Automating SR Summarization and KM Article Generation Recipe

Executive Summary

As organizations strive to improve service efficiency and knowledge management, providing a seamless way to summarize service request (SR) activities becomes vital. When a Service Representative closes an SR and there is a need to create a new Knowledge Management (KM) document, generating an accurate and concise summary of the SR activities can significantly streamline the process. In Siebel CRM, this capability can be enhanced by leveraging GenAI to automatically generate apt summaries, which can then be used as attachments or message bodies with the "New Knowledge Doc Create" functionality in Fusion KM. This integration not only showcases the innovative use of Gen AI in Siebel but also highlights productivity improvements for the KM process by automating content generation and ensuring consistent, high-quality documentation.

Introduction

In industries utilizing Siebel CRM to manage service requests (SRs), documenting key insights from resolved SRs into Knowledge Management (KM) systems can be time-consuming and prone to inconsistencies. When organizations require new KM articles based on SR activities, manual efforts often hinder productivity. Introducing an automated system for SR summarization and KM article creation can greatly enhance efficiency and accuracy.

For example, when a Service Representative closes an SR, the system uses the OCI AI framework to generate a concise summary of the SR's key activities. This summary forms the content for a new KM article. The process is streamlined further through Fusion KM REST APIs, which enable seamless publication of the article in the KM system without requiring additional manual intervention.

A practical use case is seen in customer support. When a customer's issue is resolved, the system analyzes the SR details, produces a high-quality summary, and publishes it as a KM article. This document becomes an accessible resource for addressing similar issues in the future, reducing redundancy in operations.

By automating the summarization and publishing workflow, this integration enhances productivity, ensures consistency in KM documentation, and improves the overall service delivery experience.

Challenge: Automating Service Request Summarization and KM Article Creation

Manually summarizing service requests (SRs) and creating Knowledge Management (KM) articles is a time-consuming and error-prone process. Service Representatives must extract key details from SR activities, which often delays article creation and impacts productivity. Automating this workflow with AI-driven summarization and seamless integration with Fusion KM for article publishing addresses these challenges, ensuring efficiency, accuracy, and consistent knowledge documentation.

Workflow Overview

- **Service Representative Closes Service Request**
  The Service Representative updates the Service Request status to "Closed."
- **Option to Create Article**
  A "Create Article" button is enabled in the Service Request UI, allowing the representative to initiate the process.
- **Service Request and Activity Data Consolidation**
  Upon clicking "Create Article," the Service Request Summary and Activity Description are consolidated.
- **AI-Generated Article Creation**
  The consolidated data is sent to a GenAI service, which generates a draft knowledge article using a predefined template.
- **Review Article in Popup**
  The generated article is displayed in a popup for the representative to review and make any necessary edits.
- **Article Confirmation and Submission**
  The representative confirms the article and saves it, completing the process of creating a new Knowledge Management article.
- **Optional Navigation to KM for Further Edits**
  If required, the representative is navigated to the Knowledge Management application for additional modifications.

Key Technologies Involved

Item	Description
Siebel Open UI	Updated Service Request UI to include the "Create Article" button, triggered when SR status is closed.
Business Service	Collects Service Request and Activity details, formats the prompt, and integrates with AI services.
GenAI API (OCI or Siebel)	Generates the Knowledge Article content using the consolidated SR data and a predefined article template.
Custom Physical Renderer	Displays a popup for article review, enabling editable content for the representative.
Wrapper Service (Node.js)	Handles authentication and interacts with GenAI or KM APIs, ensuring seamless integration.
Knowledge Management API	Facilitates the creation and storage of the Knowledge Article within the Fusion Knowledge Management system.

Implementation Strategy

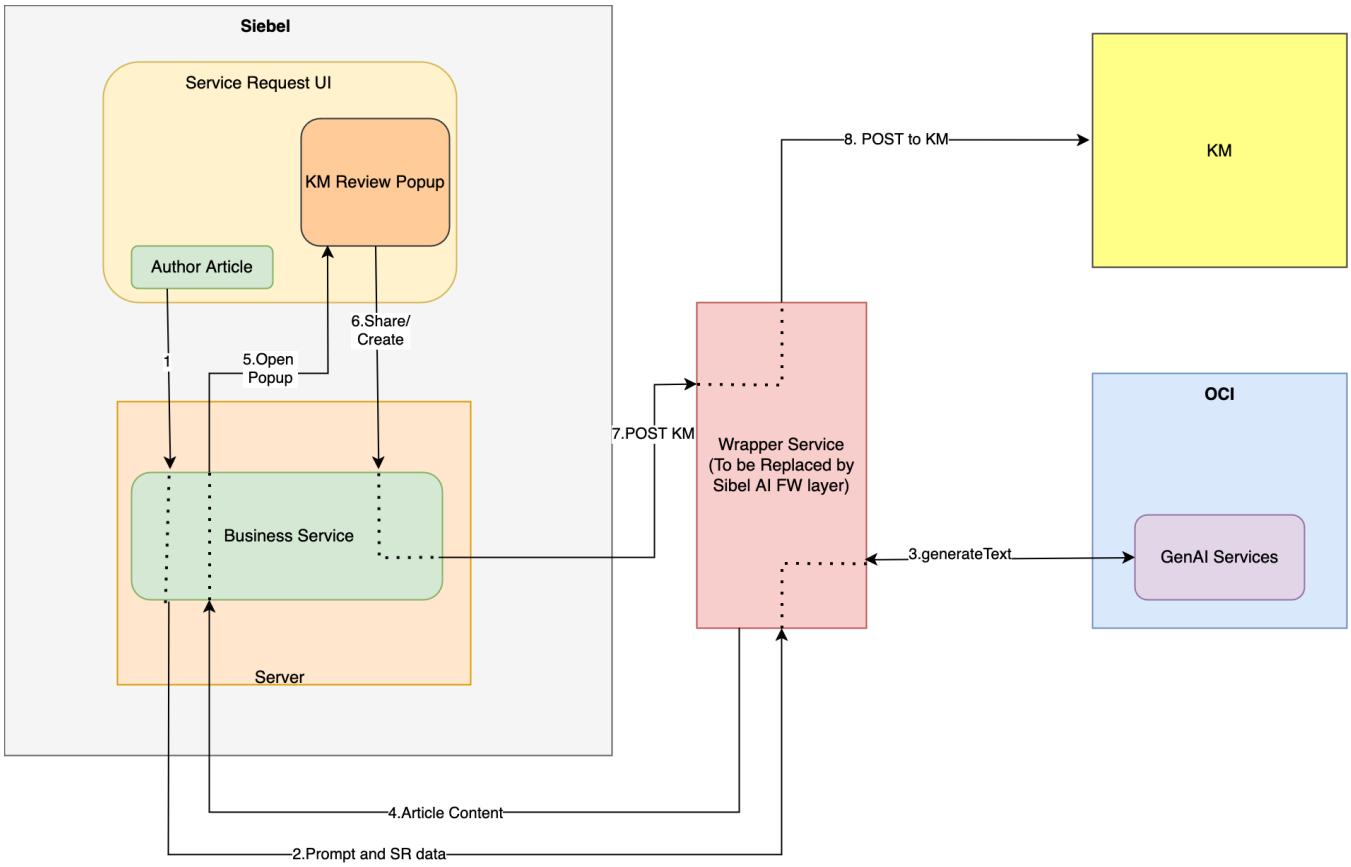
The implementation of this flow involves

- Add a "Create Article" button to the Service Request UI, enabled upon closure.
- Develop a business service to consolidate SR and Activity details for the GenAI prompt.
- Use OCI GenAI or Siebel AI Framework to generate Knowledge Article content or Create a wrapper service for GenAI and KM API integration.
- Implement a popup using a custom renderer to display and edit the article content.
- Automate Knowledge Article creation using the KM POST API.

