config/[database.js](http://database.js)

const mongoose = require('mongoose');

require('dotenv').config();

const connectDB = async () => {

try {

const conn = await mongoose.connect(process.env.MONGODB\_URI, {

useNewUrlParser: true,

useUnifiedTopology: true,

});

console.log(`MongoDB Connected: ${conn.connection.host}`);

// Create indexes for better performance

await createIndexes();

} catch (error) {

console.error('Database connection error:', error);

process.exit(1);

}

};

const createIndexes = async () => {

try {

const User = require('../models/User');

const Token = require('../models/Token');

const Session = require('../models/Session');

// Create indexes

await User.createIndexes();

await Token.createIndexes();

await Session.createIndexes();

console.log('Database indexes created successfully');

} catch (error) {

console.error('Error creating indexes:', error);

}

};

module.exports = connectDB;

middleware/[cleanup.js](http://cleanup.js)

const Token = require('../models/Token');

const Session = require('../models/Session');

/\*\*

\* Cleanup expired tokens and sessions

\*/

async function cleanupExpiredData() {

try {

console.log('Running cleanup for expired data...');

// Cleanup expired tokens

const expiredTokens = await Token.cleanupExpiredTokens();

if (expiredTokens > 0) {

console.log(`Marked ${expiredTokens} tokens as expired`);

}

// Cleanup expired sessions

const expiredSessions = await Session.cleanupExpiredSessions();

if (expiredSessions > 0) {

console.log(`Marked ${expiredSessions} sessions as expired`);

}

} catch (error) {

console.error('Error during cleanup:', error);

}

}

/\*\*

\* Auto-delete expired tokens if enabled

\*/

async function autoDeleteExpiredTokens() {

try {

const deletedCount = await Token.deleteExpiredTokens();

if (deletedCount > 0) {

console.log(`Auto-deleted ${deletedCount} expired tokens`);

}

return deletedCount;

} catch (error) {

console.error('Error auto-deleting expired tokens:', error);

return 0;

}

}

module.exports = {

cleanupExpiredData,

autoDeleteExpiredTokens

};

//models/[session.js](http://session.js)

const mongoose = require('mongoose');

const sessionSchema = new mongoose.Schema({

username: {

type: String,

required: true,

trim: true

},

device\_id: {

type: String,

required: true,

trim: true

},

ip: {

type: String,

required: true

},

started\_at: {

type: Date,

default: Date.now

},

expires\_at: {

type: Date,

required: true

},

status: {

type: String,

enum: ['ACTIVE', 'FORCED\_LOGOUT', 'EXPIRED'],

default: 'ACTIVE'

}

});

// Indexes for performance

sessionSchema.index({ username: 1 });

sessionSchema.index({ device\_id: 1 });

sessionSchema.index({ status: 1 });

sessionSchema.index({ expires\_at: 1 });

sessionSchema.index({ started\_at: -1 });

// Static method to create new session

sessionSchema.statics.createSession = async function(username, device\_id, ip, expiryMinutes = 10) {

const expires\_at = new Date(Date.now() + (expiryMinutes \* 60 \* 1000));

// End any existing active sessions for this user

await this.updateMany(

{ username, status: 'ACTIVE' },

{ $set: { status: 'EXPIRED' } }

);

return await this.create({

username,

device\_id,

ip,

expires\_at

});

};

// Static method to validate session

sessionSchema.statics.validateSession = async function(sessionId) {

const session = await this.findById(sessionId);

if (!session) {

return { valid: false, reason: 'SESSION\_NOT\_FOUND' };

}

if (session.status !== 'ACTIVE') {

return { valid: false, reason: 'SESSION\_INACTIVE' };

}

if (session.expires\_at < new Date()) {

// Mark as expired

session.status = 'EXPIRED';

await session.save();

return { valid: false, reason: 'SESSION\_EXPIRED' };

}

return { valid: true, session: session };

};

// Static method to force logout

sessionSchema.statics.forceLogout = async function(sessionId) {

const session = await this.findById(sessionId);

if (session && session.status === 'ACTIVE') {

session.status = 'FORCED\_LOGOUT';

await session.save();

return true;

}

return false;

};

// Static method to cleanup expired sessions

sessionSchema.statics.cleanupExpiredSessions = async function() {

const result = await this.updateMany(

{

status: 'ACTIVE',

expires\_at: { $lt: new Date() }

},

{

$set: { status: 'EXPIRED' }

}

);

return result.modifiedCount;

};

// Static method to get active sessions

sessionSchema.statics.getActiveSessions = async function() {

return await this.find({

status: 'ACTIVE',

expires\_at: { $gt: new Date() }

}).sort({ started\_at: -1 });

};

// Virtual for time remaining

sessionSchema.virtual('timeRemaining').get(function() {

if (this.status !== 'ACTIVE') return 0;

const now = new Date();

const remaining = Math.max(0, this.expires\_at - now);

return Math.floor(remaining / 1000); // Return seconds

});

// Virtual for session duration

sessionSchema.virtual('duration').get(function() {

const now = new Date();

const duration = now - this.started\_at;

return Math.floor(duration / 1000); // Return seconds

});

module.exports = mongoose.model('Session', sessionSchema);

//models/[Token.js](http://token.js)

const mongoose = require('mongoose');

const crypto = require('crypto');

const tokenSchema = new mongoose.Schema({

device\_id: {

type: String,

required: true,

trim: true

},

token\_hash: {

type: String,

required: true,

unique: true

},

status: {

type: String,

enum: ['ACTIVE', 'USED', 'EXPIRED'],

default: 'ACTIVE'

},

created\_at: {

type: Date,

default: Date.now

},

expires\_at: {

type: Date,

required: true

},

used\_at: {

type: Date,

default: null

}

});

// Indexes for performance

tokenSchema.index({ device\_id: 1 });

tokenSchema.index({ token\_hash: 1 });

tokenSchema.index({ status: 1 });

tokenSchema.index({ expires\_at: 1 });

tokenSchema.index({ created\_at: -1 });

// Static method to generate token hash

tokenSchema.statics.generateTokenHash = function(device\_id, plain\_token) {

return crypto.createHash('sha512').update(device\_id + plain\_token).digest('hex');

};

// Static method to generate random 6-character token

tokenSchema.statics.generateRandomToken = function() {

const chars = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789';

let token = '';

for (let i = 0; i < 6; i++) {

token += chars.charAt(Math.floor(Math.random() \* chars.length));

}

return token;

};

// Static method to create new token

tokenSchema.statics.createToken = async function(device\_id, expirySeconds = 300) {

const plain\_token = this.generateRandomToken();

const token\_hash = this.generateTokenHash(device\_id, plain\_token);

const expires\_at = new Date(Date.now() + (expirySeconds \* 1000));

const token = await this.create({

device\_id,

token\_hash,

expires\_at

});

return {

token: token,

plain\_token: plain\_token

};

};

// Static method to verify token

tokenSchema.statics.verifyToken = async function(device\_id, plain\_token) {

const token\_hash = this.generateTokenHash(device\_id, plain\_token);

const token = await this.findOne({

device\_id,

token\_hash,

status: 'ACTIVE'

});

if (!token) {

return { valid: false, reason: 'TOKEN\_NOT\_FOUND' };

}

if (token.expires\_at < new Date()) {

// Mark as expired

token.status = 'EXPIRED';

await token.save();

return { valid: false, reason: 'TOKEN\_EXPIRED' };

}

// Mark as used

token.status = 'USED';

token.used\_at = new Date();

await token.save();

return { valid: true, token: token };

};

// Static method to cleanup expired tokens

tokenSchema.statics.cleanupExpiredTokens = async function() {

const result = await this.updateMany(

{

status: 'ACTIVE',

expires\_at: { $lt: new Date() }

},

{

$set: { status: 'EXPIRED' }

}

);

return result.modifiedCount;

};

// Static method to delete expired tokens

tokenSchema.statics.deleteExpiredTokens = async function() {

const result = await this.deleteMany({

status: 'EXPIRED'

});

return result.deletedCount;

};

// Virtual for time remaining

tokenSchema.virtual('timeRemaining').get(function() {

if (this.status !== 'ACTIVE') return 0;

const now = new Date();

const remaining = Math.max(0, this.expires\_at - now);

return Math.floor(remaining / 1000); // Return seconds

});

module.exports = mongoose.model('Token', tokenSchema);

//models/[user.js](http://user.js)

const mongoose = require('mongoose');

const bcrypt = require('bcrypt');

const { encrypt, decrypt } = require('../utils/encryption');

const userSchema = new mongoose.Schema({

username: {

type: String,

required: true,

unique: true,

trim: true,

minlength: 3,

maxlength: 50

},

password\_hash: {

type: String,

required: true

},

vauth\_device\_ID: {

type: String,

required: true,

unique: true,

trim: true

},

// Encrypted PII fields

name\_enc: {

type: String,

required: true

},

email\_enc: {

type: String,

required: true

},

mobile\_enc: {

type: String,

required: true

},

operating\_country\_enc: {

type: String,

required: true

},

created\_at: {

type: Date,

default: Date.now

}

});

// Indexes for performance

userSchema.index({ username: 1 });

userSchema.index({ vauth\_device\_ID: 1 });

userSchema.index({ created\_at: -1 });

// Static method to hash password

userSchema.statics.hashPassword = async function(password) {

const saltRounds = 12;

return await bcrypt.hash(password, saltRounds);

};

// Instance method to verify password

userSchema.methods.verifyPassword = async function(password) {

return await bcrypt.compare(password, this.password\_hash);

};

// Static method to create user with encrypted PII

userSchema.statics.createUser = async function(userData) {

const { name, email, mobile, operating\_country, username, password, vauth\_device\_ID } = userData;

// Hash password

const password\_hash = await this.hashPassword(password);

// Encrypt PII data

const name\_enc = encrypt(name);

const email\_enc = encrypt(email);

const mobile\_enc = encrypt(mobile);

const operating\_country\_enc = encrypt(operating\_country);

return await this.create({

username,

password\_hash,

vauth\_device\_ID,

name\_enc,

email\_enc,

mobile\_enc,

operating\_country\_enc

});

};

// Instance method to get decrypted user data

userSchema.methods.getDecryptedData = function() {

return {

username: this.username,

vauth\_device\_ID: this.vauth\_device\_ID,

name: decrypt(this.name\_enc),

email: decrypt(this.email\_enc),

mobile: decrypt(this.mobile\_enc),

operating\_country: decrypt(this.operating\_country\_enc),

created\_at: this.created\_at

};

};

// Virtual for getting decrypted name (for display purposes)

userSchema.virtual('name').get(function() {

return decrypt(this.name\_enc);

});

module.exports = mongoose.model('User', userSchema);

//routes/[adminroutes.js](http://adminroutes.js)

const express = require('express');

const { body, validationResult } = require('express-validator');

const User = require('../models/User');

const Token = require('../models/Token');

const Session = require('../models/Session');

const { autoDeleteExpiredTokens } = require('../middleware/cleanup');

const router = express.Router();

// Auto-delete expired tokens setting

let autoDeleteEnabled = false;

/\*\*

\* Admin authentication middleware

\*/

function requireAdminAuth(req, res, next) {

if (!req.session.isAdmin) {

return res.status(401).json({

success: false,

message: 'Admin authentication required'

});

}

next();

}

/\*\*

\* POST /api/admin/login

\* Admin login with credentials from .env

\*/

router.post('/login', [

body('username').trim().isLength({ min: 1 }).withMessage('Username is required'),

body('password').isLength({ min: 1 }).withMessage('Password is required')

], async (req, res) => {

try {

const errors = validationResult(req);

if (!errors.isEmpty()) {

return res.status(400).json({

success: false,

message: 'Validation failed',

errors: errors.array()

});

}

const { username, password } = req.body;

// Check against environment variables

if (username === process.env.ADMIN\_USERNAME && password === process.env.ADMIN\_PASSWORD) {

req.session.isAdmin = true;

req.session.adminUsername = username;

res.json({

success: true,

message: 'Admin login successful'

});

} else {

res.status(401).json({

success: false,

message: 'Invalid admin credentials'

});

}

} catch (error) {

console.error('Admin login error:', error);

res.status(500).json({

success: false,

message: 'Internal server error'

});

}

});

/\*\*

\* POST /api/admin/logout

\* Admin logout

\*/

router.post('/logout', (req, res) => {

req.session.destroy((err) => {

if (err) {

console.error('Admin session destruction error:', err);

}

res.json({

success: true,

message: 'Admin logged out successfully'

});

});

});

/\*\*

\* POST /api/admin/add-user

\* Add new user with encrypted PII

\*/

router.post('/add-user', requireAdminAuth, [

body('name').trim().isLength({ min: 1 }).withMessage('Name is required'),

body('email').isEmail().withMessage('Valid email is required'),

body('mobile').trim().isLength({ min: 10 }).withMessage('Valid mobile number is required'),

body('operating\_country').trim().isLength({ min: 1 }).withMessage('Operating country is required')

], async (req, res) => {

try {

const errors = validationResult(req);

if (!errors.isEmpty()) {

return res.status(400).json({

success: false,

message: 'Validation failed',

errors: errors.array()

});

}

const { name, email, mobile, operating\_country } = req.body;

// Generate credentials

const username = generateUsername(name);

const password = generatePassword();

const vauth\_device\_ID = generateDeviceID();

// Check for uniqueness

const existingUser = await User.findOne({

$or: [

{ username },

{ vauth\_device\_ID }

]

});

if (existingUser) {

return res.status(400).json({

success: false,

message: 'Generated credentials conflict with existing user. Please try again.'

