Cash Register

Kevin Chancey and Kiran Limbu

IS312 Web Design, School of Technology and Computing, City University of Seattle

[boojiboy@cityuniversity.edu](mailto:boojiboy@cityuniversity.edu), [limbukiran@cityuniversity.edu](mailto:limbukiran@cityuniversity.edu)

**Abstract**

Our team will work and collaborate throughout the course to construct a fully operational cash register application with a full interactive user interface. We will build this using HTML, CSS, Javascript, and possibly more based on our needs. The aim is to use JavaScript to manipulate HTML elements making user interaction possible. We will go into depth about the steps we take and our thought process in overcoming the challenges that we will face. The team will collaborate to create, debug, and update our cash register application to be easy to use and functional.

**Keywords:** HTML, CSS, Typescript, JavaScript, DOM elements, Tags, Objects, Events, Functions

1. **INTRODUCTION**

When someone thinks of the world of computer programming, most of the time, they think of lonely people in a dark room typing away at a keyboard with no human contact. In fact, the coding world is nothing like that. Computer programming is a very social career and personal skills are searched for by many companies that are hiring. Paired and team programming also comes with many benefits for the individuals and for the companies themselves.

Teamwork and collaboration are extremely important to be successful in computer programming. All the major companies such as Google, Facebook, Amazon, etc. focus on teams to get daily task accomplished and keep their sites running. Even small companies use teams to meet the software development demands. When programmers work together, they all can benefit as the more senior level programmers gain experience by teaching others while the ones being taught can gain great knowledge from the people that have been doing it every day. Even if 2 programmers are equal in experience, they can both still teach each other new things as no one has the same experiences. Therefore, universities and coding bootcamps heavily focus on paired and team programming. This prepares students and future engineers for the technology world they decided to get into.

This paper and team project will get more into depth on how we overcame technical problems, maintained virtual communication as a team, created working functions, made a visually pleasing user interface, debugged our way through issues, and came together as a team to produce a working cash register application. We will discuss how we were able to create this functional application using only HTML, CSS, Javascirpt, and Typescript. Our paper will show how we were able to create a schedule, follow a structured guideline, separate responsibilities, share our code with each other, and maintain communication throughout the course to produce a successful application.

**2. PREPARATION OF GUIDELINES & SCHEDULE**

In the early stages of formation of our team, we created a team room over Microsoft Teams and decided that would be our main hub for all group communication. At the same time, we started a shared Google Drive document and slideshow for collaborative writing, viewing, and for the final presentation. This allowed us to work on the team project while maintaining our own life schedules.

We also schedule an initial chat for introductions and to discuss our goals, deadlines, schedule, and what we want our final product to look like. A document was created to plan out our project separated in 3 main steps: To Do List, Creating Modules, and Adding it all together.

The To Do List, in order, consisted of adding event handlers, getting input values, adding the new items to data structure, adding the new items to the user interface, calculating the total, then updating the user interface.

Creating Modules represented the Interface Module, the Data Module, and the Controller Module. Our plan was to now add each To Do item to the corresponding module.

The meeting ended with our schedule created, plans laid out, goals established, and our forms of communication formed. The next step was to get started. Figure 0 shows our agreed upon goad for what our cash register should look like in the end.



Figure 0. Image of generic cash register application

**3.** **THE** **CODING PROCESS**

The plan discussed in the first team meeting was to create a new project using Visual Studio Code. We agreed on keeping the organization and structure simple. An index.html, script.js, and style.css files were created as our starting point.

Our first major step was to code a working calculator since the functions are the same as a cash register. We needed to allow the user to click on each number and the user interface should show them the number they clicked on. We also needed the ability for the program to know when the user clicked on an operation button such as the plus sign or minus sign. Finally, we needed to calculator to take the user’s input and execute the problems submitted into it. For example, if the user inputs 5 + 5 then hits the equals button, we needed the calculator to save everything the user inputted and calculate the output of that math problem.

The first step we went with was to create the user interface to create a simple calculator image with div tags as buttons separated by rows. As seen in Figure 1, each div has its own id and class for easy styling and Javascript manipulation.

