**Comprehensive Assessment**

BOOK CONTROLLER:

using LibraryManagementSystem.Interface;

using LibraryManagementSystem.Model;

using Microsoft.AspNetCore.Cors;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace LibraryManagementSystem.Controllers

{

[EnableCors("AllowOrigin")]

[Route("api/[controller]")]

[ApiController]

public class BookController : ControllerBase

{

private readonly IBook \_iBook;

public BookController(IBook iBook)

{

\_iBook = iBook;

}

// GET: api/<BookController>

[HttpGet("")]

public async Task<IEnumerable<Book>> GetBookAsync()

{

var details = await \_iBook.GetBookAsync();

return details;

}

// GET api/<BookController>/5

[HttpGet("{bookId}")]

[ActionName("GetBook")]

public async Task<IActionResult> GetBook([FromRoute] int bookId)

{

var detail = await \_iBook.GetBook(bookId);

if (detail != null)

{

return Ok(detail);

}

return NotFound("Not Found");

}

// POST api/<BookController>

[HttpPost("")]

public async Task<IActionResult> AddBook([FromBody] Book book)

{

await \_iBook.AddBook(book);

return CreatedAtAction(nameof(GetBook), new { bookId = book.BookId }, book);

}

// PUT api/<BookController>/5

[HttpPut("")]

public async Task<IActionResult> UpdateBookAsync([FromBody] Book book)

{

var detail = await \_iBook.UpdateBookAsync(book);

if (detail != null)

{

return Ok(detail);

}

else

{

return NotFound("Not found");

}

}

// DELETE api/<BookController>/5

[HttpDelete("{customerId}")]

public async Task<IActionResult> DeleteBookAsync([FromRoute] int bookId)

{

await \_iBook.DeleteBookAsync(bookId);

return Ok();

}

}

}

BOOK STATUS CONTROLLER:

using LibraryManagementSystem.Interface;

using LibraryManagementSystem.Model;

using Microsoft.AspNetCore.Cors;

using Microsoft.AspNetCore.Mvc;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

// For more information on enabling Web API for empty projects, visit

https://go.microsoft.com/fwlink/?LinkID=397860

namespace LibraryManagementSystem.Controllers

{

[EnableCors("AllowOrigin")]

[Route("api/[controller]")]

[ApiController]

public class BookStatusController : ControllerBase

{

private readonly IBookStatus \_bookStatus;

public BookStatusController(IBookStatus bookStatus)

{

\_bookStatus = bookStatus;

}

// GET: api/<BookStatusController>

[HttpGet]

public async Task<IEnumerable<BookStatus>> Get()

{

return await \_bookStatus.GetBookStatus();

}

// GET api/<BookStatusController>/5

[HttpGet("{id}")]

public async Task<IActionResult> Get(int id)

{

BookStatus bookStatus = await \_bookStatus.GetBookStatus(id);

if (bookStatus == null)

{

return NotFound("Not found the bookStatus");

}

return Ok(bookStatus);

}

// POST api/<BookStatusController>

[HttpPost]

public async Task<IActionResult> Post([FromBody] BookStatus bookStatus)

{

await \_bookStatus.AddBookStatus(bookStatus);

return Ok("Successfully added");

}

// PUT api/<BookStatusController>/5

[HttpPut]

public async Task<IActionResult> Put([FromBody] BookStatus bookStatus)

{

BookStatus bookStatus1 = await \_bookStatus.UpdateBookStatus(bookStatus);

if (bookStatus1 == null)

{

return NotFound("Not found");

}

return Ok(bookStatus1);

}

// DELETE api/<BookStatusController>/5

[HttpDelete("{id}")]

public async Task<IActionResult> Delete(int id)

{

await \_bookStatus.DeleteBookStatus(id);

return Ok("Successfully deleted");

}

}

}

DESIGNATION CONTROLLER:

using LibraryManagementSystem.Interface;

using LibraryManagementSystem.Model;

using Microsoft.AspNetCore.Cors;

using Microsoft.AspNetCore.Mvc;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

// For more information on enabling Web API for empty projects, visit

https://go.microsoft.com/fwlink/?LinkID=397860

namespace LibraryManagementSystem.Controllers

{

[EnableCors("AllowOrigin")]

[Route("api/[controller]")]

[ApiController]

public class DesignationController : ControllerBase

{

private readonly IDesignation \_designation;

public DesignationController(IDesignation designation)

{

\_designation = designation;

}

// GET: api/<DesignationController>

[HttpGet]

public async Task<IEnumerable<Designation>> Get()

{

return await \_designation.GetDesignations();

}

// GET api/<DesignationController>/5

[HttpGet("{id}")]

public async Task<IActionResult> Get(int id)

{

Designation designation = await \_designation.GetDesignation(id);

if (designation == null)

{

return NotFound("Not found");

}

return Ok(designation);

}

// POST api/<DesignationController>

[HttpPost]

public async Task<IActionResult> Post([FromBody] Designation designation)

{

await \_designation.AddDesignation(designation);

return Ok("Successfully added");

}

// PUT api/<DesignationController>/5

[HttpPut]

public async Task<IActionResult> Put([FromBody] Designation designation)

{

Designation designation1 = await \_designation.UpdateDesignation(designation);

if (designation1 == null)

{

return NotFound("Not found");

}

return Ok(designation1);

}

// DELETE api/<DesignationController>/5

[HttpDelete("{id}")]

public async Task<IActionResult> Delete(int id)

{

await \_designation.DeleteDesignation(id);

return Ok("Successfully deleted");

}

}

}

EBOOK CONTROLLER:

using LibraryManagementSystem.Interface;

using LibraryManagementSystem.Model;

using Microsoft.AspNetCore.Cors;

using Microsoft.AspNetCore.Mvc;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

// For more information on enabling Web API for empty projects, visit

https://go.microsoft.com/fwlink/?LinkID=397860

namespace LibraryManagementSystem.Controllers

{

[EnableCors("AllowOrigin")]

[Route("api/[controller]")]

[ApiController]

public class EBookController : ControllerBase

{

private readonly IEBook \_eBook;

public EBookController(IEBook eBook)

{

\_eBook = eBook;

}

// GET: api/<EBookController>

[HttpGet]

public async Task<IEnumerable<EBook>> Get()

{

return await \_eBook.GetEbookAsync();

}

// GET api/<EBookController>/5

[HttpGet("{id}")]

public async Task<IActionResult> Get(int id)

{

EBook eBook = await \_eBook.GetEbook(id);

if (eBook == null)

{

return NotFound("Not found the EBook");

}

return Ok(eBook);

}

// POST api/<EBookController>

[HttpPost]

public async Task<IActionResult> Post([FromBody] EBook eBook)

{

await \_eBook.AddEbook(eBook);

return Ok("Successfully added");

}

// PUT api/<EBookController>/5

[HttpPut]

public async Task<IActionResult> Put([FromBody] EBook eBook)

{

EBook eBook1 = await \_eBook.UpdateEbookAsync(eBook);

if (eBook1 == null)

{

return NotFound("Not found");

}

return Ok(eBook1);

}

// DELETE api/<EBookController>/5

[HttpDelete("{id}")]

public async Task<IActionResult> Delete(int id)

{

await \_eBook.DeleteEbookAsync(id);

return Ok("Successfully deleted");

}

}

}

FACULTY CONTROLLER:

using LibraryManagementSystem.Interface;

using LibraryManagementSystem.Model;

using Microsoft.AspNetCore.Cors;

using Microsoft.AspNetCore.Mvc;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

// For more information on enabling Web API for empty projects, visit

https://go.microsoft.com/fwlink/?LinkID=397860

namespace LibraryManagementSystem.Controllers

{

[EnableCors("AllowOrigin")]

[Route("api/[controller]")]

[ApiController]

public class FacultyController : ControllerBase

{

private readonly IFaculty \_faculty;

public FacultyController(IFaculty faculty)

{

\_faculty = faculty;

}

// GET: api/<FacultyController>

[HttpGet]

public async Task<IEnumerable<Faculty>> Get()

{

return await \_faculty.GetFacultys();

}

// GET api/<FacultyController>/5

[HttpGet("{id}")]

public async Task<IActionResult> Get(int id)

{

Faculty faculty = await \_faculty.GetFaculty(id);

if (faculty == null)

{

return NotFound("Not found");

}

return Ok(faculty);

}

// POST api/<FacultyController>

[HttpPost]

public async Task<IActionResult> Post([FromBody] Faculty faculty)

{

await \_faculty.AddFaculty(faculty);

return Ok("Successfully added");

}

// PUT api/<FacultyController>/5

[HttpPut]

public async Task<IActionResult> Put( [FromBody] Faculty faculty)

{

Faculty faculty1 = await \_faculty.UpdateFaculty(faculty);

if (faculty1 == null)

{

return NotFound("Not found");

}

return Ok(faculty1);

}

// DELETE api/<FacultyController>/5

[HttpDelete("{id}")]

public async Task<IActionResult> Delete(int id)

{

await \_faculty.DeleteFaculty(id);

return Ok("Successfully deleted");

}

}

}

STUDENT CONTROLLER:

using LibraryManagementSystem.Interface;

using LibraryManagementSystem.Model;

using Microsoft.AspNetCore.Cors;

using Microsoft.AspNetCore.Mvc;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

// For more information on enabling Web API for empty projects, visit

https://go.microsoft.com/fwlink/?LinkID=397860

namespace LibraryManagementSystem.Controllers

{

[EnableCors("AllowOrigin")]

[Route("api/[controller]")]

[ApiController]

public class StudentController : ControllerBase

{

private readonly IStudent \_iStudent;

public StudentController(IStudent student)

{

\_iStudent = student;

}

// GET: api/<StudentController>

[HttpGet]

public async Task<IEnumerable<Student>> Get()

{

return await \_iStudent.GetStudents();

}

// GET api/<StudentController>/5

[HttpGet("{id}")]

public async Task<IActionResult> Get(int id)

{

Student student = await \_iStudent.GetStudent(id);

if (student == null)

{

return NotFound("Not found");

}

return Ok(student);

}

// POST api/<StudentController>

[HttpPost]

public async Task<IActionResult> Post([FromBody] Student student)

{

await \_iStudent.AddStudent(student);

return Ok("Successfully added");

}

// PUT api/<StudentController>/5

[HttpPut]

public async Task<IActionResult> Put([FromBody] Student student)

{

Student student1 = await \_iStudent.UpdateStudent(student);

if (student1 == null)

{

return NotFound("Not found");

}

return Ok(student);

}

// DELETE api/<StudentController>/5

[HttpDelete("{id}")]

public async Task<IActionResult> Delete(int id)

