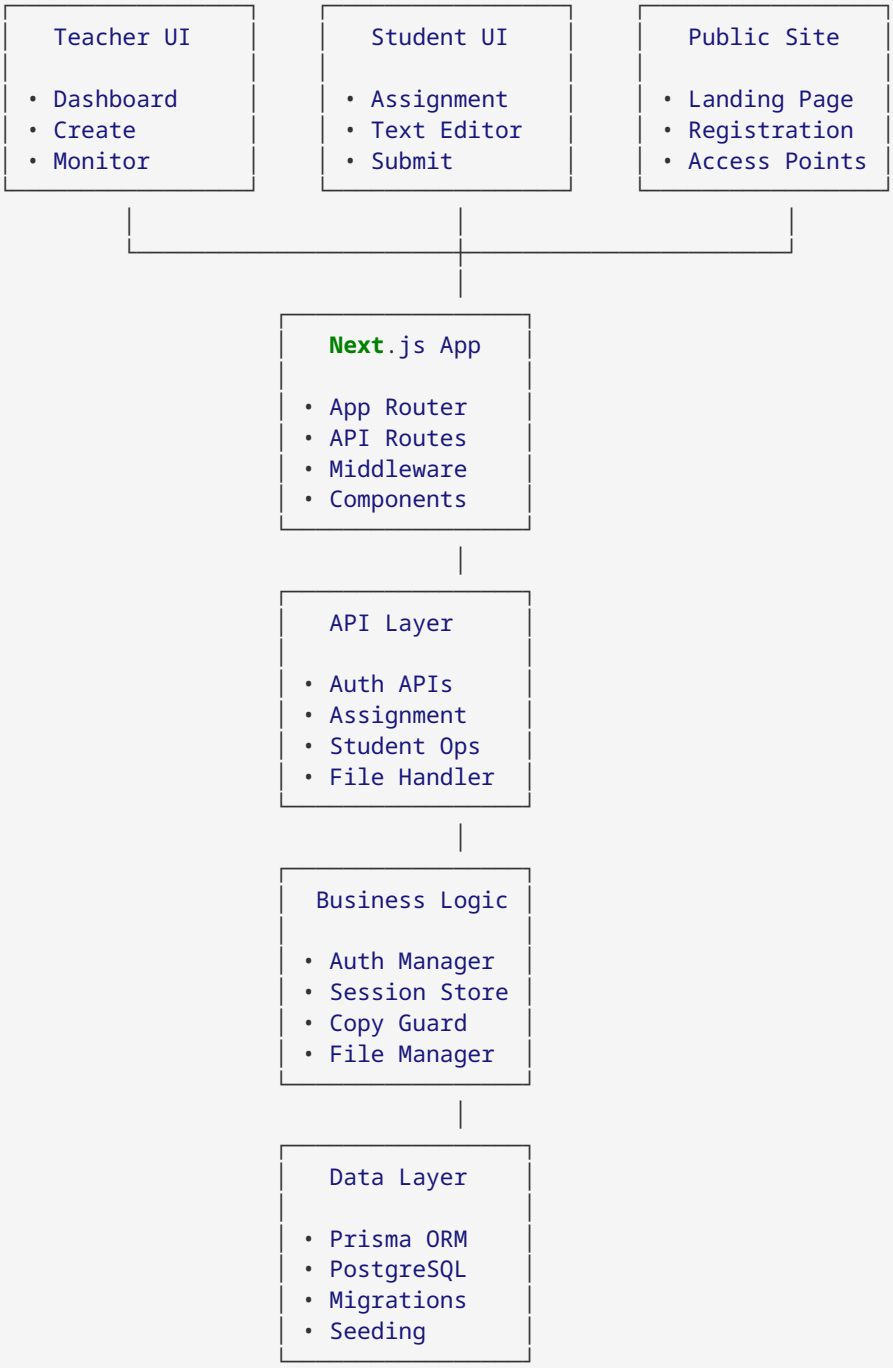
System Architecture Documentation

🏗️ Overview

The Homework Assignment System follows a modern full-stack architecture built with Next.js 14, featuring a PostgreSQL database, RESTful APIs, and a React-based frontend with advanced security measures.

