Process & Decision Documentation

Week 3 Side Quests

Focus on:

- Build an interactive story game that would lead to different game state
- Multiple game states and files, branching like a small decision tree

GenAI Documentation

If GenAI was used (keep each response as brief as possible):

Date Used: Feb 1, 2026

Tool Disclosure: Gemini Pro

Purpose of Use: Structuring the multi-file architecture, generating code for the state router, and debugging button interaction issues.

Summary of Interaction: The tool provided the initial code for my interactive story and helped troubleshoot a bug where the button was not able to clicks.

Human Decision Point(s): I rejected some codes and some incorrect file naming that provided by AI. I modified the game logic code in order to match my game story theme. I did not fully accept AI's code and move the css codes section to a separate page instead putting them all together.

Integrity & Verification Note: I verified the code by click go live in the vs code, and running it in the Chrome browser. When errors found, I verify the error and fix it by myself before asking the AI for a specific help.

Scope of GenAI Use: The AI handled the file linking and basic code logic structure, I handled the original story narrative concept idea, text aesthetic and variable optimization. I edited the codes based on the structure that GenAI gave me.

Limitations or Misfires: The tool initially provided code where the click detection coordinates in start.js did not match the drawing coordinates, causing the button to be unclickable. As well as the incorrect file naming, lead to disconnect with the pages.

Summary of Process (Human + Tool)

I started by defining the story concept and asking the AI to set up the file structure. Once the base files were created, I realized the variable names didn't make sense for my story, so I manually renamed the variables to systemHealth and ask AI to help me modify based on my current codes. And I edited the codes for every section based on the code structure that GenAI gave me. I then tested the game and found a bug where the button wasn't working, so I tried debug myself before asking AI to fix it. Finally, I test multiple times, and made some small modifications for example like css code section and the appearance of the text.

Appendix

Hi, I am working on a p5.js project for my class using the Game States architecture. Please refer to the instruction and the example that I uploaded.

I want to build a simple interactive story about：
The Concept: I want to build a short interactive story called “The Mysterious File”.
Start Screen: Shows a retro computer desktop background with a title 'System Alert'.
Game Screen: A popup appears saying 'Unknown file detected'. The player has a stat called curiosityScore starting at 0.
Choice A (Safe): ' Reject File'. This sets curiosityScore to 0 and goes to the 'Safe Ending'.
Choice B (Risky): ' Accept File'. This increases curiosityScore to 100 and goes to the 'Hacked Ending'.

Can you help me with the index.html (to load the scripts) and main.js? In main.js, please set up the state router and the global curiosityScore variable. As well as the instruction page.
Please modify based on the example.
Here are the updated files for your "The Mysterious File" project.

I have updated the architecture to include the global curiosityScore variable and renamed the ending states to safe and hacked to match your story concept.

1. index.html

I have updated the title and the script tags. Note that I have changed win.js to safe.js and lose.js to hacked.js to match your story endings.

HTML

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#1e3a3a" />
    <title>The Mysterious File</title>
    <link rel="preconnect" href="https://cdn.jsdelivr.net" />
    <link rel="stylesheet" href="style.css" />
  </head>
  <body>
    <header class="site-header">
      <div class="wrap">
        <h1 class="title">System Alert</h1>
        <p class="subtitle">Interactive Story • Week 3 Demo</p>
      </div>
    </header>

    <main class="wrap">
      <section class="card">
        <div class="canvas-frame">
          </div>
        <p class="hint">
          Tip: Navigate using the mouse or keyboard shortcuts provided.
        </p>
      </section>
    </main>

    <footer class="site-footer">
      <div class="wrap">
        <small>© 2026 Interactive Story Inc · Built with p5.js</small>
      </div>
    </footer>

    <script src="https://cdn.jsdelivr.net/npm/p5@1/lib/p5.min.js"></script>

    <script src="start.js"></script>
    <script src="instructions.js"></script>
    <script src="game.js"></script>

    <script src="safe.js"></script>
    <script src="hacked.js"></script>

    <script src="main.js"></script>
  </body>
</html>
```

2. main.js

This file now acts as the central router for your specific story states (safe, hacked) and initializes your curiosityScore.

JavaScript

```
// ------------------------------------------------------------
// main.js
// ------------------------------------------------------------

// ------------------------------
// Global game state
// ------------------------------
// 1. Current Screen Router
let currentScreen = "start"; // "start" | "instr" | "game" | "safe" | "hacked"

// 2. Story Variables
// We define this here so it is accessible by game.js, safe.js, and hacked.js
let curiosityScore = 0;

