Campus Shield: Privacy-First Campus Safety Platform :

{if you wish to see the rest of the files exported in md, you can check out the repo itself under sparrowdex_Campus-Shield folder}

{"files": [{ "name": "Campus-Shield/server/index.js", "description": "JavaScript/TypeScript implementation file", "file_url": "https://github.com/sparrowdex/Campus-Shield/blob/main/server/index.js", "directory": "https:/github.com/sparrowdex/Campus-Shield/blob/main/server" }, { "name": "Campus-Shield/client/src/pages/MyReports.tsx", "description": "Implementation file", "file_url": "https://github.com/sparrowdex/Campus-Shield/blob/main/client/src/pages/MyReports.tsx", "directory": "https:/github.com/sparrowdex/Campus-Shield/blob/main/client/src/pages" }, { "name": "Campus-Shield/server/routes/admin.js", "description": "JavaScript/TypeScript implementation file", "file_url": "https://github.com/sparrowdex/Campus-Shield/blob/main/server/routes/admin.js", "directory": "https:/github.com/sparrowdex/Campus-Shield/blob/main/server/routes" }, { "name": "Campus-Shield/README.md", "description": "Documentation file", "file_url": "https://github.com/sparrowdex/Campus-Shield/blob/main/README.md", "directory": "https:/github.com/sparrowdex/Campus-Shield/blob/main" }]}

Campus Shield Architecture and Core Components

Campus Shield is a comprehensive campus safety platform designed to enable anonymous incident reporting, real-time communication, and efficient management of safety concerns. The system follows a client-server architecture with a React frontend and Node.js backend.

Frontend Architecture

The frontend is built using React and TypeScript, leveraging modern web technologies for a responsive and interactive user interface.

- **Component Structure**: The `src/components` directory contains reusable UI components like `LoadingSpinner` and `NotificationBar`.
- **Routing**: React Router is used for navigation, with routes defined in `App.tsx`.
- **State Management**: Context API is utilized for global state management, with `AuthContext` and `NotificationContext` providing authentication and notification functionalities respectively.

Backend Architecture

The backend is powered by Node.js with Express.js, following a modular structure for scalability and maintainability.

- **API Routes**: Defined in the `server/routes` directory, handling authentication, reporting, chat, and administrative functions.
- **Middleware**: Custom middleware in `server/middleware` for error handling, authentication, and role-based access control.
- **Database Integration**: MongoDB is used as the primary database, with Mongoose for object modeling.

Real-time Communication and Socket Integration

Real-time features are implemented using Socket.IO, enabling instant messaging and live updates.

- **Socket Service**: The `socketService.js` file manages WebSocket connections and event handling.

- **Chat Functionality**: Real-time chat is implemented in the `Chat.tsx` component, allowing users to communicate with campus security.

Data Models and Schemas

The system uses several key data models to represent entities:

- **User**: Stores user information, including role and authentication details.
- **Report**: Represents incident reports with details like location, category, and status.
- **ChatRoom**: Manages chat sessions between users and administrators.
- **AdminRequest**: Handles requests for administrative access.

Security and Privacy Features

Authentication and Authorization Implementation

- **JWT-based Authentication**: Implemented in `auth.js` middleware, ensuring secure access to protected routes.
- **Role-based Access Control**: Different user roles (user, admin, moderator) with specific permissions.
- **Anonymous Reporting**: Users can submit reports without revealing their identity.

Admin and Moderation Features

- **Admin Dashboard**: Provides an overview of reports, user statistics, and management tools.
- **Moderation Queue**: Allows moderators to review and process admin access requests.

AI Integration and Data Processing

- **AI Service**: The `aiService.js` file suggests integration of AI for report categorization and sentiment analysis.

Deployment and Infrastructure

{"files": [{ "name": "Campus-Shield/server/index.js", "description": "JavaScript/TypeScript implementation file", "file_url": "https://github.com/sparrowdex/Campus-Shield/blob/main/server/index.js", "directory": "https:/github.com/sparrowdex/Campus-Shield/blob/main/server" }, { "name": "Campus-Shield/client/src/pages/MyReports.tsx", "description": "Implementation file", "file_url": "https://github.com/sparrowdex/Campus-Shield/blob/main/client/src/pages/MyReports.tsx", "directory": "https:/github.com/sparrowdex/Campus-Shield/blob/main/client/src/pages" }, { "name": "Campus-Shield/server/routes/admin.js", "description": "JavaScript/TypeScript implementation file", "file_url": "https://github.com/sparrowdex/Campus-Shield/blob/main/server/routes/admin.js", "directory": "https:/github.com/sparrowdex/Campus-Shield/blob/main/server/routes" }, { "name": "Campus-Shield/README.md", "description": "Documentation file", "file_url": "https://github.com/sparrowdex/Campus-Shield/blob/main/README.md", "directory": "https:/github.com/sparrowdex/Campus-Shield/blob/main" }]}

The project includes deployment guides for various platforms, with a focus on cloud-based solutions for scalability and reliability.

- **Environment Configuration**: Utilizes `.env` files for managing environment-specific settings.

- **Deployment Instructions**: Detailed in `DEPLOYMENT_GUIDE.md`, covering backend deployment on platforms like Render.

This comprehensive architecture enables Campus Shield to provide a secure, efficient, and user-friendly platform for managing campus safety concerns while prioritizing user privacy and data security.

References:

Code:

