DIL-LMS Technical Architecture Documentation

System Overview

DIL-LMS (Digital Interactive Learning - Learning Management System) is a comprehensive educational platform built with modern web technologies. The system supports multi-role access (Students, Teachers, Admins), AI-powered learning features, and multi-tenancy architecture.

Technology Stack

- **Frontend**: React 18 + TypeScript + Vite
- **UI Framework**: Radix UI + Tailwind CSS + shadcn/ui
- **Backend**: Supabase (PostgreSQL + Edge Functions)
- **Authentication**: Supabase Auth with MFA support
- **AI Integration**: OpenAI GPT-4 + Custom AI Assistants (IRIS, APEX)
- **Real-time Features**: Supabase Realtime + WebSockets
- **File Storage**: Supabase Storage
- **Notifications**: Firebase Cloud Messaging
- **State Management**: React Query + Context API

System Architecture

```
graph TB
    subgraph "Client Layer"
        WEB[Web Application
React + TypeScript]
        MOBILE[Mobile App
PWA Support]
    end

    subgraph "API Gateway"
        SUPABASE[Supabase
API Gateway]
        EDGE[Edge Functions
Serverless]
    end

    subgraph "Authentication"
        AUTH[Supabase Auth
JWT + MFA]
        SESSION[Session Management
RLS Policies]
    end

    subgraph "Core Services"
        DB[(PostgreSQL
Database)]
        STORAGE[File Storage
```

```
Supabase Storage]
        REALTIME[Real-time
Subscriptions]
    end

    subgraph "AI Services"
        OPENAI[OpenAI GPT-4
API]
        IRIS[IRIS Assistant
Analytics AI]
        APEX[APEX Assistant
Support AI]
        MCP[MCP Adapter
Database Tools]
    end

    subgraph "External Services"
        FCM[Firebase
Cloud Messaging]
        EMAIL[Email Service
SMTP]
    end

    WEB --> SUPABASE
    MOBILE --> SUPABASE
    SUPABASE --> AUTH
    SUPABASE --> DB
    SUPABASE --> STORAGE
    SUPABASE --> REALTIME
    EDGE --> OPENAI
    EDGE --> MCP
    IRIS --> MCP
    APEX --> DB
    SUPABASE --> FCM
    SUPABASE --> EMAIL
```

Database Schema Overview

Core Tables

```
erDiagram
    profiles ||--o{ course_members : enrolls
    profiles ||--o{ class_students : "enrolled in"
    profiles ||--o{ class_teachers : teaches
    profiles ||--o{ user_sessions : has

    courses ||--o{ course_members : contains
    courses ||--o{ assignments : has
    courses ||--o{ quizzes : includes

    classes ||--o{ class_students : contains
```

```
    classes ||--o{ class_teachers : "taught by"
    classes }o--|| schools : "belongs to"
    classes }o--|| boards : "follows"

    schools }o--|| cities : "located in"
    cities }o--|| regions : "part of"
    regions }o--|| countries : "within"

    profiles {
        uuid id PK
        string email
        string full_name
        string role
        string phone_number
        string timezone
        string avatar_url
        timestamp created_at
    }

    courses {
        uuid id PK
        string title
        text description
        string status
        uuid school_id FK
        uuid board_id FK
        json metadata
        timestamp created_at
    }

    classes {
        uuid id PK
        string name
        string code
        string grade
        uuid school_id FK
        uuid board_id FK
        string academic_year
        integer max_students
        integer current_students
    }

    schools {
        uuid id PK
        string name
        string code
        string school_type
        uuid city_id FK
        uuid board_id FK
        string status
    }
```

AI and Analytics Tables

```
erDiagram
    ai_prompts ||--o{ iris_chat_logs : uses
    apex_faqs ||--o{ apex_knowledge_base : relates
    ai_tutor_user_progress_summary ||--o{ profiles : tracks

    ai_prompts {
        uuid id PK
        string name
        string role
        text content
        integer version
        boolean is_active
        integer usage_count
    }

    iris_chat_logs {
        uuid id PK
        uuid user_id FK
        string user_role
        text query
        text response_preview
        string[] tools_used
        integer tokens_used
        boolean success
    }

    apex_faqs {
        uuid id PK
        text question
        text answer
        string category
        string[] tags
        string priority
        boolean is_active
    }

    ai_tutor_user_progress_summary {
        uuid id PK
        uuid user_id FK
        string stage
        integer total_sessions
        integer completed_exercises
        float average_score
        timestamp last_activity
    }
```

