EduLearn - Intelligent Learning Management System

NEXT.JS 14 TYPESCRIPT 5.0 TAILWIND CSS 3.0 AI POWERED 🤖

🎓 Overview

EduLearn is a next-generation Learning Management System (LMS) that combines traditional educational tools with cutting-edge AI technology. Our platform provides personalized learning experiences, intelligent analytics, and automated evaluation systems to enhance education for students, teachers, and administrators.

✨ Core Features

🔐 Authentication & Role-Based Access Control

- **Multi-role support**: Admin, Teacher, Student
- **JWT token-based authentication** with secure session management
- **Role-specific dashboards** and permissions
- **Protected routes** and component-level access control
- **Secure password hashing** with bcrypt and token refresh mechanism

🤖 AI-Powered Assistant

- **Intelligent chatbot** with role-specific interactions and context awareness
- **File attachment support** (PDF, Word, Excel, PowerPoint, text, CSV) with drag-and-drop interface
- **Smart file detection** - automatically processes files when mentioned in conversation
- **Real-time typing indicators** and modern chat UI with message bubbles
- **Quick reply suggestions** based on user roles and context
- **External webhook integration** (n8n) for advanced AI processing and natural language understanding
- **Session management** with persistent conversation history
- **SQL query assistance** for database exploration and reporting

📊 Dynamic Analytics Dashboards

- **Real-time data visualization** with interactive charts and graphs
- **Customizable dashboard widgets** with drag-and-drop layout management
- **Role-specific analytics**:
  - **Admin**: System-wide metrics, user statistics, course performance, enrollment trends
  - **Teacher**: Class performance, student progress, assignment analytics, engagement metrics
  - **Student**: Personal progress tracking, grade trends, learning insights, goal achievement
- **Export capabilities** for reports (PDF, Excel, CSV formats)
- **Automated report generation** with scheduled delivery
- **Comparative analytics** with historical data trends

🎯 Personalized Feedback System

- **AI-driven feedback generation** based on individual student performance patterns
- **Adaptive learning recommendations** tailored to learning styles and pace

- **Real-time progress tracking** with personalized milestones and achievements
- **Learning path optimization** using machine learning algorithms
- **Strength and weakness identification** with targeted improvement suggestions
- **Peer comparison analytics** (privacy-conscious and optional)
- **Motivational insights** and goal-setting assistance
- **Parent/guardian progress reports** (for K-12 implementations)

🔄 Automatic Evaluation System

- **AI-powered assignment grading** with customizable rubrics and criteria
- **Automated quiz scoring** with detailed explanations and instant feedback
- **Plagiarism detection** and academic integrity monitoring
- **Code evaluation** for programming assignments with syntax and logic checking
- **Natural language processing** for essay and written work assessment
- **Multi-format support** for various assignment types (multiple choice, essay, coding, etc.)
- **Instant feedback delivery** with detailed explanations and improvement suggestions
- **Grade book automation** with seamless integration and analytics
- **Appeal and review process** for contested grades

📚 Comprehensive Course Management

- **Interactive course creation** with multimedia support and templates
- **Assignment and quiz builder** with various question types and difficulty levels
- **Calendar integration** for scheduling, deadlines, and event management
- **Resource library** with organized file management and version control
- **Discussion forums** and collaborative spaces for peer interaction
- **Progress tracking** with completion certificates and badges
- **Content delivery** with adaptive pacing and prerequisites

👥 Advanced User Management

- **Comprehensive user profiles** with academic history and achievements
- **Bulk enrollment management** and automated class organization
- **Communication tools** (messaging, announcements, notifications)
- **Parent/guardian access portals** with progress monitoring
- **Bulk user import/export** with CSV and integration capabilities
- **Role hierarchy management** with custom permissions

🏗 Technical Architecture

Frontend Stack

- **Next.js 14** - React framework with App Router for modern web development
- **TypeScript** - Type-safe development with enhanced IDE support
- **Tailwind CSS** - Utility-first CSS framework for rapid UI development
- **Shadcn/UI** - Professional component library with accessibility features
- **Lucide Icons** - Consistent and customizable icon system

