PASTA worksheet

Stages	**Sneaker company**
I. Define business and security objectives	● *Users must be able to sign-up* ● *Users must be able to log in* ● *Users must be able to manage their account* ● *Data privacy is important* ● *Buyers and sellers must be able to communicate* ● *Buyers should be able to rate sellers* ● *Sales should be quick and easy to process* ● *Users should have several payment options* ● *Proper payment handling is critical in order to avoid legal issues*
II. Define the technical scope	List of technologies used by the application: ● *API* ● *PKI* ● *AES* ● *SHA-256* ● *SQL* Out of the above list of technology to be employed, we should prioritize investigating possible threats caused by any APIs selected for use in this application, as they will likely represent the majority of data flowing back and forth over the platform.

III. Decompose application	Sample data flow:
	1. User Registration and Login
	    a. New User -> Sign Up Form -> Verification (via Email/SMS) -> User Database
	    b. Returning User -> Login Form -> Authentication -> User Session
	2. Account Management
	    a. Logged-in User -> Account Settings Form -> User Account Updates -> User Database
	3. Communication Between Buyers and Sellers
	    a. Buyer -> Message Form -> Message Database ->

	Seller (and vice versa) 4. Rating Sellers a. Buyer -> Rating Form -> Rating Database -> Seller Rating Update 5. Processing Sales a. Buyer -> Product Selection -> Shopping Cart -> Checkout Form -> Payment Gateway -> Payment Confirmation -> Order Database -> Seller 6. Payment Options a. Buyer -> Checkout Form -> Selection of Payment Method -> Payment Gateway 7. Payment Handling a. Payment Gateway -> Payment Validation -> Secure Transaction Database
IV. Threat analysis	1. An internal threat could come from a disgruntled employee, an employee who mistakenly exposes sensitive data, or an employee who is tricked into giving access to secure data (often through social engineering). For example, an employee with access to the app's database might intentionally or accidentally expose user information, including personal and payment data. This type of threat could also come from poor internal security policies or lack of employee training. 2. An external threat from outside the organization could include hackers or other malicious actors attempting to exploit vulnerabilities in the app to gain unauthorized access, disrupt services, or steal sensitive information. One example could be a hacker launching a phishing campaign aimed at the application's users, tricking them into providing their login credentials, which the attacker then uses to access their accounts and potentially steal personal and payment information.
V. Vulnerability analysis	*1. If the app's codebase doesn't properly validate user input, it could be vulnerable to injection attacks, such as SQL injection or Cross-Site Scripting (XSS). In an SQL injection attack, a malicious actor could potentially manipulate SQL queries to access, modify, or delete data in the database. In an XSS attack, an attacker could inject malicious scripts into web pages viewed by other users, potentially leading to unauthorized access to their data.*

	2. *If the network on which the app's servers are hosted isn't properly secured, it could be vulnerable to various types of attacks. For example, if there are unsecured access points or open ports, an attacker could potentially gain unauthorized access to the network and, from there, access the app's servers or other sensitive resources.*
VI. Attack modeling	Sample attack tree diagram 1. User Data: Attackers can use user data for identity theft or phishing. 2. SQL Injection: Attackers can input malicious SQL code into user inputs to manipulate databases. 3. Lack of Prepared Statements: This can make an application vulnerable to SQL injection attacks. 4. Session Hijacking: Attackers can steal a user's session cookie to impersonate them. 5. Weak Login Credentials: Attackers can guess or crack weak passwords to gain unauthorized access to user accounts.
VII. Risk analysis and impact	List **4 security controls** that you've learned about that can reduce risk. Four (4) security controls that could reduce risk to this application: 1. Input Sanitization: Validate and sanitize user inputs to prevent SQL injection. 2. Strong Authentication: Enforce complex password policies and multi-factor authentication (MFA) to strengthen login security. 3. Prepared Statements: Use prepared statements in database queries to prevent SQL injection attacks. 4. Data Encryption: Encrypt sensitive data both at rest and in transit to protect against unauthorized access.