**Sompalli Rajesh**

1. **Tools: NuGet Packge Manager: Manage NuGet Packages for Solution:**

**Browse:**

Microsoft.EntityFrameworkCore

Microsoft.EntityFrameworkCore.SqlServer

Microsoft.EntityFrameworkCore.Design

Microsoft.EntityFrameworkCore.Tools

**appsettings.json:**

"ConnectionStrings": {

"DefaultConnection": "Server=DESKTOP-54HLCQ0\\SQLEXPRESS;Database=StudentDB;Trusted\_Connection=True;TrustServerCertificate=True;"

Models:

ApplicationDbContext.cs:

using Microsoft.EntityFrameworkCore;

namespace StudentTest.Models

{

public class ApplicationDbContext:DbContext

{

public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)

: base(options)

{

}

public DbSet<User> Users { get; set; }

public DbSet<Question> Questions { get; set; }

public DbSet<Log> Logs { get; set; }

public DbSet<Answer> Answers { get; set; }

public DbSet<UserLogin> UserLogins { get; set; }

public DbSet<TestSubmission> TestSubmissions { get; set; }

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

// Seed data for questions

modelBuilder.Entity<Question>().HasData(

new Question

{

QuestionId = 1,

QuestionText = "Which of the following is the correct syntax for declaring a character in C#?",

Explanation = "The 'char' keyword is used to declare a character in C#."

},

new Question

{

QuestionId = 2,

QuestionText = "Which of the following is a value type in C#?",

Explanation = "int is a value type in C#, while string is a reference type."

},

new Question

{

QuestionId = 3,

QuestionText = "Which operator is used for concatenation in C#?",

Explanation = "The '+' operator is used to concatenate strings in C#."

},

new Question

{

QuestionId = 4,

QuestionText = "What is the default value of a Boolean in C#?",

Explanation = "The default value of a Boolean variable is 'false' in C#."

},

new Question

{

QuestionId = 5,

QuestionText = "What does the keyword 'static' mean in C#?",

Explanation = "The 'static' keyword means that a member belongs to the type itself rather than to a specific object."

},

new Question

{

QuestionId = 6,

QuestionText = "Which keyword is used to handle exceptions in C#?",

Explanation = "The 'try' keyword is used to start an exception handling block in C#."

},

new Question

{

QuestionId = 7,

QuestionText = "Which method is used to output a line of text to the console in C#?",

Explanation = "The 'Console.WriteLine()' method outputs a line of text to the console."

},

new Question

{

QuestionId = 8,

QuestionText = "What is the size of an int in C#?",

Explanation = "An 'int' in C# is 32-bits, which is 4 bytes."

},

new Question

{

QuestionId = 9,

QuestionText = "What is boxing in C#?",

Explanation = "Boxing is the process of converting a value type to an object type in C#."

},

new Question

{

QuestionId = 10,

QuestionText = "What is the base class of all classes in C#?",

Explanation = "'System.Object' is the base class of all classes in C#."

},

new Question

{

QuestionId = 11,

QuestionText = "What is an interface in C#?",

Explanation = "An interface defines a contract that implementing classes must follow in C#."

},

new Question

{

QuestionId = 12,

QuestionText = "Which of the following types in C# does not allow for inheritance?",

Explanation = "'sealed' types cannot be inherited in C#."

},

new Question

{

QuestionId = 13,

QuestionText = "What is polymorphism in C#?",

Explanation = "Polymorphism allows methods to have different implementations depending on the object calling them."

},

new Question

{

QuestionId = 14,

QuestionText = "What does the keyword 'void' indicate in a method declaration?",

Explanation = "The 'void' keyword indicates that the method does not return any value."

},

new Question

{

QuestionId = 15,

QuestionText = "Which of the following is a reference type in C#?",

Explanation = "String is a reference type in C#."

},

new Question

{

QuestionId = 16,

QuestionText = "What is the purpose of the 'readonly' keyword in C#?",

Explanation = "'readonly' variables can only be assigned values during declaration or in the constructor."

},

new Question

{

QuestionId = 17,

QuestionText = "What is the maximum value of an int in C#?",

Explanation = "The maximum value of an 'int' in C# is 2,147,483,647."

