

Deliverable 3:

Real Estate Database System

Team Starter Immanuel

Christian Malan

Student ID#: 991720322

SYST17796: Fundamentals of Software Design and Development

CLASS: 1249_83050

Class Schedule: Tuesday 11am – 2pm

Table of Contents

Project Background and Description	4
Introduction	4
Project Expectation	7
Project Scope	9
System Design Overview and Design Considerations	11
System Use Case and User Interaction	11
Class Diagram, Applications and Source Code Design	20
Key Design Concepts	42
Encapsulation	42
Delegation	42
Cohesion	43
Coupling	43
Inheritance	43
Aggregation	44
Composition	44
Flexibility/Maintainability	44
Additional Design Elements	45
Implementation and Testing	55

Implementation and Git Repository	55
Testing and Data Validation.....	55
Project Feasibility and Operation.....	59
Project Summary and Conclusion	91
References	92

Project Background and Description

Introduction

The trend in finding homes and properties is always moving and constant, reflecting the ever-changing dynamics of the real estate market. It is crucial for both business owners and customers to make informed decisions when searching for the “right property” that meets their needs and preferences. The RealEstateApp project addresses these challenges by offering a solution through a computer-based system platform that allows business owners to showcase their properties while giving customers and house owners the opportunity to book appointments with their selected agents. The project not only enhances the efficiency of property management but also provides an interactive platform to bridge the gap between property seekers and providers.

The goal of this project is to develop a computer-based system and establish a database for commercial use in the real estate business. It adheres to the deliverables outlined in the Fundamentals of Software Design and Development course, ensuring a structured and methodical approach throughout its development. The system includes a comprehensive sign-in or sign-up feature, enabling users to either log in or register. While customers and house owners can register, admins and agents have direct access via sign-in credentials. This differentiation supports a streamlined workflow for each role. Admins, also referred to as business owners, have exclusive access to manage property listings through an intuitive interface. Their capabilities include adding, editing, and removing properties, approving pending submissions from house owners, and accessing detailed dashboards to view all users, properties, and appointments, thus centralizing property management functions.

On the customer side, users are guided through intuitive questions to help identify their ideal homes or browse the entire range of available property listings. The system is designed to enhance user experience by providing recommendations if no exact matches are found, based on the customer's preferred criteria. Additionally, customers can easily book appointments with selected agents, further personalizing their journey. House owners, on the other hand, have a dedicated dashboard allowing them to view all properties and book appointments with agents. Their role is enhanced by the ability to submit properties for admin approval, promoting a collaborative environment within the platform. For agents, the system offers tools to view properties, manage appointments, and approve or decline bookings, thus streamlining their workflow and ensuring a professional service experience.

The RealEstateApp, developed using Java, takes inspiration from a database system example in the course textbook and is structured with object-oriented programming principles. Each property is modeled as an object within the Property class, encapsulating essential details such as name, address, number of floors, rooms, property type, proximity to amenities, and schools. These properties are managed by the DataStoreProperty class, which not only stores the listings but also facilitates property searches and other inventory-related functions, reflecting a real-world inventory management system.

The project employs the MVC (Model View Controller) architectural pattern to ensure a clear separation between data management, user interface, and control logic. The Model is responsible for handling core data and algorithms within the database and includes classes for managing user roles such as admin, customer, owner, and agent, each equipped with specific permissions and functions accessible from their respective dashboards. The View delivers a user-

friendly interface, enabling users to easily navigate the platform, whether signing in, registering, or accessing role-specific features. Agents interact with the system through an AgentPage, where they manage appointments and communicate with owners and customers. The Controller processes all user actions, including property submissions and appointment requests, updating the Model and returning results to the View to ensure a cohesive and interactive user experience.

In addition to property management, the project features an advanced booking system where customers and house owners can schedule appointments with agents. Agents can approve or decline these bookings, with approved appointments displayed on the respective users' interfaces for clarity and transparency. The use of a UserRole enum categorizes users, ensuring that all interactions are appropriately handled according to each role, further enhancing security and functionality. Moreover, the system's potential to generate revenue by attracting house owners and property sellers to its services positions it as a valuable tool in the real estate industry.

Overall, the RealEstateApp project not only simulates a realistic and comprehensive real estate management experience but also integrates inventory and booking features within a well-structured, MVC compliant framework. It empowers all stakeholders such as admins, customers, house owners, and agents by providing tailored functionalities and an intuitive platform for collaboration, making it a significant step forward in modernizing property transactions and enhancing user satisfaction.

Project Expectation

The *RealEstateApp* project aims to streamline real estate management by providing a user-friendly platform where admins manage property listings, customers and house owners find and book ideal properties, and agents handle appointments efficiently. It incorporates advanced search, booking features, and role-based functionalities, ensuring an intuitive and collaborative experience while modernizing property transactions.

The *RealEstateApp* project is designed to provide specific functionalities for each user role:

1. User Sign-In and Sign-Up:

- Customers and house owners can sing-in or register via sign-up.
- Admins and agents can sign in with pre-registered credentials.

2. Admin:

- Manage property listings like add, edit, or remove.
- Approve or reject pending property submissions from house owners.
- View all users, properties, and appointments.

3. Customer:

- Search for properties using filters or browse all listings.
- Receive recommendations if no exact matches are found.
- Book appointments with selected agents.

4. Property Owner:

- Submit properties for admin approval.
- View all properties.
- Book appointments with agents.

5. Agent:

- View all properties and appointments.
- Approve or decline booking requests.
- Manage appointments with customers and house owners.

Each user has a tailored experience, ensuring efficiency and collaboration across the platform.

Project Scope

Member: Christian Malan

As the sole member of the team, I managed all aspects of the *RealEstateApp* project, taking on multiple roles to ensure its successful completion. This included conceptualizing the system and developing key documentation, such as the use case diagram, use case narrative, and UML class diagram. I also performed the following tasks across various roles:

1. Lead Developer and Backend Developer:

- Designed and implemented the program flow to align with project goals.
- Developed algorithms and methods to manage property listings, including adding, editing, and removing properties.
- Wrote validation logic for data input to ensure system accuracy and reliability.

2. Database Manager:

- Designed and implemented the database schema to support the system's functionality.
- Created methods to manage property data effectively within the database.

3. Frontend Developer:

- Developed and coded the user interface for a seamless user experience.
- Implemented filtering functionality to display properties based on customer preferences.

4. QA Tester:

- Tested system functionality to confirm alignment with project goals.
- Verified user authentication for different roles, such as admin, customer, and owner.
- Ensured the system was responsive, user-friendly, and robust across various scenarios.
- Provided feedback during testing to enhance the overall quality of the system.

Technical Scope:

The system features a multi-user interface tailored for four types of users: admins (business owners), agents, customers, and house owners. Admins can manage property listings by adding, editing, or removing properties, approve or reject submissions from house owners, and view all users, properties, and appointments. Agents can view property listings, manage appointments, and approve or decline booking requests. Customers can search for properties using filters such as location, house type, and the number of bedrooms, with guided prompts to refine their search for an ideal property, as well as book appointments with agents. House owners can submit properties for admin approval, view available properties, and schedule appointments with agents. The project's success is defined by enabling seamless property management for admins and house owners while providing a personalized property search and booking experience for customers and agents.

The use case diagram on figure 1 for the real estate application outlines interactions among five main actors: Admin, Agent, Customer, Owner, and the System. Admin manages properties and users by adding, editing, removing, or approving properties and viewing user details. Agent is responsible for approving or declining bookings and viewing their scheduled appointments. Customer searches for properties, views property details, and schedules appointments with agents for property viewings. Owner submits properties for approval, monitors the status of these properties, and can book appointments with agents if necessary. System verifies user credentials during login, and if a user is not registered, it allows them to register with a role such as Customer and Owner. Once registered, System enforces role-based access to ensure each user can only interact with features relevant to their role, such as AdminPage, AgentPage, CustomerPage, or OwnerPage. This diagram visually captures each actor's goals and interactions, supporting a clear understanding of user flows and access within the application.

The use case diagram and user interactions were described as such in the use case narratives with main and alternate path below.

1. Prompt User to Select Sign in or sign up

1.1 Sign In (Main Path)

1.1.1. Prompt User to enter username and password.

1.1.2. Enter username and password.

1.1.3. Verify credentials.

Successful Login (Main Path):

1.1.3.1. Grant access to the account.

1.1.3.2. Display user dashboard with options based on role (Admin, Agent, Customer, and Owner).

Alternate Path (Incorrect Credentials):

1.1.3.3. System rejects login attempt.

1.1.3.4. Re-prompt User to enter valid credentials until they are correct.

1.2 Sign Up (Alternate Path)

1.2.1. Select sign up option.

1.2.2. System prompts User to enter registration details (e.g., username, password, email).

1.2.3. User enters registration details and submits.

1.2.4. The system confirms registration with a success message.

1.2.5. Proceed to sign in by following steps in the Main Path (1.1).

2. User Dashboard

Once logged in, the User navigates to their specific dashboard based on role:

- 2.1 Admin Dashboard
- 2.2 Agent Dashboard

- 2.3 Customer Dashboard
- 2.4 Owner Dashboard

2.1 Admin Dashboard

2.1.1. Display Admin dashboard with management options: property management, user management, and appointment approvals.

2.1.1.1. Add a New Property

2.1.1.1.1. Select the option to add a new property.

2.1.1.1.2. Display property submission form.

2.1.1.1.3. Enter property details and submit.

2.1.1.1.4. The system confirms that the property has been successfully added.

Alternate Path (Incomplete Form):

2.1.1.1.5. System displays "Please complete all fields."

2.1.1.1.6. Return to submission form to complete.

2.1.1.2. Edit Property

2.1.1.2.1. Select options to edit an existing property.

2.1.1.2.2. Display list of properties.

2.1.1.2.3. Select property to edit.

2.1.1.2.4. Display property details form for editing.

2.1.1.2.5. Enter changes and submit.

2.1.1.2.6. System confirms "Edit successful."

Alternate Path (Cancel Edit):

2.1.1.2.7. Select cancel to abandon changes.

2.1.1.2.8. Return to Admin dashboard.

2.1.1.3. Delete Property

2.1.1.3.1. Select an option to delete a property.

2.1.1.3.2. Display list of properties.

2.1.1.3.3. Select property and confirm deletion.

2.1.1.3.4. The system removes the property.

Alternate Path (Cancel Deletion):

2.1.1.3.5. Select cancel.

2.1.1.3.6. Return to Admin dashboard.

2.1.1.4. View All Properties

2.1.1.4.1. Select options to view all properties.

2.1.1.4.2. The system displays a list of all properties.

2.1.1.4.3. Return to Admin dashboard.

2.1.1.5. Approve or Reject Pending Properties

2.1.1.5.1. View list of pending properties.

2.1.1.5.2. Select property to review.

Main Path (Approve Property):

2.1.1.5.3. Approve property.

2.1.1.5.4. Update status to "Approved" and notify Owner.

Alternate Path (Reject Property):

2.1.1.5.5. Reject property.

2.1.1.5.6. Update status to "Rejected" and notify Owner.

2.1.1.6. Manage Users and Appointments

2.1.1.6.1. View all user accounts.

2.1.1.6.2. Select user to manage permissions or view details.

2.1.1.6.3. View appointments and manage scheduling.

2.2 Agent Dashboard

2.2.1. Display Agent dashboard with options to view appointments, manage bookings, and access assigned properties.

2.2.1.1. View Appointments

2.2.1.1.1. Select options to view appointments.

2.2.1.1.2. The system displays a list of upcoming appointments.

2.2.1.1.3. Return to Agent dashboard.

2.2.1.2. Approve or Reject Bookings

2.2.1.2.1. View pending bookings.

2.2.1.2.2. The agent selects bookings to review.

Main Path (Approve Booking):

2.2.1.2.3. Approved booking.

2.2.1.2.4. Update status to "Approved" and notify Customer.

Alternate Path (Reject Booking):

2.2.1.2.5. Reject booking.

2.2.1.2.6. Update status to "Rejected" and notify Customer.

2.2.1.3. View Assigned Properties

2.2.1.3.1. Select option to view properties assigned to the Agent.

2.2.1.3.2. System displays assigned properties.

2.2.1.3.3. Return to Agent dashboard.

2.3 Customer Dashboard

2.3.1. Display Customer dashboard with options to search for properties, view property details, and book viewings.

2.3.1.1. Search for Properties

2.3.1.1.1. Enter search criteria.

2.3.1.1.2. The system displays properties matching criteria.

2.3.1.1.3. Return to Customer dashboard.

Alternate Path (No Properties Found):

2.3.1.1.4. The system displays "No properties found."

2.3.1.1.5. Return to search form to modify criteria.

2.3.1.2. View Property Details

2.3.1.2.1. Select property from search results.

2.3.1.2.2. The system displays property details.

2.3.1.2.3. Return to Customer dashboard.

2.3.1.3. Book a Property Viewing

2.3.1.3.1. Select property to book a viewing.

2.3.1.3.2. Fill out the booking form with date and time.

2.3.1.3.3. System confirms booking and notifies Customer.

Alternate Path (Booking Error):

2.3.1.3.4. The system displays an error message "Unable to complete booking."

2.3.1.3.5. Return to booking form to retry.

2.4 Owner Dashboard

2.4.1. Display Owner dashboard with options to submit properties, view submitted properties, and book appointments with Agents.

2.4.1.1. Submitting a New Property

2.4.1.1.1. Select an option to submit a property.

2.4.1.1.2. Fill out property submission form.

2.4.1.1.3. System saves property as "Pending Approval."

Alternate Path (Incomplete Form):

2.4.1.1.4. System displays "Please complete all fields."

2.4.1.1.5. Return to submission form to complete.

2.4.1.2. View Submitted Properties

2.4.1.2.1. View list of submitted properties with status.

2.4.1.2.2. Select property to view details.

2.4.1.2.3. Return to Owner dashboard.

2.4.1.3. Book Appointment with Agent

2.4.1.3.1. Select the option to book an appointment with an Agent.

2.4.1.3.2. Fill out appointment details and submit.

2.4.1.3.3. System confirms booking and notifies Owner.

Alternate Path (Booking Error):

2.4.1.3.4. The system displays an error message "Unable to complete booking."

2.4.1.3.5. Return to the appointment form to retry.

Class Diagram, Applications and Source Code Design

The UML class diagram for the real estate application provides a structured view of the system's classes, their attributes, methods, and relationships. It captures the essential components involved in managing properties, bookings, and user roles, including Admin, Agent, Customer, and Owner. Key classes such as Property, Booking, User, and DataStore are depicted, along with specialized classes like AddProperty, ApproveBooking, and Search, which represent actions specific to different user roles. The diagram illustrates associations, dependencies, inheritance, and aggregations between classes, showing how data flows within the application and how each class interacts to fulfill tasks like property management, appointment scheduling, and user verification. Overall, this UML class diagram provides a comprehensive blueprint for understanding the system's architecture, the roles of individual components, and the interconnections that enable core functionalities within the application.

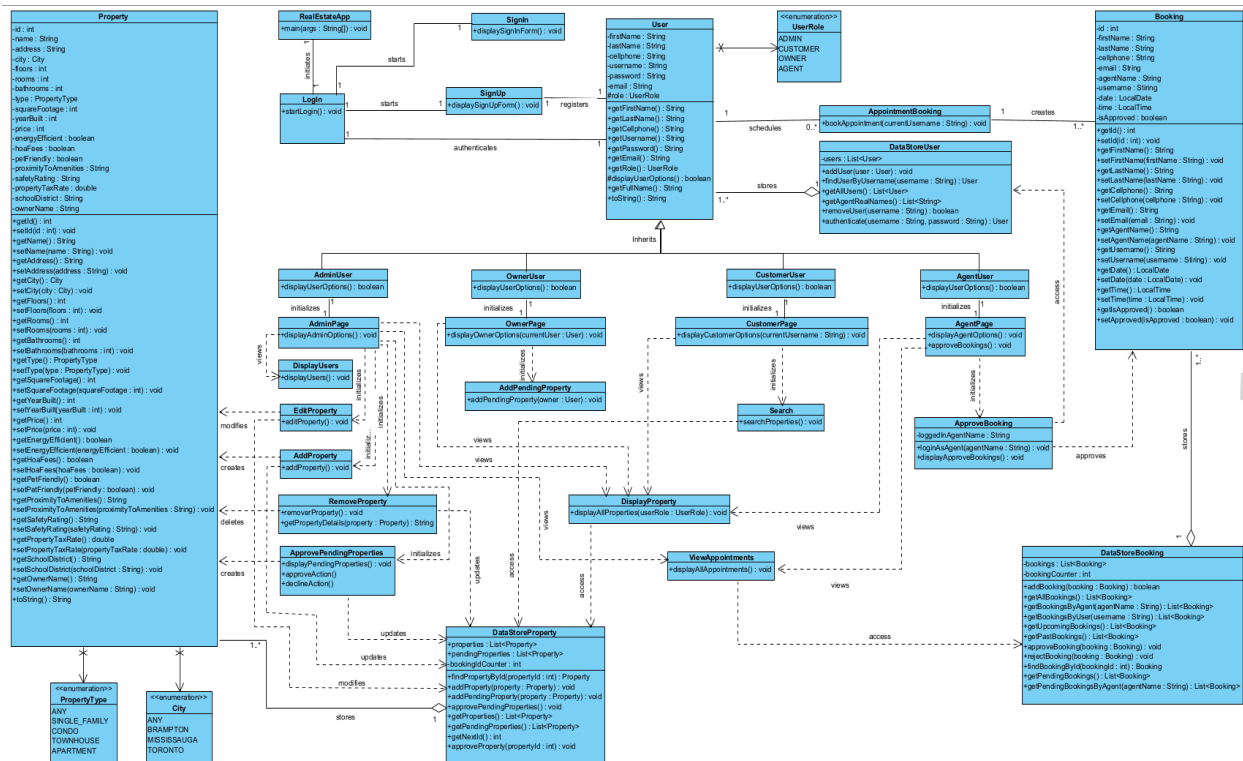


Figure 2. Real Estate System UML Class Diagram

The class diagram in Figure 2 represents the design of the real estate application, showing the relationships, associations, and multiplicities between core classes such as User, Property, Booking, DataStore, and action classes like AddProperty and ApproveBooking. The diagram includes associations and multiplicities to indicate how many instances of one class relate to instances of another, such as a single User being able to have multiple Booking instances (1 to many) and DataStoreUser aggregating multiple User objects (1 to many). Methods, modifiers, and return types are defined within each class, supporting encapsulation and allowing other classes to interact with specific functionalities without direct access to the underlying data.

Table 1. RealEstateApp Class Attributes and Methods Definitions

RealEstateApp Class Attributes and Methods Definitions		
Method	main	Initialize the application

Table 2. Login Class Attributes and Methods Definitions

Login Class Attributes and Methods Definitions		
Method	startLogin	Initiates the login process for users. It verifies user credentials and determines the next steps based on the user's role. Whether they registered or not.

Table 3. SignIn Class Attributes and Methods Definitions

SignIn Class Attributes and Methods Definitions		
Method	displaySignInForm	Shows the login form for user access.

Table 4. SignUp Class Attributes and Methods Definitions

SignUp Class Attributes and Methods Definitions		
Method	displaySignUpForm	Shows the registration form for new users.

Table 5. User Class Attributes and Methods Definitions

User Class Attributes and Methods Definitions	
Attributes	
firstName	The user's first name.

lastName	The user's last name.
cellphone	The user's cellphone number.
username	The user's chosen username.
password	The user's account password.
email	The user's email address.
role	The role of the user (e.g., admin, customer).
Methods	
getFirstName	Returns the user's first name.
getLastName	Returns the user's last name.
getCellphone	Returns the user's cellphone number.
getUsername	Returns the user's username.
getPassword	Returns the user's password.
getEmail	Returns the user's email address.
getRole	Returns the user's role.
displayUserOptions	Displays available options based on the user's role.
getFullName	Returns the user's full name (first and last name).
toString	Returns a string representation of the user object.

