Unicom TIC
Training & Innovation Centre
Skills Today, Success Tomorrow

Unicom TIC Management System - Project Report Submission

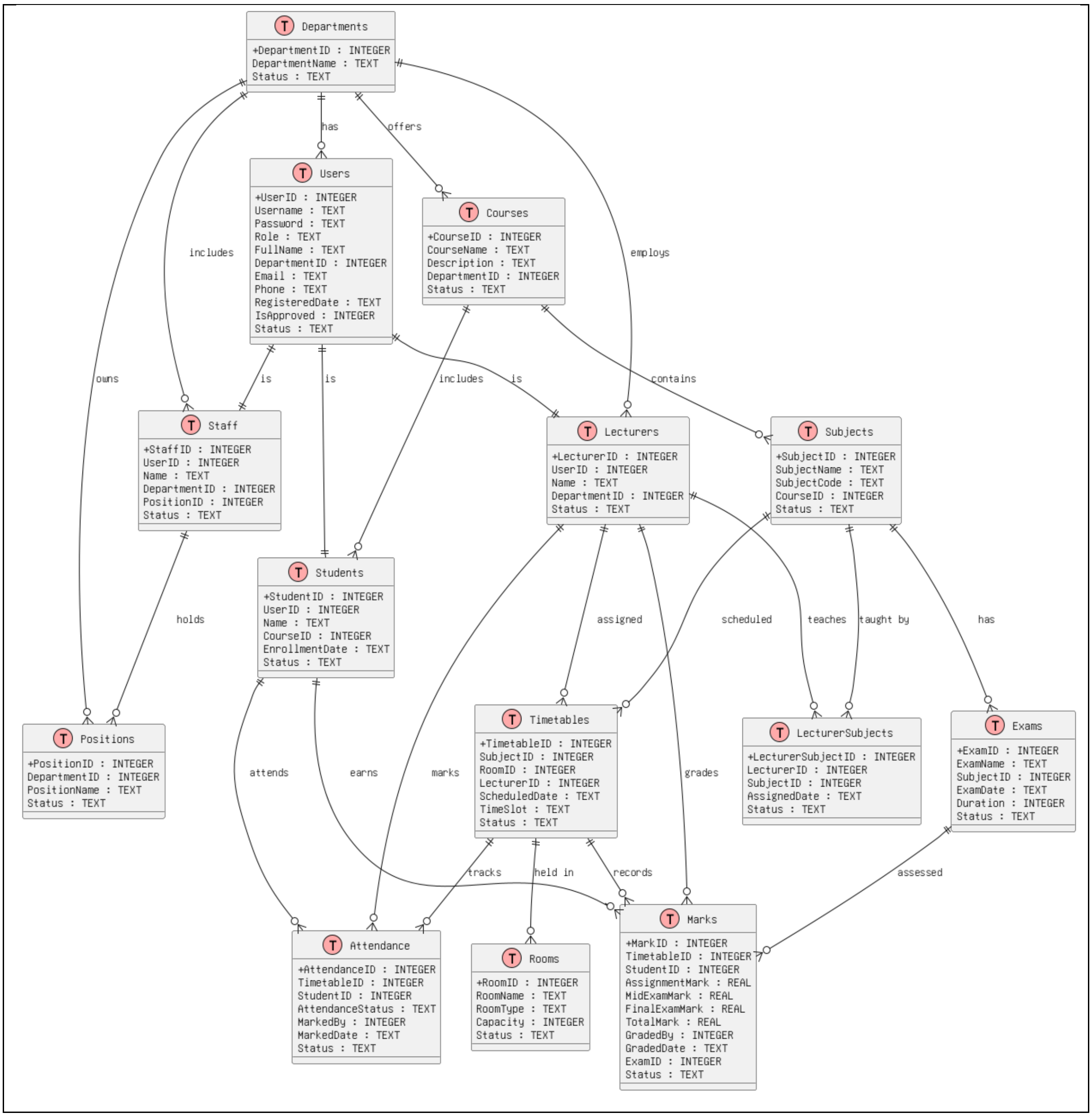
Table of Contents

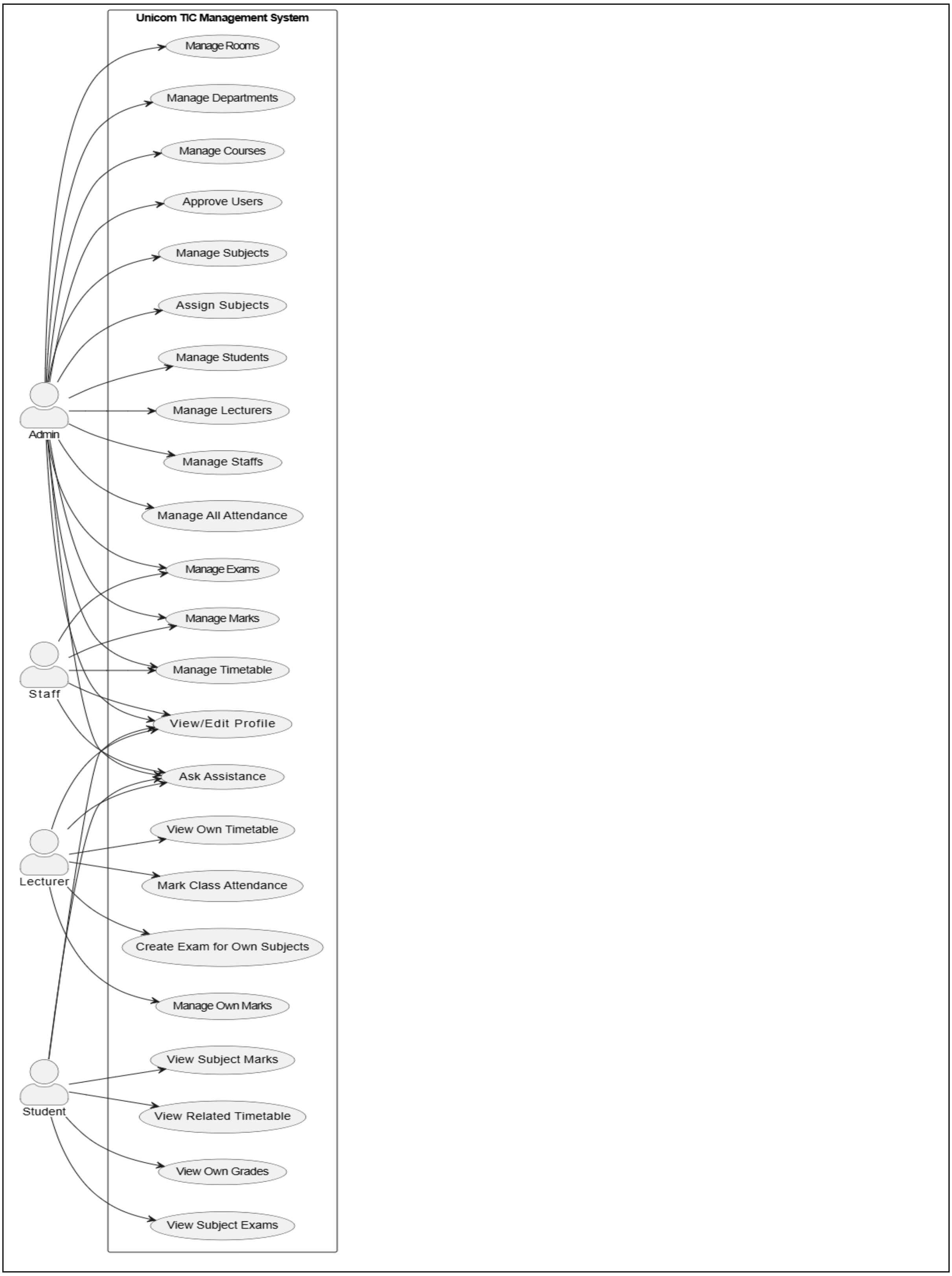
1. Project Overview

Key Features Implemented:

- Role-based login and access control (Admin, Lecturer, Staff, Student)
- Student registration with admin approval process
- Subject and course management with lecturer assignments
- Timetable creation with room allocation (Lab/Hall)
- Marks management (Total)
- Attendance management per subject and timetable
- **Email integration using MailKit for registration and approval**
- **GPT-3.5 Turbo-based chatbot for 'Ask Assistance' module via OpenRouter API**
- Profile management (edit/view based on role)
- **Secure password hashing with BCrypt**
- SQLite database integration with **soft delete** functionality
- Admin approval workflow for student registration
- Ask Assistance support for all roles
- Students and lecturers can edit their own profiles
- Staff can manage exams, marks and view all student data
- Admin has full access to all features and data

Diagrams


Departments
+DepartmentID : INTEGER
DepartmentName : TEXT
Status : TEXT
has
offers
includes
employs
owns
Users
+UserID : INTEGER
Username : TEXT
Password : TEXT
Role : TEXT
FullName : TEXT
DepartmentID : INTEGER
Email : TEXT
Phone : TEXT
RegisteredDate : TEXT
IsApproved : INTEGER
Status : TEXT
Courses
+CourseID : INTEGER
CourseName : TEXT
Description : TEXT
DepartmentID : INTEGER
Status : TEXT
is
is
includes
is
contains
Staff
+StaffID : INTEGER
UserID : INTEGER
Name : TEXT
DepartmentID : INTEGER
PositionID : INTEGER
Status : TEXT
Lecturers
+LecturerID : INTEGER
UserID : INTEGER
Name : TEXT
DepartmentID : INTEGER
Status : TEXT
Subjects
+SubjectID : INTEGER
SubjectName : TEXT
SubjectCode : TEXT
CourseID : INTEGER
Status : TEXT
Students
+StudentID : INTEGER
UserID : INTEGER
Name : TEXT
CourseID : INTEGER
EnrollmentDate : TEXT
Status : TEXT
holds
assigned
scheduled
teaches
taught by
has
Positions
+PositionID : INTEGER
DepartmentID : INTEGER
PositionName : TEXT
Status : TEXT
Timetables
+TimetableID : INTEGER
SubjectID : INTEGER
RoomID : INTEGER
LecturerID : INTEGER
ScheduledDate : TEXT
TimeSlot : TEXT
Status : TEXT
LecturerSubjects
+LecturerSubjectID : INTEGER
LecturerID : INTEGER
SubjectID : INTEGER
AssignedDate : TEXT
Status : TEXT
Exams
+ExamID : INTEGER
ExamName : TEXT
SubjectID : INTEGER
ExamDate : TEXT
Duration : INTEGER
Status : TEXT
attends
earns
marks
grades
tracks
held in
records
assessed
Attendance
+AttendanceID : INTEGER
TimetableID : INTEGER
StudentID : INTEGER
AttendanceStatus : TEXT
MarkedBy : INTEGER
MarkedDate : TEXT
Status : TEXT
Rooms
+RoomID : INTEGER
RoomName : TEXT
RoomType : TEXT
Capacity : INTEGER
Status : TEXT
Marks
+MarkID : INTEGER
TimetableID : INTEGER
StudentID : INTEGER
AssignmentMark : REAL
MidExamMark : REAL
FinalExamMark : REAL
TotalMark : REAL
GradedBy : INTEGER
GradedDate : TEXT
ExamID : INTEGER
Status : TEXT


Unicom TIC Management System
Manage Rooms
Manage Departments
Manage Courses
Approve Users
Manage Subjects
Assign Subjects
Manage Students
Manage Lecturers
Manage Staffs
Manage All Attendance
Manage Exams
Manage Marks
Manage Timetable
View/Edit Profile
Ask Assistance
View Own Timetable
Mark Class Attendance
Create Exam for Own Subjects
Manage Own Marks
View Subject Marks
View Related Timetable
View Own Grades
View Subject Exams
Admin
Staff
Lecturer
Student

Technologies Used

• Language: C# 7.3 (WinForms)

• Architecture: MVC (Controller-Service-Repository)

• Database: SQLite

• APIs: OpenAI GPT-3.5 Turbo (via OpenRouter API), MailKit for email communication

• Security: BCrypt password hashing, **appsettings.json** for API key handling

Challenges Faced and Solutions

• Async integration in WinForms: Addressed using async/await in service, Controller and Repository layers.

