

Process & Decision Documentation

Side Quests

Focus on:

This game is an interactive “Island Survival” strategy game. Players must select 5 items from a set of 10 to maximize their chances of survival. Each item possesses a specific survival value. The game’s outcome is determined by the player’s specific choices, resulting in one of three outcomes: Survival Master, Barely Survived, or Perished on the Island.

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

Date Used: Feb 1, 2026

Tool Disclosure: Gemini 3

Purpose of Use: Generate game code and make improvements.

Summary of Interaction:

- Click buttons or use keyboard shortcuts (ENTER to Start, I for Instructions) to start the game.
- Click an item card to add it to your backpack. Click again to remove it. You must select exactly 5 items from 10 items.
- Once 5 items are selected, click the EMBARK button at the bottom of the screen to lock in your choice.
- Receive one of three results based on your selection: Survival Master, Barely Survived, or Perished on the Island.
- On the result screen, press R or click anywhere to play again.

Human Decision Point(s): All game content, 10 items, the value of each item, and the final three endings are entirely my own original ideas.

Integrity & Verification Note: I verified the generated code by manually running the game in the browser and testing every feature. I also play tested the game UI, such as the new backpack area, to confirm the updates solved all the visual issues before finalizing the code.

Scope of GenAI Use: GenAI was primarily used as a coding assistant to implement my design concepts. I used it to write the JavaScript syntax for the state management system, the array shuffling algorithm, and the scoring logic.

Limitations or Misfires: GenAI initially provided code snippets rather than complete files, which caused the game to fail to start. It also sorted items numerically by value instead of randomly, requiring me to manually fix the gameplay logic. Additionally, the layout generated by GenAI was not user-friendly due to overlapping features, so I had to adjust the code to resolve these issues.

Summary of Process (Human + Tool)

- I started by defining the game concept (Survival Strategy) and used GenAI to generate the initial JavaScript structure. The first draft was incomplete and caused a crash (missing setup() function). I then iterated on the gameplay mechanics, such as moving from a static item list to a randomized grid. I then iterated on the gameplay mechanics, moving from a static item list to a randomized grid to improve replayability. After several rounds of iteration, the game finally achieved its basic interface and gameplay.

Decision Points & Trade-offs

- After considering numerous game options, I ultimately chose the island survival game because it's a relatively basic and fun game. After many revisions, I finally finalized all the game settings and options.

Verification & Judgement

- I manually verified the game logic by testing different item combinations to ensure it correctly triggered the three distinct endings.

Limitations, Dead Ends, or Open Questions

- Due to time constraints, items are currently represented by text boxes rather than images.

Appendix

Appendix: GenAI Transcript

Tool Used: Gemini **Date:** Feb 1, 2026

Interaction 1

User Prompt: I want to add a survival game based on this code. At the start, users can choose 5 items from a total of 10. Each item has a survival value. If the final score is:

Score > 80: “Survival Master”

Score 50-80: “Barely Survived”

Score < 50: “Perished on the Desert Island”

Items are as follows:

Flint (+25)

Water Tablets (+20)

Multi-tool Swiss Army Knife (+15)

First Aid Kit (+15)

Fishing Gear (+10)

Solar-Powered Flashlight (+5)

Thick Blanket (+5)

Bag of Snacks (-5)

Umbrella (-10)

Heavy Encyclopedia (-15)

Items are represented by text. When a user clicks an item, its icon or text turns gray (indicating selection) and appears in the “Backpack Bar” at the bottom of the screen.

AI Response: To implement the Survival Game, we need to add two new files (survival.js and survivalResult.js) and update your existing main.js and start.js to link them together. Here is the step-by-step implementation plan:

1. New File: survival.js

Create this file to handle the item selection, backpack logic, and drawing the game interface.

