DHARMSINH DESAI UNIVERSITY, NADIAD

FACULTY OF TECHNOLOGY

DEPARTMENT OF COMPUTER ENGINEERING

BTech CE Semester-VI **Subject:** (CE624) Web

Service Development **Project Title:**

Hostel Complaint Management Portal

Submitted by:

Name: Manav Ketanbhai Shah

Roll No: CE125

ID: 20CEUOS079

Guided By:

Prof. Ankit P. Vaishnav

DHARMSINH DESAI UNIVERSITY

Certificate

This is to certify the practical / term work carried out in the subject of Web Service Development *and recorded in this journal is the bonafide work of Mr* Manav Ketanbhai Shah *Roll No:* CE125 *Identity No:* 20CEUOS079 *of BTech semester* VI *in the branch of* Computer Engineering *during the academic year 2022-2023*

Table of Content

1. Abstract

A “Hostel Complaint Management portal” is an online tool created to give students a way to express their complaints about a variety of problems they encounter about the hostel. Users can submit complaints through the site, follow the status of their complaints, and get in touch with the administrators at the hostel who will be addressing their concerns.

2. Introduction

The hostel complaint management portal is an online platform designed to help students living in hostels report and resolve their grievances effectively. Hostels are a crucial part of the student life experience, but they can also be source of frustration when issue arise. With the hostel complaint management portal, students can submit their complaints, track their progress, and receive updates on their status in real-time. This platform aims to streamline the process of addressing student complaints by providing a centralized database that enables hostel staff to efficiency manage and prioritize complaints.

Technology and Tools used

Technology

- Asp.Net Core
- React
- CSS
- MUI
- Tailwind CSS
- Web API

Tools

- Git
- Visual Studio 2022
- Visual Studio Code

3. Software Requirement Specification (SRS)

USERS OF THE SYSTEM:

1. User
2. Admin

FUNCTIONAL REQUIREMENTS:

1) Authentication

1.1 Registration

Description: Users can register themselves by giving their unique university id and password.
Input: name, university, email, role, password
Output: Registered successfully

1.2 Login

Description: The user needs to login with their username and password to have access to their account.
Input: username and password
Output: logged in successfully

1.3 Logout

Description: Once users have done their work, they can log out from the system for security purposes.

2) Manage Complaint

2.1 Create Complaint

Description: Users can raise their complaint which will display on the front page of the website.

2.2 Update Complaint
Description: User can update the complaint if he/she wants to.

2.3 Delete Complaint
Description: User can delete the complaint if he/she finds that complaint is irrelevant.

2.4 Review/Resolve Complaint
Description: Admin have to review each and every complaint and resolve them at appropriate time.

NON-FUNCTIONAL REQUIREMENTS:

N.1: The server hardware can be any computer capable of running both the web and database servers and handling the expected traffic.

N.2: System should be easily used by the users.

N.3: The system should be always available, meaning the user can access it using a web-browser.

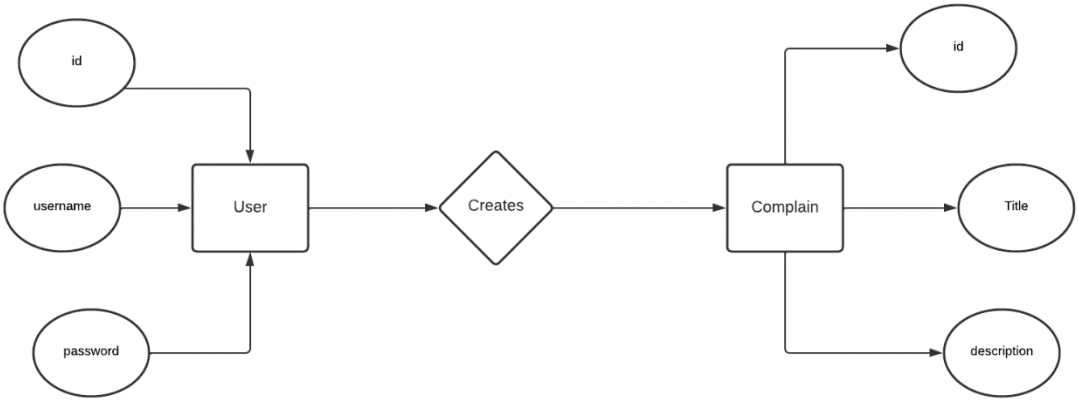
N.4: Secure access to confidential information of the users.

