

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>AI Quiz Generator Pro - Advanced Educational Platform</title>

  <link
href="https://fonts.googleapis.com/css2?family=Inter:wght@300;400;500;600;700;800&family=JetBrains+Mono:wght@400;500&display=swap" rel="stylesheet">

  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.4.0/css/all.min.css">

  <style>

    :root {

      --primary-gradient: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
      --secondary-gradient: linear-gradient(135deg, #f093fb 0%, #f5576c 100%);
      --success-gradient: linear-gradient(135deg, #4facfe 0%, #00f2fe 100%);
      --warning-gradient: linear-gradient(135deg, #43e97b 0%, #38f9d7 100%);
      --danger-gradient: linear-gradient(135deg, #fa709a 0%, #fee140 100%);
      --dark-gradient: linear-gradient(135deg, #2c3e50 0%, #34495e 100%);

      --glass-bg: rgba(255, 255, 255, 0.15);
      --glass-border: rgba(255, 255, 255, 0.2);
      --glass-shadow: 0 8px 32px 0 rgba(31, 38, 135, 0.37);

      --text-primary: #2c3e50;
      --text-secondary: #7f8c8d;
      --text-light: #ecf0f1;
```

```
--border-radius: 20px;

--border-radius-sm: 12px;

--transition: all 0.4s cubic-bezier(0.25, 0.46, 0.45, 0.94);

--shadow: 0 20px 60px rgba(0, 0, 0, 0.1);

--shadow-hover: 0 25px 80px rgba(0, 0, 0, 0.15);

}
```

```
* {

  margin: 0;

  padding: 0;

  box-sizing: border-box;

}
```

```
body {

  font-family: 'Inter', sans-serif;

  background: var(--primary-gradient);

  min-height: 100vh;

  overflow-x: hidden;

  position: relative;

}
```

```
/* Animated Background */

.background-animation {

  position: fixed;

  top: 0;
```

```
left: 0;
width: 100%;
height: 100%;
z-index: -1;
opacity: 0.1;
}
```

```
.floating-shapes {
  position: absolute;
  width: 100%;
  height: 100%;
}
```

```
.shape {
  position: absolute;
  border-radius: 50%;
  animation: float 6s ease-in-out infinite;
}
```

```
.shape:nth-child(1) {
  width: 80px;
  height: 80px;
  background: var(--success-gradient);
  top: 20%;
  left: 10%;
  animation-delay: 0s;
```

```
}
```

```
.shape:nth-child(2) {  
    width: 120px;  
    height: 120px;  
    background: var(--warning-gradient);  
    top: 60%;  
    right: 15%;  
    animation-delay: 2s;  
}
```

```
.shape:nth-child(3) {  
    width: 60px;  
    height: 60px;  
    background: var(--danger-gradient);  
    bottom: 20%;  
    left: 20%;  
    animation-delay: 4s;  
}
```

```
@keyframes float {  
    0%, 100% { transform: translateY(0px) rotate(0deg); }  
    33% { transform: translateY(-30px) rotate(120deg); }  
    66% { transform: translateY(15px) rotate(240deg); }  
}
```

```
/* Glassmorphism Container */
```

```
.main-container {  
  backdrop-filter: blur(16px) saturate(180%);  
  -webkit-backdrop-filter: blur(16px) saturate(180%);  
  background: var(--glass-bg);  
  border: 1px solid var(--glass-border);  
  border-radius: var(--border-radius);  
  box-shadow: var(--glass-shadow);  
  margin: 2rem;  
  overflow: hidden;  
  position: relative;  
}
```

```
/* Header with Particle Effect */
```

```
.hero-header {  
  background: linear-gradient(135deg, rgba(102, 126, 234, 0.9) 0%, rgba(118, 75, 162, 0.9) 100%);  
  padding: 4rem 2rem;  
  text-align: center;  
  position: relative;  
  overflow: hidden;  
}
```

```
.hero-header::before {  
  content: "";  
  position: absolute;
```

top: 0;

left: 0;

right: 0;

bottom: 0;

```
background: url('data:image/svg+xml,<svg xmlns="http://www.w3.org/2000/svg"
viewBox="0 0 100 100"><defs><pattern id="grain" width="100" height="100"
patternUnits="userSpaceOnUse"><circle cx="25" cy="25" r="1"
fill="rgba(255,255,255,0.1)" /><circle cx="75" cy="75" r="1.5"
fill="rgba(255,255,255,0.08)" /><circle cx="50" cy="10" r="0.8"
fill="rgba(255,255,255,0.12)" /></pattern></defs><rect width="100" height="100"
fill="url(%23grain)" /></svg>');
```

animation: grain 20s linear infinite;

}

@keyframes grain {

0%, 100% { transform: translate(0, 0); }

10% { transform: translate(-5%, -10%); }

20% { transform: translate(-15%, 5%); }

30% { transform: translate(7%, -25%); }

40% { transform: translate(-5%, 25%); }

50% { transform: translate(-15%, 10%); }

60% { transform: translate(15%, 0%); }

70% { transform: translate(0%, 15%); }

80% { transform: translate(3%, 35%); }

90% { transform: translate(-10%, 10%); }

}

.hero-content {

```
    position: relative;
    z-index: 2;
}
```

```
.hero-title {
    font-size: clamp(2.5rem, 5vw, 4rem);
    font-weight: 800;
    color: var(--text-light);
    margin-bottom: 1rem;
    text-shadow: 0 4px 20px rgba(0, 0, 0, 0.3);
    animation: slideInUp 1s ease-out;
}
```

```
.hero-subtitle {
    font-size: clamp(1.1rem, 2vw, 1.4rem);
    color: rgba(255, 255, 255, 0.9);
    margin-bottom: 2rem;
    animation: slideInUp 1s ease-out 0.2s both;
}
```

```
.feature-pills {
    display: flex;
    justify-content: center;
    flex-wrap: wrap;
    gap: 1rem;
    margin-top: 2rem;
```

```
    animation: slideInUp 1s ease-out 0.4s both;
}
```

```
.feature-pill {
    background: rgba(255, 255, 255, 0.2);
    backdrop-filter: blur(10px);
    padding: 0.8rem 1.5rem;
    border-radius: 50px;
    color: var(--text-light);
    font-weight: 500;
    font-size: 0.9rem;
    border: 1px solid rgba(255, 255, 255, 0.3);
    transition: var(--transition);
    position: relative;
    overflow: hidden;
}
```

```
.feature-pill::before {
    content: "";
    position: absolute;
    top: 0;
    left: -100%;
    width: 100%;
    height: 100%;
    background: linear-gradient(90deg, transparent, rgba(255, 255, 255, 0.2),
transparent);
}
```



```
    transition: left 0.5s;
}
```

```
.feature-pill:hover::before {
    left: 100%;
}
```

```
.feature-pill:hover {
    transform: translateY(-2px);
    box-shadow: 0 10px 25px rgba(0, 0, 0, 0.2);
}
```

```
/* Upload Section */
```

```
.upload-section {
    padding: 3rem;
    background: rgba(255, 255, 255, 0.05);
}
```

```
.upload-card {
    background: rgba(255, 255, 255, 0.95);
    backdrop-filter: blur(20px);
    border-radius: var(--border-radius);
    padding: 2.5rem;
    box-shadow: var(--shadow);
    transition: var(--transition);
}
```