Table 6. UserRole Enum Definitions

UserRole Enum Definitions	
ADMIN	Represents a system administrator or business owner who manages properties and users.

CUSTOMER	Represents a customer searching for properties and booking appointments.
OWNER	Represents a house owner submitting properties for approval and booking appointments.
AGENT	Represents an agent managing appointments and interacting with customers and owners.

Table 7. AppointmentBooking Class Attributes and Methods Definitions

AppointmentBooking Class Attributes and Methods Definitions	
bookAppointment(currentUsername)	Books an appointment for the given username.

Table 8. AppointmentBooking Class Attributes and Methods Definitions

AppointmentBooking Class Attributes and Methods Definitions	
Attributes	
users	A list containing all users in the system.
Methods	
addUser(user)	Add a new user to the list.
findUserByUsername(username)	Finds and returns a user by their username.
getAllUsers()	Returns a list of all users in the system.
getAgentRealNames()	Retrieves a list of real names of all agent users.

removeUser(username)	Removes a user from the list based on their username and returns a boolean indicating success.
authenticate(username, password)	Authenticates a user by checking their username and password combination.

Table 9. Booking Class Attributes and Methods Definitions

Booking Class Attributes and Methods Definitions	
Attributes	
id	A unique identifier for each booking.
firstName	The first name of the customer or owner.
lastName	The last name of the customer or owner.
cellphone	The contact number of the customer or owner.
email	The email address of the customer or owner.
agentName	The name of the assigned agent.
username	The username of the customer or owner.
date	The date of the booking (type: LocalDate).
time	The time of the booking (type: LocalTime).
isApproved	Indicates if the booking is approved (true or false).

Methods	
getId	Returns the booking ID.
setId	Sets the booking ID.
getFirstName	Returns the first name.
setFirstName	Sets the first name.
getLastName	Returns the last name.
setLastName	Sets the last name.
getCellphone	Returns the cellphone number.
setCellphone	Sets the cellphone number.
getEmail	Returns the email address.
setEmail	Sets the email address.
getAgentName	Returns the assigned agent's name.
setAgentName	Sets the assigned agent's name.
getUsername	Returns the username.
setUsername	Sets the username.
getDate	Return the date of the booking.
setDate	Set the date of the booking.

getTime	Return the time of the booking.
setTime	Sets the time of the booking.
getIsApproved	Returns whether the booking is approved.
setIsApproved	Sets the approval status of the booking.

Table 10. DataStoreBooking Class Attributes and Methods Definitions

DataStoreBooking Class Attributes and Methods Definitions	
Attributes	
bookings	A list containing all the booking objects.
bookingCounter	A counter to track the total number of bookings.
Methods	
addBooking(booking : Booking)	Adds a new booking to the list and returns a boolean indicating success.
getAllBookings()	Retrieves a list of all bookings.
getBookingsByAgent(agentName : String)	Retrieves bookings assigned to a specific agent.
getBookingsByUser(username : String)	Retrieves bookings made by a specific user.
getUpcomingBookings()	Retrieves a list of bookings scheduled for future dates.

getPastBookings()	Retrieves a list of bookings that have already occurred
approveBooking(booking : Booking)	Approves a specified booking.
rejectBooking(booking : Booking)	Rejects a specified booking.
findBookingById(bookingId : int)	Finds a booking by its unique ID and returns the booking object.
getPendingBookings()	Retrieves a list of all pending bookings.
getPendingBookingsByAgent(agentName : String)	Retrieves pending bookings for a specific agent.

Table 11. Property Class Attributes and Methods Definitions

Property Class Attributes and Methods Definitions	
Attributes	
id	A unique identifier for each property.
name	The name or title of the property.
address	The full address of the property.
city	The city where the property is located.
floors	The number of floors in the property.
rooms	The total number of rooms on the property.

bathrooms	The total number of bathrooms in the property.
type	The type of property (e.g., house, apartment).
squareFootage	The total square footage of the property.
yearBuilt	The year the property was constructed.
price	The price of the property.
energyEfficient	Indicates whether the property is energy efficient.
hoaFees	Indicates if homeowner association fees apply.
petFriendly	Indicates if the property is pet friendly.
proximityToAmenities	A description of nearby amenities.
safetyRating	A safety rating for the property location.
propertyTaxRate	The tax rate applied to the property.
schoolDistrict	The school district where the property located.
ownerName	The name of the property owner.
Methods	
getId	Returns the ID of the property.
setId	Sets the ID of the property.

getName	Returns the name of the property.
setName	Sets the name of the property.
getAddress	Returns the address of the property.
setAddress	Sets the address of the property.
getCity	Returns the city of the property.
setCity	Sets the city of the property.
getFloors	Returns the number of floors in the property.
setFloors	Sets the number of floors on the property.
getRooms	Returns the number of rooms on the property.
setRooms	Sets the number of rooms in the property.
getBathrooms	Returns the number of bathrooms on the property.
setBathrooms	Sets the number of bathrooms in the property.
getType	Returns the type of property.
setType	Sets the type of property.
getSquareFootage	Returns the square footage of the property.
setSquareFootage	Sets the square footage of the property.
getYearBuilt	Returns the year the property was built.

setYearBuilt	Sets the year the property was built.
getPrice	Returns the price of the property.
setPrice	Sets the price of the property.
getEnergyEfficient	Returns whether the property is energy efficient.
setEnergyEfficient	Sets the energy efficiency status of the property.
getHoaFees	Returns whether the property has HOA fees.
setHoaFees	Sets the HOA fees status of the property.
getPetFriendly	Returns whether the property is pet friendly.
setPetFriendly	Sets the pet-friendly status of the property.
getProximityToAmenities	Returns the description of the property's proximity to amenities.
setProximityToAmenities	Sets the description of the property's proximity to amenities.
getSafetyRating	Returns the safety rating of the property location.
setSafetyRating	Sets the safety rating of the property location.
getPropertyTaxRate	Returns the property tax rate.
setPropertyTaxRate	Sets the property tax rate.
getSchoolDistrict	Returns the school district where the property is located.
setSchoolDistrict	Sets the school district where the property is located.

getOwnerName	Returns the name of the property owner.
setOwnerName	Sets the name of the property owner.
toString	Returns a string representation of the property object.

Table 12. PropertyType Enum Definitions

PropertyType Enum Definitions	
ANY	Represents any type of property.
SINGLE_FAMILY	Represents single-family homes.
CONDO	Represents condominiums.
TOWNHOUSE	Represents townhouses.
APARTMENT	Represents apartments.

Table 13. City Enum Definitions

City Enum Definitions	
ANY	Represents any city.
BRAMPTON	Represents properties located in Brampton.
MISSISSAUGA	Represent properties located in Mississauga.
TORONTO	Represents properties located in Toronto.

Table 14. DataStoreProperty Class Attributes and Methods Definitions

DataStoreProperty Class Attributes and Methods Definitions	
Attributes	
properties	A list that stores all approved properties.
pendingProperties	A list that stores all properties pending approval.
bookingIdCounter	A counter to track the unique IDs for bookings.
Methods	
findPropertyById(propertyId)	Finds and returns a property by its unique ID.
addProperty(property)	Adds a property to the list of approved properties.
addPendingProperty(property)	Adds a property to the list of pending properties.
approvePendingProperties()	Approves all pending properties and moves them to the approved list.
getProperties()	Returns a list of all approved properties.
getPendingProperties()	Returns a list of all pending properties.
getNextId()	Returns the next unique property ID.
approveProperty(propertyId)	Approves a specific property by its ID and moves it to the approved list.

Table 15. AdminUser Class Attributes and Methods Definitions

AdminUser Class Attributes and Methods Definitions		
Method	displayUserOptions	Displays the available options for an admin user.

Table 16. AdminPage Class Attributes and Methods Definitions

AdminPage Class Attributes and Methods Definitions		
Method	displayAdminOptions	Displays the main admin page options.

Table 17. DisplayUsers Class Attributes and Methods Definitions

DisplayUsers Class Attributes and Methods Definitions		
Method	displayUsers	Displays a list of all users in the system.

Table 18. DisplayUsers Class Attributes and Methods Definitions

EditProperty Class Attributes and Methods Definitions		
Method	editProperty	Allows admins to edit details of an existing property.

Table 19. AddProperty Class Attributes and Methods Definitions

AddProperty Class Attributes and Methods Definitions		
Method	addProperty	Enables admins to add a new property to the system.

Table 20. RemoveProperty Class Attributes and Methods Definitions

RemoveProperty Class Attributes and Methods Definitions		
Method	removeProperty	Removes property from the system.
	getPropertyDetails	Retrieves the details of a specific property.

Table 21. ApprovePendingProperties Class Attributes and Methods Definitions

ApprovePendingProperties Class Attributes and Methods Definitions		
Method	displayPendingProperties	Displays all properties awaiting approval.
	approveAction	Approves pending property.
	declineAction	Declines pending property.

Table 22. OwnerUser Class Attributes and Methods Definitions

OwnerUser Class Attributes and Methods Definitions		
Method	displayUserOptions	Displays the available options for an owner user.

Table 23. OwnerPage Class Attributes and Methods Definitions

OwnerPage Class Attributes and Methods Definitions		
Method	displayOwnerOptions(currentUser)	Displays the owner's dashboard options based on the current user.

Table 24. AddPendingProperty Class Attributes and Methods Definitions

AddPendingProperty Class Attributes and Methods Definitions		
Method	addPendingProperty(owner)	Allows the owner to add a property for admin approval.

Table 25. CustomerUser Class Attributes and Methods Definitions

CustomerUser Class Attributes and Methods Definitions		
Method	displayUserOptions	Displays the available options for a customer user.

Table 26. CustomerPage Class Attributes and Methods Definitions

CustomerPage Class Attributes and Methods Definitions		
Method	displayCustomerOptions(currentUsername)	Displays the customer's dashboard based on the current username.

Table 27. Search Class Attributes and Methods Definitions

Search Class Attributes and Methods Definitions		
Method	searchProperties	Allow customers to search for properties based on filters and preferences.

Table 28. AgentUser Class Attributes and Methods Definitions

AgentUser Class Attributes and Methods Definitions		
Methods	displayUserOptions	Displays the available options for an agent user.

Table 29. AgentPage Class Attributes and Methods Definitions

AgentPage Class Attributes and Methods Definitions		
Methods	displayAgentOptions	Displays the agent's dashboard options.
	approveBookings	Allows the agent to review and approve pending bookings.

Table 30. DisplayProperty Class Attributes and Methods Definitions

DisplayProperty Class Attributes and Methods Definitions		
Methods	displayAllProperties(userRole : UserRole)	Displays all properties based on the user's role (e.g., Admin, Agent, Customer).

Table 31. ViewAppointmentsClass Attributes and Methods Definitions

ViewAppointmentsClass Attributes and Methods Definitions		
Methods	displayAllAppointments	Displays all appointments for the relevant user.

Table 32. ApproveBooking Class Attributes and Methods Definitions

ApproveBooking Class Attributes and Methods Definitions	
Attributes	
loggedInAgentName	Stores the name of the currently logged-in agent.
Methods	
loginAsAgent(agentName)	Log in the agent and set the current agent's name.
displayApproveBookings	Displays the list of bookings for the agent to approve or decline.

Table 1 explains the RealEstateApp class, which serves as the main entry point for the application. This class is responsible for initializing the entire system and starting the application's workflow. It acts as the foundation for all other components of the project.

Table 2 covers the Login class, which handles the login functionality for users. This class ensures that users can log in with valid credentials and directs them to the appropriate pages or actions based on their role, whether they are registered or new users.

Table 3 describes the SignIn class, which is responsible for presenting the login form to users. It allows users to input their credentials and proceed to access the system if they are authenticated.

Table 4 talks about the SignUp class, which handles the registration process for new users. This class displays the sign-up form and collects user details for creating an account in the system.

Table 5 explains the User class, which represents the basic structure of a user in the system. It includes all the essential details about a user, such as their personal information and their assigned role, like admin, customer, owner, or agent.

Table 6 focuses on the UserRole enum, which defines the different types of users in the system. These roles include Admin, Customer, Owner, and Agent. Each role has specific responsibilities and access privileges within the application.

Table 7 explains the AppointmentBooking class, which manages the booking of appointments. This class allows users, such as customers and house owners, to schedule meetings with agents.

Table 8 introduces the DataStoreUser class, which handles the storage and management of all user data in the system. It includes methods to add, remove, and retrieve user details, making it a key part of managing user information.

Table 9 talks about the Booking class, which represents a scheduled appointment. This class holds all the relevant details for a booking, such as the customer's information, agent details, and the status of the booking.

Table 10 describes the DataStoreBooking class, which manages all bookings in the system. This class handles tasks such as adding new bookings, retrieving past or upcoming bookings, and managing the approval process for pending bookings. Table 11 focuses on the Property class, which represents individual properties in the system. This class includes all the information related to a property, such as its address, type, and price.

Table 12 introduces the `PropertyType` enum, which defines the various categories of properties in the system. Examples include Single Family, Condo, Townhouse, and Apartment.

Table 13 explains the “City” enum, which standardizes the city names used in the system. It ensures that properties are categorized correctly by their location.

Table 14 talks about the `DataStoreProperty` class, which manages property data in the system. It handles both approved properties and pending properties waiting for admin approval. This class ensures that property records are organized and accessible.

Table 15 describes the `AdminUser` class, which represents the admin users in the system. Admins are responsible for managing properties, users, and other administrative tasks.

Table 16 focuses on the `AdminPage` class, which provides the interface for admin users. This class displays options and allows admins to perform their tasks effectively.

Table 17 introduces the `DisplayUsers` class, which allows admin users to view a list of all users in the system. This class is helpful for managing user accounts.

Table 18 explains the `EditProperty` class, which enables admins to edit the details of existing properties. It ensures that property information remains accurate and up to date.

Table 19 describes the `AddProperty` class, which allows admins to add new properties to the system. This class is essential for expanding the property listings available to users.

Table 20 focuses on the `RemoveProperty` class, which gives admins the ability to remove properties from the system. This is important for keeping the property database relevant and organized.

Table 21 talks about the ApprovePendingProperties class, which allows admins to review and approve or decline properties submitted by house owners. This ensures only valid listings are made available to users.

Table 22 explains the OwnerUser class, which represents the house owners in the system. Owners can add properties for admin approval and manage their property details.

Table 23 describes the OwnerPage class, which provides an interface for house owners. This class displays options for managing properties and booking appointments.

Table 24 focuses on the AddPendingProperty class, which allows house owners to submit properties for admin approval. It ensures a streamlined process for adding new property listings.

Table 25 introduces the CustomerUser class, which represents the customers in the system. Customers can search for properties and book appointments with agents.

Table 26 describes the CustomerPage class, which provides the dashboard for customer users. It displays options for property searches and other customer-related tasks.

Table 27 explains the Search class, which enables customers to search for properties based on their preferences. It uses filters to refine the search results and display relevant properties.

Table 28 focuses on the AgentUser class, which represents agents in the system. Agents handle tasks like managing appointments and interacting with customers and owners.

Table 29 introduces the `AgentPage` class, which provides the interface for agents. This class allows agents to approve bookings and view their assigned tasks.

Table 30 explains the `DisplayProperty` class, which displays all property listings to users based on their roles. It ensures that users see only the information they are authorized to access.

Table 31 talks about the `ViewAppointments` class, which displays all scheduled appointments to the relevant users. This class helps users keep track of their meetings.

Table 32 describes the `ApproveBooking` class, which allows agents to review and approve or decline booking requests. It ensures that the booking process remains organized and efficient. This breakdown shows how each class contributes to the functionality of the `RealEstateApp` system and ensures that all user roles and their respective tasks are effectively managed.

Key Design Concepts

Encapsulation

Encapsulation is used extensively in Figure 2 to ensure that each class only exposes relevant data and behaviors to other classes. For example, `User` has private attributes (such as `firstName`, `lastName`, and `role`) with public getter and setter methods, ensuring that other classes can access or modify these properties only through controlled methods. This design choice restricts direct access to class internals, protecting the data and maintaining the integrity of each object.

Delegation

Delegation is implemented through action classes like `AddProperty`, `ApproveBooking`, and `Search`. Rather than having complex logic within `UserPage` classes (such as `AdminPage`,

AgentPage), these responsibilities are delegated to specialized action classes. For instance, ApproveBooking is responsible for handling the approval of booking requests, while AddProperty manages the addition of properties. This delegation improves modularity and allows specific tasks to be handled by dedicated classes, which simplifies maintenance and makes the codebase more organized.

Cohesion

The design emphasizes high cohesion by grouping related functions within specific classes. Each class has a clear purpose and responsibility, DataStoreProperty stores and manages Property instances, “Booking” handles booking details, and UserPage classes initialize options specific to user roles. High cohesion ensures that each class is focused on a single aspect of functionality, making the system easier to understand, test, and maintain.

Coupling

The diagram aims for low coupling by minimizing dependencies between classes. For example, UserPage classes (such as AdminPage and AgentPage) interact with DataStore and action classes through well-defined interfaces rather than having direct dependencies on the details of Property or Booking. Where coupling is necessary, it is managed through dependencies and aggregations rather than direct associations, improving flexibility and reducing the impact of changes in one class on other classes.

Inheritance

Inheritance is used to define user roles through a “User” superclass, which is extended by AdminUser, AgentUser, OwnerUser, and CustomerUser. This approach leverages polymorphism, allowing different user types to have distinct behaviors while sharing common properties and

methods defined in User. The UserRole enumeration further supports role-based functionality, associating each User instance with a specific role.

Aggregation

Aggregation is depicted in classes like DataStoreUser and DataStoreProperty, which maintain collections of User and Property instances, respectively. Aggregation is indicated by the open diamond notation, showing that DataStore classes manage but do not own the User or Property objects. This relationship allows DataStore classes to retrieve and store data without creating strong ownership ties, making it easy to manage data without tightly coupling classes.

Composition

Composition is applied to establish a strong relationship between classes where needed. For example, AppointmentBooking might compose Booking objects, as AppointmentBooking is responsible for creating and managing bookings on behalf of User. Since Booking is integral to the appointment functionality, AppointmentBooking could have a composition relationship with Booking, meaning that if AppointmentBooking is destroyed, the associated Booking instances would also be removed.

Flexibility/Maintainability

The design emphasizes flexibility and maintainability by isolating responsibilities through cohesive, loosely coupled classes and using interfaces for action-based interactions. Delegation and encapsulation contribute to flexibility, allowing each class to evolve independently as long as the defined interfaces remain consistent. For example, if additional booking options or property attributes need to be added, these can be introduced without significantly impacting the rest of

the system. Furthermore, the use of enumeration (UserRole) for roles makes the system adaptable to adding more roles or modifying role-based behaviors.

Additional Design Elements

Methods and Return Types: Each class includes public methods with specific return types that align with its responsibilities, such as `findUserByUsername()` in `DataStoreUser` (returning a `User` object) and `getAllBookings()` in `DataStoreBooking` (returning a list of `Booking` objects). These methods provide clear, purposeful interactions while maintaining encapsulation.

Enumeration for User Roles: The `UserRole` enumeration supports role-based functionality and allows each `User` to be assigned a specific role. This enumeration ensures type safety, reducing the chance of invalid role assignments and allowing for future role expansion without disrupting the existing system structure.

In summary, Figure 2 illustrates a well-structured class diagram for the real estate application, with thoughtful attention to encapsulation, delegation, cohesion, and low coupling. The use of inheritance and aggregation promotes efficient data management, while encapsulation and composition contribute to the integrity and maintainability of the design.

The Real Estate System UML class relationships and multiplicities are described as follows:

1. RealEstateApp ↔ Login

- Relationship Type: Association
- Multiplicity: 1 ↔ 1
- Label: "starts"

- Description: RealEstateApp initializes the Login process, allowing users to authenticate into the system.

