Flawless Academy - V1

Grade settings: Maximum grade: 100
Based on: Flawless Academy - V1
Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum execution time**: 240 s **Maximum memory used**: 1.50 GiB **Maximum execution file size**: 1.50 GiB

Flawless Academy

Click here to download the code skeleton

Flawless Academy offers programs for dance, vocals, and instruments. They are in need of a web application to customers to show their available packages and the estimated cost for it. Help them by developing an application

Create a Spring MVC Spring Boot Application for developing a flawless Academy Application. Design a enrollment page to choose an appropriate package **(â€œClassicalDanceâ€• , â€ KarnaticVocalsâ€ , â€œWesternDanceâ€ , â€œDrawingâ€ , â€œInstrumentsâ€•)** and enter the Number of sessions per week

On clicking program cost button, the application should calculate the program cost depending on the program chosen and the cost per session which the customers need to pay for the package The customer has to then be redirected to estimationPage.jsp page that displays the message **â€œ**Welcome to Flawless Academy. Your preferred program cost is Rs. **<<cost>>â€•**

Application Work Flow:

- **AcademyController** is the Controller class.
- **Academy** is the model class with three attributes program, costPerSession, sessionsPerWeek and weeksPerMonth along with its getters and setters.
- **AcademyService** is the Service class which has a method called calculateProgramCost that takes Academy object as its argument and returns double.
- This method needs to set the **costPerSession** instance variable based on the cost per session (in Rs) value given in the below mentioned table.
- Calculate the program cost depending on the **costPerSession** for the program chosen and the **sessionsPerWeek**.
- By default the **weeksPerMonth** value should be 4.
- Program cost should be returned as double.

Program	Cost per session (in Rs)
ClassicalDance	150.0
KarnaticVocals	100.0
WesternDance	125.0
Drawing	130.0
Instruments	200.0

Initially, the customer should be routed via the AcademyController's **showPage** method to e**nrollmentPage.jsp** that allows user to choose the program and enter the **sessionsPerWeek**

[Note: showPage method has to be written inside the AcademyController]

A method in the AcademyController known as **buildState** should be annotated with the ModelAttribute **â€œprogramListâ€•** . This method should populate the preferred

programs **(ClassicalDance, KarnaticVocals,WesternDance,Drawing,Instruments)** in the Map as key-value pair. Both key and value be the same program. (Example: Key- ClassicalDance, Value- ClassicalDance, Key- KarnaticVocals, Value- KarnaticVocals, etc) and then return the Map. This should be then used to auto-populate the program in the **enrollmentPage**.jsp

[Note: buildState method should be written inside the AcademyController]

On clicking the program Cost button, the AcademyController's calculateProgramCost method should be called. This method takes three arguments - model attribute named â€œ**academyBean**â€• which holds the form populated Academy Object, BindingResult and the ModelMap.

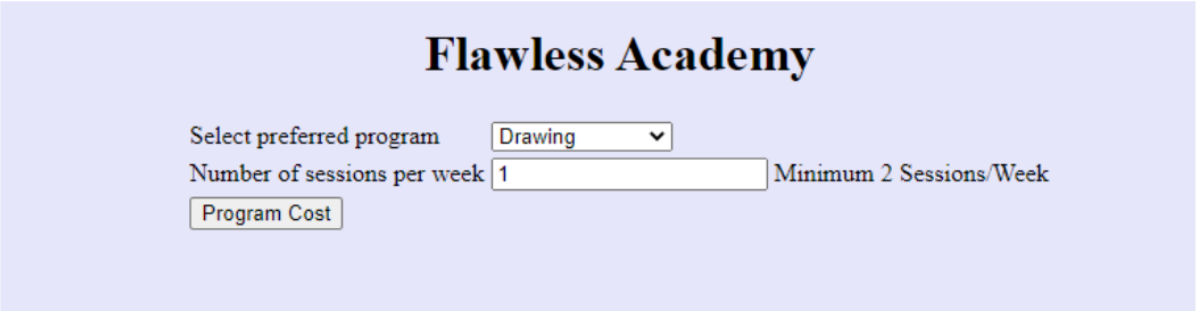
- This method should calculate the program cost by invoking the **calculateProgramCost** method of the **AcademyService**.

