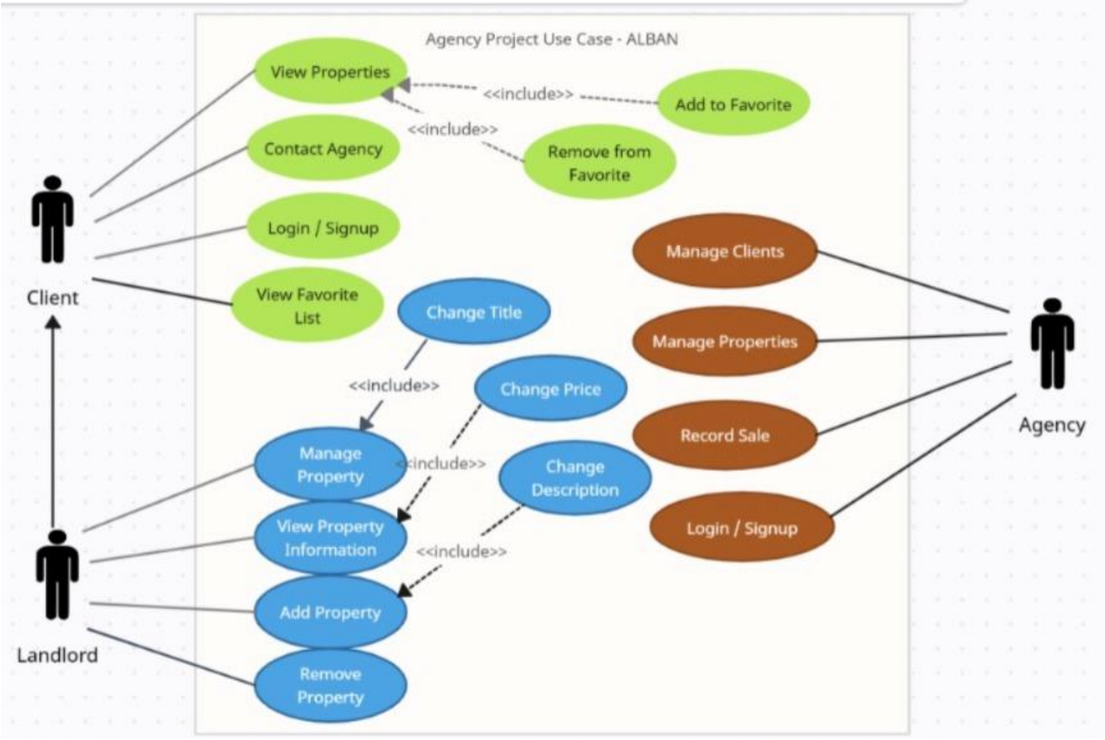
Members: Sulpice Alban, Abouleila Selim

Title: UML DIAGRAMS

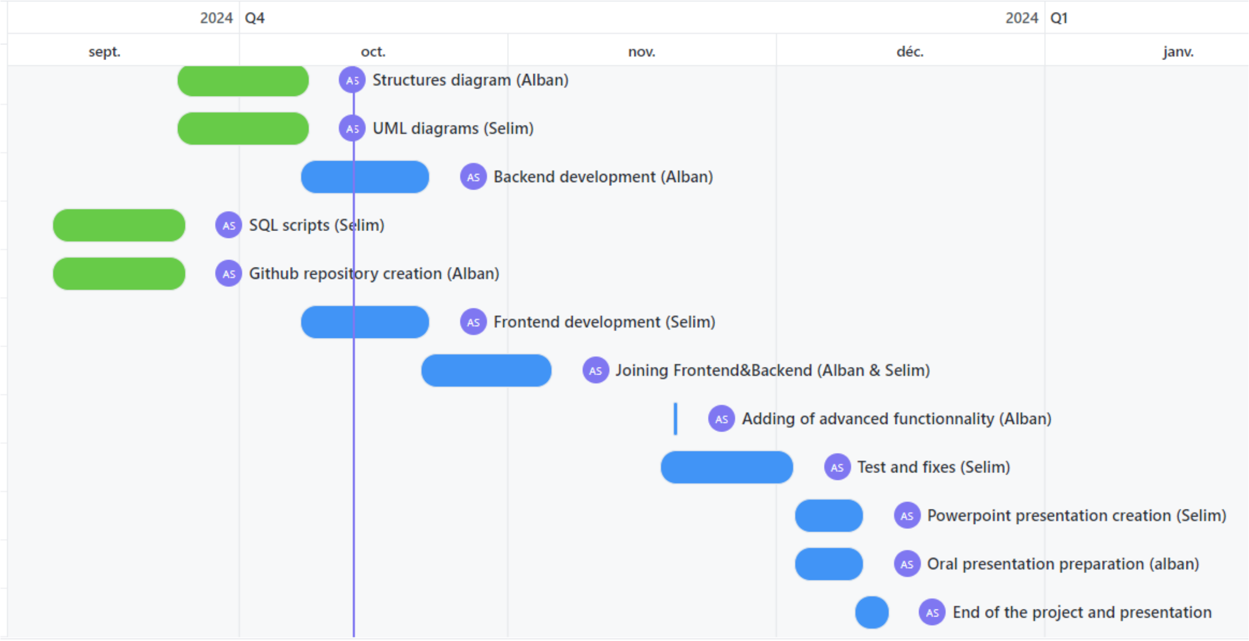


This is a **UML Use Case Diagram** illustrating the interactions between three main actors—**Client**, **Landlord**, and **Agency**—within an agency project.

