

Day 1

Topic:Creating Salesforce development account and activating it.

Activity:

Task1:Creating a salesforce developer account

The following steps are involved in creating the salesforce developer account

Creating a developer org in salesforce.

1.Go to [developers.salesforce.com/](https://developer.salesforce.com/)

2.Click on sign up.

3.On the sign up form, enter the following details :

4.First name & Last name

5.Email

6.Role : Developer

7.Company : College Name

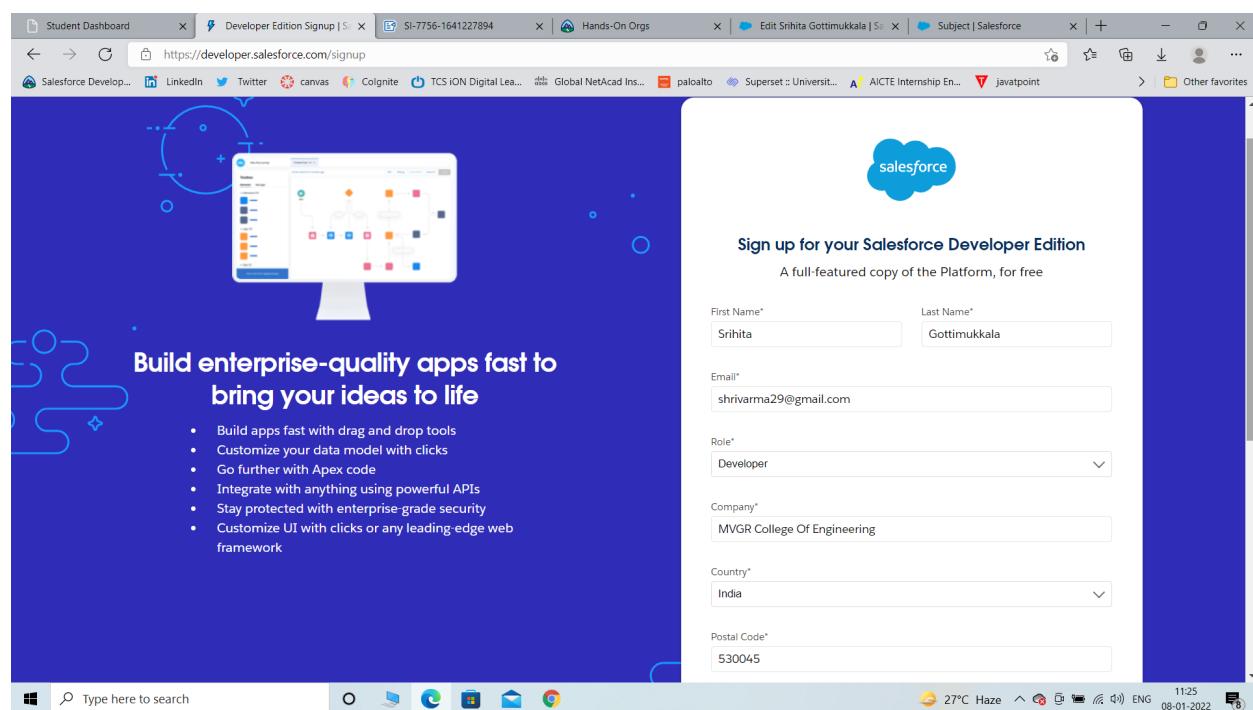
8.County : India

9.Postal Code : pin code

10.Username : should be a combination of your name and company

This need not be an actual email id, you can give anything in the format :

username@organization.com



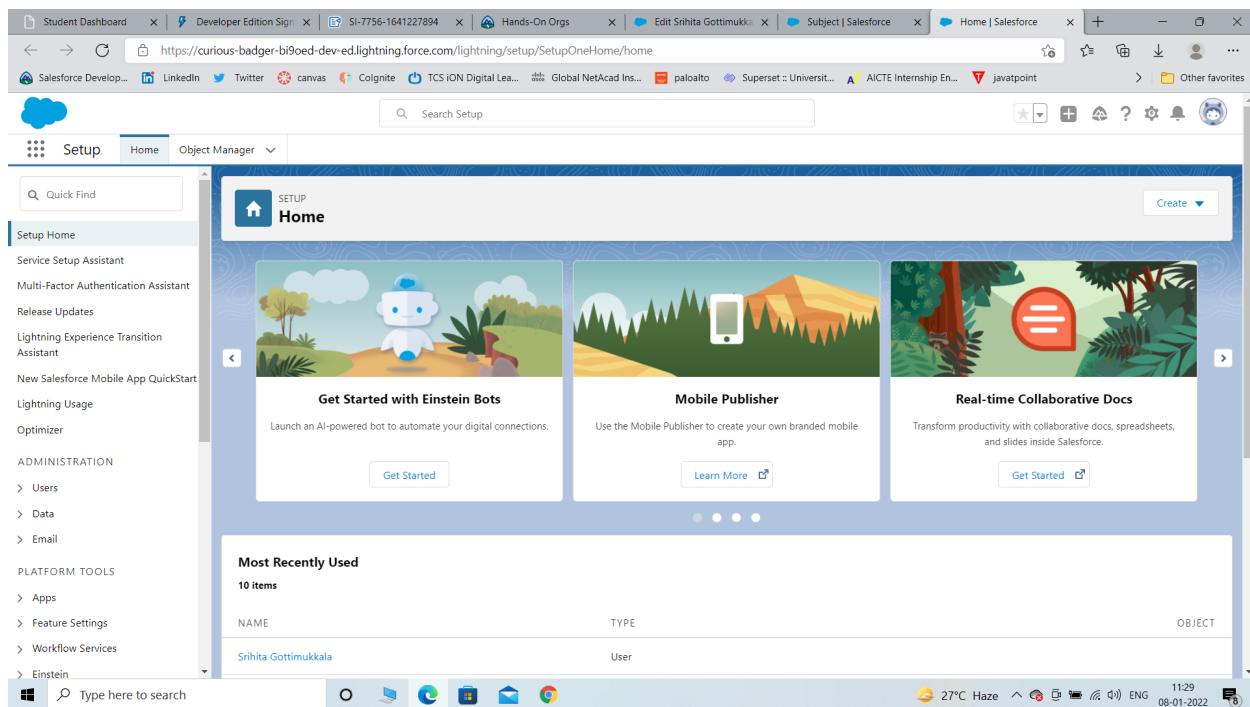
Task2:verifying the account through email verification

You will be receiving a mail from salesforce developer org with a click on button "verify your email" click on it.After clicking on it you will be redirected to the salesforce login page where you will be asked your login credentials to proceed further.

Task 3 Login into your salesforce developer account

The steps that are needed to follow are:

1. Go to salesforce.com and click on login.
2. Enter the username and password that you just created.
3. After login this is the home page which you will see.



