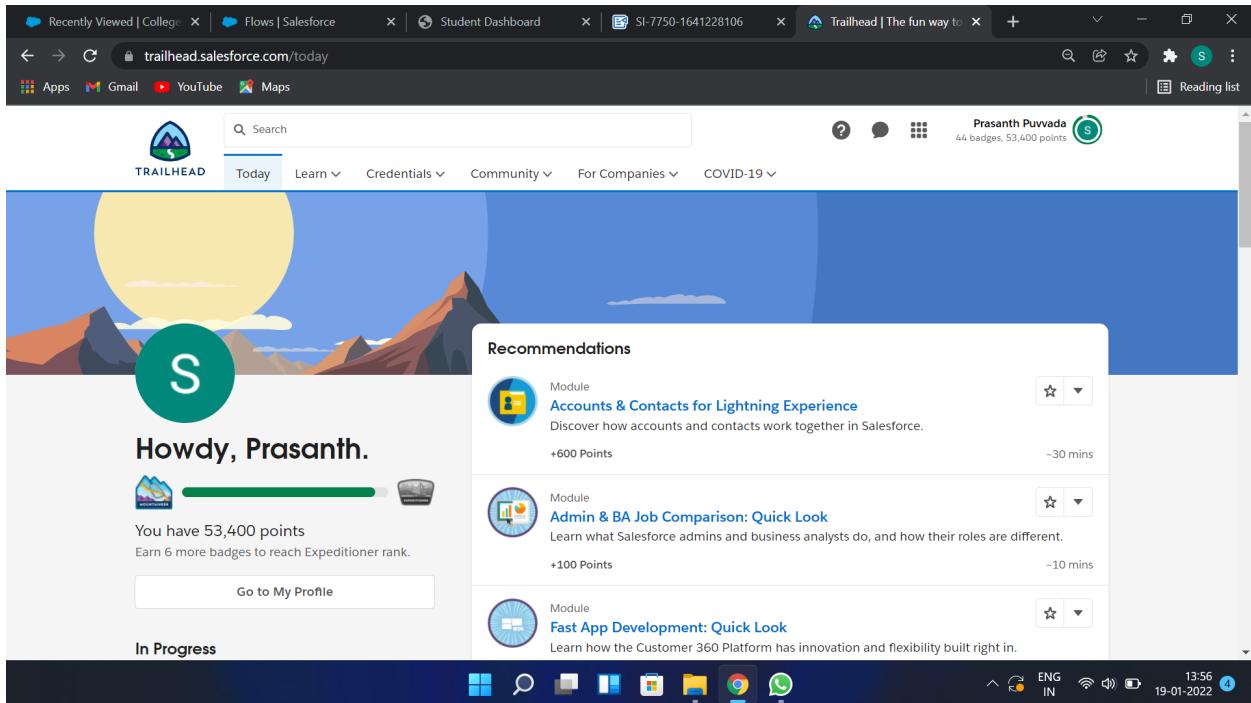


COLLEGE MANAGEMENT APPLICATION

Day1: Activities

1. Created my Salesforce Development org account
2. Activated my account
3. Logged into my account

Upload the Screenshot of Milestone:



Day2: Activities

Topic: Setting up First App, Custom Object and Fields Creation

Milestone: Custom Object Creation and Fields Creation

Detailed Description:

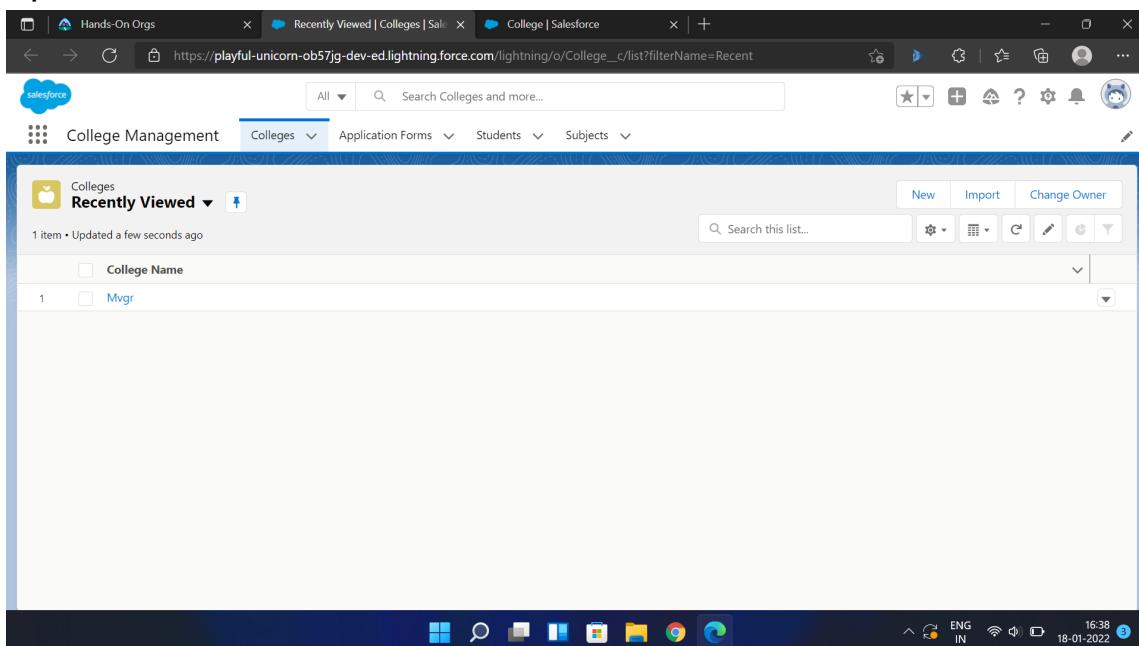
In this, Firstly Create an Application called College Management Application and then create the Objects which are required for the Application.

