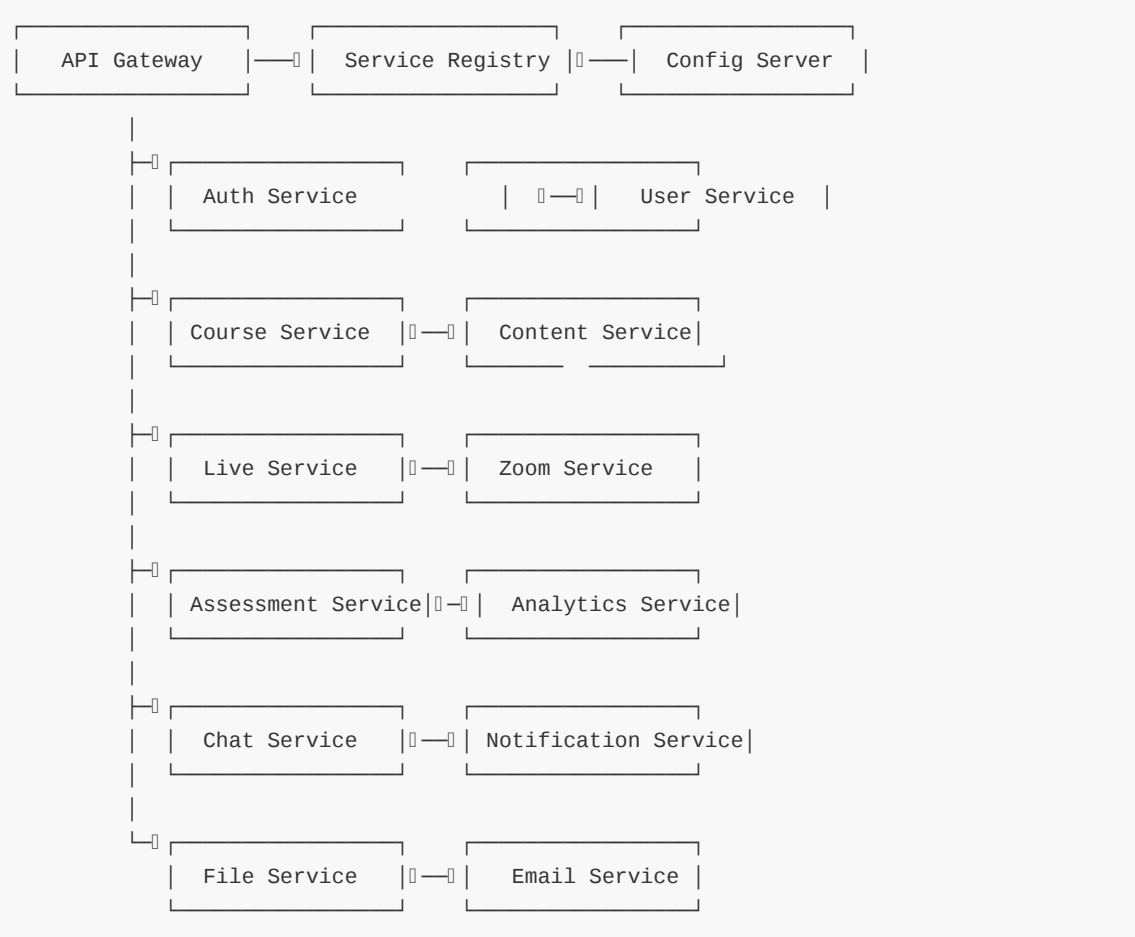
EduConnect Platform - Comprehensive Backend System Design

