Persistent

SASVA North Star

User Guide

V2.1.1

Table of Contents

Overview

North Star is designed to provide a comprehensive view of your product or application, helping you achieve your mission. It identifies gaps in your current implementation compared to competitors and market standards. Here's how North Star works:

1. **Current State Analysis**: North Star examines what has been implemented in your product so far. It assesses your codebase, including changes made in repositories like GitHub, and reviews past releases documented in JIRA.

2. **Market and Competitor Analysis**: The application studies market trends, innovations in your domain, and competitor activities. It also considers analyst insights to provide a well-rounded understanding of your product's position.

3. **Backlog Review**: North Star reviews your project management backlog to identify pending tasks and features that haven't been added yet. You can also input additional requirements.

4. **Future Planning**: Based on the information gathered, North Star provides a detailed plan for future releases. This includes the necessary additions to your product, the time required, and the number of releases needed to achieve your vision. Each release plan includes epics, stories, and tasks to guide your development process.

Use Cases

Product Enhancement: A company wants to enhance its existing product to stay competitive. North Star analyzes the current state, identifies gaps, and provides a roadmap for future enhancements, ensuring the product meets market demands.

Market Entry: A startup is planning to enter a new market with its product. North Star helps by analyzing market trends and competitor products and providing a detailed plan to align the product with market expectations.

Feature Prioritization: A development team is overwhelmed with a long list of potential features. North Star reviews the backlog, assesses market needs, and helps prioritize features that will provide the most value to users.

Release Planning: A company needs to plan its upcoming releases efficiently. North Star provides a structured plan with epics, stories, and tasks for each release, ensuring a smooth and strategic rollout of new features.

Competitive Analysis: A business wants to understand how its product compares to competitors. North Star conducts a thorough analysis of competitor products and market trends, offering insights and recommendations to improve the product's competitive edge.

Getting Started

Pre-requisites

Make sure to have the credentials below to add your project details:

Hardware Requirement

VCPU	Memory	OS Disk	Data Disk
8GB	32GB	100GB	500GB

Code Repository

For **GitHub:** Repository URL and Valid Token
For **GitLab**: Repository URL and Valid Token
For **Azure**: Repository URL, Username, and Valid Token
For **Bitbucket**: Repository URL, Username, and Valid Token

Project Management tool

For **Jira:** URL, Username, Key, and Token.
For **Aha**: URL and Token

Compatibility Matrix

This matrix outlines the supported and tested third-party tools and platforms for the current release of the North Star. Only the versions and configurations listed below are officially supported.

Category	Supported Tools/Platforms	Notes
Project Management Tools	\ Jira (Cloud & Server)	\ Integration via REST APIs
	\ Aha	
Code Repositories	\ GitHub \ GitLab (SaaS) \ Azure_repo \ Bitbucket	\ A personal access token is utilized to clone the repository.
Operating System	\ Ubuntu 20.04,22.04 \ RHEL 8.x, 9.x	\ Only x86_64 architectures are supported
Ticketing System	\ Jira	\ Integration via REST APIs
	\ Service Now	
Cloud Providers	\ AWS \ Microsoft Azure \ Google Cloud Platform (GCP) \ On-premises (VMWare/OpenStack)	\ Cloudnative deployment templates are available.
Deployments Modes	\ Docker (Compose) \ Kubernetes (v1.32 and above) \ VMs (QEMU/KVM, VMWare)	\ The Helm chart is under development by the DevOps Team.

North Star installation

Connect with the SASVA Support team for Application Installation and access.

Signing In to SASVA North Star

To access the SASVA North Star, you can choose from the following sign-in options based on your account type and preferences.

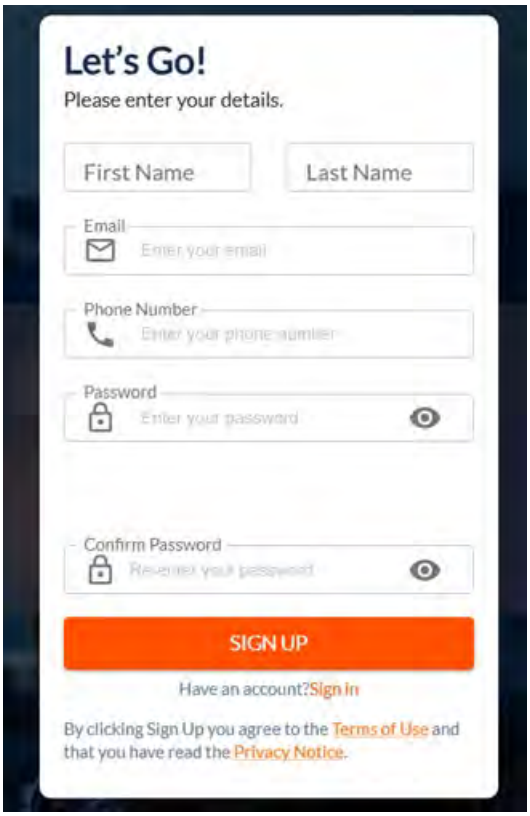
Sign in with SSO

To sign in with SSO (Single Sign-On), follow the given instructions:

1. On the SASVA North Star User Management **Login page**.
   Click the **SignIn with SSO** button.
2. Authenticate using your credentials.
3. Once authentication is complete, you will see a confirmation message.
4. Switch back to SASVA North Star.

New Account Creation

1. Click the **Create Account** option within the Login page.
   Sign Up Details page will appear.



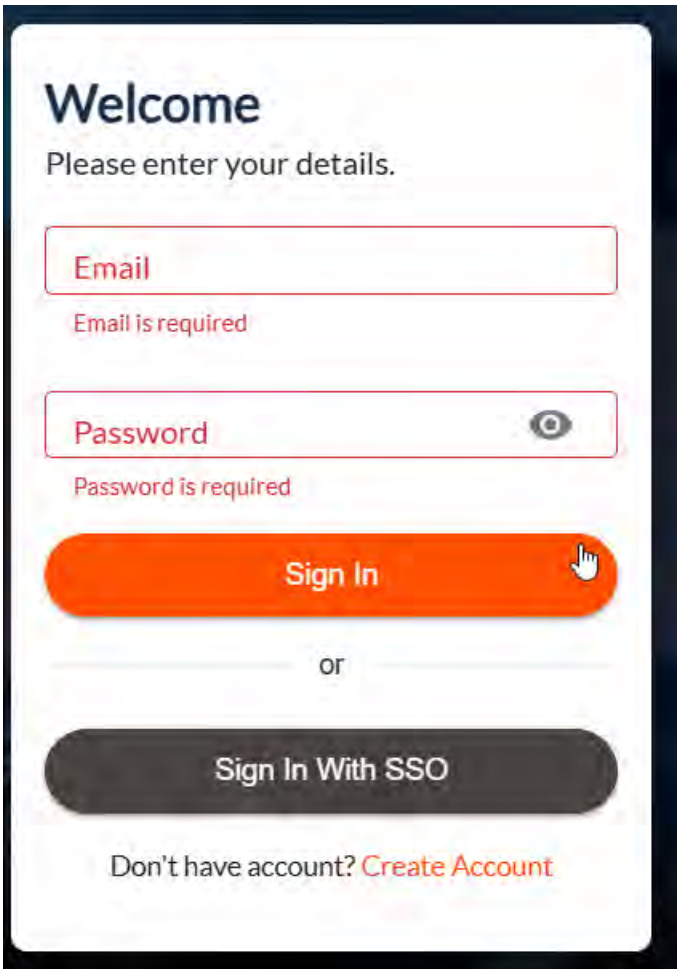
2. Enter Registration Information. The mandatory fields are marked with an asterisk (*) sign.
   a. **First Name**- Enter your first name.
   b. **Last Name**- Enter your last name.
   c. **Email**- Provide a valid email address.
   d. **Phone Number**- Enter a valid contact number for any communication.
   e. **Password**- Create a strong password for your account.
   f. **Confirm Password**- Re-enter the password to confirm.
3. Click the **Sign Up** button.
4. Once your email is verified, Click the **Login**.

5. Sign in using your new account credentials.

Existing Account Holders

To sign in, follow the given instructions:

1. Input your registered **Email** address and **Password**.
2. Click the **Sign in** button to complete the Login process.



North Star Operations

There are three types of operation we can perform using the North Star that are as follows:

Perform Assessment

The Assessment features are designed to provide a comprehensive understanding of your project's performance, enabling you to make informed decisions and optimize processes. By evaluating your Agile processes, you ensure that sprints and releases are efficient and effective.

Monitoring code quality and review processes helps identify areas for improvement, enhancing overall project health.

These features also offer insights into key metrics related to project health, quality issues, productivity, and DevOps performance, helping you maintain high standards and accelerate delivery. By analyzing code and build quality, you can address defects and technical debt, improving the robustness of your codebase. Understanding the speed of product releases and tracking team performance ensures that goals are met and changes are managed effectively.

Key Capabilities

Release Overview: Provides a detailed view of your release, including progress tracking and task management.
Agile Process Evaluation: Assesses the effectiveness of your Agile processes to ensure smooth sprints and releases.
Code and Review Monitoring: Tracks code quality and review processes to identify areas for improvement.
Project Health Metrics: Displays key metrics related to project health, quality issues, productivity, and DevOps performance.
Code and Build Quality Analysis: Highlights defects and technical debt to improve overall code health.
Release Speed and Team Tracking: Monitors the speed of product releases and tracks team performance to ensure goals are met.
Comprehensive Performance View: Covers release analysis, DevSecOps, test case metrics, and more for a complete understanding of project performance.

Generate North Star View

The NorthStar View feature is designed to enhance your project management by streamlining the creation and management of new release plans. It provides an interactive platform to describe requirements, generate detailed release plans, and ensure they align with your needs. By leveraging insights from previous releases and current market trends, NorthStar View helps you build on past successes and address current demands. Additionally, it generates a dashboard similar to the Assessment feature, analyzing your project's previous releases and creating new release plans based on source code and project management data. This comprehensive approach ensures your project remains efficient and aligned with industry standards.

Key Capabilities
Custom Release: Simplifies the ideation-to-release journey by generating use cases, tech stack recommendations, and detailed release plans. Allows for review and editing to ensure accuracy and relevance.
Improve Security Posture: Detects and upgrades vulnerable open-source libraries, providing recommendations and automated task execution to enhance security.

Release Details: Guides you through a step-by-step process to add new release plans, including specifying release details, timelines, resources, and intent.
Create Future Release Items: Analyzes previous releases and market trends to develop comprehensive future release plans. Includes defining release strategy, selecting technical stack, and generating use cases.
Build Project Use Cases: Automatically generates use cases based on selected criteria, allowing for review, editing, and detailed project plan creation.
Publish and Export: This enables exporting the release plan in PDF format and publishing changes to platforms like Jira or Aha for seamless integration and tracking.

Manage Backlog

The Manage Backlog feature is designed to streamline the process of managing project backlogs, making it easier to create, prioritize, and track tasks. This feature supports various product domains, including IT services and e-commerce, and integrates with popular management tools like JIRA and AHA.
Key Capabilities:
Backlog Creation and Management:
Users can create backlogs from customer reports and select from suggested backlogs, such as market trend expansions. The feature supports creating backlogs in JIRA, ensuring seamless integration with existing project management workflows.
Prioritization of Backlogs:
Users can prioritize existing backlogs, including features and epics, based on various criteria such as market trends, revenue impact, and customer impact. The priority of tasks can be modified (e.g., high, medium, low) and updated in real-time within JIRA.
Comprehensive Analysis and Reporting:
The feature provides a detailed analysis of files and folders, displaying the tree structure and code details. Users can download reports in PDF or Excel format, capturing all relevant content and trends in a graphical format.
Project Diagrams and Documentation:
Users can view and download project diagrams, including flowcharts and consolidated PDFs of all diagrams. The feature also supports viewing business cases and detailed project documentation.
Integration with GitHub:
The backlog management tool supports GitHub for version control, allowing users to manage private and public repositories with ease.

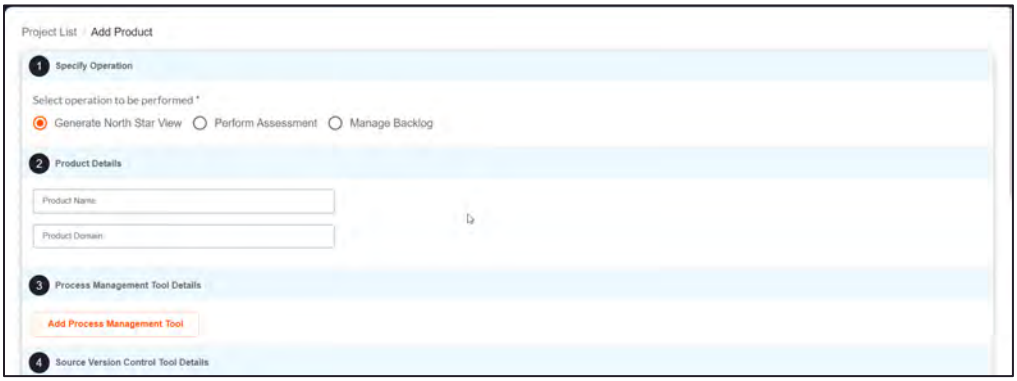
Add a New Project

There are three types of project operations you can perform using North Star that are as follows:

Add new North Star View Project

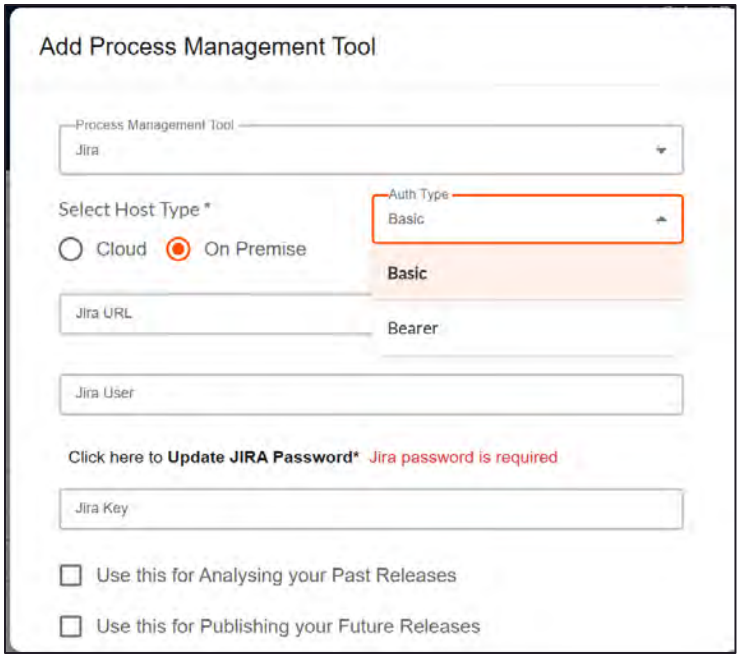
Follow the steps to Add a new project:

1. Click **Add Project**.
2. Select the **Generate North Star View** radio button.
3. Enter the product details:
   - Project Name.
   - Project Domain.



