PROPERTY RENTAL MANAGEMENT SYSTEM

Course Number: 420-DW4-AS - Course Title: Web Server Applications Development II - Teacher: Quang Hoang Cao - Session: Autumn 2024 - Group: 7278

OCTOBER 29, 2024

VIRAJ KETAN PATEL

Contents

Project Description

The **Property Rental Management System (PRMS)** is a web-based application designed to streamline the management of rental properties. The system allows property owners, managers, and tenants to efficiently handle property listings, rental agreements, maintenance requests, and payments.

Project Development

Phase I Analysis

1. Software Requirements

To develop the Property Rental Management System (PRMS) following technologies and tools used:

- **Visual Studio 2022:** An IDE for C# and ASP.NET Core, enhancing coding efficiency and debugging.
- **SQL Server 2022:** Provides robust data storage and management features, ensuring data integrity.
- **SQL Server Management Studio/ Azure Data Studio:** Offers effective tools for database management.
- **ASP.Net Core MVC:** A flexible framework using MVC architecture for scalable web applications and clean code separation.
- **Entity Framework:** Streamlines data operations with seamless integration through an object-oriented approach.
- **Bootstrap:** Ensures responsive design and a visually appealing user interface across various devices.
- **jQuery:** Enhances user interactivity through simplified DOM manipulation and efficient event handling.

2. Functional Requirements

Who are the users of this web app?

- Property Owner
- Property Managers
- Potential Tenants

User Operations:

Users	Operations
Property Owner	• Create/Update/Delete/Search/List any property manager account. • Update/Delete/Search/List any potential tenant account. • Access the reports for building property that was reported by property managers.
Property Managers	• Perform CRUD operations related to buildings. • Perform CRUD operations related to apartments. • Keep track of apartments status. • Schedule potential tenants ‘appointments. • Respond to potential tenants ‘messages. • Report any events to the property owner when necessary.
Potential Tenants	• Create an on-line account through Property Rental Management Web Site. • View any apartment suitable for their needs. • Make an appointment with the property manager. • Send necessary messages to the property manager.

3. Non-Functional Requirements

1) Security:

- User Authentication: The system implements user authentication to verify user identities.
- Authorization: The system ensure that only authorized users can perform specific actions, protecting sensitive data.

Phase II Design

Database Schema

Database Name: **PropertyRentalManagementDB**

Tables:

1) Status:

Fields	**Data Types**	**Design Notes**	**Example**
StatusID (PK)	Int	Auto Generated	1
Description	String	Status Description	Pending, Confirmed ,etc.

Category	String	Status’ Category	Appointments

Examples:

StatusID	**Description**	**Category**
1	Available	Apartments
2	Rented	Apartments
3	Unavailable	Apartments
4	Pending	Appointments
5	Confirmed	Appointments
6	Canceled	Appointments
7	Read	Messages
8	Not Read	Messages
9	New	Events
10	Pending	Events
11	Solved	Events

2) UserRoles:

Fields	**Data Types**	**Design Notes**	**Example**
RoleID (PK)	Int	Auto Generated	1
Role	String	User role in web app	Owner, Manager, Tenant

Examples:

RoleID	Role
1	Owner
2	Manager
3	Tenant

3) Users:

Fields	**Data Types**	**Design Notes**	**Example**
UserID (PK)	Int	Auto Generated	1111111
Username	String	Unique	virajpatel
FirstName	String		Viraj
LastName	String		Viraj
Email	String	Email	viraj@gmail.com
PhoneNumber	String	In format XXX-XXX-XXXX	514-123-1235
Password	String	Min 8 character atleast one number, upper letter, lower letter and special character	Vir@j123

4) Buildings:

Fields	**Data Types**	**Design Notes**	**Example**
BuildingID (PK)	String	In format BULXXXX	BUL0001
OwnerID (FK)	Int	FK Users	1111111
ManagerID (FK)	Int	FK Users	1111112
Name	String	Building Name	King's Building
Description	String		This is largest building.
Address	String		1234 - Atwater
City	String		Montreal
Province	String		Quebec
ZipCode	String	In format A1A-1A1	A1A-1A1

5) ApartmentTypes:

Fields	**Data Types**	**Design Notes**	**Example**
ApartmentTypesID (PK)	Int	Auto Generated	1
ApartmentTypeDescription	String		Studio, Loft

Examples:

ApartmentTypeID	ApartmentTypeDescription
1	Studio
2	1 Bedroom
3	2 Bedroom
4	3 Bedroom
5	Loft
6	Penthouse

6) Apartments:

Fields	**Data Types**	**Design Notes**	**Example**
ApartmentID (PK)	Int	Auto Generated	1
ApartmentCode	String		A-314
BuildingCode (FK)	Int	FK Building	BUL0001
ApartmentTypeID(FK)	Int	FK ApartmentTypeID	1
Description	String		This is nice apartment.
Rent	Decimal	Currency	1000.00
StatusID	Int	FK Status (Only 1,2,3)	1, 2, 3

7) Appointments:

Fields	Data Types	Design Notes	Example
AppointmentID(PK)	Int	Auto Generated	1
TenantID(FK)	Int	FK Users	1111113
ManagerID(FK)	Int	FK Users	1111112
ApartmentID(FK)	Int	FK Apartments	1
AppointmentDateTime	DateTime		2024-10-22 12:00 PM.
Description	String		Example Description
StatusID (FK)	Int	FK Status (Only 4,5,6)	4,5,6

8) Messages:

Fields	Data Types	Design Notes	Example
MessageID(PK)	Int	Auto Generated	1
SenderUserID(FK)	Int	FK Users	1111113
ReceiverUserID(FK)	Int	FK Users	1111112
Subject	String		Message Subject.
MessageBody	String		Message Body.
MessageDateTime	DateTime		2024-10-22 12:00 PM
StatusID	Int	FK Status (Only 7,8)	7, 8

9) Events:

Fields	Data Types	Design Notes	Example
EventID	Int	Auto Generated	1
ManagerID	Int	FK Users	1111112
ApartmentID	Int	FK Apartments	1
Description	String		
EventDate	Date		2024-10-22
StatusID	Int	FK Status (Only 9,10,11)	9,10,11

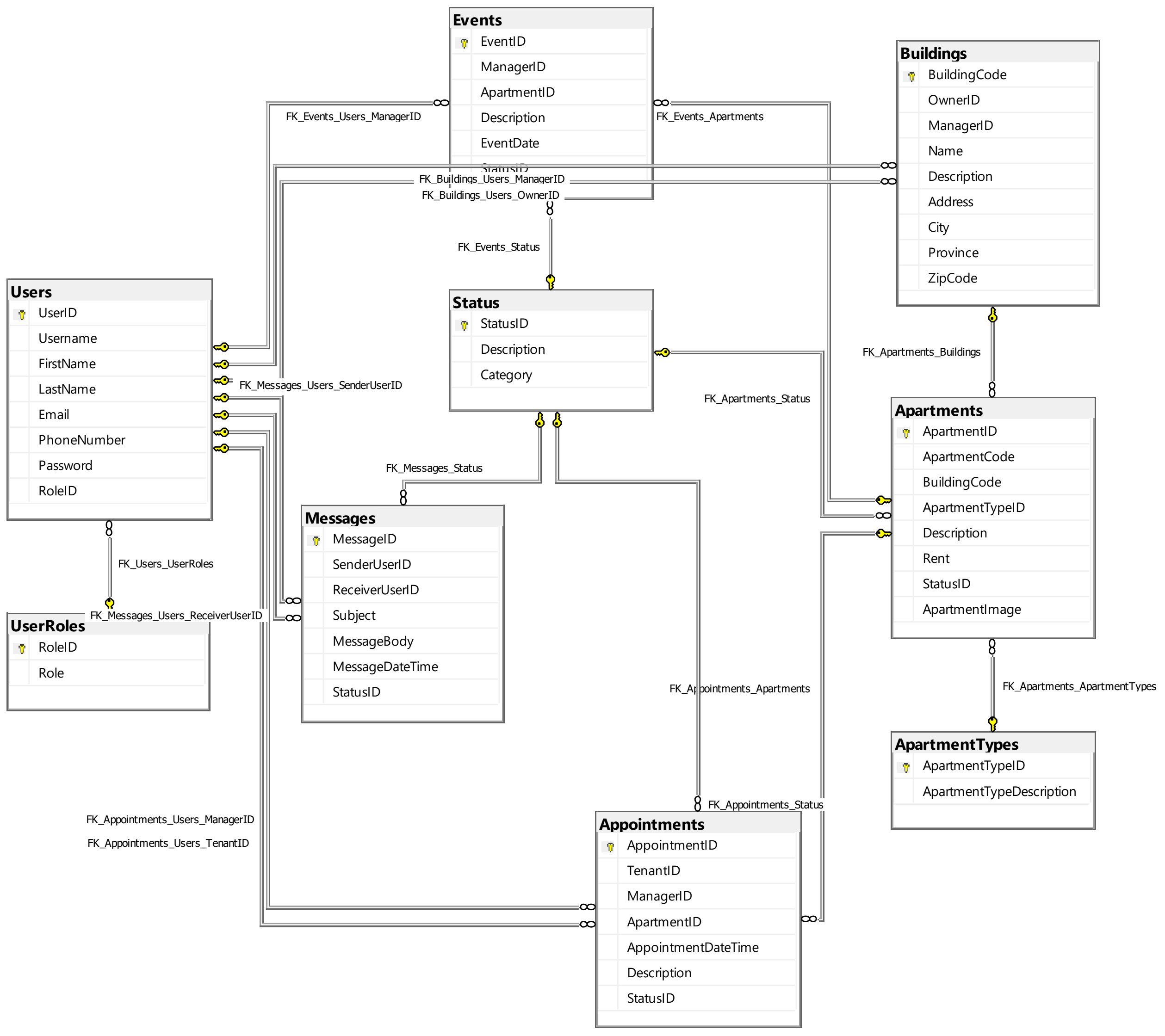
Relationships:

One-to-Many Relationships:

- **UserRoles to Users (FK_Users_UserRoles):** One user role can be assigned to multiple users.
- **Users to Buildings (FK_Buildings_Users_OwnerID):** One user (owner) can own multiple buildings.
- **Users to Buildings (FK_Buildings_Users_ManagerID):** One user (manager) can manage multiple buildings.
- **ApartmentTypes to Apartments (FK_Apartments_ApartmentTypes):** One apartment type can be assigned to multiple apartments.