[{"file_name": "Campus-Shield/server/index.js", "file_path": "https://github.com/sparrowdex/Campus-Shield/blob/main/server/index.js", "markdown_link": "- Campus-Shield/server/index.js\n", "code_chunk": "const express = require('express');\nconst http = require('http');\nconst socketIo = require('socket.io');\nconst cors = require('cors');\nconst helmet = require('helmet');\nconst compression = require('compression');\nconst morgan = require('morgan');\nconst rateLimit = require('express-rate-limit');\nconst path = require('path');\nrequire('dotenv').config();\nconst mongoose = require('mongoose');\n\n// Debug environment variables\nconsole.log('\ud83d\udd0d Environment variables check:');\nconsole.log('NODE_ENV:', process.env.NODE_ENV);\nconsole.log('PORT:', process.env.PORT);\nconsole.log('MONGODB_URI exists:', !!process.env.MONGODB_URI);\nconsole.log('MONGODB_URI length:', process.env.MONGODB_URI ? process.env.MONGODB_URI.length : 0);\nconsole.log('All env vars:', Object.keys(process.env).filter(key => key.includes('MONGO') || key.includes('MONGODB')));\n\nconst connectDB = require('./config/database');\nconst authRoutes = require('./routes/auth');\nconst reportRoutes = require('./routes/reports');\nconst chatRoutes = require('./routes/chat');\nconst adminRoutes = require('./routes/admin');\nconst notificationsRoutes = require('./routes/notifications');\nconst { initializeSocket } = require('./services/socketService');\nconst { errorHandler } = require('./middleware/errorHandler');\n\nconst app = express();\nconst server = http.createServer(app);\nconst io = socketIo(server, {\n cors: {\n origin: process.env.CORS_ORIGIN || "http://localhost:3000",\n methods: ["GET", "POST"]\n }\n});\n\n// Connect to MongoDB (non-blocking)\nconnectDB().catch(err => {\n console.error('Failed to connect to database:', err);\n // Don't exit the process, let it continue\n});\n\n// Initialize Socket.io\ninitializeSocket(io);\n\n// Security middleware\napp.use(helmet({\n contentSecurityPolicy: {\n directives: {\n defaultSrc: ["'self'"],\n styleSrc: ["'self'", "'unsafe-inline'"],\n scriptSrc: ["'self'"],\n imgSrc: ["'self'", "[REMOVED_DATA_URI]\n connectSrc: ["'self'", "ws:", "wss:"]\n }\n }\n}));\n\n// Rate limiting\nconst limiter = rateLimit({\n windowMs: parseInt(process.env.RATE_LIMIT_WINDOW) || 15 * 60 * 1000, // 15 minutes\n max: parseInt(process.env.RATE_LIMIT_MAX_REQUESTS) || 100,\n message: 'Too many requests from this IP, please try again later.',\n standardHeaders: true,\n legacyHeaders: false,\n});\napp.use('/api/', limiter);\n\n// Middleware\napp.use(compression());\napp.use(morgan('combined'));\n// Apply CORS globally before all routes\nconst allowedOrigins = process.env.CORS_ORIGIN\n ? process.env.CORS_ORIGIN.split(',').map(origin => origin.trim())\n : ["http://localhost:3000"];\nconsole.log('Allowed CORS origins:', allowedOrigins);\n\napp.use(cors({\n origin: function(origin, callback) {\n if (!origin) return callback(null, true);\n if (allowedOrigins.includes(origin)) return callback(null, true);\n return callback(new Error('Not allowed by CORS'));\n },\n credentials: true\n}));\napp.use(express.json({ limit: '10mb' }));\napp.use(express.urlencoded({ extended: true, limit: '10mb' }));\n\n// Serve uploaded files\napp.use('/uploads', express.static(path.join(__dirname, 'uploads')));\n\n// Health check endpoint\napp.get('/health', (req, res) => {\n const health = {\n status: 'OK',\n timestamp: new Date().toISOString(),\n uptime: process.uptime(),\n environment: process.env.NODE_ENV || 'development',\n database: mongoose.connection.readyState === 1 ? 'connected' : 'disconnected'\n };\n \n res.status(200).json(health);\n});\n\n// API Routes\napp.use('/api/auth', authRoutes);\napp.use('/api/reports', reportRoutes);\napp.use('/api/chat', chatRoutes);\napp.use('/api/admin', adminRoutes);\napp.use('/api/notifications', notificationsRoutes);\n\n// Error handling middleware\napp.use(errorHandler);\n\n// 404 handler\napp.use('', (req, res) => {\n res.status(404).json({\n success: false,\n message: 'Route not found'\n });\n});\n\nconst PORT = process.env.PORT || 5000;"}, {"file_name": "Campus-Shield/server/routes/reports.js", "file_path": "https://github.com/sparrowdex/Campus-Shield/blob/main/server/routes/reports.js", "markdown_link": "- Campus-Shield/server/routes/reports.js\n", "code_chunk": "const express = require('express');\nconst multer = require('multer');\nconst path = require('path');\nconst { body, validationResult, query } = require('express-validator');\nconst Report = require('../models/Report');\nconst auth = require('../middleware/auth');\nconst { categorizeReport, analyzeSentiment } = require('../services/aiService');\nconst memoryStore = require('../services/memoryStore');\n\nconst router = express.Router();\n\n// Check if MongoDB is connected\nconst isMongoConnected = () => {\n return Report.db && Report.db.readyState === 1;\n};\n\n// Configure multer for file uploads\nconst storage = multer.diskStorage({\n destination: (req, file, cb) => {\n cb(null, 'uploads/');\n },\n filename: (req, file, cb) => {\n const uniqueSuffix = Date.now() + '-' + Math.round(Math.random() * 1E9);\n cb(null, file.fieldname + '-' + uniqueSuffix + path.extname(file.originalname));\n }\n});\n\nconst upload = multer({\n storage: storage,\n limits: {\n fileSize: 10 * 1024 * 1024, // 10MB limit\n },\n fileFilter: (req, file, cb) => {\n const allowedTypes = /jpeg|jpg|png|gif|mp4|avi|mov|wav|mp3|pdf/;\n const extname = allowedTypes.test(path.extname(file.originalname).toLowerCase());\n const mimetype = allowedTypes.test(file.mimetype);\n \n if (mimetype && extname) {\n return cb(null, true);\n } else {\n cb(new Error('Only image, video, audio, and PDF files are allowed'));\n }\n }\n});\n\n// @route POST /api/reports\n// @desc Submit a new incident report\n// @access Private\nrouter.post('/', auth, upload.array('attachments', 5), [\n body('title').isLength({ min: 5, max: 200 }).withMessage('Title must be between 5 and 200 characters'),\n body('description').isLength({ min: 10, max: 2000 }).withMessage('Description must be between 10 and 2000 characters'),\n body('category').isIn([\n 'harassment', 'assault', 'theft', 'vandalism', 'suspicious_activity',\n 'emergency', 'safety_hazard', 'discrimination', 'bullying', 'other'\n]).withMessage('Invalid category'),\n // Add a custom sanitizer for coordinates\n body('location.coordinates').customSanitizer(value => {\n if (typeof value === 'string') {\n try {\n return JSON.parse(value);\n } catch {\n return value;\n }\n }\n return value;\n }),\n body('location.coordinates').isArray({ min: 2, max: 2 }).withMessage('Location coordinates are required'),\n body('location.coordinates.').isFloat().withMessage('Coordinates must be numbers'),\n body('incidentTime').optional().isISO8601().withMessage('Invalid date format')\n], async (req, res) => {\n try {\n const errors = validationResult(req);\n if (!errors.isEmpty()) {\n return res.status(400).json({ success: false, errors: errors.array() });\n }\n\n const {\n title,\n description,\n category,\n location,\n incidentTime\n } = req.body;\n\n // Get user info\n const user = req.user;\n\n // Process attachments\n const attachments = req.files ? req.files.map(file => ({\n filename: file.filename,\n originalName: file.originalname,\n mimetype: file.mimetype,\n size: file.size,\n path: file.path\n })) : [];\n\n // Parse coordinates if sent as a string\n let coordinates = location.coordinates;\n if (typeof coordinates === 'string') {\n try {\n coordinates = JSON.parse(coordinates);\n } catch (e) {\n coordinates = undefined;\n }\n }\n\n // Create report data\n const reportData = {\n reporterId: user.userId,\n title,\n description,\n category,\n location: {\n type: 'Point',\n coordinates: coordinates || location.coordinates,\n address: location.address || '',\n building: location.building || '',\n floor: location.floor || ''\n },\n incidentTime: incidentTime || new Date(),\n attachments,\n isAnonymous: user.isAnonymous || true,\n ipAddress: req.ip,\n userAgent: req.get('User-Agent')\n };"}, {"file_name": "Campus-Shield/SETUP_GUIDE.md", "file_path": "https://github.com/sparrowdex/Campus-Shield/blob/main/SETUP_GUIDE.md", "markdown_link": "- Campus-Shield/SETUP_GUIDE.md\n", "code_chunk": "# \ud83d\udee1\ufe0f CampusShield Setup Guide for Beginners\n\n## \ud83d\udccb Table of Contents\n1. What is CampusShield?\n2. Cloning from GitHub\n3. Prerequisites\n4. Installation Steps\n5. Running the

