Well-Architected Component Documentation

Overview

The well-architected component in this Angular application is designed to ensure an almost full integration of AWS Well-Architected functionalities following best practices. It allows users to create, manage, and review lenses for AWS workloads, ensuring high standards of performance, security, and cost-efficiency.

Components Overview

1. Access Component

Functionality
Handles user access and permissions, enabling secure interactions with the system's features.

Key Features
Download AWS CloudFormation Template: Users can download a predefined template to deploy resources efficiently.
Enter and Verify AWS Account IDs: Allows users to enter their AWS account IDs to check If the template was deployed successfully.

2. List Component

Functionality
Displays a comprehensive list of workloads and lenses along with associated risks and management options.

Key Features
Lenses Management: Features options for editing, deleting, and viewing detailed information about each lens.
Workload Management: Features options for editing, deleting, and viewing detailed information about each workload.
Risk Assessment Visualization: Displays risk levels (high, medium, low) associated with each workload, helping prioritize management efforts.

3. Create Component

Functionality
Facilitates the creation of custom lenses tailored to specific AWS workloads.

Key Features

Custom Lens Creation Form: A detailed form allowing input of lens name, description, and associated questions and choices concerning risk management.

4. Review Component

Functionality

Provides a detailed review of lenses, highlighting areas of risk and requiring attention.

Key Features

Risk Level Indicators: Indicates the number of high, medium, and no risk questions unanswered within a lens.

Interactive Review Summaries: Users can interact with each pillar of the lens to view specific risks associated and manage them accordingly.

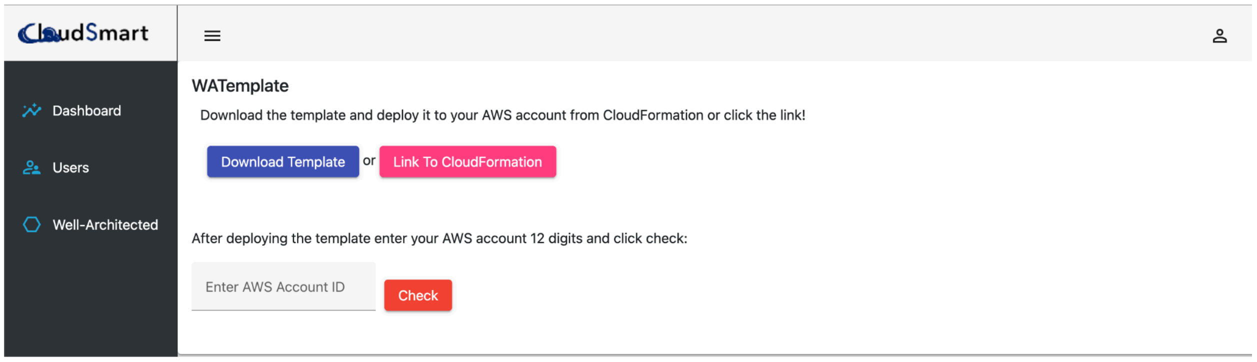
Pillar answers: Users can view each pillar's answers highlighted within the question's choices according to its risk type and edit them.

User Interface

The application features a clean and intuitive interface, with each component designed to facilitate ease of use and accessibility.

Screenshots

Access Component:

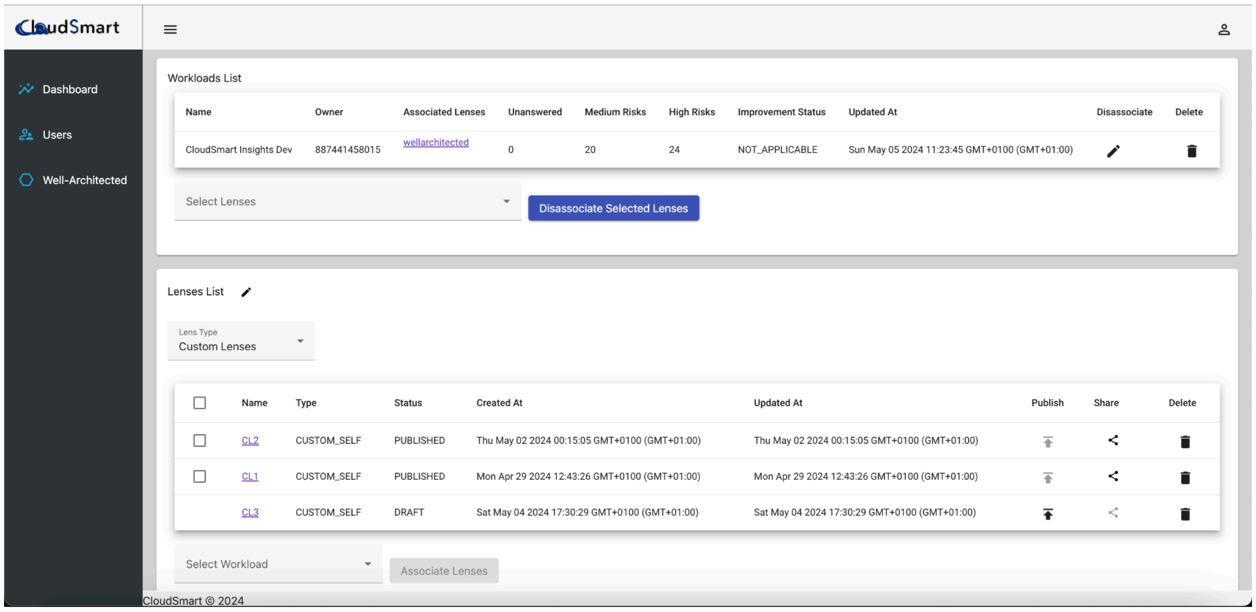


WATemplate

Download the template and deploy it to your AWS account from CloudFormation or click the link!

