S11 Group #27 Members:

De Guzman, Gabriele

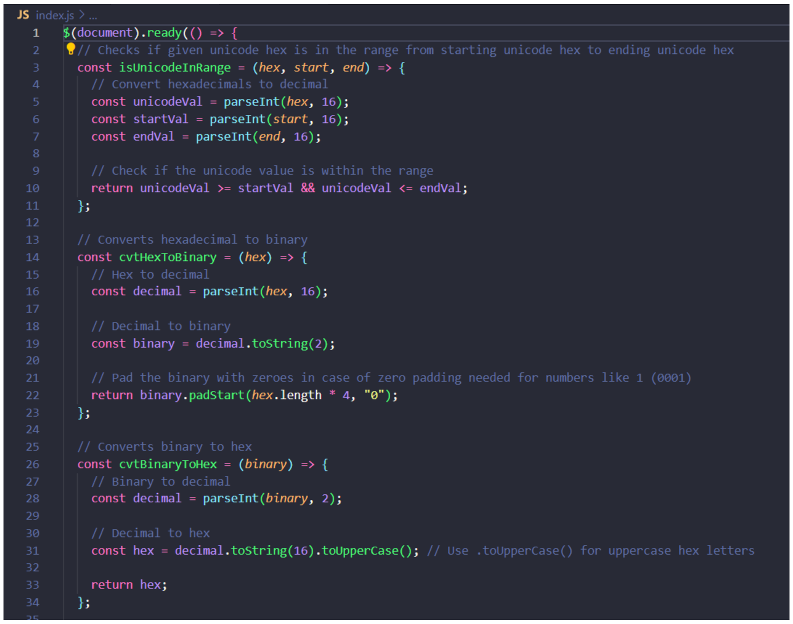
Dulin, Andrea

Javier, Darryl

CSARCH2 Simulation Project Documentation – Unicode to UTF Converter

In this simulation project, we were tasked to create a web-based application with a graphical user interface that can convert Unicode to UTF. In creating the website, we made use of HTML, CSS and JavaScript wherein JavaScript was responsible for the functionality while HTML and CSS were for the GUI. In our JavaScript file, we coded different functions such as the ff: the website will be able to check if the given Unicode is in hex, converting hexadecimal to binary, converting decimal to binary, converting binary to hex, converting binary to decimal, converting decimal to hexadecimal, and so on. Moreover, we made sure that our application is straightforward, simple and easy to use, that's why after the user has input a Unicode in HEX value, the results will be displayed immediately after clicking a button.

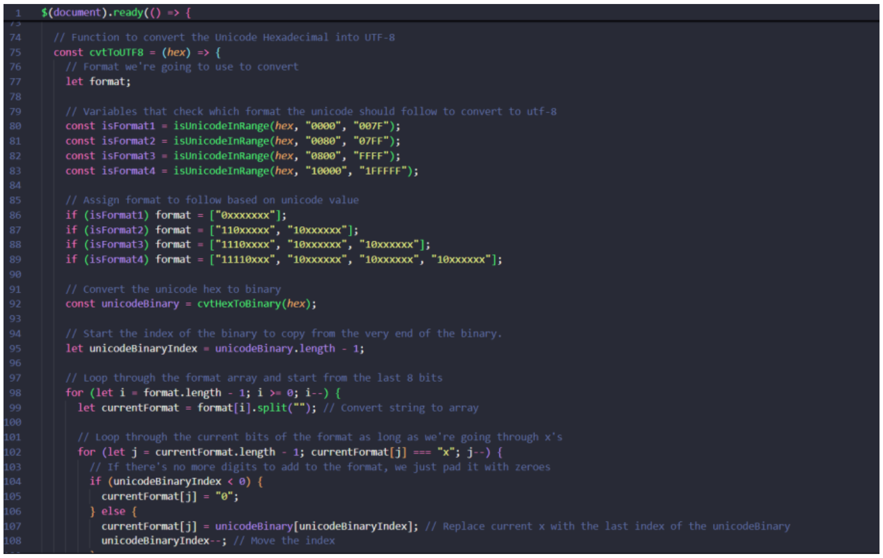
We created the functionality of the website like how we learned the conversions from Hexadecimal to UTF Formats within our course learnings. We have implemented multiple functions for conversions (aside from main conversions) such as converting hexadecimal to binary and vice versa, binary to decimal and vice versa, and many more functions that are used in the functions that serve as main conversions.



```javascript
$(document).ready(() => {
  // Checks if given unicode hex is in the range from starting unicode hex to ending unicode hex
  const isUnicodeInRange = (hex, start, end) => {
    // Convert hexadecimals to decimal
    const unicodeVal = parseInt(hex, 16);
    const startVal = parseInt(start, 16);
    const endVal = parseInt(end, 16);

    // Check if the unicode value is within the range
    return unicodeVal >= startVal && unicodeVal <= endVal;
  };

  // Converts hexadecimal to binary
  const cvtHexToBinary = (hex) => {
    // Hex to decimal
    const decimal = parseInt(hex, 16);

    // Decimal to binary
    const binary = decimal.toString(2);

    // Pad the binary with zeroes in case of zero padding needed for numbers like 1 (0001)
    return binary.padStart(hex.length * 4, "0");
  };

  // Converts binary to hex
  const cvtBinaryToHex = (binary) => {
    // Binary to decimal
    const decimal = parseInt(binary, 2);

    // Decimal to hex
    const hex = decimal.toString(16).toUpperCase(); // Use .toUpperCase() for uppercase hex letters

    return hex;
  };
```

