**Strategy Document: Property Management Software**

**Role-Based Access Control (RBAC)**

1. Identify Roles

Role-Based Access Control (RBAC) is a fundamental security feature in Azure. It allows you to define fine-grained access permissions based on roles. Here’s how to incorporate RBAC into your Property Management software:

Identify Roles: To start, list the many positions that your property management system may have. Typical roles could be:  
Property managers are in charge of overall oversight and making decisions.  
Maintaining Employees: Manages maintenance, inspections, and repairs.  
Tenants: See payment history, submit requests, and access their lease agreements.  
Administrators: They are in charge of overseeing user accounts, system configuration, and security settings.

2.) Create custom roles.

Despite the fact that Azure comes with pre-built roles (such Owner, Contributor, and Reader), you may also create unique roles that are specific to your application. At the level of a resource group, subscription, or individual resource, define permissions.

3.) Assign Permissions

Allocate distinct authorization to every position. As an illustration:  
Property managers have complete command over all property resources, such as databases, storage accounts, and virtual machines.  
Maintenance Employees: Write access to work orders and read access to maintenance logs.  
Tenants: Access to their lease information and payment history is read-only.  
Administrators: Take charge of audit logs and RBAC assignments.

**4.) Scope Assignments:**

Apply RBAC assignments at the appropriate scope:

Subscription Level: For cross-property management tasks.

Resource Group Level: For specific properties or portfolios.

Resource Level: For individual assets (e.g., a specific building or unit).

5.) Use Azure AD Integration: For easy user authentication and authorization, combine RBAC with Azure Active Directory (Azure AD). This guarantees that the system can only be accessed by authorized users.  
6.) Audit & Observe: Examine audit logs and RBAC assignments on a regular basis. Access trends and any security issues are shown by Azure Security Center and Azure Monitor.

* **Multi-Tenancy**: If managing properties for different clients (tenants), design your RBAC model to support multi-tenancy.
* **Conditional Access Policies**: Implement policies based on conditions (e.g., location, device type) to enhance security.
* **Just-In-Time Access**: Use Azure AD’s Just-In-Time access to limit exposure to privileged roles.

### 2. User Authentication

1. Azure Active Directory (Azure AD):
   * Single Sign-On (SSO): Implement SSO using Azure AD to allow users to sign in once and access multiple applications seamlessly. This enhances user experience and reduces password fatigue.
   * Multi-Factor Authentication (MFA): Enable MFA to add an extra layer of security. Users must provide additional verification (e.g., SMS code, phone call) in addition to their password.
   * Conditional Access Policies: Define policies based on conditions (e.g., location, device type) to control access. For example, restrict access from certain IP ranges or require MFA for specific roles.
2. OAuth 2.0 and OpenID Connect:
   * Use OAuth 2.0 for authorization and token-based authentication. It allows third-party applications (e.g., mobile apps) to securely access your APIs.
   * OpenID Connect builds on OAuth 2.0 and provides identity information about the user. It’s commonly used for web and mobile applications.
3. Managed Identity:
   * For services running within Azure (e.g., virtual machines, Azure Functions), use managed identities. These identities are automatically managed by Azure and allow secure access to other Azure resources.
4. Custom Authentication Providers:
   * If your property management system integrates with external identity providers (e.g., social logins, custom databases), create custom authentication providers.
   * Implement token validation and user profile retrieval.
5. Authorization Code Flow:
   * When users sign in, they receive an authorization code. Exchange this code for an access token using a secure backend service.
   * Validate the token and extract user information.

Token Management:

* Use short-lived access tokens and refresh tokens. Rotate tokens periodically to minimize security risks.
* Store tokens securely (e.g., Azure Key Vault)

Additional Considerations:

* Session Management: Implement session timeouts and secure session handling.
* Logging and Auditing: Log authentication events for auditing and troubleshooting.
* Password Policies: Enforce strong password policies and regular password changes.
* User Self-Service: Provide self-service password reset and account recovery options.

### 3.) Data Encryption

### Data in Transit Encryption

#### a. **Transport Layer Security (TLS)**

* **Implementation**: Enforce TLS for communication between property management applications and clients.
* **Purpose**: TLS ensures that data exchanged between users and the system remains confidential and tamper-proof during transmission.