Key User Flows

1. User Authentication Flow

```
sequenceDiagram
    participant U as User
    participant C as Client App
    participant SA as Supabase Auth
    participant DB as Database
    participant MFA as MFA Service

    U->>C: Enter credentials
    C->>SA: signInWithPassword()
    SA->>DB: Validate user
    DB-->>SA: User data

    alt MFA Required
        SA-->>C: MFA Challenge
        C-->>U: Request MFA code
        U->>C: Enter MFA code
        C->>MFA: Verify MFA
        MFA-->>SA: MFA verified
    end

    SA-->>C: JWT Token + Session
    C->>DB: Create session record
    C->>DB: Log access event
    C-->>U: Redirect to Dashboard

    Note over C,DB: Session tracked with RLS policies
```

2. AI Assistant (IRIS) Query Flow

```
sequenceDiagram
    participant U as User
    participant C as Client
    participant EF as Edge Function
    participant AI as OpenAI API
    participant MCP as MCP Adapter
    participant DB as Database

    U->>C: Ask question about analytics
    C->>EF: POST /iris-chat
    Note over EF: iris-chat Edge Function

    EF->>MCP: GET /tools (fetch available tools)
    MCP-->>EF: Database tools list

    EF->>AI: Chat completion with tools
    AI-->>EF: Response with tool calls

    loop For each tool call
        EF->>MCP: POST /invoke (execute SQL)
        MCP->>DB: Execute query with RLS
```

```
        DB-->>MCP: Query results
        MCP-->>EF: Formatted results
    end

    EF->>AI: Final completion with results
    AI-->>EF: Natural language response

    EF->>DB: Log interaction
    EF-->>C: AI response with insights
    C-->>U: Display formatted response
```

3. Course Enrollment Flow

```
sequenceDiagram
    participant S as Student
    participant C as Client
    participant DB as Database
    participant T as Teacher
    participant N as Notification Service

    S->>C: Browse available courses
    C->>DB: Query courses with RLS
    DB-->>C: Available courses list
    C-->>S: Display courses

    S->>C: Click "Enroll" on course
    C->>DB: Check enrollment capacity
    DB-->>C: Capacity available

    C->>DB: Insert course_member record
    DB->>DB: Update course enrollment count
    DB->>DB: Trigger enrollment notification

    DB-->>N: Send notification to teacher
    N->>T: New student enrolled notification

    C-->>S: Enrollment successful

    Note over DB: RLS ensures student can only enroll themselves
```

4. Assignment Submission Flow

```
sequenceDiagram
    participant S as Student
    participant C as Client
    participant ST as Supabase Storage
    participant DB as Database
    participant T as Teacher
    participant AI as AI Grading
```

```
    S->>C: Upload assignment file
    C->>ST: Upload file to storage
    ST-->>C: File URL

    C->>DB: Create submission record
    DB->>DB: Update assignment status

    alt Auto-grading enabled
        DB->>AI: Trigger AI grading
        AI->>ST: Download and analyze file
        AI-->>DB: Store AI feedback
    end

    DB-->>C: Submission recorded
    C-->>S: Submission successful

    DB->>T: Notify teacher of submission

    Note over DB: RLS ensures student can only submit to their courses
```

5. AI Learning Session Flow

```
sequenceDiagram
    participant S as Student
    participant C as Client
    participant AI as AI Tutor
    participant STT as Speech-to-Text
    participant TTS as Text-to-Speech
    participant DB as Database

    S->>C: Start AI learning session
    C->>DB: Get user progress & stage
    DB-->>C: Current learning stage

    C->>AI: Initialize session with stage
    AI-->>C: Learning content & prompts

    loop Learning Interaction
        C-->>S: Present exercise
        S->>C: Voice/text response

        alt Voice Response
            C->>STT: Convert speech to text
            STT-->>C: Transcribed text
        end

        C->>AI: Evaluate response
        AI-->>C: Feedback & next step

        alt AI Response includes audio
```

```
            C->>TTS: Generate speech
            TTS-->>C: Audio response
        end

        C-->>S: Present feedback
    end

    C->>DB: Update progress summary
    C->>DB: Log session analytics
    DB-->>C: Progress updated

    C-->>S: Session complete with progress
```