},

new Question

{

QuestionId = 18,

QuestionText = "What is the result of the expression '3 + 4 \* 2' in C#?",

Explanation = "The result is 11 due to operator precedence (multiplication before addition)."

},

new Question

{

QuestionId = 19,

QuestionText = "Which of the following loops in C# is guaranteed to execute at least once?",

Explanation = "'do...while' loops are guaranteed to execute at least once."

},

new Question

{

QuestionId = 20,

QuestionText = "Which access modifier in C# allows a class member to be accessible only within its own class?",

Explanation = "The 'private' access modifier restricts access to the class in which the member is defined."

}

);

// Seed data for answers

modelBuilder.Entity<Answer>().HasData(

new Answer { AnswerId = 1, AnswerText = "char", IsCorrect = true, QuestionId = 1 },

new Answer { AnswerId = 2, AnswerText = "string", IsCorrect = false, QuestionId = 1 },

new Answer { AnswerId = 3, AnswerText = "bool", IsCorrect = false, QuestionId = 1 },

new Answer { AnswerId = 4, AnswerText = "int", IsCorrect = false, QuestionId = 1 },

new Answer { AnswerId = 5, AnswerText = "int", IsCorrect = true, QuestionId = 2 },

new Answer { AnswerId = 6, AnswerText = "string", IsCorrect = false, QuestionId = 2 },

new Answer { AnswerId = 7, AnswerText = "bool", IsCorrect = false, QuestionId = 2 },

new Answer { AnswerId = 8, AnswerText = "decimal", IsCorrect = false, QuestionId = 2 },

new Answer { AnswerId = 9, AnswerText = "+", IsCorrect = true, QuestionId = 3 },

new Answer { AnswerId = 10, AnswerText = "&", IsCorrect = false, QuestionId = 3 },

new Answer { AnswerId = 11, AnswerText = "!", IsCorrect = false, QuestionId = 3 },

new Answer { AnswerId = 12, AnswerText = "-", IsCorrect = false, QuestionId = 3 },

new Answer { AnswerId = 13, AnswerText = "false", IsCorrect = true, QuestionId = 4 },

new Answer { AnswerId = 14, AnswerText = "true", IsCorrect = false, QuestionId = 4 },

new Answer { AnswerId = 15, AnswerText = "null", IsCorrect = false, QuestionId = 4 },

new Answer { AnswerId = 16, AnswerText = "0", IsCorrect = false, QuestionId = 4 },

new Answer { AnswerId = 17, AnswerText = "It belongs to the class, not the instance", IsCorrect = true, QuestionId = 5 },

new Answer { AnswerId = 18, AnswerText = "It is inherited", IsCorrect = false, QuestionId = 5 },

new Answer { AnswerId = 19, AnswerText = "It is abstract", IsCorrect = false, QuestionId = 5 },

new Answer { AnswerId = 20, AnswerText = "It is readonly", IsCorrect = false, QuestionId = 5 },

new Answer { AnswerId = 21, AnswerText = "The 'try' keyword is used to start an exception handling block in C#", IsCorrect = true, QuestionId = 6 },

new Answer { AnswerId = 23, AnswerText = "It is inherited", IsCorrect = false, QuestionId = 6 },

new Answer { AnswerId = 24, AnswerText = "It is abstract", IsCorrect = false, QuestionId = 6 },

new Answer { AnswerId = 25, AnswerText = "It is readonly", IsCorrect = false, QuestionId = 6 },

new Answer { AnswerId = 26, AnswerText = "Console.WriteLine()", IsCorrect = true, QuestionId = 7 },

new Answer { AnswerId = 27, AnswerText = "Console.Output()", IsCorrect = false, QuestionId = 7 },

new Answer { AnswerId = 28, AnswerText = "Console.PrintLine()", IsCorrect = false, QuestionId = 7 },

new Answer { AnswerId = 29, AnswerText = "System.Console()", IsCorrect = false, QuestionId = 7 },

new Answer { AnswerId = 30, AnswerText = "4 bytes", IsCorrect = true, QuestionId = 8 },

new Answer { AnswerId = 31, AnswerText = "8 bytes", IsCorrect = false, QuestionId = 8 },

new Answer { AnswerId = 32, AnswerText = "2 bytes", IsCorrect = false, QuestionId = 8 },