{

await \_iStudent.DeleteStudent(id);

return Ok("Successfully deleted");

}

}

}

SUPPLIER CONTROLLER:

using LibraryManagementSystem.Interface;

using LibraryManagementSystem.Model;

using LibraryManagementSystem.Repository;

using Microsoft.AspNetCore.Cors;

using Microsoft.AspNetCore.Mvc;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

// For more information on enabling Web API for empty projects, visit

https://go.microsoft.com/fwlink/?LinkID=397860

namespace LibraryManagementSystem.Controllers

{

[EnableCors("AllowOrigin")]

[Route("api/[controller]")]

[ApiController]

public class SupplierController : ControllerBase

{

private readonly ISupplier \_iSupplier;

public SupplierController(ISupplier supplier)

{

\_iSupplier = supplier;

}

// GET: api/<SupplierController>

[HttpGet]

public async Task<IEnumerable<Supplier>> Get()

{

return await \_iSupplier.GetSuppliers();

}

// GET api/<SupplierController>/5

[HttpGet("{id}")]

public async Task<IActionResult> Get(int id)

{

Supplier supplier = await \_iSupplier.GetSupplier(id);

if(supplier==null)

{

return NotFound("Not found");

}

return Ok(supplier);

}

// POST api/<SupplierController>

[HttpPost]

public async Task<IActionResult> Post([FromBody] Supplier supplier)

{

await \_iSupplier.AddSupplier(supplier);

return Ok("Successfully added");

}

// PUT api/<SupplierController>/5

[HttpPut]

public async Task<IActionResult> Put([FromBody] Supplier supplier)

{

Supplier suplier = await \_iSupplier.UpdateSupplierDetail(supplier);

if(supplier==null)

{

return NotFound("Not found");

}

return Ok(suplier);

}

// DELETE api/<SupplierController>/5

[HttpDelete("{id}")]

public async Task<IActionResult> Delete(int id)

{

await \_iSupplier.DeleteSupplier(id);

return Ok("Successfully deleted");

}

}

}

INTERFACE :

IBOOK INTERFACE :

using LibraryManagementSystem.Model;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace LibraryManagementSystem.Interface

{

public interface IBook

{

Task<IEnumerable<Book>> GetBookAsync();

Task<Book> GetBook(int bookid);

Task AddBook(Book book);

Task<Book> UpdateBookAsync(Book book);

Task DeleteBookAsync(int bookId);

}

}

IBOOKSTATUS INTERFACE:

using LibraryManagementSystem.Model;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace LibraryManagementSystem.Interface

{

public interface IBookStatus

{

Task<IEnumerable<BookStatus>> GetBookStatus();

Task<BookStatus> GetBookStatus(int bookStatusId);

Task AddBookStatus(BookStatus bookStatus);

Task<BookStatus> UpdateBookStatus(BookStatus bookStatus);

Task DeleteBookStatus(int bookStatusId);

}

}

IDESGINATION:

using LibraryManagementSystem.Model;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace LibraryManagementSystem.Interface

{

public interface IDesignation

{

Task<IEnumerable<Designation>> GetDesignations();

Task<Designation> GetDesignation(int designationId);

Task AddDesignation(Designation designation);

Task<Designation> UpdateDesignation(Designation designation);

Task DeleteDesignation(int designationId);

}

}

IEBOOK:

using LibraryManagementSystem.Model;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace LibraryManagementSystem.Interface

{

public interface IEBook

{

Task<IEnumerable<EBook>> GetEbookAsync();

Task<EBook> GetEbook(int eBookId);

Task AddEbook(EBook eBook);

Task<EBook> UpdateEbookAsync(EBook eBook);

Task DeleteEbookAsync(int eBookId);

}

}

IFACULTY:

using LibraryManagementSystem.Model;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace LibraryManagementSystem.Interface

{

public interface IFaculty

{

Task<IEnumerable<Faculty>> GetFacultys();

Task AddFaculty(Faculty faculty);

Task<Faculty> UpdateFaculty(Faculty faculty);

Task DeleteFaculty(int id);

Task<Faculty> GetFaculty(int id);

}

}

ISTUDENT:

using LibraryManagementSystem.Model;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace LibraryManagementSystem.Interface

{

public interface IStudent

{

Task<IEnumerable<Student>> GetStudents();

Task<Student> GetStudent(int studentId);

Task AddStudent(Student student);

Task<Student> UpdateStudent(Student student);

Task DeleteStudent(int studentId);

}

}

ISUPPLIER:

using LibraryManagementSystem.Model;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace LibraryManagementSystem.Interface

{

public interface ISupplier

{

Task<IEnumerable<Supplier>> GetSuppliers();

Task<Supplier> GetSupplier(int supplierId);

Task AddSupplier(Supplier supplier);

Task<Supplier> UpdateSupplierDetail(Supplier supplier);

Task DeleteSupplier(int supplierId);

}

}

DB CONTEXT:

using LibraryManagementSystem.Model;

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace LibraryManagementSystem.DBContext

{

public class LibraryDBContext : DbContext

{

public LibraryDBContext() {}

public LibraryDBContext(DbContextOptions<LibraryDBContext> dbContextOptions) :

base(dbContextOptions) {}

public DbSet<Book> Books { get; set; }

public DbSet<BookStatus> BookStatus { get; set; }

public DbSet<Designation> Designations { get; set; }

public DbSet<EBook> EBooks { get; set; }

public DbSet<Faculty> Faculties { get; set; }

public DbSet<Student> Students { get; set; }

public DbSet<Supplier> Suppliers { get; set; }

}

}

MODEL:

BOOK:

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

using System.Threading.Tasks;

namespace LibraryManagementSystem.Model

{

public class Book

{

[Key]

[DatabaseGenerated(DatabaseGeneratedOption.Identity)]

public int BookId { get; set; }

public string BookName { get; set; }

public string ISBN { get; set; }

public string Description { get; set; }

public string Publisher { get; set; }

public string Author { get; set; }

public string Location { get; set; }

public int Quantity { get; set; }

public int Issued { get; set; }

[ForeignKey("BookStatusId")]

public BookStatus BookStatus { get; set; }

public int BookStatusId { get; set; }

}

}

BOOK STATUS:

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

using System.Threading.Tasks;

namespace LibraryManagementSystem.Model

{

public class BookStatus

{

[Key]

[DatabaseGenerated(DatabaseGeneratedOption.Identity)]

public int BookStatusId { get; set; }

[MaxLength(50)]

public string Status { get; set; }

}

}

DESIGNATION:

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.SqlServer;

namespace LibraryManagementSystem.Model

{

public class Designation

{

[Key]

[DatabaseGenerated(DatabaseGeneratedOption.Identity)]

public int DesignationId { get; set; }

[MaxLength(50)]

public string DesignationName { get; set; }

}

}

EBOOK:

using Microsoft.AspNetCore.Http;

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

using System.Threading.Tasks;

namespace LibraryManagementSystem.Model

{

public class EBook

{

[Key]

[DatabaseGenerated(DatabaseGeneratedOption.Identity)]

public int EBookId { get; set; }

public string EBookName { get; set; }

public string ISBN { get; set; }

public string Description { get; set; }

public string Publisher { get; set; }

public string Author { get; set; }

//public IFormFile File { get; set; }

}

}

FACULTY:

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

using System.Threading.Tasks;

namespace LibraryManagementSystem.Model

{

public class Faculty

{

[Key]

public int FacultyId { get; set; }

public string FacultyName { get; set; }

public string FacultyEmail { get; set; }

[MaxLength(50)]

public string FacultyPhone { get; set; }

public string FacultyAddress { get; set; }

public string DesignationName { get; set; }

}

}

STUDENT:

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

using System.Threading.Tasks;

namespace LibraryManagementSystem.Model

{

public class Student

{

[Key]

[DatabaseGenerated(DatabaseGeneratedOption.Identity)]

public int StudentId { get; set; }

public string StudentName { get; set; }

public string StudentEmail { get; set; }

public string FatherName { get; set; }

[MaxLength(50)]

public string Password { get; set; }

[MaxLength(50)]

public string Phone { get; set; }

public string Address { get; set; }

[MaxLength(50)]

public string Class { get; set; }

public int RollNo { get; set; }

}

}

SUPPLIER:

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

using System.Threading.Tasks;

namespace LibraryManagementSystem.Model

{

public class Supplier

{

[Key]

[DatabaseGenerated(DatabaseGeneratedOption.Identity)]

public int SupplierId { get; set; }

public string SupplierName { get; set; }

public int PurchaseNumber { get; set; }

}

}

REPOSITORY:

BOOK REPOSITORY:

using LibraryManagementSystem.DBContext;

using LibraryManagementSystem.Interface;

using LibraryManagementSystem.Model;

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace LibraryManagementSystem.Repository

{

public class BookRepository:IBook

{

private LibraryDBContext \_context;

public BookRepository(LibraryDBContext context)

{

\_context = context;

}

public async Task AddBook(Book book)

{

await \_context.Books.AddAsync(book);

await \_context.SaveChangesAsync();

}

public async Task DeleteBookAsync(int bookId)

{

Book bk = new Book()

{

BookId = bookId

};

\_context.Books.Remove(bk);

await \_context.SaveChangesAsync();

}

public async Task<Book> GetBook(int bookid)

{

var book = await \_context.Books.FindAsync(bookid);

return book;

}

public async Task<IEnumerable<Book>> GetBookAsync()

{

var records = await \_context.Books.ToListAsync();

return records;

}

public async Task<Book> UpdateBookAsync(Book book)

{

var bk = new Book()

{

BookId = book.BookId,

BookName = book.BookName,

ISBN = book.ISBN,

Description = book.Description,

Publisher = book.Publisher,

Author = book.Author,

Location = book.Location,

Quantity = book.Quantity,

Issued = book.Issued,

BookStatus = book.BookStatus

};

\_context.Books.Update(bk);

await \_context.SaveChangesAsync();

return bk;

}

}

}

BOOK STATUS REPOSITORY:

using LibraryManagementSystem.DBContext;

using LibraryManagementSystem.Interface;

using LibraryManagementSystem.Model;

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace LibraryManagementSystem.Repository

{

public class BookStatusRespository : IBookStatus

{

private readonly LibraryDBContext \_libraryDBContext;

public BookStatusRespository(LibraryDBContext libraryDBContext)

{

\_libraryDBContext = libraryDBContext;

}

public async Task AddBookStatus(BookStatus bookStatus)

{

await \_libraryDBContext.BookStatus.AddAsync(bookStatus);

await \_libraryDBContext.SaveChangesAsync();