Backend Integration

- **RESTful API** architecture with Express.js
- **PostgreSQL Database** with connection pooling and optimized queries
- **JWT Authentication** with secure token management and refresh tokens
- **External AI Webhook** (n8n) for intelligent processing and automation
- **Environment-based configuration** for scalability across environments
- **Webhook-based Architecture** for scalable AI processing and third-party integrations

Key Libraries & Dependencies

```
{
  "ui": ["@radix-ui/react-*", "lucide-react", "tailwindcss"],
  "forms": ["react-hook-form", "@hookform/resolvers", "zod"],
  "charts": ["recharts", "chart.js", "react-chartjs-2"],
  "file-handling": ["react-dropzone", "file-saver", "multer"],
  "state-management": ["react-query", "zustand", "context-api"],
  "animations": ["framer-motion", "react-spring"],
  "security": ["bcrypt", "helmet", "cors", "rate-limiter"]
}
```

📁 Project Structure

```
EduLearn/
├── app/                    # Next.js App Router pages
│   ├── admin/             # Admin dashboard and management
│   │   ├── analytics/     # System-wide analytics
│   │   ├── users/         # User management
│   │   └── settings/      # System configuration
│   ├── auth/              # Authentication pages
│   │   ├── login/         # Login interface
│   │   └── register/      # Registration interface
│   ├── courses/           # Course management and viewing
│   │   └── [id]/          # Dynamic course pages
│   ├── dashboard/         # Role-based dashboards
│   │   ├── admin/         # Admin dashboard
│   │   ├── teacher/       # Teacher dashboard
│   │   └── student/       # Student dashboard
│   ├── analytics/         # Analytics and reporting
│   ├── assignments/       # Assignment management
│   │   └── [id]/          # Assignment details and submission
│   └── settings/          # User preferences and configuration
├── components/            # Reusable React components
│   ├── auth/              # Authentication forms and guards
│   ├── chatbot/           # AI assistant widget and components
│   ├── courses/           # Course-related UI components
│   ├── teacher/           # Teacher-specific tools and interfaces
│   │   ├── assignment-management.tsx
│   │   ├── grade-submissions.tsx
│   │   └── teacher-analytics.tsx
│   └── ui/                # Base UI component library
```

```
├── lib/                   # Utilities and services
│   ├── auth/              # Authentication logic and context
│   ├── hooks/             # Custom React hooks
│   ├── services/          # API services and data fetching
│   └── utils/             # Helper functions and utilities
├── backend/               # Backend API server
│   ├── src/
│   │   ├── controllers/   # Request handlers and business logic
│   │   ├── middleware/    # Authentication, validation, security
│   │   ├── routes/        # API endpoint definitions
│   │   ├── services/      # Database and external service interactions
│   │   └── types/         # TypeScript type definitions
│   └── package.json       # Backend dependencies
└── database/              # Database schemas and migrations
    └── migrations/        # SQL migration files
```

🚀 Quick Start

Prerequisites

- **Node.js** 18+ and npm/yarn
- **PostgreSQL** 12+ database server
- **AI Webhook Service** (n8n or similar platform)

Installation

1. **Clone the repository**

   ```
   git clone https://github.com/yourusername/edulearn.git
   cd edulearn
   ```

2. **Install dependencies**

   ```
   # Install all project dependencies
   npm run setup
   # or manually install
   npm install && cd backend && npm install
   ```

3. **Database Setup**

   ```
   # Create PostgreSQL database
   createdb edulearn_db

   # Run the database initialization script
   psql -d edulearn_db -f backend/src/edulearn_migrations_script.sql
   ```

4. **Environment Configuration**

```
# Frontend environment
cp .env.local.example .env.local

# Backend environment
cd backend && cp .env.example .env
```

Configure your environment variables:

Frontend (.env.local):

```
NEXT_PUBLIC_API_URL=http://localhost:3001/api
NEXT_PUBLIC_CHATBOT_WEBHOOK_URL=https://your-n8n-instance.com/webhook/user-query
```

Backend (.env):

```
PORT=3001
NODE_ENV=development
DB_HOST=localhost
DB_PORT=5432
DB_NAME=edulearn_db
DB_USER=postgres
DB_PASSWORD=your_password
JWT_SECRET=your-jwt-secret-key
JWT_REFRESH_SECRET=your-refresh-secret-key
FRONTEND_URL=http://localhost:3000

# Webhook Configuration (Optional)
WEBHOOK_SECRET=your-webhook-secret
WEBHOOK_TIMEOUT=30000
WEBHOOK_RETRY_ATTEMPTS=3
```