After deploying the template enter your AWS account 12 digits and click check:

List Component:



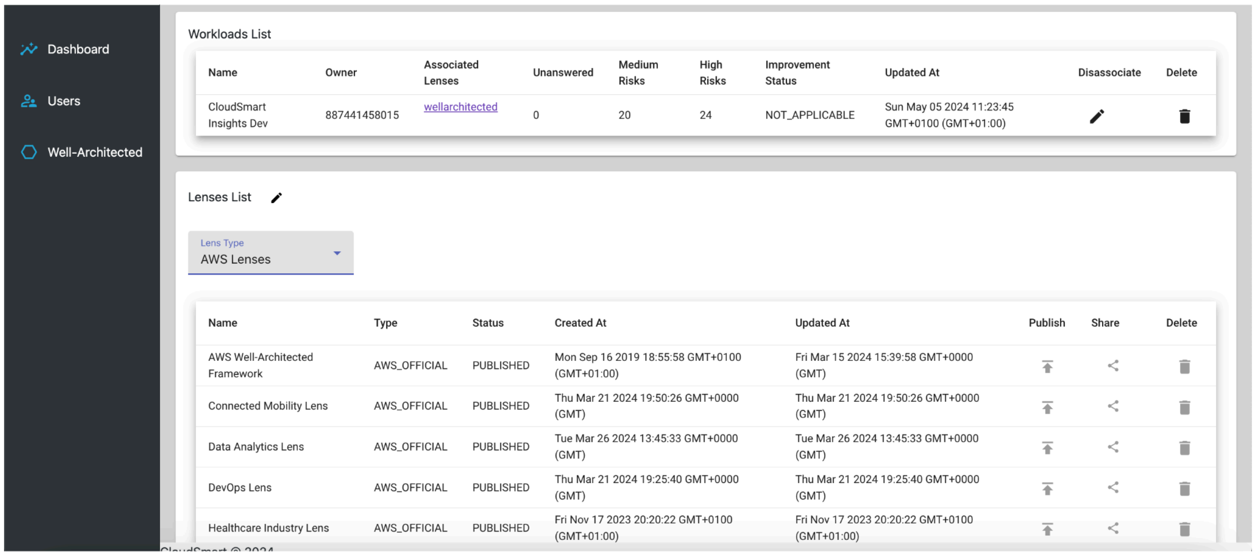
Workloads List

Name	Owner	Associated Lenses	Unanswered	Medium Risks	High Risks	Improvement Status	Updated At	Disassociate	Delete
CloudSmart Insights Dev	887441458015	wellarchitected	0	20	24	NOT_APPLICABLE	Sun May 05 2024 11:23:45 GMT+0100 (GMT+01:00)		

Lenses List

Lens Type: Custom Lenses

	Name	Type	Status	Created At	Updated At	Publish	Share	Delete
☐	CL2	CUSTOM_SELF	PUBLISHED	Thu May 02 2024 00:15:05 GMT+0100 (GMT+01:00)	Thu May 02 2024 00:15:05 GMT+0100 (GMT+01:00)			
☐	CL1	CUSTOM_SELF	PUBLISHED	Mon Apr 29 2024 12:43:26 GMT+0100 (GMT+01:00)	Mon Apr 29 2024 12:43:26 GMT+0100 (GMT+01:00)			
	CL3	CUSTOM_SELF	DRAFT	Sat May 04 2024 17:30:29 GMT+0100 (GMT+01:00)	Sat May 04 2024 17:30:29 GMT+0100 (GMT+01:00)			



Workloads List

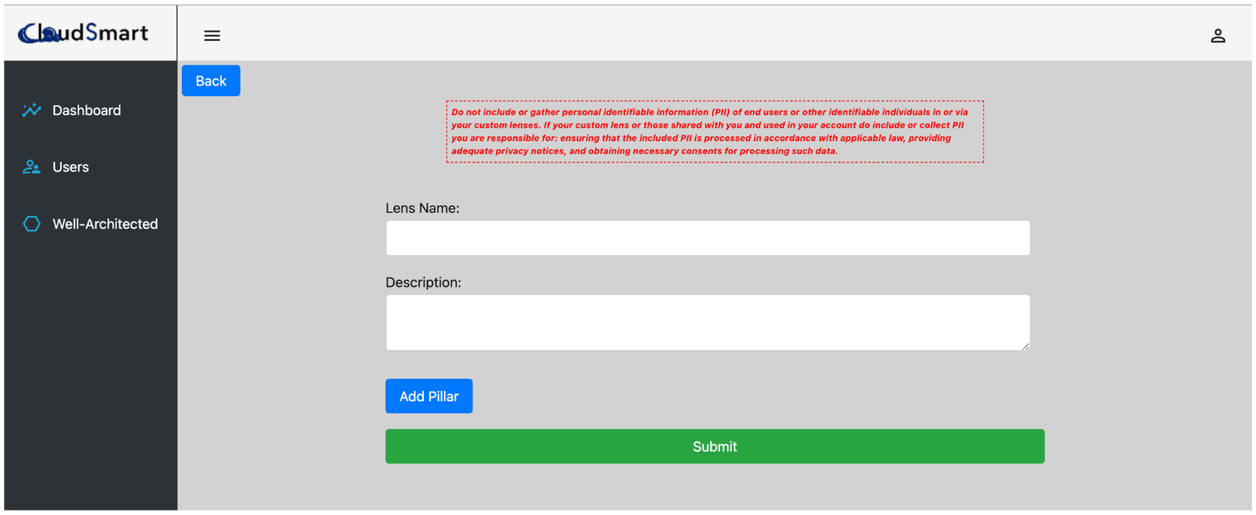
Name	Owner	Associated Lenses	Unanswered	Medium Risks	High Risks	Improvement Status	Updated At	Disassociate	Delete
CloudSmart Insights Dev	887441458015	wellarchitected	0	20	24	NOT_APPLICABLE	Sun May 05 2024 11:23:45 GMT+0100 (GMT+01:00)		