• Real-time email and chatbot integration: Handled via HttpClient and MailKit async calls.

• Role-based feature segregation: Used conditional logic across views and dynamic UI controls.

• Data consistency and relationship management: Resolved using foreign keys and soft delete constraints in SQLite.

• Exposing API Key risk: API keys managed via **appsettings.json** .

User Roles and Their Powers (Permissions)

Admin

- Full access to the entire system
- Create/edit/delete: Students, Lecturers, Staff, Courses, Departments, Subjects, Rooms
- Approve newly registered students and staff
- Assign subjects to lecturers
- Allocate lecture halls/labs via timetable module
- Create exams and assign marks
- Manage all attendance records
- Send approval/welcome emails

Lecture

- View only assigned subjects
- Mark attendance for students based on timetable
- Add/edit/delete marks for their assigned subjects
- View exam schedules and student performance
- Access own timetable
- Edit their profile
- Ask questions via Ask Assistance (ChatGPT)

Staff

- View their timetable with subject, time, and room info
- View marks for all exams (assignment, mid, final)
- View/edit their profile (limited to own data)
- Ask for support via the chatbot (GPT-based Ask Assistance)

Student

- View their timetable with subject, time, and room info
- View marks for all exams (assignment, mid, final)
- View/edit their profile (limited to own data)
- Ask for support via the chatbot (GPT-based Ask Assistance)

How the App Works – Functional Flow

1. Login Flow

- All users enter their username & password.
- Role is determined from the database (Users table).
- Based on the role:
- Admin is redirected to AdminDashboard .
- Lecturer to LecturerDashboard (Buttons Will Hide).
- Staff to StaffDashboard (Buttons Will Hide).
- Student to StudentDashboard (Buttons Will Hide).
- Invalid login triggers an error with guidance.

2. Registration & Approval

- New students/staff register via a form.
- Status is set to "Pending" in the database.
- Admin logs in → views pending requests → clicks Approve.
- On approval:
- Status changes to "Active".
- A welcome email is sent using MailKit.
- Approved users can now log in.

3. Course, Subject & Room Management (Admin)

- Admin adds courses (e.g., BSc IT), then subjects under each.
- Admin creates Rooms as Labs or Halls with types.
- Rooms are stored and selectable during timetable creation.

4. Assigning Subjects to Lecturers

- Admin links Subjects → Lecturers.
- Lecturer sees only their assigned subjects when entering marks or attendance.

5. Timetable Management (Admin)

- Admin schedules classes via:
- Subject
- Room (Lab or Hall)
- TimeSlot (e.g., morning,evening)
- Timetables are visible to:
- Students (for their enrolled course)
- Lecturers (for their assigned subjects)

6. Attendance Management (Lecturer)

- Lecturers view timetable → mark attendance for each student.
- Data is saved against:
- StudentID, SubjectID, TimetableID
- MarkedBy (Lecturer), Date

7. Exam & Marks Management

- Admin creates exams with:
- Subject, Date, Duration
- Lecturers/Staff add marks per student per exam.
- Marks split: Total .

8. Student View

- Students
- Their Timetable
- Their Marks
- Their Profile
- Use Ask Assistance

9. Ask Assistance (All Roles)

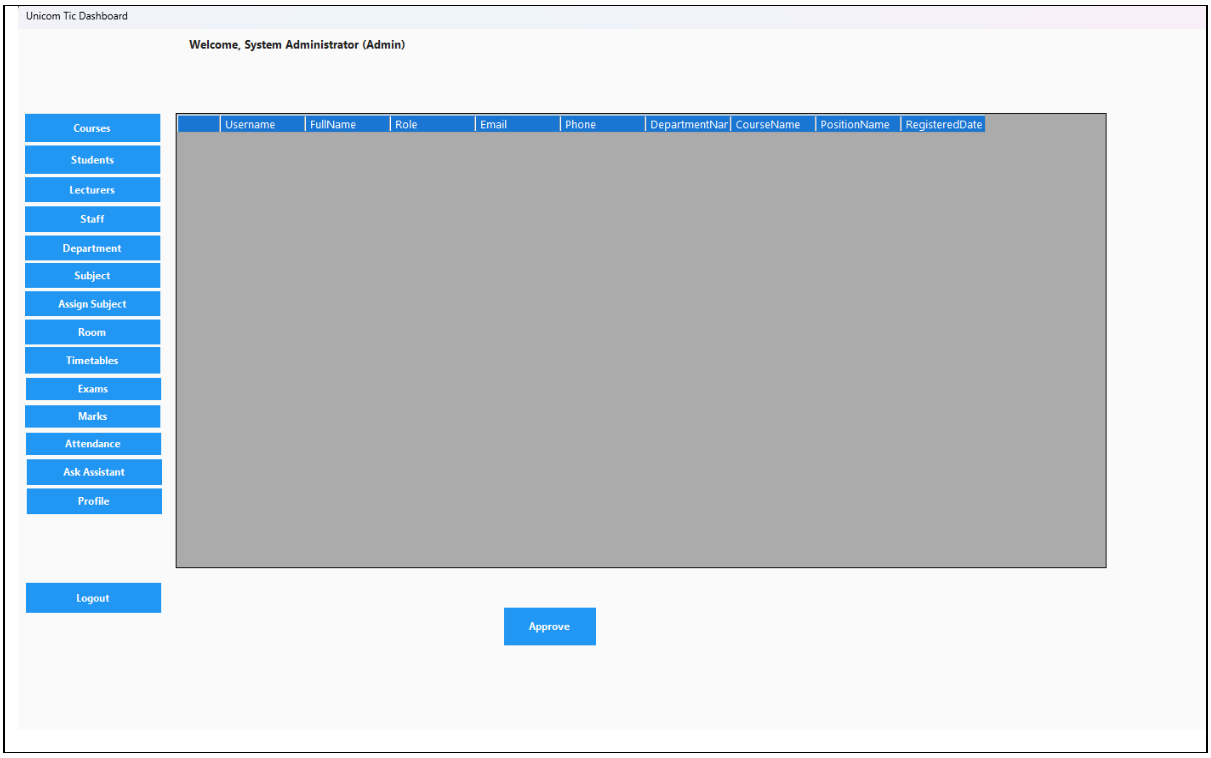
- Built-in chatbot using OpenRouter's GPT-3.5 Turbo
- Securely uses API key stored in config.json
- Users can ask academic/admin questions
- Responses fetched in real-time via async API call

10. Security & Performance

- Passwords hashed with BCrypt.
- All database operations are async, improving responsiveness.
- SQLite is used with soft delete (Status column).

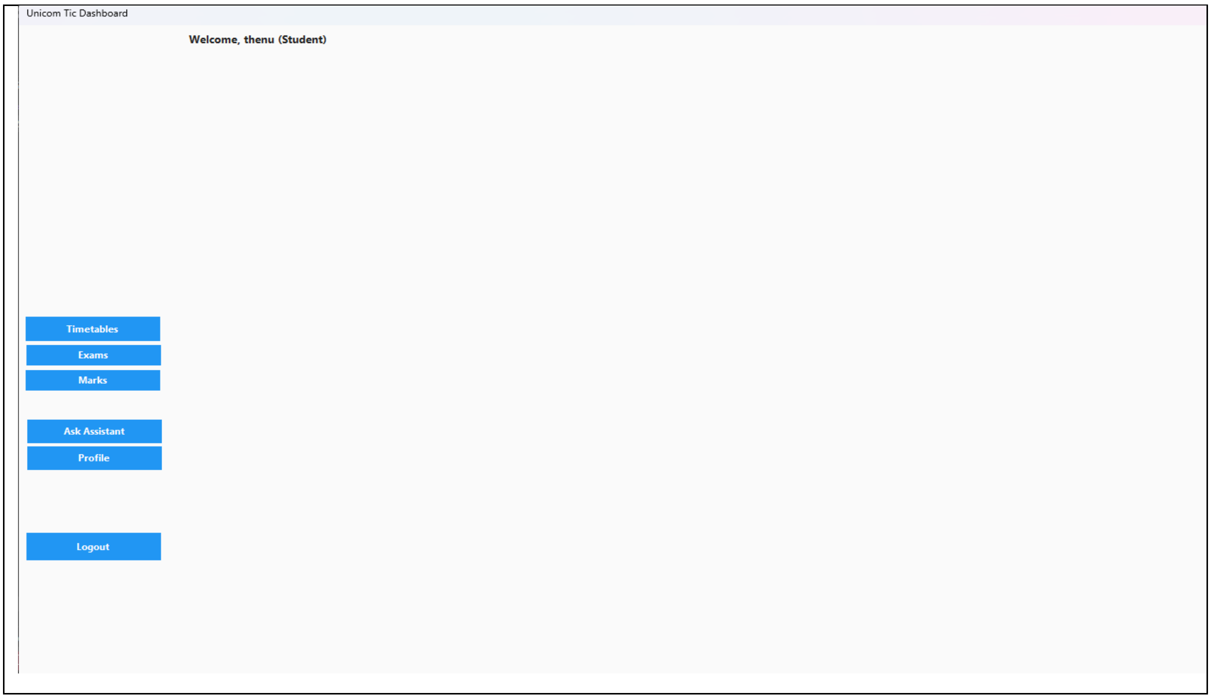
Code Samples

1. Role-Based Dashboard Initialization

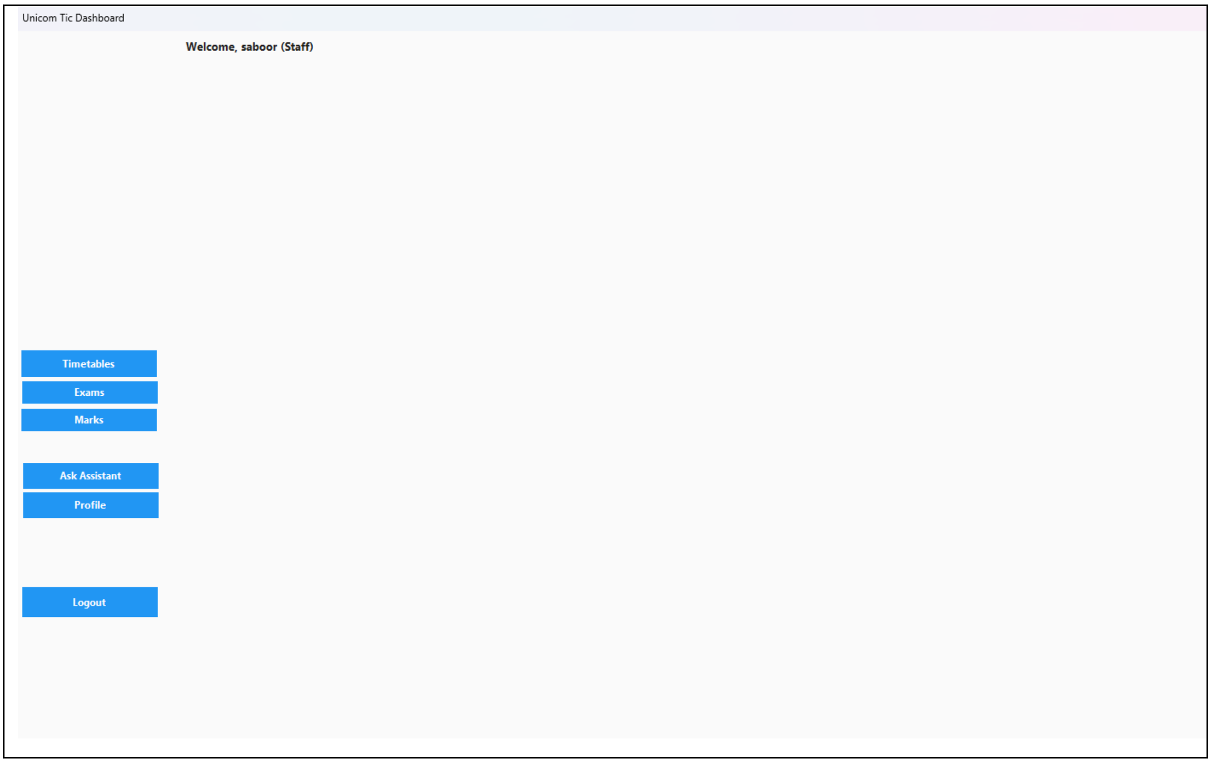


Welcome, System Administrator (Admin)