Figure 1 shows a code snippet for the input checking and conversion

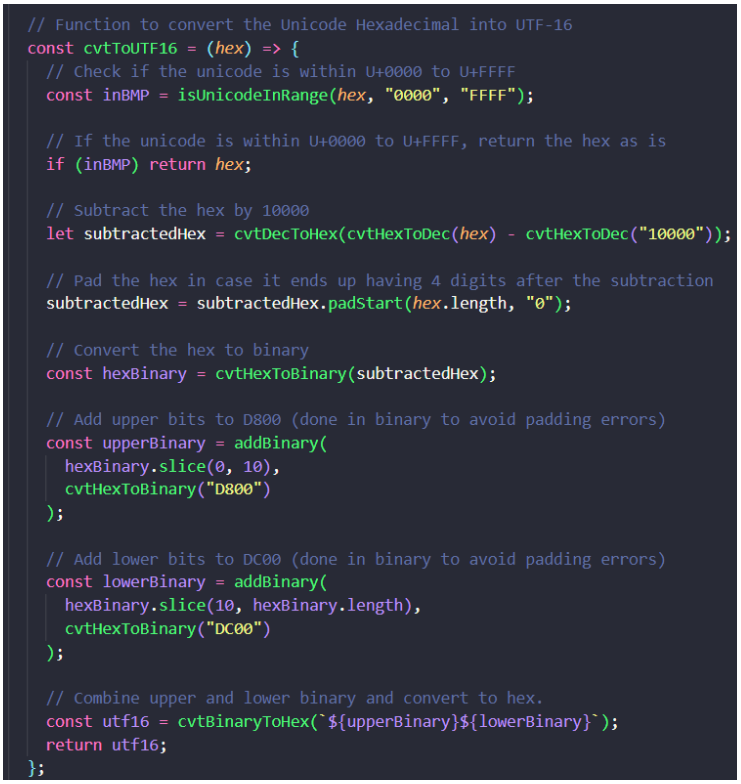
These main conversion functions are the cvtToUTF8 (Converting to UTF-8) function, the cvtToUTF16 (Converting to UTF-16) function, and the cvtToUTF32 (Converting to UTF-32) function. The conversion from hexadecimal to UTF-8 is rather straightforward, just like how we learned it within the course. There are multiple formats to follow based on the input hexadecimal, and based on which range it belongs to. These formats are represented as string arrays that contain “x” that will soon be replaced with binary digits. Then, this array will be joined together as a single binary string, and will be converted back to hexadecimal, which will serve as the UTF-8 format of the hexadecimal input.



```
$(document).ready(() => {
  // Function to convert the Unicode Hexadecimal into UTF-8
  const cvtToUTF8 = (hex) => {
    // Format we're going to use to convert
    let format;

    // Variables that check which format the unicode should follow to convert to utf-8
    const isFormat1 = isUnicodeInRange(hex, "0000", "007F");
    const isFormat2 = isUnicodeInRange(hex, "0080", "07FF");
    const isFormat3 = isUnicodeInRange(hex, "0800", "FFFF");
    const isFormat4 = isUnicodeInRange(hex, "10000", "1FFFFF");

    // Assign format to follow based on unicode value
    if (isFormat1) format = ["0xxxxxxx"];
    if (isFormat2) format = ["110xxxxx", "10xxxxxx"];
    if (isFormat3) format = ["1110xxxx", "10xxxxxx", "10xxxxxx"];
    if (isFormat4) format = ["11110xxx", "10xxxxxx", "10xxxxxx", "10xxxxxx"];

    // Convert the unicode hex to binary
    const unicodeBinary = cvtHexToBinary(hex);

    // Start the index of the binary to copy from the very end of the binary.
    let unicodeBinaryIndex = unicodeBinary.length - 1;

    // Loop through the format array and start from the last 8 bits
    for (let i = format.length - 1; i >= 0; i--) {
      let currentFormat = format[i].split(""); // Convert string to array

      // Loop through the current bits of the format as long as we're going through x's
      for (let j = currentFormat.length - 1; currentFormat[j] === "x"; j--) {
        // If there's no more digits to add to the format, we just pad it with zeroes
        if (unicodeBinaryIndex < 0) {
          currentFormat[j] = "0";
        } else {
          currentFormat[j] = unicodeBinary[unicodeBinaryIndex]; // Replace current x with the last index of the unicodeBinary
          unicodeBinaryIndex--; // Move the index
```