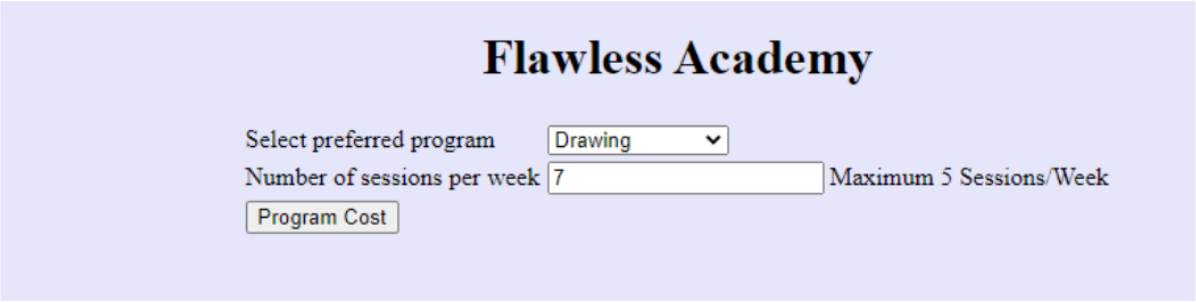
Calculation of program cost:
If the preferred program is "Drawing" and sessionsPerWeek is 3.
program cost= sessionsPerWeek* costPerSession * weeksPerMonth (by default weeksPerMonth is 4)
program cost= 3 * 130.0 * 4 => 1560.0

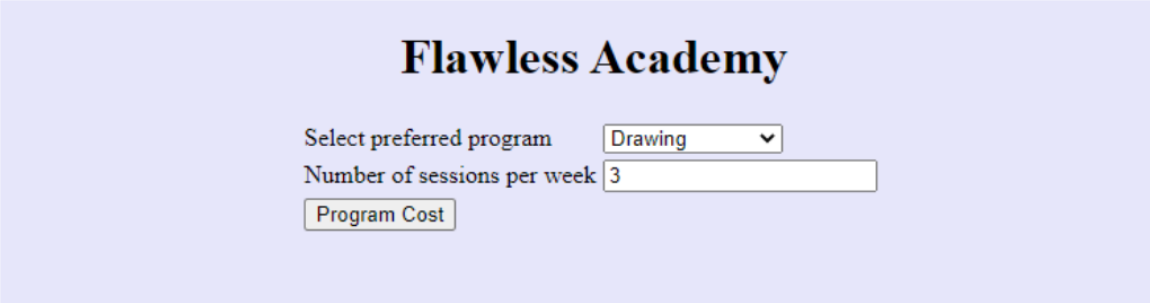
If the **sessionsPerWeek** entered by the customer is less than 2 then an error message **"Minimum 2 Sessions/Week"** has to be displayed in the **enrollmentPage.jsp**. If the **sessionsPerWeek** entered by the customer is greater than 5 then an error message **"Maximum 5 Sessions/Week"** has to be displayed in the **enrollmentPage.jsp** To do this validation use appropriate annotation above **sessionsPerWeek** attribute in **Academy** model class.

Screen designs and Expected Output:

enrollmentPage.jsp

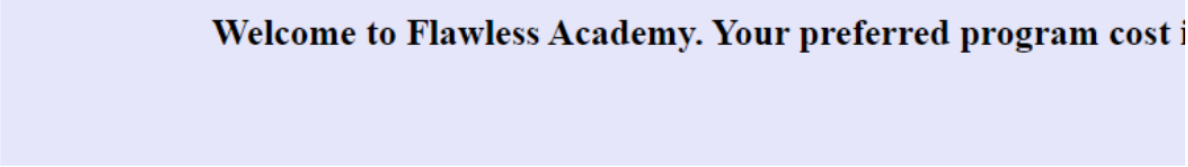






- Redirect the user to **estimationPage.jsp** page with a message **â€œWelcome to Flawless Academy. Your preferred program cost is Rs. << cost>>â€**

estimationPage.jsp



Design Constraints:

UI Design Constraints:

enrollmentPage.jsp		
Component	**ID**	**Constraints**
Select	program	Should be auto-populated using the model attribute written above the buildState method inside the AcademyController. Do not hard code the values
Textbox	sessionsPerWeek	Should not be less than 2 and should not be greater than 5
submit	submit	Name and id attribute should be submit -

Note: In **estimationPage.jsp**, the Result has to be rendered in the **<h2>** tag

Component Specification

Controller

AcademyController			
AttributeName	**AttributeType**	**Access Specifier**	**Constraints**
AcademyService	service	private	Use annotation to Autowire

AcademyController			
Method Name	**Method Argument name:type**	**Return type**	**RequestMapping URL & Request Method**

showPage	modelAttribute â€œacademyBeanâ€ • : Academy	String	/enrollmentPage & GET
calculateProgramCost	modelAttribute â€œacademyBeanâ€ • :Academy, result:BindingResult, model:ModelMap	String	/progEstimation & POST
buildState		Map <String,String>	Should be annotated with ModelAttribute with name â€œprogramListâ€ •

Service:

AcademyService		
Method Name	**Method Argument name:type**	**Return type**
calculateProgramCost	academyBean:Academy	double

Model:

Academy	
AttributeName	**AttributeType**
program	String
costPerSession	double
sessionsPerWeek	int

Overall Design Constraints:

- **AcademyController** should be inside the package **com.controller**
- **AcademyService** should be inside the package **com.service**
- **Academy** should be in the package **com.model**
- Use appropriate annotation to configure **AcademyService** as a Service
- Use appropriate annotation to configure **cademyController** as a Controller
- **AcademyService** should be **autowired** inside the**AcademyController**.
- Use annotations to implement the business Validation as specified in the screenshot [That is when the **sessionsPerWeek** is less than 2 then error message **"Minimum 2 Sessions/Week"** and when sessionsPerWeek greater than 5 then error message **â€œMaximum 5 Sessions/Weekâ€•** should be rendered in the UI.]
- Do not change the property name given in the application.properties files, you can change the value and you can include additional property if needed.
- In the pom.xml you are provided with all the dependencies needed for developing the application.
- You will not be evaluated based on the UI design (layout, color, formatting, etc.). You are free to have a basic UI with all the required UI components (input fields, buttons, labels, etc.). Expected components with the id alone should be designed as per the requirement.
- Adhere to the design specifications mentioned in the case study.

- **Do not change or delete the class/method names or return types that are provided to you as a part of the base code skeleton.**
- Please make sure that your code does not have any compilation errors while submitting your case study solution.

Result Description

Submitted by sathya .

Automatic evaluation[+]

FlawlessAcademy/pom.xml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
3      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-
4.0.0.xsd">
4      <modelVersion>4.0.0</modelVersion>
5      <parent>
6                  <groupId>org.springframework.boot</groupId>
7                  <artifactId>spring-boot-starter-parent</artifactId>
8      <version>2.1.8.RELEASE</version>
9                  <relativePath/> <!-- lookup parent from repository -->
10     </parent>
11     <groupId>com.example</groupId>
12     <artifactId>FlawlessAcademy</artifactId>
13     <version>0.0.1-SNAPSHOT</version>
14     <name>FlawlessAcademy</name>
15     <description>Demo project for Spring Boot</description>
16     <properties>
17                 <java.version>1.8</java.version>
18     </properties>
19
20     <dependencies>
21                 <dependency>
22                             <groupId>org.springframework.boot</groupId>
23                             <artifactId>spring-boot-starter-web</artifactId>
24                 </dependency>
25
26                  <dependency>
27     <groupId>org.apache.tomcat.embed</groupId>
28     <artifactId>tomcat-embed-jasper</artifactId>
29     <scope>provided</scope>
30    </dependency>
31    <dependency>
32   <groupId>javax.servlet</groupId>
33   <artifactId>servlet-api</artifactId>
34   <version>2.5</version>
35   <scope>provided</scope>
36 </dependency>
37 <dependency>
38   <groupId>javax.servlet.jsp</groupId>
39   <artifactId>jsp-api</artifactId>
40   <version>2.1</version>
41   <scope>provided</scope>
42 </dependency>
43
44 <dependency>
45   <groupId>taglibs</groupId>
```