Application\n6. Understanding the Project Structure\n7. Common Issues & Solutions\n8. Development Workflow\n9. Testing the Features\n10. Contributing\n\n---\n\n## \ud83c\udfaf What is CampusShield?\n\nCampusShield is a **privacy-first campus safety platform** that allows students to:\n- **Report incidents anonymously** with location tracking\n- **Chat securely** with campus authorities\n- **Track report status** and updates\n- **View safety analytics** and heatmaps\n\n---\n\n## \ud83d\ude80 Cloning from GitHub\n\nIf you are starting from the GitHub repository:\n`bash\ngit clone https://github.com/YOUR_USERNAME/YOUR_REPO.git\ncd CampusShield\n`\n\n---\n\n## \ud83d\udccb Prerequisites\n\nBefore you start, make sure you have these installed:\n\n### **Required Software:**\n- \u2705 **Node.js** (v16 or higher) - Download here\n- \u2705 **MongoDB** (v5 or higher) - Download here\n- \u2705 **Git** (optional) - Download here\n\n### **How to Check if Installed:**\nOpen Command Prompt and type:\n`bash\nnode --version\nnpm --version\nmongod --version\n`\n\n---\n\n## \ud83d\ude80 Installation Steps\n\n### **Step 1: Download the Project**\n1. **Download** the CampusShield project files (or clone from GitHub)\n2. **Extract** to a folder (e.g., `C:\\CampusShield`)\n3. **Open Command Prompt** in that folder\n\n### **Step 2: Install Dependencies**\n`bash\n# Install backend dependencies\ncd server\nnpm install\n\n# Install frontend dependencies\ncd ../client\nnpm install\n`\n\n### **Step 3: Set Up MongoDB**\n1. **Download MongoDB** from mongodb.com\n2. **Install with default settings** (Complete installation)\n3. **MongoDB will run as a Windows Service** (starts automatically)\n\n### **Step 4: Create Environment File**\n1. **Navigate to server folder**: `cd server`\n2. **Copy `.env.example` to `.env`**:\n `bash\n copy .env.example .env\n # Or manually create .env and copy the contents from .env.example\n `\n3. **Edit `.env` as needed** (set your MongoDB URI, JWT secret, etc.)\n\n### **Step 5: (Optional) Seed Admin/Moderator Accounts**\nIf you want demo admin/moderator accounts, run:\n`bash\ncd server\nnode seedAdmins.js\n`\n\n---\n\n## \ud83c\udfc3\u200d\u2642\ufe0f Running the Application\n\n### **You Need 2 Command Prompt Windows:**\n\n#### **Window 1: Backend Server**\n`bash\ncd server\nnpm run dev\n`\n**Expected Output:**\n`\n\ud83d\udce6 MongoDB Connected: localhost\n\ud83d\ude80 CampusShield server running on port 5000\n\ud83d\udcca Health check: http://localhost:5000/health\n\ud83d\udd12 Environment: development\n`\n\n#### **Window 2: Frontend Server**\n`bash\ncd client\nnpm start\n`\n**Expected Output:**\n`\nCompiled successfully!\n\nYou can now view campus-shield in the browser.\n\n Local: http://localhost:3000\n On Your Network: http://192.168.x.x:3000\n`\n\n### **Access the Application:**\n- **Frontend**: http://localhost:3000\n- **Backend API**: http://localhost:5000\n- **Health Check**: http://localhost:5000/health\n\n---\n\n## \ud83d\udcf1 Mobile Responsiveness\nCampusShield is fully mobile responsive and works best on modern browsers. For the best experience, use Chrome, Firefox, or Edge on desktop or mobile.\n\n---\n\n## \ud83d\udcc1 Understanding the Project Structure"}, {"file_name": "Campus-Shield/DEPLOYMENT_GUIDE.md", "file_path": "https://github.com/sparrowdex/Campus-Shield/blob/main/DEPLOYMENT_GUIDE.md", "markdown_link": "- Campus-Shield/DEPLOYMENT_GUIDE.md\n", "code_chunk": "# Campus Shield Deployment Guide\n\n## \ud83d\ude80 Backend Deployment (Render Recommended)\n\n### Prerequisites\n- GitHub account\n- Render account (free at render.com)\n\n### Step 1: Prepare Your Repository\n1. Make sure your code is pushed to GitHub\n2. Verify your `server/package.json` has the correct scripts\n\n### Step 2: Deploy to Render\n1. Go to render.com and sign up/login\n2. Click "New Web Service" and connect your GitHub repo\n3. Set the root directory to `server`\n4. Set build command to `npm install` and start command to `node index.js`\n5. Add environment variables as needed\n\n### Step 3: Set Up MongoDB Database\n1. Use MongoDB Atlas (see below) or Render's managed database\n2. Copy the MongoDB connection string\n\n### Step 4: Configure Environment Variables\nAdd these to your Render service:\n\n`env\nNODE_ENV=production\nPORT=10000\nCORS_ORIGIN=https://your-frontend-

