EdGame – Technical Specification v2.0

Document Version: 2.0
Status: Architecture Definition (Phased)
Authors: EdGame Technical Team
Last Updated: February 2026

---

1. System Overview

1.1 Product Description

EdGame is a game-based educational assessment platform focused on **Math and Science** that:

- Delivers 3 polished game environments in Phase 1 (expanding to 25+ by Year 5)
- Captures 15 core behavioral metrics beyond traditional correctness
- Provides actionable analytics to teachers (and later parents/administrators)
- Supports web-first deployment with mobile responsiveness

1.2 Strategic Architecture Principle

Start simple, scale when needed. The original spec proposed enterprise-grade infrastructure from Day 1 (AWS EKS, Kafka, Flink, Spark). This is inappropriate for a 2-person team with zero users. Instead, we follow a phased approach:

Phase	Architecture	Trigger to Migrate
Phase 1	Supabase + Vercel + Phaser	Default
Phase 2	Add Redis caching, dedicated game servers	>5K concurrent, multiplayer launch
Phase 3	AWS/GCP, Kubernetes, Kafka	>50K concurrent, SLA requirements

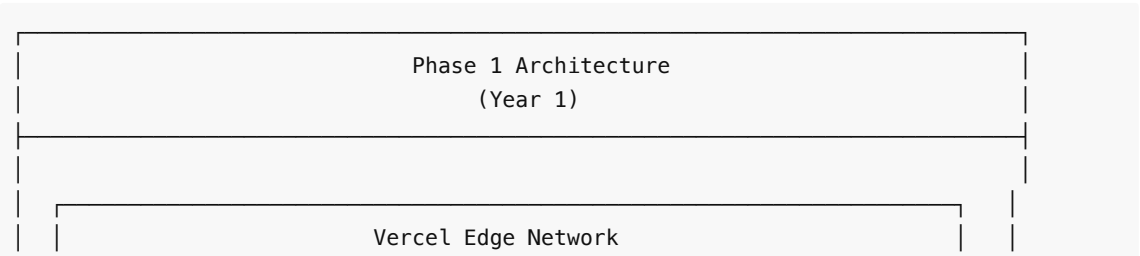
1.3 Design Principles

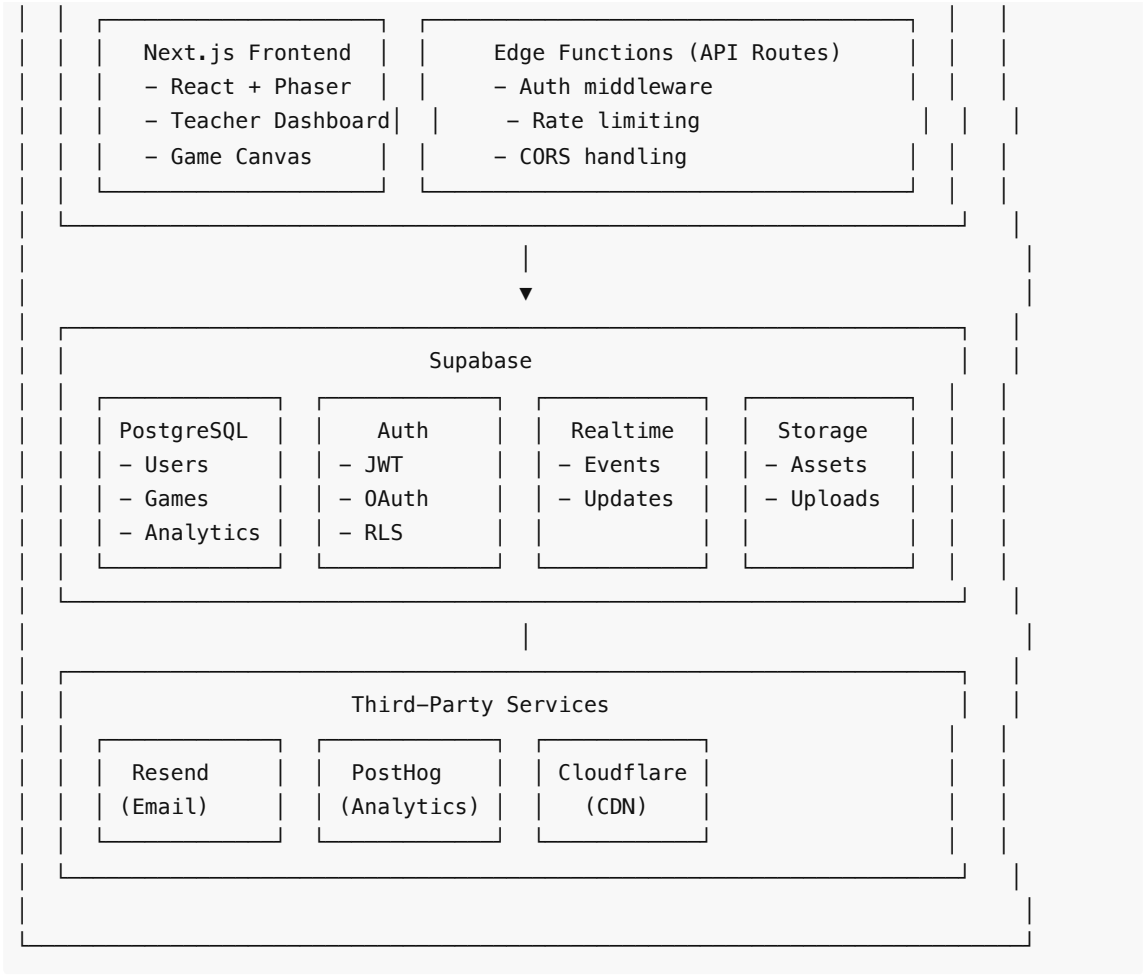
1. **Leverage managed services** — No self-hosted databases, auth systems, or real-time infrastructure until scale demands it