4. Click **Add Process Management Tool** to add your Jira or Aha process management tools details.
   a. **For Jira**:
   - Select **Jira** from the process management tools drop-down
   - Select Host Type from the available options:
     - **Cloud** – Select this option if your environment is deployed on a cloud infrastructure.
     - **On-Premises** – Select this option if your deployment is within the internal infrastructure or local network.

       Select authentication type (Applicable only for On-Premises)
       - Basic
       - Bearer



- Enter **Jira URL**
- Enter Jira **Username** details
- Enter Jira **key** token
- Check the box **Use this for Analyzing your Past Releases** if you want to use previous details for analysis.
- Check the box **Use this for Publishing your Future Releases** if you want to publish your future releases
- Click **Add** to store the Jira details.

b. **For Aha**:

- Select Aha from the process management tools drop-down
- Enter Aha **URL**
- Enter Aha **Key**
- Check the **Use this for Analyzing your Past Releases** if you want to use previous details for analysis
- Check the **Use this for Publishing your Future Releases** if you want to publish your future releases
- Click **Add** to store details.

5. Provide Source Version Control Tool details:
   a. GitHub
   - Select the **GitHub** source version control tool drop-down menu.

\ Enter the **GitHub Repository URL** that is associated with the project.

\ Add the GitHub token details by clicking **Update GitHub Token.**

b. Bitbucket

\ Select the **Bitbucket** source version control tool drop-down menu.

\ Specify the **Bitbucket Repository URL** that is associated with the project.

\ Specify the **Bitbucket Username.**

\ Add the Bitbucket token details by clicking **Update Bitbucket App Password.**

c. Gitlab

\ Select the **Gitlab** source version control tool drop-down menu.

\ Enter the **Gitlab Repository URL** that is associated with the project.

\ Add the GitHub token details by clicking **Update Gitlab Token.**

d. Azure

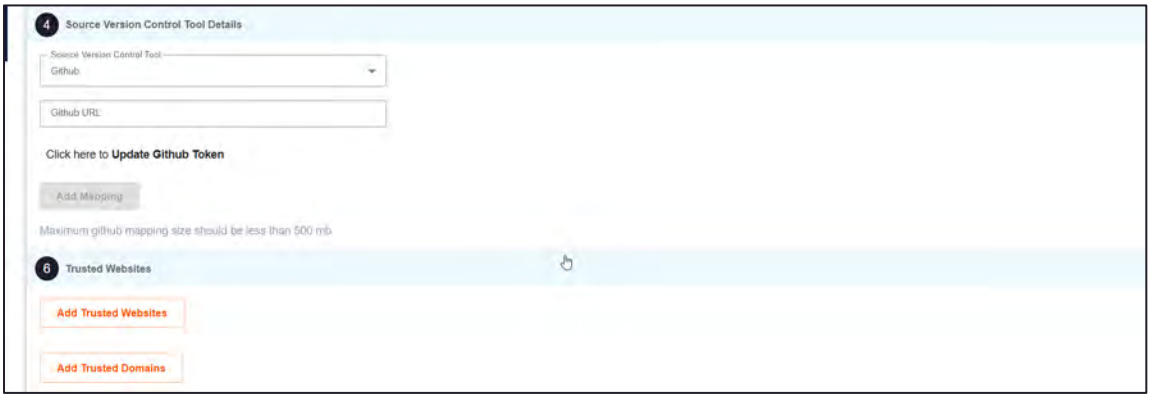
\ Select the **Azure** source version control tool drop-down menu.

\ Specify the **Azure** Repository URL that is associated with the project.

\ Specify the **Azure** Username.

Add the Azure token details by clicking **Update Azure Token** Password.

6. Provide ticketing system details:
   a. Service now: Enter the URL, Username, Release Field, and Password.
   b. Zendesk: Enter the URL, Email, and Token.
   c. Jira: Enter the URL, Email, Key, and Token.
7. Add trusted website details:
   a. Add Trusted Websites: Click **Add Trusted Websites**, enter the **URL,** and click **Add**.
   b. Add Trusted Domain: Click **Add Trusted Domain,** enter the **domain name**, and click **Add**.



8. Check the box to **Summarize** the information collected from the websites and domains.
9. Enter the number of websites selected for

10. Click **Save** to store the project details.

Note: For instructions on generating GitHub classic Personal Access Token (PAT) and Jira API Token, see the Reference section.

Add Performance Assessment Project

Follow the steps to Add a new project:

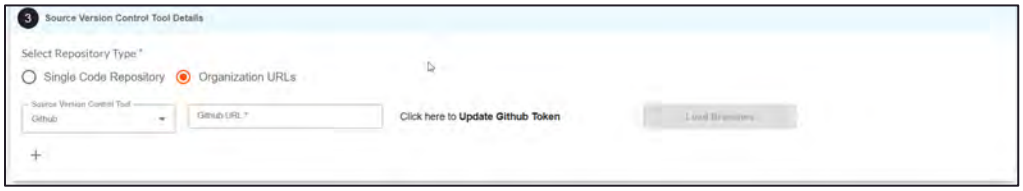
1. Click **Add Project**.
2. Select the **Perform Assessment** radio button.
3. Enter the product details:
   - Project Name.
   - Project Domain.
4. Provide **Source Version Control Tool** details.
   - Select Repo type from the following:
   a. **Single Code Repository-** To add a single repository.
   b. **Organization URLs**- To add multiple product repositories.
5. If you select **Single Code Repository** follow the steps below:
   a. GitHub
   - Select the **GitHub** source version control tool drop-down menu.
   - Enter the **GitHub Repository URL** that is associated with the project.
   - Add the GitHub token details by clicking **Update GitHub Token.**

   b. Bitbucket
   - Select the **Bitbucket** source version control tool drop-down menu.
   - Specify the **Bitbucket Repository URL** that is associated with the project.
   - Specify the **Bitbucket Username.**
   - Add the Bitbucket token details by clicking **Update Bitbucket App Password.**

   c. Gitlab
   - Select the **Gitlab** source version control tool drop-down menu.
   - Enter the **Gitlab Repository URL** that is associated with the project.
   - Add the GitHub token details by clicking **Update Gitlab Token.**

   d. Azure
   - Select the **Azure** source version control tool drop-down menu.
   - Specify the **Azure** Repository URL that is associated with the project.
   - Specify the **Azure** Username.
   - Add the Azure token details by clicking **Update Azure Token** Password.

6. If you select **Organization URLs** follow the steps below:
   e. Click the + button and add the repositories, as mentioned in **Step 5**.



7. Click **Load Branches** and select the appropriate branch for assessment.
8. Click **Save** to store the project details.

Note: For instructions on generating GitHub classic Personal Access Token (PAT) and Jira API Token, see Reference section.

Add Backlog View Project

Follow the steps to Add a new project:

1. Click **Add Project**.
2. Select the **Manage Backlog** radio button.
3. Enter the product details:
   \ Project Name
   \ Project Domain
   \ Product Portfolio
   \ Product Group
4. Click **Add Process Management Tool** to add your Jira or Aha process management tools details.
   a. **For Jira**:
   \ Select **Jira** from the process management tools drop-down
   \ Enter **Jira URL**
   \ Enter Jira **Username** details
   \ Enter Jira **key** token
   \ Check the box **Use this for Analyzing your Past Releases** if you want to use previous details for analysis.
   \ Check the box **Use this for Publishing your Future Releases** if you want to publish your future releases
   \ Click **Add** to store the Jira details.
   b. **For Aha**:
   \ Select Aha from the process management tools drop-down

- Enter Aha **URL**
- Enter Aha **Key**
- Check the **Use this for Analyzing your Past Releases** if you want to use previous details for analysis
- Check the **Use this for Publishing your Future Releases** if you want to publish your future releases
- Click **Add** to store details.

5. Provide Source Version Control Tool details:
   a. GitHub
   - Select the **GitHub** source version control tool drop-down menu.
   - Enter the **GitHub Repository URL** that is associated with the project.
   - Add the GitHub token details by clicking **Update GitHub Token.**

   b. Bitbucket
   - Select the **Bitbucket** source version control tool drop-down menu.
   - Specify the **Bitbucket Repository URL** that is associated with the project.
   - Specify the **Bitbucket Username.**
   - Add the Bitbucket token details by clicking **Update Bitbucket App Password.**

   c. Gitlab
   - Select the **Gitlab** source version control tool drop-down menu.
   - Enter the **Gitlab Repository URL** that is associated with the project.
   - Add the GitHub token details by clicking **Update Gitlab Token.**

   d. Azure
   - Select the **Azure** source version control tool drop-down menu.
   - Specify the **Azure** Repository URL that is associated with the project.
   - Specify the **Azure** Username.
   - Add the Azure token details by clicking **Update Azure Token** Password.

6. Add trusted website details:
   a. Add Trusted Websites: Click **Add Trusted Websites**, enter the **URL,** and click **Add**.
   b. Add Trusted Domain: Click **Add Trusted Domain,** enter the **domain name**, and click **Add**.
7. Check the box to **Summarize** the information collected from the websites and domains.
8. Enter the number of websites selected.

9. Click **Save** to store the project details.

Note: For instructions on generating GitHub classic Personal Access Token (PAT) and Jira API Token, see the Reference section.

View Product Feature Timeline

The Product Feature Timeline provides a comprehensive view of various project parameters through detailed reports and dashboards. This feature allows users to select the default project timeline and access the information, including:

Info

The Info Tab provides a comprehensive overview of the release, featuring tables for Epics, Stories, and Tasks. Each table includes columns for ID, Summary, Assignee, Reporter, Progress, Status, Labels, and Description, with an export option available. Additionally, the tab offers detailed code analysis, project diagrams, and in-depth file analysis, covering aspects like methods, logic, design patterns, and interactions. Contains the following tabs:

Summary

The Summary tab provides a detailed overview of the release. It includes a table with an export button, featuring the following columns: ID, Summary, Assignee, Reporter, Progress, Status, Labels, and Description. Separate tables are created for Epics, Stories, and Tasks to organize the information effectively.

Code Analysis

The Code Analysis tab shows detailed information about the files present in the code repository. It has project structure window and 2 tabs that as follows:

Details

Allows you to select project files from the project structure. This tab shows details such as File Name, Description, Detailed Summary, Importance, and References.

Code

Displays the actual code in the selected file.

Project Diagrams

The Project Diagram offers a visual representation of the project's structure, components, and relationships, simplifying the understanding of complex systems for developers. Here is the list of diagrams and their significance:

Diagram Name	Description
Class Diagram	A diagram that shows the classes and their relationships in your project, helping you understand the object-oriented design
Sequence Diagram	A diagram that shows the sequence of events and interactions between objects in your project, helping you understand the flow of your application
Logical Diagram	A diagram that shows the logical structure of your project, including the relationships between components and the data flow.
Activity Diagram	A diagram that shows the activities and tasks involved in your project, helping you understand the workflow and business processes.
Component Diagram	A diagram that shows the components and their relationships in your project, helping you understand the architecture and design.
ER (Entity-Relationship) Diagram	A diagram that shows the entities and their relationships in your project, helping you understand the data modeling and database design.
Use Case Diagram	A diagram that shows the use cases and their relationships in your project, helping you understand the functional requirements and user interactions.
Deployment Diagram	A diagram that shows the deployment of your project, including the hardware and software components, and their relationships.
Composite Structure Diagram	A diagram that shows the composite structure of your project, including the components and their relationships.
Object Diagram	A diagram that shows the objects and their relationships in your project, helping you understand the object-oriented design and implementation.
User Journey Map	A diagram that shows the user's journey and interactions with your project, helping you understand the user experience and usability.
State Diagram	A diagram that shows the states and transitions of your project, helping you understand the behavior and functionality.
Architecture Diagram	A diagram that shows the overall architecture of your project, including the components, relationships, and interactions.
Workflow Diagram	A diagram that shows the workflow and business processes involved in your project, helping you understand the operations and management.
Communication Diagram	A diagram that shows the communication and interactions between components and objects in your project, helping you understand the data flow and integration.

Detailed Analysis

Provides comprehensive details about each file, including:

- **Methods or Functions**: Lists the methods or functions defined in the file.

- **Logic and Functionality**: Describes the logic and functionality implemented in the file.
- **Summary**: Offers a brief overview of the file's purpose and content.
- **File Modified Date**: Indicates the last date the file was modified.
- **Design Patterns**: Identifies any design patterns used in the file.
- **File Created Date**: Shows the date the file was initially created.
- **File Extension**: Specifies the file type based on its extension.
- **File Author**: Names the author who created or last modified the file.
- **File Name**: Provides the name of the file.
- **Internal and External Interactions**: Details the interactions within the file and with external components.
- **Framework or Technology Used**: Lists the frameworks or technologies utilized in the file.

Features

The Features tab provides a comprehensive list of features developed for the selected release. It offers an overview of new enhancements and functionalities introduced, detailing their impact on the product. This tab helps stakeholders understand the scope of improvements and the value added in each release, ensuring transparency and alignment with project goals.

Developers Insights

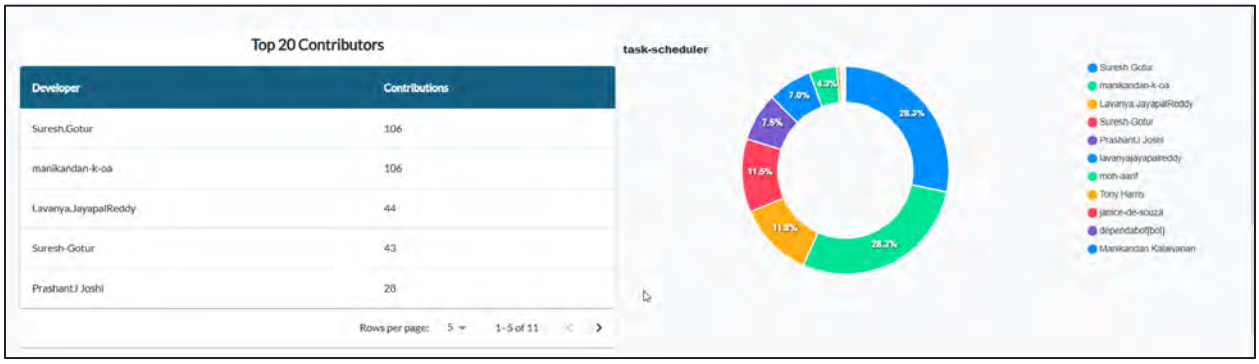
This tab contains information about the developers who worked on the project and their contributions. The tab is a combination of various cards, charts, and tables.