📊 Architecture Diagram



🎯 Core Components

1. Frontend Architecture

Next.js 14 App Router

- **Server Components**: Used for static content and initial data loading
- **Client Components**: Interactive UI elements and copy protection
- **Route Groups**: Organized by user type (`/teacher/*` , `/student/*`)

UI Component Structure

```
components/
├── ui/                    # Reusable UI components (shadcn/ui)
│   ├── button.tsx
│   ├── card.tsx
│   ├── input.tsx
│   └── ...
├── copy-protection.tsx    # Copy protection wrapper
├── assignment-editor.tsx  # Student text editor
└── dashboard-stats.tsx    # Teacher analytics
```

Page Organization

```
app/
├── page.tsx              # Landing page
├── teacher/
│   ├── login/page.tsx
│   ├── register/page.tsx
│   ├── dashboard/page.tsx
│   └── create-assignment/page.tsx
├── student/
│   ├── page.tsx          # Access form
│   └── assignment/page.tsx
└── layout.tsx            # Root layout
```

2. Backend Architecture

API Route Structure

```
app/api/
├── auth/
│   ├── login/route.ts     # Teacher authentication
│   ├── register/route.ts  # Account creation
│   └── logout/route.ts    # Session termination
├── assignments/
│   ├── route.ts           # CRUD operations
│   └── [id]/
│       ├── route.ts       # Update/delete specific assignment
│       └── download/route.ts # File downloads
└── student/
    ├── access/route.ts    # Assignment access
    ├── save/route.ts      # Draft saving
    └── submit/route.ts    # Final submission
```

Business Logic Layer

```
lib/
├── auth.ts                # Authentication utilities
├── session.ts             # Session management
├── validation.ts          # Input validation
├── file-processor.ts      # File handling
└── copy-protection.ts     # Security measures
```

3. Database Architecture

Schema Design

```
Teachers (1) ──────── (N) Assignments (1) ──────── (N) StudentWork
    │                        │                          │
    ├ id (PK)                ├ id (PK)                  ├ id (PK)
    ├ username               ├ teacher_id (FK)          ├ assignment_id (FK)
    ├ password_hash          ├ title                    ├ student_name
    ├ active_sessions        ├ content                  ├ content
    ├ failed_attempts        ├ assignment_code          ├ status
    └ locked_until           ├ status                   ├ word_count
                             ├ student_count            └ timestamps
                             └ max_students

SystemLog
├ id (PK)
├ event_type
├ user_id
├ assignment_id
├ details
└ timestamp
```

Data Relationships

- **Teacher → Assignments**: One-to-many (1:N)
- **Assignment → StudentWork**: One-to-many (1:N)
- **Composite Unique Key**: (assignment_id, student_name) prevents duplicate submissions

🔄 Data Flow

1. Teacher Workflow

```
Registration → Authentication → Session Creation → Dashboard Access
                                      ↓
Assignment Creation → Code Generation → Student Monitoring
                                      ↓
Progress Tracking → File Downloads → Assignment Management
```

2. Student Workflow

```
Assignment Access → Validation → Content Display → Text Editor
                                      ↓
Auto-save Loop → Draft Storage → Manual Save → Final Submission
                                      ↓
Submission Lock → Confirmation → Assignment Closure
```

3. Security Workflow

```
Request → Authentication Check → Authorization → Rate Limiting
            ↓
Input Validation → Business Logic → Database Operation → Response
            ↓
Audit Logging → Error Handling → Security Monitoring
```

🛡️ Security Architecture

1. Authentication Layer

- **JWT Tokens**: Stateless authentication with HTTP-only cookies
- **Password Hashing**: bcrypt with 12 salt rounds
- **Session Management**: 2-hour expiration with refresh capability
- **Account Lockout**: 5 failed attempts → 15-minute lockout

