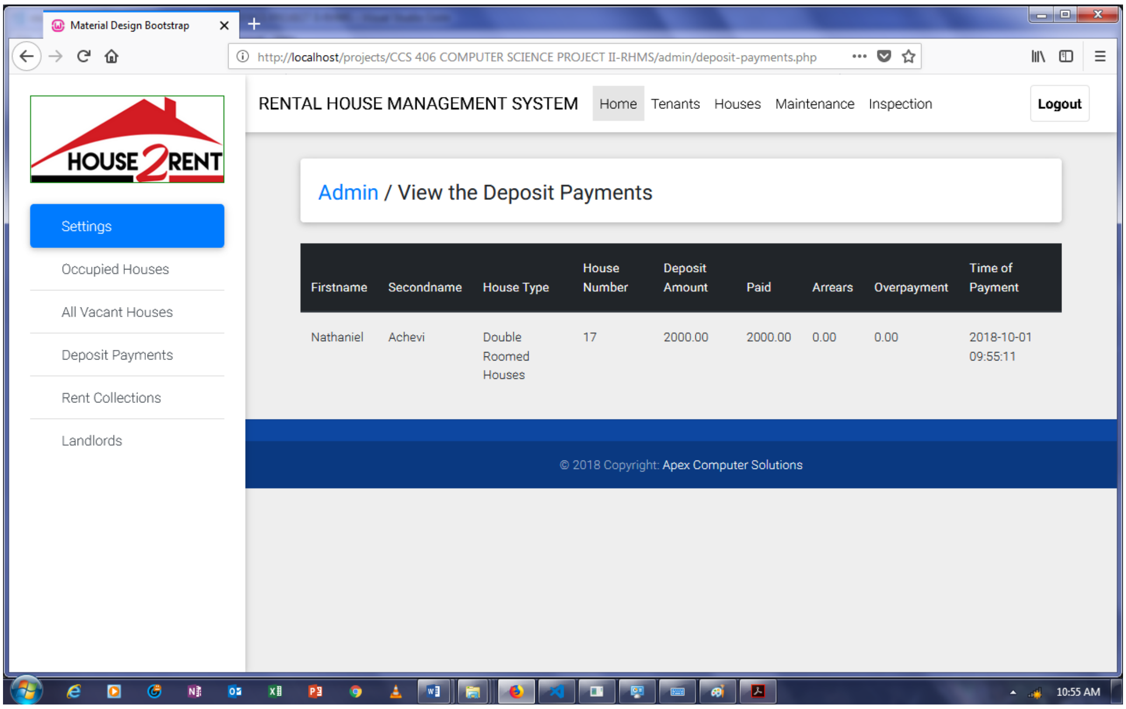
Your Password

User Role:

admin

© 2018 Copyright: Apex Computer Solutions

Figure 16: Login

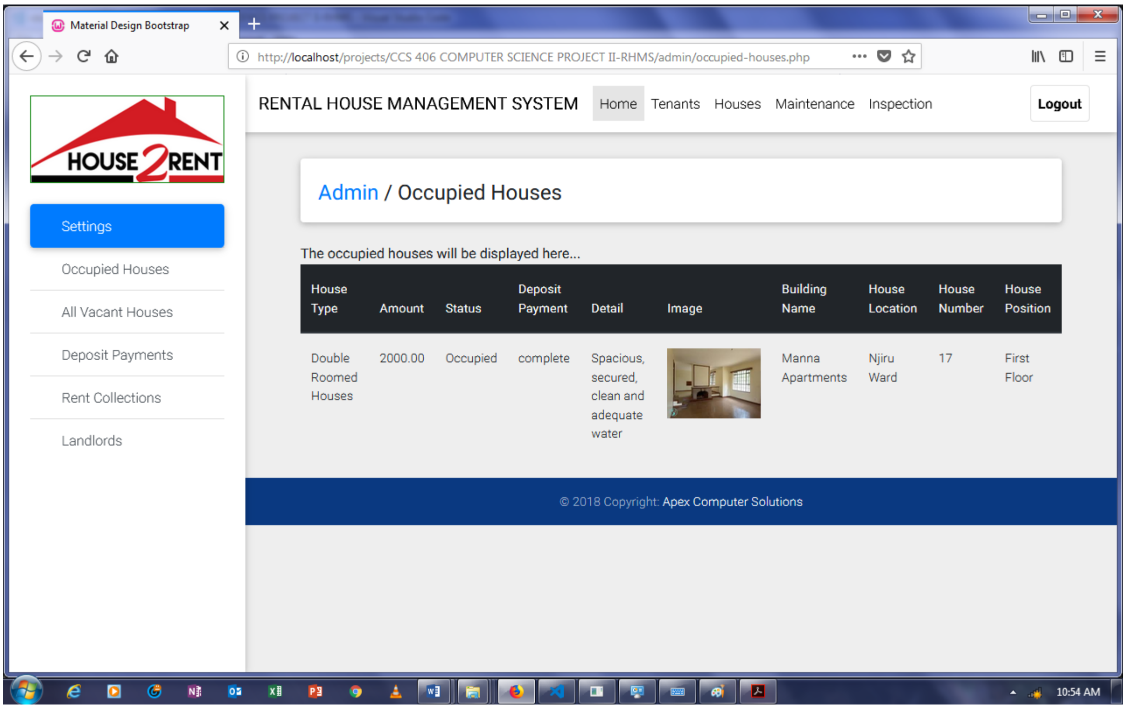

Material Design Bootstrap
http://localhost/projects/CCS 406 COMPUTER SCIENCE PROJECT II-RHMS/admin/deposit-payments.php
HOUSE 2 RENT
Settings
Occupied Houses
All Vacant Houses
Deposit Payments
Rent Collections
Landlords
RENTAL HOUSE MANAGEMENT SYSTEM
Home
Tenants
Houses
Maintenance
Inspection
Logout
10:55 AM

Admin / View the Deposit Payments

Firstname	Secondname	House Type	House Number	Deposit Amount	Paid	Arrears	Overpayment	Time of Payment
Nathaniel	Achevi	Double Roomed Houses	17	2000.00	2000.00	0.00	0.00	2018-10-01 09:55:11

© 2018 Copyright: Apex Computer Solutions

Figure 17: Deposit Payments


Material Design Bootstrap
http://localhost/projects/CCS 406 COMPUTER SCIENCE PROJECT II-RHMS/admin/occupied-houses.php
HOUSE 2 RENT
RENTAL HOUSE MANAGEMENT SYSTEM
Home
Tenants
Houses
Maintenance
Inspection
Logout
Settings
Occupied Houses
All Vacant Houses
Deposit Payments
Rent Collections
Landlords
10:54 AM

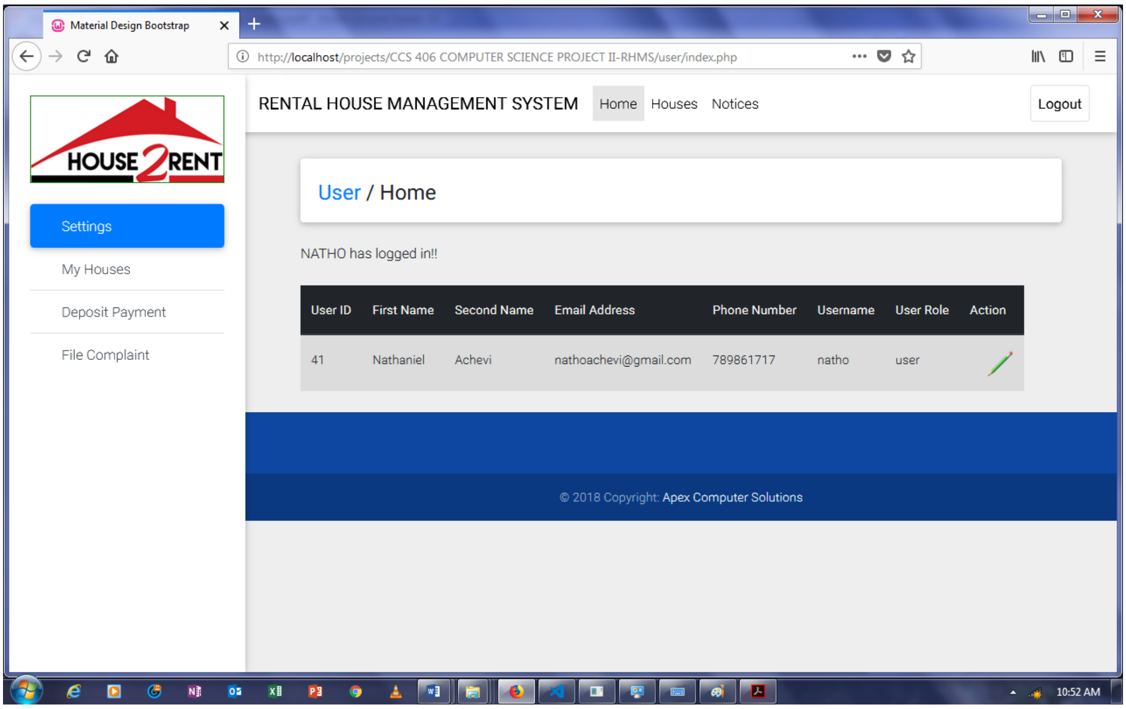
Admin / Occupied Houses

The occupied houses will be displayed here...

