Detailed breakdown of **epic-wise user stories** for the **Generative AI for Security Vulnerability Remediation** project. The user stories are designed to align with the **agentic AI execution** approach, ensuring modularity, portability, and comprehensive documentation 🚀.

Epic 1: Web Dashboard and REST API

User Stories:

US-1 : As a user, I want to access a web dashboard to visualize potential vulnerabilities affecting my application.

- **Acceptance Criteria:**
  - A web dashboard is available with a clean and intuitive UI.
  - The dashboard displays a summary of vulnerabilities (e.g., critical, high, medium, low).
  - The dashboard allows users to initiate scans and view scan results.
- **Execution:**
  - Implement a React-based web dashboard.
  - Integrate the REST API to fetch and display scan results.
  - Generate documentation with mermaid diagrams for the UI flow.

US-2 : As a user, I want to upload my codebase as a zip file so that I can analyze it for vulnerabilities.

- **Acceptance Criteria:**
  - A file upload button is available on the frontend.
  - The backend accepts zip files and extracts them for scanning.
  - The system validates the zip file structure to ensure it contains valid code.

    - The user is notified if the upload fails (e.g., invalid file type or corrupted zip).

- **Execution:**
    - Implement a file upload component in React.
    - Backend API to handle file uploads and extraction.
    - Add validation logic for zip files.
    - Generate documentation with mermaid diagrams for the file upload flow.

US-3 : As a user, I want to connect my GitHub repository so that I can analyze my codebase directly from GitHub.

- **Acceptance Criteria:**
    - A GitHub integration button is available on the frontend.
    - The user can authenticate with GitHub using OAuth.
    - The system fetches the repository and allows the user to select a branch or commit for scanning.
    - The user is notified if the GitHub integration fails (e.g., invalid token or repository not found).
- **Execution:**
    - Implement GitHub OAuth integration in React.
    - Backend API to handle GitHub repository fetching.
    - Add error handling for GitHub API failures.
    - Generate documentation with mermaid diagrams for GitHub integration flow.

Epic 2: Vulnerability Scanning Service

User Stories:

US-4 : As a developer, I want the system to use industry-standard scanning tools to identify vulnerabilities.

- **Acceptance Criteria:**
  - The system integrates with OWASP ZAP or similar scanning tools.
  - The scanning parameters are configurable based on project type.
  - The system supports multiple scanning profiles for different security needs.
  - The scan results are standardized for further processing.
- **Execution:**
  - Implement integration with OWASP ZAP API.
  - Add configuration options for scanning parameters.
  - Create scanning profiles for common project types.
  - Generate documentation with mermaid diagrams for the scanning flow.

US-5 : As a security analyst, I want detailed vulnerability reports generated after each scan.

- **Acceptance Criteria:**
  - The system generates comprehensive reports with vulnerability details.
  - The reports include severity ratings, affected components, and CWE references.
  - The reports are available in multiple formats (e.g., PDF, JSON).
  - The reports are stored in the database for future reference.
- **Execution:**
  - Implement report generation functionality.

    - Add severity rating calculation.
    - Create export options for different formats.
    - Generate documentation with mermaid diagrams for the reporting flow.

Epic 3: AI Model Selection and Configuration

User Stories:

US-6 : As a user, I want to select different LLM models (e.g., GPT-4, Claude, etc.) so that I can choose the best model for my needs.

- **Acceptance Criteria:**
    - A dropdown menu is available on the frontend to select LLM models.
    - The backend supports multiple LLM APIs (e.g., OpenAI, Anthropic, etc.).
    - The user can switch between models without restarting the application.
    - The system defaults to a pre-configured model if no selection is made.
- **Execution:**
    - Implement a dropdown component in React for model selection.
    - Backend API to handle LLM model switching.
    - Add configuration files for supported LLM models.
    - Generate documentation with mermaid diagrams for LLM model integration.

US-7 : As a user, I want to set my own API keys for LLM models so that I can use my own credits.

- **Acceptance Criteria:**
    - An input field is available on the frontend for API key entry.
    - The backend securely stores the API keys (encrypted).
    - The user can update or remove API keys at any time.

    - The system validates the API key before allowing usage.
- **Execution:**
    - Implement an API key input component in React.
    - Backend API to handle API key storage and validation.
    - Add encryption logic for API key storage.
    - Generate documentation with mermaid diagrams for API key management.