```
.upload-card:hover {  
  transform: translateY(-5px);  
  box-shadow: var(--shadow-hover);  
}
```

```
.upload-zone {  
  border: 3px dashed #e0e6ed;  
  border-radius: var(--border-radius-sm);  
  padding: 3rem 2rem;  
  text-align: center;  
  transition: var(--transition);  
  background: linear-gradient(145deg, #f8fafc, #e2e8f0);  
  position: relative;  
  overflow: hidden;  
  cursor: pointer;  
}
```

```
.upload-zone::before {  
  content: "";  
  position: absolute;  
  top: 0;  
  left: 0;  
  right: 0;  
  bottom: 0;  
  background: var(--success-gradient);  
}
```

```
    opacity: 0;
    transition: var(--transition);
}
```

```
.upload-zone:hover::before {
    opacity: 0.05;
}
```

```
.upload-zone.dragover {
    border-color: #4facfe;
    background: rgba(79, 172, 254, 0.05);
    transform: scale(1.02);
}
```

```
.upload-zone.file-selected {
    border-color: #43e97b;
    background: var(--warning-gradient);
    background-size: 200% 200%;
    animation: gradientShift 3s ease infinite;
}
```

```
.upload-zone.error {
    border-color: #ef4444;
    background: rgba(239, 68, 68, 0.05);
    animation: shake 0.5s ease-in-out;
}
```

```
@keyframes gradientShift {  
  0% { background-position: 0% 50%; }  
  50% { background-position: 100% 50%; }  
  100% { background-position: 0% 50%; }  
}
```

```
@keyframes shake {  
  0%, 100% { transform: translateX(0); }  
  25% { transform: translateX(-5px); }  
  75% { transform: translateX(5px); }  
}
```

```
.upload-icon {  
  font-size: 4rem;  
  color: #64748b;  
  margin-bottom: 1rem;  
  transition: var(--transition);  
}
```

```
.upload-zone:hover .upload-icon {  
  transform: scale(1.1) rotate(5deg);  
  color: #4facfe;  
}
```

```
.upload-text {
```

```
font-size: 1.2rem;

font-weight: 600;

color: var(--text-primary);

margin-bottom: 0.5rem;
}
```

```
.upload-subtext {

  color: var(--text-secondary);

  font-size: 0.9rem;
}
```

```
/* File Validation Messages */

.file-validation {

  margin-top: 1rem;

  padding: 1rem;

  border-radius: var(--border-radius-sm);

  display: none;
}
```

```
.file-validation.success {

  background: linear-gradient(135deg, #d1fae5, #a7f3d0);

  color: #065f46;

  border: 1px solid #10b981;
}
```

```
.file-validation.error {
```

```
background: linear-gradient(135deg, #fee2e2, #fecaca);  
color: #991b1b;  
border: 1px solid #ef4444;  
}
```

```
.file-validation.warning {  
  background: linear-gradient(135deg, #fef3c7, #fde68a);  
  color: #92400e;  
  border: 1px solid #f59e0b;  
}
```

```
/* Controls Grid */  
.controls-grid {  
  display: grid;  
  grid-template-columns: repeat(auto-fit, minmax(280px, 1fr));  
  gap: 2rem;  
  margin: 2rem 0;  
}
```

```
.control-card {  
  background: rgba(255, 255, 255, 0.8);  
  backdrop-filter: blur(15px);  
  border-radius: var(--border-radius-sm);  
  padding: 2rem;  
  border: 1px solid rgba(255, 255, 255, 0.3);  
  transition: var(--transition);
```

```
    position: relative;
    overflow: hidden;
}
```

```
.control-card::before {
    content: "";
    position: absolute;
    top: 0;
    left: 0;
    right: 0;
    height: 4px;
    background: var(--primary-gradient);
    transform: scaleX(0);
    transition: var(--transition);
}
```

```
.control-card:hover::before {
    transform: scaleX(1);
}
```

```
.control-card:hover {
    transform: translateY(-5px);
    box-shadow: 0 15px 40px rgba(0, 0, 0, 0.1);
}
```

```
.control-label {
```

```
display: flex;
align-items: center;
gap: 0.8rem;
font-weight: 600;
color: var(--text-primary);
margin-bottom: 1rem;
font-size: 1.1rem;
}
```

```
.control-icon {
  font-size: 1.3rem;
  color: #667eea;
}
```

```
.form-input {
  width: 100%;
  padding: 1rem 1.2rem;
  border: 2px solid #e2e8f0;
  border-radius: var(--border-radius-sm);
  font-size: 1rem;
  transition: var(--transition);
  background: rgba(255, 255, 255, 0.9);
  font-family: inherit;
}
```

```
.form-input:focus {
```



```
outline: none;

border-color: #667eea;

box-shadow: 0 0 0 3px rgba(102, 126, 234, 0.1);

transform: translateY(-2px);
}
```

```
.form-input.error {

border-color: #ef4444;

box-shadow: 0 0 0 3px rgba(239, 68, 68, 0.1);
}
```

```
.input-hint {

font-size: 0.85rem;

color: var(--text-secondary);

margin-top: 0.5rem;

display: flex;

align-items: center;

gap: 0.4rem;
}
```

```
.input-error {

font-size: 0.85rem;

color: #ef4444;

margin-top: 0.5rem;

display: none;

align-items: center;
```

```
    gap: 0.4rem;
}
```

```
/* Generate Button */
```

```
.generate-section {
    text-align: center;
    margin-top: 2rem;
}
```

```
.generate-btn {
    background: var(--primary-gradient);
    color: white;
    border: none;
    padding: 1.2rem 3rem;
    border-radius: 50px;
    font-size: 1.2rem;
    font-weight: 600;
    cursor: pointer;
    transition: var(--transition);
    position: relative;
    overflow: hidden;
    box-shadow: 0 10px 30px rgba(102, 126, 234, 0.3);
    min-width: 280px;
}
```

```
.generate-btn::before {
```

```
    content: "";
    position: absolute;
    top: 0;
    left: -100%;
    width: 100%;
    height: 100%;
    background: linear-gradient(90deg, transparent, rgba(255, 255, 255, 0.2),
transparent);
    transition: left 0.5s;
}
```

```
.generate-btn:hover::before {
    left: 100%;
}
```

```
.generate-btn:hover {
    transform: translateY(-3px);
    box-shadow: 0 15px 40px rgba(102, 126, 234, 0.4);
}
```

```
.generate-btn:active {
    transform: translateY(-1px);
}
```

```
.generate-btn:disabled {
    opacity: 0.7;
```

```
    cursor: not-allowed;

    transform: none;
}
```

```
.btn-icon {

    margin-right: 0.8rem;

    font-size: 1.3rem;
}
```

```
/* Enhanced Loading Animation */

.loading-overlay {

    display: none;

    position: fixed;

    top: 0;

    left: 0;

    width: 100%;

    height: 100%;

    background: rgba(0, 0, 0, 0.8);

    backdrop-filter: blur(10px);

    z-index: 2000;

    animation: fadeIn 0.5s ease-out;
}
```

```
.loading-card {

    position: absolute;

    top: 50%;
```

```
left: 50%;  
  
transform: translate(-50%, -50%);  
  
background: rgba(255, 255, 255, 0.98);  
  
backdrop-filter: blur(20px);  
  
border-radius: var(--border-radius);  
  
padding: 3rem;  
  
text-align: center;  
  
box-shadow: var(--shadow);  
  
min-width: 450px;  
  
border: 1px solid rgba(255, 255, 255, 0.3);  
}
```

```
.loading-spinner {  
  width: 80px;  
  height: 80px;  
  border: 6px solid #f3f4f6;  
  border-top: 6px solid #667eea;  
  border-radius: 50%;  
  animation: spin 1s linear infinite;  
  margin: 0 auto 2rem;  
}
```

