# GOLDVISOR(Source Code)

## GoldVisor.vfp:

<apex:page showHeader="false" sidebar="false" standardStylesheets="false" >

<meta charset="utf-8" />

<meta http-equiv="X-UA-Compatible" content="IE=edge"/>

<meta name="viewport" content="width=device-width, initial-scale=1" />

<title>GoldVisor</title>

<apex:includeScript value="/lightning/lightning.out.js"/>

<apex:includeLightning />

<style>

body

{

margin : 0;

padding: 0;

}

</style>

<div id="lightning" />

<script>

$Lightning.use("c:GoldVisorApp", function()

{

$Lightning.createComponent("c:GVComponent", {}, "lightning", function(cmp){

});

});

</script>

</apex:page>

# GoldVisorApp.app

<aura:application access="GLOBAL" extends="ltng:outApp">

<aura:dependency resource="c:GVComponent" />

</aura:application>

# GoldVisorheader.cmp

<aura:component >

<div style="background-color:rgb(10,50,150);color:white;width:auto;height:100px;padding:10px;font-size:35pt;">

<center>GoldVisor Pvt. Ltd.</center>

</div>

</aura:component>

# GVCtrl.apxc

public class GVCtrl {

@AuraEnabled

public static String getData(){

List<Customers\_\_c> listOfCustomers = CustomerDao.getAllCustomers();

List<Loan\_\_c> listOfLoans=LoanDao.getAllLoans();

List<Employee\_\_c> listOfEmployees = EmployeeDao.getAllEmployees();

List<Branch\_\_c> listOfBranches=BranchDao.getAllBranches();

Map<String, String> result=new Map<String, String>();

result.put('listOfCustomers', JSON.serialize(listOfCustomers));

result.put('listOfLoans', JSON.serialize(listOfLoans));

result.put('listOfEmployees', JSON.serialize(listOfEmployees));

result.put('listOfBranches', JSON.serialize(listOfBranches));

return JSON.serialize(result);

}

@AuraEnabled

public static String loginAccount(String Employee){

Employee\_\_c loginEmployee = (Employee\_\_c)System.JSON.deserialize(Employee, Employee\_\_c.class);

List<Employee\_\_c> listOfEmployees = EmployeeDAO.getAllEmployees();

for(Employee\_\_c u:listOfEmployees){

System.debug(loginEmployee.Email\_\_c+':'+u.Email\_\_c+':'+loginEmployee.Password\_\_c+':'+u.Password\_\_c);

if(loginEmployee.Email\_\_c == u.Email\_\_c && loginEmployee.Password\_\_c == u.Password\_\_c){

return JSON.serialize(u);

}

}

return null;

}

@AuraEnabled

public static String deleteCustomerDetail(String Customer){

System.debug(Customer);

Customers\_\_c newCustomer = (Customers\_\_c)JSON.deserialize(Customer, Customers\_\_c.class);

List<Customers\_\_c> listOfCustomers = CustomerDAO.removeCustomer(newCustomer);

return JSON.serialize(listOfCustomers);

}

@AuraEnabled

public static String saveCustomerInfo(String CustomerJSON){

Customers\_\_c Customer = (Customers\_\_c)System.JSON.deserialize(CustomerJSON, Customers\_\_c.class);

System.debug(Customer);

upsert Customer;

System.debug(Customer);

return JSON.serialize(CustomerDao.getAllCustomers());

}

@AuraEnabled

public static String deleteLoanDetail(String Loan){

System.debug(Loan);

Loan\_\_c newLoan = (Loan\_\_c)JSON.deserialize(Loan, Loan\_\_c.class);

List<Loan\_\_c> listOfLoans = LoanDAO.removeLoan(newLoan);

return JSON.serialize(listOfLoans);

}

@AuraEnabled

public static String saveLoanInfo(String LoanJSON){

Loan\_\_c Loan = (Loan\_\_c)System.JSON.deserialize(LoanJSON, Loan\_\_c.class);

System.debug(Loan);

upsert Loan;

System.debug(Loan);

return JSON.serialize(LoanDao.getAllLoans());

}

@AuraEnabled

public static String deleteEmployeeDetail(String Employee){

System.debug(Employee);

Employee\_\_c newEmployee = (Employee\_\_c)JSON.deserialize(Employee, Employee\_\_c.class);

List<Employee\_\_c> listOfEmployees = EmployeeDAO.removeEmployee(newEmployee);

return JSON.serialize(listOfEmployees);

}

@AuraEnabled

public static String saveEmployeeInfo(String EmployeeJSON){

Employee\_\_c Employee = (Employee\_\_c)System.JSON.deserialize(EmployeeJSON, Employee\_\_c.class);

System.debug(Employee);

upsert Employee;

System.debug(Employee);

return JSON.serialize(EmployeeDao.getAllEmployees());

}

@AuraEnabled

public static String deleteBranchDetail(String Branch){

System.debug(Branch);

Branch\_\_c newBranch = (Branch\_\_c)JSON.deserialize(Branch, Branch\_\_c.class);

List<Branch\_\_c> listOfBranches = BranchDAO.removeBranch(newBranch);

return JSON.serialize(listOfBranches);

}

@AuraEnabled

public static String saveBranchInfo(String BranchJSON){

Branch\_\_c Branch = (Branch\_\_c)System.JSON.deserialize(BranchJSON, Branch\_\_c.class);

System.debug(Branch);

upsert Branch;

System.debug(Branch);

return JSON.serialize(BranchDao.getAllBranches());

}

public Class Option {

public String label;

public String value;

public Option(String label, String value) {

this.label = label;

this.value = value;

}

}

# }

# GVComponent.cmp

<aura:component controller="GVCtrl">

<aura:attribute name="listOfCustomers" type="Object[]" />

<aura:attribute name="listOfLoans" type="Object[]" />

<aura:attribute name="listOfEmployees" type="Object[]" />

<aura:attribute name="listOfBranches" type="Object[]" />

<aura:attribute name="isLogin" type="Boolean" default="false" />

<aura:attribute name="isNotLogin" type="Boolean" default="true" />

<aura:attribute name="EmployeeInfo" type="Employee\_\_c" default="{'sobjectType':'Employee\_\_c'}" />

<aura:handler event="aura:waiting" action="{!c.showSpinner}"/>

<aura:handler event="aura:doneWaiting" action="{!c.hideSpinner}"/>

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

<aura:if isTrue="{!v.isLogin}">

<div class="slds-is-relative">

<aura:if isTrue="{!v.spinner}">

<div aura:id="loadingSpinner" class="slds-spinner\_container">

<div class="slds-spinner--brand slds-spinner slds-spinner--large slds-is-relative" role="alert">

<span class="slds-assistive-text">Loading</span>

<div class="slds-spinner\_\_dot-a"></div>

<div class="slds-spinner\_\_dot-b"></div>

</div>

</div>

</aura:if>

<aura:if isTrue="{!v.isError}">

<div class="slds-notify slds-notify\_alert slds-theme\_alert-texture slds-theme\_error" role="alert">

<h2>{!v.errorMessage}</h2>

</div> <br/><br/>

</aura:if>

<c:GVHeader />

<lightning:carousel disableAutoRefresh="false" disableAutoScroll="false">

<lightning:carouselImage

src = "{!$Resource.carousal1}"

alternativeText = "Second card accessible description.">

</lightning:carouselImage>

<lightning:carouselImage

src = "{!$Resource.carousal2}"

alternativeText = "Second card accessible description.">

</lightning:carouselImage>

<lightning:carouselImage

src = "{!$Resource.carousal3}"

alternativeText = "Third card accessible description.">

</lightning:carouselImage>

<lightning:carouselImage

src = "{!$Resource.carousal4}"

alternativeText = "Third card accessible description.">

</lightning:carouselImage>

<lightning:carouselImage

src = "{!$Resource.carousal5}"

alternativeText = "Third card accessible description.">

</lightning:carouselImage>

<lightning:carouselImage

src = "{!$Resource.carousal6}"

alternativeText = "Third card accessible description.">

</lightning:carouselImage>

</lightning:carousel>

<div class="slds-grid">

<lightning:tabset variant="{!'vertical'}">

<lightning:tab label="Customers"><c:Customers listOfCustomers="{!v.listOfCustomers}" /></lightning:tab>

<lightning:tab label="Loans"><c:Loans listOfLoans="{!v.listOfLoans}" /></lightning:tab>

<lightning:tab label="Employee"><c:Employees listOfEmployees="{!v.listOfEmployees}" /></lightning:tab>

