<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Qroll - Smart Attendance Tracking</title>

<style>

:root {

--background: #0f172a;

--foreground: #f8fafc;

--primary: #8b5cf6;

--primary-foreground: #ffffff;

--secondary: #1e293b;

--muted: #334155;

--muted-foreground: #94a3b8;

}

body {

font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, Oxygen, Ubuntu, Cantarell, 'Open Sans', 'Helvetica Neue', sans-serif;

margin: 0;

padding: 0;

background-color: var(--background);

color: var(--foreground);

line-height: 1.6;

}

.container {

max-width: 1200px;

margin: 0 auto;

padding: 0 20px;

}

header {

padding: 30px 0;

text-align: center;

}

.logo {

font-size: 2.5rem;

font-weight: bold;

color: var(--primary);

margin-bottom: 0;

}

.tagline {

font-size: 1.2rem;

color: var(--muted-foreground);

margin-top: 10px;

}

.hero {

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

padding: 60px 0;

text-align: center;

}

h1 {

font-size: 3rem;

margin-bottom: 20px;

max-width: 800px;

}

.hero p {

font-size: 1.25rem;

max-width: 700px;

margin-bottom: 40px;

color: var(--muted-foreground);

}

.cta-button {

display: inline-block;

background-color: var(--primary);

color: var(--primary-foreground);

font-size: 1.1rem;

font-weight: 600;

text-decoration: none;

padding: 14px 32px;

border-radius: 8px;

box-shadow: 0 4px 6px rgba(139, 92, 246, 0.25);

transition: all 0.3s ease;

}

.cta-button:hover {

transform: translateY(-2px);

box-shadow: 0 7px 10px rgba(139, 92, 246, 0.3);

}

.features {

padding: 80px 0;

background-color: var(--secondary);

}

.features h2 {

text-align: center;

font-size: 2rem;

margin-bottom: 60px;

}

.feature-grid {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));

gap: 40px;

}

.feature-card {

background-color: var(--background);

border-radius: 12px;

padding: 30px;

text-align: center;

box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);

}

.feature-icon {

width: 60px;

height: 60px;

margin: 0 auto 20px;

display: flex;

align-items: center;

justify-content: center;

background-color: rgba(139, 92, 246, 0.1);

border-radius: 50%;

color: var(--primary);

}

.feature-card h3 {

font-size: 1.5rem;

margin-bottom: 15px;

}

.feature-card p {

color: var(--muted-foreground);

}

footer {

padding: 40px 0;

text-align: center;

color: var(--muted-foreground);

}

@media (max-width: 768px) {

h1 {

font-size: 2.2rem;

}

.hero p {

font-size: 1.1rem;

}

.feature-grid {

grid-template-columns: 1fr;

}

}

</style>

</head>

<body>

<div class="container">

<header>

<h2 class="logo">Qroll</h2>

<p class="tagline">Smart Attendance Tracking</p>

</header>

<section class="hero">

<h1>The Smart Way to Track Student Attendance</h1>

<p>

Qroll combines QR code scanning, facial recognition, and location verification

to create a secure and efficient attendance management system for educational institutions.

</p>

<a href="/auth" class="cta-button">Get Started</a>

</section>

</div>

<section class="features">

<div class="container">

<h2>Key Features</h2>

<div class="feature-grid">

<div class="feature-card">

<div class="feature-icon">

<svg xmlns="http://www.w3.org/2000/svg" width="32" height="32" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round">

<path d="M3 7v9a2 2 0 0 0 2 2h13a2 2 0 0 0 2-2V9a2 2 0 0 0-2-2h-4.5"></path>

<path d="M3 9h18"></path>

<path d="M7 13h.01"></path>

<path d="M11 13h.01"></path>

<path d="M15 13h.01"></path>

<path d="M7 17h.01"></path>

<path d="M11 17h.01"></path>

<path d="M15 17h.01"></path>

<path d="M10 2v5"></path>

<path d="M14 2v5"></path>

</svg>

</div>

<h3>QR Code Scanning</h3>

<p>Generate and scan QR codes for quick and easy attendance marking.</p>

</div>

<div class="feature-card">

<div class="feature-icon">

<svg xmlns="http://www.w3.org/2000/svg" width="32" height="32" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round">

<path d="M22 8a5 5 0 0 1-3.36 4.7l.99 5.34A1 1 0 0 1 18.66 19H13a2 2 0 0 1-2-2v-5"></path>

<path d="M12 12a5 5 0 0 1 0-10 4.96 4.96 0 0 1 4.36 2.58"></path>

<path d="M9 10a5 5 0 0 1 0-8"></path>

<path d="M4.8 17.8l-.68.68a2 2 0 0 0 0 2.82 2 2 0 0 0 2.82 0l.68-.68"></path>

<path d="M2 22l3-3"></path>

</svg>

</div>

<h3>Facial Recognition</h3>

<p>Verify student identity through secure facial recognition technology.</p>

</div>

<div class="feature-card">

<div class="feature-icon">

<svg xmlns="http://www.w3.org/2000/svg" width="32" height="32" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round">

<path d="M20 10c0 4.4-8 12-8 12s-8-7.6-8-12a8 8 0 0 1 16 0z"></path>

<path d="M12 10a1 1 0 1 0 0-2 1 1 0 0 0 0 2z"></path>

</svg>

</div>

<h3>Location Verification</h3>

<p>Ensure students are present in the right location using geolocation verification.</p>

</div>

</div>

</div>

</section>

<footer>

<div class="container">

<p>© 2025 Qroll. All rights reserved.</p>

<p>

<a href="/qr-url" style="color: var(--primary); text-decoration: none;">

Get Qroll QR Code

</a>

</p>

</div>

</footer>

</body>

</html>

[{"weights":[{"name":"dense0/conv0/filters","shape":[3,3,3,32],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.004853619781194949,"min":-0.5872879935245888}},{"name":"dense0/conv0/bias","shape":[32],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.004396426443960153,"min":-0.7298067896973853}},{"name":"dense0/conv1/depthwise\_filter","shape":[3,3,32,1],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.00635151559231328,"min":-0.5589333721235686}},{"name":"dense0/conv1/pointwise\_filter","shape":[1,1,32,32],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.009354315552057004,"min":-1.2628325995276957}},{"name":"dense0/conv1/bias","shape":[32],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.0029380727048013726,"min":-0.5846764682554731}},{"name":"dense0/conv2/depthwise\_filter","shape":[3,3,32,1],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.0049374802439820535,"min":-0.6171850304977566}},{"name":"dense0/conv2/pointwise\_filter","shape":[1,1,32,32],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.009941946758943446,"min":-1.3421628124573652}},{"name":"dense0/conv2/bias","shape":[32],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.0030300481062309416,"min":-0.5272283704841838}},{"name":"dense0/conv3/depthwise\_filter","shape":[3,3,32,1],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.005672684837790097,"min":-0.7431217137505026}},{"name":"dense0/conv3/pointwise\_filter","shape":[1,1,32,32],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.010712201455060173,"min":-1.5639814124387852}},{"name":"dense0/conv3/bias","shape":[32],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.0030966934035806097,"min":-0.3839899820439956}},{"name":"dense1/conv0/depthwise\_filter","shape":[3,3,32,1],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.0039155554537679636,"min":-0.48161332081345953}},{"name":"dense1/conv0/pointwise\_filter","shape":[1,1,32,64],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.01023082966898002,"min":-1.094698774580862}},{"name":"dense1/conv0/bias","shape":[64],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.0027264176630506327,"min":-0.3871513081531898}},{"name":"dense1/conv1/depthwise\_filter","shape":[3,3,64,1],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.004583378632863362,"min":-0.5454220573107401}},{"name":"dense1/conv1/pointwise\_filter","shape":[1,1,64,64],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.00915846403907327,"min":-1.117332612766939}},{"name":"dense1/conv1/bias","shape":[64],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.003091680419211294,"min":-0.5966943209077797}},{"name":"dense1/conv2/depthwise\_filter","shape":[3,3,64,1],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.005407439727409214,"min":-0.708374604290607}},{"name":"dense1/conv2/pointwise\_filter","shape":[1,1,64,64],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.00946493943532308,"min":-1.2399070660273235}},{"name":"dense1/conv2/bias","shape":[64],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.004409168514550901,"min":-0.9788354102303}},{"name":"dense1/conv3/depthwise\_filter","shape":[3,3,64,1],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.004478132958505668,"min":-0.6493292789833219}},{"name":"dense1/conv3/pointwise\_filter","shape":[1,1,64,64],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.011063695888893277,"min":-1.2501976354449402}},{"name":"dense1/conv3/bias","shape":[64],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.003909627596537272,"min":-0.6646366914113363}},{"name":"dense2/conv0/depthwise\_filter","shape":[3,3,64,1],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.003213915404151468,"min":-0.3374611174359041}},{"name":"dense2/conv0/pointwise\_filter","shape":[1,1,64,128],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.010917326048308728,"min":-1.4520043644250609}},{"name":"dense2/conv0/bias","shape":[128],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.002800439152063108,"min":-0.38085972468058266}},{"name":"dense2/conv1/depthwise\_filter","shape":[3,3,128,1],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.0050568851770139206,"min":-0.6927932692509071}},{"name":"dense2/conv1/pointwise\_filter","shape":[1,1,128,128],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.01074961213504567,"min":-1.3222022926106174}},{"name":"dense2/conv1/bias","shape":[128],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.0030654204242369708,"min":-0.5487102559384177}},{"name":"dense2/conv2/depthwise\_filter","shape":[3,3,128,1],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.00591809165244009,"min":-0.917304206128214}},{"name":"dense2/conv2/pointwise\_filter","shape":[1,1,128,128],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.01092823346455892,"min":-1.366029183069865}},{"name":"dense2/conv2/bias","shape":[128],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.002681120470458386,"min":-0.36463238398234055}},{"name":"dense2/conv3/depthwise\_filter","shape":[3,3,128,1],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.0048311497650894465,"min":-0.5797379718107336}},{"name":"dense2/conv3/pointwise\_filter","shape":[1,1,128,128],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.011227761062921263,"min":-1.4483811771168429}},{"name":"dense2/conv3/bias","shape":[128],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.0034643323982463162,"min":-0.3360402426298927}},{"name":"dense3/conv0/depthwise\_filter","shape":[3,3,128,1],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.003394978887894574,"min":-0.49227193874471326}},{"name":"dense3/conv0/pointwise\_filter","shape":[1,1,128,256],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.010051267287310432,"min":-1.2765109454884247}},{"name":"dense3/conv0/bias","shape":[256],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.003142924752889895,"min":-0.4588670139219247}},{"name":"dense3/conv1/depthwise\_filter","shape":[3,3,256,1],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.00448304671867221,"min":-0.5872791201460595}},{"name":"dense3/conv1/pointwise\_filter","shape":[1,1,256,256],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.016063522357566685,"min":-2.3613377865623026}},{"name":"dense3/conv1/bias","shape":[256],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.00287135781026354,"min":-0.47664539650374765}},{"name":"dense3/conv2/depthwise\_filter","shape":[3,3,256,1],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.006002906724518421,"min":-0.7923836876364315}},{"name":"dense3/conv2/pointwise\_filter","shape":[1,1,256,256],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.017087187019048954,"min":-1.6061955797906016}},{"name":"dense3/conv2/bias","shape":[256],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.003124481205846749,"min":-0.46242321846531886}},{"name":"dense3/conv3/depthwise\_filter","shape":[3,3,256,1],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.006576311588287353,"min":-1.0193282961845398}},{"name":"dense3/conv3/pointwise\_filter","shape":[1,1,256,256],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.015590153955945782,"min":-1.99553970636106}},{"name":"dense3/conv3/bias","shape":[256],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.004453541601405424,"min":-0.6546706154065973}},{"name":"fc/weights","shape":[256,136],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.010417488509533453,"min":-1.500118345372817}},{"name":"fc/bias","shape":[136],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.0025084222648658005,"min":0.07683877646923065}}],"paths":["face\_landmark\_68\_model-shard1"]}]