Architecture Diagram

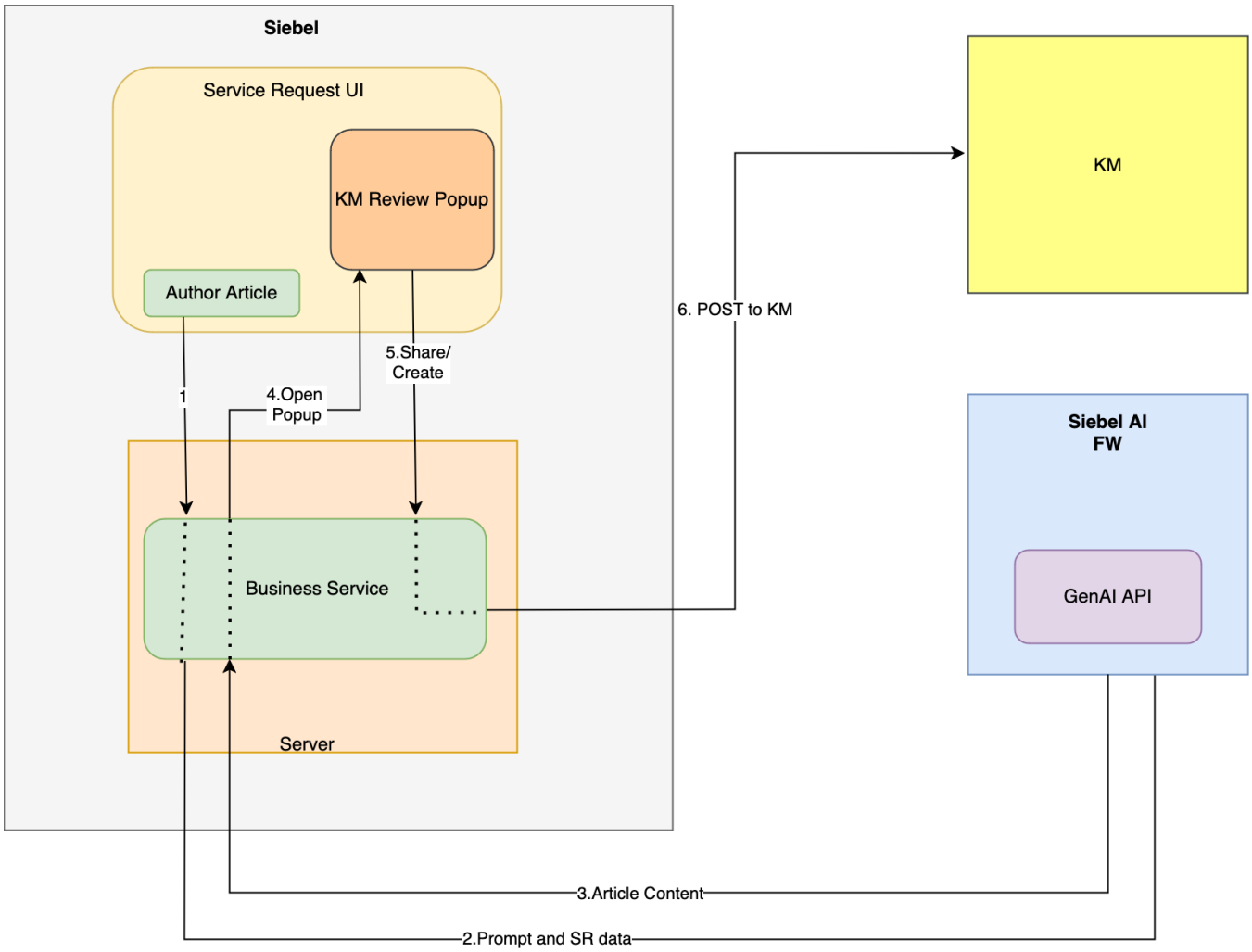
Using wrapper service for OCI GenAI

- **Service Request UI Update**: Added a new button, 'Create Article,' which is enabled when the SR status changes to closed.
- **Business Service Addition**: Added a new business service invoked on button click. This service consolidates SR (Summary field) and Activity details (Description field) , then passes them along with the prompt to an AI service for article generation. GenAI will return the data in specified template for the KM document.
- **Wrapper Service Implementation**: Added a new wrapper service (implemented in Node.js) to call the OCI generate-text API, handling authentication from this layer. This service can be replaced with the Siebel AI Framework service.
- **Popup for Generated Article**: Once the generated article content is available in the response, a new popup will be displayed in the UI with the article content as editable. The popup logic is implemented using a custom physical renderer.
- **KM POST API Invocation**: When the apply button is clicked in the popup, the KM POST API will be invoked in the following flow: Business service > Wrapper service > KM API. The wrapper service is used to overcome the access/authentication issues faced when invoking the service from the business service.



Using Siebel AI Framework

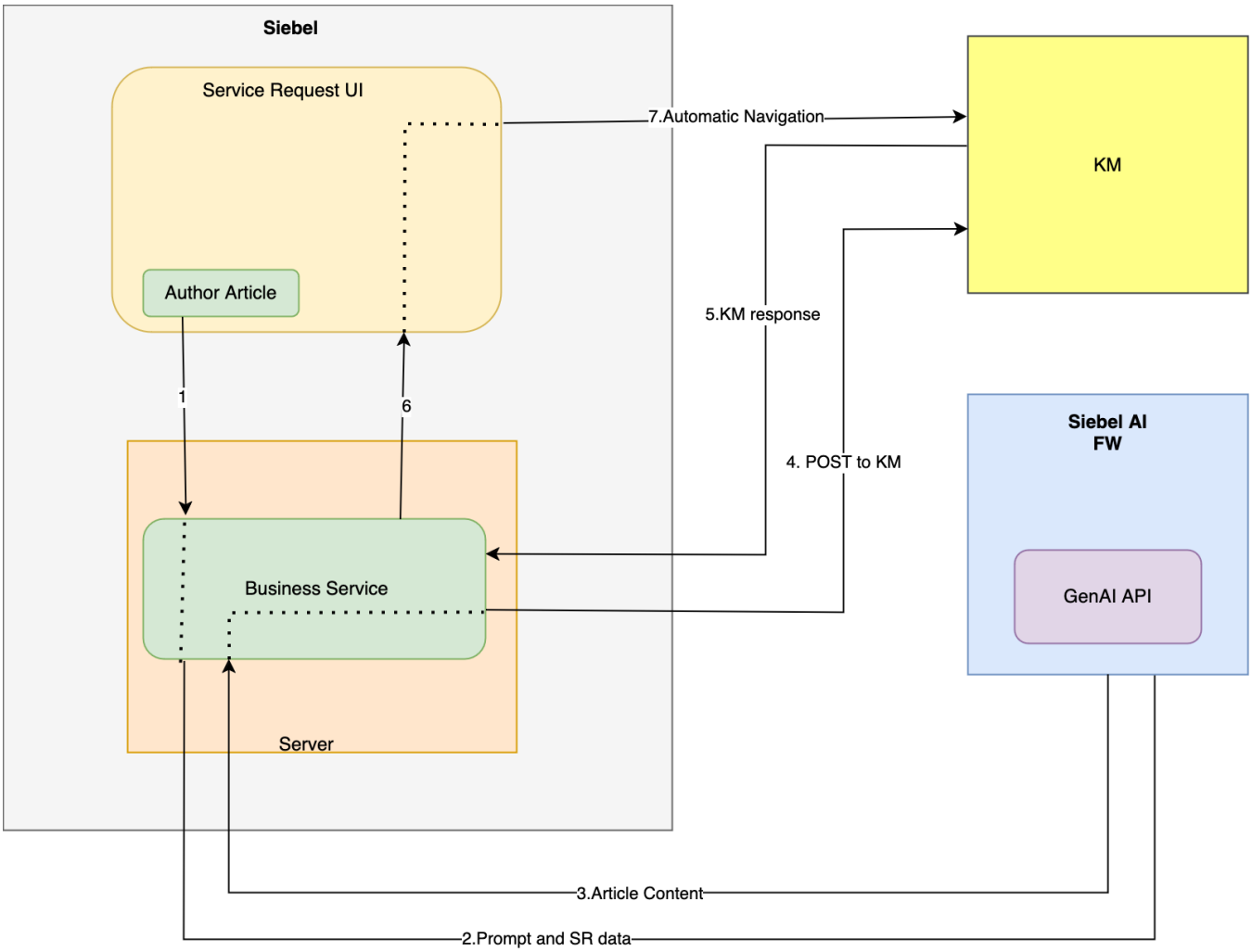
- **Service Request UI Update**: Added a new button, 'Create Article,' which is enabled when the SR status changes to closed.
- **Business Service Addition**: Added a new business service invoked on button click. This service consolidates SR (Summary field) and Activity details (Description field) , then passes them along with the prompt to Siebel AI service for article generation. Siebel GenAI will return the data in specified template for the KM document.
- **Popup for Generated Article**: Once the generated article content is available in the response, a new popup will be displayed in the UI with the article content as editable. The popup logic is implemented using a custom physical renderer.
- **KM POST API Invocation**: When the apply button is clicked in the popup, the KM POST API will be invoked in the following flow: Business service > KM API.