<lightning:tab label="Branch"><c:Branch listOfBranches="{!v.listOfBranches}" /></lightning:tab>

<lightning:tab label="Rates">

<!--gold-feed.com metal pricing start-->

<div style="width: 100%; height: 578px">

<iframe src="https://livegoldfeed.com/goldfeed/profile/frame/b196d09a7559cfede488b783c9a474b0" scrolling="no" height="100%" width="100%" frameborder="0"></iframe>

</div>

<!--gold-feed.com metal pricing end-->

</lightning:tab>

<lightning:tab label="Profile"><c:Profile EmployeeInfo="{!v.EmployeeInfo}" /></lightning:tab>

<lightning:tab label="Sign Out">

<lightning:button variant="brand" label="Sign out" title="Sign out" onclick="{!c.logoutEmployeeProfile}"></lightning:button>

</lightning:tab>

</lightning:tabset>

</div>

</div>

</aura:if>

<aura:if isTrue="{!v.isNotLogin}">

<c:SignIn EmployeeInfo="{!v.EmployeeInfo}" isLogin="{!v.isLogin}" isNotLogin="{!v.isNotLogin}"/>

</aura:if>

</aura:component>

# GVComponentController.js

({

doInit : function(component, event, helper) {

helper.getInitialData(component, event);

},

showSpinner: function(component) {

component.set("v.spinner", true);

},

hideSpinner : function(component){

component.set("v.spinner", false);

},

logoutEmployeeProfile: function(component){

component.set("v.isLogin", false);

component.set("v.isNotLogin", true);

}

})

# GVComponentHelper.js

({

getInitialData : function(component, event) {

console.log("GVComponentHelper.js:3");

var action = component.get("c.getData");

action.setCallback(this, function(response){

var state=response.getState();

if(component.isValid() && state==="SUCCESS"){

var result=JSON.parse(response.getReturnValue());

var listOfCustomers=JSON.parse(result['listOfCustomers']);

var listOfLoans=JSON.parse(result['listOfLoans']);

var listOfEmployees=JSON.parse(result['listOfEmployees']);

var listOfBranches=JSON.parse(result['listOfBranches']);

component.set("v.listOfCustomers", listOfCustomers);

component.set("v.listOfLoans", listOfLoans);

component.set("v.listOfEmployees", listOfEmployees);

component.set("v.listOfBranches", listOfBranches);

}

});

$A.enqueueAction(action);

}

})

# Customer.cmp

<aura:component >

<aura:attribute name="listOfCustomers" type="Object[]" />

<div class="slds-grid">

<lightning:tabset variant="{!'scoped'}">

<lightning:tab label="Create"><c:CreateCustomer listOfCustomers="{!v.listOfCustomers}" /></lightning:tab>

<lightning:tab label="View">

<c:ViewCustomer listOfCustomers="{!v.listOfCustomers}" />

</lightning:tab>

</lightning:tabset>

</div>

</aura:component>

# CreateCustomer.cmp

<aura:component implements="force:appHostable" controller="GVCtrl">

<aura:attribute name="showCreateCustomerForm" type="Boolean" default="false" />

<aura:attribute name="isEditCustomer" type="Boolean" default="false" />

<aura:attribute name="ErrorMessage" type="String" default="date Invalid"/>

<aura:attribute name="isError" type="Boolean" default="false"/>

<aura:attribute name="listOfCustomers" type="Object[]" />

<aura:attribute name="newCustomer" type="Customers\_\_c" default="{'sobjectType':'Customers\_\_c',

'Name':''}"/>

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

<div class="slds-form slds-form\_stacked">

<div class="slds-form-element">

<lightning:input type="text" label="Customer Name" value="{! v.newCustomer.Name}"

required="true"

pattern="^[A-Za-z ]\*$"

messagewhenPatternMismatch="Only Alphabets are allowed" />

</div>

<div class="slds-form-element">

<lightning:input type="text" label="Phone No" value="{! v.newCustomer.Phone\_No\_\_c}"

pattern="^[0-9]\*$" minlength="10" maxlength="10"

messagewhenPatternMismatch="Only numbers and spaces are allowed" />

</div>

<div class="slds-form-element">

<lightning:input type="text" label="Alternate Phone No " value="{! v.newCustomer.Alternate\_Phone\_No\_\_c}"

pattern="^[0-9]\*$" minlength="10" maxlength="10"

messagewhenPatternMismatch="Only numbers and spaces are allowed" />

</div>

<div class="slds-form-element">

<lightning:input type="text" label="Permanent Address" value="{! v.newCustomer.Permanent\_Address\_\_c}"

pattern="^[A-Za-z0-9 - ]\*$"

messagewhenPatternMismatch="Only alphanumeric and are allowed" />

</div>

<div class="slds-form-element">

<lightning:input type="text" label="Current Address" value="{! v.newCustomer.Current\_Address\_\_c }"

pattern="^[A-Za-z0-9 - ]\*$"

messagewhenPatternMismatch="Only alphanumeric and are allowed" />

</div>

<div class="slds-form-element">

<lightning:input type="email" label="Email" value="{! v.newCustomer.Email\_\_c }" />

</div>

<div class="slds-form-element">

<lightning:input type="text" label="Weight of Gold " value="{! v.newCustomer.Weight\_of\_Gold\_\_c }" />

</div>

<div class="slds-form-element">

<lightning:input type="text" label="Time Period(enter in months)" value="{! v.newCustomer.Time\_Period\_\_c }" />

</div>

<!-- <div class="slds-form-element">

<lightning:select label="Status" value="{!v.newProject.Status\_\_c}">

<aura:iteration items="{!v.listOfProjectStatus}" var="item">

<option text="{!item.label}" value="{!item.value}"

selected="{! item.value == v.newProject.Status\_\_c}"/>

</aura:iteration>

</lightning:select>

</div>

<div class="slds-form-element">

<lightning:dualListbox name="Technology" label= "Technology" sourceLabel="Available"

selectedLabel="Selected" options="{!v.listOfProjectTech}" onchange=""/>

</div>

<div class="slds-form-element">

<lightning:textarea name="Description" value="{!v.newProject.Description\_\_c}" label="Description" placeholder="type here..."/>

</div>-->

<div class="slds-form-element">

<lightning:button variant="brand" label="Save" title="Save" onclick="{!c.AddCustomer}" />

</div>

</div>

</aura:component>

# CreateCustomerController.js

({

AddCustomer : function(component, event, helper) {

helper.saveCustomer(component, event, helper);

}

})

# CreateCustomerHelper.js

({

/\*getInitData : function(component,event) {

var action = component.get("c.getInitialData");

action.setCallback(this, function(response) {

var state = response.getState();

console.log(state);

if (component.isValid() && state === "SUCCESS") {

var result = JSON.parse(response.getReturnValue());

component.set("v.listOfTaskType", JSON.parse(result['PROJECT\_TYPE\_LIST']));

component.set("v.listOfTaskStatus", JSON.parse(result['STATUS\_LIST']));

component.set("v.listOfUsers", JSON.parse(result['USER\_LIST']));

component.set("v.listOfPriority", JSON.parse(result['PROJECT\_PRIORITY\_LIST']));

component.set("v.listOfProject", JSON.parse(result['PROJECT\_LIST']));

}

else if (state === "ERROR") {

var errors = response.getError();

if (errors) {

if (errors[0] && errors[0].message) {

this.handleErrors(component,errors[0].message);

}

}

}

});

$A.enqueueAction(action);

},

getType : function(component,event) {

var action = component.get("c.getProjectType");

action.setCallback(this, function(response) {

var state = response.getState();

console.log(state);

if (component.isValid() && state === "SUCCESS") {

var listOfTaskType = JSON.parse(response.getReturnValue());

component.set("v.listOfTaskType", listOfTaskType);

}

else if (state === "ERROR") {

var errors = response.getError();

if (errors) {

if (errors[0] && errors[0].message) {

this.handleErrors(component,errors[0].message);

}

}

}

});

$A.enqueueAction(action);

},

getStatus : function(component, event) {

var action = component.get("c.getStatus");

action.setCallback(this, function(response) {

var state = response.getState();

console.log(state);

if (component.isValid() && state === "SUCCESS") {

var listOfTaskStatus = JSON.parse(response.getReturnValue());

component.set("v.listOfTaskStatus", listOfTaskStatus);

}

else if (state === "ERROR") {

var errors = response.getError();

if (errors) {

if (errors[0] && errors[0].message) {

this.handleErrors(component,errors[0].message);