}

public async Task DeleteBookStatus(int bookStatusId)

{

BookStatus bookStatus = new BookStatus()

{

BookStatusId = bookStatusId

};

\_libraryDBContext.BookStatus.Remove(bookStatus);

await \_libraryDBContext.SaveChangesAsync();

}

public async Task<IEnumerable<BookStatus>> GetBookStatus()

{

return await \_libraryDBContext.BookStatus.ToListAsync();

}

public async Task<BookStatus> GetBookStatus(int bookStatusId)

{

return await \_libraryDBContext.BookStatus.FindAsync(bookStatusId);

}

public async Task<BookStatus> UpdateBookStatus(BookStatus bookStatus)

{

\_libraryDBContext.BookStatus.Update(bookStatus);

await \_libraryDBContext.SaveChangesAsync();

return bookStatus;

}

}

}

DESIGNATION REPOSITORY:

using LibraryManagementSystem.DBContext;

using LibraryManagementSystem.Interface;

using LibraryManagementSystem.Model;

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace LibraryManagementSystem.Repository

{

public class DesignationRepository : IDesignation

{

private readonly LibraryDBContext \_libraryDBContext;

public DesignationRepository(LibraryDBContext libraryDBContext)

{

\_libraryDBContext = libraryDBContext;

}

public async Task AddDesignation(Designation designation)

{

await \_libraryDBContext.Designations.AddAsync(designation);

await \_libraryDBContext.SaveChangesAsync();

}

public async Task DeleteDesignation(int designationId)

{

Designation designation = new Designation()

{

DesignationId = designationId

};

\_libraryDBContext.Designations.Remove(designation);

await \_libraryDBContext.SaveChangesAsync();

}

public async Task<Designation> GetDesignation(int designationId)

{

return await \_libraryDBContext.Designations.FindAsync(designationId);

}

public async Task<IEnumerable<Designation>> GetDesignations()

{

return await \_libraryDBContext.Designations.ToListAsync();

}

public async Task<Designation> UpdateDesignation(Designation designation)

{

\_libraryDBContext.Designations.Update(designation);

await \_libraryDBContext.SaveChangesAsync();

return designation;

}

}

}

EBOOK REPOSITORY:

using LibraryManagementSystem.DBContext;

using LibraryManagementSystem.Interface;

using LibraryManagementSystem.Model;

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace LibraryManagementSystem.Repository

{

public class EBookRespository : IEBook

{

private readonly LibraryDBContext \_libraryDBContext;

public EBookRespository(LibraryDBContext libraryDBContext)

{

\_libraryDBContext = libraryDBContext;

}

public async Task AddEbook(EBook eBook)

{

await \_libraryDBContext.EBooks.AddAsync(eBook);

await \_libraryDBContext.SaveChangesAsync();

}

public async Task DeleteEbookAsync(int eBookId)

{

EBook eBook = new EBook()

{

EBookId = eBookId

};

\_libraryDBContext.EBooks.Remove(eBook);

await \_libraryDBContext.SaveChangesAsync();

}

public async Task<EBook> GetEbook(int eBookId)

{

return await \_libraryDBContext.EBooks.FindAsync(eBookId);

}

public async Task<IEnumerable<EBook>> GetEbookAsync()

{

return await \_libraryDBContext.EBooks.ToListAsync();

}

public async Task<EBook> UpdateEbookAsync(EBook eBook)

{

\_libraryDBContext.EBooks.Update(eBook);

await \_libraryDBContext.SaveChangesAsync();

return eBook;

}

}

}

FACULTY REPOSITORY:

using LibraryManagementSystem.DBContext;

using LibraryManagementSystem.Interface;

using LibraryManagementSystem.Model;

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace LibraryManagementSystem.Repository

{

public class FacultyRepository : IFaculty

{

private readonly LibraryDBContext \_libraryDBContext;

public FacultyRepository(LibraryDBContext libraryDBContext)

{

\_libraryDBContext = libraryDBContext;

}

public async Task AddFaculty(Faculty faculty)

{

await \_libraryDBContext.Faculties.AddAsync(faculty);

await \_libraryDBContext.SaveChangesAsync();

}

public async Task DeleteFaculty(int facultyId)

{

Faculty faculty = new Faculty()

{

FacultyId = facultyId

};

\_libraryDBContext.Faculties.Remove(faculty);

await \_libraryDBContext.SaveChangesAsync();

}

public async Task<Faculty> GetFaculty(int facultyId)

{

return await \_libraryDBContext.Faculties.FindAsync(facultyId);

}

public async Task<IEnumerable<Faculty>> GetFacultys()

{

return await \_libraryDBContext.Faculties.ToListAsync();

}

public async Task<Faculty> UpdateFaculty(Faculty faculty)

{

\_libraryDBContext.Faculties.Update(faculty);

await \_libraryDBContext.SaveChangesAsync();

return faculty;

}

}

}

STUDENT REPOSITORY:

using LibraryManagementSystem.DBContext;

using LibraryManagementSystem.Interface;

using LibraryManagementSystem.Model;

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace LibraryManagementSystem.Repository

{

public class StudentRepository : IStudent

{

private readonly LibraryDBContext \_libraryDBContext;

public StudentRepository(LibraryDBContext libraryDBContext)

{

\_libraryDBContext = libraryDBContext;

}

public async Task AddStudent(Student student)

{

await \_libraryDBContext.Students.AddAsync(student);

await \_libraryDBContext.SaveChangesAsync();

}

public async Task DeleteStudent(int studentId)

{

Student student = new Student()

{

StudentId = studentId

};

\_libraryDBContext.Students.Remove(student);

await \_libraryDBContext.SaveChangesAsync();

}

public async Task<Student> GetStudent(int studentId)

{

return await \_libraryDBContext.Students.FindAsync(studentId);

}

public async Task<IEnumerable<Student>> GetStudents()

{

return await \_libraryDBContext.Students.ToListAsync();

}

public async Task<Student> UpdateStudent(Student student)

{

\_libraryDBContext.Students.Update(student);

await \_libraryDBContext.SaveChangesAsync();

return student;

}

}

}

SUPPLIER REPOSITORY:

using LibraryManagementSystem.DBContext;

using LibraryManagementSystem.Interface;

using LibraryManagementSystem.Model;

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace LibraryManagementSystem.Repository

{

public class SupplierRepository : ISupplier

{

private readonly LibraryDBContext \_libraryDBContext;

public SupplierRepository(LibraryDBContext libraryDBContext)

{

\_libraryDBContext = libraryDBContext;

}

public async Task AddSupplier(Supplier supplier)

{

await \_libraryDBContext.Suppliers.AddAsync(supplier);

await \_libraryDBContext.SaveChangesAsync();

}

public async Task DeleteSupplier(int supplierId)

{

Supplier supplier = new Supplier()

{

SupplierId = supplierId

};

\_libraryDBContext.Suppliers.Remove(supplier);

await \_libraryDBContext.SaveChangesAsync();

}

public async Task<Supplier> GetSupplier(int supplierId)

{

return await \_libraryDBContext.Suppliers.FindAsync(supplierId);

}

public async Task<IEnumerable<Supplier>> GetSuppliers()

{

return await \_libraryDBContext.Suppliers.ToListAsync();

}

public async Task<Supplier> UpdateSupplierDetail(Supplier supplier)

{

\_libraryDBContext.Suppliers.Update(supplier);

await \_libraryDBContext.SaveChangesAsync();

return supplier;

}

}

}

APP SETTINGS :

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft": "Warning",

"Microsoft.Hosting.Lifetime": "Information"

}

},

"ConnectionStrings": {

"ConnString": "Data Source =MINDJAN387; Initial Catalog =LibraryManagementSystem; User

ID=sa;Password=pass@word1"

},

"AllowedHosts": "\*"

}

PROGRAM.CS:

using Microsoft.AspNetCore.Hosting;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.Hosting;

using Microsoft.Extensions.Logging;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace LibraryManagementSystem

{

public class Program

{

public static void Main(string[] args)

{

CreateHostBuilder(args).Build().Run();

}

public static IHostBuilder CreateHostBuilder(string[] args) =>

Host.CreateDefaultBuilder(args)

.ConfigureWebHostDefaults(webBuilder =>

{

webBuilder.UseStartup<Startup>();

});

}

}

STARTUP.CS

using LibraryManagementSystem.DBContext;

using LibraryManagementSystem.Interface;

using LibraryManagementSystem.Repository;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.HttpsPolicy;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using Microsoft.Extensions.Logging;

using Microsoft.OpenApi.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace LibraryManagementSystem

{

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

// This method gets called by the runtime. Use this method to add services to the

container.

public void ConfigureServices(IServiceCollection services)

{

//services.AddCors();

services.AddControllers();

services.AddDbContext<LibraryDBContext>(options =>

options.UseSqlServer(Configuration.GetConnectionString("ConnString")));

services.AddCors(options =>

{

options.AddPolicy("AllowOrigin",builder=>

{

builder.AllowAnyOrigin().AllowAnyMethod().AllowAnyHeader();

});

});

services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo { Title = "LibraryManagementSystem",

Version = "v1" });

});

services.AddTransient<ISupplier, SupplierRepository>();

services.AddTransient<IStudent, StudentRepository>();

services.AddTransient<IBook, BookRepository>();

services.AddTransient<IFaculty, FacultyRepository>();

services.AddTransient<IEBook, EBookRespository>();

services.AddTransient<IDesignation, DesignationRepository>();

services.AddTransient<IBookStatus, BookStatusRespository>();

}

// This method gets called by the runtime. Use this method to configure the HTTP

request pipeline.