- **Cards**:
  - **Commits from** card shows the date from which commits are stated in the release
  - **Total Commits** card shows the number of total commits in the release
  - **Lines of Code(LOC)** card shows line of code written by developers in the release.
  - **Current Active Authors**: Count of active authors in the release.
  - **Code Languages**: Count of languages used in the release.

- The distribution chart illustrates the contributions of developers using squares that vary in size based on the number of commits made by each developer.

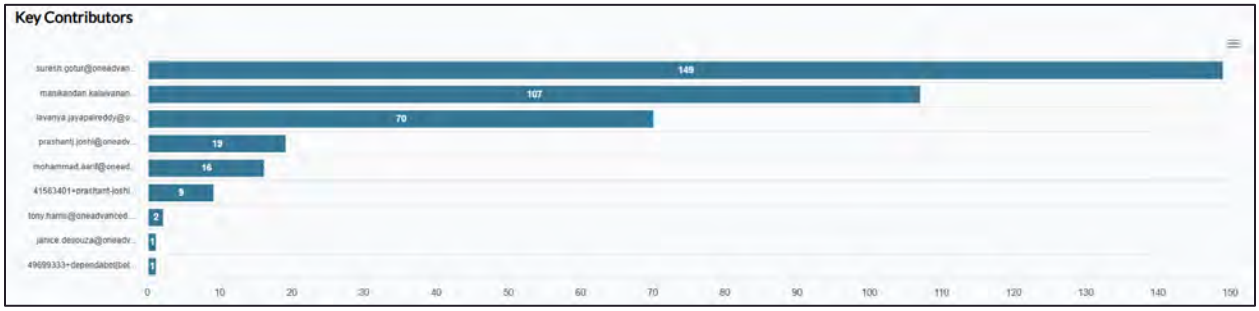
Top 20 Contributors Table and Pie Chart

Table with the list of the top 20 developers who contributed to the release. The table has column name developer and contributions made by each developer.
Pie chart shows the percentage contributions made by developers in visual form.



Key Contributors (bar chart with commits or file count)

The bar chart shows the list of top contributors participated in release. Chart shows name of the developers on Y axis and number of commits on X axis.



Developer productivity Table

The Developer Productivity Table is a comprehensive tool designed to track and analyze the contributions of developers within a project. This table includes several key columns that provide detailed insights into each developer's activity and productivity. The columns are as follows:
Author Name: The name of the developer who made the commit.
Email: The email address of the developer. This is useful for communication and for maintaining a record of the contributors.

Commit Date: The date when the commit was made. This helps in tracking the timeline of changes and understanding the development pace.
Files Modified: The number of files that were modified in the commit. This gives an idea of the scope of changes made by the developer.
Lines Added: The number of lines of code that were added in the commit. This metric helps in understanding the extent of new code contributions.
Lines Deleted: The number of lines of code that were removed in the commit. This is useful for tracking code refactoring and cleanup activities.
Lines Modified: The number of lines of code that were modified (both added and deleted). This provides a comprehensive view of the changes made in the codebase.

By analyzing this table, project managers and team leads can gain valuable insights into the productivity and contributions of each developer. It helps in identifying high-performing team members, understanding the impact of their work, and making informed decisions to improve overall project efficiency.

Author Name	Author Email	Commit Date	Files Modified	Lines Added	Lines Deleted	Lines Modified
lavanyajayapalreddy	lavanya.jayapalreddy@oneadvanced.com	2024-12-20	1. Quartz Scheduler ...	10727	3019	13746
Manikandan Kalaiva...	manikandan.kalaivanan@oneadvanced.com	2024-12-24	Merge pull request #...	9793	3769	13562
Suresh.Gotur	suresh.gotur@oneadvanced.com	2024-12-24	Merge pull request #...	10406	1525	11931
Prashant J Joshi	prashant.j.joshi@oneadvanced.com	2025-01-03	Updating push.regist...	1146	580	1726
janice-de-souza	janice.desouza@oneadvanced.com	2025-03-12	Cypress test script a...	909	0	909

1-5 of 9

Code Quality

The Code Quality section provides a detailed analysis of a release, offering insights into various metrics that help assess code health and project structure. It presents key statistics in both tabular and visual formats.

\ **Cards**:

- **Commits from** card shows the date from which commits are stated in the release
- **Total Commits** card shows the number of total commits in the release
- **Lines of Code** (**LOC**) card shows line of code written by developers in the release.
- **Current Active Authors**: Count of active authors in the release.
- **Code Languages**: Count of languages used in the release.

\ **Technology and Project Metrics**:

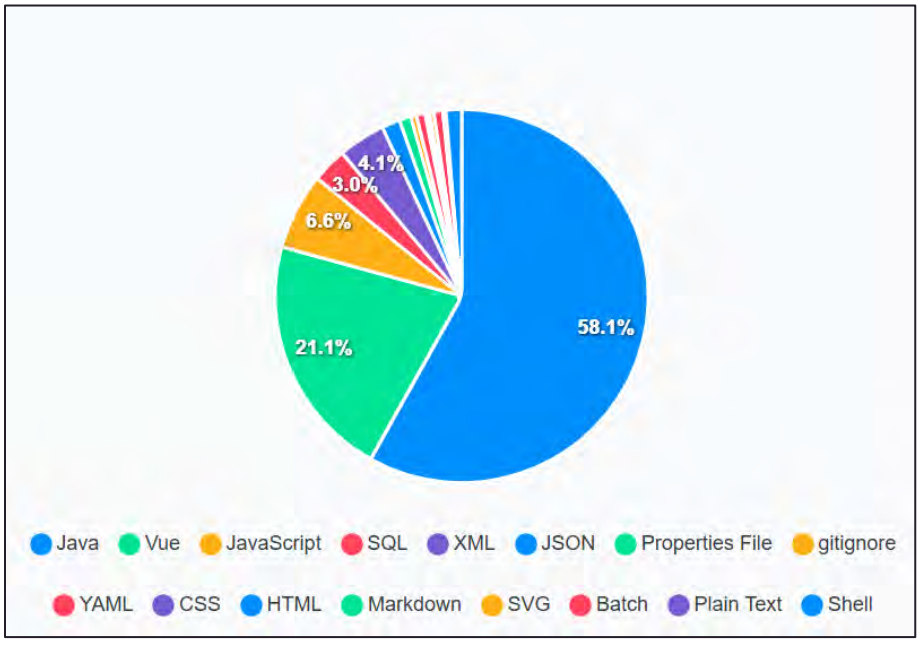
Technology	Number of Files	Lines of Code	Projects Count
Java	106	11241	4392
Vue	22	4089	2235
JavaScript	17	1283	764
SQL	9	587	351
XML	5	792	500

This table provides an overview of the technologies and components used in the release:

- **Number of Technologies Used**: Count of different technologies integrated into the project.
- **Number of Files**: Total files in the project.
- **Lines of Code (LOC)**: Aggregate LOC across all project files.
- **Project Count**: Number of projects included in the release.

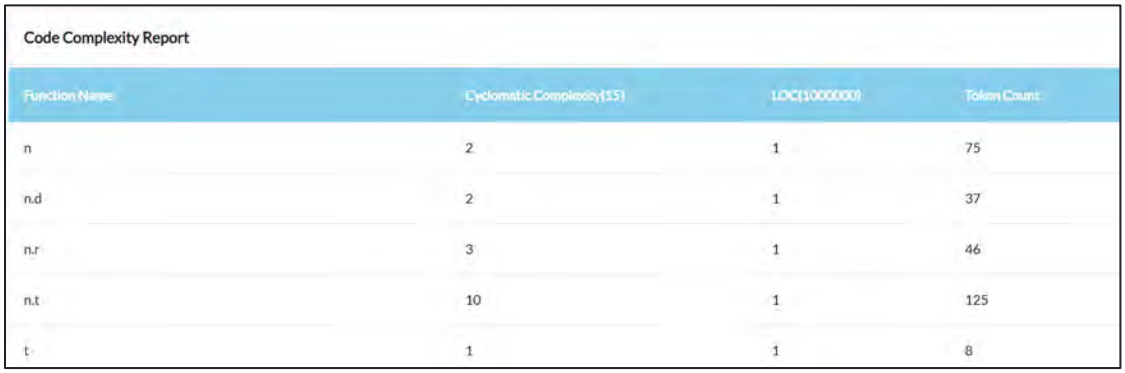
\ **Visual Representation**:

The Code Quality section provides insights into the programming languages used in the project through a pie chart representation. The chart visually displays the percentage distribution of various technologies, highlighting their contribution to the code base.



- **Code Complexity Report**

  Code Complexity Report presents detailed insights into code complexity in tabular form. It helps in identifying complex and high-risk areas in the codebase. Provides insights for refactoring and optimization. It helps in understanding overall project structure and growth over time.



Code Complexity Report

Function Name	Cyclomatic Complexity(15)	LOC(1000000)	Token Count
n	2	1	75
n.d	2	1	37
n.r	3	1	46
n.t	10	1	125
t	1	1	8

Security Assessment

The Security Assessment tab offers a thorough analysis of your project's code repository, focusing on identifying and displaying potential security issues. It meticulously identifies all components used in the project, including open source, third party, private, and unknown libraries. By scanning these components for vulnerabilities, it categorizes them by severity-critical, high, medium, low, and unknown. The results are presented through various metrics such as risk level, severity, total components, and vulnerabilities, complemented by visual representations like graphs and tables. These visuals help in understanding the issues by severity, code quality, and security posture. Additionally, the feature offers actionable insights to improve the project's overall security posture, providing guidelines for interpreting the data and reducing risk levels. This comprehensive approach ensures that any potential vulnerabilities are promptly identified and addressed, maintaining the security and integrity of your project.
The security assessment tab is divided into two sections:

- Dashboard: The security assessment report is displayed in Cards, Graphs, and a table.
- Graph: This Information is presented in an intuitive graphical presentation.

Note: Experimental Feature - This feature is in active development and may not work as expected.

Dashboard

The security assessment data is presented in Cards, Graphs, and Table format.
Cards
There are 11 graphs and they are as follows:
Risk Level

The Risk Level card provides a quick snapshot of the overall risk associated with the project, rated on a scale from 1 to 5. This score is calculated based on the identified vulnerabilities and their severity, offering a concise measure of the project's security risk. A higher score indicates greater potential for security issues, helping teams prioritize their remediation efforts effectively. This card is essential for quickly assessing the project's security posture and understanding the urgency of addressing the identified risks.

Severity

The Severity card provides an overview of the project's security severity level. It indicates the potential impact of identified vulnerabilities on the project. This metric helps in understanding the seriousness of the security issues present and guides the prioritization of remediation efforts. The severity level is categorized into different levels such as Critical, High, Medium, and Low.

Component Assessed

This card displays the total number of components that have been scanned for vulnerabilities. These components are categorized into four sections: Open Source, Third Party, Private, and Unknown. The percentage contribution of each category is also shown on the card, providing a clear breakdown of the types of components used in the project.

Vulnerabilities

The Vulnerabilities card displays the total number of vulnerabilities identified in the project, along with their distribution by severity. The vulnerabilities are categorized into Critical, High, Medium, Low, and Unknown, with each category showing the percentage of total vulnerabilities. This card provides a clear overview of the security issues present in the project, helping to prioritize which vulnerabilities need immediate attention based on their severity. This detailed breakdown helps in understanding the overall security posture and the specific areas that require focus.

Total Components

The Total Components card provides a summary of the number of components used in the project. It categorizes these components into four sections: Open Source, Third Party, Private, and Unknown. Each category shows the count of components, helping to understand the composition of the project's dependencies. This breakdown is essential for identifying the types of libraries and dependencies used in the project, which can influence the overall security and maintenance efforts.

Total Vulnerabilities

The Total Vulnerabilities card provides a comprehensive count of all vulnerabilities identified in the project. It also breaks down these vulnerabilities by their severity levels, offering a clear view of the security landscape. The vulnerabilities are categorized into Critical, High, Medium, Low, and Unknown, with each category showing the count of vulnerabilities.

Code Quality

The Code Quality card provides an assessment of the overall quality of the project's code. It is divided into several categories: Critical, High, Medium, and Low. Code quality is evaluated by running a comprehensive set of checks on the source code of each component. These checks include both direct dependencies (libraries explicitly used in the project) and transitive dependencies (libraries used by the direct dependencies). The results help identify areas where

the code may need improvement to enhance maintainability, performance, and security. This card is crucial for understanding the robustness of the codebase and guiding efforts to improve code quality.

Security Posture

The Security Posture card provides an overview of the project's security health by categorizing identified security issues based on their severity. It is divided into four categories: Critical, High, Medium, and Low. This helps in understanding the potential impact of these issues. This metric is calculated by evaluating the security vulnerabilities found in the project's components, including both direct and transitive dependencies.

Components

The Components card provides detailed information about the project's components and metadata. This card includes the following details:

Components: The total number of components (libraries and dependencies) used in the project.
Project Name: The name of the project being assessed.
Project Version: The current version of the project.
Project Created On: The date and time when the project was created.
Project Generation Method: Information about how the project was generated.
Source Code Repository: The repository where the project's source code is stored.
Branch / Tag: The specific branch or tag of the source code being assessed.
Project Created by: The name of the person who created the project.

Vulnerabilities (CVSS)

This card provides a detailed breakdown of the vulnerabilities identified in the project, categorized by their Common Vulnerability Scoring System (CVSS) scores. This helps in understanding the severity and potential impact of each vulnerability.

Components by Age

The Components by Age card provides an overview of the age distribution of the components used in the project. This information helps in understanding the lifecycle and potential obsolescence of the components, which can impact the project's security and maintainability. The components are categorized by their age, with each category showing the corresponding count. This card is essential for tracking the age of the project's components, helping to ensure that outdated and potentially vulnerable components are identified and managed appropriately.

Unique Components by Classification

The Unique Components by Classification card provides a detailed breakdown of the project's components based on their classification and dependency type. This table helps in understanding the composition and origin of the components used in the project. The classifications include Open Source, Third Party, Private, and Unknown, with each category showing the count of direct and transitive dependencies.

This card is essential for providing a clear and concise summary of the types of components and their dependencies, helping to understand the project's dependency structure.

Graphs

Issue Count By Vulnerability Severity

The Issue Count By Vulnerability Severity graph provides a visual representation of the number of vulnerabilities identified in the project, categorized by their severity. The graph features the following elements:

Y-Axis: Represents the count of vulnerabilities.
X-Axis: Represents the classification of components, including Open Source, Third Party, and Private.
The graph has four categories: Critical, High, Medium, and Low. You can click on each category to isolate the view. This allows to focus on specific severity levels and better understand the distribution of vulnerabilities within each component classification. Hovering over the color in the graph will give the count of the individual category.