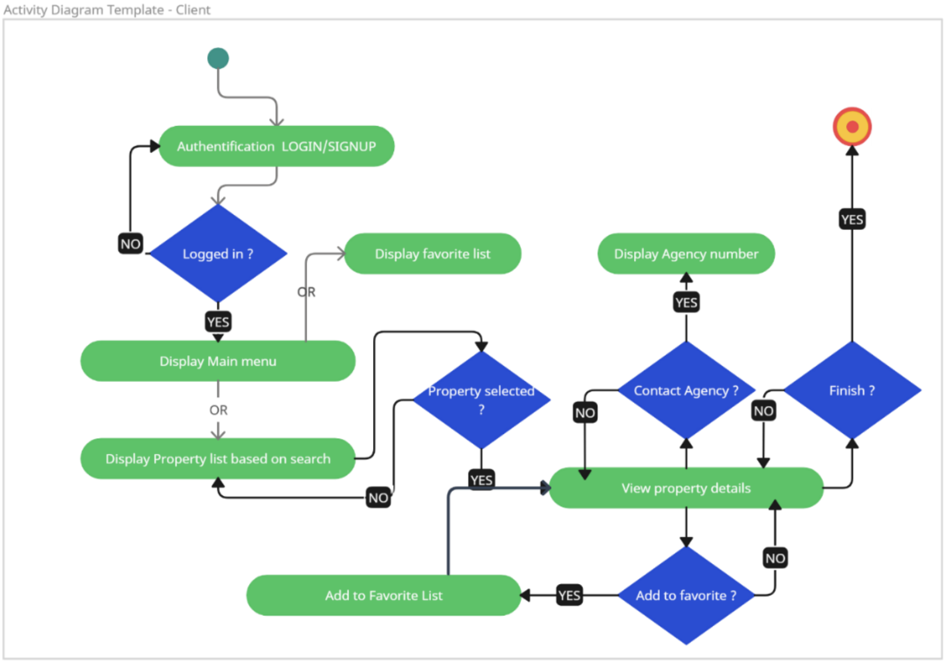
Relations:

- 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.



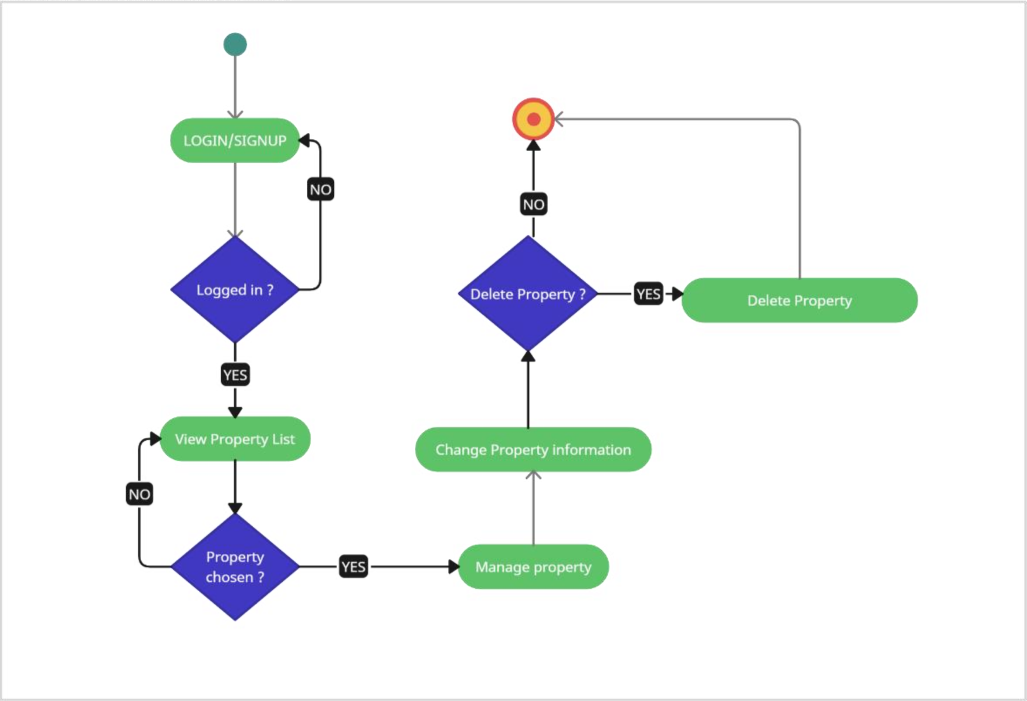
This is an **Activity Diagram** for a **Client's process flow** in an agency system.