5. **Start the application**

```
# Terminal 1: Start backend API server
npm run backend:dev

# Terminal 2: Start frontend development server
npm run dev
```

6. **Access the application**

   - Frontend: http://localhost:3000

    - Backend API: http://localhost:3001/api

👤 Demo Credentials

The application comes with pre-configured demo accounts for testing:

- **Student Account**: `student@demo.com` / `password123`
- **Teacher Account**: `teacher@demo.com` / `password123`
- **Admin Account**: `admin@demo.com` / `password123`

💡 AI Assistant Capabilities

Smart Interactions

- **Role-aware responses** tailored to user permissions and context
- **Context-aware conversations** with persistent session management
- **File analysis** and document summarization with content extraction
- **SQL query assistance** for database exploration and custom reporting
- **Learning content recommendations** based on user behavior and performance
- **Natural language processing** for complex educational queries

Supported File Types

- 📄 **Documents**: PDF, Microsoft Word (.doc, .docx)
- 📊 **Spreadsheets**: Microsoft Excel (.xls, .xlsx)
- 📈 **Presentations**: Microsoft PowerPoint (.ppt, .pptx)
- 📝 **Text Files**: Plain text (.txt), CSV (.csv)
- **File Size Limit**: 10MB per file with batch upload support

Quick Actions by Role

Admin Quick Actions

- "Show all users" - Display comprehensive user statistics
- "Show user statistics" - Generate user engagement reports
- "List all courses" - Overview of all platform courses
- "Show system analytics" - Platform-wide performance metrics
- "Database schema help" - Technical database assistance

Teacher Quick Actions

- "Show my courses" - Display assigned courses and enrollment
- "Show my students" - Student roster with performance overview
- "Course enrollment stats" - Enrollment trends and analytics
- "Assignment submissions" - Review and grade submissions
- "Help with SQL" - Database query assistance for reporting

Student Quick Actions

- "Show my courses" - Display enrolled courses and progress
- "My assignments" - View pending and completed assignments
- "My grades" - Grade history and performance tracking
- "Course schedule" - Upcoming classes and deadlines
- "Help with SQL" - Learning assistance for database concepts

🔗 Webhook Integration & Architecture

AI Processing Webhook (n8n Integration)

EduLearn uses a sophisticated webhook-based architecture to handle AI processing, ensuring scalability and separation of concerns. The AI chatbot communicates with external services through secure webhook endpoints.

Webhook Configuration

Environment Variables:

```
# Frontend (.env.local)
NEXT_PUBLIC_CHATBOT_WEBHOOK_URL=https://your-n8n-instance.com/webhook/user-query

# Backend (.env) - Optional for webhook validation
WEBHOOK_SECRET=your-webhook-secret
WEBHOOK_TIMEOUT=30000
```

Webhook Payload Structure

When a user interacts with the AI assistant, the following payload is sent to the configured webhook:

```json
{
  "query": "Show me my course analytics",
  "attachments": [
    {
      "name": "document.pdf",
      "size": 2048576,
      "type": "application/pdf",
      "base64": "base64-encoded-file-content"
    }
  ],
  "user": {
    "id": "user-123",
    "email": "student@example.com",
    "firstName": "John",
    "lastName": "Doe",
    "role": "student"
  },
  "sessionId": "session_1719705600000_abc123",
  "timestamp": "2025-06-30T10:30:00.000Z"
}
```

Expected Webhook Response

The webhook should return a JSON response in one of these formats:

Simple Response:

```json
{
  "response": "Here are your course analytics...",
  "quick_replies": [
    { "text": "Show detailed breakdown" },
    { "text": "Export to PDF" },
    { "text": "Compare with peers" }
  ]
}
```

Array Response (for multiple messages):

```json
[
  {
    "text": "I found 3 courses in your enrollment.",
    "type": "info"
  },
  {
    "text": "Your average grade is 87%. Great work!",
    "type": "success"
  }
]
```