});

}

// Create user with encrypted PII

const user = await User.createUser({

name,

email,

mobile,

operating\_country,

username,

password,

vauth\_device\_ID

});

res.json({

success: true,

message: 'User created successfully',

credentials: {

username,

password,

vauth\_device\_ID

}

});

} catch (error) {

console.error('Add user error:', error);

res.status(500).json({

success: false,

message: 'Internal server error'

});

}

});

/\*\*

\* GET /api/admin/users

\* Get list of all users

\*/

router.get('/users', requireAdminAuth, async (req, res) => {

try {

const users = await User.find({}, {

password\_hash: 0 // Exclude password hash

}).sort({ created\_at: -1 });

const userList = users.map(user => user.getDecryptedData());

res.json({

success: true,

users: userList

});

} catch (error) {

console.error('Get users error:', error);

res.status(500).json({

success: false,

message: 'Internal server error'

});

}

});

/\*\*

\* GET /api/admin/tokens

\* Get list of all tokens with status and expiry

\*/

router.get('/tokens', requireAdminAuth, async (req, res) => {

try {

const tokens = await Token.find({}).sort({ created\_at: -1 });

const tokenList = tokens.map(token => ({

\_id: token.\_id,

device\_id: token.device\_id,

status: token.status,

created\_at: token.created\_at,

expires\_at: token.expires\_at,

used\_at: token.used\_at,

timeRemaining: token.timeRemaining

}));

res.json({

success: true,

tokens: tokenList,

autoDeleteEnabled

});

} catch (error) {

console.error('Get tokens error:', error);

res.status(500).json({

success: false,

message: 'Internal server error'

});

}

});

/\*\*

\* DELETE /api/admin/tokens/:tokenId

\* Delete specific token

\*/

router.delete('/tokens/:tokenId', requireAdminAuth, async (req, res) => {

try {

const { tokenId } = req.params;

const result = await Token.findByIdAndDelete(tokenId);

if (!result) {

return res.status(404).json({

success: false,

message: 'Token not found'

});

}

res.json({

success: true,

message: 'Token deleted successfully'

});

} catch (error) {

console.error('Delete token error:', error);

res.status(500).json({

success: false,

message: 'Internal server error'

});

}

});

/\*\*

\* PATCH /api/admin/auto-delete-expired

\* Toggle auto-delete expired tokens

\*/

router.patch('/auto-delete-expired', requireAdminAuth, async (req, res) => {

try {

autoDeleteEnabled = !autoDeleteEnabled;

if (autoDeleteEnabled) {

// Run auto-delete immediately

const deletedCount = await autoDeleteExpiredTokens();

res.json({

success: true,

message: `Auto-delete enabled. Deleted ${deletedCount} expired tokens.`,

autoDeleteEnabled

});

} else {

res.json({

success: true,

message: 'Auto-delete disabled.',

autoDeleteEnabled

});

}

} catch (error) {

console.error('Toggle auto-delete error:', error);

res.status(500).json({

success: false,

message: 'Internal server error'

});

}

});

/\*\*

\* GET /api/admin/sessions

\* Get list of active sessions

\*/

router.get('/sessions', requireAdminAuth, async (req, res) => {

try {

const sessions = await Session.getActiveSessions();

const sessionList = sessions.map(session => ({

\_id: session.\_id,

username: session.username,

device\_id: session.device\_id,

ip: session.ip,

started\_at: session.started\_at,

expires\_at: session.expires\_at,

status: session.status,

timeRemaining: session.timeRemaining,

duration: session.duration

}));

res.json({

success: true,

sessions: sessionList

});

} catch (error) {

console.error('Get sessions error:', error);

res.status(500).json({

success: false,

message: 'Internal server error'

});

}

});

/\*\*

\* POST /api/admin/sessions/:sessionId/force-logout

\* Force logout specific session

\*/

router.post('/sessions/:sessionId/force-logout', requireAdminAuth, async (req, res) => {

try {

const { sessionId } = req.params;

const success = await Session.forceLogout(sessionId);

if (!success) {

return res.status(404).json({

success: false,

message: 'Session not found or already inactive'

});

}

// Emit real-time event

req.io.to('admin').emit('session-force-logout', {

sessionId,

timestamp: new Date()

});

res.json({

success: true,

message: 'Session force logout successful'

});

} catch (error) {

console.error('Force logout error:', error);

res.status(500).json({

success: false,

message: 'Internal server error'

});

}

});

/\*\*

\* POST /api/admin/virtual-device/generate-token

\* Generate token for virtual device

\*/

router.post('/virtual-device/generate-token', requireAdminAuth, [

body('device\_id').trim().isLength({ min: 1 }).withMessage('Device ID is required')

], async (req, res) => {

try {

const errors = validationResult(req);

if (!errors.isEmpty()) {

return res.status(400).json({

success: false,

message: 'Validation failed',

errors: errors.array()

});

}

const { device\_id } = req.body;

// Check if device exists

const user = await User.findOne({ vauth\_device\_ID: device\_id });

if (!user) {

return res.status(404).json({

success: false,

message: 'Device ID not found'

});

}

// Generate token

const expirySeconds = parseInt(process.env.TOKEN\_EXPIRY\_SECONDS) || 300;

const tokenResult = await Token.createToken(device\_id, expirySeconds);

// Emit real-time event

req.io.to('admin').emit('virtual-token-generated', {

device\_id,

token: tokenResult.plain\_token,

expires\_at: tokenResult.token.expires\_at,

timestamp: new Date()

});

res.json({

success: true,

message: 'Token generated successfully',

token: tokenResult.plain\_token,

expires\_at: tokenResult.token.expires\_at

});

} catch (error) {

console.error('Generate virtual token error:', error);

res.status(500).json({

success: false,

message: 'Internal server error'

});

}

});

/\*\*

\* GET /api/admin/dashboard-stats

\* Get dashboard statistics

\*/

router.get('/dashboard-stats', requireAdminAuth, async (req, res) => {

try {

const totalUsers = await User.countDocuments();

const activeTokens = await Token.countDocuments({ status: 'ACTIVE' });

const activeSessions = await Session.countDocuments({ status: 'ACTIVE' });

const expiredTokens = await Token.countDocuments({ status: 'EXPIRED' });

res.json({

success: true,

stats: {

totalUsers,

activeTokens,

activeSessions,

expiredTokens,

autoDeleteEnabled

}

});

} catch (error) {

console.error('Get dashboard stats error:', error);

res.status(500).json({

success: false,

message: 'Internal server error'

});

}

});

/\*\*

\* Helper functions for generating credentials

\*/

function generateUsername(name) {

const cleanName = name.toLowerCase().replace(/[^a-z]/g, '');

const randomNum = Math.floor(Math.random() \* 9999).toString().padStart(4, '0');

return `${cleanName}${randomNum}`;

}

function generatePassword() {

const chars = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789!@#$%^&\*';

let password = '';

for (let i = 0; i < 12; i++) {

password += chars.charAt(Math.floor(Math.random() \* chars.length));

}

return password;

}

function generateDeviceID() {

const chars = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789';

let deviceId = 'VAUTH-';

for (let i = 0; i < 8; i++) {

deviceId += chars.charAt(Math.floor(Math.random() \* chars.length));

}

return deviceId;

}

// Auto-delete expired tokens if enabled (runs every 10 minutes)

setInterval(async () => {

if (autoDeleteEnabled) {

try {

await autoDeleteExpiredTokens();

} catch (error) {

console.error('Auto-delete interval error:', error);

}

}

}, 10 \* 60 \* 1000);

module.exports = router;

//routes/[userRoutes.js](http://userroutes.js)

const express = require('express');

const { body, validationResult } = require('express-validator');

const User = require('../models/User');

const Token = require('../models/Token');

const Session = require('../models/Session');

const router = express.Router();

// Track failed login attempts for security alerts

const failedAttempts = new Map();

/\*\*

\* POST /api/user/login

\* User login with username and password

\*/

router.post('/login', [

body('username').trim().isLength({ min: 3 }).withMessage('Username must be at least 3 characters'),

body('password').isLength({ min: 1 }).withMessage('Password is required')

], async (req, res) => {

try {

// Check validation errors

const errors = validationResult(req);

if (!errors.isEmpty()) {

return res.status(400).json({

success: false,

message: 'Validation failed',

errors: errors.array()

});

}

const { username, password } = req.body;

const clientIP = req.ip || req.connection.remoteAddress;

// Find user by username

const user = await User.findOne({ username });

if (!user) {

// Track failed attempt

trackFailedAttempt(username, clientIP, 'USER\_NOT\_FOUND', req.io);

return res.status(401).json({

success: false,

message: 'User not exists'

});

}

// Verify password

const isValidPassword = await user.verifyPassword(password);

if (!isValidPassword) {

// Track failed attempt

trackFailedAttempt(username, clientIP, 'INVALID\_PASSWORD', req.io);

return res.status(401).json({

success: false,

message: 'Invalid credentials'

});

}

// Clear failed attempts on successful login

failedAttempts.delete(username);

// Emit real-time event for admin dashboard

req.io.to('admin').emit('user-login-attempt', {

username,

ip: clientIP,

success: true,

timestamp: new Date()

});

res.json({

success: true,

message: 'Login successful',

vauth\_device\_ID: user.vauth\_device\_ID

});

} catch (error) {

console.error('Login error:', error);

res.status(500).json({

success: false,

message: 'Internal server error'

});

}

});

/\*\*

\* POST /api/user/verify-token

\* Verify 2FA token and create session

\*/

router.post('/verify-token', [

body('device\_id').trim().isLength({ min: 1 }).withMessage('Device ID is required'),

body('token').trim().isLength({ min: 6, max: 6 }).withMessage('Token must be 6 characters')

], async (req, res) => {

try {

// Check validation errors

const errors = validationResult(req);

if (!errors.isEmpty()) {

return res.status(400).json({

success: false,

message: 'Validation failed',

errors: errors.array()

});

}

const { device\_id, token } = req.body;

const clientIP = req.ip || req.connection.remoteAddress;

// Find user by device ID

const user = await User.findOne({ vauth\_device\_ID: device\_id });

if (!user) {

trackTokenFailure(device\_id, clientIP, 'DEVICE\_NOT\_FOUND', req.io);

return res.status(401).json({

success: false,

message: 'Invalid device'

});

}

// Verify token

const tokenResult = await Token.verifyToken(device\_id, token);

if (!tokenResult.valid) {

let message = 'Invalid token';

if (tokenResult.reason === 'TOKEN\_EXPIRED') {

message = 'Token expired';

} else if (tokenResult.reason === 'TOKEN\_NOT\_FOUND') {

message = 'Token invalid';

}

trackTokenFailure(device\_id, clientIP, tokenResult.reason, req.io, user.username);

return res.status(401).json({

success: false,

message: message

});

}

// Create session

const expiryMinutes = parseInt(process.env.SESSION\_EXPIRY\_MINUTES) || 10;

const session = await Session.createSession(user.username, device\_id, clientIP, expiryMinutes);

// Store session in express session

req.session.userId = user.\_id;

req.session.username = user.username;

req.session.sessionId = session.\_id;

// Emit real-time events

req.io.to('admin').emit('token-verification', {

username: user.username,

device\_id,

ip: clientIP,

success: true,

timestamp: new Date()

});

req.io.to('admin').emit('session-created', {

username: user.username,

device\_id,

ip: clientIP,

sessionId: session.\_id,

timestamp: new Date()

});

res.json({

success: true,

message: 'Token verified successfully',

sessionId: session.\_id

});

} catch (error) {

console.error('Token verification error:', error);

res.status(500).json({

success: false,

message: 'Internal server error'

});

}

});

/\*\*

\* POST /api/user/logout

\* End user session

\*/

router.post('/logout', async (req, res) => {

try {

if (req.session.sessionId) {

// Mark session as expired

await Session.findByIdAndUpdate(req.session.sessionId, {

status: 'EXPIRED'

});

// Emit real-time event

req.io.to('admin').emit('user-logout', {

username: req.session.username,

sessionId: req.session.sessionId,

timestamp: new Date()

});

}

// Destroy express session

req.session.destroy((err) => {

if (err) {

console.error('Session destruction error:', err);

}

});

res.json({

success: true,

message: 'Logged out successfully'

});

} catch (error) {

console.error('Logout error:', error);

res.status(500).json({

success: false,

message: 'Internal server error'

});

}

});

/\*\*

\* GET /api/user/session-status

\* Check current session status

\*/

router.get('/session-status', async (req, res) => {

try {

if (!req.session.sessionId) {

return res.json({

success: false,

authenticated: false,

message: 'No active session'

});

}

const sessionResult = await Session.validateSession(req.session.sessionId);

if (!sessionResult.valid) {

// Clear invalid session

req.session.destroy();

return res.json({

success: false,

authenticated: false,

message: sessionResult.reason

});

}

res.json({

success: true,

authenticated: true,

username: req.session.username,

timeRemaining: sessionResult.session.timeRemaining

});

} catch (error) {

console.error('Session status error:', error);

res.status(500).json({

success: false,

message: 'Internal server error'

});

}

});

/\*\*

\* Track failed login attempts

\*/

function trackFailedAttempt(username, ip, reason, io) {

const key = username;

const attempts = failedAttempts.get(key) || [];

attempts.push({

ip,

reason,

timestamp: new Date()

});

failedAttempts.set(key, attempts);

// Check for multiple failed attempts

if (attempts.length >= 3) {

// Send alert to admin

io.to('admin').emit('security-alert', {

type: 'MULTIPLE\_FAILED\_LOGINS',

username,

attempts: attempts.length,

lastIP: ip,

timestamp: new Date()

});

}

// Emit failed login event

io.to('admin').emit('user-login-attempt', {

username,

ip,

success: false,

reason,

timestamp: new Date()

});

}

/\*\*

\* Track failed token verification attempts

\*/

function trackTokenFailure(device\_id, ip, reason, io, username = null) {

if (username) {

const key = username;

const attempts = failedAttempts.get(key) || [];

attempts.push({

ip,

reason: 'INVALID\_TOKEN',

timestamp: new Date()

});

failedAttempts.set(key, attempts);

// Check for multiple failed token attempts

if (attempts.length >= 3) {

io.to('admin').emit('security-alert', {

type: 'MULTIPLE\_FAILED\_TOKENS',

username,

attempts: attempts.length,

lastIP: ip,

timestamp: new Date()

});

}

}

// Emit failed token verification event

io.to('admin').emit('token-verification', {

username,

device\_id,

ip,

success: false,

reason,

timestamp: new Date()

});

}

module.exports = router;

//utils/[encryption.js](http://encryption.js)

const crypto = require('crypto');

require('dotenv').config();

const ALGORITHM = 'aes-256-gcm';

const KEY = Buffer.from(process.env.ENCRYPTION\_KEY, 'utf8').slice(0, 32); // Ensure 32 bytes

/\*\*

\* Encrypt text using AES-256-GCM

\* @param {string} text - Text to encrypt

\* @returns {string} - Encrypted text with IV and auth tag (base64 encoded)

\*/

function encrypt(text) {

try {

if (!text) return '';

const iv = crypto.randomBytes(16); // 128-bit IV

const cipher = crypto.createCipher('aes-256-cbc', KEY); // Use CBC instead of GCM for compatibility

let encrypted = cipher.update(text, 'utf8', 'hex');

encrypted += cipher.final('hex');

// Combine IV and encrypted data

const combined = Buffer.concat([

iv,

Buffer.from(encrypted, 'hex')

]);

return combined.toString('base64');

} catch (error) {

console.error('Encryption error:', error);

throw new Error('Failed to encrypt data');

}

}

/\*\*

\* Decrypt text using AES-256-CBC

\* @param {string} encryptedData - Encrypted text (base64 encoded)

\* @returns {string} - Decrypted text

\*/

function decrypt(encryptedData) {

try {

if (!encryptedData) return '';

const combined = Buffer.from(encryptedData, 'base64');

// Extract IV and encrypted data

const iv = combined.slice(0, 16);

const encrypted = combined.slice(16);

const decipher = crypto.createDecipher('aes-256-cbc', KEY);

let decrypted = decipher.update(encrypted, null, 'utf8');

decrypted += decipher.final('utf8');

return decrypted;

} catch (error) {

console.error('Decryption error:', error);

throw new Error('Failed to decrypt data');

