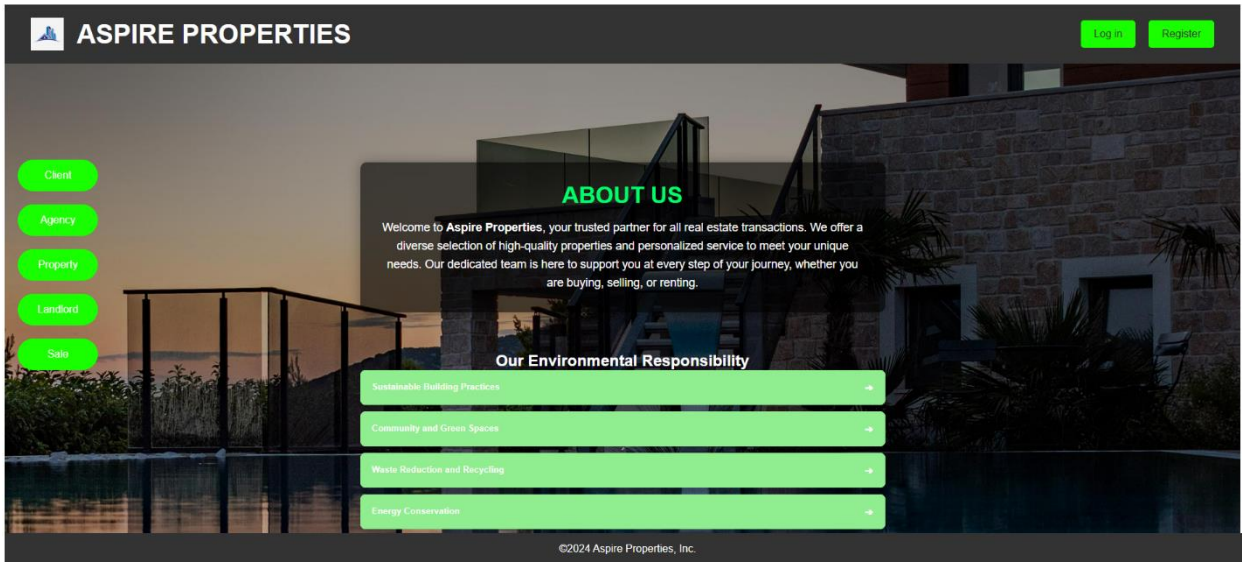
PARIS PANTHÉON-ASSAS UNIVERSITÉ

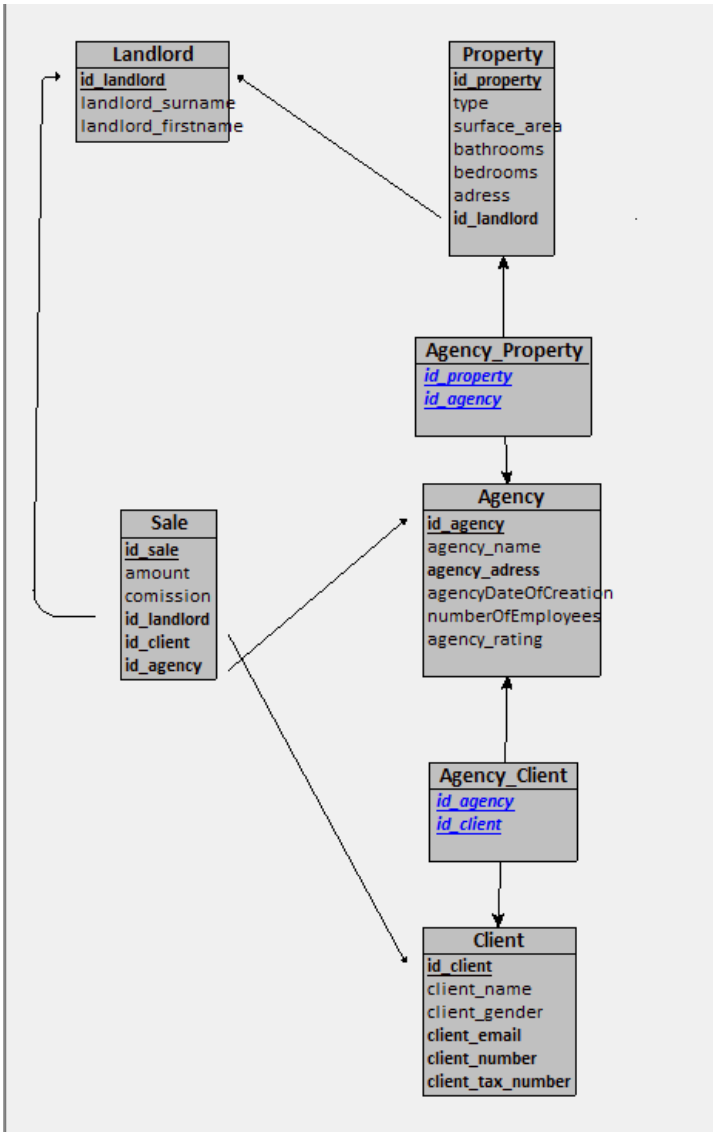
Members: Sulpice Alban, Abouleila Selim

Title: UML DIAGRAMS

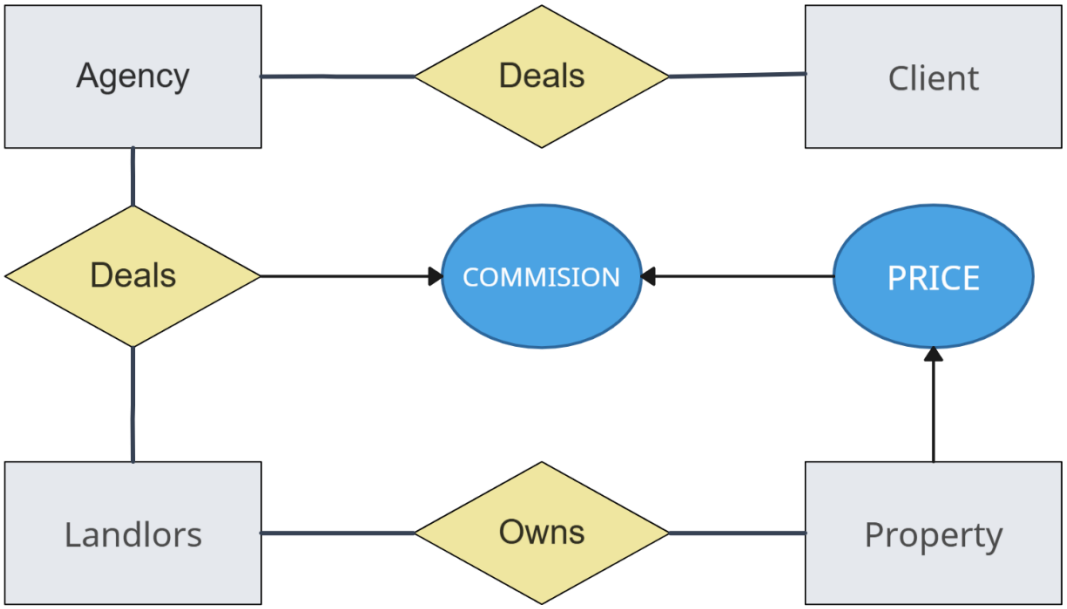


The aim of this agency project website is to streamline the operations of a real estate management system, enhancing the sale process between the agencies, landlords, clients, and properties. By integrating modern technology and user-friendly interfaces, the website seeks to eliminate manual processes, reduce inefficiencies, and provide a transparent and intuitive experience for all stakeholders.

Ultimately, the agency project website aims to bridge the gap between landlords and clients while empowering agencies to deliver better services. Its goal is to create a unified platform that fosters efficiency, transparency, and ease of use, driving the real estate sector into the digital era.