Code Quality By Classification

The Code Quality By Classification graph provides a visual representation of the results from code quality checks, categorized by the type of components. The graph features the following elements:
Y-Axis: Represents the count of code quality checks.
X-Axis: Represents the classification of components, including Open Source, Third Party, and Private.
The graph has four categories: Critical, High, Medium, and Low. You can click on each category to isolate the view. Hovering over individual colors in the graph provides a tooltip with the count of code quality issues for that specific category and component classification.

Security Posture By Classification

The Security Posture By Classification graph provides a visual representation of the security posture checks, categorized by the type of components. The graph features the following elements:
Y-Axis: Represents the count of security posture checks.
X-Axis: Represents the classification of components, including Open Source, Third Party, and Private.
The graph has four categories: Critical, High, Medium, and Low. You can click on each category to isolate the view. Hovering over individual colors in the graph provides a tooltip with the count of code quality issues for that specific category and component classification.

Table

Top Vulnerabilities

The table provides detailed information about the most critical vulnerabilities identified in the project. The table includes the following columns:
CVE Details: Comprehensive information about each vulnerability.
Package Id: The identifier of the package where the vulnerability was found.
Package Version: The version of the package affected by the vulnerability.
CVE ID: The unique identifier assigned to vulnerability.
CVE Severity: The severity level of vulnerability.

CVE Score: The score assigned to vulnerability based on the Common Vulnerability Scoring System (CVSS).
You can sort the data or select specific columns to display by clicking the column button, providing a customizable view of the information. You can export the data in PDF format, making it easy to share and document. The PDF will also include a detailed analysis of your project.

Graph

The Graph in the Security Assessment section provides an interactive and visual representation of the project's security data. This tab enhances the user experience by making complex data more accessible and easier to understand through graphical elements. The bubble graph format makes it easy to visualize the relationships and hierarchies within the data.
In the Interactive Bubble Graph:
The Center Bubble: The central bubble represents the Repository. Clicking on this bubble reveals other key categories such as Code Quality, Components, Aging Components, and Security Posture.
Expanding Bubbles: Each category expands into sub-categories when clicked. For example:
Code Quality: Expands to show Open Source, Third Party, Private, and Unknown.
Components: Expands to show Open Source, Third Party, Private, and Unknown.
Aging Components: Shows categorization by age with different bubbles.
Security Posture: Expands to show Open Source, Third Party, Private, and Unknown.
Detailed View: Clicking on any sub-category bubble displays a list of associated data, providing detailed insights into each category.
In the Repository Details Table:
Located on the left side of the Graph Tab, this vertical table provides essential information about the repository:
Project Name: The name of the project being assessed.
Project Version: The current version of the project.
Input Details: N/A
Input Type: N/A
Creator: The name of the person who created the project.
Created Date: The date and time when the project was created.
Risk Level: risk

Process Assessment

Agile Assessment Report

Overall Assessment

Overall Process Effectiveness indicator

This metric provides a single aggregated value representing the average effectiveness of various defined processes or checkpoints.

Agile Process efficiency split

These metrics represent the defined process checkpoints across different performance levels: High, Medium, and Low - based on evaluation criteria. This is presented in the form of a Pie chart.

Efficiency split by Dimension

These metrics represent the defined process checkpoints across different performance levels: High, Medium, and Low - based on evaluation criteria across different dimensions. This is the bar chart which has dimensions on the Y axis and Efficiency Percentage on the X axis. The Dimensions are as follows Sprint Management, Release Management, and Backlog Management.

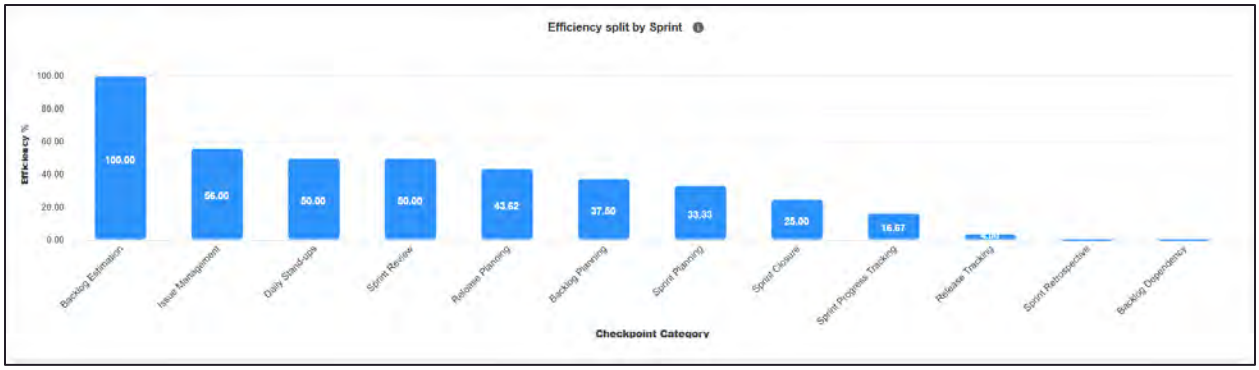
Checkpoint Category Table

The table shows a detailed explanation of the checkpoint categorization and their calculations. The categories are as follows:

Checkpoint Category	Description
Sprint Planning	Defining what can be delivered in the upcoming sprint and how that work will be achieved.
Daily Stand-ups	Short, daily meetings to discuss progress and obstacles.
Sprint Progress Tracking	Monitoring the progress of the sprint to ensure it stays on track.
Sprint Review	A meeting at the end of the sprint to review what was accomplished.
Sprint Retrospective	A meeting to reflect on the sprint and identify improvements.
Sprint Closure	Finalizing the sprint and preparing for the next one.
Release Planning	Planning for the release of the product increment.
Release Tracking	Monitoring the progress of the release.
Backlog Planning	Organizing and prioritizing the product backlog.
Backlog Dependency	Identifying dependencies in the backlog.
Backlog Estimation	Estimating the effort required for backlog items.
Issue Management	Handling issues that arise during the sprint.

Efficiency split by Sprint

This graph represents the defined efficiency percentage of various checkpoints categories.



Sprint Process

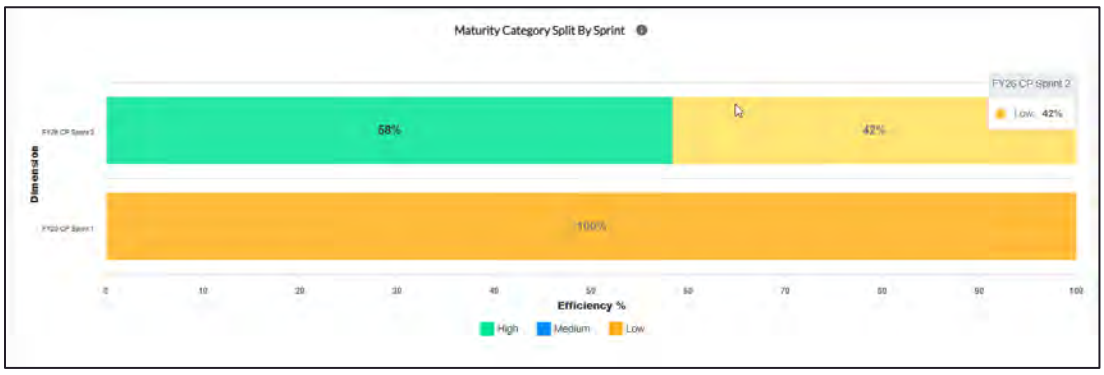
The tab shows the overall sprint metrics of your project. This tab has three subtabs that are as follows:

Overview

This tab has two graphs:

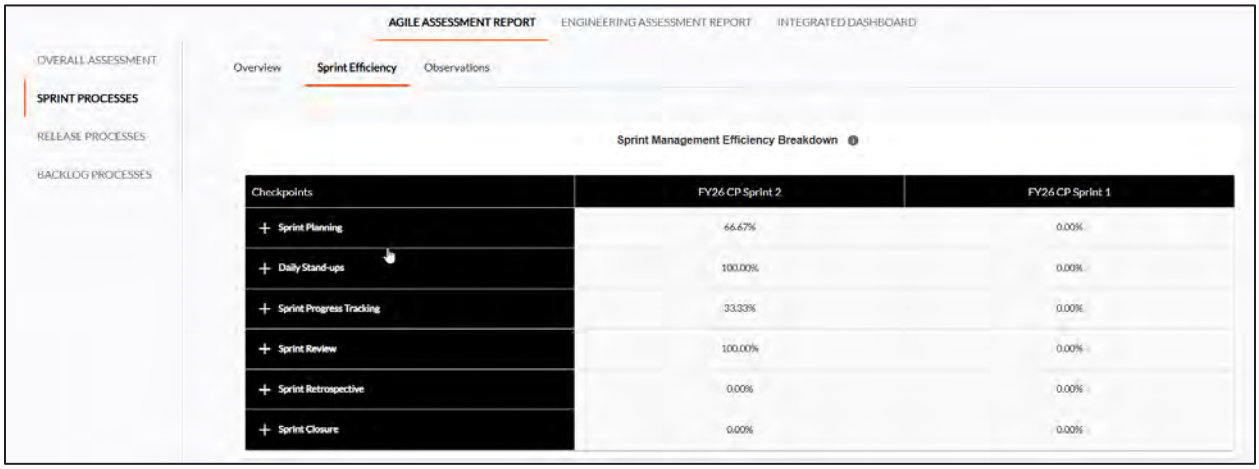
- Efficiency split by sprint: This bar chart represents the efficiency percentage across major sprints.

- Maturity Category Sprint by Sprint - This graph represents the defined process checkpoints across different performance levels: High, Medium, and Low - based on evaluation criteria across different dimensions.



Sprint efficiency

The Sprint Management Efficiency Breakdown table shows the detailed breakdown of checkpoints and their maturity level across various entities, organized by checkpoint categories.



Observations

This table represents observations across various checkpoint categories. Each category includes details such as the checkpoint description, value, observations, and recommendations to address any issues. Assists you to find the issues in the current process.

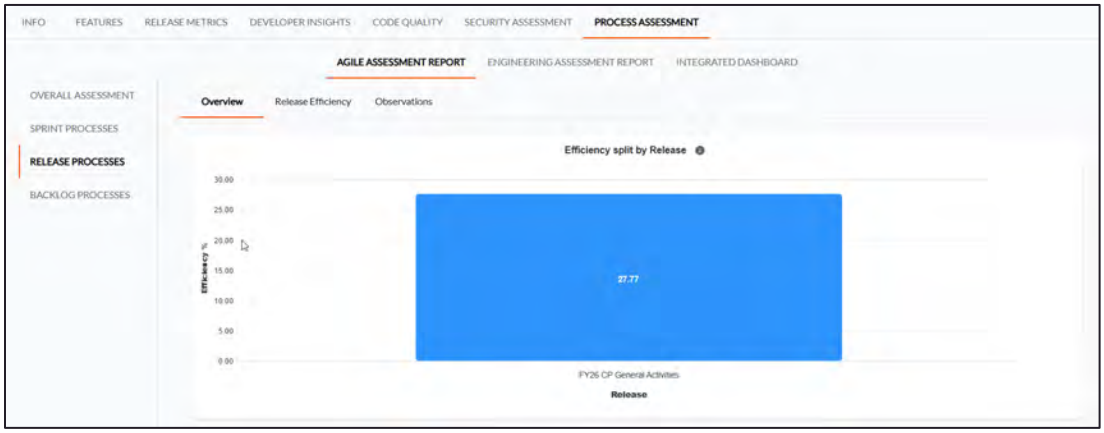
Release Process

The tab shows the overall release metrics of your project. This tab has the following subtabs:

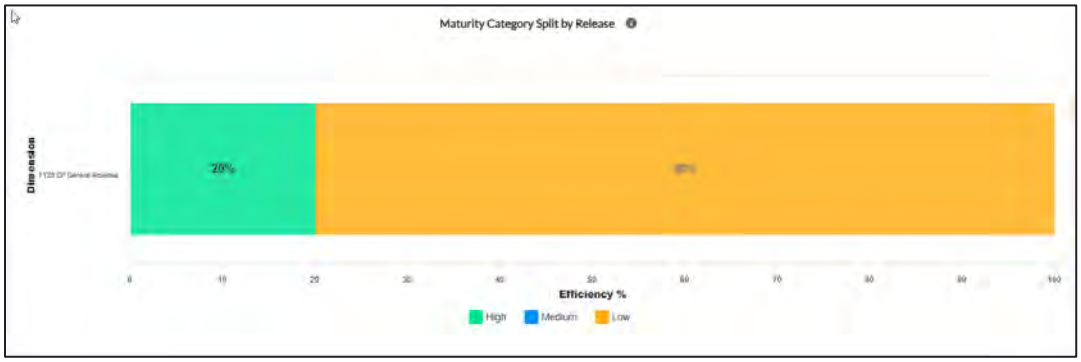
Overview

This tab has two graphs:

- Efficiency split by Release: This bar chart represents the efficiency percentage across major releases.

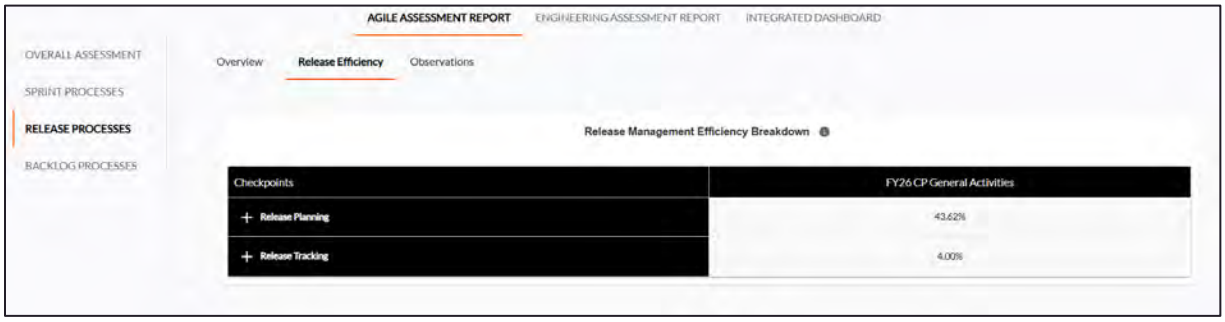


\ Maturity Category Sprint by Release - This graph represents the defined process checkpoints across different performance levels: High, Medium, and Low - based on evaluation criteria across different dimensions.