Lenses List

Lens Type: AWS Lenses

Name	Type	Status	Created At	Updated At	Publish	Share	Delete
AWS Well-Architected Framework	AWS_OFFICIAL	PUBLISHED	Mon Sep 16 2019 18:55:58 GMT+0100 (GMT+01:00)	Fri Mar 15 2024 15:39:58 GMT+0000 (GMT)			
Connected Mobility Lens	AWS_OFFICIAL	PUBLISHED	Thu Mar 21 2024 19:50:26 GMT+0000 (GMT)	Thu Mar 21 2024 19:50:26 GMT+0000 (GMT)			
Data Analytics Lens	AWS_OFFICIAL	PUBLISHED	Tue Mar 26 2024 13:45:33 GMT+0000 (GMT)	Tue Mar 26 2024 13:45:33 GMT+0000 (GMT)			
DevOps Lens	AWS_OFFICIAL	PUBLISHED	Thu Mar 21 2024 19:25:40 GMT+0000 (GMT)	Thu Mar 21 2024 19:25:40 GMT+0000 (GMT)			
Healthcare Industry Lens	AWS_OFFICIAL	PUBLISHED	Fri Nov 17 2023 20:20:22 GMT+0100 (GMT+01:00)	Fri Nov 17 2023 20:20:22 GMT+0100 (GMT+01:00)			

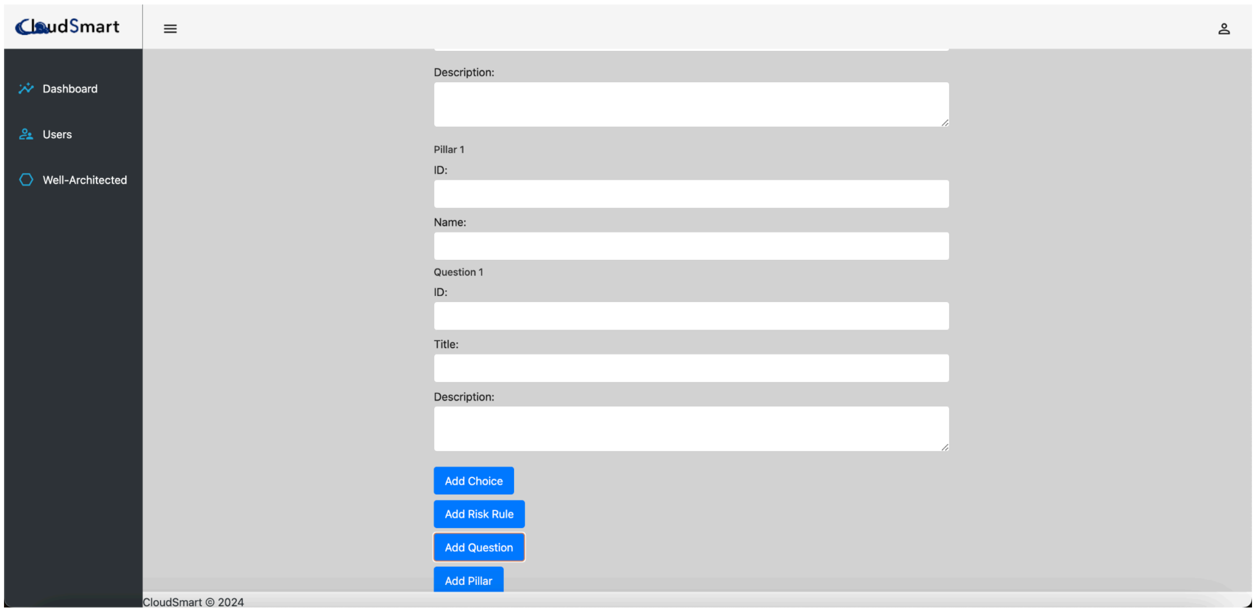
Create Component:



Do not include or gather personal identifiable information (PII) of end users or other identifiable individuals in or via your custom lenses. If your custom lens or those shared with you and used in your account do include or collect PII you are responsible for: ensuring that the included PII is processed in accordance with applicable law, providing adequate privacy notices, and obtaining necessary consents for processing such data.

