ONLINE HELP DESK SYSTEM

A Campus Facilities Service Request Management Application

Prepared By:

Hafiz Muhammad Abdullah Khan
Muhmmad Ahsan
Muhammad Harmain Shahid
Muhammad Haris Khan

Project Guide:

Miss Kiran Shakeel

Institution:

Aptech Limited

Branch Name:

Metro Star Gate

Submission Date:

July 24 2025

TABLE OF CONTENTS

Sr. No.	Section Title	Page No.
1.	Certificate of Completion	1
2.	Declaration	2
3.	Acknowledgement	3
4.	Executive Summary	4
5.	Introduction	5
6.	Problem Definition	6
7.	Objectives of the Project	7
8.	Scope of the Project	8
9.	Roles & Functionalities	9
10.	System Architecture	10
12.	System Architecture Diagram	11
13.	ER Diagram	12
14.	Use Case Diagrams	13
15.	Screenshots of the Application UI	14
16.	Appendix	20
17.	Technology Stack Used	38
18.	Database Table Design	39
19.	Database Schema Design	40
20.	Testing Summary	41
21.	Unit Test Checklist	42
22.	Security Measures	43
23.	Limitations & Future Enhancements	44
24.	Final Checklist	45

CERTIFICATE

OF COMPLETION

This is to certify that
Hafiz Muhammad Abdullah Khan, Muhammad Haris Khan,
Muhammad Harmain Shahid, Muhmmad Ahsan
has successfully completed the project titled
"Online Help Desk (OHD) System"
under the guidance of Miss Kiran Shakeel.
The project encompasses the design, development,
testing, and deployment of an Intranet-based application
to automate service request workflows across campus
facilities, including classrooms, laboratories, hostels,
mess, canteen, gymnasium, and more.

Project Duration: June 24 2025 – July 24 2025
Organization: Aptech Limited (Metro Star Gate)
Awarded on this day, July 24 2025

Muhammad Abdullah
SIGNATURE

PROJECT LEADER:
Hafiz Muhammad Abdullah Khan

Kiran Shakeel
SIGNATURE

PROJECT GUDIER:
Miss Kiran Shakeel

DECLARATION

I hereby declare that the project titled
"Online Help Desk (OHD) System"
submitted in partial fulfillment of the requirements for the award of the diploma in [Your Program Name] is a record of my original work. This project has not been submitted previously by me or any other individual for any degree, diploma, or certificate at any other institute or university.

This work was carried out under the guidance of Miss Kiran Shakeel, and I have duly acknowledged all sources of information that have been used in this document.

I understand that any violation of academic integrity or plagiarism may result in disciplinary action in accordance with institutional policies.

SUBMITTED BY:

Name: Hafiz Muhammad Abdullah Khan
Id Number: 1546475
Batch: PR2-2024-01B2
Date: July 24 2025

Muhammad Abdullah

SIGNATURE

ACKNOWLEDGEMENT

بسم الله الرحمن الرحيم

First and foremost, I would like to express my deepest gratitude to Almighty Allah for granting me the strength, patience, and opportunity to successfully complete this project.

I would like to extend my sincere thanks to my respected instructor Miss Kiran Shakeel, whose valuable guidance, encouragement, and constructive feedback have been instrumental throughout the development of this project titled "Online Help Desk (OHD) System."

I am also thankful to the entire faculty of Aptech Limited, whose teachings laid the foundation for the technical and analytical skills required for this project.

Special thanks to my classmates and friends for their continuous support, collaborative spirit, and motivation during challenging phases of the project.

Lastly, I am extremely grateful to my parents and family, whose unwavering support and prayers have always been a source of inspiration and strength for me.

EXECUTIVE SUMMARY

The Online Help Desk (OHD) System is a centralized, web-based platform designed to streamline the reporting, assignment, and resolution of facility-related complaints within an organization or institution. The system aims to eliminate inefficiencies associated with manual complaint handling and introduces an automated, transparent, and role-based workflow that improves accountability and operational responsiveness.

The project was developed using the ASP.NET Core framework, leveraging C# as the primary backend language and a SQL Server database for persistent storage. It provides distinct access levels for four user roles: Admin, End User, Facility Head, and Assignee. Each role is equipped with customized functionalities to ensure seamless interaction with the system

- End Users can submit and track complaints in real time.
- Facility Heads can review incoming requests and assign them to appropriate personnel.
- Assignees are responsible for executing and updating the status of assigned tasks.
- Admins manage users, facilities, and monitor system performance.

The OHD System features intuitive dashboards, role-specific navigation, dynamic request tracking, and structured status transitions (Pending → Assigned → In Progress → Closed). It incorporates responsive UI design and modern user experience practices for both desktop and mobile platforms.

