# Access Request System

Comprehensive guide to the Access Request system implemented in the RAS Dashboard API, providing complete workflow management from user request submission to admin approval/rejection with automated notifications and user account creation.

## 🎯 Overview

The Access Request system provides: - **Public Request Submission** - No authentication required for initial requests - **Admin Approval Workflow** - Complete admin review and decision process - **Automatic User Creation** - User accounts created upon approval - **Notification Integration** - Email and in-app notifications throughout workflow - **Comprehensive Analytics** - Request statistics and trend analysis - **Audit Trail** - Complete tracking of all decisions and actions

## 🏗️ Database Schema

### Core Table

-- Access Requests: User access request records  
access\_requests (id, first\_name, last\_name, email, status, reason,   
 rejection\_reason, processed\_at, processed\_by,   
 created\_at, updated\_at)

### Relationships

Users ←→ AccessRequests (processed\_by)  
AccessRequests → Users (created upon approval)

### Status Values

const ACCESS\_REQUEST\_STATUSES = [  
 'pending', // Awaiting admin review  
 'approved', // Approved by admin, user account created  
 'rejected' // Rejected by admin with reason  
];

## 🔄 Complete Workflow

### 1. Request Submission (Public)

User submits request → Validation → Duplicate check → Save to database →   
Send confirmation email → Notify admins → Return success response

### 2. Admin Review

Admin views requests → Filter/search → Select request → Review details →   
Make decision (approve/reject)

### 3. Approval Process

Admin approves → Update status → Create user account → Send welcome email →   
Create in-app notification → Return success response

### 4. Rejection Process

Admin rejects → Update status → Add rejection reason → Send rejection email →   
Return success response

## 📝 Request Structure

### Request Data

const accessRequest = {  
 firstName: 'Jane',  
 lastName: 'Smith',  
 email: 'jane.smith@example.com',  
 reason: 'I need access to the security dashboard to review vulnerability reports for my department. I am the IT Security Manager and require this access to perform my daily responsibilities.',  
 status: 'pending',  
 createdAt: '2024-01-15T10:30:00Z',  
 updatedAt: '2024-01-15T10:30:00Z'  
};

### Processed Request

const processedRequest = {  
 id: 123,  
 firstName: 'Jane',  
 lastName: 'Smith',  
 email: 'jane.smith@example.com',  
 status: 'approved',  
 reason: 'IT Security Manager requiring dashboard access...',  
 rejectionReason: null,  
 processedAt: '2024-01-15T14:45:00Z',  
 processedBy: 456, // Admin user ID  
 createdAt: '2024-01-15T10:30:00Z',  
 updatedAt: '2024-01-15T14:45:00Z',  
 processedByName: 'Admin',  
 processedByLastName: 'User',  
 processedByEmail: 'admin@example.com'  
};

## 🚀 API Endpoints

### Public Endpoints

// No authentication required  
POST /api/v1/access-requests/submit // Submit access request

### Admin Endpoints

// Authentication and admin permissions required  
GET /api/v1/access-requests // Get all requests with filtering  
GET /api/v1/access-requests/:id // Get request by ID  
PATCH /api/v1/access-requests/:id/approve // Approve request  
PATCH /api/v1/access-requests/:id/reject // Reject request  
DELETE /api/v1/access-requests/:id // Delete request  
GET /api/v1/access-requests/stats // Get statistics

## 🛠️ Usage Examples

### Submitting a Request (Public)

// No authentication required  
const requestData = {  
 firstName: 'John',  
 lastName: 'Doe',  
 email: 'john.doe@example.com',  
 reason: 'I need access to review security reports for compliance auditing purposes.'  
};  
  
const response = await fetch('/api/v1/access-requests/submit', {  
 method: 'POST',  
 headers: {  
 'Content-Type': 'application/json'  
 },  
 body: JSON.stringify(requestData)  
});  
  
const result = await response.json();  
console.log('Request submitted:', result.data.id);

### Admin: Getting All Requests

// Admin authentication required  
const response = await fetch('/api/v1/access-requests?status=pending&page=1&limit=20', {  
 headers: {  
 'Authorization': `Bearer ${adminToken}`,  
 'Content-Type': 'application/json'  
 }  
});  
  
const requests = await response.json();  
console.log(`Found ${requests.pagination.totalCount} requests`);

