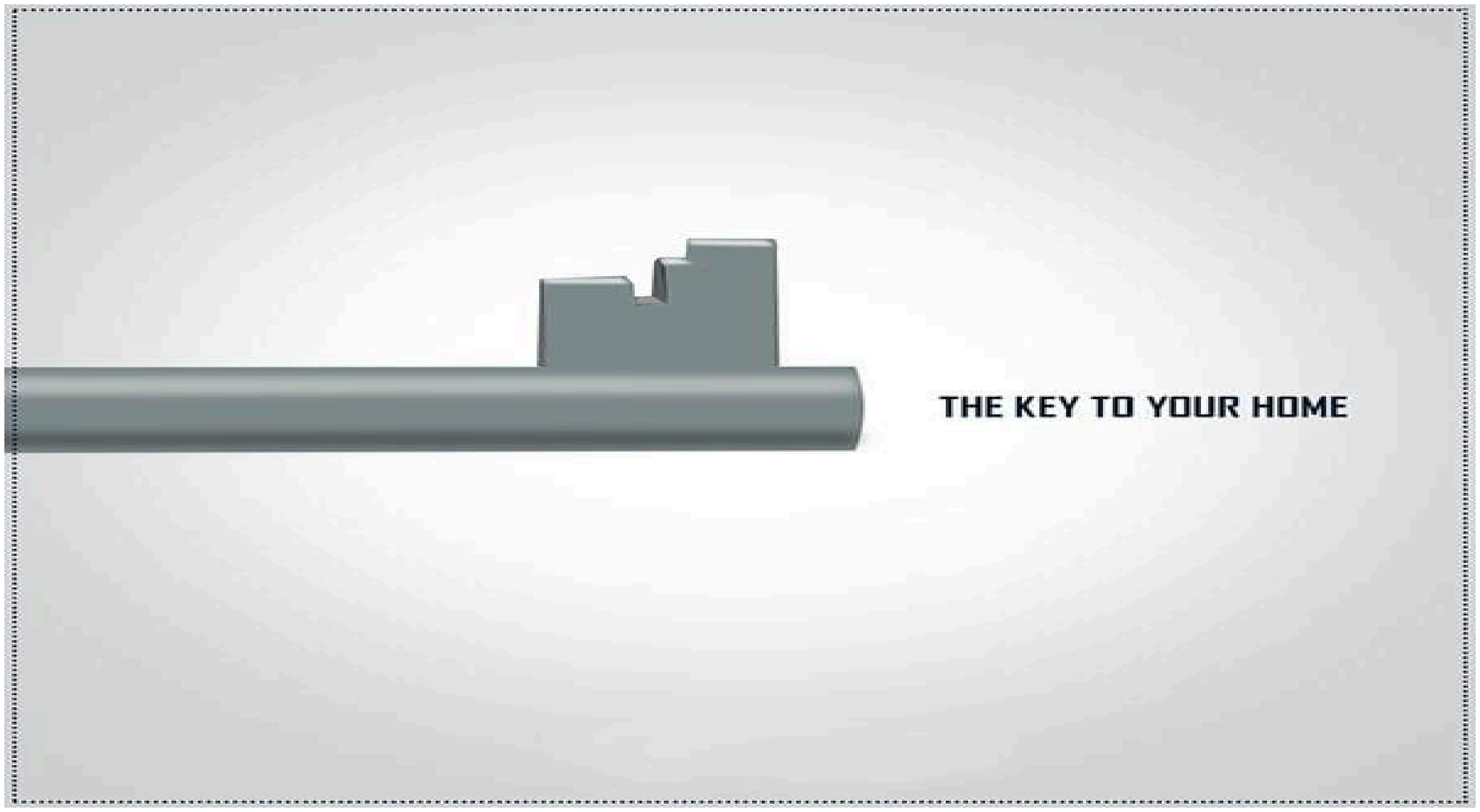


Real Estate Management System

- **Project members**

- Mahmoud Mahdy
- Youssif Mohamed
- Mohamed Emad
- Yassen Ehab
- Mohamed Hany

● Introduction

In today’s technology-driven world, the real estate sector increasingly relies on digital solutions to enhance efficiency and streamline operations. Our project aims to develop a Real Estate Management System designed to help real estate businesses manage their operations more effectively.