This project not only demonstrates technical implementation skills across frontend and backend development but also emphasizes the importance of user experience, data integrity, and systematic problem-solving within software engineering practices. The solution is scalable and can be adapted for universities, offices, hospitals, or any institution requiring structured help desk support.

In summary, the OHD System offers a reliable and professional approach to managing internal support requests—enhancing communication, reducing response time, and increasing overall efficiency across departments.

INTRODUCTION

In today's fast-paced and digitally connected environments, managing infrastructure and facility-related issues efficiently is a critical requirement for any organization, whether educational, corporate, or governmental. Traditional methods of complaint logging—such as handwritten registers, verbal communication, or informal emails—often lead to delays, lack of accountability, miscommunication, and unresolved issues. These inefficiencies directly affect the productivity and satisfaction of both users and support teams.

To address these challenges, the Online Help Desk (OHD) System has been conceptualized and developed as a centralized web-based platform that streamlines the entire lifecycle of facility-related complaint management. The system is designed to be scalable, secure, and role-driven, providing tailored access and actions based on user roles such as Admin, End User, Facility Head, and Assignee.

The OHD System empowers end users to easily submit complaints, track the real-time status of their requests, and ensure that their issues are addressed systematically. It enhances operational efficiency by allowing Facility Heads to monitor all requests under their assigned facilities and assign tasks based on staff availability. Meanwhile, Assignees receive clear task allocations and can update progress seamlessly, ensuring visibility and accountability.

From a technical standpoint, the system is built using ASP.NET MVC for server-side logic and SQL Server as the backend database. It features a responsive and intuitive frontend interface built with HTML, CSS (Tailwind), and JavaScript, making it accessible on both desktop and mobile devices.

The development of this project demonstrates key software engineering competencies, including requirement analysis, database design, system modeling, user interface development, and secure role-based access control. The OHD System not only offers a practical solution to real-world problems but also provides a strong foundation for further expansion into advanced modules such as analytics, notifications, and integration with IoT-based facility sensors.

Ultimately, the Online Help Desk System enhances internal service management by introducing transparency, efficiency, and structure—transforming the way complaints are handled across departments.

PROBLEM DEFINITION

In many educational and organizational campuses, maintenance and support requests for facilities—such as classrooms, laboratories, hostels, mess halls, canteens, gymnasiums, and computer centers—are handled through disparate channels (email, phone calls, paper slips). This fragmented approach leads to:

- Delayed Response: Requests often get lost or overlooked, resulting in slow resolution times.
- Lack of Transparency: End users cannot track the real-time status of their complaints, and facility managers lack a consolidated view of pending tasks.
- Inefficient Assignment: Facility heads must manually triage and forward requests, which can be error-prone and time-consuming.
- No Historical Insight: There is no centralized record of past requests, preventing data-driven decision making and trend analysis.

Consequently, both end users and support personnel experience frustration, reduced accountability, and wasted administrative effort. The problem, therefore, is to design and implement an integrated, web-based Online Help Desk (OHD) system that streamlines the lifecycle of facility service requests—from submission and assignment to resolution—while providing real-time tracking, automated notifications, and comprehensive reporting. This centralized solution will enhance operational efficiency, improve user satisfaction, and enable campus administrators to make informed decisions based on request analytics.

OBJECTIVE OF THE PROJECT

The primary objective of the Online Help Desk (OHD) System is to create a unified, user-friendly platform that automates and streamlines the workflow for facility service requests across a campus. Specifically, the project aims to:

1. Simplify Request Submission:
   - Provide registered end users (students, faculty, staff) with an intuitive interface to log new requests by selecting the relevant facility, specifying the severity, and adding a brief description.
2. Enable Real-Time Tracking:
   - Allow requestors to view the current status of their submissions (Pending, Assigned, In Progress, Closed, Rejected) and access a history of past requests.
3. Facilitate Efficient Assignment:
   - Equip facility heads with a dashboard to review incoming requests and assign them to appropriate assignees based on availability and expertise.
4. Support Transparent Resolution:
   - Empower assignees to update request statuses, add remarks, and mark tasks as completed, ensuring clear communication among all stakeholders.
5. Enhance Accountability and Auditability:
   - Maintain a comprehensive audit trail of status changes, timestamps, and user actions for compliance and continuous improvement.
6. Scalable Architecture:
   - Design the system with extensible data models (Users, Requests, Facilities, Status Histories) to accommodate future enhancements—such as additional roles, severity levels, or integration with external systems.
7. Responsive, Role-Based UI:
   - Deliver distinct, responsive dashboards tailored to each user role (Admin, End User, Facility Head, Assignee), ensuring an optimal experience on desktop and mobile devices.