Rich Response with Data:

```json
{
  "response": "Here's your performance summary:",
  "data": {
    "charts": [
      {
        "type": "line",
        "title": "Grade Trends",
        "data": [...],
        "config": {...}
      }
    ],
    "tables": [
      {
        "title": "Recent Assignments",
        "headers": ["Assignment", "Grade", "Date"],
        "rows": [...]
      }
    ]
  },
  "quick_replies": [
    { "text": "Show more details" },
    { "text": "Set improvement goals" }
  ]
}
```

Webhook Security & Authentication

Request Headers

```
Content-Type: application/json
Accept: application/json
Authorization: Bearer <jwt-token>
X-User-Role: student|teacher|admin
X-User-ID: <user-id>
X-Webhook-Signature: <hmac-signature> (optional)
```

Security Implementation

- **JWT Token Validation**: All webhook requests include the user's JWT token
- **Role-based Access**: User role is passed in headers for permission checking
- **HMAC Signature**: Optional webhook signature validation for additional security
- **Rate Limiting**: Built-in rate limiting to prevent abuse
- **Timeout Protection**: 30-second timeout for webhook responses

n8n Workflow Example

Here's a sample n8n workflow configuration for handling EduLearn webhooks:

```
{
  "nodes": [
    {
      "name": "Webhook",
      "type": "n8n-nodes-base.webhook",
      "parameters": {
        "path": "user-query",
        "httpMethod": "POST",
        "responseMode": "responseNode"
      }
    },
    {
      "name": "Extract User Data",
      "type": "n8n-nodes-base.function",
      "parameters": {
        "functionCode": "const { query, user, attachments } = $json;\nreturn {\n
userQuery: query,\n  userId: user.id,\n  userRole: user.role,\n  hasFiles:
attachments && attachments.length > 0\n};"
      }
    },
    {
      "name": "Process Query",
      "type": "n8n-nodes-base.switch",
      "parameters": {
        "values": {
          "string": [
            {
              "value": "admin"
            },
            {
              "value": "teacher"
            },
            {
              "value": "student"
            }
          ]
        }
      }
    },
    {
      "name": "AI Processing",
      "type": "n8n-nodes-base.openAi",
      "parameters": {
        "operation": "text",
        "prompt": "{{ $json.userQuery }}\n\nUser Role: {{ $json.userRole
}}\nContext: Educational LMS"
      }
    },
    {
      "name": "Format Response",
      "type": "n8n-nodes-base.function",
      "parameters": {
        "functionCode": "const aiResponse =
$json.choices[0].message.content;\nreturn {\n  response: aiResponse,\n
quick_replies: [\n    { text: 'Tell me more' },\n    { text: 'Help with something
else' }\n  ]\n};"
      }
    },
    {
      "name": "Send Response",
      "type": "n8n-nodes-base.respondToWebhook",
      "parameters": {
        "responseBody": "={{ $json }}"
      }
    }
  ]
}
```

Webhook Error Handling

Error Response Format

```json
{
  "error": true,
  "message": "Unable to process your request at this time.",
  "code": "PROCESSING_ERROR",
  "retry": true
}
```

Common Error Scenarios

- **Timeout**: Webhook doesn't respond within 30 seconds
- **Authentication Failed**: Invalid or expired JWT token
- **Rate Limited**: Too many requests from the same user
- **Service Unavailable**: External AI service is down
- **Invalid Payload**: Malformed request data

Alternative Webhook Providers

While EduLearn is configured for n8n, it can work with any webhook service that supports the expected payload and response format:

Zapier Integration

```javascript
// Zapier Code Step Example
const { query, user, attachments } = inputData;

// Process the query based on user role
let response = "";
const quickReplies = [];

if (user.role === "student") {
  response = await processStudentQuery(query, user.id);
  quickReplies.push(
    { text: "Show my grades" },
    { text: "My assignments" }
  );
} else if (user.role === "teacher") {
  response = await processTeacherQuery(query, user.id);
  quickReplies.push(
    { text: "Class analytics" },
    { text: "Grade submissions" }
  );
}

return {
  response: response,
  quick_replies: quickReplies
};
```