2. **Web-first, mobile-responsive** — Students access via browser; native apps are Phase 2
3. **Single-player reliability** — Multiplayer is Phase 2; ensure rock-solid single-player first
4. **Minimal viable analytics** — 15 metrics that drive teacher action, not 40+ metrics that overwhelm
5. **Monolith first** — No microservices until clear boundaries emerge from usage patterns

---

2. Phase 1 Architecture (Year 1)

2.1 Architecture Diagram





2.2 Technology Stack (Phase 1)

Layer	Technology	Justification
Frontend	Next.js 14+ (App Router)	SSR, API routes, excellent DX, Vercel-native
Game Engine	Phaser 3	Mature, large community, HTML5 Canvas/WebGL
Styling	Tailwind CSS + shadcn/ui	Rapid UI development, consistent design
Backend	Supabase (BaaS)	PostgreSQL, Auth, Realtime, Storage — all managed
Database	Supabase PostgreSQL	Row-Level Security, JSON support, full SQL
Auth	Supabase Auth	JWT, social OAuth, magic links, RBAC
Real-time	Supabase Realtime	Live dashboard updates, game state sync (light)
Hosting	Vercel	Edge deployment, automatic scaling, CI/CD
CDN	Vercel + Cloudflare	Asset delivery, DDoS protection
Email	Resend	Transactional email (assignments, reports)
Analytics	PostHog	Product analytics, event tracking, free tier