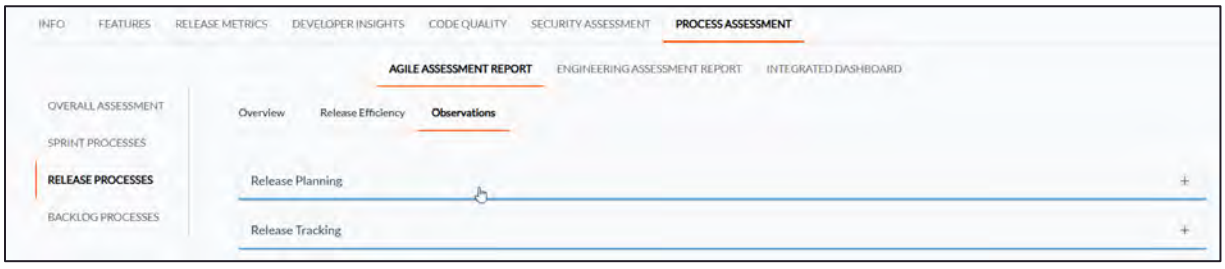
Release efficiency

The Release Management Efficiency Breakdown table shows the detailed breakdown of checkpoints and their maturity level across various entities, organized by checkpoint categories.



Observations

This table represents observations across various checkpoint categories in the release. Each category includes details such as the checkpoint description, value, observations, and recommendations to address any issues. Assists you to find the issues in the current process.



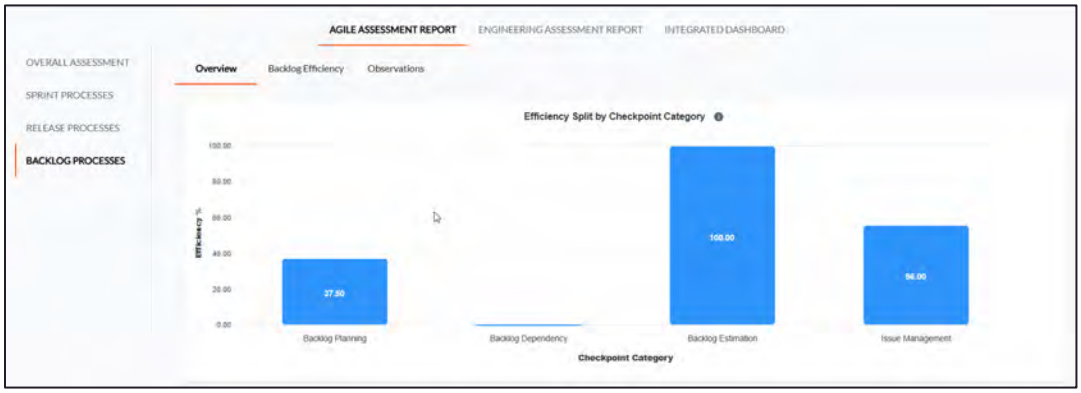
Backlog process

The tab shows the overall backlog metrics of your project. This tab has the following subtabs:

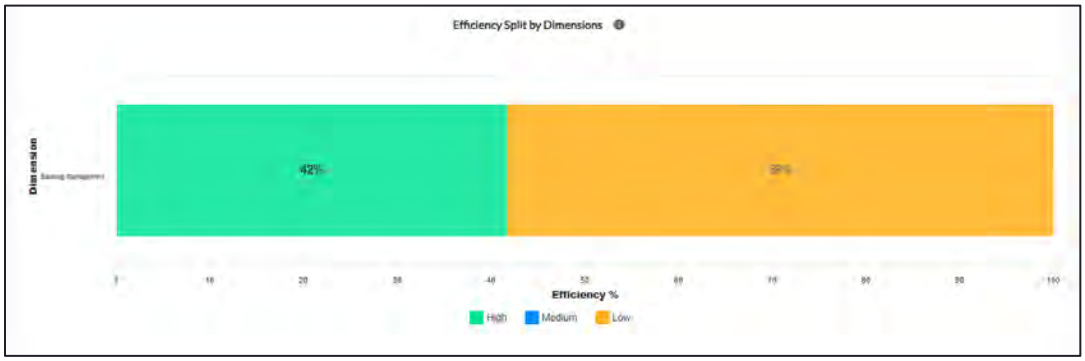
Overview

This tab has two graphs:

- Efficiency split by Checkpoint Categories: This bar chart represents the efficiency percentage across the major release.

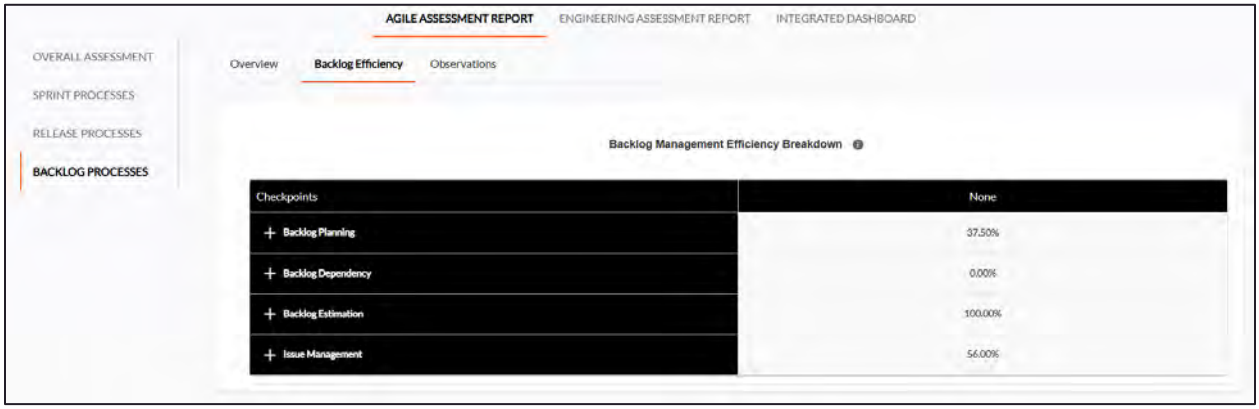


- Efficiency Split by Dimensions- This graph represents the defined process checkpoints across different categories: High, Medium, and Low - based on evaluation criteria across different dimensions.



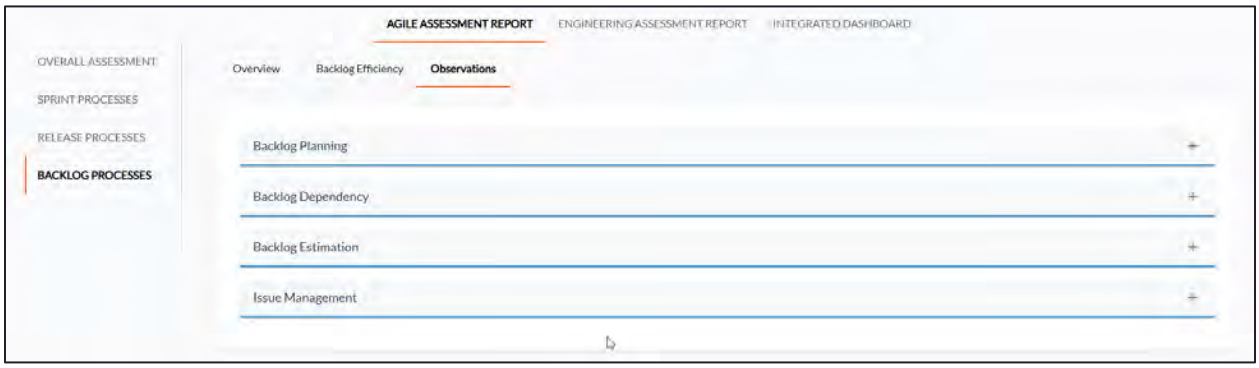
Backlog efficiency

The Backlog Management Efficiency Breakdown table shows the detailed breakdown of checkpoints and their maturity level across various entities, organized by checkpoint categories.



Observations

This table represents observations across various checkpoint categories for the backlog. Each category includes details such as the checkpoint description, value, observations, and recommendations to address any issues. Assists you to find the issues in the current process.



Engineering Assessment Report

Code Repository

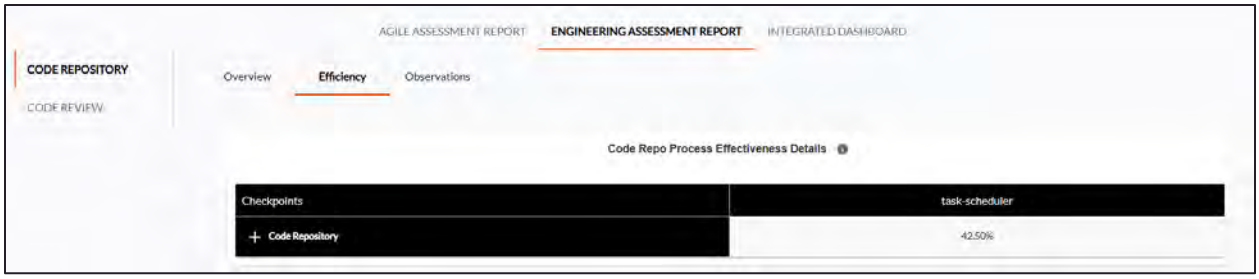
Overview

This tab has two graphs:

- **Code Repo Process Effectiveness**: This Metrics provides a single aggregated value representing the Average effectiveness of various defined processes/Checkpoints.
- **Code Repo Process Effectiveness Split**: This Metrics represents the defined process checkpoints across different performance levels: High, Medium, and Low - based on evaluation criteria.

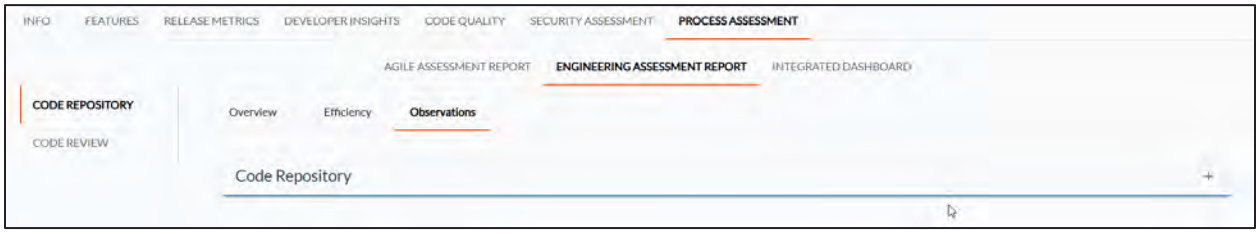
Efficiency

The Code Repo Process Effectiveness Details table shows the detailed breakdown of checkpoints and their maturity level across various entities, organized by checkpoint categories.



Observation

This table represents observations across various checkpoint categories for the code repository. Each category includes details such as the checkpoint description, Gen AI response, observations, and recommendations to address the issues. Assists you to find the issues in the current process.

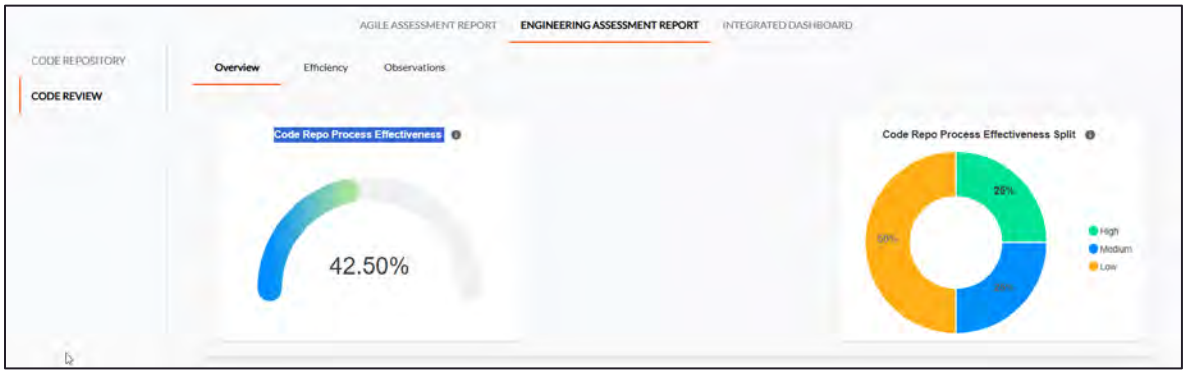


Code Review

Overview

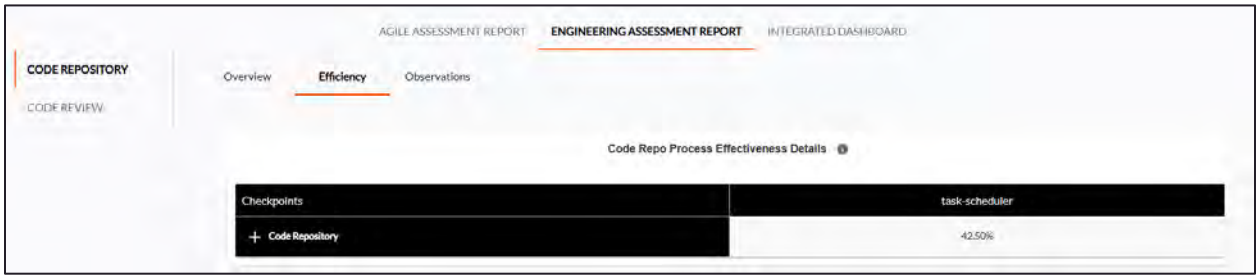
This tab has two graphs:

- **Code Repo Process Effectiveness**: This Metrics provides a single aggregated value representing the Average effectiveness of various defined processes/Checkpoints.
- **Code Repo Process Effectiveness Split**: This Metrics represents the defined process checkpoints across different performance levels: High, Medium, and Low - based on evaluation criteria.



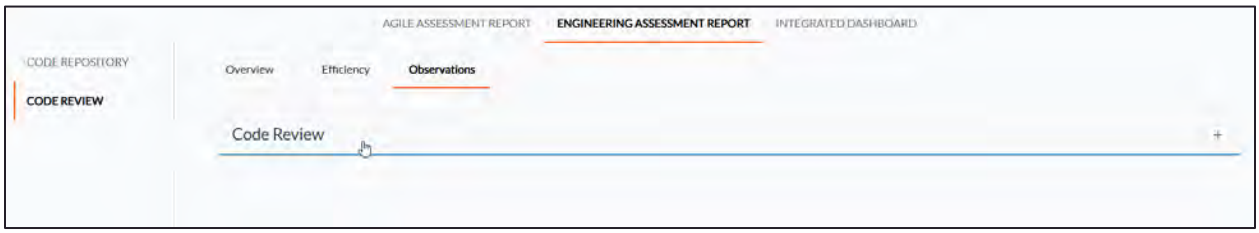
Efficiency

The Code Review Process Effectiveness Details table shows the detailed breakdown of checkpoints and their maturity level across various entities, organized by checkpoint categories.



