Home Page Homework

Table of Contents

1. BEFORE YOU START .................................................................................................................................................. 2
2. CHOOSE A WEB TOPIC AND DATABASE TABLE DESIGN ......................................................................................... 3
3. REQUIREMENTS FOR YOUR WEB SITE FOR THIS HOMEWORK .................................................................................. 4
4. SUBMISSION ................................................................................................................................................................. 6
5. GRADING AND SAMPLE DEDUCTIONS ....................................................................................................................... 7
6. SUGGESTED APPROACH ............................................................................................................................................... 8

APPENDIX A: HOW TO DEBUG HTML .......................................................................................................................... 8
APPENDIX B: HOW TO DEBUG CSS .................................................................................................................................. 9

OVERVIEW: In this homework, you will come up with a topic for your web site, design and publish your home page, and submit a zip file of your web site into Canvas.
1. **Before You Start**

   - From the class website, follow the instructions on how to **set up your IDE** –
     - the NetBeans Bundle (not just NetBeans but the Bundle – make sure you have the correct versions of java and NetBeans),
     - a secure FTP program (e.g., WinSCP for PC or CyberDuck for Mac), and
     - a recent version of Chrome (we will be using the Chrome Developer Tools).

   **Email your instructor ASAP if you have ANY installation and/or setup problems.**

   - Follow the links from this homework (of the course web page) to learn about **Web Design**, listen to course videos, and study the **sample code**.
     - HTML/CSS introduction (and know where to find reference/help)
     - Page Layout (and Fixed Positioning, e.g., of titleNav and footer)
     - CSS Measurements (recommend use of % for widths and rem for everything else)
     - Flex box (e.g., for making divs display side by side),
     - Moving Styles to an External style sheet and Disabling Browser Cache so that:
       - You are not viewing an old version of your style sheet.
       - You are aware of any images files that are too large, making the page load slowly
     - Responsive Design (layout changes according to width of viewport) – for TitleNav, Images, and Content.
     - Background images (and Working With Images)

   - Learn how to use W3Schools.com to explain any concept you need and/or as a reference. The W3Schools“try it yourself” pages are awesome.

   - If you have web design experience, at least skim over the materials to be sure you are familiar with the information presented (and email me with suggestions if you have suggestions as to how I can better I’ve summarize things for web design newcomers).
2. **Choose a Web Topic and Database Table Design**

Every student shall build a web application by the end of the semester. Each week, students shall complete a homework that expands the functionality of their web application. Each web application shall access a database with tables: user_role, web_user, and another table that is unique to each student. By the end of the semester, your web application will allow the user to add/edit/delete data from the web_user table and from this other table (that each student designs themselves). To keep the work load manageable, the web applications shall be very simple. Students will receive sample code for add/edit/delete of the user table and they will write their own code for add/edit/delete of their other table.

Every student’s web_user table will have the same attributes as the sample code. That design is as follows:

- auto-increment primary key,
- email address (example of attribute required by the DBMS to be unique),
- password,
- membership_fee (example of $ amount data type),
- birthday (example of type date), and
- user_role (example of foreign key).

Each student shall select what they want to implement for their “other” table, but you need at least these attributes (don’t add too many attributes or you make more work for yourself):

- auto increment primary key,
- unique name or description,
- several other fields (you choose, but must include user enterable non-character fields like date, integer, or decimal which is good to represent $).
  - The date that data was inserted/updated does not count as a user enterable field since we would not ask the user to type this in (the system should know).
- Foreign key to user table.

For your “other” table, do not choose any type of “people” (customers, players, etc) because you can use the web_user table for that purpose. Try to fit your idea to the database restrictions, not the other way around. Try to keep things realistic but simple.

Some examples of database designs that would work for your web application project in this class:

- Users are chefs, other table is recipe (one chef can contribute many recipes)
- Users are travelers, other table is places visited (one traveler can visit many places).
- Users are salesmen, other table is products (one salesman can sell many products).
- Users (or their secretaries) are football players, the other table is highlights of their career.
- Users (or their secretaries) are musicians, the other table is songs or albums they have created.
- PLUS MANY MORE IDEAS ... Perhaps you are starting to see a pattern of what can be done with the “shape” of database design that you are being asked to implement.

So you are being asked to **decide on a topic** for your web application and decide (roughly) on an **idea for a database table** that supports that topic.
3. Requirements for your Web Site for this Homework

A. Your index.html shall:

1. include HTML elements with the following attributes: id="title", id="nav" (the navigation bar), id="content", and id="footer". You can have more elements, of course.
   I ask for specific id attributes to ensure that students don't just submit a page that they copied from the internet or from a web site they previously designed. It also facilitates grading and layout help.