Review in KM Application

steps:

- **Service Request UI Update**: Added a new button, 'Create Article,' which is enabled when the SR status changes to closed.
- **Business Service Addition**: Added a new business service invoked on button click. This service consolidates SR (Summary field) and Activity details (Description field) , then passes them along with the prompt to Siebel AI service for article generation. Siebel GenAI will return the data in specified template for the KM document. The article data will be passed to KM API.
- **Navigation To KM for Article Edit**: Once the generated article id is available in the response, Naviage to the KM article edit screen .Here User can add modifications to the article.



Siebel Customisations

Adding new button in Service Request UI

A new button to be added in Siebel Service Request screen for the summarisation of Service Request and invocation of article creation. This button should be enabled to user only if the status of SR is Closed.

Applet: Service Request Detail Applet

Applet Server Script

Script to enable and disable the button based on the SR status

```
function WebApplet_PreCanInvokeMethod (MethodName, &CanInvoke) {
    if (MethodName == "CreateArticle") {
        var bo = TheApplication().ActiveBusObject();
        var bc = bo.GetBusComp("Service Request");
        var statusField = bc.GetFieldValue("Status");
        TheApplication().Trace("Service Request Status: " + statusField);
        if (statusField == "Closed") {
            CanInvoke = "TRUE";
        } else {
            CanInvoke = "FALSE";
        }
        return(CancelOperation);
    }
    return(ContinueOperation);
}
```

Knowledge Article review option

This section describes how a popup can be added in Service Request screen to display the Knowledge Article generated and the option to edit and send that to KM. Also the option to allow the edit of article directly using KM UI.

Logic

- In the PR, check for the Create Article button using jQuery selector.
- Remove the existing click events and add a new event.
- In the new event, invoke the SR Summarization Business Service and show the response in the UI.
- Add a button to edit the details.
- Once user clicks on Apply, invoke the KM Article creation API and close the popup if success. If error, show the error message in popup itself

PR Code:

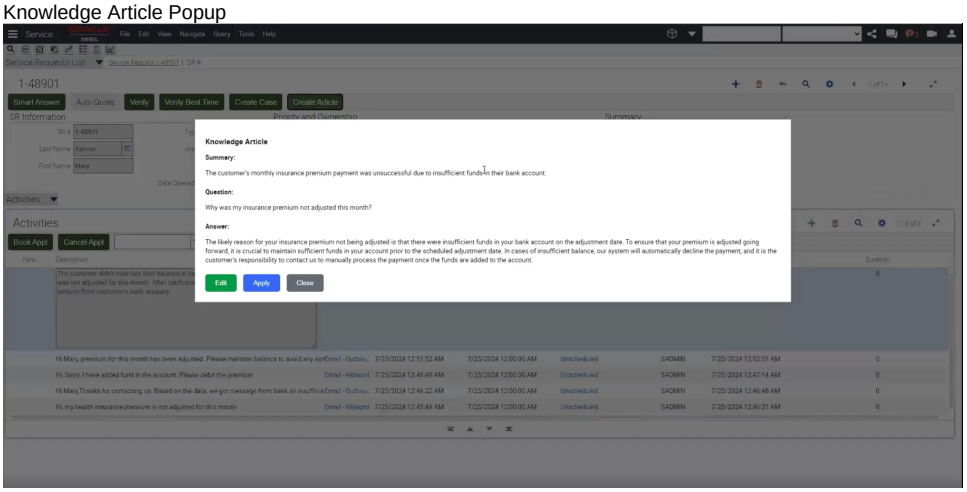
```
if (typeof SiebelAppFacade.ServiceRequestDetailAppletCustomPR === "undefined") {
    SiebelJS.Namespace('SiebelAppFacade.ServiceRequestDetailAppletCustomPR');

    define("siebel/custom/ServiceRequestDetailAppletCustomPR", ["siebel/phyrenderer"], function (PhyRenderer) {
        SiebelAppFacade.ServiceRequestDetailAppletCustomPR = (function () {

            function ServiceRequestDetailAppletCustomPR(pm) {
                console.log('ServiceRequestDetailAppletCustomPR constructor called.');
                SiebelAppFacade.ServiceRequestDetailAppletCustomPR.superclass.constructor.call(this, pm);
            }

            SiebelJS.Extend(ServiceRequestDetailAppletCustomPR, SiebelAppFacade.PhysicalRenderer);

            ServiceRequestDetailAppletCustomPR.prototype.Init = function () {
                console.log('ServiceRequestDetailAppletCustomPR Init method called.');
                SiebelAppFacade.ServiceRequestDetailAppletCustomPR.superclass.Init.call(this);
            };

            ServiceRequestDetailAppletCustomPR.prototype.BindEvents = function () {
                console.log('ServiceRequestDetailAppletCustomPR BindEvents method called.');
                SiebelAppFacade.ServiceRequestDetailAppletCustomPR.superclass.BindEvents.call(this);

                var self = this;
                var buttonControl = $('#s_1_1_12_0_Ctrl');

                if (buttonControl.length) {
                    console.log("Button control found, setting up click event.");
                    buttonControl.off("click").on("click", function () {
                        console.log("Button clicked: Create Article");
                        var inputs = SiebelApp.S_App.NewPropertySet();
                        inputs.SetProperty("SRNumber", this.GetPM().Get("GetRecordSet")[0]['SR Number']);
                        let response = SiebelApp.S_App.GetService("KMService").InvokeMethod("GetSRSummary",
inputs);
                        console.log(response);
                        var resultSet = response.GetChildByType("ResultSet");
                        if (resultSet) {
                            var result = resultSet.GetProperty("Summary");
                            self.showPopup(JSON.parse(result));
                        } else {
                            console.log("No data received from business service.");
                        }
                    }.bind(this));
                } else {
                    console.log("Button control not found.");
                }
            };

            ServiceRequestDetailAppletCustomPR.prototype.showPopup = function (data) {
                console.log(data);
                var popupHtml = `
                    <div id="custom-popup" style="display: none; position: fixed; top: 20%; left: 20%; width:
60%; background: white; border: 1px solid #ccc; padding: 20px; box-shadow: 0 0 10px rgba(0,0,0,0.1); z-index:
1000;">
                        <div id="popup-error" style="color: red; margin-top: 10px; display: none;"></div>
                        <h3>Knowledge Article</h3>
                        <div style="margin-bottom: 20px;">
```

```
                        <h4>Summary</h4>
                        <p id="popup-summary">${data.summary}</p>
                        <input type="text" id="popup-summary-edit" style="display: none; width: 100%;
border: 1px solid #ddd; padding: 5px;" value="${data.summary}" />
                    </div>
                    <div style="margin-bottom: 20px;">
                        <h4>Question</h4>
                        <p id="popup-question">${data.question}</p>
                        <input type="text" id="popup-question-edit" style="display: none; width: 100%;
border: 1px solid #ddd; padding: 5px;" value="${data.question}" />
                    </div>
                    <div style="margin-bottom: 20px;">
                        <h4>Answer</h4>
                        <p id="popup-answer">${data.answer}</p>
                        <textarea id="popup-answer-edit" style="display: none; width: 100%; height: 100px;
border: 1px solid #ddd; padding: 5px;">${data.answer}</textarea>
                    </div>
                    <button id="popup-edit-btn" style="padding: 10px 20px; background-color: #28a745;
color: white; border: none; border-radius: 5px; cursor: pointer;">Edit</button>
                    <button id="popup-apply-btn" style="padding: 10px 20px; background-color: #007bff;
color: white; border: none; border-radius: 5px; cursor: pointer; margin-left: 10px;">Apply</button>
                    <button id="popup-close-btn" style="padding: 10px 20px; background-color: #6c757d;
color: white; border: none; border-radius: 5px; cursor: pointer; margin-left: 10px;">Close</button>
                </div>
                <div id="popup-overlay" style="display: none; position: fixed; top: 0; left: 0; width:
100%; height: 100%; background: rgba(0,0,0,0.5); z-index: 999;"></div>
            `;

            $('body').append(popupHtml);

            $('#custom-popup, #popup-overlay').show();

            $('#popup-edit-btn').on('click', function () {
                $('#popup-summary, #popup-question, #popup-answer').hide();
                $('#popup-summary-edit, #popup-question-edit, #popup-answer-edit').show();
            });

            $('#popup-apply-btn').on('click', function () {
                var updatedSummary = $('#popup-summary-edit').val();
                var updatedQuestion = $('#popup-question-edit').val();
                var updatedAnswer = $('#popup-answer-edit').val();
                console.log("Updated Summary: " + updatedSummary);
                console.log("Updated Question: " + updatedQuestion);
                console.log("Updated Answer: " + updatedAnswer);

                $('#popup-summary-edit, #popup-answer-edit, #popup-answer-edit').hide();
                $('#popup-summary').text(updatedSummary).show();
                $('#popup-question').text(updatedQuestion).show();
                $('#popup-answer').text(updatedAnswer).show();

                var inputs = SiebelApp.S_App.NewPropertySet();
                inputs.SetProperty("Title", "Title");
                inputs.SetProperty("Summary", updatedSummary);
                inputs.SetProperty("Question", updatedQuestion);
                inputs.SetProperty("Answer", updatedAnswer);
                let response = SiebelApp.S_App.GetService("KMService").InvokeMethod("CreateArticle",
inputs);
                console.log(response);
                var resultSet = response.GetChildByType("ResultSet");
                if (resultSet) {
                    var result = resultSet.GetProperty("Status");
                    if(result == "Success"){
                        $('#custom-popup, #popup-overlay').remove();
                    } else {
                        $('#popup-error').text(`Error: ${result}`).show();
                    }
                } else {
                    console.log("No data received from business service.");
                    $('#popup-error').text(`Error: Create Article Service Failed`).show();
                }
            });
```