2. Login ↔ SignIn

- Relationship Type: Association
- Multiplicity: 1 ↔ 1
- Label: "starts"
- Description: Login uses the SignIn process to authenticate users.

3. Login ↔ SignUp

- Relationship Type: Association
- Multiplicity: 1 ↔ 1
- Label: "starts"
- Description: Login uses the SignUp process to register new users.

4. SignIn ↔ User

- Relationship Type: Association
- Multiplicity: 1 ↔ 1
- Label: "authenticates"
- Description: The SignIn class authenticates a User objects.

5. SignUp ↔ User

- Relationship Type: Association

- Multiplicity: 1 ↔ 1
- Label: "registers"
- Description: The SignUp class registers new User instances.

6. User ↔ UserRole (Enum)

- Relationship Type: Association
- Label: "assigned role"
- Description: Each User has a specific UserRole, which can be ADMIN, CUSTOMER, OWNER, or AGENT.

7. User ↔ AppointmentBooking

- Relationship Type: Association
- Multiplicity: * ↔ *
- Label: "schedules"
- Description: User can schedule multiple AppointmentBooking entries, and each AppointmentBooking can involve multiple users.

8. DataStoreUser ↔ User

- Relationship Type: Aggregation
- Multiplicity: 1 ↔ *
- Label: "stores"
- Description: DataStoreUser stores multiple instances of User without owning them, managing user data in the system.

9. AdminUser ↔ AdminPage

- Relationship Type: Association
- Multiplicity: 1 ↔ 1
- Label: "initializes"
- Description: AdminUser interacts with AdminPage to perform admin-related functionalities.

10. AgentUser ↔ AgentPage

- Relationship Type: Association
- Multiplicity: 1 ↔ 1
- Label: "initializes"
- Description: AgentUser interacts with AgentPage to perform agent-related functionalities.

11. CustomerUser ↔ CustomerPage

- Relationship Type: Association
- Multiplicity: 1 ↔ 1
- Label: "initializes"
- Description: CustomerUser interacts with CustomerPage to perform customer-related functionalities.

12. OwnerUser ↔ OwnerPage

- Relationship Type: Association

- Multiplicity: 1 ↔ 1
- Label: "initializes"
- Description: OwnerUser interacts with OwnerPage to manage owner-specific options like property submission.

13. OwnerUser ↔ AddPendingProperty

- Relationship Type: Association
- Multiplicity: 1 ↔ *
- Label: "creates"
- Description: OwnerUser uses AddPendingProperty to submit properties for approval.

14. CustomerUser ↔ Search

- Relationship Type: Association
- Multiplicity: 1 ↔ *
- Label: "searches"
- Description: CustomerUser interacts with Search to find properties within the system.

15. Property ↔ PropertyType (Enum)

- Relationship Type: Association
- Multiplicity: 1 ↔ 1
- Label: "classified as"

- Description: Each Property has a PropertyType defining its type (e.g., SINGLE_FAMILY, CONDO).

16. Property ↔ City (Enum)

- Relationship Type: Association
- Multiplicity: 1 ↔ 1
- Label: "located in"
- Description: Each Property is located in a specific City.

17. DataStoreProperty ↔ Property

- Relationship Type: Aggregation
- Multiplicity: 1 ↔ *
- Label: "stores"
- Description: DataStoreProperty stores multiple Property objects for management.

18. DisplayProperty ↔ DataStoreProperty

- Relationship Type: Association
- Multiplicity: 1 ↔ 1
- Label: "views"
- Description: DisplayProperty retrieves property data from DataStoreProperty for display purposes.

19. EditProperty ↔ DataStoreProperty

- Relationship Type: Association
- Multiplicity: 1 ↔ 1
- Label: "modifies"
- Description: EditProperty allows modifications to properties stored within DataStoreProperty.

20. AddProperty ↔ DataStoreProperty

- Relationship Type: Association
- Multiplicity: 1 ↔ 1
- Label: "adds"
- Description: AddProperty enables adding new properties to DataStoreProperty.

21. RemoveProperty ↔ DataStoreProperty

- Relationship Type: Association
- Multiplicity: 1 ↔ 1
- Label: "deletes"
- Description: RemoveProperty allows properties to be deleted from DataStoreProperty.

22. ApprovePendingProperties ↔ DataStoreProperty

- Relationship Type: Association
- Multiplicity: 1 ↔ *

- Label: "initializes"
- Description: ApprovePendingProperties processes pending properties submitted for approval.

23. Booking ↔ DataStoreBooking

- Relationship Type: Aggregation
- Multiplicity: 1 ↔ *
- Label: "manages"
- Description: DataStoreBooking manages multiple instances of Booking.

24. User ↔ Booking

- Relationship Type: Association
- Multiplicity: * ↔ *
- Label: "creates"
- Description: Users can create multiple bookings, and each booking is linked to a user.

25. AgentUser ↔ ApproveBooking

- Relationship Type: Association
- Multiplicity: * ↔ *
- Label: "approves"
- Description: AgentUser can approve or reject booking requests through ApproveBooking.

26. DataStoreBooking ↔ ApproveBooking

- Relationship Type: Association
- Multiplicity: 1 ↔ *
- Label: "access"
- Description: DataStoreBooking provides access to booking data for approval or rejection by agents.

27. AppointmentBooking ↔ User

- Relationship Type: Association
- Multiplicity: * ↔ *
- Label: "schedules"
- Description: Users can schedule appointments with agents through AppointmentBooking.

28. DataStoreUser ↔ AppointmentBooking

- Relationship Type: Association
- Multiplicity: 1 ↔ *
- Label: "stores"
- Description: DataStoreUser stores appointment data for various users.

29. Search ↔ DataStoreProperty

- Relationship Type: Association
- Multiplicity: 1 ↔ 1

- Label: "accesses"
- Description: Search accesses DataStoreProperty to find and filter properties based on user criteria.

Enums

1. UserRole

- Used to specify roles like ADMIN, CUSTOMER, OWNER, and AGENT for User.
- Association: User ↔ UserRole
- Label: "assigned role"

2. PropertyType

- Defines property types like SINGLE_FAMILY, CONDO, TOWNHOUSE, APARTMENT.
- Association: Property ↔ PropertyType
- Label: "classified as"

3. City

- Lists cities like TORONTO, MISSISSAUGA, BRAMPTON.
- Association: Property ↔ City
- Label: "located in"

Implementation and Testing

Implementation and Git Repository

Git Repository URL: GitHub Link (<https://github.com/chris-malan/SYST17796malanc.git>)

The Git repository is being used to manage and track the progress of the project. Since the development team consists of one member, Git is being utilized to check code progress weekly during the laboratory sessions on Tuesdays or whenever there is an important update to the code. The repository is organized into the following directories to ensure the project is well-structured.

The repository contains the project and has been pushed to the Git remote repository, including the directories: src, uml, and docs. The src directory contains all the source code files related to the project. The uml directory stores diagrams that visualize the structure of the system. Lastly, the docs directory includes all text files, documents, and other project requirements.

In this project, coding standards are being followed to ensure consistency and readability. These standards include using camelCase for naming conventions, adding comments for maintainability and better understanding of the code, and ensuring modularity by keeping methods and classes short, with each handling only one task. The project is being developed using the NetBeans IDE for programming and Visual Paradigm for creating UML diagrams.

Testing and Data Validation

For testing, the group began by verifying the sign-in and sign-up functionalities to ensure users could log in or register with valid credentials. The system's behavior was checked for both

valid and invalid inputs to confirm proper error handling and authentication. The first issue encountered during testing was with the logout functionality. When a user clicked "Log Out," the application stopped entirely instead of allowing another user to sign in or sign up. This issue was addressed to ensure the application would return to the login or sign-up page after a user logged out, enabling continuous usage.

For the admin role, testing focuses on the ability to view all properties, add new properties, edit existing ones, and manage appointments. During the add and edit property tests, the validation of all input fields was checked to ensure accurate and complete information was entered. Any added or edited properties were verified to appear correctly in the "View All Properties" section. Placeholders were included in the input fields to guide users on the required information. The ability to view and manage appointments was also tested to ensure functionality and usability.

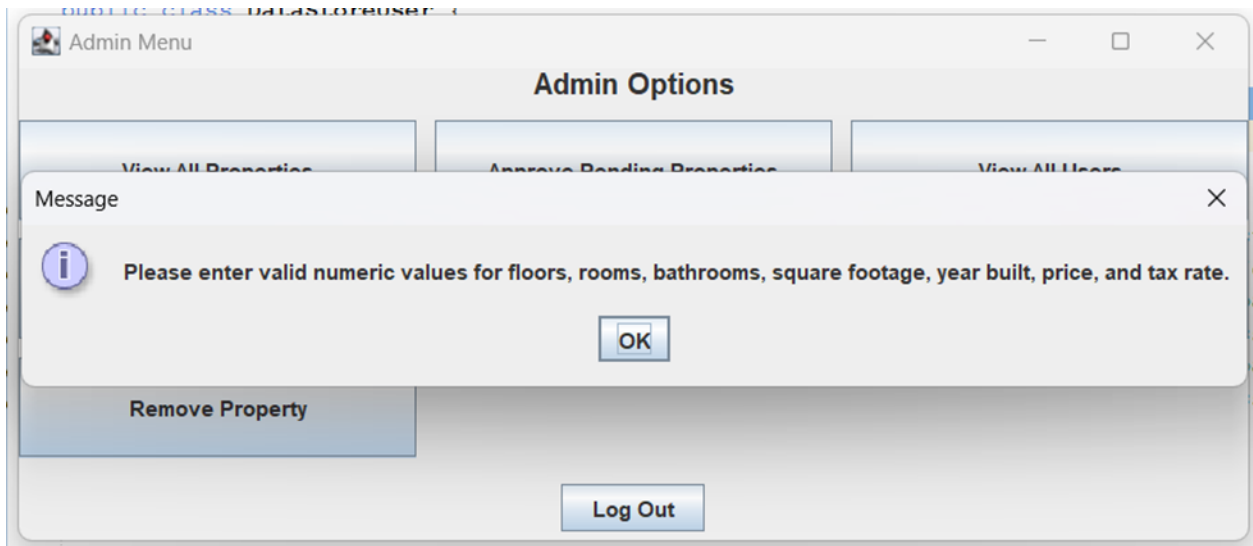


Figure 3. Result of invalid inputs in admin add property.

On figure 3, during testing, the validation of numeric input fields was a key focus to ensure the system handled invalid data effectively. For the admin interface, when invalid or non-numeric values were entered into fields such as floors, rooms, bathrooms, square footage, year built, price, or tax rate, a message was displayed, prompting the user to enter valid numeric values. This ensured that only accurate data could be submitted to the system, reducing potential errors.

For house owners, testing ensured that they could submit properties for admin approval and book appointments. Pending properties were verified to display correctly in the admin's approval section, and appointments were tested to ensure they were booked successfully and visible to the relevant agents.

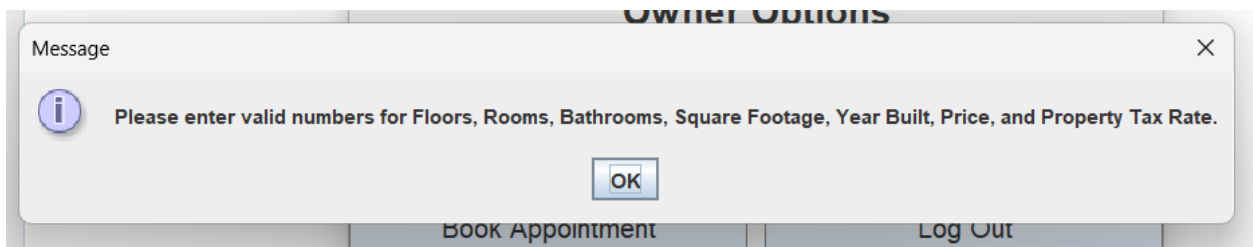


Figure 4. Result of invalid inputs in owner add pending property.

On figure 4, for the owner interface, validation checks were implemented for input fields requiring numeric values. If invalid data was entered, a similar message informed the user of the need to provide valid numbers. This approach enhanced the user experience by guiding users to correct their inputs and ensured data integrity within the system.

For customers, testing focuses on the property search functionality. Filters were checked to confirm they worked correctly, and in cases where no exact matches were found, the system

was tested to provide relevant recommendations. The booking process was also validated to ensure customers could seamlessly book appointments with agents and receive confirmation.

For agents, testing involved verifying their ability to view appointments, approve or decline booking requests, and ensure their interactions with customers and owners were accurately reflected in the system. Approved or declined bookings were tested to ensure they were updated correctly and displayed appropriately.

In all cases, the system was tested for responsiveness, input validation, and the correct data flow between the user interface and the database. The focus was on delivering a smooth and intuitive user experience for all roles, resolving issues like the logout functionality to maintain continuous usability.

Project Feasibility and Operation

The project is a real estate management system that allows users to perform role-specific tasks. Admins can manage properties, approve pending submissions, and view users and appointments. House owners can submit properties for approval and book appointments with agents. Customers can search for properties using filters and book agent appointments, while agents manage and approve bookings. The system ensures seamless user experience through validation, clear role-based interfaces, and efficient data handling.

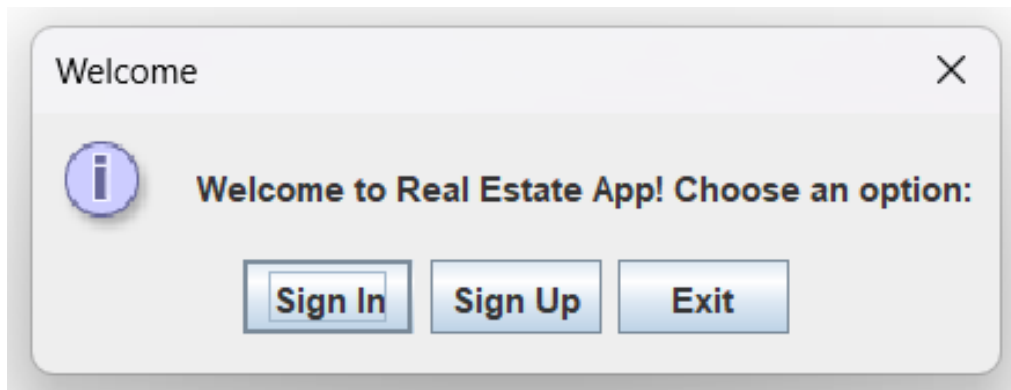


Figure 5. Welcome Screen of the Real Estate App

Figure 5 shows the Welcome Screen of the Real Estate App, where users can choose one of three options: Sign In, Sign Up, or Exit. The Sign In option allows registered users to log in based on their roles, such as Admin, Customer, Owner, or Agent. The Sign-Up option enables new users, specifically house owners or customers, to create an account. The Exit button closes the application. This screen serves as the starting point for navigating the system.

There is a list of users that were hardcoded into the system. You can use these credentials to test the system. The user roles and their credentials are listed as follows:

Admin:

- First Name: John
- Last Name: Doe
- Phone Number: 1234567890
- Username: admin
- Password: adminpass
- Email: admin@example.com

Customers:

First Name: Jane

First Name: Emily

Last Name: Smith

Last Name: Johnson

Phone Number: 0987654321

Phone Number: 5678901234

Username: customer1

Username: customer2

Password: custpass

Password: custpass2

Email: customer1@example.com

Email: customer2@example.com

Owners:

First Name: Michael

First Name: Sarah

Last Name: Brown

Last Name: Davis

Phone Number: 2345678901

Phone Number: 6789012345

Username: owner1

Username: owner2

Password: ownerpass

Password: ownerpass2

Email: owner1@example.com

Email: owner2@example.com

Agents:

First Name: Daniel

First Name: Emma

Last Name: Wilson

Last Name: Taylor

Phone Number: 3456789012

Phone Number: 7890123456

Username: agent1

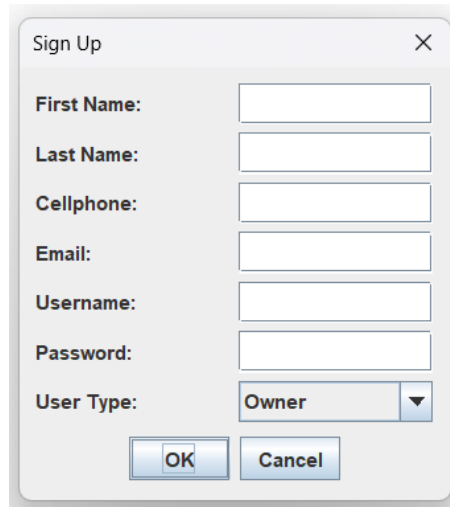
Username: agent2

Password: agentpass

Password: agentpass2

Email: agent1@example.com

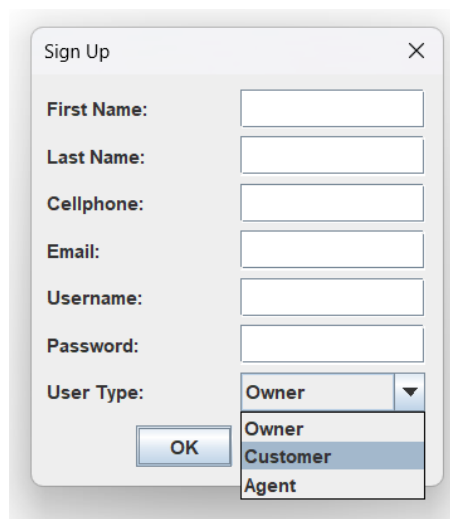
Email: agent2@example.com



A screenshot of a 'Sign Up' dialog box. It contains seven input fields: 'First Name', 'Last Name', 'Cellphone', 'Email', 'Username', 'Password', and 'User Type'. The 'User Type' field is a dropdown menu currently showing 'Owner'. At the bottom are 'OK' and 'Cancel' buttons.

Figure 6. Sign Up Screen

Figure 6 shows the Sign-Up Screen, where new users can create an account. The form collects user information, including First Name, Last Name, Cellphone, Email, Username, and Password. Users must also select their User Type from the dropdown menu, such as "Owner" or "Customer." Clicking OK submits the registration, while Cancel exits the form without saving. This screen ensures proper account creation for house owners and customers in the system.



A screenshot of the 'Sign Up' dialog box with the 'User Type' dropdown menu open. The menu lists three options: 'Owner', 'Customer', and 'Agent'. The 'OK' button is visible to the left of the dropdown.

Figure 7. User Type Dropdown

Figure 7 shows the User Type Dropdown within the Sign-Up Screen, allowing new users to select their role in the system. The dropdown options include Owner, Customer, and Agent, ensuring that the user's role is correctly assigned during registration. This selection determines the features and functionalities available to the user after they log in. This feature ensures that each user is categorized accurately for role-based access.

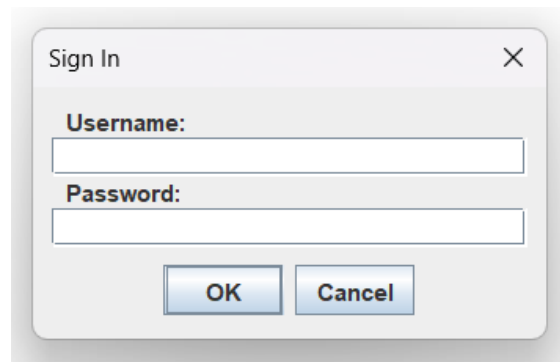


Figure 8. Sign In Screen

Figure 8 shows the Sign in Screen, where existing users can log into their accounts by entering their Username and Password. Clicking OK submits the credentials for authentication, allowing access to the system if valid. The Cancel button exits the sign-in process, returning the user to the previous screen. This feature ensures secure access to role-specific functionalities.