System Architecture Overview

Microservices Architecture



Detailed Database Schema

Core Tables with Complete Fields

```sql
-- Enhanced Users Table
CREATE TABLE users (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    email VARCHAR(255) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    role VARCHAR(20) NOT NULL CHECK (role IN ('TEACHER', 'TA', 'STUDENT', 'ADMIN')),
    first_name VARCHAR(100) NOT NULL,
    last_name VARCHAR(100) NOT NULL,
    profile_picture_url TEXT,
    phone_number VARCHAR(20),
```

```sql
    date_of_birth DATE,
    gender VARCHAR(10) CHECK (gender IN ('MALE', 'FEMALE', 'OTHER')),
    bio TEXT,
    timezone VARCHAR(50) DEFAULT 'UTC',
    language_preference VARCHAR(10) DEFAULT 'en',
    is_verified BOOLEAN DEFAULT FALSE,
    is_active BOOLEAN DEFAULT TRUE,
    last_login_at TIMESTAMP,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    version INTEGER DEFAULT 0
);

-- Institutions & Departments
CREATE TABLE institutions (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    name VARCHAR(255) NOT NULL,
    code VARCHAR(50) UNIQUE NOT NULL,
    logo_url TEXT,
    website_url TEXT,
    contact_email VARCHAR(255),
    phone_number VARCHAR(20),
    address TEXT,
    is_active BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE departments (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    institution_id UUID REFERENCES institutions(id),
    name VARCHAR(255) NOT NULL,
    code VARCHAR(50) NOT NULL,
    description TEXT,
    head_of_department UUID REFERENCES users(id),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Enhanced Courses Structure
CREATE TABLE courses (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    department_id UUID REFERENCES departments(id),
    code VARCHAR(50) NOT NULL,
    name VARCHAR(255) NOT NULL,
    description TEXT,
    credits INTEGER DEFAULT 0,
    academic_year INTEGER NOT NULL,
    semester VARCHAR(20) NOT NULL,
    course_logo_url TEXT,
    syllabus_url TEXT,
    max_students INTEGER,
    is_active BOOLEAN DEFAULT TRUE,
    created_by UUID REFERENCES users(id),
```

```
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE course_enrollments (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    course_id UUID REFERENCES courses(id),
    user_id UUID REFERENCES users(id),
    enrollment_type VARCHAR(20) CHECK (enrollment_type IN ('TEACHER', 'TA',
'STUDENT')),
    enrollment_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    enrollment_status VARCHAR(20) DEFAULT 'ACTIVE' CHECK (enrollment_status IN
('ACTIVE', 'INACTIVE', 'SUSPENDED')),
    UNIQUE(course_id, user_id)
);

-- Enhanced Subjects & Lessons
CREATE TABLE subjects (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    course_id UUID REFERENCES courses(id),
    name VARCHAR(255) NOT NULL,
    code VARCHAR(50),
    description TEXT,
    objectives TEXT,
    order_index INTEGER DEFAULT 0,
    estimated_hours INTEGER DEFAULT 0,
    is_active BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE lessons (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    subject_id UUID REFERENCES subjects(id),
    title VARCHAR(255) NOT NULL,
    description TEXT,
    lesson_type VARCHAR(20) CHECK (lesson_type IN ('LECTURE', 'PRACTICAL', 'TUTORIAL',
'LAB')),
    scheduled_date TIMESTAMP,
    duration_minutes INTEGER DEFAULT 60,
    order_index INTEGER DEFAULT 0,
    learning_objectives TEXT,
    prerequisites TEXT,
    is_published BOOLEAN DEFAULT FALSE,
    created_by UUID REFERENCES users(id),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Enhanced Live Sessions with Zoom Integration
CREATE TABLE live_sessions (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    lesson_id UUID REFERENCES lessons(id),
```

```
    zoom_meeting_id BIGINT UNIQUE,
    meeting_url TEXT NOT NULL,
