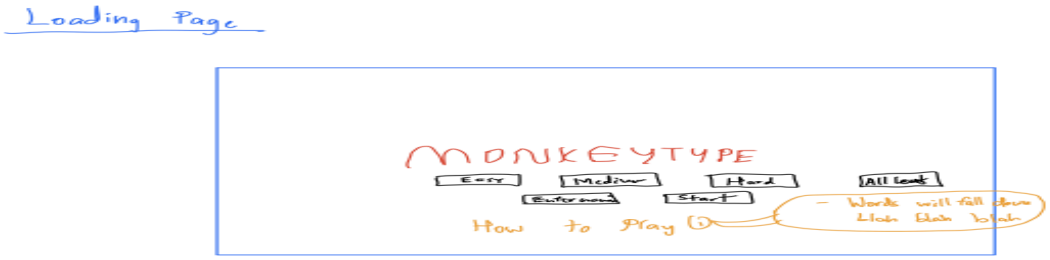
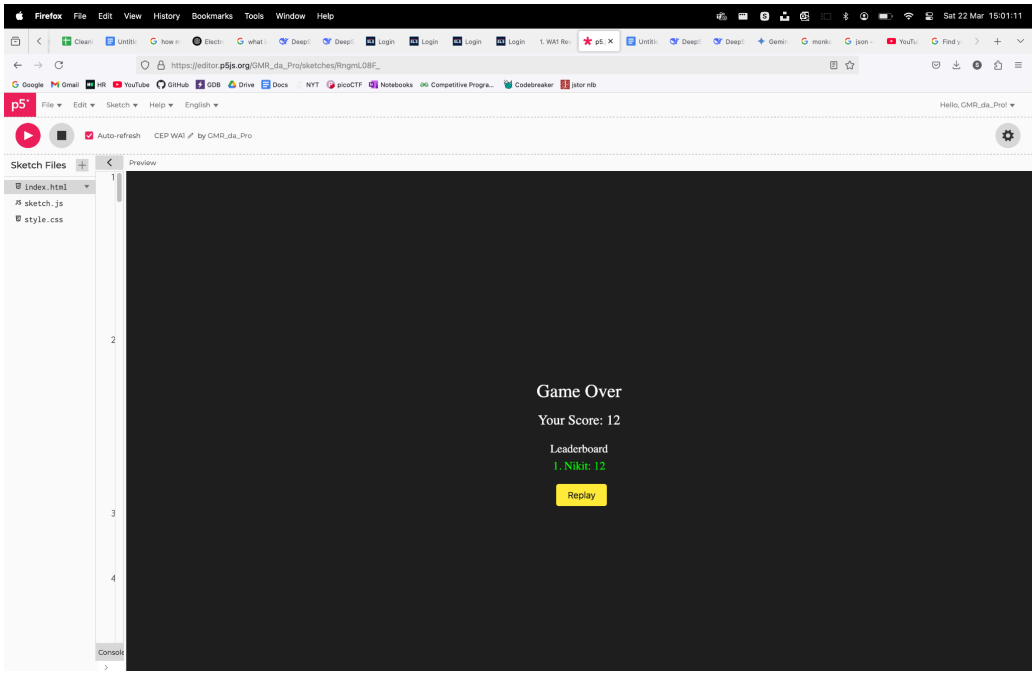
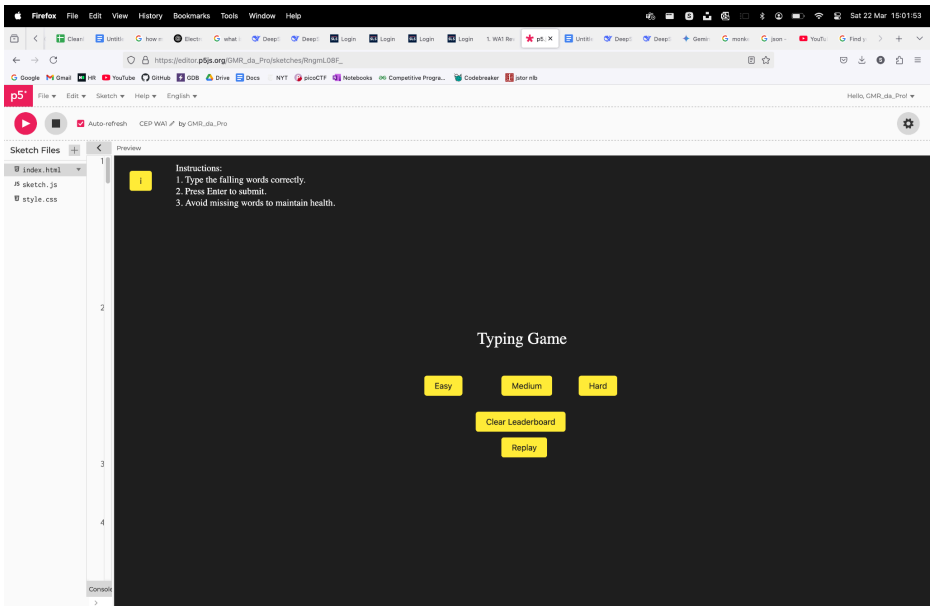
Documentation