Custom API Endpoint

```javascript
// Express.js endpoint example
app.post('/webhook/edulearn-query', async (req, res) => {
  try {
    const { query, user, attachments, sessionId } = req.body;

    // Validate JWT token
    const token = req.headers.authorization?.replace('Bearer ', '');
    const validUser = await validateToken(token);

    if (!validUser || validUser.id !== user.id) {
      return res.status(401).json({ error: 'Unauthorized' });
    }

    // Process the query with AI service
    const aiResponse = await processWithAI(query, user, attachments);

    // Return formatted response
    res.json({
      response: aiResponse.text,
      quick_replies: getQuickRepliesForRole(user.role)
    });
  } catch (error) {
    res.status(500).json({
      error: true,
      message: 'Processing failed',
      retry: true
    });
  }
});
```

Webhook Testing & Development

Local Development Setup

```bash
# Use ngrok to expose local webhook for testing
npm install -g ngrok
ngrok http 3001

# Update .env.local with ngrok URL
NEXT_PUBLIC_CHATBOT_WEBHOOK_URL=https://abc123.ngrok.io/webhook/user-query
```

Testing Tools

- **Postman**: Test webhook payloads and responses
- **ngrok**: Expose local development server for webhook testing
- **Webhook.site**: Debug webhook requests and responses
- **n8n Local**: Run n8n locally for development and testing

Monitoring & Logging

```
// Add webhook monitoring to your n8n workflow
{
  "name": "Log Request",
  "type": "n8n-nodes-base.function",
  "parameters": {
    "functionCode": "console.log('EduLearn Webhook Request:', JSON.stringify($json, null, 2));\nreturn $json;"
  }
}
```

Performance Optimization

Webhook Best Practices

- **Response Time**: Keep webhook response time under 5 seconds
- **Caching**: Implement caching for frequently requested data
- **Async Processing**: Use queues for long-running AI tasks
- **Error Recovery**: Implement retry logic with exponential backoff
- **Load Balancing**: Distribute webhook load across multiple instances

Scaling Considerations

- **Queue System**: Use Redis or RabbitMQ for handling high-volume requests
- **Database Optimization**: Optimize queries for user data retrieval
- **CDN Integration**: Cache static responses and file attachments
- **Monitoring**: Set up alerts for webhook failures and performance issues

📊 Analytics & Insights

Student Analytics Dashboard

- **Performance Tracking**: Grade trends, assignment completion rates, time-to-completion
- **Learning Patterns**: Study time analysis, engagement metrics, peak activity periods
- **Skill Assessment**: Strength and weakness identification with visual representation
- **Goal Setting**: Personalized learning objectives with progress tracking
- **Predictive Analytics**: Early warning systems for academic risk identification
- **Achievement Badges**: Gamification elements with milestone celebrations

Teacher Analytics Dashboard

- **Class Performance**: Overall grade distributions, participation rates, attendance tracking
- **Assignment Analytics**: Completion times, common mistakes, difficulty analysis
- **Student Progress**: Individual student tracking with intervention recommendations
- **Curriculum Effectiveness**: Content engagement metrics and learning outcome analysis
- **Workload Management**: Time allocation insights and efficiency recommendations
- **Parent Communication**: Automated progress reports and communication logs

Admin Analytics Dashboard

- **System Usage**: Active user metrics, feature adoption rates, session analytics
- **Course Performance**: Enrollment trends, completion rates, satisfaction scores
- **Resource Utilization**: Server metrics, storage usage, bandwidth analysis
- **Financial Insights**: Cost per student, revenue tracking, ROI calculations
- **Platform Health**: Performance monitoring, error tracking, uptime statistics
- **User Satisfaction**: Feedback analysis, support ticket trends, feature requests

🎯 Feature Roadmap

✅ Completed Features

- ☑ Authentication & comprehensive role-based access control
- ☑ AI chatbot with advanced file support and natural language processing
- ☑ Basic course management with multimedia support
- ☑ User dashboards with role-specific interfaces
- ☑ Modern, responsive UI with accessibility features
- ☑ Environment-based configuration for scalable deployment

🚧 Currently In Development

- ☐ **Dynamic Analytics Dashboards**
  - Real-time data visualization with interactive charts
  - Customizable dashboard widgets with drag-and-drop functionality
  - Advanced filtering and drill-down capabilities
  - Automated report generation and scheduled delivery
- ☐ **Personalized Feedback System**
  - AI-driven learning recommendations engine
  - Adaptive learning path optimization
  - Intelligent tutoring system integration
  - Behavioral pattern analysis for intervention
- ☐ **Automatic Evaluation System**
  - AI-powered assignment grading with natural language processing
  - Advanced plagiarism detection with similarity analysis
  - Code evaluation system for programming courses
  - Instant feedback generation with improvement suggestions