Lens Name:

Description:



Description:

Pillar 1

ID:

Name:

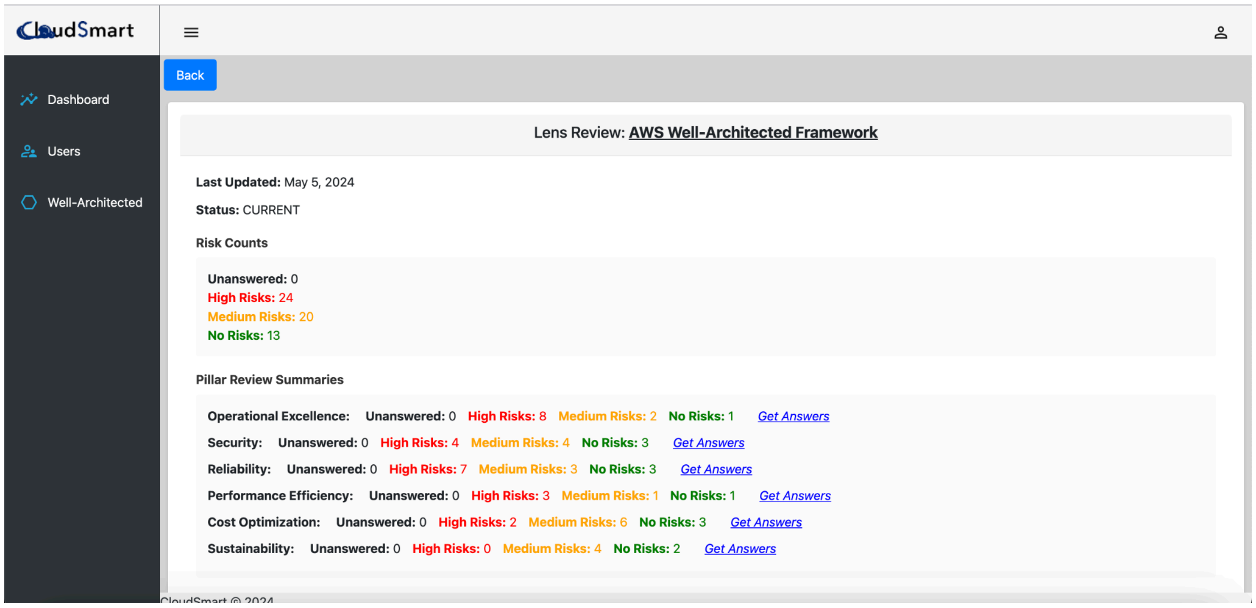
Question 1

ID:

Title:

Description:

Review Component:



Lens Review: **AWS Well-Architected Framework**

Last Updated: May 5, 2024

Status: CURRENT

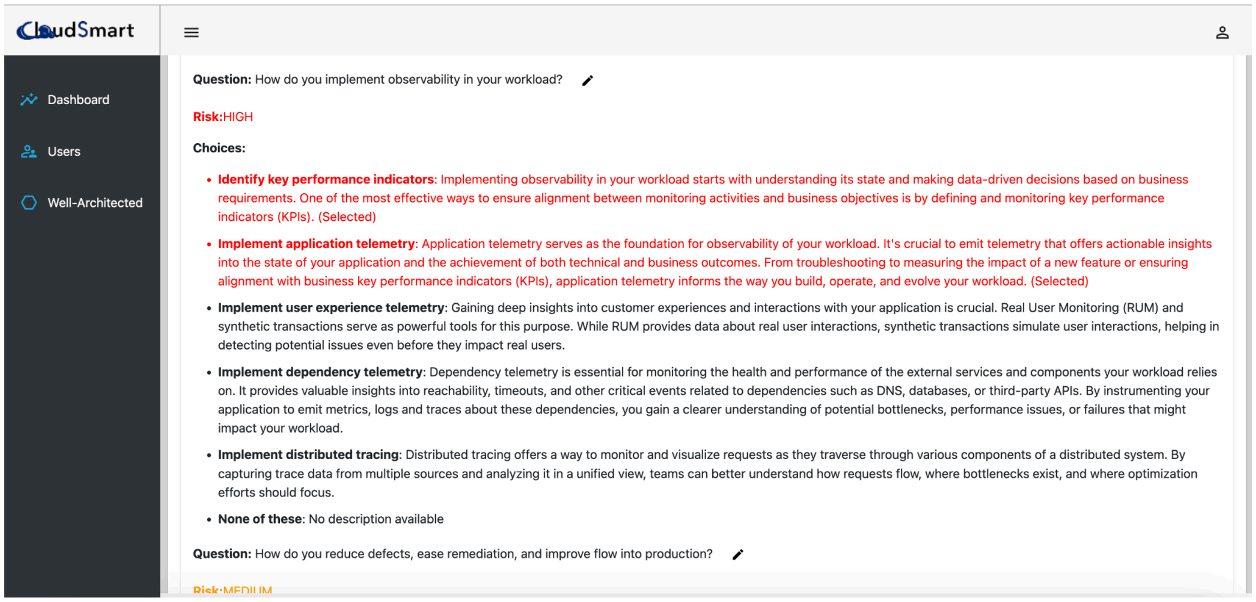
Risk Counts

Unanswered: 0
High Risks: 24
Medium Risks: 20
No Risks: 13

Pillar Review Summaries

Operational Excellence:	**Unanswered:** 0	**High Risks:** 8	**Medium Risks:** 2	**No Risks:** 1	Get Answers
Security:	**Unanswered:** 0	**High Risks:** 4	**Medium Risks:** 4	**No Risks:** 3	Get Answers
Reliability:	**Unanswered:** 0	**High Risks:** 7	**Medium Risks:** 3	**No Risks:** 3	Get Answers
Performance Efficiency:	**Unanswered:** 0	**High Risks:** 3	**Medium Risks:** 1	**No Risks:** 1	Get Answers
Cost Optimization:	**Unanswered:** 0	**High Risks:** 2	**Medium Risks:** 6	**No Risks:** 3	Get Answers
Sustainability:	**Unanswered:** 0	**High Risks:** 0	**Medium Risks:** 4	**No Risks:** 2	Get Answers

CloudSmart © 2024



Question: How do you implement observability in your workload?

