

**BAHIRDAR INSTITUTE OF TECHNOLOGY**

**PROJECT TITLE: House Rental Information Management Database System**

**Dept: Software Engineering**

**GROUP MEMBERS**

**NAME ID**

**HANA DAREGOT BDU 1410528**

**GELILA DAWIT BDU 1410517**

**SAMUEL CHALACHEW BDU 1404541**

**ASER ALEMU BDU 1405002**

**REDIET MOLLA BDU 1405466**

**YONATAN ASSEFA BDU 1405457**

**Submitted to: MR. Belete M.**

**Date of submission: 19/08/2015**

**TABLE OF CONTENTS**

**CONTENTS PAGE**

CHAPTER……………………………………………………………………………………………..…...1

1.Introduction 2

1.1. Background of the system 2

1.2. Problems of current system 3

1.3. Objectives 3

[1.3.1. General objectives 3](file:///C:\\Users\\Daregot\\Downloads\\Telegram%20Desktop\\Mini%20project%20document%20template%20(3).docx" \l "_Toc2153344)

[5](file:///C:\\Users\\Daregot\\Downloads\\Telegram%20Desktop\\Mini%20project%20document%20template%20(3).docx" \l "_Toc2153344)1.3.2. Specific objective 4

[1.4. Scope of the project 7](file:///C:\Users\Daregot\Downloads\Telegram%20Desktop\Mini%20project%20document%20template%20(3).docx#_Toc2153346)

1.5. Limitations 7

1.6. Significance of the System 8

1.7. Methodology 8

Chapter two 9

Database design 8

2.1. Conceptual database design 8

[2.1.1. Identify entity& attribute 8](file:///C:\Users\Daregot\Downloads\Telegram%20Desktop\Mini%20project%20document%20template%20(3).docx#_Toc2153357)

[2.1.2. Construct schema 9](file:///C:\Users\Daregot\Downloads\Telegram%20Desktop\Mini%20project%20document%20template%20(3).docx#_Toc2153358)

[2.1.3. Role of attribute in relationship 9](file:///C:\Users\Daregot\Downloads\Telegram%20Desktop\Mini%20project%20document%20template%20(3).docx#_Toc2153359)

[2.1.4. Entity, relationship, cardinality &participation 11](file:///C:\Users\Daregot\Downloads\Telegram%20Desktop\Mini%20project%20document%20template%20(3).docx#_Toc2153360)

2.1.6. ER\_digram 12

2.2. Logical data base design 13

[2.2.1. Converting ER-diagram into Relation 15](file:///C:\Users\Daregot\Downloads\Telegram%20Desktop\Mini%20project%20document%20template%20(3).docx#_Toc2153364)

2.2.2 Normalization 15

2.3. Physical database design 16

2.3.1. Physical implementation on SQL server 16

[2.3.2.Database diagrams of the system 18](file:///C:\Users\Daregot\Downloads\Telegram%20Desktop\Mini%20project%20document%20template%20(3).docx#_Toc2153369)

Chapter Three

3.1 Conclusion………………………………………………………………………………18

Appends

**CHAPTER 1**

**Introduction**

A home renting information management system is a software application designed to help landlords or property managers manage their rental properties more efficiently. The system provides a centralized platform for managing lease agreements, collecting rent, handling maintenance requests, screening tenants, and ensuring compliance with legal requirements and regulations. The system also provides tenants with a secure platform to communicate with their landlords or property managers, make rent payments, and submit maintenance requests.

The use of a home renting information management system can help landlords or property managers save time, reduce errors, and improve communication with their tenants. The system can also help tenants by providing them with a convenient and secure platform to manage their rental properties, make rent payments, and communicate with their landlords or property managers.

The features of a home renting information management system may vary depending on the specific needs of the stakeholders. However, some common features include lease management, rent collection and payment, maintenance request management, tenant screening, compliance management, reporting and analytics, and communication and collaboration.

Overall, a home renting information management system can help improve the rental experience for both landlords or property managers and tenants, making it an essential tool for managing rental properties in today's digital age.

Our project entitled Home Renting Managment System will provide a brief understanding about background of the study, problems, objectives, scope, significance, methodology and database design.