Create the Objects named

1. College
2. Application Form
3. Subjects
4. Students

1. Create the fields for College Object like Record Info, College Fees, Hostel Fees, College Name, EMail, Capacity of Students.
2. Create the fields for Application Form like Application Form ID, Address, College, College Fees, Hostel Fees, date of Birth, Email, Guardian Name, Looking For Hostel Stay, Ready To Join, Student Name.
3. Create the Fields for Student Object like Student Name, Address, Application Form, College Name, Date Of Birth, Guardian Name, Phone.
4. Create the Fields for Subject Object like Subject Id, Paper 1, Paper2, Student.

Upload the Screenshot of Milestone:



Hands-On Orgs | Recently Viewed | Colleges | Sales | College | Salesforce

https://playful-unicorn-ob57jg-dev-ed.lightning.force.com/lightning/setup/ObjectManager/015j000000RQE5/Fields...

Setup Home Object Manager

College

SETUP > OBJECT MANAGER

Fields & Relationships

Details

Page Layouts

Lightning Record Pages

Buttons, Links, and Actions

Compact Layouts

Field Sets

Object Limits

Record Types

Related Lookup Filters

Search Layouts

Search Layouts for Salesforce Classic

Restriction Rules

Triggers

Fields & Relationships

10 Items. Sorted by Field Label

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Capacity Of Students	Capacity__Of_Students__c	Picklist	College Name	
College Fees	College_Fees__c	Currency(7, 2)		
College Name	College__Name__c	Picklist		
College Name	Name	Text(80)		✓
Created By	CreatedById	Lookup(User)		
Email	Email__c	Picklist	College Name	
Hostel Fees	Hostel_Fees__c	Currency(6, 2)		
Last Modified By	LastModifiedById	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		✓
Record Info	Record_Info__c	Text(255)		

16:39 ENG IN 18-01-2022

Hands-On Orgs | Recently Viewed | Colleges | Sales | Application Form | Salesforce | New Tab

playful-unicorn-ob57jg-dev-ed.lightning.force.com/lightning/setup/ObjectManager/015j000000RQEe/FieldsAndRelationships/view

Gmail YouTube Maps

Setup Home Object Manager

Application Form

SETUP > OBJECT MANAGER

Fields & Relationships

Details

Page Layouts

Lightning Record Pages

Buttons, Links, and Actions

Compact Layouts

Field Sets

Object Limits

Record Types

Related Lookup Filters

Search Layouts

Search Layouts for Salesforce Classic

Restriction Rules

Triggers

Validation Rules

Fields & Relationships

15 Items. Sorted by Field Label

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Address	Address__c	Text(255)		
Application Form ID	Application_Form_ID__c	Auto Number		
Application Form Name	Name	Text(80)		✓
College	College__c	Master-Detail(College)		✓
College Fees	College_Fees__c	Formula (Currency)		
Created By	CreatedById	Lookup(User)		
date of Birth	date_of_Birth__c	Date		
Email	Email__c	Email (Unique)		✓
Guardian Name	Guardian_Name__c	Text(30)		
Hostel Fees	Hostel_Fees__c	Formula (Currency)		
Last Modified By	LastModifiedById	Lookup(User)		
Looking For Hostel Stay	Looking_For_Hostel_Stay__c	Checkbox		

16:39 ENG IN 18-01-2022

The screenshot shows the Salesforce Object Manager interface. The URL in the browser is <playful-unicorn-ob57jg-dev-ed.lightning.force.com/lightning/setup/ObjectManager/01I5J000000RRS1/FieldsAndRelationships/view>. The page title is "Student | Salesforce". The main content area is titled "Fields & Relationships" and displays a table of 11 items, sorted by Field Label. The columns in the table are FIELD LABEL, FIELD NAME, DATA TYPE, CONTROLLING FIELD, and INDEXED. The fields listed are:

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Address	Address__c	Text(255)		
Application Form	Application_Form__c	Lookup(Application Form)		
College Name	College_Name__c	Formula (Text)		
Created By	CreatedById	Lookup(User)		
Date Of Birth	DateOfBirth__c	Date		
Guardian Name	Guardian_Name__c	Text(30)		
Last Modified By	LastModifiedById	Lookup(User)		
Owner	OwnerId	Lookup(User/Group)		
Phone	Phone__c	Phone		
Student Name	Student_Name__c	Text(30)		
Student Name	Name	Text(30)		

Day 3:Activities

Topic: Adding Business Logic to Application

MileStone: Creating Global Picklist Values, Field Dependencies, Validation Rules, Process Automation and Student Record using Flow.

Detailed Description:

Firstly, Create the Global Picklist Values for the Application.

a.) College:-

Picklist Value Name:- College

Values:- MIT-HYD

MIT-BLR

MIT-MUM
MIT-MAA
MIT-DEL
MIT-CCU

b.) Paper 1:-

Picklist Value Name:- Paper1

Values:- APEX

JAVA
C
C++

c.) Paper 2:-

Picklist Value Name:- Paper2

Values:- MATHEMATICS

ENGLISH
STATISTICS

Now, Create the Field Dependencies.

- 1.) Create Field Dependency for College Object where Controlling Field is college Name and Dependent Field is E-Mail. Select E-Mail Ids according to Colleges.
- 2.) Create Field Dependency for College Object where Controlling Field is college Name and Dependent Field is Capacity of Students. Select values according to your wish.

