# Project 2 – Intro to AI-Assisted Programming

## PLEASE LIST YOUR NAMES

## (You and your project partners. Please note that you have new project partners!)

|  |
| --- |
| NoName McHiggens  |
| Ima Test  |
| Sam Student |
| Anne NotherWon |

## Expectations

* Instructor Guidance – High
* Independent Effort – **Moderate**
* Originality – Low
* Teamwork – **Moderate**

Be advised that every member of your project team should expect to receive the same grade on this project. That means that the mistakes of one student can bring down the grade of the whole team. Students are expected to help their project buddies and check each other’s work.

## Description

This project (Project 2) steps students through the process of creating a very simple web application that creates a amortization schedule for loan repayment. This calculator is applicable to common loan types such as mortgages, auto loans, student loans, or personal loans.

The purpose of the project is to introduce students to the features of the GitHub Co-Pilot Extension built into Visual Studio Code.

In this project, there is no web service component. That is to say: there is no “server side” code. This will be introduced in the next project.

## Instructions

1. Did you notice the “PLEASE LIST YOUR NAMES” box at the very beginning of this document? If you haven’t already done so, replace “NoName McHiggens” and the rest of the silly names with your name, and the names of your project buddies.

|  |
| --- |
| Students should plan on turning in ***one*** document to represent the work of the whole team. It would be wise to set this document up as a shared document so that you and your buddies can all edit it together, each student adding their own part.One of you will be “student 1”, the next will be “student 2”, and so on.I expect you to help each other. If you get stuck, reach out to your project buddies for help!  |

1. If you have not already done so, set up GitHub Co-Pilot. This is the AI “magic” that you will need to use this semester. It is essential that you do this work before we begin Project 2 in class.
	1. Set up your GitHub Account <https://youtu.be/aRoiZuaxtYE> (~ 6 minutes)
	2. Set up the VS Code extension <https://youtu.be/WyJqr0xgygc> (~ 6 minutes)

|  |
| --- |
| ***Be a team player*** – Remind your project buddies to do this! |

1. See the figures at the end of this document. They depict portions of the VS Code / GitHub CoPilot UI. You are responsible for knowing both the name and purpose of the visual elements indicated there.
2. In class (the week of 9/8) we will use GitHub Co-Pilot to create our Loan Repayment Calculator.

***Be sure that you are working in “Ask” mode.***

Students should all begin their work with the same initial prompt:

|  |
| --- |
| Create a web page that uses JavaScript, jQuery, HTML, CSS, and Bootstrap to implement a loan repayment calculator. Given a loan amount, a loan term (in years), and an annual percentage rate (APR), calculate a monthly payment, the total amount paid, and the total amount of interest paid. Here are some specifications:- Present the user with an HTML form and get the loan amount, the loan term (in years) and the annual percentage rate from the user.- Use this input to calculate the monthly payment, the total amount paid, and the total amount of interest paid.- Assume that payments are made on the loan reliably each month- Store the HTML, CSS, and JavaScript for the pate in separate files. Put CSS in a css folder. Put JavaScript in a js folder. Put the HTML in an index.html file.- Reference Bootstrap using the free, publicly available, most recent, most reliable CDN- Reference jQuery using the free, publicly available, most recent, most reliable CDN- Use jQuery to manipulate the DOM where needed- The bootstrap layout should be a 2 span column, 8 span column, 2 span column layout with the calculator in the middle- Do not use HTML buttons of type submit. Use JavaScript click events only. |

This prompt is an example of “Role Playing Prompting” in the sense that I am asking the AI to create content as if it was a Web Developer. See the appendix for different kinds of prompting.

**Alert!** You need to be aware that an AI can generate unexpected results. Modern AI models are **nondeterministic**, which is a fancy way of saying unpredictable. Students may get different responses to the prompt above, even though every student in the class is using the exact same prompt.

**It is important to note that** the specifications for the AI documented above are ***your*** specifications too. To make an extreme (absurd) example, if my prompt was to have the AI to write some JavaScript code, and the AI generated Python code instead, then the burden is on the student spot the difference and fix it (with or without more AI assistance!)

1. Provide a screenshot, showing your Visual Studio Code window. It should show your directory structure (folders and subfolders) as depicted in the “Explorer” panel, some of your code in the “Code Editor” panel, and the CoPilot “Chat” panel. To be clear, you should have a index.html file at the top of your folder hierarchy, and separate subfolders for CSS and JavaScript files.