**1.1BACKGROUND OF THE STUDY**

Housing has a central importance to quality of life with considerable economic, social, cultural and personal significance. Housing plays a huge role in revitalizing economic growth in any country, with shelter being among key indicators of development.Housing as a basic human right demands that urban dwellers should have access to a decent housing, defined as one that provides a foundation for rather than being a barrier to good physical and mental health, personal development and fulfillment of life objectives. The focus of this research project is basically managing housing for low income, medium and high incomes households or what is commonly known as affordable housing. Most families choose to rent houses based on their income and family situations.

**1.2 Problem of current system**

Not having online home rental management system for who wants to rent home gets a lot of trouble to find home even the home is available.There is no properly allocate home and the system does not easily arrange according to their user interest. And also the home rental management system almost is done through the manual system. The administrative system doesn't have the facility to make home rental management system through online and the most time the work done through illegal intermediate person without awareness of the administrative and this make more complex and more cost to find home for the customer. This leads to customer in to more trouble, cost, dishonest and time wastage.

The problem found in the current system:

- Complexity of finding home is not easy and more tedious.

- Extra money to find home.

- The user cannot get information about home when they need.

- There is too match time consumption find home

-Complexity of the system for payment.

Management has become difficult because of issues that include:

* Data growth:Data increase day to day. Storing and maintaining all data manually is very difficultiy
* Lack of computerized system:Currently most landlords/ property managers use the manual system in recording and maintaining their property and customers data may easily get damaged leading to loss of data.
* There is no database to store information.
* Potential of data loss or damage is very high because data is stored on tangible files.
* Lack of these crucial requirements makes management of the houses very difficult .

**1.3 Objective**

**1.3.1 General objective**

The general objectives of a home renting information management system are:

1. To streamline the rental process by providing a centralized platform for landlords or property managers to manage their rental properties.

2. To improve communication between landlords or property managers and tenants, resulting in a better rental experience for both parties.

3. To automate rent collection and payment processes, reducing the risk of errors and late payments.

4. To provide tenants with a secure platform to submit maintenance requests and communicate with their landlords or property managers.

5. To ensure compliance with legal requirements and regulations related to rental properties, such as lease agreements, tenant screening, and property maintenance.

6. To provide landlords or property managers with real-time insights into the performance of their rental properties, such as occupancy rates, rental income, and maintenance costs.

7. To increase the efficiency of property management operations, saving time and resources for landlords or property managers.

**1.3.2 Specific objectives**

Some specific objectives of a home renting information management system include:

1. Efficient management of lease agreements: The system should allow landlords or property managers to create, store, and manage lease agreements for their rental properties. It should also provide reminders for lease renewals and terminations.

2. Rent collection and payment automation: The system should automate the rent collection and payment process, including sending reminders for rent payments, accepting online rent payments, and generating rent receipts.

3. Maintenance request management: The system should provide tenants with a platform to submit maintenance requests, and landlords or property managers should be able to track, prioritize, and assign tasks to maintenance staff.

4. Tenant screening and management: The system should provide tools for tenant screening, including background checks and credit reports. It should also store tenant information, such as contact details, lease agreements, and payment history.

5. Compliance with legal requirements: The system should ensure that landlords or property managers comply with legal requirements and regulations related to rental properties, such as safety inspections, zoning laws, and fair housing regulations.

6. Reporting and analytics: The system should generate reports and analytics on the performance of rental properties, including occupancy rates, rental income, maintenance costs, and tenant satisfaction.

7. Communication and collaboration: The system should provide a platform for communication and collaboration between landlords or property managers and tenants through features such as messaging, document sharing, and notifications.

**1.4 Scope of The system**

The scope of a home renting information management system will depend on the specific needs of the landlords and tenants using it. However, some common features that can be included in the system are:

1. Property listing management: The system can be used to create, edit, and manage property listings, including property details, photographs, and rental rates.

2. Tenant management: The system can be used to manage tenant information, including contact details, lease agreements, and rent payments.

3. Maintenance management: The system can be used to track maintenance requests and schedule repairs.

4. Communication: The system can provide a platform for landlords and tenants to communicate, including through messaging or email.

5. Payment processing: The system can be used to process rent payments and track payment history.

6. Reporting: The system can generate reports on key metrics such as rental income, occupancy rates, and maintenance costs.