Epic 4: Generative AI Analysis Module

User Stories:

US-8 : As a DevOps user, I want the AI module to analyze vulnerability data and suggest actionable remediation steps.

- **Acceptance Criteria:**
    - The Generative AI module processes scan results and generates remediation steps.
    - The AI-generated recommendations are displayed on the dashboard.
    - The user can view detailed explanations for each recommendation.
    - The recommendations include code snippets for fixes when applicable.
- **Execution:**
    - Implement the Generative AI module using a transformer-based model.
    - Integrate the AI module with the REST API to fetch scan results.
    - Add explanation generation for recommendations.
    - Generate documentation with mermaid diagrams for the AI integration flow.

US-9 : As a developer, I want to see AI-generated recommendations integrated dynamically in my workflow.

- **Acceptance Criteria:**
    - The AI recommendations are available in the developer's IDE (e.g., VS Code).
    - The recommendations are actionable and include code snippets for fixes.
    - The user can accept or reject the recommendations within the IDE.
    - The system tracks which recommendations were accepted or rejected.
- **Execution:**
    - Implement IDE integration for AI recommendations.
    - Add functionality to accept or reject recommendations.
    - Create tracking system for recommendation outcomes.
    - Generate documentation with mermaid diagrams for IDE integration.

Epic 5: Remediation Execution

User Stories:

US-10 : As a user, I want the system to automatically apply recommended fixes to vulnerabilities.

- **Acceptance Criteria:**
    - The automated remediation service executes fixes based on AI recommendations.
    - The user can review the fixes before they are applied.
    - The system creates backup copies before applying fixes.
    - The system logs all remediation actions for future reference.

- **Execution:**
    - Implement the automated remediation service.
    - Add functionality to review and approve fixes.
    - Create backup mechanism for code changes.
    - Generate documentation with mermaid diagrams for the remediation flow.

US-11 : As a developer, I want to approve or reject automated fixes before application.

- **Acceptance Criteria:**
    - The system presents a diff view of proposed changes.
    - The user can approve or reject individual changes.
    - The system provides explanations for each proposed change.
    - The approved changes are applied automatically.
- **Execution:**
    - Implement diff view for proposed changes.
    - Add approval/rejection functionality.
    - Create explanation generation for changes.
    - Generate documentation with mermaid diagrams for the approval flow.

Epic 6: Remediation Logging and Reporting

User Stories:

US-12 : As an admin, I want to review a log of remediation actions taken, ensuring transparency.

- **Acceptance Criteria:**
  - A log of all remediation actions is available in the dashboard.
  - The log includes details such as the vulnerability, the fix applied, and the timestamp.
  - The log can be filtered and searched for specific actions.
  - The log can be exported for compliance reporting.
- **Execution:**
  - Implement a remediation log in the dashboard.
  - Add filtering and search functionality.
  - Create export options for logs.
  - Generate documentation with mermaid diagrams for the logging system.

US-13 : As a security analyst, I want to track the effectiveness of automated remediations.

- **Acceptance Criteria:**
  - The system tracks success/failure metrics for remediation attempts.
  - The dashboard displays effectiveness metrics with visualizations.
  - The user can view trend reports for security improvements.
  - The metrics can be exported for reporting.
- **Execution:**
  - Implement metrics tracking for remediations.

    - Add visualization components for effectiveness metrics.
    - Create trend report generation.
    - Generate documentation with mermaid diagrams for metrics tracking.

Epic 7: Database and Monitoring

User Stories:

US-14 : As a user, I want the system to store vulnerability data and remediation logs in a database for future reference.

- **Acceptance Criteria:**
    - The database stores vulnerability reports, AI recommendations, and remediation logs.
    - The data is accessible via the REST API.
    - The database is backed up regularly to prevent data loss.
    - The data is securely stored with proper access controls.
- **Execution:**
    - Implement the database using MongoDB or SQLite.
    - Add backup functionality for the database.
    - Create access control for data security.
    - Generate documentation with mermaid diagrams for the database schema.

US-15 : As an admin, I want to monitor the system's performance and logs to ensure it is running smoothly.

- **Acceptance Criteria:**
    - A monitoring dashboard is available with real-time metrics (e.g., CPU usage, memory usage).

  - The system logs are accessible via the dashboard.
  - The admin is notified of any critical issues (e.g., database failure).
  - The monitoring data is stored for historical analysis.
- **Execution:**
  - Implement a monitoring dashboard using Prometheus or ELK Stack.
  - Add notification functionality for critical issues.
  - Create log storage for historical analysis.
  - Generate documentation with mermaid diagrams for the monitoring system.

