**Course Number: 420-DW4-AS**

**Course Title: Web Server Applications Development II**

**Session: Autumn 2024**

**Group: 7278**

**FINAL PROJECT REPORT**

**DUE DATE:**

**December 4th, 2024**

**SUBMITTED TO:**

**Teacher: QUANG HOANG CAO**

**BY:**

**Student ID: 2231473**

**Student Name: HUYNH TU ANH CHAU**

Contents

[Project description 4](#_Toc184040286)

[Project development 4](#_Toc184040287)

[Phase I – Analysis 4](#_Toc184040288)

[The Requirements 4](#_Toc184040289)

[Functional Requirements Document 4](#_Toc184040290)

[Phase II – Design 5](#_Toc184040291)

[Tables 5](#_Toc184040292)

[Relationship Diagram 8](#_Toc184040293)

[System Architecture 8](#_Toc184040294)

[Phase III – Implementation 8](#_Toc184040295)

[Home Page (after running this program): 8](#_Toc184040296)

[Owner Dashboard (after logging-in as an owner): 8](#_Toc184040297)

[Manager Dashboard (after logging-in as a manager): 8](#_Toc184040298)

[Tenant Dashboard (after logging-in as a tenant): 8](#_Toc184040299)

[Phase IV – Testing 8](#_Toc184040300)

[Conclusion 10](#_Toc184040301)

[Sample Data for User Login 10](#_Toc184040302)

[Login as an Owner 10](#_Toc184040303)

[Login as a Manager 10](#_Toc184040304)

[Login as a Tenant 10](#_Toc184040305)

[Bibliography 11](#_Toc184040306)

# Project description

The project is a *Property Rental Management System* designed for Property Owners, Managers, and Tenants to interact with the system and perform their respective operations seamlessly. The system is developed using the **ASP.NET MVC Framework and Entity Framework** for efficient database management. Development was carried out in **Visual Studio 2022**, while the database was designed and managed using **SQL Server Management Studio (SSMS) 19**.

This project incorporates **role-based access control (RBAC)** to ensure that each user type **(Owner, Manager, Tenant)** has access only to functionalities specific to their role. The interface is designed with a focus on user-friendliness and responsiveness, allowing each user type to perform **CRUD** (Create, Read, Update, Delete) operations easily and securely.

# Project development

## Phase I – Analysis

### The Requirements

include providing roles for Property Owners, Managers, and Tenants to interact with the system and perform specific CRUD operations.

### Functional Requirements Document

#### Owners can:

* + - * 1. Manage managers account (CRUD operations).
        2. Search for any managers by email.
        3. Edit, view details, and delete any tenant accounts.
        4. Search for any tenants by email.
        5. View all buildings and apartments (optional)
        6. Search for buildings by address, city, province, etc. (optional)
        7. Search for apartments by address, city, postal code, etc. (optional)
        8. Send messages to a manager.
        9. Edit message’s content, view details of message, delete sent messages which sent to a manager. (optional)
        10. View, update, and view details assigned events which are created by managers.
        11. Edit or view details or delete their own accounts. (optional)

#### Managers can:

* + - * 1. Manage buildings, apartments (CRUD operations).
        2. Search for assigned buildings by address, city, province, etc. (optional)
        3. Search for assigned apartments by address, city, postal code, etc. (optional)
        4. Manage appointments with tenants (CRUD operations).
        5. Response(Edit) to owners’ messages.
        6. View details or delete owners’ messages.
        7. Response(Edit) to tenants’ messages.
        8. View details or delete tenants' messages.
        9. Manage events with owners (CRUD operations).
        10. Edit or view details or delete their own accounts. (optional)

#### Tenants can:

* + - * 1. Register to Property Rental Management Web Site.
        2. View all apartments.
        3. Search for apartments by address, city, province, etc. (optional)
        4. View Details for each apartment. (optional)
        5. Send messages to a manager.
        6. Edit message’s content, view details of message, delete sent messages which sent to a manager. (optional)
        7. Book appointments with a manager.
        8. Edit or view details or delete their own accounts. (optional)

## Phase II – Design

### Tables

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

### A computer screen shot of a computer flow chart Description automatically generatedRelationship Diagram

### System Architecture

* + - * MVC (Model-View-Controller) pattern.
      * Role-based access control.