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

app.UseSwagger();

app.UseSwaggerUI(c => c.SwaggerEndpoint("/swagger/v1/swagger.json",

"LibraryManagementSystem v1"));

}

app.UseHttpsRedirection();

app.UseRouting();

// with a named pocili

app.UseCors("AllowOrigin");

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllers();

});

}

}

}

ANGULAR CODES:

AUTHENTICATION:

*/\* You can add global styles to this file, and also import other style files \*/*

body{

background-color: #fdfdfd;;

}

input*.form-control*,input*.form-control:focus*{

background-color: #636b7b;

*/\* #636b7b \*/*

color: #fff;

border: 1*px* solid #636b7b;

}

input*.form-control:focus*{

box-shadow: none;

}

button[*type=*submit]*.btn*{

border-radius: 20*px*;

background-color: #6c3ae2;

color:rgb(255, 255, 255);

font-weight: 400;

}

div*.form-group* label*:not*(*.text-danger*){

color: #b4b3b3;

font-weight: 500;

}

div*.form-group.required>*label*:first-child:after*{

content:'\*';

color: rgb(105, 46, 46);

padding-left: 5*px*;

}

*#toast-container >* div {

opacity:1;

}

*.wrapper* {

display: flex;

align-items: center;

flex-direction: column;

justify-content: center;

width: 100*%*;

padding: 20*px*;

}

*.wrapped-div* {

-webkit-border-radius: 10*px* 10*px* 10*px* 10*px*;

border-radius: 10*px* 10*px* 10*px* 10*px*;

background: #161016;

*/\* 2e3137 \*/*

padding: 30*px*;

width: 90*%*;

max-width: 450*px*;

position: relative;

padding: 0*px*;

-webkit-box-shadow: 0 30*px* 60*px* 0 rgba(0,0,0,0.3);

box-shadow: 0 30*px* 60*px* 0 rgba(0,0,0,0.3);

}

form{

margin: 0*px* 16*px*;

}

div*.form-group* input*.invalid*{

border: 1*px* solid #dc3545;

}

div*.form-group* label*:first-child*{

text-transform: uppercase;

font-size: 0.9*rem*;

}

LOGIN COMPONENT.HTML:

<form *#form*='ngForm' *class*="mb-4" *autocomplete*="off" *(submit)*="*onSubmit*(*form*)">

<div *class*="form-group">

<label>Username</label>

<input *class*="form-control" *#UserName*="ngModel" *name*="UserName"

*[(ngModel)]*="*formModel*.*UserName*" *required*>

</div>

<div *class*="form-group" *style*="margin-top: 20px;">

<label>Password</label>

<input *type*="password" *class*="form-control" *#Password*="ngModel" *name*="Password"

*[(ngModel)]*="*formModel*.*Password*" *required*>

</div>

<div *class*="form-row">

<div *class*="form-group col-md-4 offset-md-4 mt-4">

<button *type*="submit" *class*="btn btn-lg btn-block"

*[disabled]*="*form*.*invalid*">Login</button>

</div>

</div>

</form>

LOGIN COMPONENT.TS:

*import* { *ToastrService* } *from* 'ngx-toastr';

*import* { *UserService* } *from* './../../shared/user.service';

*import* { *Component*, *OnInit* } *from* '@angular/core';

*import* { *NgForm* } *from* '@angular/forms';

*import* { *Router* } *from* '@angular/router';

*@Component*({

selector: 'app-login',

templateUrl: './login.component.html',

styleUrls: ['./login.component.css']

})

*export class* LoginComponent *implements* OnInit {

*formModel =* {

UserName: '',

Password: ''

}

*constructor*(*private* service*:* UserService, *private* router*:* Router, *private* toastr*:*

ToastrService) { }

*ngOnInit*() {

*if* (*localStorage*.*getItem*('token') *!=* null)

*this*.*router*.*navigateByUrl*('/home');

}

*onSubmit*(form*:* NgForm) {

*this*.*service*.*login*(*form*.*value*).*subscribe*(

(res*:* any) *=>* {

*localStorage*.*setItem*('token', *res*.*token*);

*this*.*router*.*navigateByUrl*('/home');

},

err *=>* {

*if* (*err*.*status ==* 400)

*this*.*toastr*.*error*('Incorrect username or password.', 'Authentication

failed.');

*else*

*console*.*log*(*err*);

}

);

}

}

AUTHENTICATION COMPONENT:

*/\* Tabs \*/*

*/\* You can add global styles to this file, and also import other style files \*/*

body{

background-color: #fdfdfd;;

}

input*.form-control*,input*.form-control:focus*{

background-color: #636b7b;

*/\* #636b7b \*/*

color: #fff;

border: 1*px* solid #636b7b;

}

input*.form-control:focus*{

box-shadow: none;

}

button[*type=*submit]*.btn*{

border-radius: 20*px*;

background-color: #6c3ae2;

color:rgb(255, 255, 255);

font-weight: 400;

}

div*.form-group* label*:not*(*.text-danger*){

color: #b4b3b3;

font-weight: 500;

}

div*.form-group.required>*label*:first-child:after*{

content:'\*';

color: rgb(105, 46, 46);

padding-left: 5*px*;

}

*#toast-container >* div {

opacity:1;

}

*.wrapper* {

display: flex;

align-items: center;

flex-direction: column;

justify-content: center;

width: 100*%*;

padding: 20*px*;

}

*.wrapped-div* {

-webkit-border-radius: 10*px* 10*px* 10*px* 10*px*;

border-radius: 10*px* 10*px* 10*px* 10*px*;

background: #161016;

*/\* 2e3137 \*/*

padding: 30*px*;

width: 90*%*;

max-width: 450*px*;

position: relative;

padding: 0*px*;

-webkit-box-shadow: 0 30*px* 60*px* 0 rgba(0,0,0,0.3);

box-shadow: 0 30*px* 60*px* 0 rgba(0,0,0,0.3);

}

form{

margin: 0*px* 16*px*;

}

div*.form-group* input*.invalid*{

border: 1*px* solid #dc3545;

}

div*.form-group* label*:first-child*{

text-transform: uppercase;

font-size: 0.9*rem*;

}

*.tab-header*{

text-align: center;

}

*.tab-header* h2*.active* {

color: #fff;

border-bottom: 4*px* solid #fff;

}

*.tab-header* h2 {

text-align: center;

font-size: 18*px*;

font-weight: 400;

display:inline-block;

padding: 30*px* 40*px* 10*px* 40*px*;

cursor: pointer;

color: #545d6a;

border-bottom: 2*px* solid #545d6a;

}

*.tab-header* h2*:focus* {

outline: none;

}

html, body { height: 100*%*; }

body { margin: 0; font-family: Roboto, "Helvetica Neue", sans-serif; }

AUTHENTICATION COMPONENT.HTML:

<div *class*="wrapper">

<div *class*="wrapped-div">

<div *class*="tab-header">

<h2 *routerLink*='/authentication/login' *routerLinkActive*='active'>Sign In</h2>

<h2 *routerLink*='/authentication/registration' *routerLinkActive*='active'>Sign

Up</h2>

</div>

<div *class*="row">

<div *class*="col-md-10 offset-md-1">

<router-outlet> </router-outlet>

</div>

</div>

</div>

</div>

AUTHENTICATION COMPONENT.TS:

*import* { *Component*, *OnInit* } *from* '@angular/core';

*@Component*({

selector: 'app-authentication',

templateUrl: './authentication.component.html',

styleUrls: ['./authentication.component.css']

})

*export class* AuthenticationComponent *implements* OnInit {

*constructor*() { }

*ngOnInit*()*:* void {

}

}

DASHBOARD:

DASHBOARD.CSS:

*#main-navbar* {

margin-bottom: 20*px*;

}

*#library-logo* {

margin-left: 10*px*;

margin-top: -10*px*;

margin-bottom: -10*px*;

width: 55*px*;

height: auto;

}

*#logo-title* {

color: white;

position: fixed;

margin-left: 70*px*;

}

*#setting-logo* {

margin-left: 10*px*;

margin-top: -5*px*;

margin-bottom: -5*px*;

width: 30*px*;

height: auto;

background-color: rgb(255, 253, 253);

border-radius: 20*px*;

margin-right: 5*px*;

}

*#dashboard-button* {

margin-left: 10*px*;

}

*#books-button* {

margin-left: 10*px*;

}

*/\* CONTAINER \*/*

*#dashboard-title* {

text-decoration: underline;

font-weight: bold;

}

*#dashboard-student-button* {

display: block;

margin-bottom: 10*px*;

margin-left: 5*px*;

}

*#dashboard-teacher-button* {

margin-bottom: 10*px*;

margin-left: 5*px*;

}

DASHBOARD.HTML:

<nav *class*="navbar sticky-top navbar navbar-dark bg-dark" *id*="main-navbar">

<img *src*="assets/img/bookshelf.png" *id*="library-logo">

<label *id*="logo-title" > LIBRARY MANAGEMENT </label>

<div *class*="navbar-right">

*<!-- HOME BUTTON -->*

<button *type*="button" *class*="btn btn-outline-primary btn-sm" *id*="home-button">

Home

</button>

*<!-- DASHBOARD BUTTON -->*

<button *type*="button" *class*="btn btn-outline-info btn-sm"

*id*="dashboard-button" *(click)*="*goToPage1*('dashboard')">

Dashboard

</button>

*<!-- SETTING ICON -->*

<a *href*="#"> <img *src*="assets/img/setting.png" *id*="setting-logo"> </a>

</div>

</nav>

<div *class*="container" *style*="width: 10rem;">

<div *class*="card border-dark" *style*="background-color: rgb(243, 236, 236);">

<div *class*="card-body">

<h3 *id*="dashboard-title"> DASHBOARD </h3>

<button *type*="button" *class*="btn btn-outline-dark"

*id*="dashboard-student-button" *routerLink*="student"

*(click)*="*goToPage2*('student/viewStudent')">

Student

</button>

<button *type*="button" *class*="btn btn-outline-dark"

*id*="dashboard-teacher-button" *routerLink*="teacher"

*(click)*="*goToPage3*('teacher/viewTeacher')">

Teacher

</button>

</div>

</div>

</div>

<router-outlet> </router-outlet>

DASHBOARD.TS:

*import* { *Component*, *OnInit* } *from* '@angular/core';

*import* { *Router* } *from* '@angular/router';

*@Component*({

selector: 'app-dashboard',

templateUrl: './dashboard.component.html',

styleUrls: ['./dashboard.component.css']

})

*export class* DashboardComponent *implements* OnInit {

*constructor*(*private* router*:* Router) { }

*ngOnInit*()*:* void {

}

*goToPage1*(pageName*:* string)*:*void {

*this*.*router*.*navigate*([`${*pageName*}`])

}

*goToPage2*(pageName*:* string)*:*void {

*this*.*router*.*navigate*([`${*pageName*}`])

}

*goToPage3*(pageName*:* string)*:*void {

*this*.*router*.*navigate*([`${*pageName*}`])

}

}

ROUTING PAGE:

*import* { *NgModule* } *from* '@angular/core';

*import* { *RouterModule*, *Routes* } *from* '@angular/router';

*import* { *AuthenticationComponent* } *from*

'./authentication/authentication.component';

*import* { *StudentComponent* } *from* './student/student.component';

*import* { *LoginComponent* } *from* './authentication/login/login.component';

*import* { *RegistrationComponent* } *from*

'./authentication/registration/registration.component';

*import* { *AddStudentDetailsComponent* } *from* './student/add-student-details/addstudent-

details.component';

*import* { *AddTeacherDetailsComponent* } *from* './teacher/add-teacher-details/addteacher-

details.component';