2. have title text (that stands out visibly on the page) – as well as a <title> tag specified in the <head> section (which shows up in your browser tab heading).

3. have a navigation bar with two links in it (home and blog). For this homework, neither link will work...

4. The title and nav area shall be fixed, meaning that it will always be present on the page regardless of scrolling (and, of course, the top of the content area shall not be covered up by the titleNav area).

5. The titleNav shall be responsive meaning that in narrow widths the title and nav are on different rows but in wide widths, they are on the same row (modify the top content padding/margin to accommodate the change in height of the titleNav).

6. have a fixed footer area with your name in it (make sure the footer does not cover up the bottom of the content area, regardless of how much content, how tall the browser).

7. have a content area that looks professional and includes:
   - 1-2 paragraphs that describe the functionality you propose for your website and entice viewers to visit the site. In this description, include at least one external link (href= "http://...")
   - Description of the database table you propose. This table must support the above functionality. You could use an unordered list (<ul> <li>) to say what fields you plan to include in your database table...
   - a paragraph that describes your web design/development experience.
   - paragraph(s) that describe what you found easy, hard/confusing, and valuable about this HW.
   
   Note: for this week, your content may look silly (too many unrelated things), BUT in a future assignment, you’ll move all but the first 1-2 paragraphs out of the home page and into the blog page and then your web site will look more realistic.

8. include at least one image (either <img> tag or background image) somewhere on the page.
   - If you select a regular <img> tag, use a media query to display the image in multiple columns in desktop width, but single column in mobile width and the image should be “responsive” (use percentage width so the image grows/shrinks with the width of the viewport).
   - If you select a background image, that image shall have low contrast and all the text on top of it shall be easily readable. The background image shall look good in various browser widths and heights (probably by using background-attachment:fixed).

9. load quickly (by keeping your image file sizes around 500K or less). Either select images with small file sizes or reduce image file sizes as described in the "Working With Images" section of the Web Design Tutorial page of my website. To check for page load speed, empty your browser’s cache before rendering your page.
B. The **external style** sheet that’s referenced by your index.html shall:

1. implement a **professional looking color scheme** that is appropriate for the topic of your web site.
   - As suggested from my Web Design tutorial, select your layout colors from an image on your page (as opposed to selecting colors from some color palette). This approach makes your layout colors look more natural and blend more nicely with the image on the page. You can use the ColorZilla Chrome plugin to select colors from a web page. Google “Meyer Web Color Blender” to help generate darker/lighter or more/less saturated versions of your selected colors.
2. have a **fluid layout**, which means that your HTML elements nicely wrap when you narrow the browser (no horizontal scroll bar except when the viewport is super narrow, no wasted screen width even when the viewport is really wide).
   - To achieve this, avoid specifying widths, but when you must, use **percentage widths**.
   - Use box-sizing: border-box (counts padding and border in the size of the divs). If you are speciying the width of various elements to be displayed side by side, make sure that the % widths of all these elements plus left/right margin add up to around 100% (not > 100%).
   - **Prevent your title text from wrapping, by applying white-space: nowrap.**
   - Test your layout for fluidity by slowly narrowing your browser while ensuring unwanted wrapping or overlapping occurs. Your layout shall look good in all widths.
3. make all text (**including links**) **readable** (large enough, with enough contrast to every background).
4. provide **padding and/or margins** so that no text is too close to any visible border.
5. have **nav bar link text** that
   - are not underlined (CSS declaration "text-decoration:none").
   - are styled differently than the links in your content. Achieve this by using compound CSS selectors (also called combinator selectors). Example #nav a { ... } styles anchor tags (links) but only those that are inside of id="nav".
   - **Prevent your link title text from wrapping, by applying white-space: nowrap.** You will not notice a difference (with or without the nowrap) this week due to the simple nav bar, but next week, I it will really make a difference.
   - have colors that always stand out visually against their background: before clicking (a:link), after clicking (a:visited), and while hovering (a:hover). These rules selectors are called “pseudo-selectors” because they only style elements that are in a specific state (like unvisited or hover).
     Note: Compound (combinator) selectors and pseudo selectors are discussed in the [Intro to HTML and CSS](#) section of my Web Design tutorial or google it (and learn from W3Schools).

C. **Your home page homework shall NOT use Bootstrap.** Bootstrap is a great CSS framework, but it obfuscates many of the concepts we are trying to learn here. You may want to read a bit about Bootstrap from my Web Design page. You need to know what Bootstrap is, but you are not expected to know how to use it.

