**Send Bulk Emails**

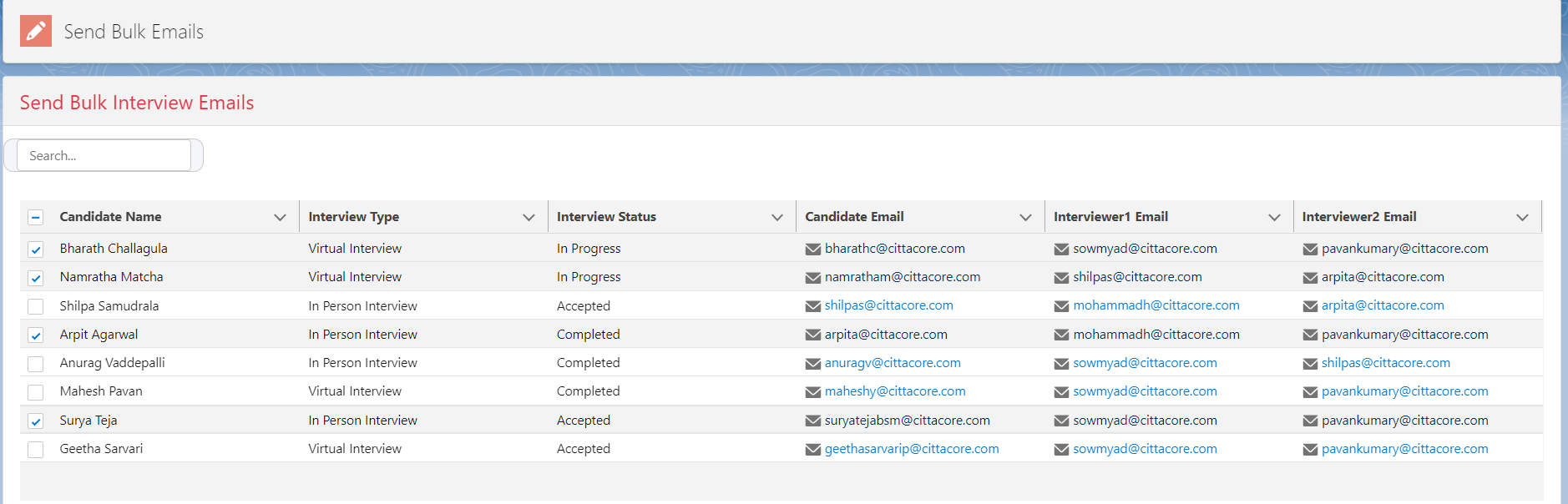
**Overview**

This Lightning Web Component displays a list of interview records and allows the user to select multiple records and send bulk emails. It consists of a search input field, a datatable, and a Send Bulk Emails button.

**Requirements**

This component has the following requirements:

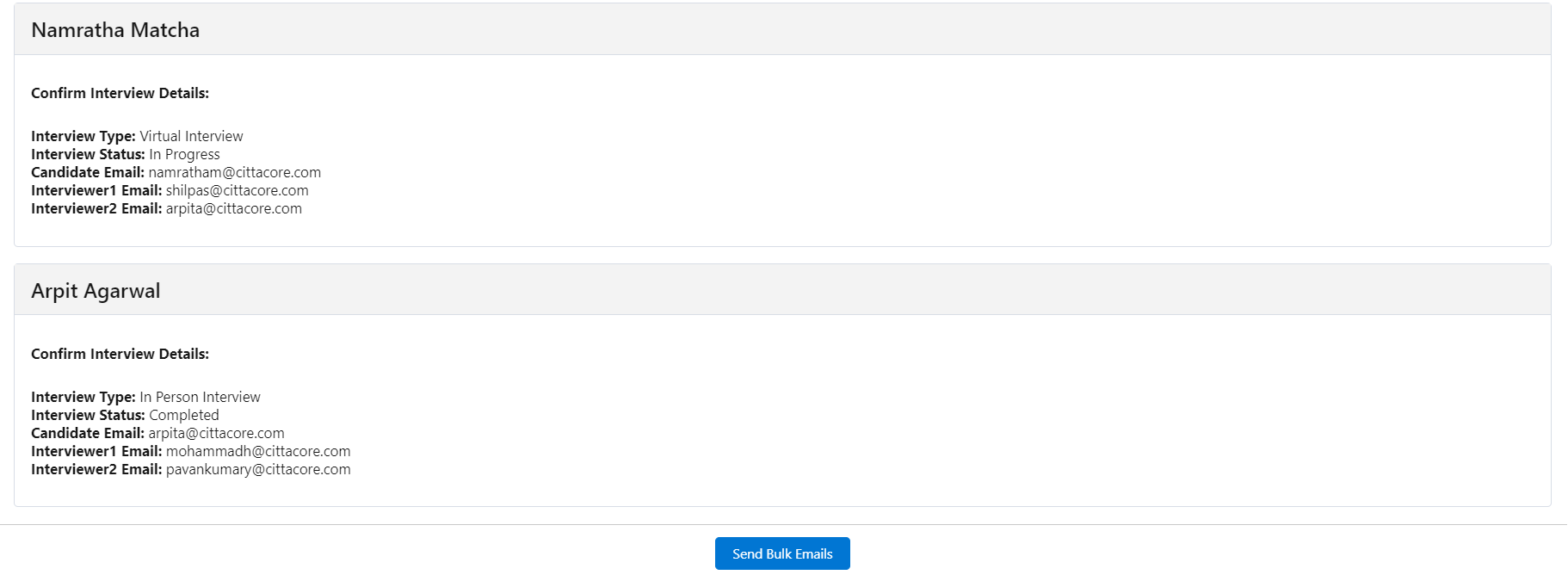
* It should fetch the interview records from the server when the component is loaded.
* It should allow the user to search for specific interview records by typing in the search input field.
* It should allow the user to select multiple interview records.
* It should display the selected interview records in separate cards.
* It should display the interview details for each selected record.
* It should allow the user to send bulk emails to the selected interview records.



