Interactive Playground Environment Specification

Overview

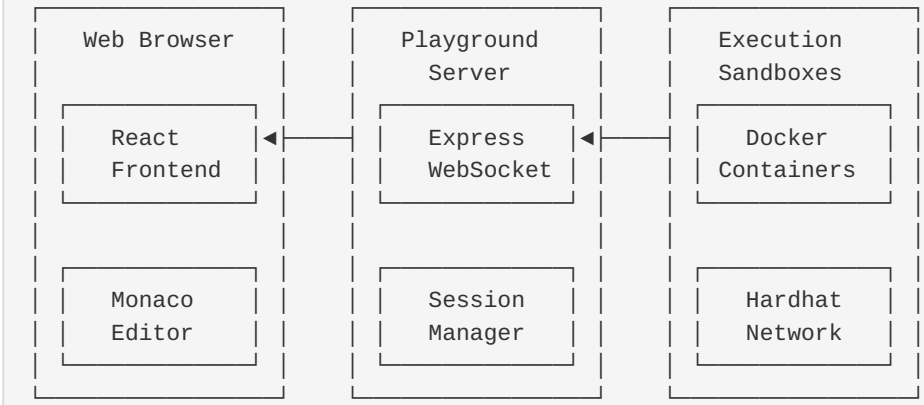
Create an interactive, browser-based playground environment where users can experiment with Audityzer features, test vulnerability patterns, and learn Web3 security concepts without local installation. This will serve as both an educational tool and a powerful demonstration platform.

Architecture Overview

Technology Stack

- **Frontend**: React + TypeScript + Vite
- **Backend**: Node.js + Express + WebSocket
- **Containerization**: Docker + Kubernetes
- **Code Execution**: Isolated sandboxes with resource limits
- **Real-time Communication**: Socket.io for live updates
- **State Management**: Zustand + React Query

System Architecture



User Interface Design

Main Playground Layout

```
const PlaygroundLayout = () => {
  return (
    <div className="h-screen flex flex-col bg-gray-900 text-white">
      {/* Header */}
      <PlaygroundHeader />

      {/* Main Content */}
      <div className="flex-1 flex">
        {/* Sidebar */}
        <div className="w-80 bg-gray-800 border-r border-gray-700">
          <PlaygroundSidebar />
        </div>

        {/* Editor Area */}
        <div className="flex-1 flex flex-col">
          <div className="flex-1 flex">
            {/* Code Editor */}
            <div className="flex-1">
              <CodeEditor />
            </div>

            {/* Results Panel */}
            <div className="w-96 border-l border-gray-700">
              <ResultsPanel />
            </div>
          </div>

          {/* Terminal */}
          <div className="h-48 border-t border-gray-700">
            <Terminal />
          </div>
        </div>
      </div>
    </div>
  );
};
```

Interactive Code Editor

```
const CodeEditor = () => {
  const [code, setCode] = useState(DEFAULT_CONTRACT);
  const [language, setLanguage] = useState('solidity');
  const { runAnalysis, isRunning } = usePlayground();

  return (
    <div className="h-full flex flex-col">
      {/* Editor Toolbar */}
      <div className="bg-gray-800 border-b border-gray-700 px-4 py-2 flex items-center
justify-between">
        <div className="flex items-center space-x-4">
          <select
            value={language}
            onChange={(e) => setLanguage(e.target.value)}
            className="bg-gray-700 text-white rounded px-3 py-1"
          >
            <option value="solidity">Solidity</option>
            <option value="javascript">JavaScript</option>
            <option value="typescript">TypeScript</option>
          </select>

          <div className="flex items-center space-x-2">
            <Button
              onClick={() => runAnalysis(code)}
              disabled={isRunning}
              className="bg-blue-600 hover:bg-blue-700"
            >
              {isRunning ? <SpinnerIcon /> : <PlayIcon />}
              Run Analysis
            </Button>

            <Button variant="outline" onClick={() => setCode(DEFAULT_CONTRACT)}>
              Reset
            </Button>
          </div>
        </div>

        <div className="flex items-center space-x-2">
          <ShareButton code={code} />
          <SaveButton code={code} />
        </div>
      </div>

      {/* Monaco Editor */}
      <div className="flex-1">
        <MonacoEditor
          language={language}
          value={code}
          onChange={setCode}
          theme="vs-dark"
          options={{
            minimap: { enabled: false },
            fontSize: 14,
            lineNumbers: 'on',
            roundedSelection: false,
            scrollBeyondLastLine: false,
            automaticLayout: true
          }}
```

```
        />
      </div>
    </div>
  );
};
```