Now, Create Validarion Rules.

- 1.) Create a validation rule on the college object such that the college name and record info should have the same name.
- 2.) Create a Validation Rule on the Application Form object to stop any modification on the application form once a student record is created.

Now, Process Automation.

1.)Create an automation process such that when the "ready to join" field is checked on the application form obj we need to create the student record automatically with the info specified in the application form record.

2.)Go to Setup-> select "Process Builder" from quick find.

Create a Process Builder on the "Application Form" object with a condition as "When a record change". And select "When a record is selected or edited".

a.) In the diamond shape box, select the criteria which trigger the process builder to fire. In our example, it is "When Ready to Join field is checked."

b.) Once the node is setup, click on the adjacent box called "Immediate Action". And select create a record on the student object.

Now, Student Record Using Flow.

1. First deactivate the process builder which we created earlier.
2. Now search for flos and select new flow->record triggered flow.
3. For object select application form, in configure trigger select when the record is updated for entry criteria select as ready to join equals to true.
4. Now create a variable named student in the resource section.
5. Now add the assignment.
6. Now add Create Records.

The screenshot shows the Salesforce Setup interface with the following details:

- Page Header:** Hands-On Orgs, Recently Viewed | Colleges | Sales, Picklist Value Sets | Salesforce, Student Dashboard.
- Left Sidebar:** Setup, Home, Object Manager, Data (Picklist Settings, State and Country/Territory, Picklists), Objects and Fields (Picklist Value Sets).
- Search Bar:** Search Setup.
- Main Content Area:**
 - Picklist Value Sets:** Global picklist value sets let you share values across objects. Base custom picklist fields on a global value set to inherit its values. The value set is restricted so users can't add unapproved values through the API.
 - View:** All, Create New View.
 - Global Value Sets Table:**

Action	Label	Description
Edit Del	College Name	
Edit Del	Paper1	
Edit Del	Paper2	
- Bottom Status Bar:** Didn't find what you're looking for? Try using Global Search. ENG IN 18-01-2022 16:50.

The screenshot shows the Salesforce Setup interface with the following details:

- Page Header:** Hands-On Orgs, Recently Viewed | Colleges | Sales, College | Salesforce, Student Dashboard.
- Left Sidebar:** Setup, Home, Object Manager, Object Manager (College).
- Search Bar:** Search Setup.
- Main Content Area:**
 - College Field Dependencies:** This page allows you to define dependencies between fields (e.g., dependent picklists).
 - Field Dependencies Table:**

Action	Controlling Field	Dependent Field	Modified By
Edit Del	College Name	Email	Prasanth Puvada 1/13/2022, 7:39 PM
Edit Del	College Name	Capacity Of Students	Prasanth Puvada 1/13/2022, 7:40 PM
- Bottom Status Bar:** ENG IN 18-01-2022 16:53.

Hands-On Orgs | Recently Viewed | Colleges | Sales | College | Salesforce | Student Dashboard

https://playful-unicorn-ob57jg-dev-ed.lightning.force.com/lightning/setup/ObjectManager/01I5j000000RQE5/Valid...

Setup Home Object Manager

College Validation Rule

Validation Rule Edit

Rule Name: College_Name_Similar
Active:

Description:

Error Condition Formula

Example: Discount_Percent__c>30 | More Examples...
Display an error if Discount is more than 30%
If this formula expression is true, display the text defined in the Error Message area

Insert Field Insert Operator `TEXT(College_Name__c) <> Name`

Functions

- All Function Categories --
- ABS
- ADDMONTHS
- AND
- BEGINS
- BLANKVALUE
- BR

Insert Selected Function ABS(number)
Returns the absolute value of a number, a number without its sign
Help on this function

Help for this Page Quick Tips Operators & Functions

Required Information

16:52 ENG IN 18-01-2022

Hands-On Orgs | Recently Viewed | Colleges | Sales | Application Form | Salesforce | Student Dashboard

https://playful-unicorn-ob57jg-dev-ed.lightning.force.com/lightning/setup/ObjectManager/01I5j000000RQEe/Valid...

Setup Home Object Manager

Application Form Validation Rule

Validation Rule Edit

Rule Name: Stop_Modifications
Active:

Description:

Error Condition Formula

Example: Discount_Percent__c>30 | More Examples...
Display an error if Discount is more than 30%
If this formula expression is true, display the text defined in the Error Message area

Insert Field Insert Operator `AND(Ready_To_Join__c == true,
OR(ISCHANGED(Address__c),
ISCHANGED(College__c),
ISCHANGED(date_of_Birth__c),
ISCHANGED>Email__c),
ISCHANGED(Guardian_Name__c),
ISCHANGED(Phone__c))`