**Design**

This component has the following design:

* The interview records are displayed in a datatable with the Candidate Name, Interview Date, Interview Type, Interview Status, Candidate Email, and Interviewer Email fields.
* The selected interview records are displayed in separate cards with the Candidate Name as the header.
* The interview details for each selected record are displayed below the header.
* The Send Bulk Emails button is displayed at the bottom of the component.

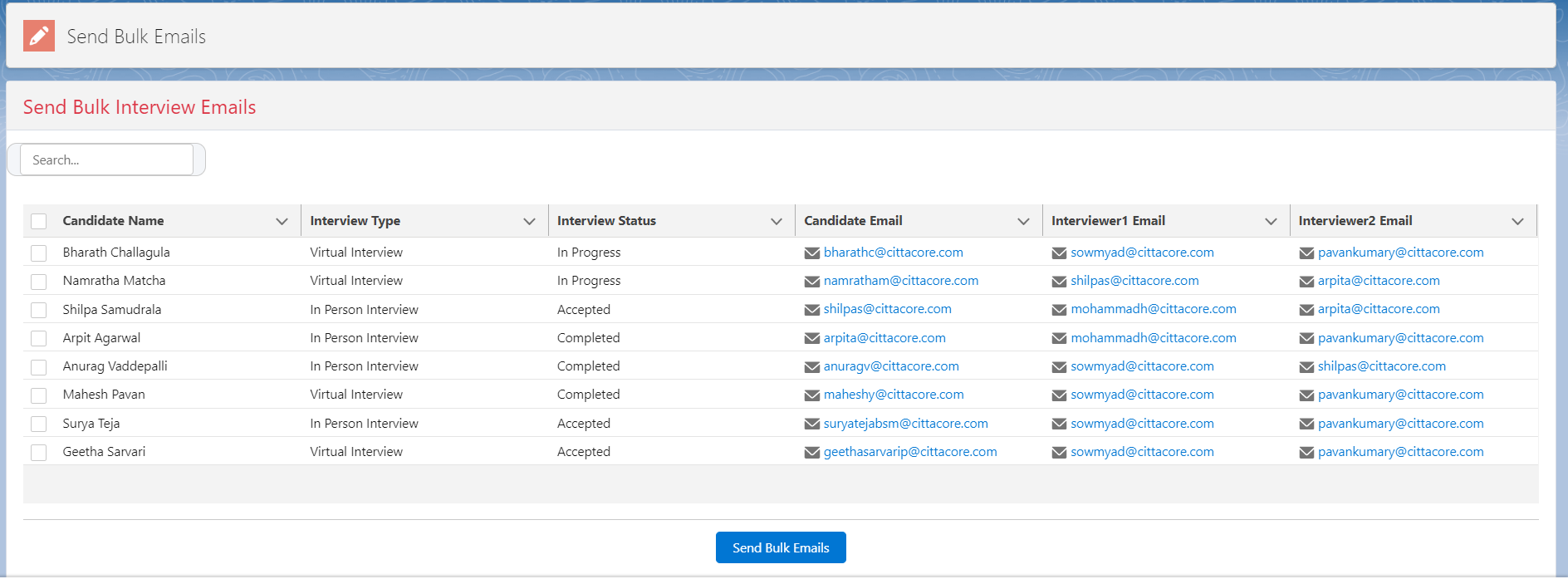


**Implementation**

This component is implemented using Lightning Web Components and Apex. The component code consists of a template and a JavaScript file.

**Template**

* The template contains the HTML markup for the component.
* It consists of a div with the slds-card class that contains a div with the slds-card\_\_header class and a h2 tag that displays the title. Below the header, there is a lightning-input field that allows the user to search for specific interview records.
* The search input field is followed by a datatable that displays the interview records.
* The datatable contains a lightning-input checkbox for each row, which allows the user to select multiple records.
* The selected records are displayed in separate cards below the datatable. Each card displays the Candidate Name as the header and the interview details below the header.
* The Send Bulk Emails button is displayed at the bottom of the component.



**The JavaScript code contains the following functions:**

* getInterviewRecords - This function calls the Apex method to fetch interview records from Salesforce.
* handleRowSelection - This function handles the selection of interview records in the datatable.
* handleSearchTermChange - This function handles the change event of the search input field.
* sendBulkEmails - This function calls the Apex method to send bulk emails to the selected candidates**.**

**The Apex code contains the following methods:**

* This Apex class, InterviewEmailController, has two methods that are exposed as AuraEnabled Apex methods to be used in Lightning Components or other Aura frameworks.
* The first method, sendBulkEmails(), takes a list of Interview Scheduling record IDs as a parameter and sends emails to the applicants and interviewers based on the Interview Type. The email contains details about the scheduled interview such as date, time, location, etc. This method is using the Salesforce SingleEmailMessage class to send emails. It retrieves the Interview Scheduling records based on the given IDs, creates the email content, and sends emails using Messaging.sendEmail() method.
* The second method, getInterviewRecords(), returns a list of all Interview Scheduling records in the Salesforce instance.

**The CSS code contains the following styles:**

* This CSS code is intended to style a component that has a search input, a datatable, and other elements. Here is a description of each style rule:
* .slds-card

This rule sets the border, border radius, and box shadow of the component.

* .slds-input.search-input

This rule styles the search input. It sets the border radius, padding, box shadow, background color, and width of the input.

* .slds-icon.search-icon

This rule styles the search icon. It sets the color of the icon.

* .slds-card\_\_header

This rule styles the header of the component. It sets the background color, border bottom, and padding of the header.

* .slds-card\_\_body

This rule styles the body of the component. It sets the padding of the body.

* .datatable-container

This rule styles the container of the datatable. It sets the height and overflow of the container.