#### b. **Azure Application Gateway**

* **Role**: Use Azure Application Gateway as a reverse proxy with SSL termination.
* **Function**: It terminates SSL connections, ensuring encrypted traffic between clients and property management services.

### Data at Rest Encryption

#### a. **Azure SQL Database Transparent Data Encryption (TDE)**

* **Configuration**: Enable TDE for Azure SQL databases.
* **Benefit**: Automatically encrypts data at rest, safeguarding sensitive information.

#### b. **Azure Disk Encryption**

* **Usage**: Encrypt virtual machine disks containing property management data.
* **Advantage**: Protects data even if physical disks are compromised.

#### **c. Azure Storage Service Encryption**

* **Scope**: Enable encryption for data stored in Azure Blob Storage, Azure Files, and Azure Queues.
* **Significance**: Ensures data remains encrypted while at rest in Azure storage services.

### Conclusion

By implementing robust data encryption practices, property management software on the Azure platform can achieve the following:

* **Confidentiality**: Sensitive data remains private.
* **Integrity**: Data remains unaltered during storage and transmission.
* **Availability**: Encrypted data is accessible only to authorized users

### 4.) SSL Certificates

### a. What is SSL?

### SSL (Secure Sockets Layer) is a cryptographic protocol that ensures secure communication over the internet.

### It provides encryption and authentication for data exchanged between a client (user’s browser) and a server (property management application).

### b. Why SSL Matters

### Data Encryption: SSL encrypts data during transmission, preventing unauthorized interception.

### Trust and Authentication: SSL certificates validate the identity of the server, building trust with users.

### Implementing SSL Certification

### a. Obtain an SSL Certificate

### Certificate Authorities (CAs) issue SSL certificates.

### Choose an appropriate certificate type:

### Domain Validated (DV): Basic validation, suitable for most property management applications.

### Organization Validated (OV): Validates the organization’s identity.

### Extended Validation (EV): Highest level of validation, displays the organization’s name in the address bar.

### b. Configure Azure Services

### i. Azure App Service

### Enable SSL for your property management web application hosted on Azure App Service:

### Purchase an SSL certificate (either from a CA or Azure Key Vault).

### Bind the certificate to your custom domain in Azure App Service.

### ii. Azure Front Door

### Use Azure Front Door as a global entry point for your application:

### Configure SSL settings in Front Door.

### Route traffic to your backend services (e.g., property management APIs) securely.

### Azure Kubernetes Service (AKS)

### If your property management system runs on AKS:

### Secure communication within the AKS cluster using SSL.

### Use Ingress controllers with SSL termination.

### Best Practices

### a. Regular Certificate Renewal

### SSL certificates have an expiration date. Set up automated renewal processes.

### b. Mixed Content Handling

### Ensure all resources (images, scripts, stylesheets) are served over HTTPS to avoid mixed content warnings.

### c. HSTS (HTTP Strict Transport Security)

### Enable HSTS headers to enforce HTTPS connections.

**5.)** **Audit Trail**

1.) Introduction

An audit trail is a critical component of any property management software. It provides a detailed record of user activities, login attempts, and modifications made to sensitive data. By implementing an effective audit trail, property managers can enhance accountability, traceability, and security.

**2. Key Objectives**

The primary objectives of the audit trail feature are as follows:

Capture User Activities: Record all user interactions within the system, including property viewings, lease agreements, maintenance requests, and financial transactions.

Track Login Attempts: Monitor login events, failed login attempts, and successful logins to identify potential security threats.

Document Data Modifications: Keep a log of changes made to critical data, such as lease terms, rent payments, and property details.

**3. Implementation Steps**

3.1. Azure Security Logging and Auditing

Azure provides robust security logging and auditing options. Consider the following steps:

Enable Azure Audit Logs:

Configure Azure services (such as Azure Active Directory, Azure SQL Database, and Azure Storage) to generate audit logs.

These logs capture relevant events related to user access, resource management, and data modifications.

Centralize Log Collection:

Use Azure Monitor or Azure Sentinel to collect and store audit logs centrally.

Set up log retention policies to ensure historical data availability.

Analyze and Alert:

Create custom alerts based on specific audit events (e.g., failed logins, role changes).