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.

Activity Diagram Landlord Manage Property



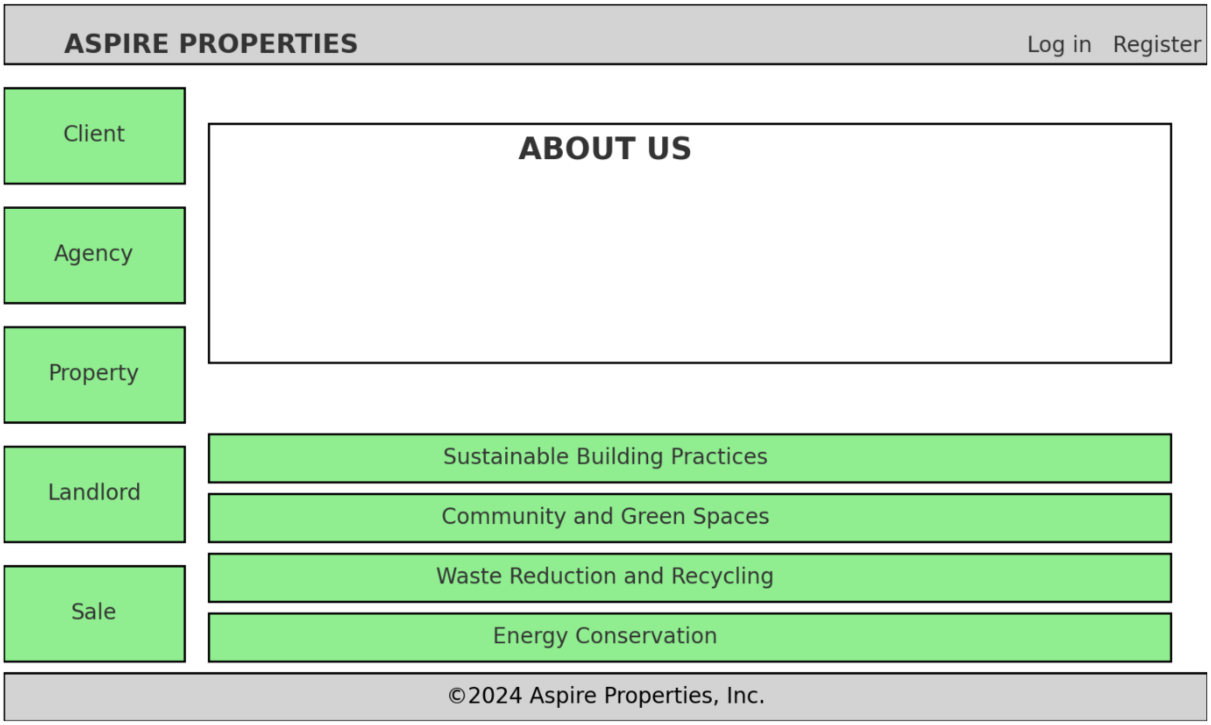
This is anther **Activity Diagram** focused on the **Landlord's property management process** within the system.

