

HEALTHY WEALTHY CARE CENTER

**A Project Report Submitted
In Partial Fulfillment of the Requirements
for the Degree of**

MASTER OF COMPUTER APPLICATIONS

**by
Deepak Goel
(1802914002)**

**Under the Supervision of
Dr. Sangeeta Arora
KIET Group of Institutions, Ghaziabad**



**to the
FACULTY OF MCA**

**DR. APJ ABDUL KALAM TECHNICAL UNIVERSITY
(Formerly Uttar Pradesh Technical University) LUCKNOW**

July 2021

DECLARATION

I hereby declare that the work presented in this report entitled “HEALTHY WEALTHY CARE CENTER”, was carried out by me. I have not submitted the matter embodied in this report for the award of any other degree or diploma of any other University or Institute.

I have given due credit to the original authors/sources for all the words, ideas, diagrams, graphics, computer programs, experiments, results, that are not my original contribution.

I have used quotation marks to identify verbatim sentences and given credit to the original authors/sources.

I affirm that no portion of my work is plagiarized, and the experiments and results reported in the report are not manipulated. In the event of a complaint of plagiarism and the manipulation of the experiments and results, I shall be fully responsible and answerable.

Name : Deepak Goel

Roll. No. : 1802914002

Branch : Master of Computer Applications

(Candidate Signature)

TRAINING CERTIFICATE

DocuSign Envelope ID: E5D191CD-D561-4FCE-9B4C-AEF6638EC63A



2 August 2021

To Whomsoever It May Concern

Dear Sir/ Madam,

This is to confirm that Mr. Deepak Goel has completed his Training/ Internship Program with Cloud Analogy Softech Pvt. Ltd. and now he is working as a Salesforce Developer full time employee with us.

Please feel free to contact us if your organization should require any further information.

Sincerely,

DocuSigned by:

ED73D47D985E452


Divya Dang Jethi

Mail: divya.dang@cloudanalogy.com

(Head HR)

R Cloud Computing Solution Company

 A-17, Sector-63, Noida-201307

 + (0120) 414-7360

 SALESFORCE
TRUSTED CONSULTING
PARTNER

CERTIFICATE

Certified that **Deepak Goel (1802914002)** has carried out the project work presented in this report entitled “**HEALTHY WEALTHY CARE CENTER**” for the award of **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University, Lucknow under my supervision. The report embodies result of original work, and studies are carried out by the student himself and the contents of the report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University.

Dr. Sangeeta Arora

External Examiner

Associate Professor

Dept. of Computer Applications

KIET Group of Institutions, Ghaziabad

Dr. Ajay Kumar Srivastava

Professor & Head

Department of Computer Applications

KIET Group of Institutions, Ghaziabad

Date:

HEALTHY WEALTHY CARE CENTER

DEEPAK GOEL

ABSTRACT

Cloud Computing is a crowd/group of unknown resources that are giving for a specific purpose to the user. This CRM is intended for a particular client (Hospital) i.e., HEALTHY WEALTHY CARE CENTER to switch their current manual, paper-based system. The proposed new structure is to control the patient information, room availability, staff and operating room schedules, and patient invoices. These facilities are to be provided in a well-organized, cost effective manner, with the objective of reducing the time and resources currently required for such tasks.

An important part of the process of any hospital involves the acquirement, management and timely renewal of great volumes of information. This information usually involves; patient individual information and medicinal history, staff information, room and ward arrangement, staff arrangement, operating theatre scheduling and various facilities waiting lists. All of this data must be achieved in a capable and price wise manner so that an organization's properties may be effectually utilized HMS will automate the administration of the hospital making it more well-organized and error free.

All the service industries in modern world are highly dependent on the quality of the data, defining objectives from available data (historical data) and utilizing the same to achieve those objectives. Our mission here was to understand and gather the requirement for an enterprise level quality dashboard for a hospital client, so that the business and executives get man overall understanding of the Hospital management growth to help them take high level decisions based on the data represented on this Dashboard.

ACKNOWLEDGEMENT

Success in life is never attained single handedly. My deepest gratitude goes to my Project supervisor, **Dr. Sangeeta Arora, Associate Professor, Department of Computer Applications** for his guidance, help and encouragement throughout my research work. Their enlightening ideas, comments, and suggestions. Words are not enough to express my gratitude for his insightful comments and administrative help at various occasions. Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

Deepak Goel
1802914002

Table of Contents

Training Certificate	iii
Certificate	iv
Abstract	v
Acknowledge	vi
Table of content	vii
List of figures	viii
1 Introduction	
1.1 Project description	1
1.2 Project Scope	2
1.3 Hardware/Software Requirement Specification	
1.3.1 Introduction	3
1.3.2 Benefit of cloud apps	4
1.3.3 How cloud apps work	7
2 Feasibility Study	
2.1 Technical feasibility	10
2.2 Economical Feasibility	11
2.3 Operational Feasibility	12
3 Design & Code	
3.1 Design Data Model	13
3.2 Screenshots	16
3.3 Source Code	21
4 Testing	65
5 Conclusion	67
6 Bibliography	68
References	69

Lists of Figures

1.1 Benefits of Clouds	8
2.1 Data Model	14
2.2 Data Model	15
3.1 SMS Sender	16
3.2 Attachment Meen (Gallery Component)	17
3.3 Gallery Component	18
3.4 Email Tab	19
3.5 Attachment Menu (Email Tab)	20

Chapter – 1

INTRODUCTION

1.1PROJECT DESCRIPTION

For a huge organization like HEALTHY WEALTHY CARE CENTER which holds the data of many diverse fields, it is a backbreaker for the higher personnel to monitor petty level tasks to transaction level tasks. Heedlessness in monitoring the progress and funds usage is depreciating organizations discipline and advancement

There should be bridge of trust between hospital and customer.

Our client HEALTHY WEALTHY CARE CENTER want to maintain good relationship with their customers. So, we propose a Customer Relationship Management (CRM) System. Organizations want to attract new patients or maintain repeating patients so they use a CRM to manage the relationships with the public. This might be task like sending check-up appointment reminders. The CRM system enables organisation to get essential customer information and use it as efficiently as possible. This CRM plans a number of methodologies in a synchronized approach to deliver healthcare. This system maintains details of both in-patient and out-patient. In Patient and Out-Patient Details will be shown in effective graphical user interface and other additional technical aspects will also be integrated along with this module. This system will help many health care sectors to maintain good customer relationship. This system manages hospital's current and future customers. Hospitals will keep track of their current and future customers. Manually it is not possible for the hospital admin to keep records of all the patients. Customer Relationship Management System helps to keep track of the appointments made by the customer.

1.2 PROJECT SCOPE

The recent rapid increase in the amount of medical information has pushed hospitals to confront an essential issue which is how to utilize healthcare information technology to improve healthcare services quality. Customer relationship management system (CRMS) is an innovative technology which facilitates the process to acquire, develop, and maintain customer relationships more efficiently and effectively. Customer relationship management (CRM) for healthcare providers is an approach to learn all they can about their customers and prospects, to communicate relevant, timely information to them, and to track results to make program adjustments necessary. The rapid increase in the amount of medical information has pushed hospitals to confront a critical issue, which is how to utilize information technologies to manage large amounts of customer information and then improve the quality of customer services. The adoption of a customer relationship management system (CRMS) thus is increased globally among hospitals. The percentage of hospitals which utilize Web sites for sales and marketing purposes has increased 2.47 times from 1995 (17%) to 2000 (59%) in the US. Also, customer satisfaction is measured by healthcare providers as a key factor of strategy and an important determinant of long-term feasibility and success under competitive situation. In addition, maintaining and increasing customer loyalty level is essential for any service company's long-term success. Creating a conceptual framework for the CRM health care system using multivariate measurement system such as factor analysis or principal component analysis causes improvement in business by increasing the level of customer satisfaction and customer loyalty.

1.3 HARDWARE/SOFTWARE USED IN THIS PROJECT

1.3.1 Introduction

Information technology (IT) is the use of any computers, storage, networking and other physical devices, infrastructure and processes to create, process, store, secure and exchange all forms of electronic data. Typically, IT is used in the context of business operations, as opposed to technology used for personal or entertainment purposes. The commercial use of IT encompasses both computer technology and telecommunications. [14]

As this is a cloud-based application so there is no need of sophisticated hardware only there is a need of Browser application and Internet connection. Cloud provider provides the development infrastructure to create application.

A cloud application, or cloud app, is a software program where cloud-based and local components work together. This model relies on remote servers for processing logic that is accessed through a web browser with a continual internet connection.

Cloud development involves, well, developing the cloud. ... This involves developing the cloud architecture such as planning, organizing, and designing to implementing and structuring cloud delivery models (IaaS, PaaS, SaaS). Additional tasks in cloud development include managing the cloud service delivery models.[3]

Cloud application servers typically are located in a remote data center operated by a third-party cloud services infrastructure provider. Cloud-based application tasks may encompass email, file storage and sharing, order entry, inventory management, word processing, customer relationship management (CRM), data collection, or financial accounting features.

There are many CRM software choices available with a wide range of features and add-on integrations. While it is great to have so many options, it can be difficult to determine which CRM software is the best fit for your business. The key is to examine your goals and objectives for the project and how these align with your overall business goals; determine what your user base needs the software to do; assess what features are most important to your business; evaluate the integration potential of the software; make sure that any solution you select is mobile-friendly; and resist the urge to build a custom solution. While this process takes time and resources, a thorough evaluation will help you select a system that yields maximum ROI.[2]

1.3.2 Benefits of cloud apps

Fast response to business needs. Cloud applications can be updated, tested and deployed quickly, providing enterprises with fast time to market and agility. This speed can lead to culture shifts in business operations.

Simplified operation. Infrastructure management can be outsourced to third-party cloud providers.

Instant scalability. As demand rises or falls, available capacity can be adjusted.

API use. Third-party data sources and storage services can be accessed with an application programming interface (API). Cloud applications can be kept smaller by using APIs to hand data to applications or API-based back-end services for processing or analytics computations, with the results handed back to the cloud application. Vetted APIs impose passive consistency that can speed development and yield predictable results.

Gradual adoption. Refactoring legacy, on-premises applications to a cloud architecture in steps, allows components to be implemented on a gradual basis.

Reduced costs. The size and scale of data centers run by major cloud infrastructure and service providers, along with competition among providers, has led to lower prices. Cloud-based applications can be less expensive to operate and maintain than equivalents on-premises installation.

Improved data sharing and security. Data stored on cloud services is instantly available to authorized users. Due to their massive scale, cloud providers can hire world-class security experts and implement infrastructure security measures that typically only large enterprises can obtain. Centralized data managed by IT operations personnel is more easily backed up on a regular schedule and restored should disaster recovery become necessary.

Cloud computing strategies will undoubtedly continue to be a huge part of every organization's network strategy in the coming years. With scalable power and flexible services, cloud providers give companies the tools they need to drive better business results. Data centers have a key role to play in building these infrastructures, making it critical that IT professionals keep a close eye on the latest developments in these interconnected industries.[1]

cloud application, or cloud app, is a software program where cloud-based and local components work together. This model relies on remote servers for processing logic that is accessed through a web browser with a continual internet connection.

Cloud application servers typically are located in a remote data center operated by a third-party cloud services infrastructure provider. Cloud-based application tasks may encompass email, file storage and sharing, order entry, inventory management, word processing, customer relationship management (CRM), data collection, or financial accounting features.

Fast response to business needs: Cloud applications can be updated, tested and deployed quickly, providing enterprises with fast time to market and agility. This speed can lead to culture shifts in business operations.

Simplified operation: Infrastructure management can be outsourced to third-party cloud providers.

Instant scalability: As demand rises or falls, available capacity can be adjusted.

API use: Third-party data sources and storage services can be accessed with an application programming interface (API). Cloud applications can be kept smaller by using APIs to hand data to applications or API-based back-end services for processing or analytics computations, with the results handed back to the cloud application. Vetted APIs impose passive consistency that can speed development and yield predictable results.

Gradual adoption: Refactoring legacy, on-premises applications to a cloud architecture in steps, allows components to be implemented on a gradual basis.

Reduced costs: The size and scale of data centers run by major cloud infrastructure and service providers, along with competition among providers, has led to lower prices.

Cloud-based applications can be less expensive to operate and maintain than equivalents on-premises installation.

Improved data sharing and security: Data stored on cloud services is instantly available to authorized users. Due to their massive scale, cloud providers can hire world-class security experts and implement infrastructure security measures that typically only large enterprises can obtain. Centralized data managed by IT operations personnel is more easily backed up on a regular schedule and restored should disaster recovery become necessary.

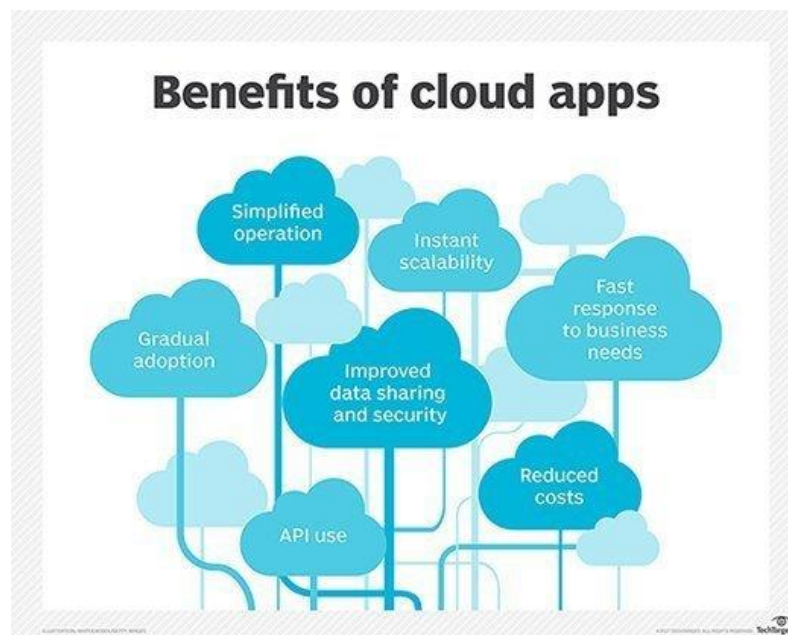


Fig 1.1 Benefits of Clouds

1.3.3 How cloud apps work

Data is stored and compute cycles occur in a remote data center typically operated by a third-party company. A back end ensures uptime, security and integration and supports multiple access methods.