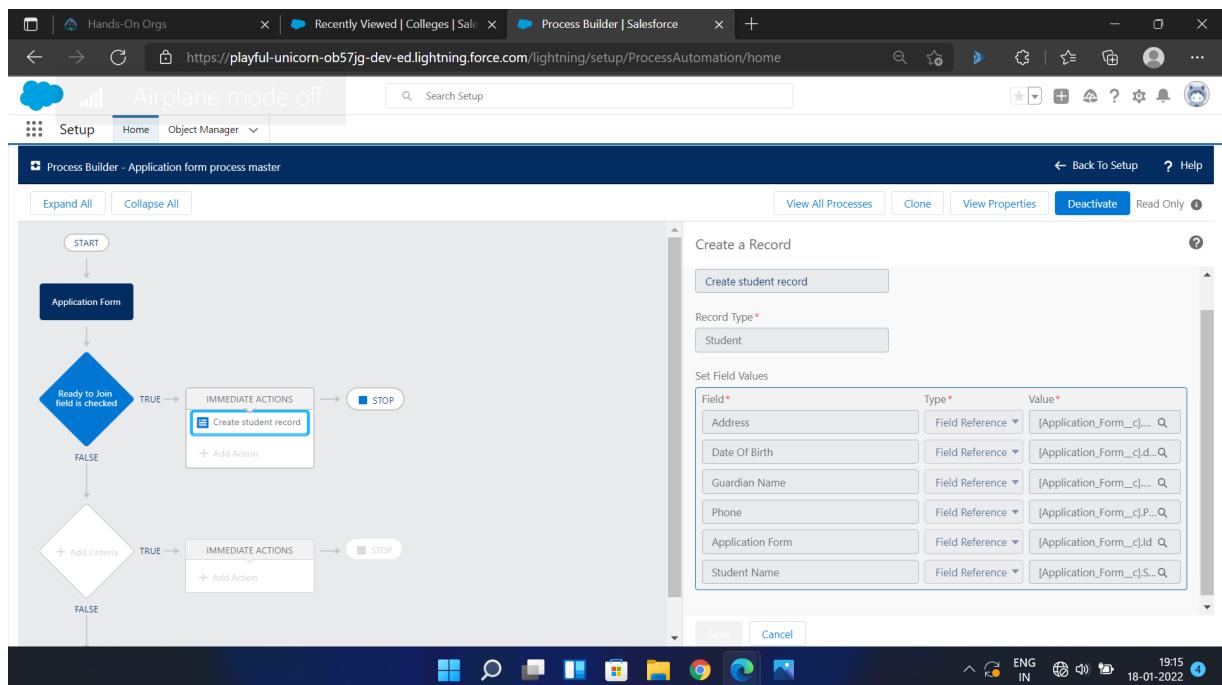
Functions

- All Function Categories --
- ABS
- ADDMONTHS
- AND
- BEGINS
- BLANKVALUE
- BR

Insert Selected Function ABS(number)
Returns the absolute value of a number, a number without its sign
Help on this function

Help for this Page Quick Tips Operators & Functions

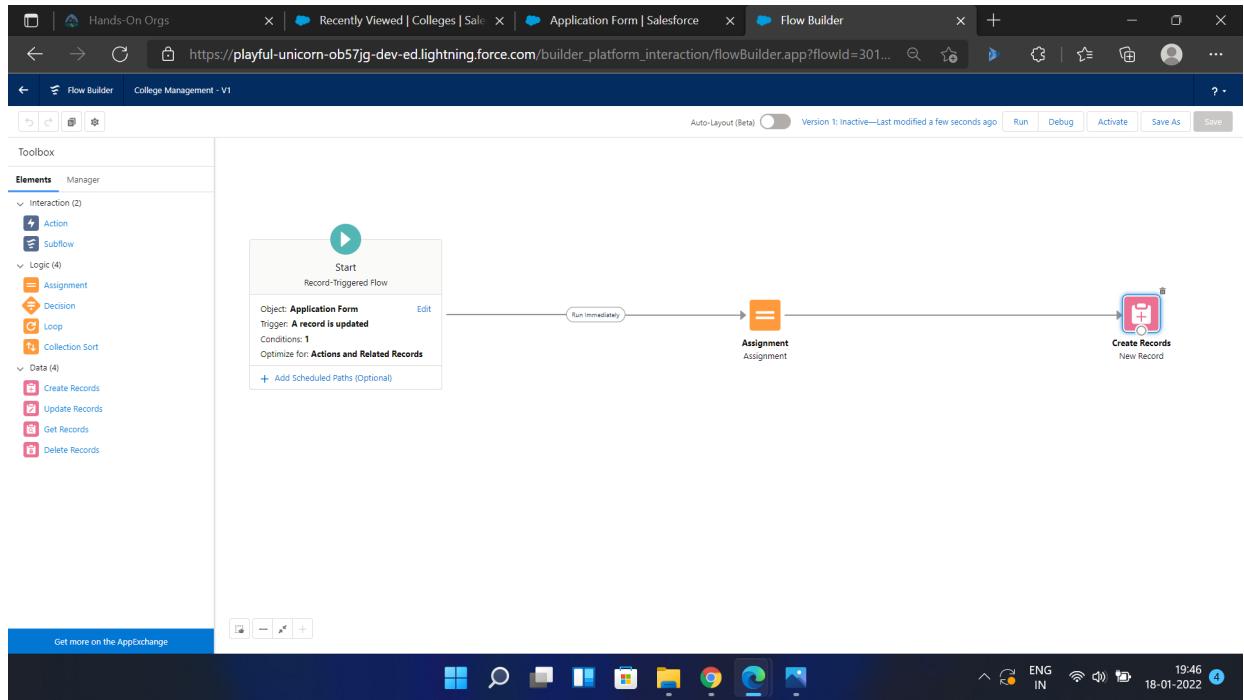
Required Information



The screenshot shows the Salesforce Flow Builder interface. A flow named "College Management - V1" is being edited. The main area is titled "Edit Assignment" and contains a table with seven rows, each defining a variable assignment. The variables and their assignments are:

Variable	Operator	Value
<code>A_a Student > Address</code>	Equals	<code>\$Record > Address</code>
<code>A_a Student > Application Form</code>	Operator	Value
<code>A_a Student > College Name</code>	Operator	Value
<code>Student > Date Of Birth</code>	Operator	Value
<code>A_a Student > Guardian Name</code>	Operator	Value
<code>A_a Student > Phone</code>	Operator	Value
<code>A_a Student > Student Name</code>	Operator	Value

At the bottom of the screen, there are buttons for "Run", "Debug", "Activate", "Save As", and "Save". The status bar at the bottom right shows the date and time as 19-01-2022 13:38.