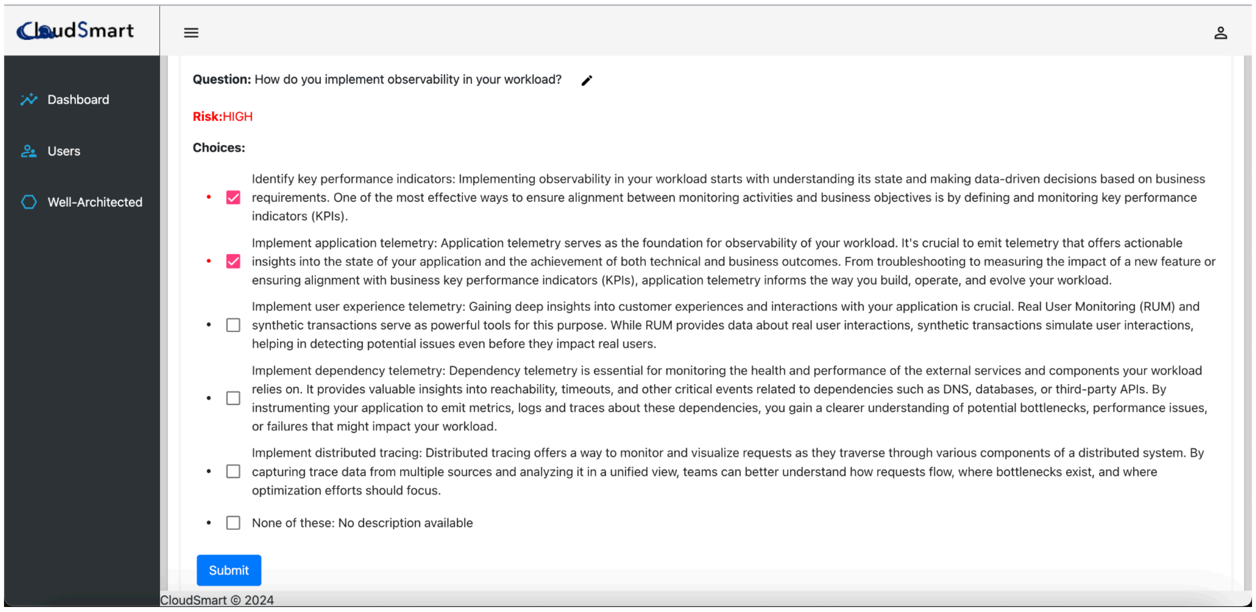
Risk:HIGH

Choices:

- **Identify key performance indicators**: Implementing observability in your workload starts with understanding its state and making data-driven decisions based on business requirements. One of the most effective ways to ensure alignment between monitoring activities and business objectives is by defining and monitoring key performance indicators (KPIs). (Selected)
- **Implement application telemetry**: Application telemetry serves as the foundation for observability of your workload. It's crucial to emit telemetry that offers actionable insights into the state of your application and the achievement of both technical and business outcomes. From troubleshooting to measuring the impact of a new feature or ensuring alignment with business key performance indicators (KPIs), application telemetry informs the way you build, operate, and evolve your workload. (Selected)
- **Implement user experience telemetry**: Gaining deep insights into customer experiences and interactions with your application is crucial. Real User Monitoring (RUM) and synthetic transactions serve as powerful tools for this purpose. While RUM provides data about real user interactions, synthetic transactions simulate user interactions, helping in detecting potential issues even before they impact real users.
- **Implement dependency telemetry**: Dependency telemetry is essential for monitoring the health and performance of the external services and components your workload relies on. It provides valuable insights into reachability, timeouts, and other critical events related to dependencies such as DNS, databases, or third-party APIs. By instrumenting your application to emit metrics, logs and traces about these dependencies, you gain a clearer understanding of potential bottlenecks, performance issues, or failures that might impact your workload.
- **Implement distributed tracing**: Distributed tracing offers a way to monitor and visualize requests as they traverse through various components of a distributed system. By capturing trace data from multiple sources and analyzing it in a unified view, teams can better understand how requests flow, where bottlenecks exist, and where optimization efforts should focus.
- **None of these**: No description available

Question: How do you reduce defects, ease remediation, and improve flow into production?

Risk:MEDIUM



Question: How do you implement observability in your workload?