**<< EACH STUDENT SHOULD COPY / PASTE A SCREEN SHOT HERE >>**

**<< STUDENT 1 >>**

**<< STUDENT 2 >>**

**<< STUDENT 3 >>**

**<< STUDENT 4 >>**

1. Check your work. Your work does not need to look exactly like this. The screenshots are provided here so that you have some numbers you can test your work against:

 

1. Upload your work to your AWS S3 bucket. You don’t need to create a new bucket. Just create a folder inside the bucket you already have!

|  |
| --- |
| I named my folder project2shafer and you follow that same convention … “project2” followed by your last name … with no dashes or spaces. **I EXPECT YOU TO FOLLOW THE SAME NAMING CONVENTION.** |

### Video material begins here

1. **(UI) User Interface Enhancement:** Our AI generated code will probably be pretty good right from the start. Part of our confidence comes from the quality of our initial prompt. We \*told\* the AI to use Bootstrap, and it did, so we get all the benefits of Bootstrap.

In this video we will make a slight change to our user interface. It is going to be important to limit our changes to CSS changes. We don’t want to change our HTML document’s structure, and we don’t want to change the logic of our code, we just want to improve its appearance a little bit.

Here is the video link: <https://youtu.be/H2MFLn5W1S0> (~ 30 mins)

Follow along with the video to do three things:

* 1. Change the background color of your web application
	2. Clean up any visual anomalies (you may or may not have any!)
	3. Make sure your “Calculate” button has the correct Bootstrap contextual class

It is advantageous to create applications that are: **Responsive**, **Cross-Platform /Browser Agnostic**, and **Mobile-Ready**.

Our UI should be implemented using **Single Page Architecture** (SPA), **Semantic Markup** & **Contextual Classes**.

*If you are not sure what those words mean, you should refer to the appendix at the end of this document*.

1. **(UX) User Experience Enhancement:** We want our final product to be both useful and easy to use.

Here is the video link: <https://youtu.be/0f_maZPzxyU> (~ 30 mins)

Follow along with the video to do these three things:

* 1. Make sure that calculations are performed using a loop. (This is necessary for the next step!)
	2. Improve your application’s utility by adding a table. Remember “utility” means: “is it useful?” In our case, it is more useful to see both the answer, and where it came from!
	3. Make navigation as intuitive as possible by showing and hiding visual elements as needed.
1. **(A11y) Check for A11y problems and remediate:** We want our final product to be accessible and compliant with accessibility regulations.

Here is the video link: <https://youtu.be/xY9kQnNSB08> (~ 30 mins)

You should use the A11y checker found at webaim.org. While I don’t require it, I strongly recommend setting up Firefox as your “testing” browser. Also, please note the following:

* 1. JavaScript alerts are not ok.
	2. Placeholder text is not ok.
1. **(Security) Check for security concerns:** We want our code to be secure!

Here is the video link: <https://youtu.be/vtbimEl-SaI>

Use GitHub CoPilot to evaluate your code for potential security concerns / vulnerabilities.

You must either remediate the issue or document why you did not remediate the issue. It is understood that different student solutions will have different issues. Please synthesize all the issues encountered by your group into one spreadsheet. Turn the spreadsheet in along with this document. Only one spreadsheet per group.

**The spreadsheet should have five columns that document the following:**

 **Issue Number:** (Just a number: 1 ,2, 3, etc.)

 **Students Affected:** (List who on your team had this vulnerability. It is ok to say “All” where appropriate.)

 **Description:** (Concise text describing the security issue.)

 **Remediation:** (Describe what was done to address the problem. If “none” or “N/A” explain why you chose to not remediate the issue.)

1. Each student should provide the URL for their work. It is OK for there to be some subtle variations in presentation, wording, and aesthetic.

However, defects in functionality, utility, accessibility and security are not ok. A defect (in functionality, utility, accessibility and/or security) in one student’s solution negatively affects the grade of every student on the team!

So, check your own work. Check the work of your project buddies. Help each other out.

Student 1: Put your client-side code URL here:

|  |
| --- |
|  |

Student 2: Put your client-side code URL here:

|  |
| --- |
|  |

Student 3: Put your client-side code URL here:

|  |
| --- |
|  |

Student 4: Put your client-side code URL here:

|  |
| --- |
|  |

1. Turn in your work. On canvas, turn in this completed Word document (one per team) and the Excel Spreadsheet requested on step 11.

## How will this project be graded?

Be advised that every member of the team will receive the same grade and that one student’s error can bring down the grade of the whole team. Students should help each other and check each other’s work.

Your solution must use the latest Bootstrap for layout, and the latest jQuery for document model manipulation. Student work that does not adhere to these requirements will be regarded as late, with no opportunity for resubmission or revision.