}

}

/\*\*

\* Generate a random encryption key

\* @returns {string} - Random 32-byte key in hex format

\*/

function generateKey() {

return crypto.randomBytes(32).toString('hex');

}

module.exports = {

encrypt,

decrypt,

generateKey

};

//[server.js](http://server.js)

const express = require('express');

const http = require('http');

const socketIo = require('socket.io');

const path = require('path');

const session = require('express-session');

const MongoStore = require('connect-mongo');

const helmet = require('helmet');

const cors = require('cors');

const rateLimit = require('express-rate-limit');

require('dotenv').config();

// Import database connection

const connectDB = require('./config/database');

// Import routes

const userRoutes = require('./routes/userRoutes');

const adminRoutes = require('./routes/adminRoutes');

// Import middleware

const { cleanupExpiredData } = require('./middleware/cleanup');

const app = express();

const server = http.createServer(app);

const io = socketIo(server, {

cors: {

origin: "\*",

methods: ["GET", "POST"]

}

});

// Connect to database

connectDB();

// Security middleware

app.use(helmet({

contentSecurityPolicy: false, // Disable for development

crossOriginEmbedderPolicy: false

}));

app.use(cors({

origin: true,

credentials: true

}));

// Rate limiting

const loginLimiter = rateLimit({

windowMs: parseInt(process.env.LOGIN\_RATE\_LIMIT\_WINDOW\_MS) || 900000, // 15 minutes

max: parseInt(process.env.LOGIN\_RATE\_LIMIT\_MAX\_ATTEMPTS) || 5,

message: 'Too many login attempts, please try again later.',

standardHeaders: true,

legacyHeaders: false,

});

const tokenLimiter = rateLimit({

windowMs: parseInt(process.env.TOKEN\_RATE\_LIMIT\_WINDOW\_MS) || 300000, // 5 minutes

max: parseInt(process.env.TOKEN\_RATE\_LIMIT\_MAX\_ATTEMPTS) || 10,

message: 'Too many token verification attempts, please try again later.',

standardHeaders: true,

legacyHeaders: false,

});

// Body parsing middleware

app.use(express.json({ limit: '10mb' }));

app.use(express.urlencoded({ extended: true, limit: '10mb' }));

// Session configuration

app.use(session({

secret: process.env.SESSION\_SECRET,

resave: false,

saveUninitialized: false,

store: MongoStore.create({

mongoUrl: process.env.MONGODB\_URI,

touchAfter: 24 \* 3600 // lazy session update

}),

cookie: {

secure: process.env.NODE\_ENV === 'production',

httpOnly: true,

maxAge: 1000 \* 60 \* 60 \* 24 // 24 hours

},

name: 'vauth.sid'

}));

// Static files

app.use(express.static(path.join(\_\_dirname, 'public')));

// Make io available to routes

app.use((req, res, next) => {

req.io = io;

next();

});

// Apply rate limiting to specific routes

app.use('/api/user/login', loginLimiter);

app.use('/api/user/verify-token', tokenLimiter);

app.use('/api/admin/login', loginLimiter);

// Routes

app.use('/api/user', userRoutes);

app.use('/api/admin', adminRoutes);

// Serve main pages

app.get('/', (req, res) => {

res.sendFile(path.join(\_\_dirname, 'public', 'index.html'));

});

app.get('/login', (req, res) => {

res.sendFile(path.join(\_\_dirname, 'public', 'login.html'));

});

app.get('/2fa', (req, res) => {

res.sendFile(path.join(\_\_dirname, 'public', '2fa.html'));

});

app.get('/dashboard', (req, res) => {

res.sendFile(path.join(\_\_dirname, 'public', 'dashboard.html'));

});

app.get('/admin/login', (req, res) => {

res.sendFile(path.join(\_\_dirname, 'public', 'admin-login.html'));

});

app.get('/admin', (req, res) => {

res.sendFile(path.join(\_\_dirname, 'public', 'admin-dashboard.html'));

});

// Socket.io connection handling

io.on('connection', (socket) => {

console.log('Client connected:', socket.id);

socket.on('join-admin', () => {

socket.join('admin');

console.log('Admin joined real-time monitoring');

});

socket.on('disconnect', () => {

console.log('Client disconnected:', socket.id);

});

});

// Cleanup expired data every 5 minutes

setInterval(cleanupExpiredData, 5 \* 60 \* 1000);

// Error handling middleware

app.use((err, req, res, next) => {

console.error('Error:', err);

res.status(500).json({

success: false,

message: 'Internal server error'

});

});

// 404 handler

app.use((req, res) => {

res.status(404).json({

success: false,

message: 'Route not found'

});

});

const PORT = process.env.PORT || 3000;

server.listen(PORT, '0.0.0.0', () => {

console.log(`VAUTH Server running on port ${PORT}`);

console.log(`Environment: ${process.env.NODE\_ENV}`);

console.log(`Access the application at: http://localhost:${PORT}`);

});

module.exports = { app, io };

//public/2fa.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>2FA Verification - VAUTH</title>

<script src="https://cdn.tailwindcss.com"></script>

<script>

tailwind.config = {

theme: {

extend: {

colors: {

primary: '#3B82F6',

secondary: '#1E40AF',

accent: '#F59E0B',

dark: '#1F2937',

light: '#F9FAFB'

}

}

}

}

</script>

<style>

.gradient-bg {

background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);

}

.token-input {

transition: all 0.3s ease;

letter-spacing: 0.5em;

text-align: center;

font-family: 'Courier New', monospace;

}

.token-input:focus {

transform: translateY(-2px);

box-shadow: 0 10px 25px -5px rgba(59, 130, 246, 0.25);

}

.pulse-animation {

animation: pulse 2s infinite;

}

@keyframes pulse {

0%, 100% { opacity: 1; }

50% { opacity: 0.5; }

}

</style>

</head>

<body class="bg-gray-50 min-h-screen">

<!-- Navigation -->

<nav class="bg-white shadow-lg">

<div class="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">

<div class="flex justify-between h-16">

<div class="flex items-center">

<a href="/" class="flex-shrink-0">

<h1 class="text-2xl font-bold text-primary">VAUTH</h1>

</a>

</div>

<div class="flex items-center space-x-4">

<a href="/" class="text-gray-600 hover:text-primary transition duration-300">Home</a>

<a href="/login" class="text-gray-600 hover:text-primary transition duration-300">Back to Login</a>

</div>

</div>

</div>

</nav>

<!-- Main Content -->

<div class="min-h-screen flex items-center justify-center py-12 px-4 sm:px-6 lg:px-8">

<div class="max-w-md w-full space-y-8">

<!-- Header -->

<div class="text-center">

<div class="mx-auto h-12 w-12 bg-accent rounded-full flex items-center justify-center">

<svg class="h-6 w-6 text-white" fill="none" stroke="currentColor" viewBox="0 0 24 24">

<path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M12 15v2m-6 4h12a2 2 0 002-2v-6a2 2 0 00-2-2H6a2 2 0 00-2 2v6a2 2 0 002 2zm10-10V7a4 4 0 00-8 0v4h8z"></path>

</svg>

</div>

<h2 class="mt-6 text-3xl font-extrabold text-gray-900">

Two-Factor Authentication

</h2>

<p class="mt-2 text-sm text-gray-600">

Enter the 6-character token from your VAUTH device

</p>

<div class="mt-4 bg-blue-50 border border-blue-200 rounded-lg p-3">

<p class="text-sm text-blue-700">

<span class="font-medium">Device ID:</span>

<span id="deviceIdDisplay" class="font-mono">Loading...</span>

</p>

</div>

</div>

<!-- 2FA Form -->

<div class="bg-white py-8 px-6 shadow-xl rounded-xl">

<form id="tokenForm" class="space-y-6">

<!-- Token Field -->

<div>

<label for="token" class="block text-sm font-medium text-gray-700 mb-2">

VAUTH Token

</label>

<input

id="token"

name="token"

type="text"

maxlength="6"

required

class="token-input appearance-none relative block w-full px-3 py-4 border border-gray-300 placeholder-gray-500 text-gray-900 rounded-lg focus:outline-none focus:ring-primary focus:border-primary focus:z-10 text-2xl font-bold"

placeholder="000000"

autocomplete="off"

>

<p class="mt-2 text-xs text-gray-500">

Enter the 6-character code displayed on your VAUTH device

</p>

</div>

<!-- Error Message -->

<div id="errorMessage" class="hidden bg-red-50 border border-red-200 text-red-700 px-4 py-3 rounded-lg">

<div class="flex">

<svg class="h-5 w-5 text-red-400 mr-2" fill="currentColor" viewBox="0 0 20 20">

<path fill-rule="evenodd" d="M10 18a8 8 0 100-16 8 8 0 000 16zM8.707 7.293a1 1 0 00-1.414 1.414L8.586 10l-1.293 1.293a1 1 0 101.414 1.414L10 11.414l1.293 1.293a1 1 0 001.414-1.414L11.414 10l1.293-1.293a1 1 0 00-1.414-1.414L10 8.586 8.707 7.293z" clip-rule="evenodd"></path>

</svg>

<span id="errorText"></span>

</div>

</div>

<!-- Success Message -->

<div id="successMessage" class="hidden bg-green-50 border border-green-200 text-green-700 px-4 py-3 rounded-lg">

<div class="flex">

<svg class="h-5 w-5 text-green-400 mr-2" fill="currentColor" viewBox="0 0 20 20">

<path fill-rule="evenodd" d="M10 18a8 8 0 100-16 8 8 0 000 16zm3.707-9.293a1 1 0 00-1.414-1.414L9 10.586 7.707 9.293a1 1 0 00-1.414 1.414l2 2a1 1 0 001.414 0l4-4z" clip-rule="evenodd"></path>

</svg>

<span>Token verified successfully! Redirecting...</span>

</div>

</div>

<!-- Submit Button -->

<div>

<button

type="submit"

id="verifyButton"

class="group relative w-full flex justify-center py-3 px-4 border border-transparent text-sm font-medium rounded-lg text-white bg-accent hover:bg-yellow-600 focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-accent transition duration-300"

>

<span class="absolute left-0 inset-y-0 flex items-center pl-3">

<svg class="h-5 w-5 text-yellow-300 group-hover:text-yellow-200" fill="currentColor" viewBox="0 0 20 20">

<path fill-rule="evenodd" d="M5 9V7a5 5 0 0110 0v2a2 2 0 012 2v5a2 2 0 01-2 2H5a2 2 0 01-2-2v-5a2 2 0 012-2zm8-2v2H7V7a3 3 0 016 0z" clip-rule="evenodd"></path>

</svg>

</span>

<span id="verifyButtonText">Verify Token</span>

</button>

</div>

</form>

<!-- Additional Info -->

<div class="mt-6 text-center">

<p class="text-sm text-gray-600">

Having trouble?

<a href="/login" class="font-medium text-primary hover:text-secondary transition duration-300">

Go back to login

</a>

</p>

</div>

</div>

<!-- Device Info -->

<div class="bg-yellow-50 border border-yellow-200 rounded-lg p-4">

<div class="flex">

<svg class="h-5 w-5 text-yellow-400 mr-2" fill="currentColor" viewBox="0 0 20 20">

<path fill-rule="evenodd" d="M8.257 3.099c.765-1.36 2.722-1.36 3.486 0l5.58 9.92c.75 1.334-.213 2.98-1.742 2.98H4.42c-1.53 0-2.493-1.646-1.743-2.98l5.58-9.92zM11 13a1 1 0 11-2 0 1 1 0 012 0zm-1-8a1 1 0 00-1 1v3a1 1 0 002 0V6a1 1 0 00-1-1z" clip-rule="evenodd"></path>

</svg>

<div class="text-sm text-yellow-700">

<p class="font-medium">Token Expiry Notice</p>

<p class="mt-1">VAUTH tokens expire after 5 minutes. Generate a new token if yours has expired.</p>

</div>

</div>

</div>

</div>

</div>

<script>

// Check if we have device ID from login

window.addEventListener('load', function() {

const deviceId = sessionStorage.getItem('vauth\_device\_id');

const username = sessionStorage.getItem('username');

if (!deviceId || !username) {

// Redirect to login if no device ID

window.location.href = '/login';

return;

}

// Display device ID

document.getElementById('deviceIdDisplay').textContent = deviceId;

});

// Auto-format token input

document.getElementById('token').addEventListener('input', function(e) {

let value = e.target.value.toUpperCase().replace(/[^A-Z0-9]/g, '');

if (value.length > 6) {

value = value.substring(0, 6);

}

e.target.value = value;

// Auto-submit when 6 characters entered

if (value.length === 6) {

setTimeout(() => {

document.getElementById('tokenForm').dispatchEvent(new Event('submit'));

}, 500);

}

});

document.getElementById('tokenForm').addEventListener('submit', async function(e) {

e.preventDefault();

const token = document.getElementById('token').value;

const deviceId = sessionStorage.getItem('vauth\_device\_id');

const errorDiv = document.getElementById('errorMessage');

const successDiv = document.getElementById('successMessage');

const errorText = document.getElementById('errorText');

const verifyButton = document.getElementById('verifyButton');

const verifyButtonText = document.getElementById('verifyButtonText');

if (!deviceId) {

window.location.href = '/login';

return;

}

if (token.length !== 6) {

errorText.textContent = 'Token must be 6 characters';

errorDiv.classList.remove('hidden');

successDiv.classList.add('hidden');

return;

}

// Hide previous messages

errorDiv.classList.add('hidden');

successDiv.classList.add('hidden');

// Show loading state

verifyButton.disabled = true;

verifyButtonText.textContent = 'Verifying...';

verifyButton.classList.add('opacity-75');

try {

const response = await fetch('/api/user/verify-token', {

method: 'POST',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify({

device\_id: deviceId,

token: token

})

});

const data = await response.json();

if (data.success) {

// Show success message

successDiv.classList.remove('hidden');

// Clear session storage

sessionStorage.removeItem('vauth\_device\_id');

sessionStorage.removeItem('username');

// Redirect to dashboard after delay

setTimeout(() => {

window.location.href = '/dashboard';

}, 2000);

} else {

// Show error

errorText.textContent = data.message || 'Token verification failed';

errorDiv.classList.remove('hidden');

// Clear token field

document.getElementById('token').value = '';

document.getElementById('token').focus();

}

} catch (error) {

console.error('Token verification error:', error);

errorText.textContent = 'Network error. Please try again.';

errorDiv.classList.remove('hidden');

} finally {

// Reset button state

verifyButton.disabled = false;

verifyButtonText.textContent = 'Verify Token';

verifyButton.classList.remove('opacity-75');

}

});

// Focus token input on load

window.addEventListener('load', function() {

setTimeout(() => {

document.getElementById('token').focus();

}, 500);

});

</script>

</body>

</html>

//public/admin-dashboard.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Admin Dashboard - VAUTH</title>

<script src="https://cdn.tailwindcss.com"></script>

<script src="/socket.io/socket.io.js"></script>

<script>

tailwind.config = {

theme: {

extend: {

colors: {

primary: '#3B82F6',

secondary: '#1E40AF',

accent: '#F59E0B',

dark: '#1F2937',

light: '#F9FAFB',

admin: '#7C3AED'