7. Security: The system can include security features such as user authentication, data encryption, and access controls to protect sensitive information.

**1.5 Limitation of the system**

While a home renting information management system can provide many benefits to landlords and tenants, there are also some limitations to consider, including:

1. Initial setup and learning curve: Setting up and learning how to use a new system can take time and resources, which may be a limitation for landlords who are already managing multiple properties.

2. Cost: Depending on the system, there may be a cost associated with using it, which can be a limitation for landlords with small rental properties or tight budgets.

3. Technical issues: Like any software or technology, a home renting information management system may experience technical issues that can disrupt the rental management process.

4. Lack of personal touch: While the system can provide a convenient platform for communication, it may lack the personal touch of in-person interactions, which can be important for building relationships and addressing complex issues.

5. Limited access: Depending on the system, there may be limitations on who can access certain features, which can be a limitation for tenants who need to communicate or access information outside of normal business hours.

**1.6 Significance**

i. The System Which Will allow the user to quickly and easily search a property for buy and Sell.

ii. The register user can upload his property for sale or rent out.

iii. The System is design and developed in such way that it tries to overcome all the prescribe problem.

iv. The system being an online system will give accurate information regarding the property which helps to view all the stuff information directly from anywhere

**1.7. Methodology**

The methodology of a home renting information management system typically involves the following steps:

1. Planning: The first step is to define the scope and objectives of the system, identify the target users, and establish the budget and timeline for the project.

2. Requirements gathering: The next step is to gather requirements from the stakeholders, including landlords or property managers, tenants, and maintenance staff. This involves understanding the key features and functionalities required, such as lease management, rent payment, maintenance request management, tenant screening, compliance, reporting, and communication.

3. Design: Based on the requirements, the system should be designed, including the user interface, database schema, and system architecture. The design should consider factors such as scalability, security, and integration with third-party tools.

4. Development: The system should be developed using appropriate technologies and programming languages. The development process should follow industry best practices, including version control, documentation, and testing.

5. Testing: The system should be tested thoroughly to ensure that it meets the functional and non-functional requirements, is free from bugs and errors, and performs well under different scenarios.

6. Deployment: The system should be deployed in the production environment, and necessary configurations should be made for hosting, security, and performance optimization.

7. Training: The stakeholders should be provided with adequate training and support to use the system effectively, including how to manage properties, tenants, rent payments, maintenance requests, and other features.

8. Maintenance and support: The system should be maintained and updated regularly to fix bugs, improve performance, and add new features. Adequate support should also be provided to resolve issues and answer queries from the stakeholders.

Overall, the methodology should be flexible, adaptive, and collaborative, involving frequent communication and feedback from the stakeholders throughout the development and implementation process.

**CHAPTER 2**

**DATABASE DESIGN**

**2.1. Conceptual database design**

**2.1.1. Identify entity& attribute**

Rent\_house : This entity represents the owner's house that the tenant agrees to rent

Attribute:

* No\_of\_floors
* H\_ID(unique identifier)
* No\_of\_rooms
* Address(city, state, reg\_cod)

Tenant: This entity represents the person that is obligated to pay the owner for renting the house.

Attribute:

* T\_name(T\_name, T\_name)
* T\_ID(unique identifier)
* T\_email
* T\_tel

House\_Contract: This entity represents the terms agreed upon by the owner and the tenant for rent.

Attribute:

* C\_ID(unique identifier)
* Start\_date
* End\_date

House\_Owner: This entity represents the person who owns the house to be rented.

Attribute:

* Own\_name(Own\_Fname, Own\_Lname)
* Own\_email
* Own\_ID(unique identifier)
* Own\_tel

Payment: This entity represents the amount of money payed to the owner by the renter.

Attribute:

* P\_ID(unique identifier)
* P\_date
* Fee\_amt

**2.1.3. Role of attribute in relationship**

Rent\_house

Attribute:

* No\_of\_floors: holds the number of floors on the house.
* H\_ID: holds the registered legal identifier of the house.
* No\_of\_rooms: holds the number of rooms in the rent house.
* Address: holds the physical address of the rent house.

Tenant

Attribute:

* T\_name: holds the name of the tenant.
* T\_ID: holds the legal registered ID number of the tenant.
* T\_email: holds the email the tenant uses.
* T\_tel: holds the telephone number the tenant uses most often.