Set up automated responses (e.g., notifications, remediation actions) for critical events.

Regularly Review Logs:

Periodically review audit logs to identify anomalies, unauthorized access, or suspicious activities.

Investigate any unusual patterns promptly.

3.2. Application-Level Audit Trail

Implement an application-level audit trail within your property management software:

**User Activity Logging:**

Log user actions, such as property viewings, lease modifications, and maintenance requests.

Include details like timestamp, user ID, action performed, and affected property.

**Login Attempts:**

Record successful and failed login attempts.

Capture IP addresses, device information, and login timestamps.

Sensitive Data Modifications:

Track changes to critical data (e.g., lease terms, rent payments, property details).

Include the old and new values, user responsible, and timestamp.

**4. Best Practices**

Granularity: Ensure that the audit trail captures sufficient detail without overwhelming the system.

Retention Period: Define a retention policy for audit logs based on compliance requirements.

Access Controls: Limit access to audit logs to authorized personnel only.

Regular Reviews: Regularly review audit logs to maintain data integrity.

## 6.) ****Password Policies****

Azure AD provides robust features for managing user authentication and access. Let’s set up password policies to enhance security:

1. **Minimum Password Complexity:**

Define criteria for strong passwords, including:

Minimum length (e.g., 8 characters).

Mix of uppercase and lowercase letters.

Special characters (e.g., !, @, #).

Implement these requirements using Azure AD policies.

1. **Periodic Password Changes:**

Regularly prompt users to change their passwords. Set up password expiration policies in Azure AD. Encourage users to choose unique passwords during resets.

Steps to Configure Password Policies in Azure AD

Follow these steps to configure password policies:

**Access the Azure Portal:**

Sign in with your Azure AD administrator account.

1. **Navigate to Azure AD:**

Click on the Azure Active Directory blade.

1. **Configure Password Protection:**

In the left navigation pane, select Security.

Under Authentication methods, click Password protection.

Click the Create policy button.

1. **Define Password Policy Settings:**

Enter a name for the policy.

Specify the following settings:

Minimum password length: Set the minimum character count.

Password complexity: Define complexity requirements.

Password history: Specify the number of previous passwords that cannot be reused.

Password expiration: Set the number of days before a password expires.

1. **Account Lockout Policy:**

Configure settings related to failed sign-in attempts: Account lockout threshold: Number of failed attempts before an account is locked. Account lockout duration: Duration of account lockout after exceeding the threshold.

1. **Apply the Policy:**

Click the Create button. The policy will be applied to all users in your Azure AD tenant.

**7.). session management**

Azure provides several features and services that can enhance session management. Let’s explore some relevant strategies:

**Azure Bastion:**

Purpose: Securely manage and monitor remote sessions to virtual machines (VMs) without exposing them to the public internet.

**Features:**

Session Monitoring: Monitor distant sessions and perform management actions.

Session Termination: Swiftly terminate sessions when necessary.

Reference: Azure Bastion1.

**Azure Active Directory (Azure AD):**

**Session Expiration:**

Configure session expiration settings in Azure AD.

Set a reasonable session duration (e.g., 30 minutes).

**Inactivity Timeout:**

Define an inactivity timeout (e.g., 15 minutes).

Automatically log out users after a period of inactivity.

Reference: Power Platform Admin Center - User Session Management2.

Steps to Configure Secure Session Management in Azure

Follow these steps to implement secure session management:

**Azure Bastion Setup:**

Deploy Azure Bastion for secure remote access to VMs.

Monitor sessions and terminate them as needed.

**Azure AD Configuration:**

Access the Azure Portal.

Navigate to Azure AD settings.

Configure session expiration and inactivity timeout.

**Test and Monitor:**

Test session duration and inactivity timeout.

Monitor user sessions for any anomalies.

## 8.) ****Data Backup and Recover****

Azure provides powerful tools and services for data protection. Let’s explore the strategies:

**Azure Backup:**

Automated Backups:

Set up automated backups for your Property Management database.

Schedule regular backups to capture changes incrementally.

Choose the appropriate retention period based on business requirements.

**Azure Recovery Services Vault:**

Create a Recovery Services Vault in Azure.

Configure backup policies for your database.

Ensure that backups are stored securely in Azure.