// ------------------------------
// setup() runs ONCE
// ------------------------------
function setup() {
  createCanvas(800, 600);

  // A retro font style works well for "System Alert" themes
  textFont("Courier New");
}

// ------------------------------
// draw() runs every frame
// ------------------------------
```

```
function draw() {
  // Router: Check currentScreen and call the specific file's draw function
  if (currentScreen === "start") {
    drawStart();
  } else if (currentScreen === "instr") {
    drawInstr();
  } else if (currentScreen === "game") {
    drawGame();
  } else if (currentScreen === "safe") {
    // Replaces 'win'
    drawSafe();
  } else if (currentScreen === "hacked") {
    // Replaces 'lose'
    drawHacked();
  }
}

// ------------------------------
// mousePressed()
// ------------------------------
function mousePressed() {
  if (currentScreen === "start") startMousePressed();
  else if (currentScreen === "instr") instrMousePressed();
  else if (currentScreen === "game") gameMousePressed();

  // Use ?.() to prevent errors if you haven't created safe.js/hacked.js yet
  else if (currentScreen === "safe") safeMousePressed?.();
  else if (currentScreen === "hacked") hackedMousePressed?.();
}

// ------------------------------
// keyPressed()
// ------------------------------
function keyPressed() {
  if (currentScreen === "start") startKeyPressed();
  else if (currentScreen === "instr") instrKeyPressed();
  else if (currentScreen === "game") gameKeyPressed?.();
  else if (currentScreen === "safe") safeKeyPressed?.();
  else if (currentScreen === "hacked") hackedKeyPressed?.();
}

// ------------------------------
// Helper: isHover()
// ------------------------------
```

```
function isHover({ x, y, w, h }) {
  return (
    mouseX > x - w / 2 &&
    mouseX < x + w / 2 &&
    mouseY > y - h / 2 &&
    mouseY < y + h / 2
  );
}
```

3. instructions.js

I have customized the text to fit the "Unknown File" narrative.

JavaScript

```
// ------------------------------
// instructions.js
// ------------------------------

function drawInstr() {
  // Retro blue screen background
  background(0, 0, 100);

  // ---- Screen title ----
  fill(255);
  textAlign(CENTER, TOP);
  textSize(32);
  text("SYSTEM MANUAL", width / 2, 80);

  // ---- Instruction text ----
  textSize(18);
  fill(200); // Light grey text

  const lines =
    "An unknown file has appeared on your desktop.\n\n" +
    "Your Goal:\n" +
    "Decide whether to ACCEPT or REJECT the file.\n\n" +
    "Warning:\n" +
    "Curiosity may lead to system corruption.";

  text(lines, width / 2, 180);

  // ---- Back button ----
  const backBtn = {
    x: width / 2,
    y: 500,
    w: 200,
    h: 60,
    label: "RETURN TO DESKTOP",
  };

  drawInstrButton(backBtn);

  // Change cursor on hover
  cursor(isHover(backBtn) ? HAND : ARROW);
}

// ------------------------------
// Inputs
// ------------------------------
function instrMousePressed() {
  const backBtn = { x: width / 2, y: 500, w: 200, h: 60 };
  if (isHover(backBtn)) {
    currentScreen = "start";
  }
}

function instrKeyPressed() {
  if (keyCode === ESCAPE || key === "b" || key === "B") {
    currentScreen = "start";
  }
}

// ------------------------------
// Button Helper
// ------------------------------
function drawInstrButton({ x, y, w, h, label }) {
  rectMode(CENTER);
  const hover = isHover({ x, y, w, h });

  noStroke();
  // Green retro button style
  fill(hover ? color(0, 255, 0) : color(0, 180, 0));
  rect(x, y, w, h, 4);

  fill(0);
  textSize(16);
  textAlign(CENTER, CENTER);
  text(label, x, y);
}
```

Next Steps for You:

To make the game run without errors, you must create the remaining files referenced in the HTML, even if they are empty placeholders for now.

Would you like me to write the code for **game.js** next to handle the logic of setting the curiosityScore based on the user's choice?

uhh change the variable of curiosityScore to systemHealth. And do the following:

Now please write the code for the other 4 files:

start.js: A green-on-black terminal style start screen with a 'BOOT SYSTEM' button.
game.js: Show a warning message 'Incoming Suspicious File'. Draw two buttons:
Button A ('ACCEPT'): If clicked, set systemHealth = 0 and go to 'lose' screen.
Button B ('REJECT'): If clicked, keep systemHealth = 100 and go to 'win' screen.
win.js (Good Ending): Green background. Text: 'Threat Blocked. System Safe.' Show the final systemHealth.
lose.js (Bad Ending): Red background. Text: 'CRITICAL ERROR. SYSTEM COMPROMISED.' Show the final systemHealth.
Please write the code for these files based on the example I gave
Here are the updated files.

First, strictly to ensure your **main.js** works with these new files, please update the global variable name at the top of **main.js** from curiosityScore to systemHealth.

JavaScript