Error Tracking	Sentry	Exception monitoring, performance

2.3 Data Model (Phase 1)

```sql
-- Core user model with role-based access
CREATE TABLE users (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  email TEXT UNIQUE NOT NULL,
  role TEXT NOT NULL CHECK (role IN ('teacher', 'student', 'admin')),
  first_name TEXT,
  last_name TEXT,
  school_id UUID REFERENCES schools(id),
  created_at TIMESTAMPTZ DEFAULT now(),
  updated_at TIMESTAMPTZ DEFAULT now()
);

-- School/organization model
CREATE TABLE schools (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  name TEXT NOT NULL,
  type TEXT CHECK (type IN ('private', 'public', 'charter', 'international')),
  country TEXT,
  region TEXT,
  license_type TEXT CHECK (license_type IN ('trial', 'standard', 'premium')),
  license_expires_at DATE,
  created_at TIMESTAMPTZ DEFAULT now()
);

-- Game environments catalog
CREATE TABLE game_environments (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  slug TEXT UNIQUE NOT NULL,
  name TEXT NOT NULL,
  subject TEXT NOT NULL CHECK (subject IN ('math', 'science', 'language_arts')),
  grade_range INT4RANGE,
  description TEXT,
  thumbnail_url TEXT,
  config JSONB DEFAULT '{}',
  is_active BOOLEAN DEFAULT true,
  created_at TIMESTAMPTZ DEFAULT now()
);

-- Teacher-created assignments
CREATE TABLE assignments (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  teacher_id UUID NOT NULL REFERENCES users(id),
  environment_id UUID NOT NULL REFERENCES game_environments(id),
  class_id UUID REFERENCES classes(id),
  title TEXT NOT NULL,
  instructions TEXT,
```

```sql
  due_at TIMESTAMPTZ,
  config JSONB DEFAULT '{}',
  created_at TIMESTAMPTZ DEFAULT now()
);

-- Game sessions (raw event capture)
CREATE TABLE game_sessions (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  student_id UUID NOT NULL REFERENCES users(id),
  assignment_id UUID REFERENCES assignments(id),
  environment_id UUID NOT NULL REFERENCES game_environments(id),
  started_at TIMESTAMPTZ DEFAULT now(),
  ended_at TIMESTAMPTZ,
  duration_seconds INTEGER,
  completed BOOLEAN DEFAULT false,
  score NUMERIC,
  raw_events JSONB DEFAULT '[]',
  computed_metrics JSONB DEFAULT '{}'
);

-- Aggregated student metrics (daily rollup)
CREATE TABLE student_metrics_daily (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  student_id UUID NOT NULL REFERENCES users(id),
  date DATE NOT NULL,
  environment_id UUID REFERENCES game_environments(id),
  -- Engagement metrics
  total_time_minutes NUMERIC,
  session_count INTEGER,
  completion_rate NUMERIC,
  -- Performance metrics
  accuracy NUMERIC,
  avg_response_time_ms INTEGER,
  concepts_practiced TEXT[],
  concepts_mastered TEXT[],
  -- Behavioral metrics
  help_requests INTEGER,
  retry_count INTEGER,
  strategy_changes INTEGER,
  UNIQUE(student_id, date, environment_id)
);

-- Row-Level Security
ALTER TABLE game_sessions ENABLE ROW LEVEL SECURITY;

CREATE POLICY "Students see own sessions"
  ON game_sessions FOR SELECT
  USING (student_id = auth.uid());

CREATE POLICY "Teachers see class sessions"
  ON game_sessions FOR SELECT
  USING (
```