- **Status to Apartments (FK_Apartments_Status):** One status can be assigned to multiple apartments.
- **Buildings to Apartments (FK_Apartments_Buildings):** One building can contain multiple apartments.
- **Users to Appointments (FK_Appointments_Users_ManagerID):** One user (manager) can have multiple appointments.
- **Users to Appointments (FK_Appointments_Users_TenantID):** One user (tenant) can have multiple appointments.
- **Apartments to Appointments (FK_Appointments_Apartments):** One apartment can have multiple appointment requests.
- **Status to Appointments (FK_Appointments_Status):** One status can be assigned to multiple appointments.
- **Users to Messages (FK_Messages_Users_SenderUserID):** One user can send multiple messages.
- **Users to Messages (FK_Messages_Users_ReceiverUserID):** One user can receive multiple messages.
- **Status to Messages (FK_Messages_Status):** One status can be assigned to multiple messages.
- **Users to Events (FK_Events_Users_ManagerID):** One user (manager) can manage multiple events.
- **Status to Events (FK_Events_Status):** One status can be assigned to multiple events.
- **Apartments to Events (FK_Events_Apartments):** One apartment can have multiple events.

Relationship Diagram:



Phase III Implementation

1. **Creation of the Database:** Established the database in SQL Server Management Studio.

2. **Creation of ASP.NET Core Web App MVC:** Developed a web application using ASP.NET Core MVC in Visual Studio 2022 to facilitate a structured approach to building the application.

3. **Installation of Necessary Packages:** Installed essential NuGet packages to enable Entity Framework Core functionalities:

- ***Microsoft.EntityFrameworkCore:*** Core package for the Entity Framework, facilitating data access.

- ***Microsoft.EntityFrameworkCore.SqlServer:*** Provides SQL Server database provider for Entity Framework Core.
- ***Microsoft.EntityFrameworkCore.Tools:*** Tools for Entity Framework Core that assist in database migration and model generation.
- ***Microsoft.EntityFrameworkCore.Proxies:*** Allows for lazy loading of related entities in Entity Framework Core.

4. **Model Creation from Database:** Utilized the Scaffold-DbContext command in the Package Manager Console to generate models based on the existing database schema. Command used:

`Scaffold-DbContext "Server=LAPTOP-4072SUH7\SQL2022EXPRESS;Initial Catalog=PropertyRentalManagementDB;User=sa;Password=123456;Integrated Security=True;TrustServerCertificate=True;" Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models -Force`

5. **Configuration of Connection String:** Moved the database connection string to the appsettings.json file and updated Program.cs to register the DbContext, enhancing maintainability and security.

6. **Creation of Controllers and Views:** Developed the necessary controllers and views to facilitate interaction between the user interface and the application's data, ensuring a seamless user experience.

Login Authentication/Authorization

The application implements robust login authentication and authorization using cookie-based authentication in ASP.NET Core MVC. The following configuration is applied:

```
using Microsoft.EntityFrameworkCore;

using Microsoft.AspNetCore.Authentication.Cookies;


var builder = WebApplication.CreateBuilder(args);

var connection =
builder.Configuration.GetConnectionString("PropertyRentalManagementDB");
builder.Services.AddDbContext<PropertyRentalManagement.Models.PropertyRentalMana
gementDbContext>(options =>
options.UseLazyLoadingProxies().UseSqlServer(connection));

// Add services to the container.

builder.Services.AddControllersWithViews();


builder.Services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme
).AddCookie(option =>

  {

      option.LoginPath = "/Access/Login";

      option.LogoutPath = "/Home/Index";

      option.AccessDeniedPath = "/Access/Denied"; // Redirects unauthorized users to the
Denied page

      option.ExpireTimeSpan = TimeSpan.FromMinutes(20);

  });

var app = builder.Build();


// Configure the HTTP request pipeline.

if (!app.Environment.IsDevelopment())

{

  app.UseExceptionHandler("/Home/Error");

  // The default HSTS value is 30 days. You may want to change this for production
scenarios, see https://aka.ms/aspnetcore-hsts.

  app.UseHsts();

}


app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseRouting();

app.UseAuthentication();

app.UseAuthorization();


app.MapControllerRoute(

   name: "default",

   pattern: "{controller=Home}/{action=Index}/{id?}");



// Custom 404 handling

app.UseStatusCodePagesWithReExecute("/Home/NotFound");



app.Run();
```

This code sets up cookie authentication with a designated login path (/Access/Login) and an expiration time of 20 minutes. Authentication is enabled in the application pipeline, and a default route is established to direct requests to the **Home** controller's **Index** action.

The **Access** controller is responsible for managing user authentication-related actions, including login and signup processes. The Login action checks user authentication status, redirecting already authenticated users to the home page. The SignUp action validates user input – first name, last name, username, password, email, and phone number - and saves new users to the database if all criteria are met.

In the **Login** action, user credentials are validated against the database. Upon successful authentication, the user is signed in using cookie-based authentication, with user information stored in claims. If authentication fails, appropriate error messages are displayed. The controller leverages Entity Framework Core for database operations and incorporates basic input validation using regular expressions to enhance security and data integrity.

The **AccessDeniedPath** option specifies the URL to redirect users who attempt to access protected resources without sufficient permissions. This ensures that unauthorized users are guided to a designated denial page (/Access/Denied), enhancing the application's security and user experience.

This implementation ensures a secure and user-friendly authentication process, facilitating access to the Property Rental Management System.

Models

1. **Apartment:** Represents rental apartments with details like type, rent, and availability status.
2. **ApartmentType:** Defines categories of apartments such as studio, one-bedroom, loft, etc.
3. **Appointment:** Tracks meetings between tenants and managers for apartment viewings.
4. **Building:** Stores information about rental properties, including location and ownership.
5. **Event:** Logs incidents or events related to property management and maintenance.
6. **Message:** Manages communication between tenants and managers via a messaging system.
7. **Status:** Defines the current state of apartments, appointments, messages, and events.
8. **User:** Captures personal and account details of system users, including tenants and managers.
9. **UserRole:** Identifies the role of each user in the system, such as Owner, Manager, or Tenant.
10. **VMLogin:** ViewModel used for processing user login credentials and authentication.

Controllers

1. **AccessControllers:** Manages user login, registration, and authorization for the system.
2. **ApartmentsController:** Handles apartment-related operations, including adding, updating, deleting, and listing apartments.
3. **AppointmentsController:** Manages scheduling and handling of tenant-manager appointments.
4. **AvailableApartmentsController:** Lists available apartments for potential tenants to view and book appointments.

5. **BuildingsController:** Facilitates CRUD operations for managing property buildings in the system.
6. **DashboardController:** Provides role-based dashboards for managers, owners, and tenants.
7. **EventsController:** Handles the management of apartment-related events reported by managers to owners.
8. **HomeController:** Manages navigation for the home page and general application pages.
9. **ManagersController:** Manages property manager profiles and assigns them to buildings.
10. **MessagesController:** Oversees tenant-manager communication via a messaging platform.
11. **TenantsController:** Handles tenants-related operations, including updating, deleting, and listing tenants.