```
@keyframes spin {  
  0% { transform: rotate(0deg); }  
  100% { transform: rotate(360deg); }  
}
```

```
.loading-text {  
  font-size: 1.4rem;  
  font-weight: 700;  
  color: var(--text-primary);  
  margin-bottom: 1rem;  
}
```

```
.loading-subtext {  
  color: var(--text-secondary);  
  font-size: 1rem;  
  line-height: 1.6;  
  margin-bottom: 1.5rem;  
}
```

```
.loading-steps {  
  text-align: left;  
  background: #f8fafc;  
  padding: 1.5rem;  
  border-radius: var(--border-radius-sm);  
  margin: 1rem 0;  
}
```

```
.loading-step {  
  display: flex;  
  align-items: center;
```

```
gap: 0.8rem;
padding: 0.5rem 0;
color: var(--text-secondary);
transition: var(--transition);
}
```

```
.loading-step.active {
  color: #667eea;
  font-weight: 600;
}
```

```
.loading-step.completed {
  color: #10b981;
}
```

```
.step-icon {
  width: 20px;
  text-align: center;
}
```

```
.progress-bar {
  width: 100%;
  height: 8px;
  background: #e5e7eb;
  border-radius: 4px;
  margin: 1.5rem 0;
```

```
    overflow: hidden;
}
```

```
.progress-fill {
    height: 100%;
    background: var(--primary-gradient);
    border-radius: 4px;
    transition: width 0.5s ease;
    width: 0%;
}
```

```
/* Enhanced Error Handling */
.error-card {
    background: linear-gradient(135deg, #fee2e2, #fecaca);
    border: 2px solid #ef4444;
    border-radius: var(--border-radius-sm);
    padding: 2rem;
    margin: 1rem 0;
    display: none;
}
```

```
.error-header {
    display: flex;
    align-items: center;
    gap: 1rem;
    margin-bottom: 1rem;
```



```
}
```

```
.error-icon {
```

```
    font-size: 2rem;
```

```
    color: #ef4444;
```

```
}
```

```
.error-title {
```

```
    font-size: 1.3rem;
```

```
    font-weight: 600;
```

```
    color: #991b1b;
```

```
}
```

```
.error-message {
```

```
    color: #7f1d1d;
```

```
    line-height: 1.6;
```

```
    margin-bottom: 1rem;
```

```
}
```

```
.error-suggestions {
```

```
    background: rgba(255, 255, 255, 0.7);
```

```
    padding: 1rem;
```

```
    border-radius: var(--border-radius-sm);
```

```
    border-left: 4px solid #f59e0b;
```

```
}
```

```
.error-suggestions h4 {  
    color: #92400e;  
    margin-bottom: 0.5rem;  
}
```

```
.error-suggestions ul {  
    color: #78350f;  
    padding-left: 1.2rem;  
}
```

```
.error-suggestions li {  
    margin-bottom: 0.3rem;  
}
```

```
/* Retry Button */
```

```
.retry-btn {  
    background: var(--warning-gradient);  
    color: white;  
    border: none;  
    padding: 0.8rem 1.5rem;  
    border-radius: 50px;  
    font-weight: 600;  
    cursor: pointer;  
    transition: var(--transition);  
    margin-top: 1rem;  
}
```

```
.retry-btn:hover {  
    transform: translateY(-2px);  
    box-shadow: 0 5px 15px rgba(67, 233, 123, 0.3);  
}
```

```
/* Results Section */
```

```
.results-section {  
    display: none;  
    padding: 3rem;  
    background: rgba(255, 255, 255, 0.02);  
    animation: slideInUp 0.8s ease-out;  
}
```

```
.results-header {  
    display: flex;  
    justify-content: space-between;  
    align-items: center;  
    margin-bottom: 2rem;  
    flex-wrap: wrap;  
    gap: 1rem;  
}
```

```
.results-title {  
    font-size: 2.2rem;  
    font-weight: 700;
```

```
color: var(--text-light);  
text-shadow: 0 2px 10px rgba(0, 0, 0, 0.3);  
}
```

```
.results-actions {  
  display: flex;  
  gap: 1rem;  
  flex-wrap: wrap;  
}
```

```
.action-btn {  
  background: rgba(255, 255, 255, 0.2);  
  backdrop-filter: blur(10px);  
  color: var(--text-light);  
  border: 1px solid rgba(255, 255, 255, 0.3);  
  padding: 0.8rem 1.5rem;  
  border-radius: 50px;  
  font-weight: 500;  
  cursor: pointer;  
  transition: var(--transition);  
  text-decoration: none;  
  display: flex;  
  align-items: center;  
  gap: 0.5rem;  
}
```

```
.action-btn:hover {  
    background: rgba(255, 255, 255, 0.3);  
    transform: translateY(-2px);  
}
```

```
/* Stats Dashboard */
```

```
.stats-dashboard {  
    background: rgba(255, 255, 255, 0.95);  
    backdrop-filter: blur(20px);  
    border-radius: var(--border-radius);  
    padding: 2rem;  
    margin-bottom: 2rem;  
    box-shadow: var(--shadow);  
}
```

```
.stats-grid {  
    display: grid;  
    grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));  
    gap: 1.5rem;  
}
```

```
.stat-card {  
    text-align: center;  
    padding: 1.5rem;  
    background: linear-gradient(145deg, #f8fafc, #e2e8f0);  
    border-radius: var(--border-radius-sm);
```

```
    transition: var(--transition);  
}
```

```
.stat-card:hover {  
    transform: translateY(-3px);  
    box-shadow: 0 10px 25px rgba(0, 0, 0, 0.1);  
}
```

```
.stat-number {  
    font-size: 2.5rem;  
    font-weight: 800;  
    color: #667eea;  
    margin-bottom: 0.5rem;  
}
```

```
.stat-label {  
    color: var(--text-secondary);  
    font-weight: 500;  
    text-transform: uppercase;  
    font-size: 0.8rem;  
    letter-spacing: 0.5px;  
}
```

```
/* Question Cards */
```

```
.questions-container {  
    display: grid;
```

```
    gap: 2rem;
}
```

```
.question-card {
    background: rgba(255, 255, 255, 0.95);
    backdrop-filter: blur(20px);
    border-radius: var(--border-radius);
    overflow: hidden;
    box-shadow: var(--shadow);
    transition: var(--transition);
    border: 1px solid rgba(255, 255, 255, 0.3);
}
```

```
.question-card:hover {
    transform: translateY(-5px);
    box-shadow: var(--shadow-hover);
}
```