    meeting_password VARCHAR(255),
    start_url TEXT, -- For host
    join_url TEXT,  -- For participants
    topic VARCHAR(255),
    agenda TEXT,
    start_time TIMESTAMP NOT NULL,
    duration_minutes INTEGER DEFAULT 60,
    status VARCHAR(20) DEFAULT 'SCHEDULED' CHECK (status IN ('SCHEDULED', 'STARTED',
'ENDED', 'CANCELLED')),
    recording_url TEXT,
    recording_status VARCHAR(20) DEFAULT 'PENDING' CHECK (recording_status IN
('PENDING', 'PROCESSING', 'AVAILABLE', 'FAILED')),
    participant_count INTEGER DEFAULT 0,
    max_participants INTEGER DEFAULT 100,
    settings JSONB, -- Zoom meeting settings
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Enhanced Study Materials with Versioning
CREATE TABLE study_materials (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    lesson_id UUID REFERENCES lessons(id),
    title VARCHAR(255) NOT NULL,
    description TEXT,
    file_url TEXT NOT NULL,
    file_name VARCHAR(255) NOT NULL,
    file_type VARCHAR(50) NOT NULL,
    file_size BIGINT NOT NULL,
    mime_type VARCHAR(100),
    version VARCHAR(20) DEFAULT '1.0',
    is_primary BOOLEAN DEFAULT TRUE,
    access_level VARCHAR(20) DEFAULT 'PUBLIC' CHECK (access_level IN ('PUBLIC',
'RESTRICTED', 'PRIVATE')),
    download_count INTEGER DEFAULT 0,
    view_count INTEGER DEFAULT 0,
    uploader_id UUID REFERENCES users(id),
    uploaded_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Enhanced Assessment System
CREATE TABLE assessments (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    lesson_id UUID REFERENCES lessons(id),
    title VARCHAR(255) NOT NULL,
    description TEXT,
    assessment_type VARCHAR(20) CHECK (assessment_type IN ('QUIZ', 'ASSIGNMENT',
'EXAM', 'SURVEY')),
    duration_minutes INTEGER NOT NULL,
```

```
    total_marks DECIMAL(10,2) NOT NULL,
    passing_marks DECIMAL(10,2),
    max_attempts INTEGER DEFAULT 1,
    shuffle_questions BOOLEAN DEFAULT FALSE,
    shuffle_options BOOLEAN DEFAULT FALSE,
    show_results_immediately BOOLEAN DEFAULT FALSE,
    show_correct_answers BOOLEAN DEFAULT FALSE,
    start_date TIMESTAMP,
    end_date TIMESTAMP,
    time_limit_type VARCHAR(20) DEFAULT 'FIXED' CHECK (time_limit_type IN ('FIXED',
'FLEXIBLE')),
    status VARCHAR(20) DEFAULT 'DRAFT' CHECK (status IN ('DRAFT', 'SCHEDULED',
'ACTIVE', 'COMPLETED', 'CANCELLED')),
    instructions TEXT,
    created_by UUID REFERENCES users(id),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE questions (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    assessment_id UUID REFERENCES assessments(id),
    question_text TEXT NOT NULL,
    question_type VARCHAR(20) CHECK (question_type IN ('MCQ_SINGLE', 'MCQ_MULTIPLE',
'TRUE_FALSE', 'SHORT_ANSWER', 'ESSAY')),
    options JSONB, -- {"A": "Option 1", "B": "Option 2"}
    correct_answers JSONB, -- ["A"] or ["A", "B"] for multiple
    explanation TEXT,
    marks DECIMAL(5,2) DEFAULT 1.00,
    difficulty_level VARCHAR(20) DEFAULT 'MEDIUM' CHECK (difficulty_level IN ('EASY',
'MEDIUM', 'HARD')),
    order_index INTEGER DEFAULT 0,
    tags TEXT[], -- Array of tags for categorization
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE student_attempts (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    assessment_id UUID REFERENCES assessments(id),
    student_id UUID REFERENCES users(id),
    attempt_number INTEGER DEFAULT 1,
    started_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    submitted_at TIMESTAMP,
    time_spent_seconds INTEGER DEFAULT 0,
    total_score DECIMAL(10,2) DEFAULT 0,
    max_score DECIMAL(10,2),
    percentage DECIMAL(5,2),
    status VARCHAR(20) DEFAULT 'IN_PROGRESS' CHECK (status IN ('IN_PROGRESS',
'SUBMITTED', 'AUTO_SUBMITTED', 'GRADED')),
    answers JSONB, -- Student's answers
    ip_address INET,
    user_agent TEXT,