Cloud applications provide quick responsiveness and don't need to permanently reside on the local device. They can function offline, but can be updated online.

While under constant control, cloud applications don't always consume storage space on a computer or communications device. Assuming a reasonably fast internet connection, a well-written cloud application offers all the interactivity of a desktop application, along with the portability of a web application.

With the advancement of remote computing technology, clear lines between cloud and web applications have blurred. The term cloud application has gained great cachet, sometimes leading application vendors with any online aspect to brand them as cloud applications.

Cloud and web applications access data residing on distant storage. Both use server processing power that may be located on premises or in a distant data center.

A key difference between cloud and web applications is architecture. A web application or web-based application must have a continuous internet connection to function. Conversely, a cloud application or cloud-based application performs processing tasks on a local computer or workstation. An internet connection is required primarily for downloading or uploading data.

A web application is unusable if the remote server is unavailable. If the remote server becomes unavailable in a cloud application, the software installed on the local user device can still operate, although it cannot upload and download data until service at the remote server is restored.

The difference between cloud and web applications can be illustrated with two common productivity tools, email and word processing. Gmail, for example, is a web application that requires only a browser and internet connection. Through the browser, it's possible to open, write and organize messages using search and sort capabilities. All processing logic occurs on the servers of the service provider (Google, in this example) via either the internet's HTTP or HTTPS protocols.

A CRM application accessed through a browser under a fee-based software as a service (SaaS) arrangement is a web application. Online banking and daily crossword puzzles are also considered web applications that don't install software locally.

An example of a word-processing cloud application that is installed on a workstation is Word's Microsoft Office 365. The application performs tasks locally on a machine without an internet connection. The cloud aspect comes into play when users save work to an Office 365 cloud server.

Chapter – 2

Feasibility Study

A feasibility study is a preliminary study which investigates the information of prospective users and determines the resources requirements, costs, benefits and feasibility of proposed system. A feasibility study takes into account various constraints within which the system should be implemented and operated. In this stage, the resource needed for the implementation such as computing equipment, manpower and costs are estimated. The estimated are compared with available resources and a cost benefit analysis of the system is made. The feasibility analysis activity involves the analysis of the problem and collection of all relevant information relating to the project. The main objectives of the feasibility study are to determine whether the project would be feasible in terms of economic feasibility, technical feasibility and operational feasibility and schedule feasibility or not. It is to make sure that the input data which are required for the project are available. Thus, we evaluated the feasibility of the system in terms of the following categories:

- **Technical feasibility**
- **Operational feasibility**
- **Economical feasibility**

2.1 Technical Feasibility:

Evaluating the technical feasibility is the trickiest part of a feasibility study. This is because, at the point in time there is no any detailed designed of the system, making it difficult to access issues like performance, costs (on account of the kind of technology to be deployed) etc. A number of issues have to be considered while doing a technical

analysis; understand the different technologies involved in the proposed system. Before commencing the project, we have to be very clear about what are the technologies that are to be required for the development of the new system. Is the required technology available? HR's register is technically feasible.

2.2 Operational Feasibility:

Operational feasibility is the measure of how well a proposed system solves the problems, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development.

The operational feasibility assessment focuses on the degree to which the proposed development project fits in with the existing business environment and objectives with regard to development schedule, delivery date, corporate culture and existing business processes.

To ensure success, desired operational outcomes must be imparted during design and development. These include such design-dependent parameters as reliability, maintainability, supportability, usability, producibility, disposability, sustainability, affordability and others. These parameters are required to be considered at the early stages of design if desired operational behaviours are to be realised. A system design and development require appropriate and timely application of engineering and management efforts to meet the previously mentioned parameters. A system may serve its intended purpose most effectively when its technical and operating characteristics are engineered into the design. Therefore, operational feasibility is a critical aspect of systems engineering that needs to be an integral part of the early design phases We have created

the project which is operational feasible too. And this project will meet the needs by completing the project.

2.3 Economical Feasibility:

Our project is economical feasible. Economic feasibility attempts to weigh the costs of developing and implementing a new system, against the benefits that would accrue from having the new system in place. This feasibility study gives the top management the economic justification for the new system. A simple economic analysis which gives the actual comparison of costs and benefits are much more meaningful in this case. In addition, this proves to be useful point of reference to compare actual costs as the project progresses. There could be various types of intangible benefits on account of automation. These could increase customer satisfaction, improvement in product quality, better decision making, and timeliness of information, expediting activities, improved accuracy of operations, better documentation and record keeping, faster retrieval of information, better employee morale.

Chapter – 3

Design & Coding

3.1 Design Data Model

Schema Builder provides a dynamic environment for viewing and modifying all the objects and relationships in your app. This greatly simplifies the task of designing, implementing, and modifying your data model, or schema. Schema Builder is enabled by default. We can view our existing schema and interactively add new custom objects, custom fields, and relationships, simply by dragging and dropping. Schema Builder automatically implements the changes and saves the layout of our schema any time you move an object. This eliminates the need to click from page to page to find the details of a relationship or to add a new custom field to an object in our schema. Schema Builder provides details like the field values, required fields, and how objects are related by displaying lookup and master-detail relationships. We can view the fields and relationships for both standard and custom objects.

Schema Builder lets you add the following to your schema:

- Custom objects
- Lookup relationships (through blue line)
- Master-detail relationships (through purple line)
- All custom fields except: Geolocation

Here is our project's Data model created with the help of Schema Builder –

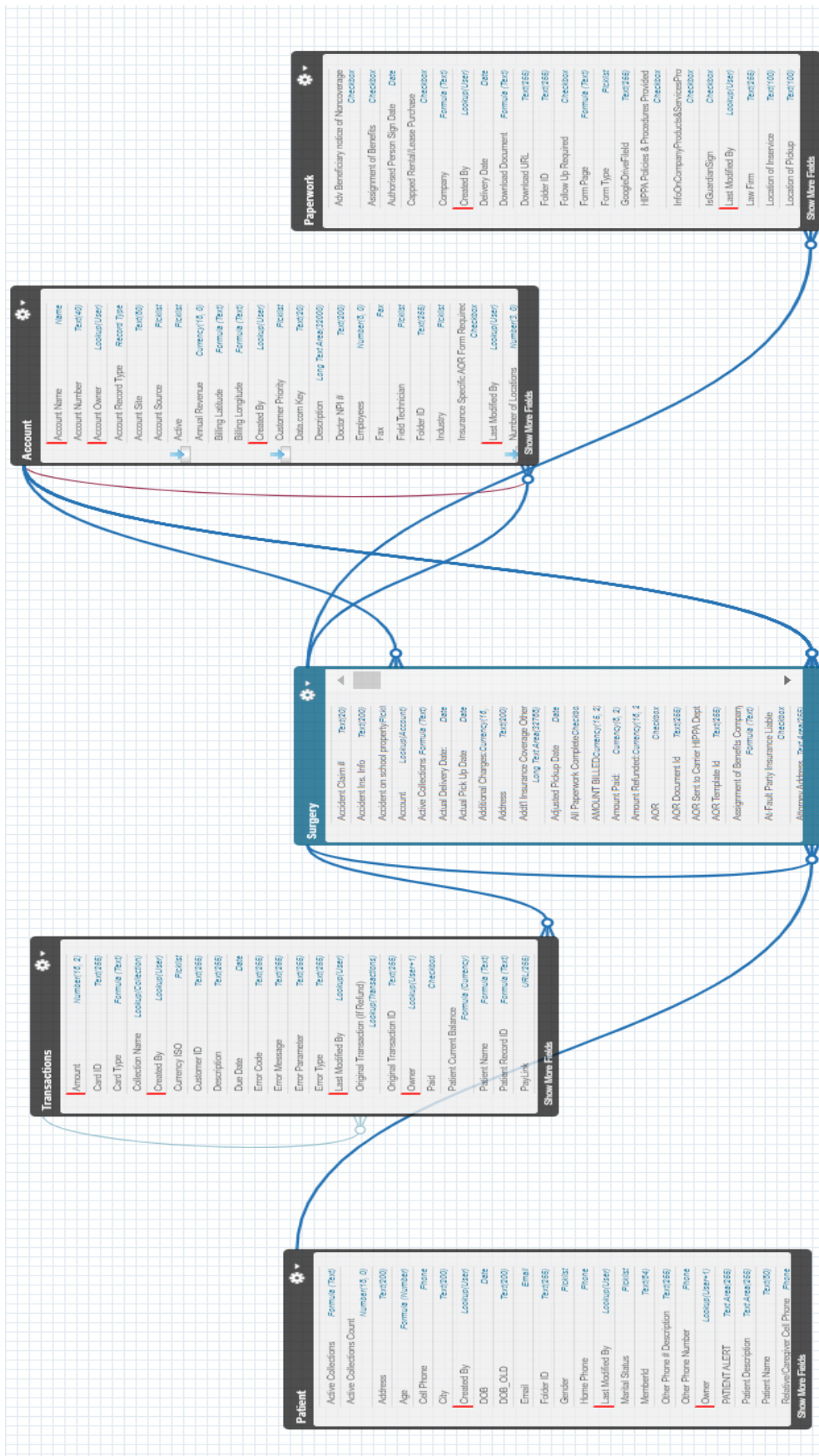


Fig 2.1 Data Model



Fig 2.2 Data Model

3.2 Screenshots

3.2.1 SMS Sender Component

SMS sender component was developed on client's requirements through which operator can send any type of information like reminder of appointment, confirmation, cancellation, etc on patient's personal number, his family or on office number. This component is developed in Lightning Web Component as it is light in weight and fast in accessing. In this component we integrate with "Twilio" third party app to send messages. This component can send message to numbers present in the details of patient. Send button can be used to send any message after selecting target receiver and will show bell notification whenever there will be any incoming message for us.

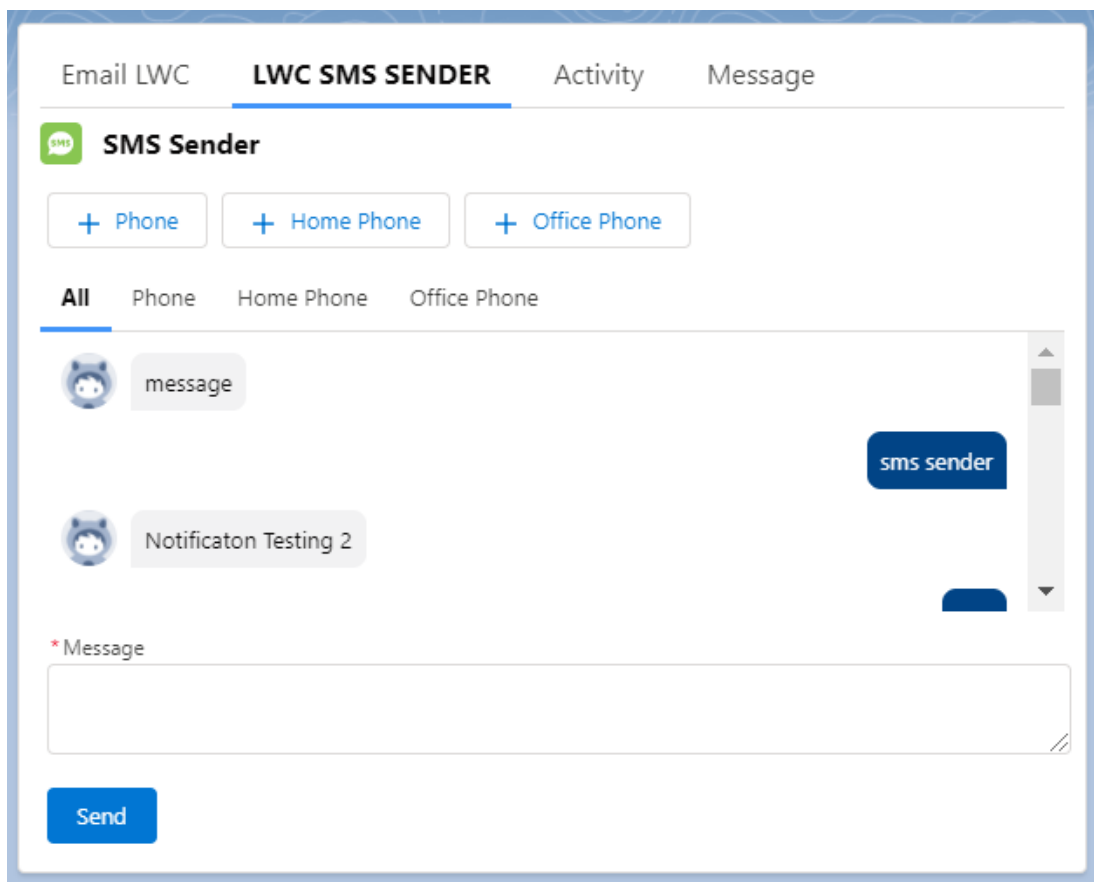


Fig 3.1 SMS Sender

3.2.2 Gallery Component

Gallery component is aura component developed to showcase all the necessary documents, reports, etc in the form of images. Although a Notes and Attachment related list is present in salesforce on every record page but that have some limitations that's why Gallery Component was developed. In this component we can set priority of images according to importance.

In this component there are two main parts i.e., Upload Image button and Home Page. Upload Image button pops up a modal where we can upload new images, rearrange priorities and can delete images as shown in Fig 3.2.