new Answer { AnswerId = 33, AnswerText = "16 bytes", IsCorrect = false, QuestionId = 8 },

new Answer { AnswerId = 34, AnswerText = "Converting a value type to an object type", IsCorrect = true, QuestionId = 9 },

new Answer { AnswerId = 35, AnswerText = "Converting a reference type to a value type", IsCorrect = false, QuestionId = 9 },

new Answer { AnswerId = 36, AnswerText = "Converting an int to a string", IsCorrect = false, QuestionId = 9 },

new Answer { AnswerId = 37, AnswerText = "Converting an object type to a value type", IsCorrect = false, QuestionId = 9 },

new Answer { AnswerId = 38, AnswerText = "System.Object", IsCorrect = true, QuestionId = 10 },

new Answer { AnswerId = 39, AnswerText = "System.Base", IsCorrect = false, QuestionId = 10 },

new Answer { AnswerId = 40, AnswerText = "System.Class", IsCorrect = false, QuestionId = 10 },

new Answer { AnswerId = 41, AnswerText = "System.Root", IsCorrect = false, QuestionId = 10 },

new Answer { AnswerId = 42, AnswerText = "A contract that classes must implement", IsCorrect = true, QuestionId = 11 },

new Answer { AnswerId = 43, AnswerText = "A class with only static methods", IsCorrect = false, QuestionId = 11 },

new Answer { AnswerId = 44, AnswerText = "A sealed class", IsCorrect = false, QuestionId = 11 },

new Answer { AnswerId = 45, AnswerText = "A method that cannot return a value", IsCorrect = false, QuestionId = 11 },

new Answer { AnswerId = 46, AnswerText = "Sealed", IsCorrect = true, QuestionId = 12 },

new Answer { AnswerId = 47, AnswerText = "Abstract", IsCorrect = false, QuestionId = 12 },

new Answer { AnswerId = 48, AnswerText = "Virtual", IsCorrect = false, QuestionId = 12 },

new Answer { AnswerId = 49, AnswerText = "Static", IsCorrect = false, QuestionId = 12 },

new Answer { AnswerId = 50, AnswerText = "The ability of a method to have different implementations", IsCorrect = true, QuestionId = 13 },

new Answer { AnswerId = 51, AnswerText = "The ability to inherit from multiple classes", IsCorrect = false, QuestionId = 13 },

new Answer { AnswerId = 52, AnswerText = "The ability of a class to have multiple constructors", IsCorrect = false, QuestionId = 13 },

new Answer { AnswerId = 53, AnswerText = "The ability to overload operators", IsCorrect = false, QuestionId = 13 },

new Answer { AnswerId = 54, AnswerText = "The method does not return a value", IsCorrect = true, QuestionId = 14 },

new Answer { AnswerId = 55, AnswerText = "The method returns an integer", IsCorrect = false, QuestionId = 14 },

new Answer { AnswerId = 56, AnswerText = "The method can return any type", IsCorrect = false, QuestionId = 14 },

new Answer { AnswerId = 57, AnswerText = "The method is static", IsCorrect = false, QuestionId = 14 },

new Answer { AnswerId = 58, AnswerText = "String", IsCorrect = true, QuestionId = 15 },

new Answer { AnswerId = 59, AnswerText = "Int", IsCorrect = false, QuestionId = 15 },

new Answer { AnswerId = 60, AnswerText = "Char", IsCorrect = false, QuestionId = 15 },

new Answer { AnswerId = 61, AnswerText = "Float", IsCorrect = false, QuestionId = 15 },

new Answer { AnswerId = 62, AnswerText = "It makes a variable assignable only in declaration or constructor", IsCorrect = true, QuestionId = 16 },

new Answer { AnswerId = 63, AnswerText = "It prevents the variable from being assigned a value", IsCorrect = false, QuestionId = 16 },

new Answer { AnswerId = 64, AnswerText = "It makes a variable accessible only within its class", IsCorrect = false, QuestionId = 16 },

new Answer { AnswerId = 65, AnswerText = "It allows the variable to be modified from anywhere", IsCorrect = false, QuestionId = 16 },

new Answer { AnswerId = 66, AnswerText = "2,147,483,647", IsCorrect = true, QuestionId = 17 },