Day 4:Activities

Topic: Batch Apex

Milestone: Create a Batchapex for Application Form

Detailed Description:

Firstly, Create a new Apex Class in Developer Console and enter the following code:

Now, change the Application Form API name to the name which we got while we have created the Application Form Object in College Management Application.

Save the File Name as ApplicationBatchTest and press Ctrl+E.

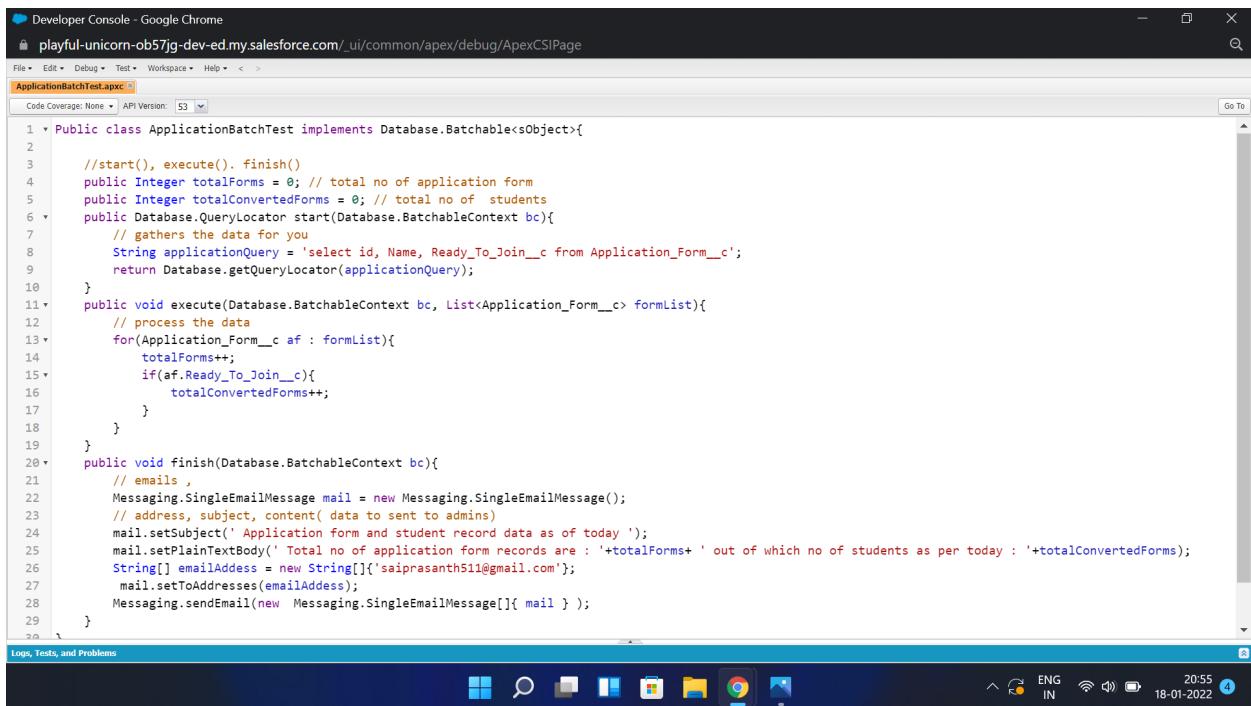
Then we get a Console. In it enter the Apex Code and execute it.

Now, remove "your E-Mail Address" text and put your personal Mail ID.

Press Execute.

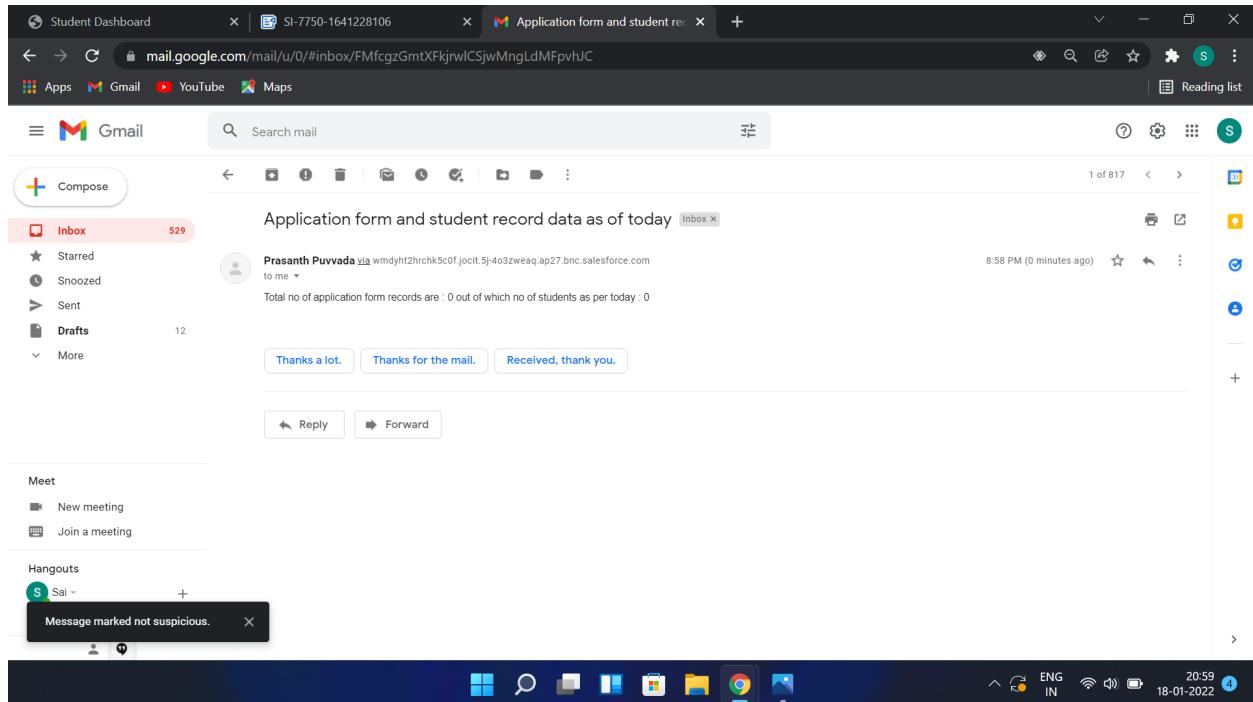
Then we get a Mail regarding the number of records which we have created in Application Form Object till date.

