Automated Phishing Detection and Mitigation System

I. Software Architecture Style Selection

Selected Architecture: Layered Architecture (N-Tier Architecture)

A. Justification and Component Granularity

The Automated Phishing Detection and Mitigation System follows a Layered Architecture pattern with the following characteristics:

1. Clear Separation of Concerns:

- Each layer has distinct responsibilities with minimal inter-layer dependencies
- Components within each layer are cohesive and loosely coupled
- Dependencies flow unidirectionally from top to bottom layers

2. Layer Structure and Granularity:

Presentation Layer (Coarse-grained components):

- User: Handles end-user interactions (email submission, alert viewing, phishing reporting)
- Admin: Manages system configuration and monitoring operations

Business Logic Layer (Fine-grained, specialized components):

- EmailAnalyzer: Rule-based detection engine using keyword matching, link analysis, and sender verification
- MLModelDetector: Machine learning-based detection using trained models and feature extraction
- RiskClassifier: Risk assessment engine that combines rule-based and ML results to determine threat level
- AlertGenerator: Alert creation and notification dispatch system

Data Access Layer (Medium-grained components):

- Blocklist: Manages blocked domains and email addresses with database persistence
- DetectionRules: Stores and manages detection rules, keywords, and trusted domains
- Database: Provides data persistence and retrieval operations

Service Layer (Cross-cutting concerns):

- NotificationService: Handles external communication channels for alert delivery

3. Key Architectural Characteristics:

- **Hierarchical Organization:** Upper layers depend on lower layers but not vice versa
- **Encapsulation:** Each layer exposes well-defined interfaces while hiding implementation details
- **Component Modularity:** EmailAnalyzer and MLModelDetector operate independently, allowing parallel execution
- **Data Flow:** Email → Analysis → Risk Classification → Alert Generation

B. Justification for Architecture Choice

The Layered Architecture is the optimal choice for this phishing detection system based on the following criteria:

1. Scalability:

- **Horizontal Scaling:** Business logic layer can be scaled independently by deploying multiple instances of EmailAnalyzer and MLModelDetector
- **Load Distribution:** High email volumes can be distributed across multiple analyzer instances without affecting other layers
- **Database Optimization:** Data layer can be scaled with read replicas and caching strategies without modifying business logic

2. Maintainability:

- **Modular Updates:** Detection rules, ML models, and blocklists can be updated independently without system-wide changes
- **Testability:** Each layer can be unit tested in isolation with mock interfaces
- **Clear Responsibilities:** Developers can work on specific layers without understanding the entire system architecture
- **Technology Flexibility:** ML models can be upgraded (e.g., switching frameworks) without affecting other components

3. Performance:

- **Caching Optimization:** Frequently accessed data (blocklists, detection rules) can be cached at the business logic layer
- **Parallel Processing:** EmailAnalyzer and MLModelDetector can analyze emails concurrently
- **Resource Management:** Heavy ML inference operations are isolated in dedicated components, preventing resource contention

4. Security Requirements:

- **Access Control:** Admin operations are separated from User operations at the presentation layer
- **Data Protection:** Database layer provides centralized security controls and encryption
- **Audit Trail:** Each layer can implement logging independently for compliance and debugging

5. Extensibility:

- **New Detection Methods:** Additional analyzers (e.g., behavioral analysis, image recognition) can be added to the business logic layer
- **Integration Capabilities:** New notification channels can be added through the NotificationService without modifying core logic
- **API Development:** RESTful APIs can be exposed at the presentation layer for third-party integrations

6. Development Efficiency:

- **Team Organization:** Different teams can work on UI, detection algorithms, and database optimization simultaneously
- **Reusability:** Core components like RiskClassifier and AlertGenerator can be reused across different phishing detection modules
- **Deployment Flexibility:** System can be deployed as a monolith initially and later migrated to distributed deployment if needed

II. Application Components

The Automated Phishing Detection and Mitigation System comprises the following application components:

1. Email (Entity Component)

- **Purpose:** Core data structure representing email messages
- **Attributes:** emailId, sender, recipient, subject, body, headers, attachments, receivedDate
- **Key Functions:** Extract sender information, retrieve email body, extract hyperlinks, access attachments
- **Relationships:** Contains 0..* Attachment objects; analyzed by EmailAnalyzer and MLModelDetector

2. Attachment (Entity Component)

- **Purpose:** Represents email attachments with metadata
- **Attributes:** filename, fileType, size
- **Key Functions:** Retrieve file metadata for suspicious extension detection