House Type	Amount	Status	Deposit Payment	Detail	Image	Building Name	House Location	House Number	House Position
Double Roomed Houses	2000.00	Occupied	complete	Spacious, secured, clean and adequate water		Manna Apartments	Njiru Ward	17	First Floor

© 2018 Copyright: Apex Computer Solutions

Figure 18: Occupied Houses


Material Design Bootstrap
http://localhost/projects/CCS 406 COMPUTER SCIENCE PROJECT II-RHMS/user/index.php
HOUSE 2 RENT
RENTAL HOUSE MANAGEMENT SYSTEM
Home
Houses
Notices
Logout
Settings
My Houses
Deposit Payment
File Complaint
10:52 AM

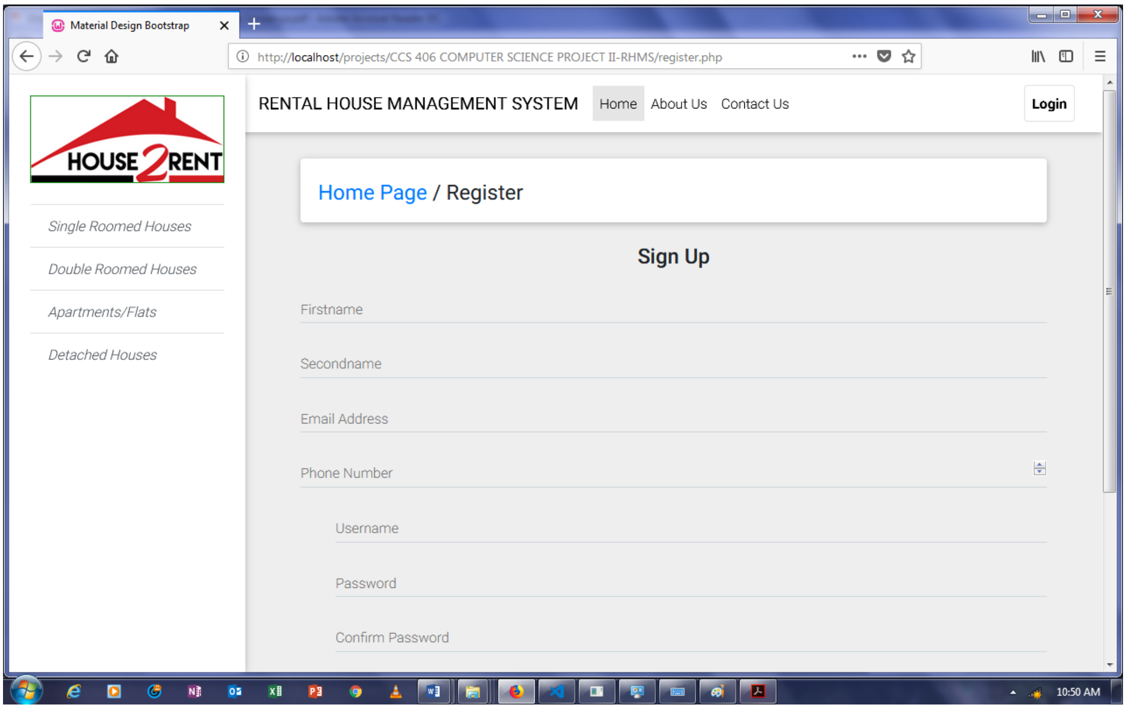
User / Home

NATHO has logged in!!

User ID	First Name	Second Name	Email Address	Phone Number	Username	User Role	Action
41	Nathaniel	Achevi	nathoachevi@gmail.com	789861717	natho	user	✎

© 2018 Copyright: Apex Computer Solutions

Figure 19: Profile


Material Design Bootstrap
http://localhost/projects/CCS 406 COMPUTER SCIENCE PROJECT II-RHMS/register.php
HOUSE 2 RENT
RENTAL HOUSE MANAGEMENT SYSTEM
Home
About Us
Contact Us
Login
Single Roomed Houses
Double Roomed Houses
Apartments/Flats
Detached Houses
10:50 AM