Risk:HIGH

Choices:

- [✓] Identify key performance indicators: Implementing observability in your workload starts with understanding its state and making data-driven decisions based on business requirements. One of the most effective ways to ensure alignment between monitoring activities and business objectives is by defining and monitoring key performance indicators (KPIs).
- [✓] Implement application telemetry: Application telemetry serves as the foundation for observability of your workload. It's crucial to emit telemetry that offers actionable insights into the state of your application and the achievement of both technical and business outcomes. From troubleshooting to measuring the impact of a new feature or ensuring alignment with business key performance indicators (KPIs), application telemetry informs the way you build, operate, and evolve your workload.
- [ ] Implement user experience telemetry: Gaining deep insights into customer experiences and interactions with your application is crucial. Real User Monitoring (RUM) and synthetic transactions serve as powerful tools for this purpose. While RUM provides data about real user interactions, synthetic transactions simulate user interactions, helping in detecting potential issues even before they impact real users.
- [ ] Implement dependency telemetry: Dependency telemetry is essential for monitoring the health and performance of the external services and components your workload relies on. It provides valuable insights into reachability, timeouts, and other critical events related to dependencies such as DNS, databases, or third-party APIs. By instrumenting your application to emit metrics, logs and traces about these dependencies, you gain a clearer understanding of potential bottlenecks, performance issues, or failures that might impact your workload.
- [ ] Implement distributed tracing: Distributed tracing offers a way to monitor and visualize requests as they traverse through various components of a distributed system. By capturing trace data from multiple sources and analyzing it in a unified view, teams can better understand how requests flow, where bottlenecks exist, and where optimization efforts should focus.
- [ ] None of these: No description available

Code Snippets

Here are some important code snippets from each component:

Access Component

```
async assumeRole(awsAccountId: string) {
   const stsClient = new STSClient({
     region: 'us-east-1',
     credentials: {
       accessKeyId: environment.AccessKeyId,
       secretAccessKey: environment.SecretAccessKey
     }
   });
   const roleArn = `arn:aws:iam::${awsAccountId}:role/TenantCustomLens`;
   const roleSessionName = 'TenantLensSession';
   const assumeRoleCommand = new AssumeRoleCommand({
     RoleArn: roleArn,
     RoleSessionName: roleSessionName,
   });

   const { Credentials } = await stsClient.send(assumeRoleCommand);
   if (!Credentials || !Credentials.AccessKeyId || !Credentials.SecretAccessKey ||
!Credentials.SessionToken) {
     throw new Error('Failed to retrieve AWS credentials');
   }
   return Credentials;
 }
```

List Component

```
<div class="cs-list">
   <mat-card class="card">
       <mat-card-content>
       <mat-card-title>
           Workloads List
       </mat-card-title>
           <div style="margin: 10px;">
               <div>
                   <table mat-table [dataSource]="workloads" class="mat-elevation-z8"
style="width: 100%;">
                       <!-- Workload Name Column -->
                       <ng-container matColumnDef="name">
                           <th mat-header-cell *matHeaderCellDef>Name</th>
                           <td mat-cell *matCellDef="let workload ">{{
workload.WorkloadName }}</td>
                       </ng-container>

                       <!-- Workload Owner Column -->
                       <ng-container matColumnDef="owner">
                           <th mat-header-cell *matHeaderCellDef>Owner</th>
                           <td mat-cell *matCellDef="let workload">{{ workload.Owner
}}</td>
                       </ng-container>

                       <ng-container matColumnDef="lenses">
                           <th mat-header-cell *matHeaderCellDef>Associated
Lenses</th>
                           <td mat-cell *matCellDef="let workload">
                               <ng-container *ngFor="let lens of workload.Lenses; let
isLast = last">
                                   <a [routerLink]="['/wellarchitected/review',
awsAccountId, workload.WorkloadId, lens]"
                                       class="truncated-text" matTooltip="{{ lens }}">
                                       {{ lens }}
                                   </a>
                                   <br>
                               </ng-container>
```

```
                                </td>
                            </ng-container>


                            <!-- Workload Medium Risks Column -->
                            <ng-container matColumnDef="riskU">
                                <th mat-header-cell *matHeaderCellDef>Unanswered</th>
                                <td mat-cell *matCellDef="let workload">{{
workload.RiskCounts.UNANSWERED || 0 }}</td>
                            </ng-container>

                            <!-- Workload High Risks Column -->
                            <ng-container matColumnDef="riskM">
                                <th mat-header-cell *matHeaderCellDef>Medium Risks</th>
                                <td mat-cell *matCellDef="let workload">{{
workload.RiskCounts.MEDIUM || 0 }} </td>
                            </ng-container>

                            <!-- Workload High Risks Column -->
                            <ng-container matColumnDef="riskH">
                                <th mat-header-cell *matHeaderCellDef>High Risks</th>
                                <td mat-cell *matCellDef="let workload">{{
workload.RiskCounts.HIGH || 0 }} </td>
                            </ng-container>

                            <!-- Workload Improvement Status Column -->
                            <ng-container matColumnDef="improve">
                                <th mat-header-cell *matHeaderCellDef>Improvement
Status</th>
                                <td mat-cell *matCellDef="let workload">{{
workload.ImprovementStatus }} </td>
                            </ng-container>

                            <!-- Updated At Column -->
                            <ng-container matColumnDef="updated">
                                <th mat-header-cell *matHeaderCellDef>Updated At</th>
                                <td mat-cell *matCellDef="let workload">{{
workload.UpdatedAt }}</td>
                            </ng-container>

                            <!-- Edit Column -->
                            <ng-container matColumnDef="edit">
```

```
                           <th mat-header-cell *matHeaderCellDef>Disassociate</th>
                           <td mat-cell *matCellDef="let workload">
                               <button mat-icon-button
(click)="showEditDiv(workload)">
                                   <mat-icon>edit</mat-icon>
                               </button>
                           </td>
                       </ng-container>

                       <!-- Delete Column -->
                       <ng-container matColumnDef="delete">
                           <th mat-header-cell *matHeaderCellDef>Delete</th>
                           <td mat-cell *matCellDef="let workload">
                               <button mat-icon-button
(click)="deleteworkload(workload.WorkloadId)">
                                   <mat-icon>delete</mat-icon>
                               </button>
                           </td>
                       </ng-container>

                       <tr mat-header-row *matHeaderRowDef="displayedColumns0"></tr>
                       <tr mat-row *matRowDef="let workload; columns:
displayedColumns0"></tr>
                   </table>
                   <mat-card *ngIf="isLoading" style="display: flex; justify-content:
center; align-items: center">
                       <mat-progress-spinner color="primary" mode="indeterminate"
diameter="15"></mat-progress-spinner>
                   </mat-card>
               </div>
               <div *ngIf="selectedWorkload" style="margin-top: 20px;">
                   <mat-form-field style="width: 500px;">
                       <mat-label>Select Lenses</mat-label>
                       <mat-select [(value)]="selectedLensesW" multiple>
                           <mat-option *ngFor="let lens of selectedWorkload.Lenses"
[value]="lens">
                               {{ lens }}
                           </mat-option>
                       </mat-select>
                   </mat-form-field>
                   <button mat-raised-button color="primary" style="margin-left:
10px;" (click)="disassociateLenses(selectedWorkload.WorkloadId,
selectedLensesW)">Disassociate Selected Lenses</button>
               </div>
           </div>
       </mat-card-content>
   </mat-card>
</div>
```