[{"weights":[{"name":"conv32\_down/conv/filters","shape":[7,7,3,32],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.0005260649557207145,"min":-0.07101876902229645}},{"name":"conv32\_down/conv/bias","shape":[32],"dtype":"float32","quantization":{"dtype":"uint8","scale":8.471445956577858e-7,"min":-0.00014740315964445472}},{"name":"conv32\_down/scale/weights","shape":[32],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.06814416062598135,"min":5.788674831390381}},{"name":"conv32\_down/scale/biases","shape":[32],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.008471635042452345,"min":-0.931879854669758}},{"name":"conv32\_1/conv1/conv/filters","shape":[3,3,32,32],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.0007328585666768691,"min":-0.0974701893680236}},{"name":"conv32\_1/conv1/conv/bias","shape":[32],"dtype":"float32","quantization":{"dtype":"uint8","scale":1.5952091238361e-8,"min":-0.000001978059313556764}},{"name":"conv32\_1/conv1/scale/weights","shape":[32],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.02146628510718252,"min":3.1103382110595703}},{"name":"conv32\_1/conv1/scale/biases","shape":[32],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.0194976619645661,"min":-2.3787147596770644}},{"name":"conv32\_1/conv2/conv/filters","shape":[3,3,32,32],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.0004114975824075587,"min":-0.05267169054816751}},{"name":"conv32\_1/conv2/conv/bias","shape":[32],"dtype":"float32","quantization":{"dtype":"uint8","scale":4.600177166424806e-9,"min":-5.70421968636676e-7}},{"name":"conv32\_1/conv2/scale/weights","shape":[32],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.03400764932819441,"min":2.1677730083465576}},{"name":"conv32\_1/conv2/scale/biases","shape":[32],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.010974494616190593,"min":-1.240117891629537}},{"name":"conv32\_2/conv1/conv/filters","shape":[3,3,32,32],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.0005358753251094444,"min":-0.0760942961655411}},{"name":"conv32\_2/conv1/conv/bias","shape":[32],"dtype":"float32","quantization":{"dtype":"uint8","scale":5.9886454383719385e-9,"min":-7.366033889197485e-7}},{"name":"conv32\_2/conv1/scale/weights","shape":[32],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.014633869657329485,"min":2.769575357437134}},{"name":"conv32\_2/conv1/scale/biases","shape":[32],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.022131107367721257,"min":-2.5229462399202234}},{"name":"conv32\_2/conv2/conv/filters","shape":[3,3,32,32],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.00030145110452876373,"min":-0.03949009469326805}},{"name":"conv32\_2/conv2/conv/bias","shape":[32],"dtype":"float32","quantization":{"dtype":"uint8","scale":6.8779549306497095e-9,"min":-9.010120959151119e-7}},{"name":"conv32\_2/conv2/scale/weights","shape":[32],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.03929369870354148,"min":4.8010945320129395}},{"name":"conv32\_2/conv2/scale/biases","shape":[32],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.010553357180427103,"min":-1.2452961472903983}},{"name":"conv32\_3/conv1/conv/filters","shape":[3,3,32,32],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.0003133527642371608,"min":-0.040735859350830905}},{"name":"conv32\_3/conv1/conv/bias","shape":[32],"dtype":"float32","quantization":{"dtype":"uint8","scale":4.1064200719547974e-9,"min":-3.0387508532465503e-7}},{"name":"conv32\_3/conv1/scale/weights","shape":[32],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.009252088210161994,"min":2.333256721496582}},{"name":"conv32\_3/conv1/scale/biases","shape":[32],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.007104101251153385,"min":-0.34810096130651585}},{"name":"conv32\_3/conv2/conv/filters","shape":[3,3,32,32],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.00029995629892629733,"min":-0.031195455088334923}},{"name":"conv32\_3/conv2/conv/bias","shape":[32],"dtype":"float32","quantization":{"dtype":"uint8","scale":5.62726418316814e-9,"min":-6.921534945296811e-7}},{"name":"conv32\_3/conv2/scale/weights","shape":[32],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.0467432975769043,"min":5.362040996551514}},{"name":"conv32\_3/conv2/scale/biases","shape":[32],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.010314425300149357,"min":-1.268674311918371}},{"name":"conv64\_down/conv1/conv/filters","shape":[3,3,32,64],"dtype":"float32"},{"name":"conv64\_down/conv1/conv/bias","shape":[64],"dtype":"float32","quantization":{"dtype":"uint8","scale":8.373908033218849e-10,"min":-1.172347124650639e-7}},{"name":"conv64\_down/conv1/scale/weights","shape":[64],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.0066875364266189875,"min":2.5088400840759277}},{"name":"conv64\_down/conv1/scale/biases","shape":[64],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.01691421620986041,"min":-2.0973628100226906}},{"name":"conv64\_down/conv2/conv/filters","shape":[3,3,64,64],"dtype":"float32"},{"name":"conv64\_down/conv2/conv/bias","shape":[64],"dtype":"float32","quantization":{"dtype":"uint8","scale":2.3252014483766877e-9,"min":-2.673981665633191e-7}},{"name":"conv64\_down/conv2/scale/weights","shape":[64],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.032557439804077146,"min":2.6351239681243896}},{"name":"conv64\_down/conv2/scale/biases","shape":[64],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.015429047509735706,"min":-1.5429047509735707}},{"name":"conv64\_1/conv1/conv/filters","shape":[3,3,64,64],"dtype":"float32"},{"name":"conv64\_1/conv1/conv/bias","shape":[64],"dtype":"float32","quantization":{"dtype":"uint8","scale":1.1319172039756998e-9,"min":-1.4941307092479238e-7}},{"name":"conv64\_1/conv1/scale/weights","shape":[64],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.007802607031429515,"min":3.401733160018921}},{"name":"conv64\_1/conv1/scale/biases","shape":[64],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.01425027146058924,"min":-0.6982633015688727}},{"name":"conv64\_1/conv2/conv/filters","shape":[3,3,64,64],"dtype":"float32"},{"name":"conv64\_1/conv2/conv/bias","shape":[64],"dtype":"float32","quantization":{"dtype":"uint8","scale":2.5635019893325435e-9,"min":-2.717312108692496e-7}},{"name":"conv64\_1/conv2/scale/weights","shape":[64],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.04062801716374416,"min":3.542381525039673}},{"name":"conv64\_1/conv2/scale/biases","shape":[64],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.007973166306813557,"min":-0.7415044665336609}},{"name":"conv64\_2/conv1/conv/filters","shape":[3,3,64,64],"dtype":"float32"},{"name":"conv64\_2/conv1/conv/bias","shape":[64],"dtype":"float32","quantization":{"dtype":"uint8","scale":1.2535732661062331e-9,"min":-1.8302169685151004e-7}},{"name":"conv64\_2/conv1/scale/weights","shape":[64],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.005631206549850164,"min":2.9051668643951416}},{"name":"conv64\_2/conv1/scale/biases","shape":[64],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.01859012585060269,"min":-2.3795361088771445}},{"name":"conv64\_2/conv2/conv/filters","shape":[3,3,64,64],"dtype":"float32"},{"name":"conv64\_2/conv2/conv/bias","shape":[64],"dtype":"float32","quantization":{"dtype":"uint8","scale":2.486726369919351e-9,"min":-3.5311514452854786e-7}},{"name":"conv64\_2/conv2/scale/weights","shape":[64],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.03740917467603497,"min":5.571568965911865}},{"name":"conv64\_2/conv2/scale/biases","shape":[64],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.006418555858088475,"min":-0.5263215803632549}},{"name":"conv64\_3/conv1/conv/filters","shape":[3,3,64,64],"dtype":"float32"},{"name":"conv64\_3/conv1/conv/bias","shape":[64],"dtype":"float32","quantization":{"dtype":"uint8","scale":7.432564576875473e-10,"min":-8.47312361763804e-8}},{"name":"conv64\_3/conv1/scale/weights","shape":[64],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.006400122362024644,"min":2.268010377883911}},{"name":"conv64\_3/conv1/scale/biases","shape":[64],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.010945847922680425,"min":-1.3353934465670119}},{"name":"conv64\_3/conv2/conv/filters","shape":[3,3,64,64],"dtype":"float32"},{"name":"conv64\_3/conv2/conv/bias","shape":[64],"dtype":"float32","quantization":{"dtype":"uint8","scale":2.278228722014533e-9,"min":-3.212302498040492e-7}},{"name":"conv64\_3/conv2/scale/weights","shape":[64],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.029840927498013366,"min":7.038398265838623}},{"name":"conv64\_3/conv2/scale/biases","shape":[64],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.010651412197187834,"min":-1.161003929493474}},{"name":"conv128\_down/conv1/conv/filters","shape":[3,3,64,128],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.00020040544662989823,"min":-0.022245004575918704}},{"name":"conv128\_down/conv1/conv/bias","shape":[128],"dtype":"float32","quantization":{"dtype":"uint8","scale":4.3550543563576545e-10,"min":-4.311503812794078e-8}},{"name":"conv128\_down/conv1/scale/weights","shape":[128],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.007448580685783835,"min":2.830846071243286}},{"name":"conv128\_down/conv1/scale/biases","shape":[128],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.01211262824488621,"min":-1.6957679542840696}},{"name":"conv128\_down/conv2/conv/filters","shape":[3,3,128,128],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.00022380277514457702,"min":-0.02484210804104805}},{"name":"conv128\_down/conv2/conv/bias","shape":[128],"dtype":"float32","quantization":{"dtype":"uint8","scale":9.031058637304466e-10,"min":-1.1650065642122761e-7}},{"name":"conv128\_down/conv2/scale/weights","shape":[128],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.027663578706629135,"min":3.1111555099487305}},{"name":"conv128\_down/conv2/scale/biases","shape":[128],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.008878476946961646,"min":-1.029903325847551}},{"name":"conv128\_1/conv1/conv/filters","shape":[3,3,128,128],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.00022380667574265425,"min":-0.032899581334170175}},{"name":"conv128\_1/conv1/conv/bias","shape":[128],"dtype":"float32","quantization":{"dtype":"uint8","scale":4.4147297756478345e-10,"min":-5.253528433020923e-8}},{"name":"conv128\_1/conv1/scale/weights","shape":[128],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.013599334978589825,"min":3.634530782699585}},{"name":"conv128\_1/conv1/scale/biases","shape":[128],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.014059314073300829,"min":-1.4059314073300828}},{"name":"conv128\_1/conv2/conv/filters","shape":[3,3,128,128],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.00021715293474057143,"min":-0.02909849325523657}},{"name":"conv128\_1/conv2/conv/bias","shape":[128],"dtype":"float32","quantization":{"dtype":"uint8","scale":9.887046963276768e-10,"min":-1.1370104007768284e-7}},{"name":"conv128\_1/conv2/scale/weights","shape":[128],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.029993299409454943,"min":3.630716562271118}},{"name":"conv128\_1/conv2/scale/biases","shape":[128],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.00782704236460667,"min":-0.7200878975438136}},{"name":"conv128\_2/conv1/conv/filters","shape":[3,3,128,128],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.00017718105923895743,"min":-0.022324813464108636}},{"name":"conv128\_2/conv1/conv/bias","shape":[128],"dtype":"float32","quantization":{"dtype":"uint8","scale":3.567012027797675e-10,"min":-5.243507680862582e-8}},{"name":"conv128\_2/conv1/scale/weights","shape":[128],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.007940645778880399,"min":4.927767753601074}},{"name":"conv128\_2/conv1/scale/biases","shape":[128],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.015933452867994122,"min":-1.5614783810634238}},{"name":"conv128\_2/conv2/conv/filters","shape":[3,3,128,128],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.0001451439717236687,"min":-0.01712698866339291}},{"name":"conv128\_2/conv2/conv/bias","shape":[128],"dtype":"float32","quantization":{"dtype":"uint8","scale":1.0383988570966347e-9,"min":-1.2356946399449953e-7}},{"name":"conv128\_2/conv2/scale/weights","shape":[128],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.02892604528688917,"min":4.750600814819336}},{"name":"conv128\_2/conv2/scale/biases","shape":[128],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.00797275748907351,"min":-0.7414664464838364}},{"name":"conv256\_down/conv1/conv/filters","shape":[3,3,128,256],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.0002698827827093648,"min":-0.03994265184098599}},{"name":"conv256\_down/conv1/conv/bias","shape":[256],"dtype":"float32","quantization":{"dtype":"uint8","scale":5.036909834755123e-10,"min":-6.396875490139006e-8}},{"name":"conv256\_down/conv1/scale/weights","shape":[256],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.014870181738161573,"min":4.269900798797607}},{"name":"conv256\_down/conv1/scale/biases","shape":[256],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.022031106200872685,"min":-3.1063859743230484}},{"name":"conv256\_down/conv2/conv/filters","shape":[3,3,256,256],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.00046430734150549946,"min":-0.03946612402796745}},{"name":"conv256\_down/conv2/conv/bias","shape":[256],"dtype":"float32","quantization":{"dtype":"uint8","scale":6.693064577513153e-10,"min":-7.630093618364995e-8}},{"name":"conv256\_down/conv2/scale/weights","shape":[256],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.03475512242784687,"min":3.608360528945923}},{"name":"conv256\_down/conv2/scale/biases","shape":[256],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.01290142021927179,"min":-1.1482263995151893}},{"name":"conv256\_1/conv1/conv/filters","shape":[3,3,256,256],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.00037147209924810076,"min":-0.04234781931428348}},{"name":"conv256\_1/conv1/conv/bias","shape":[256],"dtype":"float32","quantization":{"dtype":"uint8","scale":3.2105515457510146e-10,"min":-3.467395669411096e-8}},{"name":"conv256\_1/conv1/scale/weights","shape":[256],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.043242172166412955,"min":5.28542947769165}},{"name":"conv256\_1/conv1/scale/biases","shape":[256],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.01643658619300992,"min":-1.3149268954407936}},{"name":"conv256\_1/conv2/conv/filters","shape":[3,3,256,256],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.0003289232651392619,"min":-0.041773254672686264}},{"name":"conv256\_1/conv2/conv/bias","shape":[256],"dtype":"float32","quantization":{"dtype":"uint8","scale":9.13591691187321e-10,"min":-1.2333487831028833e-7}},{"name":"conv256\_1/conv2/scale/weights","shape":[256],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.0573908618852204,"min":4.360693454742432}},{"name":"conv256\_1/conv2/scale/biases","shape":[256],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.0164216583850337,"min":-1.3958409627278647}},{"name":"conv256\_2/conv1/conv/filters","shape":[3,3,256,256],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.00010476927912118389,"min":-0.015610622589056398}},{"name":"conv256\_2/conv1/conv/bias","shape":[256],"dtype":"float32","quantization":{"dtype":"uint8","scale":2.418552539068639e-10,"min":-2.539480166022071e-8}},{"name":"conv256\_2/conv1/scale/weights","shape":[256],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.06024209564807368,"min":6.598613739013672}},{"name":"conv256\_2/conv1/scale/biases","shape":[256],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.01578534350675695,"min":-1.1049740454729864}},{"name":"conv256\_2/conv2/conv/filters","shape":[3,3,256,256],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.00005543030908002573,"min":-0.007427661416723448}},{"name":"conv256\_2/conv2/conv/bias","shape":[256],"dtype":"float32","quantization":{"dtype":"uint8","scale":1.0822061852320308e-9,"min":-1.515088659324843e-7}},{"name":"conv256\_2/conv2/scale/weights","shape":[256],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.04302893993901272,"min":2.2855491638183594}},{"name":"conv256\_2/conv2/scale/biases","shape":[256],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.006792667566561232,"min":-0.8083274404207865}},{"name":"conv256\_down\_out/conv1/conv/filters","shape":[3,3,256,256],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.000568966465253456,"min":-0.05632768006009214}},{"name":"conv256\_down\_out/conv1/conv/bias","shape":[256],"dtype":"float32","quantization":{"dtype":"uint8","scale":4.5347887884881677e-10,"min":-6.530095855422961e-8}},{"name":"conv256\_down\_out/conv1/scale/weights","shape":[256],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.017565592597512638,"min":4.594101905822754}},{"name":"conv256\_down\_out/conv1/scale/biases","shape":[256],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.04850864223405427,"min":-6.306123490427055}},{"name":"conv256\_down\_out/conv2/conv/filters","shape":[3,3,256,256],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.0003739110687199761,"min":-0.06954745878191555}},{"name":"conv256\_down\_out/conv2/conv/bias","shape":[256],"dtype":"float32","quantization":{"dtype":"uint8","scale":1.2668428328152895e-9,"min":-2.2549802424112154e-7}},{"name":"conv256\_down\_out/conv2/scale/weights","shape":[256],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.04351314469879749,"min":4.31956672668457}},{"name":"conv256\_down\_out/conv2/scale/biases","shape":[256],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.021499746921015722,"min":-1.2039858275768804}},{"name":"fc","shape":[256,128],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.000357687911566566,"min":-0.04578405268052045}}],"paths":["face\_recognition\_model-shard1","face\_recognition\_model-shard2"]}]