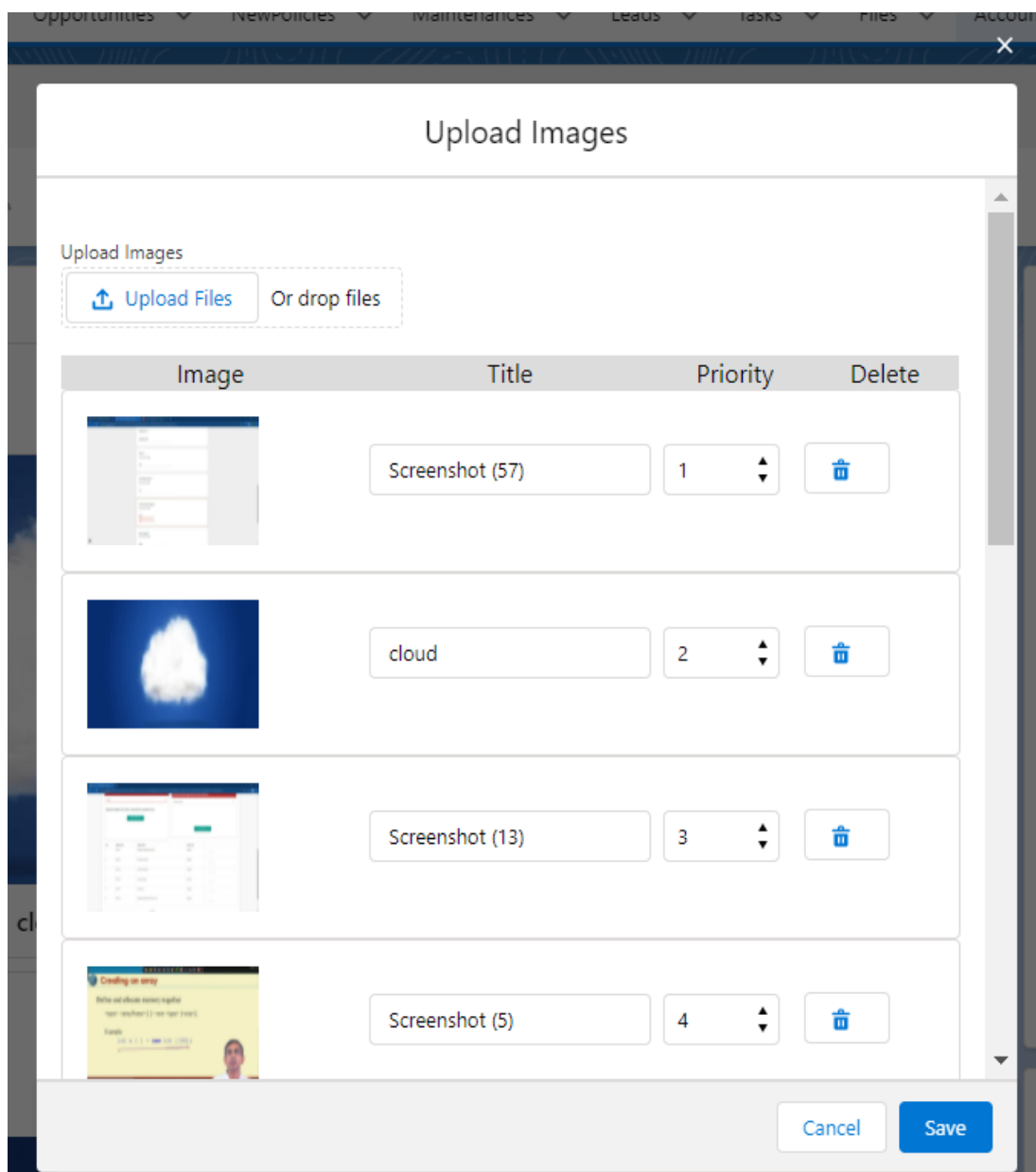


Fig 3.2. Attachment Menu

This is the home page (Fig 3.3) of Gallery component where highest priority image is shown in Image carousel and other images presents in the form of scrollable list. We can show any image on Image carousel by just clicking on the image. This will not affect the priority of images.

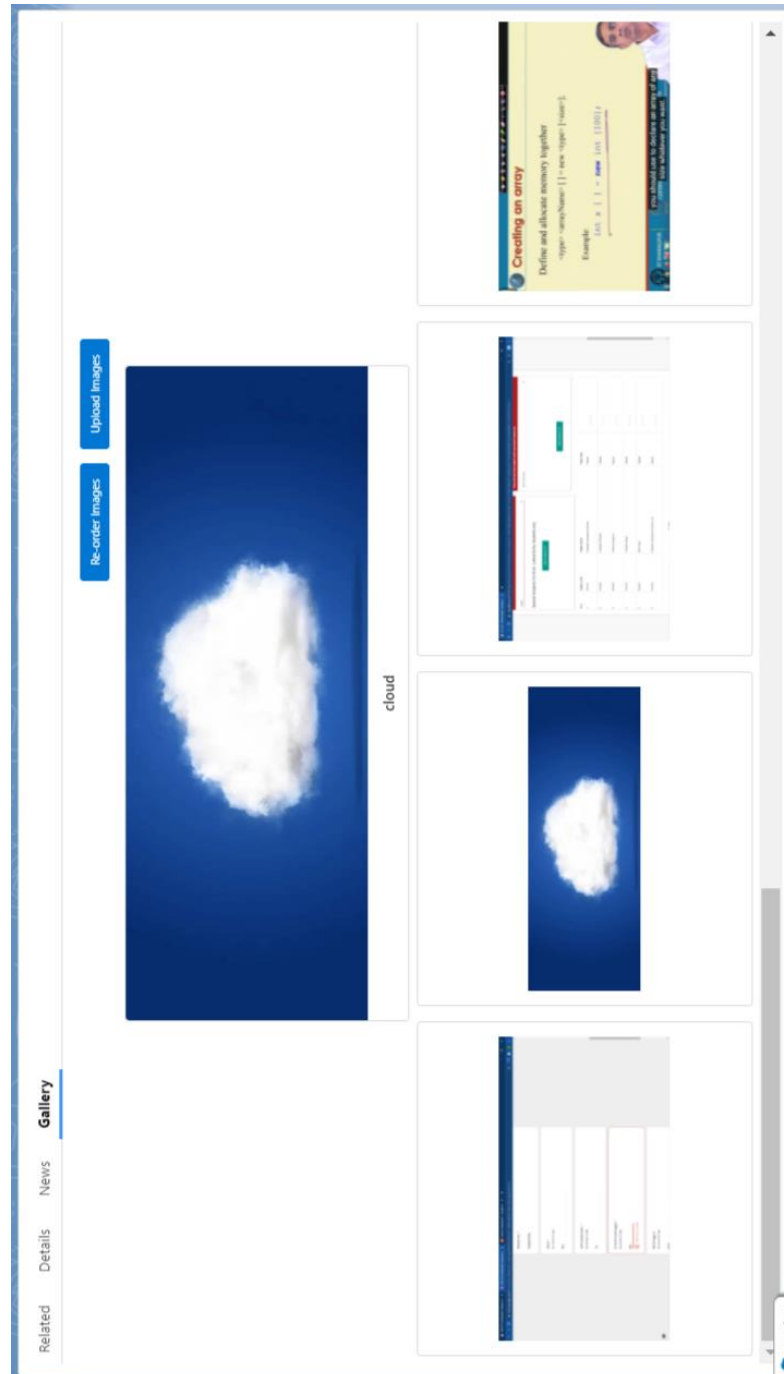


Fig 3.3 Gallery Component

3.2.3 Email Sender Component

Email component is a replica of standard Email tab present in salesforce. To remove the limitations of standard Email Sender application of Salesforce, this component was developed. This component can be used to send mails to patient, his family, doctors, staff, etc with formatted text and attachments. This component includes all the facilities like To, Cc, Bcc, Subject, etc as shown in Fig. 3.4.

The screenshot displays the 'Email' tab within a Salesforce interface. At the top, there are navigation tabs: 'New Task', 'Log a Call', 'New Event', and 'Email'. The 'Email' tab is selected. Below the tabs, the email composition form is visible. It includes fields for 'From' (pre-filled with 'Deepak Goel <deepak.goel@cloudanalog.com>'), 'To', 'Bcc', and 'Subject' (with a placeholder 'Enter Subject...'). Below these fields is a rich text editor toolbar with options for font face ('Salesforce Sans'), font size ('12'), and various text formatting options (Bold, Italic, Underline, Strikethrough, Bulleted List, Numbered List, Indent, Outdent, Link). The main text area contains the text 'Powered by Salesforce' followed by the URL 'http://www.salesforce.com/'. Below the text area is an attachment icon (a paperclip). At the bottom, there is a 'RelatedTo:' section with a dropdown menu and a search bar, and a prominent blue 'Send' button.

Fig 3.4 Email Tab

Fig 3.5 refers to the attachment menu of Email Components from which we can choose attachments to send in email body. This part of component is present in the form of modal pop up and enables user to upload new attachment files, search file from existing files, choose multiple files. Selected file will be present on home page in the form of pills.

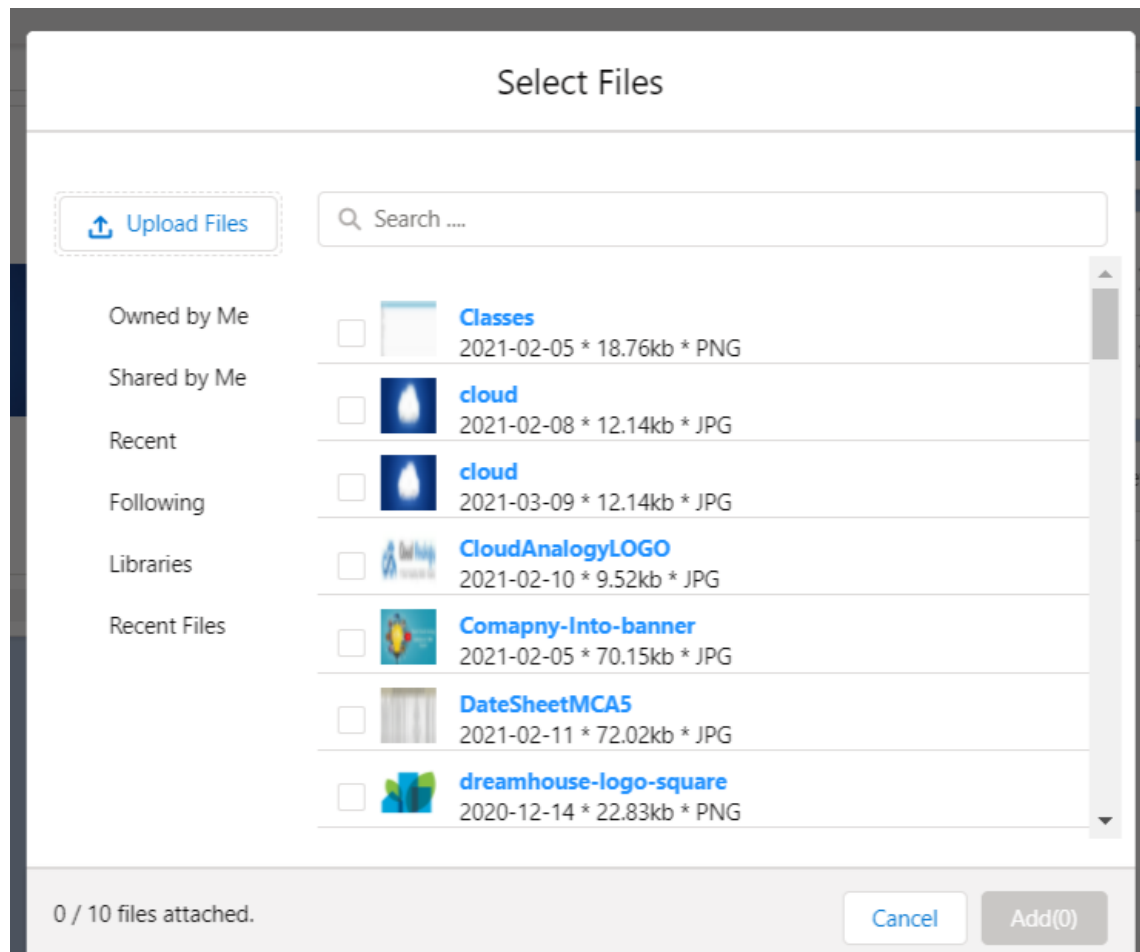


Fig 3.5 Attachment Menu

3.3 Source Code

3.3.1 Gallery Component

Component

```
<aura:component controller="GalleryController"
implements="flexipage:availableForAllPageTypes,force:hasRecordId">

    <aura:attribute name="imageList" type="List"/>
    <aura:attribute name="selectedImage" type="Object"/>
    <aura:attribute name="imageCount" type="List"/>
    <aura:attribute name="modalTitle" type="String"/>
    <aura:attribute name="filetype" type="List" default="['.png', '.jpg',
'.jpeg']"/>
    <aura:attribute name="recordId" type="String"/>
    <aura:attribute name="deleteIndex" type="Integer"/>
    <aura:attribute name="newlyAddedImages" type="List"/>

    <aura:attribute name="isModalOpen" type="boolean" default="false"/>
    <aura:attribute name="deleteImage" type="boolean" default="false"/>

    <aura:handler name="init" value="{!this}" action="{!c.doInit}"/>

    <aura:html tag="style">
        .slds-modal__container {
            width: 50%;
            height: 100%;
        }
    </aura:html>

    <div>
        <lightning:layout>
            <lightning:layoutItem size="7"></lightning:layoutItem>
            <lightning:layoutItem size="5">
                <lightning:button variant="brand" class="slds-m-top_small slds-m-
right_small" label="Re-order Images"
                    title="Brand action" onclick="{! c.reorder}"/>
                <lightning:button variant="brand" class="slds-m-top_small slds-m-
right_small" label="Upload Images"
                    title="Brand action" onclick="{! c.delete }"/>
            </lightning:layoutItem>
        </lightning:layout>
        <lightning:layout class="imageArea" horizontalAlign="center">
            <!-- <lightning:layoutItem size="2"></lightning:layoutItem> -->
            <lightning:layoutItem class="slds-m-top_medium">

                <lightning:carouselImage

src="{!'/sfc/servlet.shepherd/version/renditionDownload?rendition=THUMB720BY48
0&amp;versionId='+v.selectedImage.imageVersionId}"
                    header="{!v.selectedImage.Title}"
                    alternativeText="{!v.selectedImage.Title}"/>

            </lightning:layoutItem>
        </lightning:layout>
        <lightning:layout class="slds-scrollable_x">
            <aura:iteration items="{!v.imageList}" var="image"
indexVar="imageIndex">
```

```

        <lightning:layoutItem size="3" class="slds-m-around_x-small slds-box
slds-align_absolute-center">
            <div onclick="{!c.selectImage}" id="{!imageIndex}" class="">
                
            </div>
        </lightning:layoutItem>
    </aura:iteration>
</lightning:layout>
</div>

<aura:if isTrue="{!v.isModalOpen}">
    <!-- Modal/Popup Box starts here-->
    <section role="dialog" tabIndex="-1" aria-labelledby="modal-heading-01"
aria-modal="true"
        aria-describedby="modal-content-id-1" class="slds-modal slds-fade-in-
open">
        <div class="slds-modal__container">
            <!-- Modal/Popup Box Header Starts here-->
            <header class="slds-modal__header">
                <lightning:buttonIcon iconName="utility:close"
                    onclick="{! c.closeModel }"
                    alternativeText="close"
                    variant="bare-inverse"
                    class="slds-modal__close"/>
                <h2 id="modal-heading-01" class="slds-text-heading_medium slds-
hyphenate">{!v.modalTitle}</h2>
            </header>
            <!-- Modal/Popup Box Body Starts here -->
            <div class="slds-modal__content slds-p-around_medium slds-
scrollable_y"
                style="height:21rem;max-width:40rem"
                id="modal-content-id-1">
                <lightning:notificationsLibrary aura:id="modal"/>
                <aura:if isTrue="{!v.modalTitle == 'Upload Images'}">
                    <lightning:fileUpload label="Upload Images"
                        multiple="False"
                        accept="{!v.filetype}"
                        recordId="{!v.recordId}"
                        onuploadfinished="{!c.handleUploadFinished}"
                        class="slds-p-around_large"/>
                </aura:if>
                <lightning:layout multipleRows="true">
                    <lightning:layoutItem size="4">
                        <p style="text-align:center;background-color:Gainsboro;font-
size:medium;">Image</p>
                    </lightning:layoutItem>
                    <lightning:layoutItem size="4">
                        <p style="text-align:center;background-color:Gainsboro;font-
size:medium;">Title</p>
                    </lightning:layoutItem>
                    <lightning:layoutItem size="2">
                        <p style="text-align:center;background-color:Gainsboro;font-
size:medium;">Priority</p>
                    </lightning:layoutItem>
                    <lightning:layoutItem size="2">
                        <p style="text-align:center;background-color:Gainsboro;font-
size:medium;">Delete</p>
                    </lightning:layoutItem>
                </lightning:layout>
            </div>
        </div>
    </section>
</aura:if>