}

}

}

});

$A.enqueueAction(action);

},

getUsers : function(component,event) {

var action = component.get("c.getUsers");

action.setCallback(this, function(response) {

var state = response.getState();

console.log(state);

if (component.isValid() && state === "SUCCESS") {

var listOfUsers = JSON.parse(response.getReturnValue());

component.set("v.listOfUsers", listOfUsers);

}

else if (state === "ERROR") {

var errors = response.getError();

if (errors) {

if (errors[0] && errors[0].message) {

this.handleErrors(component,errors[0].message);

}

}

}

});

$A.enqueueAction(action);

},

getPriority : function(component,event) {

var action = component.get("c.getProjectPriority");

action.setCallback(this, function(response) {

var state = response.getState();

console.log(state);

if (component.isValid() && state === "SUCCESS") {

var listOfPriority = JSON.parse(response.getReturnValue());

component.set("v.listOfPriority", listOfPriority);

}

else if (state === "ERROR") {

var errors = response.getError();

if (errors) {

if (errors[0] && errors[0].message) {

this.handleErrors(component,errors[0].message);

}

}

}

});

$A.enqueueAction(action);

},

getProjectList : function(component,event) {

var action = component.get("c.getProjectList");

action.setCallback(this, function(response) {

var state = response.getState();

console.log(state);

if (component.isValid() && state === "SUCCESS") {

var listOfProject = JSON.parse(response.getReturnValue());

component.set("v.listOfProject", listOfProject);

}

else if (state === "ERROR") {

var errors = response.getError();

if (errors) {

if (errors[0] && errors[0].message) {

this.handleErrors(component,errors[0].message);

}

}

}

});

$A.enqueueAction(action);

},

\*/

saveCustomer : function(component, event, helper) {

debugger;

var action = component.get("c.saveCustomerInfo");

action.setParams({

"CustomerJSON" : JSON.stringify(component.get("v.newCustomer"))

});

action.setCallback(this, function(response) {

var state = response.getState();

console.log(state);

if (state === "SUCCESS") {

var Customer=JSON.parse(response.getReturnValue());

console.log(Customer);

component.set("v.listOfCustomers", Customer);

alert("CUSTOMER ADDED");

}

else if (state === "ERROR") {

var errors = response.getError();

if (errors) {

if (errors[0] && errors[0].message) {

console.log(errors[0].message);

this.handleErrors(component,errors[0].message);

}

}

}

});

$A.enqueueAction(action);

},

handleErrors : function(component,errorMessage) {

component.set('v.isError',true);

component.set('v.errorMessage',errorMessage);

},

})

# ViewCustomer.cmp

<aura:component controller="GVCtrl">

<aura:attribute name="listOfCustomers" type="Object[]" />

<aura:attribute name="selectedCustomer" type="Object" />

<aura:attribute name="displayCustomerList" type="Boolean" default="true" />

<aura:attribute name="displayCustomerDetail" type="Boolean" default="false" />

<aura:attribute name="editCustomer" type="Boolean" default="false" />

<aura:if isTrue="{! v.displayCustomerList}" >

<aura:iteration items="{! v.listOfCustomers}" var="Customer" indexVar="CustomerIndex">

<div class="slds-box slds-clearfix">

{! Customer.Name}

<lightning:buttonIcon class="slds-button\_\_icon\_small slds-float\_right" value="{!CustomerIndex}"

iconName="utility:preview" size="small" alternativeText="View"

onclick="{!c.showCustomerDetails}" variant="brand"/>

<lightning:buttonIcon class="slds-button\_\_icon\_small slds-float\_right" value="{!CustomerIndex}"

iconName="utility:edit" size="small" alternativeText="Edit"

onclick="{!c.editCustomerDetails}" variant="success"/>

<lightning:buttonIcon class="slds-button\_\_icon\_small slds-float\_right" value="{!CustomerIndex}"

iconName="utility:delete" size="small" alternativeText="Delete"

onclick="{!c.removeCustomer}" variant="destructive"/>

</div>

</aura:iteration>

</aura:if>

<aura:if isTrue="{! v.displayCustomerDetail}" >

<div class="slds-box">

<div class="button\_group slds-clearfix">

<lightning:buttonIcon class="slds-button\_\_icon\_small slds-float\_left" value="{!CustomerIndex}"

iconName="utility:back" size="small" alternativeText="Back"

onclick="{!c.showCustomerList}" variant="brand"/>

</div>

<div class="detailBlock slds-m-top\_x-small">

<c:ViewCustomerDetail Customer="{!v.selectedCustomer}" />

</div>

</div>

</aura:if>

<aura:if isTrue="{! v.editCustomer}" >

<c:CreateCustomer newCustomer="{!v.selectedCustomer}" />

</aura:if >

</aura:component>

# ViewCustomerController.js

({

showCustomerDetails : function(component, event, helper) {

component.set("v.displayCustomerList", false);

var CustomerIndex = event.getSource().get("v.value");

console.log('CustomerIndex', CustomerIndex);

var listOfCustomers = component.get("v.listOfCustomers");

component.set("v.selectedCustomer", listOfCustomers[CustomerIndex]);

component.set("v.displayCustomerDetail", true);

component.set("v.editCustomer", false);

},

showCustomerList : function(component, event, helper) {

component.set("v.displayCustomerDetail", false);

component.set("v.displayCustomerList", true);

component.set("v.editCustomer", false);

},

editCustomerDetails: function(component, event, helper) {

var CustomerIndex = event.getSource().get("v.value");

console.log('CustomerIndex', CustomerIndex);

var listOfCustomers = component.get("v.listOfCustomers");

component.set("v.selectedCustomer", listOfCustomers[CustomerIndex]);

component.set("v.editCustomer", true);

},

removeCustomer: function(component, event, helper) {

console.log("TTPLC:25");

var CustomerIndex = event.getSource().get("v.value");

console.log('CustomerIndex', CustomerIndex);

var listOfCustomers = component.get("v.listOfCustomers");

var selectedCustomer = listOfCustomers[CustomerIndex];

var action=component.get("c.deleteCustomerDetail");

action.setParams({

'Customer' : JSON.stringify(selectedCustomer)

});

action.setCallback(this, function(response){

var state=response.getState();

if(component.isValid() && state==='SUCCESS'){

var listOfCustomers = JSON.parse(response.getReturnValue());

console.log("TTPLC:36",state);

component.set("v.listOfCustomers", listOfCustomers);

}else{

alert(state);

}

});

$A.enqueueAction(action);

}

})

# ViewCustomerDetail.cmp

<aura:component >

<aura:attribute name="Customer" type="Object" />

<div class="slds-border\_top">

<div class="slds-text-heading\_medium">{!v.Customer.Name}</div>

<div class="slds-grid slds-wrap">

<div class="slds-size\_3-of-12">Phone No </div>

<div class="slds-size\_1-of-12">: </div>

<div class="slds-size\_8-of-12">{!v.Customer.Name}</div>

<div class="slds-size\_3-of-12">Email </div>

<div class="slds-size\_1-of-12">: </div>

<div class="slds-size\_8-of-12">{!v.Customer.Email\_\_c}</div>

<div class="slds-size\_3-of-12">Weight Of Gold</div>

<div class="slds-size\_1-of-12">: </div>

<div class="slds-size\_8-of-12">{!v.Customer.Weight\_of\_Gold\_\_c}</div>

<div class="slds-size\_3-of-12">Time Period </div>

<div class="slds-size\_1-of-12">: </div>

<div class="slds-size\_8-of-12">{!v.Customer.Time\_Period\_\_c}</div>

<!-- <div class="slds-size\_3-of-12">Technology </div>

<div class="slds-size\_1-of-12">: </div>

<div class="slds-size\_8-of-12">{!v.project.Technology\_\_c}</div> -->

</div>

</div>

</aura:component>

# CustomerDao.apxc

public class CustomerDao {

public static List<Customers\_\_c> getAllCustomers(){

List<Customers\_\_c> listOfCustomers = [SELECT Name, D\_O\_B\_\_c, Date\_of\_Loan\_Sanction\_\_c, Email\_\_c, Phone\_No\_\_c FROM Customers\_\_c];

return listOfCustomers;

}

public static Customers\_\_c save(Customers\_\_c Customer){

insert Customer;

return Customer;

}

public static List<Customers\_\_c> removeCustomer(Customers\_\_c Customer){

delete Customer;

return getAllCustomers();

