Property Management Portal System (Modified)

System Overview

A web-based portal connecting tenants and the admin/property owner for streamlined property management, communication, and document handling. The admin serves as both system administrator and property owner.

Core System Requirements

Functional Requirements

User Management

Registration & Authentication

- Admin creates tenant accounts only (no separate property owners)
- Auto-generated secure usernames and passwords for tenants
- Two-level access control (Admin/Property Owner, Tenant)
- Password reset functionality

Communication

Real-time Chat System

- Direct messaging between tenants and admin/property owner
- Message history and notifications
- File sharing capabilities
- Maintenance request system with priority levels

Document Management

Contract Viewing

- Tenants can view their lease contracts
- Document versioning and updates
- Download permissions
- Digital signature capability

Property Management

Property-Tenant Associations

- Admin adds and manages all properties

- Link tenants to specific properties
- Multiple tenants per property support
- Lease creation and management

Non-Functional Requirements

- **Security**: Role-based access, data encryption, secure authentication
- **Performance**: <3 second page load times, real-time messaging
- **Scalability**: Support 1000+ concurrent users
- **Reliability**: 99.5% uptime, automated backups
- **Usability**: Mobile-responsive, intuitive interface

System Architecture

Technology Stack

- **Frontend**: React.js with responsive design
- **Backend**: Node.js with Express.js
- **Database**: PostgreSQL with Redis for caching
- **Real-time**: Socket.io for chat
- **Authentication**: JWT tokens
- **File Storage**: AWS S3 or local storage
- **Deployment**: Docker containers

System Components

- Authentication Service
- User Management Service
- Property Management Service
- Messaging Service
- Document Management Service
- Notification Service

Data Models

User Model

```sql
```

```
users {
  id: UUID PRIMARY KEY
  username: VARCHAR(50) UNIQUE
  email: VARCHAR(100) UNIQUE
  password_hash: VARCHAR(255)
  role: ENUM('admin', 'tenant')
  first_name: VARCHAR(50)
  last_name: VARCHAR(50)
  phone: VARCHAR(20)
  is_active: BOOLEAN DEFAULT true
  created_at: TIMESTAMP
  updated_at: TIMESTAMP
}
```

Property Model

sql

```
properties {
  id: UUID PRIMARY KEY
  owner_id: UUID FOREIGN KEY REFERENCES users(id) -- Always references admin user
  address: TEXT
  property_type: ENUM('apartment', 'house', 'commercial')
  bedrooms: INTEGER
  bathrooms: INTEGER
  square_footage: INTEGER
  rent_amount: DECIMAL(10,2)
  description: TEXT
  is_active: BOOLEAN DEFAULT true
  created_at: TIMESTAMP
  updated_at: TIMESTAMP
}
```

Lease Model

sql

```
leases {
  id: UUID PRIMARY KEY
  tenant_id: UUID FOREIGN KEY REFERENCES users(id)
  property_id: UUID FOREIGN KEY REFERENCES properties(id)
  start_date: DATE
  end_date: DATE
  monthly_rent: DECIMAL(10,2)
  security_deposit: DECIMAL(10,2)
  status: ENUM('active', 'expired', 'terminated')
  created_at: TIMESTAMP
  updated_at: TIMESTAMP
}
```

Document Model

```sql
documents {
  id: UUID PRIMARY KEY
  lease_id: UUID FOREIGN KEY REFERENCES leases(id)
  document_type: ENUM('lease_agreement', 'addendum', 'notice')
  file_name: VARCHAR(255)
  file_path: VARCHAR(500)
  file_size: INTEGER
  mime_type: VARCHAR(100)
  uploaded_by: UUID FOREIGN KEY REFERENCES users(id)
  uploaded_at: TIMESTAMP
}
```

Message Model

```sql
messages {
  id: UUID PRIMARY KEY
  sender_id: UUID FOREIGN KEY REFERENCES users(id)
  recipient_id: UUID FOREIGN KEY REFERENCES users(id)
  property_id: UUID FOREIGN KEY REFERENCES properties(id)
  message_text: TEXT
  attachment_url: VARCHAR(500)
  is_read: BOOLEAN DEFAULT false
  sent_at: TIMESTAMP
}
```

Maintenance Request Model

```sql
maintenance_requests {
  id: UUID PRIMARY KEY
  tenant_id: UUID FOREIGN KEY REFERENCES users(id)
  property_id: UUID FOREIGN KEY REFERENCES properties(id)
  title: VARCHAR(255)
  description: TEXT
  priority: ENUM('low', 'medium', 'high', 'urgent')
  status: ENUM('pending', 'in_progress', 'completed', 'cancelled')
  created_at: TIMESTAMP
  updated_at: TIMESTAMP
}
```

