# Adventure Game - Complete Source Code

This document contains the full codebase for the Node.js Adventure Map Game.

## 1. Project Configuration

### File: package.json

Location: Root directory (/adventure-game/package.json)

Purpose: Defines dependencies and start scripts.

{  
 "name": "adventure-game-concept",  
 "version": "1.0.0",  
 "description": "Educational Adventure Map Game",  
 "main": "server.js",  
 "scripts": {  
 "start": "node server.js",  
 "dev": "nodemon server.js"  
 },  
 "dependencies": {  
 "cors": "^2.8.5",  
 "express": "^4.18.2",  
 "uuid": "^9.0.0"  
 },  
 "devDependencies": {  
 "nodemon": "^3.0.0"  
 }  
}

## 2. Backend (Node.js)

### File: server.js

Location: Root directory (/adventure-game/server.js)

Purpose: The Express server that handles API requests and file storage.

const express = require('express');  
const cors = require('cors');  
const path = require('path');  
const fs = require('fs');  
const { v4: uuidv4 } = require('uuid');  
  
const app = express();  
const PORT = 3000;  
  
// Middleware  
app.use(cors());  
app.use(express.json());  
app.use(express.static(path.join(\_\_dirname, 'public')));  
  
// Data Directory Setup  
const DATA\_DIR = path.join(\_\_dirname, 'data');  
const SESSIONS\_DIR = path.join(DATA\_DIR, 'sessions');  
  
// Ensure directories exist on start  
if (!fs.existsSync(DATA\_DIR)) fs.mkdirSync(DATA\_DIR);  
if (!fs.existsSync(SESSIONS\_DIR)) fs.mkdirSync(SESSIONS\_DIR);  
  
// --- API ROUTES ---  
  
// 1. Get Manifest (Available Themes/Games)  
// Returns hardcoded lists that match the file structure.  
app.get('/api/manifest', (req, res) => {  
 res.json({  
 themes: [  
 { id: 'forest', name: 'Whispering Woods', type: 'winding', color: '#2d5a27' },  
 { id: 'space', name: 'Galactic Route', type: 'circular', color: '#000022' }  
 ],  
 characters: [  
 { id: 'knight', name: 'Brave Knight', color: '#e74c3c' },  
 { id: 'bunny', name: 'Space Bunny', color: '#3498db' }  
 ],  
 games: [  
 { id: 'math\_add', name: 'Number Cruncher', script: 'js/minigames/game\_math\_add.js' }  
 ]  
 });  
});  
  
// 2. Start/Resume Session  
app.post('/api/session/start', (req, res) => {  
 const { profileName, charId, themeId } = req.body;  
   
 const sessionId = uuidv4();  
   
 const newSession = {  
 sessionId,  
 profileName,  
 characterId: charId,  
 themeId: themeId,  
 currentLevel: 1,  
 currentNode: 0, // Index of node (0-9)  
 totalStars: 0,  
 mercyMode: false, // If true, next game is easier  
 nodeHistory: [] // Track wins/losses  
 };  
  
 // Save to file system  
 fs.writeFileSync(  
 path.join(SESSIONS\_DIR, `${sessionId}.json`),   
 JSON.stringify(newSession, null, 2)  
 );  
  
 res.json(newSession);  
});  
  
// 3. Update Progress  
app.post('/api/session/update', (req, res) => {  
 const { sessionId, result } = req.body;  
 // result structure: { success: boolean, stars: int, usedMercy: boolean }  
  
 const filePath = path.join(SESSIONS\_DIR, `${sessionId}.json`);  
   
 if (!fs.existsSync(filePath)) {  
 return res.status(404).json({ error: "Session not found" });  
 }  
  
 const session = JSON.parse(fs.readFileSync(filePath));  
  
 // Game Logic: Update State  
 if (result.success) {  
 session.currentNode++;  
 session.totalStars += result.stars;  
 session.mercyMode = false; // Reset mercy on win  
 } else {  
 // If they failed 3 times and triggered mercy rule  
 if (result.usedMercy) {  
 session.currentNode++; // Advance  
 session.mercyMode = true; // Next game is easier  
 }  
 }  
  
 fs.writeFileSync(filePath, JSON.stringify(session, null, 2));  
 res.json({ success: true, state: session });  
});  
  
app.listen(PORT, () => {  
 console.log(`Server running at http://localhost:${PORT}`);  
});

## 3. Frontend (HTML & CSS)

### File: index.html

Location: /adventure-game/public/index.html

Purpose: The main game container.