Day 2:

Topic: Custom object creation

Activities:

Task 1: Create College management Application

The steps involved in this are

1. Click on the settings icon on the salesforce home page this will show two options
2. Click on setup
3. This will open the salesforce home page
4. Now, search for app manager and click on it
5. Click on new lightning app
6. Fill the app details and click next, In the same way fill the app options, navigation items, user profiles and then click on save this creates the new lightning app called College management application

Student Dashboard | SI-7756-1641227894 | Hands-On Orgs | All | Colleges | Salesforce | College management - Lightning

Lightning App Builder | App Settings | Pages | College management

App Settings

App Details & Branding

App Options

Utility Items (Desktop Only)

Navigation Items

User Profiles

App Details & Branding

Give your Lightning app a name and description. Upload an image and choose the highlight color for its navigation bar.

App Details

*App Name: College management

*Developer Name: College_management

Description: This is my first application

App Branding

Image: Primary Color Hex Value: #0070D2

Upload

Org Theme Options: Use the app's image and color instead of the org's custom theme

App Launcher Preview



Student Dashboard | SI-7756-1641227894 | Hands-On Orgs | All | Colleges | Salesforce | College management - Lightning

Lightning App Builder | App Settings | Pages | College management

App Settings

App Details & Branding

App Options

Utility Items (Desktop Only)

Navigation Items

User Profiles

App Options

Navigation and Form Factor

*Navigation Style: Standard navigation

Supported Form Factors:

- Desktop and phone (selected)
- Desktop
- Phone

Setup and Personalization

Setup Experience:

- Setup (full set of Setup options) (selected)
- Service Setup

App Personalization Settings

Disable end user personalization of nav items in this app

Disable temporary tabs for items outside of this app

28°C Haze 15:06 08-01-2022

Student Dashboard | SI-7756-1641227894 | Hands-On Orgs | All | Colleges | Salesforce | College management - Lightning

Lightning App Builder | App Settings | Pages | College management

App Settings

App Details & Branding

App Options

Utility Items (Desktop Only)

Navigation Items

User Profiles

Utility Items (Desktop Only)

Utility Items (Desktop Only)

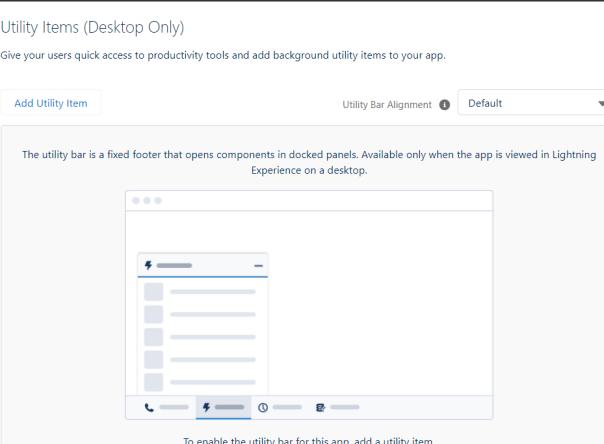
Give your users quick access to productivity tools and add background utility items to your app.

Add Utility Item

Utility Bar Alignment: Default

The utility bar is a fixed footer that opens components in docked panels. Available only when the app is viewed in Lightning Experience on a desktop.

To enable the utility bar for this app, add a utility item.



Student Dashboard | SI-7756-1641227894 | Hands-On Orgs | All | Colleges | Salesforce | College management - Lightning

Lightning App Builder | App Settings | Pages | College management

App Settings

App Details & Branding

App Options

Utility Items (Desktop Only)

Navigation Items

User Profiles

Navigation Items

Navigation Items

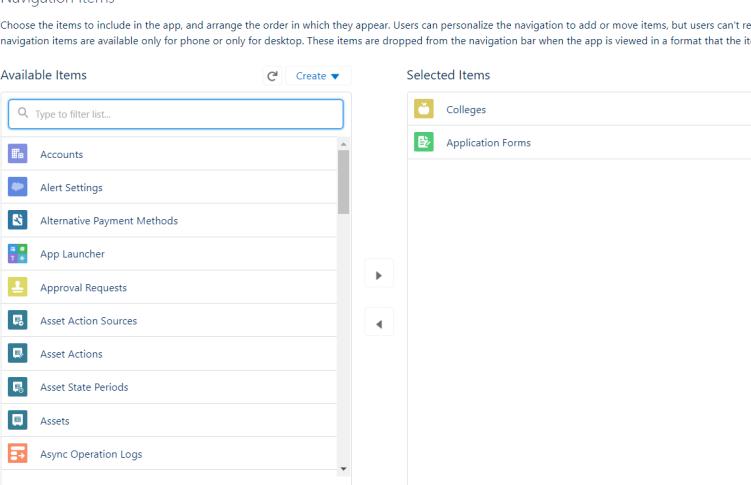
Choose the items to include in the app, and arrange the order in which they appear. Users can personalize the navigation to add or move items, but users can't remove or rename the items that you add. Some navigation items are available only for phone or only for desktop. These items are dropped from the navigation bar when the app is viewed in a format that the item doesn't support.