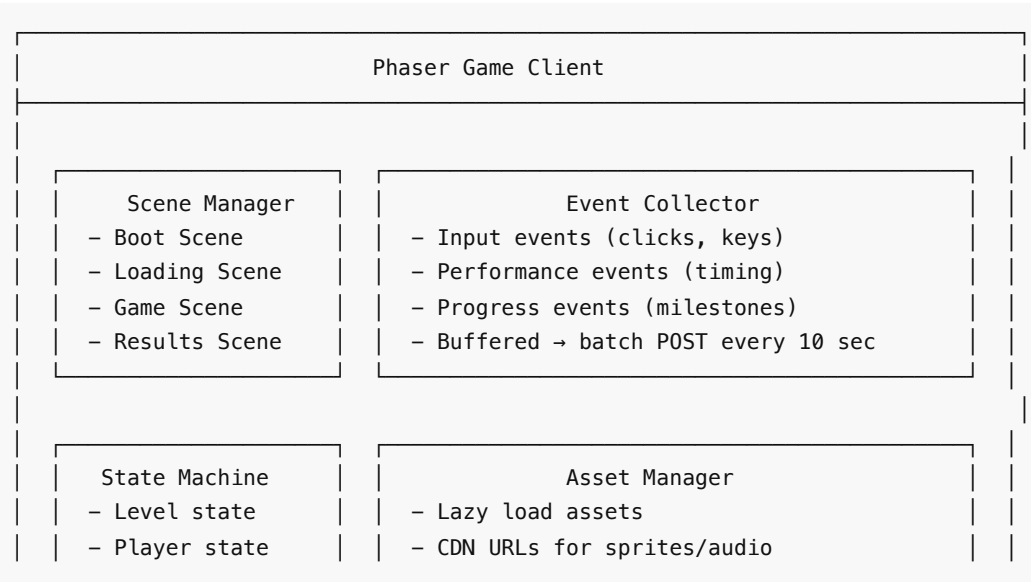
```
    EXISTS (
      SELECT 1 FROM assignments a
      JOIN classes c ON a.class_id = c.id
      WHERE a.id = game_sessions.assignment_id
      AND c.teacher_id = auth.uid()
    )
  );
```

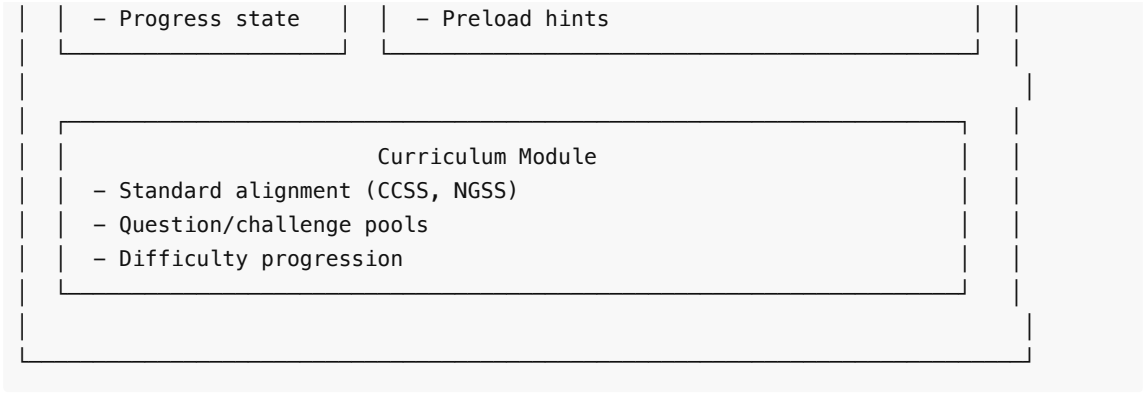
2.4 API Design (Phase 1)

Phase 1 uses Next.js API routes deployed on Vercel Edge Functions. All endpoints return JSON and use Supabase JWTs for authentication.

Endpoint	Method	Description	Auth
`/api/auth/[...supabase]`	*	Supabase Auth callbacks	Public
`/api/environments`	GET	List available game environments	Teacher+
`/api/assignments`	GET, POST	CRUD assignments	Teacher
`/api/assignments/[id]`	GET, PUT, DELETE	Single assignment	Teacher
`/api/sessions`	POST	Start new game session	Student
`/api/sessions/[id]`	PATCH	Update session (events, completion)	Student
`/api/sessions/[id]/metrics`	GET	Computed metrics for session	Teacher
`/api/analytics/class/[id]`	GET	Class-level analytics	Teacher
`/api/analytics/student/[id]`	GET	Individual student analytics	Teacher

2.5 Game Client Architecture





---

3. Phase 2 Architecture (Year 2)

3.1 Migration Triggers

Move to Phase 2 architecture when:

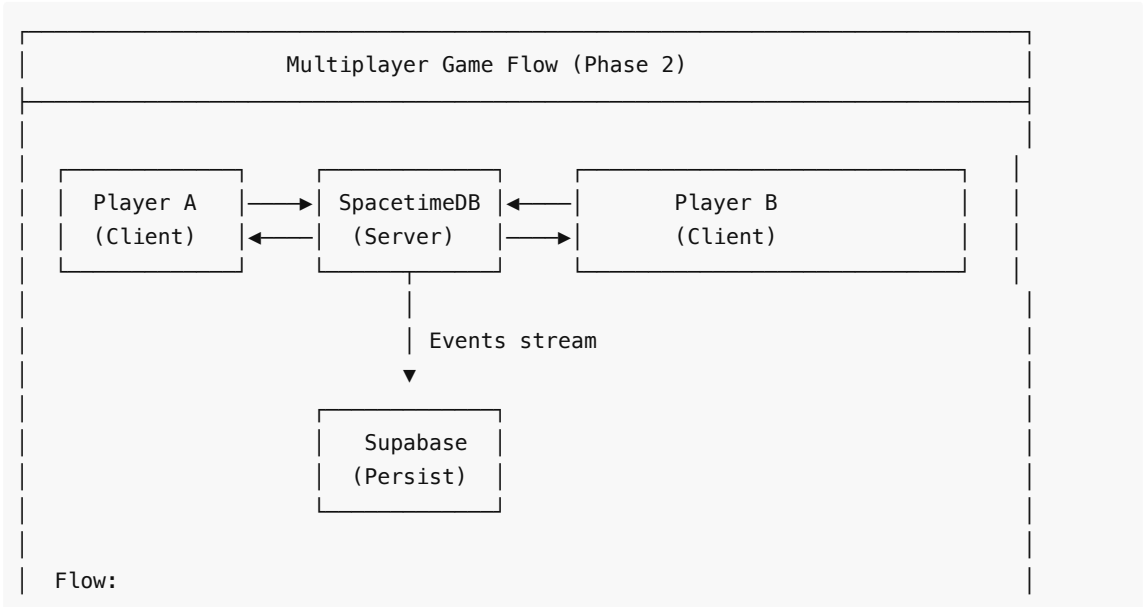
- **Concurrent users exceed 5,000** — Supabase connection limits
- **Multiplayer mode launches** — Need dedicated game server
- **Mobile apps ship** — Native apps require stable API contracts

3.2 Architecture Changes

```
Phase 1 → Phase 2 Additions:
──────────────────────────────────────────────────────────────────────────────

+ Upstash Redis         : Session caching, rate limiting, leaderboards
+ SpacetimeDB           : Real-time multiplayer state synchronization
+ Mobile Apps           : React Native or Flutter (reuse Phaser via WebView)
+ API Gateway           : Rate limiting, API versioning (Kong or AWS API GW)
+ Background Jobs       : Vercel Cron → Trigger.dev for analytics rollups
```

3.3 Multiplayer Architecture (Phase 2)



```
│  1. Teacher creates multiplayer assignment                                  │
│  2. Students join lobby (SpacetimeDB room)                                  │
│  3. Game state synced in real-time via SpacetimeDB                          │
│  4. Events buffered and persisted to Supabase on game end                   │
│  5. Analytics computed from Supabase data                                   │
│                                                                             │
└─────────────────────────────────────────────────────────────────────────────┘
```