By achieving these objectives, the OHD System will significantly reduce response times, improve satisfaction for both requestors and service providers, and provide campus leadership with actionable insights into facility maintenance operations.

SCOPE OF THE PROJECT

The Online Help Desk (OHD) System is designed to manage and track facility-related complaints in an organization such as a university or office. The system helps users submit complaints and allows Admins, Facility Heads, and Assignees to handle them efficiently.

What the System Includes:

- User Login & Roles: Admin, End User, Facility Head, and Assignee.
- Request Submission: End Users can submit complaints about issues like Wi-Fi, AC, or lighting.
- Request Tracking: Each complaint goes through statuses like Pending, Assigned, In Progress, and Closed.
- Task Assignment: Facility Heads assign tasks to Assignees who work on fixing the problem.
- Dashboards: Each role sees only what is relevant to them (e.g., My Tasks, Assign Requests).
- Facility Management: Admins manage different facilities like Labs, Libraries, or Auditoriums.
- Basic Analytics: The system shows stats like total complaints, resolved issues, etc.
- Simple UI: User-friendly interface that works well on both desktop and mobile.

ROLES & FUNCTIONALITIES

The Online Help Desk (OHD) System supports multiple user roles, each with specific access and responsibilities:

1. Admin:

- Add, edit, and delete users
- Manage facilities and facility heads
- Approve new user registrations
- Monitor overall system activity

2. End User:

- Register and log in to the system
- Submit new complaints/requests
- View and track the status of submitted requests
- Edit personal profile and password

3. Facility Head:

- View all requests related to their assigned facilities
- Assign requests to Assignees
- Monitor request progress and resolution
- View summary dashboards

4. Assignee:

- View assigned tasks
- Update request status (e.g., Assigned → In Progress → Closed)
- View task history and deadlines

SYSTEM ARCHITECTURE

The Online Help Desk (OHD) System is designed using a three-tier architecture, ensuring modularity, scalability, and maintainability:

1. Presentation Layer (Front-End)

- Built using ASP.NET Web Forms, HTML, CSS, Tailwind, and JavaScript
- Provides user interfaces for different roles: Admin, End User, Facility Head, Assignee
- Handles client-side input validation and dynamic UI components

2. Application Layer (Business Logic)

- Implements core functionalities like authentication, request routing, status updates, and user-role-based access control
- Controllers manage request handling, session management, and role-based logic
- Ensures smooth flow between user actions and database interaction

3. Data Layer (Database)

- Utilizes Entity Framework Core with SQL Server backend
- Stores structured data: Users, Facilities, Requests, Roles, Status Logs
- Supports CRUD operations and maintains referential integrity

SYSTEM ARCHITECTURE DIAGRAM

ER DIAGRAM

Users
user_id
name
role
Submits
1
Facilities
facility_id
name
*
Requests
request_id
status
*
Related To
1
requests

USE CASE DIAGRAM

SCREENSHOTS OF THE APPLICATION UI

Landing Page

SCREENSHOTS OF THE APPLICATION UI

Login Page

SCREENSHOTS OF THE APPLICATION UI

Admin Dashboard

SCREENSHOTS OF THE APPLICATION UI

User Dashboard

SCREENSHOTS OF THE APPLICATION UI

Facility Head Dashboard

SCREENSHOTS OF THE APPLICATION UI

Assignee Dashboard

APPENDIX

Index.cshtml

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>OHD System - Online Help Desk</title>
    <script src="https://cdn.tailwindcss.com"></script>
    <link rel="stylesheet" href="~/Home/style.css">
</head>
<body>
    <!-- Header/Navigation -->
    <header class="gradient-bg text-white py-6">
        <div class="container mx-auto px-4">
            <div class="flex justify-between items-center">
                <div class="flex items-center space-x-2">
                    <svg xmlns="http://www.w3.org/2000/svg" class="h-8 w-8" fill="none" viewBox="0 0 24 24" stroke="currentColor">
                        <path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M9 3v2m6-2v2M9 19v2m6-2v2M5 9H3m2 6H3m18-6h-2m2 6h-2M7 19h10a2 2 0 002-2V7a2 2 0 00-2-2H7a2
                    </svg>
                    <span class="text-xl font-bold">OHD System</span>
                </div>