Game: https://editor.p5js.org/GMR_da_Pro/full/RngmL08F_

Sketches:







I wanted to take a screenshot of the strobe, but it was too fast for me to take a picture of

How the game is meant to function:

This is a typing game which is designed to test their skill and accuracy in typing. Upon launching the game, players will be greeted by a loading screen, which features an informational button providing an overview of the gameplay mechanics and objectives. Players are then to key in their own names before selecting a difficulty level, each of which modifies 2 key parameters:

1. Word Complexity - The vocabulary set is derived from Monkeytype's categorised word difficulty system (English, English 5k, English 10k), ensuring that each level is progressively harder as the words become longer and more tedious to type out.
2. Word Descent Speed - The velocity at which the words fall from the top of the screen varies based on the chosen difficulty, with the harder levels featuring a more steeper base speed to begin with.

The primary objective of the game is for the user to type out the falling word in their entirety before they cross the bottom threshold of the screen. Each correctly typed character contributes to the player's score. Failure to entirely type out the word before it reaches the bottom results in a loss of half a heart. Each player has 5 hearts, and once all the hearts are depleted, the game is over. To aid the users, individual letters illuminate either green or red when typed correctly/incorrectly and if a letter is displayed as red, the player will be unable to type any further, and has to delete his letter and correctly type it out. To introduce an additional level of difficulty, random events (strobes) may trigger where the words will start to accelerate, testing the player's reaction.

Upon depletion of all 5 hearts, players will be directed to the "Game Over" screen, where their final score is computed based on the number of correct letters they were able to type out and WPM. Based on their final score, players will be able to receive a competitive ranking compared to the other players who have played this game before and are able to gauge their skill level (take note that each leaderboard is for the

individual difficulty level). Additionally, a replay button is also present for users to be able to replay the game under the same difficulty and username.

Additional things that were added/Cool Features:

Some elements I decided to add were as such:

1. The overall design choice. It is black and white to make it more professional. The font used is also Times New Roman and the size of the font all help to convey professionalism.
2. The jittering hearts. They were inspired by Minecraft and were also used to show an understanding of Physics and is a quick throwback to the lesson where we learnt how to use physics to utilise it.
3. Deciding to implement the code in HTML as I decided to use rounded buttons (inspired from apple's design) to support the overall aesthetic and this would only be the case if i were to combine both the .js and .css file into a html file. *(used chatgpt for this to help in overall transition)*
4. 3 JSON leaderboards which are able to store the progress of the scoreboard, and update your individual one to be green!
5. I used some quick math to devise a formula to calculate the score which would be the total number of letter * WPM / 100, to standardise the scoring system and make it more fair and accurate

References:

1. **https://github.com/MarkosComK/TyperSpace.git - Basic reference from a vanilla.js project found on github**
2. **https://github.com/monkeytypegame/monkeytype.git - Monkeytype github code, mainly referred to the themes section to get the theme (e.g. #hex…)**
3. **Chatgpt Prompts:** *"let t words = []; let inputWord = ''; let health = 100; let score = 0; let gameOver = false; let fallSpeed = 2; // Default speed let wordSpeedIncrease = 0.05; // Default increase let typingAllowed = false; let gameStarted = false; // Flag to check if the game has started let leaderboard = []; // Array to store leaderboard scores let playerName = ""; // Store the player's name let nameInput; // Input for player name let startTime; // Track the time when the game starts let wordCount = 0; // Track the number of words typed let wordTypedTime = 0; // Track the time when the last word was typed for WPM update // Word list const wordList = ["ability", "able", "about", "above", "accept", "according", "account", "across", "act", "action", "activity", "actually", "add", "address", "administration", "admit", "adult", "affect", "after", "again", "against"]; // Heart emojis const hearts = ["❤️", "❤️", "❤️", "❤️", "❤️"]; function setup() { createCanvas(600, 400); textSize(24); textAlign(CENTER, CENTER); // Use modern font textFont('Roboto'); // Set up background and font styles document.body.style.backgroundColor = '#121212'; // Dark background fill(255); // Create name input nameInput = createInput("Enter your name"); nameInput.style('background-color', '#333'); nameInput.style('color', '#fff'); nameInput.style('border', '2px solid #FFEB3B'); nameInput.position(width / 2 - 100, height / 2 - 50); // Create difficulty buttons with modern design createButton('Easy') .style('background-color', '#FFEB3B') .style('color', '#212121') .position(width / 2 - 100, height / 2 - 20) .mousePressed(() => startGame(2, 0.05)); createButton('Medium') .style('background-color', '#FFEB3B') .style('color', '#212121') .position(width / 2, height / 2 - 20) .mousePressed(() => startGame(3, 0.07)); createButton('Hard') .style('background-color', '#FFEB3B') .style('color', '#212121') .position(width / 2 + 100, height / 2 - 20) .mousePressed(() => startGame(4, 0.1)); // Load leaderboard from localStorage loadLeaderboard(); } function draw() { background(33, 33, 33); // Dark background if (!gameStarted) { // Display loading screen with a minimal style fill(255); textSize(32); text("Typing Game', width / 2, height / 2 - 80); return; // Don't run the game loop yet } if (gameOver) { // Display Game Over screen with sleek design fill(255); textSize(32); text("Game Over', width / 2, height / 2 - 50); // Display final score and WPM textSize(24); text("Your Score: ' + score, width / 2, height / 2); let minutes = (millis() - startTime) / 60000; let wpm = Math.floor(wordCount / minutes); text("WPM: ' + wpm, 50, height - 30); // Display WPM at bottom left // Display leaderboard displayLeaderboard(); // Replay button createButton('Replay') .style('background-color', '#FFEB3B') .style('color', '#121212') .position(width / 2 - 50, height / 2 + 100) .mousePressed(restartGame); return; } // Display health and score displayHealth(); fill(255); textSize(16); text('Score: ' + score, width - 50, 30); // Display WPM while the game is running let minutes = (millis() - startTime) / 60000; let wpm = Math.floor(wordCount / minutes); text('WPM: ' + wpm, 50, height - 30); // Game logic for falling words for (let i = words.length - 1; i >= 0; i--) { words[i].fall(); words[i].show(); // Remove word if it goes out of bounds if (words[i].y > height) { if (words[i].index < words[i].value.length) { loseHealth(); } words.splice(i, 1); if (health <= 0) { gameOver = true; saveScore(); // Save score when game is over } } } fill(255); // Text color for remaining instructions textSize(16); text('Type this: ' + inputWord, width / 2, 60); } function keyPressed() { if (!typingAllowed || gameOver) return; let currentWord = words[words.length - 1]; if (currentWord) { let currentLetter = currentWord.value[currentWord.index]; // Typing correct letter if (key.toLowerCase() === currentLetter.toLowerCase()) { inputWord += currentLetter; currentWord.index++; score++; } else if (keyCode !== BACKSPACE) { currentWord.typedWrong = true; } } // When the user presses the Enter key, register the word and clear input if (keyCode === ENTER) { let currentWord = words[words.length - 1]; // Only register the word if the input matches the falling word if (inputWord.toLowerCase() === currentWord.value.toLowerCase()) { words.pop(); wordCount++; // Increment word count when the word is typed correctly wordTypedTime = millis(); // Update time after a word is typed } // Clear the input field after pressing Enter, regardless of correctness inputWord = ''; } } function mousePressed() { typingAllowed = true; } function Word(value) { this.value = value; this.x = random(50, width - 50 - this.value.length * 20); this.y = -50; this.index = 0; this.typedWrong = false; this.fading = false; this.fall = function() { this.y += fallSpeed; }; this.show = function() { for (let i = 0; i < this.value.length; i++) { if (i < this.index) { fill(0, 255, 0); // Green for correct letters } else if (i === this.index && this.typedWrong) { fill(255, 0, 0); // Red for incorrect typed letter } else { fill(255); // Default color for remaining