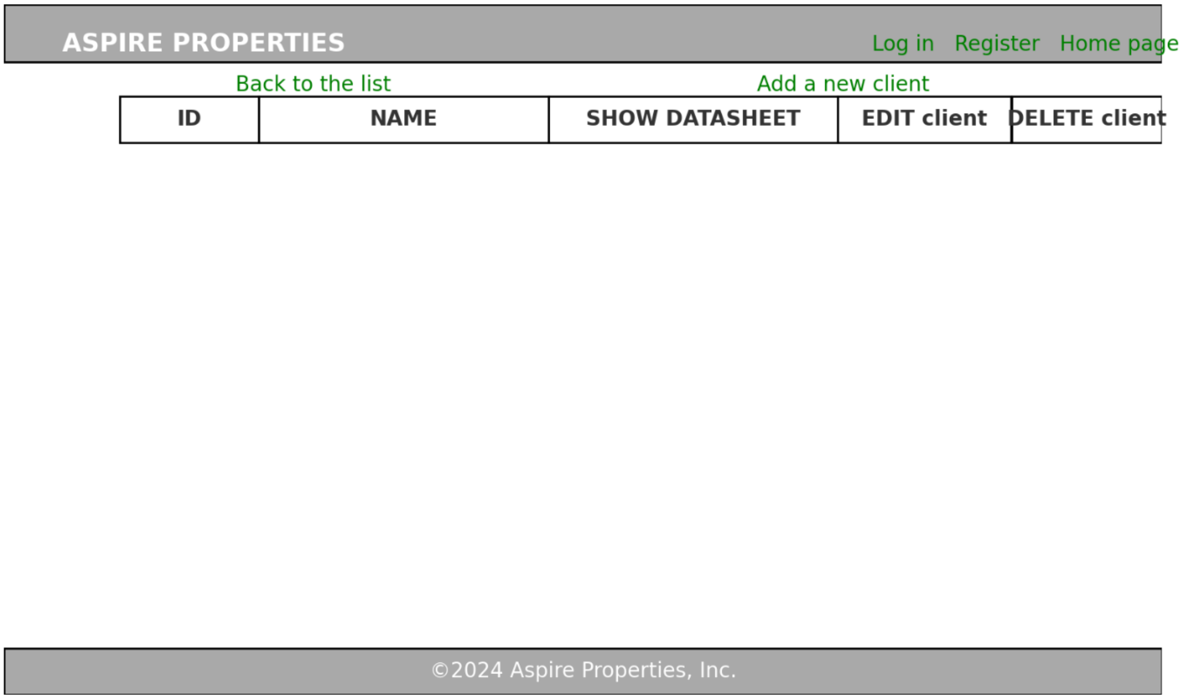
Key Flow:

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.