                <nav class="hidden md:flex space-x-6">
                    <a href="#" class="hover:text-gray-200 relative group font-medium">
                        Home
                        <span class="absolute left-0 bottom-0 h-0.5 w-0 bg-gray-200 transition-all duration-300 group-hover:w-full"></span>
                    </a>
                    <a href="#features" class="hover:text-gray-200 relative group font-medium">
                        Features
                        <span class="absolute left-0 bottom-0 h-0.5 w-0 bg-gray-200 transition-all duration-300 group-hover:w-full"></span>
                    </a>
                    <a href="#faq" class="hover:text-gray-200 relative group font-medium">
                        Help
                        <span class="absolute left-0 bottom-0 h-0.5 w-0 bg-gray-200 transition-all duration-300 group-hover:w-full"></span>
                    </a>
                    <a href="#contact" class="hover:text-gray-200 relative group font-medium">
                        Contact
                        <span class="absolute left-0 bottom-0 h-0.5 w-0 bg-gray-200 transition-all duration-300 group-hover:w-full"></span>
                    </a>
                    <a href="@Url.Action("Login", "Auth")" class="hover:text-yellow-300 relative group font-medium">
                        Login
                        <span class="absolute left-0 bottom-0 h-0.5 w-0 bg-yellow-300 transition-all duration-300 group-hover:w-full"></span>
                    </a>
                    <a href="@Url.Action("Register", "Auth")" class="hover:text-yellow-300 relative group font-medium">
```

APPENDIX

Controllers/UserController

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Online_Help_Desk.Models;

namespace Online_Help_Desk.Controllers
{
    1 reference
    public class UserController : Controller
    {
        private readonly ApplicationDbContext _context;

        0 references
        public UserController(ApplicationDbContext context)
        {
            _context = context;
        }

        5 references
        private bool IsEndUser()
        {
            return HttpContext.Session.GetString("Role") == RoleEnum.EndUser.ToString();
        }

        5 references
        private int GetUserId()
        {
            return HttpContext.Session.GetInt32("UserId") ?? 0;
        }

        // USER DASHBOARD
        0 references
        public IActionResult Dashboard()
        {
            if (!IsEndUser()) return RedirectToAction("Login", "Auth");

            int uid = GetUserId();
            var user = _context.Users.FirstOrDefault(u => u.UserId == uid);
            HttpContext.Session.SetString("Username", user?.FullName ?? "User");
            var currentUser = _context.Users.FirstOrDefault(u => u.UserId == uid);
            ViewBag.Username = currentUser?.FullName ?? "User";


            var userRequests = _context.Requests
                .Include(r => r.Facility)
```

APPENDIX

Controllers/AdminController

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Online_Help_Desk.Models;

namespace Online_Help_Desk.Controllers
{
    1 reference
    public class AdminController : Controller
    {
        private readonly ApplicationDbContext _context;

        0 references
        public AdminController(ApplicationDbContext context)
        {
            _context = context;
        }
        // Save Admin Role
        8 references
        private bool IsAdmin()
        {
            return HttpContext.Session.GetString("Role") == RoleEnum.Admin.ToString();
        }
        // Dashboard
        0 references
        public IActionResult Dashboard()
        {
            if (!IsAdmin()) return RedirectToAction("Login", "Auth");

            var recentRequests = _context.Requests
                .Include(r => r.User)
                .Include(r => r.Facility)
                .Include(r => r.AssignedToUser)
                .OrderByDescending(r => r.CreatedAt)
                .Take(10)
                .ToList();

            ViewBag.RecentRequests = recentRequests;
            ViewBag.TotalUsers = _context.Users.Count();
            ViewBag.ActiveRequests = _context.Requests.Count(r => r.Status != RequestStatus.Closed);
            ViewBag.PendingApprovals = _context.ApprovalRequests.Count(a => !a.IsApproved && !a.IsRejected);
            ViewBag.ResolvedIssues = _context.Requests.Count(r => r.Status == RequestStatus.Closed);

            return View();
        }
```

APPENDIX

Controllers/AssigneeController

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Online_Help_Desk.Models;

namespace Online_Help_Desk.Controllers
{
    1 reference
    public class AssigneeController : Controller
    {
        private readonly ApplicationDbContext _context;
        0 references
        public AssigneeController(ApplicationDbContext context) => _context = context;

        // Save Assignee Role
        5 references
        private bool IsAssignee()
        {
            return HttpContext.Session.GetString("Role") == RoleEnum.Assignee.ToString();
        }

        4 references
        private int GetUserId()
        {
            return HttpContext.Session.GetInt32("UserId") ?? 0;
        }

        // Dashboard for Assignee
        0 references
        public IActionResult Dashboard()
        {
            if (!IsAssignee()) return RedirectToAction("Login", "Auth");

            int uid = GetUserId();
            var currentUser = _context.Users.FirstOrDefault(u => u.UserId == uid);
            ViewBag.Username = currentUser?.FullName ?? "Assignee";

            var assigned = _context.Requests
            .Include(r => r.User)
            .Include(r => r.Facility)
            .Where(r => r.AssignedToUserId == uid);

            ViewBag.TotalTasks = assigned.Count();
            ViewBag.InProgress = assigned.Count(r => r.Status == RequestStatus.InProgress);
            ViewBag.Pending = assigned.Count(r => r.Status == RequestStatus.Assigned);
```

