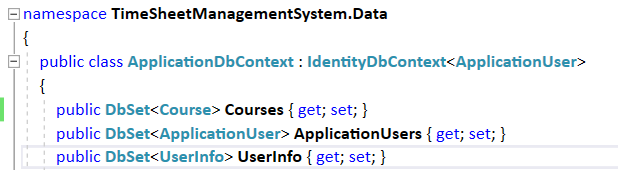
The WEBA assignment requires you to develop functionalities inside the given **TimesheetManagementSystem** web project. The **TimesheetManagementSystem** project has a **security layer**.

You will focus on reusing the files which contributes towards Manage Course functionality inside the **WEBA\_PresentationLayerDevelopment** by *integrating* them into the **TimesheetManagementSystem** web project.

|  |
| --- |
| **Listing 1** **Course.cs** class file which defines the **Course** entity |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Threading.Tasks;  namespace TimeSheetManagementSystem.Models  {  public class Course  {  public int CourseId { get; set; }  public string CourseAbbreviation { get; set; }  public string CourseName { get; set; }  public DateTime CreatedAt { get; set; }  public DateTime UpdatedAt { get; set; }  public DateTime? DeletedAt { get; set; }    public int CreatedById { get; set; }  public int UpdatedById { get; set; }  public int DeletedById { get; set; }  public UserInfo CreatedBy { get; set; }  public UserInfo UpdatedBy { get; set; }  public UserInfo DeletedBy { get; set; }  }  } |



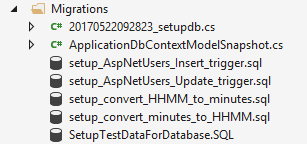
|  |
| --- |
| **Listing 2** Additional modelling code is required inside the **OnModelCreating** method |
| //----- Defining Course Entity - Start --------------  *//Make the CourseId a Primary Key*  modelBuilder.Entity<Course>()  .HasKey(input => input.CourseId)  .HasName("PrimaryKey\_CourseId");  *//Defining general properties of CourseId*  modelBuilder.Entity<Course>()  .Property(input => input.CourseId)  .HasColumnName("CourseId")  .HasColumnType("int")  .UseSqlServerIdentityColumn()  .ValueGeneratedOnAdd()  .IsRequired();  *//Defining general properties of CourseName*  modelBuilder.Entity<Course>()  .Property(input => input.CourseName)  .HasColumnName("CourseName")  .HasColumnType("VARCHAR(100)")  .IsRequired();  *//Defining general properties of CourseAbbreviation*  modelBuilder.Entity<Course>()  .Property(input => input.CourseAbbreviation)  .HasColumnName("CourseAbbreviation")  .HasColumnType("VARCHAR(10)")  .IsRequired();  modelBuilder.Entity<Course>()  .Property(input => input.CreatedAt)  .HasDefaultValueSql("GetDate()");  modelBuilder.Entity<Course>()  .Property(input => input.UpdatedAt)  .HasDefaultValueSql("GetDate()");  modelBuilder.Entity<Course>()  .Property(input => input.DeletedAt)  .IsRequired(false);  *//Enforce unique constraint on Course Abbreviation*  modelBuilder.Entity<Course>()  .HasIndex(input => input.CourseAbbreviation).IsUnique()  .HasName("Course\_CourseAbbreviation\_UniqueConstraint");  *//Setting up relationship with the UserInfo entity*  modelBuilder.Entity<Course>()  .HasOne(input => input.CreatedBy)  .WithMany()  .HasForeignKey(input => input.CreatedById)  .OnDelete(DeleteBehavior.Restrict)  .IsRequired();  modelBuilder.Entity<Course>()  .HasOne(input => input.DeletedBy)  .WithMany()  .HasForeignKey(input => input.CreatedById)  .OnDelete(DeleteBehavior.Restrict)  .IsRequired();  modelBuilder.Entity<Course>()  .HasOne(input => input.UpdatedBy)  .WithMany()  .HasForeignKey(input => input.CreatedById)  .OnDelete(DeleteBehavior.Restrict)  .IsRequired();  *//----------- Defining Course Entity - End --------------* |

After preparing the **Course** entity type inside the **ApplicationDbContext** class's **OnModelCreating** method, you have completed the Database Layer Development process for the **Course** entity type. The next step is:

i) Start the PowerShell window or the Developer Console window.

ii) Refer to the following figure, delete any existing migration class files inside the **Migrations** directory.

iii) Use the **dotnet ef migrations add setupdb** to create the migration file again.



Delete these files because you are creating a new set of migration files by using the **dotnet ef migrations add setupdb**.