new Answer { AnswerId = 67, AnswerText = "9,223,372,036,854,775,807", IsCorrect = false, QuestionId = 17 },

new Answer { AnswerId = 68, AnswerText = "32,767", IsCorrect = false, QuestionId = 17 },

new Answer { AnswerId = 69, AnswerText = "65,535", IsCorrect = false, QuestionId = 17 },

new Answer { AnswerId = 70, AnswerText = "11", IsCorrect = true, QuestionId = 18 },

new Answer { AnswerId = 71, AnswerText = "14", IsCorrect = false, QuestionId = 18 },

new Answer { AnswerId = 72, AnswerText = "7", IsCorrect = false, QuestionId = 18 },

new Answer { AnswerId = 73, AnswerText = "10", IsCorrect = false, QuestionId = 18 },

new Answer { AnswerId = 74, AnswerText = "do...while", IsCorrect = true, QuestionId = 19 },

new Answer { AnswerId = 75, AnswerText = "while", IsCorrect = false, QuestionId = 19 },

new Answer { AnswerId = 76, AnswerText = "for", IsCorrect = false, QuestionId = 19 },

new Answer { AnswerId = 77, AnswerText = "foreach", IsCorrect = false, QuestionId = 19 },

new Answer { AnswerId = 78, AnswerText = "private", IsCorrect = true, QuestionId = 20 },

new Answer { AnswerId = 79, AnswerText = "public", IsCorrect = false, QuestionId = 20 },

new Answer { AnswerId = 80, AnswerText = "protected", IsCorrect = false, QuestionId = 20 },

new Answer { AnswerId = 81, AnswerText = "internal", IsCorrect = false, QuestionId = 20 }

);

}

}

}

**Program.cs:**

builder.Services.AddDbContext<ApplicationDbContext>(options =>

options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));

Note: Update-Database

**Models:**

**User.cs:**

using System.ComponentModel.DataAnnotations;

namespace StudentTest.Models

{

public class User

{

[Key]

public int UserId { get; set; }

[Required]

public string UserName { get; set; }

[Required]

[EmailAddress]

public string Email { get; set; }

[Required]

[MinLength(6)]

public string Password { get; set; }

public UserLogin UserLogin { get; set; } // Navigation property to UserLogin

}

}

**UserLogin:**

using System.ComponentModel.DataAnnotations;

namespace StudentTest.Models

{

public class UserLogin

{

[Key]

public int UserLoginId { get; set; }

[Required]

public int UserId { get; set; } // Foreign key to User table

[Required]

public string Password { get; set; }

[Required]

public DateTime LoginTime { get; set; }

[Required]

public bool IsLoggedIn { get; set; }

public User User { get; set; } // Navigation property

}

}

**Question.cs:**

using System.ComponentModel.DataAnnotations;

namespace StudentTest.Models

{

public class Question

{

[Key]

public int QuestionId { get; set; }

[Required]

public string QuestionText { get; set; }

public ICollection<Answer> Answers { get; set; }

public string Explanation { get; set; }

public TestSubmission TestSubmission { get; set; } //Navigation Property

}

}

**Answer.cs:**

using System.ComponentModel.DataAnnotations;

namespace StudentTest.Models

{

public class Answer

{

[Key]

public int AnswerId { get; set; }

[Required]

public string AnswerText { get; set; }

public bool IsCorrect { get; set; }

public int QuestionId { get; set; }

public Question Question { get; set; } // Navigation property

}

}

**Log.cs:**

using System.ComponentModel.DataAnnotations;

namespace StudentTest.Models

{

public class Log

{

[Key]

public int LogId { get; set; }

public string Action { get; set; } // e.g., "Login", "Submit Test"

public DateTime TimeStamp { get; set; }

public string Details { get; set; } // Store additional information like user ID, etc.