```

```
    UNIQUE(assessment_id, student_id, attempt_number)
);

-- Enhanced Polling System
CREATE TABLE polls (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    lesson_id UUID REFERENCES lessons(id),
    question TEXT NOT NULL,
    options JSONB NOT NULL, -- {"A": "Option 1", "B": "Option 2"}
    poll_type VARCHAR(20) DEFAULT 'SINGLE_CHOICE' CHECK (poll_type IN
('SINGLE_CHOICE', 'MULTIPLE_CHOICE')),
    is_anonymous BOOLEAN DEFAULT TRUE,
    is_active BOOLEAN DEFAULT TRUE,
    show_results_immediately BOOLEAN DEFAULT TRUE,
    allow_multiple_responses BOOLEAN DEFAULT FALSE,
    created_by UUID REFERENCES users(id),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    expires_at TIMESTAMP
);

CREATE TABLE poll_responses (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    poll_id UUID REFERENCES polls(id),
    student_id UUID REFERENCES users(id),
    selected_options JSONB NOT NULL, -- ["A"] or ["A", "B"]
    responded_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    UNIQUE(poll_id, student_id)
);

-- Global Chat System (No Private Chats)
CREATE TABLE chat_rooms (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    course_id UUID REFERENCES courses(id),
    name VARCHAR(255) NOT NULL,
    description TEXT,
    is_active BOOLEAN DEFAULT TRUE,
    allow_file_sharing BOOLEAN DEFAULT TRUE,
    max_file_size_mb INTEGER DEFAULT 10,
    created_by UUID REFERENCES users(id),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE chat_messages (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    chat_room_id UUID REFERENCES chat_rooms(id),
    user_id UUID REFERENCES users(id),
    message_text TEXT,
    message_type VARCHAR(20) DEFAULT 'TEXT' CHECK (message_type IN ('TEXT', 'FILE',
'SYSTEM')),
    file_url TEXT,
    file_name VARCHAR(255),
    file_type VARCHAR(50),
```

```
    file_size BIGINT,
    parent_message_id UUID REFERENCES chat_messages(id), -- For replies
    is_edited BOOLEAN DEFAULT FALSE,
    edited_at TIMESTAMP,
    is_deleted BOOLEAN DEFAULT FALSE,
    deleted_at TIMESTAMP,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE chat_message_reads (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    message_id UUID REFERENCES chat_messages(id),
    user_id UUID REFERENCES users(id),
    read_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    UNIQUE(message_id, user_id)
);

-- Enhanced Attendance System
CREATE TABLE attendance (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    session_id UUID REFERENCES live_sessions(id),
    student_id UUID REFERENCES users(id),
    join_time TIMESTAMP NOT NULL,
    leave_time TIMESTAMP,
    duration_seconds INTEGER DEFAULT 0,
    attendance_status VARCHAR(20) DEFAULT 'PRESENT' CHECK (attendance_status IN
('PRESENT', 'LATE', 'ABSENT', 'EXCUSED')),
    participation_score INTEGER DEFAULT 0, -- Based on engagement
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Analytics & Reporting Tables
CREATE TABLE user_activities (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID REFERENCES users(id),
    activity_type VARCHAR(50) NOT NULL,
    resource_type VARCHAR(50),
    resource_id UUID,
    description TEXT,
    ip_address INET,
    user_agent TEXT,
    metadata JSONB,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE course_analytics (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    course_id UUID REFERENCES courses(id),
    date DATE NOT NULL,
    total_students INTEGER DEFAULT 0,
    active_students INTEGER DEFAULT 0,
    average_attendance_rate DECIMAL(5,2),
```

```
    average_assessment_score DECIMAL(5,2),
    total_materials_downloaded INTEGER DEFAULT 0,
    total_live_sessions INTEGER DEFAULT 0,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Detailed Service Architecture

1. API Gateway Service