Create Component

```
<form [formGroup]="lensForm" (ngSubmit)="onSubmit()">
   <div>
     <label for="name">Lens Name:</label>
     <input id="name" formControlName="name" type="text">
   </div>
   <div>
     <label for="description">Description:</label>
     <textarea id="description" formControlName="description"></textarea>
   </div>
    <div formArrayName="pillars">
     <div *ngFor="let pillar of getPillars().controls; let i=index"
[formGroupName]="i">
       <h3>Pillar {{ i + 1 }}</h3>
       <div>
         <label>ID:</label>
         <input formControlName="id" type="text">
       </div>
       <div>
         <label>Name:</label>
         <input formControlName="name" type="text">
       </div>
        <div formArrayName="questions">
         <div *ngFor="let question of getQuestions(i).controls; let j=index"
[formGroupName]="j">
           <h4>Question {{ j + 1 }}</h4>
           <div>
             <label>ID:</label>
             <input formControlName="id" type="text">
           </div>
           <div>
             <label>Title:</label>
             <input formControlName="title" type="text">
           </div>
           <div>
             <label>Description:</label>
             <textarea formControlName="description"></textarea>
           </div>
            <div formArrayName="choices">
               <div *ngFor="let choice of getChoices(i, j).controls; let k=index"
[formGroupName]="k">
                   <h5>Choice {{ k + 1 }}</h5>
                   <div>
                     <label>ID:</label>
                     <input formControlName="id" type="text">
                   </div>
                   <div>
                     <label>Title:</label>
                     <input formControlName="title" type="text">
                   </div>
                   <ng-container *ngIf="!choice.get('id')?.value?.endsWith('_no') &&
choice.get('title')?.value?.toLowerCase() !== 'none of these'">
                       <div formGroupName="improvementPlan">
                         <div>
                           <label>Improvement Plan Display Text:</label>
                           <input formControlName="displayText" type="text">
                         </div>
                         <div>
                           <label>Improvement Plan URL:</label>
                           <input formControlName="url" type="text">
                         </div>
                       </div>
                   </ng-container>
                 </div>
             <button type="button" (click)="addChoice(i, j)">Add Choice</button>
           </div>
            <div formArrayName="riskRules">
               <div *ngFor="let riskRule of getRiskRules(i, j).controls; let l=index"
[formGroupName]="l">
                 <h5>Risk Rule {{ l + 1 }}</h5>
                 <div>
                   <label>Condition:</label>
                   <input formControlName="condition" type="text">
                 </div>
                 <div>
                   <label>Risk:</label>
                   <select formControlName="risk">
                     <option value="HIGH_RISK">High Risk</option>
                     <option value="MEDIUM_RISK">Medium Risk</option>
                     <option value="NO_RISK">No Risk</option>
                   </select>
                 </div>
               </div>
               <button type="button" (click)="addRiskRule(i, j)">Add Risk
Rule</button>
             </div>
         </div>
         <button type="button" (click)="addQuestion(i)">Add Question</button>
       </div>
     </div>
     <button type="button" (click)="addPillar()">Add Pillar</button>
   </div>
    <button type="submit">Submit</button>
   <div *ngIf="errorMessage" class="error-message">
       {{ errorMessage }}
   </div>
   <div *ngIf="successMessage" class="success-message">
       {{ successMessage }}
   </div>
</form>
```