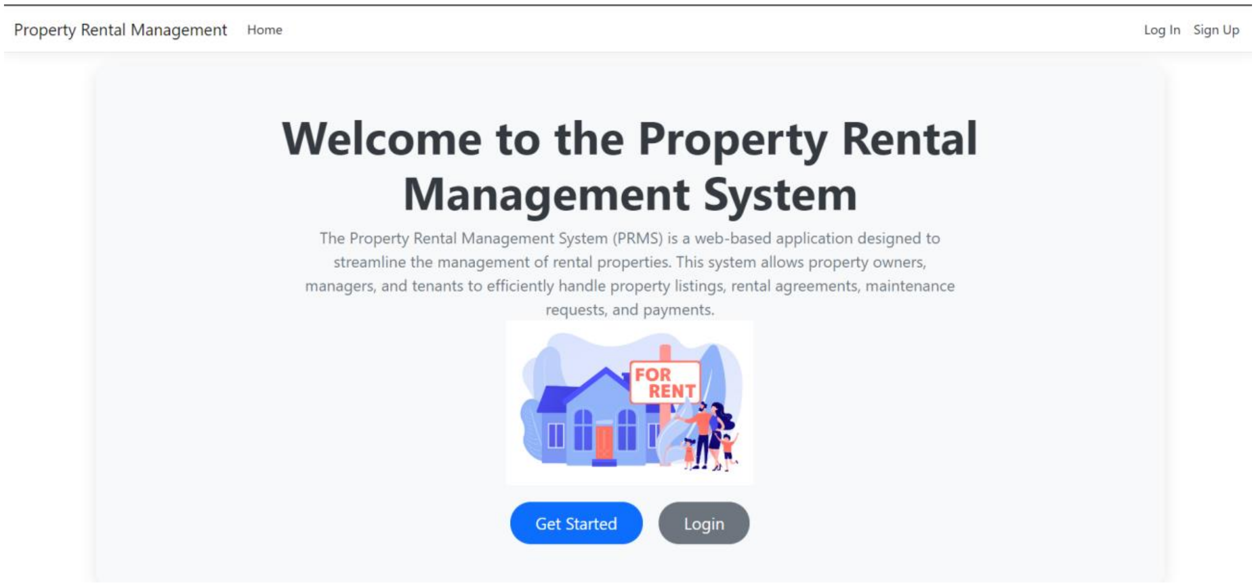
Views

1. **Access:** Provides pages for user login, registration, and access control.
    a. Denied: Displays an access denied message for unauthorized users.
    b. Login: Presents the login page for users to authenticate themselves.
    c. SignUp: Allows new potential tenants users to register for an account.
2. **Apartments:** Displays and manages apartment-related information and actions.
    a. Create: Form for adding a new apartment to the system with image.
    b. Delete: Confirms and executes the deletion of an apartment.
    c. Details: Displays detailed information about a specific apartment with image.
    d. Edit: Form for editing an existing apartment's information.
    e. Index: Lists all apartments managed by the user.
3. **Appointments:** Pages for managing tenant-manager appointments.
    a. Create: Form to schedule a new appointment.
    b. Details: Shows details of a specific appointment.
    c. Index: Lists all appointments.
4. **AvailableApartments:** Displays available apartments for tenants to browse.
    a. Details: Provides detailed information on a specific available apartment with apartment image.
    b. Index: Lists all available apartments for potential tenants to view.
5. **Buildings:** Pages for managing property buildings.
    a. Create: Form for adding a new building.
    b. Delete: Confirms and deletes a building.
    c. Details: Displays detailed information about a building.

   d. Edit: Form to modify building details.
   e. Index: Lists all buildings managed by the user.
6. **Dashboard:** Role-specific dashboard pages for users.
   a. Manager: Dashboard showing metrics and tasks for property managers.
   b. Owner: Dashboard overview for property owners.
   c. Tenant: Dashboard displaying relevant information for tenants.
7. **Events:** Handles apartment-related events in the system.
   a. Create: Form for reporting or adding a new event.
   b. Delete: Confirms and deletes an event.
   c. Details: Displays details of a specific event.
   d. Edit: Form for editing an event's information.
   e. Index: Lists all events in the system.
8. **Home:** General application pages accessible to all users.
   a. Index: The landing page of the application, welcoming unregister users.
   b. Privacy: Displays the privacy policy of the application.
9. **Managers:** Pages for managing property manager accounts and assignments.
   a. Create: Form for adding a new property manager.
   b. Delete: Confirms and deletes a property manager.
   c. Details: Shows details of a specific property manager.
   d. Edit: Form for editing property manager details.
   e. Index: Lists all property managers in the system.
10. **Messages:** Displays and manages messages between tenants and managers.
   a. Create: Form to compose a new message.
   b. Details: Displays the content of a specific message.
   c. Index: Lists all messages in a user's inbox.
11. **Tenants**: Manages potential tenant profiles and related operations.
   a. Delete: Confirms and deletes a potential tenant's profile.
   b. Details: Shows detailed information about a specific potential tenant.
   c. Edit: Form for editing a potential tenant's profile.
   d. Index: Lists all potential tenants in the system.

Pages Design

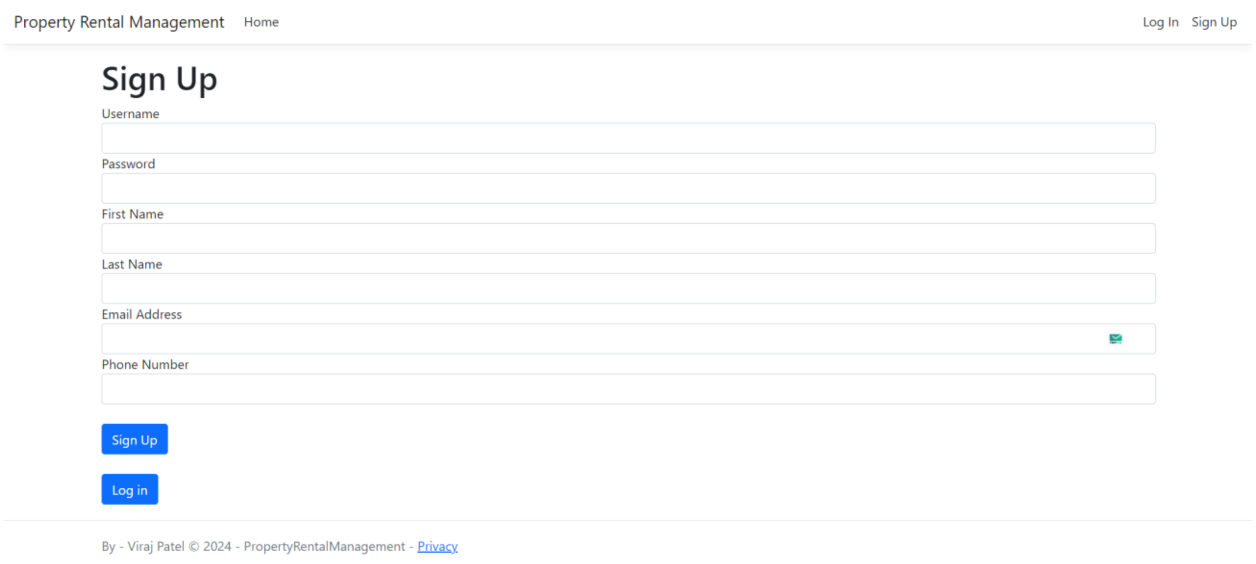
1. Home Page:


Property Rental Management Home Log In Sign Up

Welcome to the Property Rental Management System

The Property Rental Management System (PRMS) is a web-based application designed to streamline the management of rental properties. This system allows property owners, managers, and tenants to efficiently handle property listings, rental agreements, maintenance requests, and payments.