House\_Contract

Attribute:

* C\_ID: holds the legal registered identifier of the contract.
* Start\_date: holds the date from which the contract begins to take place.
* End\_date: holds the date from which the contract ends.

House\_Owner

Attribute:

* Own\_name: holds the legal name of the owner of the rent house.
* Own\_email: holds the email of the owner of the rent house.
* Own\_ID: holds the legal registered ID number of the owner of the rent house.
* Own\_tel: holds the telephone number the owner uses most often.

Payment

Attribute:

* P\_ID: holds the legal registered identifier of the payment receipt.
* P\_date: holds the date on which the payment was performed.
* Fee\_amt: holds the amount of money paid by the tenant to the owner of the rent house.

**2.1.4. Entity, relationship, cardinality &participation**

Entity:

* Rent\_house
* Tenant
* House\_Contract
* House\_Owner
* Payment

**Relationship**

Rents(Rent\_house, Tenant)

Has(Rent\_house, House\_Contract)

Has(Tenant, House\_Contract)

Has(House\_Contract, House\_Owner)

Has(House\_Contract, Payment)

Pays(Tenant, Payment)

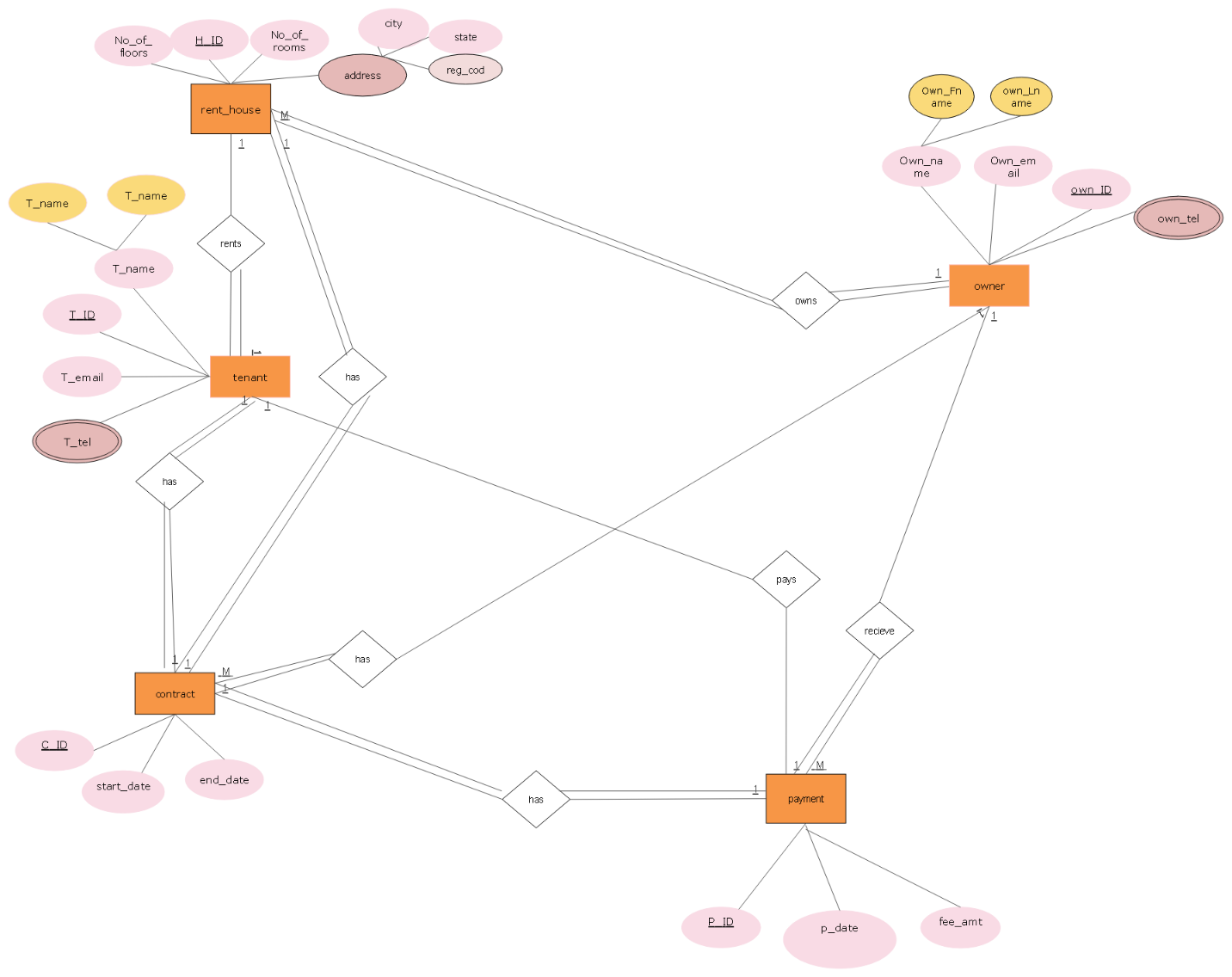
Recieve(House\_Owner, Payment)

