A
PROJECT REPORT ON

Attendance Management System

By

SHAH YASH (CE-144) (19CEUEG051)
SAKHIYA VIDUR (CE-136) (19CEUOG101)

B.Tech CE Semester-VI Subject:
(CE-619)Service Oriented Computing

Guided by: Prof.Prashant Jadav
Associate Professor Dept. of
Comp. Engg.

Faculty of Technology
Department of Computer Engineering
Dharmsinh Desai University

Faculty of Technology
Department of Computer Engineering
Dharmsinh Desai University

CERTIFICATE

This is to certify that the practical / **term work** carried out in the subject of

Service Oriented Computing and recorded in this journal is the

bonafide work of

SHAH YASH (CE-144) (19CEUEG051)
SAKHIYA VIDUR (CE-136) (19CEUOG101)

of B.Tech semester **VI** in the branch of **Computer Engineering**

during the academic year **2021-2022**.

Prof. Prasant Jadav
Associate Professor,
Dept. of Computer Engineering,
Faculty of Technology
Dharmsinh Desai University, Nadiad

Dr. C. K. Bhensdadia,
Head,
Dept. of Computer Engineering.,
Faculty of Technology
Dharmsinh Desai University, Nadiad

Contents

Introduction

- **About Project:**

Attendance Management System is WCF based web application with functionality of managing attendances of students at school level. It reduces burden of managing musters and pages of attendances.

Our application provides some of major functionalities of taking attendance of students by subject-teachers on daily basis and view attendance of their students, managing student and teachers details(admin side), viewing attendance as per student or teacher(admin side).

- **Technology:**

Our Application mainly implemented in .net technology. It is divided in three parts

i) WcfServices - It is backend part implemented in WCF Service Library

ii)ConsoleHost - It is used to host services implemented in Console App(.Net Framework).

iii)WebClient - It is client part to consume services implemented in asp.Net Web Application(.Net Framework).

We have also used Entity Framework for strong database functionality.

- **Tools:**
  - Visual Studio
  - MSSQLLocalDB(SQL Server)
  - Github

SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

Attendance Management System

R 1: Admin side

Description: Admin can Add/Update/Delete Teachers and Students details and can show attendances by teacher, by students.

R 1.1: Authentication

Description: Admin must have to login for perform actions.

R 1.1.1: Sign In

Input: User details.
Process: Validate details.
Output: Redirect to Admin page.

R 1.1.2: Log Out

Input: User Selection.
Output: Redirect to Home page.

R 1.2: Add Teacher

Input: Teacher's Details.
Process: validate.
Output: Success or Failure Message.

R 1.3: Add Student

Input: Student's Details.
Process: validate.
Output: Success or Failure Message.

R 1.4: View List

Description: it will show list of students and teachers details as per selected class.

R 1.4.1: Edit Student/Teacher

Input: User input.
Process: validate.
Output: Update list.

R 1.4.2: Delete Student/Teacher

Input: User selection.
Output: Delete record.

R 1.4.3: View Attendance (Student)

Input: User selection.
Process: get students attendance by selected student and subject.
Output: Show Student's Attendance.

R 1.4.4: View Attendance (Teacher)

Input: User selection.
Process: get Students Attendance by selected class, date and teacher.
Output: Show Students Attendance.

R 2: Teacher Side

Description: Teacher can take today's attendance and view attendance by date.

R 2.1: Authentication

Description: Teacher have to login first into application to perform task.

R 2.1.1: Sign In

Input: User details.
Process: Validate details.
Output: Redirect to Teacher Page.

R 2.1.2: Log Out

Input: User Selection.
Output: Redirect to Home page.

R 2.2: Take Attendance

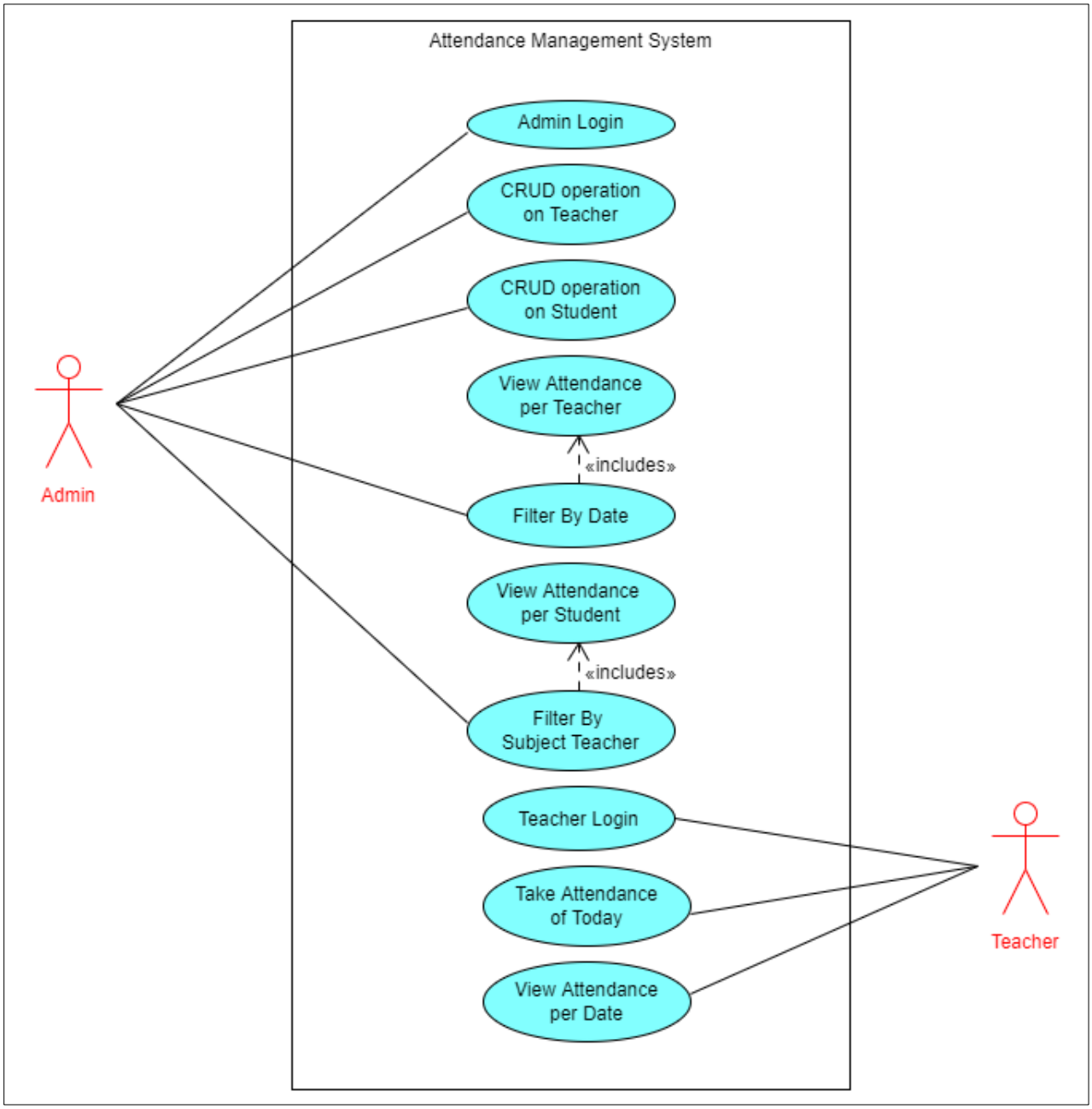
Input: List of Attendance as per Student's presence.
Process: validate.
Output: Success or Failure Message.