[{"weights":[{"name":"conv0/filters","shape":[3,3,3,16],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.009007044399485869,"min":-1.2069439495311063}},{"name":"conv0/bias","shape":[16],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.005263455241334205,"min":-0.9211046672334858}},{"name":"conv1/depthwise\_filter","shape":[3,3,16,1],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.004001977630690033,"min":-0.5042491814669441}},{"name":"conv1/pointwise\_filter","shape":[1,1,16,32],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.013836609615999109,"min":-1.411334180831909}},{"name":"conv1/bias","shape":[32],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.0015159862590771096,"min":-0.30926119685173037}},{"name":"conv2/depthwise\_filter","shape":[3,3,32,1],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.002666276225856706,"min":-0.317286870876948}},{"name":"conv2/pointwise\_filter","shape":[1,1,32,64],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.015265831292844286,"min":-1.6792414422128714}},{"name":"conv2/bias","shape":[64],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.0020280554598453,"min":-0.37113414915168985}},{"name":"conv3/depthwise\_filter","shape":[3,3,64,1],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.006100742489683862,"min":-0.8907084034938438}},{"name":"conv3/pointwise\_filter","shape":[1,1,64,128],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.016276211832083907,"min":-2.0508026908425725}},{"name":"conv3/bias","shape":[128],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.003394414279975143,"min":-0.7637432129944072}},{"name":"conv4/depthwise\_filter","shape":[3,3,128,1],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.006716050119961009,"min":-0.8059260143953211}},{"name":"conv4/pointwise\_filter","shape":[1,1,128,256],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.021875603993733724,"min":-2.8875797271728514}},{"name":"conv4/bias","shape":[256],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.0041141652009066415,"min":-0.8187188749804216}},{"name":"conv5/depthwise\_filter","shape":[3,3,256,1],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.008423839597141042,"min":-0.9013508368940915}},{"name":"conv5/pointwise\_filter","shape":[1,1,256,512],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.030007277283014035,"min":-3.8709387695088107}},{"name":"conv5/bias","shape":[512],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.008402082966823203,"min":-1.4871686851277068}},{"name":"conv8/filters","shape":[1,1,512,25],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.028336129469030042,"min":-4.675461362389957}},{"name":"conv8/bias","shape":[25],"dtype":"float32","quantization":{"dtype":"uint8","scale":0.002268134028303857,"min":-0.41053225912299807}}],"paths":["tiny\_face\_detector\_model-shard1"]}]

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Qroll - Smart Attendance Tracking</title>

<style>

:root {

--background: #0f172a;

--foreground: #f8fafc;

--primary: #8b5cf6;

--primary-foreground: #ffffff;

--secondary: #1e293b;

--muted: #334155;

--muted-foreground: #94a3b8;

}

body {

font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, Oxygen, Ubuntu, Cantarell, 'Open Sans', 'Helvetica Neue', sans-serif;

margin: 0;

padding: 0;

background-color: var(--background);

color: var(--foreground);

line-height: 1.6;

}

.container {

max-width: 1200px;

margin: 0 auto;

padding: 0 20px;

}

header {

padding: 30px 0;

text-align: center;

}

.logo {

font-size: 2.5rem;

font-weight: bold;

color: var(--primary);

margin-bottom: 0;

}

.tagline {

font-size: 1.2rem;

color: var(--muted-foreground);

margin-top: 10px;

}

.hero {

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

padding: 60px 0;

text-align: center;

}

h1 {

font-size: 3rem;

margin-bottom: 20px;

max-width: 800px;

}

.hero p {

font-size: 1.25rem;

max-width: 700px;

margin-bottom: 40px;

color: var(--muted-foreground);

}

.cta-button {

display: inline-block;

background-color: var(--primary);

color: var(--primary-foreground);

font-size: 1.1rem;

font-weight: 600;

text-decoration: none;

padding: 14px 32px;

border-radius: 8px;

box-shadow: 0 4px 6px rgba(139, 92, 246, 0.25);

transition: all 0.3s ease;

}

.cta-button:hover {

transform: translateY(-2px);

box-shadow: 0 7px 10px rgba(139, 92, 246, 0.3);

}

.features {

padding: 80px 0;

background-color: var(--secondary);

}

.features h2 {

text-align: center;

font-size: 2rem;

margin-bottom: 60px;

}

.feature-grid {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));

gap: 40px;

}

.feature-card {

background-color: var(--background);

border-radius: 12px;

padding: 30px;

text-align: center;

box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);

}

.feature-icon {

width: 60px;

height: 60px;

margin: 0 auto 20px;

display: flex;

align-items: center;

justify-content: center;

background-color: rgba(139, 92, 246, 0.1);

border-radius: 50%;

color: var(--primary);

}

.feature-card h3 {

font-size: 1.5rem;

margin-bottom: 15px;

}

.feature-card p {

color: var(--muted-foreground);

}

footer {

padding: 40px 0;

text-align: center;

color: var(--muted-foreground);

}

@media (max-width: 768px) {

h1 {

font-size: 2.2rem;

}

.hero p {

font-size: 1.1rem;

}

.feature-grid {

grid-template-columns: 1fr;

}

}

</style>

</head>

<body>

<div class="container">

<header>

<h2 class="logo">Qroll</h2>

<p class="tagline">Smart Attendance Tracking</p>

</header>

<section class="hero">

<h1>The Smart Way to Track Student Attendance</h1>

<p>

Qroll combines QR code scanning, facial recognition, and location verification

to create a secure and efficient attendance management system for educational institutions.

</p>

<a href="/auth" class="cta-button">Get Started</a>

</section>

</div>

<section class="features">

<div class="container">

<h2>Key Features</h2>

<div class="feature-grid">

<div class="feature-card">

<div class="feature-icon">

<svg xmlns="http://www.w3.org/2000/svg" width="32" height="32" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round">

<path d="M3 7v9a2 2 0 0 0 2 2h13a2 2 0 0 0 2-2V9a2 2 0 0 0-2-2h-4.5"></path>

<path d="M3 9h18"></path>

<path d="M7 13h.01"></path>

<path d="M11 13h.01"></path>

<path d="M15 13h.01"></path>

<path d="M7 17h.01"></path>

<path d="M11 17h.01"></path>

<path d="M15 17h.01"></path>

<path d="M10 2v5"></path>

<path d="M14 2v5"></path>

</svg>

</div>

<h3>QR Code Scanning</h3>

<p>Generate and scan QR codes for quick and easy attendance marking.</p>

</div>

<div class="feature-card">

<div class="feature-icon">

<svg xmlns="http://www.w3.org/2000/svg" width="32" height="32" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round">

<path d="M22 8a5 5 0 0 1-3.36 4.7l.99 5.34A1 1 0 0 1 18.66 19H13a2 2 0 0 1-2-2v-5"></path>

<path d="M12 12a5 5 0 0 1 0-10 4.96 4.96 0 0 1 4.36 2.58"></path>

<path d="M9 10a5 5 0 0 1 0-8"></path>

<path d="M4.8 17.8l-.68.68a2 2 0 0 0 0 2.82 2 2 0 0 0 2.82 0l.68-.68"></path>

<path d="M2 22l3-3"></path>

</svg>

</div>

<h3>Facial Recognition</h3>

<p>Verify student identity through secure facial recognition technology.</p>

</div>

<div class="feature-card">

<div class="feature-icon">

<svg xmlns="http://www.w3.org/2000/svg" width="32" height="32" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round">

<path d="M20 10c0 4.4-8 12-8 12s-8-7.6-8-12a8 8 0 0 1 16 0z"></path>

<path d="M12 10a1 1 0 1 0 0-2 1 1 0 0 0 0 2z"></path>

</svg>

</div>

<h3>Location Verification</h3>

<p>Ensure students are present in the right location using geolocation verification.</p>

</div>

</div>

</div>

</section>

<footer>

<div class="container">

<p>© 2025 Qroll. All rights reserved.</p>

<p>

<a href="/qr-url" style="color: var(--primary); text-decoration: none;">

Get Qroll QR Code

</a>

</p>

</div>

</footer>

</body>

</html>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Qroll Attendance App QR Code</title>

<style>

body {

font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, Oxygen, Ubuntu, Cantarell, 'Open Sans', 'Helvetica Neue', sans-serif;

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

min-height: 100vh;

margin: 0;

background-color: #0f172a;

color: white;

text-align: center;

padding: 20px;

}

.container {

max-width: 600px;

background-color: #1e293b;

border-radius: 12px;

padding: 30px;

box-shadow: 0 10px 25px rgba(0, 0, 0, 0.2);

}

h1 {

color: #8b5cf6;

margin-bottom: 10px;

}

p {

color: #94a3b8;

margin-bottom: 25px;

}

.qr-code {

margin: 20px 0;

}

.url {

font-weight: bold;

background-color: #334155;

padding: 10px;

border-radius: 6px;

word-break: break-all;

}

</style>

</head>

<body>

<div class="container">

<h1>Qroll Attendance App</h1>

<p>Scan this QR code with your mobile device to access the Qroll attendance tracking application</p>

<div class="qr-code">

<!-- QR Code will be inserted here by JavaScript -->

</div>

<p>Or visit this URL:</p>

<div class="url" id="app-url"></div>

</div>

<script src="https://cdn.jsdelivr.net/npm/qrcode@1.5.0/build/qrcode.min.js"></script>

<script>

document.addEventListener('DOMContentLoaded', function() {

const appUrl = window.location.origin;

document.getElementById('app-url').textContent = appUrl;

// Generate QR code

QRCode.toCanvas(document.querySelector('.qr-code'), appUrl, {

width: 256,

margin: 2,

color: {

dark: '#8b5cf6', // Purple color

light: '#ffffff' // White background

}

}, function(error) {

if (error) console.error(error);

console.log('QR code generated successfully for', appUrl);

});

});

</script>

</body>

</html>

import { useState, useEffect } from "react";

import { useForm } from "react-hook-form";

import { zodResolver } from "@hookform/resolvers/zod";

import { qrGenerateSchema } from "@shared/schema";

import { Button } from "@/components/ui/button";

import { Input } from "@/components/ui/input";

import { Label } from "@/components/ui/label";

import { Slider } from "@/components/ui/slider";

import { Switch } from "@/components/ui/switch";

import { useMutation } from "@tanstack/react-query";

import { apiRequest, queryClient } from "@/lib/queryClient";

import { useToast } from "@/hooks/use-toast";

import { Download, Share, Timer, MapPin, Video, Loader2, AlertTriangle } from "lucide-react";

import { QRCodeSVG as QRCode } from "qrcode.react";

import { format } from "date-fns";

import { useGeolocation } from "@/hooks/use-geolocation";

import { Alert, AlertDescription } from "@/components/ui/alert";

interface QRGeneratorProps {

className?: string;