N.5: The system will be supported by Windows.

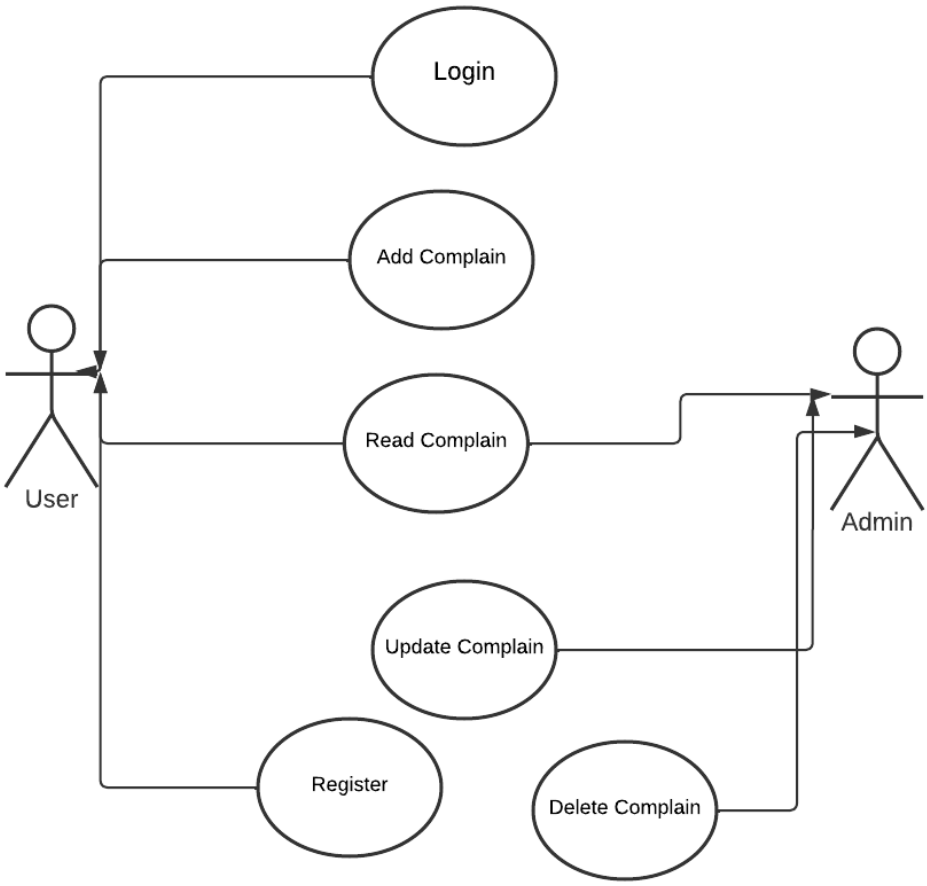
N.6: A database management system that is available free of cost in the public domain should be used.

4. Database Design

- **ER Diagram**



- **Use-case Diagram**



5. Data Dictionary

dbo.Users [Design] | dbo.Compl...[Design] | dbo.__EFM...[Design] | Complaints...troller.cs

Update | Script File: dbo.Users.sql

Name	Data Type	Allow Nulls	Default
Id	int	☐	
UserName	nvarchar(MAX)	☐	
PasswordHash	varbinary(MAX)	☐	
PasswordSalt	varbinary(MAX)	☐	
		☐	

Keys (1)
PK_Users (Primary Key, Clustered: Id)
Check Constraints (0)
Indexes (0)
Foreign Keys (0)
Triggers (0)

dbo.Com...[Design] | dbo.__EFM...[Design] | Complaints...troller.cs | Complaint.cs

Update | Script File: dbo.Complaints.sql

Name	Data Type	Allow Nulls	Default
Id	int	☐	
Title	nvarchar(MAX)	☐	
Complain	nvarchar(MAX)	☐	
UserId	int	☐	
resolve	bit	☐	(CONVERT([bit],(0)))
		☐	

Keys (1)
PK_Complaints (Primary Key, Clustered: Id)
Check Constraints (0)
Indexes (1)
IX_Complaints_UserId (UserId)
Foreign Keys (1)
FK_Complaints_Users_UserId (Id)
Triggers (0)

I. Modules created and brief description of each module

User Model:

It contains basic information.