Figure 2 shows a code snippet for the UTF-8 conversion

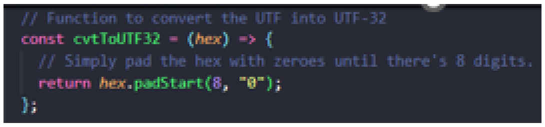
For UTF-16, we subtracted the hex value to 10,000 and convert it to binary. Afterwards, we added the upper bits to D800, while the lower bits were added to DC00. Then, we combined the result of the upper and lower bits. This function is a bit more simpler than the UTF-8 conversion function.



```
// Function to convert the Unicode Hexadecimal into UTF-16
const cvtToUTF16 = (hex) => {
  // Check if the unicode is within U+0000 to U+FFFF
  const inBMP = isUnicodeInRange(hex, "0000", "FFFF");

  // If the unicode is within U+0000 to U+FFFF, return the hex as is
  if (inBMP) return hex;

  // Subtract the hex by 10000
  let subtractedHex = cvtDecToHex(cvtHexToDec(hex) - cvtHexToDec("10000"));

  // Pad the hex in case it ends up having 4 digits after the subtraction
  subtractedHex = subtractedHex.padStart(hex.length, "0");

  // Convert the hex to binary
  const hexBinary = cvtHexToBinary(subtractedHex);

  // Add upper bits to D800 (done in binary to avoid padding errors)
  const upperBinary = addBinary(
    hexBinary.slice(0, 10),
    cvtHexToBinary("D800")
  );

  // Add lower bits to DC00 (done in binary to avoid padding errors)
  const lowerBinary = addBinary(
    hexBinary.slice(10, hexBinary.length),
    cvtHexToBinary("DC00")
  );

  // Combine upper and lower binary and convert to hex.
  const utf16 = cvtBinaryToHex(`${upperBinary}${lowerBinary}`);
  return utf16;
};
```