Reference: Azure Backup Overview1.

**Azure Site Recovery (ASR):**

Efficient Recovery Systems:

Set up ASR for disaster recovery.

Replicate your database to a secondary Azure region.

In case of system failures, fail over to the secondary region for timely restoration.

**Test Failover:**

Regularly test failover scenarios to validate recovery processes.

Ensure that your recovery systems are functioning as expected.

Reference: Azure Site Recovery Documentation2.

Steps to Implement Data Backup and Recovery in Azure

Follow these steps to establish a robust backup and recovery system:

**Azure Backup Configuration:**

Access the Azure Portal.

Create a Recovery Services Vault.

Define backup policies for your Property Management database.

Monitor backup status and alerts.

**Azure Site Recovery Setup:**

Set up ASR for your database.

Configure replication settings.

Test failover scenarios periodically.

**Regular Testing and Monitoring:**

Conduct recovery drills to ensure readiness.

Monitor backup success rates and recovery time objectives (RTOs).

## 9.) ****Access Monitoring and Alerts****

Azure provides robust tools and services for access monitoring. Let’s explore the strategies:

**Azure Monitor Alerts:**

Purpose: Proactively notify you when Azure Monitor data indicates potential issues with your infrastructure or application.

**Alert Rules:**

Create alert rules based on metric or log data sources.

Define conditions (e.g., thresholds, query results) that trigger alerts.

Specify action groups for notifications or automated workflows.

**Types of Alerts:**

Metric Alerts: Evaluate resource metrics at regular intervals (e.g., CPU usage, memory).

Log Search Alerts: Use Log Analytics queries to evaluate resource logs.

Activity Log Alerts: Triggered when specific activity log events occur.

Reference: Azure Monitor Alerts Overview1.

**Best Practices for Alerts:**

Define Clear Alert Conditions:

Set meaningful thresholds or query conditions.

Avoid excessive noise by fine-tuning alerts.

**Leverage Action Groups:**

Configure notification methods (email, SMS, push notifications).

Use automation runbooks, Azure functions, or ITSM incidents.

**Regularly Review Alerts:**

Monitor alert instances in the Azure portal.

Investigate and address alerts promptly.

Reference: Best Practices for Azure Monitor Alerts2.

Steps to Implement Access Monitoring and Alerts in Azure

Follow these steps to enhance access monitoring:

**Create Alert Rules:**

Access the Azure Portal.

Navigate to Azure Monitor.

Define alert rules based on metrics or logs.

Set appropriate thresholds or query conditions.

**Configure Action Groups:**

Create action groups for notifications.

Include relevant stakeholders (e.g., IT team, property managers).

Specify notification methods and workflows.

**Test Alerts:**

Trigger test alerts to validate the setup.

Ensure that notifications reach the intended recipients.

# 10.) **Integration with Identity Providers**

Almost all multitenant solutions require an identity system. In this article, we’ll discuss common components of identity, including both authentication and authorization, and explore how these components can be applied in a multitenant solution.

Authentication and Federation

**Authentication:**

Process: Authentication establishes a user’s identity.

Multitenant Considerations:

Special considerations for authentication in multitenant solutions.

Choose appropriate authentication mechanisms (e.g., OAuth, OpenID Connect).

Implement Azure Active Directory (Azure AD) B2C for user authentication.

**Federation**:

Federate with other identity providers (IdPs):

Social login (Google, Facebook, GitHub, or personal Microsoft account).

Tenant-specific directories (allow tenants to use their own IdPs).

Clarify supported services and protocols (OpenID Connect, SAML, etc.).

**Single Sign-On (SSO):**

Enable seamless switching between applications without reauthentication.

Federated identity models support SSO across multiple applications.

Provide a seamless experience when users change context between tenants.

Best Practices and Recommendations

**Avoid Building Your Own IdP:**

Building your own identity provider is complex, expensive, and insecure.

Leverage existing solutions like Azure AD B2C.

Avoid custom IdP development.

**Azure AD B2C Limits:**

If using Azure AD B2C as your IdP, consider custom policy limits.

Plan for scalability and tenant-specific requirements.

**Test and Monitor:**

Regularly test authentication flows.

Monitor authentication success rates and user experiences.