3. EmailAnalyzer (Core Detection Component)

- **Purpose:** Rule-based phishing detection engine
- **Attributes:** suspiciousKeywords, blocklist, detectionRules
- **Key Functions:** analyze(), checkSuspiciousLinks(), checkSuspiciousKeywords(), verifySender(), checkAttachments()
- **Detection Methods:**
  - Suspicious link detection (shortened URLs, mismatched domains)
  - Keyword-based detection (urgent action requests, financial terms)
  - Sender verification against blocklist and trusted domains
  - Attachment analysis for suspicious file types

4. Blocklist (Data Management Component)

- **Purpose:** Maintains and manages lists of blocked domains and email addresses

- **Attributes:** blockedDomains, blockedEmails, database
- **Key Functions:** isBlocked(), addToBlocklist(), removeFromBlocklist(), loadFromDatabase(), saveToDatabase()
- **Persistence:** Synchronizes with Database for persistent storage

5. DetectionRules (Configuration Component)

- **Purpose:** Stores and manages detection patterns and rules
- **Attributes:** suspiciousKeywords, suspiciousExtensions, trustedDomains
- **Key Functions:** isSuspiciousKeyword(), isTrustedDomain(), addRule(), removeRule()

6. AnalysisResult (Result Entity)

- **Purpose:** Encapsulates rule-based analysis results
- **Attributes:** suspiciousLinkCount, suspiciousKeywordCount, senderVerified, suspiciousAttachmentCount, totalRiskIndicators
- **Key Functions:** calculateTotalRisk(), toString()

7. Database (Persistence Component)

- **Purpose:** Provides data persistence layer for blocklists, rules, and system configuration
- **Attributes:** connectionString
- **Key Functions:** connect(), query(), update()

8. MLModelDetector (Machine Learning Component)

- **Purpose:** AI-powered phishing detection using machine learning models
- **Attributes:** model, featureExtractor, threshold
- **Key Functions:** predict(), extractFeatures(), trainModel(), updateModel()
- **Capabilities:**
  - Feature extraction from email content
  - Probability-based phishing prediction
  - Continuous model training and updates

9. MLResult (ML Result Entity)

- **Purpose:** Encapsulates machine learning prediction results
- **Attributes:** phishingProbability, isPhishing, confidence
- **Key Functions:** toString()

10. RiskClassifier (Classification Component)

- **Purpose:** Combines rule-based and ML-based results to determine final risk level
- **Attributes:** lowRiskThreshold, mediumRiskThreshold, highRiskThreshold
- **Key Functions:** classifyRisk(), calculateRiskScore(), determineRiskLevel()
- **Risk Levels:** LOW, MEDIUM, HIGH (enumeration)

11. AlertGenerator (Notification Component)

- **Purpose:** Generates and displays alerts based on risk classification
- **Attributes:** alertTemplates, notificationService

- **Key Functions:** generateAlert(), formatAlertMessage(), displayAlert(), sendNotification()
- **Integration:** Uses NotificationService for multi-channel alert delivery

12. Alert (Entity Component)

- **Purpose:** Represents individual alert instances
- **Attributes:** alertId, message, riskLevel, timestamp
- **Key Functions:** display(), toString()

13. User (Actor Component)

- **Purpose:** Represents end-users interacting with the system
- **Attributes:** userId, email, name, preferences
- **Key Functions:** submitEmail(), viewAlert(), reportPhishing(), viewRiskReport()
- **Interactions:** Submits 0..* emails, receives 0..* alerts

14. Admin (Administrative Component)

- **Purpose:** System administrator with elevated privileges (inherits from User)
- **Attributes:** adminLevel, permissions
- **Key Functions:** manageBlocklist(), updateDetectionRules(), trainMLModel(), viewSystemLogs(), generateStatistics()
- **Responsibilities:**
  - Blocklist management (add/remove domains)
  - Detection rule configuration
  - ML model training and updates
  - System monitoring and analytics

15. NotificationService (External Service Component)

- **Purpose:** Handles multi-channel notification delivery
- **Delivery Channels:** Email notifications, SMS alerts, push notifications, in-app alerts
- **Integration:** Used by AlertGenerator to deliver alerts to users

Component Interaction Summary

The system follows a pipeline architecture within the layered structure:

- **Step 1:** User submits Email for analysis
- **Step 2:** EmailAnalyzer performs rule-based detection → produces AnalysisResult
- **Step 3:** MLModelDetector performs ML-based prediction → produces MLResult
- **Step 4:** RiskClassifier combines both results → determines RiskLevel
- **Step 5:** AlertGenerator creates Alert and notifies User via NotificationService
- **Step 6:** Admin can update system configuration (Blocklist, DetectionRules) and retrain ML models

This component architecture ensures modularity, maintainability, and scalability while providing comprehensive phishing detection through both traditional rule-based methods and modern machine learning approaches.