Point deductions are in 5-point increments.

|  |  |
| --- | --- |
| Item | Points |
| ***All*** your names | 10 |
| ***All*** your screenshots  | 10 |
| ***All*** URLs provided  | 10 |
| ***Basic Functionality - All calculators work*** | 10 |
| ***UX/UI Improvements -*** *All calculators generate an Amortization table, navigation is good, dollar values are properly formatted* | 25 |
| ***Accessibility -*** will be checked using WebAim, do not use JavaScript Alerts, do not use HTML placeholders. | 15 |
| ***Security -*** *You must either remediate, or justify your exceptions* | 20 |
| **TOTAL** | 100 |

Please see the syllabus for the late policy.

# Appendix

Students are responsible for knowing the material here.

## Prompting Strategies (Shafer’s short list)

1. **Zero-Shot Prompting**: In zero-shot prompting, the AI model is given a task description without any examples. The AI is expected to understand and perform the task based solely on the instruction. This is useful when you want the AI to generalize from its training data to new, unseen tasks.
2. **One-Shot Prompting:** One-shot prompting is a technique in AI where the model is given a single example to understand the pattern or task it needs to perform. This is useful when you want the AI to follow a specific format or style based on a single reference.
3. **Few-Shot Prompting**: Unlike one-shot, few-shot prompting provides multiple examples to the model. This can help the AI better understand the pattern or task, especially for more complex instructions. For example, you might provide three or four examples of a task to guide the AI's responses.
4. **Chain-of-Thought Prompting**: This technique involves breaking down a complex task into a series of simpler steps, each of which is prompted individually. This can help the AI handle multi-step reasoning and problem-solving tasks more effectively.
5. **Role-Playing Prompting:** In this approach, you assign a specific role or character to the AI, and it generates responses as if it were that character. This can be useful for creative writing, dialogue generation, or simulating interactions with specific personas.

### The composition of a web application client

|  |  |
| --- | --- |
|  | * These three client-side technologies are stored in separate files.
* They work together to create the client-side application.
* When working with Copilot, it is important to keep all three of these files in context.
 |

### Visual Elements: VS Code + GitHub Copilot



Apply Insert at Cursor Copy

|  |  |
| --- | --- |
| Feature | Purpose |
| Apply | Make the AI provided recommendation to the file.  |
| Insert at Cursor | Insert the recommendation at the current cursor position. |
| Copy | Copy the recommendation so it can be pasted elsewhere. |

|  |  |
| --- | --- |
| A screenshot of a computer  AI-generated content may be incorrect. | Add ContextSet Mode (Ask, Agent, Edit)Pick Model (Use GPT-4.1 or a newer GPT model) |

|  |  |
| --- | --- |
| Feature | Purpose |
| Add Context | Add the files that the AI will use to compose its reply. These will typically be the files that you might possibly want to edit in response to an AI CoPilot prompt. For a simple web application, this will be the CSS file, the JavaScript file, and the HTML file. |
| Set Mode | There are multiple possible values for Mode. For this course we will always use “Ask” mode as it allows you the student to see what the CoPilot AI is doing, and how it is doing it. |
| Pick Model | The “model” here refers to the LLM (Large Langauge Model) that will be used to process your prompt and recommend changes. We will use GPT-4.1 in this course, but that is not the only option. |



Keep Undo Toggle Diff Editor



## User Interface Best Practices (Shafer’s Short List)

1. **Responsive**: Responsive design in web development is the practice of building websites that automatically adjust their layout, images, and functionality to look good and work properly across a wide range of devices and screen sizes (desktops, tablets, and smartphones).
2. **Cross-Platform / Browser Agnostic:** These words do mean subtly different things, but they are closely related. A Cross Platform application runs on multiple hardware platforms (Mac, PC, Android Tablet, etc.). A Browser Agnostic application is a ***web*** application that is not limited to one or two specific types of browsers. If you can expect a web application to run on Chrome, Edge, and Firefox equally well then it is considered to be Browser Agnostic. And, if it is Browser Agnostic, then it is almost always Cross Platform (because those browsers run on so many different platforms!)
3. **Mobile-ready**:In web development, “mobile-ready” means that a website is designed to function properly on smartphones and other small-screen devices. A mobile-ready site or application ensures that text is readable without zooming, navigation is easy to tap with fingers, and content loads quickly on mobile networks.
4. **Single Page Architecture:** Single Page Architecture (SPA) is a web application design approach where the entire app runs within a single HTML page. Content and the visibility of elements updates dynamically without requiring full page reloads.
5. **Semantic Markup:**Semantic markup means writing HTML that conveys the meaning and structure of content, not just how it looks.

### Bootstrap Contextual Classes

See this resource <https://www.w3schools.com/bootstrap5/bootstrap_buttons.php>