}

}

}

}

</script>

<style>

.gradient-bg {

background: linear-gradient(135deg, #7c3aed 0%, #3b82f6 100%);

}

.sidebar-item {

transition: all 0.3s ease;

}

.sidebar-item:hover {

transform: translateX(4px);

}

.sidebar-item.active {

background: linear-gradient(135deg, #7c3aed 0%, #3b82f6 100%);

color: white;

}

.card-hover {

transition: all 0.3s ease;

}

.card-hover:hover {

transform: translateY(-2px);

box-shadow: 0 10px 25px -5px rgba(0, 0, 0, 0.1);

}

.pulse-animation {

animation: pulse 2s infinite;

}

@keyframes pulse {

0%, 100% { opacity: 1; }

50% { opacity: 0.5; }

}

.live-map {

background: radial-gradient(circle at 50% 50%, #f0f9ff 0%, #ffffff 100%);

min-height: 400px;

position: relative;

overflow: hidden;

}

.floating-icon {

position: absolute;

transition: all 0.5s ease;

}

.connection-line {

position: absolute;

height: 2px;

transform-origin: left center;

transition: all 0.5s ease;

}

</style>

</head>

<body class="bg-gray-50 min-h-screen">

<!-- Navigation -->

<nav class="bg-white shadow-lg">

<div class="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">

<div class="flex justify-between h-16">

<div class="flex items-center">

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

<h1 class="text-2xl font-bold text-admin">VAUTH Admin</h1>

</div>

</div>

<div class="flex items-center space-x-4">

<div class="text-sm text-gray-600">

Welcome, <span class="font-semibold">Administrator</span>

</div>

<button

id="logoutButton"

class="bg-red-600 hover:bg-red-700 text-white px-4 py-2 rounded-lg transition duration-300"

>

Logout

</button>

</div>

</div>

</div>

</nav>

<div class="flex">

<!-- Sidebar -->

<div class="w-64 bg-white shadow-lg min-h-screen">

<div class="p-6">

<h2 class="text-lg font-semibold text-gray-900 mb-4">Admin Panel</h2>

<nav class="space-y-2">

<a href="#" class="sidebar-item active block px-4 py-3 rounded-lg text-sm font-medium" data-section="dashboard">

<svg class="inline h-5 w-5 mr-2" fill="none" stroke="currentColor" viewBox="0 0 24 24">

<path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M9 19v-6a2 2 0 00-2-2H5a2 2 0 00-2 2v6a2 2 0 002 2h2a2 2 0 002-2zm0 0V9a2 2 0 012-2h2a2 2 0 012 2v10m-6 0a2 2 0 002 2h2a2 2 0 002-2m0 0V5a2 2 0 012-2h2a2 2 0 012 2v14a2 2 0 01-2 2h-2a2 2 0 01-2-2z"></path>

</svg>

Dashboard

</a>

<a href="#" class="sidebar-item block px-4 py-3 rounded-lg text-sm font-medium text-gray-700 hover:bg-gray-100" data-section="add-user">

<svg class="inline h-5 w-5 mr-2" fill="none" stroke="currentColor" viewBox="0 0 24 24">

<path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M18 9v3m0 0v3m0-3h3m-3 0h-3m-2-5a4 4 0 11-8 0 4 4 0 018 0zM3 20a6 6 0 0112 0v1H3v-1z"></path>

</svg>

Add New User

</a>

<a href="#" class="sidebar-item block px-4 py-3 rounded-lg text-sm font-medium text-gray-700 hover:bg-gray-100" data-section="users">

<svg class="inline h-5 w-5 mr-2" fill="none" stroke="currentColor" viewBox="0 0 24 24">

<path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M12 4.354a4 4 0 110 5.292M15 21H3v-1a6 6 0 0112 0v1zm0 0h6v-1a6 6 0 00-9-5.197m13.5-9a2.5 2.5 0 11-5 0 2.5 2.5 0 015 0z"></path>

</svg>

UserDB

</a>

<a href="#" class="sidebar-item block px-4 py-3 rounded-lg text-sm font-medium text-gray-700 hover:bg-gray-100" data-section="tokens">

<svg class="inline h-5 w-5 mr-2" fill="none" stroke="currentColor" viewBox="0 0 24 24">

<path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M15 7a2 2 0 012 2m4 0a6 6 0 01-7.743 5.743L11 17H9v2H7v2H4a1 1 0 01-1-1v-2.586a1 1 0 01.293-.707l5.964-5.964A6 6 0 1121 9z"></path>

</svg>

TokenDB

</a>

<a href="#" class="sidebar-item block px-4 py-3 rounded-lg text-sm font-medium text-gray-700 hover:bg-gray-100" data-section="sessions">

<svg class="inline h-5 w-5 mr-2" fill="none" stroke="currentColor" viewBox="0 0 24 24">

<path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M9 19v-6a2 2 0 00-2-2H5a2 2 0 00-2 2v6a2 2 0 002 2h2a2 2 0 002-2zm0 0V9a2 2 0 012-2h2a2 2 0 012 2v10m-6 0a2 2 0 002 2h2a2 2 0 002-2m0 0V5a2 2 0 012-2h2a2 2 0 012 2v14a2 2 0 01-2 2h-2a2 2 0 01-2-2z"></path>

</svg>

Session Monitor

</a>

<a href="#" class="sidebar-item block px-4 py-3 rounded-lg text-sm font-medium text-gray-700 hover:bg-gray-100" data-section="virtual-device">

<svg class="inline h-5 w-5 mr-2" fill="none" stroke="currentColor" viewBox="0 0 24 24">

<path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M12 18h.01M8 21h8a2 2 0 002-2V5a2 2 0 00-2-2H8a2 2 0 00-2 2v14a2 2 0 002 2z"></path>

</svg>

Virtual Device

</a>

</nav>

</div>

</div>

<!-- Main Content -->

<div class="flex-1 p-6">

<!-- Dashboard Section -->

<div id="dashboard-section" class="section">

<!-- Stats Cards -->

<div class="grid grid-cols-1 md:grid-cols-4 gap-6 mb-8">

<div class="card-hover bg-white p-6 rounded-lg shadow-md">

<div class="flex items-center">

<div class="p-3 rounded-full bg-blue-100">

<svg class="h-6 w-6 text-blue-600" fill="none" stroke="currentColor" viewBox="0 0 24 24">

<path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M12 4.354a4 4 0 110 5.292M15 21H3v-1a6 6 0 0112 0v1zm0 0h6v-1a6 6 0 00-9-5.197m13.5-9a2.5 2.5 0 11-5 0 2.5 2.5 0 015 0z"></path>

</svg>

</div>

<div class="ml-4">

<p class="text-sm font-medium text-gray-600">Total Users</p>

<p id="totalUsers" class="text-2xl font-semibold text-gray-900">--</p>

</div>

</div>

</div>

<div class="card-hover bg-white p-6 rounded-lg shadow-md">

<div class="flex items-center">

<div class="p-3 rounded-full bg-green-100">

<svg class="h-6 w-6 text-green-600" fill="none" stroke="currentColor" viewBox="0 0 24 24">

<path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M15 7a2 2 0 012 2m4 0a6 6 0 01-7.743 5.743L11 17H9v2H7v2H4a1 1 0 01-1-1v-2.586a1 1 0 01.293-.707l5.964-5.964A6 6 0 1121 9z"></path>

</svg>

</div>

<div class="ml-4">

<p class="text-sm font-medium text-gray-600">Active Tokens</p>

<p id="activeTokens" class="text-2xl font-semibold text-gray-900">--</p>

</div>

</div>

</div>

<div class="card-hover bg-white p-6 rounded-lg shadow-md">

<div class="flex items-center">

<div class="p-3 rounded-full bg-purple-100">

<svg class="h-6 w-6 text-purple-600" fill="none" stroke="currentColor" viewBox="0 0 24 24">

<path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M9 19v-6a2 2 0 00-2-2H5a2 2 0 00-2 2v6a2 2 0 002 2h2a2 2 0 002-2zm0 0V9a2 2 0 012-2h2a2 2 0 012 2v10m-6 0a2 2 0 002 2h2a2 2 0 002-2m0 0V5a2 2 0 012-2h2a2 2 0 012 2v14a2 2 0 01-2 2h-2a2 2 0 01-2-2z"></path>

</svg>

</div>

<div class="ml-4">

<p class="text-sm font-medium text-gray-600">Active Sessions</p>

<p id="activeSessions" class="text-2xl font-semibold text-gray-900">--</p>

</div>

</div>

</div>

<div class="card-hover bg-white p-6 rounded-lg shadow-md">

<div class="flex items-center">

<div class="p-3 rounded-full bg-red-100">

<svg class="h-6 w-6 text-red-600" fill="none" stroke="currentColor" viewBox="0 0 24 24">

<path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M12 9v2m0 4h.01m-6.938 4h13.856c1.54 0 2.502-1.667 1.732-2.5L13.732 4c-.77-.833-1.964-.833-2.732 0L3.732 16.5c-.77.833.192 2.5 1.732 2.5z"></path>

</svg>

</div>

<div class="ml-4">

<p class="text-sm font-medium text-gray-600">Expired Tokens</p>

<p id="expiredTokens" class="text-2xl font-semibold text-gray-900">--</p>

</div>

</div>

</div>

</div>

<!-- Live Request Map -->

<div class="bg-white rounded-lg shadow-md p-6 mb-8">

<div class="flex justify-between items-center mb-4">

<h3 class="text-lg font-semibold text-gray-900">Live Request Map</h3>

<div class="flex space-x-2">

<div class="flex items-center">

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

<span class="text-sm text-gray-600">Valid</span>

</div>

<div class="flex items-center">

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

<span class="text-sm text-gray-600">Expired</span>

</div>

<div class="flex items-center">

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

<span class="text-sm text-gray-600">Invalid</span>

</div>

</div>

</div>

<div id="liveMap" class="live-map border-2 border-gray-200 rounded-lg">

<!-- Dynamic content will be added here -->

</div>

</div>

<!-- Recent Activity -->

<div class="bg-white rounded-lg shadow-md p-6">

<h3 class="text-lg font-semibold text-gray-900 mb-4">Recent Activity</h3>

<div id="recentActivity" class="space-y-3">

<p class="text-gray-500 text-center py-4">No recent activity</p>

</div>

</div>

</div>

<!-- Add User Section -->

<div id="add-user-section" class="section hidden">

<div class="bg-white rounded-lg shadow-md p-6">

<h3 class="text-lg font-semibold text-gray-900 mb-6">Add New User</h3>

<form id="addUserForm" class="space-y-4">

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

<div>

<label class="block text-sm font-medium text-gray-700 mb-2">Name</label>

<input type="text" name="name" required class="w-full px-3 py-2 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-admin">

</div>

<div>

<label class="block text-sm font-medium text-gray-700 mb-2">Email</label>

<input type="email" name="email" required class="w-full px-3 py-2 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-admin">

</div>

<div>

<label class="block text-sm font-medium text-gray-700 mb-2">Mobile</label>

<input type="tel" name="mobile" required class="w-full px-3 py-2 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-admin">

</div>

<div>

<label class="block text-sm font-medium text-gray-700 mb-2">Operating Country</label>

<input type="text" name="operating\_country" required class="w-full px-3 py-2 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-admin">

</div>

</div>

<button type="submit" class="bg-admin hover:bg-purple-700 text-white px-6 py-2 rounded-lg transition duration-300">

Create User

</button>

</form>

<!-- Generated Credentials Display -->

<div id="credentialsDisplay" class="hidden mt-6 p-4 bg-green-50 border border-green-200 rounded-lg">

<h4 class="font-semibold text-green-800 mb-2">User Created Successfully!</h4>

<div class="space-y-2 text-sm">

<p><strong>Username:</strong> <span id="genUsername" class="font-mono"></span></p>

<p><strong>Password:</strong> <span id="genPassword" class="font-mono"></span></p>

<p><strong>VAUTH Device ID:</strong> <span id="genDeviceId" class="font-mono"></span></p>

</div>

<p class="text-xs text-green-700 mt-2">Please save these credentials securely. They will not be shown again.</p>

</div>

</div>

</div>

<!-- Users Section -->

<div id="users-section" class="section hidden">

<div class="bg-white rounded-lg shadow-md p-6">

<div class="flex justify-between items-center mb-6">

<h3 class="text-lg font-semibold text-gray-900">User Database</h3>

<button id="refreshUsers" class="bg-primary hover:bg-secondary text-white px-4 py-2 rounded-lg transition duration-300">

Refresh

</button>

</div>

<div class="overflow-x-auto">

<table class="min-w-full divide-y divide-gray-200">

<thead class="bg-gray-50">

<tr>

<th class="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">Name</th>

<th class="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">Username</th>

<th class="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">VAUTH Device ID</th>

<th class="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">Created</th>

</tr>

</thead>

<tbody id="usersTableBody" class="bg-white divide-y divide-gray-200">

<tr>

<td colspan="4" class="px-6 py-4 text-center text-gray-500">Loading...</td>

</tr>

</tbody>

</table>

</div>

</div>

</div>

<!-- Tokens Section -->

<div id="tokens-section" class="section hidden">

<div class="bg-white rounded-lg shadow-md p-6">

<div class="flex justify-between items-center mb-6">

<h3 class="text-lg font-semibold text-gray-900">Token Database</h3>

<div class="flex space-x-2">

<button id="toggleAutoDelete" class="bg-yellow-600 hover:bg-yellow-700 text-white px-4 py-2 rounded-lg transition duration-300">

Auto-Delete: <span id="autoDeleteStatus">OFF</span>

</button>

<button id="refreshTokens" class="bg-primary hover:bg-secondary text-white px-4 py-2 rounded-lg transition duration-300">

Refresh

</button>

</div>

</div>

<div class="overflow-x-auto">

<table class="min-w-full divide-y divide-gray-200">

<thead class="bg-gray-50">

<tr>

<th class="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">Device ID</th>

<th class="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">Status</th>

<th class="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">Time Remaining</th>

<th class="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">Created</th>

<th class="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">Actions</th>

</tr>

</thead>

<tbody id="tokensTableBody" class="bg-white divide-y divide-gray-200">

<tr>

<td colspan="5" class="px-6 py-4 text-center text-gray-500">Loading...</td>

</tr>

</tbody>

</table>

</div>

</div>

</div>

<!-- Sessions Section -->

<div id="sessions-section" class="section hidden">

<div class="bg-white rounded-lg shadow-md p-6">

<div class="flex justify-between items-center mb-6">

<h3 class="text-lg font-semibold text-gray-900">Active Sessions</h3>

<button id="refreshSessions" class="bg-primary hover:bg-secondary text-white px-4 py-2 rounded-lg transition duration-300">