`domain.vercel.app\nMONGODB_URI=your-mongodb-atlas-connection-string\nJWT_SECRET=your-super-secret-jwt-key-change-this-in-production\nRATE_LIMIT_WINDOW=900000\nRATE_LIMIT_MAX_REQUESTS=100\nMAX_FILE_SIZE=10485760\nUPLOAD_PATH=uploads\nBCRYPT_ROUNDS=12\nLOG_LEVEL=info\nUSER_DATA_RETENTION_DAYS=365\nREPORT_DATA_RETENTION_DAYS=730\n`\n\n### Step 5: Deploy and Test\n1. Render will automatically deploy when you push to GitHub\n2. Check the deployment logs in Render dashboard\n3. Test your API endpoints using the provided URL\n4. Health check: `https://your-app.onrender.com/health`\n\n### Step 6: Get Your Backend URL\n- Render will provide a URL like: `https://your-app-name.onrender.com`\n- Save this URL for your frontend deployment\n\n## \ud83d\udd27 Troubleshooting\n\n### Common Issues:\n1. **Build fails**: Check Render logs for missing dependencies\n2. **Database connection fails**: Verify MongoDB URI in environment variables\n3. **CORS errors**: Update CORS_ORIGIN to match your frontend domain\n4. **Port issues**: Render automatically sets PORT environment variable\n\n### Useful Commands:\n- View Render logs in the dashboard\n\n## \ud83d\udcdd Next Steps\nAfter successful backend deployment:\n1. Deploy frontend to Vercel (see next section)\n2. Update CORS_ORIGIN with your frontend URL\n3. Test the complete application\n4. Set up custom domain (optional)\n\n## \ud83d\udd12 Security Notes\n- Change JWT_SECRET to a strong random string\n- Use HTTPS in production\n- Set up proper rate limiting\n- Consider adding API key authentication for admin routes"}, {"file_name": "Campus-Shield/client/src/pages/Home.tsx", "file_path": "https://github.com/sparrowdex/Campus-Shield/blob/main/client/src/pages/Home.tsx", "markdown_link": "- Campus-Shield/client/src/pages/Home.tsx\n", "code_chunk": "{/* CTA Section /}\n <section className="bg-primary-600 text-white py-16">\n <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8 text-center">\n <h2 className="text-3xl md:text-4xl font-bold mb-4">\n Ready to Make Campus Safer?\n \n <p className="text-xl text-primary-100 mb-8 max-w-2xl mx-auto">\n Join thousands of students who are already using CampusShield to report \n safety concerns and stay informed.\n

\n {user ? (\n <>\n {user?.role === 'user' && (\n <Link to="/report" className="btn-primary bg-white text-primary-600 hover:bg-gray-100">\n Report an Incident\n \n)}\n {user?.role === 'admin' && (\n <>\n <Link to="/admin" className="btn-primary bg-white text-primary-600 hover:bg-gray-100">\n Admin Dashboard\n \n <Link to="/chat" className="btn-secondary bg-transparent border-white text-white hover:bg-white hover:text-primary-600">\n Chat\n \n </>\n)}\n {user?.role === 'moderator' && (\n <>\n <Link to="/admin/requests" className="btn-primary bg-white text-primary-600 hover:bg-gray-100">\n Admin Requests\n \n </>\n)}\n {!user && (\n <div className="flex flex-col sm:flex-row gap-4 justify-center">\n <Link to="/register" className="btn-primary bg-white text-primary-600 hover:bg-gray-100">\n Get Started Now\n \n <Link to="/login" className="btn-secondary bg-transparent border-white text-white hover:bg-white hover:text-primary-600">\n Login\n \n \n)}\n </>\n) : (\n <div className="flex flex-col sm:flex-row gap-4 justify-center">\n <Link to="/register" className="btn-primary bg-white text-primary-600 hover:bg-gray-100">\n Get Started Now\n \n <Link to="/login" className="btn-secondary bg-transparent border-white text-white hover:bg-white hover:text-primary-600">\n Login\n \n \n)}\n \n \n\n {/ Emergency Notice /}\n <div className="bg-danger-50 border-l-4 border-danger-400 p-4">\n <div className="flex">\n <ExclamationTriangleIcon className="h-5 w-5 text-danger-400" />\n <div className="ml-3">\n <p className="text-sm text-danger-700">\n **Emergency?** If you're in immediate danger, call campus security or 911 immediately. \n CampusShield is for non-emergency safety reporting.\n

\n \n \n \n \n);\n};\n\nexport default Home;"}, {"file_name": "Campus-Shield/server/services/memoryStore.js", "file_path": "https://github.com/sparrowdex/Campus-Shield/blob/main/server/services/memoryStore.js", "markdown_link": "- Campus-Shield/server/services/memoryStore.js\n", "code_chunk": "// Admin request operations\n createAdminRequest(userId, requestData) {\n const request = {\n id: this.nextRequestId.toString(),\n userId,\n ...requestData,\n status: 'pending', // pending, approved, rejected\n createdAt: new

Date().toISOString(),\n reviewedBy: null,\n reviewedAt: null,\n reviewNotes: null\n };\n this.adminRequests.set(request.id, request);\n this.nextRequestId++;\n return request;\n }\n\n getAdminRequests(status = null) {\n const requests = Array.from(this.adminRequests.values());\n if (status) {\n return requests.filter(req => req.status === status);\n }\n return requests;\n }\n\n updateAdminRequest(requestId, updates) {\n const request = this.adminRequests.get(requestId);\n if (request) {\n Object.assign(request, updates, { \n reviewedAt: new Date().toISOString(),\n updatedAt: new Date().toISOString()\n });\n this.adminRequests.set(requestId, request);\n return request;\n }\n return null;\n }\n\n approveAdminRequest(requestId, approvedBy, notes = '') {\n const request = this.updateAdminRequest(requestId, {\n status: 'approved',\n reviewedBy: approvedBy,\n reviewNotes: notes\n });\n\n if (request) {\n // Promote user to admin\n this.updateUser(request.userId, { role: 'admin' });\n }\n\n return request;\n }\n\n rejectAdminRequest(requestId, rejectedBy, notes = '') {\n return this.updateAdminRequest(requestId, {\n status: 'rejected',\n reviewedBy: rejectedBy,\n reviewNotes: notes\n });\n }\n\n // Report operations\n createReport(reportData) {\n const report = {\n id: this.nextReportId.toString(),\n ...reportData,\n status: 'pending',\n priority: 'medium',\n createdAt: new Date().toISOString(),\n updatedAt: new Date().toISOString(),\n attachments: [],\n publicUpdates: []\n };\n this.reports.set(report.id, report);\n this.nextReportId++;\n return report;\n }\n\n findReportsByUserId(userId) {\n const userReports = [];\n for (const report of this.reports.values()) {\n if (report.userId === userId) {\n userReports.push(report);\n }\n }\n return userReports;\n }\n\n findReportById(id) {\n return this.reports.get(id) || null;\n }\n\n updateReport(id, updates) {\n const report = this.reports.get(id);\n if (report) {\n Object.assign(report, updates, { updatedAt: new Date().toISOString() });\n this.reports.set(id, report);\n return report;\n }\n return null;\n }\n\n // Chat operations\n createChatRoom(roomData) {\n const room = {\n roomId: this.nextRoomId.toString(),\n ...roomData,\n createdAt: new Date().toISOString(),\n lastMessage: null\n };\n this.chatRooms.set(room.roomId, room);\n this.nextRoomId++;\n return room;\n }\n\n findChatRoomByReportId(reportId) {\n for (const room of this.chatRooms.values()) {\n if (room.reportId === reportId) {\n return room;\n }\n }\n return null;\n }\n\n findChatRoomsByUserId(userId) {\n const userRooms = [];\n for (const room of this.chatRooms.values()) {\n if (room.userId === userId) {\n userRooms.push(room);\n }\n }\n return userRooms;\n }\n\n createMessage(messageData) {\n const message = {\n id: this.nextMessageId.toString(),\n ...messageData,\n timestamp: new Date().toISOString()\n };\n this.messages.set(message.id, message);\n this.nextMessageId++;\n return message;\n }\n\n findMessagesByRoomId(roomId) {\n const roomMessages = [];\n for (const message of this.messages.values()) {\n if (message.roomId === roomId) {\n roomMessages.push(message);\n }\n }\n return roomMessages.sort((a, b) => new Date(a.timestamp) - new Date(b.timestamp));\n }\n\n // Admin operations\n getAllReports() {\n return Array.from(this.reports.values());\n }\n\n getAllUsers() {\n return Array.from(this.users.values());\n }"}, {"file_name": "Campus-Shield/client/src/pages/Chat.tsx", "file_path": "https://github.com/sparrowdex/Campus-Shield/blob/main/client/src/pages/Chat.tsx", "markdown_link": "- Campus-Shield/client/src/pages/Chat.tsx\n", "code_chunk": "useEffect(() => {\n if (selectedRoom) {\n fetchMessages(selectedRoom);\n // Fetch report status for the selected chat room\n const room = chatRooms.find(r => r._id === selectedRoom);\n if (room && room.reportId) {\n fetch(`${process.env.REACT_APP_API_URL}/api/reports/${room.reportId}`, {\n headers: {\n 'Authorization': `Bearer ${localStorage.getItem('token')}`\n }\n })\n .then(res => res.json())\n .then(data => {\n if (data.success && data.report) {\n setReportStatus(data.report.status);\n } else {\n setReportStatus(null);\n }\n })\n .catch(() => setReportStatus(null));\n } else {\n setReportStatus(null);\n }\n }\n }, [selectedRoom, chatRooms]);\n\n // Mark notifications as read when chat room is opened\n useEffect(() => {\n if (selectedRoom) {\n notifications\n .filter(n => n.link === `/chat?roomId=${selectedRoom}` && !n.read)\n .forEach(n => markAsRead(n.id));\n }\n }, [selectedRoom, notifications, markAsRead]);\n\n useEffect(() => {\n scrollToBottom();\n }, [messages]);\n\n const scrollToBottom = () => {\n messagesEndRef.current?.scrollIntoView({ behavior: 'smooth' });\n };\n\n const fetchChatRooms = async () => {\n

