Policy Management & Risk Exception Portal

Process Flow Document

---

1. Overview

The **Policy Management & Risk Exception Portal** is designed to streamline the management of organizational policies, user acknowledgements, and risk exceptions. It ensures transparency, accountability, and compliance by providing a structured system for:

- Publishing organizational policies.
- Tracking acknowledgements from employees/users.
- Managing risk exceptions and their approval lifecycle.
- Monitoring overall system health and availability.

---

2. Core Modules

2.1 Policy Management

- **Admin publishes policies** with a name, description)
- Policies are stored in the database (Policy Masters table).
- Each policy can be linked to exceptions.

2.2 Acknowledgements

- Users are required to acknowledge published policies.
- Acknowledgements are tracked in the PolicyMasters table.
- Metrics include:
  - Total Published/Unpublished Policies
  - Total Acknowledged Policies

2.3 Exception Management

- Users may request **exceptions** to policies.
- Exceptions have attributes like:
    - PolicyId (linked policy)
    - Reason (why exception is requested)
    - Duration In Days
    - Risk Rating (Low, Medium, High)
    - isApproved (approval status: Pending/Approved)
    - SubmittedDate
- Tracked in the PolicyExceptions table.
- Metrics include:
    - Total Exceptions
    - Approved Exceptions
    - Pending Exceptions

2.4 System Health Monitoring

- Health check endpoint (/api/health) verifies system availability.
- Checks database connectivity to **SQL Server (rezilens_mvp_db_docker)**.
- Provides status: **Healthy / Degraded** along with uptime.

2.3 Audit and Reporting

- Reports are shown for Policy and Exceptions with user actions along with Timespans.
- Exportable to Excel/CSV/Print.

---

3. Process Flow

Step 1: Policy Creation

- Admin logs into the portal.
- Creates a new policy → saved in PolicyMasters table.

Outcome: Policy Published.

Step 2: User Acknowledgement

- User logs in and views assigned policies.
- Acknowledges policies → entry saved in PolicyMasters.

Outcome: Acknowledgement recorded.

Step 3: Exception Request

- User requests an exception to a policy.
- Exception stored in RiskExceptions with isApproved = 0 (Pending).

Outcome: Exception Requested.

Step 4: Exception Approval

- Admin/Reviewer reviews exceptions.
- Updates isApproved = 1 (Approved) if accepted.
- Updates isApproved = 2 (Rejected) if rejected.

Outcome: Exception Approved/Rejected or stays Pending.

Step 5: Dashboard (Admin)

The portal provides a **dashboard view** via Angular that shows:

- Total Policies Published
- Total Acknowledged Policies
- Total Exceptions Raised
- Approved Exceptions

(Fetched via backend API → Angular Service → Dashboard Components).

Step 6: System Health Check

- /api/health endpoint checks:
    - API status (up and running).
    - Database connection (to rezilens_mvp_db).

Outcome: Ensures proactive monitoring.

Database Initialization and Seeding

1. Automatic Migrations

- The API project is configured to use **Entity Framework Core migrations**.
- On the **first run**, the database rezilens_mvp_db_docker will be:
    - Created if it does not exist.
    - Migrated with all tables required for:
        - Policies (PolicyMaster)
        - Risk Exceptions (RiskExceptionMaster)
        - Users (AspNetUsers / Identity tables)
        - Any other related tables.
- No manual SQL script execution is required; EF Core handles schema creation automatically.

---

2. Initial Data Seeding

The system seeds **initial users and roles** upon first migration:

Role	**Username**	**Password (default)**	**Notes**
Admin	sam	[hashed/secure default password]	Full admin privileges, can approve/reject exceptions and manage policies.

Role	**Username**	**Password (default)**	**Notes**
User	nawal	[hashed/secure default password]	Standard user privileges, can view policies and submit exceptions.

---

3. Flow of Actions After Initialization

1. Database is automatically created and seeded on first run.
2. Admin user sam logs in and manages initial policies if required.
3. Users can submit **risk exceptions** for policies.
4. Admin approves/rejects exceptions using the **exception modal**.
5. Dashboard stats, reports, and policy acknowledgement tracking are functional from the first run.

---

4. Benefits

- No manual setup required for new deployments.
- Consistent development, testing, and production environments.
- Initial users allow immediate testing of **role-based access control** and workflow.