R 2.3: View Attendance

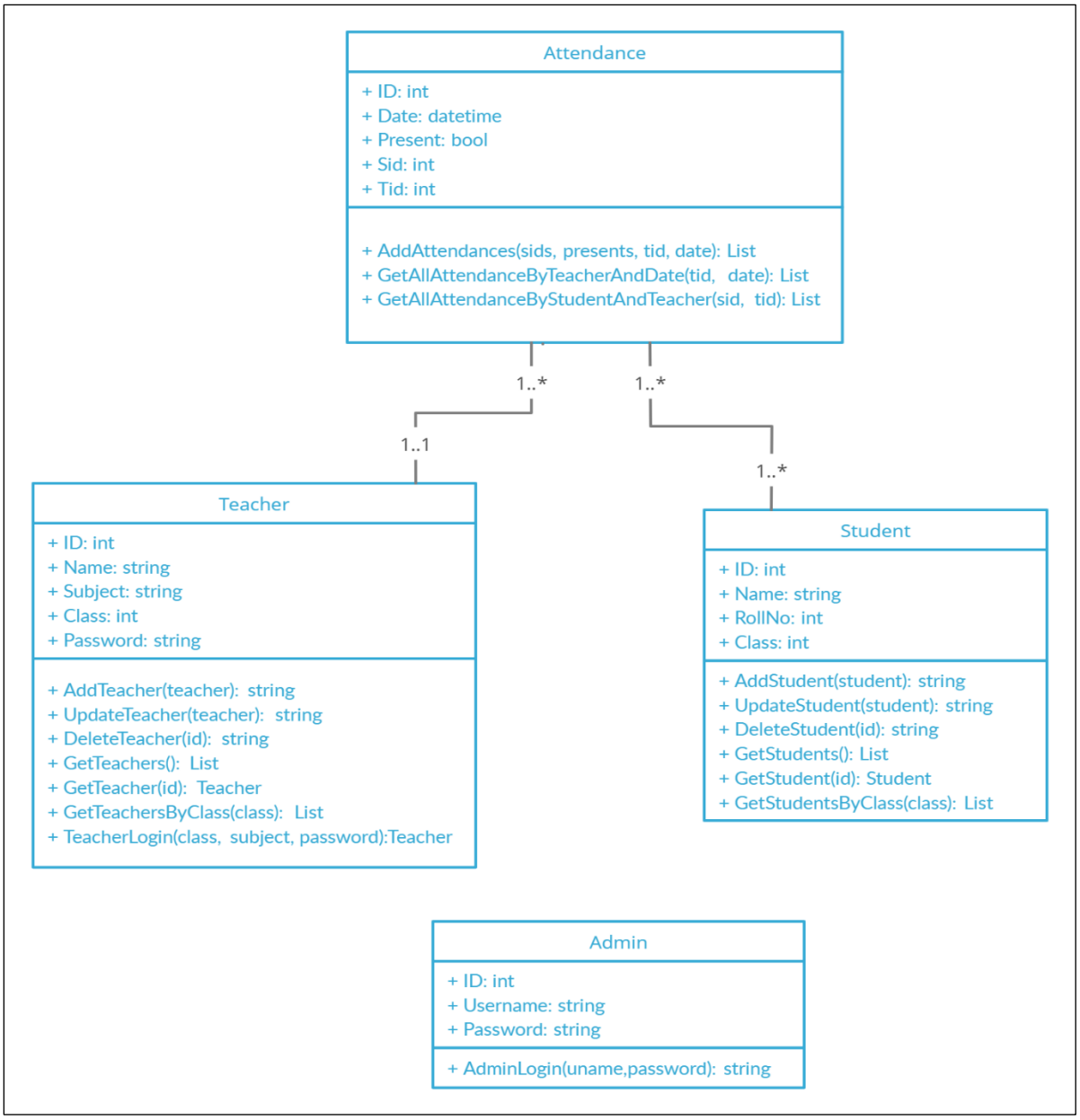
Input: Selection of Date.
Process: Get list of student Attendance as per selected date.
Output: Show list of attendance.

