Technical Specification

📋 System Requirements

Hardware Requirements

Minimum Development Environment:
- CPU: 2+ cores, 2.0 GHz
- RAM: 8 GB
- Storage: 10 GB available space
- Network: Broadband internet connection

Production Environment:
- CPU: 4+ cores, 2.4 GHz
- RAM: 16 GB
- Storage: 50 GB SSD
- Network: High-speed internet with low latency

Software Requirements

Development Dependencies:
- Node.js 18.x or higher
- Yarn 3.x package manager
- PostgreSQL 14+ database
- Git version control
- TypeScript 5.x

Runtime Environment:
- Next.js 14.x framework
- Prisma ORM 6.x
- bcrypt 2.4.x for password hashing
- jsonwebtoken 9.x for sessions
- JSZip 3.10.x for bulk file downloads
- Markdown rendering system (custom implementation)

🔧 Database Schema

Tables Structure

Teachers Table

```
CREATE TABLE teachers (
  id                    TEXT PRIMARY KEY,
  username              VARCHAR(50) UNIQUE NOT NULL,
  password_hash         VARCHAR(255) NOT NULL,
  created_at            TIMESTAMP DEFAULT NOW(),
  last_login            TIMESTAMP,
  failed_attempts       INTEGER DEFAULT 0,
  locked_until          TIMESTAMP,
  active_sessions_count INTEGER DEFAULT 0
);
```

Assignments Table

```
CREATE TABLE assignments (
  id               TEXT PRIMARY KEY,
  teacher_id       TEXT REFERENCES teachers(id),
  title            VARCHAR(100) NOT NULL,
  content          TEXT NOT NULL,
  instructions     TEXT,
  assignment_code  VARCHAR(6) UNIQUE NOT NULL,
  status           VARCHAR(20) DEFAULT 'draft',
  deadline         TIMESTAMP,
  created_at       TIMESTAMP DEFAULT NOW(),
  activated_at     TIMESTAMP,
  closed_at        TIMESTAMP,
  student_count    INTEGER DEFAULT 0,
  max_students     INTEGER DEFAULT 30
);
```

Student Work Table

```
CREATE TABLE student_work (
  id               TEXT PRIMARY KEY,
  assignment_id    TEXT REFERENCES assignments(id),
  student_name     VARCHAR(50) NOT NULL,
  content          TEXT DEFAULT '',
  word_count       INTEGER DEFAULT 0,
  status           VARCHAR(20) DEFAULT 'draft',
  created_at       TIMESTAMP DEFAULT NOW(),
  last_saved_at    TIMESTAMP DEFAULT NOW(),
  submitted_at     TIMESTAMP,
  auto_save_count  INTEGER DEFAULT 0,

  UNIQUE(assignment_id, student_name)
);
```

System Log Table

```
CREATE TABLE system_logs (
  id            TEXT PRIMARY KEY,
  event_type    VARCHAR(50) NOT NULL,
  user_id       TEXT,
  assignment_id TEXT,
  student_name  VARCHAR(50),
  ip_address    VARCHAR(45),
  user_agent    TEXT,
  details       TEXT,
  created_at    TIMESTAMP DEFAULT NOW()
);
```

Indexes

```sql
-- Performance indexes
CREATE INDEX idx_assignments_teacher_id ON assignments(teacher_id);
CREATE INDEX idx_assignments_code ON assignments(assignment_code);
CREATE INDEX idx_assignments_status ON assignments(status);
CREATE INDEX idx_student_work_assignment ON student_work(assignment_id);
CREATE INDEX idx_student_work_status ON student_work(status);
CREATE INDEX idx_system_logs_event_type ON system_logs(event_type);
CREATE INDEX idx_system_logs_created_at ON system_logs(created_at);
```

🔗 API Documentation

Authentication Endpoints

POST /api/auth/register

Create new teacher account.

Request Body:

```
{
  username: string;    // 3-50 chars, alphanumeric + underscore
  password: string;    // 8+ characters
}
```

Response (Success - 200):

```
{
  success: true;
  message: string;
  teacher: {
    id: string;
    username: string;
  }
}
```

Response (Error - 409):

```
{
  error: "Username already exists"
}
```

POST /api/auth/login

Teacher authentication.