```
// In main.js
let systemHealth = 100; // Renamed from curiosityScore
```

Here is the code for the 4 screen files. I have kept the filenames safe.js and hacked.js to match the HTML setup from the previous step, but they contain the exact "Win/Lose" logic you requested.

1. start.js

Theme: Retro Terminal (Green on Black).

JavaScript

```
// ------------------------------
// start.js - Retro Terminal Style
// ------------------------------

function drawStart() {
  // Terminal Black Background
  background(0);

  // ---- Retro CRT Lines (Optional Effect) ----
  // Draws faint lines to look like an old monitor
  stroke(0, 50, 0);
  for (let i = 0; i < height; i += 4) {
    line(0, i, width, i);
  }

  // ---- Title Text ----
  noStroke();
  fill(0, 255, 0); // Hacker Green
  textFont("Courier New"); // Monospaced font
  textAlign(CENTER, CENTER);

  textSize(50);
  text("> THE_FILE.EXE", width / 2, 200);

  textSize(20);
  // Blinking cursor effect using frameCount
  let cursor = frameCount % 60 < 30 ? "_" : " ";
  text("v.1.0.4 " + cursor, width / 2, 260);

  // ---- BOOT Button ----
  const bootBtn = {
    x: width / 2,
    y: 400,
    w: 250,
    h: 60,
    label: "[ BOOT SYSTEM ]"
  };

  drawTerminalButton(bootBtn);

  cursor(isHover(bootBtn) ? HAND : ARROW);
}

// ---- Input Handling ----
function startMousePressed() {
  const bootBtn = { x: width / 2, y: 400, w: 250, h: 60 };

  if (isHover(bootBtn)) {
    currentScreen = "game";
  }
}

function startKeyPressed() {
  if (keyCode === ENTER) {
    currentScreen = "game";
  }
}

// ---- Helper Helper ----
function drawTerminalButton({ x, y, w, h, label }) {
  rectMode(CENTER);
  const hover = isHover({ x, y, w, h });

  // Terminal style: Outline only, fill on hover
  stroke(0, 255, 0);
  strokeWeight(2);

  if (hover) {
    fill(0, 255, 0); // Fill green
    rect(x, y, w, h);
    fill(0); // Text becomes black
  } else {
    noFill(); // Transparent inside
    rect(x, y, w, h);
    fill(0, 255, 0); // Text stays green
  }

  noStroke();
  textSize(24);
  textAlign(CENTER, CENTER);
  text(label, x, y);
}
```

2. game.js

Theme: The decision point. Handles the systemHealth logic.

JavaScript

```
// ------------------------------
// game.js - The Decision
// ------------------------------

function drawGame() {
  background(50); // Dark grey desktop background

  // ---- Popup Window Design ----
  rectMode(CENTER);

  // Shadow
  fill(0, 0, 0, 100);
  rect(width / 2 + 10, height / 2 + 10, 500, 350, 4);

  // Main Window
  fill(220);
  stroke(0);
  strokeWeight(2);
  rect(width / 2, height / 2, 500, 350, 4);

  // Window Title Bar
  fill(0, 0, 150); // Navy Blue
  rect(width / 2, height / 2 - 155, 500, 40, 4, 4, 0, 0);

  fill(255);
  noStroke();
  textSize(20);
  textAlign(CENTER, CENTER);
  text("⚠ SECURITY ALERT", width / 2, height / 2 - 155);

  // ---- Message ----
  fill(0);
  textSize(24);
  text("Incoming Suspicious File", width / 2, height / 2 - 50);

  textSize(16);
  text("Do you want to accept this transfer?", width / 2, height / 2);

  // ---- Buttons ----
  // 1. Accept (Risky)
  const btnAccept = {
    x: width / 2 - 120,
    y: height / 2 + 100,
    w: 180,
    h: 50,
    label: "ACCEPT",
    type: "danger" // Custom property for styling
  };