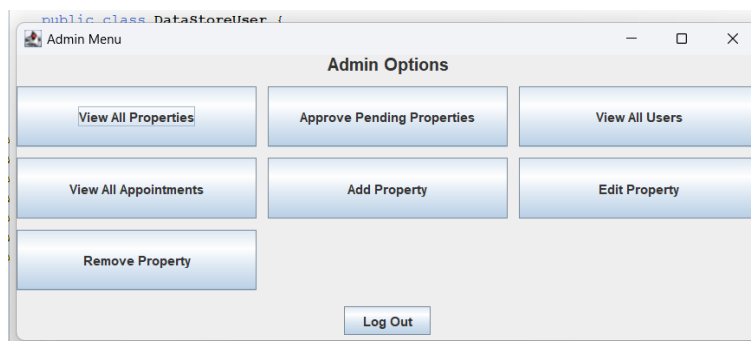
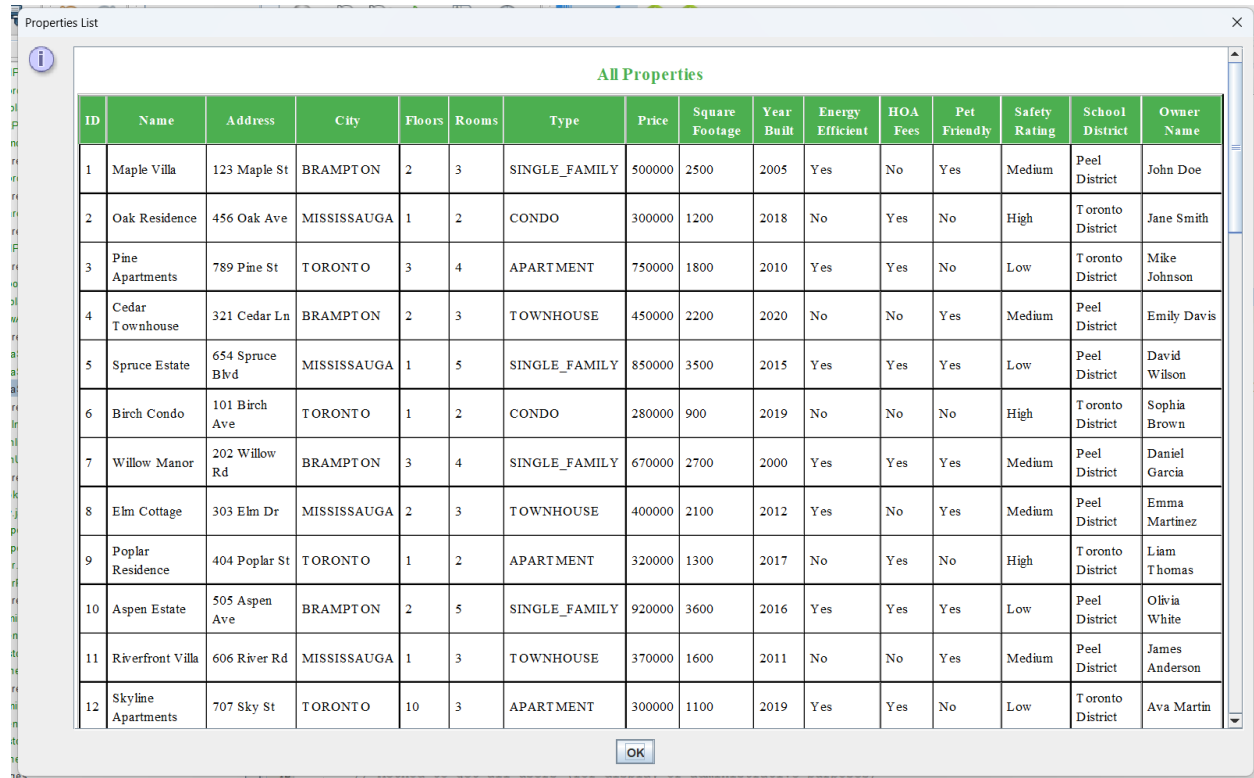


Figure 9. Admin Options Page

Figure 9 shows the Admin Options Page, which provides a dashboard for administrators to manage the system. Admins can access various functionalities, including View All Properties, Approve Pending Properties, View All Users, View All Appointments, Add Property, Edit Property, and Remove Property. The Log Out button allows the admin to exit their session securely. This page enables admins to oversee and manage properties, users, and appointments efficiently.

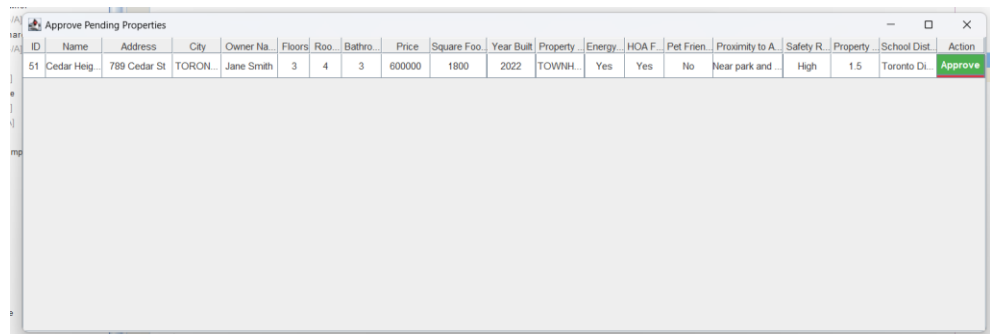


ID	Name	Address	City	Floors	Rooms	Type	Price	Square Footage	Year Built	Energy Efficient	HOA Fees	Pet Friendly	Safety Rating	School District	Owner Name
1	Maple Villa	123 Maple St	BRAMPTON	2	3	SINGLE_FAMILY	500000	2500	2005	Yes	No	Yes	Medium	Peel District	John Doe
2	Oak Residence	456 Oak Ave	MISSISSAUGA	1	2	CONDO	300000	1200	2018	No	Yes	No	High	Toronto District	Jane Smith
3	Pine Apartments	789 Pine St	TORONTO	3	4	APARTMENT	750000	1800	2010	Yes	Yes	No	Low	Toronto District	Mike Johnson
4	Cedar Townhouse	321 Cedar Ln	BRAMPTON	2	3	TOWNHOUSE	450000	2200	2020	No	No	Yes	Medium	Peel District	Emily Davis
5	Spruce Estate	654 Spruce Blvd	MISSISSAUGA	1	5	SINGLE_FAMILY	850000	3500	2015	Yes	Yes	Yes	Low	Peel District	David Wilson
6	Birch Condo	101 Birch Ave	TORONTO	1	2	CONDO	280000	900	2019	No	No	No	High	Toronto District	Sophia Brown
7	Willow Manor	202 Willow Rd	BRAMPTON	3	4	SINGLE_FAMILY	670000	2700	2000	Yes	Yes	Yes	Medium	Peel District	Daniel Garcia
8	Elm Cottage	303 Elm Dr	MISSISSAUGA	2	3	TOWNHOUSE	400000	2100	2012	Yes	No	Yes	Medium	Peel District	Emma Martinez
9	Poplar Residence	404 Poplar St	TORONTO	1	2	APARTMENT	320000	1300	2017	No	Yes	No	High	Toronto District	Liam Thomas
10	Aspen Estate	505 Aspen Ave	BRAMPTON	2	5	SINGLE_FAMILY	920000	3600	2016	Yes	Yes	Yes	Low	Peel District	Olivia White
11	Riverfront Villa	606 River Rd	MISSISSAUGA	1	3	TOWNHOUSE	370000	1600	2011	No	No	Yes	Medium	Peel District	James Anderson
12	Skyline Apartments	707 Sky St	TORONTO	10	3	APARTMENT	300000	1100	2019	Yes	Yes	No	Low	Toronto District	Ava Martin

Figure 10. View All Properties Page

Figure 10 shows the View All Properties Page, where a comprehensive list of all properties in the system is displayed. Each property is detailed with attributes such as ID, Name, Address, City, Floors, Rooms, Type, Price, Square Footage, Year Built, and additional features like Energy Efficient, HOA Fees, Pet Friendly, Safety Rating, School District, and Owner Name. This page allows users, particularly admins, to view and manage the properties efficiently, ensuring all

relevant information is easily accessible in a tabular format. The OK button closes the window after viewing.



ID	Name	Address	City	Owner Na	Floors	Roo	Bathro	Price	Square Foo	Year Built	Property	Energy	HOA F	Pet Frien	Proximity to A	Safety R	Property	School Dist	Action
51	Cedar Heig	789 Cedar St	TORON	Jane Smith	3	4	3	600000	1800	2022	TOWNH	Yes	Yes	No	Near park and	High	1.5	Toronto Di	Approve

Figure 11. Admin Screen on Approve Pending Properties

Figure 11 shows the Admin Screen on Approve Pending Properties, where administrators can view and manage properties submitted by owners for approval. Each pending property is displayed on a table with details such as ID, Name, Address, City, Owner Name, Floors, Rooms, Bathrooms, Price, Square Footage, Year Built, Property Type, and other features. The Approve button in the Action column allows the admin to approve the property, moving it to the approved properties list. This screen ensures that only validated properties are added to the system.

Figure 12 shows the Admin Screen in Viewing All Users, where administrators can see a complete list of users registered in the system. The table displays user details, including First Name, Last Name, Cellphone, Username, Email, and their Role (Admin, Customer, Owner, or Agent). This screen allows admins to easily manage and review user information, ensuring transparency and proper organization of all registered accounts. The OK button allows the admin to close the window once the review is complete.

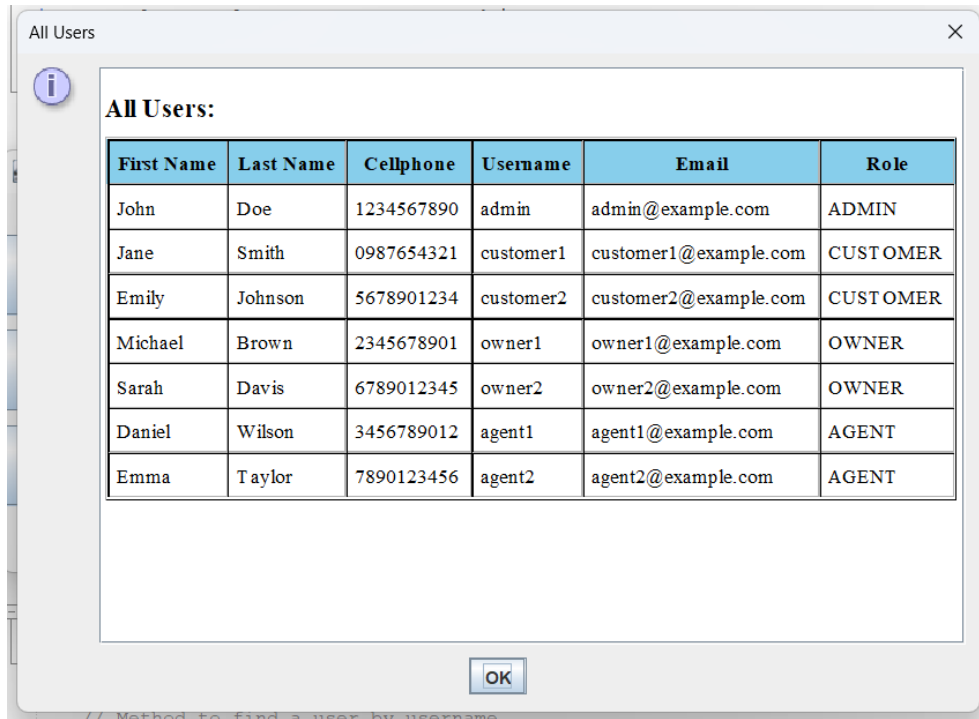


Figure 12. Admin Screen in Viewing All Users.

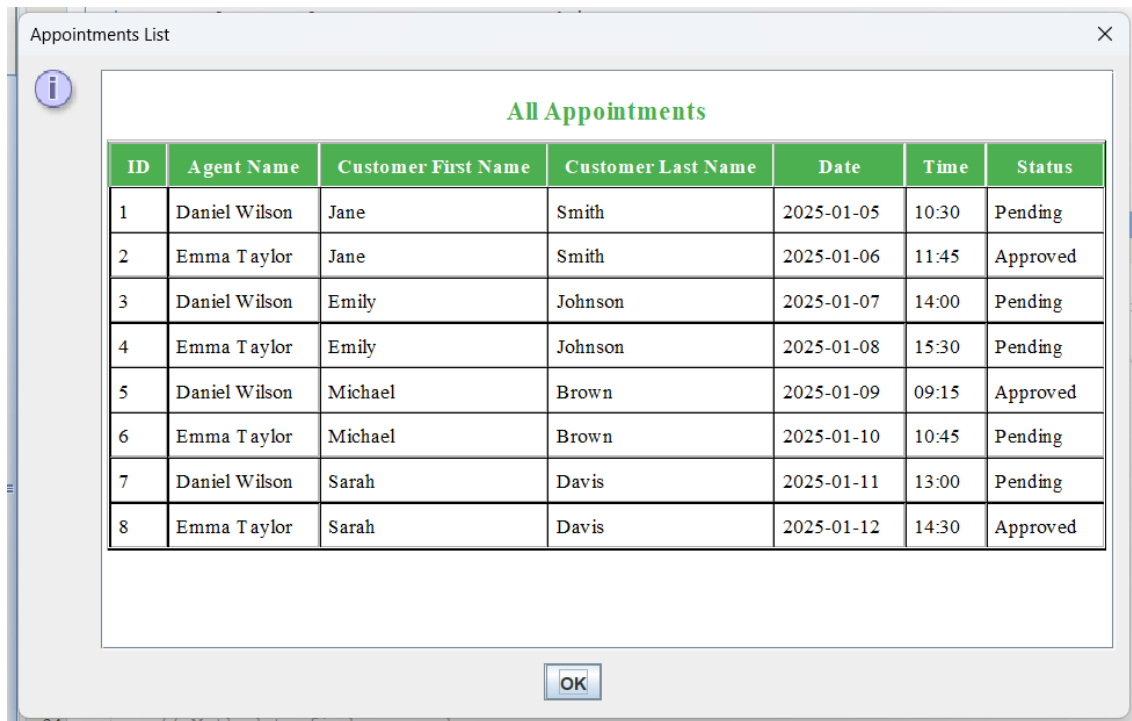


Figure 13. Admin Screen on Viewing All Appointments

Figure 13 shows the Admin Screen on Viewing All Appointments, where administrators can see a comprehensive list of appointments scheduled between agents and customers. The table includes details such as ID, Agent Name, Customer First Name, Customer Last Name, Date, Time, and the Status of each appointment. This screen allows admins to monitor and review all appointments in the system. The OK button is used to close the window after viewing the details.

The 'Add Property' dialog box contains the following fields and controls:

- Name:** Text input with placeholder 'e.g., Maple Villa'
- Address:** Text input with placeholder 'e.g., 123 Maple St'
- City:** Dropdown menu with 'ANY' selected
- Floors:** Text input with placeholder 'e.g., 2'
- Rooms:** Text input with placeholder 'e.g., 3'
- Bathrooms:** Text input with placeholder 'e.g., 2'
- Type:** Dropdown menu with 'ANY' selected
- Square Footage:** Text input with placeholder 'e.g., 1500'
- Year Built:** Text input with placeholder 'e.g., 2010'
- Price:** Text input with placeholder 'e.g., 300000'
- Energy Efficient:** Checkbox
- HOA Fees:** Checkbox
- Pet Friendly:** Checkbox
- Proximity to Amenities:** Text input with placeholder 'e.g., Near schools'
- Safety Rating:** Text input with placeholder 'e.g., High'
- Property Tax Rate:** Text input with placeholder 'e.g., 1.2'
- School District:** Text input with placeholder 'e.g., Peel District'
- Owner Name:** Text input with placeholder 'e.g., John Doe'
- Buttons:** 'OK' and 'Cancel' buttons at the bottom

Figure 14. Admin Screen on Adding a property.

Figure 14 shows the Admin Screen on adding a property, where administrators can input details to add a new property to the system. The form includes fields for Name, Address, City, Floors, Rooms, Bathrooms, Type, Square Footage, Year Built, Price, and additional attributes such as Energy Efficient, HOA Fees, and Pet Friendly. There are also fields for Proximity to Amenities, Safety Rating, Property Tax Rate, School District, and Owner Name. The OK button submits the property information, while the Cancel button exits the form without saving. This screen ensures accurate and complete property data entry.

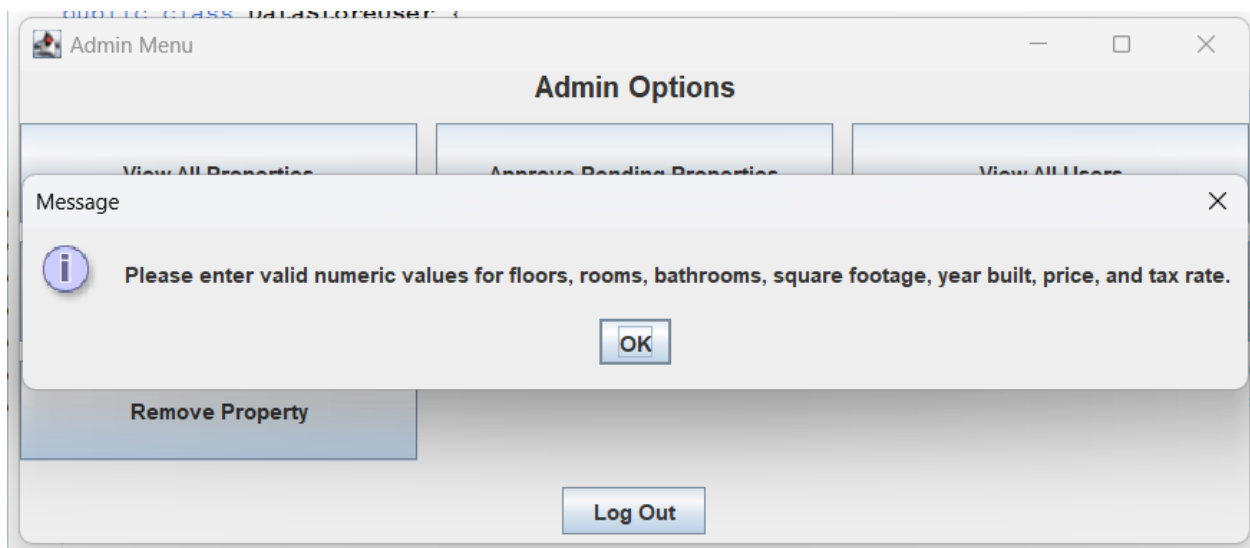


Figure 15. Warning Screen on Invalid Inputs on Adding Properties

Figure 15 shows the warning screen on invalid inputs when adding properties, which appears when an admin enters non-numeric or invalid values in fields requiring numeric data. These fields include Floors, Rooms, Bathrooms, Square Footage, Year Built, Price, and Tax Rate. The warning prompts the admin to correct their inputs before proceeding. Clicking OK closes the warning and allows the user to re-enter the data. This feature ensures that only valid and accurate data is entered into the system.