}

}

# Loan.cmp

<aura:component >

<aura:attribute name="listOfLoans" type="Object[]" />

<div class="slds-grid">

<lightning:tabset variant="{!'scoped'}">

<lightning:tab label="Create"><c:CreateLoan listOfLoans="{!v.listOfLoans}" /></lightning:tab>

<lightning:tab label="View">

<c:ViewLoan listOfLoans="{!v.listOfLoans}" />

</lightning:tab>

</lightning:tabset>

</div>

</aura:component>

# CreateLoan.cmp

<aura:component implements="force:appHostable" controller="GVCtrl">

<aura:attribute name="showCreateLoanForm" type="Boolean" default="false" />

<aura:attribute name="isEditLoan" type="Boolean" default="false" />

<aura:attribute name="listOfLoans" type="Object[]" />

<aura:attribute name="ErrorMessage" type="String" default="date Invalid"/>

<aura:attribute name="isError" type="Boolean" default="false"/>

<aura:attribute name="newLoan" type="Loan\_\_c" default="{'sobjectType':'Loan\_\_c',

'Name':''}"/>

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

<div class="slds-form slds-form\_stacked">

<div class="slds-form-element">

<lightning:input type="text" label="Loan Details Name" value="{! v.newLoan.Name}"

required="true"

pattern="^[A-Za-z ]\*$"

messagewhenPatternMismatch="Only Alphabets are allowed" />

</div>

<div class="slds-form-element">

<lightning:input type="text" label="Weight of Gold" value="{! v.newLoan.Weight\_of\_Gold\_\_c}" />

</div>

<div class="slds-form-element">

<lightning:input type="number" label="Gold rate" value="{! v.newLoan.Gold\_Rate\_\_c}" />

</div>

<div class="slds-form-element">

<lightning:input type="text" label="Time period(enter in months)"

value="{! v.newLoan.Time\_Period\_\_c}" />

</div>

<div class="slds-form-element">

<lightning:input type="text" label="Interest rate"

value="{! v.newLoan.Interest\_Rate\_\_c}" />

</div>

<div class="slds-form-element">

<lightning:input type="text" label="Guaranter"

value="{! v.newLoan.Guaranter\_\_c}" />

</div>

<div class="slds-form-element">

<lightning:input type="number" label="Deadline Exceeded Interest Rate"

value="{! v.newLoan.Deadline\_exceeded\_interest\_rate\_\_c}" />

</div>

<!-- <div class="slds-form-element">

<lightning:dualListbox name="Technology" label= "Technology" sourceLabel="Available"

selectedLabel="Selected" options="{!v.listOfProjectTech}" onchange=""/>

</div>

<div class="slds-form-element">

<lightning:textarea name="Description" value="{!v.newProject.Description\_\_c}" label="Description" placeholder="type here..."/>

</div>

<div class="slds-form-element">

<lightning:button variant="brand" label="Save" title="Save" onclick="{!c.saveProject}" />

</div>-->

<div class="slds-form-element">

<lightning:button variant="brand" label="Save" title="Save" onclick="{!c.AddLoan}" />

</div>

</div>

</aura:component>

# CreateLoanController.js

({

AddLoan : function(component, event, helper) {

helper.saveLoan(component, event, helper);

}

})

# CreateLoanHelper.js

({

/\*getInitData : function(component,event) {

var action = component.get("c.getInitialData");

action.setCallback(this, function(response) {

var state = response.getState();

console.log(state);

if (component.isValid() && state === "SUCCESS") {

var result = JSON.parse(response.getReturnValue());

component.set("v.listOfTaskType", JSON.parse(result['PROJECT\_TYPE\_LIST']));

component.set("v.listOfTaskStatus", JSON.parse(result['STATUS\_LIST']));

component.set("v.listOfUsers", JSON.parse(result['USER\_LIST']));

component.set("v.listOfPriority", JSON.parse(result['PROJECT\_PRIORITY\_LIST']));

component.set("v.listOfProject", JSON.parse(result['PROJECT\_LIST']));

}

else if (state === "ERROR") {

var errors = response.getError();

if (errors) {

if (errors[0] && errors[0].message) {

this.handleErrors(component,errors[0].message);

}

}

}

});

$A.enqueueAction(action);

},

getType : function(component,event) {

var action = component.get("c.getProjectType");

action.setCallback(this, function(response) {

var state = response.getState();

console.log(state);

if (component.isValid() && state === "SUCCESS") {

var listOfTaskType = JSON.parse(response.getReturnValue());

component.set("v.listOfTaskType", listOfTaskType);

}

else if (state === "ERROR") {

var errors = response.getError();

if (errors) {

if (errors[0] && errors[0].message) {

this.handleErrors(component,errors[0].message);

}

}

}

});

$A.enqueueAction(action);

},

getStatus : function(component, event) {

var action = component.get("c.getStatus");

action.setCallback(this, function(response) {

var state = response.getState();

console.log(state);

if (component.isValid() && state === "SUCCESS") {

var listOfTaskStatus = JSON.parse(response.getReturnValue());

component.set("v.listOfTaskStatus", listOfTaskStatus);

}

else if (state === "ERROR") {

var errors = response.getError();

if (errors) {

if (errors[0] && errors[0].message) {

this.handleErrors(component,errors[0].message);

}

}

}

});

$A.enqueueAction(action);

},

getUsers : function(component,event) {

var action = component.get("c.getUsers");

action.setCallback(this, function(response) {

var state = response.getState();

console.log(state);

if (component.isValid() && state === "SUCCESS") {

var listOfUsers = JSON.parse(response.getReturnValue());

component.set("v.listOfUsers", listOfUsers);

}

else if (state === "ERROR") {

var errors = response.getError();

if (errors) {

if (errors[0] && errors[0].message) {

this.handleErrors(component,errors[0].message);

}

}

}

});

$A.enqueueAction(action);

},

getPriority : function(component,event) {

var action = component.get("c.getProjectPriority");

action.setCallback(this, function(response) {

var state = response.getState();

console.log(state);

if (component.isValid() && state === "SUCCESS") {

var listOfPriority = JSON.parse(response.getReturnValue());

component.set("v.listOfPriority", listOfPriority);

}

else if (state === "ERROR") {

var errors = response.getError();

if (errors) {

if (errors[0] && errors[0].message) {

this.handleErrors(component,errors[0].message);

}

}

}

});

$A.enqueueAction(action);

},

getProjectList : function(component,event) {

var action = component.get("c.getProjectList");

action.setCallback(this, function(response) {

var state = response.getState();

console.log(state);

if (component.isValid() && state === "SUCCESS") {

var listOfProject = JSON.parse(response.getReturnValue());

component.set("v.listOfProject", listOfProject);

}

else if (state === "ERROR") {

var errors = response.getError();

if (errors) {

if (errors[0] && errors[0].message) {

this.handleErrors(component,errors[0].message);

}

}

}

});

$A.enqueueAction(action);

},\*/

saveLoan : function(component, event, helper) {

debugger;

var action = component.get("c.saveLoanInfo");

action.setParams({

"LoanJSON" : JSON.stringify(component.get("v.newLoan"))

});

action.setCallback(this, function(response) {

var state = response.getState();

console.log(state);

if (state === "SUCCESS") {

var Loan=JSON.parse(response.getReturnValue());

console.log(Loan);

component.set("v.listOfLoans", Loan);

alert("LOAN ADDED");

}

else if (state === "ERROR") {

var errors = response.getError();

if (errors) {

if (errors[0] && errors[0].message) {

console.log(errors[0].message);

this.handleErrors(component,errors[0].message);

}

}

}

});

$A.enqueueAction(action);

},

handleErrors : function(component,errorMessage) {

component.set('v.isError',true);

component.set('v.errorMessage',errorMessage);

},

})