```
 <artifactId>standard</artifactId>
 <version>1.1.2</version>
</dependency>
<dependency>
 <groupId>javax.servlet</groupId>
 <artifactId>jstl</artifactId>
 <version>1.2</version>
</dependency>
<!-- Spring test dependencies -->
    <dependency>
                        <groupId>org.mockito</groupId>
                        <artifactId>mockito-all</artifactId>
                        <version>1.10.19</version>
                        <scope>test</scope>
                </dependency>

                <!-- <dependency>
                        <groupId>org.seleniumhq.selenium</groupId>
                        <artifactId>selenium-java</artifactId>
                        <version>2.53.0</version>
                </dependency>-->

                <dependency>
   <groupId>org.seleniumhq.selenium</groupId>
   <artifactId>htmlunit-driver</artifactId>
   <version>2.26</version>
</dependency>

    <!-- https://mvnrepository.com/artifact/org.w3c.css/sac -->
<dependency>
   <groupId>org.w3c.css</groupId>
   <artifactId>sac</artifactId>
   <version>1.3</version>
</dependency>


<dependency>
                        <groupId>org.springframework</groupId>
                        <artifactId>spring-test</artifactId>
                         <version>5.1.7.RELEASE</version>  <!-- 4.0.5 -->
                        <scope>test</scope>
                </dependency>
<!-- End of spring test dependencies -->



                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-test</artifactId>
                        <scope>test</scope>
                        <exclusions>
                                <exclusion>
                                        <groupId>org.junit.vintage</groupId>
                                        <artifactId>junit-vintage-engine</artifactId>
                                </exclusion>
                        </exclusions>
                </dependency>
                <!-- https://mvnrepository.com/artifact/org.sonarsource.scanner.maven/sonar-maven-
plugin -->
<dependency>
   <groupId>org.sonarsource.scanner.maven</groupId>
   <artifactId>sonar-maven-plugin</artifactId>
   <version>3.8.0.2131</version>
```

```
</dependency>

    </dependencies>

    <build>
                <plugins>
                        <plugin>
                                <groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-maven-plugin</artifactId>
                        </plugin>
                </plugins>
    </build>

</project>
```

FlawlessAcademy/src/main/java/com/controller/AcademyController.java

```
package com.controller;

import java.util.HashMap;
import java.util.Map;
import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

import com.model.Academy;
import com.service.AcademyService;

@Controller
public class AcademyController {

    @Autowired
    private AcademyService service;

    @RequestMapping(value = "/enrollmentPage", method = RequestMethod.GET)
    public String showPage(@ModelAttribute("academyBean") Academy academyBean) {
                System.out.println("in controller");
                return "enrollmentPage";
    }
    @ModelAttribute("programList")
    public  Map<String, String> buildState(){

                Map<String, String> progMap = new HashMap<String, String>();
                progMap.put("ClassicalDance", "ClassicalDance");
                progMap.put("KarnaticVocals", "KarnaticVocals");
                progMap.put("WesternDance","WesternDance");
                progMap.put("Drawing", "Drawing");
                progMap.put("Instruments","Instruments");

                return progMap;

    }

    @RequestMapping(value = "/progEstimation", method = RequestMethod.POST)
    public String calculateProgramCost(@ModelAttribute("academyBean") @Valid Academy
academyBean, BindingResult result,
                                ModelMap model) {
                if (result.hasErrors()) {
```

```
                                System.out.println("Error : " + result.toString());
                                return "enrollmentPage";
                        }
                        double cost=service.calculateProgramCost(academyBean);
                        model.addAttribute("cost", cost);
                        return "estimationPage";
        }


}

```

FlawlessAcademy/src/main/java/com/example/demo/FlawlessAcademyApplication.java

```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;

@SpringBootApplication
@ComponentScan({"com.*"})
public class FlawlessAcademyApplication {

        public static void main(String[] args) {
                        SpringApplication.run(FlawlessAcademyApplication.class, args);
        }

}

```

FlawlessAcademy/src/main/java/com/model/Academy.java

```
package com.model;

import javax.validation.constraints.Min;
import javax.validation.constraints.Max;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;