Upload the Screenshot:



The screenshot shows the Salesforce Developer Console in Google Chrome. The URL is `playful-unicorn-ob57jg-dev-ed.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage`. The code editor displays the `ApplicationBatchTest.apc` file. The code implements a `Database.Batchable<sObject>` interface. It starts by initializing counters for total forms and converted forms. The `start` method queries records from the `Application_Form__c` object. The `execute` method iterates through the results, incrementing the total form counter and the converted form counter if the record has a specific field value. The `finish` method sends an email to an admin with the total counts. The developer console interface includes tabs for Logs, Tests, and Problems, and a system status bar at the bottom.

```
1 Public class ApplicationBatchTest implements Database.Batchable<sObject>{
2
3     //start(), execute(), finish()
4     public Integer totalForms = 0; // total no of application form
5     public Integer totalConvertedForms = 0; // total no of students
6     public Database.QueryLocator start(Database.BatchableContext bc){
7         // gathers the data for you
8         String applicationQuery = 'select id, Name, Ready_To_Join__c from Application_Form__c';
9         return Database.getQueryLocator(applicationQuery);
10    }
11    public void execute(Database.BatchableContext bc, List<Application_Form__c> formList){
12        // process the data
13        for(Application_Form__c af : formList){
14            totalForms++;
15            if(af.Ready_To_Join__c){
16                totalConvertedForms++;
17            }
18        }
19    }
20    public void finish(Database.BatchableContext bc){
21        // emails ,
22        Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();
23        // address, subject, content( data to sent to admins)
24        mail.setSubject(' Application form and student record data as of today ');
25        mail.setPlainTextBody(' Total no of application form records are : '+totalForms+ ' out of which no of students as per today : '+totalConvertedForms);
26        String[] emailAddess = new String[]{saiprasanth51@gmail.com};
27        mail.setToAddresses(emailAddess);
28        Messaging.sendEmail(new Messaging.SingleEmailMessage[]{ mail } );
29    }
}
```



Milestone: Create a Scheduler Class

Detiled Description:

From the Developer Console create a new Apex Class :

Day5:Activities

Topic: Lightening web components

Here we are going to create the college data table, for this we need to create an apex class, from which we are going to retrieve the data and Html, javascript files for UI .

Task1: Create college data table component(APEX CLASS)

Create an Apex Class With The Required Functionality

```
public class GetapplicationDetails
```

```
{
```

```
    @AuraEnabled(cacheable=true)
```

```
    public static List<ApplicationForm__c> oversimplification(id CollegelId)
```

```
    {
```

```
        List<ApplicationForm__c> formlist = [SELECT ID, College_Fees__c, Name,  
        Date_of_Birth__c, Email__c, Hostel_Fees__c, Student_Name__c FROM  
        ApplicationForm__c WHERE College__c =:CollegelId];
```

```
        return formList;
```

```
    }
```

```
}
```

```

11     }
12 }
13 @AuraEnabled(cacheable=true)
14 public static List<formWrapper> getApplicationDetails(id collegeId){
15     List<Application_Form__c> formList=[select id, college_fees__c, Email__c, Date_of_Birth__c, Address__c, Guardian_Name__c, Hostel_Fees__c, collegeName__c from Application_Form__c];
16     List<formWrapper> wrapperList=new List<formWrapper>();
17     for(Application_Form__c af:formList){
18         formWrapper wrap=new formWrapper();
19         wrap.Name=af.Name;
20         wrap.Student_Name=af.Student_Name__c;
21         wrap.Guardian_Name=af.Guardian_Name__c;
22         wrap.College_fees=af.college_fees__c;
23         wrap.Hostel_fees=af.Hostel_Fees__c;
24         wrap.Address=af.Address__c;
25         wrap.Email=af.Email__c;
26         wrap.collegeName=af.college__r.College_Names__c;
27         wrap.formNameUrl='/'+af.id;
28         wrap.DOB=af.Date_of_Birth__c;
29         wrapperList.add(wrap);
30     }
31     return wrapperList;
32 }
33
34 public class formWrapper{
35     @AuraEnabled public String Name {get;set;}
36     @AuraEnabled public String student_Name {get;set;}
37     @AuraEnabled public String Guardian_Name {get;set;}
38     @AuraEnabled public Decimal College_fees {get;set;}
39     @AuraEnabled public Decimal Hostel_fees {get;set;}
40     @AuraEnabled public String Address {get;set;}
41     @AuraEnabled public String Email {get;set;}
42     @AuraEnabled public String collegeName {get;set;}
43     @AuraEnabled public String formNameUrl {get;set;}
44     @AuraEnabled public Date DOB {get;set;}
45 }
46