Interactive Examples & Tutorials

Vulnerability Showcase

```
const VulnerabilityShowcase = () => {
  const vulnerabilities = [
    {
      id: 'reentrancy',
      title: 'Reentrancy Attack',
      description: 'Learn how reentrancy attacks work and how to prevent them',
      difficulty: 'Beginner',
      estimatedTime: '10 minutes',
      contract: REENTRANCY_VULNERABLE_CONTRACT,
      exploit: REENTRANCY_EXPLOIT_CODE,
      fix: REENTRANCY_FIXED_CONTRACT
    },
    {
      id: 'oracle-manipulation',
      title: 'Oracle Price Manipulation',
      description: 'Understand oracle attacks and price feed security',
      difficulty: 'Intermediate',
      estimatedTime: '15 minutes',
      contract: ORACLE_VULNERABLE_CONTRACT,
      exploit: ORACLE_EXPLOIT_CODE,
      fix: ORACLE_FIXED_CONTRACT
    },
    {
      id: 'flash-loan',
      title: 'Flash Loan Attack',
      description: 'Explore flash loan vulnerabilities and defenses',
      difficulty: 'Advanced',
      estimatedTime: '20 minutes',
      contract: FLASHLOAN_VULNERABLE_CONTRACT,
      exploit: FLASHLOAN_EXPLOIT_CODE,
      fix: FLASHLOAN_FIXED_CONTRACT
    }
  ];

  return (
    <div className="space-y-6">
      <div className="text-center">
        <h2 className="text-2xl font-bold text-white mb-2">
          Interactive Vulnerability Examples
        </h2>
        <p className="text-gray-400">
          Learn by doing - explore real vulnerability patterns and their fixes
        </p>
      </div>

      <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6">
        {vulnerabilities.map((vuln) => (
          <VulnerabilityCard
            key={vuln.id}
            vulnerability={vuln}
            onSelect={() => loadVulnerability(vuln)}
          />
        ))}
      </div>
    </div>
  );
};
```

```
const VulnerabilityCard = ({ vulnerability, onSelect }) => {
  return (
    <div className="bg-gray-800 border border-gray-700 rounded-lg p-6 hover:border-
blue-500 transition-colors cursor-pointer"
         onClick={onSelect}>
      <div className="flex items-start justify-between mb-4">
        <div className="flex items-center space-x-2">
          <VulnerabilityIcon type={vulnerability.id} />
          <h3 className="font-semibold text-white">{vulnerability.title}</h3>
        </div>
        <DifficultyBadge level={vulnerability.difficulty} />
      </div>

      <p className="text-gray-400 text-sm mb-4">
        {vulnerability.description}
      </p>

      <div className="flex items-center justify-between text-xs text-gray-500">
        <span> {vulnerability.estimatedTime}</span>
        <span> Interactive</span>
      </div>
    </div>
  );
};
```