	Username	FullName	Role	Email	Phone	DepartmentNar	CourseName	PositionName	RegisteredDate

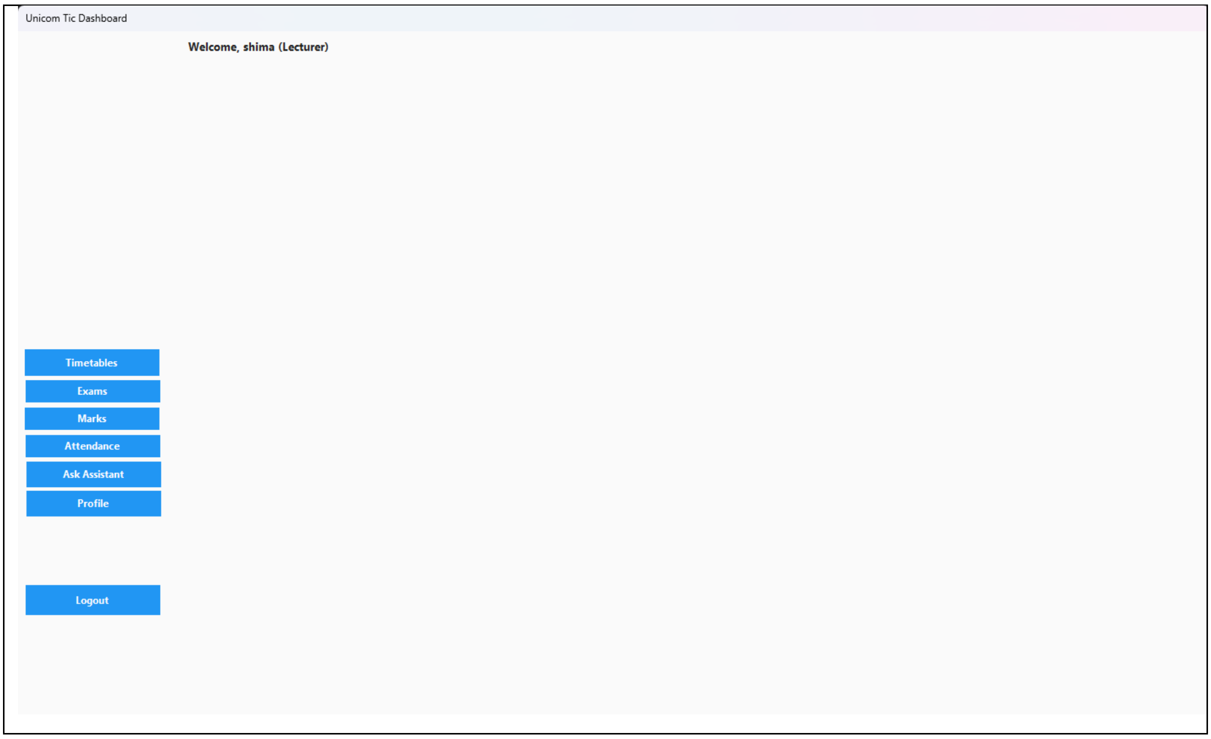


Welcome, thenu (Student)



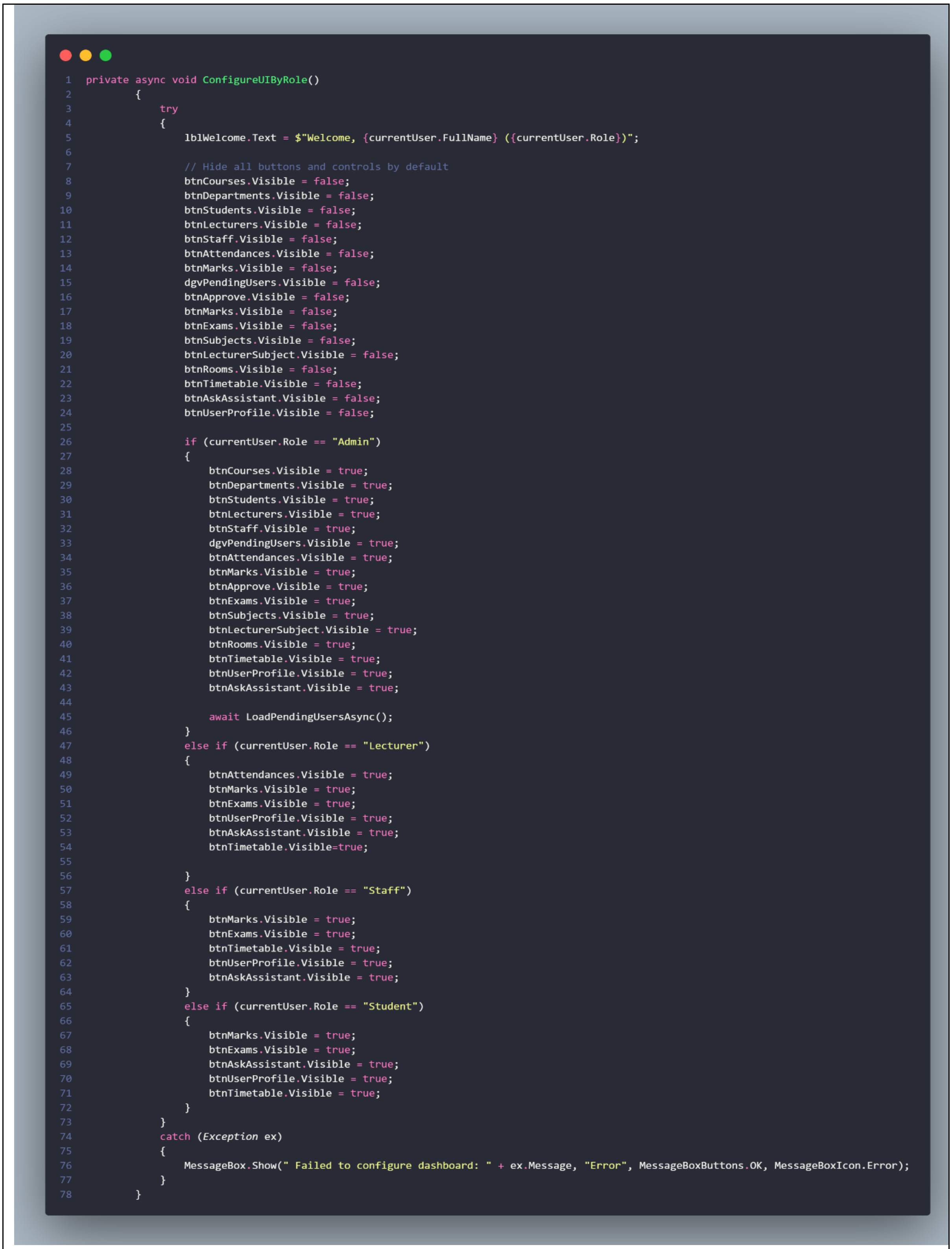
Unicom Tic Dashboard

Welcome, saboor (Staff)



Unicom Tic Dashboard

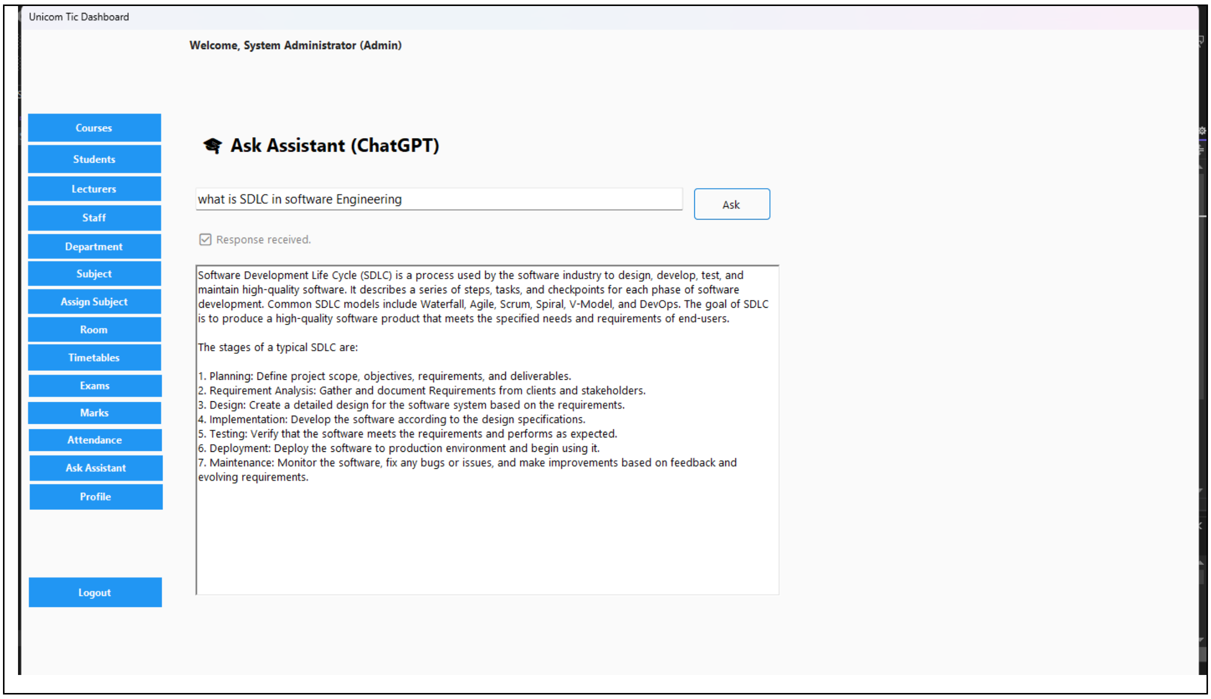
Welcome, shima (Lecturer)



```
private async void ConfigureUIByRole()
        {
            try
            {
                lblWelcome.Text = $"Welcome, {currentUser.FullName} ({currentUser.Role})";

                // Hide all buttons and controls by default
                btnCourses.Visible = false;
                btnDepartments.Visible = false;
                btnStudents.Visible = false;
                btnLecturers.Visible = false;
                btnStaff.Visible = false;
                btnAttendances.Visible = false;
                btnMarks.Visible = false;
                dgvPendingUsers.Visible = false;
                btnApprove.Visible = false;
                btnMarks.Visible = false;
                btnExams.Visible = false;
                btnSubjects.Visible = false;
                btnLecturerSubject.Visible = false;
                btnRooms.Visible = false;
                btnTimetable.Visible = false;
                btnAskAssistant.Visible = false;
                btnUserProfile.Visible = false;

                if (currentUser.Role == "Admin")
                {
                    btnCourses.Visible = true;
                    btnDepartments.Visible = true;
                    btnStudents.Visible = true;
                    btnLecturers.Visible = true;
                    btnStaff.Visible = true;
                    dgvPendingUsers.Visible = true;
                    btnAttendances.Visible = true;
                    btnMarks.Visible = true;
                    btnApprove.Visible = true;
                    btnExams.Visible = true;
                    btnSubjects.Visible = true;
                    btnLecturerSubject.Visible = true;
                    btnRooms.Visible = true;
                    btnTimetable.Visible = true;
                    btnUserProfile.Visible = true;
                    btnAskAssistant.Visible = true;

                    await LoadPendingUsersAsync();
                }
                else if (currentUser.Role == "Lecturer")
                {
                    btnAttendances.Visible = true;
                    btnMarks.Visible = true;
                    btnExams.Visible = true;
                    btnUserProfile.Visible = true;
                    btnAskAssistant.Visible = true;
                    btnTimetable.Visible=true;

                }
                else if (currentUser.Role == "Staff")
                {
                    btnMarks.Visible = true;
                    btnExams.Visible = true;
                    btnTimetable.Visible = true;
                    btnUserProfile.Visible = true;
                    btnAskAssistant.Visible = true;
                }
                else if (currentUser.Role == "Student")
                {
                    btnMarks.Visible = true;
                    btnExams.Visible = true;
                    btnAskAssistant.Visible = true;
                    btnUserProfile.Visible = true;
                    btnTimetable.Visible = true;
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show(" Failed to configure dashboard: " + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
```