6. Admin Analytics Dashboard Flow

```
sequenceDiagram
    participant A as Admin
    participant C as Client
    participant RS as Reports Service
    participant DB as Database
    participant CACHE as Cache Layer

    A->>C: Access analytics dashboard
    C->>RS: getAllReportsData()

    RS->>CACHE: Check cached data
    alt Cache Hit
        CACHE-->>RS: Return cached data
    else Cache Miss
        par Parallel Data Fetch
            RS->>DB: Practice stage performance
            RS->>DB: User engagement metrics
            RS->>DB: Time usage patterns
            RS->>DB: Top content accessed
            RS->>DB: Analytics overview
        end

        DB-->>RS: All analytics data
        RS->>CACHE: Store in cache
    end

    RS-->>C: Formatted analytics data
    C-->>A: Render dashboard with charts

    Note over RS,CACHE: 60-second cache duration
    Note over DB: Complex aggregation queries with RLS
```

7. Multi-Factor Authentication Setup Flow

```
sequenceDiagram
    participant U as User
    participant C as Client
    participant SA as Supabase Auth
    participant MFA as MFA Service
    participant DB as Database

    U->>C: Enable MFA in settings
    C->>SA: enroll() MFA
    SA->>MFA: Generate TOTP secret
    MFA-->>SA: Secret + QR code
    SA-->>C: QR code for authenticator

    C-->>U: Display QR code
    U->>U: Scan with authenticator app
    U->>C: Enter verification code

    C->>SA: challenge() with code
    SA->>MFA: Verify TOTP code
    MFA-->>SA: Verification result

    alt Verification Success
        SA-->>C: MFA enrolled successfully
        C->>DB: Update user MFA status
        C->>DB: Generate backup codes
        C-->>U: Show backup codes
    else Verification Failed
        SA-->>C: Invalid code error
        C-->>U: Request new code
    end

    Note over DB: MFA status tracked in profiles table
```

8. Real-time Notification Flow

```
sequenceDiagram
    participant T as Teacher
    participant C1 as Teacher Client
    participant DB as Database
    participant RT as Realtime Service
    participant FCM as Firebase Messaging
    participant C2 as Student Client
    participant S as Student

    T->>C1: Create new assignment
    C1->>DB: Insert assignment record
    DB->>RT: Trigger realtime event
    DB->>FCM: Queue push notification

    par Real-time Updates
        RT-->>C2: Assignment created event
```

```
        C2-->>S: Show in-app notification
    and Push Notifications
        FCM-->>C2: Push notification
        C2-->>S: System notification
    end

    S->>C2: Click notification
    C2->>DB: Fetch assignment details
    DB-->>C2: Assignment data
    C2-->>S: Display assignment

    Note over RT: WebSocket connections for real-time updates
    Note over FCM: Background notifications when app closed
```

Security Architecture

Row Level Security (RLS) Implementation

```
graph TD
    subgraph "RLS Policy Engine"
        AUTH["Authentication Context
auth.uid()"]
        ROLE["User Role Check
profiles.role"]
        TENANT["Multi-tenant Isolation
school_id/board_id"]
    end

    subgraph "Data Access Patterns"
        STUDENT["Student Access
Own data + enrolled courses"]
        TEACHER["Teacher Access
Own classes + students"]
        ADMIN["Admin Access
Full organization data"]
    end

    subgraph "Security Layers"
        JWT["JWT Token Validation"]
        MFA["Multi-Factor Authentication"]
        SESSION["Session Management"]
        AUDIT["Access Logging"]
    end

    AUTH --> STUDENT
    AUTH --> TEACHER
    AUTH --> ADMIN
    ROLE --> STUDENT
    ROLE --> TEACHER
    ROLE --> ADMIN
    TENANT --> STUDENT
```

```
    TENANT --> TEACHER
    TENANT --> ADMIN

    JWT --> AUTH
    MFA --> AUTH
    SESSION --> AUTH
    AUDIT --> AUTH
```

API Security Flow

```
sequenceDiagram
    participant C as Client
    participant AG as API Gateway
    participant AUTH as Auth Service
    participant RLS as RLS Engine
    participant DB as Database
    participant LOG as Audit Log

    C->>AG: API Request + JWT
    AG->>AUTH: Validate JWT
    AUTH-->>AG: User context

    alt Invalid Token
        AG-->>C: 401 Unauthorized
    end

    AG->>RLS: Apply RLS policies
    RLS->>DB: Execute query with policies
    DB-->>RLS: Filtered results
    RLS-->>AG: Authorized data

    AG->>LOG: Log access attempt
    AG-->>C: Response data

    Note over RLS: Policies filter data based on user role and context
    Note over LOG: All database access logged for audit
```