letters } text(this.value[i], this.x + i * 20, this.y); } if (this.index === this.value.length || this.y > height) { this.fading = true; } if (this.fading) { fill(0, 0, 0, map(this.y, height, height + 50, 255, 0)); text(this.value, this.x, this.y); } }; function spawnWord() { let newWord = random(wordList); words.push(new Word(newWord)); } function increaseFallSpeed() { fallSpeed += 0.001; } setInterval(() => { if (!gameOver && gameStarted) { spawnWord(); } }, 3000); function startGame(speed, increase) { gameStarted = true; fallSpeed = speed; wordSpeedIncrease = increase; startTime = millis(); wordCount = 0; nameInput.remove(); select.All('button').forEach(button => button.remove()); } function saveScore() { playerName = nameInput.value(); leaderboard.push({ name: playerName, score: score }); leaderboard.sort((a, b) => b.score - a.score); leaderboard = leaderboard.slice(0, 5); localStorage.setItem('leaderboard', JSON.stringify(leaderboard)); } function loadLeaderboard() { const storedLeaderboard = localStorage.getItem('leaderboard'); if (storedLeaderboard) { leaderboard = JSON.parse(storedLeaderboard); } } function displayLeaderboard() { fill(255); textSize(20); text("Leaderboard", width / 2, height / 2 + 50); for (let i = 0; i < leaderboard.length; i++) { let entry = leaderboard[i]; let textColor = (entry.name === playerName && entry.score === score) ? color(0, 255, 0) : color(255); fill(textColor); text(${i + 1}. ${entry.name}: ${entry.score}, width / 2, height / 2 + 80 + i * 25); } } function displayHealth() { let fullHearts = Math.floor(health / 20); let halfHearts = Math.ceil((health % 20) / 10); let jitter = sin(frameCount * 0.1) * 2; for (let i = 0; i < fullHearts; i++) { text(hearts[i], 30 + i * 40 + jitter, 30 + jitter); } if (halfHearts === 1) { text("💔", 30 + fullHearts * 40 + jitter, 30 + jitter); } } function loseHealth() { health -= 10; if (health < 0) health = 0; } function restartGame() { gameOver = false; score = 0; health = 100; words = []; gameStarted = false; inputWord = ''; setup(); } whats the syntax error with this code" " ok problem is when im typing one word and another word falls down from the top, my typing gets messed up, find a way which doesn't change my typing and only checks when I press enter use a complex algorithm for this" "unction spawnWord() { let wordsToSpawn = Math.floor(fallSpeed / 2); for (let i = 0; i < wordsToSpawn; i++) { let newWord = random(wordList); words.push(new Word(newWord)); } } this function is wrong -> the words.push(new Word()) no such thing as Word "" add a wpm function in this game whereby the WPM is showcased at the bottom left of the screen which changes as u type"*
4. **My own source code from last year:** *et typed = ""; let answers = ["Laughing with friends makes days brighter", "Cycling through parks brings fresh air", "Walking along beaches soothes the soul", "Reading adventure novels sparks imagination"]; let ref = ""; i = 0; state = 0; // 0 -> landing, 1 -> game let btn, btn2; let startime, endtime ; let totaltyped = 0; function setup() { createCanvas(windowWidth, windowHeight); textFont("Roboto Mono"); btn = createButton("Let's do it"); btn.position(width / 2 - 45, height / 2 + 40); btn.style("color : blue"); btn.mousePressed(toBegin); // wishful thinking btn2 = createButton("replay!") btn2.position(width/6*3, height/4) btn2.mousePressed(replay) btn2.hide(); } function draw() { textFont("Roboto Mono"); background("#3A3B3C"); if (state == 0) { textSize(30); textAlign(CENTER); text("SpeedScript: The ultimate typing challenge", width / 2, height / 2); textSize(18) text("Remember to make it upper case and follow the spaces", width/2,height/4*3.5) text("Click replay after you finish if you enjoyed the game!", width/2, height/4*3.6) text("Hope you get a good score!!",width/2,height/4*3.7) textAlign(); fill("turquoise"); // nameField = createInput("Enter your name:"); //nameField.position(50, 50); //progress bar sigma } else if (state == 1) { textSize(30); textAlign(CENTER); text(typed, width / 2, height / 2); fill("turquoise"); text(ref, width / 2, height / 2 - 30); fill("yellow"); //progress bar fill(255); rect(0, height - 50, width, 50); fill("rgb(0,255,0)"); rect(0, height - 50, map(typed.length,0,ref.length,0,width), 50); //end of progress bar //for colour fill("yellow") //accuracy // let accuracy = 100 //totaltyped=0; accuracy = round((typed.length/totaltyped * 100),1); // totaltyped = 0; textAlign(LEFT); text("ACCURACY = " + accuracy + "%", 50, 50); if (typed == ref) { noLoop(); background("#3A3B3C"); fill("rgb(0,255,0)"); textAlign(); text("YAY, you have finished the game :) ", width / 4, height / 2); textAlign(); btn2.show() endtime = millis(); let timetaken =round((endtime - startime)/1000,1); fill("600ub00") text("Time taken: "+ timetaken + " seconds", width/4, height/4*2.5); text.Align(); let WPM = (39/5)/(timetaken/60); text("WPM: " + round(WPM,2), width/4, height/4*3) text("Please replay if you have enjoyed", width/4, height/4*3.5) textAlign(); btn2.show(); } } function keyTyped() { if (ref[i] == key) { typed += key; i++; } totaltyped++; } function toBegin() { state = 1; btn.hide(); // nameField.hide(); ref = random(answers); startime = millis(); btn2.hide() } function replay(){ ref= random(answers); i= 0 typed = ''; startime = millis(); totaltyped = 0; loop() ; btn.hide() let accuracy = 100; btn2.hide() }*