```
.question-header {
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 1.5rem 2rem;
    background: linear-gradient(135deg, #f8fafc, #e2e8f0);
    border-bottom: 1px solid #e5e7eb;
}
```

```
.question-number {  
  font-size: 1.2rem;  
  font-weight: 700;  
  color: var(--text-primary);  
}
```

```
.question-type-badge {  
  padding: 0.5rem 1rem;  
  border-radius: 50px;  
  font-size: 0.8rem;  
  font-weight: 600;  
  text-transform: uppercase;  
  letter-spacing: 0.5px;  
  color: white;  
}
```

```
.question-type-badge.mcq {  
  background: linear-gradient(135deg, #10b981, #059669);  
}
```

```
.question-type-badge.true_false {  
  background: linear-gradient(135deg, #f59e0b, #d97706);  
}
```

```
.question-type-badge.fill_blank {
```



```
    background: linear-gradient(135deg, #8b5cf6, #7c3aed);  
}
```

```
.question-type-badge.essay {  
    background: linear-gradient(135deg, #ef4444, #dc2626);  
}
```

```
.question-content {  
    padding: 2rem;  
}
```

```
.question-text {  
    font-size: 1.3rem;  
    font-weight: 600;  
    color: var(--text-primary);  
    margin-bottom: 1.5rem;  
    line-height: 1.6;  
}
```

```
.options-list {  
    display: grid;  
    gap: 0.8rem;  
    margin: 1.5rem 0;  
}
```

```
.option-item {
```

```
background: linear-gradient(145deg, #f8fafc, #e2e8f0);  
padding: 1rem 1.5rem;  
border-radius: var(--border-radius-sm);  
border: 2px solid transparent;  
transition: var(--transition);  
cursor: pointer;  
font-weight: 500;  
}
```

```
.option-item:hover {  
    border-color: #667eea;  
    transform: translateX(5px);  
}
```

```
.fill-blank-display {  
    background: linear-gradient(135deg, #fef3c7, #fde68a);  
    padding: 1.5rem;  
    border-radius: var(--border-radius-sm);  
    border-left: 4px solid #f59e0b;  
    margin: 1rem 0;  
}
```

```
.blank-highlight {  
    background: #fbbf24;  
    padding: 0.3rem 1rem;  
    border-radius: 8px;
```

```
margin: 0 0.3rem;  
font-weight: 600;  
color: #92400e;  
}
```

```
.essay-guidelines {  
  background: linear-gradient(135deg, #f3f4f6, #e5e7eb);  
  padding: 1.5rem;  
  border-radius: var(--border-radius-sm);  
  border-left: 4px solid #6b7280;  
  margin: 1rem 0;  
}
```

```
.answer-section {  
  background: linear-gradient(135deg, #d1fae5, #a7f3d0);  
  padding: 1.5rem;  
  border-radius: var(--border-radius-sm);  
  border-left: 4px solid #10b981;  
  margin: 1.5rem 0;  
}
```

```
.answer-label {  
  font-weight: 700;  
  color: #065f46;  
  margin-bottom: 0.5rem;  
  display: flex;
```

```
    align-items: center;
    gap: 0.5rem;
}
```

```
.explanation-section {
    background: linear-gradient(135deg, #dbeafe, #bfdbfe);
    padding: 1.5rem;
    border-radius: var(--border-radius-sm);
    border-left: 4px solid #3b82f6;
    margin-top: 1rem;
}
```

```
.explanation-label {
    font-weight: 700;
    color: #1e40af;
    margin-bottom: 0.5rem;
    display: flex;
    align-items: center;
    gap: 0.5rem;
}
```

```
/* Enhanced Message System */
```

```
.message {
    position: fixed;
    top: 2rem;
    right: 2rem;
```

```
max-width: 450px;

padding: 1.5rem;

border-radius: var(--border-radius-sm);

color: white;

font-weight: 500;

box-shadow: var(--shadow);

z-index: 2001;

transform: translateX(100%);

transition: var(--transition);

backdrop-filter: blur(10px);
}
```

```
.message.show {

  transform: translateX(0);
}
```

```
.message.success {

  background: var(--success-gradient);

  border: 1px solid rgba(79, 172, 254, 0.3);
}
```

```
.message.error {

  background: var(--danger-gradient);

  border: 1px solid rgba(239, 68, 68, 0.3);
}
```

```
.message.warning {  
    background: var(--warning-gradient);  
    border: 1px solid rgba(245, 158, 11, 0.3);  
}
```

```
.message-content {  
    display: flex;  
    align-items: flex-start;  
    gap: 1rem;  
}
```

```
.message-icon {  
    font-size: 1.5rem;  
    margin-top: 0.2rem;  
}
```

```
.message-text {  
    flex: 1;  
    line-height: 1.5;  
}
```

```
.message-title {  
    font-weight: 700;  
    margin-bottom: 0.5rem;  
}
```

```
.message-description {  
  opacity: 0.9;  
  font-size: 0.9rem;  
}
```

```
/* Troubleshooting Panel */
```

```
.troubleshooting-panel {  
  background: rgba(255, 255, 255, 0.95);  
  backdrop-filter: blur(20px);  
  border-radius: var(--border-radius);  
  padding: 2rem;  
  margin: 2rem 0;  
  display: none;  
  border: 2px solid #f59e0b;  
}
```

```
.troubleshooting-header {  
  display: flex;  
  align-items: center;  
  gap: 1rem;  
  margin-bottom: 1.5rem;  
}
```

```
.troubleshooting-icon {  
  font-size: 2rem;  
  color: #f59e0b;
```

```
}
```

```
.troubleshooting-title {  
  font-size: 1.3rem;  
  font-weight: 700;  
  color: #92400e;  
}
```

```
.troubleshooting-steps {  
  display: grid;  
  gap: 1rem;  
}
```

```
.troubleshooting-step {  
  background: #fffbef;  
  padding: 1rem;  
  border-radius: var(--border-radius-sm);  
  border-left: 4px solid #f59e0b;  
}
```

```
.step-number {  
  background: #f59e0b;  
  color: white;  
  width: 24px;  
  height: 24px;  
  border-radius: 50%;
```



```
display: inline-flex;
align-items: center;
justify-content: center;
font-size: 0.8rem;
font-weight: 600;
margin-right: 0.8rem;
}
```

```
/* Animations */
```

```
@keyframes slideInUp {
  from {
    opacity: 0;
    transform: translateY(30px);
  }
  to {
    opacity: 1;
    transform: translateY(0);
  }
}
```

```
@keyframes fadeIn {
  from { opacity: 0; }
  to { opacity: 1; }
}
```

```
@keyframes pulse {
```

```
0%, 100% { opacity: 1; }  
50% { opacity: 0.5; }  
}
```

```
/* Responsive Design */
```

```
@media (max-width: 768px) {  
  .main-container {  
    margin: 1rem;  
  }  
}
```

```
.hero-header {  
  padding: 3rem 1.5rem;  
}
```

```
.upload-section, .results-section {  
  padding: 2rem 1.5rem;  
}
```

```
.upload-card {  
  padding: 1.5rem;  
}
```

```
.controls-grid {  
  grid-template-columns: 1fr;  
}
```

```
.results-header {  
    flex-direction: column;  
    align-items: flex-start;  
}  
  