# ViewLoan.cmp

<aura:component controller="GVCtrl">

<aura:attribute name="listOfLoans" type="Object[]" />

<aura:attribute name="selectedLoan" type="Object" />

<aura:attribute name="displayLoanList" type="Boolean" default="true" />

<aura:attribute name="displayLoanDetail" type="Boolean" default="false" />

<aura:attribute name="editLoan" type="Boolean" default="false" />

<aura:if isTrue="{! v.displayLoanList}" >

<aura:iteration items="{! v.listOfLoans}" var="Loan" indexVar="LoanIndex">

<div class="slds-box slds-clearfix">

{! Loan.Name}

<lightning:buttonIcon class="slds-button\_\_icon\_small slds-float\_right" value="{!LoanIndex}"

iconName="utility:preview" size="small" alternativeText="View"

onclick="{!c.showLoanDetails}" variant="brand"/>

<lightning:buttonIcon class="slds-button\_\_icon\_small slds-float\_right" value="{!LoanIndex}"

iconName="utility:edit" size="small" alternativeText="Edit"

onclick="{!c.editLoanDetails}" variant="success"/>

<lightning:buttonIcon class="slds-button\_\_icon\_small slds-float\_right" value="{!LoanIndex}"

iconName="utility:delete" size="small" alternativeText="Delete"

onclick="{!c.removeLoan}" variant="destructive"/>

</div>

</aura:iteration>

</aura:if>

<aura:if isTrue="{! v.displayLoanDetail}" >

<div class="slds-box">

<div class="button\_group slds-clearfix">

<lightning:buttonIcon class="slds-button\_\_icon\_small slds-float\_left" value="{!LoanIndex}"

iconName="utility:back" size="small" alternativeText="Back"

onclick="{!c.showLoanList}" variant="brand"/>

</div>

<div class="detailBlock slds-m-top\_x-small">

<c:ViewLoanDetail Loan="{!v.selectedLoan}" />

</div>

</div>

</aura:if>

<aura:if isTrue="{! v.editLoan}" >

<c:CreateLoan newLoan="{!v.selectedLoan}" />

</aura:if >

</aura:component>

# ViewLoanController.js

({

showLoanDetails : function(component, event, helper) {

component.set("v.displayLoanList", false);

var LoanIndex = event.getSource().get("v.value");

console.log('LoanIndex', LoanIndex);

var listOfLoans = component.get("v.listOfLoans");

component.set("v.selectedLoan", listOfLoans[LoanIndex]);

component.set("v.displayLoanDetail", true);

component.set("v.editLoan", false);

},

showLoanList : function(component, event, helper) {

component.set("v.displayLoanDetail", false);

component.set("v.displayLoanList", true);

component.set("v.editLoan", false);

},

editLoanDetails: function(component, event, helper) {

var LoanIndex = event.getSource().get("v.value");

console.log('LoanIndex', LoanIndex);

var listOfLoans = component.get("v.listOfLoans");

component.set("v.selectedLoan", listOfLoans[LoanIndex]);

component.set("v.editLoan", true);

},

removeLoan: function(component, event, helper) {

console.log("TTPLC:25");

var LoanIndex = event.getSource().get("v.value");

console.log('LoanIndex', LoanIndex);

var listOfLoans = component.get("v.listOfLoans");

var selectedLoan = listOfLoans[LoanIndex];

var action=component.get("c.deleteLoanDetail");

action.setParams({

'Loan' : JSON.stringify(selectedLoan)

});

action.setCallback(this, function(response){

var state=response.getState();

if(component.isValid() && state==='SUCCESS'){

var listOfLoans = JSON.parse(response.getReturnValue());

console.log("TTPLC:36",state);

component.set("v.listOfLoans", listOfLoans);

}else{

alert(state);

}

});

$A.enqueueAction(action);

}

})

# ViewLoanDetail.cmp

<aura:component >

<aura:attribute name="Loan" type="Object" />

<div class="slds-border\_top">

<div class="slds-text-heading\_medium">{!v.Loan.Name}</div>

<div class="slds-grid slds-wrap">

<div class="slds-size\_3-of-12">Weight Of Gold </div>

<div class="slds-size\_1-of-12">: </div>

<div class="slds-size\_8-of-12">{!v.Loan.Weight\_of\_Gold\_\_c }</div>

<div class="slds-size\_3-of-12">Time period</div>

<div class="slds-size\_1-of-12">: </div>

<div class="slds-size\_8-of-12">{!v.Loan.Time\_Period\_\_c}</div>

<!-- <div class="slds-size\_3-of-12">Technology </div>

<div class="slds-size\_1-of-12">: </div>

<div class="slds-size\_8-of-12">{!v.project.Technology\_\_c}</div> -->

</div>

</div>

</aura:component>

# LoanDao.apxc

public class LoanDao {

public static List<Loan\_\_c> getAllLoans(){

List<Loan\_\_c> listOfLoans=[SELECT Name, Description\_\_c, Interest\_Rate\_\_c FROM Loan\_\_c];

return listOfLoans;

}

public static Loan\_\_c save(Loan\_\_c Loan){

insert Loan;

return Loan;

}

public static List<Loan\_\_c> removeLoan(Loan\_\_c Loan){

delete Loan;

return getAllLoans();

}

}

# Employee.cmp

<aura:component >

<aura:attribute name="listOfEmployees" type="Object[]" />

<div class="slds-grid">

<lightning:tabset variant="{!'scoped'}">

<lightning:tab label="Create"><c:CreateEmployee listOfEmployees="{!v.listOfEmployees}"/></lightning:tab>

<lightning:tab label="View">

<c:ViewEmployee listOfEmployees="{!v.listOfEmployees}" />

</lightning:tab>

</lightning:tabset>

</div>

</aura:component>

# CreateEmployee.cmp

<aura:component implements="force:appHostable" controller="GVCtrl">

<aura:attribute name="showCreateEmployeeForm" type="Boolean" default="false" />

<aura:attribute name="isEditEmployee" type="Boolean" default="false" />

<aura:attribute name="listOfEmployees" type="Object[]" />

<aura:attribute name="ErrorMessage" type="String" default="date Invalid"/>

<aura:attribute name="isError" type="Boolean" default="false"/>

<aura:attribute name="newEmployee" type="Employee\_\_c" default="{'sobjectType':'Employee\_\_c',

'Name':''}"/>

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

<div class="slds-form slds-form\_stacked">

<div class="slds-form-element">

<lightning:input type="text" label="Employee Name" value="{! v.newEmployee.Name}"

required="true"

pattern="^[A-Za-z ]\*$"

messagewhenPatternMismatch="Only Alphabets are allowed" />

</div>

<div class="slds-form-element">

<lightning:input type="text" label="Employee ID" value="{! v.newEmployee.Employee\_ID\_\_c}" />

</div>

<div class="slds-form-element">

<lightning:input type="text" label="Current Contact No " value="{! v.newEmployee.Current\_Contact\_No\_\_c}"

pattern="^[0-9]\*$" minlength="10" maxlength="10"

messagewhenPatternMismatch="Only numbers and spaces are allowed" />

</div>

<div class="slds-form-element">

<lightning:input type="text" label="Permanent Contact No" value="{! v.newEmployee.Permanent\_Contact\_No\_\_c }"

pattern="^[0-9]\*$" minlength="10" maxlength="10"

messagewhenPatternMismatch="Only numbers and spaces are allowed" />

</div>

<div class="slds-form-element">

<lightning:input type="text" label="Email" value="{! v.newEmployee.Email\_\_c}" />

</div>

<div class="slds-form-element">

<lightning:input type="text" label="Current Address" value="{! v.newEmployee.Current\_Address\_\_c }"

pattern="^[A-Za-z0-9 - ]\*$"

messagewhenPatternMismatch="Only alphanumeric and are allowed" />

</div>

<div class="slds-form-element">

<lightning:input type="text" label="Permanent Address" value="{! v.newEmployee.Permanent\_Address\_\_c }"

pattern="^[A-Za-z0-9 - ]\*$"

messagewhenPatternMismatch="Only alphanumeric and are allowed" />

</div>

<div class="slds-form-element">

<lightning:input type="text" label="Branch " value="{! v.newEmployee.Branch\_\_c }" />

</div>

<div class="slds-form-element">

<lightning:input type="number" label="Salary " value="{! v.newEmployee.Salary\_\_c }" />

</div>

<div class="slds-form-element">

<lightning:input type="text" label="Password" value="{! v.newEmployee.Password\_\_c }" />

</div>

<div class="slds-form-element">

<lightning:button variant="brand" label="Save" title="Save" onclick="{!c.AddEmployee}" />

</div>

</div>

</aura:component>

# CreateEmployeeController.js

({

AddEmployee : function(component, event, helper) {

helper.saveEmployee(component, event, helper);

}

})

# CreateEmployeeHelper.js

({

/\*getInitData : function(component,event) {

var action = component.get("c.getInitialData");

action.setCallback(this, function(response) {

var state = response.getState();

console.log(state);

if (component.isValid() && state === "SUCCESS") {

var result = JSON.parse(response.getReturnValue());

component.set("v.listOfTaskType", JSON.parse(result['PROJECT\_TYPE\_LIST']));

component.set("v.listOfTaskStatus", JSON.parse(result['STATUS\_LIST']));

component.set("v.listOfUsers", JSON.parse(result['USER\_LIST']));

component.set("v.listOfPriority", JSON.parse(result['PROJECT\_PRIORITY\_LIST']));

component.set("v.listOfProject", JSON.parse(result['PROJECT\_LIST']));