try {\n const response = await fetch(`${process.env.REACT_APP_API_URL}/api/chat/rooms`, {\n headers: {\n 'Authorization': `Bearer ${localStorage.getItem('token')}`\n }\n });\n\n if (!response.ok) {\n throw new Error('Failed to fetch chat rooms');\n }\n\n const data = await response.json();\n setChatRooms(data.rooms || []);\n } catch (err: any) {\n setError(err.message);\n } finally {\n setLoading(false);\n }\n };\n\n const fetchMessages = async (roomId: string) => {\n try {\n const response = await fetch(`${process.env.REACT_APP_API_URL}/api/chat/room/${roomId}/messages`, {\n headers: {\n 'Authorization': `Bearer ${localStorage.getItem('token')}`\n }\n });\n\n if (!response.ok) {\n throw new Error('Failed to fetch messages');\n }\n\n const data = await response.json();\n setMessages(data.messages || []);\n } catch (err: any) {\n setError(err.message);\n }\n };\n\n const sendMessage = async () => {\n if (!newMessage.trim() || !selectedRoom) return;\n setSending(true);\n try {\n // Save message via REST API for persistence\n await fetch(`${process.env.REACT_APP_API_URL}/api/chat/room/${selectedRoom}/message`, {\n method: 'POST',\n headers: {\n 'Authorization': `Bearer ${localStorage.getItem('token')}`,\n 'Content-Type': 'application/json'\n },\n body: JSON.stringify({ message: newMessage })\n });\n // Optionally, still emit via Socket.IO for real-time updates\n socketRef.current?.emit('send_message', {\n roomId: selectedRoom,\n message: newMessage\n });\n setNewMessage('');\n } catch (err: any) {\n setError(err.message);\n } finally {\n setSending(false);\n }\n };\n\n const handleKeyPress = (e: React.KeyboardEvent) => {\n if (e.key === 'Enter' && !e.shiftKey) {\n e.preventDefault();\n sendMessage();\n }\n };\n\n const formatTime = (timestamp: string) => {\n return new Date(timestamp).toLocaleTimeString('en-US', {\n hour: '2-digit',\n minute: '2-digit'\n });\n };\n\n const formatDate = (timestamp: string) => {\n const date = new Date(timestamp);\n const today = new Date();\n const yesterday = new Date(today);\n yesterday.setDate(yesterday.getDate() - 1);\n\n if (date.toDateString() === today.toDateString()) {\n return 'Today';\n } else if (date.toDateString() === yesterday.toDateString()) {\n return 'Yesterday';\n } else {\n return date.toLocaleDateString('en-US', {\n month: 'short',\n day: 'numeric'\n });\n }\n };"}, {"file_name": "Campus-Shield/client/src/pages/MyReports.tsx", "file_path": "https://github.com/sparrowdex/Campus-Shield/blob/main/client/src/pages/MyReports.tsx", "markdown_link": "- Campus-Shield/client/src/pages/MyReports.tsx\n", "code_chunk": "{/ Report Detail Modal /}\n {selectedReport && (\n <div className="fixed inset-0 bg-black bg-opacity-50 flex items-center justify-center p-4 z-50">\n <div className="bg-white rounded-lg max-w-2xl w-full max-h-[90vh] overflow-y-auto">\n <div className="p-6">\n <div className="flex items-center justify-between mb-4">\n <h2 className="text-xl font-bold text-gray-900">Report Details\n <button\n onClick={() => setSelectedReport(null)}\n className="text-gray-400 hover:text-gray-600"\n >\n <XMarkIcon className="h-6 w-6" />\n \n \n\n <div className="space-y-4">\n \n <h3 className="font-semibold text-gray-900 mb-2">{selectedReport.title}\n <div className="flex items-center space-x-3 mb-3">\n <span className={`badge ${statusColors[selectedReport.status as keyof typeof statusColors]}`}>\n {selectedReport.status.replace('_', ' ').toUpperCase()}\n \n <span className={`badge ${priorityColors[selectedReport.priority as keyof typeof priorityColors]}`}>\n {selectedReport.priority.toUpperCase()}\n \n
\n \n\n
\n <h4 className="font-medium text-gray-900 mb-2">Description\n <p className="text-gray-600">{selectedReport.description}

\n
\n\n <div className="grid grid-cols-1 md:grid-cols-2 gap-4">\n
\n <h4 className="font-medium text-gray-900 mb-2">Category\n <p className="text-gray-600 capitalize">\n {categoryLabels[selectedReport.category as keyof typeof categoryLabels]}\n

\n
\n

\n <h4 className="font-medium text-gray-900 mb-2">Incident Time\n <p className="text-gray-600">{formatDate(selectedReport.incidentTime)}

\n
\n \n\n
\n <h4 className="font-medium text-gray-900 mb-2">Location\n <div className="text-gray-600">\n {selectedReport.location?.address &&