JavaScript

```
// survival.js

// -----
// Data: Items & Game State
// -----
// We define the items with their labels and score values.
// We also track their position (x, y) and selection state.
let survivalItems = [
  { label: "Flint", val: 25, selected: false },
  { label: "Water Tablets", val: 20, selected: false },
  { label: "Multi-tool", val: 15, selected: false },
  { label: "First Aid Kit", val: 15, selected: false },
  { label: "Fishing Gear", val: 10, selected: false },
  { label: "Flashlight", val: 5, selected: false },
  { label: "Thick Blanket", val: 5, selected: false },
  { label: "Bag of Snacks", val: -5, selected: false },
  { label: "Umbrella", val: -10, selected: false },
  { label: "Encyclopedia", val: -15, selected: false },
];

// We assign grid positions to the items in setup or dynamically in draw.
// To keep it simple, we'll assign them fixed positions here.
// Grid: 2 rows of 5 items.
const startX = 130;
const startY = 200;
const gapX = 135;
const gapY = 120;

survivalItems.forEach((item, index) => {
  item.w = 110;
  item.h = 80;
  // Calculate column (0-4) and row (0-1)
  const col = index % 5;
  const row = Math.floor(index / 5);
  item.x = startX + col * gapX;
  item.y = startY + row * gapY;
});

// Calculate final score shared with the result screen
let finalSurvivalScore = 0;

// -----
```

```

// Main Draw Function
// -----
function drawSurvival() {
    background(245, 245, 235); // Light beige background

    // Title
    fill(0);
    textSize(32);
    textAlign(CENTER, CENTER);
    text("Pack Your Backpack", width / 2, 80);
    textSize(18);
    text("Select exactly 5 items to survive.", width / 2, 120);

    // Draw all items
    survivalItems.forEach((item) => {
        drawSurvivalItem(item);
    });

    // Draw Backpack Bar
    drawBackpackBar();

    // Draw "Embark" Button (Confirm selection)
    // Only active if exactly 5 items are selected
    const count = survivalItems.filter((i) => i.selected).length;

    if (count === 5) {
        drawEmbarkButton(true);
    } else {
        drawEmbarkButton(false);
    }
}

// -----
// Helper: Draw Individual Item
// -----
function drawSurvivalItem(item) {
    rectMode(CENTER);

    // Hover check
    const hover = isHover(item);

    if (item.selected) {
        fill(150); // Gray if selected
        stroke(100);
    } else if (hover) {
        fill(220, 230, 255); // Light blue hover
        noStroke();
    } else {
        fill(255); // White default
        stroke(200);
    }

    rect(item.x, item.y, item.w, item.h, 8);
}

```

```

noStroke();

// Text
fill(item.selected ? 255 : 0);
textSize(14);
textAlign(CENTER, CENTER);
text(item.label, item.x, item.y, item.w - 10, item.h - 10); // box wrap
}

// -----
// Helper: Draw Backpack Bar
// -----
function drawBackpackBar() {
  fill(50);
  rectMode(CORNER);
  rect(0, height - 150, width, 150);

  fill(255);
  textSize(20);
  textAlign(LEFT, TOP);
  text("BACKPACK CONTENTS:", 20, height - 130);

  // List selected items
  let selected = survivalItems.filter((i) => i.selected);
  let xPos = 30;
  let yPos = height - 90;

  textSize(16);
  selected.forEach((item, i) => {
    text(`• ${item.label}`, xPos + i * 150, yPos);
  });
}

// -----
// Helper: Draw Embark Button
// -----
const embarkBtn = { x: 400, y: 720, w: 200, h: 60, label: "EMBARK" };

function drawEmbarkButton(active) {
  rectMode(CENTER);

  if (active) {
    const hover = isHover(embarkBtn);
    fill(hover ? color(100, 200, 100) : color(80, 180, 80)); // Green
    cursor(hover ? HAND : ARROW);
  } else {
    fill(200); // Disabled gray
  }

  rect(embarkBtn.x, embarkBtn.y, embarkBtn.w, embarkBtn.h, 20);

  fill(active ? 255 : 150);
  textSize(24);
}

```

```

textAlign(CENTER, CENTER);
text(embarkBtn.label, embarkBtn.x, embarkBtn.y);
}

// -----
// Input Handler
// -----
function survivalMousePressed() {
// 1. Check item clicks
for (let item of survivalItems) {
if (isHover(item)) {
const currentlySelected = survivalItems.filter((i) => i.selected).length;

// Toggle logic
if (item.selected) {
item.selected = false; // Deselect
} else if (currentlySelected < 5) {
item.selected = true; // Select (only if < 5)
}
return; // Stop checking after one click
}
}

// 2. Check Embark button click
const count = survivalItems.filter((i) => i.selected).length;
if (count === 5 && isHover(embarkBtn)) {
calculateAndFinish();
}
}

function calculateAndFinish() {
// Calculate total score
finalSurvivalScore = survivalItems
.filter((i) => i.selected)
.reduce((sum, i) => sum + i.val, 0);

// Go to result screen
currentScreen = "survivalResult";
}

// Helper to reset the game (call this when restarting)
function resetSurvivalGame() {
survivalItems.forEach(i => i.selected = false);
finalSurvivalScore = 0;
}

```

2. New File: survivalResult.js

This file handles the logic for displaying the three specific outcomes (Master, Barely, Perished).