ID	Name	Address	City	Floors	Rooms	Type	Price	Square Footage	Year Built	Energy Efficient	HOA Fees	Pet Friendly	Safety Rating	School District	Owner Name
1	Maple Villa	123 Maple St	BRAMPTON	2	3	SINGLE_FAMILY	500000	2500	2005	Yes	No	Yes	Medium	Peel District	John Doe
2	Oak Residence	456 Oak Ave	MISSISSAUGA	1	2	CONDO	300000	1200	2018	No	Yes	No	High	Toronto District	Jane Smith
3	Pine Apartments	789 Pine St	TORONTO	3	4	APARTMENT	750000	1800	2010	Yes	Yes	No	Low	Toronto District	Mike Johnson
4	Cedar Townhouse	321 Cedar Ln	BRAMPTON	2	3	TOWNHOUSE	450000	2200	2020	No	No	Yes	Medium	Peel District	Emily Davis
5	Spruce Estate	654 Spruce Blvd	MISSISSAUGA	1	5	SINGLE_FAMILY	850000	3500	2015	Yes	Yes	Yes	Low	Peel District	David Wilson
6	Birch Condo	101 Birch Ave	TORONTO	1	2	CONDO	280000	900	2019	No	No	No	High	Toronto District	Sophia Brown
7	Willow Manor	202 Willow Rd	BRAMPTON	3	4	SINGLE_FAMILY	670000	2700	2000	Yes	Yes	Yes	Medium	Peel District	Daniel Garcia
8	Elm Cottage	303 Elm Dr	MISSISSAUGA	2	3	TOWNHOUSE	400000	2100	2012	Yes	No	Yes	Medium	Peel District	Emma Martinez
9	Poplar Residence	404 Poplar St	TORONTO	1	2	APARTMENT	320000	1300	2017	No	Yes	No	High	Toronto District	Liam Thomas
10	Aspen Estate	505 Aspen Ave	BRAMPTON	2	5	SINGLE_FAMILY	920000	3600	2016	Yes	Yes	Yes	Low	Peel District	Olivia White
11	Riverfront Villa	606 River Rd	MISSISSAUGA	1	3	TOWNHOUSE	370000	1600	2011	No	No	Yes	Medium	Peel District	James Anderson
12	Skyline Apartments	707 Sky St	TORONTO	10	3	APARTMENT	300000	1100	2019	Yes	Yes	No	Low	Toronto District	Ava Martin

Figure 16. Displaying All Properties Before Editing properties.

Figure 16 shows the display of all properties before editing a property screen, where all properties in the system are displayed in a tabular format. This list provides details such as ID, Name, Address, City, Floors, Rooms, Type, Price, and other attributes. This screen allows admins to review the existing properties before selecting one to edit. The OK button closes the window once the admin has finished reviewing. This ensures that admins can verify property details before making changes.

Admin Menu

Admin Options

View All Properties

View All Appointments

Remove Property

View All Users

Edit Property

Log Out

Input

?

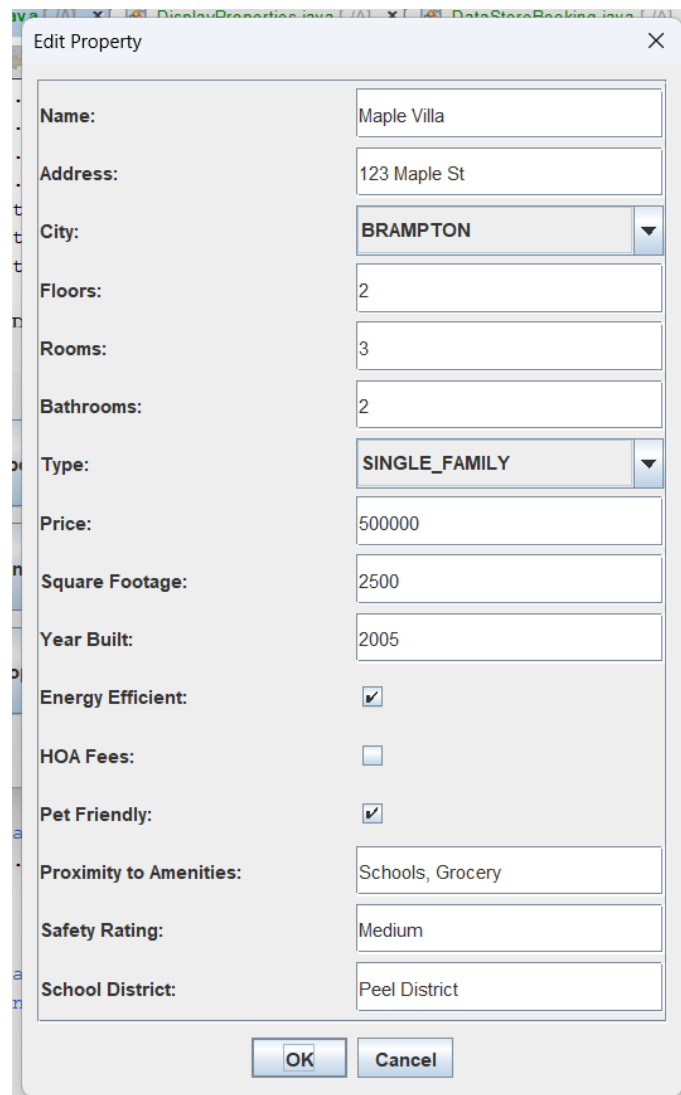
Enter the ID of the property to edit:

OK

Cancel

Figure 17. Prompt for Property ID in Editing Property.

Figure 17 shows the Prompt for Property ID in Editing Property, where the admin is required to enter the unique ID of the property they wish to edit. This prompt ensures that the system identifies the correct property to modify. After entering the ID, clicking OK proceeds to the editing screen, while clicking Cancel exits the prompt without making any changes. This feature provides precision and ensures only the intended property is edited.



The image shows a screenshot of a web application window titled "Edit Property". The window contains a form with the following fields and values:

Field	Value
Name:	Maple Villa
Address:	123 Maple St
City:	BRAMPTON
Floors:	2
Rooms:	3
Bathrooms:	2
Type:	SINGLE_FAMILY
Price:	500000
Square Footage:	2500
Year Built:	2005
Energy Efficient:	<input checked="" type="checkbox"/>
HOA Fees:	<input type="checkbox"/>
Pet Friendly:	<input checked="" type="checkbox"/>
Proximity to Amenities:	Schools, Grocery
Safety Rating:	Medium
School District:	Peel District

At the bottom of the form are two buttons: "OK" and "Cancel".

Figure 18. Edit Property Screen of Selected Property

Figure 18 shows the Edit Property Screen of Selected Property, where administrators can modify the details of a property they have chosen to edit. The form displays the current information for the property, including Name, Address, City, Floors, Rooms, Bathrooms, Type, Price, Square Footage, Year Built, and additional attributes such as Energy Efficient, HOA Fees, and Pet Friendly. Fields like Proximity to Amenities, Safety Rating, and School District can also be updated. Clicking OK saves the changes, while Cancel exits without applying any modifications. This screen ensures accurate and efficient updates to property details.

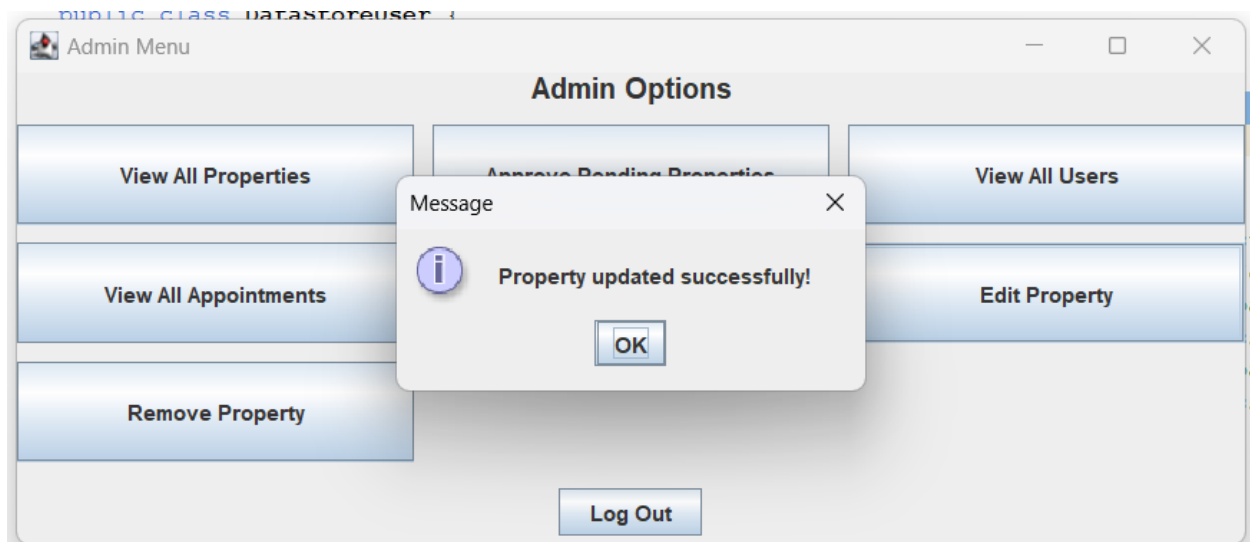


Figure 19. Successful Prompt in Editing Property.

Figure 19 shows the Successful Prompt in Editing Property, confirming that the property details have been updated successfully. After an admin modifies a property's information and clicks OK on the edit form, this message appears to assure the user that the changes have been saved. Clicking OK closes the prompt and returns the admin to the Admin Options menu. This feature provides feedback to the user, ensuring clarity and successful operation.

All Properties															
ID	Name	Address	City	Floors	Rooms	Type	Price	Square Footage	Year Built	Energy Efficient	HOA Fees	Pet Friendly	Safety Rating	School District	Owner Name
1	Maple Villa	123 Maple St	BRAMPTON	2	3	SINGLE_FAMILY	500000	2500	2005	Yes	No	Yes	Medium	Peel District	John Doe
2	Oak Residence	456 Oak Ave	MISSISSAUGA	1	2	CONDO	300000	1200	2018	No	Yes	No	High	Toronto District	Jane Smith
3	Pine Apartments	789 Pine St	TORONTO	3	4	APARTMENT	750000	1800	2010	Yes	Yes	No	Low	Toronto District	Mike Johnson
4	Cedar Townhouse	321 Cedar Ln	BRAMPTON	2	3	TOWNHOUSE	450000	2200	2020	No	No	Yes	Medium	Peel District	Emily Davis
5	Spruce Estate	654 Spruce Blvd	MISSISSAUGA	1	5	SINGLE_FAMILY	850000	3500	2015	Yes	Yes	Yes	Low	Peel District	David Wilson
6	Birch Condo	101 Birch Ave	TORONTO	1	2	CONDO	280000	900	2019	No	No	No	High	Toronto District	Sophia Brown
7	Willow Manor	202 Willow Rd	BRAMPTON	3	4	SINGLE_FAMILY	670000	2700	2000	Yes	Yes	Yes	Medium	Peel District	Daniel Garcia
8	Elm Cottage	303 Elm Dr	MISSISSAUGA	2	3	TOWNHOUSE	400000	2100	2012	Yes	No	Yes	Medium	Peel District	Emma Martinez
9	Poplar Residence	404 Poplar St	TORONTO	1	2	APARTMENT	320000	1300	2017	No	Yes	No	High	Toronto District	Liam Thomas
10	Aspen Estate	505 Aspen Ave	BRAMPTON	2	5	SINGLE_FAMILY	920000	3600	2016	Yes	Yes	Yes	Low	Peel District	Olivia White
11	Riverfront Villa	606 River Rd	MISSISSAUGA	1	3	TOWNHOUSE	370000	1600	2011	No	No	Yes	Medium	Peel District	James Anderson
12	Skyline Apartments	707 Sky St	TORONTO	10	3	APARTMENT	300000	1100	2019	Yes	Yes	No	Low	Toronto District	Ava Martin

Figure 20. View All Properties Before Removing Property.

Figure 20 shows the View All Properties Before Removing Property screen, where all the properties in the system are listed in a table. This screen allows admins to review property details, such as ID, Name, Address, City, Floors, Rooms, Type, Price, and other attributes, before selecting a property to remove. The OK button closes the window after reviewing the list. This ensures that admins can verify the property information before proceeding with the removal process.

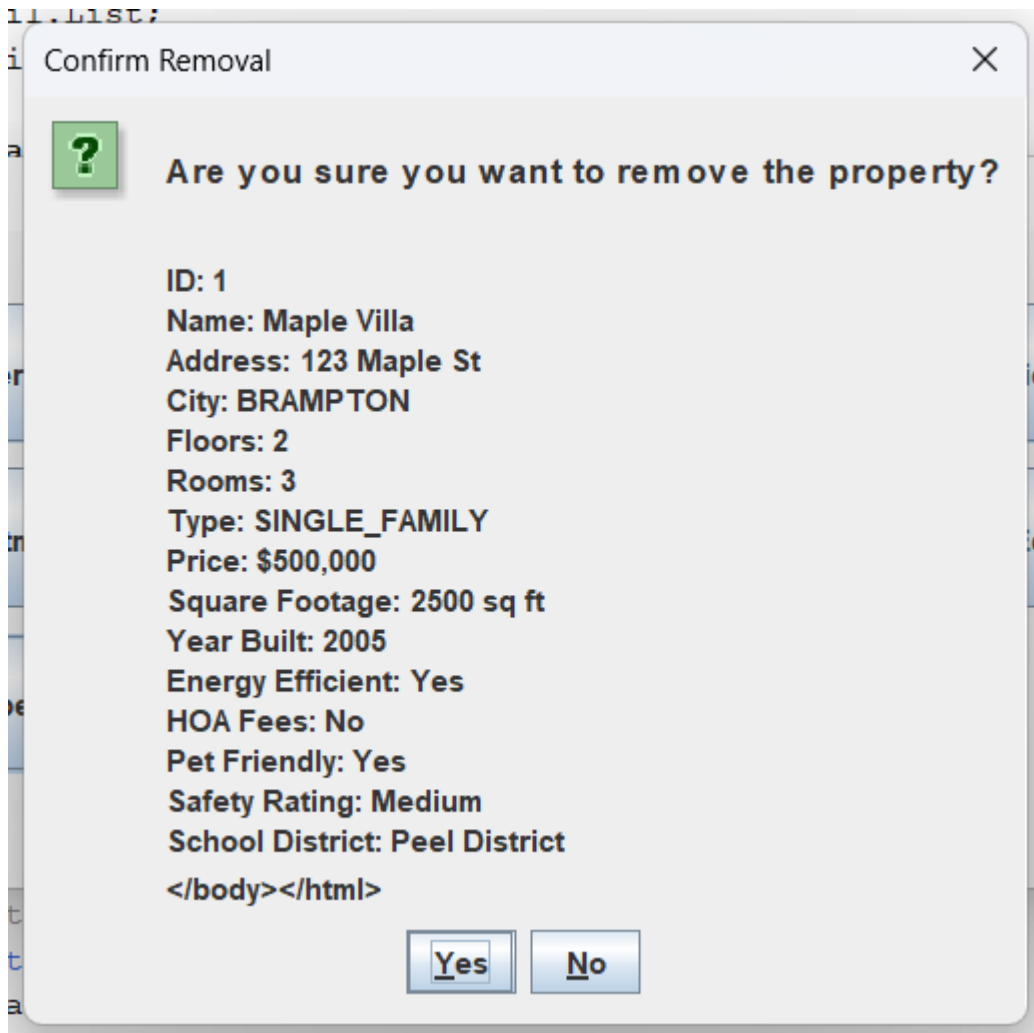


Figure 21. Warning Prompt Asking Admin Confirmation for Removing Property.

Figure 21 shows the Warning Prompt Asking Admin Confirmation for Removing Property, which appears before deleting a property from the system. The prompt displays all relevant details of the selected property, including ID, Name, Address, City, Floors, Rooms, Type, Price, and additional attributes. This ensures the admin reviews the property information before confirming the removal. Clicking Yes confirms and removes the property, while No cancels the action and returns to the previous screen. This feature prevents accidental deletions and ensures careful decision-making.

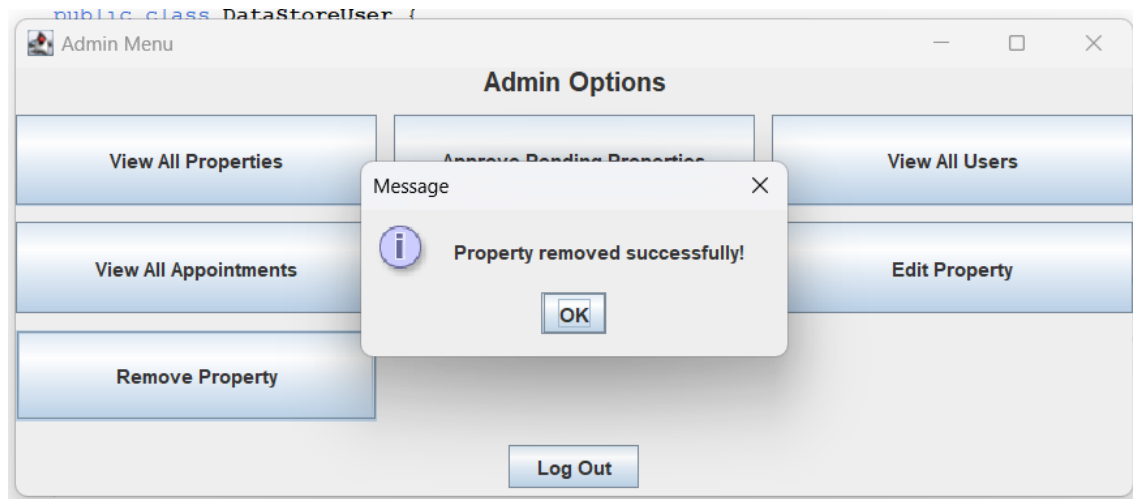


Figure 22. Prompt on Removing Property Successful.

Figure 22 shows the Prompt on Removing Property Successful, which confirms that the selected property has been successfully removed from the system. This message appears after the admin completes the removal process by confirming the action in the previous prompt. Clicking OK closes the message and returns the admin to the Admin Options menu. This feedback ensures the admin is informed of the successful completion of the action.