2. Sign Up/ Register Page:


Property Rental Management Home Log In Sign Up

Sign Up

Username

Password

First Name

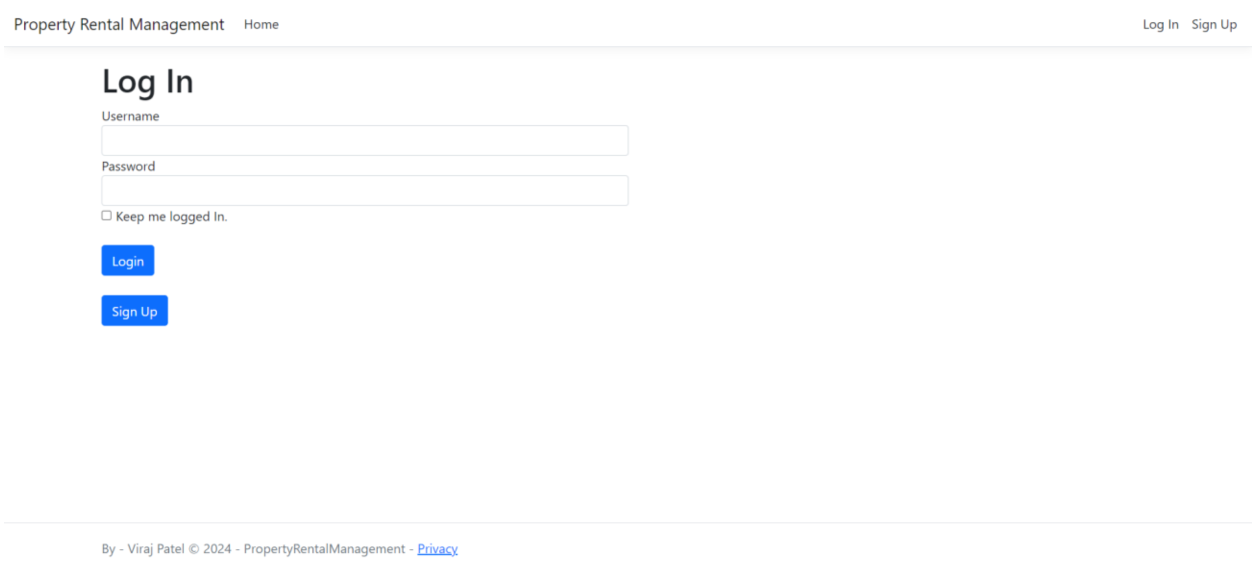
Last Name

Email Address

Phone Number

By - Viraj Patel © 2024 - PropertyRentalManagement - Privacy

3. Log in Page:



Log In

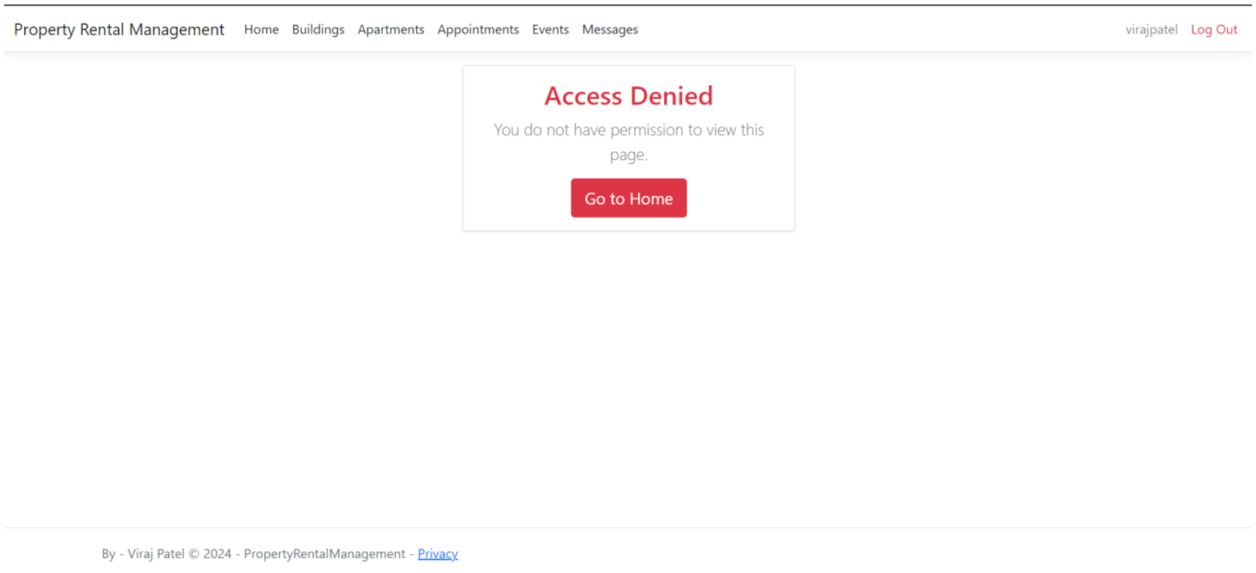
Username

Password

☐ Keep me logged In.

By - Viraj Patel © 2024 - PropertyRentalManagement - Privacy

4. Access Denied Page:

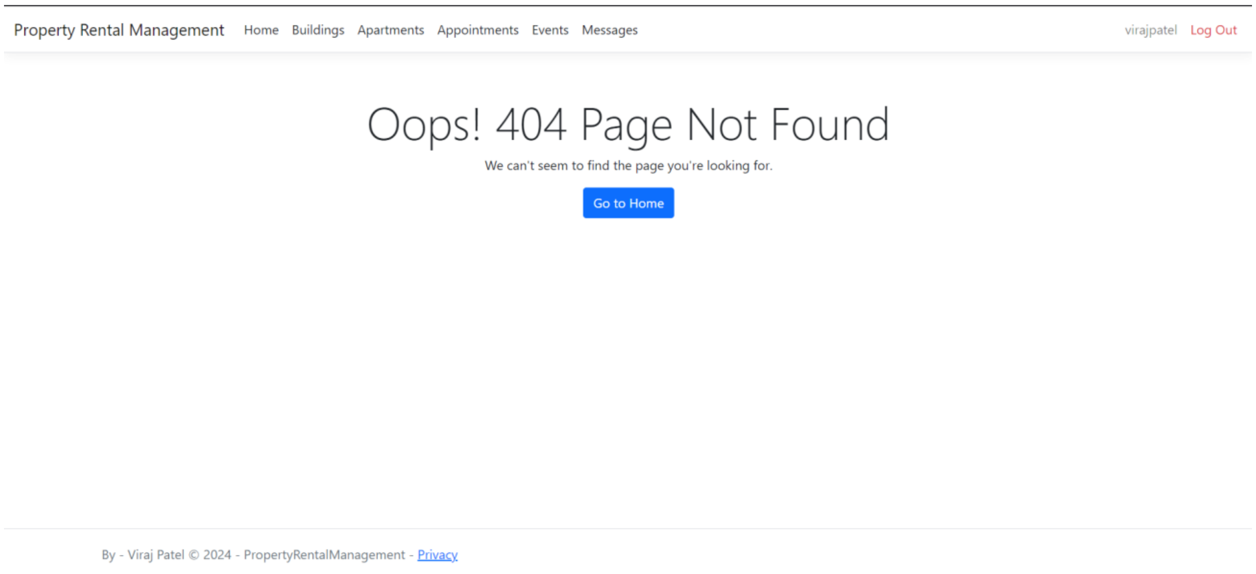


Access Denied

You do not have permission to view this page.

By - Viraj Patel © 2024 - PropertyRentalManagement - Privacy

5. Not Found Page/ 404 Page:

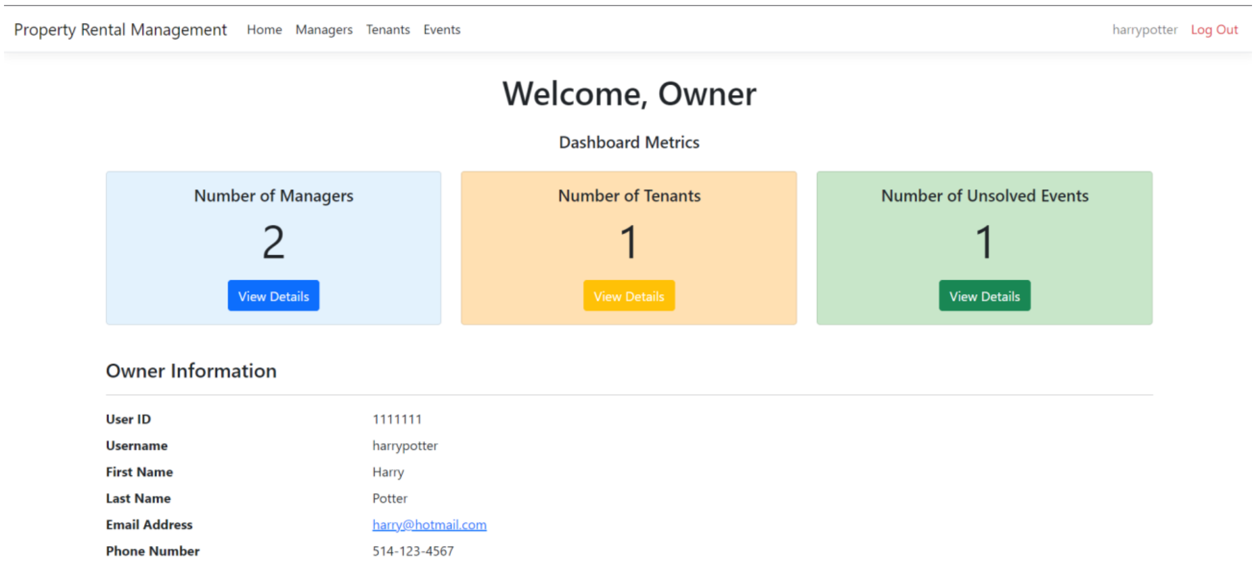


Oops! 404 Page Not Found

We can't seem to find the page you're looking for.

By - Viraj Patel © 2024 - PropertyRentalManagement - Privacy

6. Owner Dashboard:



Welcome, Owner

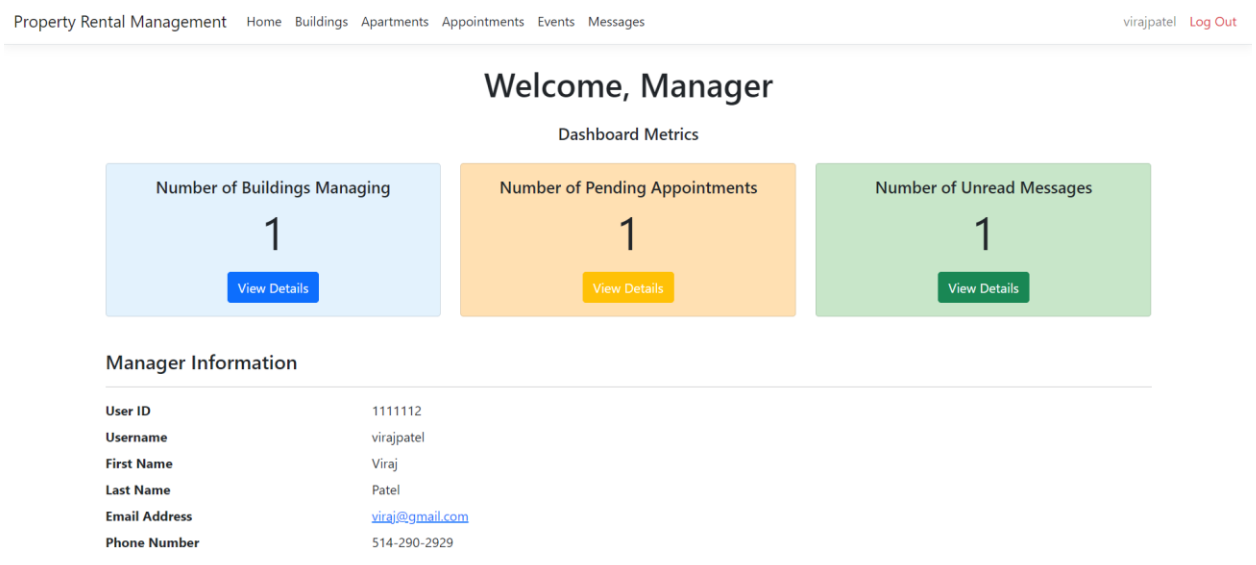
Dashboard Metrics

Number of Managers	Number of Tenants	Number of Unsolved Events
2	1	1

Owner Information

User ID	1111111
Username	harrypotter
First Name	Harry
Last Name	Potter
Email Address	harry@hotmail.com
Phone Number	514-123-4567