DESIGN

Use Case Diagram



Class Diagram



Data Dictionary

Tables

- dbo.__MigrationHistory
- dbo.Admins
- dbo.Attendances
- dbo.Students
- dbo.Teachers

Admin

Name	Data Type	Allow Nulls	Default
Id	int	☐	
Username	nvarchar(MAX)	☑	
password	nvarchar(MAX)	☑	
		☐	

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

Student

Name	Data Type	Allow Nulls	Default
Id	int	☐	
Name	nvarchar(MAX)	☑	
Class	int	☐	
RollNo	int	☐	
		☐	

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

Teacher

Name	Data Type	Allow Nulls	Default
Id	int	☐	
Name	nvarchar(MAX)	☑	
Class	int	☐	
Subject	nvarchar(MAX)	☑	
Password	nvarchar(MAX)	☑	
		☐	

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

Attendance

Name	Data Type	Allow Nulls	Default
Id	int	☐	
Date	datetime	☐	
Present	bit	☐	
S_Id	int	☑	
T_Id	int	☑	
		☐	

- **Keys** (1)
  - PK_dbo.Attendances (Primary Key, Clustered: Id)
- **Check Constraints** (0)
- **Indexes** (2)
  - IX_S_Id (S_Id)
  - IX_T_Id (T_Id)
- **Foreign Keys** (2)
  - FK_dbo.Attendances_dbo.Students_S_Id (Id)
  - FK_dbo.Attendances_dbo.Teachers_T_Id (Id)
- **Triggers** (0)

Implementation Details

- **Modules created and brief description of each module**

  This Project consists of 3 major modules.

  1) Take Attendance
  2) View Attendance
  3) CRUD operation on Student/Teacher

  Each module consists of several methods to implement the required functionality Implementation is done using WCF and Asp.net.

  1) Take Attendance

  After Successfully Login Teacher can take attendance of his/her subject day by day.

  2) View Attendance

  Admin: Admin can view attendance by selected student and subject as well as selected teacher and date.

  Teacher: Teacher can view attendance of his/her subject as per date.

  3) CRUD operation on Student/Teacher

  Only admin can perform CRUD operation on students and teachers.

➢ Function prototypes which implement major functionality

1) Attendance

```
public string AddAttendances(List<int> sids, List<bool> presents, int tid, DateTime dt)
{
    try
    {
        IEnumerable<Attendance> x = GetAllAttendanceByTeacherAndDate(tid, dt);
        int count = x.Count();
        if (count >0 )
        {
            return "Attendance Already Taken";
        }
        TeacherService.Teacher t = db.TeacherModel.Find(tid);
        for (int i = 0; i < sids.Count; i++)
        {
            StudentService.Student s = db.StudentModel.Find(sids[i]);
            Attendance a = new Attendance();
            a.S = s;
            a.T = t;
            a.Date = dt.Date;
            a.Present = presents[i];
            db.AttendanceModel.Add(a);
            db.SaveChanges();
        }
        return "Attendance Taken Succesfully.";
    }
    catch (Exception ex)
    {
        System.Diagnostics.Debug.Write(ex.Message);
        return "Something went wrong!";
    }
}

public IEnumerable<Attendance> GetAllAttendanceByTeacherAndDate(int tid, DateTime dt)
{
    IEnumerable<Attendance> lst = from a in db.AttendanceModel where a.T.Id == tid && a.Date ==dt select a;
    return lst;
}
public IEnumerable<Attendance> GetAllAttendanceByStudentAndTeacher(int sid, int tid)
{
    IEnumerable<Attendance> lst = from a in db.AttendanceModel where a.T.Id == tid && a.S.Id == sid select a;
    return lst;
}
```

2) CRUD (Student)

```
public List<Student> GetStudents()
{
    List<Student> students = db.StudentModel.ToList();
    return students;
}

public Student GetStudent(int id)
{
    Student student = db.StudentModel.Where(s => s.Id == id).FirstOrDefault();
    return student;
}
public string AddStudent(Student student)
{
    try
    {
        Student s_tmp = (from st in db.StudentModel where st.Class ==
                         student.Class && st.RollNo == student.RollNo select st ).FirstOrDefault();
        if(s_tmp != null)
        {
            return "Student Already Exist!";
        }
        Student s = new Student();
        s.Name = student.Name;
        s.Class = student.Class;
        s.RollNo = student.RollNo;

        db.StudentModel.Add(s);
        db.SaveChanges();
        return "Student Added Successfully.";
    }
    catch (Exception ex)
    {
        System.Diagnostics.Debug.Write(ex.Message);

        return "Something went wrong!";
    }
}
```

3) CRUD (Teacher)

```
public string UpdateTeacher(Teacher teacher)
{
    try
    {
        Teacher t_tmp = (from ts in db.TeacherModel where ts.Class == teacher.Class
                         && ts.Subject.ToLower() == teacher.Subject.ToLower() select ts).FirstOrDefault();
        if (t_tmp != null)
        {
            return "Teacher Already Exist!";
        }
        Teacher t = db.TeacherModel.Where(o => o.Id == teacher.Id).FirstOrDefault();
        t.Name = teacher.Name;
        t.Subject = teacher.Subject;
        t.Class = teacher.Class;
        t.Password = teacher.Password;
        db.SaveChanges();
        return "Teacher Updated Successfully.";
    }
    catch (Exception ex)
    {
        System.Diagnostics.Debug.Write(ex.Message);

        return "Something went wrong!";
    }

}

public String DeleteTeacher(int id)
{
    try
    {
        Teacher t = db.TeacherModel.Where(o => o.Id == id).FirstOrDefault();
        db.TeacherModel.Remove(t);
        db.SaveChanges();
        return "Teacher Deleted Successfully.";
    }
    catch (Exception ex)
    {
        System.Diagnostics.Debug.Write(ex.Message);

        return "Something went wrong!";
    }
}

public List<Teacher> GetTeachersByClass(int cls)
{
    List<Teacher> teachers = (from a in db.TeacherModel where a.Class == cls select a).ToList();
    return teachers;
}
```