APPENDIX

Controllers/AuthController

```
using Microsoft.AspNetCore.Mvc;
using Online_Help_Desk.Models;

namespace Online_Help_Desk.Controllers
{
    1 reference
    public class AuthController : Controller
    {
        private readonly ApplicationDbContext _context;
        0 references
        public AuthController(ApplicationDbContext context) => _context = context;

        0 references
        public IActionResult Login() => View();

        [HttpPost]
        0 references
        public IActionResult Login(string uemail, string upassword)
        {
            var user = _context.Users.FirstOrDefault(u => u.Email == uemail && u.PasswordHash == upassword && u.IsActive);
            if (user != null)
            {
                HttpContext.Session.SetInt32("UserId", user.UserId);
                HttpContext.Session.SetString("Role", user.Role.ToString());
                HttpContext.Session.SetString("FullName", user.FullName);

                return user.Role switch
                {
                    RoleEnum.Admin => RedirectToAction("Dashboard", "Admin"),
                    RoleEnum.EndUser => RedirectToAction("Dashboard", "User"),
                    RoleEnum.FacilityHead => RedirectToAction("Dashboard", "FacilityHead"),
                    RoleEnum.Assignee => RedirectToAction("Dashboard", "Assignee"),
                    _ => RedirectToAction("Login")
                };
            }
            ViewBag.Error = "Invalid credentials or not approved.";
            return View();
        }

        0 references
        public IActionResult Register() => View();

        [HttpPost]
        0 references
        public IActionResult Register(User model, RoleEnum requestedRole, string reason)
```

APPENDIX

Controllers/FacilityHeadController

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Online_Help_Desk.Models;

namespace Online_Help_Desk.Controllers
{
    1 reference
    public class FacilityHeadController : Controller
    {
        private readonly ApplicationDbContext _context;
        0 references
        public FacilityHeadController(ApplicationDbContext context) => _context = context;

        3 references
        private bool IsFacilityHead()
        {
            return HttpContext.Session.GetString("Role") == RoleEnum.FacilityHead.ToString();
        }

        3 references
        private int GetUserId()
        {
            return HttpContext.Session.GetInt32("UserId") ?? 0;
        }

        // Dashboard
        0 references
        public IActionResult Dashboard()
        {
            if (!IsFacilityHead()) return RedirectToAction("Login", "Auth");

            int uid = GetUserId();
            var currentUser = _context.Users.FirstOrDefault(u => u.UserId == uid);
            ViewBag.Username = currentUser?.FullName ?? "Facility Head";

            var assignedRequests = _context.Requests
                .Include(r => r.User)
                .Include(r => r.Facility)
                .OrderByDescending(r => r.CreatedAt)
                .ToList();

            ViewBag.TotalAssigned = assignedRequests.Count;
            ViewBag.PendingCount = assignedRequests.Count(r => r.Status == RequestStatus.Assigned);
            ViewBag.InProgressCount = assignedRequests.Count(r => r.Status == RequestStatus.InProgress);
```

APPENDIX

ApplicationDbContext

```
using Microsoft.EntityFrameworkCore;

namespace Online_Help_Desk.Models
{
    15 references
    public class ApplicationDbContext : DbContext
    {
        0 references
        public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options) : base(options) { }

        36 references
        public DbSet<User> Users { get; set; }
        7 references
        public DbSet<Facility> Facilities { get; set; }
        26 references
        public DbSet<Request> Requests { get; set; }
        2 references
        public DbSet<StatusHistory> StatusHistories { get; set; }
        10 references
        public DbSet<ApprovalRequest> ApprovalRequests { get; set; }

        0 references
        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            base.OnModelCreating(modelBuilder);

            // Prevent cascade delete on UpdatedByUser → User (to avoid multiple cascade paths)
            modelBuilder.Entity<StatusHistory>()
                .HasOne(sh => sh.UpdatedByUser)
                .WithMany()
                .HasForeignKey(sh => sh.UpdatedByUserId)
                .OnDelete(DeleteBehavior.Restrict);

            // Prevent cascade delete on AssignedToUser → User (same reason)
            modelBuilder.Entity<Request>()
                .HasOne(r => r.AssignedToUser)
                .WithMany()
                .HasForeignKey(r => r.AssignedToUserId)
                .OnDelete(DeleteBehavior.Restrict);
        }