7. Manager Dashboard:



Welcome, Manager

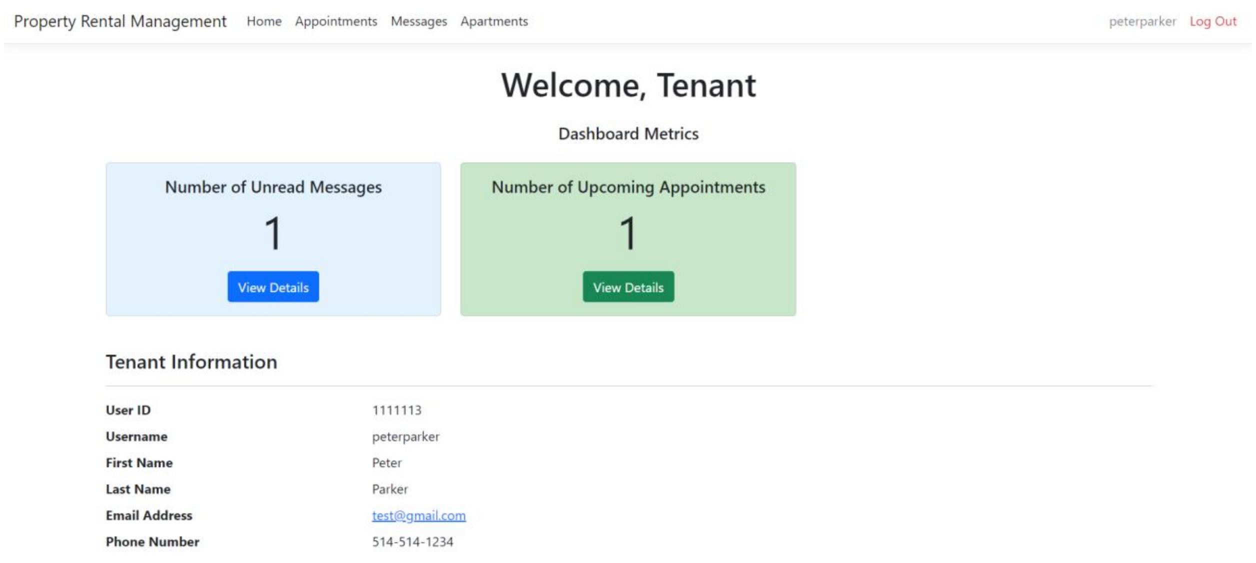
Dashboard Metrics

Number of Buildings Managing	Number of Pending Appointments	Number of Unread Messages
1	1	1
View Details	View Details	View Details

Manager Information

User ID	1111112
Username	virajpatel
First Name	Viraj
Last Name	Patel
Email Address	viraj@gmail.com
Phone Number	514-290-2929

8. Tenant Dashboard:



Welcome, Tenant

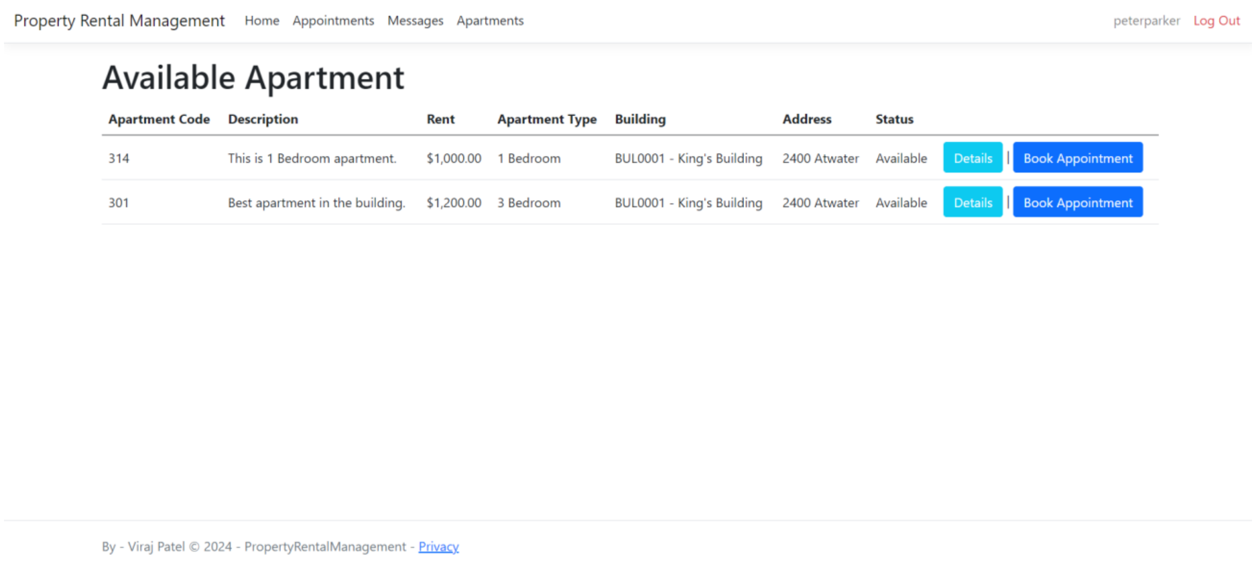
Dashboard Metrics

Number of Unread Messages	Number of Upcoming Appointments
1	1
View Details	View Details

Tenant Information

User ID	1111113
Username	peterparker
First Name	Peter
Last Name	Parker
Email Address	test@gmail.com
Phone Number	514-514-1234

9. Available Apartments:



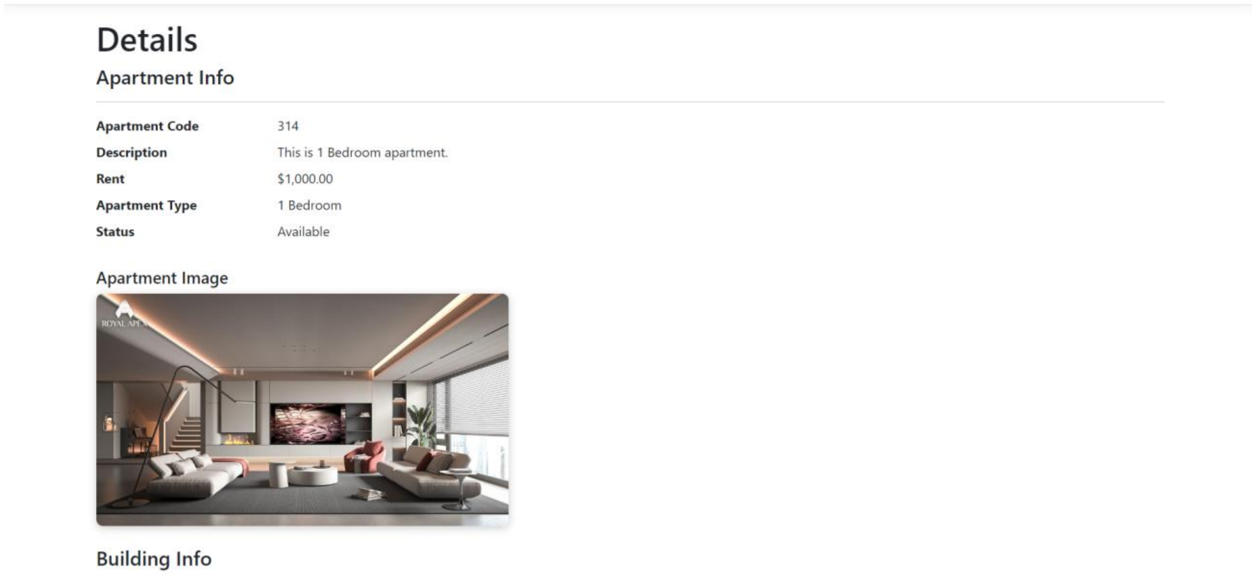
Property Rental Management Home Appointments Messages Apartments peterparker Log Out

Available Apartment

Apartment Code	Description	Rent	Apartment Type	Building	Address	Status	
314	This is 1 Bedroom apartment.	$1,000.00	1 Bedroom	BUL0001 - King's Building	2400 Atwater	Available	Details | Book Appointment
301	Best apartment in the building.	$1,200.00	3 Bedroom	BUL0001 - King's Building	2400 Atwater	Available	Details | Book Appointment

By - Viraj Patel © 2024 - PropertyRentalManagement - Privacy

10. Details Page (With Image):



Details

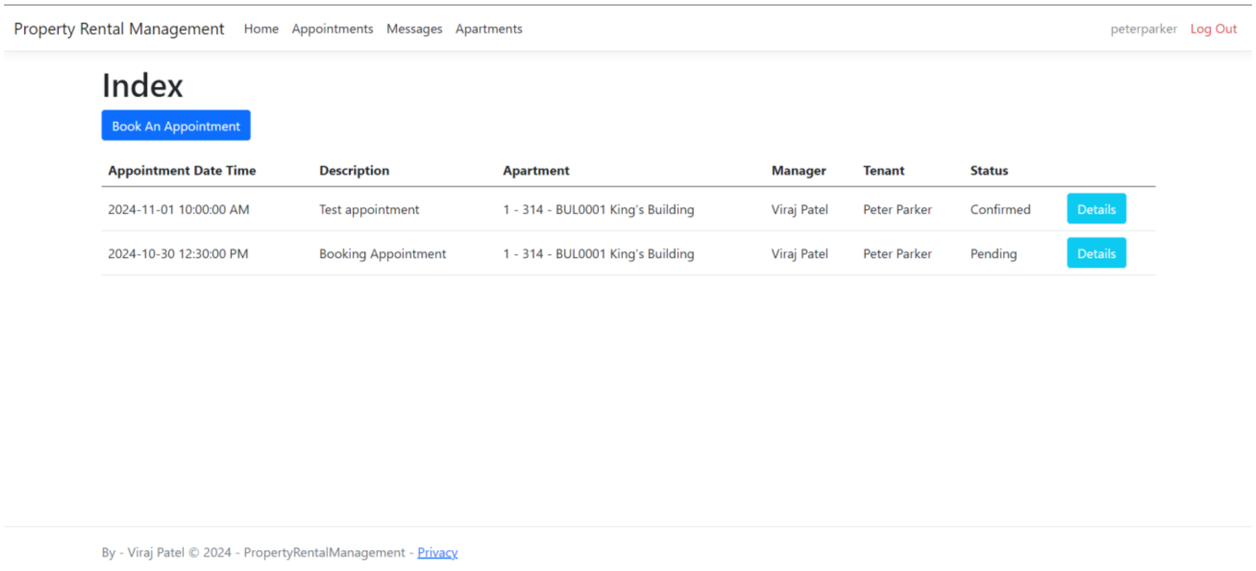
Apartment Info

Apartment Code	314
Description	This is 1 Bedroom apartment.
Rent	$1,000.00
Apartment Type	1 Bedroom
Status	Available

