

LMS Backend Technical Blueprint (Serverpod 3.0)

This document contains the complete structural and logical definition for a modern Learning Management System backend, featuring reusable content, AI-driven course generation, and flexible visibility controls.

1. Models

Course Container

Defines the high-level course settings and AI parameters.

```
class: Course
table: courses
fields:
  title: String
  description: String?
  courseImageUrl: String?
  bannerImageUrl: String?
  videoUrl: String?
  visibility: CourseVisibility
  systemPrompt: String?
  createdAt: DateTime, default=now
  updatedAt: DateTime, default=now
  modules: List<Module>?, relation
  courseIndices: List<CourseIndex>?, relation
  knowledgeFiles: List<KnowledgeFile>?, relation
```

enum: CourseVisibility

values:

- draft
- public
- private

Table of Contents Structure

Implements a hierarchical organization (Course > Module > Topic).

```
class: Module
table: modules
```

```

fields:
  title: String
  description: String?
  sortOrder: int
  imageUrl: String?
  bannerImageUrl: String?
  videoUrl: String?
  courseId: int
  course: Course?, relation(parent=courses)
  items: List<ModuleItem>?, relation

class: ModuleItem
table: module_items
fields:
  sortOrder: int
  contextualDescription: String? # Specific notes for this topic in this course
  moduleId: int
  module: Module?, relation(parent=modules)
  topicId: int
  topic: Topic?, relation(parent=topics)

class: Topic
table: topics
fields:
  title: String
  description: String? # The actual lesson content/body
  videoUrl: String?
  imageUrl: String?
  bannerImageUrl: String?
  createdAt: DateTime, default=now
  updatedAt: DateTime, default=now

```

Metadata & Source Material

Handles AI generation inputs and search performance.

```

class: KnowledgeFile
table: knowledge_files
fields:
  fileName: String
  fileUrl: String
  fileType: String?

```

```

uploadedAt: DateTime, default=now
courseld: int
course: Course?, relation(parent=courses)

class: CourseIndex
table: course_indices
fields:
  title: String
  description: String?
  imageUrl: String?
  tags: List<String>?
  sortOrder: int
  courseld: int
  course: Course?, relation(parent=courses)

```

2. API Logic Implementation (lib/src/endpoints/lms_endpoint.dart)

```

import 'package:serverpod/serverpod.dart';
import './generated/protocol.dart';

class LmsEndpoint extends Endpoint {

  // --- Course Management ---

  Future<Course> createCourse(Session session, Course course) async {
    course.createdAt = DateTime.now();
    course.updatedAt = DateTime.now();
    return await Course.db.insertRow(session, course);
  }

  Future<Course> updateCourse(Session session, Course course) async {
    course.updatedAt = DateTime.now();
    return await Course.db.updateRow(session, course);
  }

  Future<bool> deleteCourse(Session session, int id) async {
    var deleted = await Course.db.deleteRow(session, Course.include(), where: (t) =>
      t.id.equals(id));
    return deleted.isNotEmpty;
  }
}

```

```

Future<Course?> getCourseById(Session session, int id) async {
    return await Course.db.findById(
        session,
        id,
        include: Course.include(
            modules: Module.includeList(
                orderBy: (t) => t.sortOrder,
                items: ModuleItem.includeList(
                    orderBy: (t) => t.sortOrder,
                    topic: Topic.include(),
                ),
            ),
            knowledgeFiles: KnowledgeFile.includeList(),
        ),
    );
}

```

```

Future<List<Course>> listCourses(Session session, {String? keyword, CourseVisibility? visibility}) async {
    var where = Constant.bool(true);
    if (keyword != null && keyword.isNotEmpty) where = where &
    (Course.t.title.ilike('%$keyword%'));
    if (visibility != null) where = where & (Course.t.visibility.equals(visibility));

    return await Course.db.find(session, where: where, orderBy: (t) => t.createdAt,
    orderDescending: true);
}

```

// --- Knowledge File Management ---

```

Future<KnowledgeFile> addFileToCourse(Session session, KnowledgeFile file) async {
    file.uploadedAt = DateTime.now();
    return await KnowledgeFile.db.insertRow(session, file);
}

```

```

Future<List<KnowledgeFile>> getFilesForCourse(Session session, int courseId) async {
    return await KnowledgeFile.db.find(session, where: (t) => t.courseId.equals(courseId));
}

```

```

Future<bool> deleteFile(Session session, int fileId) async {
    var result = await KnowledgeFile.db.deleteRow(session, where: (t) => t.id.equals(fileId));
    return result.isNotEmpty;
}

```

```
}
```

```
// --- AI Structure Generation & Module Management ---
```

```
Future<bool> generateAiModulePlan(Session session, int courseId) async {
    // 1. Fetch KnowledgeFiles for courseId
    // 2. Extract text/metadata to build AI Context
    // 3. Request JSON structure from Gemini (Module names + Topic titles)
    // 4. Transaction:
    //   a. Insert Module records linked to courseId
    //   b. Insert Topic records (or link existing ones)
    //   c. Insert ModuleItem links with correct sortOrder
    return true;
}
```

```
Future<List<Module>> getModules(Session session, int courseId) async {
    return await Module.db.find(
        session,
        where: (t) => t.courseId.equals(courseId),
        orderBy: (t) => t.sortOrder,
        include: Module.include(
            items: ModuleItem.includeList(orderBy: (t) => t.sortOrder, topic: Topic.include()),
        ),
    );
}
```

```
Future<Module> updateModule(Session session, Module module) async {
    return await Module.db.updateRow(session, module);
}
```

```
Future<bool> deleteModule(Session session, int moduleId) async {
    var result = await Module.db.deleteRow(session, where: (t) => t.id.equals(moduleId));
    return result.isNotEmpty;
}
```

```
Future<void> deleteAllModules(Session session, int courseId) async {
    await Module.db.deleteWhere(session, where: (t) => t.courseId.equals(courseId));
}
```