JavaScript

```
// survivalResult.js
```

```

// -----
// Draw Result Screen
// -----
function drawSurvivalResult() {
  let title = "";
  let bgColor = color(255);

  // Determine outcome based on score
  if (finalSurvivalScore > 80) {
    title = "Survival Master";
    bgColor = color(150, 230, 150); // Success Green
  } else if (finalSurvivalScore >= 50) {
    title = "Barely Survived";
    bgColor = color(255, 230, 150); // Warning Yellow
  } else {
    title = "Perished on the Island";
    bgColor = color(255, 150, 150); // Failure Red
  }

  background(bgColor);

  fill(0);
  textAlign(CENTER, CENTER);

  // Outcome Title
  textSize(42);
  text(title, width / 2, 250);

  // Score display
  textSize(28);
  text("Final Score: " + finalSurvivalScore, width / 2, 350);

  // Restart instructions
  textSize(20);
  text("Press 'R' or Click to Try Again", width / 2, 500);
}

// -----
// Inputs
// -----
function survivalResultMousePressed() {
  resetSurvivalGame(); // defined in survival.js
  currentScreen = "start";
}

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

```

3. Update: main.js

We need to register the two new screens (survival and survivalResult) in the main "router".
Replace your draw(), mousePressed(), and keyPressed() functions in main.js with these updated versions:

JavaScript

```
// main.js updates

function draw() {
  if (currentScreen === "start") drawStart();
  else if (currentScreen === "instr") drawInstr();
  else if (currentScreen === "game") drawGame(); // Old game (optional)
  else if (currentScreen === "win") drawWin();
  else if (currentScreen === "lose") drawLose();

  // ---- NEW SCREENS ----
  else if (currentScreen === "survival") drawSurvival();
  else if (currentScreen === "survivalResult") drawSurvivalResult();
}

function mousePressed() {
  if (currentScreen === "start") startMousePressed();
  else if (currentScreen === "instr") instrMousePressed();
  else if (currentScreen === "game") gameMousePressed();
  else if (currentScreen === "win") winMousePressed?().;
  else if (currentScreen === "lose") loseMousePressed?().;

  // ---- NEW INPUTS ----
  else if (currentScreen === "survival") survivalMousePressed();
  else if (currentScreen === "survivalResult") survivalResultMousePressed();
}

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?().;

  // ---- NEW INPUTS ----
  else if (currentScreen === "survivalResult") survivalResultKeyPressed();
}
```

4. Update: start.js

Finally, update the Start button to point to our new Survival game instead of the old coin flip game.

In startMousePressed() inside start.js:

JavaScript

```
function startMousePressed() {
  // ... existing button definitions ...
```

```
const startBtn = { x: width / 2, y: 320, w: 240, h: 80 };
const instrBtn = { x: width / 2, y: 430, w: 240, h: 80 };

// CHANGE "game" TO "survival" HERE:
if (isHover(startBtn)) {
  // Reset the game data before starting
  if (typeof resetSurvivalGame === "function") {
    resetSurvivalGame();
  }
  currentScreen = "survival";
}
else if (isHover(instrBtn)) {
  currentScreen = "instr";
}
```

In startKeyPressed() inside start.js:

JavaScript

```
function startKeyPressed() {
  if (keyCode === ENTER) {
    if (typeof resetSurvivalGame === "function") {
      resetSurvivalGame();
    }
    currentScreen = "survival"; // Change "game" to "survival"
  }
  // ... rest of code
}
```

5. Update: index.html

Don't forget to include the two new script files in your HTML body so the browser can find them. Add them before main.js.

HTML

```
<script src="survival.js"></script>
<script src="survivalResult.js"></script>
```

Interaction 2

User Prompt: can you give me the whole code for start.js?

AI Response: Here is the complete, updated code for start.js.