Refresh

</button>

</div>

<div class="overflow-x-auto">

<table class="min-w-full divide-y divide-gray-200">

<thead class="bg-gray-50">

<tr>

<th class="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">Username</th>

<th class="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">Device ID</th>

<th class="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">IP Address</th>

<th class="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">Started</th>

<th class="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">Time Left</th>

<th class="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">Actions</th>

</tr>

</thead>

<tbody id="sessionsTableBody" class="bg-white divide-y divide-gray-200">

<tr>

<td colspan="6" class="px-6 py-4 text-center text-gray-500">Loading...</td>

</tr>

</tbody>

</table>

</div>

</div>

</div>

<!-- Virtual Device Section -->

<div id="virtual-device-section" class="section hidden">

<div class="bg-white rounded-lg shadow-md p-6">

<h3 class="text-lg font-semibold text-gray-900 mb-6">Virtual Device Token Generator</h3>

<form id="virtualTokenForm" class="space-y-4">

<div>

<label class="block text-sm font-medium text-gray-700 mb-2">VAUTH Device ID</label>

<input type="text" name="device\_id" required placeholder="Enter VAUTH Device ID" class="w-full px-3 py-2 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-admin">

</div>

<button type="submit" class="bg-admin hover:bg-purple-700 text-white px-6 py-2 rounded-lg transition duration-300">

Generate Token

</button>

</form>

<!-- Generated Token Display -->

<div id="tokenDisplay" class="hidden mt-6 p-4 bg-blue-50 border border-blue-200 rounded-lg">

<h4 class="font-semibold text-blue-800 mb-2">Token Generated Successfully!</h4>

<div class="text-center">

<p class="text-3xl font-mono font-bold text-blue-900 mb-2" id="generatedToken">------</p>

<p class="text-sm text-blue-700">Expires: <span id="tokenExpiry">--</span></p>

</div>

</div>

</div>

</div>

</div>

</div>

<!-- Security Alert Modal -->

<div id="alertModal" class="hidden fixed inset-0 bg-gray-600 bg-opacity-50 overflow-y-auto h-full w-full z-50">

<div class="relative top-20 mx-auto p-5 border w-96 shadow-lg rounded-md bg-white">

<div class="mt-3 text-center">

<div class="mx-auto flex items-center justify-center h-12 w-12 rounded-full bg-red-100">

<svg class="h-6 w-6 text-red-600" fill="none" stroke="currentColor" viewBox="0 0 24 24">

<path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M12 9v2m0 4h.01m-6.938 4h13.856c1.54 0 2.502-1.667 1.732-2.5L13.732 4c-.77-.833-1.964-.833-2.732 0L3.732 16.5c-.77.833.192 2.5 1.732 2.5z"></path>

</svg>

</div>

<h3 class="text-lg leading-6 font-medium text-gray-900 mt-2">Security Alert</h3>

<div class="mt-2 px-7 py-3">

<p id="alertMessage" class="text-sm text-gray-500"></p>

</div>

<div class="items-center px-4 py-3">

<button id="closeAlert" class="px-4 py-2 bg-red-500 text-white text-base font-medium rounded-md w-full shadow-sm hover:bg-red-700 focus:outline-none focus:ring-2 focus:ring-red-300">

Close

</button>

</div>

</div>

</div>

</div>

<script src="/js/admin-dashboard.js"></script>

</body>

</html>

//public/admin-login.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Admin Login - VAUTH</title>

<script src="https://cdn.tailwindcss.com"></script>

<script>

tailwind.config = {

theme: {

extend: {

colors: {

primary: '#3B82F6',

secondary: '#1E40AF',

accent: '#F59E0B',

dark: '#1F2937',

light: '#F9FAFB',

admin: '#7C3AED'

}

}

}

}

</script>

<style>

.gradient-bg {

background: linear-gradient(135deg, #7c3aed 0%, #3b82f6 100%);

}

.form-input {

transition: all 0.3s ease;

}

.form-input:focus {

transform: translateY(-2px);

box-shadow: 0 10px 25px -5px rgba(124, 58, 237, 0.25);

}

</style>

</head>

<body class="bg-gray-50 min-h-screen">

<!-- Navigation -->

<nav class="bg-white shadow-lg">

<div class="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">

<div class="flex justify-between h-16">

<div class="flex items-center">

<a href="/" class="flex-shrink-0">

<h1 class="text-2xl font-bold text-primary">VAUTH</h1>

</a>

</div>

<div class="flex items-center space-x-4">

<a href="/" class="text-gray-600 hover:text-primary transition duration-300">Home</a>

<a href="/login" class="bg-primary hover:bg-secondary text-white px-4 py-2 rounded-lg transition duration-300">

User Login

</a>

</div>

</div>

</div>

</nav>

<!-- Main Content -->

<div class="min-h-screen flex items-center justify-center py-12 px-4 sm:px-6 lg:px-8">

<div class="max-w-md w-full space-y-8">

<!-- Header -->

<div class="text-center">

<div class="mx-auto h-12 w-12 bg-admin rounded-full flex items-center justify-center">

<svg class="h-6 w-6 text-white" fill="none" stroke="currentColor" viewBox="0 0 24 24">

<path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M9 12l2 2 4-4m5.618-4.016A11.955 11.955 0 0112 2.944a11.955 11.955 0 01-8.618 3.04A12.02 12.02 0 003 9c0 5.591 3.824 10.29 9 11.622 5.176-1.332 9-6.03 9-11.622 0-1.042-.133-2.052-.382-3.016z"></path>

</svg>

</div>

<h2 class="mt-6 text-3xl font-extrabold text-gray-900">

Admin Access

</h2>

<p class="mt-2 text-sm text-gray-600">

Enter your administrator credentials

</p>

</div>

<!-- Admin Login Form -->

<div class="bg-white py-8 px-6 shadow-xl rounded-xl border-t-4 border-admin">

<form id="adminLoginForm" class="space-y-6">

<!-- Username Field -->

<div>

<label for="username" class="block text-sm font-medium text-gray-700 mb-2">

Admin Username

</label>

<input

id="username"

name="username"

type="text"

required

class="form-input appearance-none relative block w-full px-3 py-3 border border-gray-300 placeholder-gray-500 text-gray-900 rounded-lg focus:outline-none focus:ring-admin focus:border-admin focus:z-10 sm:text-sm"

placeholder="Enter admin username"

>

</div>

<!-- Password Field -->

<div>

<label for="password" class="block text-sm font-medium text-gray-700 mb-2">

Admin Password

</label>

<input

id="password"

name="password"

type="password"

required

class="form-input appearance-none relative block w-full px-3 py-3 border border-gray-300 placeholder-gray-500 text-gray-900 rounded-lg focus:outline-none focus:ring-admin focus:border-admin focus:z-10 sm:text-sm"

placeholder="Enter admin password"

>

</div>

<!-- Error Message -->

<div id="errorMessage" class="hidden bg-red-50 border border-red-200 text-red-700 px-4 py-3 rounded-lg">

<div class="flex">

<svg class="h-5 w-5 text-red-400 mr-2" fill="currentColor" viewBox="0 0 20 20">

<path fill-rule="evenodd" d="M10 18a8 8 0 100-16 8 8 0 000 16zM8.707 7.293a1 1 0 00-1.414 1.414L8.586 10l-1.293 1.293a1 1 0 101.414 1.414L10 11.414l1.293 1.293a1 1 0 001.414-1.414L11.414 10l1.293-1.293a1 1 0 00-1.414-1.414L10 8.586 8.707 7.293z" clip-rule="evenodd"></path>

</svg>

<span id="errorText"></span>

</div>

</div>

<!-- Submit Button -->

<div>

<button

type="submit"

id="loginButton"

class="group relative w-full flex justify-center py-3 px-4 border border-transparent text-sm font-medium rounded-lg text-white bg-admin hover:bg-purple-700 focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-admin transition duration-300"

>

<span class="absolute left-0 inset-y-0 flex items-center pl-3">

<svg class="h-5 w-5 text-purple-300 group-hover:text-purple-200" fill="currentColor" viewBox="0 0 20 20">

<path fill-rule="evenodd" d="M5 9V7a5 5 0 0110 0v2a2 2 0 012 2v5a2 2 0 01-2 2H5a2 2 0 01-2-2v-5a2 2 0 012-2zm8-2v2H7V7a3 3 0 016 0z" clip-rule="evenodd"></path>

</svg>

</span>

<span id="loginButtonText">Access Admin Panel</span>

</button>

</div>

</form>

<!-- Additional Links -->

<div class="mt-6 text-center">

<p class="text-sm text-gray-600">

Regular user?

<a href="/login" class="font-medium text-admin hover:text-purple-700 transition duration-300">

User Login

</a>

</p>

</div>

</div>

<!-- Security Notice -->

<div class="bg-purple-50 border border-purple-200 rounded-lg p-4">

<div class="flex">

<svg class="h-5 w-5 text-purple-400 mr-2" fill="currentColor" viewBox="0 0 20 20">

<path fill-rule="evenodd" d="M18 10a8 8 0 11-16 0 8 8 0 0116 0zm-7-4a1 1 0 11-2 0 1 1 0 012 0zM9 9a1 1 0 000 2v3a1 1 0 001 1h1a1 1 0 100-2v-3a1 1 0 00-1-1H9z" clip-rule="evenodd"></path>

</svg>

<div class="text-sm text-purple-700">

<p class="font-medium">Administrator Access</p>

<p class="mt-1">This area is restricted to authorized administrators only. All access attempts are logged and monitored.</p>

</div>

</div>

</div>

</div>

</div>

<script>

document.getElementById('adminLoginForm').addEventListener('submit', async function(e) {

e.preventDefault();

const username = document.getElementById('username').value;

const password = document.getElementById('password').value;

const errorDiv = document.getElementById('errorMessage');

const errorText = document.getElementById('errorText');

const loginButton = document.getElementById('loginButton');

const loginButtonText = document.getElementById('loginButtonText');

// Hide previous errors

errorDiv.classList.add('hidden');

// Show loading state

loginButton.disabled = true;

loginButtonText.textContent = 'Authenticating...';

loginButton.classList.add('opacity-75');

try {

const response = await fetch('/api/admin/login', {

method: 'POST',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify({ username, password })

});

const data = await response.json();

if (data.success) {

// Redirect to admin dashboard

window.location.href = '/admin';

} else {

// Show error

errorText.textContent = data.message || 'Admin login failed';

errorDiv.classList.remove('hidden');

}

} catch (error) {

console.error('Admin login error:', error);

errorText.textContent = 'Network error. Please try again.';

errorDiv.classList.remove('hidden');

} finally {

// Reset button state

loginButton.disabled = false;

loginButtonText.textContent = 'Access Admin Panel';

loginButton.classList.remove('opacity-75');

}

});

// Check if already logged in as admin

window.addEventListener('load', function() {

// In a real implementation, you would check admin session status

// For now, we'll just focus the username field

document.getElementById('username').focus();

});

</script>

</body>

</html>

//public/dashboard.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Dashboard - VAUTH</title>

<script src="https://cdn.tailwindcss.com"></script>

<script>

tailwind.config = {

theme: {

extend: {

colors: {

primary: '#3B82F6',

secondary: '#1E40AF',

accent: '#F59E0B',

dark: '#1F2937',

light: '#F9FAFB'

}

}

}

}

</script>

<style>

.gradient-bg {

background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);

}

.countdown-circle {

transition: stroke-dasharray 1s linear;

}

.pulse-animation {

animation: pulse 2s infinite;

}

@keyframes pulse {

0%, 100% { opacity: 1; }

50% { opacity: 0.5; }

}

</style>

</head>

<body class="bg-gray-50 min-h-screen">

<!-- Navigation -->

<nav class="bg-white shadow-lg">

<div class="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">

<div class="flex justify-between h-16">

<div class="flex items-center">

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

<h1 class="text-2xl font-bold text-primary">VAUTH</h1>

</div>

<div class="ml-6">

<span class="text-gray-600">Welcome, <span id="usernameDisplay" class="font-semibold text-gray-900">Loading...</span></span>

</div>

</div>

<div class="flex items-center space-x-4">

<!-- Session Timer -->

<div class="flex items-center space-x-2 bg-gray-100 px-3 py-2 rounded-lg">

<svg class="h-4 w-4 text-gray-600" fill="none" stroke="currentColor" viewBox="0 0 24 24">

<path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M12 8v4l3 3m6-3a9 9 0 11-18 0 9 9 0 0118 0z"></path>

</svg>

<span id="sessionTimer" class="text-sm font-medium text-gray-700">--:--</span>

</div>

<!-- Logout Button -->

<button

id="logoutButton"

class="bg-red-600 hover:bg-red-700 text-white px-4 py-2 rounded-lg transition duration-300"

>

Logout

</button>

</div>

</div>

</div>

</nav>

<!-- Main Content -->

<div class="max-w-7xl mx-auto py-6 sm:px-6 lg:px-8">

<!-- Welcome Section -->

<div class="px-4 py-6 sm:px-0">

<div class="bg-white overflow-hidden shadow-xl rounded-lg">

<div class="gradient-bg px-6 py-8 text-white">

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

<div>

<h1 class="text-3xl font-bold">Welcome to Your Dashboard</h1>

<p class="mt-2 text-lg opacity-90">You have successfully authenticated with VAUTH</p>

</div>

<div class="hidden md:block">

<svg class="h-16 w-16 opacity-75" fill="currentColor" viewBox="0 0 20 20">

<path fill-rule="evenodd" d="M6.267 3.455a3.066 3.066 0 001.745-.723 3.066 3.066 0 013.976 0 3.066 3.066 0 001.745.723 3.066 3.066 0 012.812 2.812c.051.643.304 1.254.723 1.745a3.066 3.066 0 010 3.976 3.066 3.066 0 00-.723 1.745 3.066 3.066 0 01-2.812 2.812 3.066 3.066 0 00-1.745.723 3.066 3.066 0 01-3.976 0 3.066 3.066 0 00-1.745-.723 3.066 3.066 0 01-2.812-2.812 3.066 3.066 0 00-.723-1.745 3.066 3.066 0 010-3.976 3.066 3.066 0 00.723-1.745 3.066 3.066 0 012.812-2.812zm7.44 5.252a1 1 0 00-1.414-1.414L9 10.586 7.707 9.293a1 1 0 00-1.414 1.414l2 2a1 1 0 001.414 0l4-4z" clip-rule="evenodd"></path>

</svg>

</div>

</div>

</div>

<!-- Session Info -->

<div class="px-6 py-6">

<div class="grid grid-cols-1 md:grid-cols-3 gap-6">

<div class="bg-green-50 border border-green-200 rounded-lg p-4">

<div class="flex items-center">

<svg class="h-8 w-8 text-green-500 mr-3" fill="currentColor" viewBox="0 0 20 20">

<path fill-rule="evenodd" d="M10 18a8 8 0 100-16 8 8 0 000 16zm3.707-9.293a1 1 0 00-1.414-1.414L9 10.586 7.707 9.293a1 1 0 00-1.414 1.414l2 2a1 1 0 001.414 0l4-4z" clip-rule="evenodd"></path>

