Technical Architecture Document (TAD)

Product

learntoaction – Multi-tenant Learning Execution Platform

---

1. Architecture Goals

Primary Goals

- True multi-tenancy with strict workspace isolation
- Horizontally scalable services
- Low-latency student experience
- High write-throughput for autosave & analytics
- Extensible architecture for future AI features

Non-Goals

- Real-time collaboration (v1)
- Native mobile apps (v1)
- Video hosting infrastructure

---

2. High-Level System Architecture

Logical Layers

1. **Client Applications**
2. Teacher Editor App
3. Student Runner App
4. Super Admin Console
5. **API & Gateway Layer**
6. Authentication
7. Authorization (RBAC)
8. Workspace resolution
9. Domain & slug routing
10. **Core Domain Services**
11. Workspace Service

12. Content Service (Worksheets & Workbooks)
13. Field Service
14. Response Service
15. Student Service
16. Analytics Event Service

17. Subscription & Billing Service

18. **Data & Infrastructure Layer**

19. Relational Database (PostgreSQL)
20. Object Storage (S3-compatible)
21. Event Stream / Queue
22. Cache (Redis)

---

3. Deployment Model

Cloud Strategy

- Cloud-agnostic (AWS-first)
- Containerized services (Docker)
- Orchestrated via Kubernetes (EKS)

Environments

- Production
- Staging
- Development

Each environment is isolated at network and data level.

---

4. Frontend Architecture

4.1 Teacher Editor App

Purpose - Content creation and management

Stack - React - TypeScript - Slate / ProseMirror (rich text engine) - Drag-and-drop grid layout system

Key Responsibilities - Worksheet & workbook editing - Block configuration - Field generation & validation - Autosave orchestration - Responsive previews

---

4.2 Student Runner App

Purpose - Content consumption and response capture

Stack - React (read-optimized) - Server-side rendering (optional for SEO)

Key Responsibilities - Render published worksheets/workbooks - Persist responses - Resume progress - Emit analytics events

---

4.3 Super Admin Console

Purpose - SaaS owner operations

Responsibilities - Tenant monitoring - Platform analytics - Moderation - Plan enforcement

---

5. Backend Architecture

5.1 API Gateway

Responsibilities - Auth validation (JWT / session tokens) - Role-based access control - Workspace context resolution - Rate limiting

---

5.2 Core Services

Workspace Service

- Workspace lifecycle
- Branding configuration
- Domain mappings

Content Service

- Worksheet CRUD
- Workbook CRUD
- Folder hierarchy
- Draft / publish state

Field Service

- Field generation
- Uniqueness enforcement
- Field usage tracking

Response Service

- Autosave responses
- Versioning
- Privacy flags

Student Service

- Student identity
- Enrollment tracking
- Activity status

Analytics Event Service

- Event ingestion
- Schema validation
- Asynchronous processing

---

6. Domain & Routing Architecture

Domain Resolution Flow

1. Incoming request
2. Resolve domain → workspace
3. Resolve slug → content (worksheet/workbook)
4. Apply branding
5. Serve runner app

SSL Handling

- Automatic certificate provisioning
- Managed via cloud certificate service

---

7. Data Architecture

7.1 Multi-Tenancy Strategy

- Single database
- Shared schema
- `workspace_id` on all tenant-owned tables
- Enforced via:
- Application-level guards
- Database indexes

7.2 Core Database Schema (Logical)

workspaces

- id (PK)
- owner_user_id
- brand_settings (JSONB)
- created_at

users

- id
- email
- role

worksheets

- id
- workspace_id
- title
- slug
- status (draft/published)
- schema (JSONB)
- created_at

workbooks

- id
- workspace_id
- title
- slug
- created_at

workbook_pages

- id
- workbook_id
- worksheet_snapshot (JSONB)
- page_order

fields

- id
- workspace_id
- field_name
- field_key
- created_at

students

- id
- workspace_id
- email
- name
- last_active_at

responses

- id
- worksheet_id
- student_id
- data (JSONB)
- progress
- is_shared
- updated_at

analytics_events

- id
- workspace_id
- event_type
- payload (JSONB)
- created_at

---

8. Analytics Architecture

Event-Based Model

- Every interaction emits an event
- Events are immutable
- Analytics dashboards are projections

Event Examples

- worksheet_viewed
- field_focused
- response_saved
- worksheet_completed

Processing

- Ingest → queue → worker processing → aggregates

---

9. Autosave & Versioning

Strategy

- Client-side debounce
- Partial saves
- Idempotent writes

Version Control

- Store last N versions per response
- Recoverable state

---

10. Scalability Considerations

Horizontal Scaling

- Stateless API services
- Auto-scaling via CPU / queue depth

Database Scaling

- Read replicas
- Partition analytics tables by time

Caching

- Redis for:
- Session data
- Domain resolution
- Slug resolution

---

11. Security & Compliance

- HTTPS everywhere
- JWT-based auth
- Role-based permissions
- GDPR-compliant deletion
- Audit logs

---

12. Observability

- Centralized logging

- Metrics (latency, error rate)
- Alerting on anomalies

---

13. Failure & Recovery

- Graceful autosave retry
- Circuit breakers
- Backup & restore strategy

---

14. Future Architecture Extensions

- AI inference service
- Feature flag service
- Enterprise SSO
- White-label clusters

---

15. Architecture Principle

Design for thinking latency, not streaming latency.

Learners pause, reflect, and return—architecture must preserve state reliably over time.