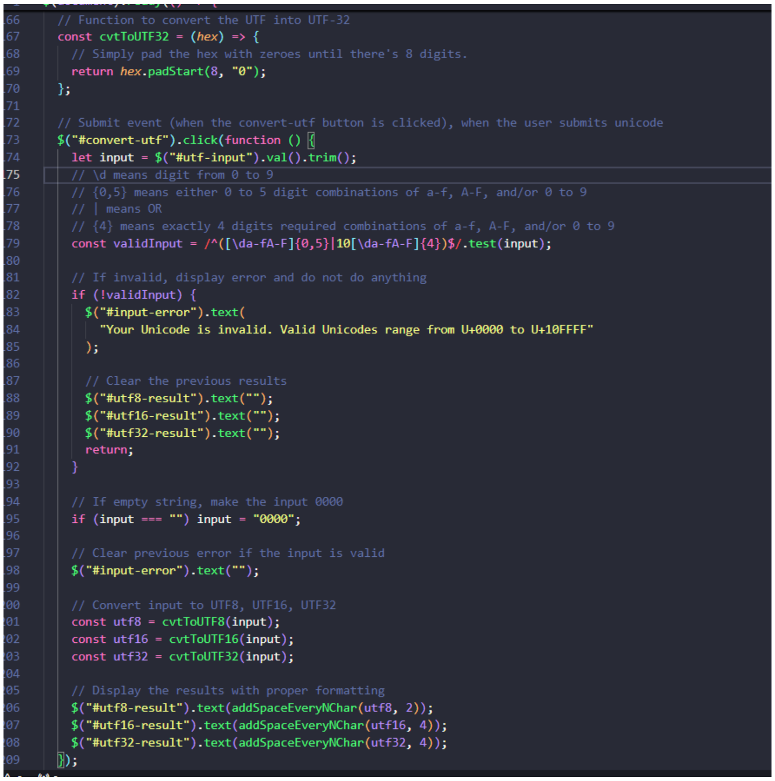
Figure 3 shows a code snippet for the UTF-16 conversion

And lastly, below is our conversion to UTF-32. We simply just padded the hexadecimal value to up to 8 bits with zeroes.



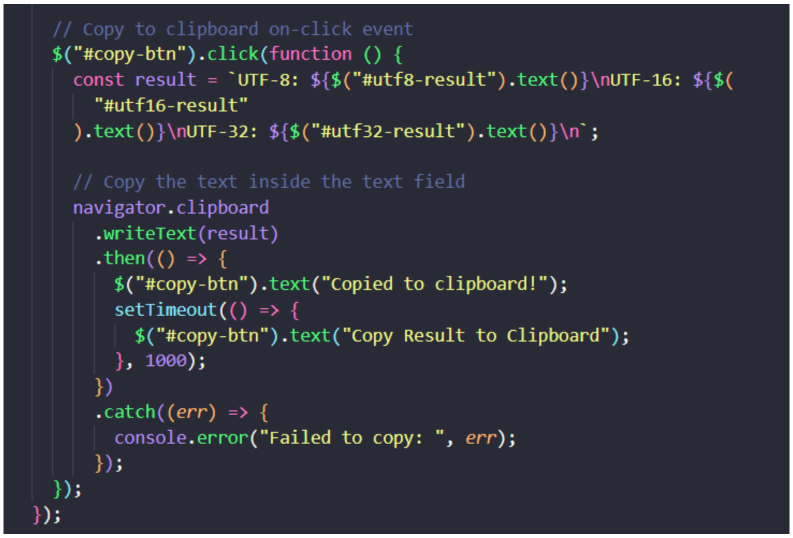
```
// Function to convert the UTF into UTF-32
const cvtToUTF32 = (hex) => {
  // Simply pad the hex with zeroes until there's 8 digits.
  return hex.padStart(8, "0");
};
```

We also implemented a function that will allow the user to copy the result to the clipboard so that they may paste the output whenever they want to do so. We also implemented an error message on the text field when the user inputs an invalid input such as inputs that exceed valid hexadecimal ranges, inputs with special and invalid characters, and inputs that are not hexadecimal. The result is also properly formatted by creating a space between each 4 bits for UTF-16 and UTF-32 conversions, and for UTF-8 conversions, we added a space every 2 bits. Screenshots of the different test inputs were also included below.