- Username
- Password

```
namespace UniComplaint.Models
{
    12 references
    public class User
    {
        6 references
        public int Id { get; set; }
        5 references
        public string UserName { get; set; } = string.Empty;
        2 references
        public byte[] PasswordHash { get; set; } = Array.Empty<byte>();
        2 references
        public byte[] PasswordSalt { get; set; } = Array.Empty<byte>();

        0 references
        public List<Complaint>? Complaints { get; set; }
    }
}
```

Complain Model:

It contains information about complaints

- Complaint Title
- Complaint Description
- Complaint Resolve

```
namespace UniComplaint.Models
{
    8 references
    public class Complaint
    {
        [Key]
        3 references
        public int Id { get; set; }

        [Required]
        0 references
        public string? Title { get; set; } = string.Empty;

        [Required]
        0 references
        public string? Complain { get; set; } = string.Empty;

        0 references
        public bool resolve { get; set; } = false;
        0 references
        public int UserId { get; set; }
        [JsonIgnore]
        0 references
        public User? User { get; set; }
    }
}
```

ComplaintDbContext:

```
UniComplaint | UniComplaint.Models.ComplaintDbC | ComplaintDbContext(DbCont

using Microsoft.EntityFrameworkCore;

namespace UniComplaint.Models
{
    14 references
    public class ComplaintDbContext:DbContext
    {
        0 references
        public ComplaintDbContext(DbContextOptions<ComplaintDbContext> options) : base(options)
        {

        }
        0 references
        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            optionsBuilder.UseSqlServer(@"Server=(localdb)\MSSQLLocalDB;Database=UniversityComp;Trusted_Connection=True");
        }

        0 references
        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {

        }
        9 references
        public DbSet<User> Users => Set<User>();

        10 references
        public DbSet<Complaint> Complaints => Set<Complaint>();
    }
}
```

II. Function Prototype which implements major functionality

- **Authentication:**

```
        [HttpPost("Login")]
        0 references
        public async Task<ActionResult> Login(userLoginDTO userDTO)
        {
            var res = await _authRepo.Login(userDTO.Username, userDTO.Password);
            Console.WriteLine(res);
            if (res == null)
            {
                return BadRequest($"Incorrect username or password!");
            }
            return Ok(new { token = res, status = 200 });
        }
    }
}
```

We will collect user input data from the registration form and determine whether or not the user's email address already exists. If the user does not exist, we will add that user to the database and redirect the user to the login page.

- **CRUD Operations:**

1) Complain Details:

```
// GET: api/Complaints
[HttpGet]
0 references
public async Task<ActionResult<IEnumerable<Complaint>>> GetComplaints()
{
  if (_context.Complaints == null)
  {
      return NotFound();
  }
    return await _context.Complaints.ToListAsync();
}
```

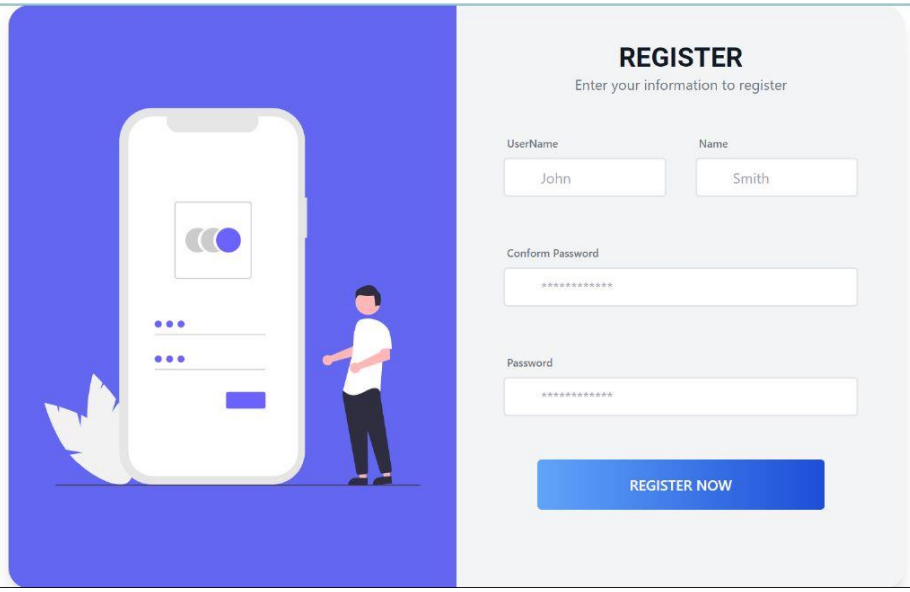
```
// GET: api/Complaints/5
[HttpGet("{id}")]
0 references
public async Task<ActionResult<Complaint>> GetComplaint(int id)
{
    if (_context.Complaints == null)
    {
        return NotFound();
    }
    var complaint = await _context.Complaints.FindAsync(id);

    if (complaint == null)
    {
        return NotFound();
    }

    return complaint;
}
```

