Security evaluation of the proposed tech stack:

Node & Express are the chosen server-side framework architecture, however most security features need to be implemented via middleware libraries. Each third-party service and libraries increases risk as each addition can potentially introduce vulnerabilities, reliance on libraries for security feature integration means regular updates and vulnerability checks are vital to mitigate the risk of using outdated or compromised packages.

It is also very important that servers are properly configured in order to secure against web vulnerabilities, Authorization protocols such as OAuth and JWTs should be implemented for proper token validation, secure storage, and handling mechanisms.

Using Joi for input validation can ensure secure user inputs if used to enforce strict data types and sanitize external input. File uploads can be handled securely with the use of Multer's size and type limiting with the use of third party malware scanning via the VirusTotal API.

Data encryption using AES256 bit level standard will be used for protecting the sensitive data this application will encounter. Table-and field-level encryption to be used to securely store data at rest. Careful management of encryption keys is required. As stated, the proposed application will undoubtedly handle sensitive data, meaning adherence to high quality standards such as GDPR, PCI DSS and ISO 27001 should be a priority in order to achieve a high level of security.

Input types:

Personally identifiable information data:

- Personal Identifiers: Usernames, passwords, and physical addresses.
- Contact Details: Email addresses and hone numbers for communication and verification purposes.
- Financial Transactions: Credit/debit card information, billing addresses, and related transaction details.

Merchant Data:

- Business Identifiers: Merchant documents such as business identification information, tax documents, address.
- Sales and Inventory: Transaction history, sales trends, product catalogs, and current stock levels.

System-Generated Data:

- Authentication Tokens: Session identifiers, cookies, and anti-CSRF tokens to secure user logins and transactions.
- Performance Metrics: Data reflecting system usage, application performance, and service health status.

External Data:

- API Communications: Interactions with external services like payment vendors, shipping couriers and any other third-party applications.

Interaction with endpoints:

- Query Data: Search terms, filter options, or other commands
- Authorization Credentials: HTTP header data containing tokens or keys for secure API access.

Administrative Inputs:

- Dashboard Management: Commands issued via the admin panel for product listings, user roles, content approval, and system settings.

Data Categories:

1. User Authentication Data

2. Personal Data

3. Payment Information

4. Order and Transaction Information

5. Customer Service Interactions

6. Access Control Data

7. System and Operational Data

8. Legal and Compliance Data

10. Application Security Data

12. External API Data

13. Dependency and Component Data

Threat Model

Spoofing Identity

Threats:

- Illicit access to merchant portals, leading to unauthorized listing or pricing changes.
- Capture of merchant credentials via social engineering or phishing.

Mitigations:

- Use MFA for all merchant logins.
- Ensure strong password policies are enforced.
- Validate all input data rigorously.
- Utilize security tokens and HTTPS for secure communication channels.

Elevation of Privilege

Threats:

- Exploiting vulnerabilities to gain unauthorized access to merchant administrative privileges.
- Merchants gaining access to another merchant's data or the platform's administrative functions.

Mitigations:

- Regularly update and patch systems to fix known security vulnerabilities.
- Conduct thorough access control checks and periodic reviews of user privileges.
- Segregate merchant accounts and restrict the scope of each merchant's permissions.

Tampering with Data

Threats:

- Unauthorized alteration of product information, prices, or descriptions.
- Modification of order data or shipping information.
- SQL injection attacks that could alter or corrupt database data.

Mitigations:

- Implement strong access controls for database management systems and use role-based access control for merchants.
- Use input validation and prepared SQL statements to prevent injection attacks.

Repudiation

Threats:

- Dispute over the integrity of transaction logs or order fulfillment records.
- Merchants denying having made changes to product listings or prices when they have done so.

Mitigations:

- Use secure logging for all critical actions by merchants and the platform itself, with timestamps and user ID’s to provide non-repudiable logs.

Information Disclosure

Threats:

- Exposure of sensitive merchant financial information such as bank account or credit card details.
- Leaking of personal data of customers or confidential product information.
- Disclosure of system configuration and internal IP addresses through error messages or logs.

Mitigations:

- Encrypt all data at rest and in transit using robust encryption standards.
- Mask personal and financial details in logs.
- Apply proper error handling to prevent sensitive data leaks through stack traces or error outputs.

Denial of Service

Threats:

- Overwhelming the site with traffic, causing slow response times or complete service outages.
- Flooding merchant inboxes with support requests, causing legitimate queries to be missed.

Mitigations:

- Employ DDoS mitigation services.
- Use rate-limiting on login and API calls to prevent brute force attacks.
- Implement a queue system for customer service requests to manage high volumes.