The diagram appears to be a flowchart or process loop that emphasizes repetitive steps within a cycle. It shows how a certain process or set of actions feeds into itself in a continuous loop, where each action is connected to the next, creating a cycle of actions or feedback. This could relate to various systems or processes, such as quality control, iterative processes, or systems with ongoing feedback mechanisms. Without more context, it's a bit hard to specify the exact nature of the process, but the looping structure indicates a recurring, cyclical activity.



This Entity-Relationship (ER) diagram represents a real estate management system, focusing on interactions between agencies, clients, landlords, and properties. It defines key entities, their attributes, and relationships to represent the business logic.

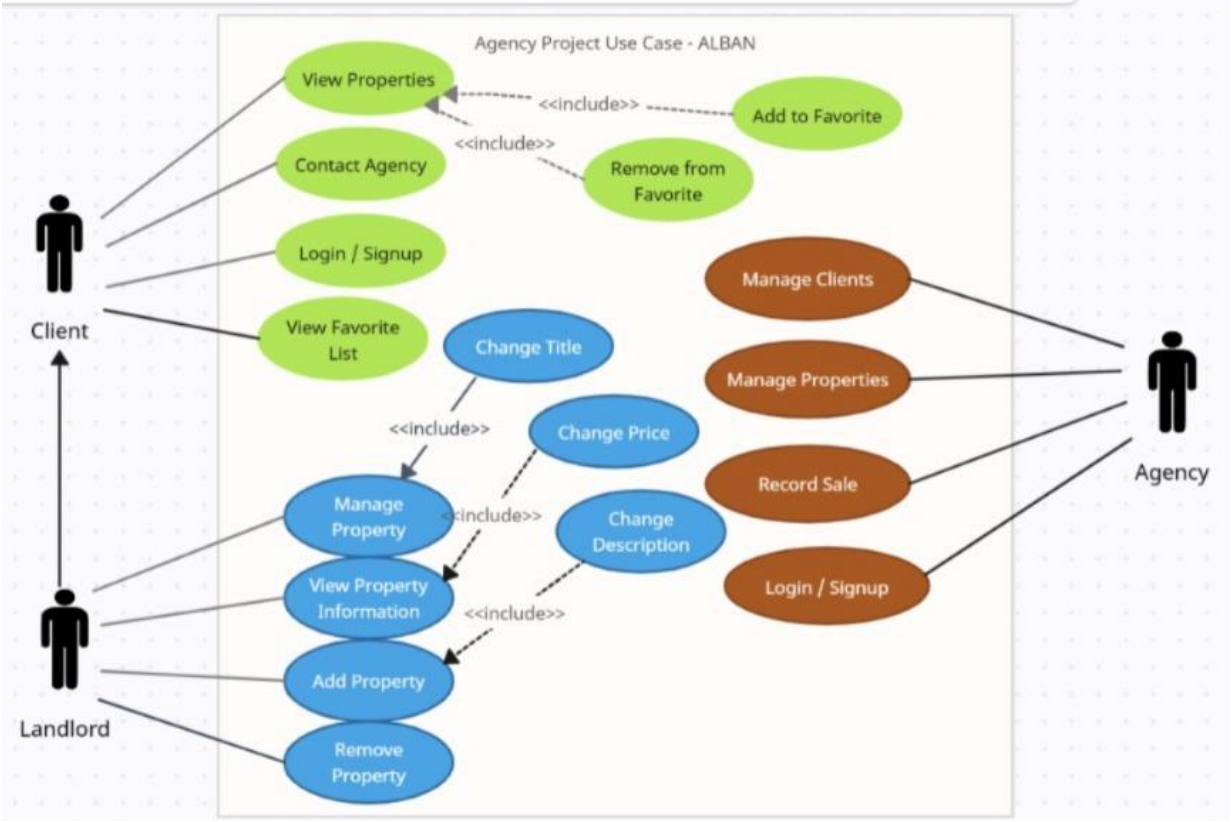
Entities:

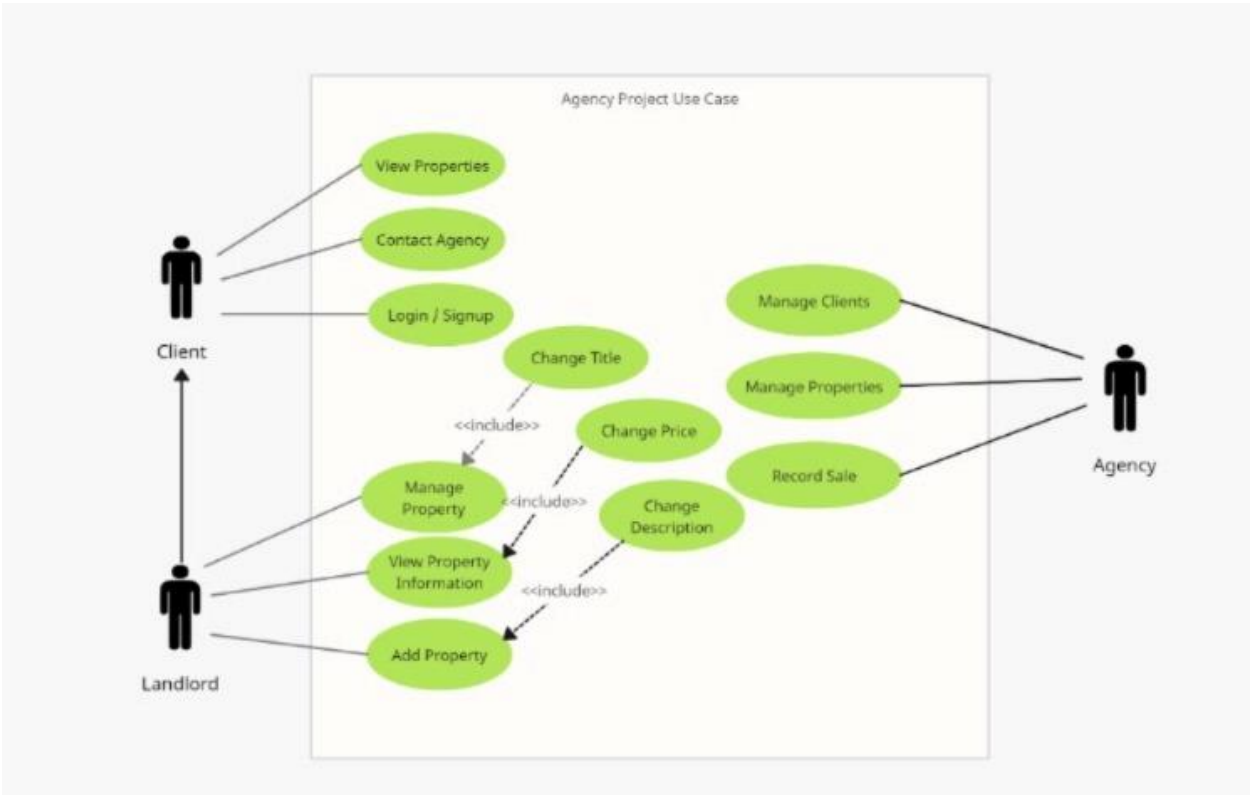
1. **Agency**:
    - Represents a real estate agency responsible for facilitating deals between landlords and clients.
2. **Client**:
    - Represents individuals or organizations looking to buy, rent, or invest in properties.
3. **Landlords**:
    - Represents property owners who collaborate with agencies to sell or rent properties.

4. **Property**:
    - Represents physical assets (houses, apartments, etc.) owned by landlords and marketed by agencies.
5. **Deals**:
    - Represents transactions or agreements involving agencies, landlords, and clients.
6. **Commission**:
    - Represents fees earned by the agency for facilitating deals.
7. **Price**:
    - Represents the financial terms agreed upon for the property within a deal.