</svg>

<div>

<p class="text-sm font-medium text-green-800">Authentication Status</p>

<p class="text-lg font-bold text-green-900">Verified</p>

</div>

</div>

</div>

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

<div class="flex items-center">

<svg class="h-8 w-8 text-blue-500 mr-3" fill="currentColor" viewBox="0 0 20 20">

<path fill-rule="evenodd" d="M12 8v4l3 3m6-3a9 9 0 11-18 0 9 9 0 0118 0z" clip-rule="evenodd"></path>

</svg>

<div>

<p class="text-sm font-medium text-blue-800">Session Time Remaining</p>

<p id="timeRemainingDisplay" class="text-lg font-bold text-blue-900">--:--</p>

</div>

</div>

</div>

<div class="bg-purple-50 border border-purple-200 rounded-lg p-4">

<div class="flex items-center">

<svg class="h-8 w-8 text-purple-500 mr-3" fill="currentColor" viewBox="0 0 20 20">

<path fill-rule="evenodd" d="M3 4a1 1 0 011-1h12a1 1 0 011 1v2a1 1 0 01-1 1H4a1 1 0 01-1-1V4zM3 10a1 1 0 011-1h6a1 1 0 011 1v6a1 1 0 01-1 1H4a1 1 0 01-1-1v-6zM14 9a1 1 0 00-1 1v6a1 1 0 001 1h2a1 1 0 001-1v-6a1 1 0 00-1-1h-2z" clip-rule="evenodd"></path>

</svg>

<div>

<p class="text-sm font-medium text-purple-800">Security Level</p>

<p class="text-lg font-bold text-purple-900">High</p>

</div>

</div>

</div>

</div>

</div>

</div>

</div>

<!-- Auto-Logout Warning -->

<div id="logoutWarning" class="hidden px-4 py-6 sm:px-0">

<div class="bg-yellow-50 border border-yellow-200 rounded-lg p-6">

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

<div class="flex items-center">

<svg class="h-8 w-8 text-yellow-500 mr-3 pulse-animation" fill="currentColor" viewBox="0 0 20 20">

<path fill-rule="evenodd" d="M8.257 3.099c.765-1.36 2.722-1.36 3.486 0l5.58 9.92c.75 1.334-.213 2.98-1.742 2.98H4.42c-1.53 0-2.493-1.646-1.743-2.98l5.58-9.92zM11 13a1 1 0 11-2 0 1 1 0 012 0zm-1-8a1 1 0 00-1 1v3a1 1 0 002 0V6a1 1 0 00-1-1z" clip-rule="evenodd"></path>

</svg>

<div>

<p class="text-lg font-medium text-yellow-800">Session Expiring Soon</p>

<p class="text-sm text-yellow-700">Your session will automatically expire in <span id="warningTimer" class="font-bold">--</span> seconds.</p>

</div>

</div>

<button

id="extendSessionButton"

class="bg-yellow-600 hover:bg-yellow-700 text-white px-4 py-2 rounded-lg transition duration-300"

>

Stay Logged In

</button>

</div>

</div>

</div>

<!-- Features Section -->

<div class="px-4 py-6 sm:px-0">

<div class="bg-white shadow-xl rounded-lg overflow-hidden">

<div class="px-6 py-4 border-b border-gray-200">

<h2 class="text-xl font-semibold text-gray-900">Account Features</h2>

</div>

<div class="p-6">

<div class="grid grid-cols-1 md:grid-cols-2 gap-6">

<div class="border border-gray-200 rounded-lg p-4 hover:shadow-md transition duration-300">

<div class="flex items-center mb-3">

<svg class="h-6 w-6 text-primary mr-2" fill="none" stroke="currentColor" viewBox="0 0 24 24">

<path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M9 12l2 2 4-4m5.618-4.016A11.955 11.955 0 0112 2.944a11.955 11.955 0 01-8.618 3.04A12.02 12.02 0 003 9c0 5.591 3.824 10.29 9 11.622 5.176-1.332 9-6.03 9-11.622 0-1.042-.133-2.052-.382-3.016z"></path>

</svg>

<h3 class="text-lg font-medium text-gray-900">Secure Authentication</h3>

</div>

<p class="text-gray-600">Your account is protected with two-factor authentication using VAUTH device tokens.</p>

</div>

<div class="border border-gray-200 rounded-lg p-4 hover:shadow-md transition duration-300">

<div class="flex items-center mb-3">

<svg class="h-6 w-6 text-primary mr-2" fill="none" stroke="currentColor" viewBox="0 0 24 24">

<path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M12 15v2m-6 4h12a2 2 0 002-2v-6a2 2 0 00-2-2H6a2 2 0 00-2 2v6a2 2 0 002 2zm10-10V7a4 4 0 00-8 0v4h8z"></path>

</svg>

<h3 class="text-lg font-medium text-gray-900">Encrypted Data</h3>

</div>

<p class="text-gray-600">All your personal information is encrypted using AES-256-GCM encryption.</p>

</div>

<div class="border border-gray-200 rounded-lg p-4 hover:shadow-md transition duration-300">

<div class="flex items-center mb-3">

<svg class="h-6 w-6 text-primary mr-2" fill="none" stroke="currentColor" viewBox="0 0 24 24">

<path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M12 8v4l3 3m6-3a9 9 0 11-18 0 9 9 0 0118 0z"></path>

</svg>

<h3 class="text-lg font-medium text-gray-900">Session Management</h3>

</div>

<p class="text-gray-600">Automatic session expiry and real-time monitoring for enhanced security.</p>

</div>

<div class="border border-gray-200 rounded-lg p-4 hover:shadow-md transition duration-300">

<div class="flex items-center mb-3">

<svg class="h-6 w-6 text-primary mr-2" fill="none" stroke="currentColor" viewBox="0 0 24 24">

<path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M13 10V3L4 14h7v7l9-11h-7z"></path>

</svg>

<h3 class="text-lg font-medium text-gray-900">Real-time Monitoring</h3>

</div>

<p class="text-gray-600">Your account activity is monitored in real-time for suspicious behavior.</p>

</div>

</div>

</div>

</div>

</div>

</div>

<script>

let sessionTimer = null;

let timeRemaining = 0;

let warningShown = false;

// Check authentication on load

window.addEventListener('load', async function() {

await checkSession();

startSessionTimer();

});

async function checkSession() {

try {

const response = await fetch('/api/user/session-status');

const data = await response.json();

if (!data.success || !data.authenticated) {

// Redirect to login if not authenticated

window.location.href = '/login';

return;

}

// Update UI with user info

document.getElementById('usernameDisplay').textContent = data.username;

timeRemaining = data.timeRemaining;

updateTimerDisplay();

} catch (error) {

console.error('Session check error:', error);

window.location.href = '/login';

}

}

function startSessionTimer() {

sessionTimer = setInterval(async () => {

timeRemaining--;

updateTimerDisplay();

// Show warning when 2 minutes remaining

if (timeRemaining <= 120 && !warningShown) {

showLogoutWarning();

warningShown = true;

}

// Auto-logout when time expires

if (timeRemaining <= 0) {

await logout();

}

// Update warning timer

if (warningShown) {

document.getElementById('warningTimer').textContent = timeRemaining;

}

}, 1000);

}

function updateTimerDisplay() {

const minutes = Math.floor(timeRemaining / 60);

const seconds = timeRemaining % 60;

const timeString = `${minutes.toString().padStart(2, '0')}:${seconds.toString().padStart(2, '0')}`;

document.getElementById('sessionTimer').textContent = timeString;

document.getElementById('timeRemainingDisplay').textContent = timeString;

// Change color based on time remaining

const timerElement = document.getElementById('sessionTimer');

if (timeRemaining <= 120) {

timerElement.classList.add('text-red-600');

timerElement.classList.remove('text-gray-700');

} else if (timeRemaining <= 300) {

timerElement.classList.add('text-yellow-600');

timerElement.classList.remove('text-gray-700');

}

}

function showLogoutWarning() {

document.getElementById('logoutWarning').classList.remove('hidden');

}

async function logout() {

try {

await fetch('/api/user/logout', {

method: 'POST'

});

} catch (error) {

console.error('Logout error:', error);

} finally {

window.location.href = '/login';

}

}

// Logout button handler

document.getElementById('logoutButton').addEventListener('click', logout);

// Extend session button handler

document.getElementById('extendSessionButton').addEventListener('click', async function() {

// In a real implementation, this would extend the session

// For now, we'll just refresh the session status

await checkSession();

document.getElementById('logoutWarning').classList.add('hidden');

warningShown = false;

});

// Refresh session status every 30 seconds

setInterval(checkSession, 30000);

</script>

</body>

</html>

//public/index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>VAUTH - Secure Authentication System</title>

<script src="https://cdn.tailwindcss.com"></script>

<script>

tailwind.config = {

theme: {

extend: {

colors: {

primary: '#3B82F6',

secondary: '#1E40AF',

accent: '#F59E0B',

dark: '#1F2937',

light: '#F9FAFB'

}

}

}

}

</script>

<style>

.gradient-bg {

background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);

}

.card-hover {

transition: all 0.3s ease;

}

.card-hover:hover {

transform: translateY(-5px);

box-shadow: 0 20px 25px -5px rgba(0, 0, 0, 0.1), 0 10px 10px -5px rgba(0, 0, 0, 0.04);

}

</style>

</head>

<body class="bg-gray-50 min-h-screen">

<!-- Navigation -->

<nav class="bg-white shadow-lg">

<div class="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">

<div class="flex justify-between h-16">

<div class="flex items-center">

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

<h1 class="text-2xl font-bold text-primary">VAUTH</h1>

</div>

</div>

<div class="flex items-center space-x-4">

<a href="/login" class="bg-primary hover:bg-secondary text-white px-4 py-2 rounded-lg transition duration-300">

User Login

</a>

<a href="/admin/login" class="bg-gray-600 hover:bg-gray-700 text-white px-4 py-2 rounded-lg transition duration-300">

Admin Login

</a>

</div>

</div>

</div>

</nav>

<!-- Hero Section -->

<section class="gradient-bg text-white py-20">

<div class="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8 text-center">

<h1 class="text-5xl font-bold mb-6">

Secure Authentication with VAUTH

</h1>

<p class="text-xl mb-8 max-w-3xl mx-auto">

Advanced two-factor authentication system with real-time monitoring,

encrypted data storage, and comprehensive admin controls.

</p>

<div class="flex justify-center space-x-4">

<a href="/login" class="bg-white text-primary px-8 py-3 rounded-lg font-semibold hover:bg-gray-100 transition duration-300">

Get Started

</a>

<a href="#features" class="border-2 border-white text-white px-8 py-3 rounded-lg font-semibold hover:bg-white hover:text-primary transition duration-300">

Learn More

</a>

</div>

</div>

</section>

<!-- Features Section -->

<section id="features" class="py-20 bg-white">

<div class="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">

<div class="text-center mb-16">

<h2 class="text-4xl font-bold text-gray-900 mb-4">

Why Choose VAUTH?

</h2>

<p class="text-xl text-gray-600 max-w-2xl mx-auto">

Built with security-first principles and modern technology stack

</p>

</div>

<div class="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-8">

<!-- Feature 1 -->

<div class="card-hover bg-gray-50 p-8 rounded-xl">

<div class="w-12 h-12 bg-primary rounded-lg flex items-center justify-center mb-6">

<svg class="w-6 h-6 text-white" fill="none" stroke="currentColor" viewBox="0 0 24 24">

<path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M12 15v2m-6 4h12a2 2 0 002-2v-6a2 2 0 00-2-2H6a2 2 0 00-2 2v6a2 2 0 002 2zm10-10V7a4 4 0 00-8 0v4h8z"></path>

</svg>

</div>

<h3 class="text-xl font-semibold text-gray-900 mb-4">Two-Factor Authentication</h3>

<p class="text-gray-600">

Secure login process with username/password and VAUTH device token verification.

</p>

</div>

<!-- Feature 2 -->

<div class="card-hover bg-gray-50 p-8 rounded-xl">

<div class="w-12 h-12 bg-accent rounded-lg flex items-center justify-center mb-6">

<svg class="w-6 h-6 text-white" fill="none" stroke="currentColor" viewBox="0 0 24 24">

<path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M9 12l2 2 4-4m5.618-4.016A11.955 11.955 0 0112 2.944a11.955 11.955 0 01-8.618 3.04A12.02 12.02 0 003 9c0 5.591 3.824 10.29 9 11.622 5.176-1.332 9-6.03 9-11.622 0-1.042-.133-2.052-.382-3.016z"></path>

</svg>

</div>

<h3 class="text-xl font-semibold text-gray-900 mb-4">Encrypted Data Storage</h3>

<p class="text-gray-600">

All personal information encrypted with AES-256-GCM encryption for maximum security.

</p>

</div>

<!-- Feature 3 -->

<div class="card-hover bg-gray-50 p-8 rounded-xl">

<div class="w-12 h-12 bg-secondary rounded-lg flex items-center justify-center mb-6">

<svg class="w-6 h-6 text-white" fill="none" stroke="currentColor" viewBox="0 0 24 24">

<path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M13 10V3L4 14h7v7l9-11h-7z"></path>

</svg>

</div>

<h3 class="text-xl font-semibold text-gray-900 mb-4">Real-time Monitoring</h3>

<p class="text-gray-600">

Live dashboard with real-time session monitoring and security alerts.

</p>

</div>

<!-- Feature 4 -->

<div class="card-hover bg-gray-50 p-8 rounded-xl">

<div class="w-12 h-12 bg-green-500 rounded-lg flex items-center justify-center mb-6">

<svg class="w-6 h-6 text-white" fill="none" stroke="currentColor" viewBox="0 0 24 24">

<path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M9 19v-6a2 2 0 00-2-2H5a2 2 0 00-2 2v6a2 2 0 002 2h2a2 2 0 002-2zm0 0V9a2 2 0 012-2h2a2 2 0 012 2v10m-6 0a2 2 0 002 2h2a2 2 0 002-2m0 0V5a2 2 0 012-2h2a2 2 0 012 2v14a2 2 0 01-2 2h-2a2 2 0 01-2-2z"></path>

</svg>

</div>

<h3 class="text-xl font-semibold text-gray-900 mb-4">Admin Dashboard</h3>

<p class="text-gray-600">

Comprehensive admin panel for user management, token control, and system monitoring.

</p>

</div>

<!-- Feature 5 -->

<div class="card-hover bg-gray-50 p-8 rounded-xl">

<div class="w-12 h-12 bg-purple-500 rounded-lg flex items-center justify-center mb-6">

<svg class="w-6 h-6 text-white" fill="none" stroke="currentColor" viewBox="0 0 24 24">

<path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M12 8v4l3 3m6-3a9 9 0 11-18 0 9 9 0 0118 0z"></path>

</svg>

</div>

<h3 class="text-xl font-semibold text-gray-900 mb-4">Session Management</h3>

<p class="text-gray-600">

Automatic session expiry, force logout capabilities, and session monitoring.