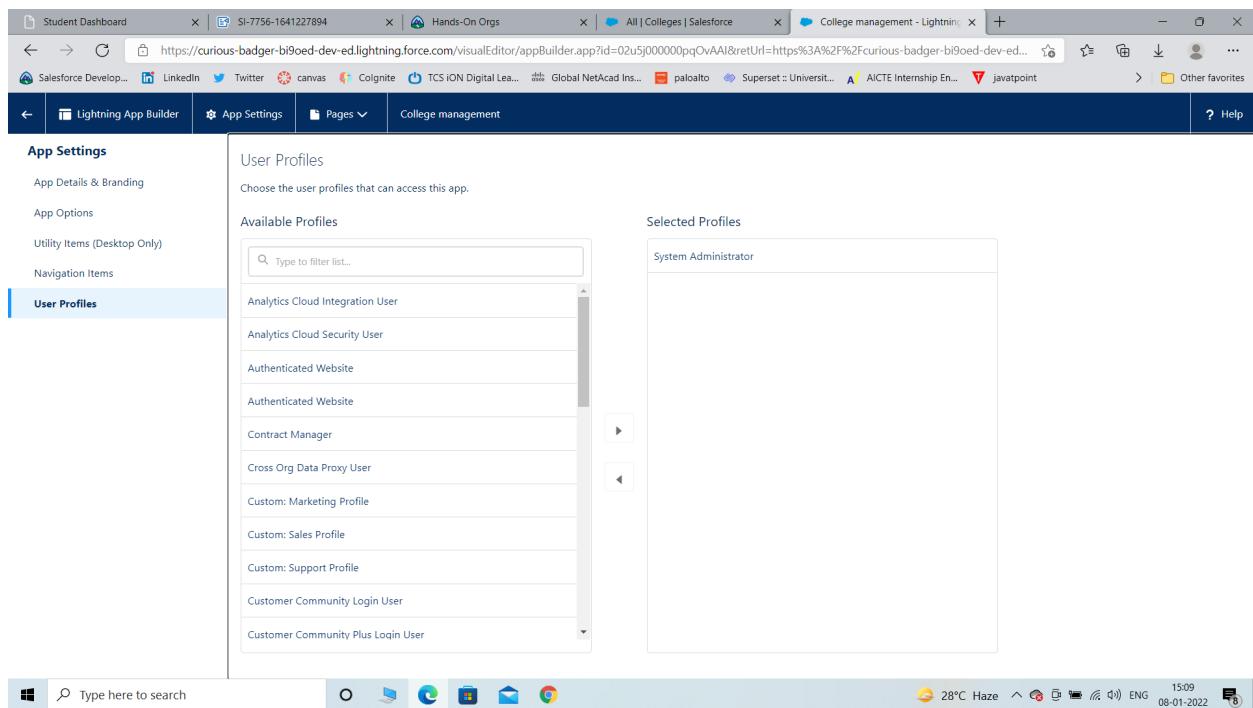
Available Items

Selected Items

Colleges

Application Forms





Task2: Create the custom objects for the college management application

The steps involved in creating custom objects are

1. Click on setup
2. Click on object manager
3. Click on create new custom object
4. Select the label and fill in the necessary details and click on save

then select a icon for the object and click on save and next to create new objects.

The objects that are needed to be created for college management application are

1. College

The screenshot shows the Salesforce Setup interface with the 'Object Manager' selected. The main page title is 'College'. On the left, a sidebar lists various configuration options under 'Details': Fields & Relationships, 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, and Restriction Rules. The right panel displays the 'Details' section for the 'College' object, which includes fields like API Name (College__c), Singular Label (College), Plural Label (Colleges), and various settings such as Enable Reports, Track Activities, and Deployment Status (Deployed). The top navigation bar shows multiple tabs open, including 'Student Dashboard', 'Hands-On Orgs', 'All | Colleges | Salesforce', 'College | Salesforce', and 'College management - Light'. The status bar at the bottom indicates it's 15:15 on 08-01-2022.

2.Application Form

The screenshot shows the Salesforce Setup interface with the 'Object Manager' selected. The main page title is 'Application Form'. On the left, a sidebar lists various configuration options under 'Details': Fields & Relationships, 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, and Restriction Rules. The right panel displays the 'Details' section for the 'Application Form' object, which includes fields like API Name (Application_Form__c), Singular Label (Application Form), Plural Label (Application Forms), and various settings such as Enable Reports, Track Activities, and Deployment Status (Deployed). The top navigation bar shows multiple tabs open, including 'Student Dashboard', 'Hands-On Orgs', 'All | Colleges | Salesforce', 'Application Form | Salesforce', and 'College management - Light'. The status bar at the bottom indicates it's 15:16 on 08-01-2022.

3.Student

The screenshot shows the Salesforce Object Manager interface for the 'Student' object. The left sidebar lists various setup options: Fields & Relationships, 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, and Restriction Rules. The main 'Details' tab displays the following configuration:

Setting	Value
Description	
API Name	Student__c
Custom	✓
Singular Label	Student
Plural Label	Students
Enable Reports	
Track Activities	
Track Field History	
Deployment Status	Deployed
Help Settings	Standard salesforce.com Help Window

4. Subjects

The screenshot shows the Salesforce Object Manager interface for the 'Subject' object. The left sidebar lists the same setup options as the Student object. The main 'Details' tab displays the following configuration:

Setting	Value
Description	
API Name	Subject__c
Custom	✓
Singular Label	Subject
Plural Label	Subjects
Enable Reports	
Track Activities	
Track Field History	
Deployment Status	Deployed
Help Settings	Standard salesforce.com Help Window

Task 3: Creating the fields on each of the above objects

The process to create fields is as follows

- 1.click on object manager
- 2.Search the required object in the search bar
- 3.click on the object
- 4.click on fields and relationships

5.Click on new

6.select the data type for the new field and click on next then enter the field label and save
The objects with their necessary fields that are required to be created for the college management application are:

1.College

The fields that are required in the college object are

Create the following fields on the college object.

Field Name	Data Type	Required	Values
Record Info	Text		
College Fees	Currency(7,2)	Yes	
Hostel Fees	Currency(6,2)	Yes	
College Name	Picklist(Refer Business Logic In Milestones)	Yes	
Email	Picklist		blr@mit.co.in hyd@mit.co.in mum@mit.co.in maa@mit.co.in ccu@mit.co.in del@mit.co.in
Capacity Of Students	Picklist		500-1000, 1000-2500, 2500-6000, 6000-10000

The screenshot shows the Salesforce Object Manager interface for the 'College' object. On the left, a sidebar lists various setup options like Page Layouts, Lightning Record Pages, and Field Sets. The main area is titled 'Fields & Relationships' and displays a table of fields. The columns include FIELD LABEL, FIELD NAME, DATA TYPE, CONTROLLING FIELD, and INDEXED. Fields listed include College Email, college_fee, College Name, Created By, Hostel fee, Last Modified By, No of Seats, Owner, and Record Info.

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
College Email	College_Email__c	Picklist	College Name	
college_fee	college_fee__c	Currency(7, 2)		
College Name	College_Name__c	Picklist		
Created By	CreatedById	Lookup(User)		
Hostel fee	Hostel_fee__c	Currency(6, 2)		
Last Modified By	LastModifiedById	Lookup(User)		
No of Seats	No_of_Seats__c	Picklist	College Name	
Owner	OwnerId	Lookup(User,Group)		✓
Record Info	Name	Text(80)		✓

2. Application Form

The fields that are needed to be created on the application form object are:

Create the following fields on the application form object.

Field Name	Data Type	Required	Values
Application Form ID	Autonumber		F-{000000} Starting Number=1
Address	Text(255)	Yes	
College	Master-Detail(College)	Yes	
College Fees	Formula(Currency)		
Hostel Fees	Formula(Currency)		
date of Birth	Date	Yes	
Email	Email(Unique)	Yes	
Guardian Name	Text(30)	Yes	
Looking For Hostel Stay	Checkbox(default=Uncheck)		
Ready To Join	Checkbox(default=Uncheck)		
Student Name	Text(30)	Yes	
Phone		Yes	

The screenshot shows the Salesforce Object Manager interface. The top navigation bar includes tabs for Student Dashboard, SI-7756-1641227894, Hands-On Orgs, All | Colleges | Salesforce, and Application Form | Salesforce. Below the navigation is a toolbar with various icons like Setup, Home, and Object Manager. The main content area is titled 'Application Form' and shows the 'Fields & Relationships' section. A table lists 16 fields: Address (Text(255)), Application Form ID (Auto Number), College (Master-Detail(College)), college fee (Formula(Currency)), Created By (Lookup(User)), Date of Birth (Date), Email (Email (Unique)), Form Number (Auto Number), Guardian Name (Text(30)), Hostel Fees (Formula(Currency)), Last Modified By (Lookup(User)), Looking For Hostel Stay (Checkbox), phone (Phone), Ready To Join (Checkbox), and Student Name (Text(30)). The table has columns for FIELD LABEL, FIELD NAME, DATA TYPE, CONTROLLING FIELD, and INDEXED.