* .slds-checkbox

This rule styles the checkboxes in the component. It sets the margin right of the checkboxes.

* .slds-card\_boundary

This rule sets the border, border radius, and margin bottom of the component.

* .slds-card\_\_header-title

This rule styles the title of the component. It sets the font size and font weight of the title.

* .slds-text-heading\_small

This rule styles the small text in the component. It sets the font size of the text.

* .slds-text-color\_error

This rule styles the error text in the component. It sets the color of the text.

* .lightning-input-container

This rule styles the input container of the component. It sets the margin bottom of the input container.

**To use the component, follow these steps:**

* Create a Lightning App or a record page and drag the Interview Scheduling Component to it.
* Set the title attribute of the component as desired.
* The component will fetch interview records from Salesforce when the page is loaded.
* Use the search input field to search for specific interview records based on candidate name, interview date, etc.
* Select one or more interview records from the datatable.
* Click the "Send Bulk Emails" button to send emails to the selected candidates.
* On the interview schedulers list view page, we have included a button layout for the list view, as well as a "send bulk emails" button. By clicking on this button, it is possible to launch the associated component.

**Send Interview Email Notifications**

**Overview**

This record-triggered flow is designed to automatically send email notifications to applicants and interviewers based on the type of interview, when an interview scheduler record is created or updated and the interview status is set to "Accepted". The flow is triggered by changes to the Interview Scheduler object.

**Flow Steps:**

**Record Lookup:** The flow first performs a lookup on the related "Job Application" record to obtain the applicant's email address.

**Decision:** The flow then determines the type of interview, by checking the value of the "Interview Type" field on the Interview Scheduler record.

**Email Notification:** The flow sends a notification email to the applicant and interviewers based on the interview type:

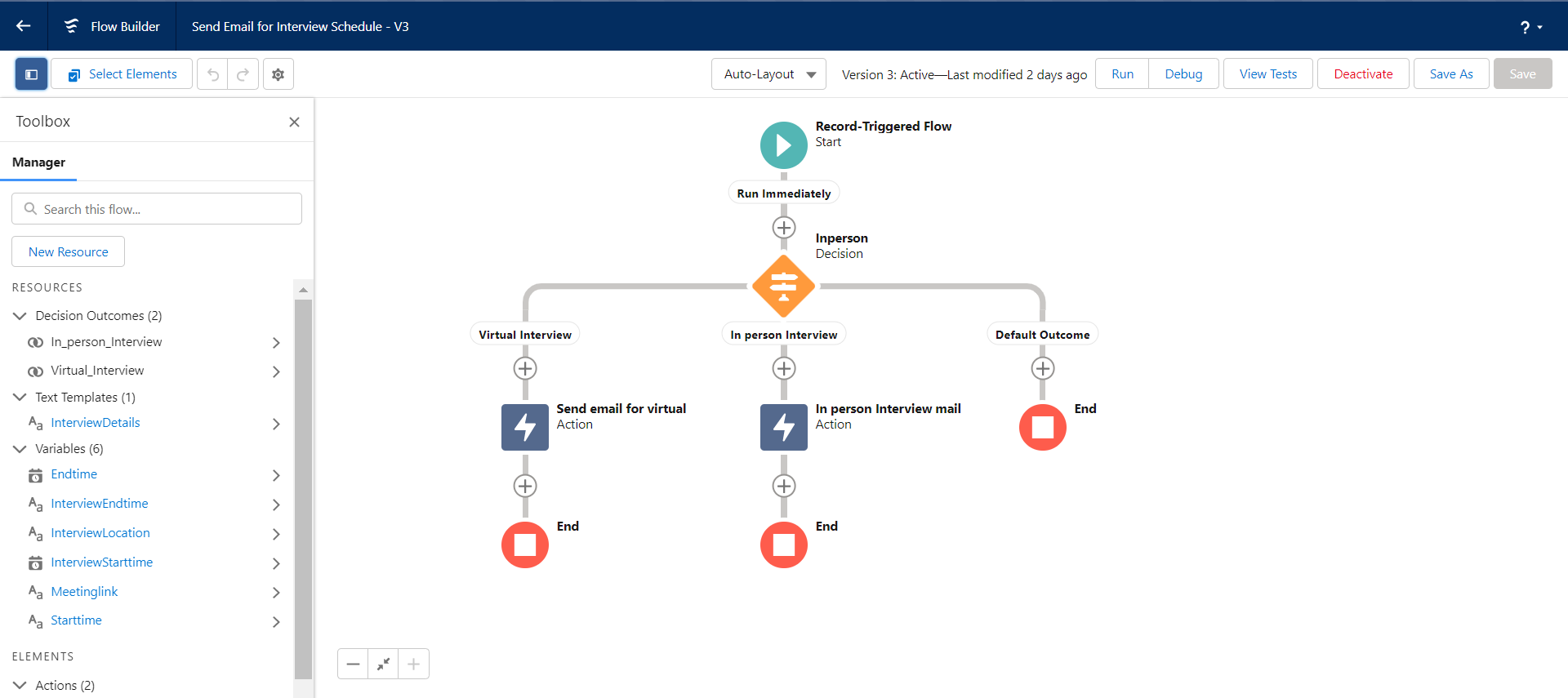
**In-Person Interview:** If the interview type is "In-Person Interview", the flow sends an email notification to the applicant and interviewers with the interview details and location.

**Virtual Interview:** If the interview type is "Virtual Interview", the flow sends an email notification to the applicant and interviewers with the interview details and a link to the video conferencing platform.

**Debug**: The flow ends by generating a debug log that records the email notifications sent and any errors encountered during the flow execution.

