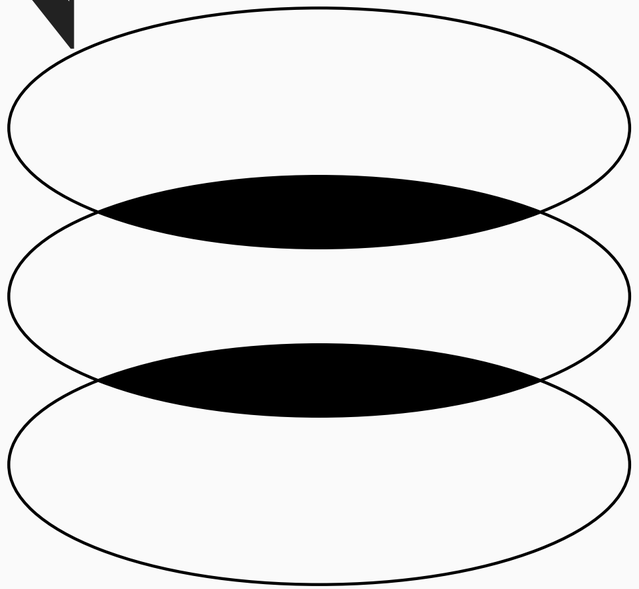
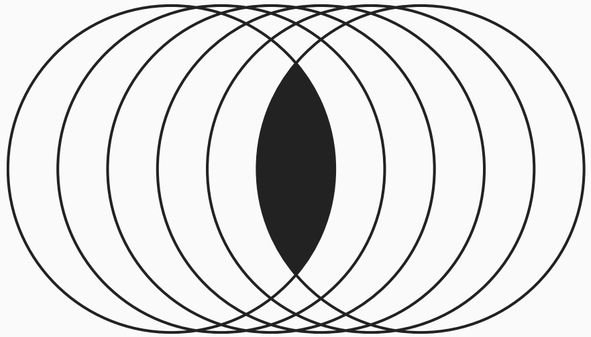
LOT RESERVATION SYSTEM



Overview

1. Introduction



Lot Reservation System

A Lot Reservation System is a digital platform for managing property lot reservations, allowing clients to browse, book, and track their reservations. Agents facilitate bookings, while lot owners manage availability and pricing.

The system ensures secure payment tracking, automates transactions, and improves efficiency by reducing manual processes. It enhances transparency and streamlines real estate reservations for a better customer experience.

1.1 Objectives

Design and develop a Lot Reservation Management system to efficiently manage lot reservations, bookings, and customer information for a real estate or property management company.

Specific Objectives:

1. Develop a user-friendly interface for easy navigation and data entry.
2. Implement a robust lot reservation management system to track availability, bookings, and customer information.
3. Create a reporting and analytics module to provide insights into reservation trends, customer behavior, and sales performance.
4. Ensure data security and integrity through robust authentication, authorization, and backup mechanisms.
5. Develop a notification system to alert customers and administrators of reservation confirmations, cancellations, and other important events.

1.2 Scope

The Lot Reservation System will include key functionalities such as lot reservation management, customer information handling, reporting and analytics, a notification system, and user access control. It will streamline the booking process, enhance security, and provide real-time insights for users.

Technically, the system will use HTML5, CSS3, and JavaScript with a framework like React or Angular for the front-end, while the back-end will be powered by Express.js or Django with a relational database like MySQL or PostgreSQL. The project will follow a 20-week timeline, covering planning, design, development, deployment, and maintenance, with cloud deployment on AWS or Google Cloud.