Performance Optimizations

Caching Strategy

```
graph LR
    subgraph "Client Side"
        RC[React Query Cache
5 min TTL]
        LC[Local Storage
User preferences]
    end
```

```
    subgraph "Service Layer"
        SC[Service Cache
60 sec TTL]
        MC[Memory Cache
In-memory results]
    end

    subgraph "Database Layer"
        IDX[Database Indexes
Optimized queries]
        MAT[Materialized Views
Pre-computed analytics]
    end

    RC --> SC
    SC --> IDX
    SC --> MAT
    LC --> RC
    MC --> SC
```

Database Optimization

```
graph TB
    subgraph "Query Optimization"
        IDX[Strategic Indexes
- Composite indexes
- Partial indexes
- GIN indexes for arrays]
        PART[Table Partitioning
- By date
- By tenant]
        MAT[Materialized Views
- Analytics summaries
- User progress]
    end

    subgraph "Connection Management"
        POOL[Connection Pooling
PgBouncer]
        PREP[Prepared Statements
Query plan caching]
    end

    subgraph "Data Architecture"
        NORM[Normalized Schema
Referential integrity]
        DENORM[Selective Denormalization
Read-heavy tables]
        ARCH[Data Archiving
Historical data]
    end
```

```
    IDX --> POOL
    PART --> POOL
    MAT --> PREP
    NORM --> DENORM
    DENORM --> ARCH
```

Deployment Architecture

Infrastructure Overview

```
graph TB
    subgraph "CDN Layer"
        CF[Cloudflare
Global CDN]
        CACHE[Edge Caching
Static assets]
    end

    subgraph "Application Layer"
        APP[React Application
Static hosting]
        PWA[PWA Support
Offline capability]
    end

    subgraph "Backend Services"
        SB[Supabase Platform
Managed PostgreSQL]
        EF[Edge Functions
Serverless compute]
        ST[Supabase Storage
File management]
    end

    subgraph "External Services"
        AI[OpenAI API
AI processing]
        FCM[Firebase
Push notifications]
        EMAIL[Email Service
Transactional emails]
    end

    CF --> APP
    CACHE --> APP
    APP --> SB
    APP --> EF
    APP --> ST
    EF --> AI
```

```
    SB --> FCM
    SB --> EMAIL
```

CI/CD Pipeline

```
graph LR
    subgraph "Development"
        DEV[Developer
Local development]
        GIT[Git Repository
Version control]
    end

    subgraph "CI/CD"
        BUILD[Build Process
Vite + TypeScript]
        TEST[Automated Tests
Unit + Integration]
        LINT[Code Quality
ESLint + Prettier]
    end

    subgraph "Deployment"
        STAGING[Staging Environment
Testing]
        PROD[Production
Live system]
        MONITOR[Monitoring
Error tracking]
    end

    DEV --> GIT
    GIT --> BUILD
    BUILD --> TEST
    TEST --> LINT
    LINT --> STAGING
    STAGING --> PROD
    PROD --> MONITOR
```

API Documentation

Core Endpoints

Endpoint Category	Base Path	Description
Authentication	`/auth`	User login, registration, MFA
User Management	`/api/users`	Profile management, roles
Course Management	`/api/courses`	Course CRUD, enrollment

Endpoint Category	Base Path	Description
Class Management	`/api/classes`	Class creation, student assignment
AI Services	`/api/ai`	IRIS, APEX, AI tutoring
Analytics	`/api/analytics`	Reports, dashboards, insights
File Management	`/storage`	File upload, download, management
Notifications	`/api/notifications`	Push notifications, email

Edge Functions

Function	Purpose	Trigger
`iris-chat`	AI analytics assistant	User query
`apex-ai-assistant`	AI support assistant	User question
`reports-assistant`	Report generation AI	Analytics request
`generate-course-thumbnail`	AI thumbnail generation	Course creation
`send-notification`	Push notification dispatch	System events
`bulk-upload-users`	Batch user import	Admin action