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>Adventure Map Game</title>  
 <link rel="stylesheet" href="css/style.css">  
</head>  
<body>  
  
 <!-- 1. SETUP SCREEN -->  
 <div id="setup-panel" class="screen active">  
 <div class="panel-card">  
 <h1>Start Adventure</h1>  
 <div class="config-group">  
 <label>Map Theme:</label>  
 <select id="config-theme"></select>  
 </div>  
 <div class="config-group">  
 <label>Your Hero:</label>  
 <select id="config-char"></select>  
 </div>  
 <button id="btn-start" class="btn-primary">Let's Go!</button>  
 </div>  
 </div>  
  
 <!-- 2. GAME SCREEN -->  
 <div id="game-container" class="screen hidden">  
 <canvas id="map-canvas"></canvas>  
   
 <!-- Character Sprite (CSS Representation) -->  
 <div id="character-sprite"></div>  
  
 <!-- UI HUD -->  
 <div id="ui-hud">  
 <div class="hud-item">⭐ <span id="score-display">0</span></div>  
 <div class="hud-item">Level 1</div>  
 </div>  
  
 <!-- Feedback Message Overlay -->  
 <div id="feedback-msg" class="hidden"></div>  
 </div>  
  
 <!-- 3. MINI-GAME MODAL -->  
 <div id="game-modal" class="modal hidden">  
 <div class="modal-content">  
 <div class="modal-header">  
 <h2 id="modal-title">Challenge!</h2>  
 </div>  
 <div id="minigame-container">  
 <!-- Game Injected Here via JS -->  
 </div>  
 </div>  
 </div>  
  
 <!-- SCRIPTS -->  
 <script src="js/api.js"></script>  
 <script src="js/map\_renderer.js"></script>  
 <script src="js/ui\_manager.js"></script>  
   
 <!-- Pre-load the Math Game -->  
 <script src="js/minigames/game\_math\_add.js"></script>  
   
 <script src="js/main.js"></script>  
</body>  
</html>

### File: style.css

Location: /adventure-game/public/css/style.css

Purpose: Global styling, modal layouts, and animations.

body, html {  
 margin: 0; padding: 0; width: 100%; height: 100%;  
 overflow: hidden; font-family: 'Verdana', sans-serif;  
 background: #333;  
}  
  
/\* Screens \*/  
.screen { position: absolute; width: 100%; height: 100%; display: flex; justify-content: center; align-items: center; }  
.screen.hidden { display: none !important; }  
  
/\* Setup Panel \*/  
#setup-panel { background: #2c3e50; }  
.panel-card { background: white; padding: 40px; border-radius: 20px; text-align: center; width: 300px; }  
.config-group { margin: 20px 0; text-align: left; }  
select { width: 100%; padding: 10px; font-size: 16px; margin-top: 5px; }  
.btn-primary {  
 background: #27ae60; color: white; border: none; padding: 15px 30px;  
 font-size: 18px; border-radius: 50px; cursor: pointer; width: 100%;  
 transition: transform 0.2s;  
}  
.btn-primary:hover { transform: scale(1.05); }  
  
/\* Game Map \*/  
#game-container { display: block; }  
canvas { display: block; width: 100%; height: 100%; }  
  
#character-sprite {  
 position: absolute; width: 40px; height: 40px;  
 background: red; border-radius: 50%; border: 3px solid white;  
 box-shadow: 0 4px 10px rgba(0,0,0,0.5);  
 transition: all 1s ease-in-out; /\* Smooth movement \*/  
 z-index: 10; display: none;  
}  
  