Review Component

```
<button class="back-button" [routerLink]="['/wellarchitected/list', awsAccountId]"
*ngIf="!showAnswers">Back</button>
<button class="back-button" (click)="toggleView()" *ngIf="showAnswers">Back</button>
<div class="cs-list" *ngIf="!showAnswers">
   <mat-card class="card" *ngIf="review">
       <mat-card-content>
           <div class="card-header">
               <mat-card-title style="text-align: center;">
                   <h5>Lens Review: <b><u>{{ review?.LensName }}</u></b></h5>
               </mat-card-title>
               <div *ngIf="review.RiskCounts?.UNANSWERED != 0 &&
review.RiskCounts?.HIGH == 0 && review.RiskCounts?.MEDIUM == 0 &&
review.RiskCounts?.NONE == 0">
```

```
                   <p class="warning-text" style="color: red; text-align:
center;">This lens has not been reviewed yet!</p>
               </div>
           </div>

           <div class="card-body">
               <p class="card-info"><strong>Last Updated:</strong> {{
review?.UpdatedAt | date }}</p>
               <p class="card-info"><strong>Status:</strong> {{ review?.LensStatus
}}</p>

               <h6 class="risk-header">Risk Counts</h6>
               <div class="risk-counts">
                   <p class="risk-info"><strong>Unanswered:</strong> {{
review.RiskCounts?.UNANSWERED || '0'  }}</p>
                   <p class="risk-info" style="color: red;"><strong>High
Risks:</strong> {{ review.RiskCounts?.HIGH || '0'  }}</p>
                   <p class="risk-info" style="color: orange"><strong>Medium
Risks:</strong> {{ review.RiskCounts?.MEDIUM || '0'  }}</p>
                   <p class="risk-info" style="color: green"><strong>No
Risks:</strong> {{ review.RiskCounts?.NONE || '0'  }}</p>
               </div>

               <h6 class="pillar-header">Pillar Review Summaries</h6>
               <div class="pillar-summaries">
                   <div *ngFor="let summary of review?.PillarReviewSummaries"
class="pillar-summary">
                       <div class="pillar-name"><strong>{{ summary?.PillarName
}}:</strong></div>
                       <div class="risk-detail">
                           <p class="risk-info"><strong>Unanswered:</strong> {{
review.RiskCounts?.UNANSWERED || '0'  }}</p>
                           <p class="risk-info" style="color: red;"><strong>High
Risks:</strong> {{ summary?.RiskCounts?.HIGH || '0' }}</p>
                           <p class="risk-info" style="color: orange;"><strong>Medium
Risks:</strong> {{ summary?.RiskCounts?.MEDIUM || '0' }}</p>
                           <p class="risk-info" style="color: green;"><strong>No
Risks:</strong> {{ summary?.RiskCounts?.NONE || '0' }}</p>
                           <span style="color: blue; margin-left: 1rem;"
(click)="getAnswers(summary?.PillarId, summary?.PillarName)"><i><u>Get
Answers</u></i></span>
                       </div>
                   </div>
               </div>
           </div>
       </mat-card-content>
   </mat-card>
   <mat-card *ngIf="isLoading" style="display: flex; justify-content: center;
align-items: center">
       <mat-progress-spinner color="primary" mode="indeterminate"
diameter="15"></mat-progress-spinner>
   </mat-card>
</div>

<div class="cs-list" *ngIf="showAnswers">
   <mat-card class="card">
       <mat-card-content>
       <div class="card-header">
           <mat-card-title style="text-align: center;">
           <h5>Questions Answers for Pillar: <b><u>{{ currentPillarName
}}</u></b></h5>
           </mat-card-title>
       </div>
       <mat-card class="card">
           <mat-card-content>
           <div *ngFor="let answer of answers; let i = index">
               <p>
               <strong>Question:</strong> {{ answer.QuestionTitle }}
               <button mat-button
(click)="toggleSelection(i)"><mat-icon>edit</mat-icon></button>
               </p>
               <p [ngStyle]="{'color': getRiskColor(answer.Risk)}">
               <strong>Risk:</strong>{{ answer.Risk }}
               </p>
               <p><strong>Choices:</strong></p>
               <ul>
               <li *ngFor="let choice of answer.Choices" [ngStyle]="{'color':
answer.SelectedChoices.includes(choice.ChoiceId) ? getRiskColor(answer.Risk) :
'inherit'}" class="choice-item">
                   <ng-container *ngIf="!answer.editMode">
                   <strong>{{ choice.Title }}</strong>
                   <span>: {{ choice.Description || 'No description available'
}}</span>
```

```
                   <span *ngIf="answer.SelectedChoices.includes(choice.ChoiceId)">
(Selected)</span>
                   </ng-container>
                   <ng-container *ngIf="answer.editMode">
                   <mat-checkbox [(ngModel)]="choice.selected">{{ choice.Title }}: {{
choice.Description || 'No description available' }}</mat-checkbox>
                   </ng-container>
               </li>
               </ul>
               <button class="back-button" *ngIf="answer.editMode"
(click)="submitAnswer(answer)">Submit</button>
           </div>
           </mat-card-content>
       </mat-card>
       </mat-card-content>
   </mat-card>
</div>
```

Integration Points
The components interact with AWS CloudFormation, AWS S3 and AWS CloudFront to perform operations such as resource deployment and data processing.

Security Considerations
Security measures are in place to handle data privacy and user authentication, ensuring that all interactions with AWS services are secure.

Deployment
To deploy this component, ensure AWS CLI is configured with the correct permissions and deploy through the Angular CLI with environment-specific configurations.

Troubleshooting
Template Download Issues: Check if the template was deployed correctly and the stack was created successfully.
Data Display Errors: Ensure API endpoints are correctly configured and accessible.

Conclusion
This documentation provides a comprehensive guide to the well-architected component of the application, covering its functionalities, integration points, and operational guidance.