### Admin: Approving a Request

// Admin authentication required  
const response = await fetch(`/api/v1/access-requests/${requestId}/approve`, {  
 method: 'PATCH',  
 headers: {  
 'Authorization': `Bearer ${adminToken}`,  
 'Content-Type': 'application/json'  
 }  
});  
  
const approvedRequest = await response.json();  
console.log('Request approved, user account created');

### Admin: Rejecting a Request

// Admin authentication required  
const rejectionData = {  
 rejectionReason: 'Insufficient business justification provided. Please provide more details about your role and specific access requirements.'  
};  
  
const response = await fetch(`/api/v1/access-requests/${requestId}/reject`, {  
 method: 'PATCH',  
 headers: {  
 'Authorization': `Bearer ${adminToken}`,  
 'Content-Type': 'application/json'  
 },  
 body: JSON.stringify(rejectionData)  
});  
  
const rejectedRequest = await response.json();  
console.log('Request rejected with reason');

### Admin: Getting Statistics

// Admin authentication required  
const response = await fetch('/api/v1/access-requests/stats', {  
 headers: {  
 'Authorization': `Bearer ${adminToken}`,  
 'Content-Type': 'application/json'  
 }  
});  
  
const stats = await response.json();  
console.log('Overall stats:', stats.data.overall);  
console.log('Recent activity:', stats.data.recent);  
console.log('Monthly trends:', stats.data.monthly);

## 📧 Notification System Integration

### Email Notifications

#### 1. Admin Notification (New Request)

<h2>New Access Request</h2>  
<p>A new access request has been submitted and requires your review:</p>  
<ul>  
 <li><strong>Name:</strong> Jane Smith</li>  
 <li><strong>Email:</strong> jane.smith@example.com</li>  
 <li><strong>Reason:</strong> IT Security Manager requiring dashboard access...</li>  
 <li><strong>Submitted:</strong> January 15, 2024 at 10:30 AM</li>  
</ul>  
<p>Please review and process this request in the admin panel.</p>

#### 2. Confirmation Email (To Requester)

<h2>Access Request Received</h2>  
<p>Dear Jane Smith,</p>  
<p>Thank you for your interest in accessing our system. We have received your access request and it is currently being reviewed by our administrators.</p>  
<p>You will receive an email notification once your request has been processed.</p>  
<p>Best regards,<br>System Administration Team</p>

#### 3. Approval Email (To Requester)

<h2>Access Request Approved</h2>  
<p>Dear Jane Smith,</p>  
<p>Great news! Your access request has been approved and your account has been created.</p>  
<p>You can now log in to the system using your email address. If you haven't set a password yet, please use the password reset feature on the login page.</p>  
<p>Welcome to the system!</p>

#### 4. Rejection Email (To Requester)

<h2>Access Request Update</h2>  
<p>Dear Jane Smith,</p>  
<p>Thank you for your interest in accessing our system. After careful review, we are unable to approve your access request at this time.</p>  
<p><strong>Reason:</strong> Insufficient business justification provided...</p>  
<p>If you believe this decision was made in error or if your circumstances have changed, please feel free to submit a new request.</p>

### In-App Notifications

#### Admin Notifications

// Notification sent to all admins when new request is submitted  
const adminNotification = {  
 title: 'New Access Request',  
 message: 'Jane Smith (jane.smith@example.com) has requested access to the system.',  
 type: 'info',  
 module: 'access\_requests',  
 eventType: 'new\_request',  
 priority: 2,  
 metadata: {  
 requesterName: 'Jane Smith',  
 requesterEmail: 'jane.smith@example.com',  
 requestId: 123  
 }  
};

#### User Notifications (Upon Approval)

// Notification sent to approved user  
const userNotification = {  
 title: 'Welcome! Your Access Request Has Been Approved',  
 message: 'Your access request has been approved and your account is now active. Welcome to the system!',  
 type: 'success',  
 module: 'access\_requests',  
 eventType: 'request\_approved',  
 priority: 2,  
 metadata: {  
 approvedAt: '2024-01-15T14:45:00Z',  
 processedBy: 456  
 }  
};

## 🔍 Advanced Filtering

### Filter Options

const filterOptions = {  
 status: 'pending', // Filter by status  
 search: 'jane', // Search in name, email, reason  
 startDate: '2024-01-01', // Filter from date  
 endDate: '2024-01-31', // Filter to date  
 processedBy: 456, // Filter by processing admin  
 page: 1, // Pagination  
 limit: 20, // Results per page  
 sortBy: 'createdAt', // Sort field  
 sortOrder: 'desc' // Sort direction  
};