#ui-hud {  
 position: absolute; top: 20px; left: 20px;  
 display: flex; gap: 20px; pointer-events: none;  
}  
.hud-item {  
 background: rgba(0,0,0,0.6); color: white; padding: 10px 20px;  
 border-radius: 30px; font-weight: bold; font-size: 20px;  
}  
  
/\* Messages \*/  
#feedback-msg {  
 position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);  
 font-size: 4rem; color: white; font-weight: bold; text-shadow: 0 5px 15px black;  
 pointer-events: none; z-index: 20;  
}  
  
/\* Modals \*/  
.modal {  
 position: fixed; top: 0; left: 0; width: 100%; height: 100%;  
 background: rgba(0,0,0,0.85);  
 display: flex; justify-content: center; align-items: center; z-index: 100;  
}  
.modal.hidden { display: none; }  
.modal-content {  
 background: #f0f3f4; width: 90%; max-width: 500px; height: 60%;  
 border-radius: 20px; display: flex; flex-direction: column;  
 box-shadow: 0 0 50px rgba(255,255,255,0.2); overflow: hidden;  
}  
.modal-header { background: #34495e; color: white; padding: 15px; text-align: center; }  
#minigame-container { flex: 1; padding: 20px; display: flex; flex-direction: column; justify-content: center; align-items: center; }  
  
/\* Generic Game Styles \*/  
.game-btn {  
 background: #3498db; color: white; border: none; padding: 20px; margin: 10px;  
 font-size: 24px; border-radius: 10px; cursor: pointer; min-width: 100px;  
}  
.game-btn:hover { background: #2980b9; }  
.game-display { font-size: 60px; font-weight: bold; color: #2c3e50; margin-bottom: 30px; }

## 4. Frontend Logic (JavaScript)

### File: api.js

Location: /adventure-game/public/js/api.js

Purpose: Wrapper for communicating with the Node backend.

const API = {  
 getManifest: async () => {  
 const res = await fetch('/api/manifest');  
 return await res.json();  
 },  
 startSession: async (profileName, charId, themeId) => {  
 const res = await fetch('/api/session/start', {  
 method: 'POST',  
 headers: { 'Content-Type': 'application/json' },  
 body: JSON.stringify({ profileName, charId, themeId })  
 });  
 return await res.json();  
 },  
 updateProgress: async (sessionId, result) => {  
 const res = await fetch('/api/session/update', {  
 method: 'POST',  
 headers: { 'Content-Type': 'application/json' },  
 body: JSON.stringify({ sessionId, result })  
 });  
 return await res.json();  
 }  
};

### File: map\_renderer.js

Location: /adventure-game/public/js/map\_renderer.js

Purpose: Handles HTML5 Canvas drawing and path generation.

const MapRenderer = {  
 canvas: null,  
 ctx: null,  
 nodes: [],  
 width: 0,   
 height: 0,  
  
 init: function(canvasId) {  
 this.canvas = document.getElementById(canvasId);  
 this.ctx = this.canvas.getContext('2d');  
 this.resize();  
 window.addEventListener('resize', () => this.resize());  
   
 // Click Handler  
 this.canvas.addEventListener('click', (e) => {  
 const rect = this.canvas.getBoundingClientRect();  
 const x = e.clientX - rect.left;  
 const y = e.clientY - rect.top;  
   
 this.nodes.forEach(node => {  
 // Simple Distance check (30px radius)  
 const dist = Math.sqrt((x - node.x)\*\*2 + (y - node.y)\*\*2);  
 if (dist < 30) {  
 // Dispatch event for Main.js to hear  
 window.dispatchEvent(new CustomEvent('node-clicked', { detail: { nodeId: node.id } }));  
 }  
 });  
 });  
 },  
  
 resize: function() {  
 this.width = window.innerWidth;  
 this.height = window.innerHeight;  
 this.canvas.width = this.width;  
 this.canvas.height = this.height;  
 if (this.nodes.length) this.draw();  
 },  
  