}

else if (state === "ERROR") {

var errors = response.getError();

if (errors) {

if (errors[0] && errors[0].message) {

this.handleErrors(component,errors[0].message);

}

}

}

});

$A.enqueueAction(action);

},

getType : function(component,event) {

var action = component.get("c.getProjectType");

action.setCallback(this, function(response) {

var state = response.getState();

console.log(state);

if (component.isValid() && state === "SUCCESS") {

var listOfTaskType = JSON.parse(response.getReturnValue());

component.set("v.listOfTaskType", listOfTaskType);

}

else if (state === "ERROR") {

var errors = response.getError();

if (errors) {

if (errors[0] && errors[0].message) {

this.handleErrors(component,errors[0].message);

}

}

}

});

$A.enqueueAction(action);

},

getStatus : function(component, event) {

var action = component.get("c.getStatus");

action.setCallback(this, function(response) {

var state = response.getState();

console.log(state);

if (component.isValid() && state === "SUCCESS") {

var listOfTaskStatus = JSON.parse(response.getReturnValue());

component.set("v.listOfTaskStatus", listOfTaskStatus);

}

else if (state === "ERROR") {

var errors = response.getError();

if (errors) {

if (errors[0] && errors[0].message) {

this.handleErrors(component,errors[0].message);

}

}

}

});

$A.enqueueAction(action);

},

getUsers : function(component,event) {

var action = component.get("c.getUsers");

action.setCallback(this, function(response) {

var state = response.getState();

console.log(state);

if (component.isValid() && state === "SUCCESS") {

var listOfUsers = JSON.parse(response.getReturnValue());

component.set("v.listOfUsers", listOfUsers);

}

else if (state === "ERROR") {

var errors = response.getError();

if (errors) {

if (errors[0] && errors[0].message) {

this.handleErrors(component,errors[0].message);

}

}

}

});

$A.enqueueAction(action);

},

getPriority : function(component,event) {

var action = component.get("c.getProjectPriority");

action.setCallback(this, function(response) {

var state = response.getState();

console.log(state);

if (component.isValid() && state === "SUCCESS") {

var listOfPriority = JSON.parse(response.getReturnValue());

component.set("v.listOfPriority", listOfPriority);

}

else if (state === "ERROR") {

var errors = response.getError();

if (errors) {

if (errors[0] && errors[0].message) {

this.handleErrors(component,errors[0].message);

}

}

}

});

$A.enqueueAction(action);

},

getProjectList : function(component,event) {

var action = component.get("c.getProjectList");

action.setCallback(this, function(response) {

var state = response.getState();

console.log(state);

if (component.isValid() && state === "SUCCESS") {

var listOfProject = JSON.parse(response.getReturnValue());

component.set("v.listOfProject", listOfProject);

}

else if (state === "ERROR") {

var errors = response.getError();

if (errors) {

if (errors[0] && errors[0].message) {

this.handleErrors(component,errors[0].message);

}

}

}

});

$A.enqueueAction(action);

},\*/

saveEmployee : function(component, event, helper) {

debugger;

var action = component.get("c.saveEmployeeInfo");

action.setParams({

"EmployeeJSON" : JSON.stringify(component.get("v.newEmployee"))

});

action.setCallback(this, function(response) {

var state = response.getState();

console.log(state);

if (state === "SUCCESS") {

var Employee=JSON.parse(response.getReturnValue());

console.log(Employee);

component.set("v.listOfEmployees", Employee);

alert("EMPLOYEE ADDED");

}

else if (state === "ERROR") {

var errors = response.getError();

if (errors) {

if (errors[0] && errors[0].message) {

console.log(errors[0].message);

this.handleErrors(component,errors[0].message);

}

}

}

});

$A.enqueueAction(action);

},

handleErrors : function(component,errorMessage) {

component.set('v.isError',true);

component.set('v.errorMessage',errorMessage);

},

})

# ViewEmployee.cmp

<aura:component controller="GVCtrl">

<aura:attribute name="listOfEmployees" type="Object[]" />

<aura:attribute name="selectedEmployee" type="Object" />

<aura:attribute name="displayEmployeeList" type="Boolean" default="true" />

<aura:attribute name="displayEmployeeDetail" type="Boolean" default="false" />

<aura:attribute name="editEmployee" type="Boolean" default="false" />

<aura:if isTrue="{! v.displayEmployeeList}" >

<aura:iteration items="{! v.listOfEmployees}" var="Employee" indexVar="EmployeeIndex">

<div class="slds-box slds-clearfix">

{! Employee.Name}

<lightning:buttonIcon class="slds-button\_\_icon\_small slds-float\_right" value="{!EmployeeIndex}"

iconName="utility:preview" size="small" alternativeText="View"

onclick="{!c.showEmployeeDetails}" variant="brand"/>

<lightning:buttonIcon class="slds-button\_\_icon\_small slds-float\_right" value="{!EmployeeIndex}"

iconName="utility:edit" size="small" alternativeText="Edit"

onclick="{!c.editEmployeeDetails}" variant="success"/>

<lightning:buttonIcon class="slds-button\_\_icon\_small slds-float\_right" value="{!EmployeeIndex}"

iconName="utility:delete" size="small" alternativeText="Delete"

onclick="{!c.removeEmployee}" variant="destructive"/>

</div>

</aura:iteration>

</aura:if>

<aura:if isTrue="{! v.displayEmployeeDetail}" >

<div class="slds-box">

<div class="button\_group slds-clearfix">

<lightning:buttonIcon class="slds-button\_\_icon\_small slds-float\_left" value="{!EmployeeIndex}"

iconName="utility:back" size="small" alternativeText="Back"

onclick="{!c.showEmployeeList}" variant="brand"/>

</div>

<div class="detailBlock slds-m-top\_x-small">

<c:ViewEmployeeDetail Employee="{!v.selectedEmployee}" />

</div>

</div>

</aura:if>

<aura:if isTrue="{! v.editEmployee}" >

<c:CreateEmployee newEmployee="{!v.selectedEmployee}" />

</aura:if >

</aura:component>

# ViewEmployeeController.js

({

showEmployeeDetails : function(component, event, helper) {

component.set("v.displayEmployeeList", false);

var EmployeeIndex = event.getSource().get("v.value");

console.log('EmployeeIndex', EmployeeIndex);

var listOfEmployees = component.get("v.listOfEmployees");

component.set("v.selectedEmployee", listOfEmployees[EmployeeIndex]);

component.set("v.displayEmployeeDetail", true);

component.set("v.editEmployee", false);

},

showEmployeeList : function(component, event, helper) {

component.set("v.displayEmployeeDetail", false);

component.set("v.displayEmployeeList", true);

component.set("v.editEmployee", false);

},

editEmployeeDetails: function(component, event, helper) {

var EmployeeIndex = event.getSource().get("v.value");

console.log('EmployeeIndex', EmployeeIndex);

var listOfEmployees = component.get("v.listOfEmployees");

component.set("v.selectedEmployee", listOfEmployees[EmployeeIndex]);

component.set("v.editEmployee", true);

},

removeEmployee: function(component, event, helper) {

console.log("TTPLC:25");

var EmployeeIndex = event.getSource().get("v.value");

console.log('EmployeeIndex', EmployeeIndex);

var listOfEmployees = component.get("v.listOfEmployees");

var selectedEmployee = listOfEmployees[EmployeeIndex];

var action=component.get("c.deleteEmployeeDetail");

action.setParams({

'Employee' : JSON.stringify(selectedEmployee)

});

action.setCallback(this, function(response){

var state=response.getState();

if(component.isValid() && state==='SUCCESS'){

var listOfEmployees = JSON.parse(response.getReturnValue());

console.log("TTPLC:36",state);

component.set("v.listOfEmployees", listOfEmployees);

}else{

alert(state);

}

});

$A.enqueueAction(action);

}

})

