Document Mess Detector (DMD) - Solution Design

Overview

Document Mess Detector (DMD) is a native Salesforce application that uses Generative AI to analyze business documents in Salesforce Files and Attachments, checking them against configurable rulesets. The application helps ensure document quality and compliance with organizational standards by leveraging Large Language Models (LLMs) through Salesforce's Einstein Prompt Templates system.

This document outlines the solution design for DMD, with a focus on security aspects relevant for Salesforce Security Review.

Personas

```
@startuml
!theme plain
skinparam actorStyle awesome

actor "Document Reviewer" as reviewer
actor "Rule Author" as author
actor "App Administrator" as admin

note right of reviewer
  Needs to analyze documents
  against defined rules
end note

note right of author
  Creates and manages rules
  and rulesets
end note

note right of admin
  Manages app configuration
  and permissions
end note

@enduml
```

Document Reviewer

- **Responsibilities**: Selects and analyzes documents against predefined rulesets
- **Permissions**: Can create analyses but cannot modify rules or rulesets
- **Primary tasks**:
  - Select documents for analysis
  - View analysis results
  - Share results with colleagues

Rule Author

- **Responsibilities**: Creates and maintains rules and rulesets
- **Permissions**: Can create/modify rules and rulesets, and run regression tests
- **Primary tasks**:
  - Create rules in natural language
  - Organize rules into rulesets
  - Generate rules with AI assistance
  - Run regression tests to ensure AI analysis consistency

App Administrator

- **Responsibilities**: Configures and maintains the application
- **Permissions**: Full access to all app functionality, including setup functions
- **Primary tasks**:
  - Configure external API settings
  - Manage user permissions
  - Monitor application usage
  - Export/Import rulesets

Use Cases

```
@startuml
!theme plain
left to right direction

actor "Document Reviewer" as reviewer
actor "Rule Author" as author
actor "App Administrator" as admin

rectangle "Document Mess Detector" {
  usecase "Analyze Document Against Ruleset" as UC1
  usecase "View Analysis Results" as UC2
  usecase "Create/Edit Rules" as UC3
  usecase "Organize Rules into Rulesets" as UC4
  usecase "Generate Rules with AI" as UC5
  usecase "Run Regression Tests" as UC6
  usecase "Configure API Settings" as UC7
  usecase "Export/Import Rulesets" as UC8
}

reviewer --> UC1
reviewer --> UC2

author --> UC3
author --> UC4
author --> UC5
author --> UC6
author --> UC8
```

```
admin --> UC7
admin --> UC8
admin .up.> UC1
admin .up.> UC2
admin .up.> UC3
admin .up.> UC4
admin .up.> UC5
admin .up.> UC6

@enduml
```

1. Analyze Document Against Ruleset

A Document Reviewer selects a document (PDF, text file) from Salesforce Files or Attachments and chooses a ruleset to analyze it against. The application extracts text from the document, processes it through an AI model against the rules, and returns results.

2. View Analysis Results

Users view analysis results showing whether each rule passed or failed, with justifications quoting relevant document sections.

3. Create/Edit Rules

Rule Authors create rules using natural language to describe requirements for documents, without needing special syntax or ML expertise.

4. Organize Rules into Rulesets

Rule Authors group related rules into rulesets, with optional context to help the AI understand the document type and purpose.

5. Generate Rules with AI

Rule Authors can request AI to generate relevant rules based on ruleset context.

6. Run Regression Tests

Rule Authors set up regression tests to ensure consistency in AI evaluations over time by using a benchmark document and comparing results.

7. Configure API Settings

Administrators configure necessary API credentials for document text extraction and maintain app settings.

8. Export/Import Rulesets

Administrators and Rule Authors can export rulesets for backup or transfer, and import them into other orgs.

System Architecture

```
@startuml
!theme plain

package "Salesforce" {
  [Document Mess Detector]

  package "Custom Objects" {
    [Rule__c]
    [Ruleset__c]
    [RuleInSet__c]
    [Analysis__c]
    [Result__c]
    [RegressionTest__c]
  }

  package "Protected Settings" {
    [DmdSettings__c]
  }

  package "Einstein Platform" {
    [Einstein Prompt Templates]
  }

  database "Files & Attachments" as Files
}

package "External Services" {
  [ExtractorAPI.com] #LightYellow
  [LLM Provider (via Salesforce)] #LightBlue
}

[Document Mess Detector] --> [Rule__c]
[Document Mess Detector] --> [Ruleset__c]
[Document Mess Detector] --> [Analysis__c]
[Document Mess Detector] --> [Result__c]
[Document Mess Detector] --> [RegressionTest__c]

[Ruleset__c] <--> [RuleInSet__c]
[Rule__c] <--> [RuleInSet__c]

[Document Mess Detector] --> [Einstein Prompt Templates]
[Document Mess Detector] --> [DmdSettings__c]
[Document Mess Detector] --> Files

[Document Mess Detector] --> [ExtractorAPI.com]: Text extraction
[Einstein Prompt Templates] --> [LLM Provider (via Salesforce)]: LLM
inferencing

@enduml
```