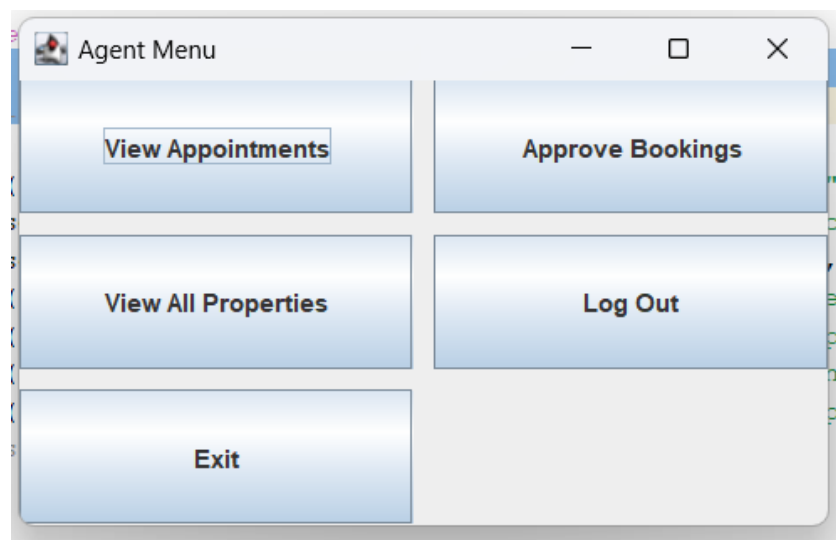
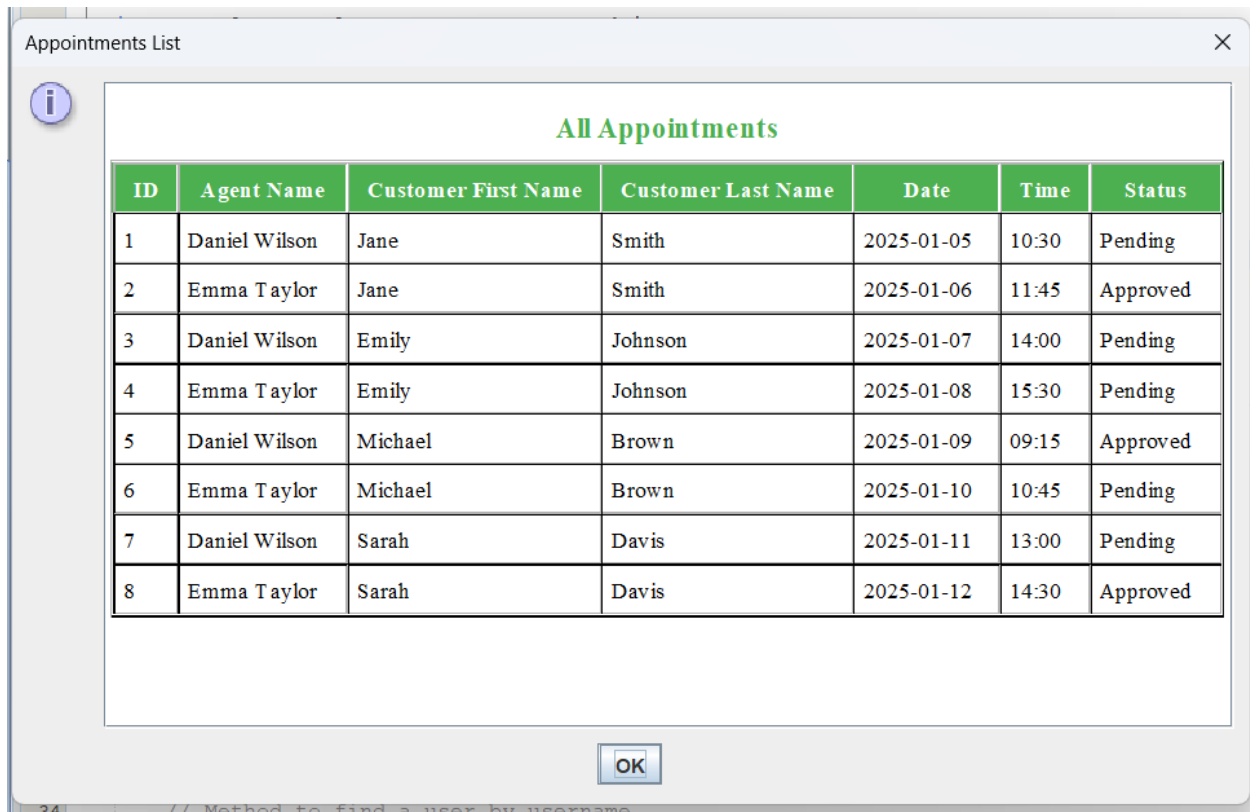


Figure 23. Agent Page

Figure 23 shows the Agent Page, which provides agents with access to their role-specific functionalities. Agents can choose from options such as View Appointments to review scheduled meetings, Approve Bookings to manage pending booking requests, View All Properties to browse listed properties, Log Out to end their session, and exit to close the application. This page ensures that agents can efficiently manage their tasks and navigate the system with ease.



ID	Agent Name	Customer First Name	Customer Last Name	Date	Time	Status
1	Daniel Wilson	Jane	Smith	2025-01-05	10:30	Pending
2	Emma Taylor	Jane	Smith	2025-01-06	11:45	Approved
3	Daniel Wilson	Emily	Johnson	2025-01-07	14:00	Pending
4	Emma Taylor	Emily	Johnson	2025-01-08	15:30	Pending
5	Daniel Wilson	Michael	Brown	2025-01-09	09:15	Approved
6	Emma Taylor	Michael	Brown	2025-01-10	10:45	Pending
7	Daniel Wilson	Sarah	Davis	2025-01-11	13:00	Pending
8	Emma Taylor	Sarah	Davis	2025-01-12	14:30	Approved

Figure 24. Agent Page on Viewing All Appointments.

Figure 24 shows the Agent Page on Viewing All Appointments, where agents can see a list of all scheduled appointments. The table includes details such as ID, Agent Name, Customer First Name, Customer Last Name, Date, Time, and the current Status (e.g., Pending or Approved). This view enables agents to manage and track their appointments effectively. The OK button

closes the window after the review is complete. This feature ensures agents have clear access to their schedule and responsibilities.

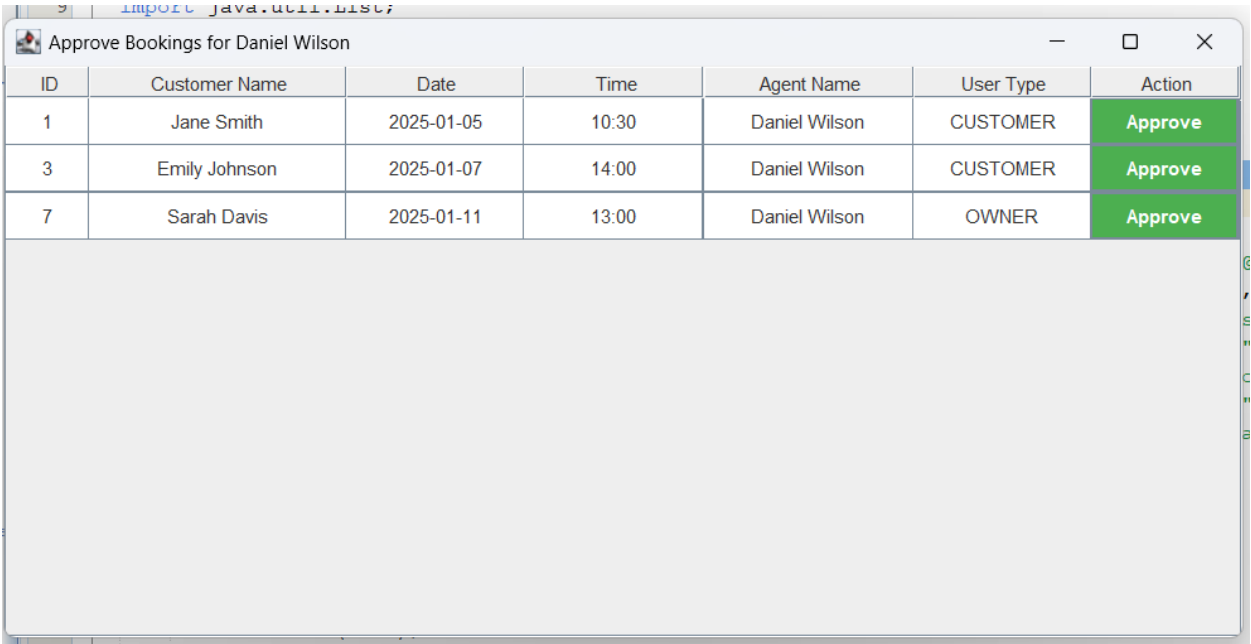
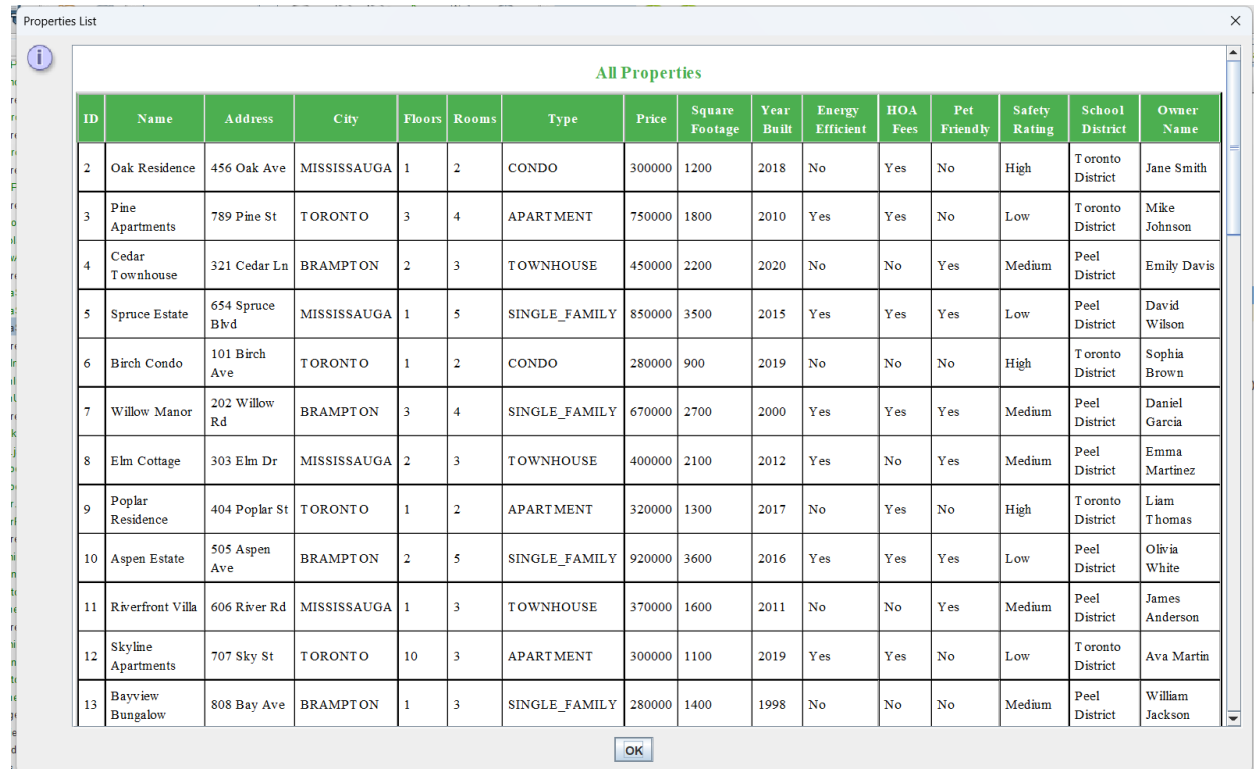


Figure 25. Agent Screen in Approving Appointments Booked to Them.

Figure 25 shows the Agent Screen in Approving Appointments Booked to Them, where agents can view and manage their pending appointment requests. The table includes details such as ID, Customer Name, Date, Time, Agent Name, User Type (Customer or Owner), and an Action column with an Approve button. Agents can approve appointments by clicking the corresponding button, ensuring efficient management of their booking schedules. This feature allows agents to confirm or handle appointments as needed.

Figure 26 shows the Agent Screen Viewing All Properties, where agents can access a complete list of all properties in the system. The table displays property details such as ID, Name, Address, City, Floors, Rooms, Type, Price, Square Footage, Year Built, and additional

attributes like Energy Efficient, HOA Fees, Pet Friendly, Safety Rating, School District, and Owner Name. This screen enables agents to review available properties to better assist customers and owners. The OK button closes the window once the agent finishes reviewing the information.



All Properties															
ID	Name	Address	City	Floors	Rooms	Type	Price	Square Footage	Year Built	Energy Efficient	HOA Fees	Pet Friendly	Safety Rating	School District	Owner Name
2	Oak Residence	456 Oak Ave	MISSISSAUGA	1	2	CONDO	300000	1200	2018	No	Yes	No	High	Toronto District	Jane Smith
3	Pine Apartments	789 Pine St	TORONTO	3	4	APARTMENT	750000	1800	2010	Yes	Yes	No	Low	Toronto District	Mike Johnson
4	Cedar Townhouse	321 Cedar Ln	BRAMPTON	2	3	TOWNHOUSE	450000	2200	2020	No	No	Yes	Medium	Peel District	Emily Davis
5	Spruce Estate	654 Spruce Blvd	MISSISSAUGA	1	5	SINGLE_FAMILY	850000	3500	2015	Yes	Yes	Yes	Low	Peel District	David Wilson
6	Birch Condo	101 Birch Ave	TORONTO	1	2	CONDO	280000	900	2019	No	No	No	High	Toronto District	Sophia Brown
7	Willow Manor	202 Willow Rd	BRAMPTON	3	4	SINGLE_FAMILY	670000	2700	2000	Yes	Yes	Yes	Medium	Peel District	Daniel Garcia
8	Elm Cottage	303 Elm Dr	MISSISSAUGA	2	3	TOWNHOUSE	400000	2100	2012	Yes	No	Yes	Medium	Peel District	Emma Martinez
9	Poplar Residence	404 Poplar St	TORONTO	1	2	APARTMENT	320000	1300	2017	No	Yes	No	High	Toronto District	Liam Thomas
10	Aspen Estate	505 Aspen Ave	BRAMPTON	2	5	SINGLE_FAMILY	920000	3600	2016	Yes	Yes	Yes	Low	Peel District	Olivia White
11	Riverfront Villa	606 River Rd	MISSISSAUGA	1	3	TOWNHOUSE	370000	1600	2011	No	No	Yes	Medium	Peel District	James Anderson
12	Skyline Apartments	707 Sky St	TORONTO	10	3	APARTMENT	300000	1100	2019	Yes	Yes	No	Low	Toronto District	Ava Martin
13	Bayview Bungalow	808 Bay Ave	BRAMPTON	1	3	SINGLE_FAMILY	280000	1400	1998	No	No	No	Medium	Peel District	William Jackson

Figure 26. Agent Screen Viewing All Properties.

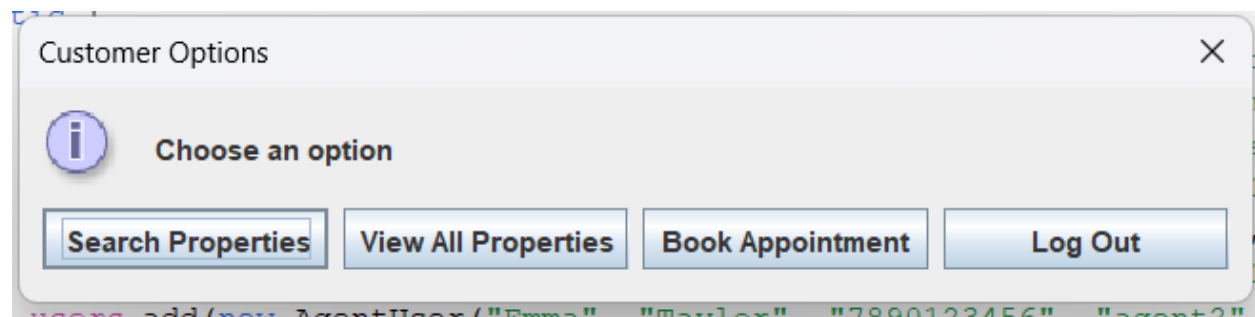
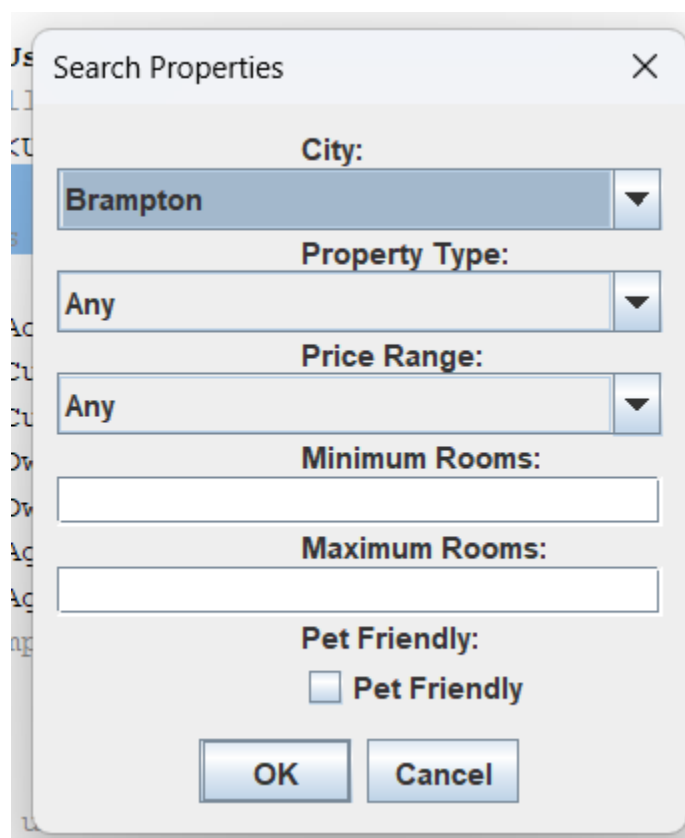


Figure 27. Customer page

Figure 27 shows the Customer Page, where customers can access various options related to their user role. They can choose to Search Properties to find listings matching their preferences, View All Properties to browse the complete property list, or Book Appointment to schedule meetings with agents. The Log Out button allows customers to securely end their session. This interface provides a simple and efficient way for customers to interact with the system and manage their property-related activities.



The image shows a 'Search Properties' dialog box with the following fields and controls:

- City:** A dropdown menu with 'Brampton' selected.
- Property Type:** A dropdown menu with 'Any' selected.
- Price Range:** A dropdown menu with 'Any' selected.
- Minimum Rooms:** An empty text input field.
- Maximum Rooms:** An empty text input field.
- Pet Friendly:** A checkbox labeled 'Pet Friendly' which is currently unchecked.
- Buttons:** 'OK' and 'Cancel' buttons at the bottom.

Figure 28. Customer Screen on Searching Properties.

Figure 28 shows the Customer Screen on Searching Properties, where customers can specify their search criteria to find properties that match their preferences. The form includes fields for selecting City, Property Type, and Price Range, as well as input fields for Minimum

Rooms and Maximum Rooms. A checkbox for Pet Friendly allows customers to filter properties that accommodate pets. Clicking OK initiates the search, while Cancel exits the form without conducting a search. This feature provides a tailored property search experience for customers.

Search Results

Showing Properties Based on this Input:

- City: Brampton
- Property Type: Any
- Price Range: Any
- Rooms: N/A - N/A
- Pet Friendly: No

Search Results:

ID	Name	Address	City	Floors	Rooms	Bathrooms	Type	Price	Square Footage	Year Built	Energy Efficient	HOA Fees	Pet Friendly	Proximity to Amenities	Safety Rating	Property
4	Cedar Townhouse	321 Cedar Ln	BRAMPTON	2	3	3	TOWNHOUSE	450000	2200	2020	No	No	Yes	Schools, Grocery, Hospital	Medium	1.3
7	Willow Manor	202 Willow Rd	BRAMPTON	3	4	3	SINGLE_FAMILY	670000	2700	2000	Yes	Yes	Yes	Schools, Park	Medium	1.4
10	Aspen Estate	505 Aspen Ave	BRAMPTON	2	5	4	SINGLE_FAMILY	920000	3600	2016	Yes	Yes	Yes	Nature Trails, Schools	Low	1.7
13	Bayview Bungalow	808 Bay Ave	BRAMPTON	1	3	1	SINGLE_FAMILY	280000	1400	1998	No	No	No	Beach, Schools	Medium	1.4
16	Forest Townhouse	202 Forest St	BRAMPTON	2	4	2	TOWNHOUSE	410000	2100	2007	Yes	No	Yes	Schools, Gym	Medium	1.3
19	Sunny Apartment	505 Sun St	BRAMPTON	1	2	1	APARTMENT	330000	1100	2020	No	Yes	No	Grocery, Bus Stop	High	1.1
22	Ocean Breeze	808 Ocean Dr	BRAMPTON	2	3	2	SINGLE_FAMILY	560000	1500	2017	Yes	Yes	Yes	Beach, Schools	Medium	1.3
25	Greenfield Villa	202 Green Ln	BRAMPTON	2	4	3	SINGLE_FAMILY	670000	2500	2011	Yes	No	Yes	Parks, Grocery	Medium	1.3
29	Suburban Home	606 Suburb Ave	BRAMPTON	2	4	3	SINGLE_FAMILY	740000	3000	2013	Yes	No	Yes	Parks, Schools	Medium	1.5
33	Forest Edge	133 Oak Ln	BRAMPTON	2	3	2	TOWNHOUSE	400000	1600	2012	Yes	Yes	No	Gym, Hospital	Low	1.3
36	Grove Residence	200 Grove Ave	BRAMPTON	1	3	2	CONDO	290000	900	2018	No	No	Yes	Public Transport	High	1.2
38	Cozy Corner	511 Willow St	BRAMPTON	1	2	1	APARTMENT	330000	1100	2019	No	Yes	No	Grocery, Bus Stop	Medium	1.1
40	West End Villa	355 West St	BRAMPTON	2	5	4	SINGLE_FAMILY	750000	3000	2015	Yes	Yes	Yes	Nature Trails	Low	1.7
43	Eco-Friendly Home	150 Green St	BRAMPTON	1	3	2	SINGLE_FAMILY	460000	1400	2018	Yes	No	Yes	Grocery, Gym	Low	1.3
46	Mountain Retreat	900 Highland Blvd	BRAMPTON	2	4	3	SINGLE_FAMILY	800000	3400	2009	Yes	Yes	Yes	Hiking Trails, Grocery	Low	1.7
48	Countryside Home	123 Country Ln	BRAMPTON	1	3	2	SINGLE_FAMILY	570000	2000	2005	Yes	No	Yes	Nature, Schools	Medium	1.4

OK

Figure 29. Customer Screen on Successful Search Results.

