Criterion C: Development

Word count: 666

The product is a web browser application and its purpose is to improve the handwriting of people who have problems with handwriting. The score of the handwriting depends on the accuracy of the written input, which is performed through touchscreen capabilities. The software also gives out exercises with a blank space where the client can use to input figures that are supposed to match the shape of the given letter or number.

List of techniques:

- HTML5 Canvas API
- Pointer and Touch Event Handling
- Dynamic Drawing and Stroke Rendering
- Image Data Analysis
- Text Measurement and Bounding Box Calculation
- Randomized Word Selection
- Data Capture using Data URLs
- External Font Integration and CSS Styling

The reason HTML5 Canvas API technique was used in my software was to provide an efficient method of rendering graphics. It plays a key role in registering the hardware input, which is evidenced by the canvas of the software giving clients/users of the application the ability to write on it.

```
const fontSize = "150";
const fontFamily = "'Satisfy'";
const letterSpacing = 50;

function writeWord(word) {
    ctx.clearRect(0, 0, canvas.width, canvas.height);

    // Background
    ctx.fillStyle = "#f0f0f0";
    ctx.fillRect(0, 0, canvas.width, canvas.height);

    ctx.font = `${fontSize}px ${fontFamily}`;
    ctx.textBaseline = "middle";
    ctx.textAlign = "left";

    // Measure total word width with custom spacing
    let totalWidth = 0;
    for (let i = 0; i < word.length; i++) {
        totalWidth += ctx.measureText(word[i]).width + letterSpacing;
    }

    const startX = (canvas.width - totalWidth + letterSpacing) / 2;
    const centerY = canvas.height / 2;

    let x = startX;

    for (let i = 0; i < word.length; i++) {
        const letter = word[i];
```

The Point and Touch Event Handling technique provides a secure way for the application to function on both desktop and mobile devices. The idea of it being compatible across both types of devices stems from Criterion A's planning phase (the text before the bulleted list of criteria) where me and my cousin were still planning it all out. For point handling, I utilized pointerdown, pointermove, and pointerup. All of these 3 ensure smoothness in interaction, which erases any potential delays from interaction with the application.

```
// Stop Drawing
canvas.addEventListener("pointerup", function() {
    drawing = false;
    handwritingData = canvas.toDataURL(); // Save the handwriting data
});

// Prevent accidental scrolling on touch devices
canvas.addEventListener("touchstart", (e) => e.preventDefault(), { passive: false });
canvas.addEventListener("touchmove", (e) => e.preventDefault(), { passive: false });
```

The Dynamic Drawing and Stroke Rendering technique allows users to see themselves writing/drawing the shape of the letter given in the canvas in real-time. I used stroke styles like ctx.strokeStyle and ctx.strokelineWidth because I wanted to enhance the visibility of the handwriting input.

```
// Outline
ctx.strokeStyle = "red";
ctx.lineWidth = 20;
ctx.strokeText(letter, x, centerY);

// Fill
ctx.fillStyle = "white";
ctx.fillText(letter, x, centerY);

// Advance cursor
x += ctx.measureText(letter).width + letterSpacing;
```

The Image Data Analysis technique was used to evaluate the handwriting accuracy of a client when filling the shape of the characters of the presented word. If the user writes only one character or two but not the full word, it will give back the message "Please write all letters first!", which prompts the user to write the rest of the characters if they want feedback from the statistics feature of the software. All of this is done by getImageData(). This function determines which areas of the shapes have handwriting in them. It also helps for identification of incorrect strokes.

```
function getFillRatio(ctx, box) {
    const imageData = ctx.getImageData(box.x, box.y, box.width, box.height);
    const data = imageData.data;
    let emptyPixelsCount = 0;
    let filledPixelsCount = 0;
```

The Randomized Word Selection technique was used to present the client of the software with words which they can exercise on writing. And because the app is intended for kids to improve their handwriting, simple words like “hug” or “cat” were given for the user(s) of the application to practice writing. This is done by making a const (constant) to define the variable (all the words given for practice). Once submitted, the board will reset and randomly choose one of the given words from the constant.

```
const words = [
    "cat", "dog", "sun", "hat", "bat",
    "run", "bug", "hug", "car", "bus",
    "red", "hop", "top", "box",
    "fox", "cup", "bed", "pen", "zip"
];

document.fonts.ready.then(() => {
    resetBoard();
});
```

The Data Capture using Data URLs technique was used to collect bits of handwriting of the user and keep them in a place where the user can see them (in the statistics section of the application). This is an efficient strategy because it allows users to track records of letters they’ve got wrong (saves progress basically) and how with time, the overall score (in percent) becomes higher or lower, depending on the type of handwriting used.

```
// Stop Drawing
canvas.addEventListener("pointerup", function() {
    drawing = false;
    handwritingData = canvas.toDataURL(); // Save the handwriting data
});
```

The External Font and CSS styling technique’s used to help users navigate with the software more easily. For font, I used “Satisfy” because it has a visually appealing design to me with all of the letters’ designing. Now, by applying the latter, CSS styling, it allowed for

organization of elements which would enhance the experience of the user software.

```
</head>
<body>
    <h1 style="font-family: Satisfy">Handwriting Improvement</h1>
    <p>Write a letter in the canvas below using your device's touchscreen.</p>
    <canvas id="handwritingCanvas" width="400" height="400"></canvas>
    <br>
    <button id="clearButton">Clear</button>
    <button id="submitButton">Submit</button>
    <div id="stats">
        <h2>Statistics</h2>
        <p id="incorrectStats">No data yet.</p>
    </div>
```

Overall, all of these techniques helped me to produce the handwriting application I was striving for in the end. Techniques like External Font and CSS styles deals with the visually appealing design of it, while techniques like Dynamic Drawing and Stroke Rendering deal more with the application's functionality behind the scenes.