Key Components

1. **Custom Objects**:
    - `Rule__c`: Stores individual rules defined in natural language
    - `Ruleset__c`: Groups related rules with optional context information
    - `RuleInSet__c`: Junction object connecting rules to rulesets
    - `Analysis__c`: Represents a document analysis against a ruleset
    - `Result__c`: Stores individual rule results within an analysis
    - `RegressionTest__c`: Configuration for regression testing

2. **Protected Custom Settings**:
    - `DmdSettings__c`: Organization-wide settings, including API keys

3. **External Integrations**:
    - **ExtractorAPI.com**: 3rd party service for document text extraction
    - **Salesforce Einstein Platform**: For secure access to LLM models

Security Architecture

```
@startuml
!theme plain

skinparam linetype ortho

package "Salesforce" {
  [User Interface]
  [DMD Application]

  database "Protected Settings" as Settings {
    [API Keys (Encrypted)] #Yellow
  }

  package "Permission Sets" {
    [Document Reviewer]
    [Rule Author]
    [App Administrator]
  }

  package "Security Controls" {
    [Object-Level Security]
    [Field-Level Security]
    [Apex Security]
    [WITH USER_MODE]
  }
}

cloud "External Services" {
  [ExtractorAPI.com]
  [Salesforce Einstein Platform] #LightBlue
  [LLM Provider] #LightBlue
}
```

```
[User Interface] --> [DMD Application]: 1. User inputs
[DMD Application] --> [Security Controls]: 2. Security enforcement
[DMD Application] --> [Settings]: 3. Retrieve credentials
[DMD Application] --> [ExtractorAPI.com]: 4. HTTPS callout
[DMD Application] --> [Salesforce Einstein Platform]: 5. Prompt Template
execution
[Salesforce Einstein Platform] --> [LLM Provider]: 6. LLM inference
(handled by Salesforce)

[Document Reviewer] --> [Object-Level Security]: Configures
[Rule Author] --> [Object-Level Security]: Configures
[App Administrator] --> [Object-Level Security]: Configures

note right of [API Keys (Encrypted)]
  Stored in protected custom settings,
  encrypted at rest by Salesforce
end note

note bottom of [ExtractorAPI.com]
  Text extraction API
  Secure HTTPS communication
end note

note bottom of [Salesforce Einstein Platform]
  Salesforce Trust Layer
  No direct LLM provider access
end note

@enduml
```

Security Aspects

1. Authentication and Authorization

- **Permission Sets**:
  - Three distinct permission sets define access levels:
    - `DocumentReviewer`: Limited to analyzing documents and viewing results
    - `RuleAuthor`: Can create/modify rules and rulesets, run regression tests
    - `AppAdministrator`: Full access including setup functionality
  - All implement proper object and field-level security restrictions
- **Object and Field-Level Security**:
  - Custom objects are secured through permission sets with proper CRUD and FLS settings
  - Apex code uses `WITH USER_MODE` to enforce object and field-level security

2. Credential Storage

- **API Key Management**:

    - ExtractorAPI key stored in protected custom settings (`DmdSettings__c`)
    - Protected custom settings are encrypted at rest by Salesforce
    - Only accessible through Apex, not directly through UI or reports
    - `SetupExtractApi` class handles secure storage and retrieval

- **LLM Access**:

    - No direct LLM API keys stored in the application
    - All LLM access through Salesforce's Einstein Platform (Prompt Templates)
    - Salesforce handles authentication to LLM providers

3. Data Protection

- **Document Processing**:

    - Documents remain within Salesforce ecosystem
    - Only document text (not binary content) is sent to external services
    - Text extraction via secure HTTPS calls to ExtractorAPI.com
    - All network communication uses TLS

- **Sensitive Information**:

    - Application doesn't permanently store document text, only analysis results
    - Results may contain short quotes from documents for justification
    - API keys are never exposed in UI or logs

4. External Service Integration

- **ExtractorAPI.com**:

    - Remote site setting configured to allow HTTPS callouts
    - Secure document text extraction with API key authentication
    - Properly handles document size limitations to prevent heap issues

- **Einstein Prompt Templates**:

    - Leverages Salesforce's trusted infrastructure for LLM access
    - Prompt templates defined within Salesforce, not externally
    - Uses Salesforce's secure channel to LLM providers