Owns(Rent\_house, Owner)

**Cardinality & Participation**

|  |  |
| --- | --- |
| Cardinality | Participation |
| <Rent\_house, Tenant>  1:1 | <Partial, Total> |
| <Rent\_house, House\_Contract>  1:1 | <Total, Total> |
| <Tenant, House\_Contract>  1:1 | <Total, Total> |
| < House\_Owner, House\_contract, >  1:M | <Partial, Total> |
| <House\_Contract, Payment>  1:1 | <Total, Total> |
| <Tenant, Payment>  1:1 | <Partial, Partial> |
| <House\_Owner, Payment>  1:M | <Partial, Total> |
| <Rent\_house, House\_Owner>  M:1 | <Total, Total> |

**2.1.5 ER Diagram**



**2.2 Logical Database Design**

**2.2.1 Converting ER-Diagram in to Relation**

Tenant

|  |  |  |  |
| --- | --- | --- | --- |
| T\_ID | T\_Fname | T\_Lname | T\_email |

Tel

|  |  |
| --- | --- |
| T\_tel | T\_ID |

House Owner

|  |  |  |  |
| --- | --- | --- | --- |
| Own\_ID | Own\_Fname | Own\_Lname | Own\_email |

Owner\_tel

|  |  |
| --- | --- |
| Own\_tel | Own\_ID |

Rent\_house

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| H\_ID | No\_of\_rooms | No\_of\_floors | city | Rh\_state | Reg\_cod |

House Contract

|  |  |  |
| --- | --- | --- |
| C\_ID | StartDate | EndDate |

Payment

|  |  |  |
| --- | --- | --- |
| P\_ID | PDate | Fee\_amt |

Mapping”rents”relationship -Tenant-Rent house

Tenant

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| T\_ID | T\_Fname | T\_Lname | T\_email | H\_ID |

Mapping”owns”relationship -Rent house- House owner

Rent\_house

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| H\_ID | No\_of\_rooms | No\_of\_floors | city | Rh\_state | Reg\_cod | Own\_ID |

Mapping”Has”relationship- Rent house - House contract

Rent\_house

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| H\_ID | No\_of\_rooms | No\_of\_floors | city | state | Reg\_cod | C\_ID |

Mapping”has”relationship-House contract- Tenant

House Contract

|  |  |  |  |
| --- | --- | --- | --- |
| C\_ID | StartDate | EndDate | T\_ID |

Mapping”has”relationship- House contract- House owner

House Contract

|  |  |  |  |
| --- | --- | --- | --- |
| C\_ID | StartDate | EndDate | Own\_ID |

* After mapping we will have the following relation schema for the logical database design stage:

Tenant

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| T \_id | T\_Fname | T\_Lname | T\_email | H\_ID |

Tenant\_tel

|  |  |
| --- | --- |
| T\_tel | T\_ID |

House\_Owner

|  |  |  |  |
| --- | --- | --- | --- |
| Own\_ID | Own\_Fname | Own\_Lname | Own\_email |

Owner\_tel

|  |  |
| --- | --- |
| .Own\_tel | Own\_ID |

Rent house

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| H\_ID | No\_of floors | No\_of rooms | city | Rh\_state | Reg\_cod | Own\_id | T\_ID |

House\_Contract

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| C\_ID | StartDate | EndDate | T\_ID | Own\_ID | H\_ID |

Payment

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P\_ID | PDate | Fee\_amt | Own\_ID | C\_ID | T\_ID |

**2.2.2 Normalization**

Data normalization is the process of reorganizing data within a database so that users can utilize it for further queries and analysis. Simply put, it is the process of developing clean data. This includes eliminating redundant and unstructured data and making the data appear similar across all records and fields.