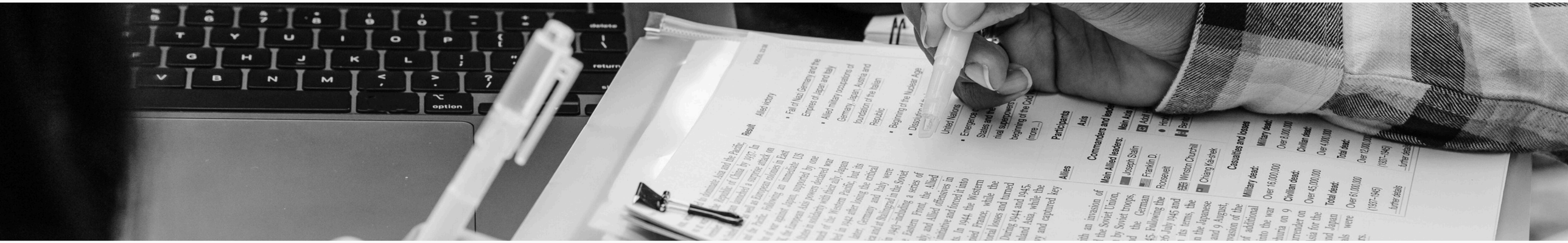
1.3 Limitations

The Lot Reservation System has several constraints that impact its functionality, technology, and resources:

- Technical Limitations:
  - Not optimized for mobile devices
  - No multi-language support
- Functional Limitations:
  - No advanced analytics or business intelligence features
  - No customer-facing portal or e-commerce functionality
- Resource Limitations:
  - Budget of approximately more or less 80,000 pesos
  - Strict 20-week project timeline
- Assumptions and Dependencies:
  - Client will provide timely and accurate feedback
  - Availability of team members and client stakeholders