Request Body:

```
{
  username: string;
  password: string;
}
```

Response (Success - 200):

```
{
  success: true;
  teacher: {
    id: string;
    username: string;
    active_sessions_count: number;
  }
}
```

Headers Set:

```
Set-Cookie: session-token=JWT_TOKEN; HttpOnly; Secure; SameSite=Lax; Max-Age=7200
```

POST /api/auth/logout

Terminate session.

Response (200):

```
{
  success: true;
}
```

Assignment Management Endpoints

GET /api/assignments

List teacher's assignments.

Headers Required:

```
Cookie: session-token=JWT_TOKEN
```

Response (200):

```
{
  assignments: Array<{
    id: string;
    title: string;
    assignment_code: string;
    status: 'draft' | 'active' | 'closed';
    created_at: string;
    activated_at?: string;
    closed_at?: string;
    student_count: number;
    max_students: number;
    student_work: Array<{
      id: string;
      student_name: string;
      status: 'draft' | 'submitted';
      last_saved_at: string;
      submitted_at?: string;
      word_count: number;
    }>;
  }>
}
```

POST /api/assignments

Create new assignment.

Request Body:

```
{
  title: string;        // 5-100 characters
  content: string;      // Assignment text/question
  instructions?: string; // Optional additional instructions
  deadline?: string;    // ISO datetime string
}
```

Response (Success - 200):

```
{
  success: true;
  assignment: {
    id: string;
    title: string;
    assignment_code: string;
    status: string;
  }
}
```

Response (Limit Reached - 409):

```
{
  error: "You have reached the maximum of 3 active assignments. Please close an
existing assignment first."
}
```

PATCH /api/assignments/[id]

Update assignment status.

Request Body:

```
{
  action: 'close' | 'reopen';
}
```

Response (200):

```
{
  success: true;
  message: string;
}
```

DELETE /api/assignments/[id]

Delete assignment and all related student work.

Response (200):

```
{
  success: true;
  message: "Assignment deleted";
}
```

GET /api/assignments/[id]/download

Download submissions.

Query Parameters:
- `type`: 'all' | 'drafts' | 'submitted' | 'bulk'
- `student`: Student name for individual download

Response:
- **Single File**: `text/plain` with formatted submission
- **Bulk Download**: `application/zip` with multiple files

Student Endpoints

POST /api/student/access

Access assignment using code.

Request Body:

```
{
  studentName: string;    // 2-50 chars, letters and spaces only
  assignmentCode: string; // 6-character code
}
```

Response (Success - 200):

```
{
  assignment: {
    id: string;
    title: string;
    content: string;
    instructions?: string;
  };
  studentWork: {
    id: string;
    content: string;
    status: 'draft' | 'submitted';
    word_count: number;
    submitted_at?: string;
  };
  isReturning: boolean;
  isSubmitted: boolean;
}
```

Response (Not Found - 404):

```
{
  error: "Assignment not found. Please check the assignment code."
}
```

Response (Capacity Full - 403):

```
{
  error: "This assignment has reached its maximum capacity of 30 students."
}
```

POST /api/student/save

Save draft work.

Request Body:

```
{
  studentWorkId: string;
  content: string;
}
```

Response (200):

```
{
  success: true;
  studentWork: {
    id: string;
    content: string;
    word_count: number;
    last_saved_at: string;
    status: string;
  };
  message: "Draft saved successfully";
}
```

POST /api/student/submit

Submit final answer.

Request Body:

```
{
  studentWorkId: string;
  content: string;
}
```

Response (200):

```
{
  success: true;
  studentWork: {
    id: string;
    content: string;
    word_count: number;
    status: 'submitted';
    submitted_at: string;
  };
  message: "Assignment submitted successfully!";
}
```

🛡️ Security Specifications

Authentication & Authorization

Password Security

- **Hashing Algorithm**: bcrypt with 12 salt rounds
- **Minimum Requirements**: 8 characters minimum length
- **Storage**: Hashed passwords only, never plain text

Session Management

- **Token Type**: JWT (JSON Web Tokens)
- **Storage**: HTTP-only cookies for security
- **Expiration**: 2 hours (7200 seconds)
- **Security Flags**: `Secure`, `SameSite=Lax`

Account Lockout

```
interface LockoutPolicy {
  maxFailedAttempts: 5;
  lockoutDuration: 900; // 15 minutes in seconds
  attemptWindow: 3600;  // 1 hour window
}
```

