BillBuddy Security Architecture Report

1. Common Security Threats in Software Architecture

- **Authentication and Authorization Attacks** – Weak authentication mechanisms may allow attackers to gain unauthorized access to sensitive data.
- **SQL Injection** – Malicious users can inject SQL queries to manipulate or delete financial records in the database.
- **Cross-Site Scripting (XSS) and Cross-Site Request Forgery (CSRF)** – Attackers can inject malicious scripts into the application or force users to execute unintended actions.
- **Data Breaches and Exposure** – Inadequate encryption and improper access controls can lead to sensitive financial data being leaked or stolen.
- **Denial of Service (DoS) and Distributed Denial of Service (DDoS) Attacks** – Attackers can overwhelm the system with excessive requests, causing service disruption.

2. BillBuddy Security Architecture

User Interface Security

The mobile application uses OAuth 2.0 and JWT (JSON Web Tokens) for user authentication. Client-side input validation is implemented to prevent common injection attacks before data is sent to the backend.

Back-End Security

The backend is built using Spring Boot with Spring Security, enforcing role-based access control (RBAC) to limit user permissions. API requests are secured using JWT authentication to ensure only authorized users can access protected resources. Rate limiting is implemented to prevent brute-force attacks.

Database Security

The database layer enforces prepared statements to prevent SQL injection attacks. Sensitive user data, such as passwords, is hashed using bcrypt, and financial records are encrypted using AES-256 encryption.

3. Security Risks and Mitigation Strategies

- **Unauthorized Access -** where attackers attempt to bypass authentication to gain access to financial records. This risk is mitigated through Spring Security with JWT authentication and multi-factor authentication to add an additional layer of security.
- **SQL injection -** where attackers craft malicious SQL queries to manipulate the database. To prevent this, the system strictly enforces prepared statements and ORM-based data access to sanitize inputs before executing database operations.
- **Data Breaches -** which could expose users' financial records. To reduce this risk, AES-256 encryption is used for sensitive data storage, and all passwords are stored using bcrypt hashing. Regular security audits and automated penetration testing help identify vulnerabilities before they can be exploited.

4. Impacts

- **Security Patch Management**

Timely updates: Regular application of security patches can reduce vulnerabilities, but frequent updates may increase the maintenance burden.
Compatibility: Security patches may cause compatibility issues that require additional testing and tuning, increasing maintenance complexity.

- **Access Control**

Permission Management: Strict access control can reduce risk, but complex permission settings increase management difficulties.
Auditing and monitoring: Real-time monitoring and auditing of access behavior contributes to security, but requires additional resources to maintain these systems.
User management: Managing user accounts and permissions (e.g., adding, deleting, modifying) increases the maintenance effort.

- **Code Complexity**

Security mechanisms: Security measures such as encryption and input validation add to code complexity, affecting readability and maintainability.
Technical Debt: Quick fixes to security issues may introduce technical debt, increasing maintenance costs in the long term.