---

4. Database Schema

Database

- **Name: rezilens_mvp_db**
- **Properties:**
    - **Recovery Model: Simple**
    - **Auto Close: ON**
    - **Read Committed Snapshot: ON**
    - **Query Store: Enabled (Read/Write)**
    - **Page Verify: CHECKSUM**

- **Containment: NONE**

Core Tables

1. Policies

Stores all published organizational policies.

```
CREATE TABLE Policies (
  PolicyId INT IDENTITY(1,1) PRIMARY KEY,
  PolicyName NVARCHAR(255) NOT NULL,
  Description NVARCHAR(MAX) NULL,
  CreatedBy NVARCHAR(100) NOT NULL,
  CreatedDate DATETIME DEFAULT GETDATE()
);
```

---

2. PolicyAcknowledgements

Tracks user acknowledgements of policies.

```
CREATE TABLE PolicyAcknowledgements (
  AcknowledgementId INT IDENTITY(1,1) PRIMARY KEY,
  PolicyId INT NOT NULL FOREIGN KEY REFERENCES Policies(PolicyId),
  UserId NVARCHAR(100) NOT NULL,
  AcknowledgedDate DATETIME DEFAULT GETDATE()
);
```

---

3. PolicyExceptions

Manages exceptions requested against policies.

```
CREATE TABLE PolicyExceptions (
  ExceptionId INT IDENTITY(1,1) PRIMARY KEY,
  PolicyId INT NOT NULL FOREIGN KEY REFERENCES Policies(PolicyId),
  UserId NVARCHAR(100) NOT NULL,
  Reason NVARCHAR(MAX) NOT NULL,
  isApproved BIT DEFAULT 0, -- 0 = Pending, 1 = Approved
  CreatedDate DATETIME DEFAULT GETDATE()
);
```

---

5. API Endpoints

1. Authentication

Method	Endpoint	Description
POST	/api/Authentication/login	Authenticates a user and returns JWT token.

---

2. Dashboard

Method	Endpoint	Description
GET	/api/Dashboard/get-stats	Returns counts of total policies, acknowledged policies, total exceptions, and approved exceptions for dashboard display.

---

3. Health Check

Method	Endpoint	Description
GET	/api/Health	Returns API and system health status. Useful for monitoring.

---

4. Policy Management

Method	Endpoint	Description
POST	/api/Policy/create-policy	Creates a new policy.

Method	Endpoint	Description
GET	/api/Policy/get-policies	Retrieves all policies in the system.
GET	/api/Policy/get-policies-for-user	Retrieves policies specific to the logged-in user.
GET	/api/Policy/get-policy/{id}	Retrieves details of a specific policy by ID.
PUT	/api/Policy/update-policy	Updates an existing policy.
PUT	/api/Policy/acknowledge-policy	Marks a policy as acknowledged by a user.
PUT	/api/Policy/publish-policy	Publishes a policy to make it visible to users.

---

5. Reports

Method	Endpoint	Description
GET	/api/Reports/GetPoliciesWithExceptionsAndAcknowledgements	Returns policies with linked exceptions and acknowledgement status for reporting purposes.

---

6. Risk Exception Management

Method	Endpoint	Description
GET	/api/RiskException/published-policies-for-exception	Retrieves published policies available for requesting exceptions.

Method	**Endpoint**	**Description**
POST	**/api/RiskException/submit-risk-exception**	**Submits a new risk exception request for a policy.**
GET	**/api/RiskException/all-risk-exceptions**	**Retrieves all risk exceptions in the system.**
POST	**/api/RiskException/update-risk-exception-status**	**Updates the status of a risk exception (approve/reject) with admin comments.**

---

7. Test / Miscellaneous

Method	**Endpoint**	**Description**
GET	**/weatherforecast**	**Default test endpoint for API project.**

---

6. Angular Frontend Flow

- **Services**:
  - PolicyService → calls /api/policies
  - ExceptionService → calls /api/exceptions
  - ReportService → calls /api/summary
- **Components**:
  - Dashboard → shows summary stats.
  - Policy List → display policies & acknowledgements.
  - Exceptions → raise & track status.
  - Admin Panel → approve exceptions.

---

7. System Health & Maintenance

- Health checks validate API + Database.

- Docker container (rezilens_mvp_db) must be running for DB availability.

---

8. Benefits

- Centralized policy repository.
- Transparent acknowledgement tracking.
- Controlled exception management.
- Real-time health visibility.
- Modular design for future expansion (e.g., notifications, audits, compliance reports).