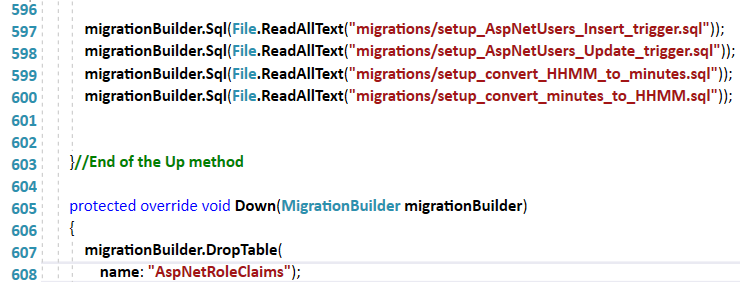
Open the **<timestamp>\_setupdb.cs** class file inside the **Migrations** directory. There are two methods. One is **Up** method and the other is **Down** method. You need to apply the following four lines of command at the *end* of the **Up** method before the **Up** method *exits*.

migrationBuilder.Sql(File.ReadAllText("migrations/setup\_AspNetUsers\_Insert\_trigger.sql"));

migrationBuilder.Sql(File.ReadAllText("migrations/setup\_AspNetUsers\_Update\_trigger.sql"));

migrationBuilder.Sql(File.ReadAllText("migrations/setup\_convert\_HHMM\_to\_minutes.sql"));

migrationBuilder.Sql(File.ReadAllText("migrations/setup\_convert\_minutes\_to\_HHMM.sql"));



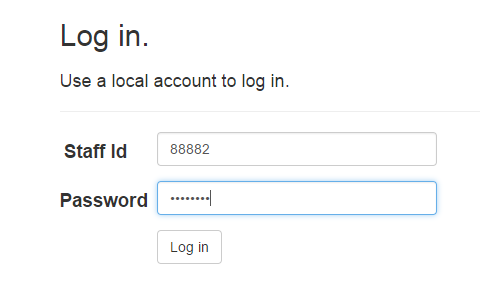
Paste additional commands before the **Up** method exits.

| **Listing** **3** **Courses** Web API controller class initial code pattern which has no Web API methods. |
| --- |
| using TimeSheetManagementSystem.Models;  using TimeSheetManagementSystem.Data;  using TimeSheetManagementSystem.Services;  using TimeSheetManagementSystem.Controllers;  using System;  using System.Collections.Generic;  using System.Linq;  using Microsoft.AspNetCore.Mvc;  using Newtonsoft.Json;  using Microsoft.EntityFrameworkCore;  using Microsoft.AspNetCore.Identity;  using Microsoft.AspNetCore.Authorization;  using Microsoft.Extensions.Logging;  using Microsoft.Extensions.Configuration;  namespace TimeSheetManagementSystem.APIs  {  [Route("api/[controller]")]  public class CoursesController : Controller  {  *//The following five properties are required for every web api controller*  *//class.*  private readonly UserManager<ApplicationUser> \_userManager;  private readonly SignInManager<ApplicationUser> \_signInManager;  private readonly IEmailSender \_emailSender;  private readonly ISmsSender \_smsSender;  private readonly ILogger \_logger;  *//Define two important properties which are required for "every"*  *//web api controller class.*  public ApplicationDbContext Database { get; }  public IConfigurationRoot Configuration { get; }  *//The following constructor code pattern is required for every Web API*  *//controller class.*  public CoursesController(UserManager<ApplicationUser> userManager,  SignInManager<ApplicationUser> signInManager,  IEmailSender emailSender,  ISmsSender smsSender,  ILoggerFactory loggerFactory, ApplicationDbContext database)  {  Database = database; *//Initialize the Database property*  \_userManager = userManager;  \_signInManager = signInManager;  \_emailSender = emailSender;  \_smsSender = smsSender;  \_logger = loggerFactory.CreateLogger<AccountController>();  }  }*//End of Web API controller class*  }*//End of namespace* |