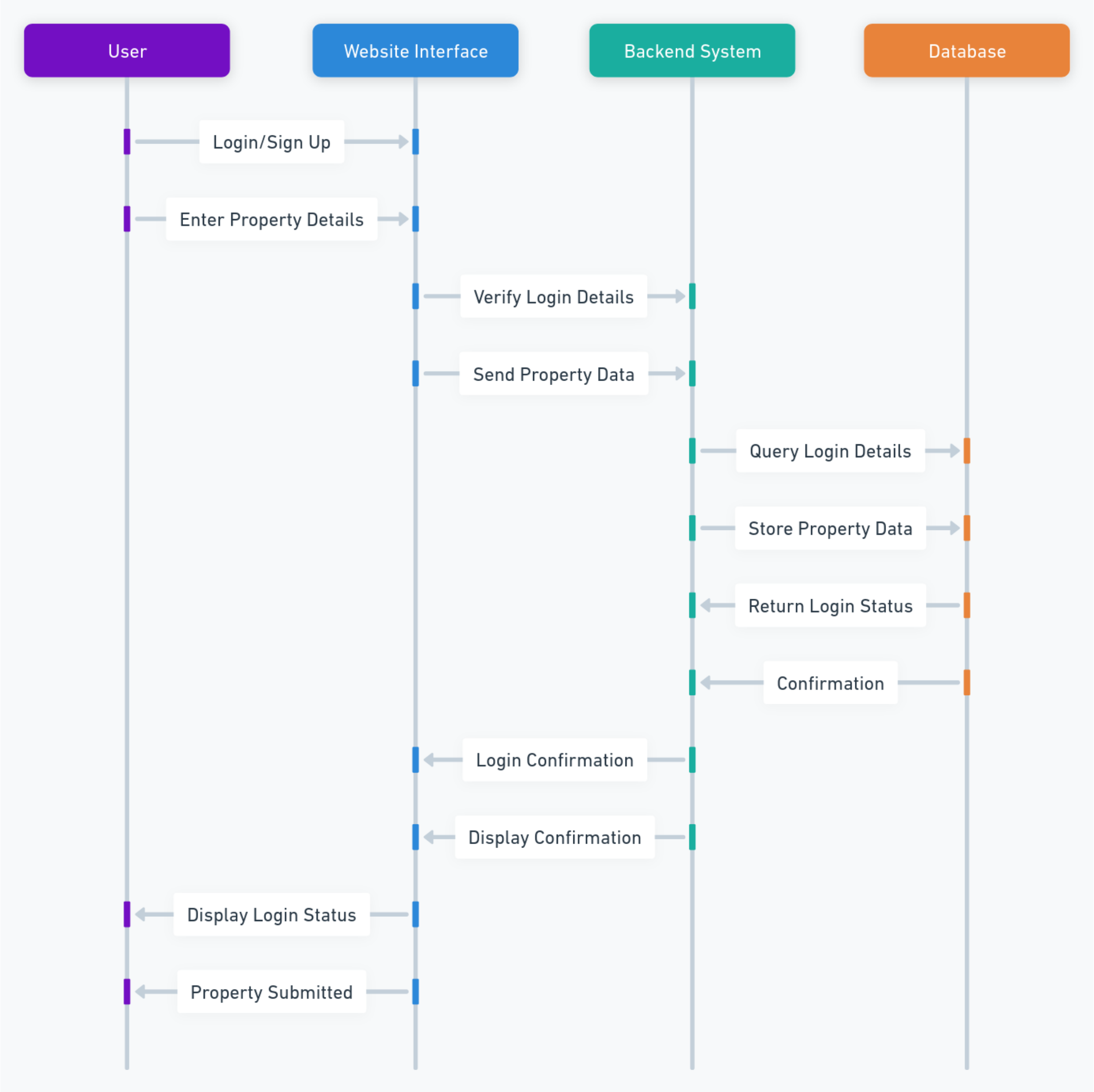

ASPIRE PROPERTIES
Log in
Register
Client
Agency
Property
Landlord
Sale
ABOUT US
Sustainable Building Practices
Community and Green Spaces
Waste Reduction and Recycling
Energy Conservation
©2024 Aspire Properties, Inc.



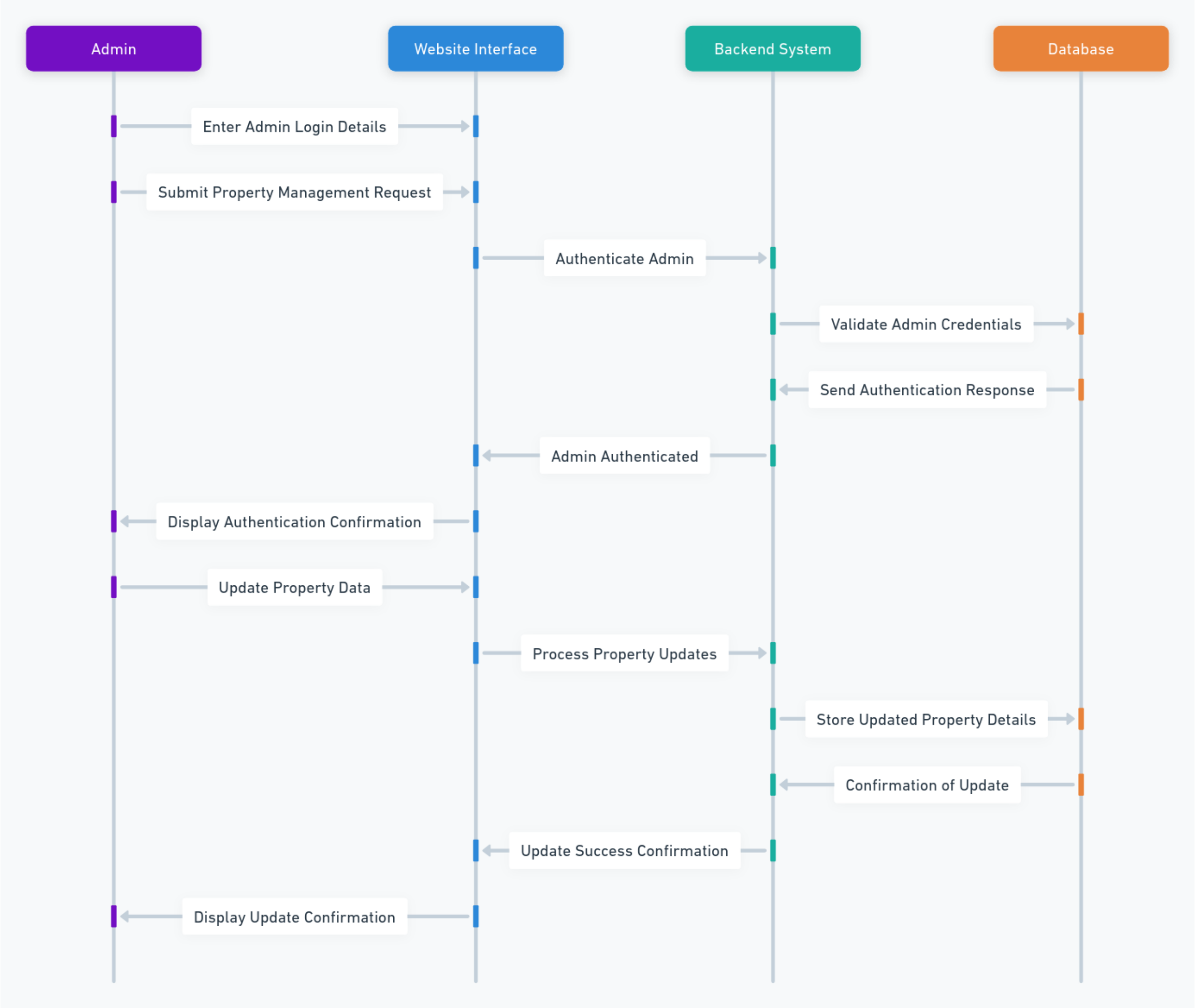
ID	NAME	SHOW DATASHEET	EDIT client	DELETE client