Relationships:

1. **Agency - Deals**:
    - Agencies participate in deals by connecting clients and landlords.
2. **Deals - Client**:
    - Clients are part of deals, representing their interest in purchasing or renting properties.
3. **Deals - Landlords**:
    - Landlords participate in deals by providing their properties.
4. **Deals - Commission**:
    - A commission is tied to deals, representing the agency's earnings.
5. **Property - Price**:
    - Price is associated with a property and defined in the deal.
6. **Landlords - Owns - Property**:
    - Landlords own properties, forming the foundation of transactions.


Agency Project Use Case - ALBAN
View Properties
<<include>>
Add to Favorite
<<include>>
Contact Agency
Remove from Favorite
Login / Signup
Client
View Favorite List
Change Title
<<include>>
Change Price
Manage Property
<<include>>
Change Description
View Property Information
<<include>>
Add Property
Landlord
Remove Property
Manage Clients
Manage Properties
Record Sale
Login / Signup
Agency

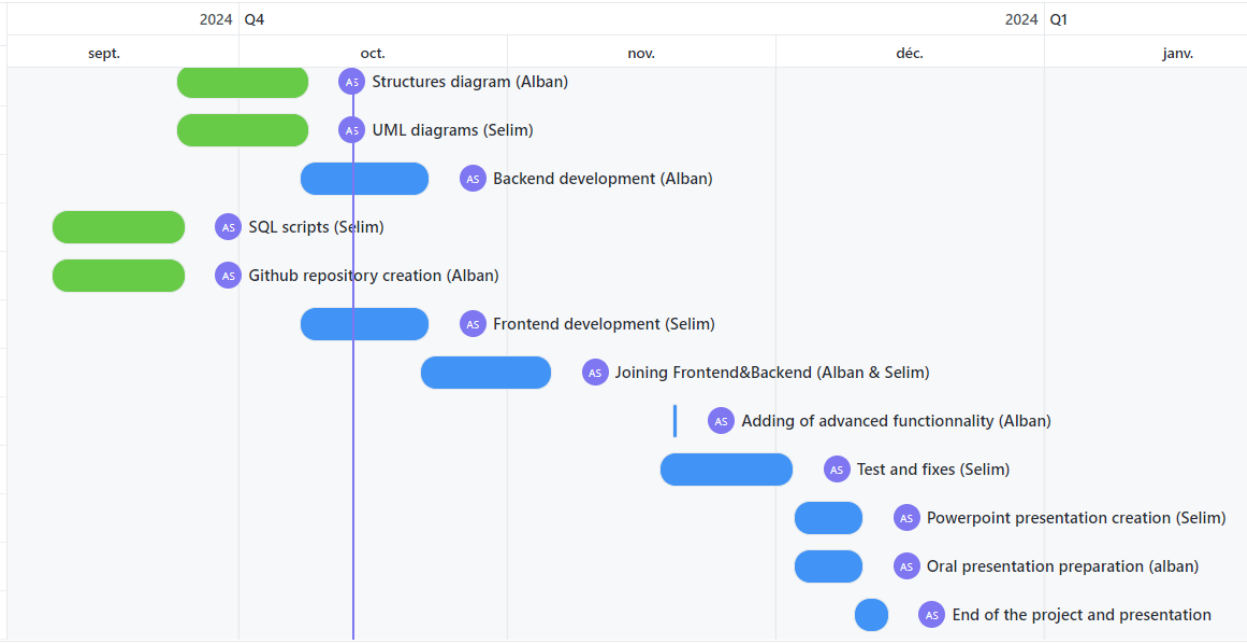

Agency Project Use Case
View Properties
Contact Agency
Login / Signup
Client
Change Title
<<include>>
Change Price
Manage Property
<<include>>
Change Description
View Property Information
<<include>>
Add Property
Landlord
Manage Clients
Manage Properties
Record Sale
Agency

These two Diagrams are **UML Use Case Diagrams** illustrating the interactions between three main actors—**Client**, **Landlord**, and **Agency**—within an agency project.

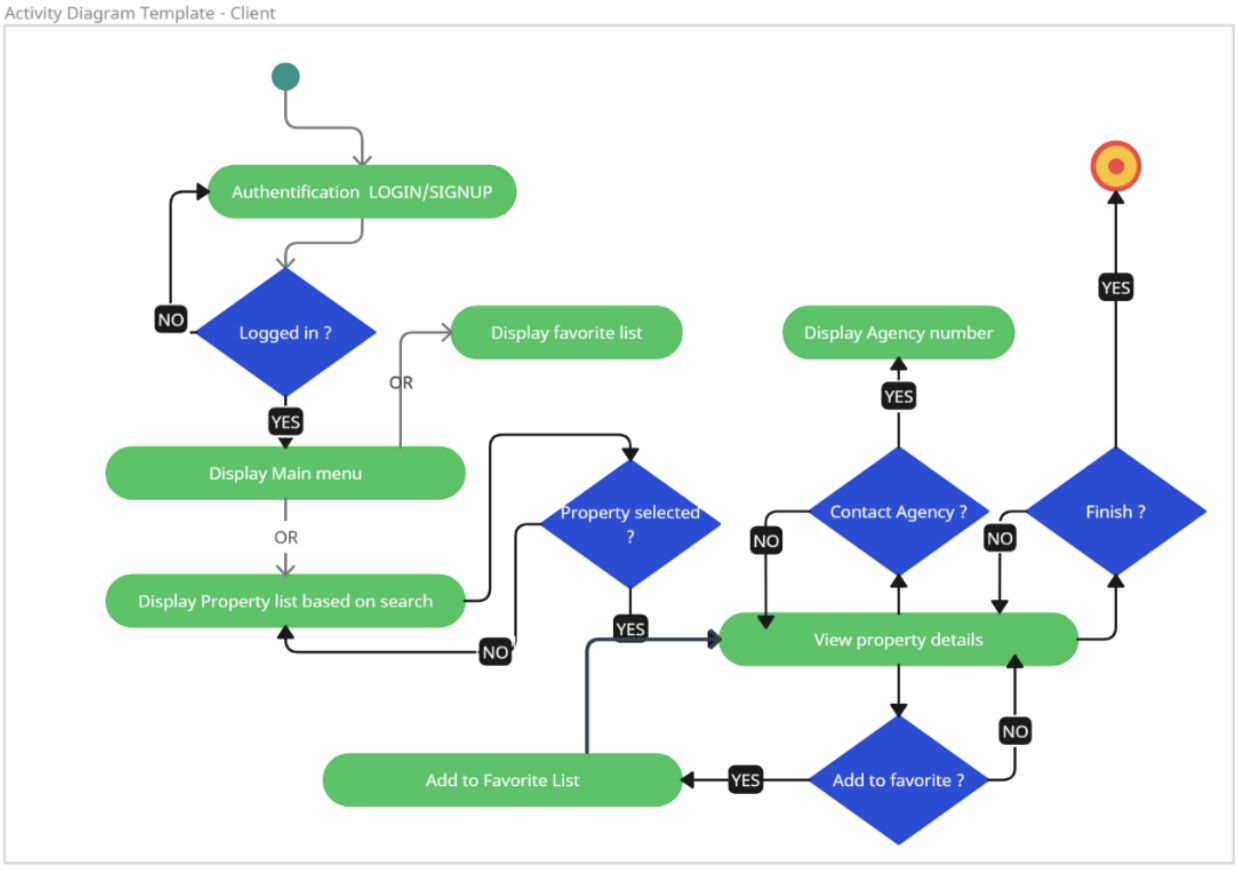
Relations:

- Use cases often include dependencies such as:
    - <<include>> relationships signify mandatory steps or sub-actions (e.g., "View Properties" includes "Add to Favorite").

This structured diagram maps out the system functionality for all involved stakeholders.



This is a **Gantt chart** that outlines a project timeline for the **Agency Project** from September 2024 to January 2025. It organizes tasks assigned to two team members, **Alban** and **Selim**, and indicates their collaboration phases.



