“Just Divide — Kid Mode” Game Development Task

---

1. Introduction

Welcome to your Game Development Assignment.

Your task is to **recreate the complete “Just Divide — Kid Mode” game** using **Phaser 3**, based on the functional and visual description provided in this document.

This assignment evaluates:

- Your UI recreation accuracy
- Your understanding of game logic
- Your ability to work with responsive layouts
- Your coding structure and clarity
- Your problem-solving and polish

You will be given all required assets separately (PNG + SVG).

Use them exactly as provided.

---

2. Game Overview

Just Divide — Kid Mode is a math-based puzzle game designed for children (ages 7–12).
The objective is to help kids understand **division, factors, multiples, equal values**, and develop strategic thinking.

How It Works

The player drags numbered tiles into a **4×4 grid**.
When the tile touches neighbor tiles, it may **merge** based on mathematical logic:

Merge Rules

1. **Equal Tiles → Both Disappear**
   Example: 4 next to 4 → (4,4) becomes empty.

2. **Divisible Tiles → Merge**

   - Larger ÷ Smaller must be a whole number
   - Result replaces the larger tile
   - Smaller tile disappears

3. Example:

   - Placing 12 next to 3 → 12 ÷ 3 = 4 → Tile becomes 4
     Placing 15 next to 5 → becomes 3
   - Placing 9 next to 3 → becomes 3

4. **If Result is 1 → Remove the Tile**
   Example:

- 3 next to 3 (equal → vanish)
- 2 next to 2 (equal → vanish)
- 1 is not placed as a tile, but can appear as a quotient

---

3. Core Gameplay Flow

Each turn the player:

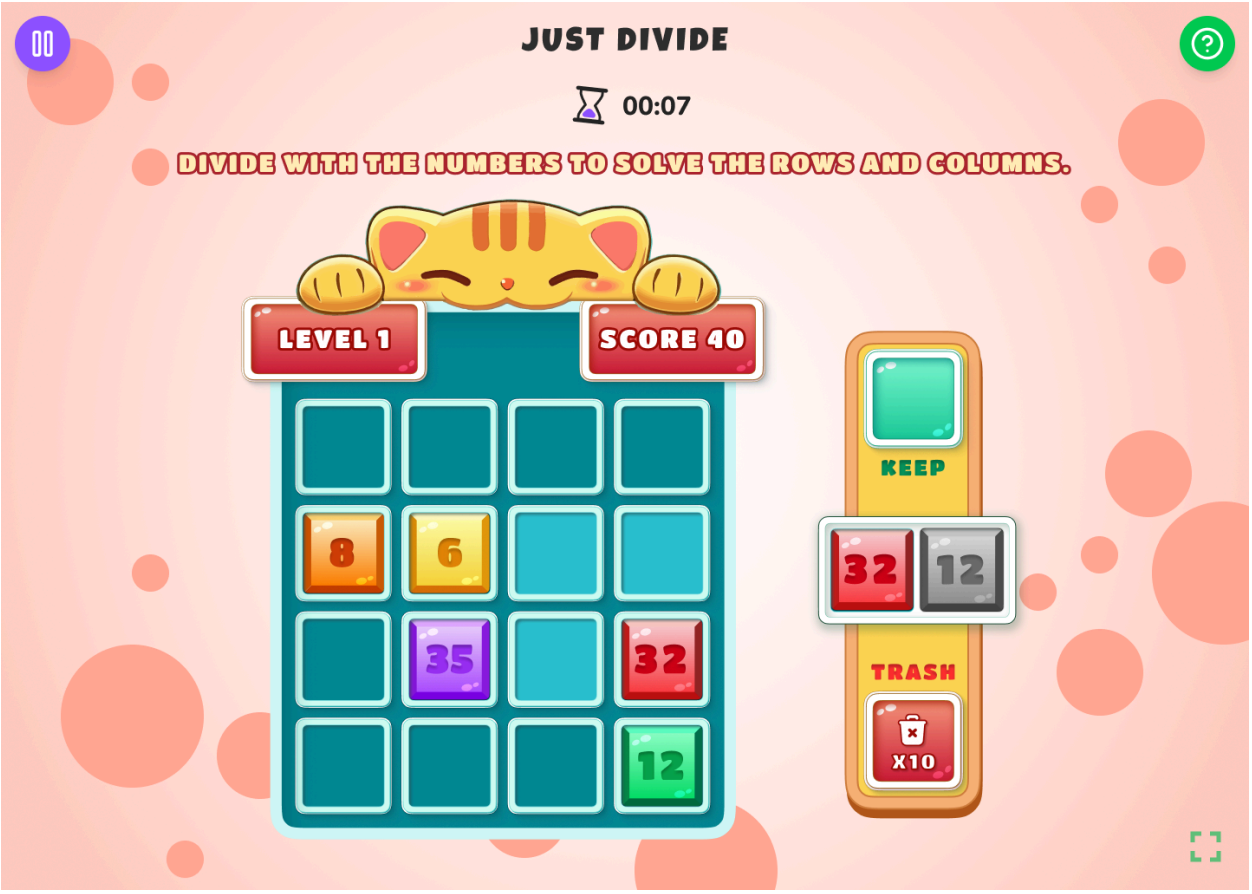
1. Takes the **top tile** from the upcoming stack
2. **Drags it** into the grid
3. It may merge immediately
4. Score updates
5. Next tile appears
6. Game continues until:
    - Grid is full AND
    - No valid merges are possible