  // 2. Reject (Safe)
  const btnReject = {
    x: width / 2 + 120,
    y: height / 2 + 100,
    w: 180,
    h: 50,
    label: "REJECT",
    type: "safe"
  };

  drawChoiceButton(btnAccept);
  drawChoiceButton(btnReject);

  cursor(isHover(btnAccept) || isHover(btnReject) ? HAND : ARROW);
}

// ---- Input Handling ----
function gameMousePressed() {
  const btnAccept = { x: width / 2 - 120, y: height / 2 + 100, w: 180, h: 50 };
  const btnReject = { x: width / 2 + 120, y: height / 2 + 100, w: 180, h: 50 };

  // BUTTON A: ACCEPT (Fail State)
  if (isHover(btnAccept)) {
    systemHealth = 0;      // Set health to 0
    currentScreen = "hacked"; // Go to Lose screen
  }

  // BUTTON B: REJECT (Win State)
  else if (isHover(btnReject)) {
    systemHealth = 100;    // Keep health at 100
    currentScreen = "safe";   // Go to Win screen
  }
}

// ---- Helper ----
function drawChoiceButton({ x, y, w, h, label, type }) {
  rectMode(CENTER);
  const hover = isHover({ x, y, w, h });
  stroke(0);
  strokeWeight(1);

  // Color logic based on button type
  if (type === "danger") {
    fill(hover ? color(255, 100, 100) : color(255, 200, 200));
  } else {
    fill(hover ? color(100, 255, 100) : color(200, 255, 200));
  }

  rect(x, y, w, h, 5);

  fill(0);
  noStroke();
  textSize(18);
  text(label, x, y);
}
```

3. safe.js (The "Win" Screen)

Theme: Success, Green Background.

JavaScript

```
// ------------------------------
// safe.js - Good Ending (Win)
// ------------------------------

function drawSafe() {
  // Green background (Success)
  background(100, 200, 100);

  fill(255);
  noStroke();
  textAlign(CENTER, CENTER);

  // Icon (Simple Shield shape using text or shapes)
  textSize(80);
  text("🛡️", width / 2, 200);

  // Main Text
  fill(0, 50, 0);
  textSize(36);
  text("Threat Blocked.", width / 2, 300);
  text("System Safe.", width / 2, 350);

  // Show System Health
  textSize(24);
  fill(0);
  // Using the global variable systemHealth
  text("System Health: " + systemHealth + "%", width / 2, 420);

  // Restart Instruction
  textSize(18);
  fill(255);
  text("Press R to Reboot System", width / 2, 520);
}

// ---- Inputs ----
function safeMousePressed() {
  // Click anywhere to restart
  currentScreen = "start";
}

function safeKeyPressed() {
  if (key === 'r' || key === 'R') {
    currentScreen = "start";
  }
}
```

4. hacked.js (The "Lose" Screen)

Theme: Failure, Red Background.

JavaScript

```
// ------------------------------
// hacked.js - Bad Ending (Lose)
// ------------------------------

function drawHacked() {
  // Red background (Critical Error)
  background(180, 0, 0);

  fill(255);
  noStroke();
  textAlign(CENTER, CENTER);
  // Skull Icon
  textSize(80);
  text("☠️", width / 2, 200);

  // Main Text
  textSize(40);
  textLeading(45); // Spacing between lines
  text("CRITICAL ERROR\nSYSTEM COMPROMISED", width / 2, 320);

  // Show System Health
  textSize(28);
  fill(255, 200, 200);
  text("System Health: " + systemHealth + "%", width / 2, 430);

  // Restart Instruction
  textSize(18);
  fill(255);
  text("Press R to Reformat Drive", width / 2, 520);
}

// ---- Inputs ----
function hackedMousePressed() {
  // Reset health on restart is optional here
  // because start.js doesn't use it, but good practice:
  systemHealth = 100;
  currentScreen = "start";
}

function hackedKeyPressed() {
  if (key === 'r' || key === 'R') {
    systemHealth = 100;
    currentScreen = "start";
  }
}
```

modify the instructions.js, main.ja and html based on the change I made. There are 7 page in total: html, instructions.js, main.js, start.js, game.js, win.js and lose.js
make sure their name is correct.
Here are the updated **index.html**, **main.js**, and **instructions.js**.

I have adjusted them to match your **7-file structure** exactly
(using win.js and lose.jsinstead of "safe/hacked") and implemented
the **systemHealth** logic.

1. index.html

I updated the script tags to load the file names you specified (win.js, lose.js).

HTML

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <title>The Mysterious File</title>
    <link rel="preconnect" href="https://cdn.jsdelivr.net" />
    <link rel="stylesheet" href="style.css" />
  </head>
  <body>
    <header class="site-header">
      <div class="wrap">
        <h1 class="title">System Alert</h1>
        <p class="subtitle">GBDA302 • Week 3 — Interactive Story</p>
      </div>
    </header>

    <main class="wrap">
      <section class="card">
        <div class="canvas-frame">
          </div>
        <p class="hint">
          Tip: Use the Mouse to click buttons or Keyboard (Enter/R/B) to navigate.
        </p>
      </section>
    </main>

    <footer class="site-footer">
      <div class="wrap">
        <small>© 2026 Interactive Story Inc · Built with p5.js</small>
      </div>
    </footer>

    <script src="https://cdn.jsdelivr.net/npm/p5@1/lib/p5.min.js"></script>

    <script src="start.js"></script>
    <script src="instructions.js"></script>
    <script src="game.js"></script>
    <script src="win.js"></script>
    <script src="lose.js"></script>

    <script src="main.js"></script>
  </body>
</html>
```