Home Page / Register

Sign Up

Firstname

Secondname

Email Address

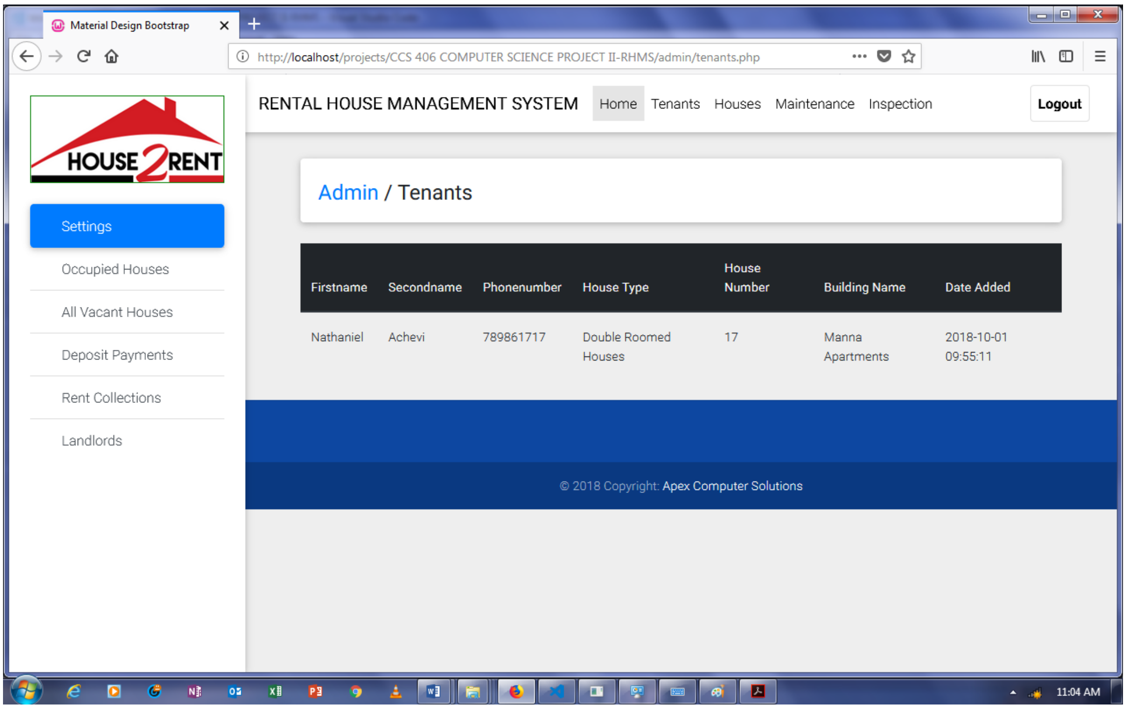
Phone Number

Username

Password

Confirm Password

Figure 20: Registration

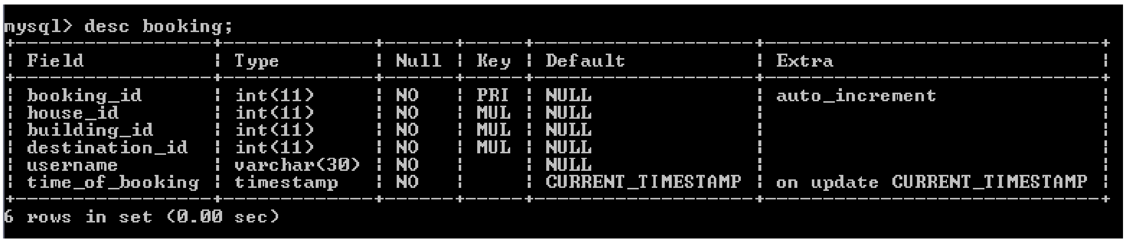


Admin / Tenants

Firstname	Secondname	Phonenumber	House Type	House Number	Building Name	Date Added
Nathaniel	Achevi	789861717	Double Roomed Houses	17	Manna Apartments	2018-10-01 09:55:11

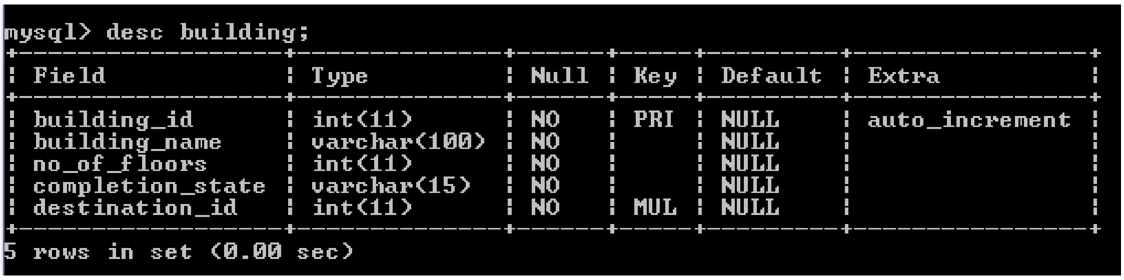
© 2018 Copyright: Apex Computer Solutions