Logic to load the correct dashboard depending on login role

2.GPT-3.5 Turbo API Integration



Welcome, System Administrator (Admin)

Ask Assistant (ChatGPT)

what is SDLC in software Engineering

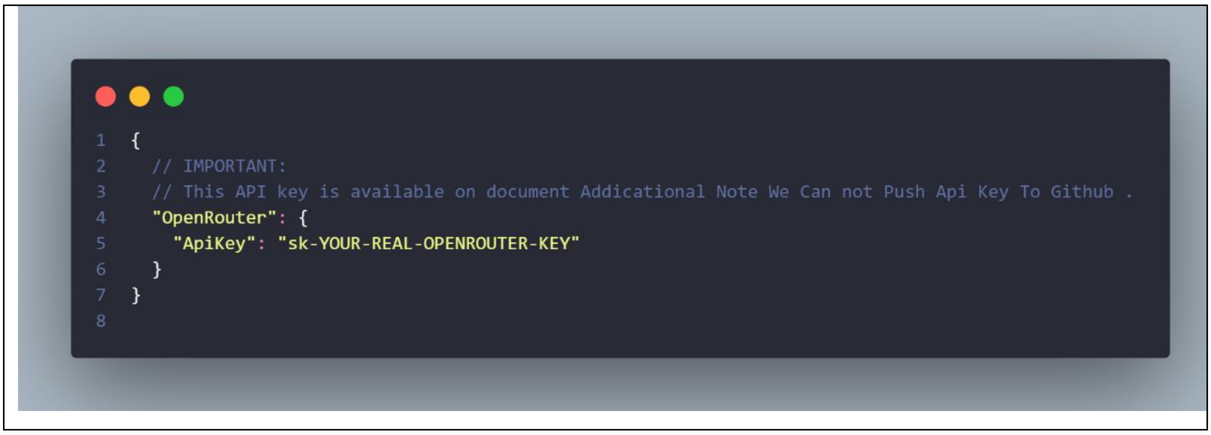
☑ Response received.

Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop, test, and maintain high-quality software. It describes a series of steps, tasks, and checkpoints for each phase of software development. Common SDLC models include Waterfall, Agile, Scrum, Spiral, V-Model, and DevOps. The goal of SDLC is to produce a high-quality software product that meets the specified needs and requirements of end-users.

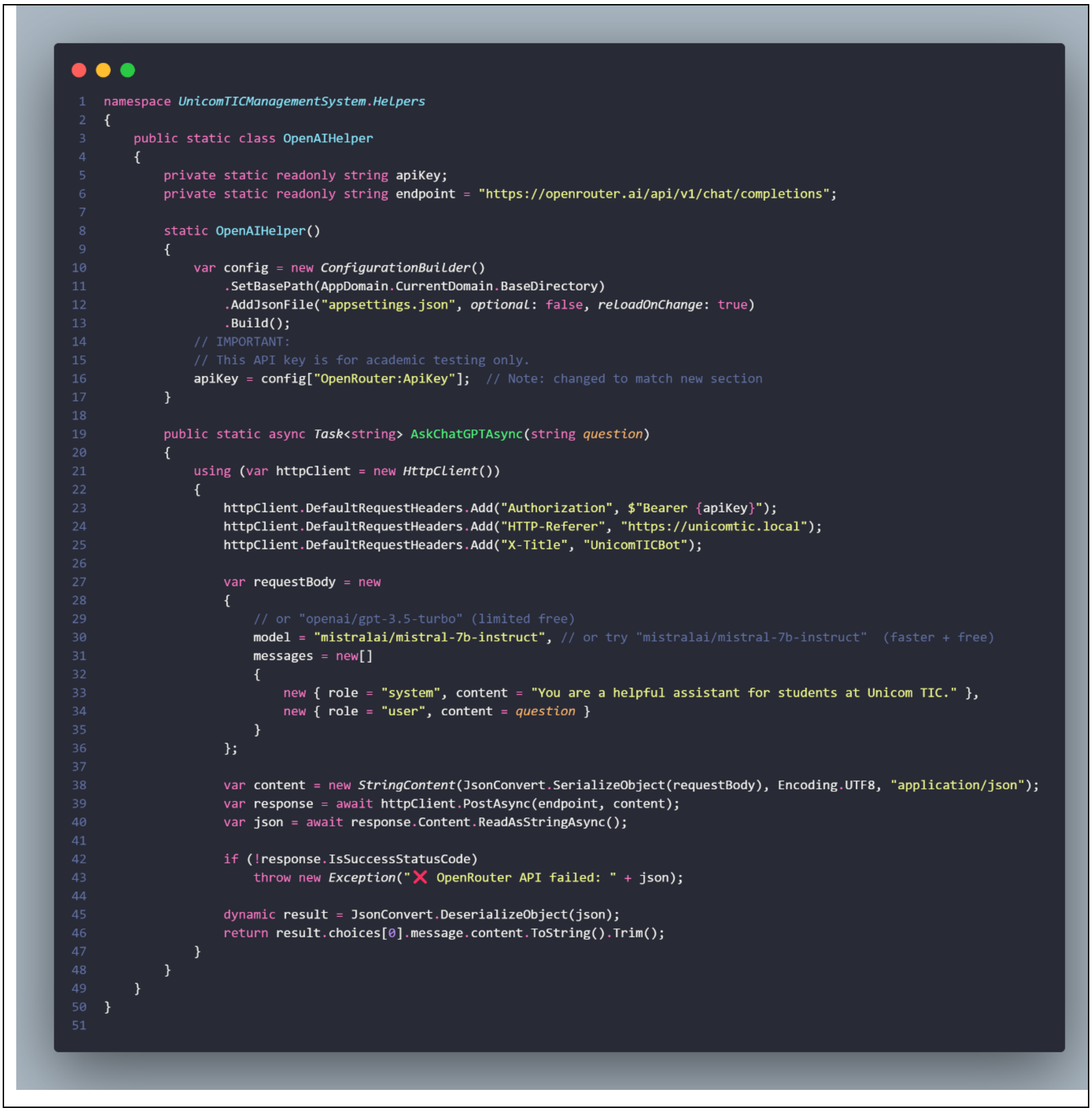
The stages of a typical SDLC are:

1. Planning: Define project scope, objectives, requirements, and deliverables.
2. Requirement Analysis: Gather and document Requirements from clients and stakeholders.
3. Design: Create a detailed design for the software system based on the requirements.
4. Implementation: Develop the software according to the design specifications.
5. Testing: Verify that the software meets the requirements and performs as expected.
6. Deployment: Deploy the software to production environment and begin using it.
7. Maintenance: Monitor the software, fix any bugs or issues, and make improvements based on feedback and evolving requirements.

appsettings.json



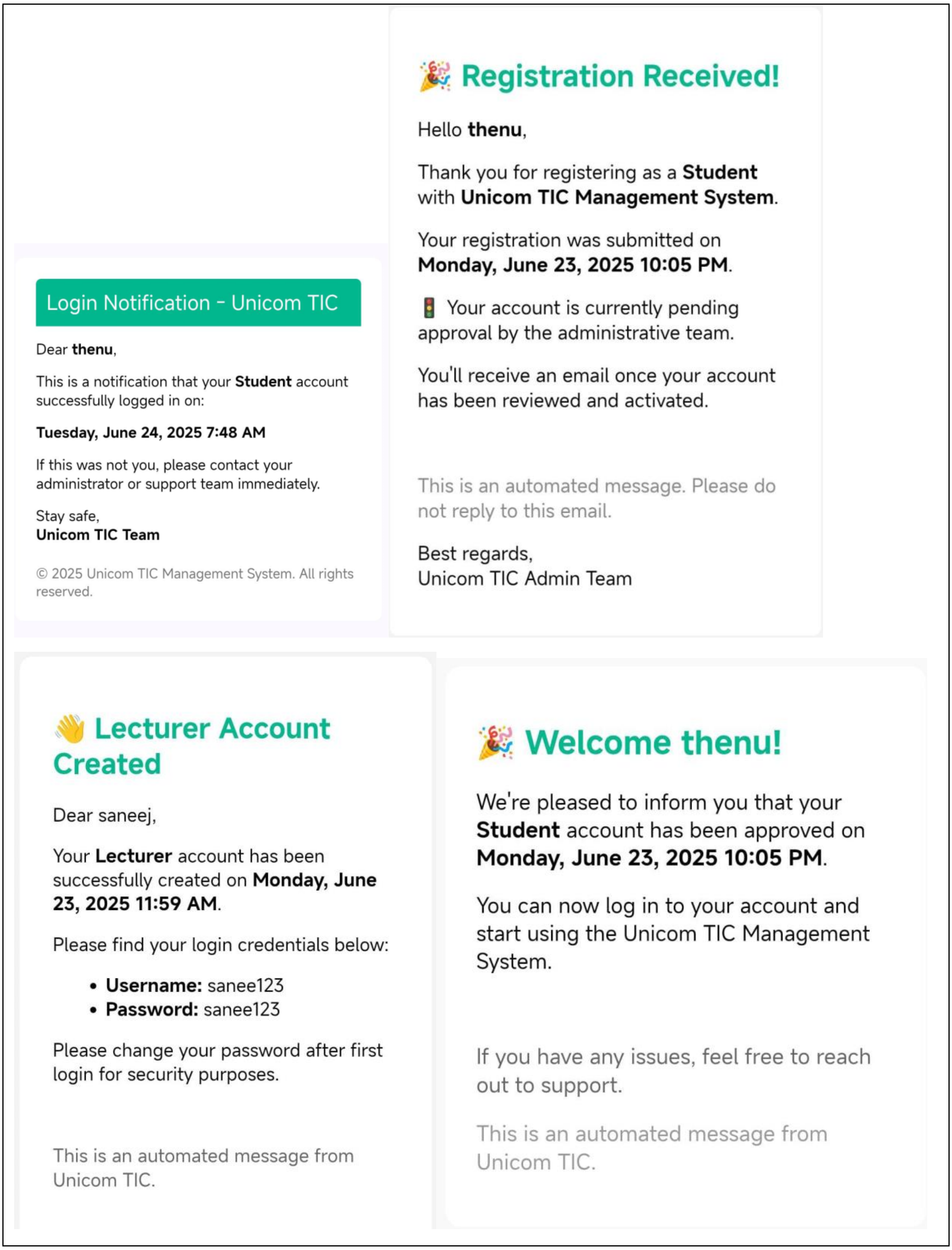
```
{
  // IMPORTANT:
  // This API key is available on document Addicational Note We Can not Push Api Key To Github .
  "OpenRouter": {
    "ApiKey": "sk-YOUR-REAL-OPENROUTER-KEY"
  }
}