### Search Functionality

// Search across multiple fields  
const searchQuery = 'security manager';  
// Searches in: firstName, lastName, email, reason  
  
const results = await accessRequestService.getAllAccessRequests({  
 search: searchQuery  
}, {  
 page: 1,  
 limit: 10  
});

## 📊 Analytics and Reporting

### Overall Statistics

const overallStats = {  
 total: 1250, // Total requests ever submitted  
 pending: 45, // Currently pending review  
 approved: 980, // Total approved requests  
 rejected: 225 // Total rejected requests  
};

### Monthly Trends

const monthlyTrends = [  
 {  
 month: '2024-01-01T00:00:00Z',  
 total: 85,  
 pending: 12,  
 approved: 65,  
 rejected: 8  
 },  
 {  
 month: '2023-12-01T00:00:00Z',  
 total: 92,  
 pending: 0,  
 approved: 78,  
 rejected: 14  
 }  
 // ... last 12 months  
];

### Recent Activity (Last 30 Days)

const recentActivity = {  
 total: 45, // Requests in last 30 days  
 pending: 12, // Still pending  
 approved: 28, // Approved in last 30 days  
 rejected: 5 // Rejected in last 30 days  
};

## ⚡ Performance Optimization

### Duplicate Prevention

// Prevent multiple pending requests from same email  
const existingRequest = await db.select()  
 .from(accessRequests)  
 .where(and(  
 eq(accessRequests.email, requestData.email),  
 eq(accessRequests.status, 'pending')  
 ))  
 .limit(1);  
  
if (existingRequest.length > 0) {  
 throw new Error('A pending access request already exists for this email address');  
}

### Efficient Querying

// Use indexes for common queries  
const indexes = [  
 'access\_requests\_email\_idx', // For duplicate checking  
 'access\_requests\_status\_idx', // For status filtering  
 'access\_requests\_created\_at\_idx', // For date filtering  
 'access\_requests\_processed\_by\_idx', // For admin filtering  
 'access\_requests\_status\_created\_idx' // Composite for admin dashboard  
];

### Pagination Strategy

// Efficient pagination for large datasets  
const paginatedRequests = await db.select()  
 .from(accessRequests)  
 .where(conditions)  
 .orderBy(desc(accessRequests.createdAt))  
 .limit(limit)  
 .offset((page - 1) \* limit);

## 🎯 Best Practices

### 1. Request Validation

// Comprehensive validation  
const requestSchema = Joi.object({  
 firstName: Joi.string().required().max(100).trim(),  
 lastName: Joi.string().required().max(100).trim(),  
 email: Joi.string().email().required().max(255).lowercase().trim(),  
 reason: Joi.string().max(1000).trim()  
});  
  
// Good: Detailed, business-justified request  
const goodRequest = {  
 firstName: 'Jane',  
 lastName: 'Smith',  
 email: 'jane.smith@company.com',  
 reason: 'I am the IT Security Manager and need access to the vulnerability dashboard to review security reports and coordinate remediation efforts with my team.'  
};  
  
// Avoid: Vague, insufficient requests  
const badRequest = {  
 firstName: 'John',  
 lastName: 'Doe',  
 email: 'john@email.com',  
 reason: 'I need access'  
};

### 2. Admin Decision Guidelines

// Approval criteria  
const approvalCriteria = {  
 businessJustification: 'Clear business need stated',  
 roleAlignment: 'Role matches access requirements',  
 contactVerification: 'Valid business email domain',  
 securityClearance: 'Appropriate for requested access level'  
};  
  
// Rejection reasons  
const commonRejectionReasons = [  
 'Insufficient business justification provided',  
 'Role does not require system access',  
 'Invalid or personal email address',  
 'Duplicate request - please use existing account',  
 'Access level not appropriate for stated role'  
];

### 3. Communication Standards

// Professional email templates  
const emailStandards = {  
 tone: 'Professional and helpful',  
 clarity: 'Clear next steps provided',  
 branding: 'Consistent with organization branding',  
 responsiveness: 'Timely responses within 24-48 hours'  
};

### 4. Security Considerations

// Security best practices  
const securityMeasures = {  
 emailValidation: 'Verify business email domains',  
 duplicatePrevention: 'Prevent spam and duplicate requests',  
 auditTrail: 'Complete logging of all decisions',  
 accessReview: 'Regular review of approved accounts',  
 dataRetention: 'Appropriate retention of request records'  
};