Testing

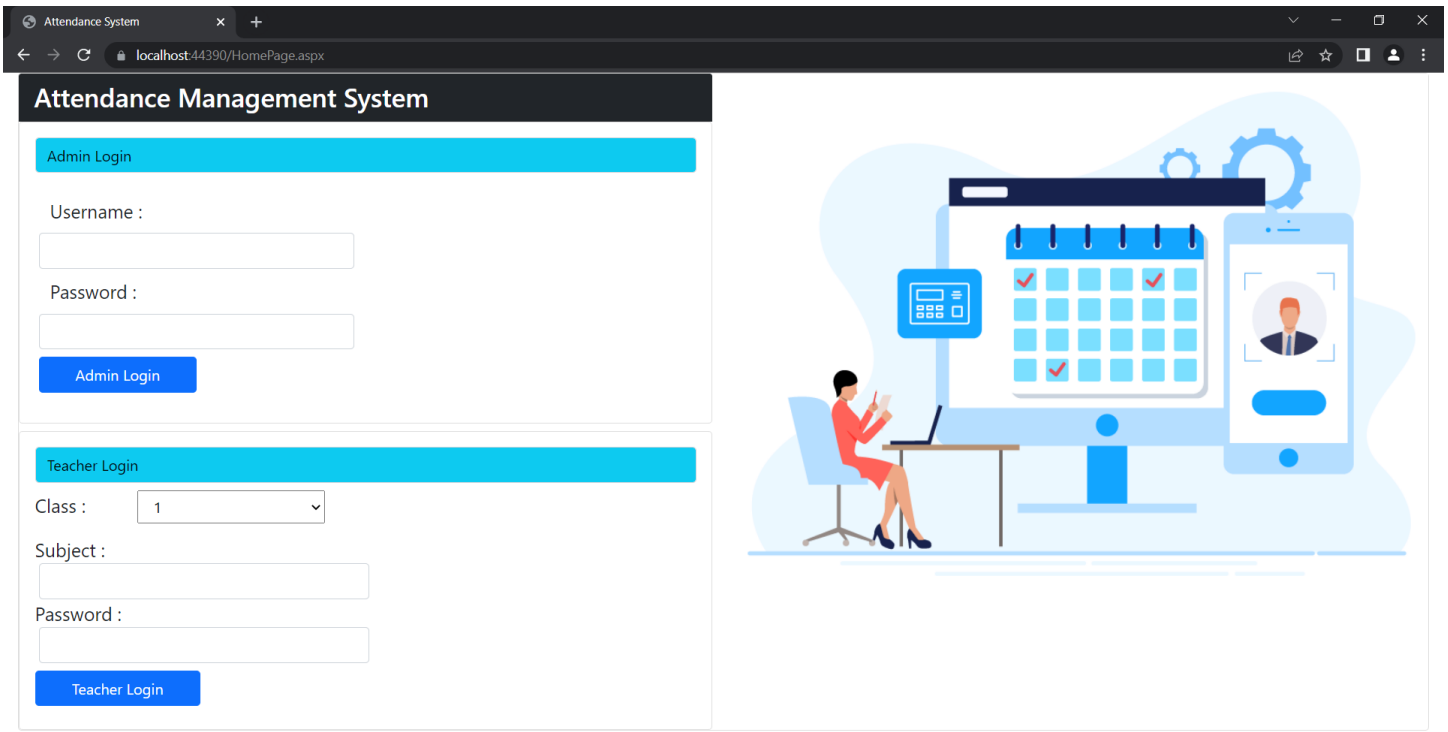
Testing Method: manual testing was performed.

Test Cases

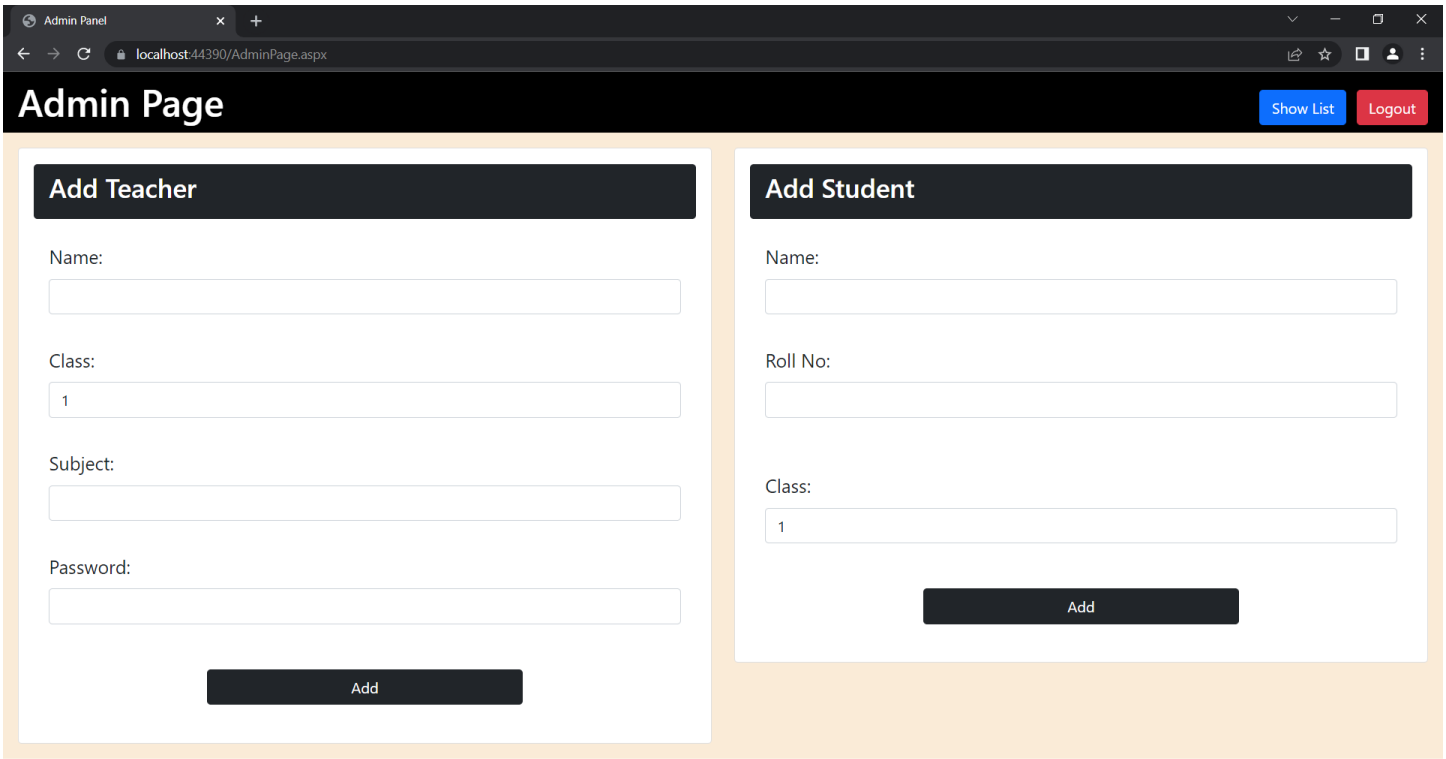
	SRS Id	Test Case Objective	Input Data	Expected Output	Actual Output	Status
1	R 1.1 R 2.1	Authentication of Admin and Teacher	Credential	Success(redirection) or error message	Success(redirection) or error message	Pass
2	R 1.2	Add Teacher	Teacher Details	Success or error message	Success or error message	Pass
3	R 1.3	Add Student	Student Details	Success or error message	Success or error message	Pass
4	R 1.4.1	Edit Student/Teacher	Student /Teacher's Updated Details	Updated details or error message	Updated details or error message	Pass
5	R 1.4.2	Delete Student/Teacher	Admin's Selection	Updated list	Updated list	Pass
6	R 1.4.3	View Attendance per Student	Selection of Student and Subject	Attendance list Of Student in selected Subject	Attendance list Of Student in selected Subject	Pass
7	R 1.4.4	View Attendance per Teacher	Selection of Teacher and Date	Attendance list Of Students in selected Teacher and Date	Attendance list Of Students in selected Teacher and Date	Pass
8	R 2.2	Take Attendance	Students list with presence	Success or error message	Success or error message	Pass
9	R 2.3	View Attendance	Teacher's Credential And Date selection	Attendance list Of Students as per selected date	Attendance list Of Students as per selected date	Pass