*import* { *TeacherComponent* } *from* './teacher/teacher.component';

*import* { *ViewStudentDetailsComponent* } *from* './student/view-student-details/viewstudent-

details.component';

*import* { *ViewTeacherDetailsComponent* } *from* './teacher/view-teacher-details/viewteacher-

details.component';

*import* { *DashboardComponent* } *from* './dashboard/dashboard.component';

*const routes: Routes = [*

*{path:* 'student'*, component: StudentComponent},*

*{path:*''*,redirectTo:*'/authentication/login'*,pathMatch:*'full'*},*

*{*

*path:* 'authentication'*, component: AuthenticationComponent,*

*children: [*

*{ path:* 'registration'*, component: RegistrationComponent },*

*{ path:* 'login'*, component: LoginComponent }*

*]*

*},*

*{*

*path:* 'student'*, component: StudentComponent,*

*children: [*

*{path:* 'addStudent'*, component: AddStudentDetailsComponent},*

*{path:* 'viewStudent'*, component: ViewStudentDetailsComponent},*

*{path:* 'dashboard'*, component: DashboardComponent},*

*]*

*},*

*{*

*path:* 'teacher'*, component: TeacherComponent,*

*children: [*

*{path:* 'addTeacher'*, component: AddTeacherDetailsComponent},*

*{path:* 'viewTeacher'*, component: ViewTeacherDetailsComponent},*

*]*

*},*

*{*

*path:* 'dashboard'*, component: DashboardComponent,*

*},*

*]*;

*@NgModule*({

imports: [*RouterModule*.*forRoot*(*routes*)],

exports: [*RouterModule*],

})

*export class* AppRoutingModule { }

APP MODULE:

*import* { *NgModule* } *from* '@angular/core';

*import* { *BrowserModule* } *from* '@angular/platform-browser';

*import* { *AppRoutingModule* } *from* './app-routing.module';

*import* { *AppComponent* } *from* './app.component';

*import* { *ToastrModule* } *from* 'ngx-toastr';

*import* { *FormsModule* } *from* '@angular/forms';

*import* { *HttpClientModule* } *from* '@angular/common/http';

*import* { *ReactiveFormsModule* } *from* '@angular/forms';

*import* { *LoginComponent* } *from* './authentication/login/login.component';

*import* { *RegistrationComponent* } *from*

'./authentication/registration/registration.component';

*import* { *HTTP\_INTERCEPTORS* } *from* '@angular/common/http';

*import* { *UserService* } *from* './shared/user.service';

*import* { *BrowserAnimationsModule* } *from* '@angular/platform-browser/animations';

*import* {*A11yModule*} *from* '@angular/cdk/a11y';

*import* {*CdkAccordionModule*} *from* '@angular/cdk/accordion';

*import* {*ClipboardModule*} *from* '@angular/cdk/clipboard';

*import* {*DragDropModule*} *from* '@angular/cdk/drag-drop';

*import* {*PortalModule*} *from* '@angular/cdk/portal';

*import* {*ScrollingModule*} *from* '@angular/cdk/scrolling';

*import* {*CdkStepperModule*} *from* '@angular/cdk/stepper';

*import* {*CdkTableModule*} *from* '@angular/cdk/table';

*import* {*CdkTreeModule*} *from* '@angular/cdk/tree';

*import* {*MatAutocompleteModule*} *from* '@angular/material/autocomplete';

*import* {*MatBadgeModule*} *from* '@angular/material/badge';

*import* {*MatBottomSheetModule*} *from* '@angular/material/bottom-sheet';

*import* {*MatButtonModule*} *from* '@angular/material/button';

*import* {*MatButtonToggleModule*} *from* '@angular/material/button-toggle';

*import* {*MatCardModule*} *from* '@angular/material/card';

*import* {*MatCheckboxModule*} *from* '@angular/material/checkbox';

*import* {*MatChipsModule*} *from* '@angular/material/chips';

*import* {*MatStepperModule*} *from* '@angular/material/stepper';

*import* {*MatDatepickerModule*} *from* '@angular/material/datepicker';

*import* {*MatDialogModule*} *from* '@angular/material/dialog';

*import* {*MatDividerModule*} *from* '@angular/material/divider';

*import* {*MatExpansionModule*} *from* '@angular/material/expansion';

*import* {*MatGridListModule*} *from* '@angular/material/grid-list';

*import* {*MatIconModule*} *from* '@angular/material/icon';

*import* {*MatInputModule*} *from* '@angular/material/input';

*import* {*MatListModule*} *from* '@angular/material/list';

*import* {*MatMenuModule*} *from* '@angular/material/menu';

*import* {*MatNativeDateModule*, *MatRippleModule*} *from* '@angular/material/core';

*import* {*MatPaginatorModule*} *from* '@angular/material/paginator';

*import* {*MatProgressBarModule*} *from* '@angular/material/progress-bar';

*import* {*MatProgressSpinnerModule*} *from* '@angular/material/progress-spinner';

*import* {*MatRadioModule*} *from* '@angular/material/radio';

*import* {*MatSelectModule*} *from* '@angular/material/select';

*import* {*MatSidenavModule*} *from* '@angular/material/sidenav';

*import* {*MatSliderModule*} *from* '@angular/material/slider';

*import* {*MatSlideToggleModule*} *from* '@angular/material/slide-toggle';

*import* {*MatSnackBarModule*} *from* '@angular/material/snack-bar';

*import* {*MatSortModule*} *from* '@angular/material/sort';

*import* {*MatTableModule*} *from* '@angular/material/table';

*import* {*MatTabsModule*} *from* '@angular/material/tabs';

*import* {*MatToolbarModule*} *from* '@angular/material/toolbar';

*import* {*MatTooltipModule*} *from* '@angular/material/tooltip';

*import* {*MatTreeModule*} *from* '@angular/material/tree';

*import* {*OverlayModule*} *from* '@angular/cdk/overlay';

*import* { *AddStudentDetailsComponent* } *from* './student/add-student-details/addstudent-

details.component';

*import* { *AuthenticationComponent* } *from*

'./authentication/authentication.component';

*import* { *StudentComponent* } *from* './student/student.component';

*import* { *AddTeacherDetailsComponent* } *from* './teacher/add-teacher-details/addteacher-

details.component';

*import* { *TeacherComponent* } *from* './teacher/teacher.component';

*import* { *ViewStudentDetailsComponent* } *from* './student/view-student-details/viewstudent-

details.component';

*import* { *ViewTeacherDetailsComponent* } *from* './teacher/view-teacher-details/viewteacher-

details.component';

*import* { *DashboardComponent* } *from* './dashboard/dashboard.component';

*import* { *RouterModule* } *from* '@angular/router';

*import* { *StudentService* } *from* './shared/student.service';

*import* { *TeacherService* } *from* './shared/teacher.service';

*import* { *EditStudentDetailsComponent* } *from* './student/edit-student-details/editstudent-

details.component';

*import* { *MatFormFieldModule* } *from* '@angular/material/form-field';

*import* { *EditTeacherDetailsComponent* } *from* './teacher/edit-teacher-details/editteacher-

details.component';

*@NgModule*({

declarations: [

*AppComponent*,

*LoginComponent*,

*RegistrationComponent*,

*AddStudentDetailsComponent*,

*AuthenticationComponent*,

*StudentComponent*,

*AddTeacherDetailsComponent*,

*TeacherComponent*,

*ViewStudentDetailsComponent*,

*ViewTeacherDetailsComponent*,

*DashboardComponent*,

*EditStudentDetailsComponent*,

*EditTeacherDetailsComponent*

],

imports: [

*BrowserModule*,

*AppRoutingModule*,

*ToastrModule*.*forRoot*({

progressBar: true

}),

*A11yModule*,

*FormsModule*,

*HttpClientModule*,

*ReactiveFormsModule*,

*BrowserAnimationsModule*,

*CdkAccordionModule*,

*ClipboardModule*,

*CdkStepperModule*,

*CdkTableModule*,

*CdkTreeModule*,

*DragDropModule*,

*MatAutocompleteModule*,

*MatBadgeModule*,

*MatBottomSheetModule*,

*MatButtonModule*,

*MatButtonToggleModule*,

*MatCardModule*,

*MatCheckboxModule*,

*MatChipsModule*,

*MatStepperModule*,

*MatDatepickerModule*,

*MatDialogModule*,

*MatDividerModule*,

*MatExpansionModule*,

*MatFormFieldModule*,

*MatGridListModule*,

*MatIconModule*,

*MatInputModule*,

*MatListModule*,

*MatMenuModule*,

*MatNativeDateModule*,

*MatPaginatorModule*,

*MatProgressBarModule*,

*MatProgressSpinnerModule*,

*MatRadioModule*,

*MatRippleModule*,

*MatSelectModule*,

*MatSidenavModule*,

*MatSliderModule*,

*MatSlideToggleModule*,

*MatSnackBarModule*,

*MatSortModule*,

*MatTableModule*,

*MatTabsModule*,

*MatToolbarModule*,

*MatTooltipModule*,

*MatTreeModule*,

*OverlayModule*,

*PortalModule*,

*RouterModule*,

*ScrollingModule*,

],

providers: [*UserService*,*StudentService*, *TeacherService*],

bootstrap: [*AppComponent*]

})

*export class* AppModule { }

MODEL:

STUDENT MODEL:

*export class* Student {

*constructor*(

*public* id*:* number *=* 0,

*public* StudentName*:* string *=* "",

*public* FatherName*:* string *=* "",

*public* Address*:* string *=* "",

*public* UserName*:* string *=* "",

*public* Class*:* string *=* "",

*public* RollNo*:* number *=* 0,

*public* Phone*:* string *=* "",

*public* StudentEmail*:* string *=* "",

*public* Password*:* string *=* "",

*public* ConfirmPassword*:* string *=* "",

){}

}

TEACHER MODEL:

*export class* Teacher {

*constructor*(

*public* FacultyId*:* number *=* 0,

*public* FacultyName*:* string *=* "",

*public* FacultyPhone*:* string *=* "",

*public* FacultyEmail*:* string *=* "",

*public* FacultyAddress*:* string *=* "",

*public* DesignationName*:* string *=* ""

){}

}

SHARED SERVICE:

STUDENT-SERVICE:

*import* { *HttpClient* } *from* '@angular/common/http';

*import* { *Injectable* } *from* '@angular/core';

*import* { *Observable* } *from* 'rxjs';

*@Injectable*({

providedIn: 'root'

})

*export class* StudentService {

*readonly APIUrl =* "https://localhost:44324/api";

*constructor*(*private* http*:* HttpClient) { }

*getStudentList*()*:*Observable<any[]>{

*return this*.*http*.*get*<any>(*this*.*APIUrl +* '/Student');