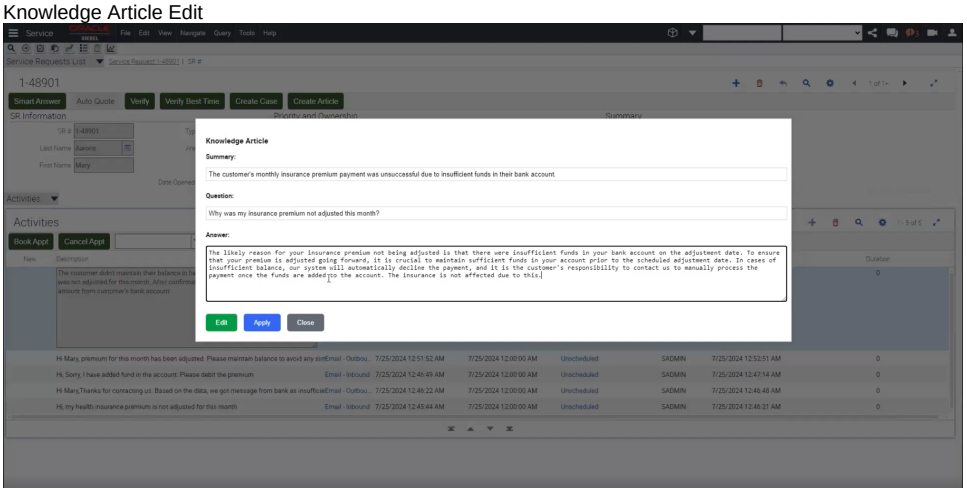
```
            $('#popup-close-btn').on('click', function () {
                $('#custom-popup, #popup-overlay').remove();
            });
        };

        return ServiceRequestDetailAppletCustomPR;

    }());

    return "SiebelAppFacade.ServiceRequestDetailAppletCustomPR";
  });
}
```

Screenshots:

1. Knowledge Article Popup



2. Knowledge Article Edit



Review And Edit Article Using KM Application

Logic

- In the PR, check for the Create Article button using jQuery selector.
- Remove the existing click events and add a new event.
- In the new event, invoke the SR Summarization Business Service.
- Once the response is available invoke the KM Article Creation Service.
- Get the article id from the response and navigate the user to edit page of that record in KM Application.
- User reviews and edit the article leveraging the features of KM UI.

PR Code:

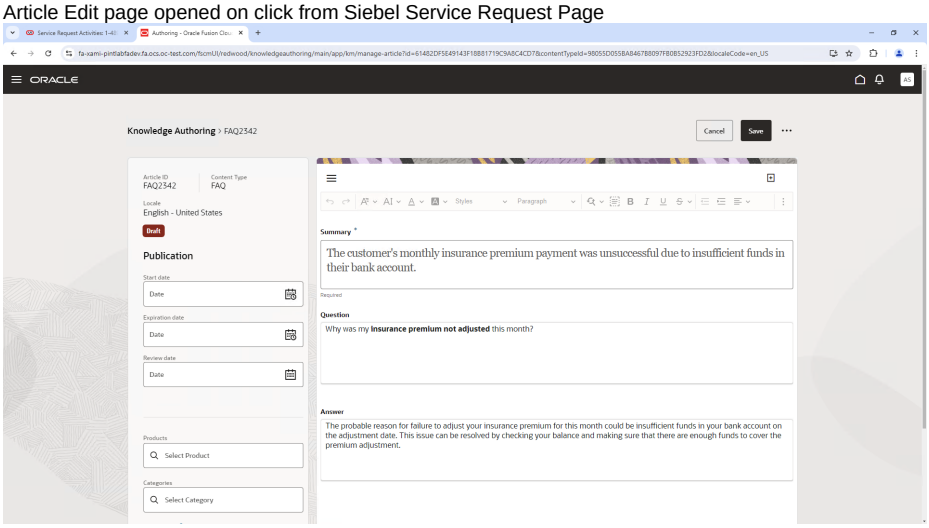
```
if (typeof SiebelAppFacade.ServiceRequestDetailAppletCustomPR === "undefined") {
    SiebelJS.Namespace('SiebelAppFacade.ServiceRequestDetailAppletCustomPR');
```

```
    define("siebel/custom/ServiceRequestDetailAppletCustomPR", ["siebel/phyrenderer"], function (PhyRenderer) {
        SiebelAppFacade.ServiceRequestDetailAppletCustomPR = (function () {

            function ServiceRequestDetailAppletCustomPR(pm) {
                console.log('ServiceRequestDetailAppletCustomPR constructor called.');
                SiebelAppFacade.ServiceRequestDetailAppletCustomPR.superclass.constructor.call(this, pm);
            }

            SiebelJS.Extend(ServiceRequestDetailAppletCustomPR, SiebelAppFacade.PhysicalRenderer);

            ServiceRequestDetailAppletCustomPR.prototype.Init = function () {
                console.log('ServiceRequestDetailAppletCustomPR Init method called.');
                SiebelAppFacade.ServiceRequestDetailAppletCustomPR.superclass.Init.call(this);
            };

            ServiceRequestDetailAppletCustomPR.prototype.BindEvents = function () {
                console.log('ServiceRequestDetailAppletCustomPR BindEvents method called.');
                SiebelAppFacade.ServiceRequestDetailAppletCustomPR.superclass.BindEvents.call(this);

                var self = this;
                var buttonControl = $('#s_1_1_12_0_Ctrl');

                if (buttonControl.length) {
                    console.log("Button control found, setting up click event.");
                    buttonControl.off("click").on("click", function () {
                        console.log("Button clicked: Create Article");
                        SiebelApp.S_App.uiStatus.Busy({mask: true});
                        var inputs = SiebelApp.S_App.NewPropertySet();
                        inputs.SetProperty("SRNumber", this.GetPM().Get("GetRecordSet")[0]['SR Number']);
                        let response = SiebelApp.S_App.GetService("KMService").InvokeMethod("GetSRSummary",
inputs);
                        console.log(response);
                        var resultSet = response.GetChildByType("ResultSet");
                        if (resultSet) {
                            var result = resultSet.GetProperty("Summary");
                            self.navigateToKM(JSON.parse(result));
                        } else {
                            console.log("No data received from business service.");
                        }
                        SiebelApp.S_App.uiStatus.Free();
                    }.bind(this));
                } else {
                    console.log("Button control not found.");
                }
            };

            ServiceRequestDetailAppletCustomPR.prototype.navigateToKM = function (data) {
                var inputs = SiebelApp.S_App.NewPropertySet();
                inputs.SetProperty("Title", "Title");
                inputs.SetProperty("Summary", data.summary);
                inputs.SetProperty("Question", data.question);
                inputs.SetProperty("Answer", data.answer);
                let response = SiebelApp.S_App.GetService("KMService").InvokeMethod("CreateArticle", inputs);
                console.log(response);
                var resultSet = response.GetChildByType("ResultSet");
                if (resultSet) {
                    var result = resultSet.GetProperty("Result");
                    const key = 'versionId":"';
                    const startIndex = result.indexOf(key) + key.length;
                    const endIndex = result.indexOf('"', startIndex);
                    const recordId = result.substring(startIndex, endIndex);
                    console.log(recordId);
                    $('#custom-popup, #popup-overlay').remove();
                    window.open("https://fa-xami-pintlabfadev.fa.ocs.oc-test.com/fscmUI/redwood
/knowledgeauthoring/main/app/km/manage-article?id="+recordId+"
&contentTypeId=98055D055BA8467B8097FB0B52923FD2&localeCode=en_US", '_blank');

                } else {
                    console.log("No data received from business service.");
                    $('#popup-error').text(`Error: Create Article Service Failed`).show();
```

```
                }
            };

            return ServiceRequestDetailAppletCustomPR;

        }());

        return "SiebelAppFacade.ServiceRequestDetailAppletCustomPR";
    });
}
```