3.Student

The fields that are needed to be created on the student object are:

Create the following fields on the student object.

Field Name	Data Type	Required	Values
Student Name	Text		
Address	Text(255)	Yes	
Application Form	Lookup(Application Form)		
College Name	Formula(Text)		
Date Of Birth	Date	Yes	
Guardian Name	Text(30)	Yes	
Phone	Phone	Yes	

The screenshot shows the Salesforce Object Manager interface for the 'Student' object. The left sidebar lists various setup options like Page Layouts, Lightning Record Pages, Buttons, Links, and Actions, etc. The main area displays the 'Fields & Relationships' section with 10 items. The columns include FIELD LABEL, FIELD NAME, DATA TYPE, CONTROLLING FIELD, and INDEXED. Key fields shown 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	CreatedBy	Lookup(User)		
Date Of Birth	DateOfBirth__c	Date		
Guardian Name	Guardian_Name__c	Text(30)		
Last Modified By	LastModifiedBy	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		
Phone	Phone__c	Phone		

4. Subjects

The fields that are required to be created on the subjects object are:

Create the following fields On the Subject object.

Field Name	Data Type	Required	Values
Subject ID	AutoNumber		S-{000000} Starting Number=1
Paper 1	Picklist(Refer Business Logic Milestone)		
Paper2	Picklist(Refer Business logic Milestone)		
Student	Lookup(Student)		

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedById	Lookup(User)		
Last Modified By	LastModifiedById	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		✓
Paper 1	Paper_1__c	Picklist		
Paper2	Paper2__c	Picklist		
Student	Student_c	Lookup(Student)		✓
Subject ID	Subject_ID__c	Auto Number		
Subject Name	Name	Text(80)		✓

After creation of the objects and their necessary fields the college management application look as follows:

Subject Name	
You haven't viewed any Subjects recently. Try switching list views.	

Day 3:

Topic: Adding Business Logic To Application

Task1: Creating global picklist value sets

The process for creating global picklist value sets is as follows:

- 1.Click on home in setup
 - 2.search for picklist value sets in the search bar and click on it
 - 3.click on new
 - 4.enter the label and values to be displayed in the picklist
 - 5.click on save
 - 6.while creating the field for picklist select global value sets option and select the required value set
1. Create the following global picklist value sets for the application.
- a)College

Picklist Value Name	Values
College	MIT-HYD MIT-BLR MIT-MUM MIT-MAA MIT-DEL MIT-CCU

Global Value Set Detail

Action	Values	API Name	Default	Chart Colors	Modified By
Edit Del Deactivate	MIT-BLR	MIT-BLR	<input type="checkbox"/>	Assigned dynamically	Srihita Gottumukkala, 1/6/2022, 4:10 AM
Edit Del Deactivate	MIT-MAA	MIT-MAA	<input type="checkbox"/>	Assigned dynamically	Srihita Gottumukkala, 1/6/2022, 4:10 AM
Edit Del Deactivate	MIT-HYD	MIT-HYD	<input type="checkbox"/>	Assigned dynamically	Srihita Gottumukkala, 1/6/2022, 4:10 AM
Edit Del Deactivate	MIT-DEL	MIT-DEL	<input type="checkbox"/>	Assigned dynamically	Srihita Gottumukkala, 1/6/2022, 4:10 AM
Edit Del Deactivate	MIT-MUM	MIT-MUM	<input type="checkbox"/>	Assigned dynamically	Srihita Gottumukkala, 1/6/2022, 4:10 AM
Edit Del Deactivate	MIT-CCU	MIT-CCU	<input type="checkbox"/>	Assigned dynamically	Srihita Gottumukkala, 1/6/2022, 4:10 AM

Global Value Set Detail

Action	Values	API Name	Default	Chart Colors	Modified By
Edit Del Deactivate	MIT-BLR	MIT-BLR	<input type="checkbox"/>	Assigned dynamically	Ramareju Pusapati, 1/6/2022, 9:22 PM
Edit Del Deactivate	MIT-MAA	MIT-MAA	<input type="checkbox"/>	Assigned dynamically	Ramareju Pusapati, 1/6/2022, 9:22 PM
Edit Del Deactivate	MIT-HYD	MIT-HYD	<input type="checkbox"/>	Assigned dynamically	Ramareju Pusapati, 1/6/2022, 9:22 PM
Edit Del Deactivate	MIT-DEL	MIT-DEL	<input type="checkbox"/>	Assigned dynamically	Ramareju Pusapati, 1/6/2022, 9:22 PM
Edit Del Deactivate	MIT-MUM	MIT-MUM	<input type="checkbox"/>	Assigned dynamically	Ramareju Pusapati, 1/6/2022, 9:22 PM
Edit Del Deactivate	MIT-CCU	MIT-CCU	<input type="checkbox"/>	Assigned dynamically	Ramareju Pusapati, 1/6/2022, 9:22 PM

b)Paper1

Picklist Value Name	Values
Paper 1	APEX JAVA C C++

Global Value Set Detail

Information

- Label: Paper
- Name: Paper_1
- Description:

Picklist Values Used

Active and inactive picklist values 4 (1,000 max)

Action	Values	API Name	Default	Chart Colors	Modified By
Edit Del Deactivate	APEX	APEX	<input checked="" type="checkbox"/>	Assigned dynamically	Sruthi Gottimukkala, 1/13/2022, 6:38 AM
Edit Del Deactivate	JAVA	JAVA	<input type="checkbox"/>	Assigned dynamically	Sruthi Gottimukkala, 1/13/2022, 6:38 AM
Edit Del Deactivate	C	C	<input type="checkbox"/>	Assigned dynamically	Sruthi Gottimukkala, 1/13/2022, 6:38 AM
Edit Del Deactivate	C++	C++	<input type="checkbox"/>	Assigned dynamically	Sruthi Gottimukkala, 1/13/2022, 6:38 AM

c)Paper2

Picklist Value Name	Values
Paper2	MATHEMATICS ENGLISH STATISTICS

Global Value Set Detail

Information

- Label: Paper 2
- Name: Paper_2
- Description:

Picklist Values Used

Active and inactive picklist values 3 (1,000 max)

Action	Values	API Name	Default	Chart Colors	Modified By
Edit Del Deactivate	MATHEMATICS	MATHEMATICS	<input checked="" type="checkbox"/>	Assigned dynamically	Sruthi Gottimukkala, 1/13/2022, 6:39 AM
Edit Del Deactivate	ENGLISH	ENGLISH	<input type="checkbox"/>	Assigned dynamically	Sruthi Gottimukkala, 1/13/2022, 6:39 AM
Edit Del Deactivate	STATISTICS	STATISTICS	<input type="checkbox"/>	Assigned dynamically	Sruthi Gottimukkala, 1/13/2022, 6:39 AM

Task2:Creating Field Dependencies

The process to create field dependencies is as follows

- 1.Go to object manager
- 2.click on college object
- 3.click on fields and relationships
- 4.click on field dependencies
- 5.click on new

6.select the controlling field and the dependent field

7.select the values to be included and save it

The field dependencies that we need to create are:

1)Create field dependency between college Name and Email, where the controlling field is college Name and dependent field is Email. Select the email ids according to the college names.