2. Copy Protection System

```
// Multi-layered protection
const CopyProtection = {
  // CSS-level protection
  userSelect: 'none',
  webkitUserSelect: 'none',

  // JavaScript event blocking
  contextmenu: false,
  selectstart: false,
  dragstart: false,

  // Keyboard shortcuts
  'Ctrl+C': blocked,
  'Ctrl+V': blocked,
  'Ctrl+A': blocked,
  'F12': blocked,

  // Developer tools
  devtools: blocked
}
```

3. Input Validation

- **Server-side Validation**: All inputs validated at API level
- **Type Safety**: TypeScript interfaces for data structure
- **Sanitization**: XSS prevention on user inputs
- **Rate Limiting**: API endpoint throttling

📊 Capacity Management

1. Teacher Limits

```
interface TeacherLimits {
  maxActiveSessions: 3;
  sessionTimeout: 7200; // 2 hours
  maxAssignmentSize: 10000; // characters
}
```

2. Assignment Limits

```
interface AssignmentLimits {
  maxStudents: 30;
  maxContentLength: 50000; // characters
  autoSaveInterval: 30000; // 30 seconds
}
```

3. Resource Management

- **Database Connections**: Prisma connection pooling
- **Memory Usage**: Efficient query optimization
- **File Storage**: Temporary file cleanup
- **Session Storage**: JWT token cleanup

🔧 Technology Stack

Frontend

- **Framework**: Next.js 14 (App Router)
- **Language**: TypeScript
- **Styling**: Tailwind CSS + shadcn/ui
- **State Management**: React hooks + Context API
- **Form Handling**: Native form APIs with validation

Backend

- **Runtime**: Node.js 18+
- **API Framework**: Next.js API Routes
- **Authentication**: JWT + bcrypt
- **File Processing**: Native Node.js + Multer
- **Validation**: Zod + custom validators

Database

- **Primary Database**: PostgreSQL 14+
- **ORM**: Prisma 6.x
- **Migration Tool**: Prisma Migrate
- **Connection Pooling**: Built-in Prisma pooling

Infrastructure

- **Package Manager**: Yarn

- **Build Tool**: Next.js compiler
- **Type Checking**: TypeScript 5.x
- **Linting**: ESLint + Prettier

🚀 Deployment Architecture

Production Environment

Environment Configuration

- **Development**: SQLite + local server
- **Staging**: PostgreSQL + preview deployment
- **Production**: PostgreSQL + optimized build

📈 Performance Considerations

Frontend Optimization

- **Code Splitting**: Automatic route-based splitting
- **Image Optimization**: Next.js built-in optimization
- **CSS Optimization**: Tailwind purging + minification
- **Bundle Analysis**: webpack-bundle-analyzer integration

Backend Optimization

- **Database Queries**: Optimized with proper indexing
- **Caching Strategy**: Session-based caching
- **API Response**: Pagination + selective fields
- **File Handling**: Streaming for large files

Security Performance

- **Rate Limiting**: Prevents abuse and DoS
- **Input Validation**: Early validation reduces processing
- **Session Management**: Efficient JWT handling
- **Copy Protection**: Lightweight JavaScript implementation

🔍 Monitoring & Logging

Application Monitoring

- **Error Tracking**: Built-in error boundaries
- **Performance Metrics**: Next.js analytics
- **User Activity**: Custom event logging
- **System Health**: Database connection monitoring

Security Monitoring

- **Failed Login Attempts**: Tracked and logged
- **Copy Protection Violations**: Event logging
- **Suspicious Activity**: Rate limiting triggers
- **Data Access Patterns**: Assignment access monitoring

This architecture provides a scalable, secure, and maintainable foundation for the homework assignment system while ensuring academic integrity through comprehensive copy protection measures.