---

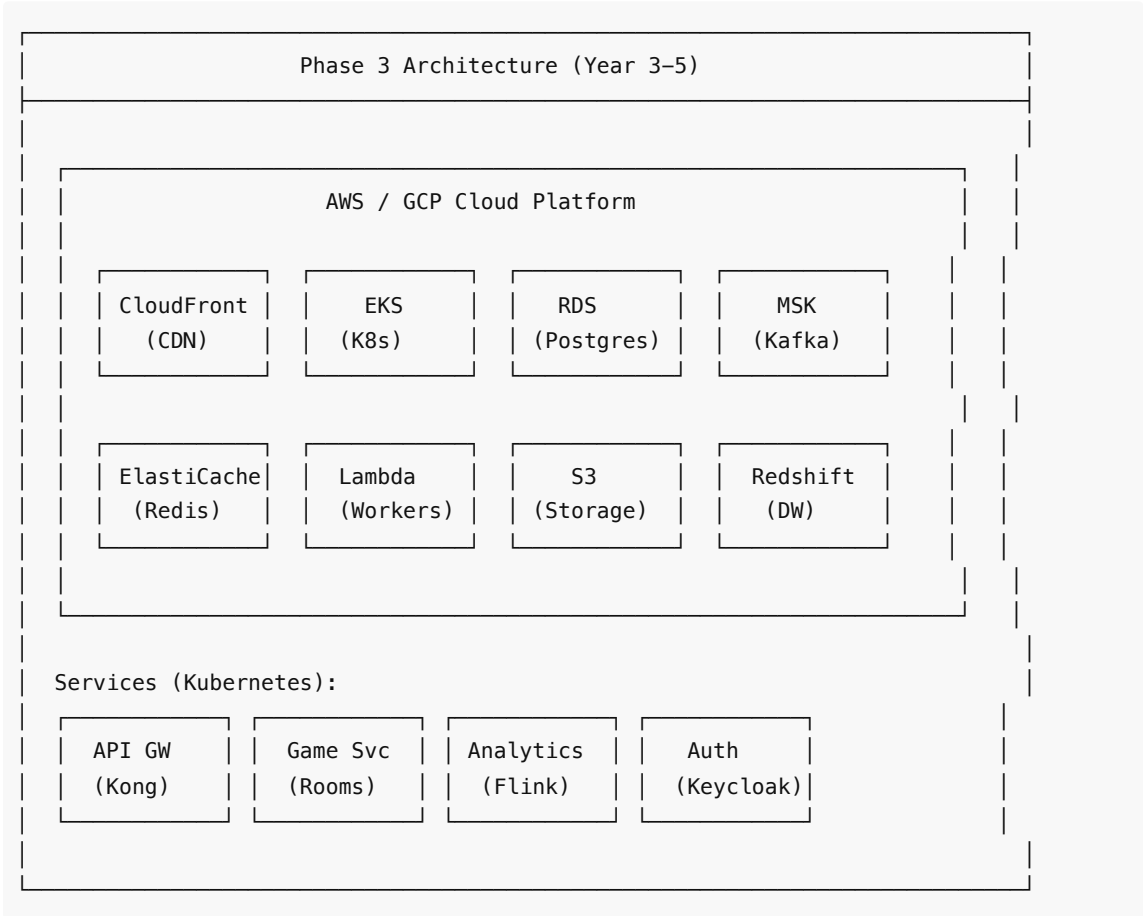
4. Phase 3 Architecture (Year 3-5)

4.1 Migration Triggers

Move to Phase 3 architecture when:

- **Concurrent users exceed 50,000** — Need horizontal scaling
- **Enterprise SLAs required** — 99.9% uptime commitments
- **Real-time analytics needed** — Streaming data pipelines
- **OEM API launches** — Third-party integrations at scale

4.2 Enterprise Architecture



4.3 Data Pipeline (Phase 3)

```
Game Events → Kafka → Flink (real-time) → Redis (hot)
                   └──────→ S3 (batch) ──→ Spark ──→ Redshift (analytics)
```

---

5. Behavioral Analytics Engine

5.1 Core 15 Metrics (Phase 1)

All metrics computed server-side from `raw_events` JSONB stored in `game_sessions`.