Screenshots

1. Article Edit page opened on click from Siebel Service Request Page



Handling SiebelOCIAIWebService Response from PR

The Response from SiebelOCIAIWebService will be in below format. This needs to be parsed to use in UI

SiebelOCIAIWebService Request and Response

```
Request url: https://den00zsb.us.oracle.com:16691/siebel/v1.0/service/SiebelOCIAIWebService/InvokeOCIAI
Request body:
{
    "body": {
        "Context": "GenAI:GenerateText",
        "GenerateText_Param_1": "prompt:Analyze the provided details of a service request and activities
carried out to resolve it. Identify the issue and the solution derived strictly based on that data. Then create
a knowledge article strictly based on the provided article template. Use the following format: Question,
Answer. The response should be strictly parseable to JSON format and should be containing only the template
data.\n\nService request data:\n{\n  \"summary\": \"Not able to login to the computer.\",\n  \"activities\":
[\n    {\"type\": \"email\", \"from\": \"customer care executive\", \"content\": \"Please restart the PC.\"},
\n    {\"type\": \"email\", \"from\": \"customer\", \"content\": \"Not working after restart.\"},\n    {\"
type\": \"call\", \"transcript\": \"agent: Reset the password. customer: How to reset the password? agent: Use
the link shared. customer: OK. agent: Are you able to login with the new password? customer: Yes.\"}\n  ]\n}
\n\nArticle template:\n{\n  \"Question\": \"Short description of the issue phrased like a question. This should
be generic and  usable while searching for similar issues. \",\n  \"Answer\": \"Solution to the issue in a
descriptive manner as a guideline to the customer care agent.Provide a step by step guide if the solution
demands it. \"\n}" ,
        "GenerateText_Param_2": "compartmentId:ocid1.tenancy.oc1..
aaaaaaaad7nju4v2lioxocqnkeqydccehtrswmbsievlnhie2rrqguu5ruxq",
        "GenerateText_Param_3" : "runtimeType:COHERE",
        "GenerateText_Param_4" : "maxTokens:2000",
        "GenerateText_Param_5" : "temperature:1"

    }
}
Response:
{
    "Status": "Success",
    "Output": "CohereLlmInferenceResponse(super=LlmInferenceResponse(super=BmcModel(__explicitlySet__=
[generatedTexts, timeCreated])), generatedTexts=[GeneratedText(super=BmcModel(__explicitlySet__=[id, text])
id=c50ebacc-337d-4341-9de3-8d2f0e82fdb3, text= {\n  \"Question\": \"Why can't I log in to my computer?\",\n  \"
Answer\": \"There could be several reasons for being unable to log in to your computer. One possible reason is
forgetting your password. If this is the case, you can follow these steps to reset your password and regain
access to your computer.\":\n   \"1. Check your email for a link from the customer care executive. This will
allow you to reset your password. If you haven't received this email, contact customer care for further
assistance. \n 2. Once you've obtained the link, follow it to the password reset page. Enter the necessary
details to verify your identity and proceed with the reset.\n3. After completing the password reset, attempt to
log in to your computer with your new password.\"\n} , likelihood=null, finishReason=null,
tokenLikelihoods=null)], timeCreated=Mon Aug 12 05:18:30 UTC 2024, prompt=null)",
    "OPCRequestId": "69741A822DB5491199823A74F557DF40/C2FF7BC16D138B08451D5D4A2729A8C9
/DF7A17E2A01A06C7DDBC6FF9B1C83EB0"
}
```

ServiceRequestDetailAppletCustomPR Changes

```
ServiceRequestDetailAppletCustomPR.prototype.BindEvents = function () {
                console.log('ServiceRequestDetailAppletCustomPR BindEvents method called.');
                SiebelAppFacade.FormControlAttrPR.superclass.BindEvents.call(this);

                var self = this;
                var buttonControl = $('#s_1_1_13_0_Ctrl');

                if (buttonControl.length) {
                    console.log("Button control found, setting up click event.");
                    buttonControl.off("click").on("click", function () {
                        console.log("Button clicked: Create Article.");

                        SiebelApp.S_App.uiStatus.Busy({mask: true});
                        var inputs = SiebelApp.S_App.NewPropertySet();
                        inputs.SetProperty("SRNumber", this.GetPM().Get("GetRecordSet")[0]['SR Number']);
                        let response = SiebelApp.S_App.GetService("KMService").InvokeMethod("GetSRSummary",
inputs);
                        console.log(response);
                        var resultSet = response.GetChildByType("ResultSet");
                        if (resultSet) {
                            var result = resultSet.GetProperty("Summary");

                            try {
                                // Parse the response string to a JavaScript object
                                const responseObject = JSON.parse(result);

                                // Safely get the 'Output' value
                                const output = responseObject.Output;

                                // Extract the 'text' content from the Output
                                const textMatch = output.match(/text\s*=\s*({[\s\S]*?})\s*,\s*likelihood=null/);

                                if (textMatch && textMatch[1]) {
                                    // Replace escape sequences for double quotes
                                    const jsonString = textMatch[1].replace(/\\n/g, '\n').replace(/\\"/g, '"');

                                    // Parse the JSON string into an object
                                    try {
                                        const parsedJson = JSON.parse(jsonString);
                                        console.log("Extracted JSON Object:", parsedJson);

                                        self.showPopup(parsedJson);

                                    } catch (error) {
                                        console.error("Error parsing JSON:", error);
                                    }
                                } else {
                                    console.error("No JSON text found in the Output.");
                                }
                            } catch (error) {
                                console.error("Error parsing the response string:", error);
                            }
                        } else {
                            console.log("No data received from business service.");
                        }
                        SiebelApp.S_App.uiStatus.Free();
                    }.bind(this));
                } else {
                    console.log("Button control not found.");
                }
            };
```

Siebel Business Service

Create New Business Service KMService in Siebel Web Tools

Service Name: KMService

Business Service Methods

GetSRSummary

Method to summarize SR.

Arguments: Input - SRNumber, Output - Summary

Process: Based on the SRNumber, fetch the SR and activity details. Create the prompt for SR summarization and invoke the OCI Gen AI service to summarize.

CreateArticle

Method to create article in Knowledge Management

Arguments: Input - Title, Summary, Question, Answer. Output - Result, Status

Process: Invoke the KM Article creation API with the input details. If the creation is successful, return the status as Success otherwise return the error message.