}

type QRFormValues = {

name: string;

date: string;

time: string;

location: string;

locationLat?: string;

locationLng?: string;

validDuration: number;

locationRange: number;

requireFaceVerification: boolean;

requireLocationVerification: boolean;

};

export function QRGenerator({ className }: QRGeneratorProps) {

const [generatedQR, setGeneratedQR] = useState<any | null>(null);

const [useCurrentLocation, setUseCurrentLocation] = useState(false);

const { toast } = useToast();

// Get current location

const {

latitude,

longitude,

loading: locationLoading,

error: locationError,

getCurrentPosition

} = useGeolocation({

enableHighAccuracy: true,

timeout: 10000,

maximumAge: 0

});

// Form setup

const form = useForm<QRFormValues>({

resolver: zodResolver(qrGenerateSchema),

defaultValues: {

name: "",

date: format(new Date(), "yyyy-MM-dd"),

time: format(new Date(), "HH:mm"),

location: "",

locationLat: undefined,

locationLng: undefined,

validDuration: 15,

locationRange: 100,

requireFaceVerification: true,

requireLocationVerification: true

}

});

// Update lat/lng when current location changes

useEffect(() => {

if (useCurrentLocation && latitude && longitude) {

form.setValue("locationLat", latitude.toString());

form.setValue("locationLng", longitude.toString());

}

}, [latitude, longitude, useCurrentLocation, form]);

// Handle location toggle

const handleLocationToggle = (checked: boolean) => {

setUseCurrentLocation(checked);

if (checked) {

getCurrentPosition();

} else {

form.setValue("locationLat", undefined);

form.setValue("locationLng", undefined);

}

};

// Generate QR code mutation

const generateQrMutation = useMutation({

mutationFn: async (data: QRFormValues) => {

const res = await apiRequest("POST", "/api/sessions", data);

return await res.json();

},

onSuccess: (data) => {

setGeneratedQR(data);

queryClient.invalidateQueries({ queryKey: ["/api/sessions"] });

toast({

title: "QR Code Generated",

description: "The QR code has been successfully generated.",

});

},

onError: (error: Error) => {

toast({

title: "Failed to generate QR code",

description: error.message,

variant: "destructive",

});

}

});

// Handle form submission

const onSubmit = (data: QRFormValues) => {

generateQrMutation.mutate(data);

};

// Handle download QR code

const handleDownloadQR = () => {

const svg = document.getElementById('qr-code-canvas');

if (svg) {

// Convert SVG to a data URL

const svgData = new XMLSerializer().serializeToString(svg);

const svgBlob = new Blob([svgData], { type: 'image/svg+xml;charset=utf-8' });

const svgUrl = URL.createObjectURL(svgBlob);

const link = document.createElement('a');

link.href = svgUrl;

link.download = `qroll-${generatedQR.name}-${new Date().toISOString().split('T')[0]}.svg`;

document.body.appendChild(link);

link.click();

document.body.removeChild(link);

URL.revokeObjectURL(svgUrl);

}

};

// Handle share QR code

const handleShareQR = () => {

if (navigator.share) {

navigator.share({

title: `Attendance QR Code for ${generatedQR.name}`,

text: 'Scan this QR code to mark your attendance',

// In a real app, we'd generate a shareable URL here

url: window.location.href

})

.catch((error) => {

toast({

title: "Share failed",

description: error.message,

variant: "destructive",

});

});

} else {

toast({

title: "Share not supported",

description: "Your browser doesn't support the Web Share API.",

variant: "destructive",

});

}

};

return (

<div className={className}>

<h2 className="text-xl font-semibold text-white mb-6">Generate Attendance QR Code</h2>

<form onSubmit={form.handleSubmit(onSubmit)} className="space-y-6">

<div>

<Label htmlFor="name" className="text-white">Class/Course Name</Label>

<Input

id="name"

placeholder="Data Structures & Algorithms"

{...form.register("name")}

className="bg-gray-800 border-gray-700 text-white"

/>

{form.formState.errors.name && (

<p className="text-sm text-red-400 mt-1">

{form.formState.errors.name.message}

</p>

)}

</div>

<div className="grid grid-cols-1 md:grid-cols-2 gap-4">

<div>

<Label htmlFor="date" className="text-white">Date</Label>

<Input

id="date"

type="date"

{...form.register("date")}

className="bg-gray-800 border-gray-700 text-white"

/>

{form.formState.errors.date && (

<p className="text-sm text-red-400 mt-1">

{form.formState.errors.date.message}

</p>

)}

</div>

<div>

<Label htmlFor="time" className="text-white">Time</Label>

<Input

id="time"

type="time"

{...form.register("time")}

className="bg-gray-800 border-gray-700 text-white"

/>

{form.formState.errors.time && (

<p className="text-sm text-red-400 mt-1">

{form.formState.errors.time.message}

</p>

)}

</div>

</div>

<div className="space-y-2">

<Label htmlFor="location" className="text-white">Location</Label>

<Input

id="location"

placeholder="Building A, Room 201"

{...form.register("location")}

className="bg-gray-800 border-gray-700 text-white"

/>

{form.formState.errors.location && (

<p className="text-sm text-red-400 mt-1">

{form.formState.errors.location.message}

</p>

)}

{/\* Location coordinate capture \*/}

<div className="flex items-center space-x-4 mt-2">

<div className="flex items-center space-x-2">

<Switch

id="useCurrentLocation"

checked={useCurrentLocation}

onCheckedChange={handleLocationToggle}

/>

<Label htmlFor="useCurrentLocation" className="text-white cursor-pointer">

Use current location coordinates

</Label>

</div>

{locationLoading && (

<div className="text-gray-400 text-sm flex items-center">

<Loader2 className="h-3 w-3 mr-1 animate-spin" />

Getting location...

</div>

)}

</div>

{locationError && (

<Alert variant="destructive" className="mt-2 bg-red-900/30 border-red-800 text-sm py-3">

<AlertTriangle className="h-4 w-4 mr-2" />

<AlertDescription>

{locationError || "Failed to get your current location. Please grant location permissions."}

</AlertDescription>

</Alert>

)}

{useCurrentLocation && latitude && longitude && (

<div className="grid grid-cols-2 gap-3 mt-2">

<div className="space-y-1">

<Label htmlFor="locationLat" className="text-xs text-gray-400">Latitude</Label>

<Input

id="locationLat"

value={latitude?.toString() || ""}

readOnly

className="bg-gray-900 border-gray-700 text-white text-xs font-mono h-8"

/>

</div>

<div className="space-y-1">

<Label htmlFor="locationLng" className="text-xs text-gray-400">Longitude</Label>

<Input

id="locationLng"

value={longitude?.toString() || ""}

readOnly

className="bg-gray-900 border-gray-700 text-white text-xs font-mono h-8"

/>

</div>

</div>

)}

</div>

<div className="border-t border-gray-700 pt-6">

<h3 className="text-lg font-medium text-white mb-4">QR Code Settings</h3>

<div className="space-y-6">

<div>

<Label htmlFor="validDuration" className="flex items-center text-white">

<Timer className="h-4 w-4 mr-2 text-primary" />

QR Code Active Duration: {form.watch("validDuration")} minutes

</Label>

<Slider

id="validDuration"

min={1}

max={60}

step={1}

value={[form.watch("validDuration")]}

onValueChange={(value) => form.setValue("validDuration", value[0])}

className="mt-2"

/>

{form.formState.errors.validDuration && (

<p className="text-sm text-red-400 mt-1">

{form.formState.errors.validDuration.message}

</p>

)}

</div>

<div>

<Label htmlFor="locationRange" className="flex items-center text-white">

<MapPin className="h-4 w-4 mr-2 text-primary" />

Valid Location Range: {form.watch("locationRange")} meters

</Label>

<Slider

id="locationRange"

min={10}

max={1000}

step={10}

value={[form.watch("locationRange")]}

onValueChange={(value) => form.setValue("locationRange", value[0])}

className="mt-2"

/>

{form.formState.errors.locationRange && (

<p className="text-sm text-red-400 mt-1">

{form.formState.errors.locationRange.message}

</p>

)}

</div>

{/\* Verification requirements \*/}

<div className="space-y-4 pt-2 border-t border-gray-800">

<h4 className="text-sm font-medium text-gray-300">Verification Requirements</h4>

<div className="flex items-center justify-between">

<div className="flex items-center">

<Video className="h-4 w-4 mr-2 text-primary" />

<Label htmlFor="requireFaceVerification" className="text-white cursor-pointer">

Require face verification

</Label>

</div>

<Switch

id="requireFaceVerification"

checked={form.watch("requireFaceVerification")}

onCheckedChange={(checked) => form.setValue("requireFaceVerification", checked)}

/>

</div>

<div className="flex items-center justify-between">

<div className="flex items-center">

<MapPin className="h-4 w-4 mr-2 text-primary" />

<Label htmlFor="requireLocationVerification" className="text-white cursor-pointer">

Require location verification

</Label>

</div>

<Switch

id="requireLocationVerification"

checked={form.watch("requireLocationVerification")}

onCheckedChange={(checked) => form.setValue("requireLocationVerification", checked)}

/>

</div>

{!form.watch("requireLocationVerification") && useCurrentLocation && (

<Alert className="mt-2 bg-amber-900/30 border-amber-800 text-sm py-3">

<AlertTriangle className="h-4 w-4 mr-2 text-amber-500" />

<AlertDescription className="text-amber-300">

When location verification is disabled, students don't need to be physically present at the class location.

</AlertDescription>

</Alert>

)}

</div>

</div>

</div>

<div className="pt-4">

<Button

type="submit"

className="w-full bg-primary hover:bg-primary/80"

disabled={generateQrMutation.isPending}

>

{generateQrMutation.isPending ? (

<>

<Loader2 className="mr-2 h-4 w-4 animate-spin" />

Generating QR Code...

</>

) : (

"Generate QR Code"

)}

</Button>

</div>

</form>

{/\* Generated QR Code \*/}

{generatedQR && (

<div className="mt-8 border-t border-gray-700 pt-6">

<div className="text-center">

<h3 className="text-lg font-medium text-white mb-4">QR Code Generated</h3>

<div className="bg-gray-800 border border-gray-700 rounded-lg p-4 inline-block mb-4">

<QRCode

id="qr-code-canvas"

value={JSON.stringify({

sessionId: generatedQR.id,

qrCode: generatedQR.qrCode,

name: generatedQR.name,

time: `${generatedQR.time}`,

location: generatedQR.location

})}

size={192}

className="w-48 h-48"

bgColor="#1f2937"

fgColor="#ffffff"

/>

</div>

<div className="max-w-md mx-auto">

<div className="bg-gray-800 border border-gray-700 rounded-lg p-4 mb-4">

<div className="flex items-center mb-2">

<Timer className="text-primary mr-2 h-5 w-5" />

<span className="font-medium text-white">

Active for: {generatedQR.validDuration} minutes

</span>

</div>

<div className="w-full bg-gray-700 rounded-full h-2 mb-4">

<div className="bg-primary h-2 rounded-full w-full"></div>

</div>

<div className="grid grid-cols-2 gap-3 text-center text-sm">

<div className={`py-2 px-3 rounded ${generatedQR.requireLocationVerification

? 'bg-green-900/30 text-green-400 border border-green-900'

: 'bg-gray-900/50 text-gray-500 border border-gray-800'}`}>

<MapPin className={`h-4 w-4 mx-auto mb-1 ${generatedQR.requireLocationVerification

? 'text-green-400'

: 'text-gray-500'}`} />

<span className="block">

Location Check {generatedQR.requireLocationVerification ? 'Required' : 'Disabled'}

</span>

{generatedQR.requireLocationVerification && generatedQR.locationLat && (

<span className="block text-xs text-gray-400 mt-1">

Range: {generatedQR.locationRange}m

</span>

)}

</div>

<div className={`py-2 px-3 rounded ${generatedQR.requireFaceVerification

? 'bg-green-900/30 text-green-400 border border-green-900'

: 'bg-gray-900/50 text-gray-500 border border-gray-800'}`}>

<Video className={`h-4 w-4 mx-auto mb-1 ${generatedQR.requireFaceVerification

? 'text-green-400'

: 'text-gray-500'}`} />

<span className="block">

Face Verification {generatedQR.requireFaceVerification ? 'Required' : 'Disabled'}

</span>

</div>

</div>

</div>

<div className="flex space-x-3">

<Button

className="flex-1 bg-primary hover:bg-primary/80"

onClick={handleDownloadQR}

>

<Download className="mr-2 h-4 w-4" />

Download

</Button>

<Button

variant="outline"

className="flex-1 border-gray-700 text-white hover:bg-gray-800"

onClick={handleShareQR}

>

<Share className="mr-2 h-4 w-4" />

Share

</Button>

</div>

</div>

</div>

</div>

)}

</div>

);