Figure 21: Tenants



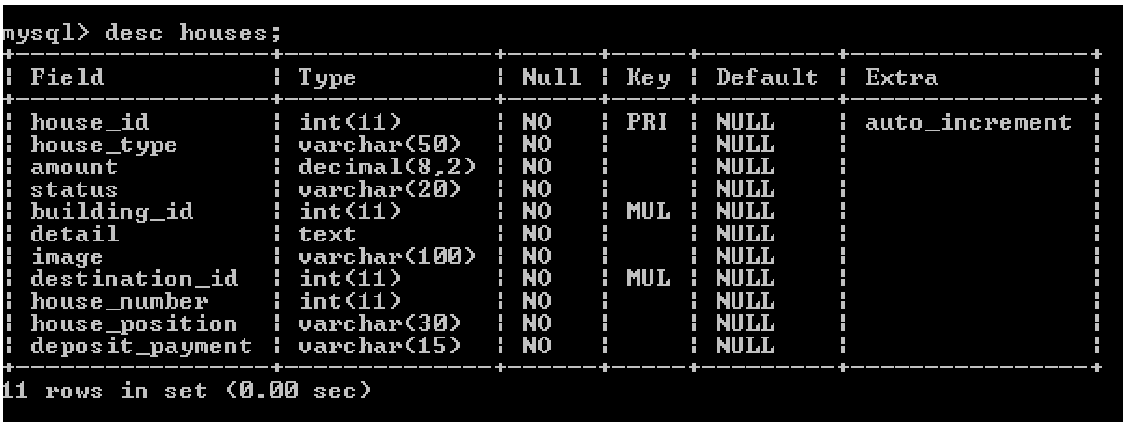
```
mysql> desc booking;
+-----------------+-------------+------+-----+-------------------+-----------------------------+
| Field           | Type        | Null | Key | Default           | Extra                       |
+-----------------+-------------+------+-----+-------------------+-----------------------------+
| booking_id      | int(11)     | NO   | PRI | NULL              | auto_increment              |
| house_id        | int(11)     | NO   | MUL | NULL              |                             |
| building_id     | int(11)     | NO   | MUL | NULL              |                             |
| destination_id  | int(11)     | NO   | MUL | NULL              |                             |
| username        | varchar(30) | NO   |     | NULL              |                             |
| time_of_booking | timestamp   | NO   |     | CURRENT_TIMESTAMP | on update CURRENT_TIMESTAMP |
+-----------------+-------------+------+-----+-------------------+-----------------------------+
6 rows in set (0.00 sec)
```

Figure 22: Booking Table



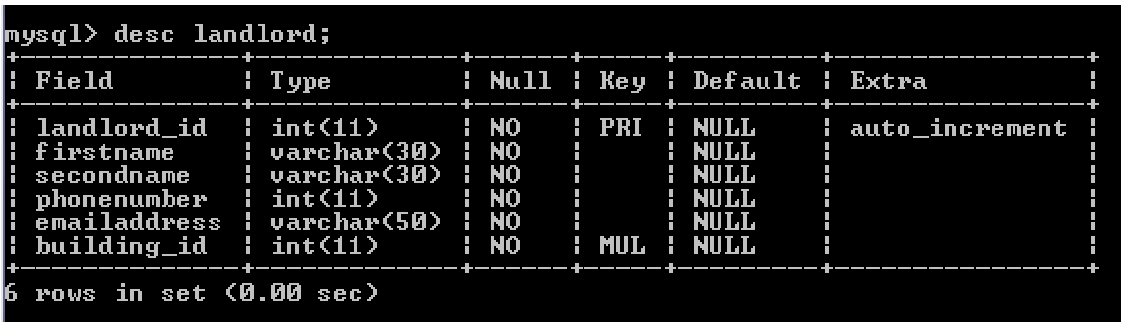
```
mysql> desc building;
+------------------+--------------+------+-----+---------+----------------+
| Field            | Type         | Null | Key | Default | Extra          |
+------------------+--------------+------+-----+---------+----------------+
| building_id      | int(11)      | NO   | PRI | NULL    | auto_increment |
| building_name    | varchar(100) | NO   |     | NULL    |                |
| no_of_floors     | int(11)      | NO   |     | NULL    |                |
| completion_state | varchar(15)  | NO   |     | NULL    |                |
| destination_id   | int(11)      | NO   | MUL | NULL    |                |
+------------------+--------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)
```