1. **First Wireframe**: A homepage layout with a navigation bar, a vertical menu, an "About Us" section, and expandable sections highlighting environmental practices.

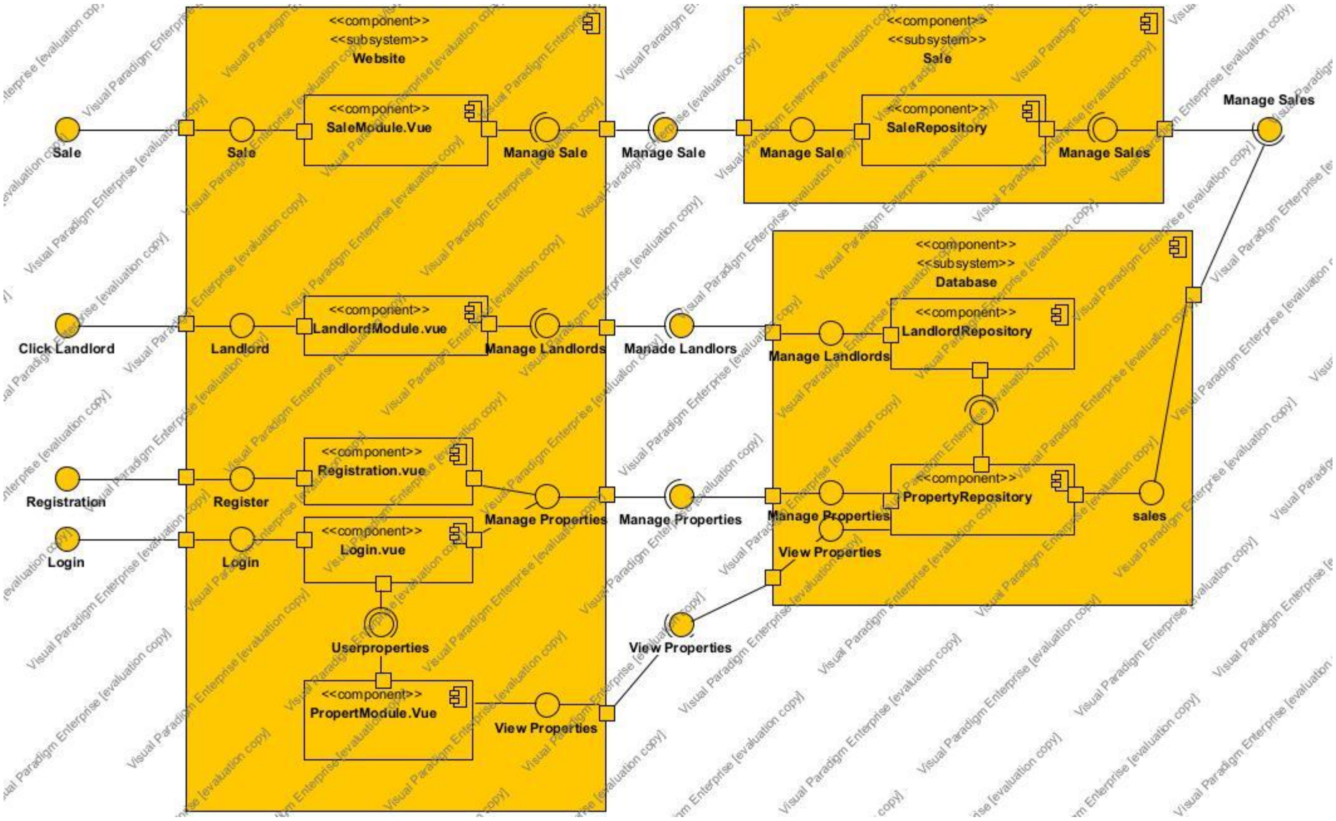
2. **Second Wireframe**: A client management page featuring a top navigation bar, links for actions, a table with client information, and options to view, edit, or delete records.