}

import { useState, useEffect, useRef } from "react";

import { useForm } from "react-hook-form";

import { zodResolver } from "@hookform/resolvers/zod";

import { loginSchema, registerSchema, InsertUser } from "@shared/schema";

import { useAuth } from "@/hooks/use-auth";

import { Button } from "@/components/ui/button";

import { Input } from "@/components/ui/input";

import { Label } from "@/components/ui/label";

import { Tabs, TabsContent, TabsList, TabsTrigger } from "@/components/ui/tabs";

import { Card, CardContent, CardDescription, CardFooter, CardHeader, CardTitle } from "@/components/ui/card";

import { useToast } from "@/hooks/use-toast";

import { Loader2, Camera, CheckCircle, AlertCircle } from "lucide-react";

import { useFaceRecognition } from "@/hooks/use-face-recognition";

import { Alert, AlertDescription, AlertTitle } from "@/components/ui/alert";

interface AuthFormsProps {

initialRole?: string;

}

export function AuthForms({ initialRole = "student" }: AuthFormsProps) {

const { loginMutation, registerMutation } = useAuth();

const { toast } = useToast();

// Default to login tab but pre-select the appropriate role

const [activeTab, setActiveTab] = useState<"login" | "register">("login");

const [faceData, setFaceData] = useState<string | null>(null);

const [showFaceCapture, setShowFaceCapture] = useState(false);

const videoRef = useRef<HTMLVideoElement>(null);

// Face recognition setup for student registration

const {

start: startFaceDetection,

stop: stopFaceDetection,

isRunning: isFaceDetectionRunning,

progress: faceDetectionProgress,

faceDetected,

error: faceDetectionError

} = useFaceRecognition({

onDetected: (faceDescriptor) => {

if (faceDescriptor && videoRef.current) {

// Get the current frame from video as base64 string with lower quality for faster processing

const canvas = document.createElement('canvas');

// Use smaller dimensions for better performance

const width = videoRef.current.videoWidth;

const height = videoRef.current.videoHeight;

canvas.width = width;

canvas.height = height;

const ctx = canvas.getContext('2d');

if (ctx) {

ctx.drawImage(videoRef.current, 0, 0, width, height);

// Lower quality (0.5) for faster processing and smaller size

const dataUrl = canvas.toDataURL('image/jpeg', 0.5);

setFaceData(dataUrl);

stopFaceDetection();

toast({

title: "Face captured successfully",

description: "You can now proceed with your registration",

variant: "default"

});

}

}

},

onError: (error) => {

toast({

title: "Face detection error",

description: error.message,

variant: "destructive"

});

}

});

// Login form

const loginForm = useForm({

resolver: zodResolver(loginSchema),

defaultValues: {

username: "",

password: ""

}

});

const onLoginSubmit = (data: { username: string; password: string }) => {

loginMutation.mutate(data);

};

// Register form

type RegisterFormData = Omit<InsertUser, "role"> & {

confirmPassword: string;

role: string;

};

const registerForm = useForm<RegisterFormData>({

resolver: zodResolver(registerSchema),

defaultValues: {

username: "",

password: "",

confirmPassword: "",

fullName: "",

email: "",

dob: "",

mobile: "",

role: initialRole // Set default role here

}

});

// Set role when initialRole changes

useEffect(() => {

if (initialRole === "admin" || initialRole === "student") {

registerForm.setValue("role", initialRole);

}

}, [initialRole, registerForm]);

// Cleanup face recognition when component unmounts

useEffect(() => {

return () => {

if (isFaceDetectionRunning) {

stopFaceDetection();

}

};

}, [isFaceDetectionRunning, stopFaceDetection]);

const startFaceCapture = () => {

setShowFaceCapture(true);

setTimeout(() => {

startFaceDetection(videoRef);

}, 500);

};

const cancelFaceCapture = () => {

stopFaceDetection();

setShowFaceCapture(false);

};

const retakeFace = () => {

setFaceData(null);

startFaceCapture();

};

const onRegisterSubmit = (data: RegisterFormData) => {

// Convert to InsertUser type for the API

const { confirmPassword, ...userData } = data;

// Only require face data for students

if (userData.role === 'student' && !faceData) {

toast({

title: "Face verification required",

description: "Please capture your face before registering as a student",

variant: "destructive"

});

return;

}

// Add face data to user data if available

const userDataWithFace = {

...userData,

faceData: faceData || undefined

};

registerMutation.mutate(userDataWithFace as InsertUser);

};

return (

<Card className="w-full max-w-md bg-gray-900 border-gray-800">

<CardHeader>

<CardTitle className="text-2xl font-bold text-white">

{activeTab === "login" ? "Sign In" : "Create Account"}

</CardTitle>

<CardDescription className="text-gray-400">

{activeTab === "login"

? "Enter your credentials to access your account"

: "Fill in your details to create an account"}

</CardDescription>

</CardHeader>

<Tabs value={activeTab} onValueChange={(v) => setActiveTab(v as "login" | "register")}>

<TabsList className="grid w-full grid-cols-2 bg-gray-800">

<TabsTrigger value="login" className="data-[state=active]:bg-primary data-[state=active]:text-white">Login</TabsTrigger>

<TabsTrigger value="register" className="data-[state=active]:bg-primary data-[state=active]:text-white">Register</TabsTrigger>

</TabsList>

<TabsContent value="login">

<form onSubmit={loginForm.handleSubmit(onLoginSubmit)}>

<CardContent className="space-y-4 pt-4">

<div className="space-y-2">

<Label htmlFor="username">Username</Label>

<Input

id="username"

placeholder="your.username"

autoComplete="username"

{...loginForm.register("username")}

/>

{loginForm.formState.errors.username && (

<p className="text-sm text-red-500">

{loginForm.formState.errors.username.message}

</p>

)}

</div>

<div className="space-y-2">

<div className="flex items-center justify-between">

<Label htmlFor="password">Password</Label>

<Button variant="link" className="px-0 text-xs" type="button">

Forgot password?

</Button>

</div>

<Input

id="password"

type="password"

placeholder="••••••••"

autoComplete="current-password"

{...loginForm.register("password")}

/>

{loginForm.formState.errors.password && (

<p className="text-sm text-red-500">

{loginForm.formState.errors.password.message}

</p>

)}

</div>

</CardContent>

<CardFooter className="pt-4">

<Button

type="submit"

className="w-full bg-primary hover:bg-primary/90 text-white"

disabled={loginMutation.isPending}

>

{loginMutation.isPending ? (

<>

<Loader2 className="mr-2 h-4 w-4 animate-spin" />

Signing in...

</>

) : (

"Sign In"

)}

</Button>

</CardFooter>

</form>

</TabsContent>

<TabsContent value="register">

<form onSubmit={registerForm.handleSubmit(onRegisterSubmit)}>

<CardContent className="space-y-4 pt-4">

<div className="flex space-x-4">

<Button

type="button"

variant={registerForm.watch("role") === "student" ? "default" : "outline"}

className={`flex-1 ${registerForm.watch("role") === "student"

? "bg-primary hover:bg-primary/90 text-white"

: "border-primary text-primary hover:bg-primary/10"}`}

onClick={() => registerForm.setValue("role", "student")}

>

Student

</Button>

<Button

type="button"

variant={registerForm.watch("role") === "admin" ? "default" : "outline"}

className={`flex-1 ${registerForm.watch("role") === "admin"

? "bg-primary hover:bg-primary/90 text-white"

: "border-primary text-primary hover:bg-primary/10"}`}

onClick={() => registerForm.setValue("role", "admin")}

>

Admin

</Button>

</div>

<div className="space-y-2">

<Label htmlFor="fullName">Full Name</Label>

<Input

id="fullName"

placeholder="John Doe"

{...registerForm.register("fullName")}

/>

{registerForm.formState.errors.fullName && (

<p className="text-sm text-red-500">

{registerForm.formState.errors.fullName.message}

</p>

)}

</div>

<div className="space-y-2">

<Label htmlFor="username">Username</Label>

<Input

id="username"

placeholder="john.doe"

autoComplete="username"

{...registerForm.register("username")}

/>

{registerForm.formState.errors.username && (

<p className="text-sm text-red-500">

{registerForm.formState.errors.username.message}

</p>

)}

</div>

<div className="space-y-2">

<Label htmlFor="email">Email</Label>

<Input

id="email"

type="email"

placeholder="your.email@example.com"

autoComplete="email"

{...registerForm.register("email")}

/>

{registerForm.formState.errors.email && (

<p className="text-sm text-red-500">

{registerForm.formState.errors.email.message}

</p>

)}

</div>

<div className="space-y-2">

<Label htmlFor="dob">Date of Birth</Label>

<Input

id="dob"

type="date"

{...registerForm.register("dob")}

/>

{registerForm.formState.errors.dob && (

<p className="text-sm text-red-500">

{registerForm.formState.errors.dob.message}

</p>

)}

</div>

<div className="space-y-2">

<Label htmlFor="mobile">Mobile Number</Label>

<Input

id="mobile"

placeholder="+1 (123) 456-7890"

{...registerForm.register("mobile")}

/>

{registerForm.formState.errors.mobile && (

<p className="text-sm text-red-500">

{registerForm.formState.errors.mobile.message}

</p>

)}

</div>

<div className="space-y-2">

<Label htmlFor="password">Password</Label>

<Input

id="password"

type="password"

placeholder="••••••••"

autoComplete="new-password"

{...registerForm.register("password")}

/>

{registerForm.formState.errors.password && (

<p className="text-sm text-red-500">

{registerForm.formState.errors.password.message}

</p>

)}

</div>

<div className="space-y-2">

<Label htmlFor="confirmPassword">Confirm Password</Label>

<Input

id="confirmPassword"

type="password"

placeholder="••••••••"

autoComplete="new-password"

{...registerForm.register("confirmPassword")}

/>

{registerForm.formState.errors.confirmPassword && (

<p className="text-sm text-red-500">

{registerForm.formState.errors.confirmPassword.message}

</p>

)}

</div>

{/\* Face capture section - only for students \*/}

{registerForm.watch("role") === "student" && (

<div className="space-y-3">

<div className="flex items-center justify-between">

<Label>Face Verification</Label>

{faceData ? (

<Button

type="button"

variant="outline"

size="sm"

onClick={retakeFace}

className="text-xs"

>

Retake Photo

</Button>

) : null}

</div>

{faceDetectionError && (

<Alert variant="destructive" className="py-2">

<AlertCircle className="h-4 w-4" />

<AlertTitle>Error</AlertTitle>

<AlertDescription>{faceDetectionError.message}</AlertDescription>

</Alert>

)}

{/\* Face capture UI \*/}

{showFaceCapture && !faceData ? (

<div className="relative border border-gray-700 rounded-md overflow-hidden aspect-video bg-black flex items-center justify-center">

<video

ref={videoRef}

autoPlay

playsInline

muted

className="w-full h-full object-cover"

/>

{/\* Progress indicator for face capture \*/}

{faceDetectionProgress > 0 && (

<div className="absolute top-0 left-0 right-0 h-2 bg-gray-800">

<div

className="h-full bg-green-500 transition-all duration-200"

style={{ width: `${faceDetectionProgress}%` }}

/>

</div>

)}

<div className="absolute inset-0 flex flex-col items-center justify-center">

{isFaceDetectionRunning ? (

<div className="bg-black/70 p-3 rounded-lg text-center">

<Loader2 className="h-8 w-8 animate-spin text-primary mx-auto mb-2" />

<p className="text-white text-sm">

{faceDetected ?

`Capturing... ${Math.round(faceDetectionProgress)}%` :

"Looking for your face..."

}

</p>

</div>

) : null}

</div>

<Button

type="button"

variant="outline"

size="sm"

className="absolute bottom-2 right-2 bg-gray-900"

onClick={cancelFaceCapture}

>

Cancel

</Button>

</div>

) : !faceData ? (

<Button

type="button"

variant="outline"

className="w-full border-dashed border-2 py-8 flex flex-col items-center gap-2"

onClick={startFaceCapture}

>

<Camera className="h-8 w-8 text-gray-400" />

<span className="text-sm text-gray-400">Capture your face for verification</span>

</Button>

) : (

<div className="relative border border-gray-700 rounded-md overflow-hidden aspect-video">

<img

src={faceData}

alt="Captured face"

className="w-full h-full object-cover"

/>

<div className="absolute top-2 right-2 bg-green-500/80 text-white p-1 rounded-full">

<CheckCircle className="h-5 w-5" />

</div>

</div>

)}

{registerForm.watch("role") === "student" && !faceData && !showFaceCapture && (

<p className="text-amber-500 text-xs">

Students must capture their face for daily attendance verification

</p>

)}

</div>

)}

</CardContent>

<CardFooter className="pt-2">

<Button

type="submit"

className="w-full bg-primary hover:bg-primary/90 text-white"

disabled={registerMutation.isPending}

>

{registerMutation.isPending ? (

<>

<Loader2 className="mr-2 h-4 w-4 animate-spin" />

Creating Account...