Category	Metric	Computation	Storage
Engagement	time_on_task	SUM(event_end - event_start) WHERE type='active'	Numeric (minutes)
Engagement	session_frequency	COUNT(DISTINCT date) per week	Integer
Engagement	completion_rate	completed_tasks / assigned_tasks	Numeric (0-1)
Engagement	voluntary_replay	COUNT sessions WHERE assignment_id IS NULL	Integer
Engagement	drop_off_point	LAST checkpoint WHERE completed=false	Text
Performance	accuracy_by_concept	correct / total per concept_id	JSONB
Performance	speed_accuracy_tradeoff	correlation(response_time, accuracy)	Numeric
Performance	error_pattern	MODE(error_type) per concept	JSONB
Performance	improvement_trajectory	REGR_SLOPE(accuracy, session_number)	Numeric
Performance	mastery_threshold	concepts WHERE accuracy > 0.8 for 3+ sessions	Array
Behavioral	help_seeking	COUNT events WHERE type='hint_request'	Integer
Behavioral	persistence	COUNT events WHERE type='retry' after failure	Integer
Behavioral	strategy_variation	COUNT DISTINCT action_patterns per session	Integer
Behavioral	response_to_feedback	accuracy_after_feedback / accuracy_before	Numeric
Behavioral	collaboration_style	TBD - Phase 2 multiplayer	Enum

5.2 Metric Computation Flow

```
1. Client captures raw events (clicks, answers, timing)
      ↓
2. Events batched and POSTed every 10 seconds
      ↓
3. Stored in game_sessions.raw_events (JSONB)
      ↓
4. On session end: compute metrics via PostgreSQL function
      ↓
5. Store in game_sessions.computed_metrics + daily rollup
      ↓
6. Teacher dashboard reads aggregated views
```

5.3 Teacher Dashboard Insights

Instead of showing all 15 metrics, the dashboard shows **"Top 3 Insights"** per student:

```typescript
interface StudentInsight {
  type: 'struggling' | 'excelling' | 'needs_attention' | 'improving';
  metric: string;
  message: string;
  action: string;
}

// Example output:
[
  {
    type: 'struggling',
    metric: 'accuracy_by_concept',
    message: 'Sarah is struggling with fractions (42% accuracy)',
    action: 'Consider reteaching equivalent fractions before proceeding'
  },
  {
    type: 'improving',
    metric: 'improvement_trajectory',
    message: 'Marcus improved 25% in algebra this week',
    action: 'Good candidate for peer tutoring role'
  },
  {
    type: 'needs_attention',
    metric: 'session_frequency',
    message: 'Emma hasn\'t logged in for 5 days',
    action: 'Check in about barriers to access'
  }
]
```

---

6. Security & Compliance

6.1 Compliance Requirements

Regulation	Applicability	Key Requirements

FERPA	US schools	Parental consent for <18, data access rights, no marketing
COPPA	US students <13	Verifiable parental consent, data minimization
GDPR	EU/UK students	Consent, right to erasure, data portability
UAE PDPL	UAE schools	Similar to GDPR, data localization preferences

6.2 Security Architecture

Layer	Implementation
Transport	HTTPS everywhere (TLS 1.3), HSTS
Authentication	Supabase Auth (JWT), OAuth 2.0, magic links
Authorization	Row-Level Security in PostgreSQL, RBAC
Data at rest	Supabase encryption (AES-256)
Secrets	Environment variables in Vercel, Supabase Vault
Logging	Audit logs for data access, retained 1 year

6.3 Data Handling

```
Student PII: Stored in users table, encrypted
Behavioral data: De-identified for analytics, linked via user_id
Parental consent: Required for COPPA, tracked in consents table
Data retention: 3 years active, 1 year archived, then deleted
Data export: GDPR/CCPA export via admin API
Data deletion: Cascade delete via admin API
```

---

7. Integrations

7.1 Phase 1 Integrations

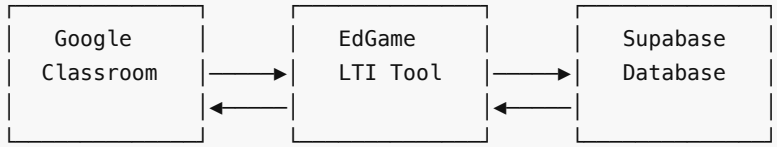
Integration	Priority	Implementation
Google Classroom	P0	LTI 1.3 for assignments, OAuth for roster import
Clever	P1	SSO, roster sync (US schools)
Google OAuth	P0	Teacher/student login
Microsoft OAuth	P1	Enterprise schools

