**EPIC:**

In the Scaled Agile Framework (SAFe), an Epic is a large body of work that can be broken down into smaller, more manageable pieces of work called Features. Epics typically span multiple sprints and often require contributions from multiple Agile Release Trains (ARTs) or teams to complete.

**Example of an Epic:**

**Epic:** Implementing a New Payment Gateway

**Description**: The Epic involves implementing a new payment gateway system to improve the online payment process for customers. It includes various functionalities such as integrating with different payment providers, enhancing security measures, updating user interfaces, and ensuring compatibility with existing systems.

**FEATURES:**

In SAFe, a Feature is a substantial piece of functionality that delivers value to customers and stakeholders. It is a key element of the Program Backlog, managed through the framework’s processes to ensure alignment with business goals and effective delivery across Agile teams.

**Example of a Features using the Epic Description given above for Implementing a New Payment Gateway:**

1. Payment Provider Integration: Integrate the new payment gateway with multiple payment providers such as PayPal, Stripe, and Square.
2. Enhanced Security Measures: Implement additional security features such as two-factor authentication and encryption to safeguard customer payment information.
3. User Interface Updates: Update the checkout process and user interfaces to provide a seamless payment experience for customers across different devices and platforms.
4. Compatibility Testing: Conduct thorough testing to ensure compatibility with existing systems, databases, and third-party applications.
5. Performance Optimization: Optimize the performance of the payment gateway to handle high volumes of transactions efficiently and minimize latency.

**Note:** Each of these Features can then be further broken down into User Stories, which are smaller, actionable items that can be completed within a single sprint.

**USER STORIES:**

User Stories are short, simple descriptions of a feature told from the perspective of the person who desires the new capability, usually a user or customer. They typically follow a template format: "As a [user role], I want [action] so that [benefit]".

**An example of a User Story using the New Payment Gateway features mention above:**

**Feature 1-User Story :**

As a user, I want to be able to pay using PayPal so that I have the option to use my PayPal account for transactions.

* Acceptance Criteria:
  + The payment gateway must be integrated with PayPal.
  + Users should see PayPal as an option during checkout.
  + Payment through PayPal should be processed correctly and reflected in the user's order status.

**TASKS:**

User stories are further split into smaller implemented activities. Here’s how you can split the user story "As a user, I want to be able to pay using PayPal so that I have the option to use my PayPal account for transactions" into specific tasks:

By breaking down the user story into these tasks, you ensure a structured approach to integrating PayPal into the payment system, covering everything from initial research and development to testing and deployment.

**An example of a Task using the User Story of the New Payment Gateway features mention above:**

1. Research PayPal Integration Requirements

* Review PayPal’s API documentation and integration guidelines.
* Identify required credentials and setup procedures for PayPal integration.

1. Obtain PayPal API Credentials

* Create a PayPal developer account if not already available.
* Generate API credentials (Client ID, Secret) for sandbox and production environments.

1. Implement PayPal Payment Gateway Integration

* Develop backend code to integrate with PayPal’s payment API.
* Implement necessary API calls for initiating and completing payments.
* Handle PayPal-specific payment data and responses.

1. Update Frontend Checkout Interface

* Add a PayPal payment option to the checkout page.
* Design and implement a PayPal button or payment method selection UI.
* Ensure the PayPal payment option is clearly labeled and accessible.

1. Develop Payment Confirmation and Handling

* Implement logic to handle PayPal payment confirmation and transaction status.
* Update order status based on PayPal payment results (e.g., success, failure).
* Ensure accurate reflection of transaction status in the user’s order history.

1. Test PayPal Payment Integration

* Conduct testing in the PayPal sandbox environment to validate integration.
* Perform test transactions to verify successful payments and proper handling of various scenarios (e.g., payment failure, refunds).
* Test the integration across different browsers and devices to ensure consistency.

1. Ensure Security and Compliance

* Verify that PayPal payment data is handled securely, following best practices (e.g., PCI compliance).
* Implement any required security measures such as tokenization and encryption.

1. Update Documentation and User Instructions

* Document the integration process, including setup and configuration steps.
* Update user-facing documentation to provide instructions for using PayPal as a payment option.

1. Deploy PayPal Integration to Production

* Deploy the integrated PayPal payment gateway to the production environment.
* Perform a final round of testing to ensure smooth operation in the live environment.

1. Monitor and Address Post-Deployment Issues

* Monitor the PayPal payment integration for any issues or errors.
* Address and resolve any bugs or user-reported issues related to PayPal payments.