Step-by-Step Tutorials

```
const InteractiveTutorial = ({ tutorialId }) => {
  const [currentStep, setCurrentStep] = useState(0);
  const [userCode, setUserCode] = useState('');
  const [stepResults, setStepResults] = useState({});

  const tutorial = useTutorial(tutorialId);
  const currentStepData = tutorial.steps[currentStep];

  const validateStep = async () => {
    const result = await validateUserCode(userCode, currentStepData.validation);
    setStepResults(prev => ({ ...prev, [currentStep]: result }));

    if (result.success) {
      setCurrentStep(prev => Math.min(prev + 1, tutorial.steps.length - 1));
    }
  };

  return (
    <div className="h-full flex">
      {/* Tutorial Steps Sidebar */}
      <div className="w-80 bg-gray-800 border-r border-gray-700 p-4">
        <div className="mb-6">
          <h2 className="text-xl font-bold text-white mb-2">{tutorial.title}</h2>
          <div className="flex items-center space-x-2 text-sm text-gray-400">
            <span>Step {currentStep + 1} of {tutorial.steps.length}</span>
            <div className="flex-1 bg-gray-700 rounded-full h-2">
              <div
                className="bg-blue-500 h-2 rounded-full transition-all"
                style={{ width: `${((currentStep + 1) / tutorial.steps.length) * 100}
%` }}
              />
            </div>
          </div>
        </div>

        <div className="space-y-4">
          {tutorial.steps.map((step, index) => (
            <TutorialStep
              key={index}
              step={step}
              index={index}
              current={index === currentStep}
              completed={stepResults[index]?.success}
              onClick={() => setCurrentStep(index)}
            />
          ))}
        </div>
      </div>

      {/* Main Tutorial Content */}
      <div className="flex-1 flex flex-col">
        {/* Step Instructions */}
        <div className="bg-gray-800 border-b border-gray-700 p-6">
          <h3 className="text-lg font-semibold text-white mb-2">
            {currentStepData.title}
          </h3>
          <div className="text-gray-300 prose prose-invert max-w-none">
            <ReactMarkdown>{currentStepData.instructions}</ReactMarkdown>
```

```
          </div>

          {currentStepData.hints && (
            <div className="mt-4">
              <HintAccordion hints={currentStepData.hints} />
            </div>
          )}
        </div>

        {/* Code Editor */}
        <div className="flex-1">
          <MonacoEditor
            language="solidity"
            value={userCode}
            onChange={setUserCode}
            theme="vs-dark"
            options={{
              minimap: { enabled: false },
              fontSize: 14,
              lineNumbers: 'on'
            }}
          />
        </div>

        {/* Step Actions */}
        <div className="bg-gray-800 border-t border-gray-700 p-4 flex items-center
justify-between">
          <div className="flex items-center space-x-4">
            <Button
              variant="outline"
              onClick={() => setUserCode(currentStepData.startingCode)}
            >
              Reset Code
            </Button>

            <Button
              onClick={validateStep}
              className="bg-green-600 hover:bg-green-700"
            >
              Check Solution
            </Button>
          </div>

          <div className="flex items-center space-x-2">
            <Button
              variant="outline"
              onClick={() => setCurrentStep(prev => Math.max(prev - 1, 0))}
              disabled={currentStep === 0}
            >
              Previous
            </Button>

            <Button
              onClick={() => setCurrentStep(prev => Math.min(prev + 1, tutori-
al.steps.length - 1))}
              disabled={currentStep === tutorial.steps.length - 1 || !stepRes-
ults[currentStep]?.success}
            >
              Next
```

```
            </Button>
          </div>
        </div>
      </div>
    </div>
  );
};
```