2. main.js

I replaced curiosityScore with systemHealth and set up the router to handle the win and lose screens.

JavaScript

```
// ------------------------------------------------------------
// main.js
// ------------------------------------------------------------

// ------------------------------
// Global game state
// ------------------------------
// 1. Current Screen Router
let currentScreen = "start"; // "start" | "instr" | "game" | "win" | "lose"

// 2. Story Variable
// Changing curiosityScore to systemHealth as requested
let systemHealth = 100;

// ------------------------------
// setup()
// ------------------------------
function setup() {
  createCanvas(800, 600);

  // A retro monospace font fits the "System Alert" theme best
  textFont("Courier New");
}

// ------------------------------
// draw()
// ------------------------------
function draw() {
  // Routes to the correct draw function based on currentScreen
  if (currentScreen === "start") {
    drawStart();
  } else if (currentScreen === "instr") {
    drawInstr();
  } else if (currentScreen === "game") {
    drawGame();
  } else if (currentScreen === "win") {
    drawWin();
  } else if (currentScreen === "lose") {
    drawLose();
  }
}

// ------------------------------
// mousePressed()
// ------------------------------
function mousePressed() {
  if (currentScreen === "start") startMousePressed();
  else if (currentScreen === "instr") instrMousePressed();
  else if (currentScreen === "game") gameMousePressed();

  // Using ?.() safely calls the function only if it exists in win.js/lose.js
  else if (currentScreen === "win") winMousePressed?.();
  else if (currentScreen === "lose") loseMousePressed?.();
}

// ------------------------------
// keyPressed()
// ------------------------------
function keyPressed() {
  if (currentScreen === "start") startKeyPressed();
  else if (currentScreen === "instr") instrKeyPressed();
  else if (currentScreen === "game") gameKeyPressed?.();
  else if (currentScreen === "win") winKeyPressed?.();
  else if (currentScreen === "lose") loseKeyPressed?.();
}

// ------------------------------
// Helper: isHover()
// ------------------------------
function isHover({ x, y, w, h }) {
  return (
    mouseX > x - w / 2 &&
    mouseX < x + w / 2 &&
    mouseY > y - h / 2 &&
    mouseY < y + h / 2
  );
}
```

3. instructions.js

I updated the text to explain the mechanics of systemHealth.

JavaScript

```
// ------------------------------
// instructions.js
// ------------------------------

function drawInstr() {
  // Dark Blue "Blue Screen of Death" style background
  background(0, 0, 150);

  // ---- Screen title ----
  fill(255);
  textAlign(CENTER, TOP);
  textSize(32);
  text("SYSTEM MANUAL", width / 2, 60);

  // ---- Instruction text ----
  textSize(18);
  textAlign(CENTER, TOP);

  const lines =
    "WARNING: SUSPICIOUS ACTIVITY DETECTED\n\n" +
    "OBJECTIVE:\n" +
    "You will be presented with an unknown file.\n" +
    "You must decide to ACCEPT or REJECT it.\n\n" +
    "GAME MECHANICS:\n" +
    "- Your System Health starts at 100%.\n" +
    "- Rejecting the file keeps the system safe.\n" +
    "- Accepting the file may compromise the system\n" +
    "  (Health drops to 0%).";

  text(lines, width / 2, 140);

  // ---- Back button ----
  const backBtn = {
    x: width / 2,
    y: 520,
    w: 220,
    h: 60,
    label: "RETURN TO ROOT",
  };

  drawInstrButton(backBtn);

  // Change cursor when hovering over the button
  cursor(isHover(backBtn) ? HAND : ARROW);
}

// ------------------------------
// Inputs
// ------------------------------
function instrMousePressed() {
  const backBtn = { x: width / 2, y: 520, w: 220, h: 60 };
  if (isHover(backBtn)) {
    currentScreen = "start";
  }
}

function instrKeyPressed() {
  // ESC or B to go back
  if (keyCode === ESCAPE || key === "b" || key === "B") {
    currentScreen = "start";
  }
}

// ------------------------------
// Button Helper
// ------------------------------
function drawInstrButton({ x, y, w, h, label }) {
  rectMode(CENTER);
  const hover = isHover({ x, y, w, h });

  noStroke();
  // Grey button style
  fill(hover ? 200 : 150);
  rect(x, y, w, h, 4);

  fill(0);
  textSize(18);
  textAlign(CENTER, CENTER);
  text(label, x, y);
}
```