Observation

This table represents observations across various checkpoint categories for the code review. Each category includes details such as the checkpoint description, Gen AI response, observations, and recommendations to address the issues. Assists you to find the issues in the current process.



Integrate Dashboard

Project Health Tab: This tab contains two sub-tabs:

Sprint Summary

Provides detailed metrics for the current and previous sprints across five tables:

Quality Defect: The Quality Defect table provides a comprehensive overview of various defect metrics across different stages of the development and testing process. It includes information on production defects, UAT defects, and defects identified during code and design reviews. The table also tracks defect removal efficiency, defect leakage at various stages, and defect density. This helps in understanding the overall quality and stability of the product, highlighting areas that need improvement to reduce defect rates and enhance product reliability. The table has the following rows:

Parameter	Description.
Production Defects	Defects found in the production environment
UAT Defects	Defects identified during User Acceptance Testing.

Total Dev and Test Defects	Combined count of defects found during development and testing phases
Code Review Defects	Defects identified during code reviews
Design Review Defects	Defects found during design reviews
Defect Removal Efficiency	Percentage of defects removed before reaching production
Defect Leakage to UAT/Customer Testing	Defects that escaped to UAT or customer testing.
Defect Leakage to Production (P1 and P2 Defects)	High-priority defects that leaked into production
Defect Leakage	Overall defect leakage across all phases.
Defect Density (Scrum-Dev)	Number of defects per unit of code in the development phase.
Defect Density (Scrum-Dev & Test)	Number of defects per unit of code in both development and testing phases
CUT DIR	Code Unit Test Defect Injection Rate.
Release DIR	Release Defect Injection Rate.

Productivity: The Productivity table measures the efficiency and output of the development team. It includes metrics such as story points completed per person per day, the done index, scope variance, effort variance, and velocity report. These metrics provide insights into the team's productivity, helping to identify trends, set realistic goals, and improve planning and execution. By analyzing these metrics, teams can optimize their workflows and enhance overall productivity. The table has the following rows:

Parameter	Description
Productivity (Story Points per Person Day)	Average story points completed per person per day
Done Index	The ratio of completed work to planned work
Scope Variance Bullet Chart	Visual representation of scope changes
Effort Variance	Difference between planned and actual effort
Velocity Report	The average amount of work completed per sprint.

Requirements: The Requirements table monitors the planning and delivery of epics and user stories. It tracks the number of epics and user stories planned versus those delivered, as well as items added after the sprint starts. This table helps in assessing the team's ability to meet planned objectives and manage scope changes. It provides a clear picture of how well the team is adhering to the planned requirements and highlights any deviations that need to be addressed. The table has the following rows:

Parameter	Description
Epics Planned	Number of epics planned for the sprint
Epics Delivered	Number of epics completed in the sprint
User Stories Planned	Number of user stories planned for the sprint

User Stories Delivered	Number of user stories completed in the sprint
Items Added Post Sprint Start	Number of items added after the sprint began.

DORA Metrics: The DORA Metrics table tracks key DevOps Research and Assessment (DORA) metrics, which are critical for evaluating the performance of the software delivery process. It includes lead time for changes, deployment frequency, change failure rate, and mean time to restore (MTTR). These metrics help in understanding the efficiency and reliability of the deployment pipeline, identifying bottlenecks, and improving the overall DevOps practices to achieve faster and more stable releases. The table has the following rows:

Parameter	Description
Lead Time for Changes	Time taken from code commit to production deployment
Deployment Frequency	How often deployments occur
Change Failure Rate	Percentage of changes that result in a failure
Mean Time to Restore (MTTR)	Average time to restore service after a failure.

Testing: The Testing table evaluates the effectiveness and coverage of the testing process. It includes metrics such as the total number of test cases executed, both manual and automated, the percentage of automation, testing cycle time, and test coverage percentage. Additionally, it tracks regression and functional test cases, as well as the automation percentages for these categories. This table provides insights into the thoroughness of the testing process, helping to ensure that the product is thoroughly tested and meets quality standards. The table has the following rows:

Parameter	Description
TCs Executed-Total	Total number of test cases executed
TCs Executed – Manual	Number of manual test cases executed
TCs Executed – Automated	Number of automated test cases executed
Automation %	Percentage of test cases that are automated
Testing Cycle Time	Time taken to complete the testing cycle
Test Coverage %	Percentage of code covered by tests
Testing Cycle Time	Time taken to complete the testing cycle
Test Coverage %	Percentage of code covered by tests
TCs Total	Total number of test cases
TCs - Manual	Number of manual test cases
TCs – Automated	Number of automated test cases
TCs – Regression	Number of regression test cases
TCs – Functional	Number of functional test cases
Regression Automation %	Percentage of regression test cases that are automated
Functional Automation %	Percentage of functional test cases that are automated
Unit Test Automation %	Percentage of unit tests that are automated.

Release Summary

Provides detailed metrics for the current and previous sprints across five tables:
Quality Defect: The Quality Defect table provides a comprehensive overview of various defect metrics across different stages of the development and testing process. It includes information on production defects, UAT defects, and defects identified during code and design reviews. The table also tracks defect removal efficiency, defect leakage at various stages, and defect density. This helps in understanding the overall quality and stability of the product, highlighting areas that need improvement to reduce defect rates and enhance product reliability. The table has following rows:

Parameter	Description.
Production Defects	Defects found in the production environment
UAT Defects	Defects identified during User Acceptance Testing.
Total Dev and Test Defects	Combined count of defects found during development and testing phases
Code Review Defects	Defects identified during code reviews
Design Review Defects	Defects found during design reviews
Defect Removal Efficiency	Percentage of defects removed before reaching production
Defect Leakage to UAT/Customer Testing	Defects that escaped to UAT or customer testing.
Defect Leakage to Production (P1 and P2 Defects)	High-priority defects that leaked into production
Defect Leakage	Overall defect leakage across all phases.
Defect Density (Scrum-Dev)	Number of defects per unit of code in the development phase.
Defect Density (Scrum-Dev & Test)	Number of defects per unit of code in both development and testing phases
CUT DIR	Code Unit Test Defect Injection Rate.
Release DIR	Release Defect Injection Rate.

Code Health: The Code Health table provides key metrics to assess the quality and activity of the codebase. It includes various parameters that help in understanding the state of code reviews, contributions, and changes made to the code. This table is essential for maintaining high code quality and ensuring efficient development practices.

Row Name	Description
Unreviewed PRs Merged	Number of pull requests (PRs) that were merged without a review.
Average Commits	Average number of commits made per PR.
PR Velocity	Average time taken to merge a PR from the time it was opened.
PRs Reviewed	Number of PRs that have been reviewed.

Row Name	Description
LOC Added	Lines of Code (LOC) added to the codebase.
LOC Deleted	Lines of Code (LOC) deleted from the codebase.
LOC Modified	Total Lines of Code (LOC) modified (added + deleted).
Active Contributors	Number of contributors actively making commits.
Average PR Size	Average size of PRs in terms of lines of code changed.

Collaboration Metrics: The Collaboration Metrics table provides key metrics to assess the efficiency and effectiveness of collaboration within the development team. It includes various parameters that help in understanding the time spent on coding, reviewing, and other development activities. This table is essential for identifying bottlenecks and improving the overall workflow.

Row Name	Description
Coding Time	Total time spent on writing code.
Pickup Time	Time taken for a task to be picked up after it is assigned.
Review Time	Average time taken to review code.
Delay in Code Review (min)	Delay in starting the code review process, measured in minutes.
Submitter Response Times	Time taken by the submitter to respond to review comments.
Total Time Spent on Dev Activities	Total time spent on all development activities, including coding, reviewing, and other tasks.

DORA Metrics: The DORA Metrics table tracks key DevOps Research and Assessment (DORA) metrics, which are critical for evaluating the performance of the software delivery process. These metrics help in understanding the efficiency and reliability of the deployment pipeline, identifying bottlenecks, and improving overall DevOps practices to achieve faster and more stable releases.

Row Name	Description
Lead Time for Changes	The time taken from code commits to production deployment. This metric helps in understanding the speed of the development process.
Deployment Frequency	The frequency at which deployments occur. This metric indicates how often new code is released to production.
Change Failure Rate	The percentage of changes that fail. This metric helps in assessing the stability and reliability of the deployment process.

Row Name	Description
Mean Time to Restore (MTTR)	The average time taken to restore service after a failure. This metric is crucial for understanding the team's ability to quickly recover from incidents.

Productivity: The Productivity table measures the efficiency and output of the development team across different releases. It includes various metrics that provide insights into the team's performance, helping to identify trends, set realistic goals, and improve planning and execution.

Row Name	Description
Scope Variance Bullet Chart	Visual representation of scope changes between releases. This metric helps in understanding how much the scope has varied from the initial plan.
Effort Variance	Difference between planned and actual effort. This metric indicates how well the team is estimating and managing their workload.
Lead Time (Release wise)	The time is taken from the start of development to the release of the product. This metric helps in understanding the speed of the development process for each release.
Cycle Time Movement (Release wise)	The time taken to complete a task from start to finish within a release. This metric helps in identifying bottlenecks and improving the efficiency of the development process.

Quality Tab

The Quality Tab is divided into three subtabs: Code Quality, Build Quality, and Defect Analysis. The graphs include a term tool in the right-hand corner to indicate the source of the information

Code Quality: This subtab combines cards and graphs to provide a detailed overview of code quality metrics.

\ Cards:

- Code Coverage: Displays the percentage of code covered by tests, indicating how much of the codebase is tested.

- Release DIR: Shows the Defect Injection Rate for the release, measuring the number of defects introduced during the release.
- CUT DIR: Displays the Code Unit Test Defect Injection Rate, indicating the number of defects found during unit testing.
- Code Smells: Indicates the number of code smells detected, which are indicators of potential issues in the code that may need refactoring.
- Code Duplicacy: Shows the percentage of code duplication, highlighting areas where code is repeated and may need consolidation.

\ Graphs:

- CUT DIR by Sprints: X Axis: Sprints, Y Axis: CUT DIR Count. This graph tracks the number of defects found during unit testing across different sprints.
- Release DIR: X Axis: Releases, Y Axis: Release DIR Count. This graph shows the number of defects introduced during each release.
- Project Code Coverage: X Axis: Sprints, Y Axis: Project Code Coverage %. This graph displays the percentage of code covered by tests over different sprints.
- Code Complexity By Project: X Axis: Sprints, Y Axis: Code Complexity Count. This graph tracks the complexity of the codebase over different sprints.
- Technical Debt By Project: X Axis: Sprints, Y Axis: Technical Debt in Minutes. This graph shows the amount of technical debt accumulated over different sprints, measured in minutes.
- Function Count By Project: X Axis: Sprints, Y Axis: Function Count. This graph tracks the number of functions in the codebase over different sprints.
- Code Duplication By Project: X Axis: Sprints, Y Axis: Code Duplication %. This graph shows the percentage of code duplication over different sprints.
- Vulnerabilities By Project: X Axis: Sprints, Y Axis: Vulnerability Count. This graph tracks the number of vulnerabilities found in the codebase over different sprints.
- Bugs By Project: X Axis: Sprints, Y Axis: Bugs Count. This graph shows the number of bugs found in the codebase over different sprints.
- Code Smell By Project: X Axis: Sprints, Y Axis: Code Smell Count. This graph tracks the number of code smells detected in the codebase over different sprints.

Build Quality: This subtab combines graphs and status cards to provide insights into build quality.

\ Graphs:

- Unreviewed PRs Merged: X Axis: Releases, Y Axis: Unreviewed PRs Merged Count. This graph shows the number of pull requests merged without review across different releases.

- Code Review Acceptance Rate: X Axis: Releases, Y Axis: Code Review Acceptance Rate. This graph tracks the rate at which code reviews are accepted across different releases.

\ Status Cards:

\ Job Statistics By Status: Displays the percentage of total successful builds, indicating the overall health of the build process.

Defect Analysis: This subtab combines cards and graphs to analyze defects.

\ Cards:

- Open Defect: Number of open defects, indicating unresolved issues.
- Defect Count: Total number of defects, providing an overview of the defect load.
- Critical & High Fixed: Number of critical and high-priority defects fixed, showing the team's focus on resolving severe issues.
- Defect Leakage: Number of defects that leaked into production, highlighting areas where quality control needs improvement.
- Defect Density: Number of defects per unit of code, indicating the overall quality of the codebase.

\ Graphs:

- Bug and Defects Count by Sprints: Tracks the number of bugs and defects per sprint, providing insights into the defect trends over time.
- Critical Defect Fix by Sprints: Tracks the number of critical defects fixed per sprint, showing the team's responsiveness to severe issues.
- Defect Accumulation by Sprint: Shows the accumulation of defects over sprints, indicating how defects are being managed over time.
- SIT Defect Leakage by Sprints: Tracks defects that leaked into System Integration Testing (SIT), highlighting areas where integration testing needs improvement.
- Defect Leakage to Production (P1 and P2 Defects) by Sprint: Tracks high-priority defects that leaked into production per sprint, indicating critical quality issues.
- Defect Density (Scrum-Dev & Test) by Sprint: Shows defect density for development and testing phases per sprint, providing insights into the overall defect rate.
- Defect Density (Scrum-Dev) by Sprint: Shows defect density for the development phase per sprint, indicating the quality of the development process.
- Defect Removal Efficiency by Sprint: Tracks the efficiency of defect removal per sprint, showing how effectively defects are being addressed.
- Bug and Defects Count by Releases: Tracks the number of bugs and defects per release, providing insights into the defect trends over different releases.
- Critical Defect Fix by Releases: Tracks the number of critical defects fixed per release, showing the team's responsiveness to severe issues.

- Defect Leakage by Releases: Shows the number of defects that leaked into production per release, highlighting areas where quality control needs improvement.
- Defect Accumulation by Releases: Shows the accumulation of defects over releases, indicating how defects are being managed over time.
- Escape Ratio by Release: Tracks the ratio of defects that escaped into production per release, indicating critical quality issues.
- SIT Defect Leakage by Releases: Tracks defects that leaked into System Integration Testing (SIT) per release, highlighting areas where integration testing needs improvement.
- Escape Defect Density (Scrum-Dev & Test): Shows defect density for development and testing phases for escaped defects, providing insights into the overall defect rate.
- Defect Leakage to Production (P1 and P2 Defects) by Releases: Tracks high-priority defects that leaked into production per release, indicating critical quality issues.
- Defect Density (Scrum-Dev & Test) by Release: Shows defect density for development and testing phases per release, providing insights into the overall defect rate.
- Defect Density (Scrum-Dev) by Release: Shows defect density for the development phase per release, indicating the quality of the development process.
- Defect Removal Efficiency by Release: Tracks the efficiency of defect removal per release, showing how effectively defects are being addressed.
- Percentage Age of Bugs by Release: Shows the age distribution of bugs per release, indicating how long bugs have been present.
- Percentage Age of Bugs by Sprint: Shows the age distribution of bugs per sprint, indicating how long bugs have been present.