screenshots

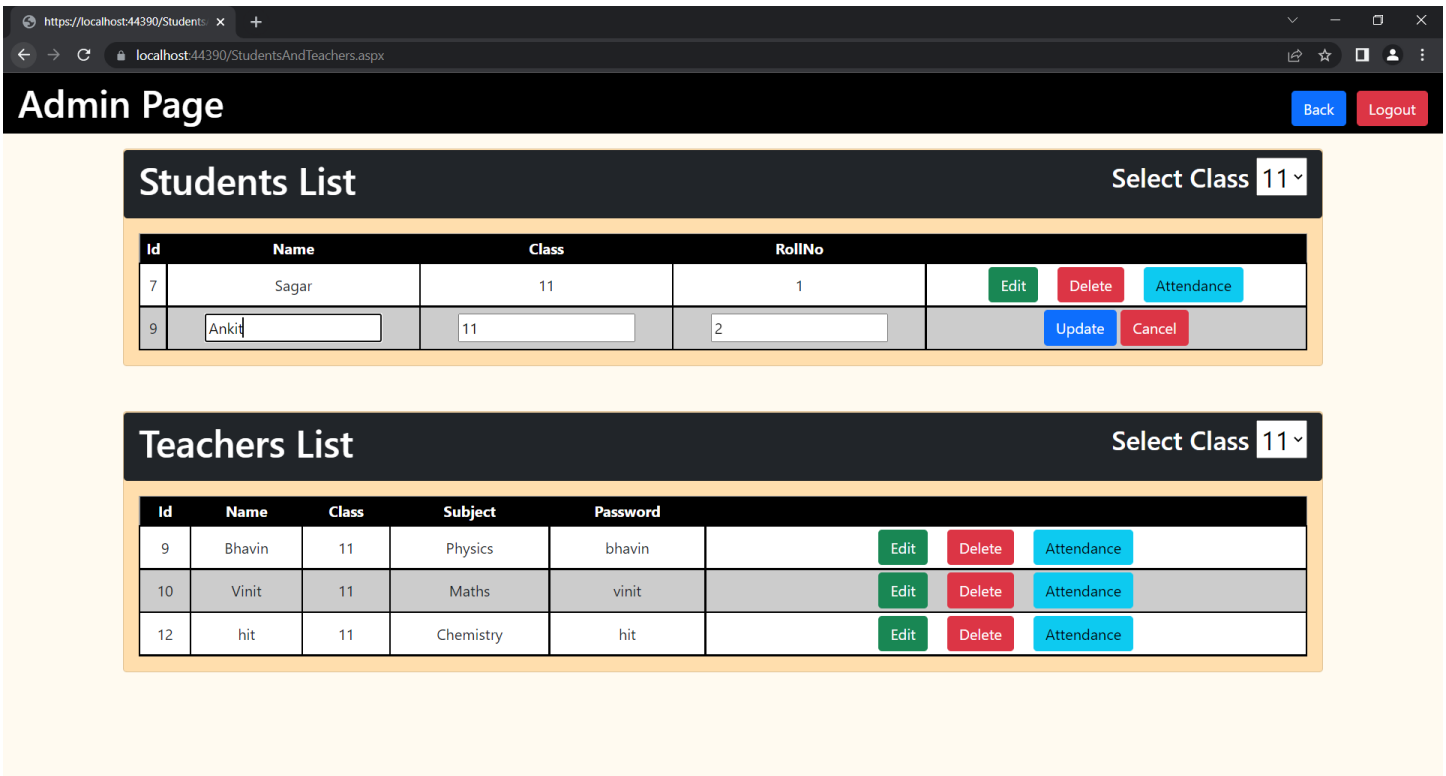
Home Page:



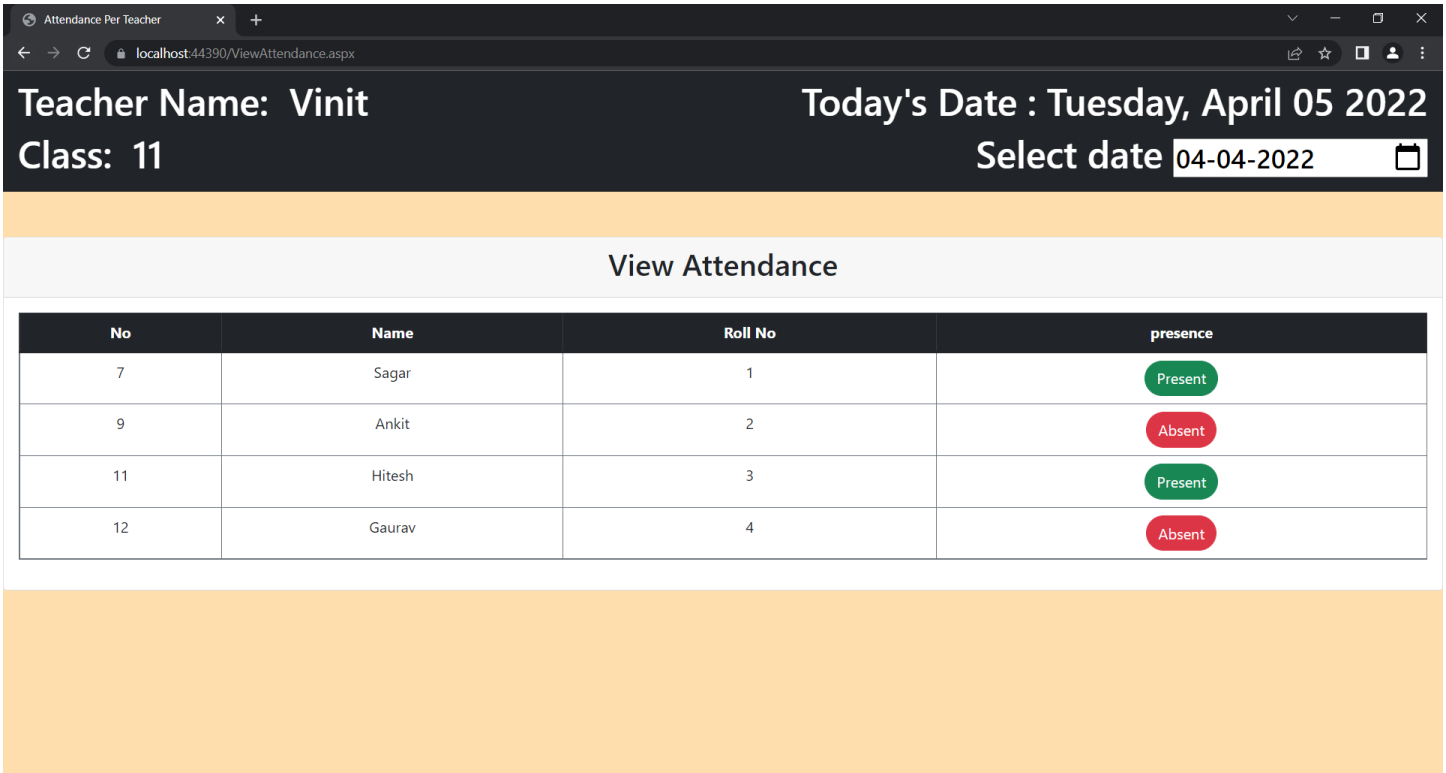
Admin Interface :