The key changes here are in startMousePressed() and startKeyPressed(). I have updated them to trigger the survival screen and reset the game data (using the resetSurvivalGame() function we planned for survival.js) instead of launching the old coin-flip game.

start.js

JavaScript

```
// NOTE: Do NOT add setup() or draw() in this file
// setup() and draw() live in main.js
```

```

// This file only defines:
// 1) drawStart() → what the start/menu screen looks like
// 2) input handlers → what happens on click / key press on this screen
// 3) a helper function to draw menu buttons

// -----
// Start screen visuals
// -----
// drawStart() is called from main.js only when:
// currentScreen === "start"
function drawStart() {
    // Background colour for the start screen
    background(180, 225, 220); // soft teal background

    // ---- Title text ----
    fill(30, 50, 60);
    textSize(46);
    textAlign(CENTER, CENTER);
    text("Island Survival", width / 2, 180); // Updated Title

    // ---- Buttons (data only) ----
    // These objects store the position/size/label for each button.
    const startBtn = {
        x: width / 2,
        y: 320,
        w: 240,
        h: 80,
        label: "START",
    };

    const instrBtn = {
        x: width / 2,
        y: 430,
        w: 240,
        h: 80,
        label: "INSTRUCTIONS",
    };

    // Draw both buttons
    drawButton(startBtn);
    drawButton(instrBtn);

    // ---- Cursor feedback ----
    // If the mouse is over either button, show a hand cursor
    const over = isHover(startBtn) || isHover(instrBtn);
    cursor(over ? HAND : ARROW);
}

// -----
// Mouse input for the start screen
// -----
// Called from main.js only when currentScreen === "start"
function startMousePressed() {

```

```

// For input checks, we only need x,y,w,h (label is optional)
const startBtn = { x: width / 2, y: 320, w: 240, h: 80 };
const instrBtn = { x: width / 2, y: 430, w: 240, h: 80 };

// If START is clicked, go to the NEW survival screen
if (isHover(startBtn)) {
    // Check if the reset function exists (from survival.js) and run it
    // This ensures the backpack is empty every time we restart
    if (typeof resetSurvivalGame === "function") {
        resetSurvivalGame();
    }

    // Switch to the survival screen
    currentScreen = "survival";
}

// If INSTRUCTIONS is clicked, go to the instructions screen
else if (isHover(instrBtn)) {
    currentScreen = "instr";
}

// -----
// Keyboard input for the start screen
// -----
// Provides keyboard shortcuts:
// - ENTER starts the game
// - I opens instructions
function startKeyPressed() {
    if (keyCode === ENTER) {
        // Reset and start survival game
        if (typeof resetSurvivalGame === "function") {
            resetSurvivalGame();
        }
        currentScreen = "survival";
    }

    if (key === "I" || key === "I") {
        currentScreen = "instr";
    }
}

// -----
// Helper: drawButton()
// -----
// This function draws a button and changes its appearance on hover.
// It does NOT decide what happens when you click the button.
function drawButton({ x, y, w, h, label }) {
    rectMode(CENTER);

    // Check if the mouse is over the button rectangle
    const hover = isHover({ x, y, w, h });

    noStroke();

```

```

// ---- Visual feedback (hover vs not hover) ----
if (hover) {
  fill(255, 200, 150, 220); // warm coral on hover

  // Shadow settings (only when hovered)
  drawingContext.shadowBlur = 20;
  drawingContext.shadowColor = color(255, 180, 120);
} else {
  fill(255, 240, 210, 210); // soft cream base

  // Softer shadow when not hovered
  drawingContext.shadowBlur = 8;
  drawingContext.shadowColor = color(220, 220, 220);
}

// Draw the rounded rectangle button
rect(x, y, w, h, 14);

// Reset shadow
drawingContext.shadowBlur = 0;

// Draw the label text on top of the button
fill(40, 60, 70);
textSize(28);
textAlign(CENTER, CENTER);
text(label, x, y);
}

```

Interaction 3

User Prompt: There's no Start button, the game can't begin.

AI Response: This sounds like a "broken link" in the code chain.

If the Start button (and likely the whole canvas) is missing, it is almost certainly because **main.js is incomplete**.

In the previous steps, I gave you snippets to *update* main.js, but if you replaced the entire file with just those snippets, you likely deleted setup() (which creates the canvas) and the isHover() helper function. Without isHover(), the code crashes instantly when it tries to check for mouse position, so nothing gets drawn.

Here is the **full, corrected code** for the files that manage the game loop. Please copy and paste the **entire content** for each file to ensure nothing is missing.