\ Tables:

- Defect by Status Table: Displays the status of defects in a table format, providing a clear view of defect resolution progress.
- Defect by Severity Table: Shows the number of defects categorized by severity, including Major, High, High (Migrated), Critical, Normal, Blocker, P2, P3, P4, Minor, Medium, and P1 defects, helping to prioritize defect resolution efforts.

Time To Market Tab

The **Time To Market** Tab provides insights into the efficiency of the development and release processes. It combines cards and graphs to present key metrics related to lead time and cycle time, helping teams understand and improve their time to market.

\ Cards:

- Lead Time (Release wise): Displays the average time taken from the start of development to the release of the product for each release. This metric helps in understanding the speed of the development process in days.
- Cycle Time (Release wise): Shows the average time taken to complete a task from start to finish within each release. This metric helps in identifying bottlenecks and improving the efficiency of the development process in days

\ Graphs:

- Lead Time by Release: A graphical representation of lead time across different releases. The X-axis represents the releases, and the Y-axis represents the lead time in days. This graph helps in visualizing trends and identifying areas for improvement.
- Cycle Time by Release: A graphical representation of cycle time across different releases. The X-axis represents the releases, and the Y-axis represents the cycle time in days . This graph helps in understanding the efficiency of the development process over time.

Productivity Tab

The Productivity Tab provides insights into the efficiency and output of the development team. It combines cards and graphs to present key metrics related to productivity, velocity, and throughput. The graphs include a term tool in the right-hand corner to indicate the source of the information.

\ Cards:

- Planned Vs Completed (SP): Displays the comparison between planned story points (SP) and completed story points for each sprint. This helps in understanding how well the team is meeting its planned objectives.
- Productivity (SP per person day) by Sprint: Shows the average story points completed per person per day for each sprint. This metric helps in assessing individual productivity.
- Velocity by Sprint: Displays the velocity of the team for each sprint, indicating the amount of work completed. This helps in understanding the team's capacity and performance over time.
- Throughput: Shows the number of tasks or stories completed in a given time period, providing insights into the team's efficiency.

\ Graphs:

- Productivity (Story Points per Person per Day): A graphical representation of productivity, showing the average story points completed per person per day. This helps in visualizing trends and identifying areas for improvement.

- Velocity Report By Sprints: A graph that tracks the team's velocity across different sprints. The X-axis represents the sprints, and the Y-axis represents the velocity. This helps in understanding the team's performance over time.
- Total Story Points Planned and Total Story Points Completed: A graph with blue and green lines representing the planned and completed story points, respectively. This helps in comparing planned work with actual outcomes.
- Sprint Remaining Burndown and Sprint Planned Burndown: These graphs allow you to choose the sprint from a drop-down menu. The X-axis represents the date, and the Y-axis represents the story points. These graphs help in tracking the progress of the sprint and identifying any deviations from the plan.
- Epic Burndown: A graph that tracks the progress of epics over time, showing how much work remains to be done. This helps in understanding the progress of larger features or initiatives.

Predictability Tab

The Predictability Tab provides insights into the team's ability to meet planned objectives and manage scope changes. It combines cards and graphs to present key metrics related to predictability, helping teams understand and improve their planning and execution processes.

\ Cards:

- Done Index: Shows the percentage of work completed compared to what was planned. This metric helps in understanding how well the team is meeting its planned objectives.
- Scope Variance: Displays the percentage of variance between the planned scope and the actual scope. This metric helps in identifying how much the scope has changed from the initial plan.
- Sprint Injection Rate: Shows the percentage of new work added to the sprint after it has started. This metric helps in understanding the impact of scope changes during the sprint.

\ Graphs:

- Done Index: A graph that tracks the Done Index across different sprints. The X-axis represents the sprints, and the Y-axis represents the Done Index percentage. This helps in visualizing the team's ability to complete planned work over time.
- Scope Variance Bullet Chart By Sprints: A graph that shows the scope variance percentage across different sprints. The X-axis represents the sprints, and the Y-axis represents the scope variance percentage. This helps in understanding how the scope has varied from the initial plan for each sprint.

- Scope Variance Bullet Chart By Release: A graph that shows the scope variance percentage across different releases. The X-axis represents the releases, and the Y-axis represents the scope variance percentage. This helps in understanding how the scope has varied from the initial plan for each release.
- Sprint Injection Bullet Chart: A graph that tracks the Sprint Injection Rate across different sprints. The X-axis represents the sprints, and the Y-axis represents the Sprint Injection Rate percentage. This helps in visualizing the impact of new work added during the sprint.

Project Related Metrics tab

This tab provides a comprehensive view of various project-related metrics through several sub-tabs: Release Analysis, DevSecOps Analysis, Test Case Analysis Metrics, DORA Metrics, and Space Metrics.

Release Analysis

This sub-tab contains cards and graphs that provide insights into the release process.

\ Cards:

- Work Added / Work Completed: Displays the amount of work added versus the work completed for the release.
- Average Velocity: Shows the average velocity, calculated as Total Story Points Completed divided by the Number of Sprints.
- Unestimated Work: Indicates the number of unestimated work items linked to the release.
- Confidence Level: Displays the confidence level in completing the remaining work within the remaining time.

\ Graphs:

- Release Report: Provides an overview of the release metrics.
- Avg. Velocity: Graph showing the average velocity across releases.
- Work Added, Completed, and Remaining: A graph with three colors representing work added (blue), work completed (green), and work remaining (yellow). The X-axis represents the metrics value, and the Y-axis represents the releases.
- Work Completed: Shows the story points completed in the release.
- Work Added: Number of story points planned for the release.
- Work Remaining: Calculated as the number of story points planned for the release minus the number of story points completed in the release.
- Sprint Remaining: Displays the number of sprints remaining to complete the release.

- Forecast By Releases: Calculated as Total Remaining Work divided by Average Velocity per Sprint. Total Remaining Work is the sum of the story points, hours, or tasks yet to be completed for the release. Average Velocity per Sprint is the average amount of work the team completes in each sprint.
- Confidence Level by Releases: Shows the velocity needed to complete the remaining work within the remaining time.
- Unestimated Work Items: Displays the number of unestimated stories linked to the release.
- Completed Story Points: Displays the number of story points completed in the release.

DevSecOps Analysis

This sub-tab contains graphs that provide insights into the DevSecOps process.

\ Graphs:

- Comment Count per PR By Release: Total author comments plus total reviewer comments per release.
- PR Reviewed by Release: Count of closed pull requests per sprint.
- Average Review Time: Calculated as PR Close Time minus PR Create Time.

Test Case Analysis Metrics

This sub-tab contains cards and graphs that provide insights into test case execution.

\ Cards:

- Failed: Number of failed test cases.
- Passed: Number of passed test cases.
- Executed: Total number of test cases executed.

\ Graphs:

- Test Case Analysis Graphs: Various graphs showing detailed metrics related to test case execution.

DORA Metrics

This sub-tab contains cards and graphs that track key DevOps Research and Assessment (DORA) metrics.

\ Cards:

- Deployment Failure Rate: Percentage of deployments that fail.

- Deployment Frequency: Frequency of deployments.
- Lead Time for Changes: Time taken from code commit to production deployment.

\ Graphs:

- Lead Time for Changes: Sum of days between the date of completion and the date of creation.

Space Metrics: This sub-tab contains cards and graphs that provide insights into various performance metrics.

\ Cards:

- Velocity: Percentage of work completed versus planned.
- Coding Time: Total coding time in days.
- Epics SP Shipped: Number of story points shipped for epics.
- Average Commits: Average number of commits.

\ Graphs:

- Performance, Activity, Collaboration, and Efficiency: Graphs with a drop-down menu to select different metrics.

Adding New Release

This section details the process for creating and managing a new release plan for upcoming features and improvements.

Custom Release

The Custom Release theme showcases how SASVA can simplify and streamline the ideation-to-release journey. In an interactive mode, describe your requirements in a few lines and generate a detailed release plan for your teams to execute.

- Generate use cases relevant to the identified theme.
- Allow customers to review and edit the auto-generated use cases.
- Provide tech stack recommendations for the implementation of the custom theme.
- Generate an optimized and detailed release plan including epics, user stories, and tasks for the identified theme.
- Allow customers to review and edit the release plan as needed to ensure it meets the requirements.
- Show tasks assigned to developers, for work that may need manual effort.

Improve Security Posture

The Improve Security Posture theme detects and upgrades vulnerable versions of open-source libraries in your project. Remove third-party software dependencies and suggest the most compatible upgrade path for your codebase

- Scans third-party library inventory
- Finds security vulnerabilities
- Provides recommendations to improve software posture
- Shows a detailed breakdown of tasks assigned to SASVA Bot for automated execution
- Enables assisted error-fixing with SASVA AI models

Release Details

Follow the instructions to add a New Release Plan:

1. Click the **Add New Release Plan** button on the Product Feature Timeline page.
2. The Release Details page will appear.
3. Select the **Release Theme** from the given options.
4. Specify the Release details:
    - Release Name
    - Version
    - Description
5. Specify Release Timelines:

- Tentative Start Date
- Tentative End Date

6. Specify Resources Details:
    - The Number of resources worked on the latest release.
    - Estimated no of resources required for the next release.
7. Click the **Next** button.
8. Select Release Intent from the provided options:
    - Plan New Feature
    - Plan Software or Product Release
    - Plan One-Time Work

    a. Select Deliverable type
        - One-Time Application
        - One-Time Product
        - One-Time Activity
9. Select from the **existing releases** to create a new release plan.
10. Select the required **GitHub Tag/ Branches**.
11. On the **Backlog Grooming** page, select pending backlog items from Jira for the release plan.
12. The **Refine Market Trends** page displays **Analyst View**, suggests **Innovative or New Features**, and conducts **Gap analysis**.

    You can review and select insights based on your preferences.
13. (Optional)Click the **Add New Requirement** button and select the requirement type:
    - **Individual Requirement**- To enter a single-line requirement.
    - **Bulk Requirement**- To add multiple requirements at once by entering them in separate lines.

    Once entered, click the **Add** button.
14. Check the boxes to **select items to consider** for the release plan and proceed.
15. **Review the inputs** thoroughly to generate future release plan.
16. A summary view of all your selections will be displayed on the **Review Input to Generate Future Release Plan** page.

Create Future Release Items

To develop the Future Release for NorthStar, previous releases are analyzed to understand what worked well and what areas need improvement. The backlog items are carefully reviewed to ensure that all pending tasks and features are considered. Additionally, current market trends are considered to align the product with the latest industry standards and customer expectations. This comprehensive approach helps create a release that not only builds on past successes but also addresses current demands and anticipates future needs. Follow the steps to create future releases:

Build Project Use Cases

1. Select Release Intent from the provided options:
    - Plan New Feature
    - Plan Software or Product Release
    - Plan One-Time Work

    \ Select Deliverable type

        o One-Time Application
        o One-Time Product
        o One-Time Activity
2. Define the release plan strategy by performing the following steps:
    a. Adjust the sliders to define team dynamics according to the proficiency levels of the developers involved in the release. Click **Next**.
    b. Select the Technical Stack for your project or check the box to get the SASVA recommended technical stack. Click **Next.**
    c. Select the desired Virtual Agents (maximum 7) and click **Next**.
    d. Choose the language style to create the stories and tasks in the preferred format and click **Next**. The formats are as follows:

    \ **Action-Oriented**: Focuses on the specific actions that need to be taken to achieve an objective. This style emphasizes the tasks and the verbs that describe the work that needs to be done. For Example: Integrate and configure an AI-based recommendation system to provide personalized suggestions for users.

    \ **Outcome-Focused**: Emphasizes the end goal or result of the action. This style focuses on the intended impact of completing the task, highlighting the value or benefit derived from it. For Example: Enhance the user experience by providing personalized content recommendations through AI-based integration.

    \ **Stakeholder-Centric**: Centers on the needs and perspectives of stakeholders, highlighting the benefits for specific groups or individuals. It emphasizes how the task aligns with stakeholder interests. For Example: As a product manager, I want to

integrate an AI-based recommendation system to deliver personalized user experiences, increasing user engagement.

- **Compliance-Driven**: Centers around meeting regulatory, security, or organizational requirements. This style emphasizes alignment with standards and legal or compliance needs. For Example: Implement AI-based recommendation systems to enhance personalization while adhering to privacy regulations, such as GDPR, and ensuring data security.
- **Functional Requirement**: Describes the required functionality or behavior of the system. This style focuses on what the system should do or how it should perform. For Example- The system shall integrate an AI-based recommendation engine to provide real-time, personalized suggestions based on user preferences and behavior.

e. Choose the approach for the release creation and click **Continue** to fetch questions that help to generate use cases automatically. Check appropriate options as per requirement.

3. The **Use Case Questionnaire** section opens. Select appropriate answers to the questions.
4. Click the **Skip Question** button, if you prefer not to answer.
5. Click **View Use Cases** when the button is enabled. A list of use cases is displayed.
6. You can also add new use cases by clicking the **Add New Use Cases** button.
7. To edit the details of a particular use case, click the pencil icon and update it.
8. Select the use cases and click the **Generate Details** button.
9. A new tab named **Final Use Cases** appears. This tab displays the revised list of use cases along with the generated details.
10. Click **Generate Project Plan** to generate a detailed project plan.
11. Click the **Proceed** button.
12. You can export the plan in PDF format.
13. Click the **Confirm Release** button to generate the final release plan.
14. Click the **Next** button, and a window will appear to confirm future releases.
15. Click the **Publish** button and the release appears in the list of releases.
16. Click **Publish to Jira/Aha**. A message will pop up asking you to confirm whether you want to publish the changes for this release.
17. Click the **Confirm** button to publish the changes.

Assessment View

For the assessment, North Star generates comprehensive dashboards that scan your entire code repository and produce various insights. To get more information about the dashboard, please refer to the Product Feature Timeline section.