}

}

**TestSubmission:**

using System.ComponentModel.DataAnnotations;

namespace StudentTest.Models

{

public class TestSubmission

{

public int QuestionId { get; set; }

[Key]

public int SelectedAnswerId { get; set; }

public Question Question { get; set; }// navigation Property

}

}

**Submit Test:**

namespace StudentTest.Models

{

public class SubmitTest

{

public int QuestionId { get; set; }

public int SelectedAnswerId { get; set; }

}

}

Note: Add-Migration and Update-Database

**UserController:**

using Microsoft.AspNetCore.Mvc;

using StudentTest.Models;

using System.ComponentModel.DataAnnotations;

namespace StudentTest.Controllers

{

public class UserController : Controller

{

public ApplicationDbContext \_context;

public UserController(ApplicationDbContext context)

{

\_context = context;

}

// GET: User/Register

[HttpGet]

public IActionResult Register()

{

return View();

}

// POST: User/Register

[HttpPost]

public IActionResult Register(User model)

{

// Check if the username or email already exists

if (\_context.Users.Any(u => u.UserName == model.UserName || u.Email == model.Email))

{

ModelState.AddModelError("", "Username or Email already exists.");

return View(model);

}

\_context.Users.Add(model);

\_context.SaveChanges();

return RedirectToAction("Login");

}

// GET: User/Login

[HttpGet]

public IActionResult Login()

{

return View();

}

// POST: User/Login

[HttpPost]

public IActionResult Login(string username, string password)

{

var user = \_context.Users.FirstOrDefault(u => u.UserName == username && u.Password == password);

if (user != null)

{

// Create a login record

var loginRecord = new UserLogin

{

UserId = user.UserId,

LoginTime = DateTime.Now,

IsLoggedIn = true

};

\_context.UserLogins.Add(loginRecord);

\_context.SaveChangesAsync();

return RedirectToAction("StartTest"); // Redirect to test

}

else

{

ModelState.AddModelError("", "Invalid username or password.");

return View(new User());

}

}

public IActionResult StartTest()

{

// Fetch 20 random questions from the database along with their answers

var questions = \_context.Questions

.Select(q => new Question

{

QuestionId = q.QuestionId,

QuestionText = q.QuestionText,

Explanation = q.Explanation,

Answers = q.Answers.Select(a => new Answer

{

AnswerId = a.AnswerId,

AnswerText = a.AnswerText

}).ToList()

})

.OrderBy(q => Guid.NewGuid()) // Shuffle to get random questions

.Take(20)

.ToList();

// Log action (Start Test)

\_context.Logs.Add(new Log

{

Action = "Start Test",

TimeStamp = DateTime.Now,

Details = "User started the test."

});

\_context.SaveChanges();

return View(questions);

}

// POST: Test/Submit

[HttpPost]

public IActionResult Submit(List<SubmitTest> Answers)

{

if (Answers == null || !Answers.Any())

{

return Content("No answers selected.");

}

int correctAnswers = 0;

foreach (var submission in Answers)

{

// Find the correct answer for the question

var correctAnswer = \_context.Answers

.FirstOrDefault(a => a.QuestionId == submission.QuestionId && a.IsCorrect);

// Check if the user's selected answer is correct

if (correctAnswer != null && correctAnswer.AnswerId == submission.SelectedAnswerId)

{

correctAnswers++;

}

// Log each submission (optional)

\_context.Logs.Add(new Log

{

Action = "Question Attempted",

TimeStamp = DateTime.Now,

Details = $"User attempted question {submission.QuestionId} with answer {submission.SelectedAnswerId}."

});

}

// Save all logs (optional)

\_context.SaveChanges();

// Calculate the score (e.g., out of 20)

int score = correctAnswers;

// Log the final submission

\_context.Logs.Add(new Log

{

Action = "Submit Test",

TimeStamp = DateTime.Now,

Details = $"User submitted the test and scored {score}."

});

\_context.SaveChanges();

// Redirect to the result page with the score

return RedirectToAction("Result", new { score = score });

}

public IActionResult Result(int score)

{

// Optional: Retrieve total questions for additional context (if needed)

int totalQuestions = \_context.Answers.Where(a => a.IsCorrect).Count(); // Assuming 1 correct answer per question

// Optional: Calculate the percentage score

double percentageScore = ((double)score / totalQuestions) \* 100;

// Pass data to the view using ViewBag or a strongly-typed view model

ViewBag.Score = score;

ViewBag.TotalQuestions = totalQuestions;

ViewBag.PercentageScore = percentageScore;

// Log action (View Results)

\_context.Logs.Add(new Log

{

Action = "View Results",

TimeStamp = DateTime.Now,

Details = $"User viewed the test results with a score of {score} out of {totalQuestions}." // You can include additional user-related information here, e.g., "UserId: 123"