the boot system button not working

If the "BOOT SYSTEM" button isn't working, it is usually because the main.js file isn't detecting the click or the button coordinates don't match where you are clicking.

Here is a troubleshooting guide and a **fixed version of start.js** with debugging tools included.

1. The "Hand Cursor" Test

Hover your mouse over the button.

- **If the cursor turns into a hand:** The code "sees" the button correctly. The issue is likely in main.js.
- **If the cursor stays an arrow:** The code (specifically isHover) isn't calculating the position correctly.

2. Check the Console

Open your browser's Developer Tools (Right Click > Inspect > Console tab).

- If you see **isHover is not defined**: Your main.js is loading *after* it's needed, or has a syntax error.
- If you see **startMousePressed is not defined**: Your start.js file isn't linked correctly in the HTML.

3. The Fix (Updated Code)

Here is a robust version of **start.js** and **main.js**. I have added console.log messages so you can see exactly what is happening in the Console.

Update start.js

I have consolidated the button variables so the "draw" logic and "click" logic are guaranteed to use the exact same numbers.

JavaScript

```
// ------------------------------
// start.js - Fixed & Debugged
// ------------------------------
```

```
function drawStart() {
  background(0); // Black terminal background

  // Retro green grid lines
  stroke(0, 50, 0);
  for (let i = 0; i < height; i += 4) {
    line(0, i, width, i);
  }

  // Title
  noStroke();
  fill(0, 255, 0);
  textAlign(CENTER, CENTER);
  textSize(50);
  text("> THE_FILE.EXE", width / 2, 200);

  // Blinking cursor
  textSize(20);
  let cursorChar = frameCount % 60 < 30 ? "_" : " ";
  text("v.1.0.4 " + cursorChar, width / 2, 260);

  // ---- BUTTON DEFINITION ----
  // We define the button here to pass it to the drawer
  const bootBtn = {
    x: width / 2,
    y: 400,
    w: 250,
    h: 60,
    label: "[ BOOT SYSTEM ]"
  };

  drawTerminalButton(bootBtn);

  // Cursor logic
  if (isHover(bootBtn)) {
    cursor(HAND);
  } else {
    cursor(ARROW);
  }
}

// ------------------------------
// Input Handling
// ------------------------------
```

```
function startMousePressed() {
  // Define the EXACT same button coordinates as above
  const bootBtn = {
    x: width / 2,
    y: 400,
    w: 250,
    h: 60
  };

  // Debugging: This will show up in your browser console (F12)
  console.log("Start screen clicked!");

  if (isHover(bootBtn)) {
    console.log("Button clicked! Switching to game...");
    currentScreen = "game";
  } else {
    console.log("Clicked, but missed the button.");
  }
}

function startKeyPressed() {
  if (keyCode === ENTER) {
    currentScreen = "game";
  }
}

// ------------------------------
// Helper
// ------------------------------
function drawTerminalButton({ x, y, w, h, label }) {
  rectMode(CENTER); // Important: This ensures x,y is the center
  const hover = isHover({ x, y, w, h });

  stroke(0, 255, 0);
  strokeWeight(2);

  if (hover) {
    fill(0, 255, 0);
    rect(x, y, w, h);
    fill(0);
  } else {
    noFill();
    rect(x, y, w, h);
    fill(0, 255, 0);
  }

  noStroke();
  textSize(24);
  textAlign(CENTER, CENTER);
  text(label, x, y);
}
```

Update main.js (Double Check This)

Ensure your mousePressed function looks **exactly** like this. A common mistake is missing the startMousePressed() line.

JavaScript

```
function mousePressed() {
  // Check which screen is active and call its specific click handler
  if (currentScreen === "start") {
    startMousePressed();
  }
  else if (currentScreen === "instr") {
    instrMousePressed();
  }
  else if (currentScreen === "game") {
    gameMousePressed();
  }
  else if (currentScreen === "win") {
    // safe navigation using ?. in case the function isn't ready
    winMousePressed?.();
  }
  else if (currentScreen === "lose") {
    loseMousePressed?.();
  }
}
```

4. Verify your index.html

Make sure main.js is loaded **LAST**, so it can "see" the functions inside start.js.