This is an **Activity Diagram** for a **Client's process flow** in an agency system.

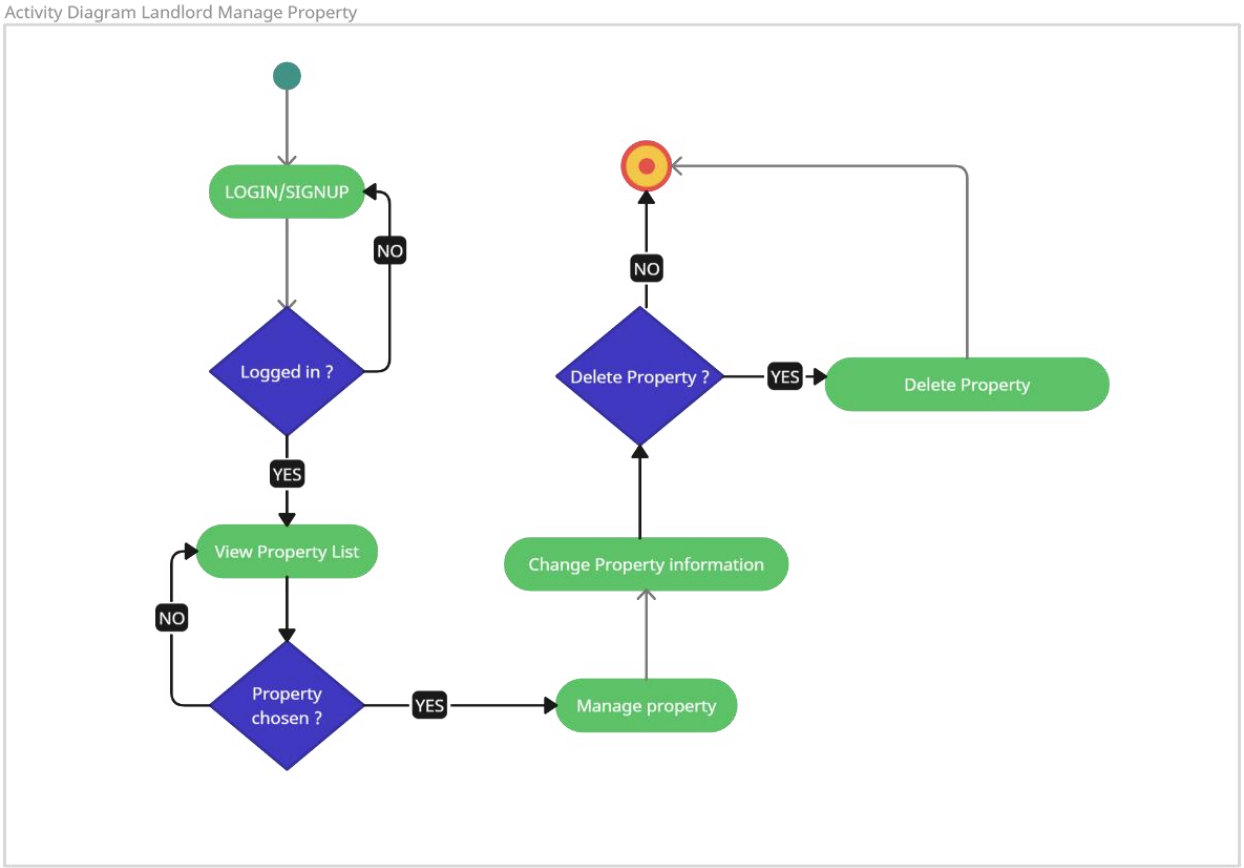
Key Flow:

1. **Login/Signup Authentication**:
    - If **not logged in**, return to authentication.
    - If **logged in**, proceed to the main menu.
2. **Display Options**:
    - View **favorite list** or search and display property lists.
3. **Property Interaction**:
    - If a **property is selected**:
        - View details.
        - Option to **add to favorites**.

- Option to **contact agency** for more information.

4. **Completion**:
    - The process ends when the user chooses to finish.

This diagram illustrates user decision points and actions while navigating the system.



This is another **Activity Diagram** focused on the **Landlord's property management process** within the system.

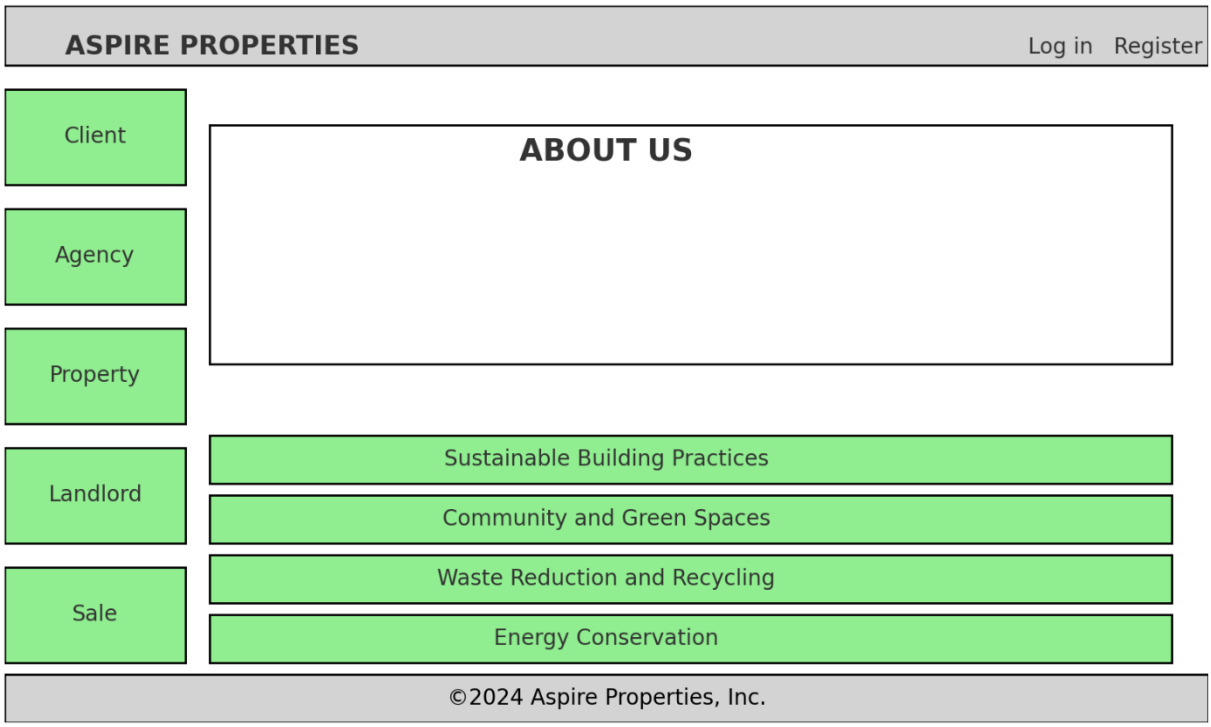
Key Flow:

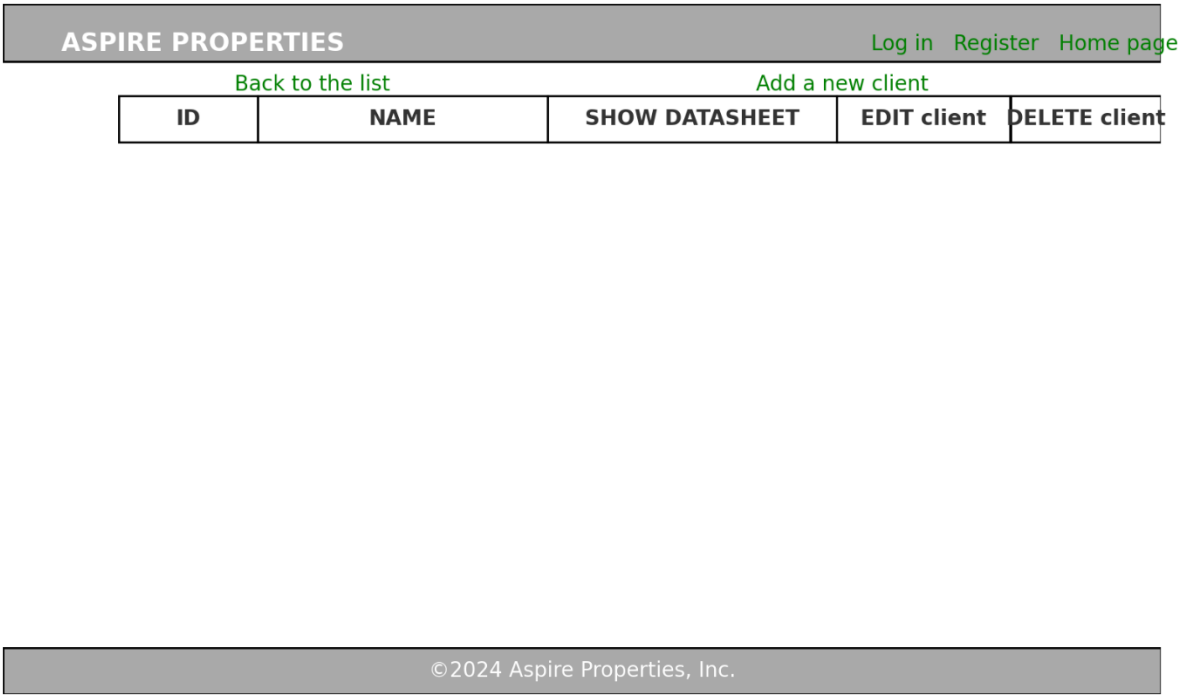
1. **Login/Signup**:
    - If **not logged in**, return to the login process.

- If **logged in**, proceed to view the property list.

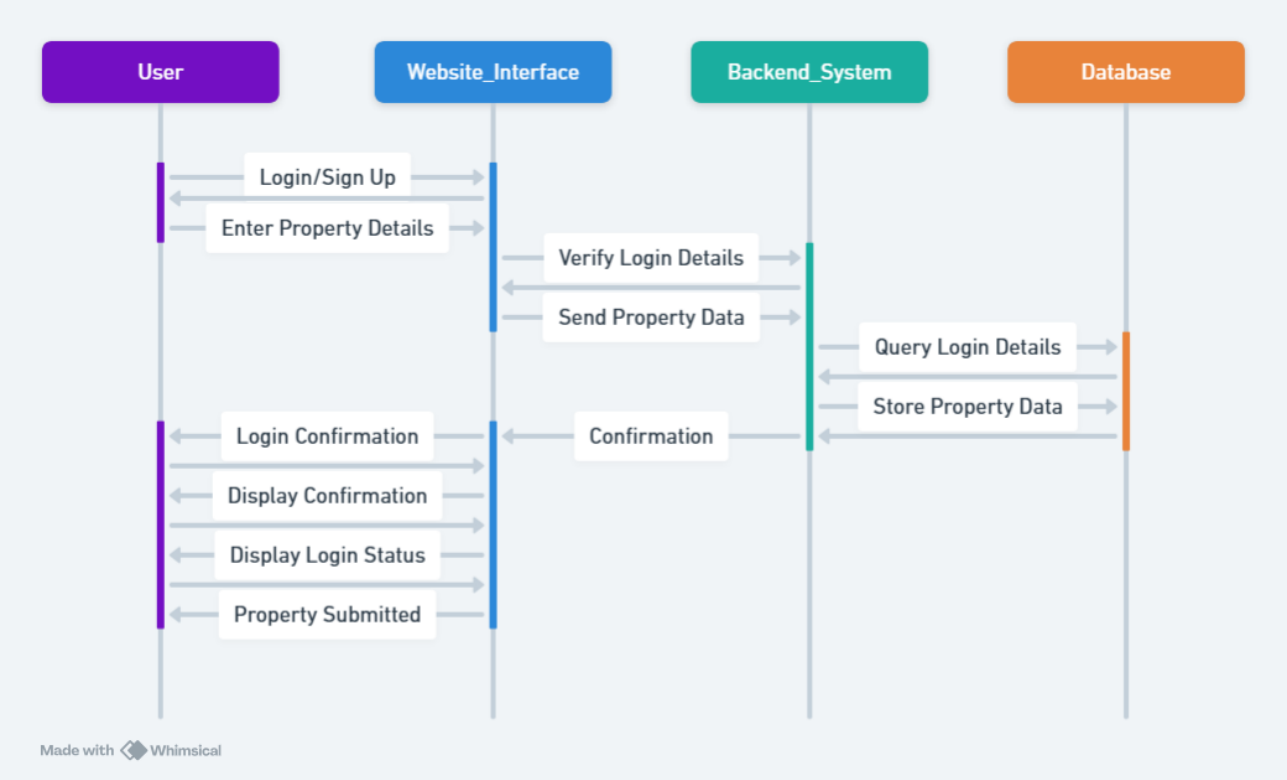
2. **Property Selection**:
    - If no property is chosen, the process halts.
    - If a property is chosen, the landlord can manage it.
3. **Property Management Options**:
    - Decide whether to:
        - **Delete the property** (if yes, the property is removed).
        - **Change property information** (e.g., update details).
4. **Completion**:
    - The process ends after managing or deleting the property.

This diagram outlines the key decision points and actions landlords can perform in the property management workflow.