    }
}
```

APPENDIX

Models/ApprovalRequest

```
using Online_Help_Desk.Models;
using System.ComponentModel.DataAnnotations;

namespace Online_Help_Desk.Models
{

    public class ApprovalRequest
    {
        [Key]

        public int ApprovalRequestId { get; set; }

        [Required]

        public int UserId { get; set; }

        public User? User { get; set; }

        [Required]

        public RoleEnum RequestedRole { get; set; }


        public bool IsApproved { get; set; } = false;

        public bool IsRejected { get; set; } = false;


        public DateTime RequestedAt { get; set; } = DateTime.Now;
        2 references
        public DateTime? ActionTakenAt { get; set; }
    }
}
```

APPENDIX

Models/Facility

APPENDIX

Models/Request

```
using System.ComponentModel.DataAnnotations;

namespace Online_Help_Desk.Models
{

        34 references
        public enum RequestStatus
        {
            Pending = 0,
            Assigned = 1,
            InProgress = 2,
            Closed = 3
        }


    10 references
    public class Request
    {
        [Key]
        10 references
        public int RequestId { get; set; }

        [Required]
        [StringLength(150)]
        3 references
        public string? Title { get; set; }

        1 reference
        public string? Description { get; set; }

        [Required]
        30 references
        public RequestStatus Status { get; set; }

        [Required]
        4 references
        public int UserId { get; set; }
        9 references
        public User User { get; set; }

        [Required]
        2 references
        public int FacilityId { get; set; }
        12 references
        public Facility? Facility { get; set; }
```

APPENDIX

Models/User

```
using System.ComponentModel.DataAnnotations;

namespace Online_Help_Desk.Models
{
    24 references
    public enum RoleEnum
    {
        Admin = 0,
        EndUser = 1,
        FacilityHead = 2,
        Assignee = 3
    }
    22 references
    public class User
    {
        [Key]
        40 references
        public int UserId { get; set; }

        [Required]
        [StringLength(100)]
        30 references
        public string? FullName { get; set; }

        [Required]
        [StringLength(50)]
        12 references
        public string? Username { get; set; }

        [Required]
        [EmailAddress]
        13 references
        public string? Email { get; set; }

        [Required]
        9 references
        public string? PasswordHash { get; set; }

        [Required]
        12 references
        public RoleEnum Role { get; set; }

        6 references
        public bool IsActive { get; set; } = false;
```

APPENDIX

Program.cs

```
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.FileSystemGlobbing.Internal.Patterns;
using Microsoft.Extensions.Options;
using Online_Help_Desk.Models;

var builder = WebApplication.CreateBuilder(args);
builder.Services.AddMvc();
builder.Services.AddSession();
builder.Services.AddDbContext<ApplicationDbContext>(Options =>
Options.UseSqlServer(builder.Configuration.GetConnectionString("mycon")));
var app = builder.Build();

//app.MapGet("/", () => "Hello World!");
app.MapControllerRoute(
    name: "default",
    pattern: "{Controller=Home}/{action=Index}"
    );

app.UseStaticFiles();
app.UseSession();
app.Run();
```

APPENDIX

appsettings.json

APPENDIX

View/Auth/Login.cshtml

```
</head>
<body>

    <div class="main-wrapper login-body">
        <div class="login-wrapper">
            <div class="container">
                <div class="loginbox">
                    <div class="login-left">
                        <img class="img-fluid" src="~/Auth/assets/img/login.png" alt="Logo">
                    </div>
                    <div class="login-right">
                        <div class="login-right-wrap">
                            <h1>Welcome to Online Help Desk</h1>
                            <p class="account-subtitle">Need an account? <a href="@Url.Action("Register","Auth")">Sign Up</a></p>
                            <h2>Sign in</h2>

                            <form action="/Auth/Login" method="post">
                                <div class="form-group">
                                    <label>Email <span class="login-danger">*</span></label>
                                    <input class="form-control" type="text" required name="uemail">
                                    <span class="profile-views"><i class="fas fa-user-circle"></i></span>
                                </div>
                                <div class="form-group">
                                    <label>Password <span class="login-danger">*</span></label>
                                    <input class="form-control pass-input" type="text" required name="upassword">
                                    <span class="profile-views feather-eye toggle-password"></span>
                                </div>
                                @ViewBag.Error
                                <div class="forgotpass" style="margin: 3px;">
                                    <div class="remember-me">
                                        <label class="custom_check mr-2 mb-0 d-inline-flex remember-me">
                                            Remember me
                                            <input type="checkbox" name="radio">
                                            <span class="checkmark"></span>
                                        </label>
                                    </div>
                                    <a href="">Forgot Password?</a>
                                </div>
                                <div class="form-group">
                                    <button class="btn btn-primary btn-block" type="submit">Login</button>
                                </div>
                            </form>
```

APPENDIX

View/Auth/Register.cshtml