This results in **Game Over**, showing score + best score.

---

4. UI Layout Description

The screen consists of the following elements:



4.1 Header Area

- Game title: **“JUST DIVIDE”**
- Subtitle: encourages dividing tiles to solve rows/columns
- Timer (optional): shows how long the session has lasted

4.2 Cat Character Header

A cartoon cat sits above the grid:

- Centrally aligned
- Arms resting over the left and right “Level” and “Score” badges
- Acts as a visual anchor for the game

4.3 Level and Score Panels (Badges)

Two rounded panels placed under the cat:

Left Panel

- Displays: **LEVEL X**

Right Panel

- Displays: **Score: N**
  Displays: **Best: N**

These must stay aligned horizontally with each other.

---

5. Grid Layout

A **4×4 grid** forms the primary play area:

- 16 turquoise slot boxes
- White borders
- Even spacing
- Centered in the gameplay area
- Each slot can hold **zero or one tile**

---

6. Number Tiles

Each tile:

- Is an **SVG/PNG rounded rectangle**, colored (blue, pink, orange, red, purple)
- Holds a **number** in bold
- Must be perfectly centered inside its slot

Tile color usually indicates "difficulty category" (not strict logic).

---

7. Right-Side Vertical Action Panel

A tall vertical panel (orange tone) aligned right of the grid.

Contains:

7.1 KEEP Slot

- Player can store **one tile**
- Pressing/dragging swaps the stored tile with the active tile

- Helps avoid bad moves

7.2 Upcoming Tiles Stack

Shows **three tiles**:

- Top tile = draggable
- Two more visible underneath
- These are the next tiles in queue

7.3 TRASH Box

- Allows discarding a tile
- Limited uses per level
- When used, tile disappears and next tile comes
- Trash counter shown below TRASH label

---

8. Background

Use the bubble/patterned wallpaper:

- Pale pink theme
- Must stretch to full screen
- No blank bars on sides

---

9. Controls & Input

Drag & Drop

- Player drags tile from the stack to:
    - Grid cell
    - KEEP slot
    - TRASH slot

Keyboard Support (Optional but recommended)

- **Z** → Undo
- **R** → Restart
- **1, 2, 3** → Difficulty toggle (Easy / Medium / Hard)
- **G** → Toggle Hints

---

10. Hint System

When hints are ON:

- Grid highlights cells where the current tile can merge
- Highlight only empty cells
- Showing where division or equality is possible

---

11. Leveling System

- Every **10 points** = next level
- Each level gives new trash uses
- Difficulty may slightly increase based on config

---

12. Game State Management

You must maintain:

- `grid[]` – 16 tile values
- `queue[]` – 3 upcoming tiles
- `keepVal` – stored tile
- `score`, `best`
- Undo stack (max 10 steps)
- Level / trash counts
- Hints on/off

---

13. Technical Requirements

You MUST use:

- You can use any of the following gaming engines Phaser 3 (latest stable)/ ReactJS/GoDot/ Unity
- Responsive scaling
- Design resolution: **1440×1024**

- Only vanilla JavaScript
- Clean folder structure
- No external JS libraries

---

14. Tasks to Complete

✔ UI Tasks

1. Build responsive scene using 1440×1024 design space
2. Position the cat, score/level badges, grid, and right panel
3. Render each SVG/PNG asset correctly
4. Center text inside tiles
5. Implement safe scaling for all screen sizes

✔ Functionality Tasks

6. Drag-drop tile system
7. Merge logic (equal/divisible)
8. Queue updating
9. KEEP tile mechanics
10. TRASH mechanics with limited uses
11. Undo system
12. Hint highlighting
13. Level-up system
14. Game-over detection
15. LocalStorage best score

16. Smooth animations

---

15. Submission Requirements

Github Repo:

- `Index.html`
- `main.js`
- `/assets` folder
- A short `README.md` explaining:
    - Approach
    - Decisions made
    - Challenges
    - What you would improve

Also provide:

- Playable build hosted on **Vercel**

15. AI Usage

Keep AI use to minimal