```



```
namespace UnicomTICManagementSystem.Helpers
{
    public static class OpenAIHelper
    {
        private static readonly string apiKey;
        private static readonly string endpoint = "https://openrouter.ai/api/v1/chat/completions";

        static OpenAIHelper()
        {
            var config = new ConfigurationBuilder()
                .SetBasePath(AppDomain.CurrentDomain.BaseDirectory)
                .AddJsonFile("appsettings.json", optional: false, reloadOnChange: true)
                .Build();
            // IMPORTANT:
            // This API key is for academic testing only.
            apiKey = config["OpenRouter:ApiKey"];  // Note: changed to match new section
        }

        public static async Task<string> AskChatGPTAsync(string question)
        {
            using (var httpClient = new HttpClient())
            {
                httpClient.DefaultRequestHeaders.Add("Authorization", $"Bearer {apiKey}");
                httpClient.DefaultRequestHeaders.Add("HTTP-Referer", "https://unicomtic.local");
                httpClient.DefaultRequestHeaders.Add("X-Title", "UnicomTICBot");

                var requestBody = new
                {
                    // or "openai/gpt-3.5-turbo" (limited free)
                    model = "mistralai/mistral-7b-instruct", // or try "mistralai/mistral-7b-instruct"  (faster + free)
                    messages = new[]
                    {
                        new { role = "system", content = "You are a helpful assistant for students at Unicom TIC." },
                        new { role = "user", content = question }
                    }
                };

                var content = new StringContent(JsonConvert.SerializeObject(requestBody), Encoding.UTF8, "application/json");
                var response = await httpClient.PostAsync(endpoint, content);
                var json = await response.Content.ReadAsStringAsync();

                if (!response.IsSuccessStatusCode)
                    throw new Exception("❌ OpenRouter API failed: " + json);

                dynamic result = JsonConvert.DeserializeObject(json);
                return result.choices[0].message.content.ToString().Trim();
            }
        }
    }
}
```

Logic to call OpenRouter API and handle responses

3. MailKit Integration



Login Notification - Unicom TIC

Dear **thenu**,

This is a notification that your **Student** account successfully logged in on:

Tuesday, June 24, 2025 7:48 AM

If this was not you, please contact your administrator or support team immediately.

Stay safe,
Unicom TIC Team

© 2025 Unicom TIC Management System. All rights reserved.

🎉 **Registration Received!**

Hello **thenu**,

Thank you for registering as a **Student** with **Unicom TIC Management System**.

Your registration was submitted on **Monday, June 23, 2025 10:05 PM**.

🚦 Your account is currently pending approval by the administrative team.

You'll receive an email once your account has been reviewed and activated.

This is an automated message. Please do not reply to this email.

Best regards,
Unicom TIC Admin Team

👋 **Lecturer Account Created**

Dear saneej,

Your **Lecturer** account has been successfully created on **Monday, June 23, 2025 11:59 AM**.

Please find your login credentials below:

- **Username:** sanee123
- **Password:** sanee123

Please change your password after first login for security purposes.

This is an automated message from Unicom TIC.

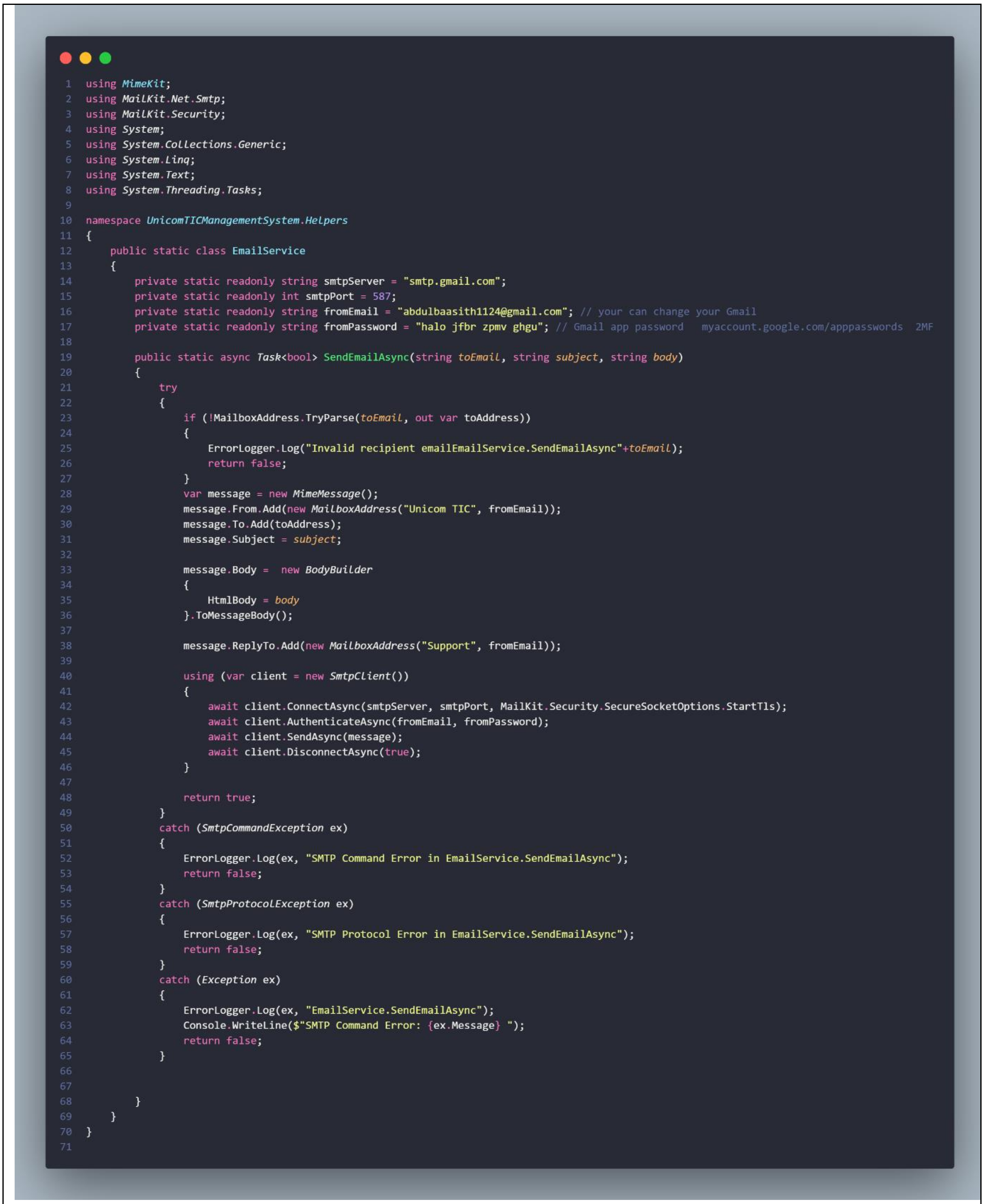
🎉 **Welcome thenu!**

We're pleased to inform you that your **Student** account has been approved on **Monday, June 23, 2025 10:05 PM**.

You can now log in to your account and start using the Unicom TIC Management System.

If you have any issues, feel free to reach out to support.

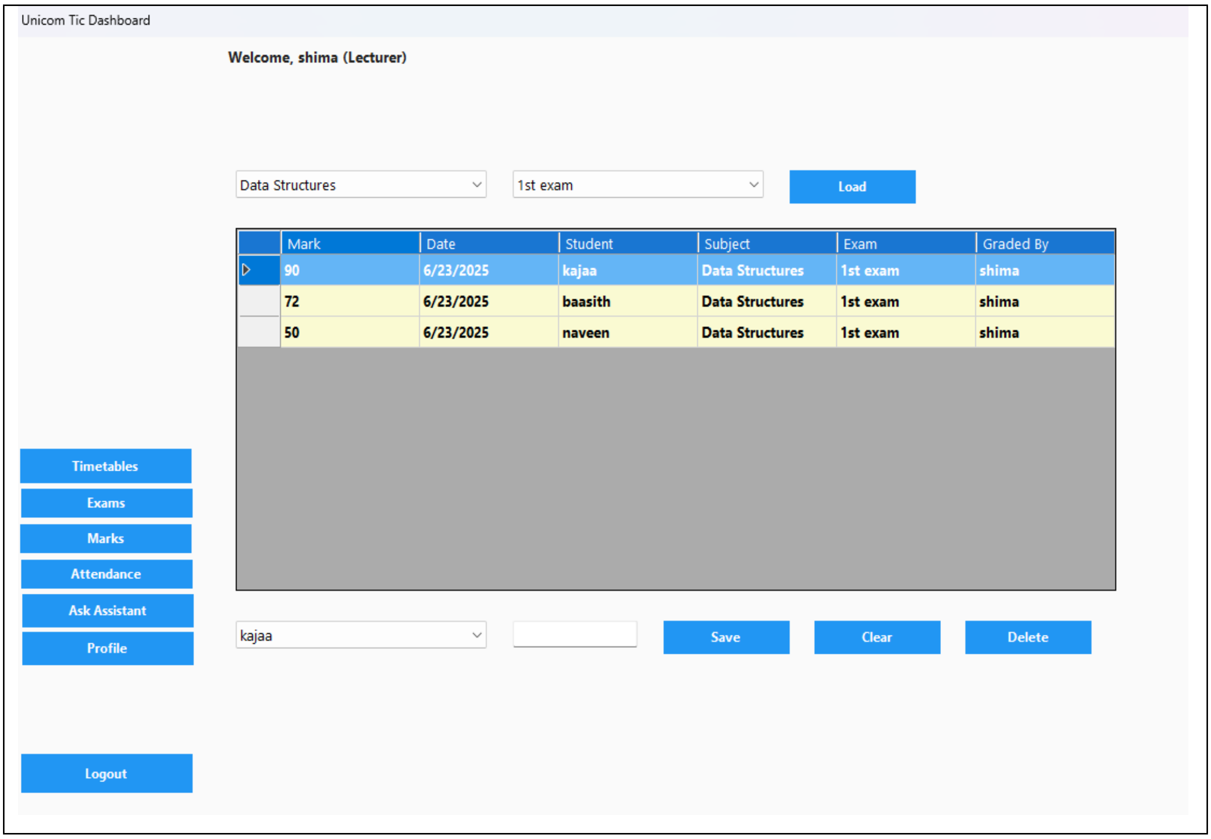
This is an automated message from Unicom TIC.



```
using MimeKit;
using MailKit.Net.Smtp;
using MailKit.Security;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace UnicomTICManagementSystem.Helpers
{
    public static class EmailService
    {
        private static readonly string smtpServer = "smtp.gmail.com";
        private static readonly int smtpPort = 587;
        private static readonly string fromEmail = "abdulbaasith1124@gmail.com"; // your can change your Gmail
        private static readonly string fromPassword = "halo jfbr zpmv ghgu"; // Gmail app password   myaccount.google.com/apppasswords  2MF

        public static async Task<bool> SendEmailAsync(string toEmail, string subject, string body)
        {
            try
            {
                if (!MailboxAddress.TryParse(toEmail, out var toAddress))
                {
                    ErrorLogger.Log("Invalid recipient emailEmailService.SendEmailAsync"+toEmail);
                    return false;
                }
                var message = new MimeMessage();
                message.From.Add(new MailboxAddress("Unicom TIC", fromEmail));
                message.To.Add(toAddress);
                message.Subject = subject;

                message.Body =  new BodyBuilder
                {
                    HtmlBody = body
                }.ToMessageBody();

                message.ReplyTo.Add(new MailboxAddress("Support", fromEmail));

                using (var client = new SmtpClient())
                {
                    await client.ConnectAsync(smtpServer, smtpPort, MailKit.Security.SecureSocketOptions.StartTls);
                    await client.AuthenticateAsync(fromEmail, fromPassword);
                    await client.SendAsync(message);
                    await client.DisconnectAsync(true);
                }

                return true;
            }
            catch (SmtpCommandException ex)
            {
                ErrorLogger.Log(ex, "SMTP Command Error in EmailService.SendEmailAsync");
                return false;
            }
            catch (SmtpProtocolException ex)
            {
                ErrorLogger.Log(ex, "SMTP Protocol Error in EmailService.SendEmailAsync");
                return false;
            }
            catch (Exception ex)
            {
                ErrorLogger.Log(ex, "EmailService.SendEmailAsync");
                Console.WriteLine($"SMTP Command Error: {ex.Message} ");
                return false;
            }


        }
    }
}
```