```

```

        </lightning:layout>

        <aura:iteration items="{!v.imageList}" var="image"
indexVar="imageIndex">
            <lightning:layout class="slds-box">
                <lightning:layoutItem size="4">
                    
                </lightning:layoutItem>
                <lightning:layoutItem size="4">
                    <lightning:input name="Image Name" value="{!image.Title}"/>
                </lightning:layoutItem>
                <lightning:layoutItem size="2">
                    <lightning:select aura:id="Priority" class="slds-m-
horizontal_x-small"
                        onchange="{!c.changePriority}">
                        <aura:iteration items="{!v.imageCount}" var="count"
indexVar="index">
                            <aura:if isTrue="{!image.priority == count}">
                                <option value="{!imageIndex+'_'+count}"
selected="true">{!count}</option>
                                <aura:set attribute="else">
                                    <option
value="{!imageIndex+'_'+count}">{!count}</option>
                                </aura:set>
                                </aura:if>
                            </aura:iteration>
                        </lightning:select>
                    </lightning:layoutItem>
                    <lightning:layoutItem size="2">
                        <lightning:button iconName="utility:delete"
                            class="slds-m-horizontal_x-small slds-m-top_medium"
                            onclick="{!c.onDeleteDetails}"
value="{!imageIndex}"/>
                    </lightning:layoutItem>
                </lightning:layout>
            </aura:iteration>

        </div>
        <!-- Modal/Popup Box Footer Starts here -->
        <footer class="slds-modal__footer">
            <lightning:button variant="neutral"
                label="Cancel"
                title="Cancel"
                onclick="{! c.closeModel }"/>
            <lightning:button variant="brand"
                label="Save"
                title="Save"
                onclick="{!c.onSave}"/>
        </footer>
    </div>
</section>
<div class="slds-backdrop slds-backdrop_open"></div>
</aura:if>

<!-- Delete confirmaton modal -->
<aura:if isTrue="{!v.deleteImage}">
    <!-- Modal/Popup Box starts here-->

```

```

    <section role="dialog" tabindex="-1" aria-labelledby="modal-heading-01"
    aria-modal="true"
        aria-describedby="modal-content-id-1" class="slds-modal slds-fade-in-
open">
    <div class="slds-modal__container">
        <!-- Modal/Popup Box Header Starts here-->
        <header class="slds-modal__header">
            <lightning:buttonIcon iconName="utility:close"
                onclick="{! c.closeModel1 }"
                alternativeText="close"
                variant="bare-inverse"
                class="slds-modal__close"/>
            <h2 id="modal-heading-01" class="slds-text-heading_medium slds-
hyphenate">Delete</h2>
        </header>
        <div style="background-color:white;">
            <p class="slds-m-vertical_large" style="display: flex;justify-
content: center;">Do you really want
                to delete selected image?</p>
        </div>
        <footer class="slds-modal__footer">
            <lightning:button variant="neutral"
                label="Cancel"
                title="Cancel"
                onclick="{! c.closeModel1 }"/>
            <lightning:button variant="brand"
                label="Confirm"
                title="Confirm"
                onclick="{!c.onDelete}"/>
        </footer>
    </div>
</section>
<div class="slds-backdrop slds-backdrop_open"></div>
</aura:if>
</aura:component>

```

JavaScript Controller

```

({
    doInit : function(component, event, helper) {
        helper.doInit_helper(component, event, helper);
    },

    onSave : function(component, event, helper) {
        try{
            var action = component.get("c.updatePriority");
            var updatedData = component.get("v.imageList");
            var count = 1;
            for(var i=0; i<updatedData.length; i++){
                for(var j=i+1; j<updatedData.length; j++){
                    if(updatedData[i].priority == updatedData[j].priority){
                        count=0;
                        break;
                    }
                }
            }
            if(count==0){

```



```

        component.find('modal').showNotice({
            "variant": "error",
            "header": "Something has gone wrong!",
            "message": "Two or more priorities are same!",
        });
    }
    else{
        action.setParams({
            "updatedData" : JSON.stringify(updatedData)
        })
        action.setCallback(this, function(a) {
            var returnValue = a.getReturnValue();
            if(!$A.util.isEmpty(returnValue)){
                if(returnValue == 'Working'){
                    console.log('working');
                    component.set("v.isModalOpen", false);
                    console.log(updatedData.length);
                }
                else
                    console.log('error from apex');
            }
            else
                console.log('no response received');
        })
        $A.enqueueAction(action);
        helper.doInit_helper(component, event, helper);
    }
}
catch(err){
    console.log(err);
}
},

changePriority : function(component, event, helper) {
    try{
        var input = event.getSource().get("v.value");
        var splitInput = input.split("_");
        var imageList = component.get("v.imageList");
        imageList[splitInput[0]].priority = splitInput[1];
    }
    catch(err){
        console.log(err);
    }
},

selectImage : function(component, event, helper) {
    try{
        var imageList = component.get("v.imageList");
        var check = event.currentTarget.id;
        component.set("v.selectedImage",imageList[check]);
    }
    catch(err){
        console.log(err);
    }
},

openModel: function(component, event, helper) {
    // Set isModalOpen attribute to true

```

```

        component.set("v.isModalOpen", true);
    },

    closeModal: function(component, event, helper) {
    try{
        var modalTitle = component.get("v.modalTitle");
        if(modalTitle == 'Upload Images'){
            var imageDeleteList = component.get("v.newlyAddedImages");
            var action = component.get("c.cancelUpload");
            action.setParams({
                "imageDetails" : imageDeleteList
            })
            action.setCallback(this, function(a) {
                var returnValue = a.getReturnValue();
                if(!$.util.isEmpty(returnValue)){
                    if(returnValue == 'Success'){
                        console.log('deleted');
                        helper.doInit_helper(component, event, helper);
                    }
                    else
                        console.log('error from apex');
                }
                else
                    console.log('no response received');
            })
            $.enqueueAction(action);
        }
        //helper.doInit_helper(component, event, helper);
        //$A.get('e.force:refreshView').fire();
        component.set("v.isModalOpen",false);
    }
    catch(err){
        console.log(err);
    }
    },

    reorder : function(component, event, helper) {
        try{
            component.set("v.isModalOpen", true);
            component.set("v.modalTitle",'Re-Order Images');
        }
        catch(err){
            console.log(err);
        }
    },

    delete : function(component, event, helper) {
        try{
            console.log(component.get("v.imageList"));
            var obj = component.get('v.newlyAddedImages');
            obj.length = 0;
            component.set("v.newlyAddedImages",obj);
            component.set("v.isModalOpen", true);
            component.set("v.modalTitle",'Upload Images');
        }
        catch(err){
            console.log(err);
        }
    },

    handleUploadFinished: function(component, event, helper) {
    try{
        var imageList = component.get("v.imageList");

```

```

var priority = imageUrl.length;
var newlyAddedImages = component.get("v.newlyAddedImages");
var uploadedFiles = event.getParam("files");
newlyAddedImages.push(uploadedFiles[0].documentId);
console.log(uploadedFiles[0].contentVersionId);
var action = component.get("c.cloudinaryIntegration");
action.setParams({
    "picture" : uploadedFiles[0].contentVersionId,
    "priority" : priority+1
})
action.setCallback(this, function(a) {
    var returnValue = a.getReturnValue();
    if(!$A.util.isEmpty(returnValue)){
        if(returnValue == 'Success'){
            console.log('working');
            console.log(component.get("v.newlyAddedImages"));
            helper.doInit_helper(component, event, helper);
        }
        else
            console.log('error from apex');
    }
    else
        console.log('no response received');
})
    $A.enqueueAction(action);
}
catch(err){
    console.log(err);
}
},
onDeleteDetails : function(component, event, helper) {
    try{
        var check = event.getSource().get("v.value");
        component.set("v.deleteIndex",check);
        component.set("v.deleteImage",true);
        component.set("v.isModalOpen",false);
    }
    catch(err){
        console.log(err);
    }
},
onDelete : function(component, event, helper) {
    try{
        console.log('working');
        //var check = event.getSource().get("v.value");
        var check = component.get("v.deleteIndex");
        var imageUrl = component.get("v.imageUrl");
        var delImage = imageUrl[check];
        var action = component.get("c.deleteImage");
        var further_items = [];
        for(var i=check+1; i<imageUrl.length; i++){
            further_items.push(imageUrl[i].imageVersionId);
        }
        console.log(further_items);
        action.setParams({
            "details" : delImage.imageId,
            "cvIDs" : further_items
        })
        var returnValue='';
        action.setCallback(this, function(a) {
            returnValue = a.getReturnValue();

```

```

        if(!$A.util.isEmpty(returnValue)){
            if(returnValue == 'Success'){
                console.log('deleted');
                component.set("v.deleteImage",false);
                component.set("v.isModalOpen",true);
                helper.doInit_helper(component, event, helper);
                //console.log(check);
                //console.log(component.get("v.imageList").length);
                //helper.changePriority_onDeletefunction(component, event, helper,
check);
                //helper.doInit_helper(component, event, helper);
            }
            else
                console.log('error from apex');
        }
        else
            console.log('no response received');
    })
    $A.enqueueAction(action);
}
catch(err){
    console.log(err);
}
},
closeModel: function(component, event, helper) {
    component.set("v.deleteImage",false);
    component.set("v.isModalOpen",true);
}
})

```

JavaScript Helper

```

({
    doInit_helper : function(component, event, helper) {
        try{
            var action = component.get("c.fetchattachmentList");
            var recId = component.get("v.recordId");
            var imageCount = [];
            action.setParams({
                "recordId" : recId
            })
            action.setCallback(this, function(a) {
                var returnValue = a.getReturnValue();
                if(!$A.util.isEmpty(returnValue)){
                    component.set("v.imageList",returnValue);
                    component.set("v.recordId",recId);
                    component.set("v.selectedImage",returnValue[0]);
                    for(var i=1;i<=returnValue.length;i++)
                        imageCount.push(i);
                    component.set("v.imageCount",imageCount);
                }
            })
            $A.enqueueAction(action);
        }
        catch(err){
            console.log(err);
        }
    },

```

```

changePriority_onDeletefunction : function(component, event, helper,check){
try{
var imageList = component.get("v.imageList");
//console.log(imageList);
var action = component.get("c.updatePriority");
for(var i=check; i<imageList.length; i++){
    imageList[i].priority--;
}
//console.log(imageList);
action.setParams({
    "updatedData" : JSON.stringify(imageList)
})
action.setCallback(this, function(a) {
    var returnValue = a.getReturnValue();
    if(!$A.util.isEmpty(returnValue)){
        if(returnValue == 'Working'){
            console.log('working');
        }
        else
            console.log('error from apex');
    }
    else
        console.log('no response received');
    })
    $A.enqueueAction(action);
}
catch(err){
    console.log(err);
}
}
})

```

CSS

```

.THIS .imageArea{
    height:90%;
}
.THIS .slds-carousel__content{
    height:fit-content;
}

```

Apex Controller

```

public with sharing class GalleryController {
    @auraEnabled
    public static List<imageDetailsWrapper> fetchattachmentList(String
recordId){
    try{
        system.debug(recordId);
        List<ContentDocumentLink> cdItemsList = new List<ContentDocumentLink>();
        List<ContentVersion> contentVersionList = new List<ContentVersion>();
        List<imageDetailsWrapper> imageList = new List<imageDetailsWrapper>();
        Set<id> contentDocId = new Set<id>();
    }
}
}