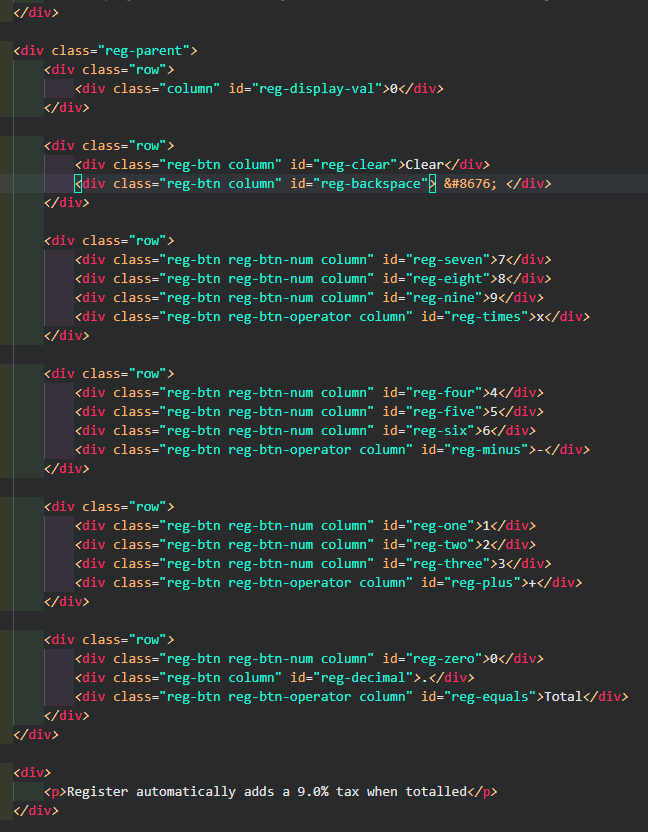


Figure 1. Screenshot of Visual Studio HTML file

Figure 2 show what our project looked like when it was originally created. Styling was added to show different colors of each button and operations, button color changing when the mouse hovered over, and basic clean design.

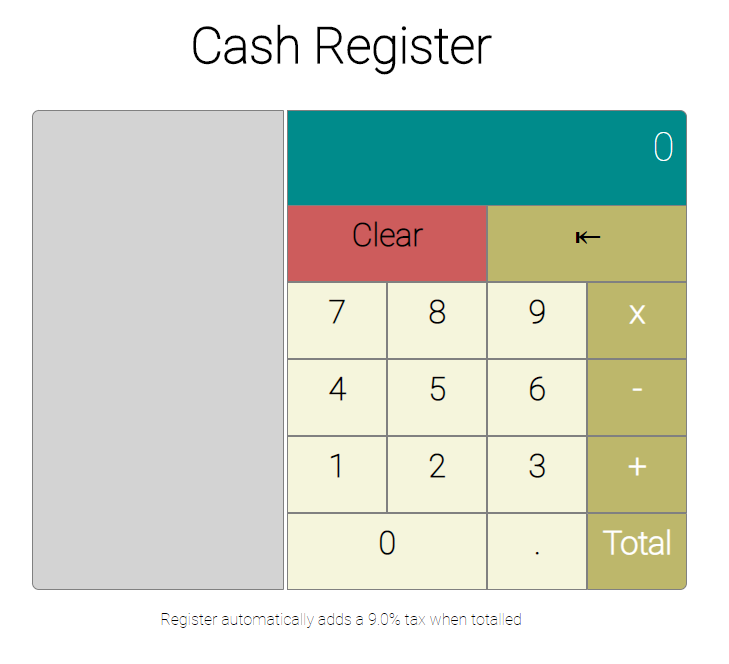


Figure2. Screenshot of user interface html with added CSS styling

The next step was to start adding the Javascript to the project. In the external Javascript file, we added the event listener and grabbed all the buttons by their id’s creating variables with each one. We also added event listeners for each number button and each operator button. The display (the dark blue section in Figure 2) was also created as a variable set to zero. Next, we created an empty array to later hold the operations being performed by the user.

The first main function created was the ‘updating the display’ function. The purpose of this is to update the display as the user clicks on different buttons. This function works by taking the inner text from the button the user clicks and makes that text what shows in the display. If there are no buttons clicked, the display is defaulted to zero.

The next function was the operations function which had the purpose of taking in the operator buttons such as plus, minus, multiply, and equals. Depending on the operator selected, the function will push it into the array created earlier to be held until the equals button is pressed. When the equals button is finally pressed, everything in the array gets joined and calculated. At the same time, for simplicity, we have the equal function add an automatic tax to the final output. That calculation is then added to the display to show the user their answer.

Three more functions were added for the decimal button, the backspace button, and the clear button. The clear button wipes out all the information in the function array that holds any user data entered. It also defaults the display back to zero. The backspace function only clears the last number added to the array; if there is nothing left in the array after the backspace is pressed, the display is defaulted to zero. The decimal function allows the user to only add 1 decimal per input. It first checks if the display has any decimals before allowing one to be added.

After this Javascript was added to the HTML and CSS, we now have a working calculator that could do simple math problems. The next step was to turn this calculator into a working cash register application.

Add what you did next with the Javascript / CSS

**4.** **TEAM MEETINGS**

About team communication throughout,

**5. TEAM PAPER & PRESENTATION**

About writing the paper and creating the presentation.

**6. OVERCOMING ISSUES**

Overcoming our issues, communication, debugging, etc.

**7. SHARING CODE AND COMBINING**

Sharing our code and putting both together.

**8. FOOTNOTES**

Footnotes if we have any.

**9.** **REFERENCES**

References used for this project.

**Appendices and Annexures**

Typically, an appendix or annexure will include actual handouts given in class, or questionnaires used for surveys, or **other presentation graphics and tables** that do not fit readily into the body of the paper but are still important for clarity or completeness. Often these items were prepared for other purposes than presentation as a part of your paper, and it is acceptable that they do not adhere to font face, font size, and column requirements observed elsewhere in your paper.

**In appendices, you are allowed to have margins that are larger but not smaller than the generally required margins, which are** 1.0 top margin, 1.0 bottom margin, 1.0 left margin, and 1.0 right margin

So long as you adhere to the margin requirements, appendices may be presented in any format that you desire.

Rotated pages (landscape orientation) are not allowed in the body of the document, but they are permitted in the appendices. This will allow the entire page to be rotated anti-clockwise 90 degrees for final typesetting.