```
spring:
  cloud:
    gateway:
      routes:
        - id: auth-service
          uri: lb://auth-service
          predicates:
            - Path=/api/auth/**
          filters:
            - RateLimit=1000,10s

        - id: user-service
          uri: lb://user-service
          predicates:
            - Path=/api/users/**
          filters:
            - JwtAuth

        - id: course-service
          uri: lb://course-service
          predicates:
            - Path=/api/courses/**,/api/subjects/**,/api/lessons/**
```

2. Authentication Service

```
@Service
public class AuthService {

    public AuthResponse login(LoginRequest request) {
        // JWT token generation
        // Refresh token management
        // Login auditing
    }

    public void logout(String token, String refreshToken) {
        // Token blacklisting
        // Session cleanup
    }

    public AuthResponse refreshToken(RefreshTokenRequest request) {
        // Token refresh logic
```

```
        // Security validation
    }
}
```

3. User Management Service

```
@Service
public class UserService {

    public UserProfile getProfile(UUID userId) {
        // User profile with statistics
    }

    public Page<UserDTO> getCourseStudents(UUID courseId, Pageable pageable) {
        // Paginated student list with progress
    }

    public void bulkEnrollUsers(BulkEnrollRequest request) {
        // CSV processing
        // Email notifications
    }
}
```

4. Course Management Service

```
@Service
public class CourseService {

    public CourseDetailDTO createCourse(CreateCourseRequest request) {
        // Course creation with default structure
        // Teacher assignment
        // Notification to department
    }

    public LessonDTO scheduleLesson(ScheduleLessonRequest request) {
        // Lesson scheduling
        // Conflict detection
        // Notification to enrolled students
    }
}
```

5. Live Session Service with Zoom Integration

```
@Service
public class LiveSessionService {

    @Async
    public LiveSessionDTO scheduleLiveSession(ScheduleLiveRequest request) {
        // Zoom API integration
        // Meeting configuration
        // Calendar integration
```

```
        // Email notifications
    }

    public void processRecording(RecordingWebhook webhook) {
        // Recording processing
        // Storage management
        // Notification to students
    }

    public AttendanceReport generateAttendanceReport(UUID sessionId) {
        // Attendance analytics
        // Participation scoring
        // Report generation
    }
}
```

6. Assessment Service

```
@Service
public class AssessmentService {

    public AssessmentDTO createAssessment(CreateAssessmentRequest request) {
        // Question bank integration
        // Validation rules
        // Scheduling
    }

    public AssessmentAttempt startAttempt(StartAttemptRequest request) {
        // Time tracking
        // Question randomization
        // Anti-cheating measures
    }

    public AssessmentResult submitAttempt(SubmitAttemptRequest request) {
        // Auto-grading
        // Result calculation
        // Analytics update
    }
}
```

7. Global Chat Service

```
@Service
public class ChatService {

    @MessageMapping("/chat/{roomId}/send")
    public void sendMessage(ChatMessage message, @DestinationVariable String roomId) {
        // Message validation
        // Profanity filtering
        // File handling
        // Real-time broadcasting
```

```
    }

    public List<ChatMessageDTO> getMessages(UUID roomId, Pageable pageable) {
        // Paginated message history
        // User presence
        // File metadata
    }

    public FileUploadResponse uploadFile(MultipartFile file, UUID roomId) {
        // File validation
        // Virus scanning
        // Cloud storage
        // Thumbnail generation
    }
}
```

Complete API Design

Authentication APIs

```
# User Registration
POST /api/auth/register
Content-Type: application/json
{
  "email": "student@university.edu",
  "password": "SecurePassword123!",
  "firstName": "John",
  "lastName": "Doe",
  "role": "STUDENT",
  "institutionCode": "UNI-001",
  "departmentCode": "CS"
}

# User Login
POST /api/auth/login
Content-Type: application/json
{
  "email": "student@university.edu",
  "password": "SecurePassword123!"
}

# Refresh Token
POST /api/auth/refresh-token
Content-Type: application/json
{
  "refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
}

# Forgot Password
POST /api/auth/forgot-password
Content-Type: application/json
{
```

```
  "email": "student@university.edu"
}

# Reset Password
POST /api/auth/reset-password
Content-Type: application/json
{
  "token": "reset-token",
  "newPassword": "NewSecurePassword123!"
}
```

Course Management APIs

```
# Create Course
POST /api/courses
Authorization: Bearer {token}
Content-Type: application/json
{
  "code": "CS101",
  "name": "Introduction to Computer Science",
  "description": "Fundamental concepts of computer science...",
  "departmentId": "uuid",
  "academicYear": 2024,
  "semester": "SPRING",
  "credits": 3,
  "maxStudents": 50,
  "syllabus": "Course outline and objectives..."
}