Figure 23: Building Table



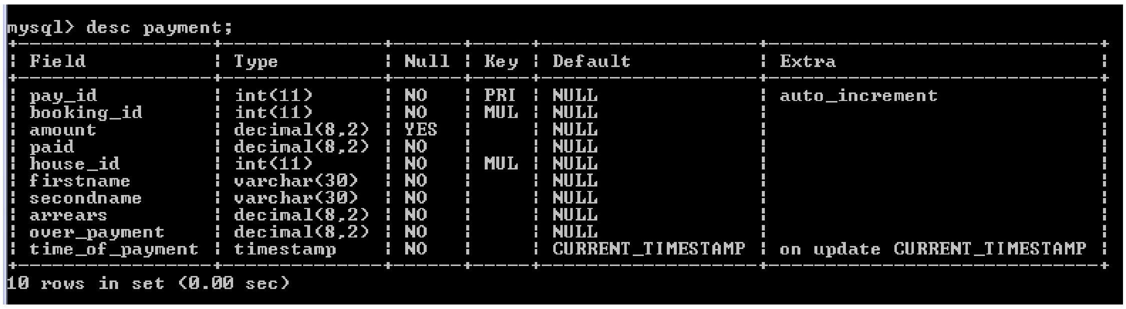
```
mysql> desc houses;
+-----------------+--------------+------+-----+---------+----------------+
| Field           | Type         | Null | Key | Default | Extra          |
+-----------------+--------------+------+-----+---------+----------------+
| house_id        | int(11)      | NO   | PRI | NULL    | auto_increment |
| house_type      | varchar(50)  | NO   |     | NULL    |                |
| amount          | decimal(8,2) | NO   |     | NULL    |                |
| status          | varchar(20)  | NO   |     | NULL    |                |
| building_id     | int(11)      | NO   | MUL | NULL    |                |
| detail          | text         | NO   |     | NULL    |                |
| image           | varchar(100) | NO   |     | NULL    |                |
| destination_id  | int(11)      | NO   | MUL | NULL    |                |
| house_number    | int(11)      | NO   |     | NULL    |                |
| house_position  | varchar(30)  | NO   |     | NULL    |                |
| deposit_payment | varchar(15)  | NO   |     | NULL    |                |
+-----------------+--------------+------+-----+---------+----------------+
11 rows in set (0.00 sec)
```

Figure 24: Houses Table



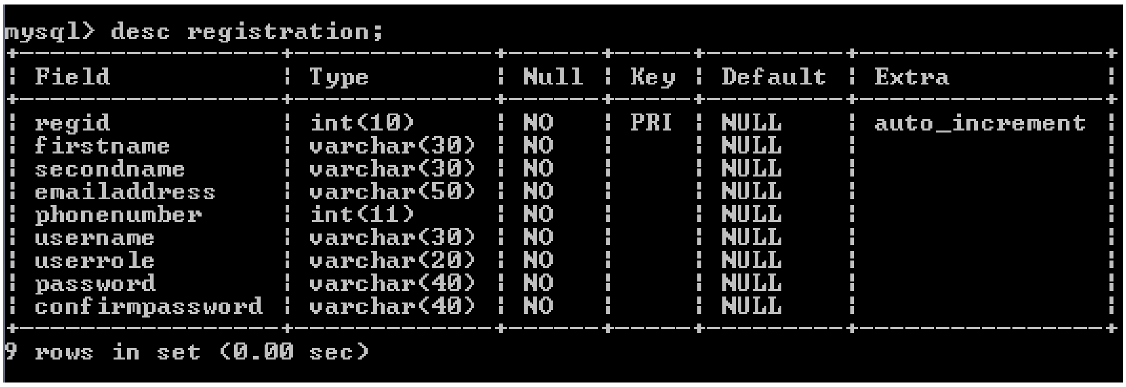
```
mysql> desc landlord;
+--------------+-------------+------+-----+---------+----------------+
| Field        | Type        | Null | Key | Default | Extra          |
+--------------+-------------+------+-----+---------+----------------+
| landlord_id  | int(11)     | NO   | PRI | NULL    | auto_increment |
| firstname    | varchar(30) | NO   |     | NULL    |                |
| secondname   | varchar(30) | NO   |     | NULL    |                |
| phonenumber  | int(11)     | NO   |     | NULL    |                |
| emailaddress | varchar(50) | NO   |     | NULL    |                |
| building_id  | int(11)     | NO   | MUL | NULL    |                |
+--------------+-------------+------+-----+---------+----------------+
6 rows in set (0.00 sec)
```