**Flow Diagram:**

Record Trigger Flow



**Flow Considerations:**

* The flow relies on the "Job Application" record being populated with the applicant's email address.
* The flow assumes that the interview scheduler record is linked to the appropriate interviewers and has the necessary interview details populated.
* The flow sends the notification email using an email alert or custom email template. These must be set up in advance in the Salesforce org.
* The flow generates debug logs that can be used to troubleshoot any issues with the email notifications.
* As this is a record-triggered flow, it can be set to run either before or after the record is saved, depending on the requirements of the business process being automated.

**Code Implemented for Send Bulk Emails**

**Code review enhancements suggested by Chandra Prasad**

* In the conditional checking section, I initially utilized two separate conditional blocks using "if" and "elseif" statements.
* The first block was implemented to set the email addresses, while the second block was developed to create the email body. Prasad suggested that there is no need to use two separate conditional blocks, resulting in a more efficient and simplified code structure.
* Additionally, I have also included try-catch blocks to enhance the code's reliability and stability during execution.

**Here is the modified code:**

**InterviewEmailController.APXC**

// server side code for sending bulk emails to selected records

public with sharing class InterviewEmailController {

 //fetching the interview scheduler records from salesforce database

*@AuraEnabled*

    public static *List*<*EMS\_Interview\_Scheduling\_\_c*> getInterviewRecords() {

*List*<*EMS\_Interview\_Scheduling\_\_c*> interviewList = new *List*<*EMS\_Interview\_Scheduling\_\_c*>();

        try {

            interviewList = [SELECT Id, EMS\_Interview\_Round\_\_c, EMS\_Interview\_Status\_\_c, EMS\_Interview\_Type\_\_c, EMS\_Meeting\_Link\_\_c, EMS\_Interviewer\_Name\_\_c, EMS\_Job\_Application\_ID\_\_c, EMS\_Candidate\_Name\_\_c, EMS\_Interview\_Location\_\_c, EMS\_Interviewer\_Email\_\_c, EMS\_Applicant\_Email\_\_c, EMS\_Interviewer\_2\_Email\_\_c, EMS\_Interview\_Stage\_\_c, EMS\_Decline\_Reason\_\_c, EMS\_Interview\_Start\_Time\_c\_\_c, EMS\_InterviewEndTime\_\_c FROM *EMS\_Interview\_Scheduling\_\_c* WITH SECURITY\_ENFORCED];

        } catch(*Exception* e) {

*System*.debug('Error fetching interview records: ' + e.getMessage());

        }

        return interviewList;

    }

*@AuraEnabled*

    public static void sendBulkEmails(*List*<*Id*> interviewIds) {

*List*<*EMS\_Interview\_Scheduling\_\_c*> interviewList = new *List*<*EMS\_Interview\_Scheduling\_\_c*>();

        try {

            interviewList = [SELECT Id, EMS\_Interview\_Start\_Time\_c\_\_c, EMS\_Interview\_Round\_\_c, EMS\_InterviewEndTime\_\_c, EMS\_Interview\_Status\_\_c, EMS\_Interview\_Type\_\_c, EMS\_Meeting\_Link\_\_c, EMS\_Interviewer\_Name\_\_c, EMS\_Job\_Application\_ID\_\_c, EMS\_Candidate\_Name\_\_c, EMS\_Interview\_Location\_\_c, EMS\_Interviewer\_Email\_\_c, EMS\_Applicant\_Email\_\_c, EMS\_Interviewer\_2\_Email\_\_c, EMS\_Interview\_Stage\_\_c, EMS\_Decline\_Reason\_\_c FROM *EMS\_Interview\_Scheduling\_\_c* WHERE EMS\_Interview\_Status\_\_c = 'Accepted' AND Id IN :interviewIds WITH SECURITY\_ENFORCED];

        } catch(*Exception* e) {

*System*.debug('Error fetching interview records: ' + e.getMessage());

        }

*List*<*Messaging*.*SingleEmailMessage*> emails = new *List*<*Messaging*.*SingleEmailMessage*>();

        for(*EMS\_Interview\_Scheduling\_\_c* interview : interviewList) {

            try {

*Messaging*.*SingleEmailMessage* email = new *Messaging*.*SingleEmailMessage*();

                String[] toAddresses = new String[] { };

                if(interview.EMS\_Interview\_Type\_\_c == 'Virtual Interview' || interview.EMS\_Interview\_Type\_\_c == 'In Person Interview') {

                    toAddresses = new String[] { interview.EMS\_Applicant\_Email\_\_c, interview.EMS\_Interviewer\_Email\_\_c, interview.EMS\_Interviewer\_2\_Email\_\_c };

                }

                email.setToAddresses(toAddresses);

                email.setSubject('Interview Schedule Details');

                String body = '';

*DateTime* startDateTime = DateTime.newInstance(interview.EMS\_Interview\_Start\_Time\_c\_\_c.year(), interview.EMS\_Interview\_Start\_Time\_c\_\_c.month(), interview.EMS\_Interview\_Start\_Time\_c\_\_c.day(), interview.EMS\_Interview\_Start\_Time\_c\_\_c.hour(), interview.EMS\_Interview\_Start\_Time\_c\_\_c.minute(), 0);

*DateTime* endDateTime = DateTime.newInstance(interview.EMS\_InterviewEndTime\_\_c.year(), interview.EMS\_InterviewEndTime\_\_c.month(), interview.EMS\_InterviewEndTime\_\_c.day(), interview.EMS\_InterviewEndTime\_\_c.hour(), interview.EMS\_InterviewEndTime\_\_c.minute(), 0);

                body += 'Your interview is scheduled for ' + startDateTime.format('MM/dd/yyyy hh:mm a') + ' - ' + endDateTime.format('hh:mm a') + ' (UTC).\n';

                if(interview.EMS\_Interview\_Type\_\_c == 'Virtual Interview') {

                    body += 'The interview will be held virtually. Please use the following link to join the virtual interview: ' + interview.EMS\_Meeting\_Link\_\_c + '\n';

                } else if(interview.EMS\_Interview\_Type\_\_c == 'In Person Interview') {

                    body += 'The interview will be held in-person at the following location: ' + interview.EMS\_Interview\_Location\_\_c + '\n';

                }

                email.setPlainTextBody(body);

                emails.add(email);

            } catch(*Exception* e) {

*System*.debug('Error creating email: ' + e.getMessage());

            }

        }

        try {

            if(!emails.isEmpty()) {

*System*.debug('Sending emails...');

                Messaging.sendEmail(emails);

            }

        } catch(*Exception* e) {

*System*.debug('Error sending emails: ' + e.getMessage());

        }

    }

**LWC Code for sending bulk emails:**

Prasad did not provide any suggestions for the LWC component.