🔮 Future Enhancements

- ☐ **Mobile Applications** (React Native for iOS and Android)

- □ **Advanced AI Tutoring** with conversational learning assistance
- □ **Virtual Classroom Integration** with video conferencing and screen sharing
- □ **Blockchain-based Certificates** for credential verification
- □ **Multi-language Support** with internationalization
- □ **Advanced Security Features** including two-factor authentication
- □ **Integration Hub** for third-party educational tools and platforms
- □ **Offline Learning** capabilities with synchronization

🔒 Security & Privacy

Security Features

- **Data Encryption**: All sensitive data encrypted at rest and in transit using AES-256
- **Authentication Security**: JWT tokens with short expiration and refresh mechanisms
- **Input Validation**: Comprehensive sanitization and validation for all user inputs
- **Rate Limiting**: API endpoint protection against abuse and DDoS attacks
- **SQL Injection Prevention**: Parameterized queries and ORM protection
- **CORS Protection**: Configured cross-origin resource sharing policies
- **Security Headers**: Helmet.js implementation for HTTP security headers

Privacy Compliance

- **GDPR Compliance**: Privacy controls, data portability, and right to be forgotten
- **FERPA Compliance**: Educational record privacy protection (US)
- **Regular Security Audits**: Automated vulnerability scanning and penetration testing
- **Access Logging**: Comprehensive audit trails for all system interactions
- **Data Minimization**: Collection and retention of only necessary user data
- **Privacy Dashboard**: User control over personal data and privacy settings

🛠️ Development

Available Scripts

```
# Development
npm run dev              # Start frontend development server
npm run backend:dev      # Start backend development server
npm run setup           # Install all dependencies

# Production
npm run build           # Build frontend for production
npm run backend:build   # Build backend for production
npm run start          # Start production servers

# Testing
npm run test           # Run test suites
npm run test:watch     # Run tests in watch mode
npm run test:coverage  # Generate test coverage reports

# Database
```

```
npm run db:migrate      # Run database migrations
npm run db:seed         # Seed database with sample data
npm run db:reset        # Reset database to initial state

# Code Quality
npm run lint            # Run ESLint for code quality
npm run type-check      # Run TypeScript type checking
npm run format          # Format code with Prettier
```

Development Workflow

1. **Fork the repository** and create a feature branch
2. **Set up development environment** with all prerequisites
3. **Create feature branch**: `git checkout -b feature/amazing-feature`
4. **Make changes** following coding standards and best practices
5. **Write tests** for new functionality and ensure existing tests pass
6. **Commit changes**: `git commit -m 'Add amazing feature'`
7. **Push to branch**: `git push origin feature/amazing-feature`
8. **Open a Pull Request** with detailed description and test results

📚 API Documentation

The backend API is available at `http://localhost:3001/api` with comprehensive endpoints:

Authentication Endpoints

- `POST /api/auth/register` - User registration with role assignment
- `POST /api/auth/login` - User authentication with JWT token generation
- `GET /api/auth/me` - Get current authenticated user profile
- `POST /api/auth/refresh` - Refresh access token using refresh token
- `POST /api/auth/logout` - Secure user logout with token invalidation

User Management

- `GET /api/users` - List users with pagination and filtering
- `GET /api/users/:id` - Get specific user profile
- `PUT /api/users/:id` - Update user information
- `DELETE /api/users/:id` - Delete user account

Course Management

- `GET /api/courses` - List available courses
- `POST /api/courses` - Create new course (teachers/admins)
- `GET /api/courses/:id` - Get course details
- `PUT /api/courses/:id` - Update course information
- `DELETE /api/courses/:id` - Delete course

Analytics Endpoints

- `GET /api/analytics/dashboard/:role` - Role-specific dashboard data
- `GET /api/analytics/performance/:userId` - Individual performance metrics
- `GET /api/analytics/courses/:courseId` - Course-specific analytics
- `GET /api/analytics/system` - System-wide statistics (admin only)

Webhook Management

- `POST /api/webhooks/test` - Test webhook connectivity and response
- `GET /api/webhooks/status` - Check webhook service status
- `PUT /api/webhooks/config` - Update webhook configuration (admin only)
- `GET /api/webhooks/logs` - View webhook request/response logs