# ViewEmployeeDetail.cmp

<aura:component >

<aura:attribute name="Employee" type="Object" />

<div class="slds-border\_top">

<div class="slds-text-heading\_medium">{!v.Employee.Name}</div>

<div class="slds-grid slds-wrap">

<div class="slds-size\_3-of-12">Employee ID </div>

<div class="slds-size\_1-of-12">: </div>

<div class="slds-size\_8-of-12">{!v.Employee.Employee\_ID\_\_c}</div>

<div class="slds-size\_3-of-12">Email </div>

<div class="slds-size\_1-of-12">: </div>

<div class="slds-size\_8-of-12">{!v.Employee.Email\_\_c}</div>

<div class="slds-size\_3-of-12">Salary </div>

<div class="slds-size\_1-of-12">: </div>

<div class="slds-size\_8-of-12">{!v.Employee.Salary\_\_c}</div>

<div class="slds-size\_3-of-12">Branch </div>

<div class="slds-size\_1-of-12">: </div>

<div class="slds-size\_8-of-12">{!v.Employee.Branch\_\_c}</div>

<!-- <div class="slds-size\_3-of-12">Technology </div>

<div class="slds-size\_1-of-12">: </div>

<div class="slds-size\_8-of-12">{!v.project.Technology\_\_c}</div> -->

</div>

</div>

</aura:component>

# EmployeeDao.apxc

public class EmployeeDao {

public static List<Employee\_\_c > getAllEmployees(){

List<Employee\_\_c> listOfEmployees=[SELECT Name, Branch\_\_c, Email\_\_c, Password\_\_c,Current\_Contact\_No\_\_c,Designation\_\_c,Employee\_ID\_\_c FROM Employee\_\_c ];

return listOfEmployees;

}

public static Employee\_\_c save(Employee\_\_c Employee){

insert Employee;

return Employee;

}

public static List<Employee\_\_c> removeEmployee(Employee\_\_c Employee){

delete Employee;

return getAllEmployees();

}

}

# Branch.cmp

<aura:component >

<aura:attribute name="listOfBranches" type="Object[]" />

<div class="slds-grid">

<lightning:tabset variant="{!'scoped'}">

<lightning:tab label="Create"><c:CreateBranch listOfBranches="{!v.listOfBranches}"/></lightning:tab>

<lightning:tab label="View">

<c:ViewBranch listOfBranches="{!v.listOfBranches}" />

</lightning:tab>

</lightning:tabset>

</div>

</aura:component>

# CreateBranch.cmp

<aura:component implements="force:appHostable" controller="GVCtrl">

<aura:attribute name="showCreateBranchForm" type="Boolean" default="false" />

<aura:attribute name="isEditBranch" type="Boolean" default="false" />

<aura:attribute name="listOfBranches" type="Object[]" />

<aura:attribute name="ErrorMessage" type="String" default="date Invalid"/>

<aura:attribute name="isError" type="Boolean" default="false"/>

<aura:attribute name="newBranch" type="Branch\_\_c" default="{'sobjectType':'Branch\_\_c',

'Name':''}"/>

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

<div class="slds-form slds-form\_stacked">

<div class="slds-form-element">

<lightning:input type="text" label="Branch Name" value="{! v.newBranch.Name}"

required="true"

pattern="^[A-Za-z ]\*$"

messagewhenPatternMismatch="Only Alphabets are allowed" />

</div>

<div class="slds-form-element">

<lightning:input type="text" label="Branch Code" value="{! v.newBranch.Branch\_Code\_\_c}"

pattern="^[0-9]\*$" minlength="10" maxlength="10"

messagewhenPatternMismatch="Only numbers and spaces are allowed" />

</div>

<div class="slds-form-element">

<lightning:input type="text" label="City" value="{! v.newBranch.City\_\_c}" />

</div>

<div class="slds-form-element">

<lightning:button variant="brand" label="Save" title="Save" onclick="{!c.AddBranch}" />

</div>

</div>

</aura:component>

# CreateBranchController.js

({

AddBranch : function(component, event, helper) {

helper.saveBranch(component, event, helper);

}

})

# CreateBranchHelper.js

({

/\*getInitData : function(component,event) {

var action = component.get("c.getInitialData");

action.setCallback(this, function(response) {

var state = response.getState();

console.log(state);

if (component.isValid() && state === "SUCCESS") {

var result = JSON.parse(response.getReturnValue());

component.set("v.listOfTaskType", JSON.parse(result['PROJECT\_TYPE\_LIST']));

component.set("v.listOfTaskStatus", JSON.parse(result['STATUS\_LIST']));

component.set("v.listOfUsers", JSON.parse(result['USER\_LIST']));

component.set("v.listOfPriority", JSON.parse(result['PROJECT\_PRIORITY\_LIST']));

component.set("v.listOfProject", JSON.parse(result['PROJECT\_LIST']));

}

else if (state === "ERROR") {

var errors = response.getError();

if (errors) {

if (errors[0] && errors[0].message) {

this.handleErrors(component,errors[0].message);

}

}

}

});

$A.enqueueAction(action);

},

getType : function(component,event) {

var action = component.get("c.getProjectType");

action.setCallback(this, function(response) {

var state = response.getState();

console.log(state);

if (component.isValid() && state === "SUCCESS") {

var listOfTaskType = JSON.parse(response.getReturnValue());

component.set("v.listOfTaskType", listOfTaskType);

}

else if (state === "ERROR") {

var errors = response.getError();

if (errors) {

if (errors[0] && errors[0].message) {

this.handleErrors(component,errors[0].message);

}

}

}

});

$A.enqueueAction(action);

},

getStatus : function(component, event) {

var action = component.get("c.getStatus");

action.setCallback(this, function(response) {

var state = response.getState();

console.log(state);

if (component.isValid() && state === "SUCCESS") {

var listOfTaskStatus = JSON.parse(response.getReturnValue());

component.set("v.listOfTaskStatus", listOfTaskStatus);

}

else if (state === "ERROR") {

var errors = response.getError();

if (errors) {

if (errors[0] && errors[0].message) {

this.handleErrors(component,errors[0].message);

}

}

}

});

$A.enqueueAction(action);

},

getUsers : function(component,event) {

var action = component.get("c.getUsers");

action.setCallback(this, function(response) {

var state = response.getState();

console.log(state);

if (component.isValid() && state === "SUCCESS") {

var listOfUsers = JSON.parse(response.getReturnValue());

component.set("v.listOfUsers", listOfUsers);

}

else if (state === "ERROR") {

var errors = response.getError();

if (errors) {

if (errors[0] && errors[0].message) {

this.handleErrors(component,errors[0].message);

}

}

}

});

$A.enqueueAction(action);

},

getPriority : function(component,event) {

var action = component.get("c.getProjectPriority");

action.setCallback(this, function(response) {

var state = response.getState();

console.log(state);

if (component.isValid() && state === "SUCCESS") {

var listOfPriority = JSON.parse(response.getReturnValue());

component.set("v.listOfPriority", listOfPriority);

}

else if (state === "ERROR") {

var errors = response.getError();

if (errors) {

if (errors[0] && errors[0].message) {

this.handleErrors(component,errors[0].message);

}

}

}

});

$A.enqueueAction(action);

},

getProjectList : function(component,event) {

var action = component.get("c.getProjectList");

action.setCallback(this, function(response) {

var state = response.getState();

console.log(state);

if (component.isValid() && state === "SUCCESS") {

var listOfProject = JSON.parse(response.getReturnValue());

component.set("v.listOfProject", listOfProject);

}

else if (state === "ERROR") {

var errors = response.getError();

if (errors) {

if (errors[0] && errors[0].message) {

this.handleErrors(component,errors[0].message);

}

}

}

});

$A.enqueueAction(action);

},\*/

saveBranch : function(component, event, helper) {

debugger;

var action = component.get("c.saveBranchInfo");

action.setParams({

"BranchJSON" : JSON.stringify(component.get("v.newBranch"))

});

action.setCallback(this, function(response) {

var state = response.getState();

console.log(state);

if (state === "SUCCESS") {

var Branch=JSON.parse(response.getReturnValue());

console.log(Branch);

component.set("v.listOfBranches", Branch);

alert("BRANCH ADDED");

}

else if (state === "ERROR") {

var errors = response.getError();

if (errors) {

if (errors[0] && errors[0].message) {

console.log(errors[0].message);

this.handleErrors(component,errors[0].message);

}

}

}

});

$A.enqueueAction(action);

},

handleErrors : function(component,errorMessage) {

component.set('v.isError',true);

component.set('v.errorMessage',errorMessage);

},

})

# ViewBrach.cmp

<aura:component controller="GVCtrl">

<aura:attribute name="listOfBranches" type="Object[]" />

<aura:attribute name="selectedBranch" type="Object" />

<aura:attribute name="displayBranchList" type="Boolean" default="true" />

<aura:attribute name="displayBranchDetail" type="Boolean" default="false" />

<aura:attribute name="editBranch" type="Boolean" default="false" />

<aura:if isTrue="{! v.displayBranchList}" >

<aura:iteration items="{! v.listOfBranches}" var="Branch" indexVar="BranchIndex">

