**《Multi-campus Student Management System Report》**

**Author：Junnan Zhao**

**Id：23010566**

**My GitHub URL: https://github.com/universaaser/23010566-Assignment2.git**

**Code files, reports, and videos are all available in the GitHub repository, please click on the link above to access them.**

**My video URL:** 【Video on Assignment2】 <https://www.bilibili.com/video/BV17sBxYTEFU/?share_source=copy_web&vd_source=c25689555dd567b22c46384421939ec4>

**My video was posted on BiliBili.**

1. **Accessing the Website**

**Main page**图形用户界面, 文本, 应用程序

描述已自动生成

It realizes the introduction of the basic functions of the website and supports the direct jump to the website functions. In addition, it has developed the school achievement display and excellent student display section, which enriches the content of the website.

Contact Page图形用户界面, 文本, 应用程序

描述已自动生成

**Student listing page（Index.cshtml）**

图形用户界面, 应用程序

描述已自动生成

**o Page layout**

**o There is a navigation bar at the top of the page that shows the current page as "Students" and a "Create New" button to add new student information.**

**o There is a data table in the middle of the page, and the table column names include Campus Name, Student Name, and Address.**

**o Each student record line also contains "Edit", "Details" and "Delete" operation links to facilitate user management operations on student information.**

**o Data filtering and sorting**

**o page provides data filtering and sorting functions. Users can select a specific campus to Filter data through the "Filter by campus" drop-down menu or select the sorting method through the "Sort By" drop-down menu.**

**o the page currently shows "Showing all 4 students", which indicates that the system is able to obtain and display the student data correctly.**

**o and the system can implement paging operations**

**2. Action functions (Create, Edit, Delete, Details)**

**o Add students (Create)**

**o by clicking the "Create New" button, the user can go to the "create. cshtml" page to add new student information. In this page, the user can input the information of the student, such as the campus, name, class and so on, and then submit the form, and the system will save the new student information to the database.**

**o Edit student information (Edit)**

**o After clicking the "Edit" link, the user will be taken to the "Edit.cshtml" page, which will display the existing information of the current student and allow the user to modify this information. After the modification is completed, the form will be submitted, and the system will update the student record in the database.**

**o View student Details**

**o the "details. cshtml" page is used to display the Details of a student, and the user can click on the "Details" link to view the full information of a particular student, which may include other additional attributes of the student, such as contact information.**

**o Delete the student (Delete)**

**o After clicking the "Delete" link, the system will pop up a confirmation dialog box, and the records of the corresponding students will be deleted from the database after confirmation.**图形用户界面, 文本, 应用程序, 网站

描述已自动生成

The operation functions of the Campus interface are as above. In addition, the correlation and independence between campus and students are technically realized. When a whole campus is deleted by mistake, the internal student data will be retained.

1. **Project structure**

**文本

描述已自动生成**

**1. Overall layout of the project**

**o the project root contains multiple folders, such as "Controllers," "Models," "Scripts," and "Views," which contain code and files for different feature modules.**

**o the "Connected Services" and "Properties" folders may be used for project configuration and files related to connecting to external services.**

**2. Controllers folder**

**o This folder contains several controller class files, such as "CampusController.cs", "HomeController.cs", and "StudentsController.cs".**

**o These controller classes are responsible for handling HTTP requests from the user, calling the appropriate business logic, and returning views or data to the frontend. For example, "StudentsController.cs" handles requests for actions related to students such as getting student list, adding student, editing student information, etc.**

**Models’ folder**

**o Contains two model class files "Campus.cs" and "Student.cs".**

**The o model class is used to define data structures that represent the table structure in the database. For example, the "Student.cs" class contains the attributes of the student such as name, campus, class, etc., which correspond to the fields of the "Student table" in the database.**

**Scripts folder**

**o This folder is usually used for JavaScript files that are used to implement the interactive logic of the front-end. Scripts for form validation, asynchronous data fetching, and more.**

**5. Views folder**

**o There are multiple subfolders under the views folder, such as "Campus" and "Students", corresponding to the view files of different modules.**

**o Each subfolder contains multiple. cshtml files, such as "Create.cshtml", "Delete.cshtml", "Details.cshtml", "Edit.cshtml", and "Index.cshtml". These view files are used to render the user interface, such as "Index.cshtml" in the "Students" folder for the student list page and "Create.cshtml" for the form page to add student information.**