</p>

</div>

<!-- Feature 6 -->

<div class="card-hover bg-gray-50 p-8 rounded-xl">

<div class="w-12 h-12 bg-red-500 rounded-lg flex items-center justify-center mb-6">

<svg class="w-6 h-6 text-white" fill="none" stroke="currentColor" viewBox="0 0 24 24">

<path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M12 9v2m0 4h.01m-6.938 4h13.856c1.54 0 2.502-1.667 1.732-2.5L13.732 4c-.77-.833-1.964-.833-2.732 0L3.732 16.5c-.77.833.192 2.5 1.732 2.5z"></path>

</svg>

</div>

<h3 class="text-xl font-semibold text-gray-900 mb-4">Security Alerts</h3>

<p class="text-gray-600">

Automatic detection and alerts for suspicious activities and failed login attempts.

</p>

</div>

</div>

</div>

</section>

<!-- How It Works Section -->

<section class="py-20 bg-gray-50">

<div class="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">

<div class="text-center mb-16">

<h2 class="text-4xl font-bold text-gray-900 mb-4">

How VAUTH Works

</h2>

<p class="text-xl text-gray-600 max-w-2xl mx-auto">

Simple and secure authentication process

</p>

</div>

<div class="grid grid-cols-1 md:grid-cols-3 gap-8">

<div class="text-center">

<div class="w-16 h-16 bg-primary rounded-full flex items-center justify-center mx-auto mb-6">

<span class="text-2xl font-bold text-white">1</span>

</div>

<h3 class="text-xl font-semibold text-gray-900 mb-4">Login with Credentials</h3>

<p class="text-gray-600">

Enter your username and password to begin the authentication process.

</p>

</div>

<div class="text-center">

<div class="w-16 h-16 bg-accent rounded-full flex items-center justify-center mx-auto mb-6">

<span class="text-2xl font-bold text-white">2</span>

</div>

<h3 class="text-xl font-semibold text-gray-900 mb-4">Verify with VAUTH Token</h3>

<p class="text-gray-600">

Use the 6-character token from your VAUTH device to complete verification.

</p>

</div>

<div class="text-center">

<div class="w-16 h-16 bg-secondary rounded-full flex items-center justify-center mx-auto mb-6">

<span class="text-2xl font-bold text-white">3</span>

</div>

<h3 class="text-xl font-semibold text-gray-900 mb-4">Access Dashboard</h3>

<p class="text-gray-600">

Successfully authenticated users gain access to their secure dashboard.

</p>

</div>

</div>

</div>

</section>

<!-- Footer -->

<footer class="bg-dark text-white py-12">

<div class="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">

<div class="text-center">

<h3 class="text-2xl font-bold mb-4">VAUTH</h3>

<p class="text-gray-400 mb-6">

Secure Authentication System with Real-time Monitoring

</p>

<div class="flex justify-center space-x-6">

<a href="/login" class="text-gray-400 hover:text-white transition duration-300">User Login</a>

<a href="/admin/login" class="text-gray-400 hover:text-white transition duration-300">Admin Login</a>

</div>

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

<p class="text-gray-400 text-sm">

© 2024 VAUTH. All rights reserved. Built with security-first principles.

</p>

</div>

</div>

</div>

</footer>

</body>

</html>

//public/login.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>User Login - VAUTH</title>

<script src="https://cdn.tailwindcss.com"></script>

<script>

tailwind.config = {

theme: {

extend: {

colors: {

primary: '#3B82F6',

secondary: '#1E40AF',

accent: '#F59E0B',

dark: '#1F2937',

light: '#F9FAFB'

}

}

}

}

</script>

<style>

.gradient-bg {

background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);

}

.form-input {

transition: all 0.3s ease;

}

.form-input:focus {

transform: translateY(-2px);

box-shadow: 0 10px 25px -5px rgba(59, 130, 246, 0.25);

}

</style>

</head>

<body class="bg-gray-50 min-h-screen">

<!-- Navigation -->

<nav class="bg-white shadow-lg">

<div class="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">

<div class="flex justify-between h-16">

<div class="flex items-center">

<a href="/" class="flex-shrink-0">

<h1 class="text-2xl font-bold text-primary">VAUTH</h1>

</a>

</div>

<div class="flex items-center space-x-4">

<a href="/" class="text-gray-600 hover:text-primary transition duration-300">Home</a>

<a href="/admin/login" class="bg-gray-600 hover:bg-gray-700 text-white px-4 py-2 rounded-lg transition duration-300">

Admin Login

</a>

</div>

</div>

</div>

</nav>

<!-- Main Content -->

<div class="min-h-screen flex items-center justify-center py-12 px-4 sm:px-6 lg:px-8">

<div class="max-w-md w-full space-y-8">

<!-- Header -->

<div class="text-center">

<div class="mx-auto h-12 w-12 bg-primary rounded-full flex items-center justify-center">

<svg class="h-6 w-6 text-white" fill="none" stroke="currentColor" viewBox="0 0 24 24">

<path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M16 7a4 4 0 11-8 0 4 4 0 018 0zM12 14a7 7 0 00-7 7h14a7 7 0 00-7-7z"></path>

</svg>

</div>

<h2 class="mt-6 text-3xl font-extrabold text-gray-900">

User Login

</h2>

<p class="mt-2 text-sm text-gray-600">

Enter your credentials to access your account

</p>

</div>

<!-- Login Form -->

<div class="bg-white py-8 px-6 shadow-xl rounded-xl">

<form id="loginForm" class="space-y-6">

<!-- Username Field -->

<div>

<label for="username" class="block text-sm font-medium text-gray-700 mb-2">

Username

</label>

<input

id="username"

name="username"

type="text"

required

class="form-input appearance-none relative block w-full px-3 py-3 border border-gray-300 placeholder-gray-500 text-gray-900 rounded-lg focus:outline-none focus:ring-primary focus:border-primary focus:z-10 sm:text-sm"

placeholder="Enter your username"

>

</div>

<!-- Password Field -->

<div>

<label for="password" class="block text-sm font-medium text-gray-700 mb-2">

Password

</label>

<input

id="password"

name="password"

type="password"

required

class="form-input appearance-none relative block w-full px-3 py-3 border border-gray-300 placeholder-gray-500 text-gray-900 rounded-lg focus:outline-none focus:ring-primary focus:border-primary focus:z-10 sm:text-sm"

placeholder="Enter your password"

>

</div>

<!-- Error Message -->

<div id="errorMessage" class="hidden bg-red-50 border border-red-200 text-red-700 px-4 py-3 rounded-lg">

<div class="flex">

<svg class="h-5 w-5 text-red-400 mr-2" fill="currentColor" viewBox="0 0 20 20">

<path fill-rule="evenodd" d="M10 18a8 8 0 100-16 8 8 0 000 16zM8.707 7.293a1 1 0 00-1.414 1.414L8.586 10l-1.293 1.293a1 1 0 101.414 1.414L10 11.414l1.293 1.293a1 1 0 001.414-1.414L11.414 10l1.293-1.293a1 1 0 00-1.414-1.414L10 8.586 8.707 7.293z" clip-rule="evenodd"></path>

</svg>

<span id="errorText"></span>

</div>

</div>

<!-- Submit Button -->

<div>

<button

type="submit"

id="loginButton"

class="group relative w-full flex justify-center py-3 px-4 border border-transparent text-sm font-medium rounded-lg text-white bg-primary hover:bg-secondary focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-primary transition duration-300"

>

<span class="absolute left-0 inset-y-0 flex items-center pl-3">

<svg class="h-5 w-5 text-primary-light group-hover:text-primary-dark" fill="currentColor" viewBox="0 0 20 20">

<path fill-rule="evenodd" d="M5 9V7a5 5 0 0110 0v2a2 2 0 012 2v5a2 2 0 01-2 2H5a2 2 0 01-2-2v-5a2 2 0 012-2zm8-2v2H7V7a3 3 0 016 0z" clip-rule="evenodd"></path>

</svg>

</span>

<span id="loginButtonText">Sign In</span>

</button>

</div>

</form>

<!-- Additional Links -->

<div class="mt-6 text-center">

<p class="text-sm text-gray-600">

Need admin access?

<a href="/admin/login" class="font-medium text-primary hover:text-secondary transition duration-300">

Admin Login

</a>

</p>

</div>

</div>

<!-- Info Box -->

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

<div class="flex">

<svg class="h-5 w-5 text-blue-400 mr-2" fill="currentColor" viewBox="0 0 20 20">

<path fill-rule="evenodd" d="M18 10a8 8 0 11-16 0 8 8 0 0116 0zm-7-4a1 1 0 11-2 0 1 1 0 012 0zM9 9a1 1 0 000 2v3a1 1 0 001 1h1a1 1 0 100-2v-3a1 1 0 00-1-1H9z" clip-rule="evenodd"></path>

</svg>

<div class="text-sm text-blue-700">

<p class="font-medium">Two-Factor Authentication Required</p>

<p class="mt-1">After login, you'll need to verify with your VAUTH device token.</p>

</div>

</div>

</div>

</div>

</div>

<script>

document.getElementById('loginForm').addEventListener('submit', async function(e) {

e.preventDefault();

const username = document.getElementById('username').value;

const password = document.getElementById('password').value;

const errorDiv = document.getElementById('errorMessage');

const errorText = document.getElementById('errorText');

const loginButton = document.getElementById('loginButton');

const loginButtonText = document.getElementById('loginButtonText');

// Hide previous errors

errorDiv.classList.add('hidden');

// Show loading state

loginButton.disabled = true;

loginButtonText.textContent = 'Signing In...';

loginButton.classList.add('opacity-75');

try {

const response = await fetch('/api/user/login', {

method: 'POST',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify({ username, password })

});

const data = await response.json();

if (data.success) {

// Store device ID for 2FA

sessionStorage.setItem('vauth\_device\_id', data.vauth\_device\_ID);

sessionStorage.setItem('username', username);

// Redirect to 2FA page

window.location.href = '/2fa';

} else {

// Show error

errorText.textContent = data.message || 'Login failed';

errorDiv.classList.remove('hidden');

}

} catch (error) {

console.error('Login error:', error);

errorText.textContent = 'Network error. Please try again.';

errorDiv.classList.remove('hidden');

} finally {

// Reset button state

loginButton.disabled = false;

loginButtonText.textContent = 'Sign In';

loginButton.classList.remove('opacity-75');

}

});

// Check if already logged in

window.addEventListener('load', async function() {

try {

const response = await fetch('/api/user/session-status');

const data = await response.json();

if (data.success && data.authenticated) {

window.location.href = '/dashboard';

}

} catch (error) {

console.error('Session check error:', error);

}

});

</script>

</body>

</html>

//public/js/[admin-dashboard.js](http://admin-dashboard.js)

// Admin Dashboard JavaScript

let socket;

let currentSection = 'dashboard';

let autoDeleteEnabled = false;

// Initialize when page loads

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

initializeSocket();

initializeSidebar();

loadDashboardStats();

loadInitialData();

// Set up event listeners

setupEventListeners();

// Refresh data every 30 seconds

setInterval(loadDashboardStats, 30000);

});

// Initialize Socket.io connection

function initializeSocket() {

socket = io();

socket.on('connect', function() {

console.log('Connected to server');

socket.emit('join-admin');

});

// Real-time event handlers

socket.on('user-login-attempt', handleLoginAttempt);

socket.on('token-verification', handleTokenVerification);

socket.on('session-created', handleSessionCreated);

socket.on('user-logout', handleUserLogout);

socket.on('security-alert', handleSecurityAlert);

socket.on('virtual-token-generated', handleVirtualTokenGenerated);

socket.on('session-force-logout', handleSessionForceLogout);

}

// Initialize sidebar navigation

function initializeSidebar() {

const sidebarItems = document.querySelectorAll('.sidebar-item');

sidebarItems.forEach(item => {

item.addEventListener('click', function(e) {

e.preventDefault();

// Remove active class from all items

sidebarItems.forEach(i => i.classList.remove('active'));

// Add active class to clicked item

this.classList.add('active');

// Show corresponding section

const section = this.getAttribute('data-section');

showSection(section);

});

});

}

// Show specific section

function showSection(sectionName) {

// Hide all sections

const sections = document.querySelectorAll('.section');

sections.forEach(section => section.classList.add('hidden'));

// Show selected section

const targetSection = document.getElementById(`${sectionName}-section`);

if (targetSection) {

targetSection.classList.remove('hidden');

currentSection = sectionName;

// Load section-specific data

loadSectionData(sectionName);

}

}

// Load section-specific data

function loadSectionData(sectionName) {

switch(sectionName) {

case 'users':

loadUsers();

break;

case 'tokens':

loadTokens();

break;

case 'sessions':

loadSessions();

break;

case 'dashboard':

loadDashboardStats();

break;

}

}

// Load dashboard statistics

async function loadDashboardStats() {

try {

const response = await fetch('/api/admin/dashboard-stats');

const data = await response.json();

if (data.success) {

document.getElementById('totalUsers').textContent = data.stats.totalUsers;

document.getElementById('activeTokens').textContent = data.stats.activeTokens;

document.getElementById('activeSessions').textContent = data.stats.activeSessions;

document.getElementById('expiredTokens').textContent = data.stats.expiredTokens;

autoDeleteEnabled = data.stats.autoDeleteEnabled;

updateAutoDeleteStatus();

}

} catch (error) {

console.error('Error loading dashboard stats:', error);

}

}

// Load users data

async function loadUsers() {

try {

const response = await fetch('/api/admin/users');

const data = await response.json();

if (data.success) {

const tbody = document.getElementById('usersTableBody');

tbody.innerHTML = '';

if (data.users.length === 0) {

tbody.innerHTML = '<tr><td colspan="4" class="px-6 py-4 text-center text-gray-500">No users found</td></tr>';

return;

}

data.users.forEach(user => {

const row = document.createElement('tr');

row.innerHTML = `

<td class="px-6 py-4 whitespace-nowrap text-sm font-medium text-gray-900">${user.name}</td>

<td class="px-6 py-4 whitespace-nowrap text-sm text-gray-500">${user.username}</td>

<td class="px-6 py-4 whitespace-nowrap text-sm text-gray-500 font-mono">${user.vauth\_device\_ID}</td>

<td class="px-6 py-4 whitespace-nowrap text-sm text-gray-500">${formatDate(user.created\_at)}</td>

`;

tbody.appendChild(row);

});

}

} catch (error) {

console.error('Error loading users:', error);

}

}

// Load tokens data

async function loadTokens() {

try {

const response = await fetch('/api/admin/tokens');

const data = await response.json();

if (data.success) {

const tbody = document.getElementById('tokensTableBody');

tbody.innerHTML = '';

autoDeleteEnabled = data.autoDeleteEnabled;

updateAutoDeleteStatus();

if (data.tokens.length === 0) {

tbody.innerHTML = '<tr><td colspan="5" class="px-6 py-4 text-center text-gray-500">No tokens found</td></tr>';

return;