**Template code : interviewSchedulingComponent.html**

<template>

    <div class="slds-card">

        <div class="slds-card\_\_header slds-grid slds-grid\_align-spread">

            <h2 class="slds-text-heading\_medium slds-text-color\_error">{title}</h2>

        </div>

        <lightning-input

        label="Search"

        value={searchTerm}

        onchange={handleSearchTermChange}

        class="slds-input--small slds-m-right\_small slds-grow search-input"

        variant="label-hidden"

        placeholder="Search...">

        <lightning-icon

          icon-name="utility:search"

          slot="icon"

          size="x-small"

          class="slds-icon slds-icon\_x-small slds-icon-text-default slds-m-left\_x-small search-icon">

        </lightning-icon>

      </lightning-input>

        <div class="slds-card\_\_body">

            <template if:true={records}>

                <div class="datatable-container">

                    <lightning-datatable

                        key-field="Id"

                        data={records}

                        columns={columns}

                        onrowselection={handleRowSelection}>

                        <template for:each={records} for:item="record">

                            <lightning-input key={record.Id} type="checkbox" data-id={record.Id} class="slds-checkbox"></lightning-input>

                        </template>

                    </lightning-datatable>

                </div>

                <div class="slds-m-around\_medium">

                    <template for:each={selectedRecords} for:item="record">

                        <div key={record.Id} class="slds-card slds-card\_boundary">

                            <div class="slds-card\_\_header slds-grid">

                                <h3 class="slds-card\_\_header-title">{record.EMS\_Candidate\_Name\_\_c}</h3>

                            </div>

                            <div class="slds-card\_\_body">

                                <template if:true={record.EMS\_Interview\_Type\_\_c}>

                                    <div class="slds-grid slds-grid\_align-spread slds-p-bottom\_medium">

                                        <div class="slds-col slds-size\_1-of-1 slds-medium-size\_1-of-2">

                                            <p class="slds-text-heading\_small slds-p-bottom\_x-small"><strong>Confirm Interview Details:</strong></p>

                                        </div>

                                    </div>

                                </template>

                                <div class="slds-grid slds-grid\_vertical-align-center slds-wrap">

                                    <div class="slds-col slds-size\_1-of-1 slds-medium-size\_1-of-2">

                                        <p class="slds-text-heading\_small"><strong>Interview Type:</strong> {record.EMS\_Interview\_Type\_\_c}</p>

                                        <p class="slds-text-heading\_small"><strong>Interview Status:</strong> {record.EMS\_Interview\_Status\_\_c}</p>

                                        <p class="slds-text-heading\_small"><strong>Candidate Email:</strong> {record.EMS\_Applicant\_Email\_\_c}</p>

                                        <p class="slds-text-heading\_small"><strong>Interviewer1 Email:</strong> {record.EMS\_Interviewer\_Email\_\_c}</p>

                                        <p class="slds-text-heading\_small"><strong>Interviewer2 Email:</strong> {record.EMS\_Interviewer\_2\_Email\_\_c }</p>

                                    </div>

                                </div>

                            </div>

                        </div>

                    </template>

                </div>

                <div class="slds-card\_\_footer">

                    <div class="slds-button-group">

                        <lightning-button

                            variant="brand"

                            label="Send Bulk Emails"

                            onclick={sendBulkEmails}>

                        </lightning-button>

                    </div>

                </div>

            </template>

        </div>

    </div>

</template>

**JavaScript code: interviewSchedulingComponent.js**

import { LightningElement, track, api } from 'lwc';

import { ShowToastEvent } from 'lightning/platformShowToastEvent';

import sendBulkEmails from '@salesforce/apex/InterviewEmailController.sendBulkEmails';

import getInterviewRecords from '@salesforce/apex/InterviewEmailController.getInterviewRecords';

export default *class* InterviewSchedulingComponent extends LightningElement {

    @api title

    @api selectedRecords = [];

    @track records = [];

    @track columns = [

        { label: 'Candidate Name', fieldName: 'EMS\_Candidate\_Name\_\_c', type: 'text' },

        //{ label: 'Interview Date', fieldName: 'EMS\_Interview\_Date\_\_c', type: 'date' },

        { label: 'Interview Type', fieldName: 'EMS\_Interview\_Type\_\_c', type: 'text' },

        //{ label: 'Interview Stage', fieldName: 'EMS\_Interview\_Stage\_\_c', type: 'picklist' },

        { label: 'Interview Status', fieldName: 'EMS\_Interview\_Status\_\_c', type: 'picklist' },

        { label: 'Candidate Email', fieldName: 'EMS\_Applicant\_Email\_\_c', type: 'email' },

        { label: 'Interviewer1 Email', fieldName: 'EMS\_Interviewer\_Email\_\_c', type: 'email' },

        { label: 'Interviewer2 Email', fieldName: 'EMS\_Interviewer\_2\_Email\_\_c', type: 'email' }

    ];

    // Fetches the interview records from the server

    connectedCallback() {

        this.getInterviewRecords();

        this.title = 'Send Bulk Interview Emails';

    }

    // Handles the selection of interview records

    handleRowSelection(*event*) {

        this.selectedRecords = *event*.detail.selectedRows;

    }

          // Handles the search term change event

          handleSearchTermChange(*event*) {

            this.searchTerm = event.target.value;

            this.getInterviewRecords();