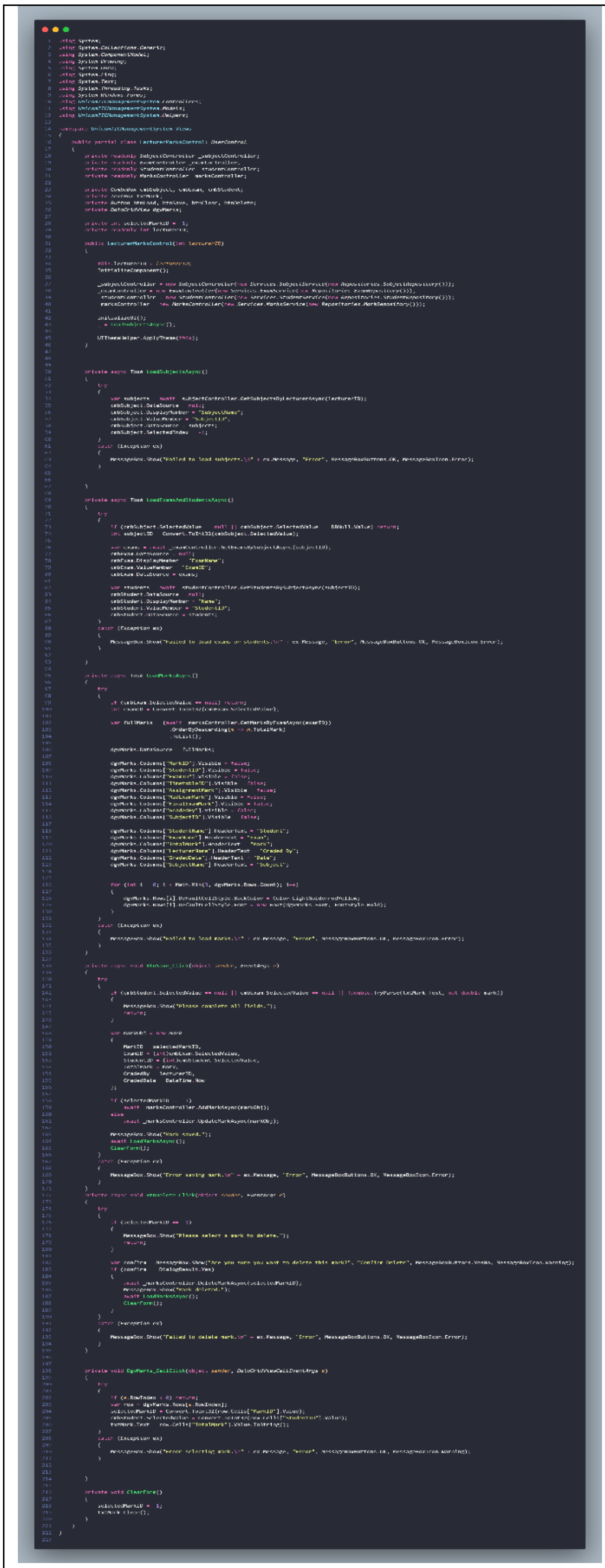
Email service using templates (registration, approval,ogin)

4. Lecturer Marks Management



Welcome, shima (Lecturer)

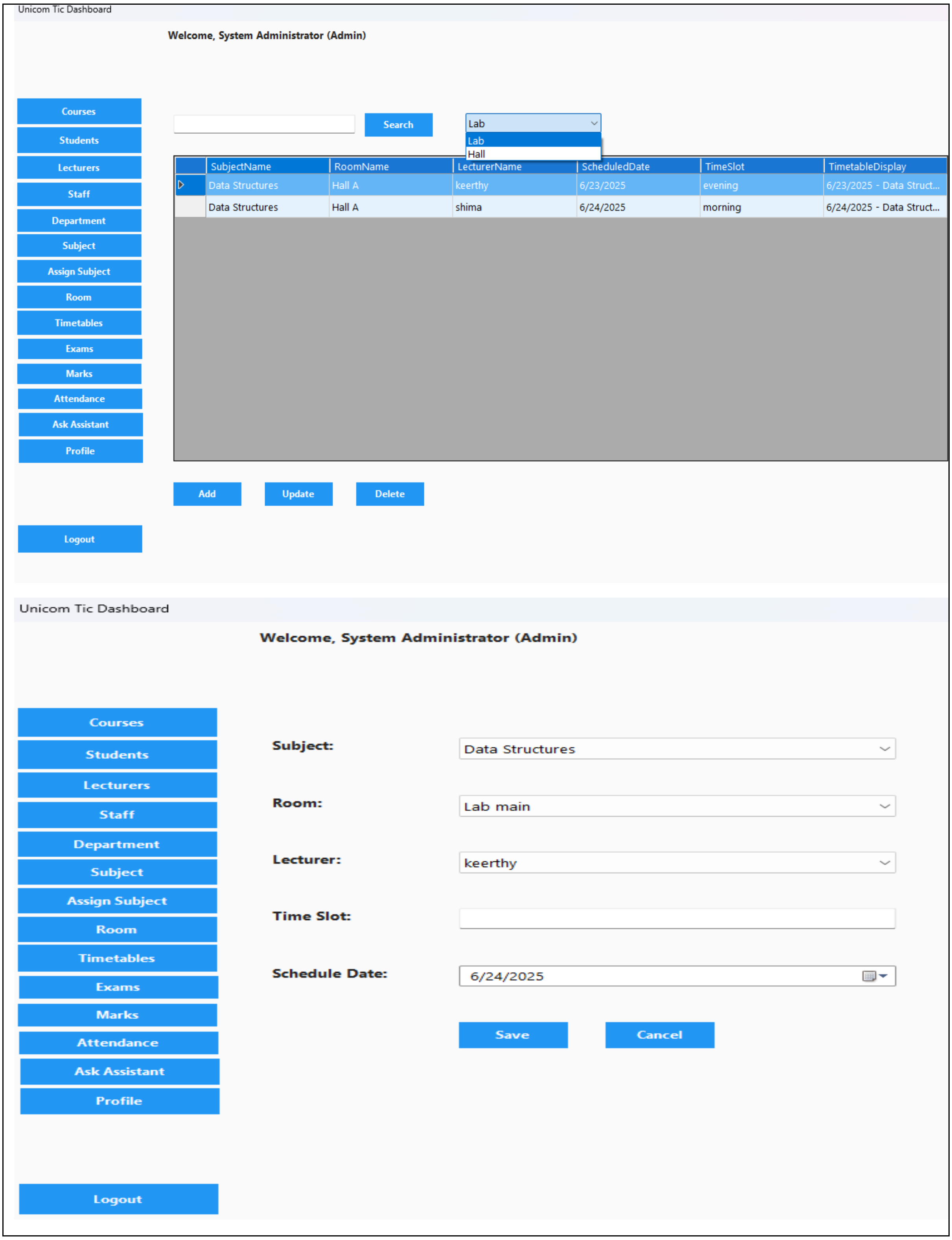
Mark	Date	Student	Subject	Exam	Graded By
90	6/23/2025	kajaa	Data Structures	1st exam	shima
72	6/23/2025	baasith	Data Structures	1st exam	shima
50	6/23/2025	naveen	Data Structures	1st exam	shima



```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using UnicomTICManagementSystem.Controllers;
using UnicomTICManagementSystem.Models;
using UnicomTICManagementSystem.Helpers;

namespace UnicomTICManagementSystem.Views
{
    public partial class LecturerMarksControl : UserControl
    {
        private readonly SubjectController _subjectController;
        private readonly ExamController _examController;
        private readonly StudentController _studentController;
        private readonly MarksController _marksController;

        private ComboBox cmbSubject, cmbExam, cmbStudent;
        private TextBox txtMark;
        private Button btnLoad, btnSave, btnClear, btnDelete;
        private DataGridView dgvMarks;

        private int selectedMarkID = -1;
        private readonly int lecturerID;

        public LecturerMarksControl(int lecturerID)
        {

            this.lecturerID = lecturerID;
            InitializeComponent();

            _subjectController = new SubjectController(new Services.SubjectService(new Repositories.SubjectRepository()));
            _examController = new ExamController(new Services.ExamService(new Repositories.ExamRepository()));
            _studentController = new StudentController(new Services.StudentService(new Repositories.StudentRepository()));
            _marksController = new MarksController(new Services.MarksService(new Repositories.MarkRepository()));

            InitializeUI();
            _ = LoadSubjectsAsync();

            UIThemeHelper.ApplyTheme(this);
        }



        private async Task LoadSubjectsAsync()
        {
            try
            {
                var subjects = await _subjectController.GetSubjectsByLecturerAsync(lecturerID);
                cmbSubject.DataSource = null;
                cmbSubject.DisplayMember = "SubjectName";
                cmbSubject.ValueMember = "SubjectID";
                cmbSubject.DataSource = subjects;
                cmbSubject.SelectedIndex = -1;
            }
            catch (Exception ex)
            {
                MessageBox.Show("Failed to load subjects.\n" + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }


        }

        private async Task LoadExamsAndStudentsAsync()
        {
            try
            {
                if (cmbSubject.SelectedValue == null || cmbSubject.SelectedValue == DBNull.Value) return;
                int subjectID = Convert.ToInt32(cmbSubject.SelectedValue);

                var exams = await _examController.GetExamsBySubjectAsync(subjectID);
                cmbExam.DataSource = null;
                cmbExam.DisplayMember = "ExamName";
                cmbExam.ValueMember = "ExamID";
                cmbExam.DataSource = exams;

                var students = await _studentController.GetStudentsBySubjectAsync(subjectID);
                cmbStudent.DataSource = null;
                cmbStudent.DisplayMember = "Name";
                cmbStudent.ValueMember = "StudentID";
                cmbStudent.DataSource = students;
            }
            catch (Exception ex)
            {
                MessageBox.Show("Failed to load exams or students.\n" + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }

        }

        private async Task LoadMarksAsync()
        {
            try
            {
                if (cmbExam.SelectedValue == null) return;
                int examID = Convert.ToInt32(cmbExam.SelectedValue);

                var fullMarks = (await _marksController.GetMarksByExamAsync(examID))
                                .OrderByDescending(m => m.TotalMark)
                                .ToList();

                dgvMarks.DataSource = fullMarks;

                dgvMarks.Columns["MarkID"].Visible = false;
                dgvMarks.Columns["StudentID"].Visible = false;
                dgvMarks.Columns["ExamID"].Visible = false;
                dgvMarks.Columns["TimetableID"].Visible = false;
                dgvMarks.Columns["AssignmentMark"].Visible = false;
                dgvMarks.Columns["MidExamMark"].Visible = false;
                dgvMarks.Columns["FinalExamMark"].Visible = false;
                dgvMarks.Columns["GradedBy"].Visible = false;
                dgvMarks.Columns["SubjectID"].Visible = false;

                dgvMarks.Columns["StudentName"].HeaderText = "Student";
                dgvMarks.Columns["ExamName"].HeaderText = "Exam";
                dgvMarks.Columns["TotalMark"].HeaderText = "Mark";
                dgvMarks.Columns["LecturerName"].HeaderText = "Graded By";
                dgvMarks.Columns["GradedDate"].HeaderText = "Date";
                dgvMarks.Columns["SubjectName"].HeaderText = "Subject";


                for (int i = 0; i < Math.Min(3, dgvMarks.Rows.Count); i++)
                {
                    dgvMarks.Rows[i].DefaultCellStyle.BackColor = Color.LightGoldenrodYellow;
                    dgvMarks.Rows[i].DefaultCellStyle.Font = new Font(dgvMarks.Font, FontStyle.Bold);
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Failed to load marks.\n" + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        private async void BtnSave_Click(object sender, EventArgs e)
        {
            try
            {
                if (cmbStudent.SelectedValue == null || cmbExam.SelectedValue == null || !double.TryParse(txtMark.Text, out double mark))
                {
                    MessageBox.Show("Please complete all fields.");
                    return;
                }

                var markObj = new Mark
                {
                    MarkID = selectedMarkID,
                    ExamID = (int)cmbExam.SelectedValue,
                    StudentID = (int)cmbStudent.SelectedValue,
                    TotalMark = mark,
                    GradedBy = lecturerID,
                    GradedDate = DateTime.Now
                };

                if (selectedMarkID == -1)
                    await _marksController.AddMarkAsync(markObj);
                else
                    await _marksController.UpdateMarkAsync(markObj);

                MessageBox.Show("Mark saved.");
                await LoadMarksAsync();
                ClearForm();
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error saving mark.\n" + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
        private async void BtnDelete_Click(object sender, EventArgs e)
        {
            try
            {
                if (selectedMarkID == -1)
                {
                    MessageBox.Show("Please select a mark to delete.");
                    return;
                }

                var confirm = MessageBox.Show("Are you sure you want to delete this mark?", "Confirm Delete", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);
                if (confirm == DialogResult.Yes)
                {
                    await _marksController.DeleteMarkAsync(selectedMarkID);
                    MessageBox.Show("Mark deleted.");
                    await LoadMarksAsync();
                    ClearForm();
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Failed to delete mark.\n" + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }


        private void DgvMarks_CellClick(object sender, DataGridViewCellEventArgs e)
        {
            try
            {
                if (e.RowIndex < 0) return;
                var row = dgvMarks.Rows[e.RowIndex];
                selectedMarkID = Convert.ToInt32(row.Cells["MarkID"].Value);
                cmbStudent.SelectedValue = Convert.ToInt32(row.Cells["StudentID"].Value);
                txtMark.Text = row.Cells["TotalMark"].Value.ToString();
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error selecting mark.\n" + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            }


        }

        private void ClearForm()
        {
            selectedMarkID = -1;
            txtMark.Clear();
        }
    }
}
```