.stats-grid {  
    grid-template-columns: repeat(2, 1fr);  
}  
  
.loading-card {  
    min-width: 90vw;  
    margin: 0 1rem;  
}  
  
.message {  
    right: 1rem;  
    left: 1rem;  
    max-width: none;  
}  
}  
  
/* Print Styles */  
@media print {  
    body {  
        background: white;  
        font-size: 12pt;  
    }  
}
```

```
}

.hero-header, .upload-section, .stats-dashboard, .results-actions, .background-
animation, .message, .loading-overlay {

    display: none !important;

}

.question-card {

    break-inside: avoid;

    box-shadow: none;

    border: 1px solid #ddd;

    margin-bottom: 1rem;

}

}

</style>
</head>
<body>

    <!-- Animated Background -->

    <div class="background-animation">

        <div class="floating-shapes">

            <div class="shape"></div>

            <div class="shape"></div>

            <div class="shape"></div>

        </div>

    </div>

    <!-- Main Container -->

    <div class="main-container">
```

```
<!-- Hero Header -->

<div class="hero-header">

  <div class="hero-content">

    <h1 class="hero-title">

      <i class="fas fa-brain"></i> AI Quiz Generator Pro

    </h1>

    <p class="hero-subtitle">

      Transform your educational PDFs into dynamic, intelligent quiz questions with
bulletproof AI technology

    </p>

    <div class="feature-pills">

      <div class="feature-pill">

        <i class="fas fa-list-check"></i> Multiple Choice

      </div>

      <div class="feature-pill">

        <i class="fas fa-toggle-on"></i> True/False

      </div>

      <div class="feature-pill">

        <i class="fas fa-edit"></i> Fill-in-the-Blank

      </div>

      <div class="feature-pill">

        <i class="fas fa-file-alt"></i> Essay Questions

      </div>

    </div>

  </div>

</div>

</div>
```

<!-- Upload Section -->

<div class="upload-section">

<div class="upload-card">

<form id="quizForm" enctype="multipart/form-data">

<!-- File Upload Zone -->

<div class="upload-zone" id="uploadZone">

<input type="file" id="pdfFile" name="pdf_file" accept=".pdf" required hidden>

<i class="fas fa-cloud-upload-alt upload-icon"></i>

<div class="upload-text" id="uploadText">

Drag & Drop your PDF or Click to Browse

</div>

<div class="upload-subtext">

Supports PDF files up to 16MB | Educational content recommended

</div>

</div>

<!-- File Validation Messages -->

<div class="file-validation" id="fileValidation">

<div id="validationContent"></div>

</div>

<!-- Error Card -->

<div class="error-card" id="errorCard">

<div class="error-header">

<i class="fas fa-exclamation-triangle error-icon"></i>

```
    <div class="error-title" id="errorTitle">Upload Error</div>
  </div>
  <div class="error-message" id="errorMessage"></div>
  <div class="error-suggestions">
    <h4><i class="fas fa-lightbulb"></i> Troubleshooting Tips:</h4>
    <ul id="errorSuggestions"></ul>
  </div>
  <button type="button" class="retry-btn" id="retryBtn">
    <i class="fas fa-redo"></i> Try Again
  </button>
</div>
```

```
<!-- Controls Grid -->
<div class="controls-grid">
  <div class="control-card">
    <label class="control-label">
      <i class="fas fa-calculator control-icon"></i>
      Number of Questions
    </label>
    <input type="number" class="form-input" id="numQuestions"
name="num_questions"
      min="4" max="20" value="8" required>
    <div class="input-hint">
      <i class="fas fa-info-circle"></i>
      Minimum 4 questions for balanced question types
    </div>
```

```
<div class="input-error" id="questionsError">
  <i class="fas fa-exclamation-circle"></i>
  <span></span>
</div>
</div>
```

```
<div class="control-card">
  <label class="control-label">
    <i class="fas fa-target control-icon"></i>
    Difficulty Level
  </label>
  <select class="form-input" id="difficulty" name="difficulty">
    <option value="Easy">Easy - Basic Understanding</option>
    <option value="Medium" selected>Medium - Applied Knowledge</option>
    <option value="Hard">Hard - Advanced Analysis</option>
  </select>
  <div class="input-hint">
    <i class="fas fa-lightbulb"></i>
    Determines cognitive complexity of questions
  </div>
</div>
</div>
```

```
<!-- Generate Button -->
<div class="generate-section">
  <button type="submit" class="generate-btn" id="generateBtn">
```


<i class="fas fa-rocket btn-icon"></i>

Generate AI-Powered Quiz

</button>

</div>

</form>

<!-- Troubleshooting Panel -->

<div class="troubleshooting-panel" id="troubleshootingPanel">

<div class="troubleshooting-header">

<i class="fas fa-tools troubleshooting-icon"></i>

<div class="troubleshooting-title">Having Issues? Let's Fix Them!</div>

</div>

<div class="troubleshooting-steps">

<div class="troubleshooting-step">

1

Check your PDF: Make sure it's a valid PDF file under 16MB with readable text content.

</div>

<div class="troubleshooting-step">

2

Network connection: Ensure you have a stable internet connection for AI processing.

</div>

<div class="troubleshooting-step">

3

File content: PDFs with educational text work best. Avoid image-only or encrypted PDFs.

```
</div>

<div class="troubleshooting-step">

    <span class="step-number">4</span>

    <strong>Try again:</strong> Sometimes AI processing can take time. Wait and
retry if needed.
```

```
</div>

</div>

</div>

</div>

</div>

<!-- Results Section -->

<div class="results-section" id="resultsSection">

    <div class="results-header">

        <h2 class="results-title">

            <i class="fas fa-graduation-cap"></i> Your AI-Generated Quiz

        </h2>

        <div class="results-actions">

            <a href="#" class="action-btn" id="downloadBtn">

                <i class="fas fa-download"></i> Download JSON

            </a>

            <button class="action-btn" onclick="window.print()">

                <i class="fas fa-print"></i> Print Quiz

            </button>

            <button class="action-btn" id="shareBtn">

                <i class="fas fa-share"></i> Share Quiz
```

```
</button>
```

```
</div>
```

```
</div>
```

```
<!-- Stats Dashboard -->
```

```
<div class="stats-dashboard">
```

```
<div class="stats-grid" id="statsGrid">
```

```
<!-- Stats will be populated by JavaScript -->
```

```
</div>
```

```
</div>
```

```
<!-- Questions Container -->
```

```
<div class="questions-container" id="questionsContainer">
```

```
<!-- Questions will be populated by JavaScript -->
```

```
</div>
```

```
</div>
```

```
</div>
```

```
<!-- Enhanced Loading Overlay -->
```

```
<div class="loading-overlay" id="loadingOverlay">
```

```
<div class="loading-card">
```

```
<div class="loading-spinner"></div>
```

```
<div class="loading-text">AI is crafting your quiz...</div>
```

```
<div class="loading-subtext">
```

Please wait while our advanced AI analyzes your content and generates diverse question types

</div>

<div class="loading-steps">

<div class="loading-step" id="step1">

<i class="fas fa-file-pdf step-icon"></i>

Extracting text from PDF

</div>

<div class="loading-step" id="step2">

<i class="fas fa-brain step-icon"></i>

AI analyzing content structure

</div>

<div class="loading-step" id="step3">

<i class="fas fa-question-circle step-icon"></i>

Generating diverse questions

</div>

<div class="loading-step" id="step4">

<i class="fas fa-check-circle step-icon"></i>

Finalizing quiz format

</div>

</div>

<div class="progress-bar">

<div class="progress-fill" id="progressFill"></div>

</div>

</div>

</div>

<script>