 generateNodes: function(type) {  
 this.nodes = [];  
 const count = 9; // Fixed for now  
 const padding = 100;  
   
 for(let i=0; i<count; i++) {  
 const progress = i / (count - 1);  
 let x, y;  
  
 if (type === 'circular') {  
 const angle = progress \* Math.PI \* 2 - (Math.PI / 2);  
 x = (this.width/2) + Math.cos(angle) \* (this.width/3);  
 y = (this.height/2) + Math.sin(angle) \* (this.height/3);  
 } else {  
 // Winding  
 x = padding + (progress \* (this.width - padding\*2));  
 y = (this.height/2) + Math.sin(progress \* Math.PI \* 3) \* 150;  
 }  
   
 this.nodes.push({ id: i, x, y, status: i === 0 ? 'current' : 'locked' });  
 }  
 this.draw();  
 },  
  
 draw: function() {  
 this.ctx.clearRect(0,0,this.width, this.height);  
  
 // Draw Lines  
 if (this.nodes.length > 1) {  
 this.ctx.beginPath();  
 this.ctx.moveTo(this.nodes[0].x, this.nodes[0].y);  
 for (let i=0; i<this.nodes.length-1; i++) {  
 const p0 = this.nodes[i];  
 const p1 = this.nodes[i+1];  
 this.ctx.lineTo(p1.x, p1.y);  
 }  
 this.ctx.strokeStyle = 'rgba(255,255,255,0.5)';  
 this.ctx.lineWidth = 5;  
 this.ctx.setLineDash([15, 15]);  
 this.ctx.stroke();  
 this.ctx.setLineDash([]);  
 }  
  
 // Draw Nodes  
 this.nodes.forEach(node => {  
 this.ctx.beginPath();  
 this.ctx.arc(node.x, node.y, 25, 0, Math.PI\*2);  
   
 if (node.status === 'completed') this.ctx.fillStyle = '#27ae60';  
 else if (node.status === 'current') this.ctx.fillStyle = '#f1c40f';  
 else this.ctx.fillStyle = '#7f8c8d';  
  
 this.ctx.fill();  
 this.ctx.strokeStyle = 'white';  
 this.ctx.lineWidth = 3;  
 this.ctx.stroke();  
   
 // Draw Number  
 this.ctx.fillStyle = 'white';  
 this.ctx.font = 'bold 16px Arial';  
 this.ctx.textAlign = 'center';  
 this.ctx.textBaseline = 'middle';  
 this.ctx.fillText(node.id + 1, node.x, node.y);  
 });  
 },  
  
 updateNodeStatus: function(nodeId, status) {  
 if (this.nodes[nodeId]) {  
 this.nodes[nodeId].status = status;  
 this.draw();  
 }  
 },  
  
 getNodePos: function(nodeId) {  
 return this.nodes[nodeId] ? { x: this.nodes[nodeId].x, y: this.nodes[nodeId].y } : {x:0,y:0};  
 }  
};

### File: ui\_manager.js

Location: /adventure-game/public/js/ui\_manager.js

Purpose: Handles DOM manipulation for menus, modals, and feedback.

const UIManager = {  
 showScreen: (id) => {  
 document.querySelectorAll('.screen').forEach(el => el.classList.add('hidden'));  
 document.querySelectorAll('.screen').forEach(el => el.classList.remove('active'));  
 const target = document.getElementById(id);  
 target.classList.remove('hidden');  
 target.classList.add('active');  
 },  
  
 populateConfig: (manifest) => {  
 const themeSelect = document.getElementById('config-theme');  
 const charSelect = document.getElementById('config-char');  
  
 manifest.themes.forEach(t => {  
 const opt = document.createElement('option');  
 opt.value = t.id; opt.textContent = t.name;  
 themeSelect.appendChild(opt);  
 });  
  