Add/Edit/Delete marks for their subject , usage of async, filtering by subject, validation

5. Timetable Creation with Room Allocation



SubjectName	RoomName	LecturerName	ScheduledDate	TimeSlot	TimetableDisplay
Data Structures	Hall A	keerthy	6/23/2025	evening	6/23/2025 - Data Struct...
Data Structures	Hall A	shima	6/24/2025	morning	6/24/2025 - Data Struct...

Subject: Data Structures

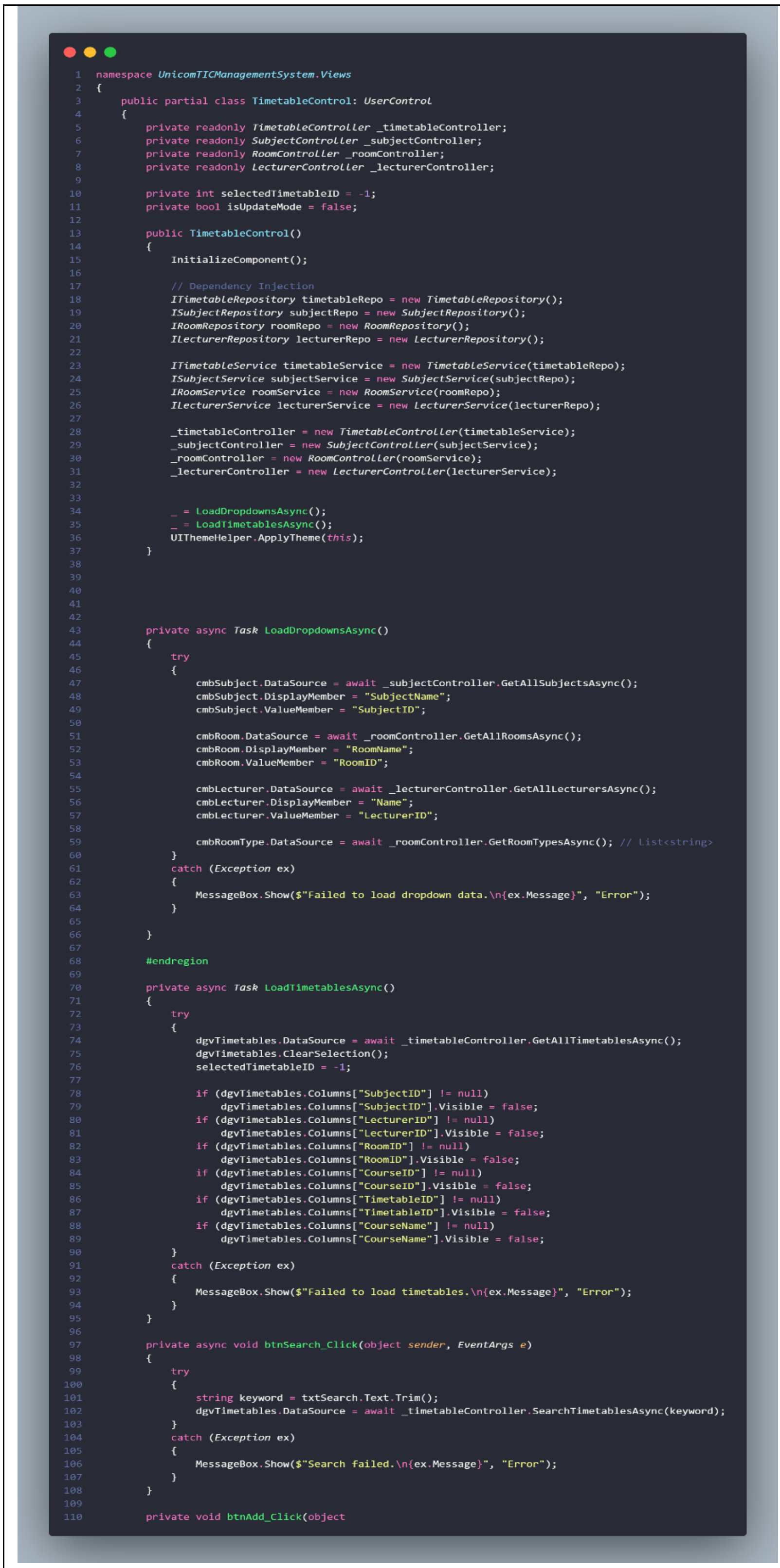
Room: Lab main

Lecturer: keerthy

Time Slot:

Schedule Date: 6/24/2025

Subject, room, slot assignment



```
namespace UnicomTICManagementSystem.Views
{
    public partial class TimetableControl: UserControl
    {
        private readonly TimetableController _timetableController;
        private readonly SubjectController _subjectController;
        private readonly RoomController _roomController;
        private readonly LecturerController _lecturerController;

        private int selectedTimetableID = -1;
        private bool isUpdateMode = false;

        public TimetableControl()
        {
            InitializeComponent();

            // Dependency Injection
            ITimetableRepository timetableRepo = new TimetableRepository();
            ISubjectRepository subjectRepo = new SubjectRepository();
            IRoomRepository roomRepo = new RoomRepository();
            ILecturerRepository lecturerRepo = new LecturerRepository();

            ITimetableService timetableService = new TimetableService(timetableRepo);
            ISubjectService subjectService = new SubjectService(subjectRepo);
            IRoomService roomService = new RoomService(roomRepo);
            ILecturerService lecturerService = new LecturerService(lecturerRepo);

            _timetableController = new TimetableController(timetableService);
            _subjectController = new SubjectController(subjectService);
            _roomController = new RoomController(roomService);
            _lecturerController = new LecturerController(lecturerService);


            _ = LoadDropdownsAsync();
            _ = LoadTimetablesAsync();
            UIThemeHelper.ApplyTheme(this);
        }






        private async Task LoadDropdownsAsync()
        {
            try
            {
                cmbSubject.DataSource = await _subjectController.GetAllSubjectsAsync();
                cmbSubject.DisplayMember = "SubjectName";
                cmbSubject.ValueMember = "SubjectID";

                cmbRoom.DataSource = await _roomController.GetAllRoomsAsync();
                cmbRoom.DisplayMember = "RoomName";
                cmbRoom.ValueMember = "RoomID";

                cmbLecturer.DataSource = await _lecturerController.GetAllLecturersAsync();
                cmbLecturer.DisplayMember = "Name";
                cmbLecturer.ValueMember = "LecturerID";

                cmbRoomType.DataSource = await _roomController.GetRoomTypesAsync(); // List<string>
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Failed to load dropdown data.\n{ex.Message}", "Error");
            }

        }

        #endregion

        private async Task LoadTimetablesAsync()
        {
            try
            {
                dgvTimetables.DataSource = await _timetableController.GetAllTimetablesAsync();
                dgvTimetables.ClearSelection();
                selectedTimetableID = -1;

                if (dgvTimetables.Columns["SubjectID"] != null)
                    dgvTimetables.Columns["SubjectID"].Visible = false;
                if (dgvTimetables.Columns["LecturerID"] != null)
                    dgvTimetables.Columns["LecturerID"].Visible = false;
                if (dgvTimetables.Columns["RoomID"] != null)
                    dgvTimetables.Columns["RoomID"].Visible = false;
                if (dgvTimetables.Columns["CourseID"] != null)
                    dgvTimetables.Columns["CourseID"].Visible = false;
                if (dgvTimetables.Columns["TimetableID"] != null)
                    dgvTimetables.Columns["TimetableID"].Visible = false;
                if (dgvTimetables.Columns["CourseName"] != null)
                    dgvTimetables.Columns["CourseName"].Visible = false;
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Failed to load timetables.\n{ex.Message}", "Error");
            }
        }

        private async void btnSearch_Click(object sender, EventArgs e)
        {
            try
            {
                string keyword = txtSearch.Text.Trim();
                dgvTimetables.DataSource = await _timetableController.SearchTimetablesAsync(keyword);
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Search failed.\n{ex.Message}", "Error");
            }
        }

        private void btnAdd_Click(object
```