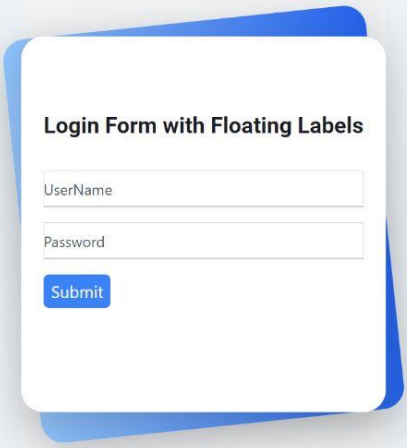
```
// PUT: api/Complaints/5
// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754
[HttpPut("{id}")]
0 references
public async Task<IActionResult> PutComplaint(int id, Complaint complaint)
{
    if (id != complaint.Id)
    {
        return BadRequest();
    }

    _context.Entry(complaint).State = EntityState.Modified;

    try
    {
        await _context.SaveChangesAsync();
    }
    catch (DbUpdateConcurrencyException)
    {
        if (!ComplaintExists(id))
        {
            return NotFound();
        }
        else
        {
            throw;
        }
    }

    return NoContent();
}
```

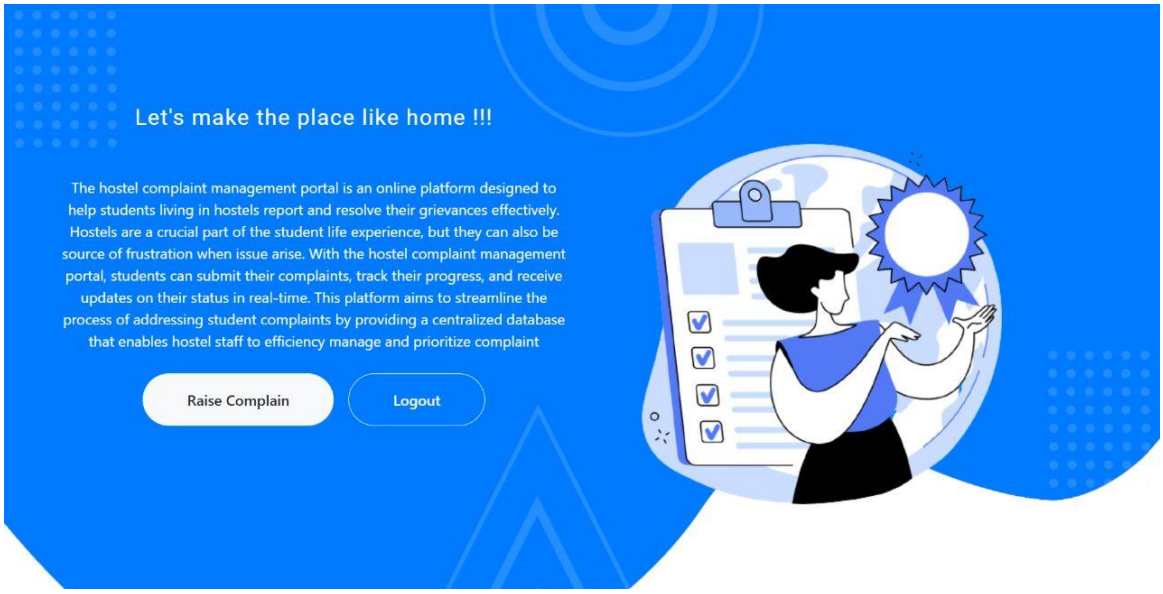
```
// POST: api/Complaints
// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754
[HttpPost]
0 references
public async Task<ActionResult<Complaint>> PostComplaint(Complaint complaint)
{
  if (_context.Complaints == null)
  {
      return Problem("Entity set 'ComplaintDbContext.Complaints'  is null.");
  }
    _context.Complaints.Add(complaint);
    await _context.SaveChangesAsync();

    return CreatedAtAction("GetComplaint", new { id = complaint.Id }, complaint);
}

// DELETE: api/Complaints/5
```

```
0 references
public async Task<IActionResult> DeleteComplaint(int id)
{
    if (_context.Complaints == null)
    {
        return NotFound();
    }
    var complaint = await _context.Complaints.FindAsync(id);
    if (complaint == null)
    {
        return NotFound();
    }

    _context.Complaints.Remove(complaint);
    await _context.SaveChangesAsync();

    return NoContent();
}
```