```

class QuizGeneratorPro {
  constructor() {
    this.currentResultFile = null;

    this.isProcessing = false;

    this.loadingSteps = ['step1', 'step2', 'step3', 'step4'];

    this.currentStep = 0;

    this.initializeEventListeners();

    this.initializeValidation();

    this.initializeAnimations();
  }

  initializeEventListeners() {
    // File upload handling with comprehensive validation

    const uploadZone = document.getElementById('uploadZone');
    const fileInput = document.getElementById('pdfFile');
    const uploadText = document.getElementById('uploadText');

    uploadZone.addEventListener('click', () => {
      if (!this.isProcessing) {
        fileInput.click();
      }
    });

    fileInput.addEventListener('change', (e) => {
      this.handleFileSelection(e.target.files[0]);
    });
  }
}

```

```
// Enhanced drag and drop

uploadZone.addEventListener('dragover', (e) => {

    e.preventDefault();

    if (!this.isProcessing) {

        uploadZone.classList.add('dragover');

    }

});
```

```
uploadZone.addEventListener('dragleave', () => {

    uploadZone.classList.remove('dragover');

});
```

```
uploadZone.addEventListener('drop', (e) => {

    e.preventDefault();

    uploadZone.classList.remove('dragover');

    if (!this.isProcessing) {

        const files = e.dataTransfer.files;

        if (files.length > 0) {

            this.handleFileSelection(files[0]);

        }

    }

});
```

```
// Form submission with validation
```

```
document.getElementById('quizForm').addEventListener('submit', (e) => {  
    e.preventDefault();  
    this.generateQuiz();  
});  
  
// Retry button  
document.getElementById('retryBtn').addEventListener('click', () => {  
    this.resetForm();  
});  
  
// Download and share buttons  
document.getElementById('downloadBtn').addEventListener('click', () => {  
    this.downloadQuiz();  
});  
  
document.getElementById('shareBtn').addEventListener('click', () => {  
    this.shareQuiz();  
});  
  
// Form validation on input change  
document.getElementById('numQuestions').addEventListener('input', (e) => {  
    this.validateQuestionCount(e.target.value);  
});  
}  
  
initializeValidation() {
```

```
this.validationRules = {  
  fileSize: 16 * 1024 * 1024, // 16MB  
  fileType: 'application/pdf',  
  minQuestions: 4,  
  maxQuestions: 20  
};  
}
```

```
initializeAnimations() {  
  // Animate elements on scroll  
  const observerOptions = {  
    threshold: 0.1,  
    rootMargin: '0px 0px -100px 0px'  
  };  
};
```

```
const observer = new IntersectionObserver((entries) => {  
  entries.forEach(entry => {  
    if (entry.isIntersecting) {  
      entry.target.style.animation = 'slideDown 0.8s ease-out';  
    }  
  });  
}, observerOptions);
```

```
document.querySelectorAll('.control-card, .question-card').forEach(el => {  
  observer.observe(el);  
});
```



```
}
```

```
handleFileSelection(file) {  
  
    const uploadZone = document.getElementById('uploadZone');  
    const uploadText = document.getElementById('uploadText');  
    const fileValidation = document.getElementById('fileValidation');  
    const validationContent = document.getElementById('validationContent');  
  
    if (!file) {  
        this.showFileValidation('Please select a file', 'error');  
        return;  
    }  
  
    // Reset previous states  
    uploadZone.classList.remove('file-selected', 'error');  
    fileValidation.style.display = 'none';  
    this.hideError();  
  
    // Validate file type  
    if (file.type !== this.validationRules.fileType) {  
        this.showError(  
            'Invalid File Type',  
            'Please upload a PDF file only.',  
            [  
                'Only PDF files are supported',  
                'Convert your document to PDF format',
```

```
        'Ensure file extension is .pdf'
    ]
);
uploadZone.classList.add('error');
return;
}
```

```
// Validate file size
```

```
if (file.size > this.validationRules.fileSize) {
    const sizeMB = (file.size / (1024 * 1024)).toFixed(2);
    this.showError(
        'File Too Large',
        `Your file is ${sizeMB}MB. Maximum allowed size is 16MB.`,
        [
            'Compress your PDF file',
            'Split large documents into smaller sections',
            'Remove unnecessary images or content'
        ]
    );
    uploadZone.classList.add('error');
    return;
}
```

```
// Validate file name
```

```
if (file.name.length > 100) {
    this.showFileValidation('File name is too long. Please rename your file.', 'warning');
```

```


    return;
}

// File is valid

const sizeMB = (file.size / (1024 * 1024)).toFixed(2);

uploadText.innerHTML = `
    <i class="fas fa-file-pdf" style="color: #ef4444;"></i>
    ${file.name} <small>${sizeMB}MB</small>
`;

uploadZone.classList.add('file-selected');

this.showFileValidation(
    `  Valid PDF file selected (${sizeMB}MB). Ready to generate quiz!`,
    'success'
);

// Update file input

const dataTransfer = new DataTransfer();

dataTransfer.items.add(file);

document.getElementById('pdfFile').files = dataTransfer.files;

this.showMessage(
    'File Validated',
    'PDF file successfully validated and ready for processing!',
    'success'
);

```

```
}
```

```
validateQuestionCount(value) {
```

```
    const input = document.getElementById('numQuestions');
```

```
    const error = document.getElementById('questionsError');
```

```
    const num = parseInt(value);
```

```
    input.classList.remove('error');
```

```
    error.style.display = 'none';
```

```
    if (num < this.validationRules.minQuestions) {
```

```
        input.classList.add('error');
```

```
        error.querySelector('span').textContent = ` Minimum  
${this.validationRules.minQuestions} questions required for all question types `;
```

```
        error.style.display = 'flex';
```

```
        return false;
```

```
    }
```

```
    if (num > this.validationRules.maxQuestions) {
```

```
        input.classList.add('error');
```

```
        error.querySelector('span').textContent = ` Maximum  
${this.validationRules.maxQuestions} questions allowed `;
```

```
        error.style.display = 'flex';
```

```
        return false;
```

```
    }
```

```
    return true;
```

```

    }

    async generateQuiz() {
        if (this.isProcessing) {
            this.showMessage('Processing', 'Quiz generation already in progress. Please wait.',
'warning');
            return;
        }

        // Validate form

        const fileInput = document.getElementById('pdfFile');

        const numQuestions =
parseInt(document.getElementById('numQuestions').value);

        if (!fileInput.files || fileInput.files.length === 0) {
            this.showMessage('No File', 'Please select a PDF file first.', 'error');
            document.getElementById('uploadZone').classList.add('error');
            return;
        }

        if (!this.validateQuestionCount(numQuestions)) {
            this.showMessage('Invalid Input', 'Please check the number of questions.', 'error');
            return;
        }

        this.isProcessing = true;

        this.showEnhancedLoading();
    }

```

```
this.hideError();
```

```
this.hideTroubleshooting();
```

```
try{
```

```
    const formData = new FormData(document.getElementById('quizForm'));
```

```
    console.log('🚀 Starting quiz generation...');
```

```
    console.log('📄 File:', fileInput.files[0].name);
```

```
    console.log('📋 Questions:', numQuestions);
```

```
    console.log('🎯 Difficulty:', document.getElementById('difficulty').value);
```