Figure 29 shows the Customer Screen on Successful Search Results, displaying a table of properties that match the customer's search criteria. The results include details such as Name, Address, City, Rooms, Bathrooms, Type, Price, Square Footage, Year Built, and additional attributes like Energy Efficient, Pet Friendly, and Proximity to Amenities. The summary of the search criteria is displayed at the top for easy reference. The OK button allows the customer to exit the results view. This feature provides a clear and organized way for customers to review and select suitable properties.

Search Results

Showing Properties Based on this Input:

- City: Brampton
- Property Type: Single-Family
- Price Range: 700,000 - 900,000
- Rooms: 4 - 7
- Pet Friendly: Yes

Search Results:

No exact results found. Showing recommended properties based on some of your criteria:

ID	Name	Address	City	Floors	Rooms	Bathrooms	Type	Price	Square Footage	Year Built	Energy Efficient	HOA Fees	Pet Friendly	Proximity to Amenities	Safety Rating	Property
4	Cedar Townhouse	321 Cedar Ln	BRAMPTON	2	3	3	TOWNHOUSE	450000	2200	2020	No	No	Yes	Schools, Grocery, Hospital	Medium	1.3
7	Willow Manor	202 Willow Rd	BRAMPTON	3	4	3	SINGLE_FAMILY	670000	2700	2000	Yes	Yes	Yes	Schools, Park	Medium	1.4
10	Aspen Estate	505 Aspen Ave	BRAMPTON	2	5	4	SINGLE_FAMILY	920000	3600	2016	Yes	Yes	Yes	Nature Trails, Schools	Low	1.7
13	Bayview Bungalow	808 Bay Ave	BRAMPTON	1	3	1	SINGLE_FAMILY	280000	1400	1998	No	No	No	Beach, Schools	Medium	1.4
16	Forest Townhouse	202 Forest St	BRAMPTON	2	4	2	TOWNHOUSE	410000	2100	2007	Yes	No	Yes	Schools, Gym	Medium	1.3
19	Sunny Apartment	505 Sun St	BRAMPTON	1	2	1	APARTMENT	330000	1100	2020	No	Yes	No	Grocery, Bus Stop	High	1.1
22	Ocean Breeze	808 Ocean Dr	BRAMPTON	2	3	2	SINGLE_FAMILY	560000	1500	2017	Yes	Yes	Yes	Beach, Schools	Medium	1.3
25	Greenfield Villa	202 Green Ln	BRAMPTON	2	4	3	SINGLE_FAMILY	670000	2500	2011	Yes	No	Yes	Parks, Grocery	Medium	1.3
29	Suburban Home	606 Suburb Ave	BRAMPTON	2	4	3	SINGLE_FAMILY	740000	3000	2013	Yes	No	Yes	Parks, Schools	Medium	1.5
33	Forest Edge	133 Oak Ln	BRAMPTON	2	3	2	TOWNHOUSE	400000	1600	2012	Yes	Yes	No	Gym, Hospital	Low	1.3
36	Grove Residence	200 Grove Ave	BRAMPTON	1	3	2	CONDO	290000	900	2018	No	No	Yes	Public Transport	High	1.2
38	Cozy Corner	511 Willow St	BRAMPTON	1	2	1	APARTMENT	330000	1100	2019	No	Yes	No	Grocery, Bus Stop	Medium	1.1
40	West End Villa	355 West St	BRAMPTON	2	5	4	SINGLE_FAMILY	750000	3000	2015	Yes	Yes	Yes	Nature Trails	Low	1.7
43	Eco-Friendly Home	150 Green St	BRAMPTON	1	3	2	SINGLE_FAMILY	460000	1400	2018	Yes	No	Yes	Grocery, Gym	Low	1.3
46	Mountain Retreat	900 Highland Blvd	BRAMPTON	2	4	3	SINGLE_FAMILY	800000	3400	2009	Yes	Yes	Yes	Hiking Trails, Grocery	Low	1.7
48	Countryside Home	123 Country Ln	BRAMPTON	1	3	2	SINGLE_FAMILY	570000	2000	2005	Yes	No	Yes	Nature, Schools	Medium	1.4

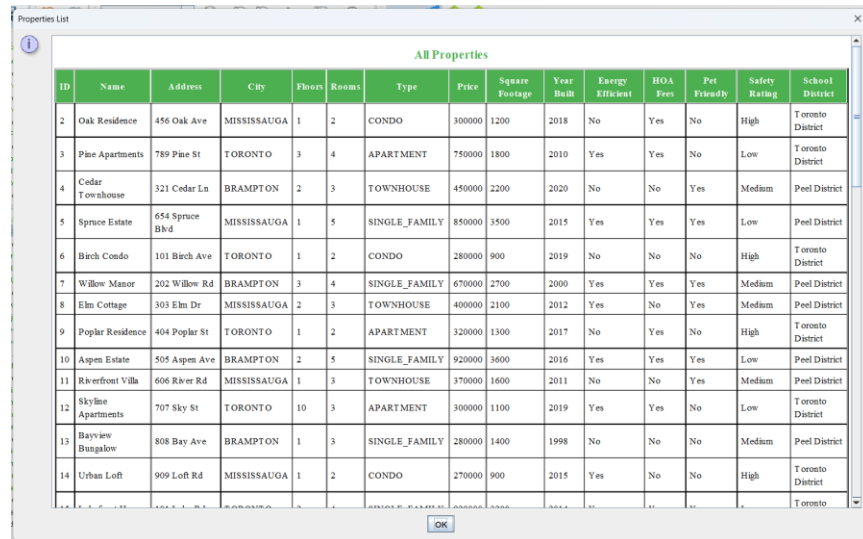
OK

Figure 30. Customer Screen if No Exact Results on Property Search.

Figure 30 shows the Customer Screen if No Exact Results on Property Search, where the system provides recommended properties that closely match some of the customer's input criteria. At the top, a summary of the search inputs is displayed, followed by a message indicating no exact matches were found. Below, a table lists recommended properties with attributes such as Name, Address, City, Rooms, Bathrooms, Type, Price, Square Footage, Year Built, and more. The OK button allows the customer to close the recommendations and return them to the previous menu. This feature ensures customers have alternative options even when their exact preferences are unavailable.

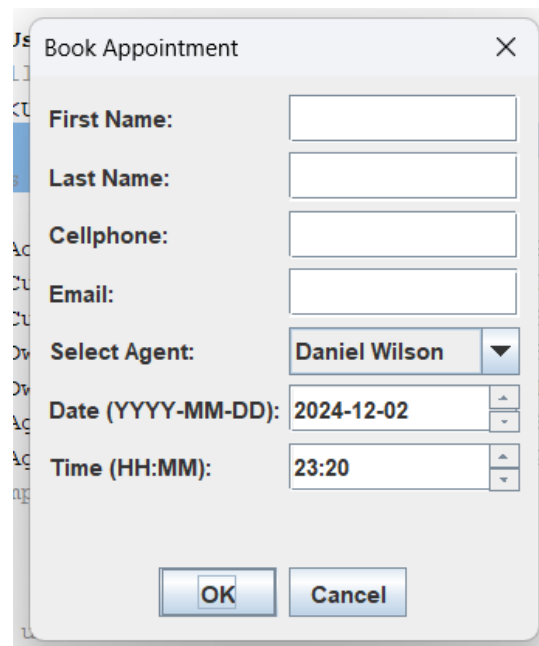
Figure 31 shows the Customer Screen in Viewing All Properties, where customers can browse a complete list of properties available in the system. The table includes details such as ID, Name, Address, City, Rooms, Bathrooms, Type, Price, Square Footage, Year Built, and

additional attributes like Energy Efficient, Pet Friendly, and Safety Rating. The OK button at the bottom allows the customer to exit the view and return to the previous menu. This screen provides customers with an overview of all available properties for exploration.



ID	Name	Address	City	Floors	Rooms	Type	Price	Square Footage	Year Built	Energy Efficient	HOA Fees	Pet Friendly	Safety Rating	School District
2	Oak Residence	456 Oak Ave	MISSISSAUGA	1	2	CONDO	300000	1200	2018	No	Yes	No	High	Toronto District
3	Pine Apartments	789 Pine St	TORONTO	3	4	APARTMENT	750000	1800	2010	Yes	Yes	No	Low	Toronto District
4	Cedar Townhouse	321 Cedar Ln	BRAMPTON	2	3	TOWNHOUSE	450000	2200	2020	No	No	Yes	Medium	Peel District
5	Spruce Estate	654 Spruce Blvd	MISSISSAUGA	1	5	SINGLE_FAMILY	850000	3500	2015	Yes	Yes	Yes	Low	Peel District
6	Birch Condo	101 Birch Ave	TORONTO	1	2	CONDO	280000	900	2019	No	No	No	High	Toronto District
7	Willow Manor	202 Willow Rd	BRAMPTON	3	4	SINGLE_FAMILY	670000	2700	2000	Yes	Yes	Yes	Medium	Peel District
8	Elm Cottage	303 Elm Dr	MISSISSAUGA	2	3	TOWNHOUSE	400000	2100	2012	Yes	No	Yes	Medium	Peel District
9	Poplar Residence	404 Poplar St	TORONTO	1	2	APARTMENT	320000	1300	2017	No	Yes	No	High	Toronto District
10	Aspen Estate	505 Aspen Ave	BRAMPTON	2	5	SINGLE_FAMILY	920000	3600	2016	Yes	Yes	Yes	Low	Peel District
11	Riverfront Villa	606 River Rd	MISSISSAUGA	1	3	TOWNHOUSE	370000	1600	2011	No	No	Yes	Medium	Peel District
12	Skyline Apartments	707 Sky St	TORONTO	10	3	APARTMENT	300000	1100	2019	Yes	Yes	No	Low	Toronto District
13	Bayview Bungalow	808 Bay Ave	BRAMPTON	1	3	SINGLE_FAMILY	280000	1400	1998	No	No	No	Medium	Peel District
14	Urban Loft	909 Loft Rd	MISSISSAUGA	1	2	CONDO	270000	900	2015	Yes	No	No	High	Toronto District

Figure 31. Customer Screen in Viewing All Properties.



Book Appointment

✕

First Name:

Last Name:

Cellphone:

Email:

Select Agent:

Daniel Wilson ▾

Date (YYYY-MM-DD):

2024-12-02

⬆ ⬇ ⬆

Time (HH:MM):

23:20

⬆ ⬇ ⬆

OK

Cancel

Figure 32. Customer Booking Screen.

Figure 32 shows the Customer Booking Screen, where customers can schedule an appointment with an agent. The form includes fields for First Name, Last Name, Cellphone, Email, and a dropdown menu to Select Agent. Additionally, customers can specify the Date and Time for the appointment using input fields. The OK button submits the booking request, while the Cancel button allows users to exit the form without scheduling. This feature facilitates seamless communication and appointment scheduling between customers and agents.

Book Appointment

First Name:

Last Name:

Cellphone:

Email:

Select Agent: Daniel Wilson ▼

Date (YYYY-MM-DD):

Time (HH:MM):

OK Cancel

Figure 33. Customer Screen with Agents Dropdown.

Figure 33. shows the Customer Screen with Agents Dropdown. This figure illustrates the customer booking screen where customers can select an agent from the Agents Dropdown. The dropdown menu lists available agents, such as Daniel Wilson and Emma Taylor, allowing

customers to choose their preferred agent for the appointment. This feature ensures flexibility in selecting an agent based on availability or preference while booking appointments.

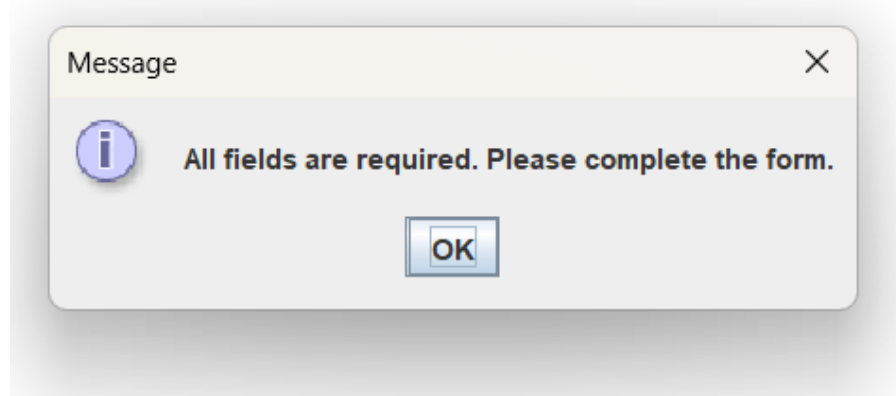


Figure 34. Warning Prompt of Invalid Inputs in Booking

In Figure 34, the warning prompt is displayed when a customer tries to book an appointment without completing all the required fields in the form. The message, "All fields are required. Please complete the form," ensures that users are reminded to provide all necessary information before proceeding. This feature helps enforce data validation and maintain the accuracy and integrity of the booking process.

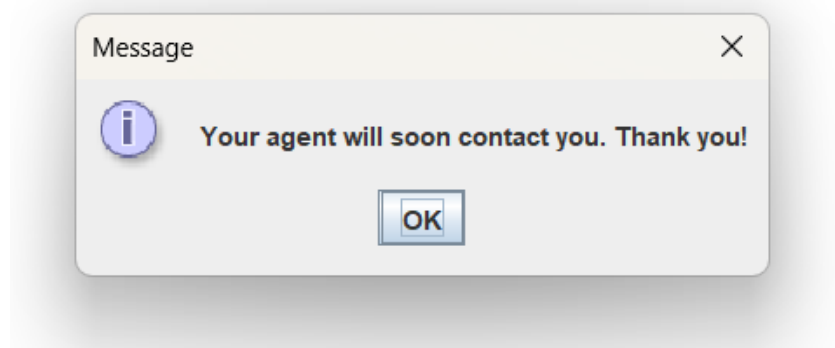


Figure 35. Customer Booking Successful.

In Figure 35, the confirmation prompt displays a success message indicating that the customer booking has been completed. The message, "Your agent will soon contact you. Thank you!" assures the user that their booking was processed, and the assigned agent will follow up regarding the appointment. This screen confirms that all booking inputs were valid and submitted successfully, providing users with confidence in the system's functionality.

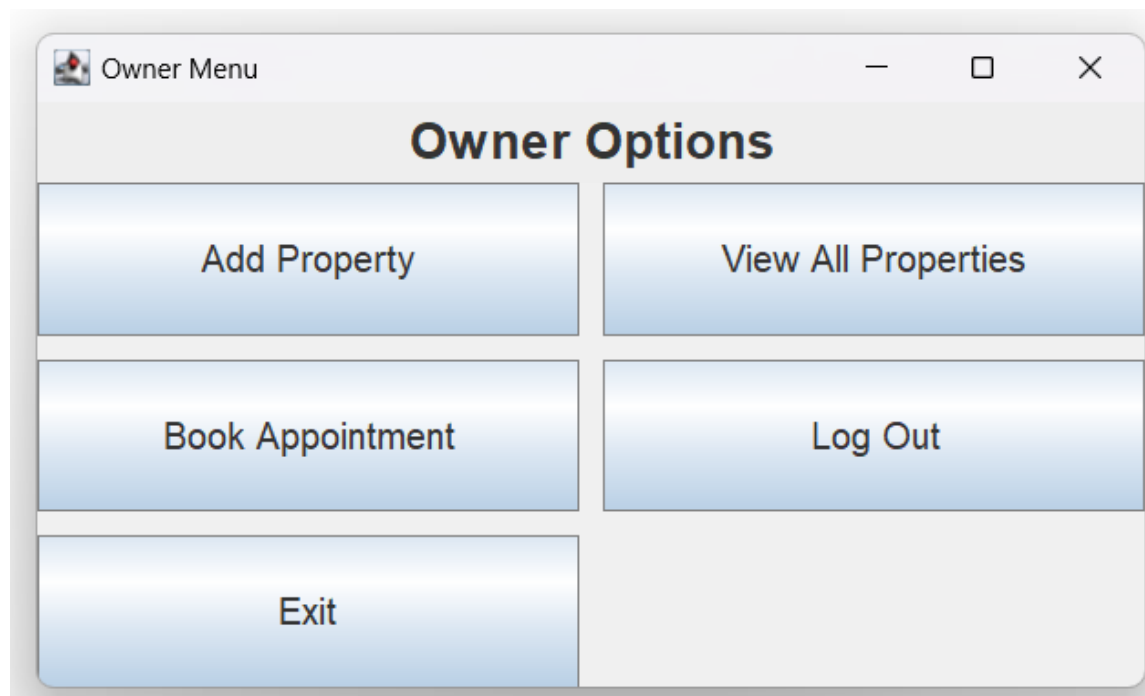


Figure 36. Owner Page

In Figure 36, the Owner Page is presented, providing a tailored menu for property owners. This screen includes options to add properties for admin approval, view all properties listed in the system, book appointments with agents, log out of the session, or exit the application. Each button offers owners a streamlined way to manage their interactions with the system while ensuring they can perform key actions efficiently. This page ensures that owners have access to tools aligned with their role in the application.

Add Property for Approval

Name: e.g., Maple Villa

Address: e.g., 123 Maple St

City: ANY

Floors: e.g., 2

Rooms: e.g., 3

Bathrooms: e.g., 2

Type: ANY

Square Footage: e.g., 1500

Year Built: e.g., 2010

Price: e.g., 300000

Energy Efficient: ☐

HOA Fees: ☐

Pet Friendly: ☐

Proximity to Amenities: e.g., Near schools

Safety Rating: e.g., High

Property Tax Rate: e.g., 1.2

School District: e.g., Peel District

Owner First Name: Michael

Owner Last Name: Brown

OK Cancel

Figure 37. Owner Screen on Adding Pending Properties.

In Figure 37, the Owner Screen for Adding Pending Properties is displayed. This screen allows property owners to input details about their properties, such as the name, address, city, number of floors, rooms, bathrooms, property type, square footage, year built, price, and various additional attributes like energy efficiency and pet-friendliness. The form also includes fields for proximity to amenities, safety rating, property tax rate, school district, and the owner's name.

Upon submission, the property is added to a pending list for admin approval, ensuring only validated entries are published in the system.

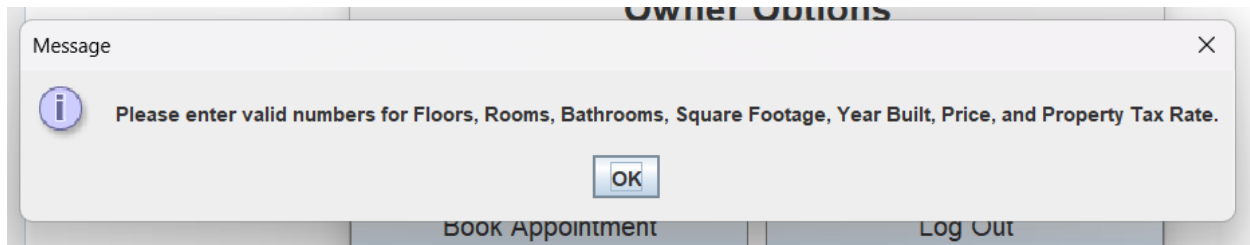


Figure 38. Warning Prompt of Invalid Inputs for Adding Pending Properties.