Epic 8: CI/CD Pipeline Integration

User Stories:

US-16 : As a DevOps user, I want the solution to integrate seamlessly with my CI/CD pipeline (e.g., Jenkins, GitHub Actions).

- **Acceptance Criteria:**
  - The solution can be integrated into existing CI/CD pipelines.
  - The system triggers scans and remediation automatically during the build process.
  - The CI/CD pipeline logs are accessible via the dashboard.
  - The integration is configurable for different CI/CD platforms.
- **Execution:**
  - Implement CI/CD integration using Jenkins or GitHub Actions.
  - Add functionality to trigger scans and remediation during the build process.
  - Create dashboard integration for CI/CD logs.
  - Generate documentation with mermaid diagrams for CI/CD integration.

US-17 : As a developer, I want the system to provide real-time feedback on vulnerabilities during the CI/CD process.

- **Acceptance Criteria:**
  - The system provides real-time feedback on vulnerabilities during the build process.
  - The developer can view the feedback in the CI/CD pipeline logs.
  - The system blocks the build if critical vulnerabilities are detected.
  - The feedback includes remediation suggestions.
- **Execution:**
  - Implement real-time feedback in the CI/CD pipeline.
  - Add functionality to block builds for critical vulnerabilities.
  - Create remediation suggestion integration.
  - Generate documentation with mermaid diagrams for real-time feedback.

Epic 9: Feedback Loop and Continuous Improvement

User Stories:

US-18 : As a user, I want the system to continuously improve its AI recommendations based on my feedback.

- **Acceptance Criteria:**
  - The system collects user feedback on AI recommendations.
  - The feedback is used to retrain the AI model periodically.
  - The user is notified when the AI model has been updated.
  - The system provides metrics on improvement over time.

- **Execution:**
    - Implement a feedback collection mechanism.
    - Add functionality to retrain the AI model using feedback data.
    - Create notification system for model updates.
    - Generate documentation with mermaid diagrams for the feedback loop.

US-19 : As an admin, I want to track the effectiveness of the AI model over time.

- **Acceptance Criteria:**
    - The system provides metrics on the accuracy of AI recommendations.
    - The metrics are displayed in the dashboard with visualizations.
    - The admin can compare metrics between model versions.
    - The metrics can be exported for reporting.
- **Execution:**
    - Implement metrics tracking for the AI model.
    - Add visualization components for model metrics.
    - Create version comparison functionality.
    - Generate documentation with mermaid diagrams for metrics tracking.

Epic 10: System Portability and Documentation

User Stories:

US-20 : As a user, I want the application to be portable and lightweight so that I can run it on any PC or device.

- **Acceptance Criteria:**
  - The application can be packaged as a standalone executable or container.
  - The application runs on Windows, macOS, and Linux.
  - The application uses minimal system resources (CPU, RAM).
  - The application can be easily updated or reinstalled.
- **Execution:**
  - Package the application using containerization or executable packaging.
  - Test the application on multiple platforms.
  - Optimize the application for performance.
  - Generate documentation with mermaid diagrams for the packaging process.

US-21 : As a developer, I want detailed documentation with diagrams (mermaid, SVG) so that I can understand the system architecture and flow.

- **Acceptance Criteria:**
  - Documentation includes architecture diagrams in mermaid and SVG formats.
  - Documentation covers all major components (frontend, backend, AI, scanning, etc.).
  - Documentation is updated automatically with each code change.
  - Documentation is accessible via a `/docs` route in the application.

- **Execution:**
    - Generate mermaid diagrams for architecture, data flow, and component interactions.
    - Automate documentation updates using CI/CD.
    - Create a `/docs` route in the frontend to display documentation.
    - Add a script to generate SVG diagrams from mermaid files.

Key Points:

- **Diagram-Based Structure: 10 epics** with **21 user stories** organized according to the phase diagram.
- **Three-Phase Approach:** Core functionality divided into UI/Scanning, AI Integration, and Remediation phases.
- **Supporting Modules:** Database, CI/CD, and Feedback Loop as essential supporting components.
- **Documentation:** Every user story includes **agentic AI execution** steps for generating **mermaid diagrams** and **documentation**.
- **Continuous Working Application:** After each user story, the application remains functional and deployable. 🚀