6. Password Hashing with BCrypt

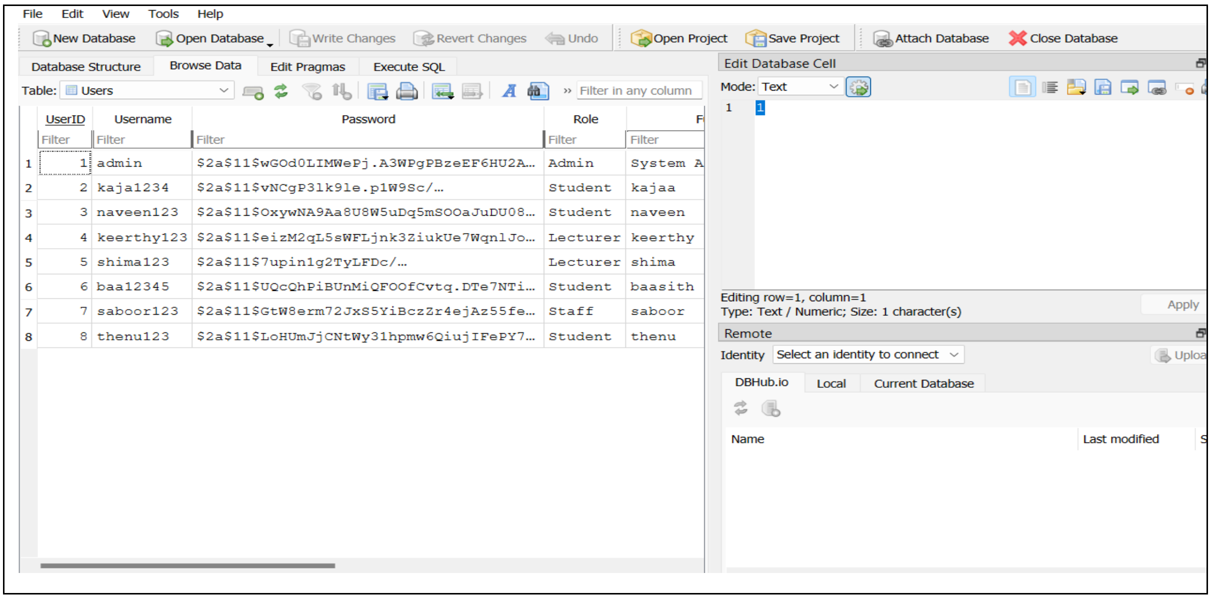


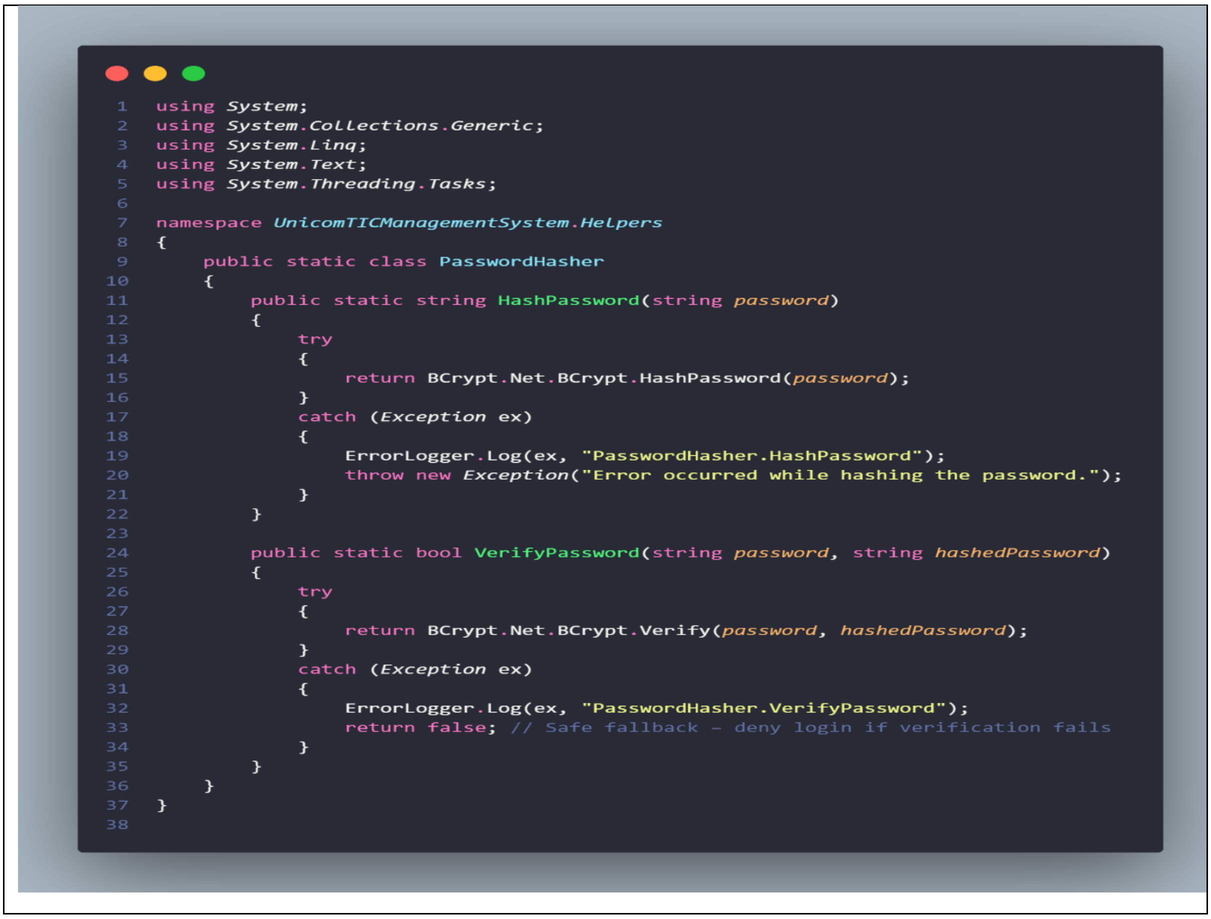
Table: Users

	UserID	Username	Password	Role	F
1	1	admin	$2a$11$wGOd0LIMWePj.A3WPgPBzeEF6HU2A…	Admin	System A
2	2	kaja1234	$2a$11$vNCgP3lk9le.p1W9Sc/…	Student	kajaa
3	3	naveen123	$2a$11$OxywNA9Aa8U8W5uDq5mSOOaJuDU08…	Student	naveen
4	4	keerthy123	$2a$11$eizM2qL5sWFLjnk3ZiukUe7WqnlJo…	Lecturer	keerthy
5	5	shima123	$2a$11$7upin1g2TyLFDc/…	Lecturer	shima
6	6	baa12345	$2a$11$UQcQhPiBUnMiQFOOfCvtq.DTe7NTi…	Student	baasith
7	7	saboor123	$2a$11$GtW8erm72JxS5YiBczZr4ejAz55fe…	Staff	saboor
8	8	thenu123	$2a$11$LoHUmJjCNtWy31hpmw6QiujIFePY7…	Student	thenu

Mode: Text

Editing row=1, column=1
Type: Text / Numeric; Size: 1 character(s)

Identity: Select an identity to connect



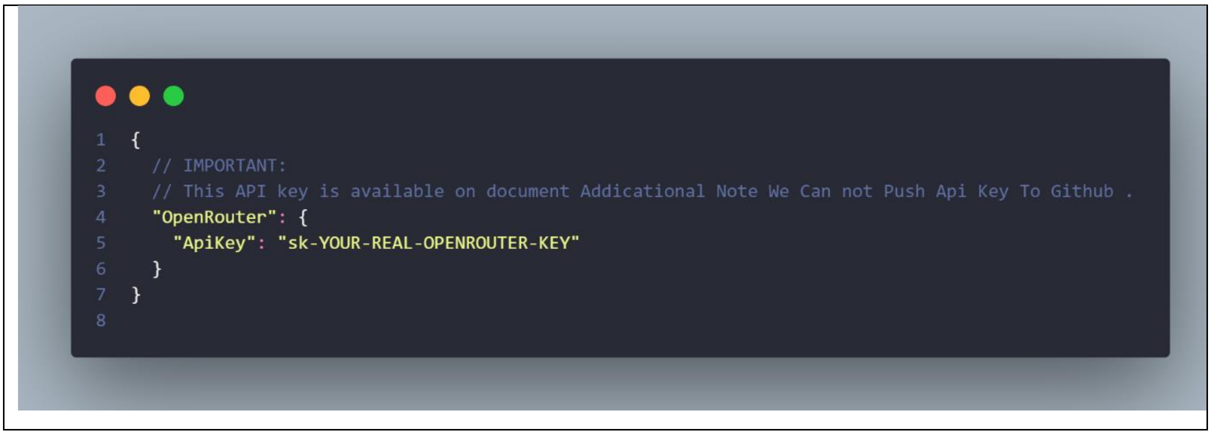
```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace UnicomTICManagementSystem.Helpers
{
    public static class PasswordHasher
    {
        public static string HashPassword(string password)
        {
            try
            {
                return BCrypt.Net.BCrypt.HashPassword(password);
            }
            catch (Exception ex)
            {
                ErrorLogger.Log(ex, "PasswordHasher.HashPassword");
                throw new Exception("Error occurred while hashing the password.");
            }
        }

        public static bool VerifyPassword(string password, string hashedPassword)
        {
            try
            {
                return BCrypt.Net.BCrypt.Verify(password, hashedPassword);
            }
            catch (Exception ex)
            {
                ErrorLogger.Log(ex, "PasswordHasher.VerifyPassword");
                return false; // Safe fallback - deny login if verification fails
            }
        }
    }
}

```

– Secure user registration/login

Additional Notes on OpenRouter API Key Handling

OpenRouter API key is required for the GPT-based chatbot feature.

The key is stored in **appsettings.json** like this:



```
{
  // IMPORTANT:
  // This API key is available on document Addicational Note We Can not Push Api Key To Github .
  "OpenRouter": {
    "ApiKey": "sk-YOUR-REAL-OPENROUTER-KEY"
  }
}

```

When Before Runnig Change the ApiKey "sk-YOUR-REAL-OPENROUTER-KEY" to this Original Key .

Api Key

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

If Not This Working Contanct Me I will Provide Another Key

Number : +94777353481

The Api must NOT be pushed to GitHub.

If accidentally exposed, Open Router revokes the key automatically for security.