}

*addStudentList*(val*:*any) {

*return this*.*http*.*post*(*this*.*APIUrl +* '/Student', *val*, {responseType: 'text'});

}

*updateStudentList*(val*:*any) {

*return this*.*http*.*put*(*this*.*APIUrl +* '/Student', *val*, {responseType: 'text'});

}

*deleteStudentList*(val*:*any) {

*return this*.*http*.*delete*(*this*.*APIUrl +* '/Student/'*+ val*, {responseType: 'text'});

}

}

TEACHER-SERVICE:

*import* { *HttpClient* } *from* '@angular/common/http';

*import* { *Injectable* } *from* '@angular/core';

*import* { *Observable* } *from* 'rxjs';

*@Injectable*({

providedIn: 'root'

})

*export class* TeacherService {

*readonly APIUrl =* "https://localhost:44324/api";

*constructor*(*private* http*:* HttpClient) { }

*getTeacherList*()*:*Observable<any[]>{

*return this*.*http*.*get*<any>(*this*.*APIUrl +* '/Faculty');

}

*addTeacher*(val*:*any) {

*return this*.*http*.*post*(*this*.*APIUrl +* '/Faculty', *val*, {responseType: 'text'});

}

*updateTeacher*(val*:*any) {

*return this*.*http*.*put*(*this*.*APIUrl +* '/Faculty', *val*);

}

*deleteTeacher*(val*:*any) {

*return this*.*http*.*delete*(*this*.*APIUrl +* '/Faculty/'*+ val*, {responseType: 'text'});

}

}

USER-SERVICE:

*import* { *Injectable* } *from* '@angular/core';

*import* { *FormBuilder*, *Validators*, *FormGroup* } *from* '@angular/forms';

*import* { *HttpClient*, *HttpHeaders* } *from* "@angular/common/http";

*@Injectable*({

providedIn: 'root'

})

*export class* UserService {

*constructor*(*private* fb*:* FormBuilder, *private* http*:* HttpClient) { }

*readonly BaseURI =* 'http://localhost:54277/api';

*formModel = this*.*fb*.*group*({

UserName: ['', *Validators*.*required*],

Email: ['', *Validators*.*email*],

FullName: [''],

Passwords: *this*.*fb*.*group*({

Password: ['', [*Validators*.*required*, *Validators*.*minLength*(4)]],

ConfirmPassword: ['', *Validators*.*required*]

}, { validator: *this*.*comparePasswords* })

});

*comparePasswords*(fb*:* FormGroup) {

*// let confirmPswrdCtrl = fb.get('ConfirmPassword');*

*// //passwordMismatch*

*// //confirmPswrdCtrl.errors={passwordMismatch:true}*

*// if (confirmPswrdCtrl.errors == null || 'passwordMismatch' in*

*confirmPswrdCtrl.errors) {*

*// if (fb.get('Password').value != confirmPswrdCtrl.value)*

*// confirmPswrdCtrl.setErrors({ passwordMismatch: true });*

*// else*

*// confirmPswrdCtrl.setErrors(null);*

*// }*

}

*register*() {

*var body = {*

*UserName: this.formModel.value.UserName,*

*Email: this.formModel.value.Email,*

*FullName: this.formModel.value.FullName,*

*Password: this.formModel.value.Passwords.Password*

*}*;

*return this*.*http*.*post*(*this*.*BaseURI +* '/ApplicationUser/Register', *body*);

}

*login*(formData*:* any) {

*return this*.*http*.*post*(*this*.*BaseURI +* '/ApplicationUser/Login', *formData*);

}

*getUserProfile*() {

*return this*.*http*.*get*(*this*.*BaseURI +* '/UserProfile');

}

}

STUDENT COMPONENT:

ADD-STUDENT-DETAILS.CSS:

*#container* {

margin-top: 20*px*;

}

*#studentNameLabel* {

color: black;

}

*#studentName* {

width: 45*%*;

background-color: rgb(240, 239, 239);

color: black;

}

*#emailIDLabel* {

color: black;

}

*#studentEmail* {

background-color: rgb(240, 239, 239);

color: black;

}

*#phoneNumberLabel* {

color: black;

}

*#phone* {

background-color: rgb(240, 239, 239);

color: black;

width: 52.5*%*;

}

*#passwordLabel* {

color: black;

}

*#password* {

background-color: rgb(240, 239, 239);

color: black;

width: 45*%*;

}

*#confirmPasswordDiv* {

margin: -81*px* 0*px* 0*px* 300*px*;

}

*#confirmPasswordLabel* {

color: black;

}

*#confirmPassword* {

background-color: rgb(240, 239, 239);

color: black;

width: 100*%*;

}

*#addressLabel* {

color: black;

}

*#address* {

width: 100*%*;

background-color: rgb(240, 239, 239);

color: black;

}

*#userNameLabel* {

color: black;

}

*#userName* {

width: 55*%*;

background-color: rgb(240, 239, 239);

color: black;

}

*#userNameDiv* {

margin-top: 10*px*;

}

*#classLabel* {

color: black;

}

*#class* {

width: 40*%*;

background-color: rgb(240, 239, 239);

color: black;

}

*#classDiv* {

margin-top: 10*px*;

}

*#rollNoLabel* {

color: black;

}

*#rollNo* {

width: 65*%*;

background-color: rgb(240, 239, 239);

color: black;

}

*#rollNoDiv* {

margin: -62*px* 0*px* 0*px* 275*px*;

}

*#fatherNameLabel* {

color: black;

}

*#fatherName* {

width: 95*%*;

background-color: rgb(240, 239, 239);

color: black;

}

*#fatherNameDiv* {

margin-top: 10*px*;

}

ADD-STUDENT-DETAILS.HTML:

<div *class*="container" *style*="width: 40rem;">

<div *class*="card">

<div *class*="card-body">

<form *#studentForm*="ngForm" *(ngSubmit)*="*onSubmit*()" *novalidate id*="form-start">

*<!-- STUDENT NAME -->*

<div *class*="form-group" *id*="studentNameDiv">

<label *for*="studentName" *id*="studentNameLabel">

<h4 *style*="margin-bottom: -10px;">

Student Name

<span *style*="color: red;"> \* </span>

</h4>

</label>

<input

*required*

*[(ngModel)]* = "studentModel.StudentName"

*name*="studentName"

*type*="text"

*id*="studentName"

*class*="form-control"

*#studentName*="ngModel"

*minlength*="5"

*pattern*="[A-za-z\s]+"

*[class.is-invalid]*="*studentName*.*touched && !studentName*.*valid*">

<div *\*ngIf*="(*studentName*.*errors && studentName*.*touched*)

*||*

(*studentName*.*errors && studentName*.*valid*)">

<small *class*="text-danger" *\*ngIf*="*studentName*?.*errors*?.['required']">

Student name is required </small>

<small *class*=text-danger *\*ngIf*="*studentName*?.*errors*?.['pattern']"> Only

alphabets are allowed </small>

<small *class*=text-danger *\*ngIf*="*studentName*?.*errors*?.['minlength']"> Minimum

length should be 5 </small>

</div>

</div>

*<!-- FATHER"S NAME -->*

<div *class*="form-group" *id*="fatherNameDiv">

<label *for*="fatherName" *id*="fatherNameLabel">

<h4 *style*="margin-bottom: -10px;">

Father Name

<span *style*="color: red;"> \* </span>

</h4>

</label>

<input

*required*

*[(ngModel)]* = "studentModel.FatherName"

*name*="fatherName"

*type*="text"

*id*="fatherName"

*class*="form-control"

*#fatherName*="ngModel"

*minlength*="5"

*pattern*="[A-za-z\s]+"

*[class.is-invalid]*="*fatherName*.*touched && !fatherName*.*valid*">

</div>

*<!-- USERNAME -->*

<div *class*="form-group" *id*="userNameDiv">

<label *for*="userName" *id*="studentNameLabel">

<h4 *style*="margin-bottom: -10px;"> User Name

<span *style*="color: red;"> \* </span>

</h4>

</label>

<input

*required*

*[(ngModel)]* = "studentModel.UserName"

*name*="userName"

*type*="text"

*id*="userName"

*class*="form-control"

*#userName*="ngModel"

*minlength*="5"

*pattern*="[A-za-z\s]+"

*[class.is-invalid]*="*userName*.*touched && !userName*.*valid*">

</div>

*<!-- CLASS -->*

<div *class*="form-group" *id*="classDiv">

<label *for*="class" *id*="classLabel">

<h4 *style*="margin-bottom: -10px;">

Class

</h4>

</label>

<input

*required*

*[(ngModel)]* = "studentModel.Class"

*type*="text"

*name*="class"

*id*="class"

*value*="class"

*#class*="ngModel"

*class*="form-control">

</div>

*<!-- ROLL NO -->*

<div *class*="form-group" *id*="rollNoDiv">

<label *for*="rollNo" *id*="rollNOLabel">

<h4 *style*="margin-bottom: -10px;">

Roll No.

</h4>

</label>

<input

*required*

*[(ngModel)]* = "studentModel.RollNo"

*type*="text"

*name*="rollNo"

*id*="rollNo"

*value*="rollNo"

*#rollNo*="ngModel"

*class*="form-control"

*pattern*="[0-9]">

</div>

*<!-- EMAIL-ID -->*

<div *class*="form-group" *id*="emailIDDiv">

<label *for*="studentEmail" *id*="emailIDLabel">

<h4 *style*="margin-bottom: -10px; margin-top: 20px;">

Email - ID.