```
<img class="img-fluid" src="~/Auth/assets/img/login.png" alt="Logo">
</div>
<div class="login-right">
    <div class="login-right-wrap">
        <h1>Sign Up</h1>
        <p class="account-subtitle">Enter details to create your account</p>

        <form method="post" action="/Auth/Register" onsubmit="return validatePasswords();">
            <div class="form-group">
                <label>Full Name <span class="login-danger">*</span></label>
                <input class="form-control" type="text" required name="FullName">
            </div>

            <div class="form-group">
                <label>Username <span class="login-danger">*</span></label>
                <input class="form-control" type="text" required name="Username">
            </div>

            <div class="form-group">
                <label>Email <span class="login-danger">*</span></label>
                <input class="form-control" type="email" required name="Email">
            </div>

            <div class="form-group">
                <label>Register As <span class="login-danger">*</span></label>
                <select class="form-control" required name="requestedRole">
                    <option value="">-- Select Role --</option>
                    <option value="1">End User</option>
                    <option value="2">Facility Head</option>
                    <option value="3">Assignee</option>
                </select>
            </div>

            <div class="form-group">
                <label>Password <span class="login-danger">*</span></label>
                <input class="form-control pass-input" type="password" required name="PasswordHash" id="password">
            </div>

            <div class="form-group">
                <label>Confirm password <span class="login-danger">*</span></label>
                <input class="form-control pass-confirm" type="password" required id="confirmPassword">
                <span id="passwordError" class="text-danger" style="display:none;">Passwords do not match</span>
            </div>
```

TECHNOLOGY STACK USED

The development of the Online Help Desk (OHD) System utilized a robust and scalable technology stack to ensure performance, reliability, and ease of maintenance:

1. FRONTEND:

- HTML5, CSS3, JavaScript – Core technologies for UI structure and interactivity.
- Tailwind CSS – Utility-first framework for responsive, modern design.
- Bootstrap 5 (selective) – For components like accordions and modals.

2. BACKEND:

- ASP.NET MVC (C#) – Structured server-side logic with Model-View-Controller architecture.
- Entity Framework Core – For ORM-based database interactions.

3. DATABASE:

- Microsoft SQL Server – Relational database for structured data and efficient queries.

4. DEVELOPMENT TOOLS:

- Visual Studio 2022 – Primary IDE for backend development.
- Canva – For UI wireframes, design assets, and documentation layout

DATABASE TABLE DESIGN

The OHD system's database is designed to ensure efficient data normalization, relational integrity, and scalability. The schema is organized around core entities such as Users, Requests, Facilities, and their relationships. Below is a high-level overview of key tables and their attributes:

1. USERS TABLE

Stores system users, including Admins, End Users, Facility Heads, and Assignees.

Column	Data Type	Description
UserId	INT (PK)	Unique identifier for the user
FullName	VARCHAR(100)	Full name of the user
Username	VARCHAR(50)	Login username
Email	VARCHAR(100)	Email address
PasswordHash	VARCHAR(255)	Hashed password
Role	ENUM	User role (Admin, EndUser, etc.)
IsActive	BOOLEAN	Account status
DateCreated	DATETIME	Registration timestamp

DATABASE TABLE DESIGN

The OHD system's database is designed to ensure efficient data normalization, relational integrity, and scalability. The schema is organized around core entities such as Users, Requests, Facilities, and their relationships. Below is a high-level overview of key tables and their attributes:

2. REQUESTS TABLE

Manages help desk complaints/requests submitted by end users.

Column	Data Type	Description
RequestId	INT (PK)	Unique ID of the request
Title	VARCHAR(150)	Brief summary of the issue
Description	TEXT	Detailed explanation
Status	ENUM	Request status (Pending, Assigned, etc.)
UserId	INT (FK)	Foreign key to Users (requester)
FacilityId	INT (FK)	Foreign key to Facilities
AssignedToUserId	INT (FK)	Foreign key to Assignee (nullable)
CreatedAt	DATETIME	Submission time
UpdatedAt	DATETIME	Last status update

DATABASE TABLE DESIGN

The OHD system's database is designed to ensure efficient data normalization, relational integrity, and scalability. The schema is organized around core entities such as Users, Requests, Facilities, and their relationships. Below is a high-level overview of key tables and their attributes:

3. FACILITIES TABLE

Lists all available facilities where issues can be reported.

Column	Data Type	Description
FacilityId	INT (PK)	Unique facility ID
Name	VARCHAR(100)	Name of the facility
Location	VARCHAR(100)	Physical or logical location

RELATIONAL OVERVIEW

- One User can submit multiple Requests
- One Facility Head manages Requests for their assigned Facility
- One Assignee can handle multiple Requests
- Each Request belongs to one Facility

DATABASE SCHEMA DESIGN