Reflections

WEEK 3.5 :

I am not that scared for this WA, as we have done somthing similar to this last year too – building a typing game for our final project. Furthermore, there are many open-source projects available online such as github repos (although might be in python) that can help to guide me along the way and there are already quite a few existing games like Typeracer and Monkeytype, enabling me to have a clear sensing of what I have to acheive.

Coming up with my plan for the game, what elements I want to include, how I want the UI to look like (i can perhaps draw something up). Based on my clear visualisation, I'll then be able to actually code them up (aka the landing page, the score system, the elements I'd want to include). I feel the main issue I should really tacke first should be the actual system of the game, meaning the physics behind the typing (words dropping down from the top of the screen etc.)

If i had unlimited resources, I would probably make the UI more minimalistic, code out an actual AI (not in JS though, use python for the backend, and create an entire VSCode project file) for the typers to race against – where basically the AI learns from all the players that have used the website before, and is just slightly better than the participant at all times. I would also use it to refine my system, and try and create an app using swift.

I think one thing which I'm currently too stuck on is just solely focusing on the elements from which the game becomes good (i.e in this case the AI i wanna build, the design elements) but instead I should be fixated on the actual playing system itself, meaning the typing boxes, typing, as they are the key details of the project and without them, the game wouldn't even be a game at all.