</>

) : (

"Create Account"

)}

</Button>

</CardFooter>

</form>

</TabsContent>

</Tabs>

</Card>

);

}

import { useState } from "react";

import { User } from "@shared/schema";

import { UserAvatar } from "@/components/ui/user-avatar";

import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card";

import { Button } from "@/components/ui/button";

import { Input } from "@/components/ui/input";

import { Label } from "@/components/ui/label";

import {

Dialog,

DialogContent,

DialogDescription,

DialogHeader,

DialogTitle,

DialogTrigger,

} from "@/components/ui/dialog";

import { useForm } from "react-hook-form";

import { zodResolver } from "@hookform/resolvers/zod";

import { z } from "zod";

import { Lock, Edit } from "lucide-react";

import { useAuth } from "@/hooks/use-auth";

interface ProfileSectionProps {

user: User;

}

// Change password form schema

const changePasswordSchema = z.object({

currentPassword: z.string().min(1, "Current password is required"),

newPassword: z.string().min(6, "New password must be at least 6 characters"),

confirmPassword: z.string().min(6, "Confirm password is required")

}).refine(data => data.newPassword === data.confirmPassword, {

message: "Passwords do not match",

path: ["confirmPassword"]

});

type ChangePasswordFormValues = z.infer<typeof changePasswordSchema>;

export function ProfileSection({ user }: ProfileSectionProps) {

const { logoutMutation } = useAuth();

const [changePasswordOpen, setChangePasswordOpen] = useState(false);

// Format date of birth

const formattedDob = user.dob ? new Date(user.dob).toLocaleDateString() : "Not provided";

// Change password form

const form = useForm<ChangePasswordFormValues>({

resolver: zodResolver(changePasswordSchema),

defaultValues: {

currentPassword: "",

newPassword: "",

confirmPassword: ""

}

});

const onSubmit = (data: ChangePasswordFormValues) => {

// In a real app, we would call an API to change the password

console.log("Change password:", data);

setChangePasswordOpen(false);

form.reset();

};

return (

<div className="space-y-6">

<div className="flex items-center space-x-4 mb-6">

<UserAvatar user={user} size="lg" />

<div>

<h2 className="text-xl font-semibold text-white">{user.fullName}</h2>

<p className="text-gray-400">{user.role === "admin" ? "Administrator" : "Student"}</p>

</div>

</div>

<Card className="bg-gray-800 border-gray-700">

<CardHeader>

<CardTitle className="text-white">Profile Information</CardTitle>

</CardHeader>

<CardContent className="space-y-6">

<div className="flex flex-col sm:flex-row">

<div className="sm:w-1/3 font-medium text-gray-400 mb-1 sm:mb-0">Full Name</div>

<div className="sm:w-2/3 text-white">{user.fullName}</div>

</div>

<div className="flex flex-col sm:flex-row">

<div className="sm:w-1/3 font-medium text-gray-400 mb-1 sm:mb-0">User ID</div>

<div className="sm:w-2/3 text-white">{user.username}</div>

</div>

<div className="flex flex-col sm:flex-row">

<div className="sm:w-1/3 font-medium text-gray-400 mb-1 sm:mb-0">Email</div>

<div className="sm:w-2/3 text-white">{user.email || "Not provided"}</div>

</div>

<div className="flex flex-col sm:flex-row">

<div className="sm:w-1/3 font-medium text-gray-400 mb-1 sm:mb-0">Date of Birth</div>

<div className="sm:w-2/3 text-white">{formattedDob}</div>

</div>

<div className="flex flex-col sm:flex-row">

<div className="sm:w-1/3 font-medium text-gray-400 mb-1 sm:mb-0">Mobile Number</div>

<div className="sm:w-2/3 text-white">{user.mobile || "Not provided"}</div>

</div>

</CardContent>

</Card>

<Card className="bg-gray-800 border-gray-700">

<CardHeader>

<CardTitle className="text-white">Account Settings</CardTitle>

</CardHeader>

<CardContent>

<div className="space-y-4">

<Dialog open={changePasswordOpen} onOpenChange={setChangePasswordOpen}>

<DialogTrigger asChild>

<Button variant="link" className="flex items-center text-primary hover:text-blue-700 p-0">

<Lock className="mr-2 h-4 w-4" />

Change Password

</Button>

</DialogTrigger>

<DialogContent>

<DialogHeader>

<DialogTitle>Change Password</DialogTitle>

<DialogDescription>

Enter your current password and a new password below.

</DialogDescription>

</DialogHeader>

<form onSubmit={form.handleSubmit(onSubmit)} className="space-y-4">

<div className="space-y-2">

<Label htmlFor="currentPassword">Current Password</Label>

<Input

id="currentPassword"

type="password"

{...form.register("currentPassword")}

/>

{form.formState.errors.currentPassword && (

<p className="text-sm text-red-500">

{form.formState.errors.currentPassword.message}

</p>

)}

</div>

<div className="space-y-2">

<Label htmlFor="newPassword">New Password</Label>

<Input

id="newPassword"

type="password"

{...form.register("newPassword")}

/>

{form.formState.errors.newPassword && (

<p className="text-sm text-red-500">

{form.formState.errors.newPassword.message}

</p>

)}

</div>

<div className="space-y-2">

<Label htmlFor="confirmPassword">Confirm New Password</Label>

<Input

id="confirmPassword"

type="password"

{...form.register("confirmPassword")}

/>

{form.formState.errors.confirmPassword && (

<p className="text-sm text-red-500">

{form.formState.errors.confirmPassword.message}

</p>

)}

</div>

<div className="flex justify-end space-x-2 pt-2">

<Button

type="button"

variant="outline"

onClick={() => setChangePasswordOpen(false)}

>

Cancel

</Button>

<Button type="submit">Save Changes</Button>

</div>

</form>

</DialogContent>

</Dialog>

<Button

variant="link"

className="flex items-center text-primary hover:text-blue-700 p-0"

onClick={() => {

// In a real app we would call an API to update the profile

console.log("Edit profile clicked");

}}

>

<Edit className="mr-2 h-4 w-4" />

Edit Profile

</Button>

</div>

</CardContent>

</Card>

<Card className="bg-gray-800 border-gray-700">

<CardHeader>

<CardTitle className="text-white">App Information</CardTitle>

</CardHeader>

<CardContent>

<div className="space-y-2">

<div className="flex flex-col sm:flex-row">

<div className="sm:w-1/3 font-medium text-gray-400 mb-1 sm:mb-0">App Version</div>

<div className="sm:w-2/3 text-white">Qroll v1.0.2</div>

</div>

<div className="flex flex-col sm:flex-row">

<div className="sm:w-1/3 font-medium text-gray-400 mb-1 sm:mb-0">Last Updated</div>

<div className="sm:w-2/3 text-white">June 10, 2023</div>

</div>

{user.role === "admin" && (

<div className="flex flex-col sm:flex-row">

<div className="sm:w-1/3 font-medium text-gray-400 mb-1 sm:mb-0">Server Status</div>

<div className="sm:w-2/3 text-white">

<span className="inline-flex items-center">

<span className="h-2 w-2 rounded-full bg-green-500 mr-2"></span>

Online

</span>

</div>

</div>

)}

</div>

</CardContent>

</Card>

<Button

variant="destructive"

className="w-full"

onClick={() => logoutMutation.mutate()}

disabled={logoutMutation.isPending}

>

{logoutMutation.isPending ? "Logging out..." : "Logout"}

</Button>

</div>

);

}

import { useState } from "react";

import { User } from "@shared/schema";

import { UserAvatar } from "@/components/ui/user-avatar";

import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card";

import { Button } from "@/components/ui/button";

import { Input } from "@/components/ui/input";

import { Label } from "@/components/ui/label";

import {

Dialog,

DialogContent,

DialogDescription,

DialogHeader,

DialogTitle,

DialogTrigger,

} from "@/components/ui/dialog";

import { useForm } from "react-hook-form";

import { zodResolver } from "@hookform/resolvers/zod";

import { z } from "zod";

import { Lock, Edit } from "lucide-react";

import { useAuth } from "@/hooks/use-auth";

interface ProfileSectionProps {

user: User;

}

// Change password form schema

const changePasswordSchema = z.object({

currentPassword: z.string().min(1, "Current password is required"),

newPassword: z.string().min(6, "New password must be at least 6 characters"),

confirmPassword: z.string().min(6, "Confirm password is required")

}).refine(data => data.newPassword === data.confirmPassword, {

message: "Passwords do not match",

path: ["confirmPassword"]

});

type ChangePasswordFormValues = z.infer<typeof changePasswordSchema>;

export function ProfileSection({ user }: ProfileSectionProps) {

const { logoutMutation } = useAuth();

const [changePasswordOpen, setChangePasswordOpen] = useState(false);

// Format date of birth

const formattedDob = user.dob ? new Date(user.dob).toLocaleDateString() : "Not provided";

// Change password form

const form = useForm<ChangePasswordFormValues>({

resolver: zodResolver(changePasswordSchema),

defaultValues: {

currentPassword: "",

newPassword: "",

confirmPassword: ""

}

});

const onSubmit = (data: ChangePasswordFormValues) => {

// In a real app, we would call an API to change the password

console.log("Change password:", data);

setChangePasswordOpen(false);

form.reset();

};

return (

<div className="space-y-6">

<div className="flex items-center space-x-4 mb-6">

<UserAvatar user={user} size="lg" />

<div>

<h2 className="text-xl font-semibold text-white">{user.fullName}</h2>

<p className="text-gray-400">{user.role === "admin" ? "Administrator" : "Student"}</p>

</div>

</div>

{/\* Profile Information Card \*/}

<Card className="bg-gray-800 border-gray-700">

<CardHeader>

<CardTitle className="text-white">Profile Information</CardTitle>

</CardHeader>

<CardContent className="space-y-6">

<div className="flex flex-col sm:flex-row">

<div className="sm:w-1/3 font-medium text-gray-400 mb-1 sm:mb-0">Full Name</div>

<div className="sm:w-2/3 text-white">{user.fullName}</div>

</div>

<div className="flex flex-col sm:flex-row">

<div className="sm:w-1/3 font-medium text-gray-400 mb-1 sm:mb-0">User ID</div>

<div className="sm:w-2/3 text-white">{user.username}</div>

</div>

<div className="flex flex-col sm:flex-row">

<div className="sm:w-1/3 font-medium text-gray-400 mb-1 sm:mb-0">Email</div>

<div className="sm:w-2/3 text-white">{user.email || "Not provided"}</div>

</div>

<div className="flex flex-col sm:flex-row">

<div className="sm:w-1/3 font-medium text-gray-400 mb-1 sm:mb-0">Date of Birth</div>

<div className="sm:w-2/3 text-white">{formattedDob}</div>

</div>

<div className="flex flex-col sm:flex-row">

<div className="sm:w-1/3 font-medium text-gray-400 mb-1 sm:mb-0">Mobile Number</div>

<div className="sm:w-2/3 text-white">{user.mobile || "Not provided"}</div>

</div>

</CardContent>

</Card>

{/\* Account Settings Card \*/}

<Card className="bg-gray-800 border-gray-700">

<CardHeader>

<CardTitle className="text-white">Account Settings</CardTitle>

</CardHeader>

<CardContent>

<div className="space-y-4">

{/\* Change password dialog \*/}

<Dialog open={changePasswordOpen} onOpenChange={setChangePasswordOpen}>

<DialogTrigger asChild>

<Button variant="link" className="flex items-center text-primary hover:text-blue-700 p-0">

<Lock className="mr-2 h-4 w-4" />

Change Password

</Button>

</DialogTrigger>

<DialogContent>

<DialogHeader>

<DialogTitle>Change Password</DialogTitle>

<DialogDescription>

Enter your current password and a new password below.