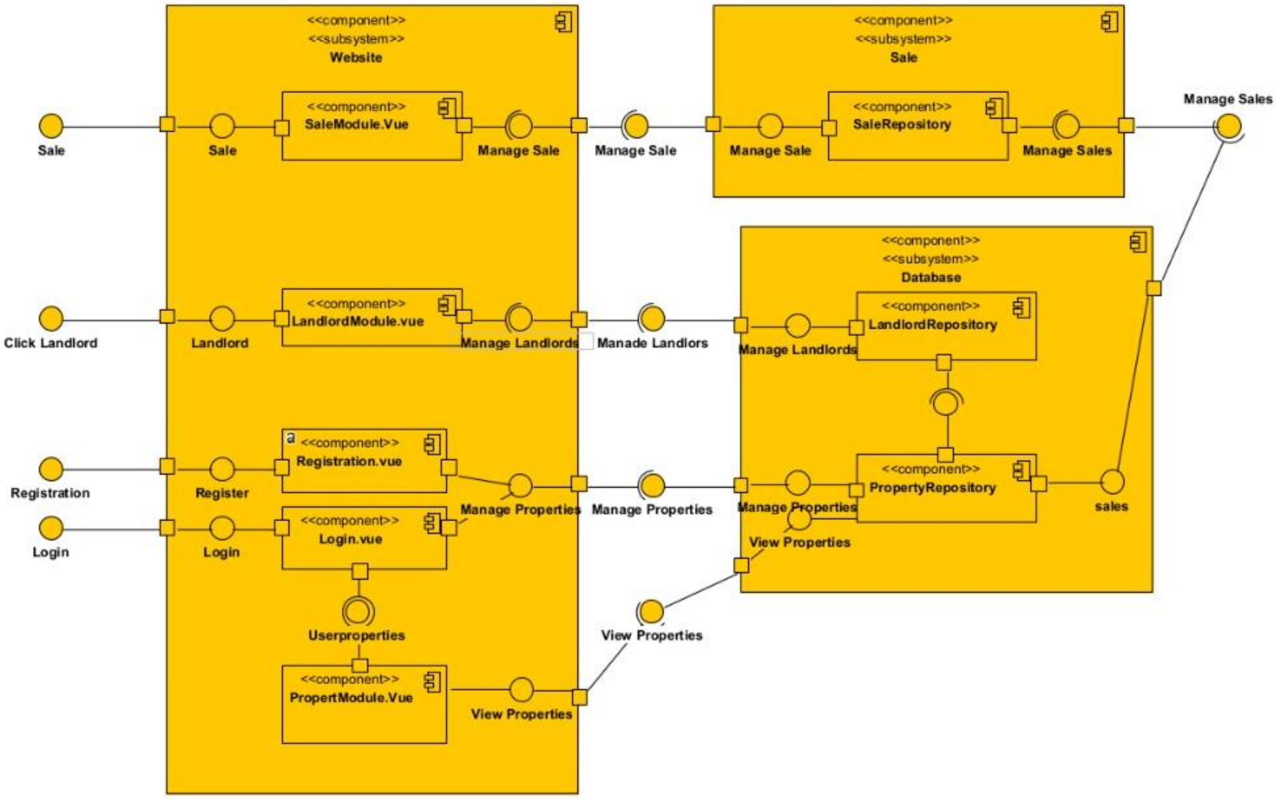


This sequence diagram illustrates the process of a user logging in or signing up, submitting property details, and the data flow through the website interface, backend system, and database. It highlights the interactions and confirmations at each step to ensure successful data submission.