WEEK 4:

My Plan:

Initial Stage:

- *Create a typing game where words fall from the top of the screen and you must type out the word before it exits the screen*
- *If it exists the screen, you lose some health; once all health is gone – game is over*
- *There will be 3 levels of difficulty which refer to the speed at which the words fall to the bottom of the screen*

- *As time goes on the words start to fall at a faster speed*
- *Each letter turns green once if it is typed correctly, and becomes red if typed incorrectly – in which case, word cannot be typed further*
- *Create a leaderboard for players to key in their names and score*
- *Have a loading page (homepage) to enter name and select level*
- *Have the page with the actual game, with your current health on top left, score on top right, box where you can see your 'userInput' aka text typed in the middle top, WPM at the bottom left (currently unfinished)*
- *Once done, there will be a replay button to replay the game and the leaderboard*

Problems/Suggestions:

1. *When a letter is red and wrong and backspaced, it should become grey (have no colour, easier for user to play)*
2. *The box where you can see the word still has the previous words – i wanna change this to where you have to press 'enter' button to actually submit the word and then make userinput = '' so it gets cleared, but I'm not too sure on how to do it*
3. *Remove replay button from the homepage when u press on it*
4. *Make the game more sleek/professional by changing the css file (adding some elements, can experiment with making the buttons round, and an actual leaderboard to scroll up & down)*
5. *WPM does not work (idk why)????*

*My current code: https://editor.p5js.org/GMR_da_Pro/sketches/RngmL08F_*

Week 5:

*https://editor.p5js.org/GMR_da_Pro/sketches/RngmL08F_*

- I was unable to acccess the Figma website as it was apparently down on the day i checked (or my wifi just has an issue)

- 1. FIX THE BUTTONS AT THE START, 2. MAYBE ADD IN SOME FEATURES WHERE IF THE WORD HITS THE LINE, IT WILL BOUNCE AND EVEN IF IT HITS THE SIDE LINE UR OUT, MAKE A INFORMATIONAL BUTTON AT THE START TO SEE HOW TO PLAY THE GAME, 4. MAKE THE WPM FEATURE WORK

1. I made this game to utilise Physics and to make it a fun way for me to practice my typing. For the Physics part, I felt that the words falling down from the screen and the Jittering Hearts would really put that physics lesson we had to good use. For the game features, I didn't really have any standardised approach, it was more that when the ideas randomly came to my mind I would just pen them down, and try my best to figure out how to implement it in my game. I would also give my game to my friends who were seated next to me in CEP class and ask them how my game felt - for example, 2 weeks ago, in my game, the word speed when they were falling down were way too slow, and hence the game would be played for approximately 2 min before it would be over. Based on this, I decided to make the words travel faster as I thought that such short games will only be captivating enough if they are quick and short. I was also going to try a cool and innovative idea whereby i add in advertisements at the end but decided against it for the cool look and partially because, u need to create an account to get monetized.
2. The most proud thing of my code is the Jittering hearts and the leaderboard. For the jittering hearts, it took me about 20 min to actually get them to jitter and another half an hour to understand how to break one into half and remove it from the array when the word falls down. For the leaderboard, i can't really be too proud of it as i used chatgpt for it and still don't really understand the *.parse* feature in it and how the JSON leaderboard worked, but i tried my best to actually double check on chatgpt what it does - i understand that JSON is a place to store data in Javascript but don't really get the backend of it.
3. Yep, i definitely was able to utilise whatever we learnt in class over the past term. For example, i managed to utilise the physics concept in the jittering hearts and the vectors in the hearts too when removing them. Using object oriented programming also helped me a lot by dividing the code into different classes (take for example the words falling & the buttons to select the difficulty level) which helped to make the code more readable and understandable for me.
4. In the prior reflections, i feel that i showed a really good understanding of the concepts but one thing which i did not expect was the sheer pain in which to utilise the concepts, cus when u have a really cool idea and u dont' know how to even do it its really painful, another thing which really surprised me is how confident i was for this project but what really let me down is that i procrastinated this cep project to the last weekend which really got me started on back foot
5. First of all, really understand your own game and vision. Once all the details are sorted out, plan it all out by drawing it on a sketchboard and start coding it out. If you have any issues with the coding bit, ask AI or your teacher-mentor to help you with it. First try to get the base of it to work before implementing the additional features. If you have any thoughts or additional features you would like to add in during the game coding itself, save them for later and first refine them before actually adding them in.
6. Yeah, I think it was honestly a worthwhile project. Perhaps we could have made it more fun, by publishing it into a website and seeing how it would actually perform or giving us free leeway and telling us to not just be stuck on one idea as I'm pretty sure half the class used the project whereby words fall from the top of the screen.