Data Flow

```
@startuml
!theme plain

actor "User" as user
participant "DMD UI" as ui
participant "AnalyseDocument" as analyser
participant "DocumentText" as extractor
participant "CompletionApi" as api
```

```
entity "Salesforce Files" as files
database "Custom Objects" as db
participant "Einstein Platform" as einstein

boundary "ExtractorAPI.com" as extractorapi #Yellow

user -> ui: 1. Select document\nand ruleset
ui -> analyser: 2. Create Analysis__c
activate analyser

analyser -> db: 3. Query ruleset and rules
analyser -> files: 4. Retrieve document
analyser -> extractor: 5. Extract text
activate extractor

extractor -> extractorapi: 6. Send document\nfor text extraction
extractorapi --> extractor: 7. Return document text
extractor --> analyser: 8. Return text
deactivate extractor

analyser -> api: 9. Prepare LLM prompt
api -> einstein: 10. Execute\nprompt template
activate einstein
einstein --> api: 11. Return LLM\nanalysis results
deactivate einstein

api --> analyser: 12. Structured results
analyser -> db: 13. Store results
analyser --> ui: 14. Update status
deactivate analyser

ui --> user: 15. Display results

@enduml
```

Security Considerations in Data Flow

1. **Document Selection (Steps 1-2)**:
    - User can only see documents they have access to (Salesforce standard security)
    - Permission to create Analysis__c records is controlled by permission sets
2. **Data Access (Steps 3-4)**:
    - Queries use `WITH USER_MODE` to enforce object-level security
    - Document access leverages Salesforce's standard file security model
3. **Text Extraction (Steps 5-8)**:
    - API key retrieved from protected custom settings
    - Secure HTTPS callout to ExtractorAPI.com
    - Document size checks prevent heap overflow attacks

4. **LLM Processing (Steps 9-12)**:
    - No direct API keys exposed
    - Leverages Salesforce's Einstein Platform security
    - Structured format prevents injection attacks
5. **Result Storage (Steps 13-15)**:
    - Results stored with proper sharing settings
    - Justifications may contain document quotes but limited in size

Regression Testing System

```
@startuml
!theme plain

package "Regression Testing" {
  [RegressionTest__c] as config
  [Analysis__c] as benchmark #LightGreen
  [Analysis__c] as regression
  [RegressionSuiteScheduler] as scheduler
  [RegressionSuiteQueue] as queue

  database "Benchmark Results" as benchResults
  database "Regression Results" as regressionResults

  [Analytics Dashboard] as dashboard
}

config --> benchmark: references
config --> scheduler: schedules
scheduler --> queue: enqueues
queue --> regression: creates
regression --> regressionResults: produces
benchmark --> benchResults: has

dashboard --> benchResults: compares
dashboard --> regressionResults: with

note right of benchmark
  "Golden standard" analysis
  selected by human
end note

note right of regression
  Automated re-analysis
  of same document
end note

note bottom of dashboard
  Monitors LLM consistency
  over time and updates
```

```
end note

@enduml
```

Security Aspects of Regression Testing

- **Schedule Security**: Only users with proper permission set can schedule jobs
- **Data Access**: Regression tests limited to documents/rulesets the user has access to
- **Automation Governance**: Scheduled jobs run with system context but respect sharing settings
- **Failure Monitoring**: System includes notification system for failures

API and Integration Security

ExtractorAPI Integration

- **Purpose**: Converts binary document formats (PDF) to plaintext
- **Security Measures**:
  - API key stored in protected custom settings, encrypted at rest
  - HTTPS callouts only (enforced via remote site settings)
  - No sensitive customer data persisted by the service
  - Proper error handling for service availability issues

Einstein Platform Integration

- **Purpose**: Provides secure access to LLM capabilities
- **Security Measures**:
  - Uses Salesforce's managed Einstein Platform (Prompt Templates)
  - No direct API keys needed
  - LLM providers authenticated through Salesforce's Trust Layer
  - Prompts structured to prevent prompt injection attacks

Conclusion

The Document Mess Detector (DMD) application is designed with security as a core principle. By leveraging Salesforce's robust security model and established best practices, DMD ensures that:

1. **Access Control**: Users only access functionality and data appropriate to their role via permission sets
2. **Sensitive Data**: API credentials are securely stored in protected settings
3. **External Communications**: All external service communications use secure protocols
4. **LLM Security**: AI functionality leverages Salesforce's secure Einstein Platform
5. **Monitoring**: Regression testing provides visibility into AI consistency over time

The application follows Salesforce security best practices including proper CRUD/FLS enforcement, protection against SOQL injection, secure credential storage, and proper callout security.