Apartment Image

Building Info

11. Tenant's Apartments Page:


Property Rental Management Home Appointments Messages Apartments peterparker Log Out

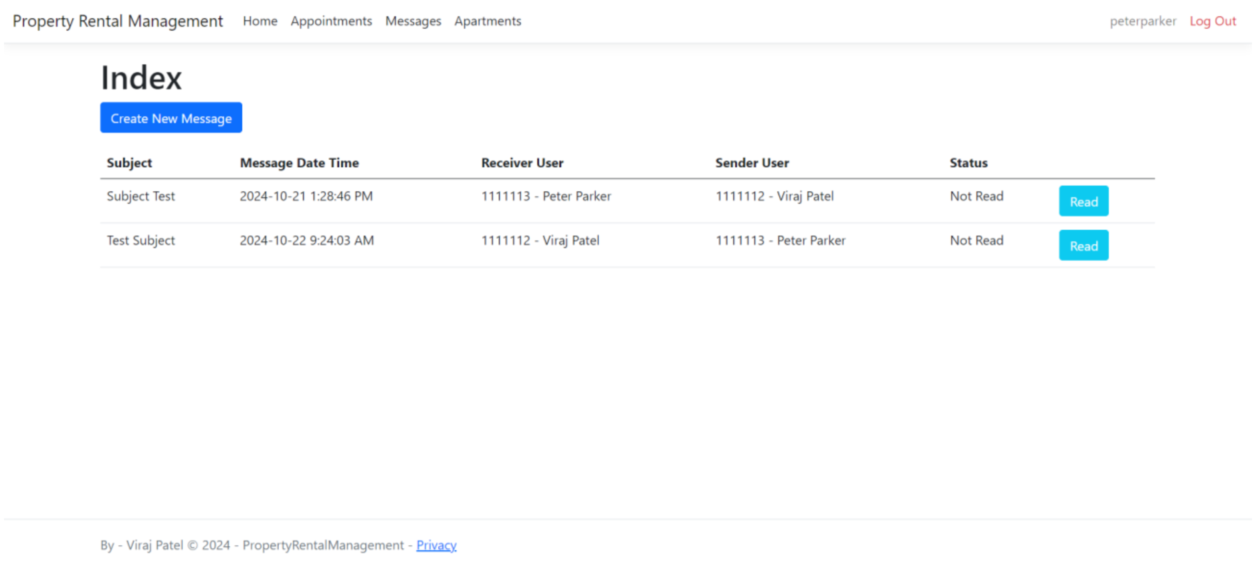
Index

Book An Appointment

Appointment Date Time	Description	Apartment	Manager	Tenant	Status	
2024-11-01 10:00:00 AM	Test appointment	1 - 314 - BUL0001 King's Building	Viraj Patel	Peter Parker	Confirmed	Details
2024-10-30 12:30:00 PM	Booking Appointment	1 - 314 - BUL0001 King's Building	Viraj Patel	Peter Parker	Pending	Details

By - Viraj Patel © 2024 - PropertyRentalManagement - Privacy

12. Tenants and Managers Messages Page:


Property Rental Management Home Appointments Messages Apartments peterparker Log Out

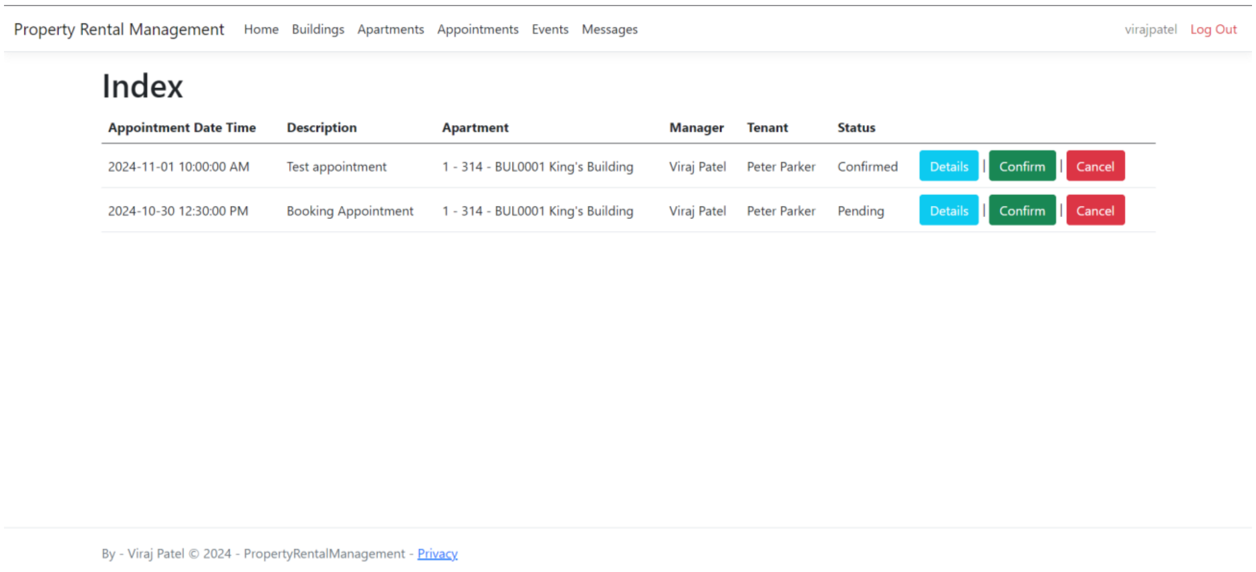
Index

Create New Message

Subject	Message Date Time	Receiver User	Sender User	Status	
Subject Test	2024-10-21 1:28:46 PM	1111113 - Peter Parker	1111112 - Viraj Patel	Not Read	Read
Test Subject	2024-10-22 9:24:03 AM	1111112 - Viraj Patel	1111113 - Peter Parker	Not Read	Read

By - Viraj Patel © 2024 - PropertyRentalManagement - Privacy

13. Managers’ Appointments Page:

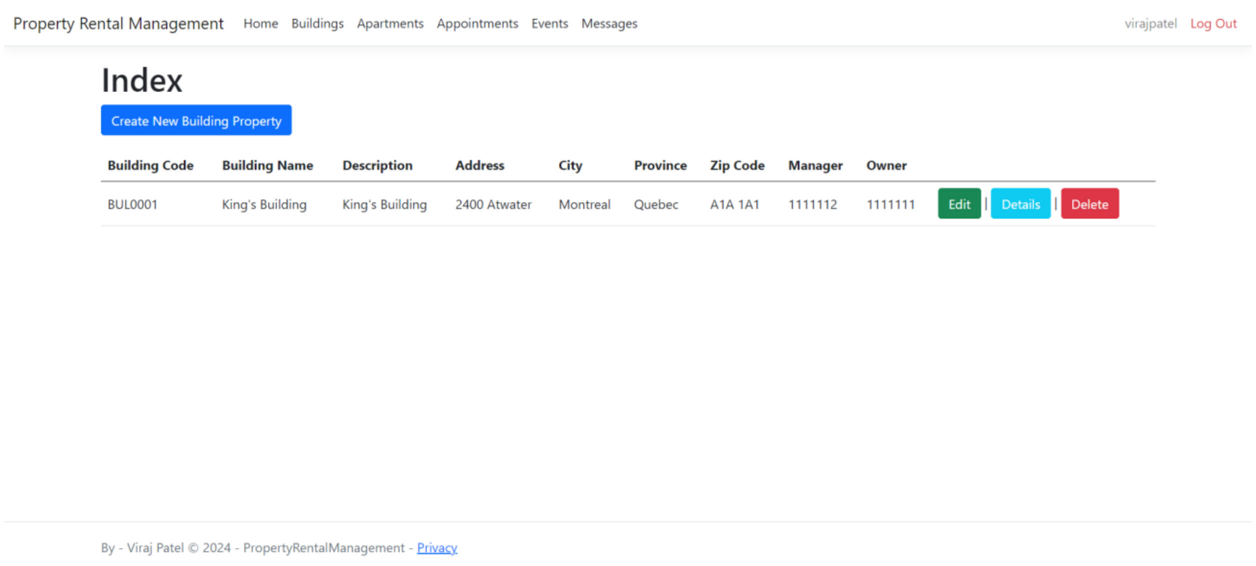

Property Rental Management   Home   Buildings   Apartments   Appointments   Events   Messages   virajpatel   Log Out