The OHD system's database is designed to ensure efficient data normalization, relational integrity, and scalability. The schema is organized around core entities such as Users, Requests, Facilities, and their relationships. Below is a high-level overview of key tables and their attributes:

TESTING SUMMARY

The Online Help Desk (OHD) system was thoroughly tested to ensure functionality, performance, and reliability across all modules. The testing process followed structured unit tests, integration checks, and user acceptance tests. Each role (Admin, End User, Facility Head, Assignee) was validated against real-world use cases to confirm seamless navigation, data integrity, and consistent role-based access control.

Key Testing Goals:

- Ensure all user actions reflect correct status and updates in real-time.
- Verify secure login, session management, and role-based redirection.
- Check responsiveness across mobile, tablet, and desktop views.
- Validate CRUD operations for users, requests, and facilities.

UNIT TEST CHECKLIST

Module	Test Case Description	Status
Login / Auth System	Validate correct and incorrect credentials	✅ Passed
Role Redirection	Ensure correct landing page for each role	✅ Passed
Submit Request	User creates a new ticket with valid fields	✅ Passed
Track Request	Request status updates as per assigned progress	✅ Passed
Assign Request	Facility Head assigns requests to correct assignees	✅ Passed
Status Change (Assignee)	Assignee updates request from Assigned → Closed	✅ Passed
Profile Update	Validate changes in profile and password functionality	✅ Passed
Admin Dashboard	View all users, facilities, and perform CRUD operations	✅ Passed
Export Feature	Export to PDF/Excel works with formatted data	✅ Passed
Responsive UI	Test layout on various screen sizes	✅ Passed

SECURITY MEASURES

The Online Help Desk (OHD) system has been designed with robust security practices to ensure data confidentiality, integrity, and authorized access. Key security protocols implemented across the platform include:

Role-Based Access Control (RBAC)

Only authenticated users can access the system, and actions are strictly limited based on user roles (Admin, End User, Facility Head, Assignee). Unauthorized role access is redirected to the login page.

Session Management

User sessions are securely handled using encrypted session tokens. Idle or expired sessions are automatically invalidated to prevent unauthorized access.

Input Validation & Sanitization

All forms and inputs are validated server-side to prevent SQL Injection, XSS, and other injection attacks. Inputs are sanitized before processing or displaying.

Database Access Control

Direct database access is restricted. Only the application server has permission to interact with the database using parameterized queries and ORM practices.

Audit Trails (Basic Logging)

Critical actions such as login, request creation, assignment, and status changes are logged for accountability and traceability.

These layered security measures collectively ensure that the OHD system maintains high trustworthiness while protecting user data and system functionality.

LIMITATIONS & FUTURE ENHANCEMENTS

Current Limitations

Despite the functional completeness of the OHD (Online Help Desk) system, a few limitations exist in the current version:

- **No Email or SMS Notifications:** Users do not receive automated alerts upon status changes or assignments.
- **Manual Admin Intervention:** Forgot password functionality and profile recovery still require administrator support.
- **Limited Reporting & Analytics:** While basic task charts are provided, detailed trend analysis and exportable reports are minimal.
- **No File Attachments:** Users cannot currently upload images or documents with their requests (e.g., photos of faulty equipment).
- **Single Language Interface:** The system is currently English-only, limiting accessibility for some local users.

LIMITATIONS & FUTURE ENHANCEMENTS

Planned Future Enhancements

To address these limitations and improve user experience, the following enhancements are proposed for upcoming versions:

- Real-Time Email/SMS Notifications for all major actions (status updates, assignments, etc.).
- Password Recovery Workflow with OTP-based reset and multi-factor authentication (MFA).
- Advanced Analytics Dashboard with export options, filtering, and visual insights.
- File Upload Support during complaint submission for better context and evidence.
- Multi-language Support to improve accessibility across diverse user bases.
- Mobile App Integration for iOS and Android for on-the-go complaint tracking.

FINAL CHECKLIST

Item	Remarks	Status
Problem Definition Clearly Stated	Well-defined and aligned with real needs	✓ Completed
Objectives Clearly Defined	Specific, measurable, and achievable	✓ Completed
User Roles and Functionalities Implemented	Admin, End User, Facility Head, Assignee	✓ Completed
System Architecture Documented	Diagram + explanation included	✓ Completed
ER Diagram & Database Schema Designed	Covers all key entities and relations	✓ Completed
Use Case Diagrams Prepared	Graphical representation for all user flows	✓ Completed
Testing Checklist Created	Functional & unit test coverage ensured	✓ Completed
Security Features Considered	Sessions, role-checks, limited access, etc.	✓ Completed
UI Screenshots Captured	All main interfaces documented	✓ Completed
Project is Deployable	Can run locally or on institution's server	✓ Completed