|  |
| --- |
| **Listing 4** The necessary Web API methods which supports the client-side JavaScript logic |
| *// GET: api/Courses/GetCoursesForControls*  [HttpGet("GetCoursesForControls")]  public JsonResult GetCoursesForControls()  {  *//Create a List object, courseList which can store anonymous objects later*.  List<object> courseList = new List<object>();  var coursesQueryResult = Database.Courses  .Where(eachCourse => eachCourse.DeletedAt == null);  *//Loop through each Course entity in the coursesQueryResult's*  *//internal List of Course entities. Create an anoymous type object which*  *//has 2 properties, courseId, courseAbbreviation*  foreach (var oneCourse in coursesQueryResult)  {  courseList.Add(new  {  courseId = oneCourse.CourseId,  courseAbbreviation = oneCourse.CourseAbbreviation  });  }*//end of foreach*  return new JsonResult(courseList);  }*//end of GetCoursesForControls()*  [HttpGet]  public JsonResult Get()  {  List<object> courseList = new List<object>();  var courses = Database.Courses  .Include(input => input.CreatedBy)  .Include(input => input.UpdatedBy)  .Where(eachCourse => eachCourse.DeletedAt == null);  foreach (var oneCourse in courses)  {  courseList.Add(new  {  courseId = oneCourse.CourseId,  courseName = oneCourse.CourseName,  courseAbbreviation = oneCourse.CourseAbbreviation,  createdAt = oneCourse.CreatedAt,  createdBy = oneCourse.CreatedBy.FullName,  updatedAt = oneCourse.UpdatedAt,  updatedBy = oneCourse.UpdatedBy.FullName  });  }//end of foreach  return new JsonResult(courseList);  }//end of Get()  // GET api/Courses/5  [HttpGet("{id}")]  public JsonResult Get(int id)  {  List<object> courseList = new List<object>();  var foundOneCourse = Database.Courses  .Where(eachCourse => eachCourse.CourseId == id).Single();  //Create an anonymous type object to build a new JsonResult type object  //to send back information to the client.  var response = new  {  courseId = foundOneCourse.CourseId,  courseName = foundOneCourse.CourseName,  courseAbbreviation = foundOneCourse.CourseAbbreviation,  createdAt = foundOneCourse.CreatedAt,  updatedAt = foundOneCourse.UpdatedAt  };//end of creation of the response object  return new JsonResult(response);  }  // PUT api/Courses/5  [HttpPut("{id}")]  public IActionResult Put(int id, [FromBody]string value)  {  string customMessage = "";  int userId = GetUserIdFromUserInfo();  var courseChangeInput = JsonConvert.DeserializeObject<dynamic>(value);  //After reconstructing the object from the JSON string residing  //in the input parameter variable, value:  //To obtain the course Abbreviation information,  //use courseChangeInput.courseAbbreviation.Value  //To obtain the course name information,  //use courseChangeInput.courseName.Value  var oneCourse = Database.Courses  .Where(courseEntity => courseEntity.CourseId == id).Single();  oneCourse.CourseAbbreviation = courseChangeInput.courseAbbreviation.Value;  oneCourse.CourseName = courseChangeInput.courseName.Value;  oneCourse.UpdatedAt = DateTime.Now;  oneCourse.UpdatedById = userId;  try  {  Database.SaveChanges();  }  catch (Exception ex)  {  if (ex.InnerException.Message  .Contains("Course\_CourseAbbreviation\_UniqueConstraint") == true)  {  customMessage = "Unable to save course record due " +  "to another record having the same name as : " +  courseChangeInput.courseAbbreviation.Value;  //Create an anonymous object that has one property, Message.  //This anonymous object's Message property contains a simple string message  object httpFailRequestResultMessage = new { message = customMessage };  //Return a bad http request message to the client  return BadRequest(httpFailRequestResultMessage);  }  }//End of try .. catch block on saving data  //Construct a custom message for the client  //Create a success message anonymous object which has a  //Message member variable (property)  var successRequestResultMessage = new  {  message = "Saved course record"  };  //Create a OkObjectResult class instance, httpOkResult.  //When creating the object, provide the previous message object into it.  OkObjectResult httpOkResult =  new OkObjectResult(successRequestResultMessage);  //Send the OkObjectResult class object back to the client.  return httpOkResult;  }//End of Put() Web API method  // POST api/Courses  [HttpPost]  public IActionResult Post([FromBody]string value)  {  string customMessage = "";  int userId = GetUserIdFromUserInfo();  //Reconstruct a useful object from the input string value.  dynamic courseNewInput = JsonConvert.DeserializeObject<dynamic>(value);  Course newCourse = new Course();  try  {  //Copy out all the course data into the new Course instance,  //new.  newCourse.CourseAbbreviation = courseNewInput.courseAbbreviation.Value;  newCourse.CourseName = courseNewInput.courseName.Value;  newCourse.CreatedById = userId;  newCourse.UpdatedById = userId;  //When I add this Course instance, newCourse into the  //Courses Entity Set, it will turn into a Course entity waiting to be mapped  //as a new record inside the actual Course table.  Database.Courses.Add(newCourse);  Database.SaveChanges();//Telling the database model to save the changes  }  catch (Exception exceptionObject)  {  if (exceptionObject.InnerException.Message  .Contains("Course\_CourseAbbreviation\_UniqueConstraint") == true)  {  customMessage = "Unable to save course record due " +  "to another record having the same abbreviation : " +  courseNewInput.courseAbbreviation.Value;  //Create an anonymous type object that has one property, message.  //This anonymous object's message property contains a simple string message  object httpFailRequestResultMessage = new { message = customMessage };  //Return a bad http request message to the client  return BadRequest(httpFailRequestResultMessage);  }  }//End of Try..Catch block  //If there is no runtime error in the try catch block, the code execution  //should reach here. Sending success message back to the client.  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //Construct a custom message for the client  //Create a success message anonymous type object which has a  //message member variable (property)  var successRequestResultMessage = new  {  message = "Saved course record"  };  //Create a OkObjectResult class instance, httpOkResult.  //When creating the object, provide the previous message object into it.  OkObjectResult httpOkResult =  new OkObjectResult(successRequestResultMessage);  //Send the OkObjectResult class object back to the client.  return httpOkResult;  }//End of POST api  // DELETE api/Courses/5  [HttpDelete("{id}")]  public IActionResult Delete(int id)  {  string customMessage = "";  try  {  var foundOneCourse = Database.Courses  .Single(eachCourse => eachCourse.CourseId == id);  foundOneCourse.DeletedAt = DateTime.Now;  foundOneCourse.DeletedById = GetUserIdFromUserInfo();  //Tell the db model to commit/persist the changes to the database,  //I use the following command.  Database.SaveChanges();  }  catch (Exception ex)  {  customMessage = "Unable to delete course record.";  object httpFailRequestResultMessage = new { message = customMessage };  //Return a bad http request message to the client  return BadRequest(httpFailRequestResultMessage);  }//End of try .. catch block on manage data  //Build a custom message for the client  //Create a success message anonymous object which has a  //Message member variable (property)  var successRequestResultMessage = new  {  message = "Deleted course record"  };  //Create a OkObjectResult class instance, httpOkResult.  //When creating the object, provide the previous message object into it.  OkObjectResult httpOkResult =  new OkObjectResult(successRequestResultMessage);  //Send the OkObjectResult class object back to the client.  return httpOkResult;  }//end of Delete() Web API method with /apis/Courses/digit route  //Helper method: To obtain the numeric user info id from the UserInfo  public int GetUserIdFromUserInfo()  {  string userLoginId = \_userManager.GetUserName(User);  int userInfoId = Database.UserInfo.Single(input => input.LoginUserName == userLoginId).UserInfoId;  return userInfoId;  } |