Using wrapper services to invoke KM and Gen AI APIs

```
function Service_PreInvokeMethod (MethodName, Inputs, Outputs)
{
    if (MethodName == "GetSRSummary")
    {
        var srNumber = Inputs.GetProperty("SRNumber");
        if (srNumber != "")
        {
            GetActivities(srNumber, Outputs);
            return (CancelOperation);
        }
    } else if(MethodName == "CreateArticle")
    {
        CreateArticle(Inputs.GetProperty("Title"), Inputs.GetProperty("Summary"),Inputs.GetProperty
("Question"), Inputs.GetProperty("Answer"), Outputs);
        return (CancelOperation);
    }
    return (ContinueOperation);
}

function CreateArticle(title, summary, question, answer, Outputs)
{
    try {
        var httpService = TheApplication().GetService("EAI HTTP Transport");
        var httpInputs = TheApplication().NewPropertySet();
        var httpOutputs = TheApplication().NewPropertySet();

        var url = "https://127.0.0.1:3000/create-content";
        var requestBody = '{"locale":{ "recordId":"en_US" },"contentType":{ "recordId":"
98055D055BA8467B8097FB0B52923FD2", "referenceKey":"FAQ", "name":"FAQ" },"xml":"<FAQ><SUMMARY><![CDATA
['+summary+']]></SUMMARY><QUESTION><![CDATA['+question+']]></QUESTION><ANSWER><![CDATA['+answer+']]></ANSWER><
/FAQ>","isForEdit":true}';


        httpInputs.SetProperty("HTTPRequestURLTemplate", url);
        httpInputs.SetProperty("HTTPRequestMethod", "POST");
        httpInputs.SetProperty("HTTPContentType", "application/json");
        httpInputs.SetProperty("CharSetConversion", "UTF-8");
        httpInputs.SetProperty("HTTPAccept", "*/*");
        httpInputs.SetValue(requestBody);
        httpService.InvokeMethod("SendReceive", httpInputs, httpOutputs);
        Outputs.SetProperty("Status", "Success");

    } catch (e) {
        Outputs.SetProperty("Status", "Failure: "+e.toString());
        TheApplication().RaiseErrorText("Error occurred in CreateArticle: " + e.toString());
    }
}
```

```
function GetActivities(srNumber, Outputs)
{
    try
    {
        var boSR = TheApplication().GetBusObject("Service Request");
        var bcSR = boSR.GetBusComp("Service Request");
        var bcActivity = boSR.GetBusComp("Action");

        // Query for the SR by SR Number
        bcSR.ClearToQuery();
        bcSR.SetViewMode(AllView);
        bcSR.SetSearchSpec("SR Number", srNumber);
        bcSR.ExecuteQuery(ForwardOnly);

        if (bcSR.FirstRecord())
        {
            // Get the Id of the SR
            var srId = bcSR.GetFieldValue("Id");

            // Query for Activities related to the SR
            bcActivity.ClearToQuery();
            bcActivity.SetViewMode(AllView);
            bcActivity.SetSearchSpec("Activity SR Id", srId);
            bcActivity.SetSortSpec("Id(ASCENDING)");
            bcActivity.ExecuteQuery(ForwardOnly);

            var activityDetails = "";
            var isRecord = bcActivity.FirstRecord();
            while (isRecord)
            {
                var type = bcActivity.GetFieldValue("Type");
                var description = bcActivity.GetFieldValue("Description");
                activityDetails += type + ":" + description + ", ";
                isRecord = bcActivity.NextRecord();
            }


            try {
                var httpService = TheApplication().GetService("EAI HTTP Transport");
                var httpInputs = TheApplication().NewPropertySet();
                var httpOutputs = TheApplication().NewPropertySet();

                var url = "https://127.0.0.1:3000/generate-text";

                var requestBody = '{"runtimeType":"COHERE", "prompt": "Analyze the below provided context of a
service request and activities carried out to resolve it.Identify the issue and the solution derived strictly
based on that data. The output should be strictly in JSON format and should be containing only the keys title,
summary,question, answer. context:[[Service request data:{summary: .,activities: '+activityDetails+'}]].
Expected output format :{title: <Short title of issue.>, summary: <Short summary of issue.>,question: <Short
description of the issue phrased like a question. This should be generic and  usable while searching for
similar issues.>,answer: <Solution to the issue in a descriptive manner.> }"}';
                httpInputs.SetProperty("HTTPRequestURLTemplate", url);
                httpInputs.SetProperty("HTTPRequestMethod", "POST");
                httpInputs.SetProperty("HTTPContentType", "application/json");
                httpInputs.SetProperty("CharSetConversion", "UTF-8");
                httpInputs.SetProperty("HTTPAccept", "*/*");
                httpInputs.SetValue(requestBody);
                httpService.InvokeMethod("SendReceive", httpInputs, httpOutputs);

                try {
                    var oTransService = TheApplication().GetService("Transcode Service");
                    var oTransOutputs = TheApplication().NewPropertySet();
                    httpOutputs.SetProperty("ConversionMode", "EncodingToString");
                    httpOutputs.SetProperty("SourceEncoding", "CP1252");
                    oTransService.InvokeMethod("Convert", httpOutputs, oTransOutputs);
                    var sResponse = oTransOutputs.GetValue();
                    TheApplication().Trace("HTTP Response sResponse: " + sResponse);
                    Outputs.SetProperty("Summary", sResponse);
                } catch (e2) {
                    Outputs.SetProperty("Summary", "Error2");
```

```
                    TheApplication().Trace("Error in Transcode Service or JSON parsing: " + e2);
                }
            } catch (e) {
                Outputs.SetProperty("Summary", e.toString());
                TheApplication().RaiseErrorText("Error occurred in GetUserEngagementData: " + e.toString());
            }
            //Outputs.SetProperty("Summary", activityDescriptions);
            TheApplication().Trace("Activity Descriptions: \n" + activityDescriptions);
        }
        else
        {
            Outputs.SetProperty("ActivityDescriptions", "No Service Request found with SR Number: " + srNumber);
        }
    }
    catch (e)
    {
        TheApplication().Trace("Error in GetActivities: " + e.toString());
        Outputs.SetProperty("ActivityDescriptions", "Error in GetActivities: " + e.toString());
    }
}
```