The screenshot shows the Salesforce Setup interface for creating a field dependency. The top navigation bar includes tabs for Hands-On Orgs, All | Colleges | Salesforce, College | Salesforce, Student Dashboard, and a specific setup page. The main area is titled "SETUP > OBJECT MANAGER" and shows the "College" object. On the left, a sidebar lists various configuration options under "Fields & Relationships". The main content area displays a table for defining field dependencies. The "Controlling Field" is set to "College Name" and the "Dependent Field" is set to "College Email". Below this, instructions advise using double-click or Shift+click to select cells and include/exclude values. A legend indicates that yellow cells represent "Included Value" and white cells represent "Excluded Value". The table lists college names (MIT-BLR, MIT-MAA, MIT-HYD, MIT-DEL, MIT-MUM) and their corresponding email addresses, with some emails highlighted in yellow to indicate they are included. The bottom of the table shows pagination and links to "View All" and "Go to".

- 2) Create field dependency between college Name and capacity of students, where the controlling field is college Name and dependent field is Capacity of Students. Select the values according to your wish.

College Name:	MIT-BLR	MIT-MAA	MIT-HYD	MIT-DEL	MIT-MUM
No of Seats:	500-1500 1500-3000 3000-5000	500-1500 1500-3000 3000-5000	500-1500 1500-3000 3000-5000	1500-3000	500-1500 1500-3000 3000-5000
	5000-8000	5000-8000	5000-8000	5000-8000	5000-8000 8000-12000
	8000-12000	8000-12000	8000-12000	8000-12000	8000-12000

Task3:Creating validation rules on objects

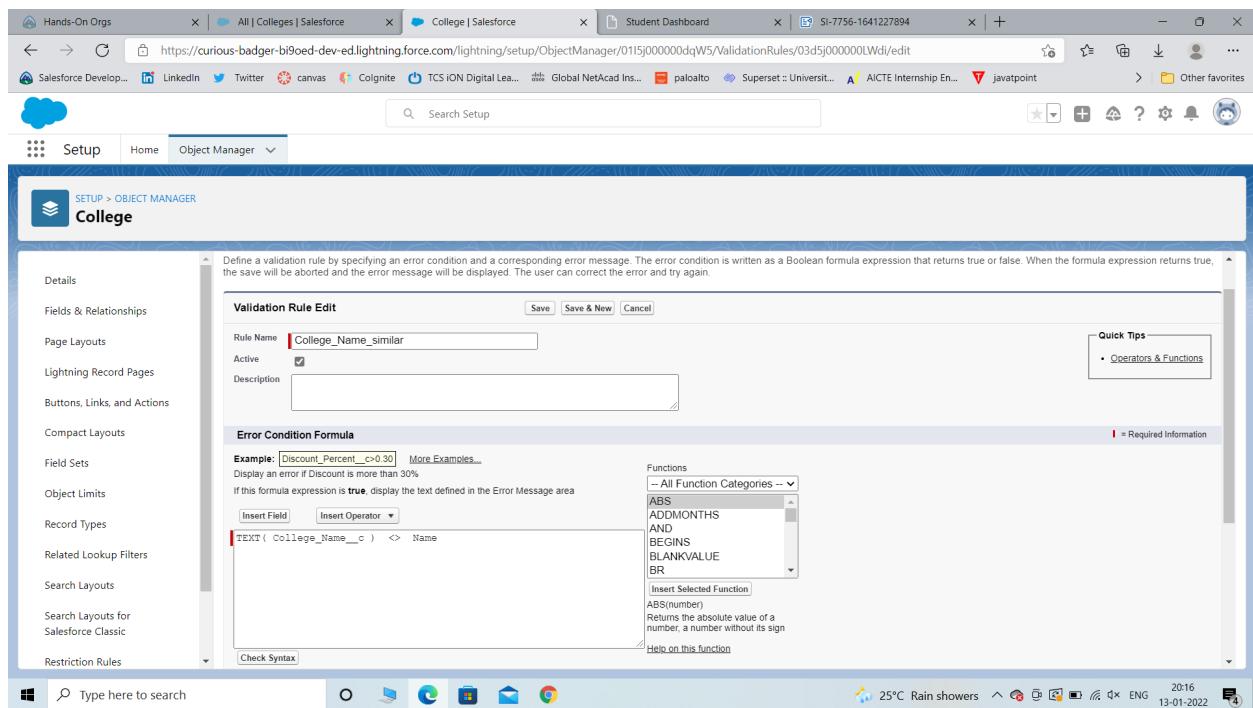
The steps required to follow for creation of validation rules are:

- 1.Click on object manager
- 2.go to the required object
- 3.click on validation rules
- 4.click on new
- 5.add the rule and save it

The objects on which the validation rules have to be created are:

- 1) Create a validation rule on the college object such that the college name and record info should have the same name.

TEXT(College_Name__c) <> 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.

```

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 )
)
)

```

The screenshot shows the Salesforce Setup interface with the Object Manager selected. A validation rule for the 'Application Form' object is being edited. The formula is as follows:

```

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 )
    )
)

```

Task4: Process Automation

Steps required for process Automation are:

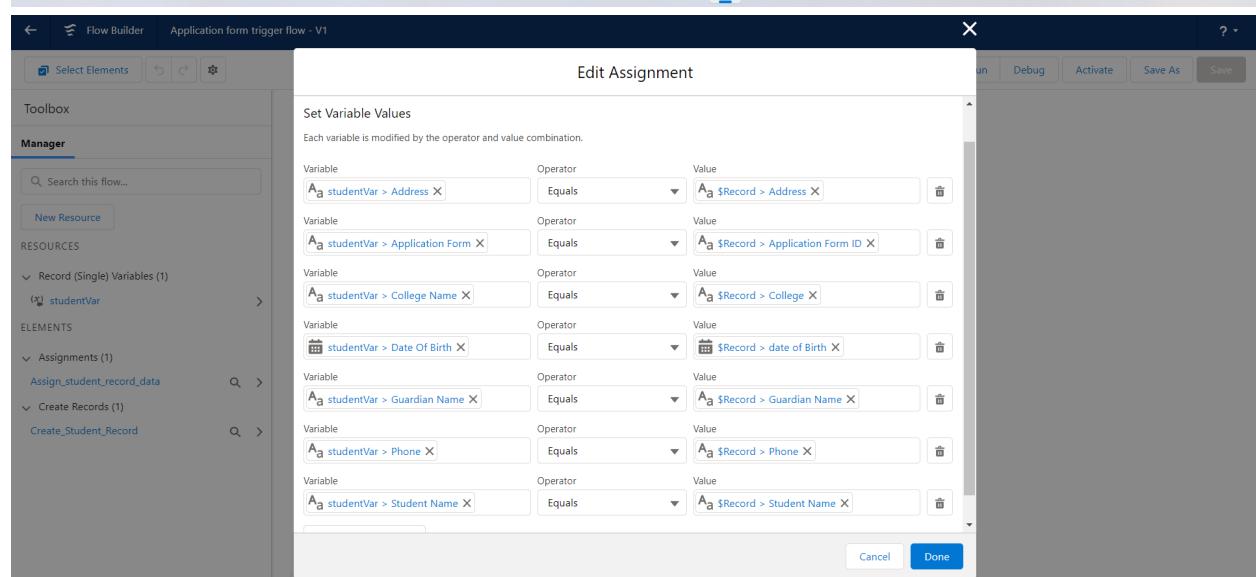
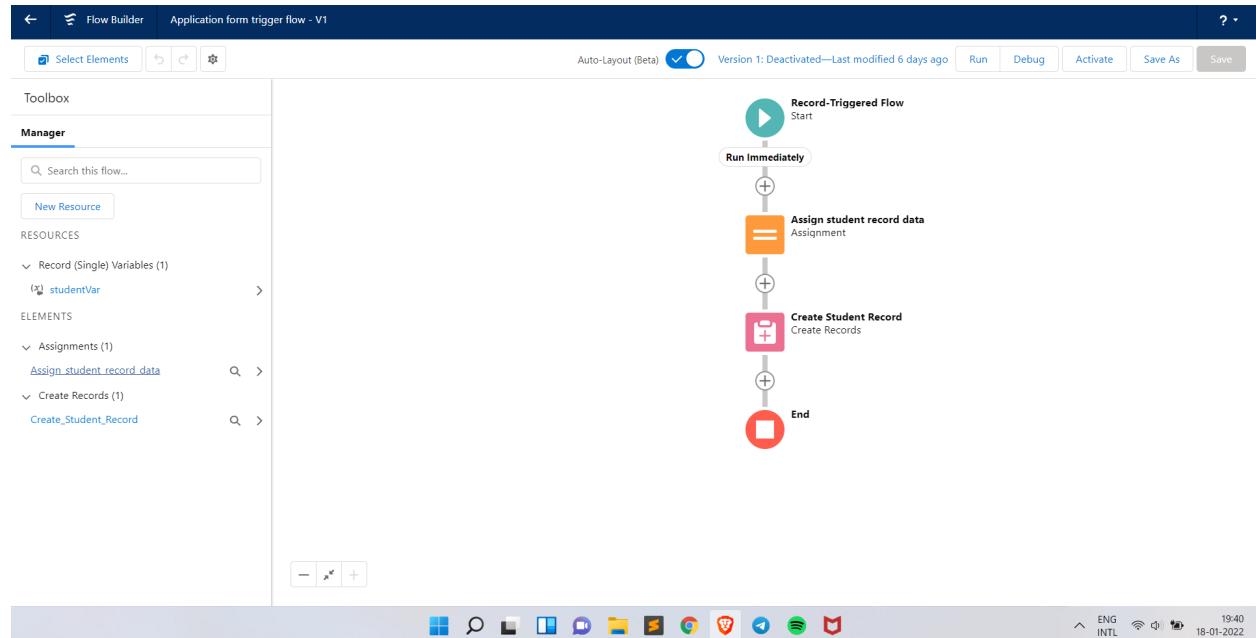
- 1.click on the home
- 2.search for process builder
- 3.click on new
- 4.enter the necessary fields
- 5.click on activate then save

The screenshot shows the Salesforce Process Builder interface. A process named 'Process master' is displayed. The flow starts with a 'START' step, followed by an 'Application Form' trigger. This triggers a decision diamond 'Ready field is checked'. If the condition is 'TRUE', it performs an 'IMMEDIATE ACTIONS' step 'create a student record' and then stops. If the condition is 'FALSE', it proceeds to another decision diamond '+ Add Criteria'. If this second decision is 'TRUE', it performs an 'IMMEDIATE ACTIONS' step '+ Add Action' and then stops. If the condition is 'FALSE', it stops. On the right, there is a configuration panel for creating a 'Student' record, setting values for various fields.

Task5:Create the student record using flow

The steps required to build the flow are as follows:

1. First deactivate the process builder which we created earlier.
2. Now search for flows 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 as follows.



Day 6

Topic:Batch Apex

Batch Apex is used to run large jobs (think thousands or millions of records!) that would exceed

normal processing limits. Using Batch Apex, you can process records asynchronously in batches (hence the name, “Batch Apex”) to stay within platform limits. If you have a lot of records to process, for example, data cleansing or archiving, Batch Apex is probably your best solution.

Syntax Of BatchApex

```
public class MyBatchClass implements Database.Batchable<sObject> {
    public (Database.QueryLocator | Iterable<sObject>) start(Database.BatchableContext bc) {
        // collect the batches of records or objects to be passed to execute
    }
    public void execute(Database.BatchableContext bc, List<P> records){
        // process each batch of records
    }
    public void finish(Database.BatchableContext bc){
        // execute any post-processing operations
    }
}
```

