# Class Design for Risk Management & User Document System

Waleed Shoaib

January 31, 2025

## 1 User Authentication & Permissions

```
class User {
    constructor(id, email, password, role = "standard",
        twoFactorEnabled = false) {
        this.id = id;
        this.email = email;
        this.password = this.hashPassword(password);
        this.role = role;
        this.twoFactorEnabled = twoFactorEnabled;
    }

    hashPassword(password) {
        return `hashed_${password}`;
    }

    enable2FA(method) {
        const validMethods = ["SMS", "Email", "Authenticator"];
        if (validMethods.includes(method)) {
            this.twoFactorEnabled = true;
            return `2FA enabled via ${method}`;
        }
        return "Invalid 2FA method";
    }

    verifyPassword(inputPassword) {
        return this.password === this.hashPassword(inputPassword);
    }
}
```

## 2 Risk Assessment & Mitigation

```
class RiskAssessment {
    constructor(id, impact, likelihood) {
        this.id = id;
        this.impact = impact;
        this.likelihood = likelihood;
        this.riskScore = this.calculateRisk();
    }

    calculateRisk() {
        return this.impact * this.likelihood;
    }

    applyControls(controls = []) {
        this.riskScore -= controls.reduce((sum, control) => sum +
            control, 0);
        this.riskScore = Math.max(this.riskScore, 0);
        return this.riskScore;
    }

    getRiskLevel() {
        if (this.riskScore <= 4) return "Low Risk";
        if (this.riskScore <= 6) return "Medium Risk";
        return "High Risk";
    }
}
```

## 3 Incident Management

```
class Incident {
    constructor(id, type, description, severity = "medium") {
        this.id = id;
        this.type = type;
        this.description = description;
        this.severity = severity;
        this.status = "Open";
    }

    resolveIncident() {
        this.status = "Resolved";
        return `Incident ID ${this.id} resolved.`;
    }
}
```

## 4 Compliance Tracking

```
class Compliance {
    constructor(id, regulation, status = "Pending") {
        this.id = id;
        this.regulation = regulation;
        this.status = status;
    }

    updateStatus(newStatus) {
        this.status = newStatus;
        return `Compliance for ${this.regulation} updated to ${
            newStatus}.`;
    }
}
```

## 5 Document Management

```
class Document {
    constructor(id, name, fileType, owner, status = "Pending
        Approval") {
        this.id = id;
        this.name = name;
        this.fileType = fileType;
        this.owner = owner;
        this.status = status;
    }

    approveDocument() {
        this.status = "Approved";
        return `Document ${this.name} approved.`;
    }

    rejectDocument(reason) {
        this.status = `Rejected: ${reason}`;
        return `Document ${this.name} rejected due to: ${reason}`;
    }
}
```

## 6 System Users with Role-Based Access

```
class SystemUser extends User {
    constructor(id, email, password, role) {
        super(id, email, password, role);
        this.permissions = this.assignPermissions(role);
    }

    assignPermissions(role) {
        const rolePermissions = {
            "admin": ["manage_users", "approve_documents", "
                manage_risks"],
            "security_officer": ["handle_compliance", "
                monitor_risks"],
            "risk_manager": ["manage_risks", "log_incidents"],
            "standard": ["upload_documents", "view_policies"]
        };
        return rolePermissions[role] || [];
    }

    hasPermission(permission) {
        return this.permissions.includes(permission);
    }
}
```

## 7 Class Diagram (Mermaid)

```
classDiagram
    class User {
        +id: number
        +email: string
        +password: string
        +role: string
        +twoFactorEnabled: boolean
        +enable2FA(method: string): string
        +verifyPassword(password: string): boolean
    }

    class SystemUser {
        +permissions: array
        +assignPermissions(role: string): array
        +hasPermission(permission: string): boolean
    }

    class RiskAssessment {
        +id: number
        +impact: number
        +likelihood: number
        +riskScore: number
        +calculateRisk(): number
        +applyControls(controls: array): number
        +getRiskLevel(): string
    }

    class Incident {
        +id: number
        +type: string
        +description: string
        +severity: string
        +status: string
        +resolveIncident(): string
    }

    class Compliance {
        +id: number
        +regulation: string
        +status: string
        +updateStatus(newStatus: string): string
    }

    class Document {
        +id: number
        +name: string
        +fileType: string
        +owner: string
        +status: string
        +approveDocument(): string
        +rejectDocument(reason: string): string
    }

    User <|-- SystemUser
```