D. **All code files** (including index.html and your style sheet) shall:

1. be **syntactically correct** and properly **indented** (use NetBeans "Source – Format"). **Tip:** If you are viewing source of an HTML page in a browser, you can click on the style sheet (or any images) that are referenced in that code.
2. be neat and organized, with **no irrelevant code**.
   - In other words, everything in your style sheet must affect the look and feel of your index page - if you try something that has no effect, remove it. Since style is hierarchical (each element inherits style from its parent element), put your styles at the highest level possible (so you do not have to copy/paste the same style to a lot of low level children elements).
E. **Project Organization.** To assist us with grading, please put your *last name in your project name*. There is a NetBeans “rename project” option if you want to modify a project and it’s folder. To do so, right click on the project name and click that you want the folder also to be renamed.

As you start working on your home page, you may just have a simple index.html file (with internal style sheet) plus a “pics” folder with some images inside.

![Folder Structure Example](image)

After you move your styles to an external style sheet and incorporate the drop down JavaScript code, your NetBeans project shall be organized like this:

![Updated Folder Structure Example](image)

F. **Organization of your NetBeans Projects.**

- If you used NetBeans for a previous class, I recommend that you “close” most of those projects (will remove them from the NetBeans project pane), then move them into a different folder (than mydocs/NetBeans Projects).
- Create one project for the sample code for each homework. But, you’ll have just one project for your homework that grows in functionality each week. Back this up often (outside of NetBeans Projects).

4. **Submission**

When you have completed all the requirements, **tested locally** (and syntax checked), and **published** (and tested what you published), submit a zip file of your project folder into **Canvas** *(No RARs)*, along with your self assessment (list of deductions you think you’ll get – see next section). Your zip file submission is important because we base the late penalty off of the date/time of submission. Also, if there is ever a question about how we graded a homework, we will go by the code that was submitted, not by what is currently published. However, we do grade from what is published.
5. Grading and Sample Deductions (out of 10)

Grading will be based on timeliness, meeting the requirements, and effort/professionalism/originality.

-9 if no zip file submitted into Canvas (by the due date).
-9 if not published by the due date.
Up to -2 for lack of professionalism.
- Does the color scheme look natural looking and does it blend well with the image(s)?
- Does your page utilize "white space" effectively to emphasize important aspects of the page?
- Does your page have any text that is too close to visible edges?
- Is all text legible (size and contrast) - including link text before clicking, after clicking, and while hovering?
- Do your nav bar links look professional (not underlined)? Have you applied whitespace:nowrap?
Up to -3 for content area.
- Does your content explain what your web site has to offer, enticing users?
- Does your content include an external link that is styled differently than your nav links?
- Did you list the fields you propose for your "other" database table and does this database table support the functionality you described (also in the content area)?
- Did you mention your web development experience, if any?
- Did you describe what you found easy, hard (confusing), and/or valuable about this assignment?
Up to -1 for not having a fixed titleNav (and/or start of content covered up, no padding added to top of content).
- Up to -1 for not having a fixed footer (and/or end of content covered up, no padding added to bottom of content).
- -0.5 for not having your name in your footer (this helps us when grading).
Up to -2 for fluidity when we narrow the browser:
- Is there unwanted wrapping or overlapping of title or nav bar (or any other) elements?
- Do some links in the nav bar become hidden, requiring horizontal scroll bar to access?
- Does the page look lopsided and/or waste space on one side or the other?
Up to -2 for not implementing at least the beginnings of a responsive design using a Media Query.
- Does the titleNav change from single row to two rows when the browser narrows?
Up to -3 if no image used (as regular image or background image).
- Up to -1 if your page loads too slowly – try to select images with small file sizes or edit the images so that the total size of all image files on the page is around 900K or less. Test for this by clearing your browser’s cache (Chrome – More Tools – Clear Browsing Data), then reload the page.
- If using “regular” images in the content area, are the images single column in mobile width, but multi-column in desktop width (using Media Query)?
- If using a background image, is it low contrast so text is text legible? Do things look good when browser narrows?
Up to -3 for lack of originality (students with less HTML/CSS experience may not be able to be “super original”)
- Is your code too similar to sample code that is provided, or to any of your classmates?
- Was your code copied from the internet or from a site you previously designed? This is one reason I asked you to specifically name your ids (to prevent your copying a layout). The other reason is so that we can easily identify areas of your layout and assist you with layout issues.
When we “View Source”:
- -1 if you did not use an external style sheet.
- Up to -3 for HTML/CSS syntax errors which would show in red font from Firefox View Source.
- Up to -3 for poor coding style. Lack of code indentation, few or no comments, including unused/unneeded code.
- Up to -2 for missing/inadequate self assessment.
6. **Suggested Approach**

1. First **read this document**, so that when you **study the sample code and listen to the videos**, you’ll know “what you are looking for”. Don’t expect to find the answer to your homework already done in the sample code. Combine techniques from various parts of the sample code.