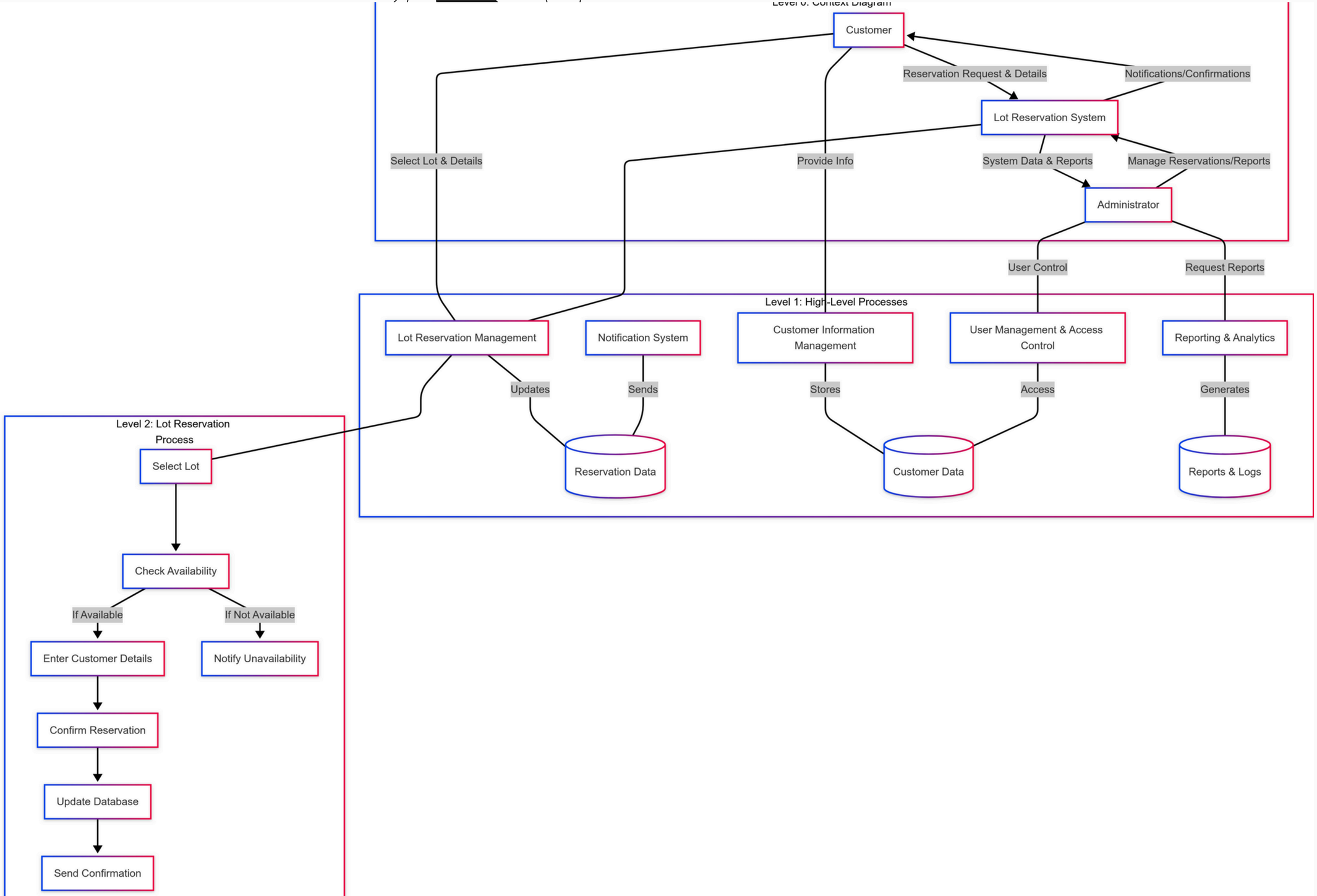
2. Process Flow

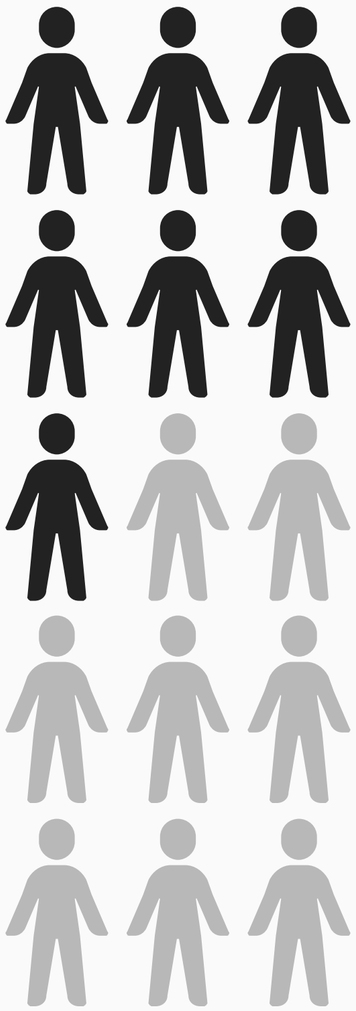
A visual representation of the steps involved in a system or workflow, illustrating how tasks, decisions, and data move from start to finish



LINK

https://www.mermaidchart.com/raw/846fc7ad-5b7f-41d7-a3e9-eld9bl52c959?theme=light&version=v0.1&format=svg


Level 0: Context Diagram
Customer
Reservation Request & Details
Notifications/Confirmations
Lot Reservation System
Select Lot & Details
Provide Info
System Data & Reports
Manage Reservations/Reports
Administrator
User Control
Request Reports
Level 1: High-Level Processes
Lot Reservation Management
Notification System
Customer Information Management
User Management & Access Control
Reporting & Analytics
Updates
Sends
Stores
Access
Generates
Reservation Data
Customer Data
Reports & Logs
Level 2: Lot Reservation Process
Select Lot
Check Availability
If Available
If Not Available
Enter Customer Details
Notify Unavailability
Confirm Reservation
Update Database
Send Confirmation



3. Database Design

The foundation of a well-structured database system, ensuring that data is stored efficiently and accurately. It involves defining tables, attributes, relationships, and constraints to maintain data integrity and support the application's functionality. A well-designed database minimizes redundancy, improves performance, and ensures consistency across the system.

The design process includes identifying key entities, establishing relationships, and selecting appropriate data types and constraints. Normalization techniques help eliminate unnecessary duplication, while indexing enhances query performance. A properly designed database ensures scalability, making it easier to manage, retrieve, and update data as the system grows.

```
CREATE TABLE Users (
    UserID INT IDENTITY(1,1) PRIMARY KEY,
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    MiddleName VARCHAR(50),
    Email VARCHAR(100) UNIQUE NOT NULL,
    Password VARCHAR(255) NOT NULL,
    Phone VARCHAR(20),
    Role VARCHAR(20) CHECK (Role IN ('Agent',
'Client', 'LotOwner'))
    NOT  NULL

);
```

```
CREATE TABLE Clients (
    ClientID INT PRIMARY KEY FOREIGN KEY REFERENCES
    Users(UserID),
     Address VARCHAR(255)
);
```