Health Check

- `GET /api/health` - API server status and database connectivity

🗄 Database Schema

The application uses PostgreSQL with optimized schema design:

Core Tables

- **users** - User accounts, profiles, and authentication data
- **courses** - Course information, metadata, and content structure
- **enrollments** - Student course enrollments with timestamps
- **assignments** - Course assignments with rubrics and deadlines
- **submissions** - Student assignment submissions and files
- **grades** - Grading information with feedback and timestamps
- **analytics** - User interaction and performance tracking data

Relationships

- Users have many enrollments (many-to-many with courses)
- Courses have many assignments and enrollments
- Assignments have many submissions from enrolled students
- Submissions are linked to grades with detailed feedback

🌐 Production Deployment

Deployment Checklist

1. **Infrastructure Setup**
    - Configure PostgreSQL database server
    - Set up Redis for session management (optional)
    - Configure load balancer and reverse proxy (nginx recommended)
2. **Environment Configuration**
    - Set production environment variables

   - Configure SSL certificates for HTTPS
   - Set up monitoring and logging systems

3. **Application Deployment**

   - Build frontend: `npm run build`
   - Build backend: `cd backend && npm run build`
   - Configure process manager (PM2 recommended)
   - Set up automated deployment pipeline

4. **Security Hardening**

   - Enable firewall and security groups
   - Configure backup and disaster recovery
   - Set up monitoring and alerting systems

Recommended Hosting Platforms

- **Vercel** - Frontend deployment with automatic scaling
- **Railway/Heroku** - Backend API with database integration
- **AWS/GCP/Azure** - Full infrastructure control and scalability
- **DigitalOcean** - Cost-effective solution with managed databases

🤝 Contributing

We welcome contributions from the educational technology community!

How to Contribute

1. **Fork the repository** and star the project
2. **Create an issue** to discuss proposed changes
3. **Create a feature branch** with descriptive naming
4. **Follow coding standards** and write comprehensive tests
5. **Submit a pull request** with detailed documentation
6. **Participate in code review** and address feedback

Contribution Guidelines

- Follow TypeScript and React best practices
- Write unit tests for new features
- Update documentation for API changes
- Ensure accessibility compliance (WCAG 2.1)
- Follow semantic versioning for releases

📞 Support & Resources

Documentation & Community

- **Technical Documentation**: docs.edulearn.com
- **API Reference**: api.edulearn.com

- **Community Forum**: community.edulearn.com
- **Developer Discord**: discord.gg/edulearn

Support Channels

- **Email Support**: support@edulearn.com
- **Bug Reports**: GitHub Issues
- **Feature Requests**: GitHub Discussions
- **Security Issues**: security@edulearn.com

Training & Resources

- **Video Tutorials**: YouTube channel with setup and usage guides
- **Webinar Series**: Monthly developer and educator training sessions
- **Documentation Wiki**: Comprehensive guides and troubleshooting
- **Sample Implementations**: Reference projects and code examples

📄 License

This project is licensed under the **MIT License** - see the LICENSE file for complete details.

Open Source Commitment

- Free for educational institutions and non-profit organizations
- Commercial licenses available for enterprise deployments
- Contributions welcome under the same MIT license terms
- Regular security updates and community support

🙏 Acknowledgments

Technology Partners

- **Next.js Team** for the excellent React framework and development experience
- **Vercel** for hosting solutions and deployment infrastructure
- **Shadcn** for the beautiful and accessible UI component library
- **Radix UI** for foundational accessibility and component primitives

Educational Community

- **Educators and Students** who provided feedback and feature requirements
- **Open Source Contributors** who helped build and improve the platform
- **Beta Testing Schools** who validated features in real classroom environments
- **Accessibility Advocates** who ensured inclusive design principles

Special Thanks

- Educational technology researchers for learning analytics insights
- AI/ML community for natural language processing guidance
- Security experts for vulnerability assessments and recommendations
- International educators for multi-cultural and multi-lingual perspectives

---

🎓 Built with 💖 for the future of education 🚀

🌐 Website • 📚 Documentation • 🐛 Report Bug • ✨ Request Feature

Empowering educators and students with intelligent learning technology