{selectedReport.location.address}

}\n {selectedReport.location?.building &&

Building: {selectedReport.location.building}

}\n {selectedReport.location?.floor &&

Floor: {selectedReport.location.floor}

}\n {!selectedReport.location?.address && !selectedReport.location?.building && !selectedReport.location?.floor && (\n <p className="text-gray-500">Location not specified

\n)}\n
\n \n\n {selectedReport.attachments && selectedReport.attachments.length > 0 && (\n
\n <h4 className="font-medium text-gray-900 mb-2">Attachments\n <div className="space-y-2">\n {selectedReport.attachments.map((file, index) => (\n <div key={index} className="flex items-center space-x-2 text-sm text-gray-600">\n <DocumentIcon className="h-4 w-4" />\n {file.originalName || file.filename || 'Unknown file'}\n
*\n))}\n \n \n)}"}, {"file_name": "Campus-Shield/docs/TECH_STACK_AND_WORKFLOW.md", "file_path": "https://github.com/sparrowdex/Campus-Shield/blob/main/docs/TECH_STACK_AND_WORKFLOW.md", "markdown_link": "- Campus-Shield/docs/TECH_STACK_AND_WORKFLOW.md\n", "code_chunk": "# CampusShield Tech Stack and Workflow Documentation\n\n## Recommended Tech Stack\n\n| Layer | Technology Options | Notes |\n|----------------------|---|-- |\n| **Front-End** | React.js, React Native (mobile), Flutter (mobile) | For web/mobile apps; Flutter enables true cross-platform |\n| **Back-End/API** | Node.js (Express.js), Python (FastAPI or Django) | Scalable REST APIs, real-time features |\n| **Database** | MongoDB (NoSQL), PostgreSQL (SQL) | MongoDB for flexible data; PostgreSQL for relational data |\n| **Authentication** | Firebase Auth, Auth0, or custom JWT-based auth | Secure sign-in, supports anonymity and OAuth |\n| **Notifications** | Firebase Cloud Messaging (push), Twilio (SMS), SendGrid (email) | Real-time and multi-channel notifications |\n| **AI/ML Integration**| Python (scikit-learn, Hugging Face Transformers, spaCy) via an API microservice | For categorization, sentiment analysis, NLP |\n| **Chat/Real-Time** | Socket.io (Node.js), WebSockets, or Firebase Realtime Database | For admin-user anonymous chat, group support |\n| **Maps/Heatmaps** | Google Maps API, Mapbox, Leaflet.js | For live incident heatmaps |\n| **File Storage** | AWS S3, Google Cloud Storage, Firebase Storage | For reports with photos, voice, or video |\n| **Admin Dashboard** | React (web-based), Chart.js/D3.js for analytics and visualizations | Data visualization and report management |\n| **Hosting/Infra** | AWS, Google Cloud Platform, Azure, Vercel, Heroku | Scalable and easy deployment |\n| **Security** | HTTPS/SSL, end-to-end encryption (Signal Protocol, custom), privacy libraries | To ensure report privacy and anonymous chat |\n| **Localization** | i18next, Google Cloud Translation | For multilingual support |\n\n## MVP Tech Stack (Phase 1)\n\nFor the initial MVP, we'll use a simplified but scalable stack:\n\n- **Frontend**: React.js with Tailwind CSS\n- **Backend**: Node.js with Express.js\n- **Database**: MongoDB*

(flexible schema for reports)\n- **Real-time**: Socket.io for chat and live updates\n- **Authentication**: JWT-based with anonymous options\n- **Maps**: Leaflet.js for heatmap visualization\n- **File Storage**: Local storage initially, cloud storage later\n- **AI/ML**: Basic text classification using natural language processing\n\n## Suggested Workflow\n\n### 1. User Onboarding & Authentication\n- Users sign up with minimal data, choose anonymity (no personal info required).\n- Optional: Offer OAuth (Google, college email) for added features with clear privacy messaging."}, {"file_name": "Campus-Shield/README.md", "file_path": "https://github.com/sparrowdex/Campus-Shield/blob/main/README.md", "markdown_link": "- Campus-Shield/README.md\n", "code_chunk": "# CampusShield\n\nCampusShield is a privacy-first campus safety platform for anonymous incident reporting, real-time chat, and admin management. Built for hackathons and real-world impact.\n\n---\n\n## \ud83d\ude80 Features\n\n- **Anonymous Incident Reporting**: Students can report safety incidents without revealing their identity.\n- **Real-time Chat**: Secure, role-based chat between users and campus security/admins.\n- **Role-based Access**: User, Admin, and Moderator roles with custom dashboards and permissions.\n- **Admin Dashboard**: Manage reports, view analytics, assign/resolve cases, and monitor campus safety.\n- **Incident Heatmap**: Visualize incident locations and patterns with Leaflet.js.\n- **AI-Powered Categorization**: Automatic classification and prioritization of reports.\n- **Notifications**: (Pluggable) Real-time in-app notifications for new messages, assignments, and status changes.\n- **Mobile Responsive**: Usable on desktop and mobile devices.\n- **Security & Privacy**: JWT authentication, minimal data collection, and strong privacy defaults.\n\n---\n\n## \ud83d\udee0\ufe0f Tech Stack\n\n- **Frontend**: React, TypeScript, Tailwind CSS\n- **Backend**: Node.js, Express.js\n- **Database**: MongoDB, Mongoose\n- **Real-time**: Socket.IO\n- **Maps**: Leaflet.js\n- **Authentication**: JWT (JSON Web Tokens)\n\n---\n\n## \ud83e\uddd1\u200d\ud83d\udcbb Demo/Test Accounts\n\n- **Admin** \n Email: `admin1@example.com` \n Password: `adminpassword1`\n\n- **Moderator** \n Email: `moderator1@example.com` \n Password: `moderatorpassword1`\n\n- **User** \n Register a new account or use anonymous login.\n Email: `user@example.com`\n Password: `userpassword`\n\n---\n\n## \u26a1 Quick Start\n\n1. **Clone the repo:**\n `bash\n git clone https://github.com/yourusername/campus-shield.git\n cd campus-shield\n` \n2. **Install dependencies:**\n `bash\n cd server && npm install\n cd ../client && npm install\n` \n3. **Set up environment variables:**\n - Copy `.env.example` to `.env` in both `server/` and `client/` if needed.\n4. **Start MongoDB locally (or use Atlas).**\n5. **Start the backend:**\n `bash\n cd server && npm start\n` \n6. **Start the frontend:**\n `bash\n cd ../client && npm start\n` \n7. **Open http://localhost:3000 to view the app.**\n\n---\n\n## \ud83d\udcf1 Mobile & Responsiveness\n- The UI is responsive and works on mobile and desktop.\n- For best results, test in Chrome DevTools mobile view.\n\n---\n\n## \ud83d\udca1 Why We Built This (Impact)\n\n- **Problem:** Students often hesitate to report safety incidents due to privacy concerns and lack of trust.\n- **Solution:** CampusShield enables anonymous, secure reporting and real-time support, empowering students and improving campus safety.\n- **Impact:** More reports, faster admin response, and a safer, more connected campus community.\n\n---\n\n## \ud83d\udce3 Notifications (Pluggable)\n- In-app notification bar for new chat messages, assignments, and status changes (see below for integration instructions).\n- (Optional) Email notifications can be added with Nodemailer.\n\n---\n\n## \ud83d\udcc2 Project Structure\n\n`\n\u251c\u2500\u2500 client/ # React frontend\n\u251c\u2500\u2500 server/ # Node.js backend\n\u251c\u2500\u2500 docs/ # Documentation\n\u2514\u2500\u2500 scripts/ # Utility scripts\n`\n\n---\n\n---\n\n## Setup\n\nFor detailed setup instructions, see SETUP_GUIDE.md.\n\n---"}, {"file_name": "Campus-Shield/server/routes/auth.js", "file_path": "https://github.com/sparrowdex/Campus-Shield/blob/main/server/routes/auth.js", "markdown_link": "- Campus-Shield/server/routes/auth.js\n", "code_chunk": "const express = require('express');\nconst bcrypt = require('bcryptjs');\nconst jwt = require('jsonwebtoken');\nconst { v4: uuidv4 } = require('uuid');\nconst { body, validationResult } = require('express-validator');\nconst User = require('../models/User');\nconst auth = require('../middleware/auth');\nconst memoryStore = require('../services/memoryStore');\nconst AdminRequest =