Backend Implementation

Sandbox Execution Engine

```
// src/server/sandbox/SandboxManager.js
class SandboxManager {
  constructor() {
    this.activeSandboxes = new Map();
    this.resourceLimits = {
      memory: '512m',
      cpu: '0.5',
      timeout: 30000,
      networkAccess: false
    };
  }

  async createSandbox(sessionId, config = {}) {
    const sandboxConfig = {
      ...this.resourceLimits,
      ...config,
      sessionId,
      image: 'auditzer-playground:latest'
    };

    const container = await this.docker.createContainer({
      Image: sandboxConfig.image,
      Cmd: ['/bin/bash'],
      WorkingDir: '/workspace',
      HostConfig: {
        Memory: this.parseMemory(sandboxConfig.memory),
        CpuQuota: this.parseCpu(sandboxConfig.cpu),
        NetworkMode: sandboxConfig.networkAccess ? 'bridge' : 'none',
        AutoRemove: true
      },
      Env: [
        'NODE_ENV=sandbox',
        'HARDHAT_NETWORK=localhost',
        `SESSION_ID=${sessionId}`
      ]
    });

    await container.start();

    const sandbox = new Sandbox(container, sandboxConfig);
    this.activeSandboxes.set(sessionId, sandbox);

    // Set up cleanup timer
    setTimeout(() => {
      this.destroySandbox(sessionId);
    }, sandboxConfig.timeout);

    return sandbox;
  }

  async executeCode(sessionId, code, language) {
    const sandbox = this.activeSandboxes.get(sessionId);
    if (!sandbox) {
      throw new Error('Sandbox not found');
    }

    return await sandbox.execute(code, language);
  }
```

```
  async runAudityzerAnalysis(sessionId, contractCode, config) {
    const sandbox = this.activeSandboxes.get(sessionId);
    if (!sandbox) {
      throw new Error('Sandbox not found');
    }

    // Write contract to sandbox
    await sandbox.writeFile('contract.sol', contractCode);

    // Write Audityzer config
    await sandbox.writeFile('audityzer.config.js', this.generateConfig(config));

    // Run Audityzer analysis
    const result = await sandbox.execute('audityzer run contract.sol --format json', 's
hell');

    return JSON.parse(result.stdout);
  }
}
```

Real-time Communication

```
// src/server/websocket/PlaygroundSocket.js
class PlaygroundSocket {
  constructor(io) {
    this.io = io;
    this.sandboxManager = new SandboxManager();
    this.setupEventHandlers();
  }

  setupEventHandlers() {
    this.io.on('connection', (socket) => {
      console.log(`User connected: ${socket.id}`);

      // Create sandbox for user
      socket.on('playground:init', async (config) => {
        try {
          const sandbox = await this.sandboxManager.createSandbox(socket.id, config);
          socket.emit('playground:ready', { sandboxId: sandbox.id });
        } catch (error) {
          socket.emit('playground:error', { message: error.message });
        }
      });

      // Execute code
      socket.on('playground:execute', async (data) => {
        try {
          const result = await this.sandboxManager.executeCode(
            socket.id,
            data.code,
            data.language
          );

          socket.emit('playground:result', result);
        } catch (error) {
          socket.emit('playground:error', { message: error.message });
        }
      });

      // Run Audityzer analysis
      socket.on('playground:analyze', async (data) => {
        try {
          socket.emit('playground:analyzing', { status: 'Starting analysis...' });

          const result = await this.sandboxManager.runAudityzerAnalysis(
            socket.id,
            data.contractCode,
            data.config
          );

          socket.emit('playground:analysis-complete', result);
        } catch (error) {
          socket.emit('playground:error', { message: error.message });
        }
      });

      // Handle disconnection
      socket.on('disconnect', () => {
        console.log(`User disconnected: ${socket.id}`);
        this.sandboxManager.destroySandbox(socket.id);
```

```
      });
    });
  }
}
```

Session Management

```
// src/server/session/SessionManager.js
class SessionManager {
  constructor() {
    this.sessions = new Map();
    this.sessionTimeout = 30 * 60 * 1000; // 30 minutes
  }

  createSession(userId = null) {
    const sessionId = this.generateSessionId();
    const session = {
      id: sessionId,
      userId,
      createdAt: new Date(),
      lastActivity: new Date(),
      code: '',
      results: [],
      tutorial: null,
      progress: {}
    };

    this.sessions.set(sessionId, session);

    // Set cleanup timer
    setTimeout(() => {
      this.cleanupSession(sessionId);
    }, this.sessionTimeout);

    return session;
  }

  updateSession(sessionId, updates) {
    const session = this.sessions.get(sessionId);
    if (!session) return null;

    Object.assign(session, updates, { lastActivity: new Date() });
    return session;
  }

  saveSessionState(sessionId, state) {
    const session = this.sessions.get(sessionId);
    if (!session) return false;

    session.code = state.code;
    session.results = state.results;
    session.tutorial = state.tutorial;
    session.progress = state.progress;
    session.lastActivity = new Date();

    return true;
  }

  getSession(sessionId) {
    return this.sessions.get(sessionId);
  }

  cleanupSession(sessionId) {
    const session = this.sessions.get(sessionId);
    if (!session) return;
```

```
    const timeSinceActivity = Date.now() - session.lastActivity.getTime();
    if (timeSinceActivity > this.sessionTimeout) {
      this.sessions.delete(sessionId);
      // Cleanup associated sandbox
      this.sandboxManager.destroySandbox(sessionId);
    }
  }
}
```