<span *style*="color: red;"> \* </span>

</h4>

</label>

<input

*[(ngModel)]* = "studentModel.StudentEmail"

*type*="text"

*name*="studentEmail"

*id*="studentEmail"

*value*="studentEmail"

*#studentEmail*="ngModel"

*class*="form-control"

*[class.is-invalid]*="*studentEmail*.*touched && !studentEmail*.*valid*">

&nbsp;

</div>

*<!-- PASSWORD & CONFIRM PASSWORD -->*

<div *id*="passwordDiv">

<label *for*="password" *id*="passwordLabel">

<h4 *style*="margin-bottom: -10px;">

Password

<span *style*="color: red;"> \* </span>

</h4>

</label>

<input

*[(ngModel)]* = "studentModel.Password"

*type*="password"

*name*="password"

*id*="password"

*value*="password"

*#password*="ngModel"

*class*="form-control"

*[class.is-invalid]*="*password*.*touched && !password*.*valid*">

&nbsp;

</div>

<div *id*="confirmPasswordDiv">

<label *for*="confirmPassword" *id*="confirmPasswordLabel">

<h4 *style*="margin-bottom: -10px;"> Confirm Password </h4> </label>

<input

*[(ngModel)]* = "studentModel.ConfirmPassword"

*type*="password"

*name*="confirmPassword"

*id*="confirmPassword"

*value*="confirmPassword"

*#confirmPassword*="ngModel"

*class*="form-control">

&nbsp;

</div>

*<!-- PHONE NUMBER -->*

<div *class*="form-group" *id*="phoneNumberDiv">

<label *for*="phone" *id*="phoneNumberLabel">

<h4 *style*="margin-bottom: -10px;">

Phone Number

<span *style*="color: red;"> \* </span>

</h4>

</label>

<input

*required*

*[(ngModel)]* = "studentModel.Phone"

*type*="text"

*name*="phone"

*id*="phone"

*value*="phone"

*#phone*="ngModel"

*class*="form-control"

*[class.is-invalid]*="*phone*.*touched && !phone*.*valid*">

*<!-- pattern="[0-9]{10}"*

*<div \*ngIf="phone.errors && (phone.touched || phone.valid)">*

*<small class="text-danger" \*ngIf="phone?.errors?.['required']"> Phone*

*number is required </small>*

*<small class=text-danger \*ngIf="phone?.errors?.['pattern']"> Enter 10*

*digits </small>*

*</div> -->*

&nbsp;

</div>

*<!-- ADDRESS -->*

<div *class*="form-group">

<div *id*="addressDiv">

<label *for*="address" *id*="addressLabel"> <h4 *style*="margin-bottom: -10px;">

Address </h4> </label>

<input

*[(ngModel)]* = "studentModel.Address"

*type*="text"

*name*="address"

*id*="address"

*value*="address"

*#streetAddress*="ngModel"

*class*="form-control">

&nbsp;

</div>

</div>

<div>

<button *class*="btn btn-primary" *type*="submit">

Add Student

</button>

</div>

</form>

</div>

</div>

</div>

*<!-- [disabled]="studentForm.invalid" -->*

*<!-- <div class="form-group" id="idDiv">*

*<label for="id" id="idLabel"> Id </label>*

*<input*

*required*

*[(ngModel)] = "userModel.id"*

*name="id"*

*type="number"*

*min="0"*

*max="100"*

*id="id"*

*class="form-control"*

*#id="ngModel"*

*[class.is-invalid]="id.touched && !id.valid">*

*</div>*

*<div class="form-group" id="costOfVaccineDiv">*

*<label for="costOfVaccine" id="costOfVaccineLabel"> Cost Of Vaccine </label>*

*<input*

*required*

*[(ngModel)] = "userModel.CostOfVaccine"*

*name="costOfVaccine"*

*type="number"*

*min="100"*

*max="10000"*

*id="costOfVaccine"*

*class="form-control"*

*#costOfVaccine="ngModel"*

*[class.is-invalid]="costOfVaccine.touched && !costOfVaccine.valid">*

*<div \*ngIf="costOfVaccine.touched && !costOfVaccine.valid">*

*<small class="text-danger" \*ngIf="costOfVaccine?.errors?.['required']"> Cost*

*of vaccine is required. </small>*

*<small class="text-danger" \*ngIf="costOfVaccine?.errors?.['min']"> Minimum*

*amount should be 100 </small>*

*</div>*

*</div>*

*<pre> </pre>*

*<pre> </pre> -->*

*<!--*

*<div class="form-group" id="gender">*

*<label for="gender" id="genderLabel"> Gender </label>*

*<div class="form-control">*

*<input*

*[(ngModel)] = "userModel.Gender"*

*type="radio"*

*name="radio-button"*

*id="male"*

*value="male"*

*#male="ngModel">*

*&nbsp;*

*<label for="male" class="radio-inline"> Male </label>*

*&nbsp;*

*<input*

*[(ngModel)] = "userModel.Gender"*

*type="radio"*

*name="radio-button"*

*id="female"*

*value="female"*

*#female="ngModel">*

*&nbsp;*

*<label for="female" class="radio-inline"> Female </label>*

*</div>*

*</div>*

*<div class="form-group" id="typeDiv">*

*<label for="type" id="typeLabel"> Type </label>*

*<div class="form-control">*

*<select (blur)="validateTopic(type.value)"*

*(change)="validateTopic(type.value)"*

*[(ngModel)] = "userModel.ChooseType" name="type" #type="ngModel"*

*[class.is-invalid]="topicHasError && type.touched">*

*<option value="default"> -- </option>*

*<option \*ngFor="let topic of topics"> {{ topic }} </option>*

*</select>*

*<small class="text-danger" [class.d-none]="!topicHasError || type.untouched">*

*Please choose a type*

*</small>*

*</div>*

*</div>*

*<pre> </pre>*

*<div id="pincodeDiv">*

*<label for="state" id="pincode"> <h4 style="margin-bottom: -10px;"> Pincode*

*</h4> </label>*

*<input*

*required*

*[(ngModel)] = "userModel.Pincode"*

*type="number"*

*name="pincode"*

*id="pincode"*

*value="pincode"*

*#pincode="ngModel"*

*class="form-control"*

*pattern="[0-9]{6}"*

*[class.is-invalid] = "pincode.touched && !pincode.valid">*

*<div \*ngIf = "pincode.errors && (pincode.touched || pincode.valid)">*

*<small class="text-danger" \*ngIf="pincode?.errors?.['required']"> Pincode*

*is required </small>*

*<small class=text-danger \*ngIf="pincode?.errors?.['pattern']"> Enter six*

*digits </small>*

*</div>*

*</div>*

*<pre> </pre>*

*<div id="cityDiv">*

*<label for="city" id="cityLabel"> <h4 style="margin-bottom: -10px;"> City*

*</h4> </label>*

*<input*

*[(ngModel)] = "userModel.City"*

*type="text"*

*name="city"*

*id="city"*

*value="city"*

*#city="ngModel"*

*class="form-control">*

*&nbsp;*

*</div>*

*<div id="stateDiv">*

*<label for="state" id="stateLabel"> <h4 style="margin-bottom: -10px;">*

*State </h4> </label>*

*<input*

*[(ngModel)] = "userModel.State"*

*type="text"*

*name="state"*

*id="state"*

*value="state"*

*#state="ngModel"*

*class="form-control">*

*&nbsp;*

*</div>*

*<div id="countryDiv">*

*<label for="country" id="countryLabel"> <h4 style="margin-bottom: -10px;">*

*Country </h4> </label>*

*<input*

*[(ngModel)] = "userModel.Country"*

*type="text"*

*name="country"*

*id="country"*

*value="country"*

*#country="ngModel"*

*class="form-control">*

*&nbsp;*

*</div>*

*</div>*

*-->*

ADD-STUDENT-DETAILS.TS:

*import* { *Component*, *OnInit*, *ViewChild* } *from* '@angular/core';

*import* { *FormGroupDirective*, *NgForm* } *from* '@angular/forms';

*import* { *ToastrService* } *from* 'ngx-toastr';

*import* { *Student* } *from* 'src/app/model/Student';

*import* { *StudentService* } *from* 'src/app/shared/student.service';

*@Component*({

selector: 'app-add-student-details',

templateUrl: './add-student-details.component.html',

styleUrls: ['./add-student-details.component.css']

})

*export class* AddStudentDetailsComponent *implements* OnInit {

*constructor*(*private* service*:* StudentService, *private* toastr*:* ToastrService) { }

*ngOnInit*()*:* void {

}

*formDirective!:* FormGroupDirective;

*@ViewChild*('studentForm') *studentForm!:* NgForm;

*// studentModel = new Student(0,'','','','','',0,0,'','','')*

*studentModel = new Student*(0,'Johnson John','Bretto','San fransisco',

'John','1',123,'9999999999','johnsonBretto@gmail.com','johnson','johnson')

*onSubmit*(){

*this*.*service*.*addStudentList*(*this*.*studentModel*).*subscribe*(data*=>* {

*this*.*toastr*.*success*('Added Successfully','Success');

*this*.*studentForm*.*reset*();

})

}

}

EDIT-STUDENT-DETAILS.CSS;

<div *class* = "form-group row">

<label *class*="col-sm-2 col-form-label"> <h2> Student Id </h2> </label>

<div *class*="col-sm-10">

<input *type*="text" *class*="form-control" *[(ngModel)]*="*studentId*"

*placeholder* = "Student Id" *disabled*>

</div>

<label *class*="col-sm-2 col-form-label"> <h2> Student Name </h2> </label>

<div *class*="col-sm-10">

<input *type*="text" *class*="form-control" *[(ngModel)]*="*studentName*"

*placeholder* = "Enter Student Name">

</div>

<label *class*="col-sm-2 col-form-label"> <h2> Father Name </h2> </label>

<div *class*="col-sm-10">

<input *type*="text" *class*="form-control" *[(ngModel)]*="*fatherName*"

*placeholder* = "Enter Father Name">

</div>

<label *class*="col-sm-2 col-form-label"> <h2> Student Email </h2> </label>

<div *class*="col-sm-10">

<input *type*="text" *class*="form-control" *[(ngModel)]*="*studentEmail*"

*placeholder* = "Enter Student Email">

</div>

<label *class*="col-sm-2 col-form-label"> <h2> Phone Number </h2> </label>

<div *class*="col-sm-10">

<input *type*="text" *class*="form-control" *[(ngModel)]*="*phone*"

*placeholder* = "Enter Phone Number">

</div>

<label *class*="col-sm-2 col-form-label"> <h2> Password </h2> </label>

<div *class*="col-sm-10">

<input *type*="password" *class*="form-control" *[(ngModel)]*="*password*"

*placeholder* = "Enter Password" *disabled*>

</div>

<label *class*="col-sm-2 col-form-label"> <h2> Address </h2> </label>

<div *class*="col-sm-10">

<input *type*="text" *class*="form-control" *[(ngModel)]*="*address*"

*placeholder* = "Enter Address">

</div>

<label *class*="col-sm-2 col-form-label"> <h2> Class </h2> </label>

<div *class*="col-sm-10">

<input *type*="text" *class*="form-control" *[(ngModel)]*="*class*"

*placeholder* = "Enter Class">

</div>

<label *class*="col-sm-2 col-form-label"> <h2> Roll Number </h2> </label>

<div *class*="col-sm-10">

<input *type*="number" *class*="form-control" *[(ngModel)]*="*rollNo*"

*placeholder* = "Enter RollNo.">

</div>

</div>

<button *(click)* = "updateStudent()" *class*="btn btn-primary">

Update

</button>

EDIT-STUDENT-DETAILS.TS;