        }

   // Fetches the interview records from the server

getInterviewRecords() {

    getInterviewRecords()

        .then(*result* *=>* {

            // Filter the records based on the search term

            if (this.searchTerm) {

*const* regex = new RegExp(this.searchTerm, "i");

                this.records = result.filter(*record* *=>*

                    regex.test(record.EMS\_Candidate\_Name\_\_c) ||

                    regex.test(record.EMS\_Interview\_Type\_\_c) ||

                    regex.test(record.EMS\_Interview\_Status\_\_c) ||

                    regex.test(record.EMS\_Applicant\_Email\_\_c) ||

                    regex.test(record.EMS\_Interviewer\_2\_Email\_\_c) ||

                    regex.test(record.EMS\_Interviewer\_Email\_\_c)

                );

            } else {

                this.records = result;

            }

        })

        .catch(*error* *=>* {

            console.error(error);

        });

}

    //Sending bulk emails

    sendBulkEmails() {

        console.log('records ' + JSON.stringify(this.selectedRecords));

*const* interviewIds = this.selectedRecords.map(*record* *=>* *record*.Id);

        console.log(JSON.stringify('Interview Ids List :' + interviewIds));

        if (interviewIds.length > 0) {

            sendBulkEmails({ interviewIds })

    .then(*result* *=>* {

        console.log('result: '+ JSON.stringify(*result*));

        this.dispatchEvent(

            new ShowToastEvent({

                title: 'Success',

                message: 'Emails sent successfully',

                variant: 'success'

            })

        );

    })

    .catch(*error* *=>* {

        this.dispatchEvent(

            new ShowToastEvent({

                title: 'Error',

                message: *error*.body.message,

                variant: 'error'

            })

        );

    });

}

    }

}

**CSS code for UI styling: interviewSchedulingComponent.css**

/\* css \*/

  /\* Add a gradient background to the entire component \*/

/\* Add a gradient background to the entire component \*/

.slds-card {

*border*: 1px solid #d8dde6;

*border-radius*: 0.25rem;

*box-shadow*: none;

  }

  /\* Style the search input \*/

.slds-input.search-input {

*border-radius*: 10px;

*padding*: 0 12px;

*box-shadow*: none;

*background-color*: #f4f6f9;

*width*: 200px;

}

/\* Style the search icon \*/

.slds-icon.search-icon {

*color*: #a3acb8;

}

  .slds-card\_\_header {

*background-color*: #f3f3f3;

*border-bottom*: 1px solid #d8dde6;

*padding*: 0.75rem 1rem;

  }

  .slds-card\_\_body {

*padding*: 1rem;

  }

  .datatable-container {

*height*: 300px;

*overflow*: auto;

  }

  .slds-checkbox {

*margin-right*: 0.5rem;

  }

  .slds-card\_boundary {

*border*: 1px solid #d8dde6;

*border-radius*: 0.25rem;

*margin-bottom*: 0.5rem;

  }

  .slds-card\_\_header-title {

*font-size*: 1.25rem;

*font-weight*: 600;

  }

  .slds-text-heading\_small {

*font-size*: 0.875rem;

  }

  .slds-text-color\_error {

*color*: #dc3545;

  }

  .lightning-input-container {

*margin-bottom*: 0.5rem;

  }

**Meta.xml file for visibility in home. record, app pages:**

<?xml version="1.0" encoding="UTF-8"?>

<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">

    <apiVersion>56.0</apiVersion>

    <isExposed>true</isExposed>

    <targets>

        <target>lightning\_\_AppPage</target>

        <target>lightning\_\_HomePage</target>

        <target>lightning\_\_RecordPage</target>

        <target>lightning\_\_RecordAction</target>

    </targets>

</LightningComponentBundle>

**Test class for InterviewEmailController: InterviewEmailControllerTest.APXC**

The following test class has achieved 100% code coverage, meeting the requirement to deploy the code to production. Achieving full code coverage ensures that all parts of the code have been executed at least once during the testing process, reducing the risk of potential errors or bugs in the production environment.

*@isTest*

public class InterviewEmailControllerTest {

*@isTest*

    static void testGetInterviewRecords() {

*EMS\_Applicant\_\_c* applicant = new *EMS\_Applicant\_\_c*(EMS\_Email\_\_c='testcandidate@example.com',EMS\_Mobile\_\_c='+918090908090', EMS\_Last\_Name\_\_c = 'Karim');

    insert applicant;

*EMS\_Employee\_\_c* interviewer = new *EMS\_Employee\_\_c*(Name='Naguru', EMS\_Type\_\_c='Full Time', EMS\_Work\_Email\_\_c='testinterviewer@example.com');

    insert interviewer;

*EMS\_Position\_\_c* position = new *EMS\_Position\_\_c*(Name='Test Position',EMS\_Required\_Skills\_\_c = 'System Administrator', EMS\_Status\_\_c='Active');

    insert position;

*EMS\_Job\_Opening\_\_c* jobopening = new *EMS\_Job\_Opening\_\_c*();

    jobopening.EMS\_Department\_\_c = 'Finance';

    jobopening.EMS\_Experience\_\_c = '1 - 2 Yrs';

     jobopening.EMS\_Educational\_Requirements\_\_c = 'B.Tech/B.E';

     jobopening.EMS\_Minimum\_Salary\_\_c = 9000;

    jobopening.EMS\_Location\_\_c = 'Hyderabad - Telangana';

    jobopening.EMS\_Open\_Date\_\_c = Date.today();

     jobopening.Hire\_by\_Date\_\_c = Date.today();

     jobopening.EMS\_Maximum\_Salary\_\_c = 70000;

    jobopening.EMS\_Job\_Title\_\_c = position.Id;

        jobopening.EMS\_No\_of\_Openings\_\_c = 10;

      insert jobopening;