This system will feature a comprehensive website tailored to meet the needs of real estate companies. It will simplify property management, tenant handling, and transaction tracking. Key features include property data management, transaction monitoring, and maintenance management, all designed to boost operational efficiency and reduce administrative tasks.

With a user-friendly interface, our platform will provide a seamless experience, enabling companies to deliver superior service to their clients. By integrating the latest technologies, we aim to create a tool that supports the growth and success of real estate management businesses, demonstrating our commitment to innovation and excellence in the industry.

Problem definition

Managing real estate is complex and time-consuming due to the abundance of data and transactions involved. This manual process often leads to errors, delays, and increased administrative burdens.

Objectives

1- Create an admin dashboard for managing properties, users, and transactions
2- Manage user accounts by adding, editing, and deleting them
3- Control property listings with tools to add, update, or remove properties.
4-Track transactions and manage payment
5- Monitor system activity to ensure smooth operation.
6- Generate reports on property and financial performance.

Stakeholders

- **The business owners:**
  - **1- Role:** Manage properties, users, and transactions.
  - **2- Needs:** An intuitive dashboard, efficient tools for managing data, and secure access.

Programming languages

1- html
2- css
3- Javascript
4- Python
5- SQL

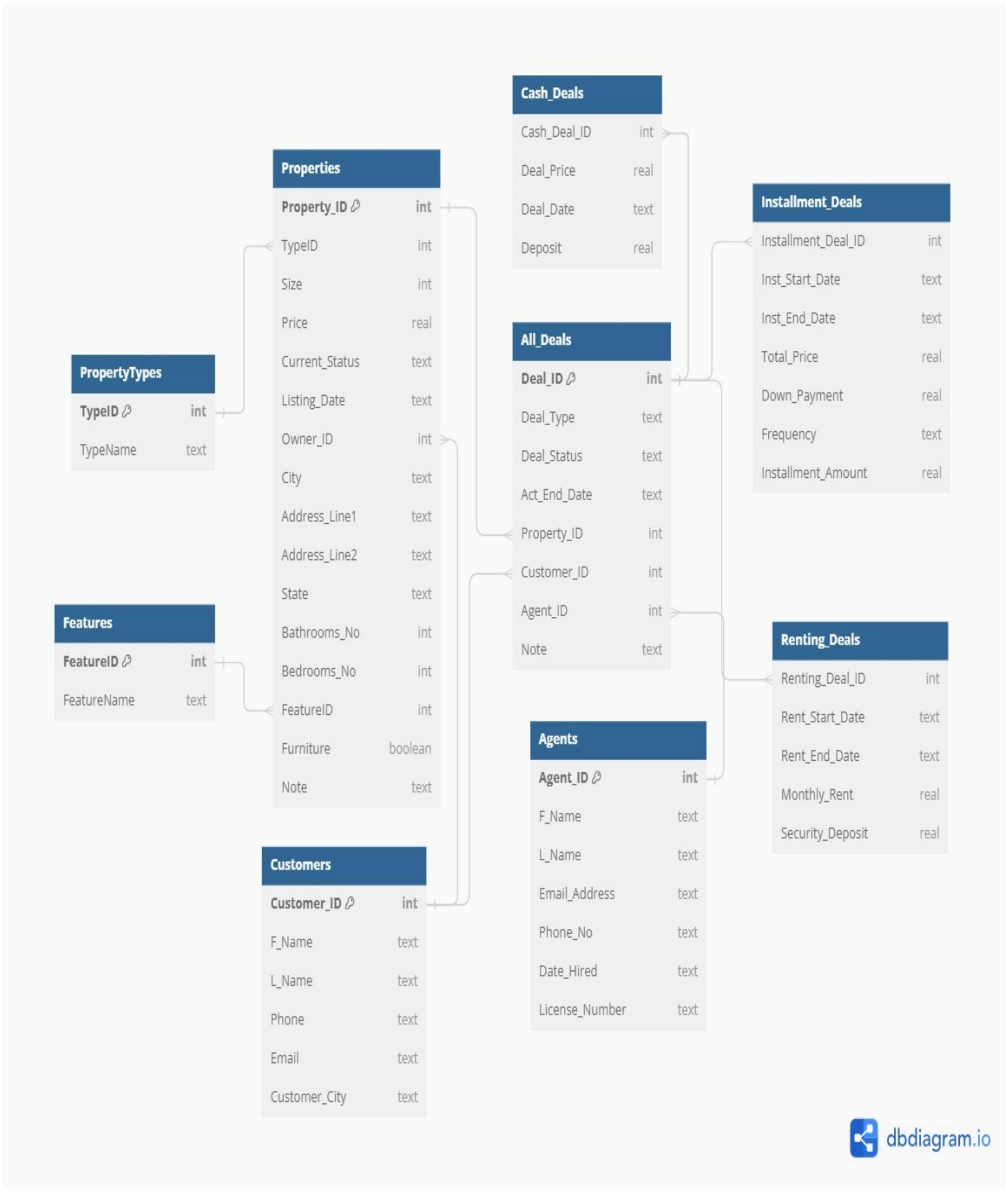
- ## Framework
  - Flask
  - SQL Server Management System (SSMS)
  - PYODBC