```
    const response = await fetch('/upload', {
```

```
        method: 'POST',
```

```
        body: formData
```

```
    });
```

```
    console.log('🔍 Response status:', response.status);
```

```
    if (!response.ok) {
```

```
        throw new Error(` HTTP ${response.status}: ${response.statusText} `);
```

```
    }
```

```
    const result = await response.json();
```

```
    console.log('✅ Response received:', result);
```

```

    if (result.success) {
        this.completeStep(4);
        await this.delay(500);

        this.displayResults(result.quiz_data, result.question_types);
        this.currentResultFile = result.result_file;

        this.showMessage(
            'Success!',
            ` 🎉 Generated ${Object.values(result.question_types || {}).reduce((a, b) => a
+ b, 0)} questions with AI!`,
            'success'
        );
    } else {
        this.handleGenerationError(result.error);
    }
} catch (error) {
    console.error(' ❌ Generation error:', error);
    this.handleGenerationError(error.message);
} finally {
    this.isProcessing = false;
    this.hideLoading();
}
}

```

```

handleGenerationError(errorMessage) {

```

```
console.error('🔥 Quiz generation failed:', errorMessage);
```

```
let title = 'Generation Failed';
```

```
let suggestions = [];
```

```
if (errorMessage.includes('Could not extract text')) {
```

```
    title = 'PDF Text Extraction Failed';
```

```
    suggestions = [
```

```
        'Ensure your PDF contains readable text (not just images)',
```

```
        'Try a different PDF file',
```

```
        'Check if the PDF is password protected',
```

```
        'Convert scanned documents to text-searchable PDFs'
```

```
    ];
```

```
} else if (errorMessage.includes('Could not generate quiz questions')) {
```

```
    title = 'AI Generation Failed';
```

```
    suggestions = [
```

```
        'Check your internet connection',
```

```
        'Try with a smaller number of questions',
```

```
        'Ensure the PDF contains educational content',
```

```
        'Wait a moment and try again'
```

```
    ];
```

```
} else if (errorMessage.includes('No file uploaded')) {
```

```
    title = 'File Upload Error';
```

```
    suggestions = [
```

```
        'Refresh the page and try again',
```

```
        'Check your file selection',
```



```
        'Ensure the file is a valid PDF',  
        'Try a different browser'  
    ];  
    } else {  
        suggestions = [  
            'Check your internet connection',  
            'Try refreshing the page',  
            'Select a different PDF file',  
            'Contact support if the issue persists'  
        ];  
    }  
}
```

```
this.showError(title, errorMessage, suggestions);  
this.showTroubleshooting();
```

```
this.showMessage(  
    'Generation Failed',  
    'There was an issue generating your quiz. Please check the troubleshooting tips  
below.',  
    'error'  
);  
}
```

```
showEnhancedLoading() {  
    document.getElementById('loadingOverlay').style.display = 'block';  
    document.getElementById('generateBtn').disabled = true;
```

```

this.currentStep = 0;

// Reset progress
document.getElementById('progressFill').style.width = '0%';
this.loadingSteps.forEach(stepId => {
    const step = document.getElementById(stepId);
    step.classList.remove('active', 'completed');
});

// Start step progression
this.progressSteps();
}

async progressSteps() {
    for (let i = 0; i < this.loadingSteps.length; i++) {
        await this.delay(800 + Math.random() * 1000);
        this.completeStep(i + 1);
    }
}

completeStep(stepNumber) {
    // Mark previous steps as completed
    for (let i = 0; i < stepNumber - 1; i++) {
        const step = document.getElementById(this.loadingSteps[i]);
        step.classList.remove('active');
        step.classList.add('completed');
    }
}

```

```
}
```

```
// Mark current step as active
```

```
if (stepNumber <= this.loadingSteps.length) {
```

```
    const currentStepElement =
```

```
document.getElementById(this.loadingSteps[stepNumber - 1]);
```

```
    currentStepElement.classList.add('active');
```

```
}
```

```
// Update progress bar
```

```
const progress = (stepNumber / this.loadingSteps.length) * 100;
```

```
document.getElementById('progressFill').style.width = `${progress}%`;
```

```
}
```

```
hideLoading() {
```

```
    document.getElementById('loadingOverlay').style.display = 'none';
```

```
    document.getElementById('generateBtn').disabled = false;
```

```
}
```

```
showFileValidation(message, type) {
```

```
    const fileValidation = document.getElementById('fileValidation');
```

```
    const validationContent = document.getElementById('validationContent');
```

```
    fileValidation.className = `file-validation ${type}`;
```

```
    validationContent.innerHTML = `
```

```
        <i class="fas fa-${type === 'success' ? 'check-circle' : type === 'warning' ?  
'exclamation-triangle' : 'times-circle'}"></i>
```

```

    ${message}
`;
fileValidation.style.display = 'block';
}

showError(title, message, suggestions = []) {
    const errorCard = document.getElementById('errorCard');
    const errorTitle = document.getElementById('errorTitle');
    const errorMessage = document.getElementById('errorMessage');
    const errorSuggestions = document.getElementById('errorSuggestions');

    errorTitle.textContent = title;
    errorMessage.textContent = message;

    errorSuggestions.innerHTML = suggestions
        .map(suggestion => `<li>${suggestion}</li>`)
        .join("");

    errorCard.style.display = 'block';

    // Scroll to error
    errorCard.scrollIntoView({ behavior: 'smooth', block: 'center' });
}

hideError() {
    document.getElementById('errorCard').style.display = 'none';
}

```

```
}
```

```
showTroubleshooting() {
```

```
    document.getElementById('troubleshootingPanel').style.display = 'block';
```

```
}
```

```
hideTroubleshooting() {
```

```
    document.getElementById('troubleshootingPanel').style.display = 'none';
```

```
}
```

```
resetForm() {
```

```
    // Reset form state
```

```
    document.getElementById('quizForm').reset();
```

```
    document.getElementById('uploadText').textContent = 'Drag & Drop your PDF or  
Click to Browse';
```

```
    document.getElementById('uploadZone').classList.remove('file-selected', 'error');
```

```
    document.getElementById('fileValidation').style.display = 'none';
```

```
    this.hideError();
```

```
    this.hideTroubleshooting();
```

```
    this.isProcessing = false;
```

```
    this.showMessage('Reset', 'Form has been reset. You can now upload a new file.',  
'success');
```

```
}
```

```
displayResults(quizData, questionTypes) {
```

```
this.displayStats(questionTypes);  
this.displayQuestions(quizData.questions);
```

```
document.getElementById('resultsSection').style.display = 'block';  
document.getElementById('resultsSection').scrollIntoView({  
  behavior: 'smooth'  
});  
}
```