# Get Course Details
GET /api/courses/{courseId}
Authorization: Bearer {token}

# Enroll Students in Course
POST /api/courses/{courseId}/enrollments/bulk
Authorization: Bearer {token}
Content-Type: application/json
{
  "studentEmails": ["student1@edu.com", "student2@edu.com"],
  "enrollmentType": "STUDENT"
}

# Get Course Analytics
GET /api/courses/{courseId}/analytics?startDate=2024-01-01&endDate=2024-12-31
Authorization: Bearer {token}
```

Live Session APIs

```
# Schedule Live Session
POST /api/lessons/{lessonId}/live-sessions
Authorization: Bearer {token}
```

```
Content-Type: application/json
{
  "topic": "Introduction to Algorithms",
  "agenda": "Discussing basic algorithms and complexity",
  "startTime": "2024-01-15T10:00:00Z",
  "durationMinutes": 90,
  "settings": {
    "waitingRoom": true,
    "muteOnEntry": true,
    "autoRecording": "cloud",
    "breakoutRooms": false
  }
}

# Join Live Session
GET /api/live-sessions/{sessionId}/join
Authorization: Bearer {token}

# End Live Session
POST /api/live-sessions/{sessionId}/end
Authorization: Bearer {token}

# Get Session Recordings
GET /api/live-sessions/{sessionId}/recordings
Authorization: Bearer {token}
```

Assessment APIs

```
# Create Assessment
POST /api/assessments
Authorization: Bearer {token}
Content-Type: application/json
{
  "lessonId": "uuid",
  "title": "Mid-term Quiz",
  "description": "Assessment covering topics 1-5",
  "assessmentType": "QUIZ",
  "durationMinutes": 45,
  "totalMarks": 100,
  "passingMarks": 50,
  "maxAttempts": 2,
  "startDate": "2024-01-20T09:00:00Z",
  "endDate": "2024-01-25T23:59:59Z",
  "shuffleQuestions": true,
  "showResultsImmediately": false,
  "questions": [
    {
      "questionText": "What is the time complexity of binary search?",
      "questionType": "MCQ_SINGLE",
      "options": {
        "A": "O(1)",
        "B": "O(log n)",
```

```
        "C": "O(n)",
        "D": "O(n log n)"
      },
      "correctAnswers": ["B"],
      "marks": 5,
      "difficultyLevel": "MEDIUM"
    }
  ]
}

# Start Assessment Attempt
POST /api/assessments/{assessmentId}/attempts
Authorization: Bearer {token}

# Submit Assessment Attempt
POST /api/assessments/{assessmentId}/attempts/{attemptId}/submit
Authorization: Bearer {token}
Content-Type: application/json
{
  "answers": {
    "question1": "A",
    "question2": ["B", "C"],
    "question3": "Binary search has O(log n) complexity"
  }
}

# Get Assessment Results
GET /api/assessments/{assessmentId}/results
Authorization: Bearer {token}
```

Global Chat APIs

```
# Get Chat Rooms for Course
GET /api/courses/{courseId}/chat-rooms
Authorization: Bearer {token}

# Send Message
POST /api/chat-rooms/{roomId}/messages
Authorization: Bearer {token}
Content-Type: application/json
{
  "messageText": "Hello everyone! Has anyone completed the assignment?",
  "parentMessageId": null
}

# Upload File to Chat
POST /api/chat-rooms/{roomId}/files
Authorization: Bearer {token}
Content-Type: multipart/form-data
file: @assignment_solution.pdf

# Get Message History
```

```
GET /api/chat-rooms/{roomId}/messages?page=0&size=50
Authorization: Bearer {token}

# Mark Messages as Read
POST /api/chat-rooms/{roomId}/messages/read
Authorization: Bearer {token}
Content-Type: application/json
{
  "messageIds": ["msg1", "msg2", "msg3"]
}
```

Polling APIs