*EMS\_Job\_Applications\_\_c* jobApp = new *EMS\_Job\_Applications\_\_c*(EMS\_Applicant\_\_c=applicant.Id, EMS\_Stage\_\_c='Pre-Screening', EMS\_Status\_\_c='Passed', EMS\_Job\_Opening\_\_c=jobopening.Id);

    insert jobApp;

        // Create test data

*EMS\_Interview\_Scheduling\_\_c* interview = new *EMS\_Interview\_Scheduling\_\_c*(

            EMS\_Interview\_Date\_\_c = Date.today().addDays(8),

            EMS\_Interview\_Start\_Time\_c\_\_c = DateTime.newInstance(2023,4,20, 10, 30, 9),

            EMS\_InterviewEndTime\_\_c = DateTime.newInstance(2023,4,20, 10, 45, 9),

            EMS\_Interview\_Type\_\_c = 'In Person Interview',

            //EMS\_Interviewer\_Email\_\_c = 'test-interviewer@test.com',

            //EMS\_Applicant\_Email\_\_c = 'test-applicant@test.com',

            EMS\_Interview\_Location\_\_c = 'Hyd',

            EMS\_Interview\_Round\_\_c = 'Technical Round',

            EMS\_Interview\_Stage\_\_c = 'Scheduled',

            EMS\_Interview\_Status\_\_c = 'Accepted',

            EMS\_Job\_Application\_ID\_\_c = jobApp.Id

        );

        insert interview;

        // Call the method

*List*<*EMS\_Interview\_Scheduling\_\_c*> interviewList = InterviewEmailController.getInterviewRecords();

        // Verify the result

*System*.assertEquals(1, interviewList.size());

*System*.assertEquals(interview.Id, interviewList[0].Id);

        //System.assertEquals(interview.EMS\_Interview\_Date\_\_c, interviewList[0].EMS\_Interview\_Date\_\_c);

*System*.assertEquals(interview.EMS\_Interview\_Start\_Time\_c\_\_c, interviewList[0].EMS\_Interview\_Start\_Time\_c\_\_c);

*System*.assertEquals(interview.EMS\_InterviewEndTime\_\_c, interviewList[0].EMS\_InterviewEndTime\_\_c);

*System*.assertEquals(interview.EMS\_Interview\_Type\_\_c, interviewList[0].EMS\_Interview\_Type\_\_c);

        //System.assertEquals(interview.EMS\_Interviewer\_Email\_\_c, interviewList[0].EMS\_Interviewer\_Email\_\_c);

        //System.assertEquals(interview.EMS\_Applicant\_Email\_\_c, interviewList[0].EMS\_Applicant\_Email\_\_c);

*System*.assertEquals(interview.EMS\_Interview\_Location\_\_c, interviewList[0].EMS\_Interview\_Location\_\_c);

*System*.assertEquals(interview.EMS\_Interview\_Round\_\_c, interviewList[0].EMS\_Interview\_Round\_\_c);

*System*.assertEquals(interview.EMS\_Interview\_Stage\_\_c, interviewList[0].EMS\_Interview\_Stage\_\_c);

*System*.assertEquals(interview.EMS\_Interview\_Status\_\_c, interviewList[0].EMS\_Interview\_Status\_\_c);

    }

*@isTest*

   static void testSendBulkEmails() {

*EMS\_Applicant\_\_c* applicant = new *EMS\_Applicant\_\_c*(EMS\_Email\_\_c='testcandidate@example.com',EMS\_Mobile\_\_c='+918090908090', EMS\_Last\_Name\_\_c = 'Karim');

    insert applicant;

*EMS\_Employee\_\_c* interviewer = new *EMS\_Employee\_\_c*(Name='Naguru', EMS\_Type\_\_c='Full Time', EMS\_Work\_Email\_\_c='testinterviewer@example.com');

    insert interviewer;

*EMS\_Position\_\_c* position = new *EMS\_Position\_\_c*(Name='Test Position',EMS\_Required\_Skills\_\_c = 'System Administrator', EMS\_Status\_\_c='Active');

    insert position;

*EMS\_Job\_Opening\_\_c* jobopening = new *EMS\_Job\_Opening\_\_c*();

    jobopening.EMS\_Department\_\_c = 'Finance';

    jobopening.EMS\_Experience\_\_c = '1 - 2 Yrs';

     jobopening.EMS\_Educational\_Requirements\_\_c = 'B.Tech/B.E';

     jobopening.EMS\_Minimum\_Salary\_\_c = 9000;

    jobopening.EMS\_Location\_\_c = 'Hyderabad - Telangana';

    jobopening.EMS\_Open\_Date\_\_c = Date.today();

     jobopening.Hire\_by\_Date\_\_c = Date.today();

     jobopening.EMS\_Maximum\_Salary\_\_c = 70000;

    jobopening.EMS\_Job\_Title\_\_c = position.Id;

        jobopening.EMS\_No\_of\_Openings\_\_c = 10;

      insert jobopening;

*EMS\_Job\_Applications\_\_c* jobApp = new *EMS\_Job\_Applications\_\_c*(EMS\_Applicant\_\_c=applicant.Id, EMS\_Stage\_\_c='Pre-Screening', EMS\_Status\_\_c='Passed', EMS\_Job\_Opening\_\_c=jobopening.Id);

    insert jobApp;

*EMS\_Employee\_\_c* emp = new *EMS\_Employee\_\_c*(Name='Arpit Agarwal', EMS\_Type\_\_c='Full Time', EMS\_Work\_Email\_\_c='test-interviewer@test.com');

       insert emp;

        // Create test data

*EMS\_Interview\_Scheduling\_\_c* interview = new *EMS\_Interview\_Scheduling\_\_c*(

            EMS\_Interview\_Date\_\_c = Date.today().addDays(8),

            EMS\_Interview\_Start\_Time\_c\_\_c = DateTime.newInstance(2023,4,20, 10, 30, 9),