```
displayStats(questionTypes) {  
  const statsGrid = document.getElementById('statsGrid');  
  const totalQuestions = Object.values(questionTypes || {}).reduce((a, b) => a + b, 0);
```

```
  const typeNames = {  
    'mcq': 'Multiple Choice',  
    'true_false': 'True/False',  
    'fill_blank': 'Fill Blanks',  
    'essay': 'Essay Questions'  
  };
```

```
  let statsHtml = `  
    <div class="stat-card">  
      <div class="stat-number">${totalQuestions}</div>  
      <div class="stat-label">Total Questions</div>  
    </div>  
  `;  
};
```

```

Object.entries(questionTypes || {}).forEach(([type, count]) => {
  statsHtml += `
    <div class="stat-card">
      <div class="stat-number">${count}</div>
      <div class="stat-label">${typeNames[type] || type}</div>
    </div>
  `;
});

statsGrid.innerHTML = statsHtml;
}

displayQuestions(questions) {
  const container = document.getElementById('questionsContainer');
  container.innerHTML = "";

  questions.forEach((q, index) => {
    const questionCard = this.createQuestionCard(q, index + 1);
    container.appendChild(questionCard);
  });
}

createQuestionCard(question, number) {
  const card = document.createElement('div');
  card.className = 'question-card';

```

```
const typeClass = question.type || 'mcq';
```

```
const typeName = {
```

```
  'mcq': 'Multiple Choice',
```

```
  'true_false': 'True/False',
```

```
  'fill_blank': 'Fill in the Blank',
```

```
  'essay': 'Essay Question'
```

```
}[typeClass];
```

```
let contentHtml = '';
```

```
if (typeClass === 'fill_blank') {
```

```
  const questionText = question.question.replace(/_{3,}/g,
```

```
    '<span class="blank-highlight">____</span>');
```

```
  contentHtml = `
```

```
    <div class="question-text">${questionText}</div>
```

```
    <div class="fill-blank-display">
```

```
      <strong><i class="fas fa-pencil-alt"></i> Instructions:</strong>
```

```
      Fill in the blank(s) with appropriate words or phrases.
```

```
    </div>
```

```
  `;
```

```
} else if (typeClass === 'essay') {
```

```
  contentHtml = `
```

```
    <div class="question-text">${question.question}</div>
```

```
    <div class="essay-guidelines">
```

```
      <strong><i class="fas fa-list-ul"></i> Essay Guidelines:</strong>
```



```

        <div style="margin-top: 0.8rem; line-height: 1.6;">
            ${question.correct_answer}
        </div>
    </div>

    `;
} else {
    const optionsHtml = (question.options || []).map(option =>
        `<div class="option-item">${option}</div>`
    ).join("");

    contentHtml = `
        <div class="question-text">${question.question}</div>
        <div class="options-list">${optionsHtml}</div>
    `;
}

card.innerHTML = `
    <div class="question-header">
        <div class="question-number">
            <i class="fas fa-question-circle"></i> Question ${number}
        </div>
        <div class="question-type-badge ${typeClass}">${typeName}</div>
    </div>
    <div class="question-content">
        ${contentHtml}
        <div class="answer-section">

```

```

        <div class="answer-label">
            <i class="fas fa-check-circle"></i> Correct Answer
        </div>

        <div>${question.correct_answer}</div>
    </div>

    <div class="explanation-section">
        <div class="explanation-label">
            <i class="fas fa-lightbulb"></i> Explanation
        </div>

        <div>${question.explanation}</div>
    </div>
</div>

`;

return card;
}

downloadQuiz() {
    if (this.currentResultFile) {
        window.open(`/download/${this.currentResultFile}`, '_blank');

        this.showMessage('Download Started', ' 📄 Quiz file download has started!',
'success');
    } else {
        this.showMessage('No Quiz Available', ' ❌ No quiz file available for download',
'error');
    }
}

```

```
}
```

```
shareQuiz() {  
  if (navigator.share) {  
    navigator.share({  
      title: 'AI-Generated Quiz',  
      text: 'Check out this AI-generated quiz created with Quiz Generator Pro!',  
      url: window.location.href  
    }).then(() => {  
      this.showMessage('Shared', 'Quiz shared successfully!', 'success');  
    }).catch(() => {  
      this.fallbackShare();  
    });  
  } else {  
    this.fallbackShare();  
  }  
}
```

```
fallbackShare() {  
  if (navigator.clipboard) {  
    navigator.clipboard.writeText(window.location.href).then(() => {  
      this.showMessage('Link Copied', '📄 Quiz link copied to clipboard!', 'success');  
    }).catch(() => {  
      this.showMessage('Share Failed', 'Unable to share or copy link', 'error');  
    });  
  } else {
```

```
        this.showMessage('Share Not Supported', 'Sharing is not supported in this  
browser', 'warning');
```

```
    }
```

```
}
```

```
showMessage(title, description, type) {
```

```
    // Remove existing messages
```

```
    document.querySelectorAll('.message').forEach(msg => msg.remove());
```

```
    const message = document.createElement('div');
```

```
    message.className = `message ${type}`;
```

```
    const icons = {
```

```
        success: 'check-circle',
```

```
        error: 'exclamation-circle',
```

```
        warning: 'exclamation-triangle'
```

```
    };
```

```
    message.innerHTML = `
```

```
        <div class="message-content">
```

```
            <i class="fas fa-${icons[type]} message-icon"></i>
```

```
            <div class="message-text">
```

```
                <div class="message-title">${title}</div>
```

```
                <div class="message-description">${description}</div>
```

```
            </div>
```

```
        </div>
```

```
`;
```

```
document.body.appendChild(message);
```

```
// Trigger animation
```

```
setTimeout(() => message.classList.add('show'), 100);
```

```
// Auto-remove
```

```
setTimeout(() => {
```

```
    message.classList.remove('show');
```

```
    setTimeout(() => message.remove(), 500);
```

```
}, 6000);
```

```
}
```

```
delay(ms) {
```

```
    return new Promise(resolve => setTimeout(resolve, ms));
```

```
}
```

```
}
```

```
// Initialize the bulletproof application
```

```
document.addEventListener('DOMContentLoaded', () => {
```

```
    const app = new QuizGeneratorPro();
```

```
    console.log('💡 AI Quiz Generator Pro initialized with bulletproof error handling!');
```

```
    console.log('💎 All previous issues resolved and handled gracefully');
```

```
});
```

```
// Global error handler
```

```
window.addEventListener('error', (e) => {  
  console.error('🔥 Global error caught:', e.error);  
});
```

```
// Unhandled promise rejection handler
```

```
window.addEventListener('unhandledrejection', (e) => {  
  console.error('🔥 Unhandled promise rejection:', e.reason);  
  e.preventDefault();  
});
```

```
// Keyboard shortcuts
```

```
document.addEventListener('keydown', (e) => {  
  if ((e.ctrlKey || e.metaKey) && e.key === 'u') {  
    e.preventDefault();  
    document.getElementById('pdfFile').click();  
  }  
  
  if ((e.ctrlKey || e.metaKey) && e.key === 'Enter') {  
    e.preventDefault();  
    const fileInput = document.getElementById('pdfFile');  
    if (fileInput.files.length > 0) {  
      document.getElementById('generateBtn').click();  
    }  
  }  
});
```

</script>

</body>

</html>