}

data.tokens.forEach(token => {

const row = document.createElement('tr');

const statusClass = getStatusClass(token.status);

const timeRemaining = token.status === 'ACTIVE' ? formatTime(token.timeRemaining) : '--';

row.innerHTML = `

<td class="px-6 py-4 whitespace-nowrap text-sm text-gray-500 font-mono">${token.device\_id}</td>

<td class="px-6 py-4 whitespace-nowrap">

<span class="px-2 inline-flex text-xs leading-5 font-semibold rounded-full ${statusClass}">

${token.status}

</span>

</td>

<td class="px-6 py-4 whitespace-nowrap text-sm text-gray-500">${timeRemaining}</td>

<td class="px-6 py-4 whitespace-nowrap text-sm text-gray-500">${formatDate(token.created\_at)}</td>

<td class="px-6 py-4 whitespace-nowrap text-sm font-medium">

<button onclick="deleteToken('${token.\_id}')" class="text-red-600 hover:text-red-900">Delete</button>

</td>

`;

tbody.appendChild(row);

});

}

} catch (error) {

console.error('Error loading tokens:', error);

}

}

// Load sessions data

async function loadSessions() {

try {

const response = await fetch('/api/admin/sessions');

const data = await response.json();

if (data.success) {

const tbody = document.getElementById('sessionsTableBody');

tbody.innerHTML = '';

if (data.sessions.length === 0) {

tbody.innerHTML = '<tr><td colspan="6" class="px-6 py-4 text-center text-gray-500">No active sessions</td></tr>';

return;

}

data.sessions.forEach(session => {

const row = document.createElement('tr');

row.innerHTML = `

<td class="px-6 py-4 whitespace-nowrap text-sm font-medium text-gray-900">${session.username}</td>

<td class="px-6 py-4 whitespace-nowrap text-sm text-gray-500 font-mono">${session.device\_id}</td>

<td class="px-6 py-4 whitespace-nowrap text-sm text-gray-500">${session.ip}</td>

<td class="px-6 py-4 whitespace-nowrap text-sm text-gray-500">${formatDate(session.started\_at)}</td>

<td class="px-6 py-4 whitespace-nowrap text-sm text-gray-500">${formatTime(session.timeRemaining)}</td>

<td class="px-6 py-4 whitespace-nowrap text-sm font-medium">

<button onclick="forceLogout('${session.\_id}')" class="text-red-600 hover:text-red-900">Force Logout</button>

</td>

`;

tbody.appendChild(row);

});

}

} catch (error) {

console.error('Error loading sessions:', error);

}

}

// Setup event listeners

function setupEventListeners() {

// Logout button

document.getElementById('logoutButton').addEventListener('click', logout);

// Add user form

document.getElementById('addUserForm').addEventListener('submit', handleAddUser);

// Virtual token form

document.getElementById('virtualTokenForm').addEventListener('submit', handleVirtualToken);

// Refresh buttons

document.getElementById('refreshUsers').addEventListener('click', loadUsers);

document.getElementById('refreshTokens').addEventListener('click', loadTokens);

document.getElementById('refreshSessions').addEventListener('click', loadSessions);

// Auto-delete toggle

document.getElementById('toggleAutoDelete').addEventListener('click', toggleAutoDelete);

// Alert modal close

document.getElementById('closeAlert').addEventListener('click', closeAlert);

}

// Handle add user form submission

async function handleAddUser(e) {

e.preventDefault();

const formData = new FormData(e.target);

const userData = Object.fromEntries(formData);

try {

const response = await fetch('/api/admin/add-user', {

method: 'POST',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify(userData)

});

const data = await response.json();

if (data.success) {

// Show generated credentials

document.getElementById('genUsername').textContent = data.credentials.username;

document.getElementById('genPassword').textContent = data.credentials.password;

document.getElementById('genDeviceId').textContent = data.credentials.vauth\_device\_ID;

document.getElementById('credentialsDisplay').classList.remove('hidden');

// Reset form

e.target.reset();

// Refresh stats

loadDashboardStats();

} else {

alert('Error creating user: ' + data.message);

}

} catch (error) {

console.error('Error adding user:', error);

alert('Network error. Please try again.');

}

}

// Handle virtual token generation

async function handleVirtualToken(e) {

e.preventDefault();

const formData = new FormData(e.target);

const deviceId = formData.get('device\_id');

try {

const response = await fetch('/api/admin/virtual-device/generate-token', {

method: 'POST',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify({ device\_id: deviceId })

});

const data = await response.json();

if (data.success) {

// Show generated token

document.getElementById('generatedToken').textContent = data.token;

document.getElementById('tokenExpiry').textContent = formatDate(data.expires\_at);

document.getElementById('tokenDisplay').classList.remove('hidden');

// Reset form

e.target.reset();

} else {

alert('Error generating token: ' + data.message);

}

} catch (error) {

console.error('Error generating virtual token:', error);

alert('Network error. Please try again.');

}

}

// Delete token

async function deleteToken(tokenId) {

if (!confirm('Are you sure you want to delete this token?')) {

return;

}

try {

const response = await fetch(`/api/admin/tokens/${tokenId}`, {

method: 'DELETE'

});

const data = await response.json();

if (data.success) {

loadTokens(); // Refresh tokens list

loadDashboardStats(); // Refresh stats

} else {

alert('Error deleting token: ' + data.message);

}

} catch (error) {

console.error('Error deleting token:', error);

alert('Network error. Please try again.');

}

}

// Force logout session

async function forceLogout(sessionId) {

if (!confirm('Are you sure you want to force logout this session?')) {

return;

}

try {

const response = await fetch(`/api/admin/sessions/${sessionId}/force-logout`, {

method: 'POST'

});

const data = await response.json();

if (data.success) {

loadSessions(); // Refresh sessions list

loadDashboardStats(); // Refresh stats

} else {

alert('Error forcing logout: ' + data.message);

}

} catch (error) {

console.error('Error forcing logout:', error);

alert('Network error. Please try again.');

}

}

// Toggle auto-delete expired tokens

async function toggleAutoDelete() {

try {

const response = await fetch('/api/admin/auto-delete-expired', {

method: 'PATCH'

});

const data = await response.json();

if (data.success) {

autoDeleteEnabled = data.autoDeleteEnabled;

updateAutoDeleteStatus();

loadTokens(); // Refresh tokens list

} else {

alert('Error toggling auto-delete: ' + data.message);

}

} catch (error) {

console.error('Error toggling auto-delete:', error);

alert('Network error. Please try again.');

}

}

// Update auto-delete status display

function updateAutoDeleteStatus() {

const statusElement = document.getElementById('autoDeleteStatus');

statusElement.textContent = autoDeleteEnabled ? 'ON' : 'OFF';

const button = document.getElementById('toggleAutoDelete');

if (autoDeleteEnabled) {

button.classList.remove('bg-yellow-600', 'hover:bg-yellow-700');

button.classList.add('bg-green-600', 'hover:bg-green-700');

} else {

button.classList.remove('bg-green-600', 'hover:bg-green-700');

button.classList.add('bg-yellow-600', 'hover:bg-yellow-700');

}

}

// Load initial data

function loadInitialData() {

loadUsers();

loadTokens();

loadSessions();

}

// Real-time event handlers

function handleLoginAttempt(data) {

addRecentActivity(`Login attempt by ${data.username} from ${data.ip} - ${data.success ? 'Success' : 'Failed'}`, data.success ? 'success' : 'error');

if (currentSection === 'dashboard') {

animateRequestFlow('login', data.success);

}

}

function handleTokenVerification(data) {

addRecentActivity(`Token verification for ${data.username || 'Unknown'} - ${data.success ? 'Valid' : 'Invalid'}`, data.success ? 'success' : 'error');

if (currentSection === 'dashboard') {

animateRequestFlow('token', data.success);

}

// Refresh tokens if on tokens page

if (currentSection === 'tokens') {

loadTokens();

}

}

function handleSessionCreated(data) {

addRecentActivity(`New session created for ${data.username}`, 'success');

loadDashboardStats();

if (currentSection === 'sessions') {

loadSessions();

}

}

function handleUserLogout(data) {

addRecentActivity(`User ${data.username} logged out`, 'info');

loadDashboardStats();

if (currentSection === 'sessions') {

loadSessions();

}

}

function handleSecurityAlert(data) {

let message = '';

if (data.type === 'MULTIPLE\_FAILED\_LOGINS') {

message = `⚠ Warning: ${data.username} has ${data.attempts} failed login attempts from ${data.lastIP}`;

} else if (data.type === 'MULTIPLE\_FAILED\_TOKENS') {

message = `⚠ Warning: ${data.username} is attempting unauthorized access with invalid tokens`;

}

showAlert(message);

addRecentActivity(message, 'warning');

}

function handleVirtualTokenGenerated(data) {

addRecentActivity(`Virtual token generated for device ${data.device\_id}`, 'info');

if (currentSection === 'tokens') {

loadTokens();

}

}

function handleSessionForceLogout(data) {

addRecentActivity(`Session force logout executed`, 'warning');

if (currentSection === 'sessions') {

loadSessions();

}

}

// Add recent activity

function addRecentActivity(message, type = 'info') {

const activityContainer = document.getElementById('recentActivity');

// Remove "No recent activity" message if present

if (activityContainer.querySelector('.text-gray-500')) {

activityContainer.innerHTML = '';

}

const activityItem = document.createElement('div');

activityItem.className = `flex items-center p-3 rounded-lg ${getActivityClass(type)}`;

const icon = getActivityIcon(type);

const timestamp = new Date().toLocaleTimeString();

activityItem.innerHTML = `

${icon}

<div class="ml-3 flex-1">

<p class="text-sm font-medium">${message}</p>

<p class="text-xs opacity-75">${timestamp}</p>

</div>

`;

// Add to top of list

activityContainer.insertBefore(activityItem, activityContainer.firstChild);

// Keep only last 10 activities

while (activityContainer.children.length > 10) {

activityContainer.removeChild(activityContainer.lastChild);

}

}

// Animate request flow on live map

function animateRequestFlow(type, success) {

const liveMap = document.getElementById('liveMap');

// Create animated elements

const sourceIcon = document.createElement('div');

sourceIcon.className = 'floating-icon w-8 h-8 bg-blue-500 rounded-full flex items-center justify-center text-white text-xs';

sourceIcon.textContent = type === 'login' ? 'PC' : 'DEV';

sourceIcon.style.left = '10%';

sourceIcon.style.top = '50%';

const targetIcon = document.createElement('div');

targetIcon.className = 'floating-icon w-8 h-8 bg-gray-500 rounded-full flex items-center justify-center text-white text-xs';

targetIcon.textContent = 'DB';

targetIcon.style.right = '10%';

targetIcon.style.top = '50%';

const line = document.createElement('div');

line.className = `connection-line ${success ? 'bg-green-500' : 'bg-red-500'}`;

line.style.left = '18%';

line.style.top = '50%';

line.style.width = '64%';

liveMap.appendChild(sourceIcon);

liveMap.appendChild(targetIcon);

liveMap.appendChild(line);

// Animate and remove after delay

setTimeout(() => {

if (sourceIcon.parentNode) sourceIcon.parentNode.removeChild(sourceIcon);

if (targetIcon.parentNode) targetIcon.parentNode.removeChild(targetIcon);

if (line.parentNode) line.parentNode.removeChild(line);

}, 3000);

}

// Show security alert modal

function showAlert(message) {

document.getElementById('alertMessage').textContent = message;

document.getElementById('alertModal').classList.remove('hidden');

}

// Close alert modal

function closeAlert() {

document.getElementById('alertModal').classList.add('hidden');

}

// Admin logout

async function logout() {

try {

await fetch('/api/admin/logout', {

method: 'POST'

});

} catch (error) {

console.error('Logout error:', error);

} finally {

window.location.href = '/admin/login';

}

}

// Utility functions

function formatDate(dateString) {

return new Date(dateString).toLocaleString();

}

function formatTime(seconds) {

if (seconds <= 0) return '00:00';

const mins = Math.floor(seconds / 60);

const secs = seconds % 60;

return `${mins.toString().padStart(2, '0')}:${secs.toString().padStart(2, '0')}`;

}

function getStatusClass(status) {

switch(status) {

case 'ACTIVE':

return 'bg-blue-100 text-blue-800';

case 'USED':

return 'bg-green-100 text-green-800';

case 'EXPIRED':

return 'bg-red-100 text-red-800';

default:

return 'bg-gray-100 text-gray-800';

}

}

function getActivityClass(type) {

switch(type) {

case 'success':

return 'bg-green-50 border border-green-200';

case 'error':

return 'bg-red-50 border border-red-200';

case 'warning':

return 'bg-yellow-50 border border-yellow-200';

default:

return 'bg-blue-50 border border-blue-200';

}

}

function getActivityIcon(type) {

const iconClass = 'h-5 w-5';

switch(type) {

case 'success':

return `<svg class="${iconClass} text-green-500" fill="currentColor" viewBox="0 0 20 20"><path fill-rule="evenodd" d="M10 18a8 8 0 100-16 8 8 0 000 16zm3.707-9.293a1 1 0 00-1.414-1.414L9 10.586 7.707 9.293a1 1 0 00-1.414 1.414l2 2a1 1 0 001.414 0l4-4z" clip-rule="evenodd"></path></svg>`;

case 'error':

return `<svg class="${iconClass} text-red-500" fill="currentColor" viewBox="0 0 20 20"><path fill-rule="evenodd" d="M10 18a8 8 0 100-16 8 8 0 000 16zM8.707 7.293a1 1 0 00-1.414 1.414L8.586 10l-1.293 1.293a1 1 0 101.414 1.414L10 11.414l1.293 1.293a1 1 0 001.414-1.414L11.414 10l1.293-1.293a1 1 0 00-1.414-1.414L10 8.586 8.707 7.293z" clip-rule="evenodd"></path></svg>`;

case 'warning':

return `<svg class="${iconClass} text-yellow-500" fill="currentColor" viewBox="0 0 20 20"><path fill-rule="evenodd" d="M8.257 3.099c.765-1.36 2.722-1.36 3.486 0l5.58 9.92c.75 1.334-.213 2.98-1.742 2.98H4.42c-1.53 0-2.493-1.646-1.743-2.98l5.58-9.92zM11 13a1 1 0 11-2 0 1 1 0 012 0zm-1-8a1 1 0 00-1 1v3a1 1 0 002 0V6a1 1 0 00-1-1z" clip-rule="evenodd"></path></svg>`;

default:

return `<svg class="${iconClass} text-blue-500" fill="currentColor" viewBox="0 0 20 20"><path fill-rule="evenodd" d="M18 10a8 8 0 11-16 0 8 8 0 0116 0zm-7-4a1 1 0 11-2 0 1 1 0 012 0zM9 9a1 1 0 000 2v3a1 1 0 001 1h1a1 1 0 100-2v-3a1 1 0 00-1-1H9z" clip-rule="evenodd"></path></svg>`;

}

}