## 🔧 Troubleshooting

### Common Issues

#### 1. Duplicate Request Error

// Error: "A pending access request already exists for this email address"  
// Solution: Check for existing pending requests  
const existingRequests = await accessRequestService.getAllAccessRequests({  
 search: userEmail,  
 status: 'pending'  
});  
  
if (existingRequests.data.length > 0) {  
 console.log('Existing pending request found:', existingRequests.data[0]);  
}

#### 2. Email Delivery Issues

// Check email service configuration  
const emailConfig = {  
 smtp\_host: process.env.SMTP\_HOST,  
 smtp\_port: process.env.SMTP\_PORT,  
 smtp\_user: process.env.SMTP\_USER,  
 smtp\_pass: process.env.SMTP\_PASS  
};  
  
// Verify all required environment variables are set  
Object.entries(emailConfig).forEach(([key, value]) => {  
 if (!value) {  
 console.error(`Missing email configuration: ${key}`);  
 }  
});

#### 3. User Account Creation Issues

// Debug user creation process  
try {  
 const newUser = await accessRequestService.createUserFromApprovedRequest(request);  
 console.log('User created successfully:', newUser.email);  
} catch (error) {  
 console.error('User creation failed:', error.message);  
 // Check for existing users, validation errors, database constraints  
}

#### 4. Permission Issues

// Verify admin permissions  
const hasPermission = await permissionService.checkPermission(  
 userId,  
 'access\_requests',  
 'admin'  
);  
  
if (!hasPermission) {  
 console.error('User lacks admin permissions for access requests');  
}

## 🚀 Advanced Features

### 1. Bulk Operations

// Process multiple requests efficiently  
const bulkApproval = async (requestIds, adminUserId) => {  
 const results = [];  
  
 for (const requestId of requestIds) {  
 try {  
 const result = await accessRequestService.approveAccessRequest(requestId, adminUserId);  
 results.push({ requestId, status: 'approved', result });  
 } catch (error) {  
 results.push({ requestId, status: 'failed', error: error.message });  
 }  
 }  
  
 return results;  
};

### 2. Custom Approval Workflows

// Multi-step approval process  
const approvalWorkflow = {  
 steps: [  
 { role: 'manager', required: true },  
 { role: 'security', required: true },  
 { role: 'admin', required: true }  
 ],  
 autoApprove: {  
 conditions: ['internal\_email', 'known\_department'],  
 skipSteps: ['manager']  
 }  
};

### 3. Integration with HR Systems

// Verify employee status  
const hrIntegration = {  
 verifyEmployee: async (email) => {  
 // Check against HR database  
 const employee = await hrService.findByEmail(email);  
 return {  
 isEmployee: !!employee,  
 department: employee?.department,  
 manager: employee?.manager,  
 startDate: employee?.startDate  
 };  
 }  
};

### 4. Automated Reminders

// Send reminders for pending requests  
const reminderSchedule = {  
 adminReminders: [  
 { after: '24 hours', message: 'Pending access request requires review' },  
 { after: '72 hours', message: 'Urgent: Access request overdue for review' }  
 ],  
 requesterUpdates: [  
 { after: '48 hours', message: 'Your access request is being reviewed' },  
 { after: '1 week', message: 'Access request status update' }  
 ]  
};

## 📋 Compliance and Auditing

### Audit Trail

// Complete audit trail for compliance  
const auditRecord = {  
 requestId: 123,  
 action: 'approved',  
 performedBy: 456,  
 performedAt: '2024-01-15T14:45:00Z',  
 previousStatus: 'pending',  
 newStatus: 'approved',  
 reason: 'Valid business justification provided',  
 ipAddress: '192.168.1.100',  
 userAgent: 'Mozilla/5.0...',  
 metadata: {  
 userCreated: true,  
 notificationsSent: ['email', 'in\_app'],  
 approvalTime: '4 hours 15 minutes'  
 }  
};

### Compliance Reporting

// Generate compliance reports  
const complianceReport = {  
 period: '2024-Q1',  
 totalRequests: 245,  
 averageProcessingTime: '2.5 days',  
 approvalRate: 0.82,  
 rejectionReasons: {  
 'insufficient\_justification': 28,  
 'invalid\_email': 12,  
 'duplicate\_request': 8,  
 'role\_mismatch': 5  
 },  
 slaCompliance: 0.95 // 95% processed within SLA  
};

This Access Request system provides a complete workflow solution for managing user access requests with professional communication, comprehensive tracking, and enterprise-grade security and compliance features. ```