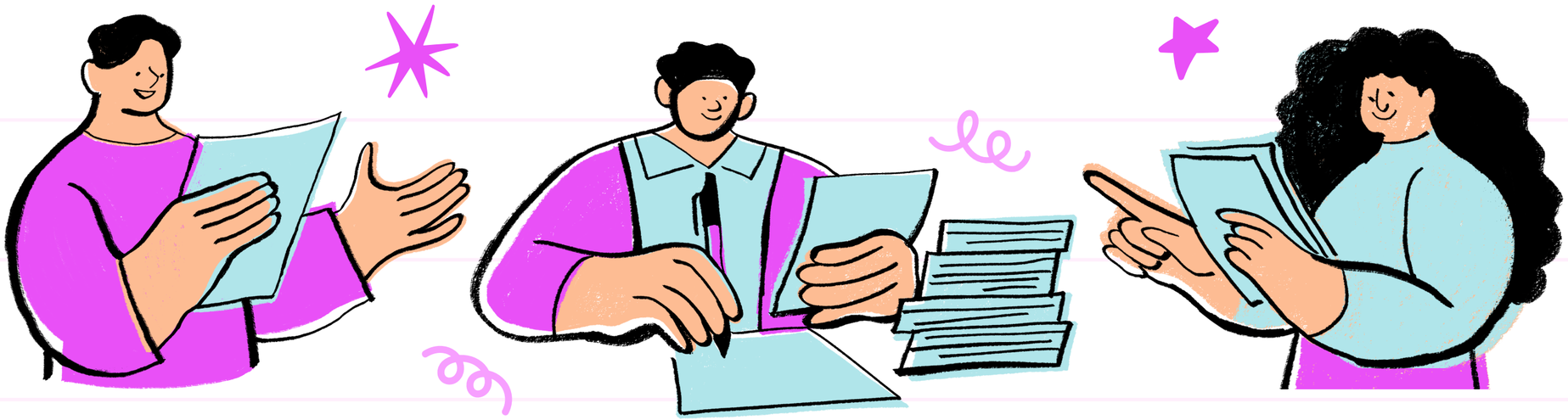
GROUP JJJ

March 26, 2025

Fateshaper

Strategic RPS



Jasmyre

Jarell

Jamille

Introduction

Game

FateShaper is an exciting twist on the classic Rock, Paper, Scissors game, incorporating RPG elements where two players duel using enhanced skills such as precision, strength, and speed, etc. Players can restore health during the match while managing fatigue, with the goal of reducing their opponent's health.

Target Audience

Our target audiences are people who enjoys being strategic in battles it can be a kid, teenager, adult and also the casual gamers

Game Idea & Concept

What is the game about?

Fate Shaper is basically a battle game that takes the old rock-paper-scissors idea and makes it way more strategic. Instead of just picking rock, paper, or scissors, you also manage things like your strength, speed, crit chance, fatigue, and momentum. Your moves (attack, skip, or heal) are affected by these stats, so every fight is a mix of smart choices and a bit of luck. Plus, you get to play different classes (like Warrior or Mage) that change how you play. It's like turning a simple game into a deep, tactical, turn-based battle!

🎮 Turn-Based Strategy

⚔️ RPG (Role-Playing Game)

🃏 Tactical Combat

🔢 Math-Based Combat

Technologies Used

HTML

HTML (HyperText Markup Language) is the backbone of the game's structure. It defines elements like buttons, health bars, and the game screen. Without HTML, there would be no visible layout for players to interact with.

CSS

CSS (Cascading Style Sheets) handles the design and animations of the game. It controls colors, fonts, layouts, and effects like transitions or glowing buttons. This makes the game visually appealing and interactive.

Typescript

Typescript Is a superset of JavaScript that adds static typing. It helps developers write more reliable and maintainable code. Typescript manages all the game logic, like calculating damage, tracking fatigue, and handling player actions.

Game Development Process

Planning

Before coding, the game structure was planned with wireframes and rough UI sketches. This included designing the game screen layout, placing health bars, action buttons, and a battle log. The goal was to create a clear and intuitive interface for smooth gameplay.

Coding Approach

- HTML sets up the structure, defining elements like the game board, buttons, and stats display.
- CSS styles everything, adding animations and making the UI visually appealing.
- TypeScript handles the game logic, including turn-based combat, stat calculations, and player interactions.

Game Development Process

Challenges & How They Were Overcome

UI Responsiveness

Some UI elements didn't display well on different screen sizes, so CSS adjustments were made for better user experience.

Game Logic Bugs

Unexpected behavior, like infinite loops in battle, was fixed through debugging and logging system improvements.

Code Snippets & Explanations

```
578
579   type Action =
580       / "attack-rock"
581       / "attack-paper"
582       / "attack-scissors"
583       / "skip"
584       / "heal";
585
586   function getRobotAction(): Action {
587       const actions: Action[] = [
588           "attack-rock",
589           "attack-paper",
590           "attack-scissors",
591           "skip",
592           "heal",
593       ];
594       return actions[Math.floor(Math.random() * actions.length)];
595   }
596
```

Game Lore

> *In a world where even the simplest gesture can change the tide of battle...*

In this universe, champions from different classes (Warrior, Mage, Rogue, Guardian, and Assassin) clash using the timeless moves of rock, paper, and scissors. Behind each move is a complex interplay of fatigue and momentum–winning builds momentum and makes your strikes more lethal, while overexertion raises fatigue and diminishes your defenses. Choose your moves carefully to outlast your opponent.

5

Unique Classes

Each with distinct strengths, weaknesses, and special abilities that dramatically change your combat strategy.

6+

Core Stats

Manage strength, precision, crit, speed, defense, and healing to create your perfect combat style.

∞

Strategic Depth

With fatigue and momentum mechanics, every decision matters in this deceptively complex battle system.

Tactical Combat

Master strategic turn-based battles with unique combat mechanics beyond simple rock-paper-scissors. Every attack is calculated using a complex system of stats and modifiers.

Dynamic Stats

Manage strength, precision, crit, speed, fatigue, and momentum to gain the upper hand in battle. Upgrade your stats after each round to customize your combat style.

Strategic Choices

Decide when to attack, when to skip turns to recover fatigue, and when to heal. Each decision has trade-offs that can turn the tide of battle.

Game Mechanics

Rounds

Core Stats

Strength

Conclusion

Key Takeaways

- Even a simple game idea like rock-paper-scissors can become engaging when combined with RPG and strategy elements.
- Organized code using TypeScript eases debugging and scaling as the project grows.

What we Learned:

- The value of thorough planning and testing in game development.
- How to leverage modern tools like Vite for faster development cycles.
- careful design, coding can turn a basic concept into a polished, playable game.

The end.