## Phase III – Implementation

### Home Page (after running this program):

* + - * Displays system information and **lists** available apartments.
      * Includes a **search bar** for filtering apartments by apartment ID.
      * Includes tabs to **login** as **owner/manager/tenant.**
      * Include tabs to **register** as **owners or tenants**.

### Owner Dashboard (after logging-in as an owner):

* + - * Edit or view details or delete their own accounts.
      * Access to manage **Managers** (CRUD operations and search functionality).
      * Access to manage **Tenants** (CRUD operations and search functionality).
      * View details of all **Buildings** and **Apartments** with search functionality.
      * Manage **messages** to a manager (CRUD operations)
      * View assigned **events** and update its status or view its details.

### Manager Dashboard (after logging-in as a manager):

* + - * Edit or view details or delete their own accounts.
      * Access to manage **Buildings** (CRUD operations and search functionality).
      * Access to manage **Apartments** (CRUD operations and search functionality).
      * Access to manage **messages** which are created by **owners**. Manager can response, view details, and delete messages.
      * Access to manage **messages** which are created by **tenants**. Manager can response, view details, and delete messages.
      * Access to manage **appointments** with **tenants** (CRUD operations).
      * Access to manage **events** with **owners** (CRUD operations).

### Tenant Dashboard (after logging-in as a tenant):

* + - * Edit or view details or delete their own accounts.
      * View all **Apartments** with search functionality.
      * Access to manage **messages** to **managers** (CRUD operations).
      * Access to manage **appointments** to **managers** (CRUD operations).

## Phase IV – Testing

|  |  |  |
| --- | --- | --- |
| **User Role** | **Functional Requirements** | **Result** |
| Owner | Create/Update/Delete/Search/List any property manager account | Passed |
| Owner | Not allowing to edit/create manager’s email that already existing. | Passed |
| Owner | View details of all Buildings and Apartments with search functionality. | Passed |
| Owner | Update/Delete/Search/List any potential tenant account. | Passed |
| Owner | * Not allowing to edit tenant ‘s email that already existing. * Allow bypassing Tenant model validation if leaving password field empty if owner does not want to edit existing password of the chosen tenant account. | Passed |
| Owner | Perform CRUD operations related to messages which will be sent to managers. | Passed |
| Owner | * Not allowing to edit manager ‘s email that already existing. * Allow bypassing Manager model validation if leaving password field empty if owner does not want to edit existing password of the chosen manager account. | Passed |
| Owner | Update assigned event’s status (Pending/Resolved) or view its details which are sent by managers. | Passed |
| Manager | Perform CRUD operations related to buildings, including search functionality. | Passed |
| Manager | Perform CRUD operations related to apartments, including search functionality. | Passed |
| Manager | Keep track of apartments status | Passed |
| Manager | Schedule potential tenants ‘appointments (CRUD operations). | Passed |
| Manager | Respond to potential tenants ‘messages (Edit, View Details, Delete operations). | Passed |
| Manager | Access to manager messages to owners (CRUD operations). | Passed |
| Manager | Report any events to the property  owner when necessary (CRUD operations). | Passed |
| Tenants | Create an on-line account through Property Rental Management Web Site | Passed |
| Tenants | View any available and already assigned apartments suitable for their needs | Passed |
| Tenants | Make an appointment with the property manager | Passed |
| Tenants | Send necessary messages to the property manager (CRUD operations). | Passed |

# Conclusion

This project provided hands-on experience with ASP.NET MVC and Entity Framework. The implementation emphasized:

* Leveraging Entity Framework for efficient database operations.
* Applying role-based access control to enhance security.
* Developing user-friendly and responsive web interfaces.

The project also reinforced the importance of planning, testing, and adhering to best practices in web development.

# Sample Data for User Login

## Login as an Owner

* + **Email:** [**tylerdurden@gmail.com**](mailto:tylerdurden@gmail.com)
  + **Password:** password123

## Login as a Manager

* + **Email:** [**ednamode@gmail.com**](mailto:ednamode@gmail.com)
  + **Password:** password123

## Login as a Tenant

* + **Email:** [**michaelcorleone@gmail.com**](mailto:michaelcorleone@gmail.com)
  + **Password:** password123

# Bibliography

 Microsoft ASP.NET Documentation: <https://learn.microsoft.com/aspnet>

 Entity Framework Documentation: <https://learn.microsoft.com/ef>

 Bootstrap Framework: <https://getbootstrap.com>