In Figure 38, the Warning Prompt for Invalid Inputs while Adding Pending Properties is displayed. This prompt appears when an owner attempts to submit the form with invalid or non-numeric values in critical fields, such as floors, rooms, bathrooms, square footage, year built, price, or property tax rate. This validation ensures data accuracy and consistency before submission, preventing errors in the property database.

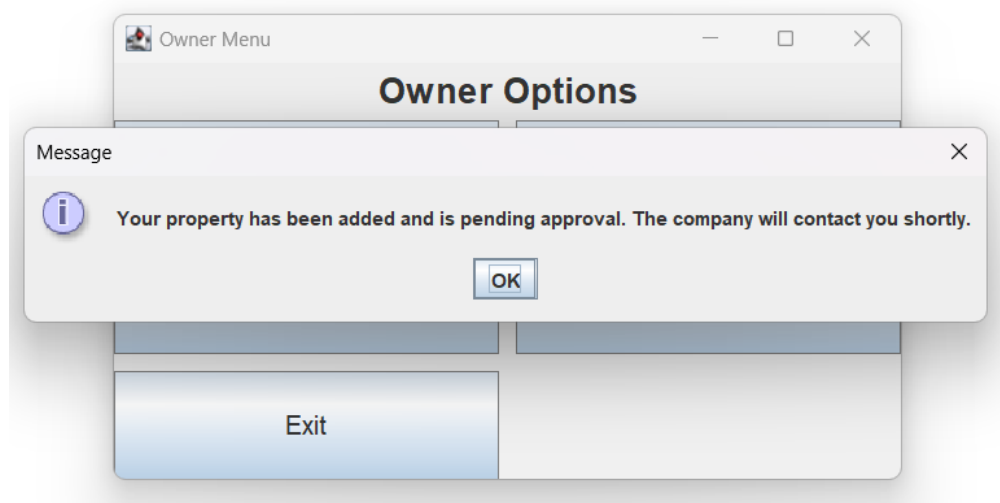
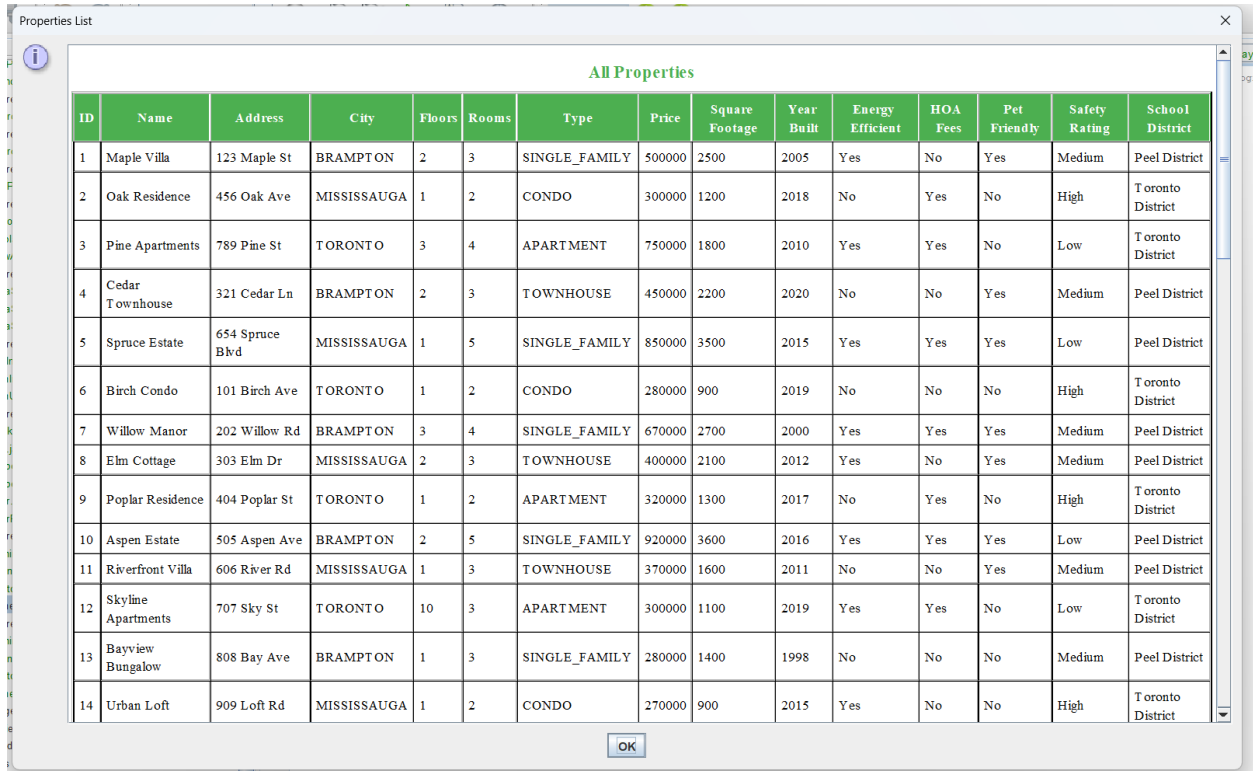


Figure 39. Owner Screen Successfully Added Pending Property.

In Figure 39, the Owner Screen confirms the successful addition of pending property. This message notifies the owner that their property has been submitted and is awaiting approval from the administrator. The prompt assures the owner that the company will contact them regarding the approval status, completing the property submission process.



All Properties														
ID	Name	Address	City	Floors	Rooms	Type	Price	Square Footage	Year Built	Energy Efficient	HOA Fees	Pet Friendly	Safety Rating	School District
1	Maple Villa	123 Maple St	BRAMPTON	2	3	SINGLE_FAMILY	500000	2500	2005	Yes	No	Yes	Medium	Peel District
2	Oak Residence	456 Oak Ave	MISSISSAUGA	1	2	CONDO	300000	1200	2018	No	Yes	No	High	Toronto District
3	Pine Apartments	789 Pine St	TORONTO	3	4	APARTMENT	750000	1800	2010	Yes	Yes	No	Low	Toronto District
4	Cedar Townhouse	321 Cedar Ln	BRAMPTON	2	3	TOWNHOUSE	450000	2200	2020	No	No	Yes	Medium	Peel District
5	Spruce Estate	654 Spruce Blvd	MISSISSAUGA	1	5	SINGLE_FAMILY	850000	3500	2015	Yes	Yes	Yes	Low	Peel District
6	Birch Condo	101 Birch Ave	TORONTO	1	2	CONDO	280000	900	2019	No	No	No	High	Toronto District
7	Willow Manor	202 Willow Rd	BRAMPTON	3	4	SINGLE_FAMILY	670000	2700	2000	Yes	Yes	Yes	Medium	Peel District
8	Elm Cottage	303 Elm Dr	MISSISSAUGA	2	3	TOWNHOUSE	400000	2100	2012	Yes	No	Yes	Medium	Peel District
9	Poplar Residence	404 Poplar St	TORONTO	1	2	APARTMENT	320000	1300	2017	No	Yes	No	High	Toronto District
10	Aspen Estate	505 Aspen Ave	BRAMPTON	2	5	SINGLE_FAMILY	920000	3600	2016	Yes	Yes	Yes	Low	Peel District
11	Riverfront Villa	606 River Rd	MISSISSAUGA	1	3	TOWNHOUSE	370000	1600	2011	No	No	Yes	Medium	Peel District
12	Skyline Apartments	707 Sky St	TORONTO	10	3	APARTMENT	300000	1100	2019	Yes	Yes	No	Low	Toronto District
13	Bayview Bungalow	808 Bay Ave	BRAMPTON	1	3	SINGLE_FAMILY	280000	1400	1998	No	No	No	Medium	Peel District
14	Urban Loft	909 Loft Rd	MISSISSAUGA	1	2	CONDO	270000	900	2015	Yes	No	No	High	Toronto District

Figure 40. Owner Screen on Viewing All Properties.

In Figure 40, the Owner Screen displays a list of all properties available in the system. The table provides comprehensive details for each property, including the name, address, city, number of floors and rooms, type, price, and additional attributes like energy efficiency and proximity to amenities. This feature allows owners to view the complete set of properties, aiding in better management and comparison.

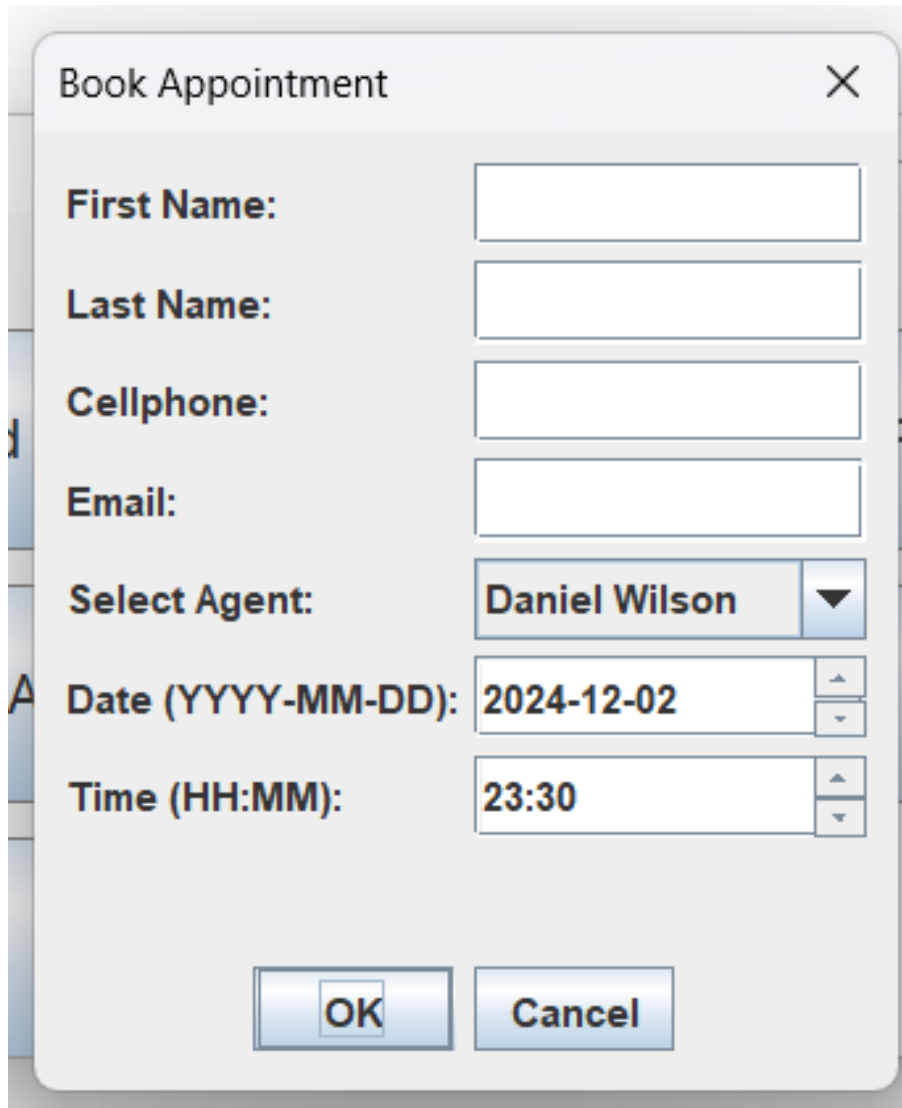
A screenshot of a 'Book Appointment' dialog box. The dialog has a title bar with the text 'Book Appointment' and a close button (X) on the right. Inside the dialog, there are several input fields: 'First Name:', 'Last Name:', 'Cellphone:', and 'Email:', each followed by a text input box. Below these is a 'Select Agent:' label followed by a dropdown menu showing 'Daniel Wilson' with a downward arrow. Then, 'Date (YYYY-MM-DD):' is followed by a date input box showing '2024-12-02' and a small calendar icon. Finally, 'Time (HH:MM):' is followed by a time input box showing '23:30' and a small clock icon. At the bottom of the dialog are two buttons: 'OK' and 'Cancel'.

Figure 41. Owner Screen for Booking an Appointment.

In Figure 41, the Owner Screen for Booking an Appointment is displayed. This interface allows property owners to schedule appointments with selected agents. The form requires input for the owner's first and last name, cellphone number, email, the agent they wish to meet, as well as the date and time of the appointment. This feature ensures an organized way for owners to connect with agents for property-related discussions.

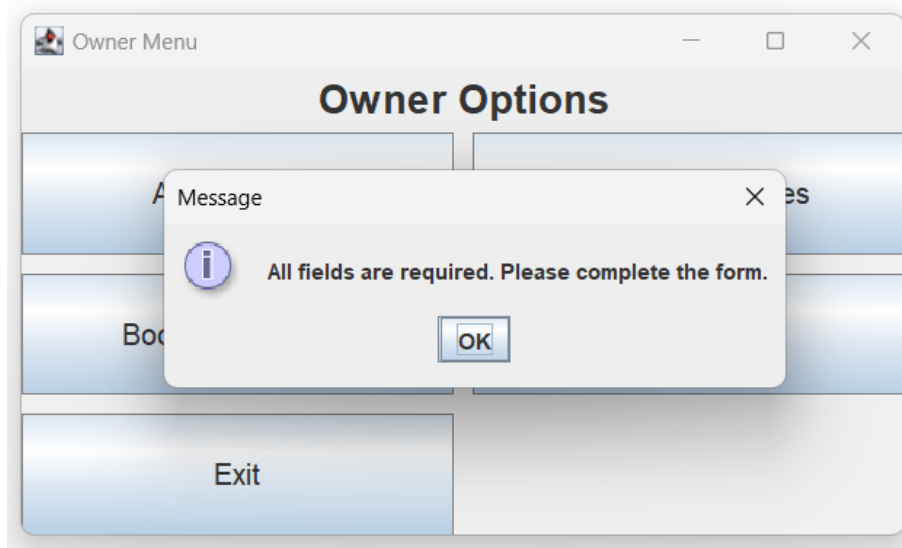


Figure 42. Owner's Warning Prompt on Invalid Inputs on Booking Appointment.

In Figure 42, the Owner's Warning Prompt for Invalid Inputs on Booking Appointment is displayed. This message alerts the owner that all fields in the booking form are required and must be completed before proceeding. This ensures the submission of complete and accurate information for successful appointment scheduling.

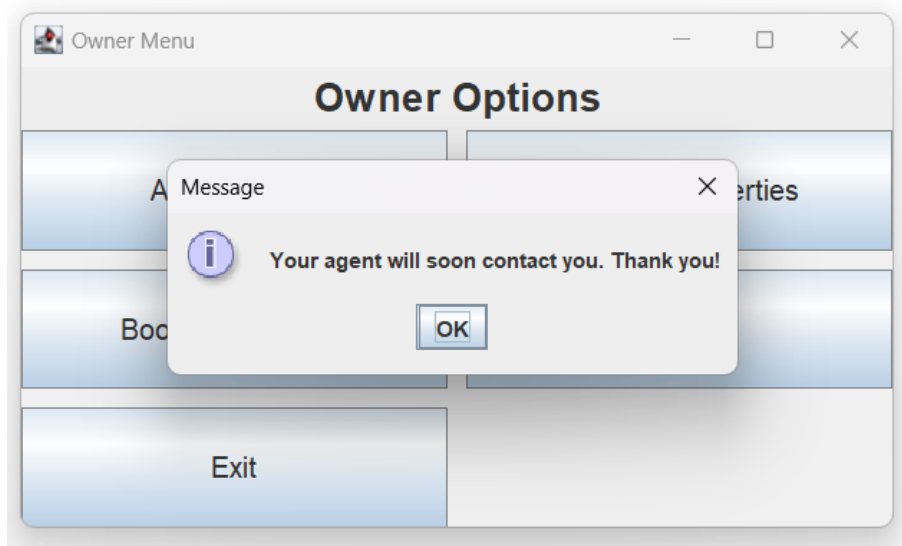


Figure 43. Booking Successful for Owner

In Figure 43, the Owner sees a confirmation prompt indicating a successful booking. The message informs the owner that their selected agent will contact them shortly, providing reassurance that the appointment has been scheduled and action will follow.

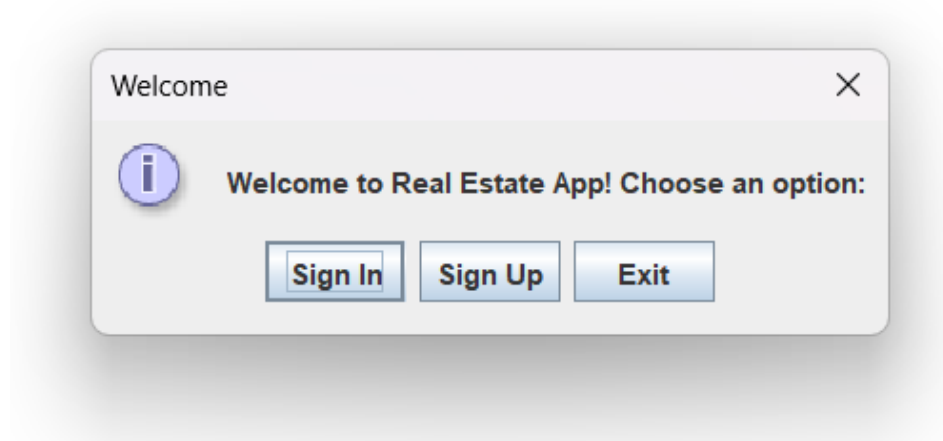


Figure 44. Going Back to Welcome Screen After Clicking Log Out.

In Figure 44, the application returns to the Welcome Screen after the user clicks the "Log Out" button. This screen provides the user with options to either sign in again, sign up as a new user, or exit the application, ensuring smooth navigation back to the starting point.

The feasibility of the Real Estate App project is strong, given its clear objectives and practical implementation. The project is technically viable, leveraging a structured design and tools like NetBeans and UML diagrams to ensure maintainability and scalability. It is economically feasible, as it focuses on efficient resource utilization and does not require extensive external dependencies. The application also meets user needs effectively by offering intuitive features such as property management, appointment booking, and role-based access, making it a viable solution for real estate interactions.

Project Summary and Conclusion

The Real Estate App successfully fulfills its intended purpose of providing a streamlined system for property management, user interactions, and appointment booking. It allows admins, agents, customers, and owners to interact through user-friendly interfaces tailored to their roles. The project integrates essential features like property searches, bookings, and property approval processes, all while maintaining clear validation and feedback mechanisms.

In conclusion, the Real Estate App achieves its goal of delivering a comprehensive platform for managing real estate activities. Admins can efficiently manage properties, approve pending submissions, and oversee appointments, ensuring smooth operations. Agents benefit from the ability to view and approve bookings, maintaining streamlined communication with customers and owners. Customers and owners are provided with intuitive tools to search for properties, book appointments, and manage their interactions seamlessly.

The project not only meets its stated objectives but also exceeds expectations in areas such as validation and error handling. Thorough testing ensured that user actions are validated, preventing errors and enhancing overall user experience.

Overall, the Real Estate App demonstrates its feasibility and reliability as a solution for real estate management. Its design prioritizes user experience, modularity, and scalability, making it a valuable tool for real estate businesses. The project's success underscores its ability to meet real-world requirements while providing a strong foundation for future enhancements and additional features.

References

McLaughlin, B., Pollice, G., & West, D. (2006). *Head First Object-Oriented Analysis and Design*.

O'Reilly Media, Inc.

<https://learning.oreilly.com/library/view/head-first-object-oriented/0596008678/?ar>