```

```

        Map<Id,ContentVersion> conDocIdVsConVersId = new
Map<Id,ContentVersion>();
        cdlItemsList = [SELECT ContentDocumentId,ContentDocument.Title FROM
ContentDocumentLink WHERE LinkedEntityId=: recordId LIMIT 100];
        for(ContentDocumentLink cdlobj : cdlItemsList)
            contentDocId.add(cdlobj.ContentDocumentId);
        contentVersionList = [SELECT
Id,ContentDocumentId,Priority__c,Cloudinary_Link__c FROM ContentVersion WHERE
ContentDocumentId =: contentDocId];
        for(ContentVersion cvObj : contentVersionList){
            if(!(conDocIdVsConVersId.containsKey(cvObj.ContentDocumentId))){
                conDocIdVsConVersId.put(cvObj.ContentDocumentId,cvObj);
            }
        }
        for(ContentDocumentLink cdlobj : cdlItemsList)
            imageList.add(new
imageDetailsWrapper(cdlobj.ContentDocumentId,cdlobj.ContentDocument.Title,conD
ocIdVsConVersId.get(cdlobj.ContentDocumentId).Id,conDocIdVsConVersId.get(cdlob
j.ContentDocumentId).Priority__c,conDocIdVsConVersId.get(cdlobj.ContentDocumen
tId).Cloudinary_Link__c));
        for(integer i=0;i<imageList.Size();i++){
            imageDetailsWrapper temp;
            for(integer j=i+1;j<imageList.Size();j++){
                if(imageList[i].priority > imageList[j].priority){
                    temp = imageList[i];
                    imageList[i] = imageList[j];
                    imageList[j] = temp;
                }
            }
        }
        return imageList;
    }
    catch(Exception e){
        System.debug('Error is ::: '+e.getMessage()+' at line no. :::
'+e.getLineNumber());
        return null;
    }
}

@AuraEnabled
public static String updatePriority(String updatedData){
    List<imageDetailsWrapper> updatedWrapData =
(List<imageDetailsWrapper>)JSON.deSerialize(updatedData,List<imageDetailsWrapp
er>.class);
    //updatedWrapData =
JSON.deSerialize(updatedData,List<imageDetailsWrapper>.class);
    try{
        List<ContentVersion> dataToUpdate = new List<ContentVersion>();
        for(imageDetailsWrapper idWObj : updatedWrapData ){
            system.debug(idWObj);
            ContentVersion cvObj = new ContentVersion();
            cvObj.Id = idWObj.imageVersionId;
            cvObj.Title = idWObj.Title;
            cvObj.Priority__c = idWObj.priority;
            dataToUpdate.add(cvObj);
        }
        update dataToUpdate;
        return 'Working';
    }
    catch(Exception e){
        System.debug('Error is ::: '+e.getMessage()+' at line no. :::

```

```

    +e.getLineNumber());
    return Null;
}
}
@AuraEnabled
public static String cloudinaryIntegration(String picture,Integer priority){
    try{
        system.debug(picture);
        ContentVersion cvObj = new ContentVersion();
        cvObj = [SELECT Title,VersionData FROM ContentVersion WHERE Id =: picture
LIMIT 1];
        system.debug(cvObj);
        String cloudName='deepakcloudanalogy';
        String apikey='421997831572282';
        String apisecret='WsYZr9tjI4Xw0dpxIaCh1_qDv88';
        Http h = new Http();
        // Instantiate a new HTTP request, specify the method (POST) as well as
the endpoint
        HttpRequest req = new HttpRequest();

req.setEndpoint('http://api.cloudinary.com/v1_1/'+cloudname+'/image/upload');
        req.setMethod('POST');
        //base64encode picture body
        String pictureString = EncodingUtil.base64Encode(cvObj.VersionData);
        // 'UTF-8' encode
        pictureString= EncodingUtil.urlEncode(pictureString, 'UTF-8');
        String tiStmp=String.valueOf(System.NOW().getTime() / 1000);
        String myData = 'public_id='+cvObj.Title+'&timestamp='+tiStmp+apisecret;
        system.debug(myData);
        Blob hash = Crypto.generateDigest('SHA1',Blob.valueOf(myData));
        String hexDigest = EncodingUtil.convertToHex(hash);
        String fileString = 'data:image/png;base64,';
        System.debug(fileString);
        String finalBodyString
='public_id='+cvObj.Title+'&api_key='+apikey+'&timestamp='+EncodingUtil.urlEnc
ode(tiStmp, 'UTF-8')+'&signature='+EncodingUtil.urlEncode(hexDigest, 'UTF-
8')+'&file='+EncodingUtil.urlEncode(fileString, 'UTF-8')+pictureString;

        req.setBody(finalBodyString);
        HttpResponse res = h.send(req);
        system.debug(res);
        Map<String, Object> dataMap = (Map<String,
Object>) JSON.deserializeUntyped(res.getBody());
        //system.debug(JSON.deserializeUntyped(res.getBody()));

        String link = (String)dataMap.get('url');
        system.debug(link);
        ContentVersion updateCV = new ContentVersion();
        updateCV.Id = picture;
        updateCV.Cloudinary_Link__c = link;
        updateCV.Priority__c = priority;
        update updateCV;
        return 'Success';
    }
    catch(Exception e) {
        System.debug('Error is ::: ' + e.getMessage() + ' at line no. ::: ' +
e.getLineNumber());
        return Null;
    }
}
}
@AuraEnabled

```

```

public static String deleteImage(String details,List<String> cvIDs){
    try{
        system.debug(details);
        ContentDocument delCD = new ContentDocument();
        delCD.Id = details;
        Delete delCD;
        List<ContentVersion> cvList = new List<ContentVersion>();
        cvList = [SELECT Id,Priority__c FROM ContentVersion WHERE Id =:cvIDs];
        if(cvIDs != Null && cvIDs.Size() > 0) {
            for (ContentVersion cvObj : cvList) {
                //ContentVersion cvItem = new ContentVersion();
                cvObj.Priority__c--;
            }
        }
        Update cvList;
        return 'Success';
    }
    catch(Exception e) {
        System.debug('Error is ::: ' + e.getMessage() + ' at line no. ::: ' +
e.getLineNumber());
        return null;
    }
}
}
@AuraEnabled
public static String cancelUpload(List<String> imageDetails){
    try{
        List<ContentDocument> deleteList = new List<ContentDocument>();
        for(String s : imageDetails){
            ContentDocument delCD = new ContentDocument();
            delCD.Id = s;
            deleteList.add(delCD);
        }
        Delete deleteList;
        return 'Success';
    }
    catch(Exception e) {
        System.debug('Error is ::: ' + e.getMessage() + ' at line no. ::: ' +
e.getLineNumber());
        return null;
    }
}
}

public class imageDetailsWrapper{
    @AuraEnabled public String imageId;
    @AuraEnabled public String Title;
    @AuraEnabled public String imageVersionId;
    @AuraEnabled public Decimal priority;
    @auraEnabled public String cloudinaryLink;
    public imageDetailsWrapper(String imageId, String Title, String
imageVersionId, Decimal priority, String cloudinaryLink) {
        this.imageId = imageId;
        this.Title = Title;
        this.imageVersionId = imageVersionId;
        this.priority = priority;
        this.cloudinaryLink = cloudinaryLink;
    }
    public imageDetailsWrapper() {
        this.imageId = '';
        this.Title = '';
        this.imageVersionId = '';
        this.priority = 0;
    }
}

```



```

    this.cloudinaryLink = '';
  }
}
}

```

3.3.2 Email Component (LWC)

HTML

```

<template>
  <template if:true={showError}>
    <div>
      <div style="background-color:#8B0000;">
        <p class="slds-p-vertical_medium slds-p-left_medium" style="color:white; font-size:large;">{ErrorMsg}
        </p>
      </div>
      <p class="slds-p-vertical_medium slds-p-left_medium slds-box" style="color:#8B0000;">{ErrorMsg2}</p>
    </div>
  </template>
  <lightning-layout multiple-rows="true">
    <lightning-layout-item size='3' class="slds-m-top_x-small"><label>From</label></lightning-layout-item>
    <lightning-layout-item size='9' class="slds-m-top_x-small">
      <lightning-combobox variant="label-hidden" onchange={handleChange} name="From" value={value} options={user}
        dropdown-alignment="left"></lightning-combobox>
    </lightning-layout-item>
    <lightning-layout-item size='3' class="slds-m-top_x-small"><label>To</label></lightning-layout-item>
    <lightning-layout-item size='9' class="slds-m-top_x-small">
      <lightning-layout multiple-rows="true" class="slds-box slds-box_xx-small">
        <lightning-layout-item>
          <lightning-layout multiple-rows="true">
            <template for:each={selectedRecord} for:item="item">
              <lightning-layout-item key={index}>
                <lightning-
pill label={item.name} onremove={handleItemRemove} data-id={item.email}>
                  <lightning-icon icon-name={item.iconname}></lightning-
icon>
                </lightning-pill>
              </lightning-layout-item>
            </template>
          </lightning-layout>
        </lightning-layout-item>
      </lightning-layout-item flexibility="auto">

```

```

        <lightning-
input value={strText} onblur={onBlur} oncommit={onCommit} onkeyup={search
Records}
        variant="label-hidden" class="test"></lightning-input>
    </lightning-layout-item>
    <template if:false={displayCc}>
        <lightning-layout-item alignment-bump="left">
            <lightning-
button variant="base" label="Cc" onclick={onCc}></lightning-button>
        </lightning-layout-item>
    </template>
</lightning-layout>
<template if:true={displayDp}>
    <div class="slds-form-element">
        <div class="slds-form-element__control">
            <div class="slds-combobox_container">
                <div class="slds-dropdown slds-dropdown_length-5 slds-
dropdown_fluid" role="listbox">
                    <ul class="slds-listbox slds-
listbox_vertical recordListBox">
                        <!-- To display Drop down List -->
                        <template for:each={objectsRecord} for:item="rec">
                            <li class="slds-
listbox__item hover" key={index} onmousedown={selectItem}
                                data-id={rec.email} data-name={rec.name}>
                                <div class="slds-media slds-
listbox__option_entity slds-border_bottom">
                                    <span>
                                        <lightning-icon icon-
name={rec.iconname}></lightning-icon>
                                    </span>
                                    <span class="verticalAlign slds-p-left_x-small">
                                        <span class="slds-
truncate">{rec.name}<br />{rec.email}</span>
                                    </span>
                                </div>
                            </li>
                        </template>
                    </ul>
                </div>
            </div>
        </div>
    </div>
</template>
</lightning-layout-item>
<template if:true={displayCc}>
    <lightning-layout-item size="3" class="slds-m-top_x-small">
        <label>Cc</label>
    </lightning-layout-item>

```

```

        <lightning-layout-item size="9" class="slds-m-top_x-small">
            <c-email-field-l-w-
c onprogressvaluechange={hanldeProgressValueChange1}></c-email-field-l-w-
c>
            </lightning-layout-item>
        </template>
        <lightning-layout-item size="3" class="slds-m-top_x-small">
            <label>Bcc</label>
        </lightning-layout-item>
        <lightning-layout-item size="9" class="slds-m-top_x-small">
            <c-email-field-l-w-
c onprogressvaluechange={hanldeProgressValueChange}></c-email-field-l-w-
c>
            </lightning-layout-item>
        <lightning-layout-item size="3" class="slds-m-top_x-small">
            <label>Subject</label>
        </lightning-layout-item>
        <lightning-layout-item size="9" class="slds-m-top_x-small">
            <lightning-input value={subject} variant="label-
hidden" onchange={changeSubject}
            placeholder="Enter Subject..."></lightning-input>
        </lightning-layout-item>
        <lightning-layout-item size="12" class="slds-m-top_x-small">
            <lightning-input-rich-
text value={mailBody} onchange={changeBody} class="slds-m-top_small">
            </lightning-input-rich-text>
        </lightning-layout-item>
    </lightning-layout>
    <div>
        <template for:each={addAttachmentPill} for:item="item">
            <lightning-pill class="slds-m-top_small slds-m-left_small slds-
size_2-of-2" key={index} data-id={item.id}
            label={item.Title} onremove={attRemove}>
            <lightning-icon icon-name="doctype:attachment" alternative-
text="Attachment">
            </lightning-icon>
            </lightning-pill>
        </template>
    </div>
    <lightning-button class="slds-m-top_x-small" icon-
name="utility:attach" icon-position="left" onclick={openModel}>
    </lightning-button>
    <lightning-button variant="brand" class="slds-m-top_x-large slds-m-
left_small" label="Send" title="Send"
    onclick={onSend}></lightning-button>

    <template if:true={isModalOpen}>
        <!-- Modal/Popup Box starts here-->

```

```

    <section role="dialog" tabindex="-1" aria-labelledby="modal-heading-
01" aria-modal="true"
        aria-describedby="modal-content-id-1" class="slds-modal slds-fade-
in-open">
        <div class="slds-modal__container">
            <!-- Modal/Popup Box Header Starts here-->
            <header class="slds-modal__header">
                <lightning-button-icon icon-
name="utility:close" onclick={closeModel} alternative-text="close"
                variant="bare-inverse" class="slds-modal__close"></lightning-
button-icon>
                <h2 id="modal-heading-01" class="slds-text-heading_medium slds-
hyphenate">Select Files</h2>
            </header>
            <div class="slds-modal__content slds-p-around_medium">
                <lightning-layout multiple-rows="true">
                    <lightning-layout-item size="3">
                        <lightning-file-
upload name="fileUploader" multiple="true" accept={filetype}
                        onuploadfinished={handleUploadFinished}></lightning-file-
upload>
                    </lightning-layout-item>
                    <lightning-layout-item size="9">
                        <div>
                            <lightning-
input type="search" placeholder="Search ...." value={searchInput}
                            onkeyup={searchContentDocs}></lightning-input>
                        </div>
                    </lightning-layout-item>
                </lightning-layout>
                <lightning-layout multiple-rows="true">
                    <lightning-layout-item size="3">
                        <lightning-vertical-navigation selected-item="Owned by Me">
                            <lightning-vertical-navigation-section label="">
                                <lightning-vertical-navigation-
item label="Owned by Me" name="Owned by Me">
                                </lightning-vertical-navigation-item>
                                <lightning-vertical-navigation-
item label="Shared by Me" name="Shared by Me">
                                </lightning-vertical-navigation-item>
                                <lightning-vertical-navigation-
item label="Recent" name="Recent">
                                </lightning-vertical-navigation-item>
                                <lightning-vertical-navigation-
item label="Following" name="Following">
                                </lightning-vertical-navigation-item>
                                <lightning-vertical-navigation-
item label="Libraries" name="Libraries">
                                </lightning-vertical-navigation-item>

```

```

        <lightning-vertical-navigation-
item label="Recent Files" name="Recent Files">
        </lightning-vertical-navigation-item>
    </lightning-vertical-navigation-section>
</lightning-vertical-navigation>
</lightning-layout-item>

    <lightning-layout-item size="9">
        <div class="slds-
scrollable_y" style="height:19rem;width:30rem">
            <br />
            <div>
                <template for:each={ContentDocItems} for:item="item">
                    <div class="hove slds-m-top_x-small" data-
id={item.id} key={index}>
                        <lightning-layout class="slds-border_bottom">
                            <lightning-layout-item size="1">
                                <lightning-input type="checkbox"
                                    class="slds-m-left_small slds-m-top_x-
small" onchange={checkCB}
                                    data-id={item.id} name="input2"></lightning-
input>
                                </lightning-layout-item>
                                <lightning-layout-item size="1">
                                    <img src={item.imageLink} style="height:2rem; width:2rem;" />
                                </lightning-layout-item>
                                <lightning-layout-item size="10" class="slds-m-
left_x-small ">
                                    <b style="color:#1E90FF;">{item.Title}</b><br />
                                    {item.ContentModifiedDate}*{item.ContentSize}*{item.FileType}
                                </lightning-layout-item>
                            </lightning-layout>
                        </div>
                    </template>
                </div>
            </div>
        </lightning-layout-item>
    </lightning-layout>
</div>

    <footer class="slds-modal__footer">
        <lightning-
button variant="neutral" label="Cancel" title="Cancel" onclick={closeMode
l}>
        </lightning-button>