```

Topic: Create college data table component(HTML CLASS)

HTML:

<template>

<h1> College and Application form list table </h1>

<template if:true={recordList}>
 <lightning-datable
 key-field="id"
 data={recordList}
 show-row-number-column
 hide-checkbox-column
 columns={columnsList}>
 >

```
</lightning-datatable>  
</template>
```

```
<template if:true={error}>  
{error}  
</template>  
</template>
```

```
1 <template>  
2     <h1>College and Application form list table</h1>  
3  
4     <template if:true={recordList}>  
5         <lightning-datatable  
6             key-field="id"  
7             data={recordList}  
8             show-row-number-column  
9             hide-checkbox-column  
10            columns={columnsList}  
11        </lightning-datatable>  
12    </template>  
13  
14    <template if:true={error}>  
15        {error}  
16    </template>  
17</template>
```

Topic: Create college data table component(SCRIPT CLASS)

After the creation of the Html file create the following js file

```
import { LightningElement, api, wire } from 'lwc';  
import getapplicationvalues  
from '@salesforce/apex/GetapplicationDetails.getapplicationvalues';  
export default class CollegeDataTable extends LightningElement {  
columnsList = [  
    {label : 'Application Form' , fieldName : 'Name', type:'text' },  
    {label : 'College Fees' , fieldName : 'College_Fees__c', type:'currency' },  
    {label : 'Date Of Birth' , fieldName : 'Date_Of_Birth__c', type:'date' },
```

```

{label : 'Email' , fieldName : 'Email__c', type:'email' },
{label : 'Hostel Fees' , fieldName : 'Hostel_Fees__c', type:'Currency' },
{label : 'StudentName' , fieldName : 'Student_Name__c', type:'text' }
];
@api recordId;
recordList;
error;
@wire(getapplicationvalues, {CollegeId : '$recordId'})
wiredCollegeData({data, error}){
if(data){
    this.recordList = data;
}
else if(error){
    this.error = error;
    this.recordList = undefined;
}
}
}

```

The screenshot shows the Salesforce LWC Editor interface. The left sidebar displays the project structure:

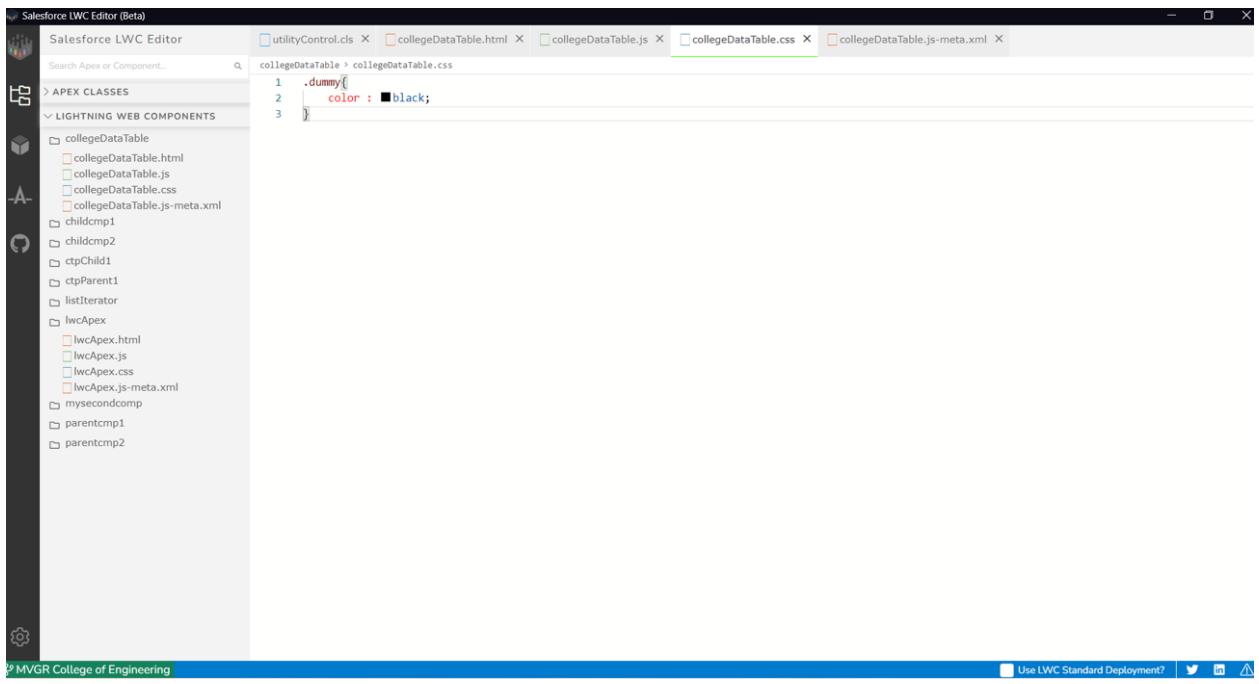
- Salesforce LWC Editor (Beta)**
- UtilityControl.cls**
- collegeDataTable.html**
- collegeDataTable.js**
- collegeDataTable.css**
- collegeDataTable.js-meta.xml**
- APEX CLASSES**
- LIGHTNING WEB COMPONENTS**
 - collegeDataTable**
 - collegeDataTable.html
 - collegeDataTable.js
 - collegeDataTable.css
 - collegeDataTable.js-meta.xml
 - childcmp1**
 - childcmp2**
 - ctpChild1**
 - ctpParent1**
 - listIterator**
 - lwcApex**
 - lwcApex.html
 - lwcApex.js
 - lwcApex.css
 - lwcApex.js-meta.xml
 - mysecondcomp**
 - parentcmp1**
 - parentcmp2**