 manifest.characters.forEach(c => {  
 const opt = document.createElement('option');  
 opt.value = c.id; opt.textContent = c.name;  
 charSelect.appendChild(opt);  
 });  
 },  
  
 moveCharacter: (x, y, color) => {  
 const sprite = document.getElementById('character-sprite');  
 sprite.style.display = 'block';  
 sprite.style.backgroundColor = color || 'red';  
 // Center sprite on node (assuming 40px width)  
 sprite.style.left = (x - 20) + 'px';   
 sprite.style.top = (y - 20) + 'px';  
 },  
  
 openModal: (title) => {  
 document.getElementById('modal-title').textContent = title;  
 document.getElementById('game-modal').classList.remove('hidden');  
 },  
  
 closeModal: () => {  
 document.getElementById('game-modal').classList.add('hidden');  
 document.getElementById('minigame-container').innerHTML = ''; // Cleanup  
 },  
  
 updateScore: (stars) => {  
 document.getElementById('score-display').textContent = stars;  
 },  
  
 showFeedback: (msg, duration = 2000) => {  
 const el = document.getElementById('feedback-msg');  
 el.textContent = msg;  
 el.classList.remove('hidden');  
 setTimeout(() => el.classList.add('hidden'), duration);  
 }  
};

### File: game\_math\_add.js

Location: /adventure-game/public/js/minigames/game\_math\_add.js

Purpose: The specific logic for the Math Addition mini-game.

// Standard Interface: window.GameLibrary["ID"]  
window.GameLibrary = window.GameLibrary || {};  
  
window.GameLibrary["math\_add"] = {  
 config: {  
 id: 'math\_add',  
 name: 'Number Cruncher',  
 type: 'math'  
 },  
  
 start: function(container, difficulty, onComplete) {  
 container.innerHTML = ''; // Clear  
  
 // 1. Generate Question based on difficulty  
 // Diff 1: 1-5, Diff 5: 10-20  
 const max = difficulty \* 4 + 1;   
 const n1 = Math.floor(Math.random() \* max) + 1;  
 const n2 = Math.floor(Math.random() \* max) + 1;  
 const answer = n1 + n2;  
  
 // 2. Create UI  
 const display = document.createElement('div');  
 display.className = 'game-display';  
 display.textContent = `${n1} + ${n2} = ?`;  
  
 const optionsDiv = document.createElement('div');  
   
 // Generate 3 options (1 correct, 2 wrong)  
 const options = [answer, answer + 1, answer - 1].sort(() => Math.random() - 0.5);  
  
 options.forEach(opt => {  
 const btn = document.createElement('button');  
 btn.className = 'game-btn';  
 btn.textContent = opt;  
 btn.onclick = () => {  
 if (opt === answer) {  
 onComplete(true); // Success  
 } else {  
 onComplete(false); // Fail  
 }  
 };  
 optionsDiv.appendChild(btn);  
 });  
  
 container.appendChild(display);  
 container.appendChild(optionsDiv);  
 }  
};

### File: main.js

Location: /adventure-game/public/js/main.js

Purpose: The central Game Engine that connects API, UI, Map, and Minigames.

const Game = {  
 manifest: null,  
 session: null,  
 currentNodeFailures: 0,  
  
 init: async () => {  
 // 1. Load Assets  
 Game.manifest = await API.getManifest();  
 UIManager.populateConfig(Game.manifest);  
   
 // 2. Setup Event Listeners  
 document.getElementById('btn-start').addEventListener('click', Game.startGame);  
   
 // Listen for Map clicks  
 window.addEventListener('node-clicked', (e) => {  
 Game.handleNodeClick(e.detail.nodeId);  
 });  
 },  
  
 startGame: async () => {  
 const themeId = document.getElementById('config-theme').value;  
 const charId = document.getElementById('config-char').value;  
   
 // Create Session on Server  
 Game.session = await API.startSession('Player', charId, themeId);  
   
 // Switch Screen  
 UIManager.showScreen('game-container');  
   