```

```

        <lightning-
button variant="brand" label="Save" title="Save" onclick={onSave}>
        </lightning-button>
    </footer>
</div>
</section>
<div class="slds-backdrop slds-backdrop_open"></div>
</template>
</template>

```

JavaScript

```

import { LightningElement, track, api } from 'lwc';
import UserDetails from '@salesforce/apex/EmailTab.UserDetails';
import searchRecords from '@salesforce/apex/EmailTab.fetchRecords';
import sendMailMethod from '@salesforce/apex/EmailTab.sendMailMethod';
import fetchContentDocuments from '@salesforce/apex/EmailTab.fetchContent
Documents';
import searchContentDocuments from '@salesforce/apex/EmailTab.searchConte
ntDocuments';

export default class EmailTabLWC extends LightningElement {
    @api user = [];
    @api value;
    @api subject;
    @api mailBody;
    @api strText;
    @api displayCc = false;
    @api displayDp = false;
    @api objectsRecord;
    @api selectedRecord = [];
    // @api emailsList = [];
    @api showError = false;
    @api errorMsg;
    @api errorMsg2;
    @api ccmails;
    @api bccmails = [];

    @api isModalOpen = false;

    @api ContentDocItems = [];
    @api addAttachmentPill = [];
    @api selectedImages = [];

    connectedCallback() {
        UserDetails({
        })
        .then(result => {

```

```

        this.response = result;
        var item = [];
        for (var i = 0; i < this.response.length; i++) {
            item = {

label: this.response[i].Name + '<' + this.response[i].Email + '>',
            value: this.response[i].Email
        }
        this.user.push(item);
        this.value = this.response[0].Email;
    }
    })
    .catch(error => {
        console.log(error.message);
    });
    fetchContentDocuments({
    })
    .then(result => {
        this.response = result;
        this.ContentDocItems.length = 0;
        for (var i = 0; i < this.response.length; i++) {
            var Cdate = this.response[i].ContentModifiedDate;
            var item = {
                "id": this.response[i].Id,

"imageLink": '/sfc/servlet.shepherd/version/download/' + this.response[i]
.Id,
                "Title": this.response[i].Title,
                "ContentModifiedDate": Cdate.slice(0, 10) + ' ',

"ContentSize": ' ' + (this.response[i].ContentSize / 1024).toFixed(2) + '
kb ',
                "FileType": this.response[i].FileType,
                "contentDocId": this.response[i].ContentDocumentId
            }
            this.ContentDocItems.push(item);
        }
        console.log(this.ContentDocItems);
    })
    .catch(error => {
        console.log(error.message);
    });
}
searchRecords(event) {
    this.strText = event.target.value;
    searchRecords({
        'searchString': this.strText
    })
    .then(result => {

```

```

        this.response = result;
        if (this.response.length > 0) {
            this.displayDp = true;
            this.objectsRecord = this.response;
        }
        else {
            this.displayDp = false;
        }
    })
    .catch(error => {
        console.log(error.message);
    });
}
onBlur() {
    this.displayDp = false;
}
onCc() {
    this.displayCc = true;
}
onCommit() {
    try {
        var text = this.strText;
        var mailformat = /^[A-Za-z0-9_\-\.]+\@([A-Za-z0-9_\-\.])+\.([A-
Za-z]{2,4})$/;
        if (text.match(mailformat)) {
            var item = {
                "name": text,
                "value": text.toString(),
                "iconname": 'standard:email',
                "email": text
            };
            var count = 1;
            for (var i = 0; i < this.selectedRecord.length; i++) {
                if (this.selectedRecord[i].email.toLowerCase() == item.email.toLowerCase(
                )) {
                    count = 0;
                    break;
                }
            }
            if (count == 1) {
                this.selectedRecord.push(item);
            }
            this.strText = null;
            this.displayDp = false;
        }
        else {
            this.ErrorMsg = 'Review the errors on this page.';
            this.ErrorMsg2 = 'Please enter valid Email address to commit!';
        }
    }
}

```



```

        this.showError = true;
    }
}
catch (err) {
    console.log(err);
}
}
selectItem(event) {
    var text = event.currentTarget.dataset.id;
    var count = 1;
    for (var i = 0; i < this.selectedRecord.length; i++) {
if (this.selectedRecord[i].email.toLowerCase() == text.toLowerCase()) {
    count = 0;
    break;
}
}
    if (count == 1) {
        for (var i = 0; i < this.objectsRecord.length; i++) {
if (this.objectsRecord[i].email.toLowerCase() == text.toLowerCase()) {
            this.selectedRecord.push(this.objectsRecord[i]);
            break;
        }
    }
    this.strText = null;
    this.displayDp = false;
}
handleItemRemove(event) {
    try {
        var text = event.currentTarget.dataset.id;
        for (var i = 0; i < this.selectedRecord.length; i++) {
            if (this.selectedRecord[i].email == text) {
                this.selectedRecord.splice(i, 1);
                break;
            }
        }
    }
    catch (err) {
        console.log(err);
    }
}
hanldeProgressValueChange(event) {
    this.bccmails = event.detail;
}
hanldeProgressValueChange1(event) {
    this.ccmails = event.detail;
}

```

```

changeSubject(event) {
    this.subject = event.target.value;
}
changeBody(event) {
    this.mailBody = event.target.value;
}
openModel() {
    this.isModalOpen = true;
    this.connectedCallback();
}
closeModel() {
    this.isModalOpen = false;
    this.selectedImages.length = 0;
}
checkCB(event) {
    var text = event.currentTarget.dataset.id;
    var count = 0;
    for (var i = 0; i < this.selectedImages.length; i++) {
        if (this.selectedImages[i] == text) {
            this.selectedImages.splice(i, 1);
            count = 1;
            break;
        }
    }
    if (count == 0)
        this.selectedImages.push(text);
}
searchContentDocs(event) {
    var text = event.target.value;
    searchContentDocuments({
        "inputStr": text
    })
    .then(result => {
        this.response = result;
        console.log(this.response);
        this.ContentDocItems.length = 0;
        var itemList = [];
        for (var i = 0; i < this.response.length; i++) {
            var Cdate = this.response[i].ContentModifiedDate;
            var item = {
                "id": this.response[i].Id,
                "imageLink": '/sfc/servlet.shepherd/version/download/' + this.response[i].Id,
                "Title": this.response[i].Title,
                "ContentModifiedDate": Cdate.slice(0, 10) + ' ',
                "ContentSize": ' ' + (this.response[i].ContentSize / 1024).toFixed(2) + ' kb '
            }
        }
    })
}

```

```

        "FileType": this.response[i].FileType,
        "contentDocId": this.response[i].ContentDocumentId
    }
    itemList.push(item);
}
this.ContentDocItems = itemList;
console.log(this.ContentDocItems);
})
.catch(error => {
    console.log(error.message);
});
}
onSave() {
    try {
        for (var i = 0; i < this.selectedImages.length; i++) {
            var count = 1;
            for (var j = 0; j < this.addAttachmentPill.length; j++) {
                if (this.addAttachmentPill[j].id == this.selectedImages[i]) {
                    count = 0;
                    break;
                }
            }
            if (count == 1) {
                for (var k = 0; k < this.ContentDocItems.length; k++) {
                    if (this.ContentDocItems[k].id == this.selectedImages[i]) {
                        this.addAttachmentPill.push(this.ContentDocItems[k]);
                        break;
                    }
                }
            }
        }
        this.isModalOpen = false;
    }
    catch (err) {
        console.log(err);
    }
}
attRemove(event) {
    var text = event.currentTarget.dataset.id;
    for (var i = 0; i < this.addAttachmentPill.length; i++) {
        if (text == this.addAttachmentPill[i].id) {
            this.addAttachmentPill.splice(i, 1);
            break;
        }
    }
}
onSend() {
    try {
        var toMails = [];

```

```

var ccmaillist = [];
var bccmaillist = [];
var attList = [];
if (this.selectedRecord != null)
    for (var i = 0; i < this.selectedRecord.length; i++)
        toMails.push(this.selectedRecord[i].email);
if (this.ccmails != null)
    for (var i = 0; i < this.ccmails.length; i++)
        ccmaillist.push(this.ccmails[i].email);
if (this.bccmails != null)
    for (var i = 0; i < this.bccmails.length; i++)
        bccmaillist.push(this.bccmails[i].email);
if (this.addAttachmentPill != null)
    for (var i = 0; i < this.addAttachmentPill.length; i++)
        attList.push(this.addAttachmentPill[i].contentDocId);
sendMailMethod({
    'mToMail': toMails,
    'mCcMail': ccmaillist,
    'mBccMail': bccmaillist,
    'mSubject': this.subject,
    'mbody': this.mailBody,
    'contentDocId': attList
})
.then(result => {
    this.response = result;
    alert(this.response);
})
.catch(error => {
    console.log(error.message);
});
}
catch (err) {
    console.log(err);
}
}
handleUploadFinished(event) {
    try {
        const uploadedFiles = event.detail.files;
        for (var i = 0; i < uploadedFiles.length; i++) {
            var att = {
                "id": uploadedFiles[i].ContentVersionId,
                "imageLink": '/sfc/servlet.shepherd/version/download/'+uploadedFiles[i].ContentVersionId,
                "Title": uploadedFiles[i].name,
                "contentDocId": uploadedFiles[i].documentId
            };
            this.addAttachmentPill.push(att);
        }
    }
}

```

```

    } catch (err) {
        console.log(err);
    }
}
}
}

```

XML

```

<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata"
>
    <apiVersion>51.0</apiVersion>
    <isExposed>true</isExposed>
    <targets>
        <target>lightning__AppPage</target>
        <target>lightning__RecordPage</target>
        <target>lightning__HomePage</target>
    </targets>
</LightningComponentBundle>

```

CSS

```

.slds-input {
    border: 0px !important;
    margin: 0px !important;
}
.test input:focus{
    box-shadow:none;
}
.slds-modal__container {
    max-width: 80rem !important;
    width: 50% !important;
}
.slds-pill {
    max-width: 150px !important;
}

```

Apex Controller

```

public class EmailTab {
    @auraEnabled
    public static List<User> UserDetails(){
        try{
            String userId = UserInfo.getUserName();
            User currentUser = new User();

```

```

currentUser = [SELECT Name,Email FROM User WHERE UserName =: userId LIMIT
1];
    List<User> userList = new List<User>();
    userList.add(currentUser);
    return userList;
}
catch(Exception e){

System.Debug('Error is :::'+e.getMessage()+' at line no. :::'+e.getLineNu
mber());
    return null;
}
}
public class RecordsData{
    @AuraEnabled public String name;
    @AuraEnabled public String recordId;
    @AuraEnabled public String email;
    @AuraEnabled public String ObjType;
    @AuraEnabled public String iconname;

public RecordsData(String name, String email, String recordId, String Obj
Type,String iconname) {
    this.name = name;
    this.email = email;
    this.recordId = recordId;
    this.ObjType = ObjType;
    this.iconname = iconname;
}
}
@AuraEnabled
public static List<ContentVersion> fetchContentDocuments(){
    try{
        List<ContentVersion> cdList = new List<ContentVersion>();

cdList = [SELECT Title,FileType,ContentSize,ContentModifiedDate,ContentDo
cumentId,Id FROM ContentVersion ORDER BY Title ASC LIMIT 100];
        return cdList;
    }
    catch(Exception e){

System.Debug('Error is :::'+e.getMessage()+' at line no. :::'+e.getLineNu
mber());
        return null;
    }
}
@AuraEnabled

```

```

public static List<ContentVersion> searchContentDocuments(String inputStr)
{
    try{
        List<ContentVersion> cdList = new List<ContentVersion>();

        cdList = Database.query('SELECT Title,FileType,ContentSize,ContentModifiedDate,ContentDocumentId,Id FROM ContentVersion WHERE Title LIKE \''+inputStr+'%\' LIMIT 100');
        return cdList;
    }
    catch(Exception e){

        System.Debug('Error is :::'+e.getMessage()+ ' at line no. :::'+e.getLineNumber());
        return null;
    }
}

@AuraEnabled
public static List<RecordsData> fetchRecords(String searchString) {
    try{
        List<RecordsData> recordsDataList = new List<RecordsData>();
        List<Contact> conList = new List<Contact>();
        List<Lead> leadList = new List<Lead>();
        List<User> userList = new List<User>();

        conList = Database.query('SELECT Email, Name FROM Contact WHERE Email LIKE \''+searchString+'%\' LIMIT 5');

        leadList = Database.query('SELECT Email, Name FROM Lead WHERE Email LIKE \''+searchString+'%\' LIMIT 5');

        userList = Database.query('SELECT Email, Name FROM User WHERE Email LIKE \''+searchString+'%\' LIMIT 5');
        if(conList != NULL && conList.size() > 0){
            for(Contact obj : conList)

            recordsDataList.add( new RecordsData(obj.Name,obj.Email,obj.Id,'contact','standard:contact'));
        }
        if(leadList != NULL && leadList.size() > 0){
            for(Lead obj : leadList)

            recordsDataList.add( new RecordsData(obj.Name,obj.Email,obj.Id,'lead','standard:lead'));
        }
        if(userList != NULL && userList.size() > 0){
            for(User obj : userList)