require('../models/AdminRequest');\n\nconst router = express.Router();\n\n// Add request logging middleware at the top of the file\nrouter.use((req, res, next) => {\n console.log(`[${req.method}] ${req.originalUrl} - Body:`, req.body);\n next();\n});\n\n// Check if MongoDB is connected\nconst isMongoConnected = () => {\n return User.db.readyState === 1;\n};\n\n// Generate JWT token\nconst generateToken = (userId, role) => {\n return jwt.sign(\n { userId, role },\n process.env.JWT_SECRET || 'your-secret-key',\n { expiresIn: '7d' }\n);\n};\n\n// @route POST /api/auth/register\n// @desc Register a new user (optional email/password)\n// @access Public\nrouter.post('/register', [\n body('email').optional().isEmail().withMessage('Please enter a valid email'),\n body('password').optional().isLength({ min: 6 }).withMessage('Password must be at least 6 characters'),\n body('campusId').optional().isString().withMessage('Campus ID must be a string')\n], async (req, res) => {\n try {\n const errors = validationResult(req);\n if (!errors.isEmpty()) {\n return res.status(400).json({ success: false, errors: errors.array() });\n }\n\n const { email, password, campusId } = req.body;\n \n // Generate anonymous ID\n const anonymousId = uuidv4();\n \n // Use MongoDB if available, otherwise use memory store\n if (isMongoConnected()) {\n // Check if email already exists (if provided)\n if (email) {\n const existingUser = await User.findOne({ email });\n if (existingUser) {\n return res.status(400).json({\n success: false,\n message: 'User with this email already exists'\n });\n }\n }\n\n // Create user\n const userData = {\n anonymousId,\n campusId,\n isAnonymous: !email // Anonymous if no email provided\n };\n\n if (email) {\n userData.email = email;\n }\n\n if (password) {\n const salt = await bcrypt.genSalt(10);\n userData.password = await bcrypt.hash(password, salt);\n }\n\n const user = await User.create(userData);\n\n // Generate token\n const token = generateToken(user._id, user.role);\n\n res.status(201).json({\n success: true,\n token,\n user: {\n id: user._id,\n anonymousId: user.anonymousId,\n email: user.email,\n role: user.role,\n isAnonymous: user.isAnonymous,\n campusId: user.campusId\n }\n });\n } else {\n // Use memory store\n if (email) {\n const existingUser = memoryStore.findUserByEmail(email);\n if (existingUser) {\n return res.status(400).json({\n success: false,\n message: 'User with this email already exists'\n });\n }\n }\n\n // Create user in memory store\n const userData = {\n anonymousId,\n campusId,\n isAnonymous: !email,\n email: email || null,\n password: password ? await bcrypt.hash(password, 10) : null,\n role: 'user'\n };\n\n const user = memoryStore.createUser(userData);\n\n // Generate token\n const token = generateToken(user.id, user.role);\n\n res.status(201).json({\n success: true,\n token,\n user: {\n id: user.id,\n anonymousId: user.anonymousId,\n email: user.email,\n role: user.role,\n isAnonymous: user.isAnonymous,\n campusId: user.campusId\n }\n });\n }"}, {"file_name": "Campus-Shield/client/src/pages/RequestAdmin.tsx", "file_path": "https://github.com/sparrowdex/Campus-Shield/blob/main/client/src/pages/RequestAdmin.tsx", "markdown_link": "- Campus-Shield/client/src/pages/RequestAdmin.tsx\n", "code_chunk": "<div className="bg-blue-50 border border-blue-200 rounded-lg p-4 mb-6">\n <div className="flex">\n <ShieldCheckIcon className="h-5 w-5 text-blue-400 mt-0.5" />\n <div className="ml-3">\n <h3 className="text-sm font-medium text-blue-800">Important Information\n <div className="mt-2 text-sm text-blue-700">\n <ul className="list-disc list-inside space-y-1">\n

- *Admin access is granted only to authorized campus personnel*

\n

- *Your request will be reviewed by existing administrators only*

\n

- *Only pre-approved admins can approve new admin requests*

\n

- *You will be notified of the decision via email*

\n

- *Please provide a detailed reason for your request*

*\n \n \n \n \n \n\n <form onSubmit={handleSubmit} className="space-y-6">\n {/ Personal Information */}\n <div className="bg-gray-50 p-4 rounded-lg">\n <h3 className="text-lg font-medium text-gray-900 mb-