2. **Choose a topic** plus attributes for your database table.

3. **Select a look and feel** for your home page that is appropriate to the topic you selected. You can do this by first picking an image that relates to your topic and that includes colors you’d like for your color scheme. Select 2 or 3 colors from that image using a color picker browser plug-in (like ColorZilla). Use a color blender (like the Meyer color blending web site) to make variations of these colors (lighter/darker or more/less saturated).

4. **Begin the web design cycle** (edit, save, view, repeat), creating your home page (index.html). At first, I recommend that you use an internal style sheet (styles inside of `<style>` tags in the `<head>` section). Once you are pretty happy with how it looks, move the styles to an external style sheet (one of the homework samples shows how to do this).
   - Once you have an external style sheet, you may have to hold down the control key when you refresh the page – to avoid browser caching – or else you might not be seeing your latest CSS changes. OR right click from NetBeans and View or Run the page (opens up a new tab) instead of just refreshing the existing page. OR clear out your browser cache (Chrome: More Tools – Clear Browsing Data). OR tell Chrome not to cache when you have the F12 developer tools open (F12 – Network – “Disable Cache”).

5. **Follow submission instructions** (about publishing, testing, submitting, self assessment).

**Appendix A: How to Debug HTML**

**Debugging HTML**: Keep an eye out for the red bubbles that NetBeans places to the left of every line that has a syntax error. Nesting is just as important in HTML as is in java.

- Take advantage of NetBeans "Source – Format" (menu option) to automatically indent your code and help you understand the nesting within your HTML code. (Proper indentation is also a homework requirement.)
- Most HTML elements have a starting and ending tag, e.g., `<div id="content"> ... </div>`. If you click on the starting or ending tag of an element, NetBeans highlights the corresponding ending or starting tag. I suggest that you put a comment next to ending tags that are far away from their matching starting tag. Remember that HTML comments look like this  
  
```html
<!-- your comment -->
```
  
not `/*...*/` and not `//`
Appendix B: How to Debug CSS

CSS TERMINOLOGY (CSS RULE SET):

- One way to debug **debug HTML/CSS** is to put temporary borders on various HTML elements (like divs). This will let you determine whether space is coming from padding or margins. For example add this:
  ```
  border: thin solid red;
  ```
- OR you can click on F12 in Chrome to bring up the debugger and click on the "elements" tab. For any element that you select, Chrome will show you all the style rules that apply to that element.

Remember these things about CSS:
- Get rid of all red bubbles (from NetBeans) that identify CSS syntax errors.
- CSS comments can only use this form: `/* ... */`
- If you are having trouble getting a CSS declaration (rule) to take effect,
  - Move the declaration to the end of the rule. You might have specified the property twice (last one wins).
  - Also, if several rules are applied to an element, a declaration (say background-color) would be based on the most specific rule (for example, inline style in the html tag overrides styles from a style sheet).
  - Make sure your browser is referencing the latest files (instead of possibly cached files) by holding down the control key while refreshing or clear your browser's cache to see your latest changes.
- With CSS, less is better. Put your declarations at the highest level. For example, put your most used font-family in the body and only specify a different font-family for the areas that need something different.
- If your style sheet becomes cluttered, clean it up, organize it, and remove any unused rules (this is also a homework requirement).
- If the top or bottom of your content is covered up by your titleNav or footer, remember that with **fixed CSS positioning**, the fixed elements are "removed from the flow" which means all other elements act as if that fixed element never existed. Add enough top padding or margin to your content to accommodate the height of the titleNav. Add enough bottom padding or margin to your content to allow for the height of your footer. The z-index of titleNav and footer must be higher than that of content (default value is 0).
- Your layout should be "fluid", using percentages to specify widths (not pixel count). If this creates unwanted wrapping (when browser is at various widths), apply min-width to one of the elements like titleNav or nav, etc. As the browser narrows to that width (that you specified with min-width), it will provide a horizontal scroll bar and you will avoid the unwanted wrapping. It is fine to use pixel count for vertical measurements (margin, padding, font-size).