```

```

recordsDataList.add( new RecordsData(obj.Name,obj.Email,obj.Id,'user','standard:user'));
    }
    system.debug(recordsDataList);
    return recordsDataList;
}
catch(exception e){

System.Debug('Error is :::'+e.getMessage()+ ' at line no. :::'+e.getLineNumber());
    return null;
}
}
@auraEnabled

public static string sendMailMethod(String recordId,List<String> mToMail,
List<String> mCcMail, List<String> mBccMail ,String mSubject,String mbody
,List<String> contentDocId){
    try{

system.debug(mToMail+' '+mCcMail+' '+mBccMail+' '+mSubject+' '+mbody+' '+
contentDocId);

        System.debug(recordId);
        PageReference brochure;
        brochure = Page.createPDF;
        brochure.getParameters().put('emailbody', mbody);
        //brochure.getParameters().put('emailbody2', part2);
        Blob pageContent;
        pageContent = brochure.getContent();
        String fileName = 'email.pdf';
        ContentVersion content = new ContentVersion();

        if (Schema.sObjectType.ContentVersion.fields.title.isCreateable()) {
            content.title = fileName;
        }

        if (Schema.sObjectType.ContentVersion.fields.origin.isCreateable()) {
            content.origin = 'C';
        }

        if (Schema.sObjectType.ContentVersion.fields.versionData.isCreateable())
        {
            content.versionData = pageContent;
        }
    }
}

```



```

if (Schema.sObjectType.ContentVersion.fields.PathOnClient.isCreateable())
{
    content.PathOnClient = fileName;
}

if (Schema.sObjectType.ContentVersion.fields.firstPublishLocationId.isCreateable()) {

//content.firstPublishLocationId = DataWrapperrObj.BuildingList[0].Id;
}

    if (Schema.sObjectType.ContentVersion.isCreateable()) {
        insert content;
    }
    List<ContentVersion> versionList = new List<ContentVersion>();
    if (Schema.sObjectType.ContentVersion.isAccessible()) {

versionList = [SELECT contentdocumentId,Id FROM ContentVersion Where id =
: String.escapeSingleQuotes(
    content.Id) LIMIT 1];
    }

    if(mToMail.size() > 0 || mCcMail.size() > 0 || mBccMail.size() > 0){
        List<String> contIds = new List<String>();
        contIds.add(content.Id);
        for(String s : contentDocId ){

ContentVersion contId =[SELECT Id FROM ContentVersion WHERE ContentDocume
ntId =: s LIMIT 1];
            contIds.add(contId.Id);
        }

List<Messaging.SingleEmailMessage> mails = new List<Messaging.SingleEmail
Message>();

Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();
        List<String> sendTo = new List<String>();
        for(String s : mToMail)
            sendTo.add(s);
        mail.setCcAddresses(mCcMail);
        mail.setBccAddresses(mBccMail);
        mail.setToAddresses(sendTo);
        mail.setSubject(mSubject);
        mail.setHtmlBody(mbody);
        mail.setEntityAttachments(contIds);
        mails.add(mail);
        Messaging.sendEmail(mails);
    }
}

```

```

        return versionList[0].Id;

    }catch(Exception ex){

System.debug('Exception :::'+ex.getMessage() + 'Line Number::'+ex.getLin
eNumber());
        return null;
    }
}
}
}

```

3.3.3 SMS Sender

HTML

```

<template>
    <div>
        <lightning-card title="SMS Sender" icon-
name="standard:sms"></lightning-card>
        <lightning-layout multiple-rows="true">
            <lightning-layout-item class="slds-m-horizontal_xx-small slds-m-
vertical_x-small">
                <lightning-button variant={phone.variant} label="Phone" icon-
name={phone.icon} title="Phone"
                onclick={selectPhone}></lightning-button>
            </lightning-layout-item>
            <lightning-layout-item class="slds-m-horizontal_xx-small slds-m-
vertical_x-small">
                <lightning-button variant={home.variant} label="Home Phone" icon-
name={home.icon} title="Home Phone"
                onclick={selectHome}></lightning-button>
            </lightning-layout-item>
            <lightning-layout-item class="slds-m-horizontal_xx-small slds-m-
vertical_x-small">
                <lightning-
button variant={office.variant} label="Office Phone" icon-
name={office.icon}
                title="Office Phone" onclick={selectOffice}></lightning-button>
            </lightning-layout-item>

            <lightning-layout-item size="12">
                <lightning-tabset>
                    <lightning-
tab label="All" value="all" onactive={changeTab} class='slds-p-
vertical_none'></lightning-tab>
                    <lightning-
tab label="Phone" value="phone" onactive={changeTab} class='slds-p-
vertical_none'></lightning-tab>

```

```

        <lightning-
tab label="Home Phone" value="home" onactive={changeTab} class='slds-p-
vertical_none'></lightning-tab>
        <lightning-
tab label="Office Phone" value="office" onactive={changeTab} class='slds-
p-vertical_none'></lightning-tab>
    </lightning-tabset>
    <div id="scroll" class="slds-scrollable_y" style="height:10rem;">
        <template if:true={isMsgListEmpty}>
            <p class="slds-align_absolute-center" style="font-
size:medium;">
                No previous chat history found!
            </p>
        </template>
        <template if:false={isMsgListEmpty}>
            <section role="log" class="slds-chat">
                <ul class="slds-chat-list">
                    <template for:each={messageList} for:item="msg">
                        <template if:true={msg.booleanType} key={index}>
                            <li class="slds-chat-listitem slds-chat-
listitem_inbound" key={index}>
                                <div class="slds-chat-message">
                                    <span aria-hidden="true"
class="slds-avatar slds-avatar_circle slds-
chat-avatar slds-avatar_profile-image-large"></span>
                                    <div class="slds-chat-message__body">
                                        <div
class="slds-chat-message__text slds-chat-
message__text_inbound">
                                            <span onclick={showDateTime}>{msg.Body__c}</span>
                                        </div>
                                        <template if:true={showTimeDate}>
                                            <div class="slds-chat-message__meta">
                                                {msg.Account__r.Name} • {msg.date} • {msg.time}
                                            </div>
                                        </template>
                                    </div>
                                </li>
                            </template>
                            <template if:false={msg.booleanType}>
                                <li class="slds-chat-listitem slds-chat-
listitem_outbound" key={index}>
                                    <div class="slds-chat-message">
                                        <div class="slds-chat-message__body">
                                            <div

```

```

        class="slds-chat-message__text slds-chat-
message__text_outbound">
            <span
onclick={showDateTime}>{msg.Body__c}</span>
            </div>
            <template if:true={showTimeDate}>
            <div class="slds-chat-message__meta">
                You • {msg.date} • {msg.time}
            </div>
            </template>
            </div>
        </div>
    </li>
</template>
</template>
</ul>
</section>
</template>
</div>
</lightning-layout-item>

    <lightning-layout-item size="12" class="slds-m-horizontal_xx-
small slds-m-vertical_x-small">
        <lightning-
textarea name="input1" label="Message" value={messageBody} required oncha
nge={changeMessage}
        message-when-value-
missing="This field is required."></lightning-textarea>
    </lightning-layout-item>
    <lightning-layout-item size="12" class="slds-m-horizontal_xx-
small slds-m-vertical_x-small">
        <lightning-button label="Send" variant="brand" class="slds-
button_stretch slds-size_2-of-2" onclick={onSend}></lightning-button>
    </lightning-layout-item>
</lightning-layout>

</div>
</template>

```

JavaScript

```

import { LightningElement, track, api } from 'lwc';
import { ShowToastEvent } from 'lightning/platformShowToastEvent';
import fetchNumbers from '@salesforce/apex/SMSSender.fetchNumbers';
import sendMessage from '@salesforce/apex/SMSSender.sendMessage';
import checkLookup from '@salesforce/apex/SMSSender.checkLookup';
import fetchMessageRecords from '@salesforce/apex/SMSSender.fetchMessageR
ecords';

```

```

export default class FirstLWC extends LightningElement {
  @api recordId;
  @track numberList;
  @track phone = { variant: 'neutral', icon: 'utility:add' };
  @track home = { variant: 'neutral', icon: 'utility:add' };
  @track office = { variant: 'neutral', icon: 'utility:add' };
  @track toList = { phone: '', homePhone: '', officePhone: '' };
  @track messageBody;
  @track tab = 'all';
  @track messageList;
  @track isMsgListEmpty = false;
  @track showTimeDate = false;

  connectedCallback() {
    fetchNumbers({
      recId: this.recordId
    })
      .then(result => {
        console.log(result);
        this.response = result;
        this.numberList = this.response;
        console.log(this.numberList);
      })
      .catch(error => {
        console.log(error.message);
      });
    this.fetchMessageRecordsMethod();
  }

  changeTab(event) {
    var tabName = event.target.value;
    console.log(tabName);
    switch (tabName) {
      case 'phone':
        //this.messageList = null;
        this.tab = this.numberList.Phone__c;
        break;
      case 'home':
        //this.messageList = null;
        this.tab = this.numberList.Home_Phone__c;
        break;
      case 'office':
        //this.messageList = null;
        this.tab = this.numberList.Office_Phone__c;
        break;
      case 'all':
        this.tab = 'all';
    }
  }
}

```

```

        break;
    }
    this.fetchMessageRecordsMethod();
}

fetchMessageRecordsMethod() {
    fetchMessageRecords({
        RecordId: this.recordId,
        tab: this.tab
    })
    .then(result => {
        this.response = result;
        if (this.response == null)
            this.isMsgListEmpty = true;
        else
            for (var i = 0; i < this.response.length; i++) {
                if (this.response[i].Type__c == 'Inbound')
                    this.response[i].booleanType = true;
                else if (this.response[i].Type__c == 'Outbound')
                    this.response[i].booleanType = false;
                var dateTime = this.response[i].CreatedDate.split('T');
                var msgTime = dateTime[1].substr(0, dateTime[1].length - 5);
                this.response[i].date = dateTime[0];
                this.response[i].time = msgTime;
                this.isMsgListEmpty = false;
            }
        this.messageList = this.response;
        console.log(JSON.stringify(this.messageList[0]));
    })
    .catch(error => {
        console.log(error.message);
    })
}

selectPhone() {
    console.log('---phoneButton---');
    if (this.phone.variant == 'neutral') {
        if (this.numberList.Phone) {
            checkLookup({
                phone: this.numberList.Phone
            })
            .then(result => {
                this.response = result;
                if (this.response == 'mobile') {
                    this.phone.variant = 'brand';
                    this.phone.icon = 'utility:check';
                    this.toList.phone = this.numberList.Phone;
                }
            })
        }
    }
}

```

```

else {
  const event = new ShowToastEvent({
    variant: 'error',
    title: "Can't add Phone!",
    message: 'Number exists in Phone field is not a Mobile Number!',
  });
  this.dispatchEvent(event);
}
})
.catch(error => {
  console.log(error.message);
})
}
else {
  const event = new ShowToastEvent({
    variant: 'error',
    title: "Can't add Phone!",
    message: 'No Number exists in Phone field!',
  });
  this.dispatchEvent(event);
}
}
else {
  this.phone.variant = 'neutral';
  this.phone.icon = 'utility:add';
  this.toList.phone = '';
}
}
selectHome() {
  console.log('---homeButton---');
  if (this.home.variant == 'neutral') {
    if (this.numberList.Home_Phone__c) {
      checkLookup({
        phone: this.numberList.Home_Phone__c
      })
      .then(result => {
        this.response = result;
        if (this.response == 'mobile') {
          this.home.variant = 'brand';
          this.home.icon = 'utility:check';
          this.toList.homePhone = this.numberList.Home_Phone__c;
        }
        else {
          const event = new ShowToastEvent({
            variant: 'error',
            title: "Can't add Home Phone!",
            message: 'Number exists in Home Phone field is not a Mobile Number!',
          });

```

```

        this.dispatchEvent(event);
    }
})
.catch(error => {
    console.log(error.message);
})
}
else {
    const event = new ShowToastEvent({
        variant: 'error',
        title: "Can't add Home Phone!",
        message: 'No Number exists in Home Phone field!',
    });
    this.dispatchEvent(event);
}
}
else {
    this.home.variant = 'neutral';
    this.home.icon = 'utility:add';
    this.toList.homePhone = '';
}
}
selectOffice() {
    console.log('---officeButton---');
    if (this.office.variant == 'neutral') {
        if (this.numberList.Office_Phone__c) {
            checkLookup({
                phone: this.numberList.Office_Phone__c
            })
            .then(result => {
                this.response = result;
                if (this.response == 'mobile') {
                    this.office.variant = 'brand';
                    this.office.icon = 'utility:check';
                    this.toList.officePhone = this.numberList.Office_Phone__c;
                }
                else {
                    const event = new ShowToastEvent({
                        variant: 'error',
                        title: "Can't add Office Phone!",
message: 'Number exists in Office Phone field is not a Mobile Number!',
                    });
                    this.dispatchEvent(event);
                }
            })
            .catch(error => {
                console.log(error.message);
            })

```



```

}
else {
  const event = new ShowToastEvent({
    variant: 'error',
    title: "Can't add Office Phone!",
    message: 'No Number exists in Office Phone field!',
  });
  this.dispatchEvent(event);
}
}
else {
  this.office.variant = 'neutral';
  this.office.icon = 'utility:add';
  this.toList.officePhone = '';
}
}
changeMessage(event) {
  this.messageBody = event.target.value;
}
onSend() {
  console.log('---onSend---');
  var toList = this.toList;
  var x = this.messageBody;
  if ((toList.phone || toList.homePhone || toList.officePhone) && x) {
    sendMessage({
      receiver: toList,
      message: x,
      AccountId: this.recordId
    })
    .then(result => {
      this.response = result;
      if (this.response == 'Success') {
        const event = new ShowToastEvent({
          variant: 'success',
          title: "Successfull",
          message: 'SMS sent successfully!',
        });
        this.dispatchEvent(event);
        this.fetchMessageRecordsMethod();
        this.messageBody = '';
        this.numberList.phone = '';
        this.numberList.homePhone = '';
        this.numberList.officePhone = '';
        this.phone = { variant: 'neutral', icon: 'utility:add' };
        this.home = { variant: 'neutral', icon: 'utility:add' };
        this.office = { variant: 'neutral', icon: 'utility:add' };
      }
      else if (this.response == 'UnSuccess') {
        const event = new ShowToastEvent({

```

```

        variant: 'error',
        title: "Error!",
message: 'No. is not verified in Twilio or any other error found!',
    });
    this.dispatchEvent(event);
}
else {
    const event = new ShowToastEvent({
        variant: 'error',
        title: "Error!",
        message: 'No response from apex received!',
    });
    this.dispatchEvent(event);
}
})
.catch(error => {
    console.log(error.message);
})
}
else {
    const event = new ShowToastEvent({
        variant: 'error',
        title: "Empty Fields founds!",
        message: 'No Phone or No message body found! ',
    });
    this.dispatchEvent(event);
}
}
showDateTime() {
    if (this.showTimeDate == false)
        this.showTimeDate = true;
    else
        this.showTimeDate = false;
}
}