1. Fix main.js (The Vital Fix)

This file **must** contain setup(), draw(), and the isHover() helper.
JavaScript

```
// -----
```

```

// main.js
// -----
let currentScreen = "start"; // "start", "instr", "survival", "survivalResult"

// -----
// setup() runs ONCE
// -----
function setup() {
  createCanvas(800, 800);
  textAlign("center");
}

// -----
// draw() runs every frame
// -----
function draw() {
  // Route to the correct screen
  if (currentScreen === "start") drawStart();
  else if (currentScreen === "instr") drawInstr();

  // New Survival Screens
  else if (currentScreen === "survival") {
    // Check if function exists to prevent crash if file is missing
    if (typeof drawSurvival === "function") drawSurvival();
    else background(255, 0, 0); // Red error screen if missing
  }
  else if (currentScreen === "survivalResult") {
    if (typeof drawSurvivalResult === "function") drawSurvivalResult();
  }

  // Old screens (optional, kept for safety)
  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 === "survival") {
    if (typeof survivalMousePressed === "function") survivalMousePressed();
  }
  else if (currentScreen === "survivalResult") {
    if (typeof survivalResultMousePressed === "function") survivalResultMousePressed();
  }
  else if (currentScreen === "game") gameMousePressed();
  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 === "survivalResult") {
    if (typeof survivalResultKeyPressed === "function") survivalResultKeyPressed();
  }
  else if (currentScreen === "game") gameKeyPressed?();
}

// -----
// Shared helper function: isHover()
// -----
// REQUIRED: This function checks if mouse is over a button.
// Without this, start.js will crash and draw nothing.
function isHover({ x, y, w, h }) {
  return (
    mouseX > x - w / 2 &&
    mouseX < x + w / 2 &&
    mouseY > y - h / 2 &&
    mouseY < y + h / 2
  );
}

```

2. Check index.html

Make sure your HTML loads the new files. If survival.js is not listed here, the game will crash when you click Start.

HTML

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>Island Survival</title>
    <link rel="stylesheet" href="style.css" />
    <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="survival.js"></script>
    <script src="survivalResult.js"></script>

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

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

```

```
<body>
  <main>
    </main>
</body>
</html>
```

3. Verify start.js

Ensure you are using the version I gave you in the last response. Specifically, check that it **does not** contain setup() or draw(), but it **does** contain drawStart() and drawButton().

Try this now:

1. Update main.js with the full code above.
 2. Update index.html to ensure survival.js and survivalResult.js are linked.
 3. Refresh the page. You should see the teal "Island Survival" screen.
-

Interaction 4

User Prompt: In the instructions, briefly summarize the game's rules and how to play in just a sentence or two.

AI Response: Here is the updated instructions.js file. I have modified the text in drawInstr() to briefly summarize the new survival rules.

instructions.js

JavaScript

```
// NOTE: Do NOT add setup() or draw() in this file
// setup() and draw() live in main.js
// This file only defines:
// 1) drawInstr() → what the instructions screen looks like
// 2) input handlers → how the player returns to the start screen
// 3) helper functions specific to this screen

// -----
// Main draw function for instructions screen
// -----
// drawInstr() is called from main.js
// only when currentScreen === "instr"
function drawInstr() {
  // Light neutral background
  background(240);

  // ---- Screen title ----
  fill(0);
  textAlign(CENTER, TOP);
  textSize(36);
  text("How to Play", width / 2, 80);

  // ---- Instruction text ----
  textSize(20);
  textAlign(CENTER, CENTER);
```

```

// \n creates a line break in the text
const lines =
  "You are stranded on a desert island!\n\n" +
  "Select exactly 5 items to pack in your backpack.\n" +
  "Choose wisely to maximize your survival score.";

text(lines, width / 2, 220);

// ---- Back button ----
// This button lets the player return to the start screen
const backBtn = {
  x: width / 2, // centred horizontally
  y: 560,
  w: 220,
  h: 70,
  label: "BACK",
};

// Draw the back button
drawInstrButton(backBtn);

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

// -----
// Mouse input for instructions screen
// -----
// Called from main.js only when currentScreen === "instr"
function instrMousePressed() {
  // Button data must match the draw position
  const backBtn = { x: width / 2, y: 560, w: 220, h: 70 };

  // If the button is clicked, return to the start screen
  if (isHover(backBtn)) {
    currentScreen = "start";
  }
}

// -----
// Keyboard input for instructions screen
// -----
// Provides keyboard-only navigation
function instrKeyPressed() {
  // ESC is a common "go back" key in games and apps
  if (keyCode === ESCAPE) {
    currentScreen = "start";
  }

  // B key is an additional, explicit shortcut for "back"
  if (key === "b" || key === "B") {
    currentScreen = "start";
  }
}

```

```

        }

// -----
// Button drawing helper (instructions screen)
// -----
function drawInstrButton({ x, y, w, h, label }) {
  rectMode(CENTER);