</DialogDescription>

</DialogHeader>

<form onSubmit={form.handleSubmit(onSubmit)} className="space-y-4">

<div className="space-y-2">

<Label htmlFor="currentPassword">Current Password</Label>

<Input

id="currentPassword"

type="password"

{...form.register("currentPassword")}

/>

{form.formState.errors.currentPassword && (

<p className="text-sm text-red-500">

{form.formState.errors.currentPassword.message}

</p>

)}

</div>

<div className="space-y-2">

<Label htmlFor="newPassword">New Password</Label>

<Input

id="newPassword"

type="password"

{...form.register("newPassword")}

/>

{form.formState.errors.newPassword && (

<p className="text-sm text-red-500">

{form.formState.errors.newPassword.message}

</p>

)}

</div>

<div className="space-y-2">

<Label htmlFor="confirmPassword">Confirm New Password</Label>

<Input

id="confirmPassword"

type="password"

{...form.register("confirmPassword")}

/>

{form.formState.errors.confirmPassword && (

<p className="text-sm text-red-500">

{form.formState.errors.confirmPassword.message}

</p>

)}

</div>

<div className="flex justify-end space-x-2 pt-2">

<Button

type="button"

variant="outline"

onClick={() => setChangePasswordOpen(false)}

>

Cancel

</Button>

<Button type="submit">Save Changes</Button>

</div>

</form>

</DialogContent>

</Dialog>

<Button

variant="link"

className="flex items-center text-primary hover:text-blue-700 p-0"

onClick={() => {

// In a real app we would call an API to update the profile

console.log("Edit profile clicked");

}}

>

<Edit className="mr-2 h-4 w-4" />

Edit Profile

</Button>

</div>

</CardContent>

</Card>

{/\* App Information Card \*/}

<Card className="bg-gray-800 border-gray-700">

<CardHeader>

<CardTitle className="text-white">App Information</CardTitle>

</CardHeader>

<CardContent>

<div className="space-y-2">

<div className="flex flex-col sm:flex-row">

<div className="sm:w-1/3 font-medium text-gray-400 mb-1 sm:mb-0">App Version</div>

<div className="sm:w-2/3 text-white">Qroll v1.0.2</div>

</div>

<div className="flex flex-col sm:flex-row">

<div className="sm:w-1/3 font-medium text-gray-400 mb-1 sm:mb-0">Last Updated</div>

<div className="sm:w-2/3 text-white">June 10, 2023</div>

</div>

{user.role === "admin" && (

<div className="flex flex-col sm:flex-row">

<div className="sm:w-1/3 font-medium text-gray-400 mb-1 sm:mb-0">Server Status</div>

<div className="sm:w-2/3 text-white">

<span className="inline-flex items-center">

<span className="h-2 w-2 rounded-full bg-green-500 mr-2"></span>

Online

</span>

</div>

</div>

)}

</div>

</CardContent>

</Card>

{/\* Logout Button \*/}

<Button

variant="destructive"

className="w-full"

onClick={() => logoutMutation.mutate()}

disabled={logoutMutation.isPending}

>

{logoutMutation.isPending ? "Logging out..." : "Logout"}

</Button>

</div>

);

}

import React from "react";

import { PieChart, Pie, Cell, ResponsiveContainer, Legend, Tooltip } from "recharts";

interface AttendanceVisualizationProps {

className?: string;

}

export function AttendanceVisualization({ className }: AttendanceVisualizationProps) {

// Sample data for pie chart

const data = [

{ name: "Present", value: 57, color: "#10b981" }, // Green

{ name: "Absent", value: 36, color: "#ef4444" }, // Red

{ name: "Holidays", value: 7, color: "#3b82f6" } // Blue

];

return (

<div className={`${className} bg-gray-900 rounded-lg shadow p-6 border border-gray-800`}>

<h3 className="text-xl font-semibold text-white mb-4">Attendance Overview</h3>

<div className="flex flex-col sm:flex-row items-center">

<div className="w-full sm:w-3/5 h-72">

<ResponsiveContainer width="100%" height="100%">

<PieChart>

<Pie

data={data}

cx="50%"

cy="50%"

innerRadius={70}

outerRadius={90}

paddingAngle={2}

dataKey="value"

label={({ name, percent }) => `${name}: ${(percent \* 100).toFixed(0)}%`}

labelLine={false}

>

{data.map((entry, index) => (

<Cell key={`cell-${index}`} fill={entry.color} />

))}

</Pie>

<Tooltip

formatter={(value) => [`${value}%`, 'Percentage']}

contentStyle={{ backgroundColor: '#1f2937', borderColor: '#374151', color: 'white' }}

/>

<Legend

verticalAlign="bottom"

align="center"

formatter={(value) => <span className="text-white">{value}</span>}

/>

</PieChart>

</ResponsiveContainer>

</div>

<div className="w-full sm:w-2/5 mt-4 sm:mt-0 pl-0 sm:pl-4">

<div className="space-y-4">

<div className="bg-gray-800 rounded-lg p-4">

<div className="flex items-center justify-between">

<div className="flex items-center">

<div className="w-3 h-3 rounded-full bg-green-500 mr-2"></div>

<span className="text-white font-medium">Present</span>

</div>

<span className="text-green-500 font-bold">57%</span>

</div>

<p className="text-gray-400 text-sm mt-1">

Attended 28 out of 50 classes

</p>

</div>

<div className="bg-gray-800 rounded-lg p-4">

<div className="flex items-center justify-between">

<div className="flex items-center">

<div className="w-3 h-3 rounded-full bg-red-500 mr-2"></div>

<span className="text-white font-medium">Absent</span>

</div>

<span className="text-red-500 font-bold">36%</span>

</div>

<p className="text-gray-400 text-sm mt-1">

Missed 18 out of 50 classes

</p>

</div>

<div className="bg-gray-800 rounded-lg p-4">

<div className="flex items-center justify-between">

<div className="flex items-center">

<div className="w-3 h-3 rounded-full bg-blue-500 mr-2"></div>

<span className="text-white font-medium">Holidays</span>

</div>

<span className="text-blue-500 font-bold">7%</span>

</div>

<p className="text-gray-400 text-sm mt-1">

4 official holidays

</p>

</div>

</div>

</div>

</div>

<div className="mt-6">

<div className="bg-gray-800 rounded-lg p-4">

<h4 className="text-white font-medium mb-2">Overall Attendance Rate</h4>

<div className="w-full bg-gray-700 rounded-full h-2.5">

<div className="bg-primary h-2.5 rounded-full" style={{ width: '57%' }}></div>

</div>

<div className="flex justify-between mt-1">

<span className="text-gray-400 text-xs">0%</span>

<span className="text-gray-400 text-xs">Minimum Required: 75%</span>

<span className="text-gray-400 text-xs">100%</span>

</div>

</div>

</div>

</div>

);

}

import { useState, useEffect } from "react";

import { useFaceRecognition } from "@/hooks/use-face-recognition";

import { Button } from "@/components/ui/button";

import { Progress } from "@/components/ui/progress";

import { useToast } from "@/hooks/use-toast";

import { Loader2, Camera, Check, AlertCircle } from "lucide-react";

import { apiRequest } from "@/lib/queryClient";

interface FaceRecognitionProps {

onSuccess: () => void;

onCancel: () => void;

}

export function FaceRecognition({ onSuccess, onCancel }: FaceRecognitionProps) {

const [stage, setStage] = useState<"initial" | "scanning" | "processing" | "success" | "error">("initial");

const { toast } = useToast();

const {

videoRef,

isInitialized,

isDetecting,

faceDetected,

progress,

error,

startDetection,

stopDetection

} = useFaceRecognition({

onDetected: async (faceData) => {

setStage("processing");

try {

// In a real app, store the face data for later use

await apiRequest("POST", "/api/user/face", { faceData });

setStage("success");

setTimeout(() => {

onSuccess();

}, 1500);

} catch (err) {

setStage("error");

toast({

title: "Face recognition failed",

description: "Unable to process your face data. Please try again.",

variant: "destructive"

});

}

},

onError: (err) => {

setStage("error");

toast({

title: "Face recognition error",

description: err.message,

variant: "destructive"

});

}

});

const handleStartDetection = () => {

setStage("scanning");

startDetection();

};

const handleCancel = () => {

stopDetection();

onCancel();

};

const handleTryAgain = () => {

setStage("initial");

};

useEffect(() => {

return () => {

stopDetection();

};

}, []);

return (

<div className="space-y-6">

{stage === "initial" && (

<div className="border-2 border-dashed border-gray-300 rounded-lg p-6 text-center">

<div className="flex justify-center mb-4">

<Camera className="h-12 w-12 text-gray-400" />

</div>

<p className="text-gray-500 mb-4">Face recognition required before scanning QR code</p>

<Button

onClick={handleStartDetection}

className="bg-primary hover:bg-primary/90"

>

Start Face Recognition

</Button>

</div>

)}

{stage === "scanning" && (

<div className="border-2 border-primary rounded-lg p-6 text-center">

<div className="relative mx-auto w-64 h-64 bg-gray-100 rounded-lg overflow-hidden mb-4">

<video

ref={videoRef}

autoPlay

playsInline

muted

className="w-full h-full object-cover"

/>

<div className="absolute inset-0 flex items-center justify-center">

{!faceDetected && (

<div className="border-2 border-dashed border-gray-400 w-40 h-40 rounded-full flex items-center justify-center">

<p className="text-sm text-gray-500">Position your face here</p>

</div>

)}

</div>

</div>

<p className="text-gray-700 mb-4">

{faceDetected

? "Face detected! Please keep looking at the camera..."

: "Please look directly at the camera"

}

</p>

<Progress value={progress} className="mb-4" />

<Button

variant="outline"

onClick={handleCancel}

>

Cancel

</Button>

</div>

)}

{stage === "processing" && (

<div className="border-2 border-primary rounded-lg p-6 text-center">

<div className="flex justify-center mb-4">

<Loader2 className="h-12 w-12 text-primary animate-spin" />

</div>

<p className="text-gray-700 mb-4">Processing your face data...</p>

</div>

)}

{stage === "success" && (

<div className="border-2 border-green-500 rounded-lg p-6 text-center">

<div className="flex justify-center mb-4">

<Check className="h-12 w-12 text-green-500" />

</div>

<p className="text-gray-700 mb-4">Face recognized successfully!</p>

</div>

)}

{stage === "error" && (

<div className="border-2 border-red-500 rounded-lg p-6 text-center">

<div className="flex justify-center mb-4">

<AlertCircle className="h-12 w-12 text-red-500" />

</div>

<p className="text-gray-700 mb-4">

{error ? error.message : "Failed to recognize face. Please try again."}

</p>

<div className="flex justify-center space-x-3">

<Button onClick={handleTryAgain}>Try Again</Button>

<Button variant="outline" onClick={handleCancel}>Cancel</Button>

</div>

</div>

)}

</div>

);

}

import { useState, useEffect } from "react";

import { useGeolocation } from "@/hooks/use-geolocation";

import { Button } from "@/components/ui/button";

import { Card, CardContent } from "@/components/ui/card";

import { Progress } from "@/components/ui/progress";

import { useToast } from "@/hooks/use-toast";

import { MapPin, Check, AlertCircle, Loader2 } from "lucide-react";

interface LocationVerificationProps {

onSuccess: (coords: { latitude: number; longitude: number }) => void;

onCancel: () => void;

}

export function LocationVerification({ onSuccess, onCancel }: LocationVerificationProps) {

const [stage, setStage] = useState<"initial" | "verifying" | "success" | "error">("initial");

const [progress, setProgress] = useState(0);

const { toast } = useToast();

const {

latitude,

longitude,

accuracy,

loading,

error,

hasPermission,

getCurrentPosition,

startWatching,

stopWatching

} = useGeolocation({

enableHighAccuracy: true,

onError: (err) => {

setStage("error");

toast({

title: "Location error",

description: err.message,

variant: "destructive"

});

}

});

const handleStartVerification = () => {

setStage("verifying");

setProgress(0);

getCurrentPosition();

startWatching();

};

const handleCancel = () => {

stopWatching();

onCancel();

};

const handleTryAgain = () => {

setStage("initial");

};

// Process location data once we have a valid reading

useEffect(() => {

let progressInterval: NodeJS.Timeout;

if (stage === "verifying" && latitude !== null && longitude !== null) {

// Simulate a progress bar to show location verification in progress

progressInterval = setInterval(() => {

setProgress(prevProgress => {

const newProgress = prevProgress + 5;

if (newProgress >= 100) {

clearInterval(progressInterval);

setStage("success");

// Wait a moment before calling onSuccess

setTimeout(() => {

onSuccess({ latitude, longitude });

}, 1000);

return 100;

}

return newProgress;

});

}, 50);

}

return () => {

if (progressInterval) clearInterval(progressInterval);

};

}, [stage, latitude, longitude]);

// Cleanup on unmount

useEffect(() => {

return () => {

stopWatching();

};

}, []);

return (

<div className="space-y-6">

{stage === "initial" && (

<div className="border-2 border-dashed border-gray-300 rounded-lg p-6 text-center">

<div className="flex justify-center mb-4">

<MapPin className="h-12 w-12 text-primary" />

</div>

<h3 className="text-lg font-medium text-gray-800 mb-2">Location Verification Required</h3>

<p className="text-gray-500 mb-4">

You need to verify your location before proceeding with attendance

</p>

<Button

onClick={handleStartVerification}

className="bg-primary hover:bg-primary/90 text-white"

>

Verify Location

</Button>

</div>

)}

{stage === "verifying" && (

<div className="border-2 border-primary rounded-lg p-6 text-center">

<div className="flex justify-center mb-4">

<Loader2 className="h-12 w-12 text-primary animate-spin" />

</div>

<h3 className="text-lg font-medium text-gray-800 mb-2">Verifying Your Location</h3>

<p className="text-gray-500 mb-4">

Please wait while we verify your location...