7.2 Phase 2+ Integrations

Integration	Phase	Implementation
Canvas/Schoology	P2	LTI 1.3

ClassDojo	P2	Parent notification API
Stripe	P2	Payment processing for B2C
Firebase	P2	Push notifications for mobile

7.3 LTI 1.3 Implementation




```
Flow:
1. Teacher clicks "Add EdGame assignment" in Google Classroom
2. LTI launch request sent to EdGame
3. EdGame validates JWT, creates/links user
4. Assignment created in EdGame, linked to Classroom
5. Grades sent back via LTI Assignment and Grade Services
```

---

8. Performance Targets

8.1 Phase 1 Targets

Metric	Target	Measurement
Page load (dashboard)	< 2s	Lighthouse, p95
Game start	< 3s	Asset load + init
API response	< 200ms	p95
Concurrent users	2,000	Load testing
Availability	99.5%	Uptime (excl. maintenance)

8.2 Phase 3 Targets

Metric	Target	Measurement
Page load	< 1.5s	CDN + edge
Game start	< 2s	Optimized assets
API response	< 100ms	p99
Concurrent users	100,000	Auto-scaling
Availability	99.9%	SLA

---

9. Development & Deployment

9.1 Development Setup

```
# Prerequisites: Node.js 20+, pnpm
git clone https://github.com/edgame/edgame
cd edgame
pnpm install

# Environment
cp .env.example .env.local
# Fill in Supabase URL, keys, etc.

# Local development
pnpm dev          # Next.js dev server
pnpm db:studio    # Supabase Studio (local)
pnpm game:dev     # Phaser dev with hot reload
```

9.2 CI/CD Pipeline

```
Push to main → GitHub Actions:
  1. Lint + Type check (2 min)
  2. Unit tests (3 min)
  3. Build (2 min)
  4. Deploy preview to Vercel (automatic)
  5. E2E tests on preview (5 min)
  6. Production deploy (if tests pass)

Branch protection:
- Require PR review
- Require CI pass
- No direct push to main
```

9.3 Monitoring & Observability

Tool	Purpose	Phase
Vercel Analytics	Web vitals, traffic	P1
PostHog	Product analytics, funnels	P1
Sentry	Error tracking, performance	P1
Supabase Dashboard	Database metrics, connections	P1
Grafana + Prometheus	Custom metrics, alerting	P3

---

10. Game Environment Specifications

10.1 Environment: Math Challenge Arena

Attribute	Value
Subject	Mathematics
Grades	4-8
Format	Single-player challenge mode
Duration	10-20 minutes
Standards	CCSS Math

Gameplay: Students solve progressively harder math problems in an arena setting. Correct answers advance them; incorrect answers provide feedback. Speed and accuracy both contribute to score.

Metrics captured: accuracy, response_time, error_patterns, hint_usage, persistence

10.2 Environment: Virtual Chemistry Lab

Attribute	Value
Subject	Science (Chemistry)
Grades	6-10
Format	Simulation / sandbox
Duration	15-30 minutes
Standards	NGSS

Gameplay: Students conduct virtual experiments, mix chemicals, observe reactions, and document findings. Open-ended exploration with guided challenges.

Metrics captured: exploration_breadth, hypothesis_testing, safety_violations, procedure_accuracy, scientific_reasoning

10.3 Environment: Physics Simulation

Attribute	Value
Subject	Science (Physics)
Grades	7-12
Format	Puzzle / simulation
Duration	15-25 minutes
Standards	NGSS

Gameplay: Students manipulate variables (mass, force, angle) to solve physics puzzles. Projectile motion, simple machines, energy transfer.

Metrics captured: trial_count, variable_manipulation, prediction_accuracy, concept_transfer, iteration_speed

---

This specification defines a pragmatic, phased approach to building EdGame. Phase 1 is deliberately simple — Supabase + Vercel + Phaser can be built by a 2-person team with AI tools. Complexity is added only when usage patterns demand it, not because "enterprise architecture" sounds impressive.