Index

Appointment Date Time	Description	Apartment	Manager	Tenant	Status	
2024-11-01 10:00:00 AM	Test appointment	1 - 314 - BUL0001 King's Building	Viraj Patel	Peter Parker	Confirmed	Details | Confirm | Cancel
2024-10-30 12:30:00 PM	Booking Appointment	1 - 314 - BUL0001 King's Building	Viraj Patel	Peter Parker	Pending	Details | Confirm | Cancel

By - Viraj Patel © 2024 - PropertyRentalManagement - Privacy

14. Buildings manage Page:


Property Rental Management   Home   Buildings   Apartments   Appointments   Events   Messages   virajpatel   Log Out

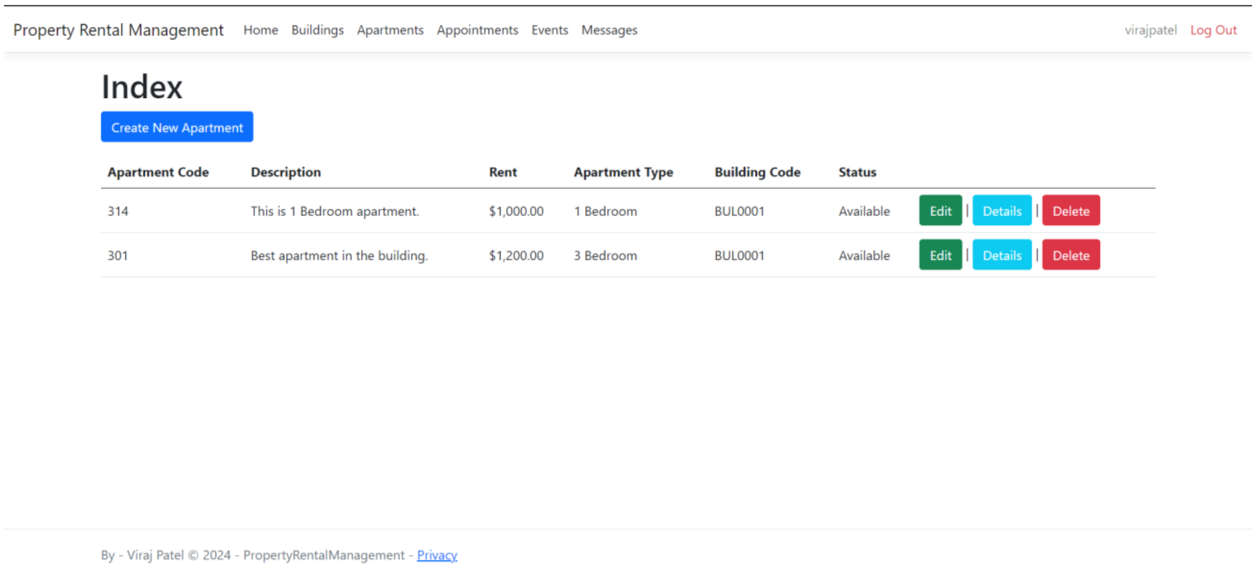
Index

Create New Building Property

Building Code	Building Name	Description	Address	City	Province	Zip Code	Manager	Owner	
BUL0001	King's Building	King's Building	2400 Atwater	Montreal	Quebec	A1A 1A1	1111112	1111111	Edit | Details | Delete

By - Viraj Patel © 2024 - PropertyRentalManagement - Privacy

15. Apartments manage Page:


Property Rental Management Home Buildings Apartments Appointments Events Messages virajpatel Log Out

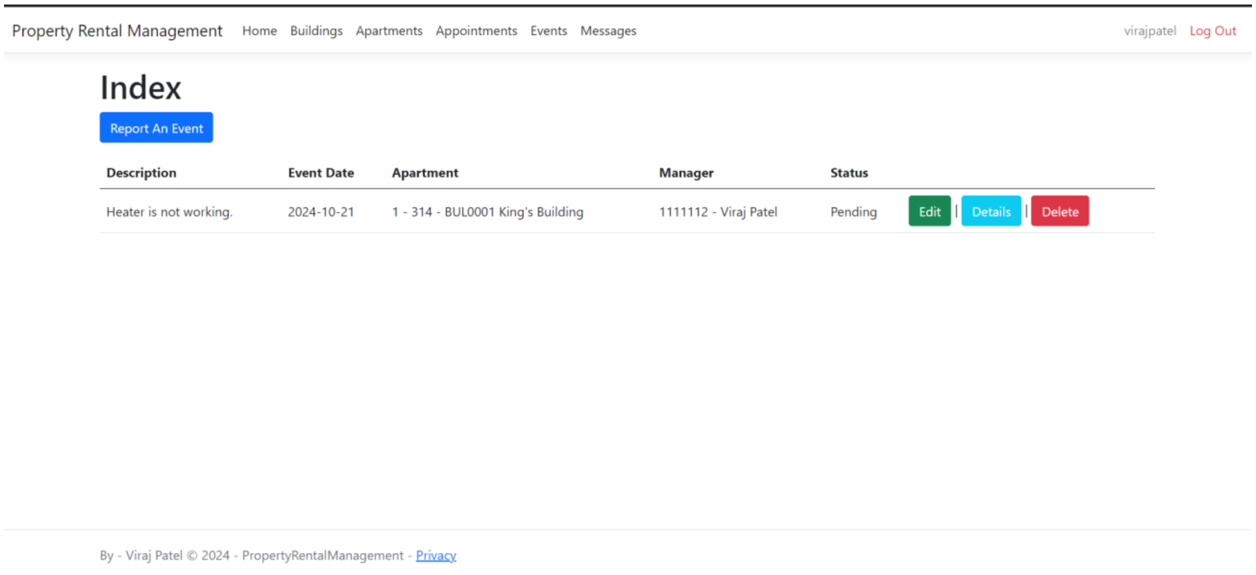
Index

Create New Apartment

Apartment Code	Description	Rent	Apartment Type	Building Code	Status	
314	This is 1 Bedroom apartment.	$1,000.00	1 Bedroom	BUL0001	Available	Edit | Details | Delete
301	Best apartment in the building.	$1,200.00	3 Bedroom	BUL0001	Available	Edit | Details | Delete

By - Viraj Patel © 2024 - PropertyRentalManagement - Privacy

16. Managers' Events Page:


Property Rental Management Home Buildings Apartments Appointments Events Messages virajpatel Log Out

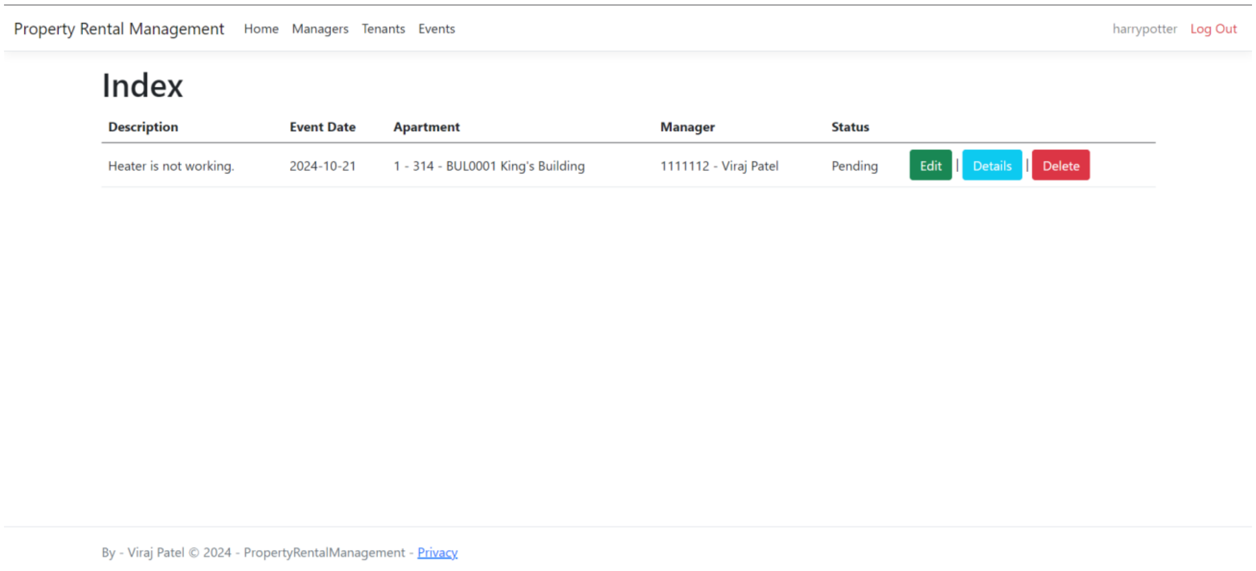
Index

Report An Event

Description	Event Date	Apartment	Manager	Status	
Heater is not working.	2024-10-21	1 - 314 - BUL0001 King's Building	1111112 - Viraj Patel	Pending	Edit | Details | Delete