Figure 25: Landlord Table



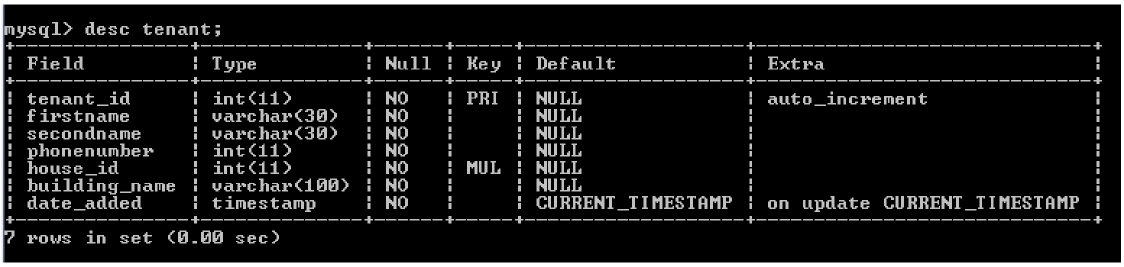
```
mysql> desc payment;
+-----------------+--------------+------+-----+-------------------+-----------------------------+
| Field           | Type         | Null | Key | Default           | Extra                       |
+-----------------+--------------+------+-----+-------------------+-----------------------------+
| pay_id          | int(11)      | NO   | PRI | NULL              | auto_increment              |
| booking_id      | int(11)      | NO   | MUL | NULL              |                             |
| amount          | decimal(8,2) | YES  |     | NULL              |                             |
| paid            | decimal(8,2) | NO   |     | NULL              |                             |
| house_id        | int(11)      | NO   | MUL | NULL              |                             |
| firstname       | varchar(30)  | NO   |     | NULL              |                             |
| secondname      | varchar(30)  | NO   |     | NULL              |                             |
| arrears         | decimal(8,2) | NO   |     | NULL              |                             |
| over_payment    | decimal(8,2) | NO   |     | NULL              |                             |
| time_of_payment | timestamp    | NO   |     | CURRENT_TIMESTAMP | on update CURRENT_TIMESTAMP |
+-----------------+--------------+------+-----+-------------------+-----------------------------+
10 rows in set (0.00 sec)
```

Figure 26: Payment Table



```
mysql> desc registration;
+-----------------+-------------+------+-----+---------+----------------+
| Field           | Type        | Null | Key | Default | Extra          |
+-----------------+-------------+------+-----+---------+----------------+
| regid           | int(10)     | NO   | PRI | NULL    | auto_increment |
| firstname       | varchar(30) | NO   |     | NULL    |                |
| secondname      | varchar(30) | NO   |     | NULL    |                |
| emailaddress    | varchar(50) | NO   |     | NULL    |                |
| phonenumber     | int(11)     | NO   |     | NULL    |                |
| username        | varchar(30) | NO   |     | NULL    |                |
| userrole        | varchar(20) | NO   |     | NULL    |                |
| password        | varchar(40) | NO   |     | NULL    |                |
| confirmpassword | varchar(40) | NO   |     | NULL    |                |
+-----------------+-------------+------+-----+---------+----------------+
9 rows in set (0.00 sec)
```

Figure 27: Registration Table



```
mysql> desc tenant;
+---------------+--------------+------+-----+-------------------+-----------------------------+
| Field         | Type         | Null | Key | Default           | Extra                       |
+---------------+--------------+------+-----+-------------------+-----------------------------+
| tenant_id     | int(11)      | NO   | PRI | NULL              | auto_increment              |
| firstname     | varchar(30)  | NO   |     | NULL              |                             |
| secondname    | varchar(30)  | NO   |     | NULL              |                             |
| phonenumber   | int(11)      | NO   |     | NULL              |                             |
| house_id      | int(11)      | NO   | MUL | NULL              |                             |
| building_name | varchar(100) | NO   |     | NULL              |                             |
| date_added    | timestamp    | NO   |     | CURRENT_TIMESTAMP | on update CURRENT_TIMESTAMP |
+---------------+--------------+------+-----+-------------------+-----------------------------+
7 rows in set (0.00 sec)
```

Figure 28: Tenant Table

5.4 Introduction to Testing

Testing is very important and critical to the success of any project that aims at delivering working software. There are many types of testing that a system may be subjected to. The overall purpose of testing is to ensure the Rental House Management System meets all of its functional and business requirements. The purpose of this section is to describe the overall test plan and strategy for testing the system. The goals in testing this system include validating the quality, usability, reliability and performance of the application. Tests will be designed around requirements and functionality.

5.5 Types of Testing Applied

Two testing approaches were used for the system:

5.5.1 Confirmation Testing

When a test fails because of the defect then that defect is reported and a new version of the software is expected to have the defect fixed. In this case we need to execute the test again to confirm that whether the defect got actually fixed or not. This is known as confirmation testing and also known as re-testing. It is important to ensure that the test is executed in exactly the same way it was the first time using the same inputs, data and environments. Hence, when the change is made to the defect in order to fix it then confirmation testing or re-testing is helpful.