});

// Save the log entry

\_context.SaveChanges();

return View(); // Or return View() if not using a view model

}

}

}

**Views: Razor View Empty:**

**Register:**

@model StudentTest.Models.User

<style>

body {

background-color:aqua

}

</style>

<h2 style="color:gold">Welcome,Fill The Form To Register</h2>

<br>

<a href="/User/Login">Already Registered? Please Login Here</a>

<br>

<**form** action="/User/Register" method="post">

<div class="form-group">

<label for="username">Username</label>

<input type="text" id="username" name="UserName" class="form-control" />

</div>

<div class="form-group">

<label for="email">Email</label>

<input type="email" id="email" name="Email" class="form-control" />

</div>

<div class="form-group">

<label for="password">Password</label>

<input type="password" id="password" name="Password" class="form-control" />

</div><br>

<button type="submit" class="btn btn-primary">Register</button>

</**form**>

**Login:**

<style>

body

{

background-color:aqua;

}

</style>

<**img** src="~/offline-omr-exam-1.png" />

</img>

<h2>Welcome To Login Page</h2>

<br>

<h4>Note:Registered Users Can Able To Perform Login</h4>

<br>

<a href="\User\Register">Not Register? Click Here</a>

<br>

<**form** action="/User/Login" method="post" id="loginForm">

<div class="form-group">

<label for="username">Username</label>

<input type="text" id="username" name="username" class="form-control" />

</div>

<div class="form-group">

<label for="password">Password</label>

<input type="password" id="password" name="password" class="form-control" />

</div><br>

<button type="submit" class="btn btn-primary">Login</button>

</**form**>

<!-- Loading GIF -->

<div id="loadingDiv" style="display:none;">

<**img** src="~/loading.gif" alt="Loading..." />

<p>Loading, please wait...</p>

</div>

<!-- JavaScript to show loading GIF on form submit -->

<script>

document.getElementById('loginForm').addEventListener('submit', function () {

// Show the loading GIF

document.getElementById('loadingDiv').style.display = 'block';

});

</script>

<style>

#loadingDiv {

position: fixed;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

z-index: 9999; /\* Ensures it stays on top of other elements \*/

text-align: center;

}

#loadingDiv img {

width: 100px; /\* Adjust size of the loading GIF \*/

}

/\* Optional: Prevent user from interacting with the page during loading \*/

body.loading {

pointer-events: none;

opacity: 0.7; /\* Dim the page while loading \*/

}

</style>

**StartTest:**

@model List<StudentTest.Models.Question>

<h2>C# Test</h2>

<br>

<h3>20 Questions With Multiple Answers,Choose One Carefully</h3>

<br>

<**form** **asp-action**="Submit" method="post">

@for (int i = 0; i < Model.Count; i++)

{

var question = Model[i];

<div class="question">

<h4>@question.QuestionText</h4>

@foreach (var answer in question.Answers)

{

<div>

<!-- Change the name attribute to correctly bind to the model -->

<input type="radio"

name="Answers[@i].SelectedAnswerId"

value="@answer.AnswerId" required />

<label>@answer.AnswerText</label>

</div>

}

<!-- Include the QuestionId as a hidden input -->

<input type="hidden"

name="Answers[@i].QuestionId"

value="@question.QuestionId" />

</div>

}

<button type="submit" class="btn btn-primary">Submit Test</button>

</**form**>

**Result:**

@{

ViewBag.Title = "Test Results";

}

<h2 class="mt-4">Test Results</h2>

<div class="alert alert-info">

<p>Your score is: <strong>@ViewBag.Score</strong></p>

<p>Total Number Of Questions: <strong>@ViewBag.TotalQuestions</strong></p>

<p>Your Percentage: <strong>@ViewBag.PercentageScore</strong></p>

</div>

<a href="@Url.Action("StartTest", "User")" class="btn btn-primary">Retake Test</a>

**Layout:**

<li class="nav-item">

<**a** class="nav-link text-dark" **asp-area**="" **asp-controller**="User" **asp-action**="Register">Register</**a**>

</li>

<li class="nav-item">

<**a** class="nav-link text-dark" **asp-area**="" **asp-controller**="User" **asp-action**="Login">Login</**a**>

</li>

Note: Compile and Run Project