  // Check whether the mouse is hovering over the button
  const hover = isHover({ x, y, w, h });

  noStroke();

  // Subtle colour change on hover for visual feedback
  fill(hover ? color(200, 200, 255, 200) : color(220, 220, 255, 170));

  // Draw the button shape
  rect(x, y, w, h, 12);

  // Draw the button text
  fill(0);
  textSize(26);
  textAlign(CENTER, CENTER);
  text(label, x, y);
}

```

Interaction 5

User Prompt: I need all survival Items to be randomly positioned, not based on their value. and change Multi-tool to Swiss Army Knife, Water Tablets to Water Purification Tablets.

AI Response: Here is the updated survival.js file.

I have renamed the items as requested and updated the `resetSurvivalGame()` function. Now, every time you press "Start" (which calls `reset`), the `survivalItems` array is **shuffled**, and the items are re-assigned to grid positions. This ensures they appear in random locations rather than being sorted by value.

`survival.js`

`JavaScript`

```

// survival.js

// -----
// Data: Items & Game State
// -----
// Defined with labels and values.
// We do not set x/y here; they are set in resetSurvivalGame()
// so they can be randomized every time the game starts.
let survivalItems = [

```

```

{ label: "Flint", val: 25, selected: false },
{ label: "Water Purification Tablets", val: 20, selected: false }, // Renamed
{ label: "Swiss Army Knife", val: 15, selected: false }, // Renamed
{ label: "First Aid Kit", val: 15, selected: false },
{ label: "Fishing Gear", val: 10, selected: false },
{ label: "Flashlight", val: 5, selected: false },
{ label: "Thick Blanket", val: 5, selected: false },
{ label: "Bag of Snacks", val: -5, selected: false },
{ label: "Umbrella", val: -10, selected: false },
{ label: "Encyclopedia", val: -15, selected: false },
];

// Layout constants (Grid: 2 rows of 5)
const GRID_START_X = 130;
const GRID_START_Y = 200;
const GRID_GAP_X = 135;
const GRID_GAP_Y = 120;

// Initialize items with default positions once on load
// (Just to ensure x,y,w,h exist before the first shuffle)
survivalItems.forEach((item, index) => {
  item.w = 110;
  item.h = 80;
  item.x = GRID_START_X + (index % 5) * GRID_GAP_X;
  item.y = GRID_START_Y + Math.floor(index / 5) * GRID_GAP_Y;
});

let finalSurvivalScore = 0;

// -----
// Main Draw Function
// -----
function drawSurvival() {
  background(245, 245, 235); // Light beige background

  // Title
  fill(0);
  textSize(32);
  textAlign(CENTER, CENTER);
  text("Pack Your Backpack", width / 2, 80);
  textSize(18);
  text("Select exactly 5 items to survive.", width / 2, 120);

  // Draw all items
  survivalItems.forEach((item) => {
    drawSurvivalItem(item);
  });

  // Draw Backpack Bar
  drawBackpackBar();

  // Draw "Embark" Button (Confirm selection)
  // Only active if exactly 5 items are selected
}

```

```

const count = survivalItems.filter((i) => i.selected).length;

if (count === 5) {
  drawEmbarkButton(true);
} else {
  drawEmbarkButton(false);
}
}

// -----
// Helper: Draw Individual Item
// -----
function drawSurvivalItem(item) {
rectMode(CENTER);

// Hover check
const hover = isHover(item);

if (item.selected) {
  fill(150); // Gray if selected
  stroke(100);
} else if (hover) {
  fill(220, 230, 255); // Light blue hover
  noStroke();
} else {
  fill(255); // White default
  stroke(200);
}

rect(item.x, item.y, item.w, item.h, 8);
noStroke();

// Text
fill(item.selected ? 255 : 0);
textSize(14);
textAlign(CENTER, CENTER);
// Using width/height in text() ensures long names wrap automatically
text(item.label, item.x, item.y, item.w - 10, item.h - 10);
}

// -----
// Helper: Draw Backpack Bar
// -----
function drawBackpackBar() {
fill(50);
rectMode(CORNER);
rect(0, height - 150, width, 150);

fill(255);
textSize(20);
textAlign(LEFT, TOP);
text("BACKPACK CONTENTS:", 20, height - 130);
}

```

```

// List selected items
let selected = survivalItems.filter((i) => i.selected);
let xPos = 30;
let yPos = height - 90;

textSize(16);
selected.forEach((item, i) => {
  // If text is very long, we might truncate it slightly or just list it
  text(`• ${item.label}`, xPos + i * 150, yPos, 140, 60);
});
}