|  |
| --- |
| **Listing 5** – **Courses** action controller class (**\Controllers\CoursesController.cs**) |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Threading.Tasks;  using Microsoft.AspNetCore.Mvc;  namespace TimeSheetManagementSystem.Controllers  {  public class CoursesController : Controller  {  // GET: /<Courses>/  public IActionResult Index()  {  return View();  }  public IActionResult ExperimentDataDrivenListBox()  {  return View();  //The web application will know it needs to process the  //ExperimentDataDrivenListBox.cshtml file.  }  public IActionResult Create()  {  return View();  }  public IActionResult Update()  {  return View();  }  }  } |

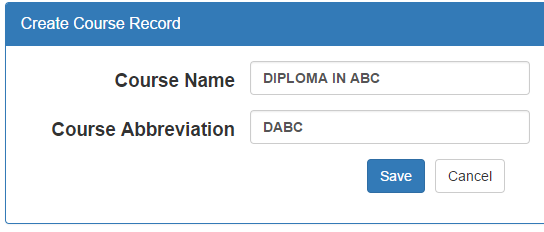
Login by using the account, JULIET. The login id should be **88882** and password should be **P@ssw0rd**.



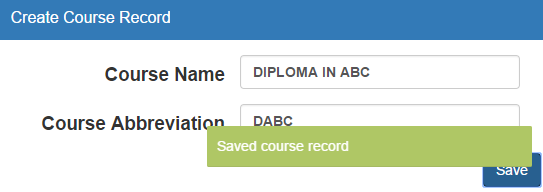
Apply the following URL pattern at the web browser to access the Create Course view interface.



Refer to the following figure. Provide **DIPLOMA** **IN** **ABC** for course name and **DABC** for course abbreviation. Then, click the **Save** button interface to proceed.



Refer to the figure below. The client-side JavaScript logic should receive a response from the server-side Web API which indicates that the course record has been saved.



Inspect the database's Course table. Find the new record and check the **CreatedById** and **UpdatedById** column. Notice that, the two fields have a value of 2.





Use the URL pattern, **http:<base address>/Courses/Index** to obtain the following view interface.