In database system normalization has up to 6 forms. One database is expected to be in 3rd normal form to be said its normalized.

Back to our system, we got to the following steps:

Unnormalized table

Rent\_house

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| H\_ID | No\_of\_rooms | No\_of\_floors | city | Rh\_state | Reg\_cod |

Based on our system assumption our table has no any multivalued attribute, which means it is in its first normal form.

Second Normal Form/2NF

In our table there is a partial dependency relationship ,which is the attributes can be accessed by both of the keys /composite keys.

Here we have the composite keys rent house-H\_ID ,and the Reg\_cod;

Non prime attributes =Np\_of rooms,No\_of floors,city,Rh-state

* {H\_ID,Reg\_cod} Np\_of rooms,No\_of floors,city,Rh-state
* {H\_ID} Np\_of rooms,No\_of floors
* {Reg\_cod} city,Rh-state

Rent house

|  |  |  |
| --- | --- | --- |
| H\_ID | No\_of floor | No\_of rooms |

Region code

|  |  |  |
| --- | --- | --- |
| Reg\_cod | city | Rh\_state |

Third Normal Form/3NF

In this form we will get ride of transitive dependency relationships from the second normal form

H\_ID🡪 No\_of floor🡪 No\_of rooms

{H\_ID} No\_of floor

{No\_ of floor} No\_of rooms

Rent house

|  |  |
| --- | --- |
| H\_ID | No\_of floors |

No\_of rooms

|  |  |
| --- | --- |
| No\_of floors | No\_of rooms |

Region code

|  |  |  |
| --- | --- | --- |
| Reg\_cod | city | Rh\_state |

Here we normalized our table properly and it is in 3NF.

**2.3 Physical Database Design**

**2.3.1 Implementation**

**The physical design of the system is put on a separate file with under the name “Final.sql” with an sql file extension.**

Here is the code for the system:

create database House\_Rental\_Database\_Sys

create table tenant

(

T\_ID int primary key,

T\_Fname varchar(20),

T\_Lname varchar(20),

H\_ID int foreign key references Rent\_house(H\_ID)

)

create table Tenant\_Tel

(

T\_tel int primary key,

T\_ID int foreign key references tenant(T\_ID)

)

create table House\_Owner

(

own\_ID int primary key,

own\_Fname varchar(20),

own\_Lname varchar(20),

own\_email varchar(20),

)

create table owner\_tel

(

own\_tel int primary key,

own\_ID int foreign key references House\_Owner(own\_ID)

)

create table Rent\_house

(

H\_ID int primary key,

no\_of\_floors int foreign key references number\_of\_floors(no\_of\_floor)

)

create table number\_of\_floors

(

no\_of\_floor int primary key,

no\_of\_room int

)

create table region\_cod

(

reg\_cod varchar(20) primary key,

city varchar(20),

RH\_state varchar(20)

)

create table House\_contract

(

C\_ID int primary key,

startDate int,

endDate int,

T\_ID int foreign key references tenant(T\_ID),

own\_ID int foreign key references House\_Owner(own\_ID),

H\_ID int foreign key references Rent\_House(H\_ID),

)