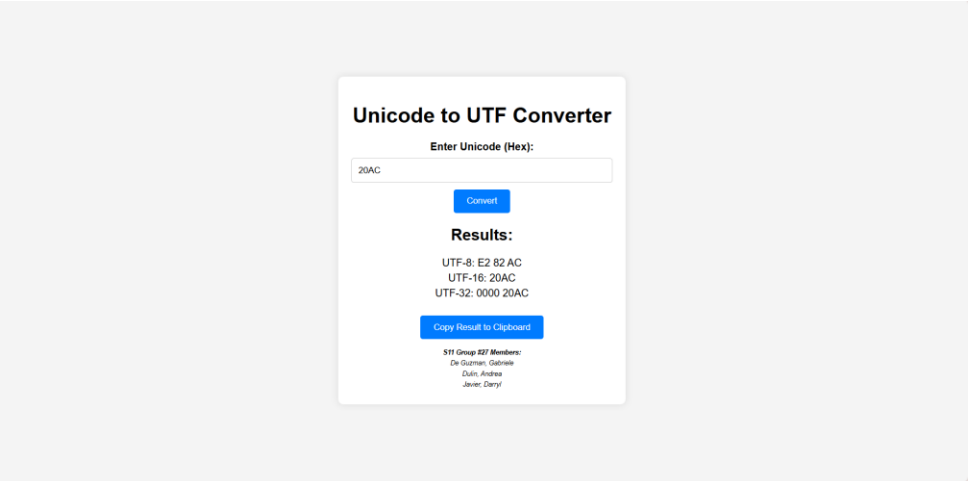
```
// Function to convert the UTF into UTF-32
const cvtToUTF32 = (hex) => {
  // Simply pad the hex with zeroes until there's 8 digits.
  return hex.padStart(8, "0");
};

// Submit event (when the convert-utf button is clicked), when the user submits unicode
$("#convert-utf").click(function () {
  let input = $("#utf-input").val().trim();
  // \d means digit from 0 to 9
  // {0,5} means either 0 to 5 digit combinations of a-f, A-F, and/or 0 to 9
  // | means OR
  // {4} means exactly 4 digits required combinations of a-f, A-F, and/or 0 to 9
  const validInput = /^([\da-fA-F]{0,5}|10[\da-fA-F]{4})$/.test(input);

  // If invalid, display error and do not do anything
  if (!validInput) {
    $("#input-error").text(
      "Your Unicode is invalid. Valid Unicodes range from U+0000 to U+10FFFF"
    );

    // Clear the previous results
    $("#utf8-result").text("");
    $("#utf16-result").text("");
    $("#utf32-result").text("");
    return;
  }

  // If empty string, make the input 0000
  if (input === "") input = "0000";

  // Clear previous error if the input is valid
  $("#input-error").text("");

  // Convert input to UTF8, UTF16, UTF32
  const utf8 = cvtToUTF8(input);
  const utf16 = cvtToUTF16(input);
  const utf32 = cvtToUTF32(input);

  // Display the results with proper formatting
  $("#utf8-result").text(addSpaceEveryNChar(utf8, 2));
  $("#utf16-result").text(addSpaceEveryNChar(utf16, 4));
  $("#utf32-result").text(addSpaceEveryNChar(utf32, 4));
});
```

Figure 4 shows a code snippet for the UTF-32 conversion



```
  // Copy to clipboard on-click event
  $("#copy-btn").click(function () {
    const result = `UTF-8: ${$("#utf8-result").text()}\nUTF-16: ${$(
      "#utf16-result"
    ).text()}\nUTF-32: ${$("#utf32-result").text()}\n`;

    // Copy the text inside the text field
    navigator.clipboard
      .writeText(result)
      .then(() => {
        $("#copy-btn").text("Copied to clipboard!");
        setTimeout(() => {
          $("#copy-btn").text("Copy Result to Clipboard");
        }, 1000);
      })
      .catch((err) => {
        console.error("Failed to copy: ", err);
      });
  });
});
```

