1. Overview

The **Trade Finance Management System (TFMS)** is a web application designed for banks and financial institutions to manage trade finance operations. The system provides functionalities for handling letters of credit, guarantees, and trade documentation. TFMS ensures compliance with international trade regulations, improves process efficiency, and minimizes risks associated with trade finance.

Built using **MVC architecture**, the system supports both **Java (Spring MVC)** and **.NET (ASP.NET Core MVC)** frameworks for implementation flexibility.

2. Core Modules Description

2.1 Letter of Credit Management

Facilitates the issuance, amendment, and closure of letters of credit (LC).

2.2 Bank Guarantee Management

Handles requests, issuance, and tracking of bank guarantees.

2.3 Trade Documentation

Enables the management of trade documents like invoices, bills of lading, and shipping details.

2.4 Risk Assessment

Analyzes risk factors associated with trade finance transactions and provides risk scoring.

2.5 Compliance and Regulatory Reporting

Ensures adherence to international trade laws and generates necessary compliance reports.

3. Module-Level Design

3.1 Letter of Credit Management Module

Purpose: Manages the lifecycle of letters of credit.

- **Controller:**
  - LetterOfCreditController
    - createLetterOfCredit()
    - amendLetterOfCredit()
    - closeLetterOfCredit()
- **Service:**
  - LetterOfCreditService
    - Processes LC requests, validates details, and manages the lifecycle.
- **Model:**
  - **LetterOfCredit Entity**
    - Attributes:
      - lcId (PK)
      - applicantName
      - beneficiaryName

            - amount
            - currency
            - expiryDate
            - status

3.2 Bank Guarantee Management Module

Purpose: Issues and tracks bank guarantees.

- **Controller:**
    - BankGuaranteeController
        - requestGuarantee()
        - issueGuarantee()
        - trackGuaranteeStatus()
- **Service:**
    - BankGuaranteeService
        - Validates guarantee requests, issues guarantees, and updates statuses.
- **Model:**
    - **BankGuarantee Entity**
        - Attributes:
            - guaranteeId (PK)
            - applicantName
            - beneficiaryName
            - guaranteeAmount
            - currency
            - validityPeriod
            - status

3.3 Trade Documentation Module

Purpose: Manages trade-related documentation.

- **Controller:**
    - TradeDocumentController
        - uploadDocument()
        - viewDocument()
        - updateDocumentDetails()
- **Service:**
    - TradeDocumentService
        - Handles document storage, retrieval, and updates.
- **Model:**
    - **TradeDocument Entity**
        - Attributes:
            - documentId (PK)
            - documentType
            - referenceNumber
            - uploadedBy
            - uploadDate
            - status

3.4 Risk Assessment Module

Purpose: Evaluates risks in trade finance transactions.

- **Controller:**
  - RiskAssessmentController
    - analyzeRisk()
    - getRiskScore()
- **Service:**
  - RiskAssessmentService
    - Implements risk algorithms and calculates scores.
- **Model:**
  - **RiskAssessment Entity**
    - Attributes:
      - riskId (PK)
      - transactionReference
      - riskFactors
      - riskScore
      - assessmentDate

3.5 Compliance and Regulatory Reporting Module

Purpose: Ensures compliance and generates regulatory reports.

- **Controller:**
  - ComplianceController
    - generateComplianceReport()
    - submitRegulatoryReport()
- **Service:**
  - ComplianceService
    - Validates trade transactions against compliance requirements.
- **Model:**
  - **Compliance Entity**
    - Attributes:
      - complianceId (PK)
      - transactionReference
      - complianceStatus
      - remarks
      - reportDate

4. Database Schema

4.1 Table Definitions

1. **LetterOfCredit Table**

```
CREATE TABLE LetterOfCredit (
  lcId INT AUTO_INCREMENT PRIMARY KEY,
  applicantName VARCHAR(100),
```

```
    beneficiaryName VARCHAR(100),
    amount DECIMAL(15, 2),
    currency VARCHAR(10),
    expiryDate DATE,
    status ENUM('Open', 'Amended', 'Closed')
);
```

2. **BankGuarantee Table**

```
CREATE TABLE BankGuarantee (
    guaranteeId INT AUTO_INCREMENT PRIMARY KEY,
    applicantName VARCHAR(100),
    beneficiaryName VARCHAR(100),
    guaranteeAmount DECIMAL(15, 2),
    currency VARCHAR(10),
    validityPeriod DATE,
    status ENUM('Pending', 'Issued', 'Expired')
);
```

3. **TradeDocument Table**

```
CREATE TABLE TradeDocument (
    documentId INT AUTO_INCREMENT PRIMARY KEY,
    documentType VARCHAR(50),
    referenceNumber VARCHAR(50),
    uploadedBy VARCHAR(50),
    uploadDate DATE,
    status ENUM('Active', 'Archived')
);
```

4. **RiskAssessment Table**

```
CREATE TABLE RiskAssessment (
    riskId INT AUTO_INCREMENT PRIMARY KEY,
    transactionReference VARCHAR(50),
    riskFactors JSON,
    riskScore DECIMAL(5, 2),
    assessmentDate DATE
);
```

5. **Compliance Table**

```
CREATE TABLE Compliance (
    complianceId INT AUTO_INCREMENT PRIMARY KEY,
    transactionReference VARCHAR(50),
    complianceStatus ENUM('Compliant', 'Non-Compliant'),
    remarks VARCHAR(255),
    reportDate DATE
);
```

5. Local Deployment Details

1. **Environment Setup:**
    - Install MySQL or SQL Server for database.
    - Configure application.properties or appsettings.json with database credentials.
    - Ensure JDK or .NET SDK is installed based on the development framework.
2. **Deployment Steps:**
    - Clone the repository.
    - Build the application using Maven (Java) or Visual Studio (ASP.NET Core).
    - Run the server and access it locally at the designated port (e.g., http://localhost:8080 or http://localhost:5000).

6. Conclusion

The **Trade Finance Management System** simplifies trade finance operations, ensuring regulatory compliance and efficient processing. Its modular architecture facilitates scalability and maintainability for banks and financial institutions.