- ## functional & features “**Popular functions & features** ”
  - Add New Property
  - View Property
  - Update Property
  - Delete Property
  - Add Deal
    - Installment
    - Cash

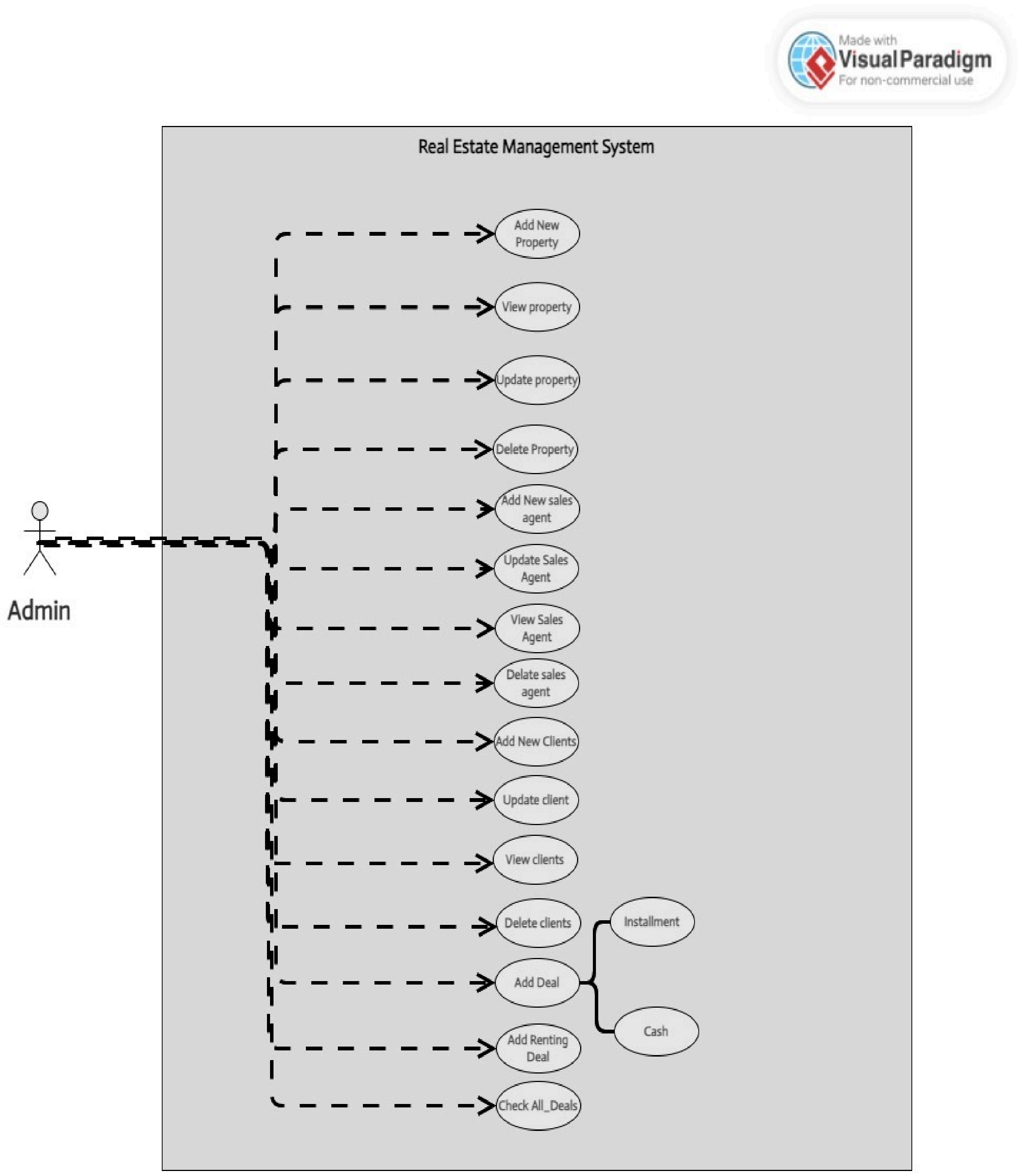
- ## Future Improvements
  - Add Agent User with special permission
  - Add Login form
  - Improve the Dashboard