Figure 5 shows a code snippet for copying to clipboard



Unicode to UTF Converter

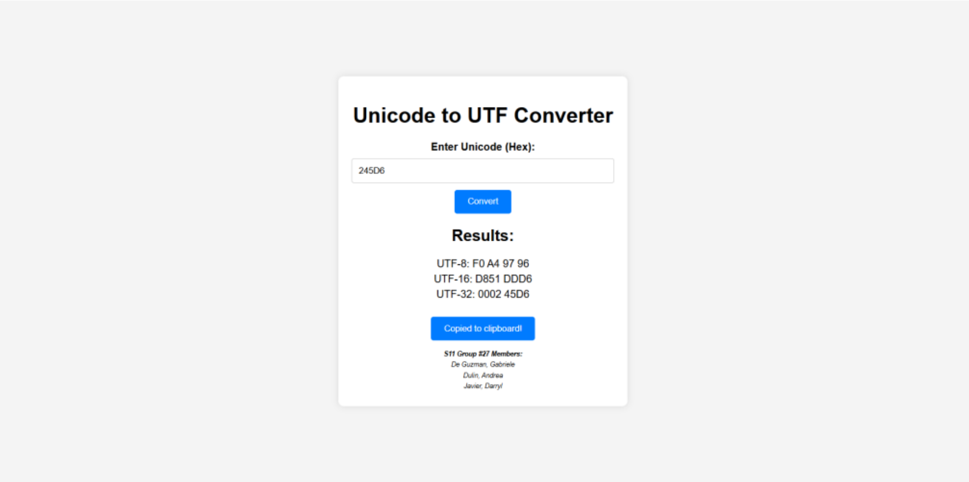
Enter Unicode (Hex): 20AC

Results:

UTF-8: E2 82 AC
UTF-16: 20AC
UTF-32: 0000 20AC

S11 Group #27 Members:
De Guzman, Gabriele
Dulin, Andrea
Javier, Darryl

Figure 6 shows a test case with a normal case Unicode

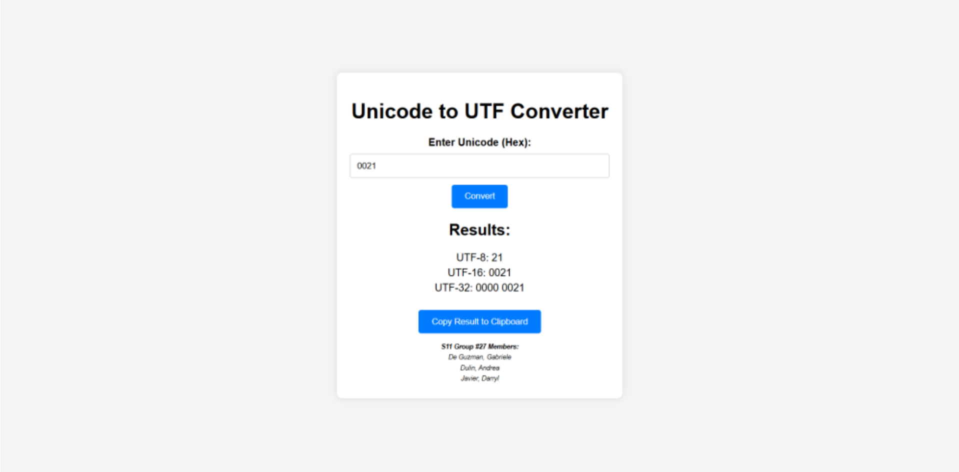


Results:

UTF-8: F0 A4 97 96
UTF-16: D851 DDD6
UTF-32: 0002 45D6

S11 Group #27 Members:
De Guzman, Gabriele
Dulin, Andrea
Javier, Darryl

Figure 7 shows that we can copy the output to the clipboard

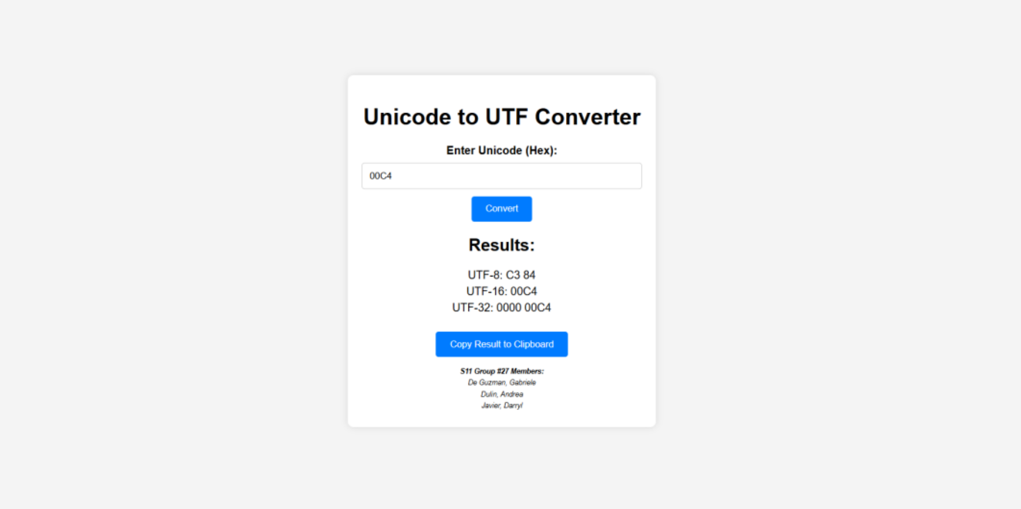


Results:

UTF-8: 21
UTF-16: 0021
UTF-32: 0000 0021

S11 Group #27 Members:
De Guzman, Gabriele
Dulin, Andrea
Javier, Darryl

Figure 8 shows the results of the special character “!” in UTF-8, UTF-16 AND UTF-32



Unicode to UTF Converter

Enter Unicode (Hex):

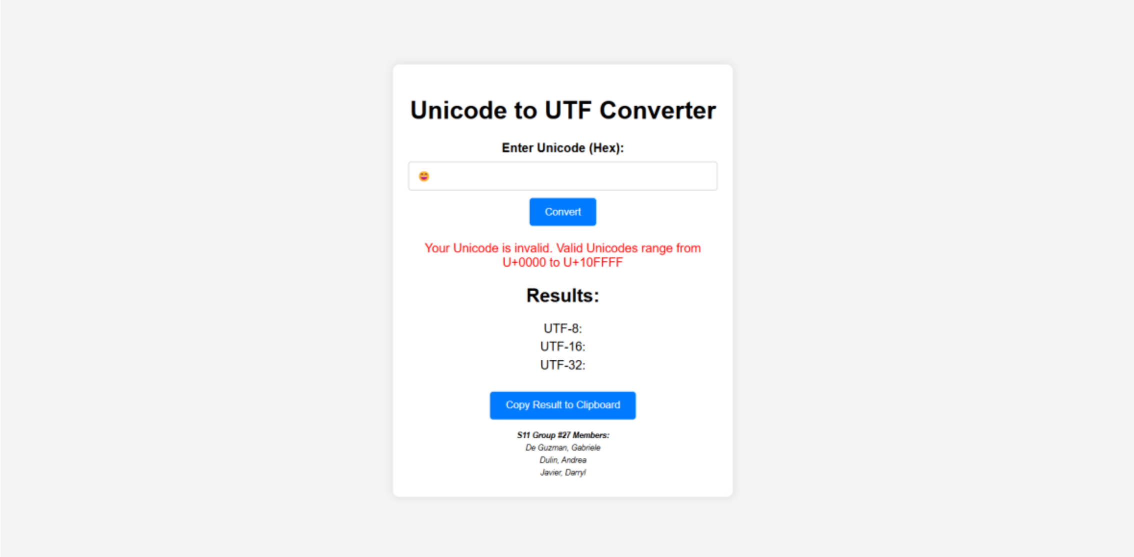
00C4

Results:

UTF-8: C3 84
UTF-16: 00C4
UTF-32: 0000 00C4

S11 Group #27 Members:
De Guzman, Gabriele
Dulin, Andrea
Javier, Darryl

Figure 9 shows the results of the LATIN CAPITAL LETTER A WITH DIAERESIS in UTF-8, UTF-16 AND UTF-32



Unicode to UTF Converter

Enter Unicode (Hex):

😄

Your Unicode is invalid. Valid Unicodes range from U+0000 to U+10FFFF

Results:

UTF-8:
UTF-16:
UTF-32:

S11 Group #27 Members:
De Guzman, Gabriele
Dulin, Andrea
Javier, Darryl

Figure 10 shows the error message if the input is not in hex format

We have tested the website with various inputs, and they have been proven to be accurate according to other online conversion tools. This website is deployed in GitHub pages, and the GitHub repository is also publicly available.

GitHub Repository: Darealtube/CSARCH2-Unicode-Simulation (github.com)
GitHub Page Link: UTF Converter (darealtube.github.io)