CIS 3342 Project 2 – Home Builder

The goal of this assignment is to give you some experience creating web applications with a GridView control and using dynamic binding for dynamic data displays. This web application will allow users to build their dream home.

Requirements:

1. Create an ASPX page and design the user interface (input form) for the home builder page
   1. Add form controls for the user to enter their information name, address, and phone number.
   2. Put a **GridView** control on your form that will be used to display all the possible rooms for a home (master bedroom, bedrooms 2 – 5, bathrooms 1 – 4, den, basement, family room, living room, kitchen, dining room, great room, media room, loft, garage, etc…). This input GridView display will be dynamic with 5 columns. Each row in the GridView will correspond to a room that can be added to the house being built.
      1. First column display a checkbox to all the user to select this room to add to the house build.
      2. Second column displays the room description coming from the database.
      3. Third column displays the price per square foot for the room.
      4. Fourth column contains a textbox for the user to enter the length of the room.
      5. Fifth column contains a textbox for the user to enter the width of the room.
   3. Add **GridView** controls to add upgrades to the amenities in the house.
      1. Examples of some upgrades would be pre-installed sound system, pool, granite counters, tile, custom kitchen, etc…
      2. Each upgrade should contain either a fixed price or a price per square foot.
         * For example, a tile upgrade is based on the square footage of the room type that gets tile, but a pool is a fixed price.
      3. Provide a GridView to display a list of upgrades based on each room type.
         * For example, the application should have a GridView to display the upgrades for a kitchen, another GridView to display the upgrades for the bathrooms, etc…
         * Provide a way for the user to select as many upgrades as they want for each room type.
      4. The upgrades the user selects will add to the price of the house.
2. Use server-side input validation
   1. The name, address, and phone number textboxes cannot be blank.
   2. The user must select at least a bedroom, kitchen, living room, and bathroom before they can proceed.
3. Server-side processing
   1. Create a separate ClassLibrary project that contains the necessary classes used in this project. Name the library project **HomeBuilderLibrary**
   2. The first class **Home** represents a newly built home. It should contain 5 elements, as class properties, to be displayed in the output GridView (in Part 4). The 5 elements pertain to the house that was built.
      1. The class should contain an arraylist of Room objects. Where each Room has a description, price (based on chosen square footage and price per square feet), Upgrade cost, and TotalCost (price plus cost for the room’s upgrades).
      2. The class should contain methods to add and remove rooms and upgrades along with any other class members that may be necessary.
   3. Classes for a **Room** and **Upgrade** should contain elements and properties to store information regarding these.
   4. One class should contain functions that deal with processing the newly built house and any other operations that may be necessary:
      1. Note: this class must perform database operations, but it turns out that a Class Library cannot see the database DBConnect object created in the Codebehind of your ASPX page (outside the scope of the ClassLibrary). The best way to solve this problem is simply to create a new DBConnect object in the classes contained in the ClassLibrary that need it, or pass the database DBConnect object to the class functions through a parameter. The tradeoff for simplicity is using more server resources since each DBConnect object will require a new connection from a server connection pool.
4. Display order output
   1. Display the user’s name, address, phone number, etc….
   2. Display a dynamic GridView as Illustrated below. It needs to display the house that was built by displaying all the selected rooms and their individual information (description, total square footage, price, total cost). The description of the room should also contain the names of the upgrades. The total square footage is based on the user’s choice for the length and width of the room. The GridView must display the information using the arraylist of rooms from part 3b; this is what should be bound to this GridView in order to display the newly built house.
   3. “Total Cost” is calculated using the price of the room plus all the upgrades added to the room.
   4. The last row of the GridView should display the grand total for the house that was built. You need to implement this as a footer of the GridView.

Output GridView

|  |  |  |  |
| --- | --- | --- | --- |
| **Room Description** | **Price** | **Upgrades Cost** | **Total Cost** |
| Kitchen (12x12) with granite counters | $14,400 | $3000 | $17,400 |
| … | … |  |  |
| **Grand Total** |  |  | $100,000 |

1. Visibility control
   1. Control the visibility of both the input GridView and the output GridView displays. You want to show the input GridView when building a new house to order and hide the output Gridview. Then, you want to show the output GridView and hide the input GridView after the build was processed.
   2. Alternatively, you could use two separate pages, but this requires passing data between two ASPX pages, which we haven’t covered yet.
2. Good Design:
   1. Use server-side input validation where it’s necessary.
   2. Implement exception handling, so your programs don’t crash for any reason.
   3. Provide a consistent and logical navigation system. The user should never have to use the browser’s Back and Forward buttons to move between pages.
   4. Make your presentation clear to the user, providing on-screen instructions wherever needed both for data entry and error correction. If required data is omitted or entries are incorrect, the user should not have to re-enter data that is already correct.
   5. Create a good data model and implement the data model by creating the necessary tables in the database. You will be graded on the implementation of your data model.
   6. You need to use a proper naming convention for all controls and in your code. I expect you to properly name your classes, variables, functions, etc…
   7. **You cannot use the SQLDataSource control or any other controls to work with the database. You need to handle the code manually using ADO.NET programming.**
   8. **You must use component-based software design. This means writing as much code in classes and functions of classes instead of in the GUI.**

**Submission:**You need to publish your web application project to the cis-iis2 web server, upload your code to Blackboard, and provide the URL to your web application’s start page (Table of Contents page). Make sure a current version of your solution is located in your G:\cis3342 folder. Projects that are not submitted properly will not be graded.   
  
The grade for the required elements will be based on the published version of your web application. This means the elements must work from the published web application, not just the solution submitted through Blackboard. It’s important to make sure your published web application is the current version and that everything works. You shouldn’t publish to the Project folder after the project has been submitted. Otherwise, it will be considered late or may not be accepted if it’s after the deadline. If you make changes to your project, you will need to republish and resubmit the project. Generally, you shouldn’t publish or make changes until after you receive a grade.   
  
You need to zip the root folder for your solution into a single zip file and submit the assignment in Blackboard. To submit the assignment, you need to click the Assignment’s Title “Project 2” to view the submission form and upload the file.

**Make sure you properly submit your assignment and that it works. Programs that don’t run or don’t contain all the necessary files will not be graded and marked late.**

Please be sure to save your work periodically as you proceed and also back it up. You may want to store it on your flash drive. If you are going to zip an application in order to store it, BE SURE TO FIRST CLOSE Visual Studio. If you do store information on your flash drive be sure to copy it to a hard drive on your computer before working with the project.