*import* { *Component*, *Input*, *OnInit* } *from* '@angular/core';

*import* { *ToastrService* } *from* 'ngx-toastr';

*import* { *StudentService* } *from* 'src/app/shared/student.service';

*@Component*({

selector: 'app-edit-student-details',

templateUrl: './edit-student-details.component.html',

styleUrls: ['./edit-student-details.component.css']

})

*export class* EditStudentDetailsComponent *implements* OnInit {

*constructor*(*private* service*:* StudentService, *private* toastr*:* ToastrService) { }

*@Input*() *student:* any;

*studentId!:* number;

*studentName!:* string;

*fatherName!:* string;

*studentEmail!:* string;

*phone!:* string;

*address!:* string;

*class!:* string;

*rollNo!:* number;

*password!:* string;

*StudentList:*any *=* [];

*ngOnInit*()*:* void {

*this*.*studentId = this*.*student*.*studentId*;

*this*.*studentName = this*.*student*.*studentName*;

*this*.*fatherName = this*.*student*.*fatherName*;

*this*.*password = this*.*student*.*password*;

*this*.*studentEmail = this*.*student*.*studentEmail*;

*this*.*phone = this*.*student*.*phone*;

*this*.*address = this*.*student*.*address*;

*this*.*class = this*.*student*.*class*;

*this*.*rollNo = this*.*student*.*rollNo*;

}

*refreshStudentList*() {

*this*.*service*.*getStudentList*().*subscribe*(data *=>* {

*this*.*StudentList = data*;

});

}

*updateStudent*() {

*var val = { studentId:this.studentId,*

*studentName:this.studentName,*

*fatherName:this.fatherName,*

*studentEmail:this.studentEmail,*

*phone:this.phone,*

*password:this.password,*

*address:this.address,*

*class:this.class,*

*rollNo:this.rollNo,}*

*this*.*service*.*updateStudentList*(*val*).*subscribe*

(res*=>*{

*this*.*toastr*.*success*('Updated Successfully','Success');

})

}

}

VIEW-STUDENT-DETAILS.HTML:

<div *class*="modal fade" *id*="exampleModal" *tabindex*="-1" *role*="dialog" *arialabelledby*="

exampleModalLabel" *aria-hidden*="true">

<div *class*="modal-dialog modal-dialog-cetnered modal-xl" *role*="document">

<div *class*="modal-content">

<div *class*="modal-header">

<h2 *class*="modal-title" *id*="exampleModalLabel">{{*ModalTitle*}}</h2>

<button *type*="button" *aria-label*="close" *data-bs-dismiss*="modal"

*class*="close" *(click)*= "closeClick()">

<span *aria-hidden*="true">&times;</span>

</button>

</div>

<div *class*="modal-body">

<app-edit-student-details *[student]*="*student*" *\*ngIf*="*ActivateEditStudentComp*">

</app-edit-student-details>

</div>

</div>

</div>

</div>

<table *class* = "table table-striped">

<thead>

<tr>

<th *style*="width: 2%;"> REG NO. </th>

<th *style*="width: 10%;"> STUDENT NAME </th>

<th *style*="width: 10%;"> FATHER'S NAME </th>

<th *style*="width: 20%;"> EMAIL </th>

<th *style*="width: 12.5%;"> PHONE </th>

<th> ADDRESS </th>

<th *style*="width: 6%;"> CLASS </th>

<th *style*="width: 6%;"> ROLL NO. </th>

</tr>

</thead>

<tbody>

<tr *\*ngFor*="*let dataItem of StudentDetails*">

<td>{{*dataItem*.*studentId*}}</td>

<td>{{*dataItem*.*studentName*}}</td>

<td>{{*dataItem*.*fatherName*}}</td>

<td>{{*dataItem*.*studentEmail*}}</td>

<td>{{*dataItem*.*phone*}}</td>

<td>{{*dataItem*.*address*}}</td>

<td>{{*dataItem*.*class*}}</td>

<td>{{*dataItem*.*rollNo*}}</td>

<td *style*="width: 3%;">

<button *type*="button" *(click)*= "editClick(dataItem)" *data-bs-toggle*="modal"

*data-bs-target*="#exampleModal"

*data-bs-backdrop*="static" *data-bs-keyboard*="false"

*class*="btn btn-outline-primary btn-sm">

Edit

</button>

</td>

<td *style*="width: 3%;">

<button *type*="button" *(click)*= "deleteClick(dataItem)"

*class*="btn btn-outline-primary btn-sm">

delete

</button>

</td>

</tr>

</tbody>

</table>

VIEW-STUDENT-DETAILS.TS:

*import* { *Component*, *OnInit* } *from* '@angular/core';

*import* { *StudentService* } *from* 'src/app/shared/student.service';

*import* { *ToastrService* } *from* 'ngx-toastr';

*@Component*({

selector: 'app-view-student-details',

templateUrl: './view-student-details.component.html',

styleUrls: ['./view-student-details.component.css']

})

*export class* ViewStudentDetailsComponent *implements* OnInit {

*constructor*(*private* service*:* StudentService, *private* toastr*:* ToastrService) { }

*StudentDetails:*any *=* [];

*ModalTitle! :* string;

*ActivateEditStudentComp:*boolean *=* false;

*student:* any;

*ngOnInit*()*:* void {

*this*.*refreshStudentList*();

}

*refreshStudentList*() {

*this*.*service*.*getStudentList*().*subscribe*(data *=>* {

*this*.*StudentDetails = data*;

});

}

*closeClick*() {

*this*.*ActivateEditStudentComp =* false;

*this*.*refreshStudentList*();

}

*editClick*(item*:* any) {

*this*.*student = item*;

*this*.*ModalTitle =* "Edit Student Details";

*this*.*ActivateEditStudentComp =* true;

}

*deleteClick*(item*:* any){

*if*(*confirm*("Are you sure?")) {

*this*.*service*.*deleteStudentList*(*item*.*studentId*).*subscribe*(

data *=>* {

*if*('Successfully Deleted') {

*this*.*toastr*.*success*('Deleted Successfully!', 'Success');

*this*.*refreshStudentList*();

}

}

)

}

}

}

STUDENT-COMPONENT.CSS:

*#main-navbar* {

margin-bottom: 20*px*;

}

*#library-logo* {

margin-left: 10*px*;

margin-top: -10*px*;

margin-bottom: -10*px*;

width: 55*px*;

height: auto;

}

*#logo-title* {

color: white;

position: fixed;

margin-left: 70*px*;

}

*#setting-logo* {

margin-left: 10*px*;

margin-top: -5*px*;

margin-bottom: -5*px*;

width: 30*px*;

height: auto;

background-color: rgb(255, 253, 253);

border-radius: 20*px*;

margin-right: 5*px*;

}

*#dashboard-button* {

margin-left: 10*px*;

}

*#books-button* {

margin-left: 10*px*;

}

*#addStudent-button* {

margin-left: 10*px*;

}

*#viewStudent-button* {

margin-left: 10*px*;

}

STUDENT.COMPONENT.HTML:

<nav *class*="navbar sticky-top navbar navbar-dark bg-dark" *id*="main-navbar">

<img *src*="assets/img/bookshelf.png" *id*="library-logo">

<label *id*="logo-title" > LIBRARY MANAGEMENT </label>

<div *class*="navbar-right">

*<!-- HOME BUTTON -->*

<button *type*="button" *class*="btn btn-outline-primary btn-sm" *id*="home-button">

Home

</button>

*<!-- DASHBOARD BUTTON -->*

<button *type*="button" *class*="btn btn-outline-info btn-sm"

*id*="dashboard-button" *(click)*="*goToPage*('dashboard')">

Dashboard

</button>

*<!-- ADD STUDENT BUTTON -->*

<button *type*="button" *class*="btn btn-outline-warning btn-sm"

*id*="addStudent-button" *routerLink*="addStudent">

Add Student

</button>

*<!-- VIEW STUDENT BUTTON -->*

<button *type*="button" *class*="btn btn-outline-warning btn-sm"

*id*="viewStudent-button" *routerLink*="viewStudent">

View Students

</button>

*<!-- SETTING ICON -->*

<a *href*="#"> <img *src*="assets/img/setting.png" *id*="setting-logo"> </a>

</div>

</nav>

<router-outlet> </router-outlet>

STUDENT.COMPONENT.TS:

*import* { *Component*, *OnInit* } *from* '@angular/core';

*import* { *Router* } *from* '@angular/router';

*@Component*({

selector: 'app-student',

templateUrl: './student.component.html',

styleUrls: ['./student.component.css']

})

*export class* StudentComponent *implements* OnInit {

*constructor*(*private* router*:*Router) { }

*ngOnInit*()*:* void {

}

*goToPage*(pageName*:* string)*:*void {

*this*.*router*.*navigate*([`${*pageName*}`])

}

}

**Git:**

**Renaming & Moving Files**

Pwd

cd project/starter-web

ls

git status

cd level 1/level2/level3

pwd

ls

git status

git mv level3-file.txt level3.txt

ls

git status

git commit -m “remaining level3 file”

cd..

clear

pwd

ls

mv level2-file.txt level2.txt

ls

git status

git add -A

git status

git commit

clear

ls

git mv level2.txt level2.txt

ls

git status

git mv 2.txt level2.txt

ls

git status

clear

ls

git mv level2.txt level3

ls

git status

cd..

git status

git commit

ls

cd/level3

ls

mv level2.txt ..

ls

cd..

pwd

ls

git status

git add -A

git status

git commit

clear

cd..

pwd

ls

git status

git add level1.txt

git add -u

git status

git commit

**Deleting Files**

pwd

cd projects/starter -web

ls

git status

clear

mate doomed.txt

ls

git status

git rm doomed.txt

rm doomed.txt

git status

clear

git ls -files

git rm newfile.txt

ls

git status

git commit -m”Deleting new File”

git status

clear

ls

git ls-files

git rm LibraryManagementSystem.txt

ls

git status

git reset HEAD LibraryManagementSystem.txt

ls

git status

git checkout -- LibraryManagementSystem.txt

ls

git status

clear

ls

rm LibraryManagementSystem.txt

ls

git status

git add -A

git status

git commit

git status

clear

ls

rm -rf level1

ls

git status

git add -A

git status

git commit -m “Deleting level1 and all children”

git status

**Revert Changes**

pwd

cd project/starter -web

ls

git status

cd level1/

ls

mate level1-file.txt

git status

git add level1 -file.txt

git status

mate level1 -file.txt

clear

git status

git rest HEAD level1-file.txt

git status

mate level1 -file.txt

clear

git status

git checkout –level1-file.txt

git status

mate level1-file.txt

clear