```

XML

```

<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata"
>
    <apiVersion>51.0</apiVersion>
    <isExposed>true</isExposed>
    <targets>
        <target>lightning__AppPage</target>
        <target>lightning__RecordPage</target>
    </targets>

```

```

    <target>lightning__HomePage</target>
  </targets>
</LightningComponentBundle>

```

Apex Controller

```

public class SMSSender {
    @AuraEnabled
    public static Account fetchNumbers(Id recId) {
        try {

            Account accObj = new Account();

accObj = [SELECT Name, Phone, Home_Phone__c, Office_Phone__c FROM Account
WHERE Id =: recId LIMIT 1];
            return accObj;
        } catch (Exception e) {

System.debug('Error is ::: ' + e.getMessage() + ' at line number ::: ' +
e.getLineNumber());
            return Null;
        }
    }

    @AuraEnabled
    public static String sendMessage(Map<String, String> receiver, String mes
sage, String AccountId) {
        system.debug(receiver + ' ' + message);
        try {
            List<String> numberList = receiver.values();
            List<Messages__c> msgList = new List<Messages__c>();
            for (String s : numberList) {
                if (s.length() > 0) {
                    String phNumber = s;
                    String accountSid = 'ACfd34dc0af59f863f60f41f7c93310da4';
                    String token = '31ea55575ad2a09b81c0482bd4877c05';
                    String fromPhNumber = '+13613011115';
                    String smsBody = message;
                    HttpRequest req = new HttpRequest();
                    req.setEndpoint('https://api.twilio.com/2010-04-
01/Accounts/' + accountSid + '/SMS/Messages.json');
                    req.setMethod('POST');
                    String VERSION = '3.2.0';
                    req.setHeader('X-Twilio-Client', 'salesforce-' + VERSION);
                    req.setHeader('User-Agent', 'twilio-salesforce/' + VERSION);
                    req.setHeader('Accept', 'application/json');

```

```

        req.setHeader('Accept-Charset', 'utf-8');
        req.setHeader('Authorization',

'Basic ' + EncodingUtil.base64Encode(Blob.valueOf(accountSid + ':' + token)));
        req.setBody('To=' + EncodingUtil.urlEncode(phoneNumber, 'UTF-8') + '&From=' +
            EncodingUtil.urlEncode(fromPhoneNumber, 'UTF-8') + '&Body=' + smsBody);
        Http http = new Http();
        HTTPResponse res = http.send(req);
        System.debug(res.getBody());
        if (res.getStatusCode() == 201) {
            System.Debug(s+'::: Message sending Successful');
            Messages__c msgObj = new Messages__c();
            msgObj.Account__c = AccountId;
            msgObj.Type__c = 'Outbound';
            msgObj.Mobile__c = s;
            msgObj.Body__c = message;
            msgList.add(msgObj);
        } else {
            System.Debug(s+'::: Message sending Unsuccessful');
        }
    }

    insert msgList;
    return 'Success';
} catch (Exception e) {

System.debug('Error is ::: ' + e.getMessage() + ' at line number ::: ' + e.getLineNumber());
    return Null;
}
}

@AuraEnabled
public static String checkLookup(String phone) {
    try {
        system.debug(phone);
        String type = 'carrier';
        Http http = new Http();
        HttpRequest request = new HttpRequest();
        String accountSid = 'ACfd34dc0af59f863f60f41f7c93310da4';
        String token = '31ea55575ad2a09b81c0482bd4877c05';
        request.setHeader('Authorization',

'Basic ' + EncodingUtil.base64Encode(Blob.valueOf(accountSid + ':' + token)));

```

```

request.setEndpoint('https://lookups.twilio.com/v1/PhoneNumbers/' + phone
+ '?Type=' + type);
request.setMethod('GET');
HttpResponse response = http.send(request);
if (response.getStatusCode() == 200) {

Map<String, Object> results = (Map<String, Object>) JSON.deserializeUntyp
ed(response.getBody());

Map<String, Object> carierDetails = (Map<String, Object>) results.get('ca
rrier');
    String check = (String)carierDetails.get('type');
    system.debug(check);
    return check;
} else {
    system.debug(response);
    return 'error';
}
} catch (Exception e) {

System.debug('Error is ::: ' + e.getMessage() + ' at line number ::: ' +
e.getLineNumber());
    return Null;
}
}
}
@AuraEnabled

public static List<Messages__c> fetchMessageRecords(String RecordId,String
tab){
    try{
        //TwilioReceiveSMS class is used to receive inbound messages...
        system.debug(tab);
        List<Messages__c> msgList = new List<Messages__c>();
        if(tab == 'all')

msgList = [SELECT Body__c,Mobile__c,Type__c,CreateDate,Account__r.Name F
ROM Messages__c WHERE Account__c=:RecordId ORDER BY CreateDate DESC LIMIT 50];
        else if(tab != Null)
            msgList = [SELECT Body__c,Mobile__c,Type__c,CreateDate,Account__
r.Name FROM Messages__c WHERE Account__c=:RecordId AND Mobile__c =: tab O
RDER BY CreateDate DESC LIMIT 50];
        else
            msgList = null;
        return msgList;
    }
    catch(Exception e){

```

```

        System.debug('Error is ::: ' + e.getMessage() + ' at line number ::
: ' + e.getLineNumber());
        return Null;
    }
}
}
}

```

Webservice

```

@RestResource(urlMapping = '/smsHandler/*')
global class TwilioReceiveSMS {
    @HttpGet
    global static void getSMS() {
        try {
            // Store the request
            RestRequest req = RestContext.request;
            // Store the HTTP parameters from the request in a Map
            Map<String, String> sms = req.params ;
            String fromPhNumber ;
            String smsBody ;
            // get the phone number from the response
            if (sms.containsKey('From')) {
                fromPhNumber = sms.get('From') ;
            }
            // get the body of sms from the response
            if (sms.containsKey('Body')) {
                smsBody = sms.get('Body') ;
            }

            System.Debug('This is you msg content you received :' + smsBody + ' ::: '
+ fromPhNumber);
            RestContext.response.statusCode = 200;
            RestContext.response.addHeader('Content-Type', 'text/plain');
            RestContext.response.responseBody =

            Blob.valueOf('Thanks, We have received your SMS and will get back to you
soon') ;
            String plus = String.valueOf('+');
            String no = fromPhNumber.remove(' ');
            String search = plus+no;

            Account accObj = [SELECT Id,Name FROM Account WHERE Phone=: search OR Hom
e_Phone__c =:search
                OR Office_Phone__c =: search LIMIT 1];
            system.debug(accObj);
            Messages__c msgObj = new Messages__c();
            msgObj.Account__c = accObj.Id;

```

```

        msgObj.Type__c = 'Inbound';
        msgObj.Mobile__c = search;
        msgObj.Body__c = smsBody;
        insert msgObj;
        system.debug(msgObj.Id);

        Set<String> idSet = new Set<String>();
        idSet.add('0055g000000sbtCAAQ');

        CustomNotificationType notificationType = [SELECT Id, DeveloperName FROM
        CustomNotificationType WHERE DeveloperName='Custom_Notification'];
        Messaging.CustomNotification notification = new Messaging.CustomNo
        tification();
        notification.setTitle('New Message received!');
        notification.setBody('New message received for '+accObj.Name);
        notification.setNotificationTypeId(notificationType.Id);
        notification.setTargetId(accObj.Id);
        notification.send(idSet);
        system.debug(notificationType);

    }
    catch(Exception e){

        System.debug('Error is ::: ' + e.getMessage() + ' at line number ::: ' +
        e.getLineNumber());
    }
}
}

```

WebHook

```

public class Webhook implements HttpCalloutMock {

    public static HttpRequest request;
    public static HttpResponse response;

    public HTTPResponse respond(HTTPRequest req) {
        request = req;
        response = new HttpResponse();
        response.setStatusCode(200);
        return response;
    }

    public static String jsonContent(List<Object> triggerNew, List<Object> tr
    iggerOld) {
        String newObjects = '[';
        if (triggerNew != null) {

```

```

        newObjects = JSON.serialize(triggerNew);
    }

    String oldObjects = '[]';
    if (triggerOld != null) {
        oldObjects = JSON.serialize(triggerOld);
    }

    String userId = JSON.serialize(UserInfo.getUserId());

String content = '{"new": ' + newObjects + ', "old": ' + oldObjects + ',
"userId": ' + userId + '}';
    return content;
}

@future(callout=true)
public static void callout(String url, String content) {

    if (Test.isRunningTest()) {
        Test.setMock(HttpCalloutMock.class, new Webhook());
    }

    Http h = new Http();

    HttpRequest req = new HttpRequest();
    req.setEndpoint(url);
    req.setMethod('POST');
    req.setHeader('Content-Type', 'application/json');
    req.setBody(content);

    h.send(req);
}
}

```

Trigger WebHook

```

trigger TwilioWebhookTrigger on Account (before insert) {

    String url = 'https://salesforce-webhook-creator.herokuapp.com/app';

    String content = Webhook.jsonContent(Triiger.new, Triiger.old);

    Webhook.callout(url, content);

}

```


Chapter – 4

Testing

The Force.com platform requires that at least 75% of the Apex Code in an org be executed via unit tests in order to deploy the code to production. You shouldn't consider 75% code coverage to be an end goal though. Instead, you should strive to increase the state coverage of your unit tests. Code has many more possible states than it has lines of code. For example, the following method has 4,294,967,296 different states: System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic.

4.1 TYPES OF TESTING

BLACK BOX TESTING:

The technique of testing without having any knowledge of the interior workings of the application is called black box testing. The tester is oblivious to the system architecture and does not have access to the source code. Typically, while performing a black box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.

WHITE BOX TESTING:

Whitebox testing is the detailed investigation of internal logic and structure of the code. Whitebox testing is also called glass testing or open box testing. In order to perform white box testing on an application, a tester needs to know the internal workings of the code.

GREY BOX TESTING:

Grey box testing is a technique to test the application with having a limited knowledge of the internal workings of an application. In software testing, the phrase the more you know, the better carries a lot of weight while testing an application.

UNIT TESTING:

Unit Testing contains the testing of each unit of Recruitment Application. We have tested each interface by input values and check whether it is working properly working or not we also tested database connectivity. We have entered value in interface and check that the values are properly goes to corresponding tuples or not.

INTEGRATION TESTING:

Integration testing is defined as the testing of combined parts of an application to determine if they function correctly. Integration testing can be done in two ways: Bottom up integration testing and Top down integration testing.

SYSTEM TESTING:

System testing tests the system as a whole. Once all the components are integrated, the application as a whole is tested rigorously to see that it meets the specified Quality Standards. This type of testing is performed by a specialized testing team

AUTOMATION T:

Automation Testing or Test Automation is a software testing technique that performs using special automated testing software tools to execute a test case suite. On the contrary, Manual Testing is performed by a human sitting in front of a computer carefully executing the test steps.[15]

CHAPTER 5

CONCLUSION

5.1 CONCLUSION

The main agenda of this project was to meet client needs within the given period of time. In this project, I have worked on different parts of Salesforce including Lightning Components, VisualForce Pages, Lightning Web Components, Apex classes, triggers etc. Through all these tasks I have learned a lot of things about Salesforce and its implementation and gained knowledge and skills. Integration with a third party app i.e. Twilio was the most interesting part for me as it was totally new thing for me and was working first time on integration and Webservices. In this whole process I have used

- Apex Language
- Java Script
- HTML
- XML
- Twilio (For integration)
- VisualForce Page
- Triggers
- Webservices

Chapter – 6

Bibliography

- trailhead.salesforce.com
- developer.salesforce.com
- <https://www.lightningdesignsystem.com/>
- <https://developer.salesforce.com/docs/component-library/overview/components>
- [Communityforce.com](https://communityforce.com)

REFERENCES

- [1] Gregory S. Smith, “Cloud Computing Strategies and Risks”, 11 April 2013, Chapter 7, Pages 125-153, Print ISBN:9781118390030
- [2] Davis D. Janowski, “Selecting the Right CRM System”, January 2013, Pages 5-15, Print ISBN:9781118434765
- [3] Mehmet N. Aydin, Nazim Ziya, “Cloud-Based Development Environments”, Pages 62-69, 13 May 2016, Print ISBN:9781118821978
- [4] Tansu Barker, “Benchmarks of Successful Salesforce Performance”, 08 April 2009
- [5] Deven N. Shah, Dilip Motwani, “Software Engineering”, Publisher: Dreamtech Press, 2010, ISBN:9350040395, 9789350040393
- [6] Gerard O'Regan, “Concise Guide to Software Engineering From Fundamentals to Application Methods”, 2017, Publisher: Springer International Publishing, ISBN: 978-3-319-57750-0
- [7] Iztok Fajfar, “Start Programming Using HTML, CSS And JavaScript”, 2015, ISBN: 9780429083457, Imprint: Chapman and Hall/CRC
- [8] Rajesh K. Maurya and Swati R. Maurya, “Software Testing”, April 21, 2021, Publisher: Dreamtech Press, ISBN: 978-9350044001
- [9] Rod Stephens, “Beginning Software Engineering”, Publisher: Wrox, Publication date: 24 April 2015, ISBN: 8126555378
- [10] Chester Bullock, Mark Pollard, “salesforce-marketing-cloud-for- dummies”, Publisher: For Dummies, 22 December 2017, ISBN: 1119122090

- [11] Rod Stephens, “Beginning Software Engineering”, Publisher: Wrox, 2 March 2015, ISBN: 978-1118969144
- [12] Saeid Abolfazli, Zohreh Sanaei, Mohammad Hadi Sanaei, Mohammad Shojafar and Abdullah Gani, “Mobile Cloud Computing”, Publisher: Wiley, 13 May 2016
- [13] Ernesto Exposito and Codé Diop, “Service-Oriented and Cloud Computing Architectures”, Publisher: Wiley, July 2014, ISBN: 978-1-118-76169-4
- [14] Xiaolong Li, “Information Technology and Applications”, Publisher: Boca Raton - CRC Press, 2014, ISBN: 9780429226373
- [15] Xiaoxu Diao, Manuel Rodriguez and Carol Smidts, “Automated Software Testing”, First published: 06 July 2018