            EMS\_InterviewEndTime\_\_c = DateTime.newInstance(2023,4,20, 10, 45, 9),

            EMS\_Interview\_Type\_\_c = 'In Person Interview',

            EMS\_Interviewer\_2\_Name\_\_c = emp.Id,

            EMS\_Interviewer\_1\_Name\_\_c = emp.Id,

            //EMS\_Interviewer\_Email\_\_c = emp.EMS\_Work\_Email\_\_c,

            //EMS\_Applicant\_Email\_\_c = 'test-applicant@test.com',

            EMS\_Interview\_Location\_\_c = 'Hyd',

            EMS\_Interview\_Round\_\_c = 'Technical Round',

            EMS\_Interview\_Stage\_\_c = 'Scheduled',

            EMS\_Interview\_Status\_\_c = 'Accepted',

            EMS\_Job\_Application\_ID\_\_c = jobApp.Id

        );

        insert interview;

*Test*.startTest();

    // Call the method under test with the interview Id

*List*<*Id*> interviewIds = new *List*<*Id*>{interview.Id};

    InterviewEmailController.sendBulkEmails(interviewIds);

*Test*.stopTest();

    String subject;

    String body;

    if(interview.EMS\_Interview\_Type\_\_c=='In Person Interview')

    {

        subject = 'Interview Schedule Details';

        body =' Your interview is scheduled for 4/13/2023 10:00 AM - 11:00 AM (UTC)';

    }

   system.assertEquals('Interview Schedule Details', subject);

    system.assertEquals(' Your interview is scheduled for 4/13/2023 10:00 AM - 11:00 AM (UTC)',body);

}

*@isTest*

static void testSendBulkEmails1() {

*EMS\_Applicant\_\_c* applicant = new *EMS\_Applicant\_\_c*(EMS\_Email\_\_c='testcandidate@example.com',EMS\_Mobile\_\_c='+918090908090', EMS\_Last\_Name\_\_c = 'Karim');

    insert applicant;

*EMS\_Employee\_\_c* interviewer = new *EMS\_Employee\_\_c*(Name='Naguru', EMS\_Type\_\_c='Full Time', EMS\_Work\_Email\_\_c='testinterviewer@example.com');

    insert interviewer;

*EMS\_Position\_\_c* position = new *EMS\_Position\_\_c*(Name='Test Position',EMS\_Required\_Skills\_\_c = 'System Administrator', EMS\_Status\_\_c='Active');

    insert position;

*EMS\_Job\_Opening\_\_c* jobopening = new *EMS\_Job\_Opening\_\_c*();

    jobopening.EMS\_Department\_\_c = 'Finance';

    jobopening.EMS\_Experience\_\_c = '1 - 2 Yrs';

     jobopening.EMS\_Educational\_Requirements\_\_c = 'B.Tech/B.E';

     jobopening.EMS\_Minimum\_Salary\_\_c = 9000;

    jobopening.EMS\_Location\_\_c = 'Hyderabad - Telangana';

    jobopening.EMS\_Open\_Date\_\_c = Date.today();

     jobopening.Hire\_by\_Date\_\_c = Date.today();

     jobopening.EMS\_Maximum\_Salary\_\_c = 70000;

    jobopening.EMS\_Job\_Title\_\_c = position.Id;

        jobopening.EMS\_No\_of\_Openings\_\_c = 10;

      insert jobopening;

*EMS\_Job\_Applications\_\_c* jobApp = new *EMS\_Job\_Applications\_\_c*(EMS\_Applicant\_\_c=applicant.Id, EMS\_Stage\_\_c='Pre-Screening', EMS\_Status\_\_c='Passed', EMS\_Job\_Opening\_\_c=jobopening.Id);

    insert jobApp;

*EMS\_Employee\_\_c* emp = new *EMS\_Employee\_\_c*(Name='Arpit Agarwal', EMS\_Type\_\_c='Full Time', EMS\_Work\_Email\_\_c='test-interviewer@test.com');

       insert emp;

        // Create test data

*EMS\_Interview\_Scheduling\_\_c* interview = new *EMS\_Interview\_Scheduling\_\_c*(

            EMS\_Interview\_Date\_\_c = Date.today().addDays(8),

            EMS\_Interview\_Start\_Time\_c\_\_c = DateTime.newInstance(2023,4,20, 10, 30, 9),

            EMS\_InterviewEndTime\_\_c = DateTime.newInstance(2023,4,20, 10, 45, 9),

            EMS\_Interview\_Type\_\_c = 'Virtual Interview',

            EMS\_Interviewer\_2\_Name\_\_c = emp.Id,

            EMS\_Interviewer\_1\_Name\_\_c = emp.Id,

            //EMS\_Interviewer\_Email\_\_c = emp.EMS\_Work\_Email\_\_c,

            //EMS\_Applicant\_Email\_\_c = 'test-applicant@test.com',

            EMS\_Interview\_Location\_\_c = 'Hyd',

            EMS\_Interview\_Round\_\_c = 'Technical Round',

            EMS\_Interview\_Stage\_\_c = 'Scheduled',

            EMS\_Interview\_Status\_\_c = 'Accepted',

            EMS\_Job\_Application\_ID\_\_c = jobApp.Id,

            RecordTypeId = '012DT000003sALJYA2'

        );

        insert interview;

*Test*.startTest();

    // Call the method under test with the interview Id

*List*<*Id*> interviewIds = new *List*<*Id*>{interview.Id};

    InterviewEmailController.sendBulkEmails(interviewIds);

*Test*.stopTest();

    String subject;

    String body;

    if(interview.EMS\_Interview\_Type\_\_c == 'Virtual Interview')

    {

        subject = 'Interview Schedule Details';

        body ='Virtual Interview';

    }

   system.assertEquals('Interview Schedule Details', subject);

    system.assertEquals( 'Virtual Interview',body);

}

}