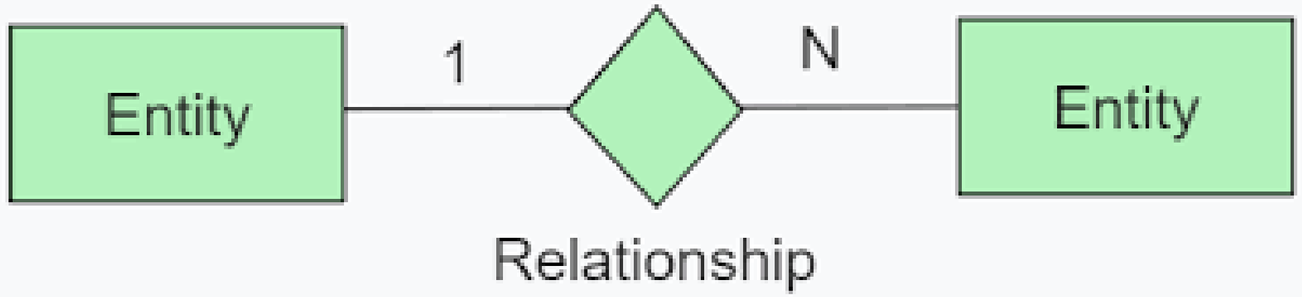
```
CREATE TABLE Agents (
     AgentID INT PRIMARY KEY FOREIGN KEY
     REFERENCES Users(UserID),
     LicenseNumber VARCHAR(50) NOT NULL
);
```

```
CREATE TABLE LotOwners (
     LotOwnerID INT PRIMARY KEY FOREIGN KEY REFERENCES
    Users(UserID),
    CompanyName VARCHAR(100)
);
```

```
CREATE TABLE Reservations (
    ReservationID INT IDENTITY(1,1) PRIMARY KEY,
    ClientID INT FOREIGN KEY REFERENCES Clients(ClientID) ON DELETE CASCADE,
    LotID INT FOREIGN KEY REFERENCES Lots(LotID) ON DELETE CASCADE,
    AgentID INT FOREIGN KEY REFERENCES Agents(AgentID) ON DELETE SET NULL,
    ReservationDate DATETIME DEFAULT GETDATE(),
    Status VARCHAR(20) CHECK (Status IN ('Pending', 'Confirmed', 'Cancelled')) DEFAULT 'Pending'
);
CREATE TABLE Lots (
    LotID INT IDENTITY(1,1) PRIMARY KEY,
    LotNumber VARCHAR(20) UNIQUE NOT NULL,
    OwnerID INT FOREIGN KEY REFERENCES LotOwners(LotOwnerID) ON DELETE CASCADE,
    Size DECIMAL(10,2) NOT NULL,
    Price DECIMAL(18,2) NOT NULL,
    Status VARCHAR(20) CHECK (Status IN ('Available', 'Reserved', 'Sold')) DEFAULT 'Available'
);
CREATE TABLE Payments (
    PaymentID INT IDENTITY(1,1) PRIMARY KEY,
    ReservationID INT FOREIGN KEY REFERENCES Reservations(ReservationID) ON DELETE CASCADE,
    AmountPaid DECIMAL(18,2) NOT NULL,
    PaymentDate DATETIME DEFAULT GETDATE(),
    PaymentMethod VARCHAR(50) CHECK (PaymentMethod IN ('Cash', 'Credit Card', 'Bank Transfer')) NOT NULL,
    Status VARCHAR(20) CHECK (Status IN ('Pending', 'Completed', 'Failed')) DEFAULT 'Pending'
);
```

4. Conceptual Framework

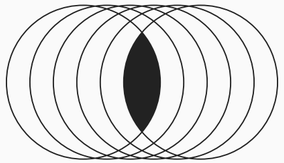
provides a structured approach to understanding the relationships between key components of the Lot Reservation System, including users, lots, reservations, and payments. It defines how different entities interact within the system, ensuring a clear and logical flow of data and processes.



