Problem Statement: Database Security and Secure Development Practices

You are tasked with securing an ASP.NET Core web application that uses a relational database (e.g., SQL Server, MySQL) for storing sensitive user data, including personal information, financial details, and authentication credentials. Your goal is to implement database security best practices, secure coding practices, and ensure that all data interactions are safe from common threats like SQL Injection, unauthorized access, and data breaches.

The application must adhere to secure development lifecycle practices, including security assessments and code reviews. It must also implement encryption for sensitive data and ensure secure connections between the application and the database.

User Stories:

User Story 1: Database Encryption and Secure Connections

As a developer,
I want to ensure that sensitive data is encrypted both at rest and in transit
So that it remains protected from unauthorized access and breaches.

Acceptance Criteria:

- Implement **Transparent Data Encryption (TDE)** or **column-level encryption** in the database to encrypt sensitive data such as user passwords, financial information, and personal details.
- Use **SSL/TLS** to ensure secure connections between the application and the database, preventing Man-in-the-Middle (MITM) attacks.
- All data transmitted between the server and the database should be encrypted using SSL, with the appropriate certificates installed.
- Sensitive data such as passwords should never be stored in plain text. Use a secure hashing algorithm (e.g., **bcrypt** or **PBKDF2**) for password storage.

User Story 2: Secure Coding Practices and Code Reviews

As a developer,
I want to follow secure coding practices and conduct code reviews
So that my application is free from vulnerabilities and secure against attacks like SQL Injection and Cross-Site Scripting (XSS).

Acceptance Criteria:

- Ensure that all **SQL queries** are parameterized or use **ORMs** like Entity Framework to avoid SQL Injection vulnerabilities.
- Review and follow secure coding guidelines to protect the application from XSS, CSRF, and other common web vulnerabilities.
- Conduct regular **code reviews** to identify potential security flaws and fix them before deployment.
- Use **static code analysis** tools to detect vulnerabilities in the codebase and ensure best practices are followed.
- Implement proper **input validation** and **output encoding** for all user inputs and outputs.

User Story 3: Security Assessments and Penetration Testing

As a security analyst,
 I want to conduct security assessments and penetration testing
 So that I can identify and address any vulnerabilities in the database and application before deployment.

Acceptance Criteria:

- Perform regular **security assessments** to identify potential weaknesses in the application and database, including improper data access, weak encryption, or inadequate input validation.
- Conduct **penetration testing** (e.g., SQL Injection, XSS) to simulate real-world attack scenarios and identify possible threats.
- Review the results of the security assessments and penetration tests, and apply necessary security patches or fixes to the codebase and database configurations.
- Ensure all vulnerabilities found during security assessments are remediated before deployment.

User Story 4: Secure Database Access and Permissions

As a database administrator,
 I want to ensure that access to the database is restricted to only authorized users
 So that sensitive data remains protected from unauthorized access.

Acceptance Criteria:

- Implement **least privilege** access control by granting only the necessary database permissions to users and applications.
- Use **role-based access control (RBAC)** to manage user permissions, ensuring that only authorized users can access sensitive tables or data.

- Ensure that **database credentials** (e.g., connection strings, passwords) are stored securely using tools like **Azure Key Vault** or **AWS Secrets Manager**, and never hardcoded in the application.
- Configure database auditing to log any suspicious access attempts and monitor for unusual activity.

User Story 5: Secure Development Lifecycle (SDLC) for Database Security

As a project manager,
I want to integrate security assessments throughout the SDLC
So that security is part of every stage of development and deployment.

Acceptance Criteria:

- Include **security assessments** in every stage of the SDLC (e.g., design, development, testing, and deployment) to identify and mitigate security risks early.
- Ensure that **code reviews** are conducted with a focus on security, especially for database interactions and user input handling.
- Implement **secure coding standards** and best practices in all development documentation, ensuring that all developers follow them.
- Use **automated security testing tools** to check for common vulnerabilities like SQL Injection, XSS, and buffer overflow during the testing phase.
- Establish a process for **vulnerability management** to prioritize and address discovered security issues before deployment.

Additional Requirements:

- The application must store user data securely, using encryption or secure hashing techniques.
- Perform **input sanitization** and **parameterized queries** to avoid SQL Injection vulnerabilities.
- Ensure that sensitive data is never logged or exposed through application logs.
- Use **HMAC (Hash-based Message Authentication Code)** for ensuring the integrity of sensitive data.