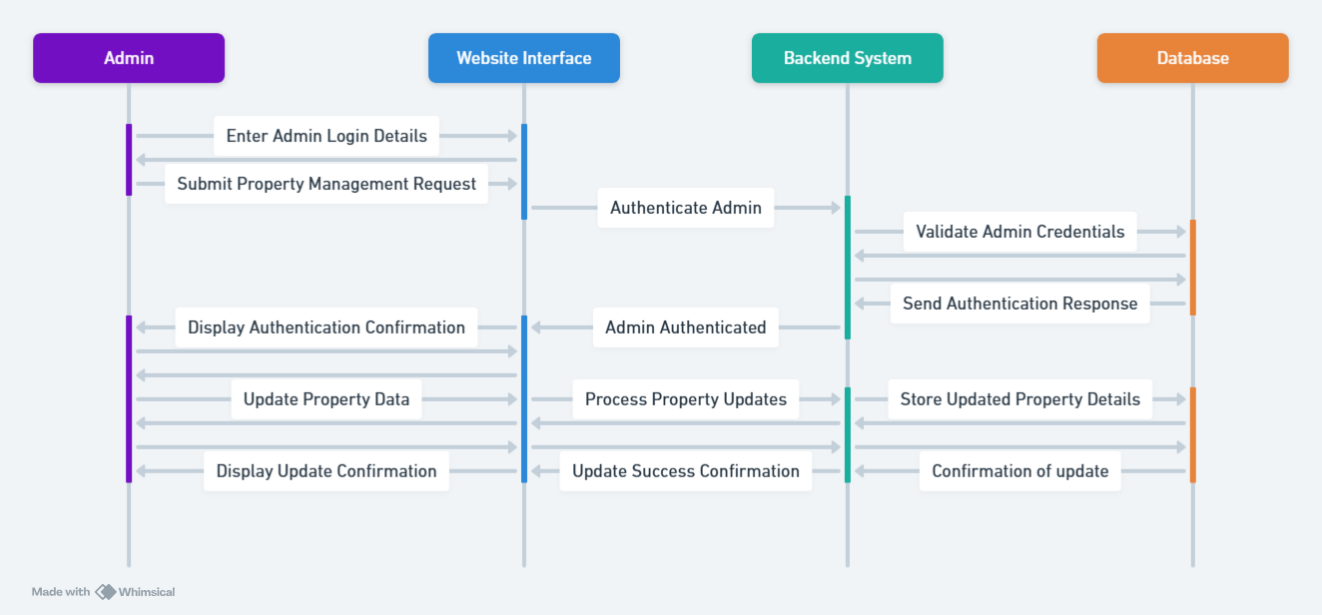




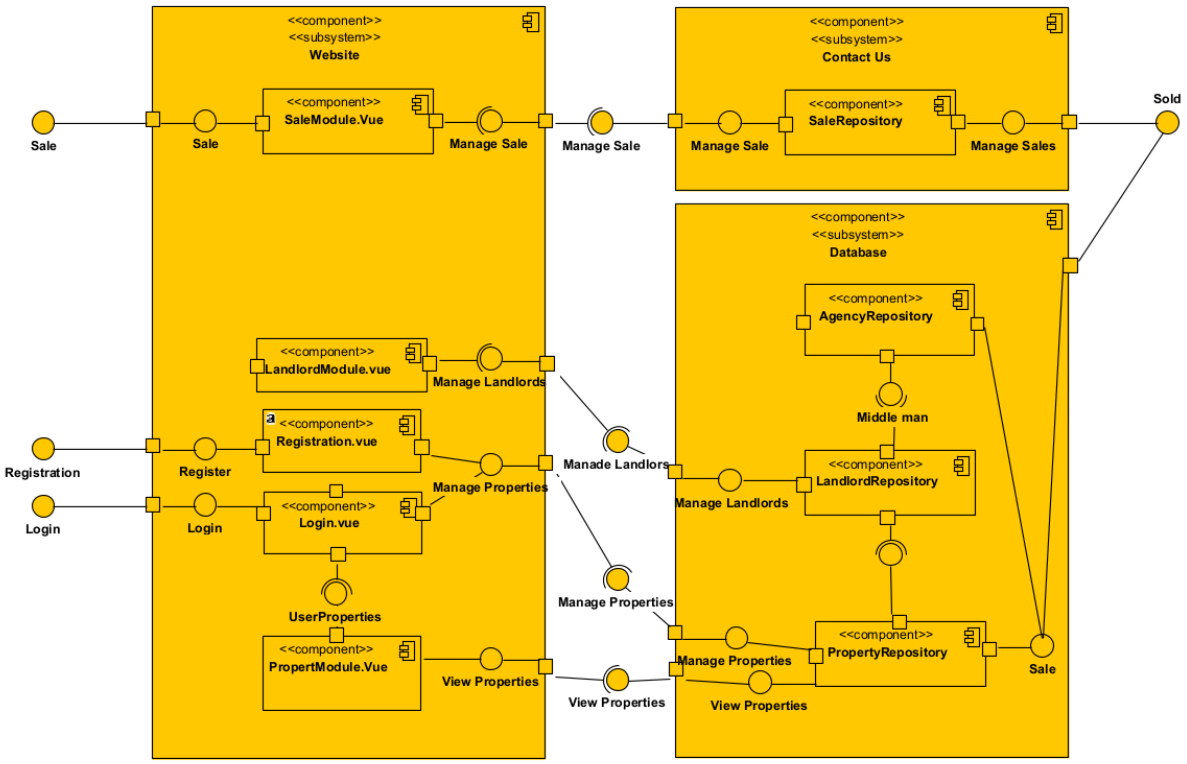
1. **First Wireframe**: A homepage layout with a navigation bar, a vertical menu, an "About Us" section, and expandable sections highlighting environmental practices.
2. **Second Wireframe**: A client management page featuring a top navigation bar, links for actions, a table with client information, and options to view, edit, or delete records.

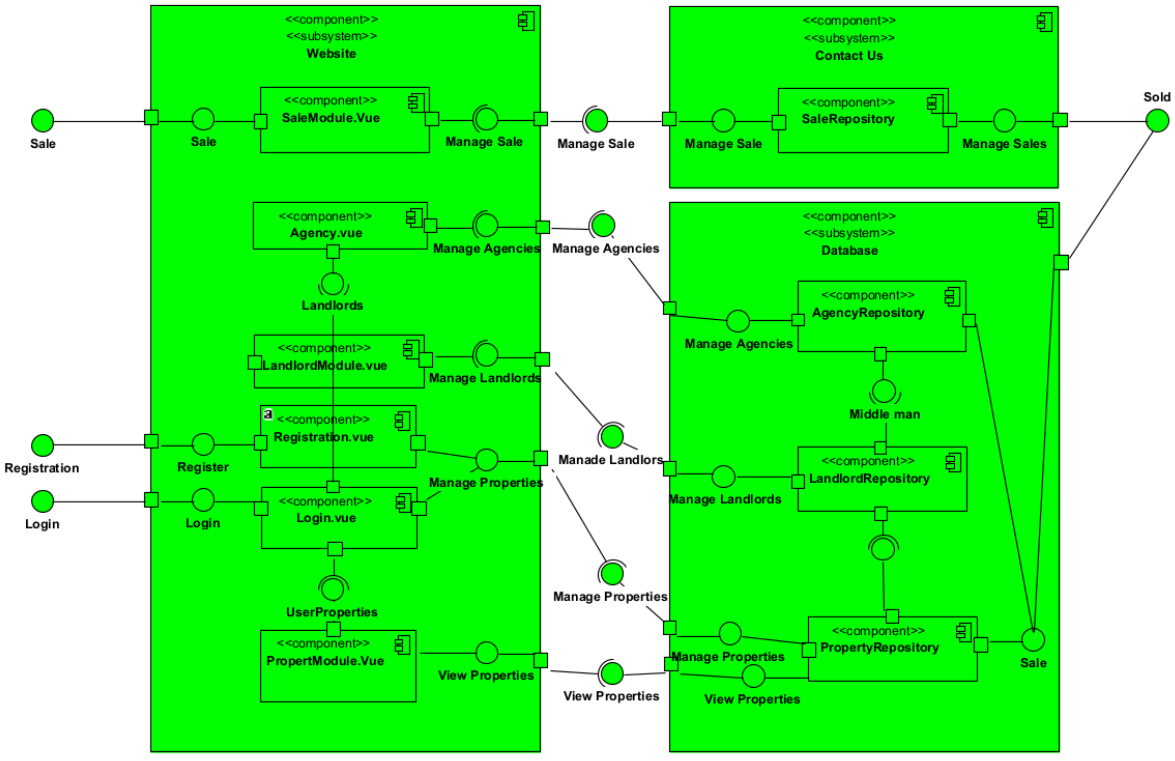


This sequence diagram illustrates the process of a user logging in or signing up, submitting property details, and the data flow through the website interface, backend system, and database. It highlights the interactions and confirmations at each step to ensure successful data submission.