// -----
// Helper: Draw Embark Button
// -----
const embarkBtn = { x: 400, y: 720, w: 200, h: 60, label: "EMBARK" };

function drawEmbarkButton(active) {
  rectMode(CENTER);

  if (active) {
    const hover = isHover(embarkBtn);
    fill(hover ? color(100, 200, 100) : color(80, 180, 80)); // Green
    cursor(hover ? HAND : ARROW);
  } else {
    fill(200); // Disabled gray
  }

  rect(embarkBtn.x, embarkBtn.y, embarkBtn.w, embarkBtn.h, 20);

  fill(active ? 255 : 150);
  textSize(24);
  textAlign(CENTER, CENTER);
  text(embarkBtn.label, embarkBtn.x, embarkBtn.y);
}

// -----
// Input Handler
// -----
function survivalMousePressed() {
  // 1. Check item clicks
  for (let item of survivalItems) {
    if (isHover(item)) {
      const currentlySelected = survivalItems.filter((i) => i.selected).length;

      // Toggle logic
      if (item.selected) {
        item.selected = false; // Deselect
      } else if (currentlySelected < 5) {
        item.selected = true; // Select (only if < 5)
      }
      return; // Stop checking after one click
    }
  }
}

```

```
}

// 2. Check Embark button click
const count = survivalItems.filter((i) => i.selected).length;
if (count === 5 && isHover(embarkBtn)) {
  calculateAndFinish();
}
}

function calculateAndFinish() {
  // Calculate total score
  finalSurvivalScore = survivalItems
    .filter((i) => i.selected)
    .reduce((sum, i) => sum + i.val, 0);

  // Go to result screen
  currentScreen = "survivalResult";
}

// -----
// Reset & Randomize
// -----
// This is called from start.js when the player clicks START.
// It randomizes the item positions every time.
function resetSurvivalGame() {
  // 1. Reset selection & Score
  survivalItems.forEach((i) => (i.selected = false));
  finalSurvivalScore = 0;

  // 2. Shuffle the array to randomize order
  // (shuffle is a p5.js function)
  shuffle(survivalItems, true);

  // 3. Re-assign Grid Positions based on the NEW shuffled order
  survivalItems.forEach((item, index) => {
    // Reset size (just in case)
    item.w = 110;
    item.h = 80;

    // Calculate grid spot
    const col = index % 5;
    const row = Math.floor(index / 5);

    item.x = GRID_START_X + col * GRID_GAP_X;
    item.y = GRID_START_Y + row * GRID_GAP_Y;
  });
}
```

Interaction 6

User Prompt: Change the backpack contents area larger, with the embark positioned underneath. currently the embark is partially obstructing the backpack contents section.

AI Response: Here is the updated survival.js.

I have increased the size of the bottom bar to **200 pixels** (up from 150) and moved the **Embark** button down to the bottom edge so it sits clearly underneath the list of items.
survival.js

JavaScript

```
// survival.js

// -----
// Data: Items & Game State
// -----
// Defined with labels and values.
let survivalItems = [
  { label: "Flint", val: 25, selected: false },
  { label: "Water Purification Tablets", val: 20, selected: false },
  { label: "Swiss Army Knife", val: 15, selected: false },
  { label: "First Aid Kit", val: 15, selected: false },
  { label: "Fishing Gear", val: 10, selected: false },
  { label: "Flashlight", val: 5, selected: false },
  { label: "Thick Blanket", val: 5, selected: false },
  { label: "Bag of Snacks", val: -5, selected: false },
  { label: "Umbrella", val: -10, selected: false },
  { label: "Encyclopedia", val: -15, selected: false },
];

// Layout constants (Grid: 2 rows of 5)
const GRID_START_X = 130;
const GRID_START_Y = 200;
const GRID_GAP_X = 135;
const GRID_GAP_Y = 120;

// Initialize items with default positions once on load
survivalItems.forEach((item, index) => {
  item.w = 110;
  item.h = 80;
  item.x = GRID_START_X + (index % 5) * GRID_GAP_X;
  item.y = GRID_START_Y + Math.floor(index / 5) * GRID_GAP_Y;
});

let finalSurvivalScore = 0;

// -----
// Main Draw Function
// -----
function drawSurvival() {
  background(245, 245, 235); // Light beige background
```

```

// Title
fill(0);
textSize(32);
textAlign(CENTER, CENTER);
text("Pack Your Backpack", width / 2, 80);
textSize(18);
text("Select exactly 5 items to survive.", width / 2, 120);