@Component
public class Academy {

        private String program;

        private double costPerSession;
        private int weeksPerMonth;
        @Min(value = 2, message = "Minimum 2 Sessions/Week")
        @Max(value = 5, message = "Maximum 5 Sessions/Week")
        private int sessionsPerWeek;

        public int getSessionsPerWeek() {
                        return sessionsPerWeek;
        }

        public void setSessionsPerWeek(int sessionsPerWeek) {
                        this.sessionsPerWeek = sessionsPerWeek;
        }

        public Academy()
        {

        }

```

```
33      public int getWeeksPerMonth() {
34              return weeksPerMonth;
35      }
36      public void setWeeksPerMonth(int weeksPerMonth) {
37              this.weeksPerMonth = weeksPerMonth;
38      }
39      public double getCostPerSession() {
40              return costPerSession;
41      }
42      public void setCostPerSession(double costPerSession) {
43              this.costPerSession = costPerSession;
44      }
45
46      public String getProgram() {
47              return program;
48      }
49      public void setProgram(String program) {
50              this.program = program;
51      }
52
53
54
55
56  }
57
58
```

FlawlessAcademy/src/main/java/com/service/AcademyService.java

```
1   package com.service;
2
3   import org.springframework.stereotype.Service;
4
5   import com.model.Academy;
6
7   @Service
8   public class AcademyService {
9
10      public double calculateProgramCost (Academy academyBean) {
11
12              double cost=0.0;
13              academyBean.setWeeksPerMonth(4);
14              System.out.println(academyBean.getSessionsPerWeek()+"
"+academyBean.getProgram()+" "+academyBean.getWeeksPerMonth());
15              if(academyBean.getProgram().equalsIgnoreCase("ClassicalDance") &&
academyBean.getSessionsPerWeek()>=1)
16              {
17                      academyBean.setCostPerSession(150.0);
18
        cost=academyBean.getCostPerSession()*academyBean.getSessionsPerWeek()*academyBean.getW
eeksPerMonth();
19              }
20      else if(academyBean.getProgram().equalsIgnoreCase("KarnaticVocals") &&
academyBean.getSessionsPerWeek()>=1 )
21      {
22              academyBean.setCostPerSession(100.0);
23
        cost=academyBean.getCostPerSession()*academyBean.getSessionsPerWeek()*academyBean.getW
eeksPerMonth();
24      }
25      else if(academyBean.getProgram().equalsIgnoreCase("WesternDance") &&
academyBean.getSessionsPerWeek()>=1)
26      {
27              academyBean.setCostPerSession(125.0);
```

```
28
        cost=academyBean.getCostPerSession()*academyBean.getSessionsPerWeek()*academyBean.getW
eeksPerMonth();
29      }
30      if(academyBean.getProgram().equalsIgnoreCase("Drawing") &&
academyBean.getSessionsPerWeek()>=1)
31      {
32                      academyBean.setCostPerSession(130.0);
33
        cost=academyBean.getCostPerSession()*academyBean.getSessionsPerWeek()*academyBean.getW
eeksPerMonth();
34      }
35  else if(academyBean.getProgram().equalsIgnoreCase("Instruments") &&
academyBean.getSessionsPerWeek()>=1)
36  {
37      academyBean.setCostPerSession(200.0);
38
        cost=academyBean.getCostPerSession()*academyBean.getSessionsPerWeek()*academyBean.getW
eeksPerMonth();
39  }
40      return cost;
41
42      }
43
44  }
45
```

FlawlessAcademy/src/main/resources/application.properties

```
1  server.port=9088
2  spring.mvc.view.prefix = /WEB-INF/views/
3  spring.mvc.view.suffix = .jsp
4  spring.mvc.static-path-pattern=/resources/**
5
6
```

FlawlessAcademy/src/main/webapp/WEB-INF/views/enrollmentPage.jsp