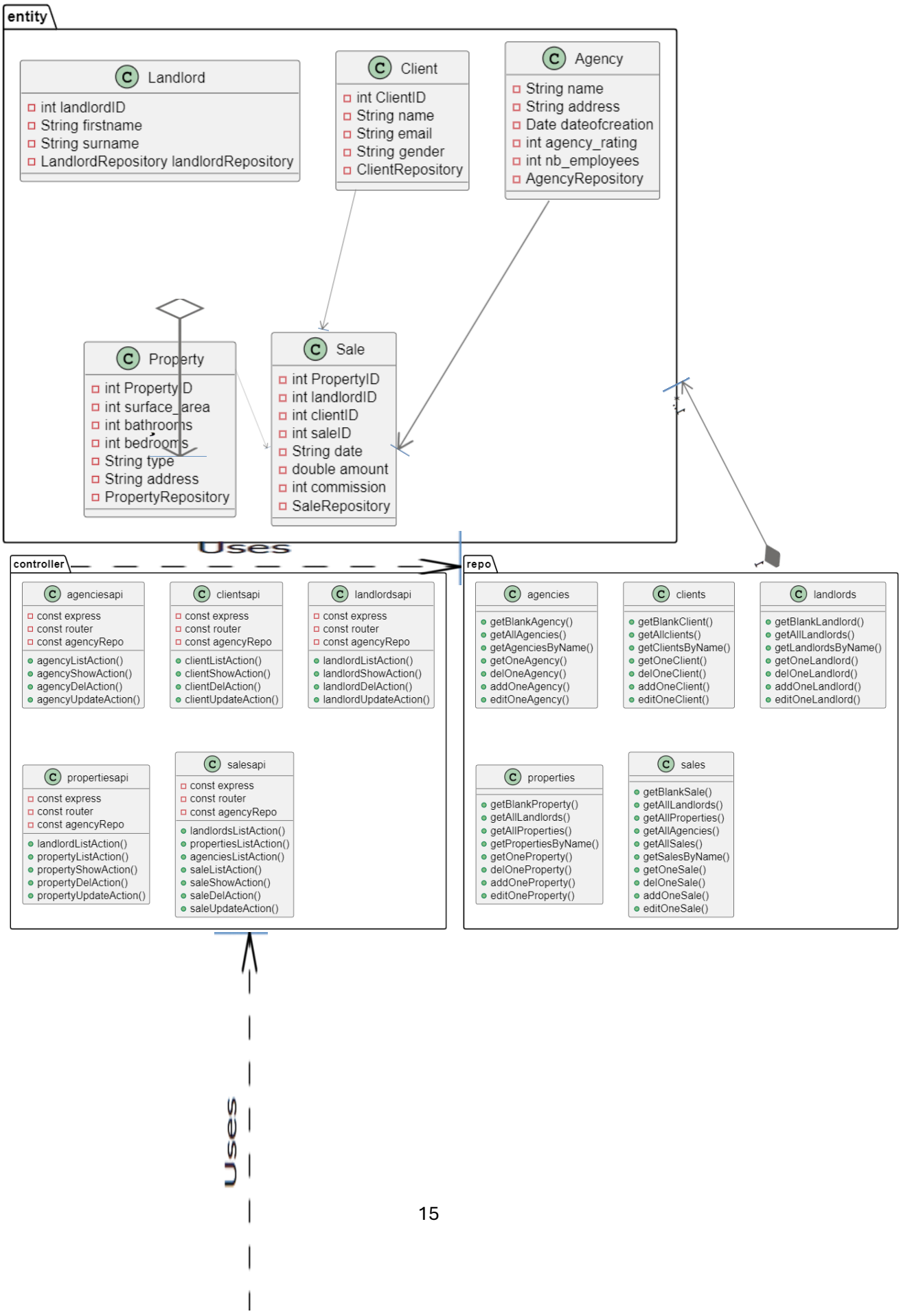
This sequence diagram outlines the process for an admin to manage property data, starting with authentication and ending with updates to the database. It highlights the interactions between the admin, website interface, backend system, and database for secure and efficient property management.

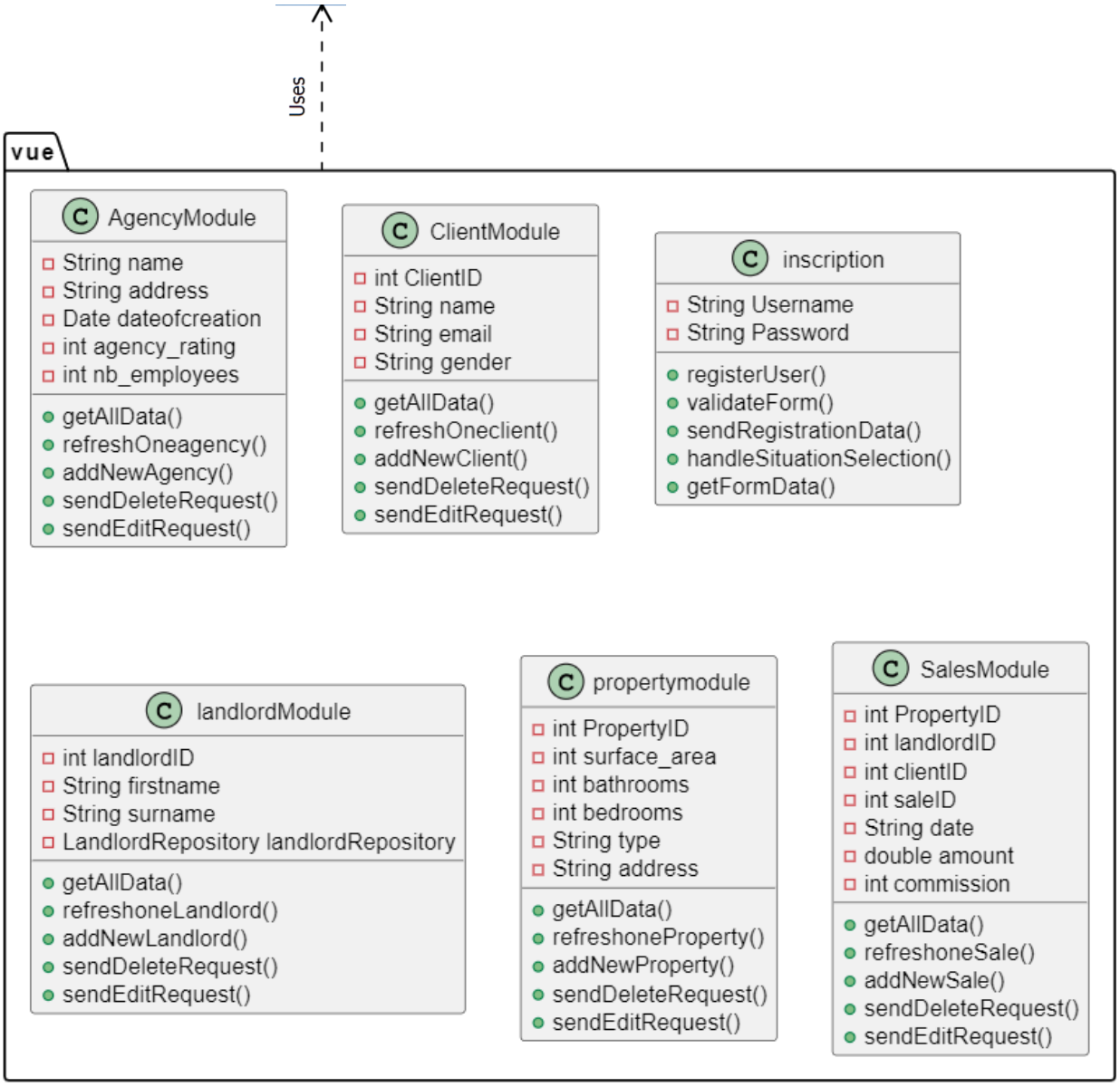

<<component>>
<<subsystem>>
Website
<<component>>
SaleModule.Vue
Sale
Sale
Manage Sale
Manage Sale
<<component>>
<<subsystem>>
Contact Us
Manage Sale
<<component>>
SaleRepository
Manage Sales
Sold
<<component>>
<<subsystem>>
Database
<<component>>
AgencyRepository
Middle man
<<component>>
LandlordRepository
Manage Landlords
<<component>>
PropertyRepository
Sale
Manage Properties
View Properties
<<component>>
LandlordModule.vue
Manage Landlords
Manade Landlors
<<component>>
Registration.vue
Registration
Register
Manage Properties
<<component>>
Login.vue
Login
Login
UserProperties
<<component>>
PropertModule.Vue
View Properties