create table payment

(

P\_ID int primary key,

P\_date int,

fee\_amt int,

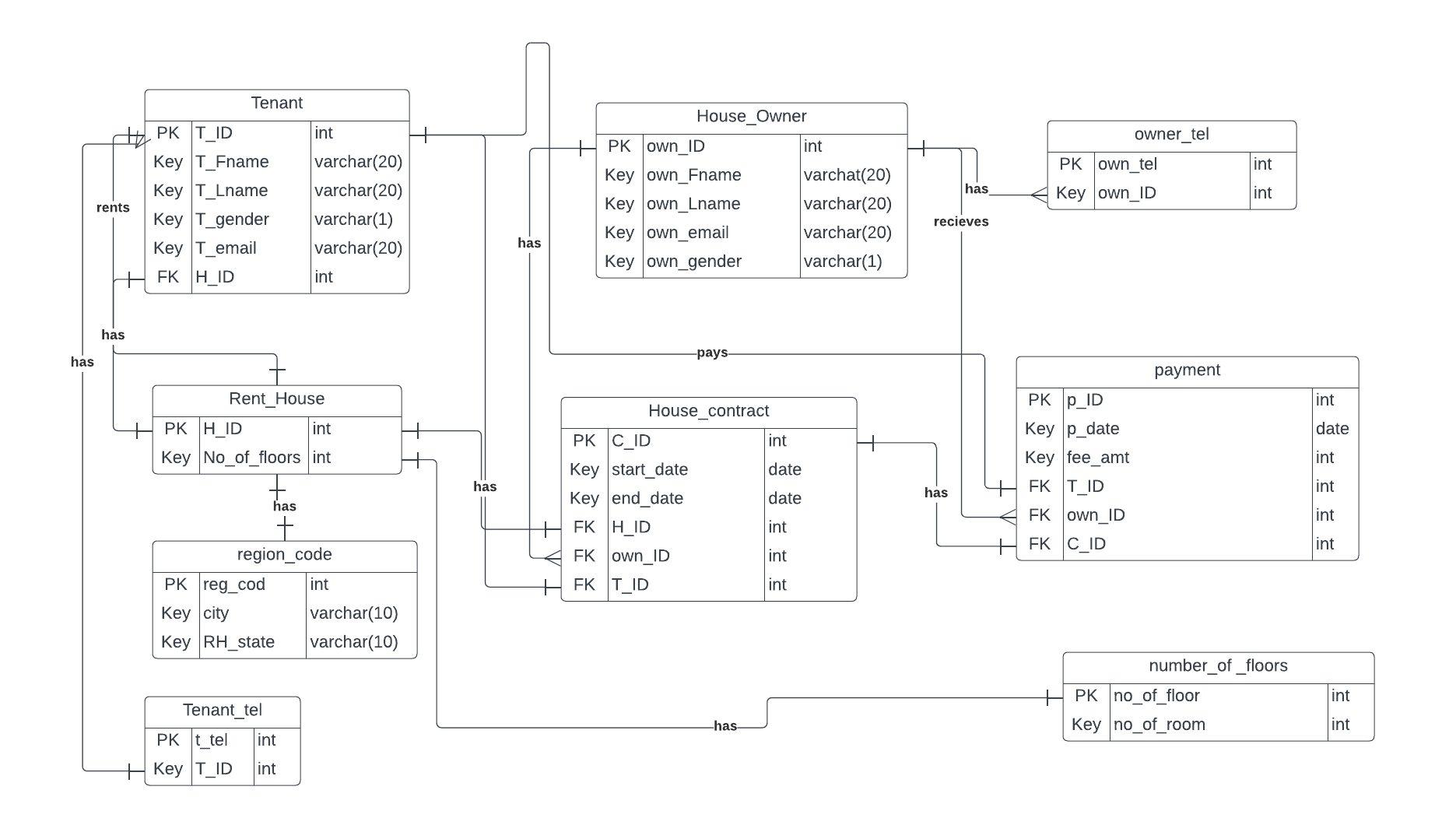
own\_ID int foreign key references House\_Owner(own\_ID),

C\_ID int foreign key references House\_contract(C\_ID),

T\_ID int foreign key references tenant(T\_ID)

)

The execution for the code is done through sequences.

**2.3.2 Database Diagram**

**CHAPTER 3**

**3.1 CONCLUSION**

In conclusion, a home renting information management system is a valuable tool for both landlords and tenants. It can help landlords streamline their rental management process, from advertising vacancies to collecting rent payments and tracking maintenance requests. For tenants, the system can provide a convenient platform for paying rent, submitting maintenance requests, and communicating with their landlord. By using a centralized system, both parties can save time and reduce the risk of miscommunication or errors. Overall, a home renting information management system can improve the efficiency and effectiveness of the rental process for all involved.

*APPENDICES*

* The ER -diagram is done with [www.smartdraw.com](file:///C:\Users\Daregot\Downloads\Telegram%20Desktop\www.smartdraw.com).
* All tables are done with Microsoft Word.
* The database diagram is done with Lucid Chart.
* The physical design is done with SQL Server Management Studio19.