Educational Content System

Tutorial Content Structure

```
// src/content/tutorials/reentrancy-basics.js
export const reentrancyBasicsTutorial = {
  id: 'reentrancy-basics',
  title: 'Understanding Reentrancy Attacks',
  description: 'Learn how reentrancy attacks work and how to prevent them',
  difficulty: 'beginner',
  estimatedTime: '15 minutes',
  prerequisites: ['solidity-basics'],

  steps: [
    {
      id: 'intro',
      title: 'What is Reentrancy?',
      instructions: `
# Understanding Reentrancy

Reentrancy occurs when a function makes an external call to another untrusted contract
before resolving its own state changes. This allows the external contract to call back
into the original function before it completes.

## The Problem

Look at this vulnerable contract:

\`\`\`solidity
contract VulnerableBank {
    mapping(address => uint256) public balances;

    function withdraw(uint256 amount) public {
        require(balances[msg.sender] >= amount, "Insufficient balance");

        // External call before state change - VULNERABLE!
        (bool success, ) = msg.sender.call{value: amount}("");
        require(success, "Transfer failed");

        balances[msg.sender] -= amount; // State change happens after external call
    }
}
\`\`\`

**Your task**: Identify the vulnerability in this contract.
      `,
      startingCode: `// Analyze this contract and identify the reentrancy vulnerability
contract VulnerableBank {
    mapping(address => uint256) public balances;

    function deposit() public payable {
        balances[msg.sender] += msg.value;
    }

    function withdraw(uint256 amount) public {
        require(balances[msg.sender] >= amount, "Insufficient balance");

        // TODO: Identify the vulnerability here
        (bool success, ) = msg.sender.call{value: amount}("");
        require(success, "Transfer failed");

        balances[msg.sender] -= amount;
```

```
    }

    function getBalance(address user) public view returns (uint256) {
        return balances[user];
    }
}`,
      validation: {
        type: 'comment-analysis',
        expectedComments: [
          'external call before state change',
          'reentrancy vulnerability',
          'state change after external call'
        ]
      },
      hints: [
        'Look at the order of operations in the withdraw function',
        'What happens between the external call and the balance update?',
        'Could an attacker call withdraw again during the external call?'
      ]
    },

    {
      id: 'exploit',
      title: 'Creating the Attack',
      instructions: `
# Building a Reentrancy Attack

Now let's create an attacker contract that exploits the vulnerability.

The attack works by:
1. Depositing some funds into the vulnerable contract
2. Calling withdraw()
3. In the receive() function, calling withdraw() again before the first call completes
4. Repeating until the contract is drained

**Your task**: Complete the attacker contract below.
      `,
      startingCode: `contract ReentrancyAttacker {
    VulnerableBank public target;
    uint256 public attackAmount;

    constructor(address _target) {
        target = VulnerableBank(_target);
    }

    function attack() public payable {
        attackAmount = msg.value;

        // Step 1: Deposit funds
        target.deposit{value: attackAmount}();

        // Step 2: Start the attack
        // TODO: Call withdraw to start the reentrancy attack
    }

    // This function will be called when the target sends ETH
    receive() external payable {
        // TODO: Implement the reentrancy logic
        // Hint: Check if the target still has balance and call withdraw again
```

```
    }

    function getBalance() public view returns (uint256) {
        return address(this).balance;
    }
}`,
      validation: {
        type: 'code-execution',
        testCases: [
          {
            description: 'Attack function calls target.withdraw()',
            test: 'checkFunctionCall("attack", "target.withdraw")'
          },
          {
            description: 'Receive function implements reentrancy',
            test: 'checkReentrancyLogic("receive")'
          }
        ]
      }
    },

    {
      id: 'fix',
      title: 'Implementing the Fix',
      instructions: `
# Fixing the Reentrancy Vulnerability

There are several ways to fix reentrancy vulnerabilities:

1. **Checks-Effects-Interactions Pattern**: Update state before external calls
2. **Reentrancy Guard**: Use a mutex to prevent recursive calls
3. **Pull Payment Pattern**: Let users withdraw funds themselves

**Your task**: Fix the vulnerable contract using the checks-effects-interactions pat-
tern.
      `,
      startingCode: `contract SecureBank {
    mapping(address => uint256) public balances;

    function deposit() public payable {
        balances[msg.sender] += msg.value;
    }

    function withdraw(uint256 amount) public {
        // TODO: Implement the checks-effects-interactions pattern
        // 1. Checks: Verify conditions
        // 2. Effects: Update state
        // 3. Interactions: External calls
    }

    function getBalance(address user) public view returns (uint256) {
        return balances[user];
    }
}`,
      validation: {
        type: 'pattern-analysis',
        expectedPattern: 'checks-effects-interactions',
        requirements: [
          'require statement before state changes',
```

```
          'balance update before external call',
          'external call at the end'
        ]
      }
    }
  ]
};
```