4.1


User
UserID INT
FirstName VARCHAR(50)
LastName VARCHAR(50)
MiddleName VARCHAR(50)
Email VARCHAR(100)
Password VARCHAR(255)
Phone VARCHAR(20)
Role VARCHAR(20), CHECK (Role IN ('Agent', 'Client', 'LotOwner'))
Clients
ClientID INT, FK -> Users.UserID
Address VARCHAR(255)
Agents
AgentID INT, FK -> Users.UserID
LicenseNumber VARCHAR(50)
LotOwners
LotOwnerID INT, FK -> Users.UserID
CompanyName VARCHAR(100)
Lots
LotID INT
LotNumber VARCHAR(20)
OwnerID INT, FK -> LotOwners.LotOwnerID
Size DECIMAL(10,2)
Price DECIMAL(18,2)
Status VARCHAR(20), CHECK (Status IN ('Available', 'Reserved', 'Sold')), DEFAULT 'Available'
Reservations
ReservationID INT
ClientID INT, FK -> Clients.ClientID
LotID INT, FK -> Lots.LotID
AgentID INT, FK -> Agents.AgentID
ReservationDate DATETIME, DEFAULT GETDATE()
Status VARCHAR(20), CHECK (Status IN ('Pending', 'Confirmed', 'Cancelled')), DEFAULT 'Pending'
Payments
PaymentID INT
ReservationID FK -> Reservations.ReservationID
AmountPaid DECIMAL(18,2)
PaymentDate DATETIME, DEFAULT GETDATE()
PaymentMethod VARCHAR(50), CHECK (PaymentMethod IN ('Cash', 'Credit Card', 'Bank Transfer'))
Status VARCHAR(20), CHECK (Status IN ('Pending', 'Completed', 'Failed')), DEFAULT 'Pending'

5. Data Dictionary



a structured repository that defines and describes the data elements used in a database, including their attributes, relationships, and constraints. It serves as a reference for developers, database administrators, and stakeholders, ensuring consistency and clarity in data management. The data dictionary typically includes details such as table names, column names, data types, constraints, and relationships between tables, helping to standardize data usage across the system.

5.1

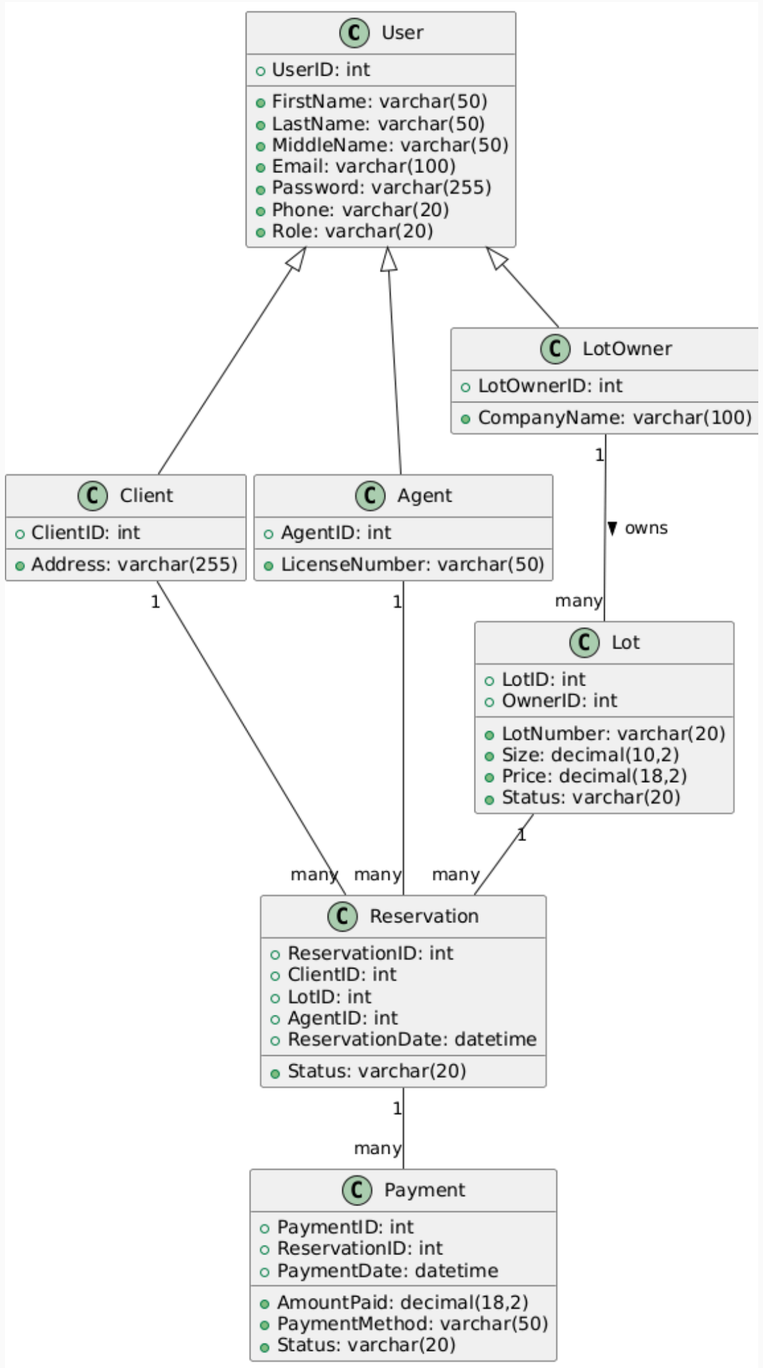
LOT RESERVATION SYSTEM			
Table Name	**Column Name**	**Data Type**	**Description**
User	UserID	INT (Primary Key)	Unique identifier for a user.
	FirstName	VARCHAR(50)	User's first name.
	LastName	VARCHAR(50)	User's last name.
	MiddleName	VARCHAR(50)	User's middle name (if any).
	Email	VARCHAR(100)	User's email address.
	Password	VARCHAR(255)	Encrypted password for authentication.
	Phone	VARCHAR(20)	User's contact number.
	Role	VARCHAR(20)	Defines the user type: Agent, Client, or LotOwner.
Clients	ClientID	INT (Primary Key, Foreign Key to UserID)	Unique identifier for a client.
	Address	VARCHAR(255)	Client's residential address.
Agents	AgentID	INT (Primary Key, Foreign Key to UserID)	Unique identifier for an agent.
	LicenseNumber	VARCHAR(50)	Agent's license number for verification.
LotOwners	LotOwnerID	INT (Primary Key, Foreign Key to UserID)	Unique identifier for a lot owner.
	CompanyName	VARCHAR(100)	Name of the company that owns the lot.