HTML

```
<script src="start.js"></script>
<script src="instructions.js"></script>
<script src="game.js"></script>
<script src="win.js"></script>
```

```
<script src="lose.js"></script>

<script src="main.js"></script>
```

Great. Now please write the code for the other 4 files:

start.js: A green-on-black terminal style start screen with a 'BOOT SYSTEM' button.
game.js: Show a warning message 'Incoming Suspicious File'. Draw two buttons:
Button A ('ACCEPT'): If clicked, set systemHealth = 0 and go to 'lose' screen.
Button B ('REJECT'): If clicked, keep systemHealth = 100 and go to 'win' screen.
win.js (clicked reject): Green background. Text: 'Threat Blocked. System Safe.' Show the final systemHealth.
lose.js (clicked accept): Red background. Text: 'CRITICAL ERROR. SYSTEM COMPROMISED.' Show the final systemHealth.
Please write the code for these files.

Here is the code for the four remaining files (start.js, game.js, win.js, lose.js) tailored to your specific story requirements and the systemHealth variable.

1. start.js

Theme: Retro Hacker Terminal (Green on Black).

JavaScript

```
// ------------------------------
// start.js - Retro Terminal Style
// ------------------------------

function drawStart() {
  // 1. Background: Terminal Black
  background(0);

  // 2. Retro Grid Lines (Optional visual flair)
  stroke(0, 50, 0); // Faint green lines
  strokeWeight(1);
  for (let i = 0; i < height; i += 10) {
    line(0, i, width, i);
  }

  // 3. Title Text
  noStroke();
  fill(0, 255, 0); // Hacker Green
  textAlign(CENTER, CENTER);

  // Large Title
  textSize(50);
  text("> THE_FILE.EXE", width / 2, 200);

  // Subtitle with blinking cursor effect
  textSize(20);
  let cursorChar = (frameCount % 60 < 30) ? "_" : " ";
  text("v.1.0.4 Ready " + cursorChar, width / 2, 260);

  // 4. Draw Button
  // We define the button object here so we can pass it to the drawer
  const bootBtn = {
    x: width / 2,
    y: 420,
    w: 260,
    h: 60,
    label: "[ BOOT SYSTEM ]"
  };

  drawTerminalButton(bootBtn);

  // Change cursor to hand if hovering
  cursor(isHover(bootBtn) ? HAND : ARROW);
}

// ---- Input Handling ----
function startMousePressed() {
  // Define button same as in drawStart
  const bootBtn = { x: width / 2, y: 420, w: 260, h: 60 };

  if (isHover(bootBtn)) {
    currentScreen = "game";
  }
}

function startKeyPressed() {
  // Allow Enter key to start
  if (keyCode === ENTER) {
    currentScreen = "game";
  }
}

// ---- Helper Function to Draw Button ----
function drawTerminalButton({ x, y, w, h, label }) {
  rectMode(CENTER);
  const hover = isHover({ x, y, w, h });

  stroke(0, 255, 0);
  strokeWeight(2);

  if (hover) {
    // On hover: Filled Green box, Black text
    fill(0, 255, 0);
    rect(x, y, w, h);
    fill(0);
  } else {
    // Normal: Empty box, Green text
    noFill();
    rect(x, y, w, h);
    fill(0, 255, 0);
  }

  noStroke();
  textSize(24);
  textAlign(CENTER, CENTER);
  text(label, x, y);
}
```

2. game.js

Theme: The Decision. Handles the logic for systemHealth.