Copy Protection Implementation

CSS-Based Protection

```
.copy-protected {
  -webkit-user-select: none;
  -moz-user-select: none;
  -ms-user-select: none;
  user-select: none;
  -webkit-touch-callout: none;
  -webkit-tap-highlight-color: transparent;
  pointer-events: none; /* For question content only */
}

.copy-protected::selection {
  background: transparent;
}

.copy-protected img {
  -webkit-user-drag: none;
  user-drag: none;
}
```

JavaScript Event Blocking

```
const copyProtectionEvents = {
  contextmenu: (e: Event) => e.preventDefault(),
  selectstart: (e: Event) => e.preventDefault(),
  dragstart: (e: Event) => e.preventDefault(),
  keydown: (e: KeyboardEvent) => {
    // Block Ctrl+C, Ctrl+V, Ctrl+A, F12, etc.
    const blockedKeys = ['c', 'v', 'a', 's', 'p', 'x'];
    if ((e.ctrlKey || e.metaKey) && blockedKeys.includes(e.key.toLowerCase())) {
      if (e.key.toLowerCase() !== 's') { // Allow Ctrl+S
        e.preventDefault();
        return false;
      }
    }
    if (e.key === 'F12' || (e.ctrlKey && e.shiftKey && e.key === 'I')) {
      e.preventDefault();
      return false;
    }
  }
};
```

Input Validation

Server-Side Validation Rules

```
const ValidationRules = {
  teacher: {
    username: {
      minLength: 3,
      maxLength: 50,
      pattern: /^[a-zA-Z0-9_]+$/,
      required: true
    },
    password: {
      minLength: 8,
      maxLength: 128,
      required: true
    }
  },
  assignment: {
    title: {
      minLength: 5,
      maxLength: 100,
      required: true
    },
    content: {
      minLength: 10,
      maxLength: 50000,
      required: true
    },
    assignment_code: {
      length: 6,
      pattern: /^[A-Z0-9]{6}$/,
      unique: true
    }
  },
  student: {
    name: {
      minLength: 2,
      maxLength: 50,
      pattern: /^[a-zA-Z\s]+$/,
      required: true
    }
  }
};
```

📁 File Management

Supported File Types

- **Text Files**: `.txt`
- **Microsoft Word**: `.doc`, `.docx`
- **PDF Documents**: `.pdf`
- **Maximum Size**: 10 MB per file

File Processing

```typescript
interface FileProcessor {
  // Extract text content from uploaded files
  extractText(file: Buffer, mimeType: string): Promise<string>;

  // Validate file type and size
  validateFile(file: File): ValidationResult;

  // Generate download files
  createSubmissionFile(assignment: Assignment, work: StudentWork): string;

  // Create ZIP archives for bulk downloads
  createBulkDownload(submissions: StudentWork[]): Promise<Buffer>;
}
```

Download File Format

```
HOMEWORK ASSIGNMENT SUBMISSION
==================================================

Assignment: {assignment.title}
Student: {work.student_name}
Status: {work.status}
Word Count: {work.word_count}
Last Saved: {work.last_saved_at}
Submitted: {work.submitted_at}

==================================================
QUESTION/PROMPT:
==================================================

{assignment.content}

==================================================
INSTRUCTIONS:
==================================================

{assignment.instructions}

==================================================
STUDENT ANSWER:
==================================================

{work.content}

==================================================
END OF SUBMISSION
==================================================
```

🔄 Auto-Save System

Implementation Details

```
interface AutoSaveConfig {
  interval: 30000;        // 30 seconds
  retryAttempts: 3;       // Retry failed saves
  retryDelay: 5000;       // 5 second delay
  conflictResolution: 'client-wins'; // Latest changes win
}

class AutoSaveManager {
  private saveInterval: NodeJS.Timeout;
  private lastSaveContent: string;
  private isSaving: boolean = false;

  startAutoSave(studentWorkId: string, getContent: () => string) {
    this.saveInterval = setInterval(async () => {
      const currentContent = getContent();
      if (currentContent !== this.lastSaveContent && !this.isSaving) {
        await this.performSave(studentWorkId, currentContent);
      }
    }, AutoSaveConfig.interval);
  }

  stopAutoSave() {
    if (this.saveInterval) {
      clearInterval(this.saveInterval);
    }
  }
}
```

📊 Performance Specifications

Response Time Requirements

- **Page Load**: < 2 seconds initial load
- **API Responses**: < 500ms average
- **Auto-save Operations**: < 200ms
- **File Downloads**: < 5 seconds for typical files