// Draw all items
survivalItems.forEach((item) => {
  drawSurvivalItem(item);
});

// Draw Backpack Bar (Background and Text)
drawBackpackBar();

// Draw "Embark" Button (Confirm selection)
// Only active if exactly 5 items are selected
const count = survivalItems.filter((i) => i.selected).length;

if (count === 5) {
  drawEmbarkButton(true);
} else {
  drawEmbarkButton(false);
}

// -----
// Helper: Draw Individual Item
// -----
function drawSurvivalItem(item) {
  rectMode(CENTER);

  // Hover check
  const hover = isHover(item);

  if (item.selected) {
    fill(150); // Gray if selected
    stroke(100);
  } else if (hover) {
    fill(220, 230, 255); // Light blue hover
    noStroke();
  } else {
    fill(255); // White default
    stroke(200);
  }

  rect(item.x, item.y, item.w, item.h, 8);
  noStroke();

  // Text
  fill(item.selected ? 255 : 0);
  textSize(14);
}

```

```

textAlign(CENTER, CENTER);
text(item.label, item.x, item.y, item.w - 10, item.h - 10);
}

// -----
// Helper: Draw Backpack Bar
// -----
function drawBackpackBar() {
  fill(40); // Dark background
  rectMode(CORNER);

  // INCREASED SIZE: Height is now 200px (was 150) to fit button
  const barHeight = 200;
  const barY = height - barHeight;

  rect(0, barY, width, barHeight);

  fill(255);
  textSize(20);
  textAlign(LEFT, TOP);
  // Positioned at top of the bar
  text("BACKPACK CONTENTS:", 20, barY + 20);

  // List selected items
  let selected = survivalItems.filter((i) => i.selected);
  let xPos = 30;
  let yPos = barY + 60; // Positioned below the title

  textSize(16);
  selected.forEach((item, i) => {
    text(`• ${item.label}`, xPos + i * 150, yPos, 140, 60);
  });
}

// -----
// Helper: Draw Embark Button
// -----
// MOVED BUTTON: y changed from 720 to 740 to sit lower in the larger bar
const embarkBtn = { x: 400, y: 740, w: 200, h: 60, label: "EMBARK" };

function drawEmbarkButton(active) {
  rectMode(CENTER);

  if (active) {
    const hover = isHover(embarkBtn);
    fill(hover ? color(100, 200, 100) : color(80, 180, 80)); // Green
    cursor(hover ? HAND : ARROW);
  } else {
    fill(100); // Darker gray for disabled state
  }

  rect(embarkBtn.x, embarkBtn.y, embarkBtn.w, embarkBtn.h, 20);
}

```

```

fill(active ? 255 : 180);
textSize(24);
textAlign(CENTER, CENTER);
text(embarkBnn.label, embarkBnn.x, embarkBnn.y);
}

// -----
// Input Handler
// -----
function survivalMousePressed() {
  // 1. Check item clicks
  for (let item of survivalItems) {
    if (isHover(item)) {
      const currentlySelected = survivalItems.filter((i) => i.selected).length;

      // Toggle logic
      if (item.selected) {
        item.selected = false; // Deselect
      } else if (currentlySelected < 5) {
        item.selected = true; // Select (only if < 5)
      }
      return; // Stop checking after one click
    }
  }

  // 2. Check Embark button click
  const count = survivalItems.filter((i) => i.selected).length;
  if (count === 5 && isHover(embarkBnn)) {
    calculateAndFinish();
  }
}

function calculateAndFinish() {
  // Calculate total score
  finalSurvivalScore = survivalItems
    .filter((i) => i.selected)
    .reduce((sum, i) => sum + i.val, 0);

  // Go to result screen
  currentScreen = "survivalResult";
}

// -----
// Reset & Randomize
// -----
function resetSurvivalGame() {
  // 1. Reset selection & Score
  survivalItems.forEach((i) => (i.selected = false));
  finalSurvivalScore = 0;

  // 2. Shuffle the array to randomize order
  shuffle(survivalItems, true);
}

```

```
// 3. Re-assign Grid Positions based on the NEW shuffled order
survivalItems.forEach((item, index) => {
    item.w = 110;
    item.h = 80;

    // Calculate grid spot
    const col = index % 5;
    const row = Math.floor(index / 5);

    item.x = GRID_START_X + col * GRID_GAP_X;
    item.y = GRID_START_Y + row * GRID_GAP_Y;
});
}
```