5.1

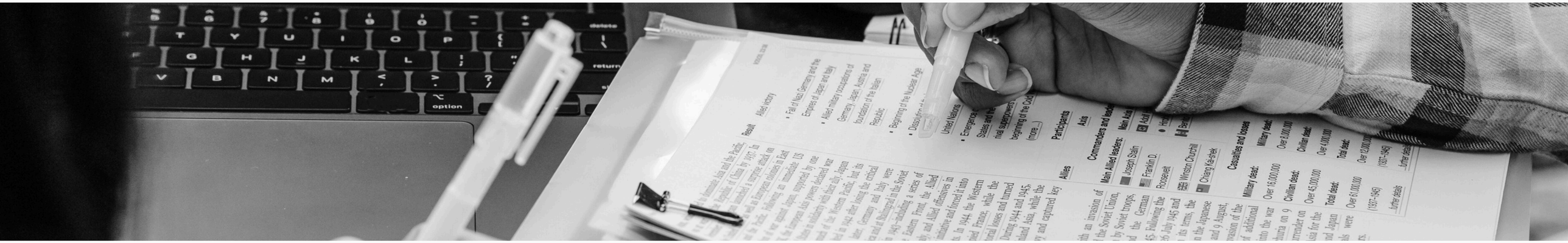
Lots	LotID	INT (Primary Key)	Unique identifier for a lot.
	LotNumber	VARCHAR(20)	Unique number assigned to the lot.
	OwnerID	INT (Foreign Key to LotOwners.LotOwnerID)	The owner of the lot.
	Size	DECIMAL(10,2)	Size of the lot in square meters.
	Price	DECIMAL(18,2)	Price of the lot.
	Status	VARCHAR(20)	Status of the lot: Available, Reserved, or Sold.
Reservations	ReservationID	INT (Primary Key)	Unique identifier for a reservation.
	ClientID	INT (Foreign Key to Clients.ClientID)	The client who made the reservation.
	LotID	INT (Foreign Key to Lots.LotID)	The lot being reserved.
	AgentID	INT (Foreign Key to Agents.AgentID)	The agent handling the reservation.
	ReservationDate	DATETIME (Default GETDATE())	Date and time of the reservation.
	Status	VARCHAR(20)	Status of the reservation: Pending, Confirmed, or Cancelled.
Payments	PaymentID	INT (Primary Key)	Unique identifier for a payment.
	ReservationID	INT (Foreign Key to Reservations.ReservationID)	The reservation related to the payment.
	AmountPaid	DECIMAL(18,2)	Amount paid by the client.
	PaymentDate	DATETIME (Default GETDATE())	Date and time of the payment.
	PaymentMethod	VARCHAR(50)	Mode of payment: Cash, Credit Card, or Bank Transfer.
	Status	VARCHAR(20)	Status of the payment: Pending, Completed, or Failed.