Interactive Code Validation

```
// src/server/validation/CodeValidator.js
class CodeValidator {
  async validateStep(code, validation) {
    switch (validation.type) {
      case 'comment-analysis':
        return this.validateComments(code, validation.expectedComments);

      case 'code-execution':
        return this.validateExecution(code, validation.testCases);

      case 'pattern-analysis':
        return this.validatePattern(code, validation.expectedPattern);

      default:
        throw new Error(`Unknown validation type: ${validation.type}`);
    }
  }

  validateComments(code, expectedComments) {
    const comments = this.extractComments(code);
    const foundComments = expectedComments.filter(expected =>
      comments.some(comment =>
        comment.toLowerCase().includes(expected.toLowerCase())
      )
    );

    return {
      success: foundComments.length === expectedComments.length,
      score: foundComments.length / expectedComments.length,
      feedback: this.generateCommentFeedback(expectedComments, foundComments),
      foundComments,
      missingComments: expectedComments.filter(c => !foundComments.includes(c))
    };
  }

  async validateExecution(code, testCases) {
    const results = [];

    for (const testCase of testCases) {
      try {
        const result = await this.runTest(code, testCase.test);
        results.push({
          description: testCase.description,
          passed: result.success,
          output: result.output,
          error: result.error
        });
      } catch (error) {
        results.push({
          description: testCase.description,
          passed: false,
          error: error.message
        });
      }
    }

    const passedTests = results.filter(r => r.passed).length;
```

```
    return {
      success: passedTests === testCases.length,
      score: passedTests / testCases.length,
      results,
      feedback: this.generateExecutionFeedback(results)
    };
  }

  validatePattern(code, expectedPattern) {
    const patterns = {
      'checks-effects-interactions': this.validateCEIPattern,
      'reentrancy-guard': this.validateReentrancyGuard,
      'pull-payment': this.validatePullPayment
    };

    const validator = patterns[expectedPattern];
    if (!validator) {
      throw new Error(`Unknown pattern: ${expectedPattern}`);
    }

    return validator.call(this, code);
  }
}
```

Deployment & Scaling

Docker Configuration

```
# Dockerfile.playground
FROM node:18-alpine

# Install security tools
RUN apk add --no-cache \
    git \
    python3 \
    make \
    g++ \
    && npm install -g hardhat

# Create workspace
WORKDIR /workspace

# Install Adudityzer
RUN npm install -g adudityzer@latest

# Copy playground utilities
COPY playground-utils/ ./utils/

# Set up security restrictions
RUN adduser -D -s /bin/bash playground
USER playground

# Default command
CMD ["/bin/bash"]
```