Directly invoking KM APIs and Siebel AI Framework

```
function Service_PreInvokeMethod (MethodName, Inputs, Outputs)
{
    if (MethodName == "GetSRSummary")
    {
        var srNumber = Inputs.GetProperty("SRNumber");
        if (srNumber != "")
        {
            GetActivities(srNumber, Outputs);
            return (CancelOperation);
        }
    } else if(MethodName == "CreateArticle")
    {
        CreateArticle(Inputs.GetProperty("Title"), Inputs.GetProperty("Summary"),Inputs.GetProperty
("Question"), Inputs.GetProperty("Answer"), Outputs);
        return (CancelOperation);
    }
    return (ContinueOperation);
}

function CreateArticle(title, summary, question, answer, Outputs)
{
    try {
        var httpService = TheApplication().GetService("EAI HTTP Transport");
        var httpInputs = TheApplication().NewPropertySet();
        var httpOutputs = TheApplication().NewPropertySet();

        var url = "https://fa-xami-pintlabfadev.fa.ocs.oc-test.com/km/api/latest/content";
        var requestBody = '{"locale":{ "recordId":"en_US" },"contentType":{ "recordId":"
98055D055BA8467B8097FB0B52923FD2", "referenceKey":"FAQ", "name":"FAQ" },"xml":"<FAQ><SUMMARY><![CDATA
['+summary+']]></SUMMARY><QUESTION><![CDATA['+question+']]></QUESTION><ANSWER><![CDATA['+answer+']]></ANSWER><
/FAQ>","isForEdit":true}';


        httpInputs.SetProperty("HTTPRequestURLTemplate", url);
        httpInputs.SetProperty("HTTPRequestMethod", "POST");
        httpInputs.SetProperty("HTTPContentType", "application/json");
        httpInputs.SetProperty("CharSetConversion", "UTF-8");
        httpInputs.SetProperty("HTTPAccept", "*/*");
        httpInputs.SetProperty("HDR.Authorization", "Basic ******************************");
        httpInputs.SetValue(requestBody);
        httpService.InvokeMethod("SendReceive", httpInputs, httpOutputs);
        Outputs.SetProperty("Status", "Success");
        Outputs.SetProperty("Result", httpOutputs);

    } catch (e) {
```

```
        Outputs.SetProperty("Status", "Failure: "+e.toString());
        TheApplication().RaiseErrorText("Error occurred in CreateArticle: " + e.toString());
    }
}

function GetActivities(srNumber, Outputs)
{
    try
    {
        var boSR = TheApplication().GetBusObject("Service Request");
        var bcSR = boSR.GetBusComp("Service Request");
        var bcActivity = boSR.GetBusComp("Action");

        // Query for the SR by SR Number
        bcSR.ClearToQuery();
        bcSR.SetViewMode(AllView);
        bcSR.SetSearchSpec("SR Number", srNumber);
        bcSR.ExecuteQuery(ForwardOnly);

        if (bcSR.FirstRecord())
        {
            // Get the Id of the SR
            var srId = bcSR.GetFieldValue("Id");

            // Query for Activities related to the SR
            bcActivity.ClearToQuery();
            bcActivity.SetViewMode(AllView);
            bcActivity.SetSearchSpec("Activity SR Id", srId);
            bcActivity.SetSortSpec("Id(ASCENDING)");
            bcActivity.ExecuteQuery(ForwardOnly);

            var activityDetails = "";
            var isRecord = bcActivity.FirstRecord();
            while (isRecord)
            {
                var type = bcActivity.GetFieldValue("Type");
                var description = bcActivity.GetFieldValue("Description");
                activityDetails += type + ":" + description + ", ";
                isRecord = bcActivity.NextRecord();
            }


            try {
                var httpService = TheApplication().GetService("EAI HTTP Transport");
                var httpInputs = TheApplication().NewPropertySet();
                var httpOutputs = TheApplication().NewPropertySet();

                var url = "https://den00zsb.us.oracle.com:16691/siebel/v1.0/service/SiebelOCIAIWebService
/InvokeOCIAI";

                var prompt = 'Analyze the below provided context of a service request and activities carried
out to resolve it.Identify the issue and the solution derived strictly based on that data. The output should be
strictly in JSON format and should be containing only the keys title, summary,question, answer. context:
[[Service request data:{summary: .,activities: '+activityDetails+'}]]. Expected output format :{title: <Short
title of issue.>, summary: <Short summary of issue.>,question: <Short description of the issue phrased like a
question. This should be generic and  usable while searching for similar issues.>,answer: <Solution to the
issue in a descriptive manner.> }';
                var requestBody = '{"body": {"Context": "GenAI:GenerateText",  "GenerateText_Param_1":
"prompt:'+prompt+'" ,"GenerateText_Param_2": "compartmentId:ocid1.tenancy.oc1..
aaaaaaaad7nju4v2lioxocqnkeqydccehtrswmbsievlnhie2rrqguu5ruxq","GenerateText_Param_3" : "runtimeType:COHERE","
GenerateText_Param_4" : "maxTokens:2000","GenerateText_Param_5" : "temperature:1"}}';


                httpInputs.SetProperty("HTTPRequestURLTemplate", url);
                httpInputs.SetProperty("HTTPRequestMethod", "POST");
                httpInputs.SetProperty("HTTPContentType", "application/json");
                httpInputs.SetProperty("CharSetConversion", "UTF-8");
                httpInputs.SetProperty("HTTPAccept", "*/*");
                httpInputs.SetProperty("HDR.Authorization", "Basic ************************");
                httpInputs.SetValue(requestBody);
                httpService.InvokeMethod("SendReceive", httpInputs, httpOutputs);
```

```
                try {
                    var oTransService = TheApplication().GetService("Transcode Service");
                    var oTransOutputs = TheApplication().NewPropertySet();
                    httpOutputs.SetProperty("ConversionMode", "EncodingToString");
                    httpOutputs.SetProperty("SourceEncoding", "CP1252");
                    oTransService.InvokeMethod("Convert", httpOutputs, oTransOutputs);
                    var sResponse = oTransOutputs.GetValue();
                    TheApplication().Trace("HTTP Response sResponse: " + sResponse);
                    Outputs.SetProperty("Summary", sResponse);
                } catch (e2) {
                    Outputs.SetProperty("Summary", "Error2");
                    TheApplication().Trace("Error in Transcode Service or JSON parsing: " + e2);
                }
            } catch (e) {
                Outputs.SetProperty("Summary", e.toString());
                TheApplication().RaiseErrorText("Error occurred in GetUserEngagementData: " + e.toString());
            }
            //Outputs.SetProperty("Summary", activityDescriptions);
            TheApplication().Trace("Activity Descriptions: \n" + activityDescriptions);
        }
        else
        {
            Outputs.SetProperty("ActivityDescriptions", "No Service Request found with SR Number: " + srNumber);
        }
    }
    catch (e)
    {
        TheApplication().Trace("Error in GetActivities: " + e.toString());
        Outputs.SetProperty("ActivityDescriptions", "Error in GetActivities: " + e.toString());
    }
}
```

Using Knowledge Management APIs to create articles

Using REST API

https://fa-xami-pintlabfadev.fa.ocs.oc-test.com/km/api/latest/content

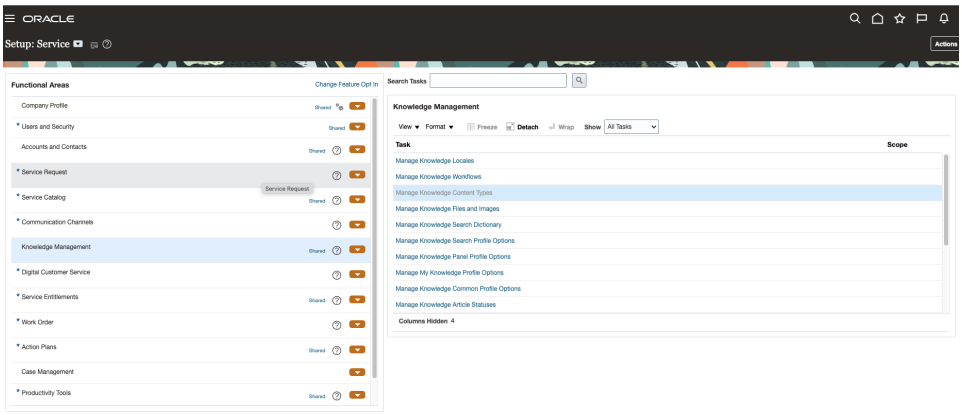
Authentication : Basic

Payload sample:

```
{
    "locale":{ "recordId":"en_US" },
    "contentType":{ "recordId":"98055D055BA8467B8097FB0B52923FD2", "referenceKey":"FAQ", "name":"FAQ" },
    "xml":"<FAQ><SUMMARY><![CDATA[High electric bill version 2]]></SUMMARY><QUESTION><![CDATA[How to reduce
electricity bill at home]]></QUESTION><ANSWER><![CDATA[use energy rated devices.]]></ANSWER></FAQ>",
    "isForEdit":true
}
```

The contentType decides the template of the KM article. Here the FAQ template is used.

Content Type Creation



Invoking OCI GEN API from Node Project

how to create node server and expose an API as a wrapper to OCI Gen-AI generate-text service. The Authentication and API call is handled in wrapper service using oci-sdk.

Steps

- Create node project and install dependencies
- Copy the below mentioned files to the project folder
- Configure ociConfiguration.js with OCI related details
- Copy the key files to project folder (key.pem)
- Run main.js (node main.js).

API

Request

url:https://localhost:3000/generate-text

payload:

```
{
    "runtimeType":"COHERE",
  "prompt": "You are a bot who can author knowledge articles for service agents which will help them to resolve service  by referring ity. Analyze the provided details of a service request and activities carried out to resolve it. Identify the issue and the solution derived strictly based on that data. Then create a knowledge article strictly based on the provided article template. Use the following format: Question, Answer. The response should be parseable to JSON format and the response should be containing only the template attributes and values for it as proper string variables.\n\nService request data:\n{\n  \"summary\": \"Not able to login to the computer.\",\n  \"activities\": [\n    {\"type\": \"email\", \"from\": \"customer care executive\", \"content\": \"Please restart the PC.\"},\n    {\"type\": \"email\", \"from\": \"customer\", \"content\": \"Not working after restart.\"},\n    {\"type\": \"call\", \"transcript\": \"agent: Reset the password. customer: How to reset the password? agent: Use the link shared. customer: OK. agent: Are you able to login with the new password? customer: Yes.\"}\n  ]\n}\n\nArticle template:\n{\n  \"Question\": \"Short description of the issue phrased like a question. This should be generic and  usable while searching for similar issues. \",\n  \"Answer\": \"Solution to the issue in a descriptive manner as a guideline to the customer care agent.Provide a step by step guide if the solution demands it. \"\n}"
}
```

Response:

```
{
"Question": "Why can't I log in to my computer?",
"Answer": "There could be several reasons for your inability to log in to your computer. One possible reason is forgetting your password. If you have forgotten your password, you can follow these steps to reset it and regain access
to your computer.":
"1. Reach out to customer care and ask them for a password reset link.",
"2. Check your email for a password reset email from the customer care executive.",
"3. Click on the link to reset your password.",
"4. Create a new password for your account and make sure to keep it safe and memorable.",
"5. Try logging in with your new password."
}
```