```
1  <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2     pageEncoding="ISO-8859-1" isELIgnored="false"%>
3  <%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
4
5  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
6  <html>
7  <body style="background-color:lavender">
8  <h1><center> Flawless Academy</center></h1>
9  <form:form method="post" action="progEstimation" modelAttribute="academyBean">
10 <table style="margin: 0px auto; margin-left: auto; margin-right:auto">
11
12
13                              <tr>
14                                      <td>Select preferred program</td>
15                                      <td>
16                                       <form:select path="program" id="program">
17                                                              <form:options items="${programList}"/>
18                                              </form:select>
19
20                                      </td>
21                              </tr>
22
23                              <tr>
24                              <td>Number of sessions per week</td>
25                              <td><form:input path="sessionsPerWeek" id="sessionsPerWeek"/></td>
26                              <td><form:errors path="sessionsPerWeek" id="errormsg"/></td>
27                              <tr>
```

```
                                        <td><input type="submit" value="Program Cost" name="submit"
/></td>

                                </tr>

                        </table>
</form:form>


</body>
</html>

```

FlawlessAcademy/src/main/webapp/WEB-INF/views/estimationPage.jsp

```
<%@page isELIgnored="false" %>
<html>
<body bgcolor="lavender">
<center><h2>Welcome to Flawless Academy. Your preferred program cost is Rs. ${cost}</h2></center>
</body>
</html>
```

Grade

Reviewed on Friday, 18 February 2022, 1:09 PM by Automatic grade
Grade 100 / 100
Assessment report
Assessment Completed Successfully
[-]**Grading and Feedback**

```
Feature Test  - 35.00 / 35.00(Success)
         * check whether the controller annotation is present above the Controller - 8.00
/ 8.00
         * check whether the ModelAttribute annotation is present above the Method - 8.00
/ 8.00
         * check whether the Service is autowired inside the Controller - 7.00 / 7.00
         * check whether Service annotation is present above the Service - 6.00 / 6.00
         * check whether the ModelAttribute annotation has the appropriate value above the
Method - 6.00 / 6.00

Functional testing - 25.00 / 25.00(Success)
         * check whether the program cost for the ClassicalDance type is calculated
properly - 5.00 / 5.00
         * check whether the program cost for the KarnaticVocals type is calculated
properly - 5.00 / 5.00
         * check whether the program cost for the WesternDance type is calculated properly
- 5.00 / 5.00
         * check whether the program cost for the Drawing type is calculated properly -
5.00 / 5.00
         * check whether the program cost for the Instruments is calculated properly -
5.00 / 5.00

Spring Testing in the Controller - 25.00 / 25.00(Success)
         * check whether the request is mapped for enrollmentPage and redirected to
enrollmentPage.jsp page - 5.00 / 5.00
         * check whether the request is mapped for estimationPage and redirected to
estimationPage.jsp page - 5.00 / 5.00
         * check whether the model attribute is specified with the name above the
buildState method that holds the Map containing the Preferred program - 5.00 / 5.00
         * check whether the request is mapped and redirected to the estimationPage page
for valid details - 5.00 / 5.00
         * check whether the request is mapped and redirected to the same page for Invalid
details - 5.00 / 5.00

UI Testing - 15.00 / 15.00(Success)
```

*   * check whether the enrollmentPage is rendered for the program type ClassicalDance - 2.00 / 2.00*
*   * check whether the enrollmentPage is rendered for the program type KarnaticVocals - 2.00 / 2.00*
*   * check whether the enrollmentPage is rendered for the program type WesternDance - 2.00 / 2.00*
*   * check whether the enrollmentPage is rendered for the program type Drawing - 2.00 / 2.00*
*   * check whether the enrollmentPage is rendered for the program type Instruments - 2.00 / 2.00*
*   * check whether the enrollmentPage is rendered with proper constraints for sessionsPerWeek - 1.00 / 1.00*
*   * check whether UI from enrollmentPage to estimationPage page is redirected after calculating the program cost for ClassicalDance - 1.00 / 1.00*
*   * check whether UI from enrollmentPage to estimationPage page is redirected after calculating the program cost for KarnaticVocals - 1.00 / 1.00*
*   * check whether UI from enrollmentPage to estimationPage page is redirected after calculating the program cost for WesternDance - 1.00 / 1.00*
*   * check whether the sessionsPerWeek field validated properly - 0.50 / 0.50*
*   * check whether the sessionsPerWeek field in Enrollment Page is validated properly - 0.50 / 0.50*