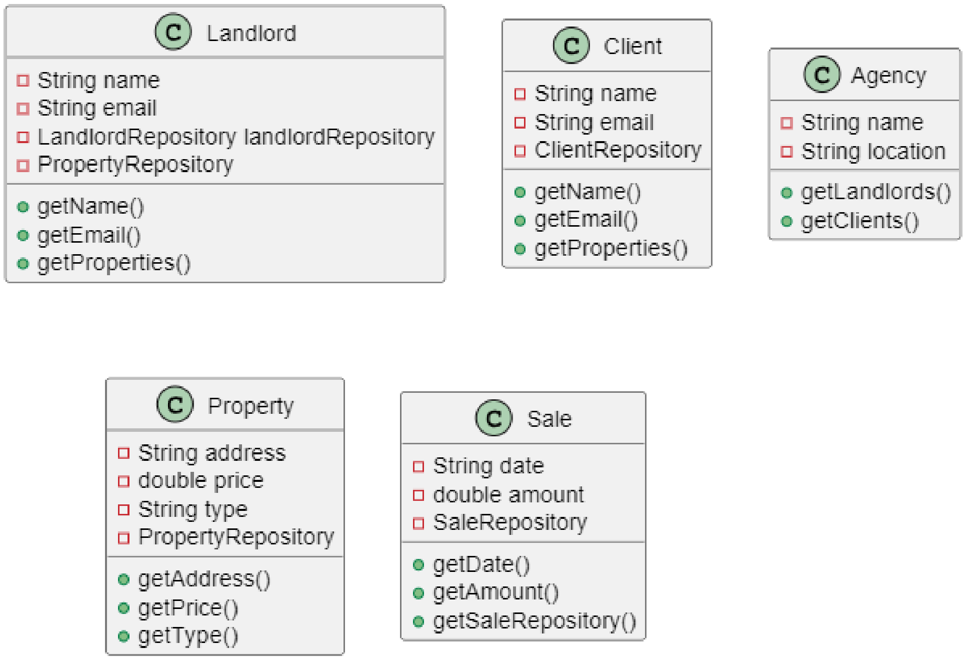

Admin
Website Interface
Backend System
Database
Enter Admin Login Details
Submit Property Management Request
Authenticate Admin
Validate Admin Credentials
Send Authentication Response
Admin Authenticated
Display Authentication Confirmation
Update Property Data
Process Property Updates
Store Updated Property Details
Confirmation of Update
Update Success Confirmation
Display Update Confirmation

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
Sale
Sale
<<component>>
SaleModule.Vue
Manage Sale
Manage Sale
<<component>>
<<subsystem>>
Sale
Manage Sale
<<component>>
SaleRepository
Manage Sales
Manage Sales
<<component>>
<<subsystem>>
Database
Click Landlord
Landlord
<<component>>
LandlordModule.vue
Manage Landlords
Manade Landlors
Manage Landlords
<<component>>
LandlordRepository
Registration
Register
<<component>>
Registration.vue
Manage Properties
Manage Properties
Manage Properties
<<component>>
PropertyRepository
sales
Login
Login
<<component>>
Login.vue
View Properties
Userproperties
View Properties
<<component>>
PropertModule.Vue
View Properties


<<component>>
<<subsystem>>
Website
Sale
Sale
<<component>>
SaleModule.Vue
Manage Sale
Manage Sale
<<component>>
<<subsystem>>
Sale
Manage Sale
<<component>>
SaleRepository
Manage Sales
Manage Sales
<<component>>
<<subsystem>>
Database
Click Landlord
Landlord
<<component>>
LandlordModule.vue
Manage Landlords
Manade Landlors
Manage Landlords
<<component>>
LandlordRepository
Registration
Register
<<component>>
Registration.vue
Manage Properties
Manage Properties
Manage Properties
<<component>>
PropertyRepository
sales
Login
Login
<<component>>
Login.vue
View Properties
Userproperties
View Properties
<<component>>
PropertModule.Vue
View Properties

The Diagram above is the component diagram showcasing a few of the components and how they interact among each other through the views, the database and the business logic



- **Landlord**: Represents a landlord with fields for name, email, a reference to a LandlordRepository, and a list of properties. It includes methods to get the name, email, and properties.
- **Client**: Represents a client with fields for name, email, a reference to a ClientRepository, and a list of properties. It includes methods to get the name, email, and properties.
- **Agency**: Represents an agency with fields for name and location. It includes methods to get lists of landlords and clients.
- **Property**: Represents a property with fields for address, price, type, and a reference to a PropertyRepository. It includes methods to get the address, price, and type.
- **Sale**: Represents a sale with fields for date, amount, and a reference to a Sale Repository. It includes methods to get the date, amount, and the sale repository