 // Render Map  
 const theme = Game.manifest.themes.find(t => t.id === themeId);  
 MapRenderer.init('map-canvas');  
 MapRenderer.generateNodes(theme.type);  
   
 // Place Character at Start  
 const startNode = MapRenderer.getNodePos(0);  
 const char = Game.manifest.characters.find(c => c.id === charId);  
 UIManager.moveCharacter(startNode.x, startNode.y, char.color);  
   
 // Intro Message  
 UIManager.showFeedback("Let's Go!");  
 },  
  
 handleNodeClick: (nodeId) => {  
 // Rule: Can only click the CURRENT node (the active challenge)  
 if (nodeId !== Game.session.currentNode) return;  
  
 Game.launchChallenge();  
 },  
  
 launchChallenge: () => {  
 // In real app: pick game from manifest based on config.  
 // Demo: Always use "math\_add"  
 const gameId = 'math\_add';   
 const gameModule = window.GameLibrary[gameId];  
  
 if (!gameModule) {  
 console.error("Game module not loaded!");  
 return;  
 }  
  
 UIManager.openModal(gameModule.config.name);  
   
 // DIFFICULTY LOGIC  
 // Base difficulty = Level (1).   
 // If Mercy Mode is on, reduce difficulty by 1 (min 1).  
 let difficulty = Game.session.currentLevel;  
 if (Game.session.mercyMode) difficulty = Math.max(1, difficulty - 1);  
  
 const container = document.getElementById('minigame-container');  
   
 // Start the Minigame  
 gameModule.start(container, difficulty, (success) => {  
 Game.handleChallengeResult(success);  
 });  
 },  
  
 handleChallengeResult: async (success) => {  
 UIManager.closeModal();  
  
 if (success) {  
 // SUCCESS LOGIC  
 UIManager.showFeedback("Awesome! ⭐⭐⭐");  
   
 // Calculate Stars (Simplified: always 3 for win)  
 const stars = 3;  
   
 // Update Server  
 const res = await API.updateProgress(Game.session.sessionId, {  
 success: true, stars: stars  
 });  
 Game.session = res.state;  
 Game.currentNodeFailures = 0; // Reset fails  
  
 // Update Map  
 MapRenderer.updateNodeStatus(Game.session.currentNode - 1, 'completed');  
 if (Game.session.currentNode < 9) {  
 MapRenderer.updateNodeStatus(Game.session.currentNode, 'current');  
   
 // Move Character  
 const nextPos = MapRenderer.getNodePos(Game.session.currentNode);  
 const char = Game.manifest.characters.find(c => c.id === Game.session.characterId);  
 UIManager.moveCharacter(nextPos.x, nextPos.y, char.color);  
 UIManager.updateScore(Game.session.totalStars);  
 } else {  
 UIManager.showFeedback("VICTORY!", 5000);  
 }  
  
 } else {  
 // FAILURE LOGIC  
 Game.currentNodeFailures++;  
   
 if (Game.currentNodeFailures >= 3) {  
 // MERCY RULE TRIGGER  
 UIManager.showFeedback("Good Try! Let's move on.");  
   
 // Tell server we used mercy  
 const res = await API.updateProgress(Game.session.sessionId, {  
 success: false, stars: 0, usedMercy: true  
 });  
 Game.session = res.state;  
 Game.currentNodeFailures = 0;  
  
 // Move Character (same as win, but no stars)  
 MapRenderer.updateNodeStatus(Game.session.currentNode - 1, 'completed');  
 MapRenderer.updateNodeStatus(Game.session.currentNode, 'current');  
   
 const nextPos = MapRenderer.getNodePos(Game.session.currentNode);  
 const char = Game.manifest.characters.find(c => c.id === Game.session.characterId);  
 UIManager.moveCharacter(nextPos.x, nextPos.y, char.color);  
 } else {  
 // Simple Fail  
 UIManager.showFeedback("Try Again!");  
 }  
 }  
 }  
};  
  
// Start the Engine  
window.onload = Game.init;