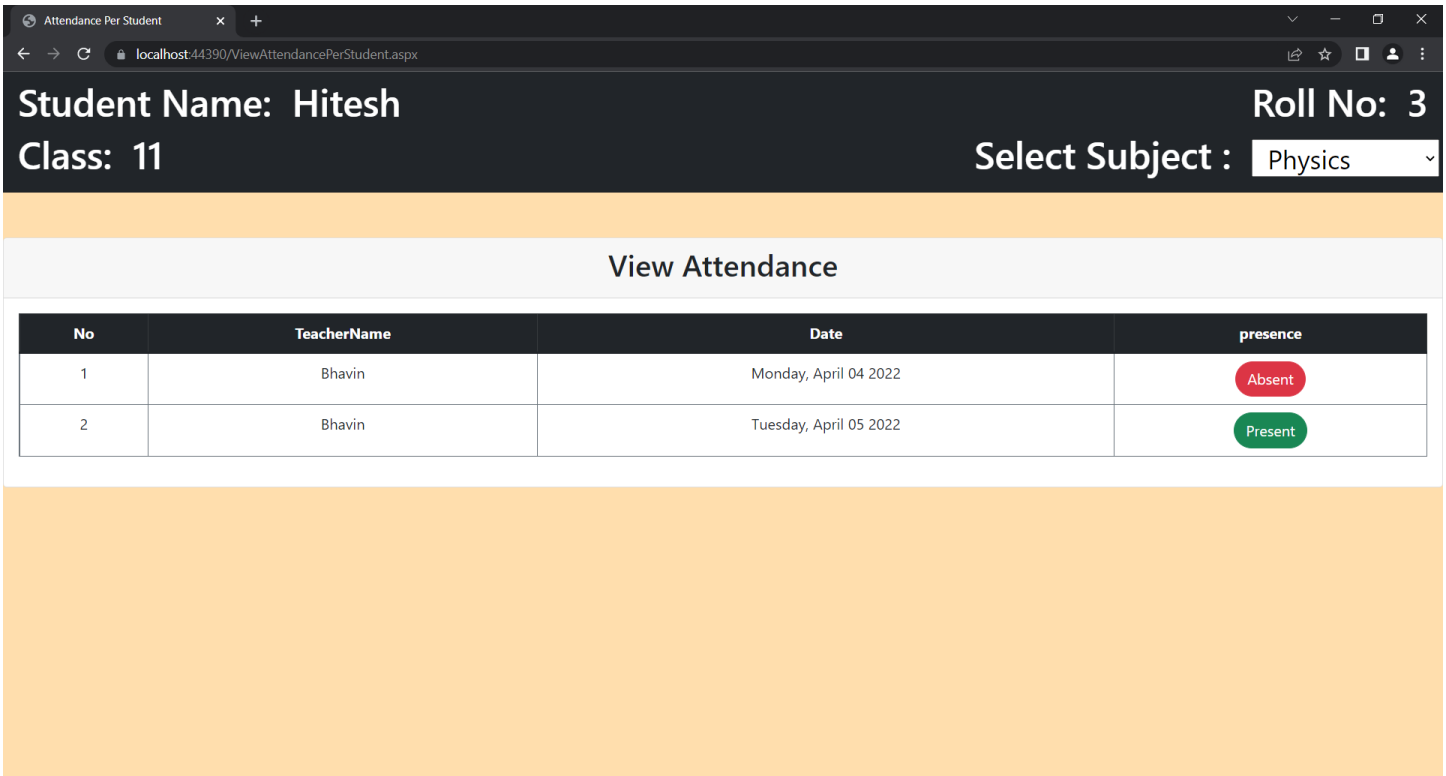
List Of Students And Teachers Per Class



View Attendance Per Teacher

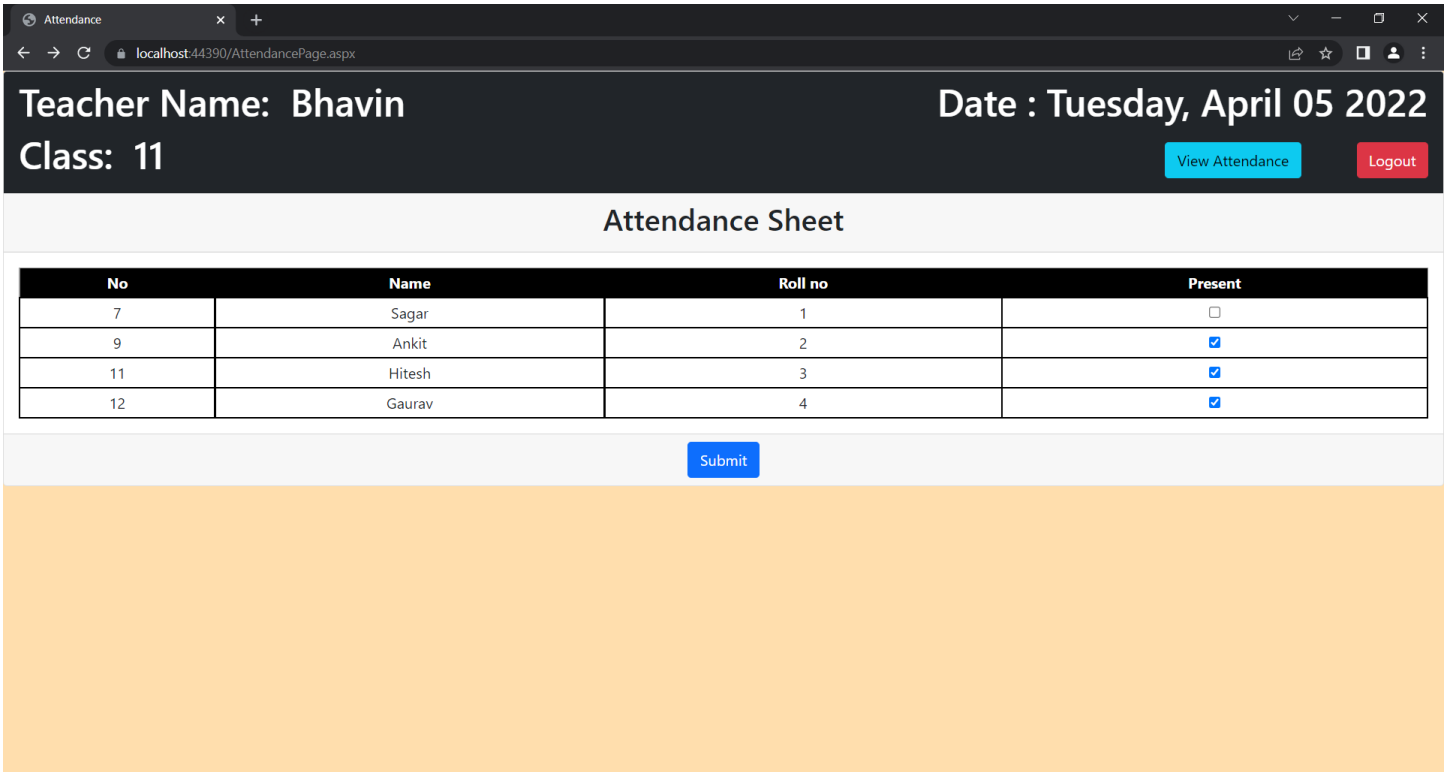


View Attendance Per Student

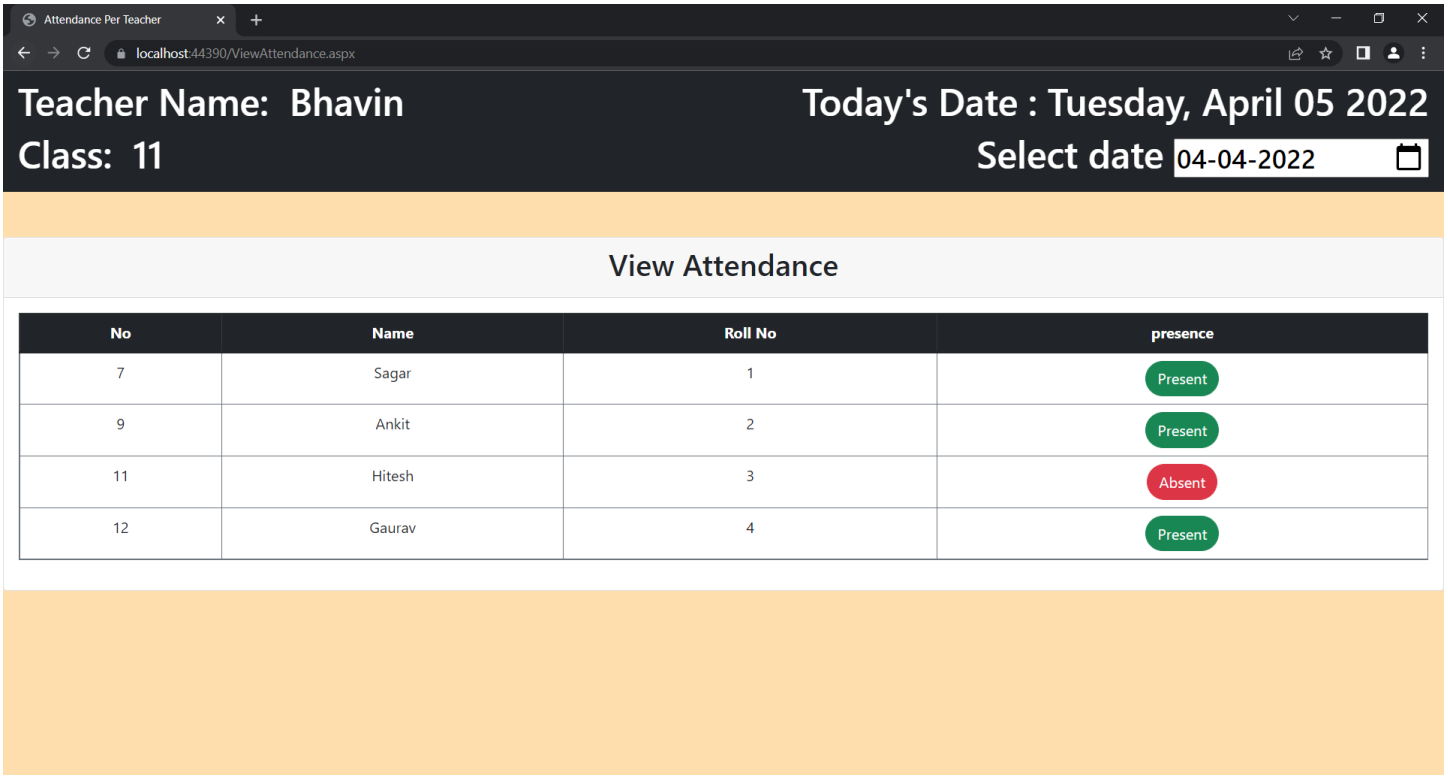


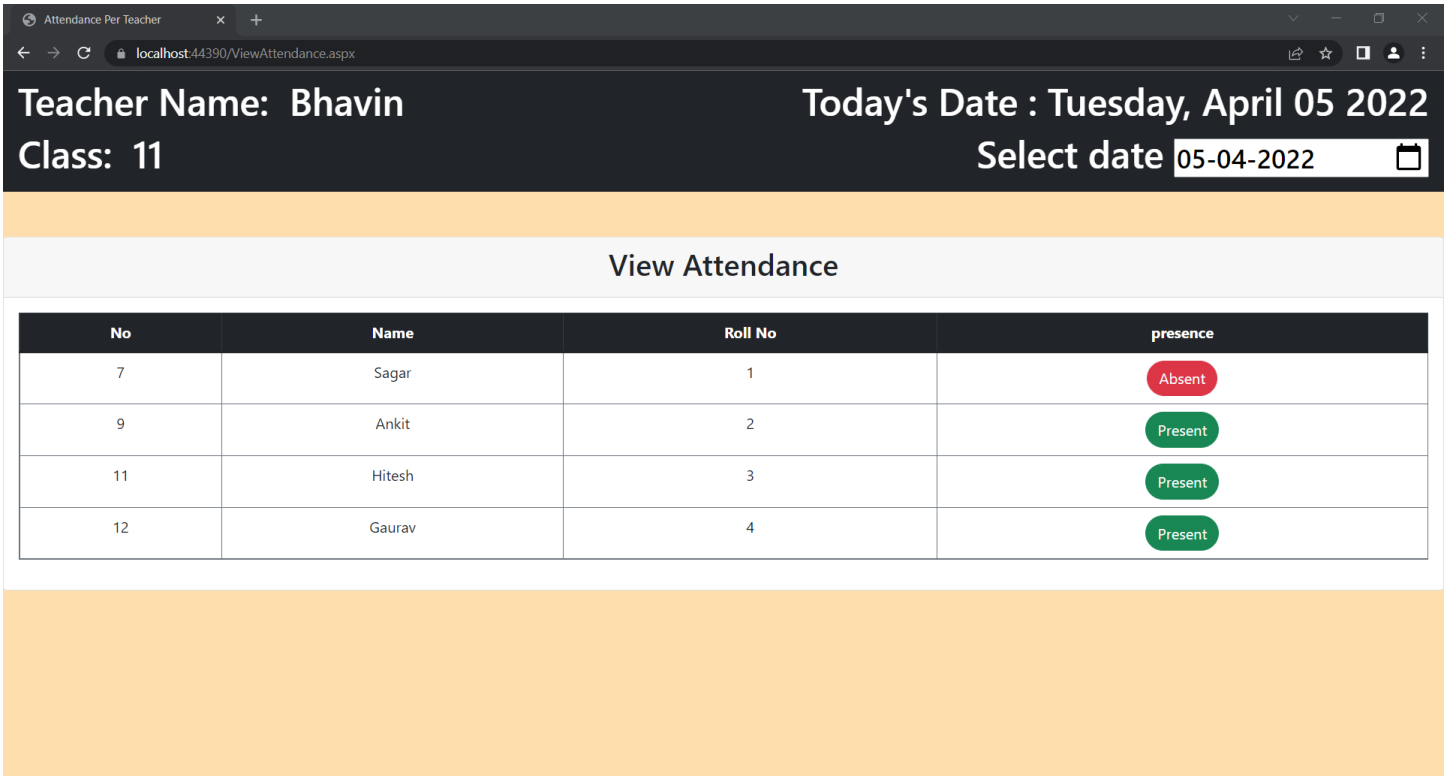
Teacher Interface:

Take Attendance



View Attendance





Conclusion

The functionalities are implemented in system after understanding all the system modules according to the requirements. Functionalities that are successfully implemented in the system are:

- Admin and Teacher Login
- Take attendance of Students (By Teacher)
- View attendance of Students (By Teacher on his/her subject)
- Students details management (By Admin)
- Teachers details management (By Admin)
- View attendance per Student with Subject (By Admin)
- View attendance per Teacher Date (By Admin)

Limitation and Future Enhancement

- In our system there is no option to update attendance once it's taken.
- In our system student interface is not available which can be provided in future extension.
- We can also provide more advance filter methods in attendance list.

Reference / Bibliography

- SERVICE-ORIENTED-ARCHITECTURE By Thomas Erl

Following links and websites were referred during the development of this project:

- Wcf
- .Net-Framework
- Entity-Framework
- Stackoverflow
- https://www.c-sharpcorner.com/