6. Class Diagram(UML)

A class diagram visually represents the structure of a system by defining its classes, attributes, methods, and the relationships between them



7. CRUD Matrix



A visual representation of the steps involved in a system or workflow, illustrating how tasks, decisions, and data move from start to finish

7.1

Matrix

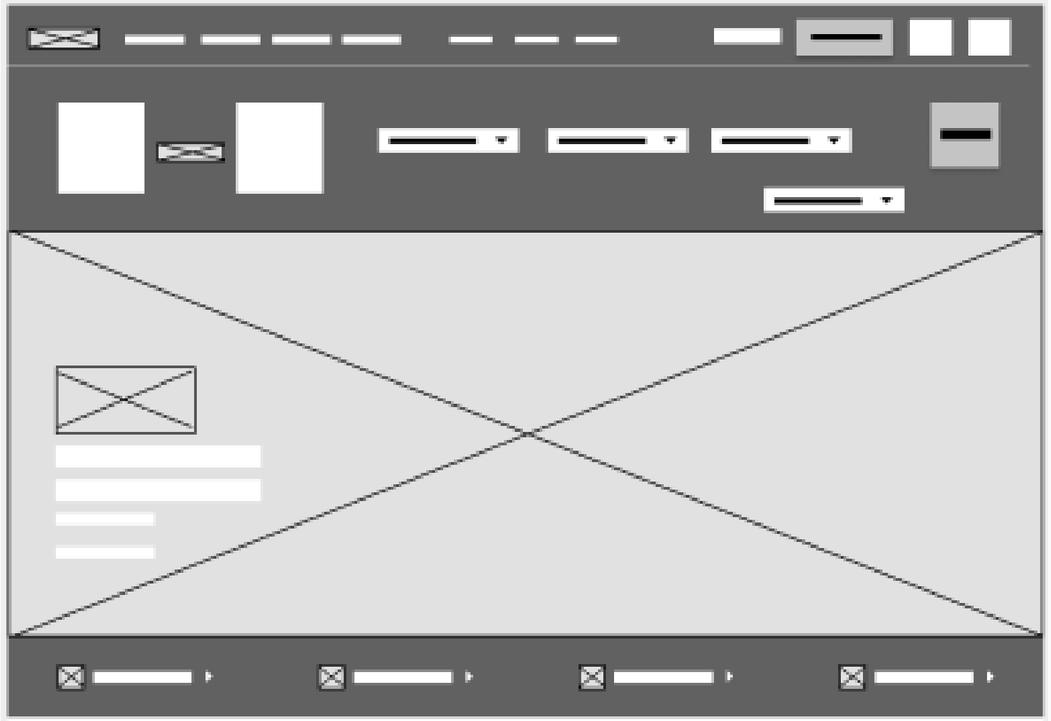
Functionality	Users	Clients	Agents	Lot Owners	Lots	Reservations	Payments
Register a user	C						
View User Profile	R	R	R	R			
Update User Profile	U	U	U	U			
Delete User Account	D	D	D	D			
Add Client Record		C					
View Client List		R	R	R			
Update Client Info		U					
Delete Client Record		D					
Add Agent Record			C				
View Agents List			R	R			
Update Agent Info			U				
Delete Agent Record			D				
Add Lot Owner Record				C			
View Lot Owners List			R	R			
Update Lot Owner Info				U			
Delete Lot Owner Record				D			