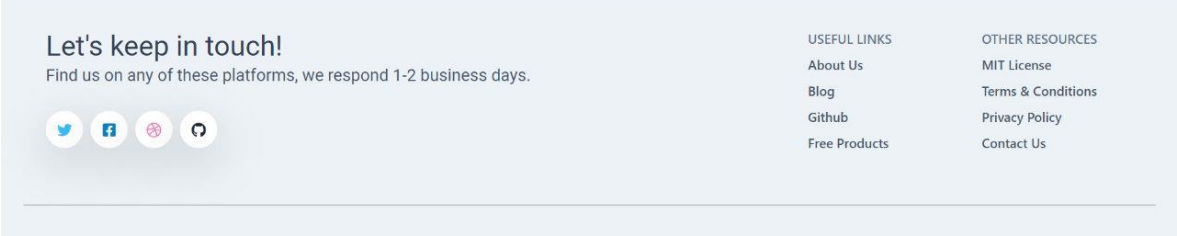
Screenshots:






Let's make the place like home !!!
The hostel complaint management portal is an online platform designed to help students living in hostels report and resolve their grievances effectively. Hostels are a crucial part of the student life experience, but they can also be source of frustration when issue arise. With the hostel complaint management portal, students can submit their complaints, track their progress, and receive updates on their status in real-time. This platform aims to streamline the process of addressing student complaints by providing a centralized database that enables hostel staff to efficiency manage and prioritize complaint
Raise Complain
Logout

Complains


Let's keep in touch!
Find us on any of these platforms, we respond 1-2 business days.
USEFUL LINKS
About Us
Blog
Github
Free Products
OTHER RESOURCES
MIT License
Terms & Conditions
Privacy Policy
Contact Us


Post your complain
Title
Complain
POST COMPLAIN

Members only

Title : Ragging

Complain : 3rd year student room number 121, they are disturbing my mental health.

Complain Id : 41
Aug 18

6. Testing:

Test Case Id	**Test Case Objective**	**Input Data**	**Expected Output**	**Actual Output**	**Status**
TC-01	Add new User into system	Credential	User Added	User Added	Pass
TC-02	Add user with already exists username	Credential			Pass
TC-03	Login into System	Credential	Token	Token	Pass
TC-04	Login into System with wrong credential	Wrong Credential			Pass
TC-05	Logout from system	Logout Button	Home Screen	Home Screen	Pass
TC-06	Complaint List	-	Complaint	Complaint	Pass
TC-07	Complaint Detail	Select Complaint	Complaint Content	Complaint Content	Pass
TC-08	Create Complaint	Complaint Details	Complaint added on home screen	Complaint added on home screen	Pass
TC-09	Update Complaint	Complaint ID along with new data	Updated Complaint	Updated Complaint	Pass
TC-10	Delete Complaint	Complaint ID	Complaint removed from site	Complaint removed from site	Pass

7.

8. Conclusion

Hostel Complaint Portal is completely free, easy to use, good user interface, and great user experience booking site which is created through ASP.NET Core Web API Technology for a Hostel.

The functionalities which are implemented in this project is:

- User Login
- User Registration
- CRUD Operations on Complaint
- User Logout

9. Limitations and Future Extension

Limitation:

- This project is limited only for CRUD oprations.

Future Extension:

- User can add comment for particular complaint.
- Separate login for hostel staff and students

9. Bibliography

- Stack Overflow
- https://learn.microsoft.com/en-us/aspnet/core/tutorials/first-web-api?view=aspnetcore-7.0&tabs=visual-studio

4.