Task1:Create a batch apex for Application form

For this you need to create a apex class and do the following

1. From the developer console create a new apex class and enter the following code.

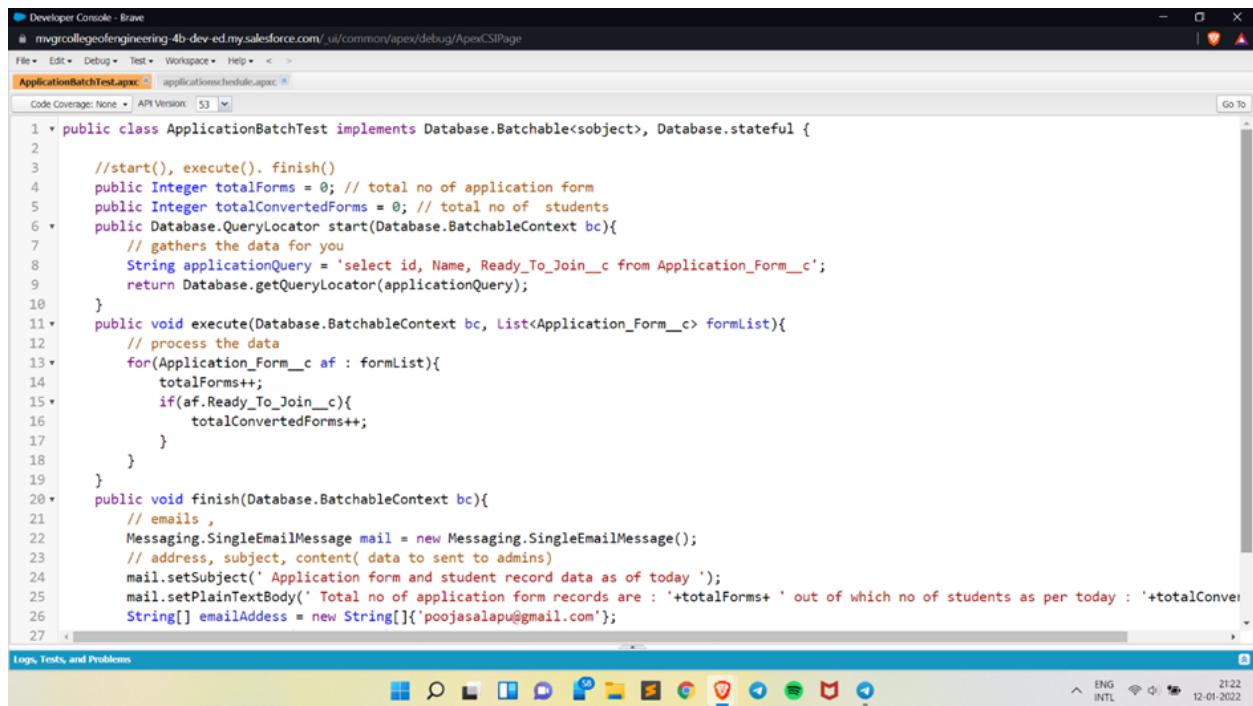
```
Public class ApplicationBatchTest implements Database.Batchable<sObject>{

    //start(), execute(). finish()
    public Integer totalForms = 0; // total no of application form
    public Integer totalConvertedForms = 0; // total no of students
    public Database.QueryLocator start(Database.BatchableContext bc){ // gathers the data for
you
        String applicationQuery = 'select id, Name, Ready_To_Join__c from ApplicationForm__c';
        return Database.getQueryLocator(applicationQuery);
    }
    public void execute(Database.BatchableContext bc, List<ApplicationForm__c> formList){
        // process the data
        for(ApplicationForm__c af : formList){
            totalForms++;
            if(af.Ready_To_Join__c){
                totalConvertedForms++;
            }
        }
    }
}
```

```

public void finish(Database.BatchableContext bc){
    // emails ,
    Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();
    // address, subject, content( data to sent to admins)
    mail.setSubject(' Application form and student record data as of today ');
    mail.setPlainTextBody(' Total no of application form records are : '+totalForms+ ' out of
which no of students as per today : '+totalConvertedForms);
    String[] emailAddess = new String[]{'your email address'};
    mail.setToAddresses(emailAddess);
    Messaging.sendEmail(new Messaging.SingleEmailMessage[]{ mail });
}
}

```



The screenshot shows the Salesforce Developer Console interface. The title bar reads "Developer Console - Brave" and the URL is "mvgcollegeofengineering-4b-dev-ed.my.salesforce.com/ui/common/apex/debug/ApexCSIPage". The tab bar has "ApplicationBatchTest.apxc" selected. The code editor displays the following Apex class:

```

1 * public class ApplicationBatchTest implements Database.Batchable<sobject>, Database.stateful {
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[]{'poojasalapu@gmail.com'};
27    }
}

```

The code editor includes line numbers, syntax highlighting, and a status bar at the bottom showing "Logs, Tests, and Problems", "ENG INTL", "21:22", and "12-01-2022".

Developer Console - Brave
 mvrgcollegeofengineering-4b-dev-ed.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage

File • Edit • Debug • Test • Workspace • Help • < >

ApplicationBatchTest.apxc applicationschedule.apxc

Code Coverage: None API Version: 53 Go To

```

5 lic Integer totalConvertedForms = 0; // total no of students
6 vlic 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•lic 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•lic 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 : '+totalConvertedForm
26 String[] emailaddress = new String[]{'poojasalapu@gmail.com'};
27 mail.setToAddresses(emailaddress);
28 Messaging.sendEmail(new Messaging.SingleEmailMessage[]{ mail } );
29
30

```

Logs, Tests, and Problems

Developer Console - Brave
 mvrgcollegeofengineering-4b-dev-ed.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage

File • Edit • Debug • Test • Workspace • Help • < >

ApplicationBatchTest.apxc applicationschedule.apxc

Code Coverage: None API Version: 53 Go To

```

1 • public class ApplicationBatchTest implements Database.Batchable<sobject>, Database.stateful {
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 : '+totalConve
26     String[] emailAddress = new String[]{'poojasalapu@gmail.com'};
27

```

Logs, Tests, and Problems

```
Developer Console - Brave
mvrgcollegeofengineering-4b-dev-ed.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage
File Edit Debug Test Workspace Help < >
ApplicationBatchTest.apc applicationschedule.apc
Code Coverage: None API Version: 53 Go To
5 lic Integer totalConvertedForms = 0; // total no of students
6 *lic 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 *lic 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 *lic 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 : '+totalConvertedForm
26 String[] emailaddress = new String[]{'poojasalapu@gmail.com'};
27 mail.setToAddresses(emailaddress);
28 Messaging.sendEmail(new Messaging.SingleEmailMessage[]{ mail });
29
30
Logs, Tests, and Problems
21:22 ENG INTL 12-01-2022
```

Task2:Create a scheduler class

1. From the developer console create a new apex class and enter the following code.

```
public class applicationschedule implements Schedulable{
    public void execute(SchedulableContext sc){
        ApplicationBatchTest abt = new ApplicationBatchTest();
        Database.executeBatch(abt, 400); // 200 to 200
    }
}
```

From setup search for apex class and click on schedule jobs and fill the details as per your requirements.

Developer Console - Brave
mvrgcollegeofengineering-4b-dev-ed.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage

File Edit Debug Test Workspace Help < >
ApplicationBatchTest.apxc applicationschedule.apxc
Code Coverage: None API Version: 53 Go To

```
1 public class applicationschedule implements Schedulable{
2
3
4
5 public void execute(SchedulableContext sc){
6
7     ApplicationBatchTest abt = new ApplicationBatchTest();
8
9     Database.executeBatch(abt, 400); // 200 to 2000
10
11
12 }
13
14 }
15 }
```

Logs, Tests, and Problems

Windows Taskbar: File Explorer, Control Panel, Device Manager, Task View, File History, Task Scheduler, Taskbar Icons, Taskbar Buttons, Taskbar Shortcuts, Taskbar Status Bar, Taskbar Buttons, Taskbar Shortcuts, Taskbar Status Bar

System Tray: ENG INTL, 21:23, 12-01-2022

Execute Anonymous - Brave
mvrgcollegeofengineering-4b-dev-ed.my.salesforce.com/_ui/common/apex/debug/ApexExecAnon

```
1 ApplicationBatchTest abt=new ApplicationBatchTest();
2 Database.executeBatch(abt, 400);
3
4
5 String crn='20 10 23 ? 1 1';
6 System.schedule('cronJob1', crn, new applicationschedule());
```