- **Shots for ERD and some general functions**

- **ERD**

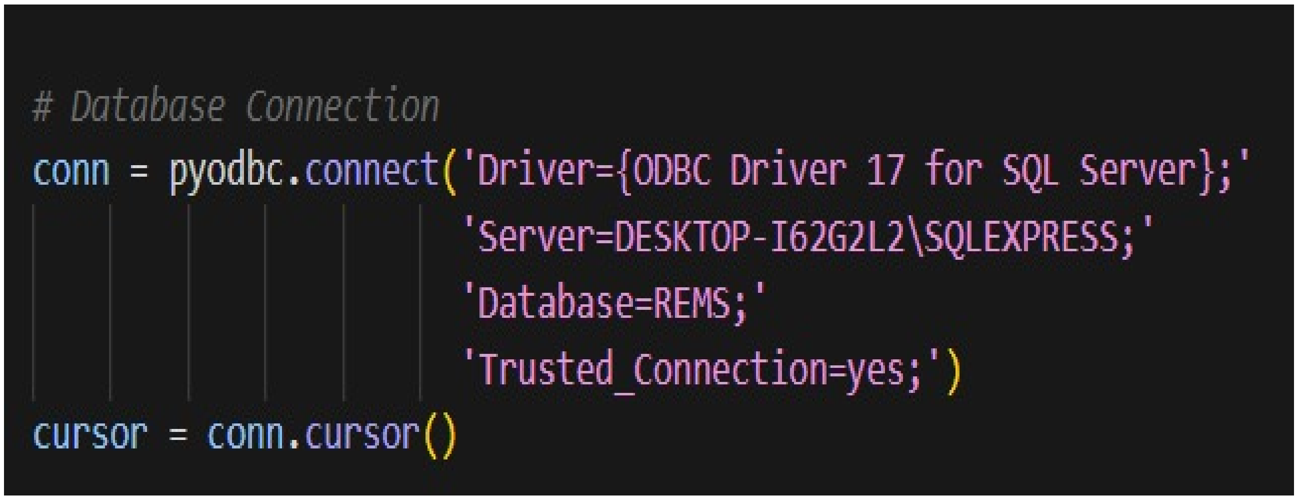


● Use case



Made with Visual Paradigm For non-commercial use
Real Estate Management System
Admin
Add New Property
View property
Update property
Delete Property
Add New sales agent
Update Sales Agent
View Sales Agent
Delate sales agent
Add New Clients
Update client
View clients
Delete clients
Add Deal
Add Renting Deal
Check All_Deals
Installment
Cash