**Feature Implementation**

**I. Model**

**1. Campus model**

**截图里有图片

描述已自动生成**

**The Campus model class is used to represent the campus information.**

**Contains an 'ID' attribute to uniquely identify each school district.**

**The 'Name' property is used to store the school district name, with strict validation constraints through a series of data annotations. The 'Required' attribute ensures that the district name cannot be empty; The 'StringLength' property limits the length of a district name to between 1 and 50 characters; The 'RegularExpression' property requires that the district name starts with a capital letter and can only contain letters and Spaces. These validations help ensure the accuracy and consistency of the data.**

**Also included is a dummy 'Students' collection property that establishes a one-to-many relationship with the' Student 'model, i.e. a campus can contain multiple students.**

**The Student model**

**文本

描述已自动生成**

**The 'Student' model class is used to represent student information.**

**The 'ID' attribute is used to uniquely identify each student.**

**The 'Name' property is used to store the student name, again with strict data validation. The 'Required' attribute ensures that names cannot be empty; The 'StringLength' property limits names to between 3 and 50 characters; The 'RegularExpression' property states that the name should only consist of letters, numbers, and Spaces.**

**The 'Address' property is used to store the student address and has similar validation rules, ensuring that the address is not empty and between 10 and 200 characters long, and can only consist of letters, numbers, and Spaces.**

**The 'CampusID' property is used to establish an association with the 'Campus' model, linking a student to the Campus through a foreign key relationship, and the' campus' property is used to navigate to the associated campus object.**

**2. CSS**

**文本

描述已自动生成**

**1. Overall navigation bar styles (.custom-Navbar)**

**- Set the navbar background color to dark gray (' #555 ') and removed the default padding to make the navbar look more compact.**

**2. Navigation bar container styles (.custom-container)**

**- Defines a container width of 100%, with a maximum width of 1200 pixels and horizontal center via 'margin: 0 auto'. Also, use 'display: flex', 'justify-content: space-between' and 'align-items: center' to lay out the navbar content so that it is aligned horizontally and centered vertically.**

**3. custom navbar-brand**

**- Specified a white branding text color, a font size of 30 pixels (' 30px '), removed the default link underlines, and set the left margin to 20 pixels to adjust the positioning of the branding text.**

**4. Navigation bar toggler style (.custom-toggler)**

**- The background color is set to transparent, no border, the mouse pointer is changed to pointer style, and the right margin is 10 pixels for adjusting the button position.**

**5. Navigation bar toggle icon styles (.custom-toggler-icon)**

**- A simple three-line icon with a width and height of 30 pixels is set via the 'background-image' property to display the toggle button icon when the navigation bar is collapsed.**

**6. Content area styles for collapsed navbars (.custom-navbar-collapse)**

**- Set 'flex-basis' to 100% and' flex-grow 'to 1 to control the width and scale of the content area after the navigation bar is collapsed.**

**7. Navigation bar list styles (.custom-navbar-nav)**

**- Remove the default styles for lists (' list-style-type: none ') and set padding to 0 and margin to 0. Use 'display: flex' and 'justify-content: space-around' to evenly distribute the navigation items across the navigation bar.**

**8. Single navigation link styles (.custom-nav-link**

**The text color is white without underscores, padding is set to 10 pixels (top and bottom) and 15 pixels (left and right), and a 'transition' property is applied on hover, which changes the background color to dark gray (' #555 ') with 5 pixels rounded corners.**

**9. Outer styles for the search form in the navigation bar (.custom-navbar-form)**

**Use 'display: flex' and 'align-items: center' to vertically center the search form content.**

**10. Search field input styles (.custom-form-group and.custom-form-control)**

**The 'custom-form-group' sets a spacing of 10 pixels from the submit button. 'custom-form-control' sets the input field to have rounded corners of 5 pixels, no border, and padding of 5 pixels (top and bottom) and 10 pixels (left and right).**

**11. Search form submit button styles (.custom-btn and.custom-btn-default)**

**- Button background color blue (' #007bff '), text color white, no border, rounded corners of 5 pixels, padding of 5 pixels (top and bottom) and 15 pixels (left and right), background color darned to '#0056b3' on mouse hover, transition effect achieved.**