4">Personal Information\n <div className="grid grid-cols-1 md:grid-cols-2 gap-4">\n
\n <label htmlFor="role" className="form-label">\n Your Role/Position *\n \n <input\n id="role"\n name="role"\n type="text"\n required\n value={formData.role}\n onChange={handleInputChange}\n className="input-field"\n placeholder="e.g., Security Officer, IT Manager, Dean"\n />\n
\n
\n <label htmlFor="department" className="form-label">\n Department/Unit \n \n <input\n id="department"\n name="department"\n type="text"\n required\n value={formData.department}\n onChange={handleInputChange}\n className="input-field"\n placeholder="e.g., Campus Security, IT Services, Student Affairs"\n />\n
\n \n \n\n {/ Experience & Qualifications */}\n <div className="bg-gray-50 p-4 rounded-lg">\n <h3 className="text-lg font-medium text-gray-900 mb-4">Experience & Qualifications\n
\n <label htmlFor="experience" className="form-label">\n Relevant Experience \n \n <textarea\n id="experience"\n name="experience"\n rows={3}\n required\n value={formData.experience}\n onChange={handleInputChange}\n className="input-field"\n placeholder="Describe your experience with campus safety, incident management, or administrative systems..."\n />\n
\n \n\n {/ Responsibilities */}\n <div className="bg-gray-50 p-4 rounded-lg">\n <h3 className="text-lg font-medium text-gray-900 mb-4">Responsibilities & Duties\n
\n <label htmlFor="responsibilities" className="form-label">\n Current Responsibilities *\n \n <textarea\n id="responsibilities"\n name="responsibilities"\n rows={3}\n required\n value={formData.responsibilities}\n onChange={handleInputChange}\n className="input-field"\n placeholder="Describe your current responsibilities that would benefit from admin access..."\n />\n
\n "}, {"file_name": "Campus-Shield/env.example", "file_path": "https://github.com/sparrowdex/Campus-Shield/blob/main/env.example", "markdown_link": "- Campus-Shield/env.example\n", "code_chunk": "# CampusShield Environment Configuration\n\n# Server Configuration\nNODE_ENV=development\nPORT=5000\nCORS_ORIGIN=http://localhost:3000\n\n# Database Configuration\nMONGODB_URI=mongodb://localhost:27017/campusshield\n\n# JWT Configuration\nJWT_SECRET=your-super-secret-jwt-key-change-this-in-production\n\n# Rate Limiting\nRATE_LIMIT_WINDOW=900000\nRATE_LIMIT_MAX_REQUESTS=100\n\n# File Upload Configuration\nMAX_FILE_SIZE=10485760\nUPLOAD_PATH=uploads\n\n# Security Configuration\nBCRYPT_ROUNDS=12\n\n# AI/ML Service Configuration (for future integration)\nAI_SERVICE_URL=\nAI_SERVICE_KEY=\n\n# Notification Services (for future integration)\nFIREBASE_PROJECT_ID=\nFIREBASE_PRIVATE_KEY=\nFIREBASE_CLIENT_EMAIL=\n\nTWILIO_ACCOUNT_SID=\nTWILIO_AUTH_TOKEN=\nTWILIO_PHONE_NUMBER=\n\nSENDGRID_API_KEY=\nSENDGRID_FROM_EMAIL=\n\n# Maps Configuration\nGOOGLE_MAPS_API_KEY=\nMAPBOX_ACCESS_TOKEN=\n\n# File Storage (for future integration)\nAWS_ACCESS_KEY_ID=\nAWS_SECRET_ACCESS_KEY=\nAWS_REGION=\nAWS_S3_BUCKET=\n\n# Logging Configuration\nLOG_LEVEL=info\nLOG_FILE=logs/app.log\n\n# Data Retention (in days)\nUSER_DATA_RETENTION_DAYS=365\nREPORT_DATA_RETENTION_DAYS=730"}, {"file_name": "Campus-Shield/client/src/pages/AdminDashboard.tsx", "file_path": "https://github.com/sparrowdex/Campus-Shield/blob/main/client/src/pages/AdminDashboard.tsx", "markdown_link": "- Campus-Shield/client/src/pages/AdminDashboard.tsx\n", "code_chunk": "useEffect(() => {\n fetchDashboardData();\n }, []);\n\n const fetchDashboardData = async () => {\n try {\n const [statsRes, activeChatsRes] = await Promise.all([\n fetch(`${process.env.REACT_APP_API_URL}/api/admin/stats`, {\n headers: {\n 'Authorization': `Bearer ${localStorage.getItem('token')}`\n }\n }),\n fetch(`${process.env.REACT_APP_API_URL}/api/admin/active-chats`, {\n headers: {\n 'Authorization': `Bearer ${localStorage.getItem('token')}`\n }\n })\n]);\n if (statsRes.ok) {\n const statsData = await

statsRes.json();\n setStats(statsData.stats);\n }\n if (activeChatsRes.ok) {\n const activeChatsData = await activeChatsRes.json();\n setActiveChats(activeChatsData.activeChats || 0);\n }\n\n // Fetch all reports\n const reportsResponse = await fetch(`${process.env.REACT_APP_API_URL}/api/admin/reports`, {\n headers: {\n 'Authorization': `Bearer ${localStorage.getItem('token')}`\n }\n });\n\n if (reportsResponse.ok) {\n const reportsData = await reportsResponse.json();\n setReports(reportsData.reports || []);\n }\n\n // Fetch heatmap data\n const heatmapResponse = await fetch(`${process.env.REACT_APP_API_URL}/api/reports/heatmap/data`, {\n headers: {\n 'Authorization': `Bearer ${localStorage.getItem('token')}`\n }\n });\n\n if (heatmapResponse.ok) {\n const heatmapData = await heatmapResponse.json();\n setHeatmapData(heatmapData.heatmapData || []);\n }\n\n } catch (err: any) {\n setError(err.message);\n } finally {\n setLoading(false);\n }\n };\n\n const updateReportStatus = async (reportId: string, newStatus: string) => {\n try {\n const response = await fetch(`${process.env.REACT_APP_API_URL}/api/admin/reports/${reportId}/status`, {\n method: 'PATCH',\n headers: {\n 'Authorization': `Bearer ${localStorage.getItem('token')}`,\n 'Content-Type': 'application/json'\n },\n body: JSON.stringify({ status: newStatus })\n });\n\n if (response.ok) {\n // Update the report in the local state\n setReports(prev => prev.map(report => \n report.id === reportId \n ? { ...report, status: newStatus }\n : report\n));\n \n // Refresh stats\n fetchDashboardData();\n }\n } catch (err: any) {\n setError(err.message);\n }\n };\n\n const filteredReports = reports.filter(report => {\n const matchesStatus = filter === 'all' || report.status === filter;\n const matchesPriority = priorityFilter === 'all' || report.priority === priorityFilter;\n const matchesCategory = categoryFilter === 'all' || report.category === categoryFilter;\n return matchesStatus && matchesPriority && matchesCategory;\n });\n\n const formatDate = (dateString: string) => {\n return new Date(dateString).toLocaleDateString('en-US', {\n year: 'numeric',\n month: 'short',\n day: 'numeric',\n hour: '2-digit',\n minute: '2-digit'\n });\n };\n\n const getCategoryStats = () => {\n const categoryCounts: { [key: string]: number } = {};\n reports.forEach(report => {\n const category = categoryLabels[report.category as keyof typeof categoryLabels] || report.category;\n categoryCounts[category] = (categoryCounts[category] || 0) + 1;\n });\n return categoryCounts;\n };\n\n if (loading) {\n return (\n <div className="max-w-7xl mx-auto">\n <div className="card">\n <LoadingSpinner text="Loading admin dashboard..." />\n \n \n);\n }\n\n const assignedToId =\n selectedReport && selectedReport.assignedTo\n ? isAssignedToObject(selectedReport.assignedTo)\n ? selectedReport.assignedTo._id\n : selectedReport.assignedTo\n : null;"}]

What's Next

Continue exploring the documentation with these detailed sections:

- **Getting Started**: Getting Started
- **Architecture Overview**: Architecture Overview
- **Core Technologies**: Core Technologies