```
# Create Poll
POST /api/lessons/{lessonId}/polls
Authorization: Bearer {token}
Content-Type: application/json
{
  "question": "Which topic did you find most challenging?",
  "options": {
    "A": "Algorithms",
    "B": "Data Structures",
    "C": "Object-Oriented Programming",
    "D": "Database Design"
  },
  "pollType": "SINGLE_CHOICE",
  "isAnonymous": true,
  "expiresAt": "2024-01-16T23:59:59Z"
}

# Respond to Poll
POST /api/polls/{pollId}/respond
Authorization: Bearer {token}
Content-Type: application/json
{
  "selectedOptions": ["A"]
}

# Get Poll Results
GET /api/polls/{pollId}/results
Authorization: Bearer {token}
```

Advanced Features Implementation

1. Real-time Communication with WebSocket

```
@Configuration
@EnableWebSocketMessageBroker
public class WebSocketConfig implements WebSocketMessageBrokerConfigurer {

    @Override
```

```
    public void configureMessageBroker(MessageBrokerRegistry config) {
        config.enableSimpleBroker("/topic", "/queue");
        config.setApplicationDestinationPrefixes("/app");
        config.setUserDestinationPrefix("/user");
    }

    @Override
    public void registerStompEndpoints(StompEndpointRegistry registry) {
        registry.addEndpoint("/ws")
                .setAllowedOriginPatterns("*")
                .withSockJS();
    }
}

@Service
public class ChatWebSocketService {

    @MessageMapping("/chat.{roomId}.send")
    @SendTo("/topic/chat.{roomId}")
    public ChatMessageDTO handleMessage(ChatMessage message,
                                        @DestinationVariable String roomId,
                                        Principal principal) {
        // Process and broadcast message
    }

    @EventListener
    public void handleSessionConnect(SessionConnectEvent event) {
        // Track user presence
    }

    @EventListener
    public void handleSessionDisconnect(SessionDisconnectEvent event) {
        // Update user presence
    }
}
```

2. Zoom Webhook Integration

```
@RestController
@RequestMapping("/api/webhooks/zoom")
public class ZoomWebhookController {

    @PostMapping("/recording-completed")
    public ResponseEntity<?> handleRecordingCompleted(@RequestBody ZoomWebhookPayload
payload) {
        // Validate webhook signature
        // Process recording
        // Update lesson materials
        // Notify students
    }

    @PostMapping("/meeting-ended")
```

```
    public ResponseEntity<?> handleMeetingEnded(@RequestBody ZoomWebhookPayload
payload) {
        // Finalize attendance
        // Generate session report
        // Trigger recording processing
    }

    @PostMapping("/participant-joined")
    public ResponseEntity<?> handleParticipantJoined(@RequestBody ZoomWebhookPayload
payload) {
        // Record attendance
        // Update live participant count
    }
}
```

3. Advanced Analytics Engine

```
@Service
public class AnalyticsService {

    @Scheduled(cron = "0 0 2 * * ?") // Daily at 2 AM
    public void generateDailyAnalytics() {
        // Course engagement metrics
        // Student performance trends
        // Resource utilization
        // Attendance patterns
    }

    public LearningAnalytics getStudentLearningPath(UUID studentId, UUID courseId) {
        // Progress tracking
        // Weak area identification
        // Personalized recommendations
    }

    public PredictiveAnalytics predictStudentPerformance(UUID studentId, UUID
courseId) {
        // Machine learning models
        // Risk identification
        // Intervention suggestions
    }
}
```

4. File Processing Pipeline

```
@Service
public class FileProcessingService {

    @Async
    public void processUploadedFile(FileUploadRequest request) {
        // Virus scanning
        // File type validation
```

```
        // Thumbnail generation for images
        // Text extraction for documents
        // Compression if needed
        // Cloud storage upload
        // Database record creation
    }

    public FilePreview generatePreview(String fileUrl, String fileType) {
        // Document to HTML conversion
        // Image resizing
        // Video thumbnail extraction
        // Audio waveform generation
    }
}
```

This comprehensive backend plan provides a robust, scalable foundation for the EduConnect platform with extensive features for modern educational needs. The system is designed for high availability, security, and excellent user experience.