Concurrent Usage Capacity

- **Teachers**: 1000 concurrent sessions
- **Students**: 10000 concurrent users
- **Active Assignments**: 3000 total system-wide
- **Database Connections**: 100 connection pool

Browser Performance

- **JavaScript Bundle**: < 500KB compressed
- **CSS Bundle**: < 100KB compressed
- **Memory Usage**: < 50MB per student session
- **CPU Usage**: < 10% for copy protection

🚨 Error Handling

HTTP Status Codes

```
const StatusCodes = {
  200: 'OK - Request successful',
  201: 'Created - Resource created successfully',
  400: 'Bad Request - Invalid input data',
  401: 'Unauthorized - Authentication required',
  403: 'Forbidden - Access denied or capacity reached',
  404: 'Not Found - Resource does not exist',
  409: 'Conflict - Resource already exists or limit reached',
  429: 'Too Many Requests - Rate limit exceeded',
  500: 'Internal Server Error - Unexpected error'
};
```

Error Response Format

```
interface ErrorResponse {
  error: string;           // Human-readable error message
  code?: string;           // Error code for client handling
  details?: any;           // Additional error context
  timestamp: string;       // ISO datetime of error
}
```

✨ Rich Text & Content System

Markdown Editor Implementation

Rich Text Features

```
interface MarkdownEditor {
  formatters: {
    bold: (text: string) => `**${text}**`;
    italic: (text: string) => `*${text}*`;
    heading: (text: string) => `## ${text}`;
    bullet: (text: string) => `• ${text}`;
  };

  preview: {
    renderMarkdown: (text: string) => string;
    livePreview: boolean;
    copyProtection: boolean;
  };
}
```

Content Rendering Pipeline

```
// Client-side rendering (teacher preview & student view)
const renderMarkdown = (text: string) => {
  return text
    .replace(/\*\*(.*?)\*\*/g, '<strong>$1</strong>')
    .replace(/\*(.*?)\*/g, '<em>$1</em>')
    .replace(/^## (.*$)/gim, '<h4>$1</h4>')
    .replace(/^• (.*$)/gim, '<li>• $1</li>')
    .replace(/\n/g, '<br>');
};

// Server-side plain text conversion (for downloads)
const markdownToPlainText = (text: string) => {
  return text
    .replace(/\*\*(.*?)\*\*/g, '$1')
    .replace(/\*(.*?)\*/g, '$1')
    .replace(/^## (.*$)/gim, '$1')
    .replace(/^• (.*$)/gim, '• $1')
    .replace(/\n/g, '\n');
};
```

Enhanced Data Flow Architecture

Student Session Management

```
interface StudentSession {
  studentName: string;
  assignmentId: string;
  studentWorkId: string;
  isReturning: boolean;
  isSubmitted: boolean;
}

// Improved data storage pattern
localStorage.setItem('student-session', JSON.stringify(sessionData));
localStorage.setItem('current-assignment', JSON.stringify({
  assignment: assignmentData,
  studentWork: workData,
  isReturning: boolean,
  isSubmitted: boolean
}));
```

Assignment Data Integration

```
// Real-time data flow (no placeholders)
const dataFlow = {
  creation: 'Teacher → Rich Editor → Preview → Database',
  access: 'Student → Code Entry → Real Data Fetch → Display',
  submission: 'Student → Auto-save → Final Submit → Download'
};
```

Download System Architecture

Enhanced JSZip Implementation

```
interface DownloadConfig {
  compression: 'DEFLATE';
  compressionOptions: { level: 6 };
  errorHandling: 'comprehensive';
  filenameSanitization: boolean;
  crossPlatformCompatibility: true;
}

// Robust download generation
const createZip = async (submissions: StudentWork[]) => {
  try {
    const zip = new JSZip();

    for (const work of submissions) {
      const content = createSubmissionFile(assignment, work);
      const sanitizedName = work.student_name.replace(/[^a-zA-Z0-9_\-\s]/g, '');
      const filename = `${sanitizedName}_${formatDate(work.last_saved_at)}_${work.statu
s}.txt`;
      zip.file(filename, content);
    }

    return await zip.generateAsync({
      type: 'nodebuffer',
      compression: 'DEFLATE',
      compressionOptions: { level: 6 }
    });
  } catch (error) {
    throw new Error('ZIP generation failed');
  }
};
```

This technical specification provides comprehensive details for implementing, maintaining, and extending the homework assignment system while ensuring security, performance, and reliability standards are met.