Notification Model

```sql
notifications {
  id: UUID PRIMARY KEY
  user_id: UUID FOREIGN KEY REFERENCES users(id)
  type: ENUM('message', 'document_update', 'lease_expiry', 'maintenance_request')
  title: VARCHAR(255)
  content: TEXT
  is_read: BOOLEAN DEFAULT false
  created_at: TIMESTAMP
}
```

User Interface Design

Admin/Property Owner Dashboard

Features:

- Property portfolio overview
- Tenant registration and management
- Property addition and management
- Lease creation and assignment
- Document upload and management
- Maintenance request tracking
- Tenant communication hub
- System analytics and reporting

Key Pages:

- `/admin/dashboard` - Overview with key metrics and quick actions
- `/admin/properties` - Property management (add, edit, view)
- `/admin/tenants` - Tenant management (add, edit, assign)
- `/admin/leases` - Lease management and assignments
- `/admin/documents` - Document management and uploads
- `/admin/messages` - Communication center with all tenants
- `/admin/maintenance` - Maintenance request management
- `/admin/reports` - Analytics and reporting

Tenant Portal

Features:

- Lease document viewing and download
- Direct messaging with property owner/admin
- Maintenance request submission
- Payment history tracking
- Personal profile management
- Notification center

Key Pages:

- `/tenant/dashboard` - Lease overview and quick actions
- `/tenant/documents` - Contract viewing and download
- `/tenant/messages` - Communication with property owner
- `/tenant/maintenance` - Service request submission and tracking
- `/tenant/profile` - Personal information management

Enhanced Features

Communication System

- **Real-time Chat**: Instant messaging between admin and tenants
- **Message Templates**: Pre-defined messages for common scenarios
- **Broadcast Messages**: Admin can send announcements to all tenants
- **File Sharing**: Share documents and images in conversations

Document Management

- **Digital Signatures**: E-signature capability for lease agreements

- **Version Control**: Track document changes and updates
- **Automatic Notifications**: Alerts when new documents are uploaded
- **Bulk Upload**: Upload multiple documents at once

Property Management

- **Property Photos**: Image galleries for each property
- **Rent Tracking**: Monitor rent payment status
- **Lease Expiry Alerts**: Automatic notifications before lease expiration
- **Vacancy Management**: Track available properties

Maintenance System

- **Priority Levels**: Categorize requests by urgency
- **Status Tracking**: Monitor request progress
- **Photo Attachments**: Visual documentation of issues
- **Vendor Assignment**: Assign maintenance to specific vendors

Security & Compliance

Security Measures

- **Role-based Access Control**: Strict permission system
- **Data Encryption**: Encrypt sensitive data at rest and in transit
- **Input Validation**: Prevent SQL injection and XSS attacks
- **Session Management**: Automatic logout and session timeout
- **Audit Logs**: Track all user actions and system changes

Compliance Features

- **Data Export**: GDPR-compliant data export functionality
- **Data Retention**: Configurable data retention policies
- **Backup Strategy**: Automated daily database and file backups
- **Privacy Controls**: User consent management

Implementation Phases

Phase 1: Core Functionality (MVP)

- Admin user authentication and tenant registration
- Basic property and lease management
- Document viewing capability

- Simple messaging system
- Tenant-property assignments

Phase 2: Enhanced Features

- Real-time chat with notifications
- File sharing in messages
- Maintenance request system
- Mobile responsiveness
- Document upload functionality

Phase 3: Advanced Features

- Digital signatures
- Advanced search and filters
- Analytics dashboard
- Reporting system
- Mobile app

Technical Considerations

Performance Optimization

- **Database Indexing**: Optimize query performance
- **Caching Strategy**: Redis for frequently accessed data
- **Image Optimization**: Compress and optimize property photos
- **Lazy Loading**: Improve page load times

Scalability

- **Microservices Architecture**: Modular system design
- **Load Balancing**: Handle increased user traffic
- **Database Optimization**: Efficient data storage and retrieval
- **CDN Integration**: Fast content delivery

Monitoring & Maintenance

- **Health Checks**: System monitoring and alerts
- **Error Logging**: Comprehensive error tracking
- **Performance Metrics**: Monitor system performance
- **Automated Testing**: Ensure system reliability

This modified system design streamlines property management by consolidating admin and property owner roles, making it ideal for single-owner property management companies or individual landlords managing multiple properties.