PASTA worksheet

Stages	Sneaker company
I. Define business and security objectives	Make **2-3 notes** of specific business requirements that will be analyzed. ● *Will the app process transactions?* *The app will handle transactions between buyers and sellers, indicating the need for secure payment processing, multiple payment options, and the prevention of fraud. Ensuring clear, fast, and reliable payment handling will also support a positive user experience and avoid legal complications.* ● *Does it do a lot of back-end processing?* *The company emphasizes data privacy as a priority, so the app should implement robust privacy protections and secure account management. Users' personal data must be safeguarded to instill confidence in data handling practices.* ● *Are there industry regulations that need to be considered?* *Direct messaging between buyers and sellers and a seller rating feature are integral to the app's functionality. These elements need to be secured against spam and unauthorized access to maintain a safe and user-friendly platform.*
II. Define the technical scope	List of technologies used by the application: ● *Application programming interface (API)* ● *Public key infrastructure (PKI)* ● *SHA-256* ● *SQL* *I would prioritize evaluating the **Application Programming Interface (API)** first. Since the app relies on APIs for interactions with external services, any vulnerabilities in the API endpoints could expose sensitive user data or allow unauthorized access.*

	Ensuring secure API access and protecting data exchanges is critical to prevent breaches and data leaks, especially when sensitive transactions are involved. After securing the APIs, the focus could shift to PKI, which safeguards key exchanges and encryption, as well as SQL, which manages user and transaction data.
III. Decompose application	Sample data flow diagram **APIs**: APIs play a crucial role in the data flow, particularly for searching and retrieving information about shoes for sale. Ensuring APIs are secure with proper authentication and authorization controls will prevent unauthorized data access and data leakage during requests. **Public Key Infrastructure (PKI)**: PKI is essential for encrypting data exchanges between the app and its servers, especially when handling sensitive data like credit card information. Using AES for data encryption and RSA for key exchange ensures that even if intercepted, data remains protected. **SHA-256**: For processes involving sensitive user data, such as storing passwords or credit card details, SHA-256 hashing ensures that this information is stored securely in the database. Hashing passwords with SHA-256 adds a layer of security by making it extremely difficult for attackers to reverse-engineer sensitive information. **SQL**: SQL databases store all sneaker listings and related user information. Implementing SQL security best practices—such as parameterized queries to prevent SQL injection and access controls for sensitive tables—ensures data integrity and protects the database against injection attacks.

IV. Threat analysis	List **2 types of threats** in the PASTA worksheet that are risks to the information being handled by the application. ● *What are the internal threats?* ***Internal Threat - SQL Injection**: Employees or contractors with malicious intent, or external attackers who manage to bypass front-end controls, could exploit SQL injection vulnerabilities to access or manipulate sensitive database information. Regularly auditing system logs and implementing parameterized queries will mitigate this threat by detecting and preventing unauthorized database access.* ● *What are the external threats?* ***External Threat - API Abuse**: Attackers might exploit insecure API endpoints to perform unauthorized actions, extract sensitive user data, or launch denial-of-service (DoS) attacks. This threat could result in data breaches or service disruptions. Monitoring API requests through internal logs can help identify unusual activity and prevent abuse.*
V. Vulnerability analysis	List **2 vulnerabilities** in the PASTA worksheet that could be exploited. ● *Could there be things wrong with the codebase?* ● *Could there be weaknesses in the database?* ● *Could there be flaws in the network?* ***API Endpoint Exposure**: Unprotected or poorly secured API endpoints are vulnerable to attacks such as unauthorized access or data extraction (sensitive data exposure). Without proper authentication and input validation, attackers could access or manipulate user data. Following OWASP guidelines for secure API development and enforcing token-based authentication can mitigate this risk.* ● ***SQL Injection**: If the app's codebase doesn't implement parameterized queries, it may be susceptible to SQL injection attacks. This could allow attackers to execute arbitrary SQL commands, gaining unauthorized access to the database and sensitive information. Regularly testing for SQL injection vulnerabilities and applying database access controls are critical defenses.*

VI. Attack modeling	Sample attack tree diagram ***API Endpoint Exposure Attack Path:*** *Inject malicious requests to access sensitive data.* *Bypass authentication controls.* ***SQL Injection Attack Path****:* *Execute unauthorized SQL queries.* *Access user account information.* *Modify or delete sensitive records.* *Insert malicious data or create backdoors for further attacks.*
VII. Risk analysis and impact	List **4 security controls** that you've learned about that can reduce risk. ***API Authentication and Authorization****: Implement token-based authentication (such as OAuth) for all API endpoints to ensure only authorized users access sensitive data. Regularly review and update permissions to minimize exposure.* ***Parameterized SQL Queries****: Use parameterized queries and prepared statements to prevent SQL injection attacks. This reduces the risk of unauthorized database access by ensuring that user inputs are handled securely.* ***Data Encryption****: Employ AES encryption for sensitive data (such as payment details) in transit and at rest. Encrypting data minimizes the potential impact of unauthorized access, as intercepted data remains protected.* ***Web Application Firewall (WAF)****: Deploy a firewall to monitor and filter incoming traffic, blocking malicious requests and protecting against common web-based attacks like API abuse and SQL injection. A WAF provides a proactive defense layer by identifying and stopping suspicious patterns in real time.*