**Game-Wise Tech Stack**

**Visual Novel**

* **Frontend**:
  + Framework: React.js or Vue.js.
  + Styling: Tailwind CSS
  + Animation: GSAP for smooth transitions and interactions.
* **Game Logic**:
  + Phaser.js for lightweight interactivity or a custom logic layer with React.js Canvas.
* **Backend**:
  + Firebase Firestore for saving user progress.
  + Firebase Functions for handling user profile updates.

**Anshul’s Game (DSA Adventure)**

* **Frontend**:
  + Phaser.js: For 2D game mechanics and interactive levels.
  + Physics Engine: Phaser’s Arcade Physics for object collisions.
* **Backend**:
  + Node.js with Express.js for handling levels, user progress, and scores.
  + Database: MongoDB or Firebase for storing user achievements and stats.

**Kunisha’s Games**

* **Biryani Run**:
  + Framework: Phaser.js for fast-paced arcade-style gameplay.
  + Physics: Phaser Arcade Physics for obstacles and power-ups.
* **Corporate Jargon**:
  + Framework: React.js for quiz UI.
  + Backend: Firebase for storing questions and user progress.

**WE Smash!**

* **Game Engine**:
  + Unity for physics-based, dynamic gameplay, with WebGL export for browser compatibility.
  + Alternative: Phaser.js for simpler 2D mechanics.
* **Backend**:
  + Firebase for real-time leaderboards and score management.
* **Graphics**:
  + Blender or Adobe Illustrator for creating dynamic backgrounds and characters.

**Treasure Hunt**

* **Platform**:
  + React.js for web-based clue interactions.
  + Backend: Node.js or Firebase Functions for managing clues and power-ups.
* **Features**:
  + WebSockets (Socket.io) for real-time updates during the hunt.

**Asokan’s Game (Shifu’s Code Warrior)**

* **Frontend**:
  + Unity (C#) for immersive gamified coding challenges with export to HTML5.
  + Alternative: Phaser.js for simpler gameplay mechanics.
* **Backend**:
  + Firebase for saving user progress on the Path of Enlightenment.
  + API Integration: Flask (Python) for dynamically fetching coding challenges.

**Aruvi’s Game (Mini Games)**

* **Don’t Pop the Duckie**:
  + Game Engine: Phaser.js or Godot for mini-game physics.
* **Wordle & Diamonds**:
  + Framework: React.js for a clean and responsive UI.
  + Backend: Firebase for storing wordlists and player progress.
* **Multiplayer Option**:
  + Backend: WebSockets (Node.js) for real-time interactions.
  + Deployment: Firebase Hosting for a seamless multiplayer experience.