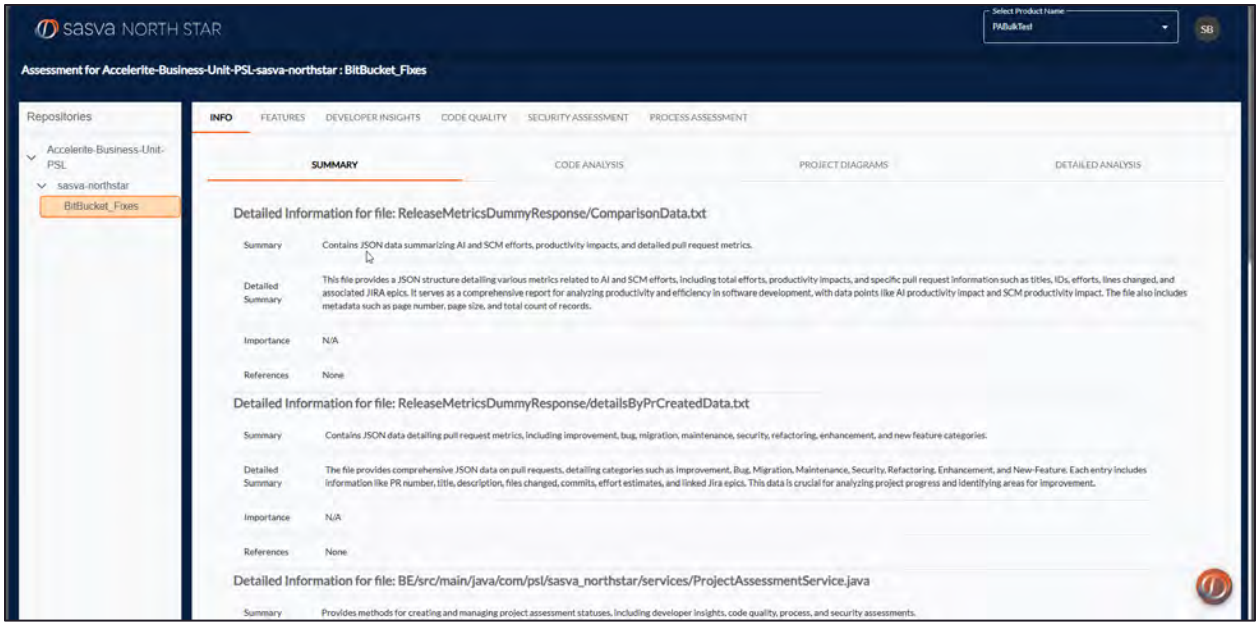
Manage Project Backlog

The Manage Backlog feature is designed to help you efficiently generate and maintain a project backlog. This functionality allows you to:

- Identify and Prioritize Tasks: Automatically generate a list of tasks and enhancements based on the latest market trends and user feedback.
- Integrate Market Features: Seamlessly add new features that align with current market demands to your application.
- Optimize Development Workflow: Streamline your development process by focusing on high-priority items and reducing bottlenecks.

Once a project is added to the Manage Backlog view, North Star automatically generates comprehensive information as follows:

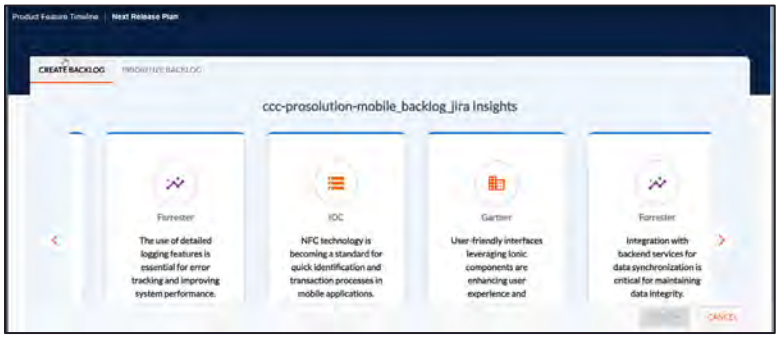
- Release History: Shows detailed analysis of release history with timeline and number of features added.
- Summary: Shows detailed information of each file that contains Detailed Summary, Importance, and references.
- Code Analysis: Detailed insights into the code quality and potential issues.
- Project Diagrams: Visual representations of the project's structure and components.
- Detailed Analysis: In-depth examination of the project's files and their interactions.
- Feature Overview: A summary of the application's features and functionalities.



Create Project Backlog

Follow the steps to manage the backlog:

1. Click **MANAGE BACKLOG**.

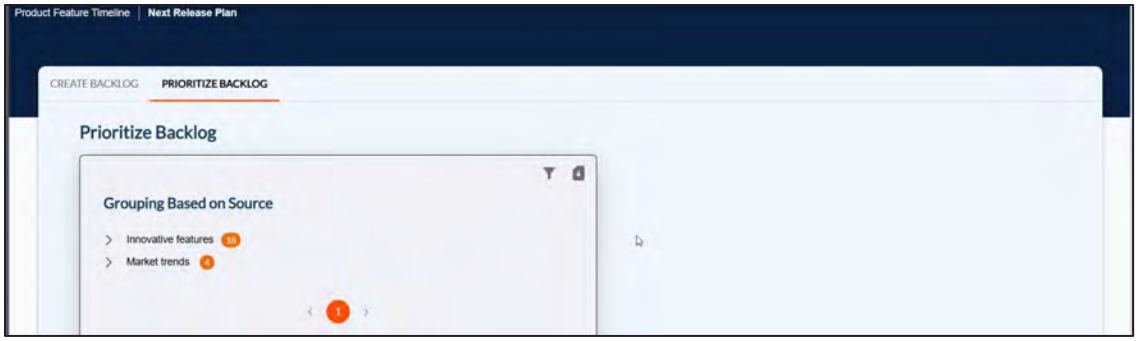


1. North Star recommends the features based on the market study. These features are divided into three categories: Competitive Features, Innovative Features, and Market Trends.
2. Select **Feature** and drag and drop into product backlog.
3. Click **Publish** and **Confirm** to release your plan in the project management tool.

Prioritize backlog

Allow you to manage and prioritize the features in the project management tool. Follow the steps to prioritize backlog:

1. Click **MANAGE BACKLOG**.
1. Click **Prioritize Backlog**.



2. Open the drop-down menu to view the list of features. Select the feature you want to prioritize by clicking the ⊕ icon. The feature will be added to the Prioritize Backlog list.

3. Select the feature, click the **Priority** drop-down, and select your priority.
4. Click the **VIEW** button to see the Business Case for the feature.
5. Click **PUBLISH** to update the priority list in your project management tool.

User Profile

The User Profile section provides access to various features and settings that allow you to manage your account and preferences. The following menu options are available:

\ Projects

This section allows you to view and manage all your projects. You can view all the configured projects and access them.

\ Settings

The Settings menu includes several important configuration options:

- Email Configuration: Set up and manage your email preferences, including notifications and communication settings.
- User Management: Manage user roles and permissions. Add or remove users and assign specific roles to control access to different features.
- Guidelines: Add the Organization or User guidelines documents to North star
- Change Password : Update your password to ensure your account remains secure.

\ Logout

Use this option to securely log out of your account when you are finished using the platform.

Email Configuration

You can update your email configuration by following these steps:

1. Login to North Star with Administrator credentials.
2. Click **User Profile**.
3. Click **Settings** and select **Email Configuration**.
4. Edit the email configuration details as follows:
    a. **Sender Email ID**: Enter the email address that will appear as the sender.
    b. **Username**: Enter the username associated with the email account.
    c. **Password**: Enter the password for the email account.
    d. **Host**: Enter the email server host (e.g., smtp.example.com).
    e. **Port Number**: Enter the port number used by the email server (e.g., 587 for TLS, 465 for SSL).
5. Click **Update** to save the details.

User Management

This section contains the user management settings:

Applications

You can view the roles assigned to your profile.

Users

In this section, you can add a user and assign them roles.

Roles and permissions

User Role	Projects Page		Settings Page		Add Project Page		Project Timeline Page	Add New Release Plan Page
	Edit/Delete Project							
	Logged-in user added	Other user added	Email Configuration	User Management	Save Project	Add Mapping	All sections	
Super Administrator	Yes	Yes	Yes	Yes	Yes		Yes	Yes
Tenant Administrator	Yes	Yes	Yes	Yes	Yes		Yes	Yes
Primary Administrator	Yes	Yes	No		Yes		Yes	Yes
Secondary Administrator	Yes	No	No		Yes		Yes	No
Operator	No	No	No		Yes	No	Yes	No

1. To add a user click **Add User**.
    a. Type the **First name**, **Last name**, **Email**, and **Phone number** of the user.
    b. Select **Status** for the user.
    c. Assign the **Role** for the user.
    d. Click **Submit**.
2. To edit existing user details.
    a. Select a user for which you wish to edit the details.
    b. Click **Edit** and update the information.
    c. Click **Update** to save updated information.
3. To remove the user access.
    a. Select the user for which you wish to revoke the access.
    b. Click **Delete** button.
    c. A confirmation pop-up appears. Click **Confirm** to delete access.

Guidelines

This feature allows you to upload organizational or personal guidelines documents relevant to your project. By leveraging machine learning, the system extracts valuable insights from these documents. These guidelines will be followed while generating project data. Follow the steps to upload document:

1. Click the **User Profile**.
2. Select **Settings** and click **Guidelines**.
3. **Provide** the following details:
    - **Document Title**: Enter the title of the document.
    - **Document Description**: Provide a brief description of the document.
    - Document Category:
        - **Organization**: Select this if the document pertains to organizational guidelines.
        - **User**: Select this if the document pertains to personal guidelines
    - Source Type:
        - **File**: Choose the option to upload the documents. Upload files by clicking the **File Upload** box.
        - **URL**: Choose this option to provide SharePoint or Confluence URL. Provide the necessary details to connect to SharePoint or Confluence, then click **Get Document List** and select the necessary documents.
    - Choose **Document Type**(s): Check the appropriate boxes to define the document category (e.g., Security, Privacy, Quality, etc.).

.

4. Click **UPLOAD** to submit the document. The document will be added to the list and will be visible in the Uploaded document list.
5. You can add more documents, by clicking **Add Another Document**.

Change Password

To update your password, follow these steps:

1. Click the **User Profile**.
2. Select **Settings** and click **Change Password**.
3. Enter the **New Password**.
4. Confirm the **New Password**.
5. Click **Change Password** to save the new password.

Reference

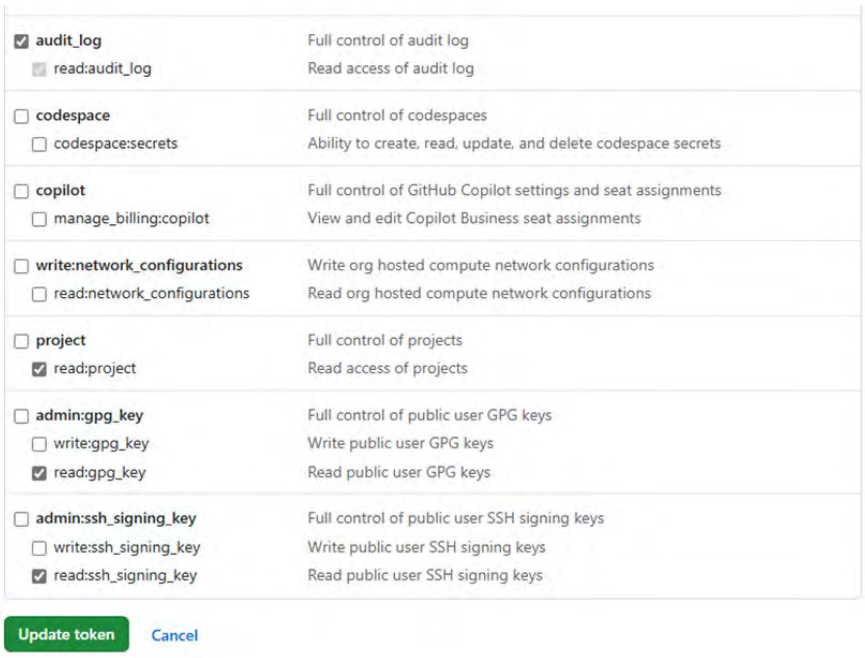
PAT Classic token for GitHub

Follow these steps to generate a Git Personal Access Token (PAT):

1. Log in to your **GitHub** account.
2. Click your **profile picture** in the top right corner.
3. Select **Settings**.
4. In the left sidebar, go to **Developer settings**.
5. Click **Personal access token**>**Token (classic).**
6. Click **Generate new token**>**Generate new token (Classic)**.
7. Provide a **note** to describe the purpose of the token.
8. Select an **expiration date** for the token.
9. Choose the necessary **scopes/permissions** for the token.

Scopes define the access for personal tokens. Read more about OAuth scopes.

☑ **repo**	Full control of private repositories
☑ repo:status	Access commit status
☑ repo_deployment	Access deployment status
☑ public_repo	Access public repositories
☑ repo:invite	Access repository invitations
☑ security_events	Read and write security events
☐ **workflow**	Update GitHub Action workflows
☐ **write:packages**	Upload packages to GitHub Package Registry
☑ read:packages	Download packages from GitHub Package Registry
☐ **delete:packages**	Delete packages from GitHub Package Registry
☐ **admin:org**	Full control of orgs and teams, read and write org projects
☐ write:org	Read and write org and team membership, read and write org projects
☑ read:org	Read org and team membership, read org projects
☐ manage_runners:org	Manage org runners and runner groups
☐ **admin:public_key**	Full control of user public keys
☐ write:public_key	Write user public keys
☑ read:public_key	Read user public keys
☐ **admin:repo_hook**	Full control of repository hooks
☐ write:repo_hook	Write repository hooks
☑ read:repo_hook	Read repository hooks
☐ **admin:org_hook**	Full control of organization hooks
☐ **gist**	Create gists
☐ **notifications**	Access notifications
☐ **user**	Update ALL user data
☑ read:user	Read ALL user profile data
☐ user:email	Access user email addresses (read-only)
☐ user:follow	Follow and unfollow users
☐ **delete_repo**	Delete repositories
☐ **write:discussion**	Read and write team discussions
☑ read:discussion	Read team discussions
☐ **admin:enterprise**	Full control of enterprises
☐ manage_runners:enterprise	Manage enterprise runners and runner groups
☐ manage_billing:enterprise	Read and write enterprise billing data
☑ read:enterprise	Read enterprise profile data
☐ scim:enterprise	Provisioning of users and groups via SCIM



10. Click **Generate token**.
11. Copy the generated token to your clipboard.

API Token for Jira

Follow these steps to generate a Jira API Token:

1. Log in to Jira.
2. Click your profile picture.
3. Select **Profile.**
4. Click **Manage Your Account**.
5. Navigate to the **Security** tab.
6. Click **Create and manage API tokens**.
7. Click **Create API token**.
8. Provide a label for the token.
9. Select an expiration date for the token.
10. Click **Create**.
11. Copy the token to your clipboard.

Related Documentation

You can access the SASVA product documentation from the following link:
SASVA Support Portal

SASVA Support

You can reach out for support at the support@sasva.ai.