The main code editor window contains the following JavaScript code:

```

import { LightningElement, api, wire } from 'lwc';
import getForm from '@salesforce/apex/utilityControl.getApplicationDetails';
export default class CollegeDataTable extends LightningElement {
    columnsList=[{
        label:'College Name', fieldName: 'collegeName', type: 'text' },
        {label:'Application Form', fieldName: 'formnameurl', type: 'url', typeAttributes:{label:{fieldName:'Name'}, target:'_blank'}},
        {label:'Application Name', fieldName: 'student_name', type: 'text' },
        {label:'Email', fieldName: 'Email', type: 'email' },
        {label:'Address', fieldName: 'Address', type: 'text' },
        {label:'DoB', fieldName: 'DoB', type: 'date' },
        {label:'Guardian Name', fieldName: 'Guardian_Name', type: 'text' },
        {label:'College Fee', fieldName: 'College_fees', type: 'currency' },
    ];
    recordId;
    recordList;
    error;
    @wire(getForm, {collegeId: '$recordId'})
    wiredAccounts(({data, error})=>{
        if(data){
            this.recordList=data;
        }
        else if(error){
            this.error=error;
            this.recordList=undefined;
        }
    })
}

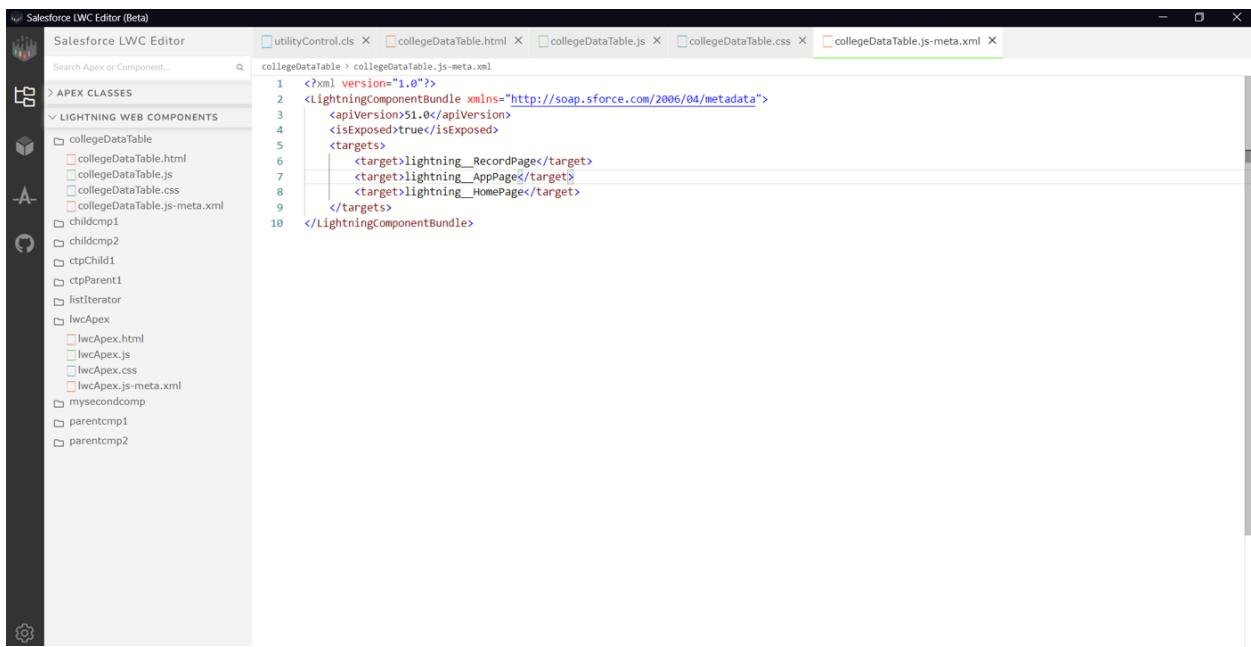
```



Topic: Create college data table component(META CLASS)

Change the Meta File as follows:

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



The screenshot shows the Salesforce LWC Editor (Beta) interface. On the left is a file tree with the following structure:

- Salesforce LWC Editor
- Search Apex or Component...
- APEX CLASSES
- LIGHTNING WEB COMPONENTS
 - collegeDataTable
 - collegeDataTable.html
 - collegeDataTable.js
 - collegeDataTable.css
 - collegeDataTable.js-meta.xml
 - childcmp1
 - childcmp2
 - ctpChild1
 - ctpParent1
 - listIterator
 - lwcApex
 - lwcApex.html
 - lwcApex.js
 - lwcApex.css
 - lwcApex.js-meta.xml
 - mysecondcomp
 - parentcmp1
 - parentcmp2

The main area is a code editor showing the following XML code:

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

CONCLUSION:

In this project we have learned the various stepd of a college management application development with the help of milestones like creating object creating fields for the objects, defining the dependencies between the fields, creating validation rules,Custom object creation, adding business logic to the application,creating apex class the lightening wed compoments.