Kubernetes Deployment

```yaml
# k8s/playground-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: adityzer-playground
spec:
  replicas: 3
  selector:
    matchLabels:
      app: adityzer-playground
  template:
    metadata:
      labels:
        app: adityzer-playground
    spec:
      containers:
      - name: playground-server
        image: adityzer/playground:latest
        ports:
        - containerPort: 3000
        env:
        - name: NODE_ENV
          value: "production"
        - name: REDIS_URL
          valueFrom:
            secretKeyRef:
              name: playground-secrets
              key: redis-url
        resources:
          requests:
            memory: "512Mi"
            cpu: "250m"
          limits:
            memory: "1Gi"
            cpu: "500m"
        livenessProbe:
          httpGet:
            path: /health
            port: 3000
          initialDelaySeconds: 30
          periodSeconds: 10
        readinessProbe:
          httpGet:
            path: /ready
            port: 3000
          initialDelaySeconds: 5
          periodSeconds: 5
```

Auto-scaling Configuration

```yaml
# k8s/playground-hpa.yaml
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: playground-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: audityzer-playground
  minReplicas: 2
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 70
  - type: Resource
    resource:
      name: memory
      target:
        type: Utilization
        averageUtilization: 80
```

Analytics & Monitoring

Usage Analytics

```
// src/server/analytics/PlaygroundAnalytics.js
class PlaygroundAnalytics {
  constructor() {
    this.events = [];
    this.sessions = new Map();
  }

  trackEvent(sessionId, event, data = {}) {
    const eventData = {
      sessionId,
      event,
      data,
      timestamp: new Date(),
      userAgent: data.userAgent,
      ip: data.ip
    };

    this.events.push(eventData);
    this.updateSessionMetrics(sessionId, event, data);
  }

  updateSessionMetrics(sessionId, event, data) {
    if (!this.sessions.has(sessionId)) {
      this.sessions.set(sessionId, {
        startTime: new Date(),
        events: [],
        codeExecutions: 0,
        tutorialProgress: {},
        vulnerabilitiesExplored: new Set()
      });
    }

    const session = this.sessions.get(sessionId);
    session.events.push({ event, data, timestamp: new Date() });

    switch (event) {
      case 'code_executed':
        session.codeExecutions++;
        break;
      case 'tutorial_step_completed':
        session.tutorialProgress[data.tutorialId] = data.stepIndex;
        break;
      case 'vulnerability_explored':
        session.vulnerabilitiesExplored.add(data.vulnerabilityType);
        break;
    }
  }

  generateReport() {
    const totalSessions = this.sessions.size;
    const totalEvents = this.events.length;
    const avgSessionDuration = this.calculateAvgSessionDuration();
    const popularVulnerabilities = this.getPopularVulnerabilities();
    const tutorialCompletionRates = this.getTutorialCompletionRates();

    return {
      totalSessions,
      totalEvents,
```

```
      avgSessionDuration,
      popularVulnerabilities,
      tutorialCompletionRates,
      userEngagement: this.calculateEngagementMetrics()
    };
  }
}
```

Success Metrics

Educational Impact

- **Tutorial Completion Rate**: Target 70%
- **Concept Retention**: Measure through follow-up quizzes
- **User Progression**: Track advancement through difficulty levels
- **Community Contributions**: User-generated content and examples

Technical Performance

- **Response Time**: < 2 seconds for code execution
- **Uptime**: 99.9% availability
- **Concurrent Users**: Support 1000+ simultaneous users
- **Resource Efficiency**: < 512MB RAM per sandbox

User Engagement

- **Session Duration**: Target 20+ minutes average
- **Return Rate**: 60% users return within 7 days
- **Sharing Rate**: 30% of sessions shared with others
- **Feedback Score**: 4.5/5 user satisfaction

This interactive playground will serve as a powerful educational tool and demonstration platform, making Web3 security concepts accessible to developers of all skill levels while showcasing Audityzer's capabilities.