Code

- **main.js**

```
const genai = require("oci-generativeaiinference");
const common = require("oci-common");
const ociConfig = require('./ociConfigurations');
const fs = require("fs");


const {  user, fingerprint, tenancy, region, key_file } = ociConfig;


const privateKey = fs.readFileSync(key_file, { encoding: "utf-8" });


let ociRegion = {}; // Change to appropriate region

if (region === "us-chicago-1") {
  //ociRegion = common.Region.US_ASHBURN_1;
  ociRegion = common.Region.US_CHICAGO_1;

}  else if (region === "us-ashburn-1") {
  ociRegion = common.Region.US_ASHBURN_1;
}else if (region === "us-phoenix-1") {
  ociRegion = common.Region.US_PHOENIX_1;
}

const provider = new common.SimpleAuthenticationDetailsProvider(
    tenancy, user, fingerprint, privateKey, '', ociRegion
);

// Set up the Generative AI client
const endpoint = 'https://generativeai.aiservice.us-chicago-1.oci.oraclecloud.com'; // Replace with
your service endpoint if different

const compartmentId = "ocid1.tenancy.oc1..
aaaaaaaad7nju4v2lioxocqnkeqydccehtrswmbsievlnhie2rrqguu5ruxq"

const generativeAiClient = new genai.GenerativeAiInferenceClient({
  authenticationDetailsProvider: provider,
  endpoint: endpoint,
 // retryConfiguration: new genai.retry.NoneRetryConfiguration(), // No retries
  //requestSigner: new genai.Common.DefaultRequestSigner(provider)
});

const modelId = 'cohere.command';
const servingMode = {
  modelId,
  servingType: genai.models.OnDemandServingMode.servingType,
};
// Define the request details
const generateTextDetails = {
  inferenceRequest: {
    prompt: "write a poem",
    maxTokens: 1000,
    numGenerations: 1,
    returnLikelihoods: "GENERATION",
    isStream: false,
    runtimeType: "COHERE",
  },
  servingMode: {
    servingType: "ON_DEMAND",
    modelId: "cohere.command"
  },  // : new genai.GenerativeAi.Models.OnDemandServingMode({modelId: "cohere.command"}), //
Replace with your model ID
  compartmentId: compartmentId,

};

// Call the generateText API
generativeAiClient.generateText({ generateTextDetails }).then(response => {
  console.log("Generated Text:", response);
  const generatedTexts = response.generateTextResult.inferenceResponse.generatedTexts;
  generatedTexts.forEach((textObj, index) => {
    console.log(`Generated Text ${index + 1}: ${textObj.text}`);
  });
}).catch(error => {
  console.error("Error generating text:", error);
});
```

- **ociConfigurations.js**

```
const ociConfig =
    {
        "user": "ocid1.user.oc1..aaaaaaaaslawrnuafvbftc3ew7k7zhm5w757bxplxqra4kwhnjqpln52hlta",
        "fingerprint": "c7:96:fa:50:1f:9a:91:97:94:da:ce:3e:7a:4f:54:81",
        "tenancy": "ocid1.tenancy.oc1..xxxx",
        "region": "us-chicago-1",
        "key_file": "key.pem"
      }


module.exports = ociConfig;
```

- package.json
- **package.json**

```
{
  "name": "test",
  "version": "1.0.0",
  "description": "",
  "main": "main.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node main.js"
  },
  "dependencies": {
    "node-os-walk": "^1.0.2",
    "oci-sdk": "^2.69.0"
  },
  "author": "rahul",
  "license": ""
}
```

- **Invoking different models**:

```
const genai = require("oci-generativeaiinference");
const common = require("oci-common");
const fs = require("fs");
const ociConfig = require('./ociConfigurations');

const { user, fingerprint, tenancy, region, key_file } = ociConfig;

const privateKey = fs.readFileSync(key_file, { encoding: "utf-8" });

let ociRegion;

if (region === "us-chicago-1") {
    ociRegion = common.Region.US_CHICAGO_1;

}  else if (region === "us-ashburn-1") {
    ociRegion = common.Region.US_ASHBURN_1;
}else if (region === "us-phoenix-1") {
    ociRegion = common.Region.US_PHOENIX_1;
}

const provider = new common.SimpleAuthenticationDetailsProvider(
  tenancy, user, fingerprint, privateKey, '', ociRegion
);

const endpoint = 'https://generativeai.aiservice.us-chicago-1.oci.oraclecloud.com';

const compartmentId = "ocid1.tenancy.oc1..
aaaaaaaad7nju4v2lioxocqnkeqydccehtrswmbsievlnhie2rrqguu5ruxq";

const generativeAiClient = new genai.GenerativeAiInferenceClient({
  authenticationDetailsProvider: provider,
  endpoint: endpoint,
});

const defModelId = 'cohere.command';//meta.llama-2-70b-chat
const defRuntimeType= 'COHERE';//LLAMA

async function generateText(prompt, runtimeType, modelId) {

  console.log('using runtimeType ', runtimeType, modelId)
```

```
var inferenceRequest;
  if(runtimeType=="COHERE"){
    if(!modelId){
      modelId = "cohere.command";

    }
    inferenceRequest={
      prompt: prompt,
      maxTokens: 1000,
      numGenerations: 1,
      returnLikelihoods: "GENERATION",
      isStream: false,
      "runtimeType": "COHERE",
    }
  }else if (runtimeType=="LLAMA"){
    if(!modelId){
    modelId = "meta.llama-2-70b-chat";
    }

    inferenceRequest= {
      prompt: prompt,
      "maxTokens": 1000,
      "isStream": false,
      "numGenerations": 1,
      "runtimeType": "LLAMA",
      "frequencyPenalty": 0,
      "presencePenalty": 0,
      "temperature": 1,
      "topP": 0.75,
      "topK": -1,
      "stop": []
    }
  }
  const generateTextDetails = {

    inferenceRequest,
    servingMode: {
      servingType: "ON_DEMAND",
      modelId
    },
    compartmentId: compartmentId,
  };
  /*const generateTextDetails= {
    "compartmentId": compartmentId,
    "servingMode": {
      "modelId": "meta.llama-2-70b-chat",
      "servingType": "ON_DEMAND"
    },
    "inferenceRequest": {
      "prompt": "tell me something about earth",
      "maxTokens": 300,
      "isStream": false,
      "numGenerations": 1,
      "runtimeType": "LLAMA",
      "frequencyPenalty": 0,
      "presencePenalty": 0,
      "temperature": 1,
      "topP": 0.75,
      "topK": -1,
      "stop": []
    }
  }*/

  console.log('REQUEST:',generateTextDetails)

  try {
    const response = await generativeAiClient.generateText({ generateTextDetails });
    console.log('response:',response);
    if(runtimeType=="LLAMA"){
      const choices = response.generateTextResult.inferenceResponse.choices;

      // If choices array is not empty, parse the first choice text
      if (choices && choices.length > 0) {
        const firstChoiceText = choices[0].text; // Adjust this based on the actual
structure of each choice
        console.log(firstChoiceText);
```

```
        choices.forEach((choice, index) => {
          try {
            const parsedResponse = choice.text;
            console.log(`Parsed Response for Choice ${index + 1}:`, parsedResponse);
          } catch (parseError) {
            console.log(`Error parsing choice ${index + 1} text:`, parseError);
          }
        });


        return firstChoiceText;
      }
    }else{

      return response.generateTextResult.inferenceResponse.generatedTexts[0]?response.
generateTextResult.inferenceResponse.generatedTexts[0].text: response.generateTextResult.
inferenceResponse;
    }
    return "Cannot process response, check runtimeType"
  } catch (error) {
      throw new Error("Error generating text: " + error.message);
    }
  }

  module.exports = { generateText };
```

Functionality

SR Summarization Demo

Editing KM Demo

Benefits

- **Improved Efficiency**: Automating the SR summarization process reduces manual effort and speeds up the creation of Knowledge Management articles.
- **Consistency in Documentation**: The use of AI ensures that SR summaries are consistently formatted and high-quality, reducing human error.
- **Real-Time Access to updated Knowledge Base**: Automatically generated articles ensure that new knowledge articles are available in KM nstantly, enhancing support and decision-making.
- **Seamless Integration**: The system integrates well with Siebel CRM and Fusion KM, streamlining the workflow from SR closure to article publication.
- **Enhanced User Experience**: Users can edit the generated summary, giving them control over the final content while maintaining efficiency.

Conclusion

By integrating Generative AI into the SR summarization and Knowledge Management (KM) process, organizations can significantly enhance efficiency and consistency in documentation. Automating the creation of knowledge articles reduces manual effort, ensures high-quality content, and accelerates the publishing workflow. However, a limitation of this feature is that the AI-generated summary may miss crucial details, making human review essential for accuracy. This approach streamlines knowledge transfer, improves service resolution times, and enables seamless integration with Fusion KM, ultimately enhancing productivity and the overall customer support experience.