7.1

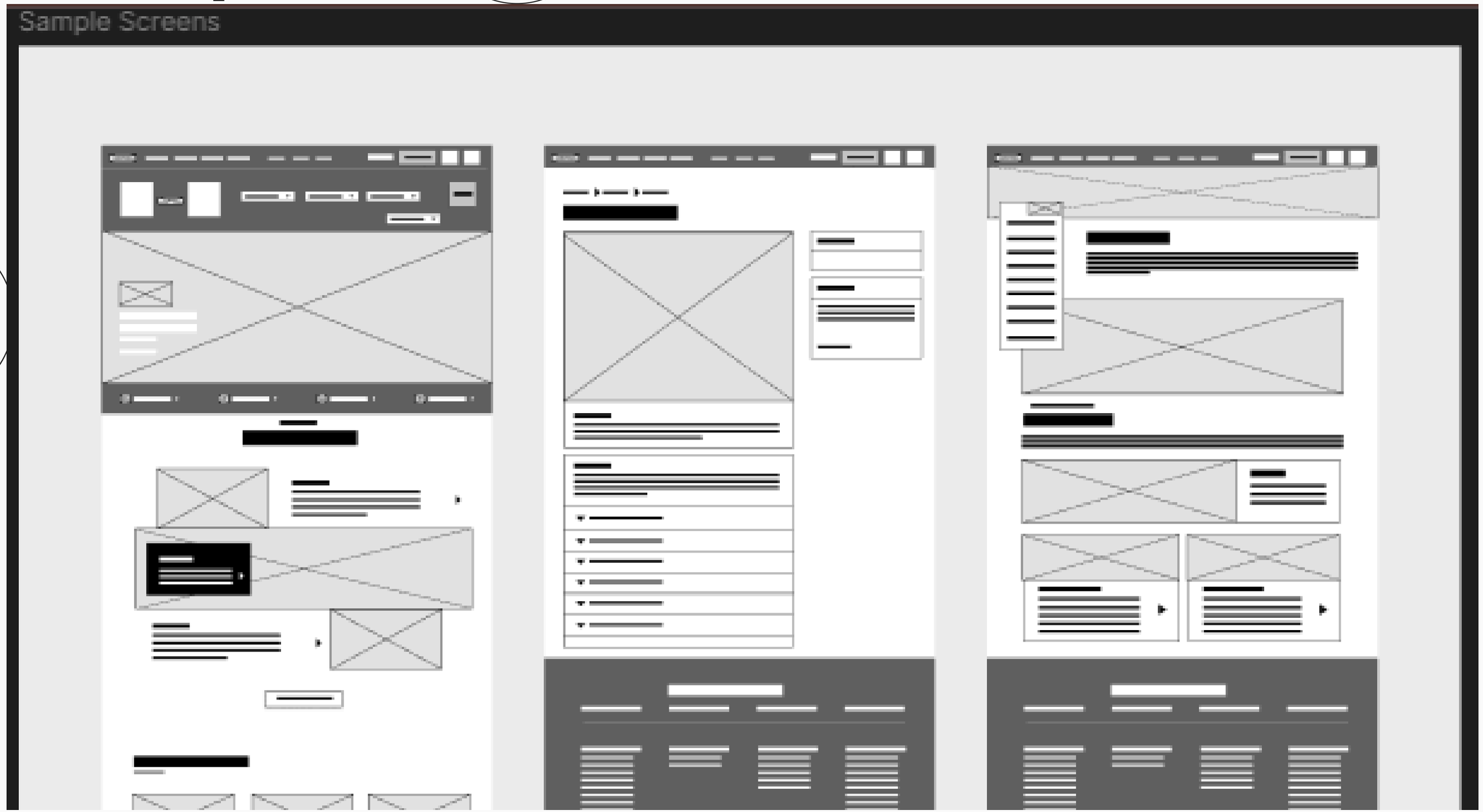
Add a Lot							C					
View Lot Details		R		R		R		R				
Update Lot Information								U				
Delete a Lot								D				
Change Lot Status								U				
Reserve a Lot		C								C		
View Reservation List		R		R		R				R		
Approve a Reservation				U						U		
Cancel a Reservation		U	D	U						U	D	
Add a Payment										C		C
View Payment Details		R		R		R				R		R
Update Payment Status												U
Delete a Payment												D

8. Wireframe

a visual blueprint of a system's user interface, outlining the layout and structure of key components without focusing on design details. It helps in planning user interactions, ensuring a clear and functional flow before development begins



8.1 Sample Screens



THANKYOU!

- Jhon Lester