API Design Documentation

Waleed Shoaib

January 31, 2025

1 API Endpoints Structure

```
/api
          /auth
                POST   /register
                POST   /login
                POST   /enable-2fa
                POST   /verify-2fa
                POST   /logout

          /users
                GET    /profile
                PUT    /update-role/:id
                DELETE /delete/:id

          /risk
                POST   /assess
                GET    /score/:id
                PUT    /update/:id
                DELETE /delete/:id

          /incidents
                POST   /log
                GET    /list
                PUT    /resolve/:id

          /compliance
                GET    /status
                PUT    /update/:id

          /documents
                POST   /upload
                GET    /list
                PUT    /approve/:id
                DELETE /delete/:id
```

2 Authentication & Authorization (JWT + 2FA)

2.1 Register a New User

```
POST /api/auth/register
Content-Type: application/json

{
  "email": "user@example.com",
  "password": "securepassword",
  "role": "standard"
}
```

Response:

```
{
  "message": "User registered successfully",
  "userId": "12345"
}
```

2.2 Login & Generate JWT

```
POST /api/auth/login
Content-Type: application/json

{
  "email": "user@example.com",
  "password": "securepassword"
}
```

Response:

```
{
  "message": "Login successful",
  "token": "jwt_token_here",
  "requires2FA": true
}
```

2.3 Enable Two-Factor Authentication (2FA)

```
POST /api/auth/enable-2fa
Content-Type: application/json

{
  "userId": "12345",
  "method": "authenticator"  // Options: SMS, Email, Authenticator
}
```

Response:

```
{
  "message": "2FA enabled via Authenticator"
}
```

3 User Management

3.1 Get User Profile

```
GET /api/users/profile
Authorization: Bearer jwt_token
```

Response:

```
{
  "email": "user@example.com",
  "role": "standard",
  "2FA_enabled": true
}
```

3.2 Update User Role (Admin Only)

```
PUT /api/users/update-role/12345
Content-Type: application/json
Authorization: Bearer admin_token

{
  "role": "risk_manager"
}
```

Response:

```
{
  "message": "User role updated to risk_manager"
}
```

4 Risk Assessment

4.1 Assess a New Risk

```
POST /api/risk/assess
Content-Type: application/json

{
  "impact": 3,
  "likelihood": 4
}
```

Response:

```
{
  "riskScore": 12,
  "recommendation": "Mitigation Required"
}
```

4.2 Get Risk Score

```
GET /api/risk/score/9876
Authorization: Bearer jwt_token
```

Response:

```
{
  "riskScore": 12,
  "status": "Mitigation Required"
}
```

5 Incident Logging

5.1 Log an Incident

```
POST /api/incidents/log
Content-Type: application/json

{
  "type": "Unauthorized Access",
  "description": "Multiple failed login attempts detected",
  "severity": "high"
}
```

Response:

```
{
  "message": "Incident logged successfully",
  "incidentId": "54321"
}
```

6 Compliance & Monitoring

6.1 Get Compliance Status

```
GET /api/compliance/status
Authorization: Bearer jwt_token
```

Response:

```
[
  {
    "regulation": "ISO 27001",
    "status": "Compliant"
  },
  {
    "regulation": "GDPR",
    "status": "Pending"
  }
]
```

6.2 Update Compliance Status

```
PUT /api/compliance/update/2023
Content-Type: application/json
Authorization: Bearer jwt_token

{
  "status": "Compliant"
}
```

Response:

```
{
  "message": "Compliance updated successfully"
}
```

7 Document Management

7.1 Upload a Document

```
POST /api/documents/upload
Content-Type: multipart/form-data
Authorization: Bearer jwt_token

{
  "file": "risk_policy.pdf"
}
```

Response:

```
{
  "message": "Document uploaded successfully",
  "documentId": "22222"
}
```