By - Viraj Patel © 2024 - PropertyRentalManagement - Privacy

17. Owner' Event Page:

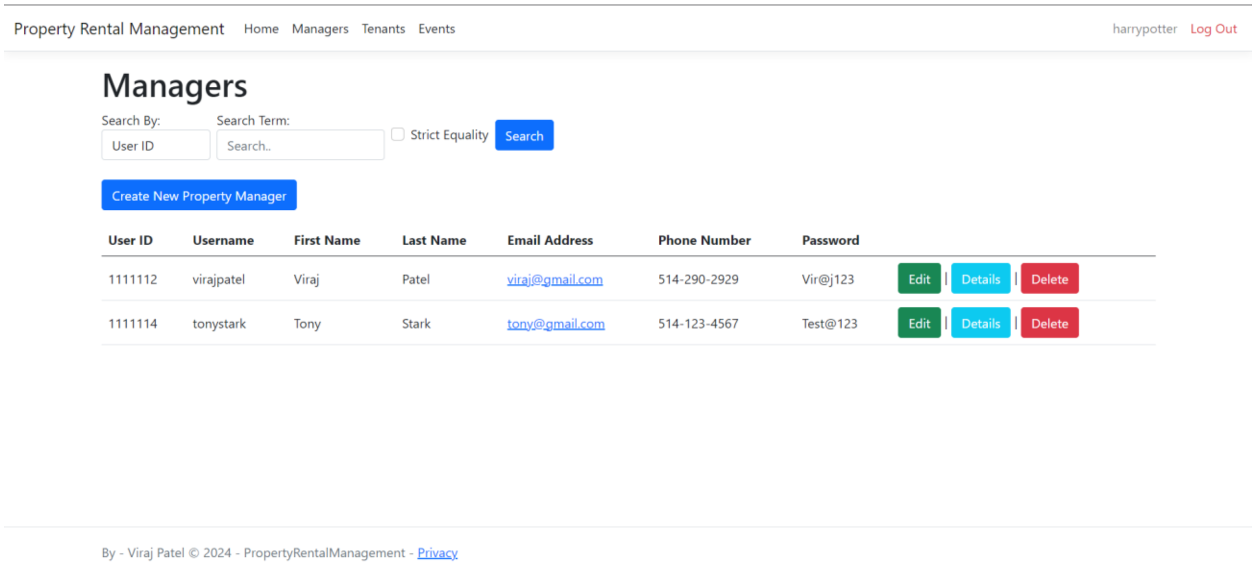

Property Rental Management Home Managers Tenants Events harrypotter Log Out

Index

Description	Event Date	Apartment	Manager	Status	
Heater is not working.	2024-10-21	1 - 314 - BUL0001 King's Building	1111112 - Viraj Patel	Pending	Edit | Details | Delete

By - Viraj Patel © 2024 - PropertyRentalManagement - Privacy

18. Managers manage Page:


Property Rental Management Home Managers Tenants Events harrypotter Log Out

Managers

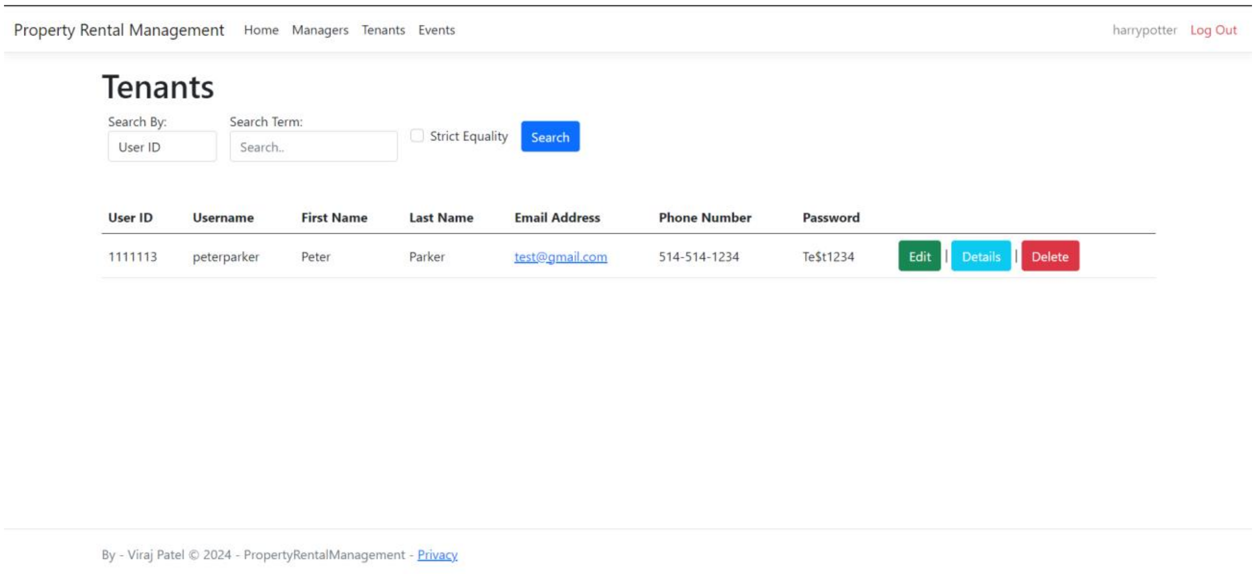
Search By: User ID Search Term: Search.. Strict Equality Search

Create New Property Manager

User ID	Username	First Name	Last Name	Email Address	Phone Number	Password	
1111112	virajpatel	Viraj	Patel	viraj@gmail.com	514-290-2929	Vir@j123	Edit | Details | Delete
1111114	tonystark	Tony	Stark	tony@gmail.com	514-123-4567	Test@123	Edit | Details | Delete

By - Viraj Patel © 2024 - PropertyRentalManagement - Privacy

19. Tenants manage Page:


Property Rental Management Home Managers Tenants Events harrypotter Log Out

Tenants

Search By: User ID Search Term: Search.. Strict Equality Search

User ID	Username	First Name	Last Name	Email Address	Phone Number	Password	
1111113	peterparker	Peter	Parker	test@gmail.com	514-514-1234	Te$t1234	Edit | Details | Delete

By - Viraj Patel © 2024 - PropertyRentalManagement - Privacy

Users Login Credentials

1. Owner:

Username: harrypotter

Password: H@rry123

2. Manager:

Username: virajpatel

Password: Vir@j123

3. Tenant

Username: peterparker

Password: Te$t1234

Phase IV Testing the Program

User	**Functional requirement**	**Test result/ problems**
User (Potential Tenants, Owners, Managers)	Sign in	Successful Users can sign in and are redirected to the correct dashboard based on their roles. However, if already logged in, using another browser requires re-login.
Potential Tenants	Sign Up	Successful. New potential tenants can create an account using the sign-up page.
	Access Restrictions	Successful. Potential tenants do not have access to pages restricted to managers and owners.
	View Available Apartments	Successful. Potential tenants can browse and view all available apartments directly from the Available Apartments page.
	Make an Appointment with Property Manager	Successful. Potential Tenants can schedule appointments with property managers through the Appointments page or Available Apartments Page.
	Send Messages to Property Manager	Successful. Potential Tenants can send messages to property managers from the apartment details page.
Property Managers	Access Restrictions	Successful. Property managers do not have access to pages restricted to potential tenants and owners.
	Perform CRUD Operations on Buildings	Successful. Property Managers can create, update, delete, and view building details via the navigation bar.
	Perform CRUD Operations on Apartments	Successful. Property Managers can manage apartment details (create, update, delete, list) through the navigation bar.
	Track Apartment Status	Successful. Property Managers can track the status of

		apartments (available, rented, unavailable) from apartment’s edit page.
	Schedule Appointments with Potential Tenants	Successful. Property managers can schedule appointments from the Appointments page.
	Respond to Tenant Messages	Successful. Property managers can send messages to potential tenants and respond to messages through the Messages page.
	Report and manage Events to Property Owner	Successful. Managers can report maintenance events or incidents to property owners via the Events page.
Property Owner	Access Restrictions	Successful. Property Owner do not have access to pages restricted to potential tenants and property managers.
	Manage Property Manager Accounts (CRUD)	Successful. Property Owner can create, update, delete, and view property managers from the navigation bar under the "Managers" section of the navigation bar.
	Manage Potential Tenant Accounts	Successful. Property Owner can update, delete, search, and view tenant accounts through the "Tenants" section of the navigation bar.
	View, Edit, Delete Responded Events	Successful. Property Owner can access, edit, and delete all reported events through the "Events" section of the navigation bar.

Summary of Test Results:

All functional requirements were thoroughly tested and successfully validated. The system handled operations efficiently for all types of users (potential tenants, property managers, and property owners) without any major issues. Security testing confirmed that unauthorized users were properly restricted from accessing protected resources. The user interfaces for each role were intuitive, and the navigation experience was smooth. Minor UI improvements were suggested to further enhance user experience, particularly in message notifications, but no critical defects were identified.

Conclusion

The development of the Property Rental Management System (PRMS) offered invaluable learning experiences, enhancing my proficiency in full-stack web development. Through this project, I gained a solid understanding of key modern technologies, including ASP.NET Core MVC, Entity Framework Core, and Razor Pages. The project reinforced my skills in designing efficient database schemas, implementing object-relational mapping (ORM), and building secure authentication and authorization mechanisms. Using Visual Studio and SQL Server Management Studio further refined my abilities to manage and develop complex applications.

Moreover, the project strengthened my ability to create responsive, user-friendly interfaces through Bootstrap and jQuery, seamlessly integrating them with server-side technologies. By organizing the development process into phases - requirements analysis, development, and testing- I learned the importance of a structured approach to building scalable, maintainable systems. This course also honed my critical thinking skills, enabling me to analyze problems, identify potential issues, and implement effective solutions. It has deepened my technical expertise and prepared me for the demands of real-world software development, particularly in creating secure, enterprise-level applications.

Bibliography

1. Login Authentication/Authorization in ASP.NET CORE 6 MVC: https://www.youtube.com/watch?v=uGoNCJf0t1g
2. ChatGPT: https://chat.openai.com/chat
3. ASP.NET Core Documentation: https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-8.0
4. Stack Overflow: https://stackoverflow.com/