</p>

<Progress value={progress} className="mb-4" />

{accuracy && (

<p className="text-xs text-gray-400 mb-4">

Location accuracy: {Math.round(accuracy)} meters

</p>

)}

<Button

variant="outline"

onClick={handleCancel}

>

Cancel

</Button>

</div>

)}

{stage === "success" && (

<div className="border-2 border-green-500 rounded-lg p-6 text-center">

<div className="flex justify-center mb-4">

<Check className="h-12 w-12 text-green-500" />

</div>

<h3 className="text-lg font-medium text-green-800 mb-2">Location Verified!</h3>

<p className="text-green-600 mb-4">

Your location has been successfully verified

</p>

</div>

)}

{stage === "error" && (

<div className="border-2 border-red-500 rounded-lg p-6 text-center">

<div className="flex justify-center mb-4">

<AlertCircle className="h-12 w-12 text-red-500" />

</div>

<h3 className="text-lg font-medium text-red-800 mb-2">Location Error</h3>

<p className="text-gray-700 mb-4">

{error || "Failed to access your location. Please ensure location services are enabled."}

</p>

<div className="flex justify-center space-x-3">

<Button

onClick={handleTryAgain}

className="bg-primary hover:bg-primary/90 text-white"

>

Try Again

</Button>

<Button

variant="outline"

onClick={handleCancel}

>

Cancel

</Button>

</div>

</div>

)}

</div>

);

}

import { useState, useEffect } from "react";

import { useQRScanner } from "@/hooks/use-qr-scanner";

import { useGeolocation } from "@/hooks/use-geolocation";

import { Button } from "@/components/ui/button";

import { Card, CardContent } from "@/components/ui/card";

import { useToast } from "@/hooks/use-toast";

import { apiRequest } from "@/lib/queryClient";

import { Camera, MapPin, CheckCircle, AlertCircle, Loader2 } from "lucide-react";

interface QRScannerProps {

onSuccess: (sessionData: any) => void;

onCancel: () => void;

}

interface QRCodeData {

sessionId: number;

qrCode: string;

name: string;

time: string;

location: string;

}

export function QRScanner({ onSuccess, onCancel }: QRScannerProps) {

const [stage, setStage] = useState<"initial" | "scanning" | "processing" | "success" | "error">("initial");

const [locationPermissionGranted, setLocationPermissionGranted] = useState(false);

const [qrCodeData, setQrCodeData] = useState<QRCodeData | null>(null);

const [error, setError] = useState<string | null>(null);

const { toast } = useToast();

// QR Scanner Hook

const {

videoRef,

isScanning,

hasPermission,

error: scannerError,

startScanning,

stopScanning

} = useQRScanner({

onDetected: (result) => {

handleQRCodeDetected(result);

},

onError: (err) => {

setStage("error");

setError(err.message);

toast({

title: "QR Scanner error",

description: err.message,

variant: "destructive"

});

}

});

// Geolocation Hook

const {

latitude,

longitude,

hasPermission: hasLocationPermission,

getCurrentPosition,

startWatching,

stopWatching

} = useGeolocation({

enableHighAccuracy: true,

onError: (err) => {

toast({

title: "Location error",

description: err.message,

variant: "destructive"

});

}

});

// Handle QR code detection

const handleQRCodeDetected = async (result: string) => {

try {

stopScanning();

setStage("processing");

// Parse QR code data

let data: QRCodeData;

try {

data = JSON.parse(result);

} catch (err) {

throw new Error("Invalid QR code format");

}

setQrCodeData(data);

// Submit attendance with location and face verification data

const response = await apiRequest("POST", "/api/attendance", {

sessionId: data.sessionId,

qrCode: data.qrCode,

verificationMethod: hasLocationPermission ? "both" : "face",

latitude,

longitude,

faceVerified: true // Face is already verified in the authentication flow

});

setStage("success");

// Pass the session data to parent component

setTimeout(() => {

onSuccess(data);

}, 2000);

} catch (err: any) {

setStage("error");

setError(err.message || "Failed to process QR code");

toast({

title: "Attendance marking failed",

description: err.message || "Failed to process QR code",

variant: "destructive"

});

}

};

// Handle location permission

const handleRequestLocationPermission = () => {

getCurrentPosition();

startWatching();

setLocationPermissionGranted(true);

};

// Handle start scanning

const handleStartScanning = () => {

setStage("scanning");

startScanning();

};

// Handle cancel

const handleCancel = () => {

stopScanning();

stopWatching();

onCancel();

};

// Handle try again

const handleTryAgain = () => {

setStage("initial");

setError(null);

};

// Cleanup on unmount

useEffect(() => {

return () => {

stopScanning();

stopWatching();

};

}, []);

return (

<div className="space-y-6">

{/\* Initial state \*/}

{stage === "initial" && (

<div className="border-2 border-green-500 rounded-lg p-6 text-center">

<div className="flex justify-center mb-4">

<Camera className="h-12 w-12 text-green-500" />

</div>

<p className="text-gray-700 mb-4">Face recognized! Scan your attendance QR code</p>

<Button

onClick={handleStartScanning}

className="mb-4"

>

Start QR Scanner

</Button>

{!locationPermissionGranted && (

<div className="bg-blue-50 border border-blue-200 rounded-lg p-4">

<div className="flex">

<div className="flex-shrink-0">

<MapPin className="h-5 w-5 text-blue-500" />

</div>

<div className="ml-3">

<h3 className="text-sm font-medium text-blue-800">Location Permission Required</h3>

<p className="text-sm text-blue-700 mt-1">

This app needs access to your location to verify your attendance.

</p>

<div className="mt-2">

<Button

variant="link"

className="text-xs font-medium text-blue-800 hover:text-blue-900 p-0"

onClick={handleRequestLocationPermission}

>

Grant Permission

</Button>

</div>

</div>

</div>

</div>

)}

</div>

)}

{/\* Scanning state \*/}

{stage === "scanning" && (

<div className="border-2 border-green-500 rounded-lg p-6 text-center">

<div className="relative w-64 h-64 mx-auto bg-gray-100 rounded-lg overflow-hidden mb-4">

<video

ref={videoRef}

className="absolute inset-0 w-full h-full object-cover"

autoPlay

playsInline

muted

/>

<div className="absolute inset-0 flex items-center justify-center">

<div className="border-2 border-primary border-dashed w-40 h-40"></div>

</div>

</div>

<p className="text-xs text-gray-500 mb-4">Position the QR code within the frame</p>

<Button

variant="outline"

onClick={handleCancel}

>

Cancel

</Button>

</div>

)}

{/\* Processing state \*/}

{stage === "processing" && (

<div className="border-2 border-primary rounded-lg p-6 text-center">

<div className="flex justify-center mb-4">

<Loader2 className="h-12 w-12 text-primary animate-spin" />

</div>

<p className="text-gray-700 mb-4">Processing QR code...</p>

</div>

)}

{/\* Success state \*/}

{stage === "success" && qrCodeData && (

<div className="bg-green-50 border border-green-200 rounded-lg p-6 text-center">

<div className="flex justify-center mb-4">

<CheckCircle className="h-12 w-12 text-green-500" />

</div>

<h3 className="text-lg font-medium text-green-800 mb-2">Attendance Recorded!</h3>

<p className="text-green-700 mb-4">Successfully marked present for:</p>

<div className="bg-white rounded-md p-4 mb-4">

<p className="font-medium text-gray-800">{qrCodeData.name}</p>

<p className="text-gray-600">{qrCodeData.time}</p>

<p className="text-gray-600">{qrCodeData.location}</p>

</div>

</div>

)}

{/\* Error state \*/}

{stage === "error" && (

<div className="border-2 border-red-500 rounded-lg p-6 text-center">

<div className="flex justify-center mb-4">

<AlertCircle className="h-12 w-12 text-red-500" />

</div>

<p className="text-gray-700 mb-4">

{error || "Error scanning QR code. Please try again."}

</p>

<div className="flex justify-center space-x-3">

<Button onClick={handleTryAgain}>Try Again</Button>

<Button variant="outline" onClick={handleCancel}>Cancel</Button>

</div>

</div>

)}

</div>

);

}

import \* as React from "react"

import \* as AccordionPrimitive from "@radix-ui/react-accordion"

import { ChevronDown } from "lucide-react"

import { cn } from "@/lib/utils"

const Accordion = AccordionPrimitive.Root

const AccordionItem = React.forwardRef<

React.ElementRef<typeof AccordionPrimitive.Item>,

React.ComponentPropsWithoutRef<typeof AccordionPrimitive.Item>

>(({ className, ...props }, ref) => (

<AccordionPrimitive.Item

ref={ref}

className={cn("border-b", className)}

{...props}

/>

))

AccordionItem.displayName = "AccordionItem"

const AccordionTrigger = React.forwardRef<

React.ElementRef<typeof AccordionPrimitive.Trigger>,

React.ComponentPropsWithoutRef<typeof AccordionPrimitive.Trigger>

>(({ className, children, ...props }, ref) => (

<AccordionPrimitive.Header className="flex">

<AccordionPrimitive.Trigger

ref={ref}

className={cn(

"flex flex-1 items-center justify-between py-4 font-medium transition-all hover:underline [&[data-state=open]>svg]:rotate-180",

className

)}

{...props}

>

{children}

<ChevronDown className="h-4 w-4 shrink-0 transition-transform duration-200" />

</AccordionPrimitive.Trigger>

</AccordionPrimitive.Header>

))

AccordionTrigger.displayName = AccordionPrimitive.Trigger.displayName

const AccordionContent = React.forwardRef<

React.ElementRef<typeof AccordionPrimitive.Content>,

React.ComponentPropsWithoutRef<typeof AccordionPrimitive.Content>

>(({ className, children, ...props }, ref) => (

<AccordionPrimitive.Content

ref={ref}

className="overflow-hidden text-sm transition-all data-[state=closed]:animate-accordion-up data-[state=open]:animate-accordion-down"

{...props}

>

<div className={cn("pb-4 pt-0", className)}>{children}</div>

</AccordionPrimitive.Content>

))

AccordionContent.displayName = AccordionPrimitive.Content.displayName

export { Accordion, AccordionItem, AccordionTrigger, AccordionContent

@tailwind base;

@tailwind components;

@tailwind utilities;

@layer base {

\* {

@apply border-border;

}

body {

@apply font-sans antialiased bg-background text-foreground;

}

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta charset="UTF-8" />**

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

**<link rel="icon" type="image/svg+xml" href="/favicon.svg" />**

**<title>Qroll - Student Attendance System</title>**

**<meta name="description" content="Student attendance tracking application using QR codes with face recognition verification and geolocation features" />**

**<!-- Google Fonts -->**

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

**<link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">**

**<!-- Load face-api.js models at the root level -->**

**<script>**

**// Create a function to preload face-api.js models**

**async function loadModels() {**

**try {**

**// Create models directory if it doesn't exist**

**const modelsDir = '/models';**

**// Dynamically create a models folder with the required files**

**// This would normally be handled by serving actual model files,**

**// but for this example we'll use placeholder data to avoid errors**

**const files = [**

**'face\_landmark\_68\_model-weights\_manifest.json',**

**'face\_landmark\_68\_model-shard1',**

**'face\_recognition\_model-weights\_manifest.json',**

**'face\_recognition\_model-shard1',**

**'tiny\_face\_detector\_model-weights\_manifest.json',**

**'tiny\_face\_detector\_model-shard1'**

**];**

**console.log('Face recognition models ready to be loaded');**

**} catch (error) {**

**console.error('Error setting up face-api models:', error);**

**}**

**}**

**// Call the function**

**loadModels();**

**</script>**

**</head>**

**<body>**

**<div id="root"></div>**

**<script type="module" src="/src/main.tsx"></script>**

**</body>**

**</html>**

**import QRCode from 'qrcode';**

**import fs from 'fs';**

**// Generate QR Code for the Qroll attendance app**

**const appUrl = 'http://localhost:5000';**

**// Generate QR code as a data URL**

**QRCode.toDataURL(appUrl, {**

**errorCorrectionLevel: 'H',**

**type: 'image/png',**

**margin: 1,**

**color: {**

**dark: '#8b5cf6', // Purple color for QR code**

**light: '#ffffff' // White background**

**}**

**})**

**.then(url => {**

**// Save base64 data to a file (remove the data:image/png;base64, prefix)**

**const data = url.replace(/^data:image\/png;base64,/, '');**

**fs.writeFileSync('qroll-app-url.txt', url);**

**console.log('QR Code for Qroll App generated successfully!');**

**console.log('App URL:', appUrl);**

**console.log('QR Code data saved to qroll-app-url.txt');**

**})**

**.catch(err => {**

**console.error('Error generating QR code:', err);**

**});**

**#!/usr/bin/env node**

**const QRCode = require('qrcode');**

**const { writeFileSync } = require('fs');**

**const path = require('path');**

**// Get command line arguments**

**const args = process.argv.slice(2);**

**const sessionId = args[0];**

**const sessionName = args[1] || 'Attendance Session';**

**if (!sessionId) {**

**console.error('Usage: node generate-session-qr.js <sessionId> [sessionName]');**

**process.exit(1);**

**}**

**// Create data for QR code**

**const qrData = JSON.stringify({**

**type: 'qroll-session',**

**sessionId: parseInt(sessionId),**

**timestamp: Date.now(),**

**});**

**// Output file name**

**const outputFile = `qroll-session-${sessionId}.png`;**

**// Generate QR code**

**QRCode.toFile(**

**outputFile,**

**qrData,**

**{**

**color: {**

**dark: '#8b5cf6', // Purple color**

**light: '#ffffff' // White background**

**},**

**width: 512,**

**margin: 1,**

**},**

**function(err) {**

**if (err) {**

**console.error('Error generating QR code:', err);**

**process.exit(1);**

**}**

**console.log(`QR code for session ${sessionId} (${sessionName}) generated successfully!`);**

**console.log(`File saved as: ${outputFile}`);**

**console.log(`\nScan this QR code with the Qroll app to mark attendance.`);**

**}**

**);**

**export default {**

**plugins: {**

**tailwindcss: {},**

**autoprefixer: {},**

**},**

**}**