5.5.2 Regression Testing

During confirmation testing the defect got fixed and that part of the application started working as intended. But there might be a possibility that the fix may have introduced or uncovered a different defect elsewhere in the software. The way to detect these **'**unexpected side-effects**'** of fixes is to do regression testing. The purpose of a regression testing is to verify that modifications in the software or the environment have not caused any unintended adverse side effects and that the system still meets its requirements. Regression testing are mostly automated because in order to fix the defect the same test is carried out again and again and it will be very tedious to do it manually. Regression tests are executed whenever the software changes, either as a result of fixes or new or changed functionality.

5.5.3 Test Plans and Results

Table 1: Test Plans and Results

Test Case	Test Purpose	Test Condition	Expected Outcome	Actual Result
Login	Check username and password.	If user details are not correct display error message.	Grant access to the applicable part of the system.	User successfully logs upon submission of correct credentials.

Test Case	Test Purpose	Test Condition	Expected Outcome	Actual Result
Add new tenant	To ensure that a new tenant is added to the system successfully.	If the tenant already exists in the system, an error message is displayed.	New tenant should be successfully added to the system.	If email entered exists in the system, error message is displayed else new tenant is successfully added.
Add new landlord	To ensure that a new landlord is added to the system successfully.	If the landlord already exists in the system, an error message is displayed.	New landlord should be successfully added to the system.	If email entered exists in the system, error message is displayed else a new landlord is successfully added.
Add new house	To ensure that a new house is added into the system successfully.	If the house already exists in the system an error message is displayed.	New rental house should be added successfully.	If a record exists with the same house number and building id, error message displayed otherwise house is added.

Test Case	Test Purpose	Test Condition	Expected Outcome	Actual Result
Rent house	To ensure that a vacant house is rented successfully.	Error message is displayed if an error is encountered in the process.	A vacant rental house should be rented successfully.	If the state of the house is occupied, error message is displayed otherwise the house is rented successfully.
File complaint	To test whether all tenants can successfully file complaints.	Whenever a tenant files a complaint the relevant information should be submitted.	Complaint filed after the tenant supplies the required details.	Complaint filed successfully and a success message displayed.
Maintenance Request	To test whether all tenants can successfully request maintenance operations.	Whenever a tenant requests for a maintenance operation the relevant information should be submitted.	Maintenance request made after the tenant supplies the required details.	Maintenance request made successfully and a success message displayed.

CHAPTER SIX: CONCLUSION AND RECOMMENDATIONS

6.1 Conclusion and Recommendations

The software product produced was fairly good. It achieved most of the user requirements. The user interface is good and is very easy to navigate and even novice users can find their way around the web application easily. The client-side validation is excellent. The lack of integration with a payment API is the major drawback of the system as the system could not make payment implementation a reality but rather simulated the functionality of a payment system. I can confidently state that I have achieved the main aims and core requirements stated at the beginning of the report. The project will not only have potential commercial benefits but is something that can benefit society as it has been designed to provide a solution to rental property managers. Given more time and resources I believe I would have been able to develop a better system. A number of things can be done in future to improve the effectiveness and efficiency of the system.

a) Information archiving. A system holding all the tenant information should have some form of archiving so that details of tenants whose tenancy has been terminated for whichever reason or landlords who no longer require the services of a given management agency are archived rather than been completely deleted from the system. This is essential especially in an event where details may be required for future references. For instance, a new landlord or managing agency for that matter may be interested in knowing the history of a given tenant for them to gauge whether the prospect tenant is sufficiently reliable to pay rent and in due time without delay.

b) Improvements can be made as far as handling of rent payment is done. Proper coordination between the respective banks and the database of the application such that as soon as a tenant makes a rent payment the details of the payment are captured in the database.

c) Incorporation of a payment API so as to fully actualize the system. This would involve liaising with banks to ensure credit card information about a particular tenant is processed through the system and the necessary payments are made online.

REFERENCES

C. R. Kothari (2004). Research Methodology-Methods and Techniques. Second Revised Edition. New Age International Publishers.

Ian Sommerville (2011). Software Engineering. Ninth Edition. Pearson Education Limited

Gary B. Shelly and Harr J. Rosenblatt (2012). Systems Analysis and Design. Ninth Edition. Course Technology Cengage Learning.

United Nations, The Bill of Human Rights (1948)

Basorun J.O. and Fadairo G. (2012) “Government Challenges in Housing the Urban Poor in Ado~Ekiti, Nigeria” Journal of Sustainable Society Vol. 1, No.2, 2012, 31~35

Hal Pawson & Filip Sosenko (2009), “Assessing Resident Satisfaction” Heriot~Watt University and Ipos MORI