<div class="slds-box slds-clearfix">

{! Branch.Name}

<lightning:buttonIcon class="slds-button\_\_icon\_small slds-float\_right" value="{!BranchIndex}"

iconName="utility:preview" size="small" alternativeText="View"

onclick="{!c.showBranchDetails}" variant="brand"/>

<lightning:buttonIcon class="slds-button\_\_icon\_small slds-float\_right" value="{!BranchIndex}"

iconName="utility:edit" size="small" alternativeText="Edit"

onclick="{!c.editBranchDetails}" variant="success"/>

<lightning:buttonIcon class="slds-button\_\_icon\_small slds-float\_right" value="{!BranchIndex}"

iconName="utility:delete" size="small" alternativeText="Delete"

onclick="{!c.removeBranch}" variant="destructive"/>

</div>

</aura:iteration>

</aura:if>

<aura:if isTrue="{! v.displayBranchDetail}" >

<div class="slds-box">

<div class="button\_group slds-clearfix">

<lightning:buttonIcon class="slds-button\_\_icon\_small slds-float\_left" value="{!BranchIndex}"

iconName="utility:back" size="small" alternativeText="Back"

onclick="{!c.showBranchList}" variant="brand"/>

</div>

<div class="detailBlock slds-m-top\_x-small">

<c:ViewBranchDetail Branch="{!v.selectedBranch}" />

</div>

</div>

</aura:if>

<aura:if isTrue="{! v.editBranch}" >

<c:CreateBranch newBranch="{!v.selectedBranch}" />

</aura:if >

</aura:component>

# ViewBranchController.js

({

showBranchDetails : function(component, event, helper) {

component.set("v.displayBranchList", false);

var BranchIndex = event.getSource().get("v.value");

console.log('BranchIndex', BranchIndex);

var listOfBranches = component.get("v.listOfBranches");

component.set("v.selectedBranch", listOfBranches[BranchIndex]);

component.set("v.displayBranchDetail", true);

component.set("v.editBranch", false);

},

showBranchList : function(component, event, helper) {

component.set("v.displayBranchDetail", false);

component.set("v.displayBranchList", true);

component.set("v.editBranch", false);

},

editBranchDetails: function(component, event, helper) {

var BranchIndex = event.getSource().get("v.value");

console.log('BranchIndex', BranchIndex);

var listOfBranches = component.get("v.listOfBranches");

component.set("v.selectedBranch", listOfBranches[BranchIndex]);

component.set("v.editBranch", true);

},

removeBranch: function(component, event, helper) {

console.log("TTPLC:25");

var BranchIndex = event.getSource().get("v.value");

console.log('BranchIndex', BranchIndex);

var listOfBranches = component.get("v.listOfBranches");

var selectedBranch = listOfBranches[BranchIndex];

var action=component.get("c.deleteBranchDetail");

action.setParams({

'Branch' : JSON.stringify(selectedBranch)

});

action.setCallback(this, function(response){

var state=response.getState();

if(component.isValid() && state==='SUCCESS'){

var listOfBranches = JSON.parse(response.getReturnValue());

console.log("TTPLC:36",state);

component.set("v.listOfBranches", listOfBranches);

}else{

alert(state);

}

});

$A.enqueueAction(action);

}

})

# ViewBranchDetail.cmp

<aura:component >

<aura:attribute name="Branch" type="Object" />

<div class="slds-border\_top">

<div class="slds-text-heading\_medium">{!v.Branch.Name}</div>

<div class="slds-grid slds-wrap">

<div class="slds-size\_3-of-12">Branch Code </div>

<div class="slds-size\_1-of-12">: </div>

<div class="slds-size\_8-of-12">{!v.Branch.Branch\_Code\_\_c}</div>

<div class="slds-size\_3-of-12">City </div>

<div class="slds-size\_1-of-12">: </div>

<div class="slds-size\_8-of-12">{!v.Branch.City\_\_c}</div>

<!-- <div class="slds-size\_3-of-12">Technology </div>

<div class="slds-size\_1-of-12">: </div>

<div class="slds-size\_8-of-12">{!v.project.Technology\_\_c}</div> -->

</div>

</div>

</aura:component>

# BranchDao.apxc

public class BranchDao {

public static List<Branch\_\_c> getAllBranches(){

List<Branch\_\_c> listOfBranches=[SELECT Name, Branch\_Code\_\_c , City\_\_c ,Invested\_Bank\_\_c FROM Branch\_\_c];

return listOfBranches;

}

public static Branch\_\_c save(Branch\_\_c Branch){

insert Branch;

return Branch;

}

public static List<Branch\_\_c> removeBranch(Branch\_\_c Branch){

delete Branch;

return getAllBranches();

}

}

# SIgnIn.cmp

<aura:component controller="GVCtrl">

<aura:attribute name="loginEmployee" type="Employee\_\_c" default="{'sobjectType':'Employee\_\_c'}" />

<aura:attribute name="EmployeeInfo" type="Employee\_\_c" default="{'sobjectType':'Employee\_\_c'}" />

<aura:attribute name="isLogin" type="Boolean" default="false" />

<aura:attribute name="isNotLogin" type="Boolean" default="true" />

<aura:attribute name="isError" type="Boolean" default="false" />

<aura:attribute name="errorMessage" type="String" default="" />

<aura:if isTrue="{!v.isNotLogin}">

<div class="slds-form slds-form\_stacked">

<div class="row">

<div class="col-sm-4">

<aura:if isTrue="{!v.isError}">

<span style="color:white;font-size:18px;">{!v.errorMessage}</span>

</aura:if>

</div>

<div class="col-sm-4">

<div class="slds-form-element">

<br/>

<br/>

<br/><br/><br/><br/><br/>

<lightning:input type="email" label="Employee Name" value="{! v.loginEmployee.Email\_\_c}"

/><br/>

</div>

<div class="slds-form-element">

<lightning:input type="text" label="Password" value="{! v.loginEmployee.Password\_\_c}"

/>

</div>

<div class="slds-form-element">

<lightning:button variant="brand" label="Sign in" title="Sign in" onclick="{!c.loginEmployeeProfile}" class="slds-align\_absolute-center"></lightning:button>

</div>

<br/><br/><br/><br/><br/><br/><br/><br/><br/><br/>

<br/>

</div></div>

</div>

</aura:if>

</aura:component>

# SignInCntroller.js

({

loginEmployeeProfile : function(component, event, helper) {

helper.getEmployeeLogin(component, event);

}

})

# SignInHelper.js

({

getEmployeeLogin : function(component, event) {

var Employee=component.get("v.loginEmployee");

if(Employee.Email\_\_c==null || Employee.Email\_\_c==""){

component.set("v.isError", true);

component.set("v.errorMessage", "Email can not be left blank!");

}else if(Employee.Password\_\_c==null || Employee.Password\_\_c==""){

component.set("v.isError", true);

component.set("v.errorMessage", "Password can not be left blank!");

}else if(Employee.Password\_\_c.length!=10){

component.set("v.isError", true);

component.set("v.errorMessage", "Password length must be 10!")

}else if(isNaN(Employee.Password\_\_c)){

component.set("v.isError", true);

component.set("v.errorMessage", "Password pattern is invalid!")

}else{

var action=component.get("c.loginAccount");

action.setParams({

'Employee' : JSON.stringify(Employee)

});

action.setCallback(this, function(response){

var state=response.getState();

if(component.isValid() && state==='SUCCESS'){

var login=JSON.parse(response.getReturnValue());

if(login==null || login==''){

alert("Wrong Credentials. Try Again!");

component.set("v.isLogin", false);

component.set("v.isNotLogin", true);

}else{

component.set("v.EmployeeInfo", login);

component.set("v.isLogin", true);

component.set("v.isNotLogin", false);

}

}else{

alert(state);

}

});

$A.enqueueAction(action);

}

}

})

# Rate.cmp

<aura:component >

<aura:attribute name="EmployeeInfo" type="Employee\_\_c" default="{'sobjectType':'Employee\_\_c'}" />

<div>

<table>

<tr>

<td>Name:</td>

<td>{!v.EmployeeInfo.Name}</td>

</tr>

<tr>

<td>Email:</td>

<td>{!v.EmployeeInfo.Email\_\_c}</td>

</tr>

</table>

</div>

</aura:component>