**文本

描述已自动生成**

**12. Overall page styles (body and main)**

**- The 'body' font is set to 'Arial, sans-serif', line height is 1.6, default margins and padding are removed, and a gradient background color is set (from '#e6f7ff' to '#d0e9ff'). 'main' sets a semi-transparent white background (' rgba(255, 255, 255, 0.8) '), rounded corners of 10 pixels, a black shadow with 0.1 opacity, and a padding of 30 pixels to highlight the main content of the page.**

**13. title and lead styles (#title,.center-text, and.lead)**

**- '#title' sets the title text to center, 20 pixels margin at the bottom, a blue color (' #007bff '), and a font size of 36 pixels. '. Center-text 'is used to center the text. The '.lead 'sets the lead text to center, a bottom margin of 30 pixels, and a dark gray color (' #333').**

**14..Card**

**- The card has a 0.15 opacity black shadow with a transition effect that darkens to 0.2 opacity on mouseover, 8 pixels rounded corners, 20 pixels bottom margin, and a white background. '. Card-body 'sets the padding of the card content area to 20 pixels. '.card h2 'sets the bottom margin of the header to 15 pixels and a dark gray color (' #333'). The '.card p 'centers the text and gives it a dark gray color (' #333').**

**15. Button Styles (.btn)**

**- Button appears as an inline block element with a top margin of 10 pixels, padding of 10 pixels (top and bottom) and 20 pixels (left and right), rounded corners of 5 pixels, no border, background color of light blue (' #00bfff '), text color of white, and background color of '#0056b3' on mouse hover to achieve transition effect.**

**16. Text box styles (textarea and.textarea-label)**

**- 'textarea' is set to a width of 45%, a height of 150 pixels, a top margin of 20 pixels, a padding of 10 pixels, a gray border of 1 pixel, and rounded corners of 4 pixels. '.textarea-label 'sets the text label to center and 10 pixels margin at the bottom.**

**JavaScript**

**1. Navigation bar toggle button interaction logic, such as expanding or collapsing the navigation bar menu when a button is clicked.**

**2. The submit event handling of the search form. When the user enters the search keyword and clicks the submit button, the corresponding search operation is triggered, which may send an AJAX request to the server to obtain the search results and dynamically update the page display.**

**3. Form validation enhancement: In addition to model validation on the server side, JavaScript is used in the front-end to verify the user's input in real time. When the user enters the name or address of the student, it immediately checks whether it conforms to the prescribed format, providing more timely user feedback.**

**4. Dynamic effects of page elements, such as card fade in and out effects, button click animations, etc., to improve the user experience.**

**Four, function realization**

**图形用户界面, 文本, 应用程序

描述已自动生成**

**1. Search functionality**

**- Implement the search function in 'Index' method of 'StudentsController'.**

**- First get the query object 'students' (via' db.Students.Include(s => s.Campus) ') that contains the students and the district information.**

**- When the user enters a search term on the page and submits it, the 'search' parameter is not empty, then the 'students' query object is filtered by the' Where 'method, If the student's name, address, or school district Contains a search term (' students = students.where (s => s.Name.Contains(search) || s.dress.contains (search) ||) S.camus.name.contains (search)) ').**

**- The filtered results are used to generate a 'viewModel' and eventually passed to the view for display, which displays a list of students that match the search criteria.**

**2. Sorting function**

**- Also implement sorting in the 'Index' method.**

**- Receive the user's chosen sort via the 'sortBy' parameter.**

**- Sort the 'students' query object based on the' sortBy 'value in the' switch 'statement. For example, when 'sortBy' is' "name\_ascending" ', sort by student name in ascending order using 'OrderBy(s => s.Name)'; When 'sortBy' is' "name\_descending" ', use 'OrderByDescending(s => s.Name)' to sort by student name in descending order. If the value of 'sortBy' doesn't match any of the preset sorts, it will default to sorting by name in ascending order.**

**- The sorted results are also reflected in the 'viewModel', which finally displays the list of students in the view in sorted order. The view model also maintains a 'Sorts' dictionary, which is used to generate drop-down lists of sorts options in the view, allowing the user to select different sorts.图形用户界面, 文本, 应用程序

描述已自动生成**

**V. Summary**

**The multi-campus student management system has certain rationality and practicability in the project structure and function realization. The back-end logic is realized by ASP.NET framework, and the front-end interface and interaction are realized by combining HTML, CSS and JavaScript, which can meet the basic needs of student information management. However, there is still room for improvement in terms of user experience, performance and functional extension, and the system can be made more perfect and efficient through further optimization and extension.**