<<component>>
<<subsystem>>
Website
<<component>>
SaleModule.Vue
Sale
Sale
Manage Sale
Manage Sale
<<component>>
<<subsystem>>
Contact Us
Manage Sale
<<component>>
SaleRepository
Manage Sales
Sold
<<component>>
Agency.vue
Manage Agencies
Manage Agencies
<<component>>
<<subsystem>>
Database
Landlords
<<component>>
AgencyRepository
Manage Agencies
<<component>>
LandlordModule.vue
Manage Landlords
Middle man
<<component>>
LandlordRepository
Manage Landlords
<<component>>
PropertyRepository
Sale
Manage Properties
View Properties
<<component>>
Registration.vue
Registration
Register
Manade Landlors
Manage Properties
<<component>>
Login.vue
Login
Login
UserProperties
<<component>>
PropertModule.Vue
View Properties

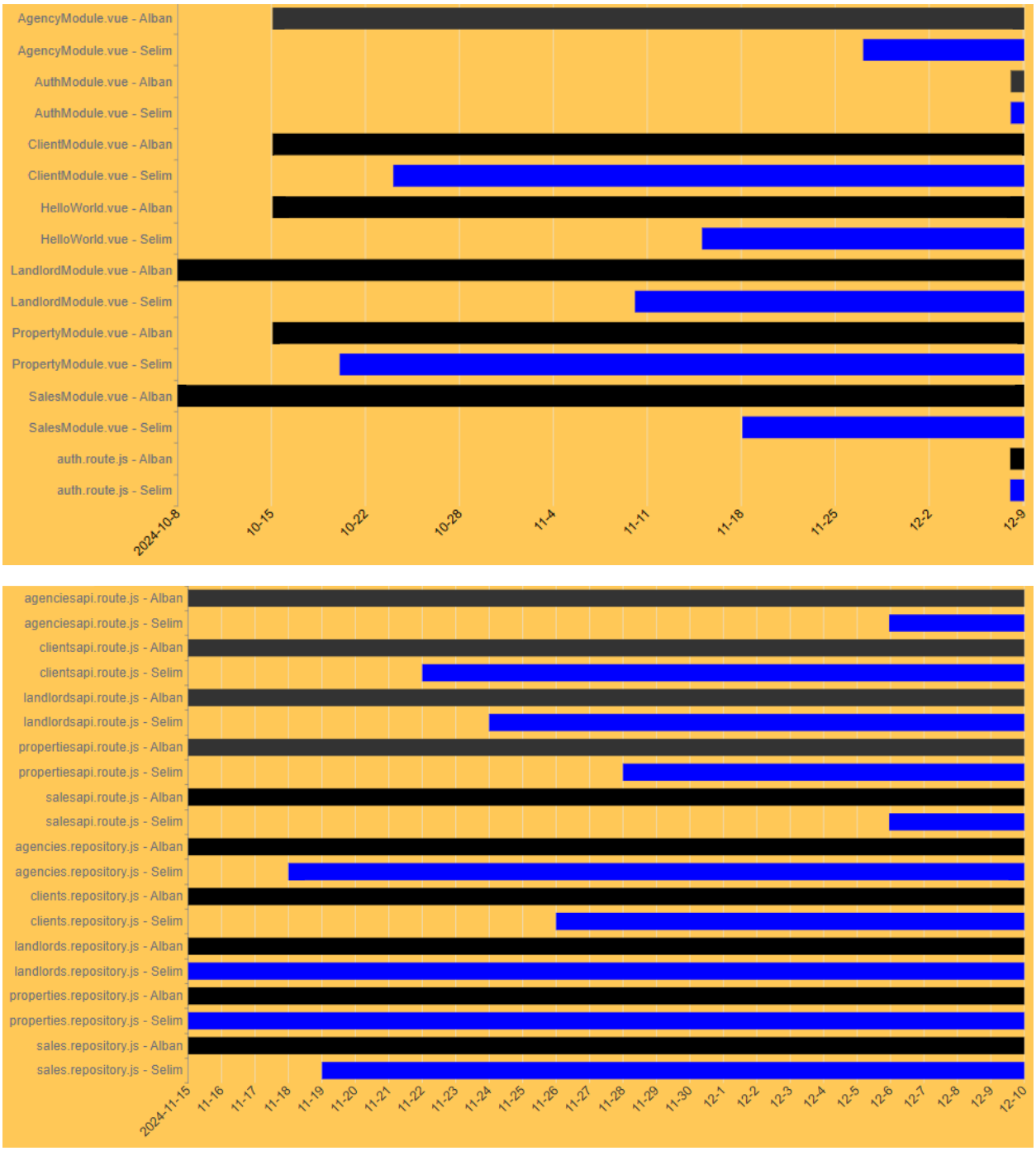
This diagram depicts a component-based system architecture with a focus on modularity and functionality distribution. Key modules, such as SaleModule.vue, LandlordModule.vue, and PropertyModule.vue, represent front-end functionalities like sales management, landlord registration, and property handling. Back-end repositories (SaleRepository, AgencyRepository, etc.) manage data operations, ensuring separation of concerns. The diagram highlights workflows like registration, login, and property management, showcasing interactions between components and repositories. The yellow background enhances visual clarity, and arrows illustrate the flow of data and actions between modules. The structure supports scalability and maintainability, aligning front-end interfaces with back-end logic.

This second diagram illustrates a component-based architecture for a web application. It features modules for managing sales, agencies, landlords, and properties, interconnected via repositories that handle data persistence. Each component, such as SaleModule.vue or PropertyModule.vue, represents specific functionalities like registration, login, or property management. The design emphasizes modularity and separation of concerns, with components communicating through defined interfaces. Green highlights the modules and repositories, ensuring clarity of functionality and interaction. The flow between modules, repositories, and endpoints illustrates the system's operational structure, focusing on user actions (e.g., register, manage) and corresponding backend operations.


entity
Landlord
int landlordID
String firstname
String surname
LandlordRepository landlordRepository
Client
int ClientID
String name
String email
String gender
ClientRepository
Agency
String name
String address
Date dateofcreation
int agency_rating
int nb_employees
AgencyRepository
Property
int PropertyID
int surface_area
int bathrooms
int bedrooms
String type
String address
PropertyRepository
Sale
int PropertyID
int landlordID
int clientID
int saleID
String date
double amount
int commission
SaleRepository
Uses
controller
agenciesapi
const express
const router
const agencyRepo
agencyListAction()
agencyShowAction()
agencyDelAction()
agencyUpdateAction()
clientsapi
const express
const router
const agencyRepo
clientListAction()
clientShowAction()
clientDelAction()
clientUpdateAction()
landlordsapi
const express
const router
const agencyRepo
landlordListAction()
landlordShowAction()
landlordDelAction()
landlordUpdateAction()
propertiesapi
const express
const router
const agencyRepo
landlordListAction()
propertyListAction()
propertyShowAction()
propertyDelAction()
propertyUpdateAction()
salesapi
const express
const router
const agencyRepo
landlordsListAction()
propertiesListAction()
agenciesListAction()
saleListAction()
saleShowAction()
saleDelAction()
saleUpdateAction()
repo
agencies
getBlankAgency()
getAllAgencies()
getAgenciesByName()
getOneAgency()
delOneAgency()
addOneAgency()
editOneAgency()
clients
getBlankClient()
getAllclients()
getClientsByName()
getOneClient()
delOneClient()
addOneClient()
editOneClient()
landlords
getBlankLandlord()
getAllLandlords()
getLandlordsByName()
getOneLandlord()
delOneLandlord()
addOneLandlord()
editOneLandlord()
properties
getBlankProperty()
getAllLandlords()
getAllProperties()
getPropertiesByName()
getOneProperty()
delOneProperty()
addOneProperty()
editOneProperty()
sales
getBlankSale()
getAllLandlords()
getAllProperties()
getAllAgencies()
getAllSales()
getSalesByName()
getOneSale()
delOneSale()
addOneSale()
editOneSale()
Uses



The Complex Diagram above is the class diagram showcasing the classes of the project and how they interact with each other via arrows (HAS, USES, ENROLLS, CONTAINS)



Gaunt Diagram illustrating the modifications of each consequential file over the span of time