- **Database connection string**



```
# Database Connection
conn = pyodbc.connect('Driver={ODBC Driver 17 for SQL Server};'
                      'Server=DESKTOP-I62G2L2\SQLEXPRESS;'
                      'Database=REMS;'
                      'Trusted_Connection=yes;')
cursor = conn.cursor()
```

- **Overview**

This Flask application manages a real estate system with properties, agents, customers, and deals.

## Main Components

### Properties
- List: /properties
- Add: /properties/add
- Update: /properties/update/<property_id>
- Delete: /properties/delete/<property_id>

### Agents
- List: /agents
- Add: /agents/add
- Update: /agents/update/<agent_id>
- Delete: /agents/delete/<agent_id>

### Customers
- List: /customers
- Add: /customers/add
- Update: /customers/update/<customer_id>
- Delete: /customers/delete/<customer_id>

### Deals
- List: /deals
- Add: /deals/add
- Update: /deals/update/<deal_id>
- Delete: /deals/delete/<deal_id>

### Lookup Tables
- Property Types: /property_types
- Features: /features

### Deal-Dependent Tables
- Cash Deals: /cash_deals
- Renting Deals: /renting_deals
- Installment Deals: /installment_deals

## Key Functions

1. Database Connection: get_db_connection()
2. Error Handling: Try-except blocks for database operations
3. Form Validation: Input validation in POST requests
4. Status Management: Updating property statuses based on deal status

## Templates
- Each component has corresponding HTML templates in their respective folders

## Running the Application

Execute app.run(debug=True) to start the Flask development server.

## Note
Ensure proper database setup and ODBC driver configuration before running the application.

templates html

The application uses HTML templates for rendering views. These are organized in subdirectories within the templates folder:

### Properties
- properties/list.html: Displays all properties
- properties/add.html: Form for adding a new property
- properties/update.html: Form for updating an existing property

### Agents
- agents/list.html: Shows all agents
- agents/add.html: Form for adding a new agent
- agents/update.html: Form for updating an existing agent

### Customers
- customers/list.html: Displays all customers
- customers/add.html: Form for adding a new customer
- customers/update.html: Form for updating an existing customer

### Deals
- deals/list.html: Shows all deals
- deals/add.html: Form for adding a new deal
- deals/update.html: Form for updating an existing deal
- deals/error.html: Displays error messages related to deal operations
- deals/cash_list.html: Lists all cash deals
- deals/renting_list.html: Lists all renting deals

- deals/installment_list.html: Lists all installment deals

### Property Types
- property_types/list.html: Displays all property types
- property_types/add.html: Form for adding a new property type
- property_types/update.html: Form for updating an existing property type

### Features
- features/list.html: Shows all features
- features/add.html: Form for adding a new feature
- features/update.html: Form for updating an existing feature

### Index
- index.html: The main landing page of the application

Each template extends a base template (likely layout.html or base.html, not explicitly shown in the code) which provides the common structure and styling for all pages. The templates use Jinja2 templating language for dynamic content rendering and form handling.

- **CRUD Operations on** :

  - Props Table
  - PropTypes Table
  - Prop Features Table
  - Clients Table
  - Agents Table
  - Deals Table
  - CashDeals Table
  - Installment Deals Table
  - RentinDeals Table

● Conclusion

Our Real Estate Management System is designed to streamline the operations of real estate businesses, making property management, tenant handling, and transaction tracking more efficient. By providing a user-friendly platform with advanced features, we aim to help companies improve their services and grow their business. This project reflects our commitment to using technology to create practical solutions that meet the needs of the real estate industry.