JavaScript

```
// ------------------------------
// game.js - The Decision Screen
// ------------------------------

function drawGame() {
  background(50); // Dark Grey Desktop

  // ---- Popup Window Container ----
  rectMode(CENTER);

  // Shadow
  fill(0, 0, 0, 100);
  rect(width / 2 + 8, height / 2 + 8, 500, 350, 4);

  // Window Body
```

```
fill(220); // Light grey window
stroke(0);
strokeWeight(2);
rect(width / 2, height / 2, 500, 350, 4);

// Window Title Bar (Blue)
fill(0, 0, 150);
rect(width / 2, height / 2 - 155, 500, 40, 4, 4, 0, 0);

fill(255);
noStroke();
textSize(18);
textAlign(CENTER, CENTER);
text("⚠ SECURITY ALERT", width / 2, height / 2 - 155);

// ---- Warning Message ----
fill(0);
textSize(28);
text("Incoming Suspicious File", width / 2, height / 2 - 50);

textSize(16);
text("Do you want to accept this transfer?", width / 2, height / 2);

// ---- Buttons ----

// Button A: ACCEPT (Risky/Bad)
const btnAccept = {
 x: width / 2 - 110,
 y: height / 2 + 100,
 w: 180,
 h: 50,
 label: "ACCEPT",
 color: color(255, 100, 100) // Reddish
};

// Button B: REJECT (Safe/Good)
const btnReject = {
 x: width / 2 + 110,
 y: height / 2 + 100,
 w: 180,
 h: 50,
 label: "REJECT",
 color: color(100, 255, 100) // Greenish
};
```

```
  drawGameButton(btnAccept);
  drawGameButton(btnReject);

  // Cursor handling
  if (isHover(btnAccept) || isHover(btnReject)) {
    cursor(HAND);
  } else {
    cursor(ARROW);
  }
}

// ---- Input Handling ----
function gameMousePressed() {
  const btnAccept = { x: width / 2 - 110, y: height / 2 + 100, w: 180, h: 50 };
  const btnReject = { x: width / 2 + 110, y: height / 2 + 100, w: 180, h: 50 };

  // Logic: ACCEPT -> LOSE (Health 0)
  if (isHover(btnAccept)) {
    systemHealth = 0;
    currentScreen = "lose";
  }

  // Logic: REJECT -> WIN (Health 100)
  else if (isHover(btnReject)) {
    systemHealth = 100;
    currentScreen = "win";
  }
}

// ---- Helper ----
function drawGameButton({ x, y, w, h, label, color }) {
  rectMode(CENTER);
  const hover = isHover({ x, y, w, h });

  stroke(0);
  strokeWeight(1);

  // Darken color slightly if not hovering
  if (hover) {
    fill(color);
  } else {
    // Create a slightly darker version manually or just use the color
    fill(red(color)-30, green(color)-30, blue(color)-30);
  }

  rect(x, y, w, h, 5);

  fill(0);
  noStroke();
  textSize(18);
  text(label, x, y);
}
```

3. win.js

Theme: Success (Good Ending).

JavaScript

```
// ------------------------------
// win.js - Good Ending (Reject)
// ------------------------------

function drawWin() {
  // Green background to indicate safety
  background(100, 220, 100);

  fill(255);
  stroke(0);
  strokeWeight(4);
  textAlign(CENTER, CENTER);

  // Big Icon
  textSize(80);
  text("🛡", width / 2, 200);

  // Main Text
  noStroke();
  fill(0, 50, 0); // Dark Green Text
  textSize(40);
  text("Threat Blocked.", width / 2, 300);
  text("System Safe.", width / 2, 350);

  // Show System Health (Global Variable)
  textSize(30);
  fill(0);
  text("System Health: " + systemHealth + "%", width / 2, 450);
  // Restart Instruction
  textSize(18);
  fill(255);
  text("[ Click or Press R to Restart ]", width / 2, 550);
}

// ---- Input Handling ----
function winMousePressed() {
  currentScreen = "start";
}

function winKeyPressed() {
  if (key === 'r' || key === 'R') {
    currentScreen = "start";
  }
}
```

4. lose.js

Theme: Critical Error (Bad Ending).

JavaScript

```
// ------------------------------
// lose.js - Bad Ending (Accept)
// ------------------------------

function drawLose() {
  // Red background for danger
  background(200, 50, 50);

  fill(255);
  textAlign(CENTER, CENTER);

  // Big Icon
  textSize(80);
  text("☠", width / 2, 200);

  // Main Text
  textSize(38);
  textStyle(BOLD);
  text("CRITICAL ERROR", width / 2, 300);
  textStyle(NORMAL);
  textSize(32);
  text("SYSTEM COMPROMISED", width / 2, 350);

  // Show System Health (Global Variable)
  // This should display "0%" based on game.js logic
  textSize(30);
  fill(0);
  text("System Health: " + systemHealth + "%", width / 2, 450);

  // Restart Instruction
  textSize(18);
  fill(255);
  text("[ Click or Press R to Restart ]", width / 2, 550);
}

// ---- Input Handling ----
function loseMousePressed() {
  // Reset health (optional, but good practice for restart)
  systemHealth = 100;
  currentScreen = "start";
}

function loseKeyPressed() {
  if (key === 'r' || key === 'R') {
    systemHealth = 100;
    currentScreen = "start";
  }
}
```