Monitoring and Observability

System Monitoring

```
graph TB
    subgraph "Application Metrics"
        PERF[Performance Metrics
Response times, throughput]
        ERROR[Error Tracking
Exception monitoring]
        USER[User Analytics
Engagement, retention]
    end

    subgraph "Infrastructure Metrics"
        DB[Database Performance
Query times, connections]
        API[API Gateway Metrics
Request rates, latency]
        STORAGE[Storage Metrics
Usage, bandwidth]
    end

    subgraph "Business Metrics"
        ENROLL[Enrollment Rates
Course popularity]
```

```
        ENGAGE[Learning Engagement
Session duration, completion]
        AI[AI Usage
Query volume, accuracy]
    end

    PERF --> DB
    ERROR --> API
    USER --> STORAGE
    ENROLL --> ENGAGE
    ENGAGE --> AI
```

Logging Strategy

```
graph LR
    subgraph "Log Sources"
        APP[Application Logs
User actions, errors]
        DB[Database Logs
Query performance]
        AI[AI Service Logs
Token usage, responses]
    end

    subgraph "Log Processing"
        COLLECT[Log Collection
Structured logging]
        FILTER[Log Filtering
Noise reduction]
        ENRICH[Log Enrichment
Context addition]
    end

    subgraph "Log Storage"
        SEARCH[Search & Analysis
Query capabilities]
        ALERT[Alerting
Anomaly detection]
        ARCHIVE[Log Archival
Long-term storage]
    end

    APP --> COLLECT
    DB --> COLLECT
    AI --> COLLECT
    COLLECT --> FILTER
    FILTER --> ENRICH
    ENRICH --> SEARCH
    SEARCH --> ALERT
    SEARCH --> ARCHIVE
```

Scalability Considerations

Horizontal Scaling

```
graph TB
    subgraph "Frontend Scaling"
        CDN[Global CDN
Edge locations]
        LB[Load Balancer
Traffic distribution]
        CACHE[Browser Caching
Asset optimization]
    end

    subgraph "Backend Scaling"
        EF[Edge Functions
Auto-scaling serverless]
        DB[Database Scaling
Read replicas, sharding]
        STORAGE[Storage Scaling
Distributed file system]
    end

    subgraph "Service Scaling"
        AI[AI Service Limits
Rate limiting, queuing]
        QUEUE[Background Jobs
Async processing]
        MICRO[Microservices
Service isolation]
    end

    CDN --> LB
    LB --> EF
    EF --> DB
    DB --> STORAGE
    AI --> QUEUE
    QUEUE --> MICRO
```

Performance Bottlenecks

Component	Potential Bottleneck	Mitigation Strategy
Database	Complex analytics queries	Materialized views, caching
AI Services	OpenAI API rate limits	Request queuing, fallbacks
File Storage	Large file uploads	Chunked uploads, compression
Real-time	WebSocket connections	Connection pooling, clustering
Authentication	MFA verification	Async processing, caching

Future Enhancements

Planned Features

1. **Advanced AI Tutoring**
    - Personalized learning paths
    - Adaptive difficulty adjustment
    - Multi-modal learning support
2. **Enhanced Analytics**
    - Predictive analytics
    - Learning outcome prediction
    - Behavioral pattern analysis
3. **Mobile Applications**
    - Native iOS/Android apps
    - Offline learning support
    - Push notification enhancements
4. **Integration Ecosystem**
    - Third-party LMS integration
    - Single Sign-On (SSO)
    - API marketplace
5. **Advanced Security**
    - Zero-trust architecture
    - Advanced threat detection
    - Compliance frameworks (GDPR, COPPA)

Technical Debt

Area	Issue	Priority	Timeline
Code Quality	Legacy component refactoring	Medium	Q2 2024
Performance	Database query optimization	High	Q1 2024
Security	Enhanced audit logging	High	Q1 2024
Testing	Increased test coverage	Medium	Q2 2024
Documentation	API documentation completion	Low	Q3 2024

Conclusion

The DIL-LMS system represents a modern, scalable educational platform with advanced AI capabilities. The architecture supports multi-tenancy, role-based access control, and real-time collaboration while maintaining

high security standards and performance optimization.

The system's modular design allows for future enhancements and integrations while the comprehensive monitoring and observability ensure reliable operation at scale.

For technical questions or contributions, please refer to the development team or create an issue in the project repository.

Last Updated: December 2024 Version: 1.0 Document Status: Complete