Windows Taskbar: File Explorer, Control Panel, Device Manager, Task View, File History, Taskbar Icons, Taskbar Buttons, Taskbar Shortcuts, Taskbar Status Bar, Taskbar Buttons, Taskbar Shortcuts, Taskbar Status Bar

System Tray: ENG INTL, 17:50, 12-01-2022

WhatsApp Student Dashboard Triggers | SmartBridge SI-7885-1641227630 Apex Jobs | Salesforce mvgrcollegeengineering-4b-dev-ed.lightning.force.com/lightning/setup/AsyncApexJobs/home

Action	Submitted Date	Job Type	Status	Status Detail	Total Batches	Batches Processed	Failures	Submitted By	Completion Date	Apex Class	Apex Method	Apex Job ID
1/12/2022, 2:30 AM	Batch Apex	Completed			1	1	0	Salapu, Pooja Madhurima	1/12/2022, 2:30 AM	ApplicationBatchTest		7075g00001jTdSE
1/12/2022, 2:30 AM	Scheduled Apex	Queued			0	0	0	Salapu, Pooja Madhurima		applicationschedule		7075g00001jTdSG
1/12/2022, 2:15 AM	Scheduled Apex	Aborted			0	0	0	Salapu, Pooja Madhurima		applicationschedule		7075g00001jTcZB
1/12/2022, 2:12 AM	Batch Apex	Completed			1	1	0	Salapu, Pooja Madhurima	1/12/2022, 2:12 AM	ApplicationBatchTest		7075g00001jTdF6
1/12/2022, 2:09 AM	Batch Apex	Completed		First error: SendEmail failed. First exception on row 0; first error: INVALID_ID: Invalid ID. ID is invalid or you do not have access to the record.: [toAddresses, your]	1	1	0	Salapu, Pooja Madhurima	1/12/2022, 2:09 AM	ApplicationBatchTest		7075g00001jTcbu

Help for this Page

WhatsApp Student Dashboard Triggers | SmartBridge SI-7885-1641227630 Scheduled Jobs | Salesforce mvgrcollegeengineering-4b-dev-ed.lightning.force.com/lightning/setup/ScheduledJobs/home

Action	Job Name	Submitted By	Submitted	Started	Next Scheduled Run	Type
Del	cronJob1	Salapu, Pooja Madhurima	1/12/2022, 2:30 AM		1/15/2022, 11:10 PM	Scheduled Apex
Del	Analytics Data Loader Job for Org 00D5g000004zBVk	User_Integration	6/12/2021, 8:13 PM	1/11/2022, 8:37 PM	1/12/2022, 8:37 PM	Autonomous Data Loader Job

Help for this Page

A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | Other All

A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | Other All

https://mvgrcollegeengineering-4b-dev-ed.lightning.force.com/one/one.app#/setup/ScheduledJobs/home

Developer Console - Brave
mvrgcollegeofengineering-4b-dev-ed.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage

File • Edit • Debug • Test • Workspace • Help • < >
applicationFormTrigger.apxc formhelper.apxc formhelper_TestLapxc

Code Coverage: None • API Version: 53

```
1 public class formhelper {
2     public static void getoldNewValues(Map<id, Application_Form__c> formoldMap, List<Application_Form__c> formlist)
3     {
4         for(Application_Form__c fm:formlist)
5         {
6             Application_Form__c oldFormRecord = formoldMap.get(fm.id);
7             System.debug('Old value : '+oldFormRecord.Student_Name__c);
8             System.debug('New Values : '+fm.Student_Name__c + 'Old Values : '+formoldMap.get(fm.id).Student_Name__c);
9             if(fm.Student_Name__c!=formoldMap.get(fm.id).Student_Name__c)
10            {
11            }
12        }
13    }
14    public static void preventRecordDeletion(List<Application_Form__c> formlist)
15    {
16        Set<id> formIdSet=new Set<id>();
17        Map<id,Students__c> formidStudentRecMap=new Map<id,Students__c>();
18        for(Application_Form__c fm:formlist)
19        {
20            formIdSet.add(fm.id);
21        }
22        if(formIdSet!=null && formIdSet.size()>0)
23        {
24            List<Students__c> stdList=[select id,Application_Form__c from Students__c where Application_Form__c IN :formIdSet];
25            if(stdList.size()>0)
26            {
27
```

Logs, Tests, and Problems

Developer Console - Brave
mvrgcollegeofengineering-4b-dev-ed.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage

File • Edit • Debug • Test • Workspace • Help • < >
applicationFormTrigger.apxc formhelper.apxc formhelper_TestLapxc

Code Coverage: None • API Version: 53

```
27        {
28            for(Students__c std:stdList){
29                formidStudentRecMap.put(std.Application_Form__c,std);
30            }
31        }
32        if(formidStudentRecMap!=null && formidStudentRecMap.values().size()>0)
33        {
34            for(Application_Form__c fm:formlist){
35                if(formidStudentRecMap.containsKey(fm.id) && formidStudentRecMap.get(fm.id)!=null){
36                    fm.addError(System.label.Application_form_deletion_error_messages);
37                }
38            }
39        }
40    }
41    public static void preventMultipleStudentRecordCreation(List<Application_Form__c> formlist) {
42        Set<id> formIdSet=new Set<id>();
43        Map<id,Students__c> formidStudentRecMap=new Map<id,Students__c>();
44        List<Subjects__c> stdlist2 = new List<Subjects__c>();
45        for(Application_Form__c fm:formlist)
46        {
47            formIdSet.add(fm.id);
48        }
49        if(formIdSet!=null && formIdSet.size()>0)
50        {
51            List<Students__c> stdList=[select id, Application_Form__c from Students__c where Application_Form__c IN :formIdSet];
52            if(stdList.size()>0)
53
```

```

Developer Console - Brave
mgcollegeofengineering-4b-dev-ed.my.salesforce.com/ui/common/apex/debug/ApexCSIPage
File Edit Debug Test Workspace Help < >
applicationFormTrigger.apxc formhelper.apxc formhelper_Test.apxc
Code Coverage: None API Version: 53
2 • public class formhelper_Test {
3 •     testMethod static void getoldNewValuesTest(){
4 •         College__c clg=new College__c();
5 •         clg.College_names__c = 'MIT - BLR';
6 •         clg.Email__c='mit@blr.com';
7 •         clg.College_Fees__c=52877;
8 •         clg.Name='MIT - BLR';
9 •         insert clg;
10 •        Application_Form__c af=new Application_Form__c();
11 •        af.College__c=clg.id;
12 •        af.date_of_Birth__c=System.today()-720;
13 •        af.Guardian_Name__c='Guardian';
14 •        af.Email__c='herry@herry.com';
15 •        af.Student_Name__c='Herry herry';
16 •        af.Phone__c='1234567891';
17 •        af.Address__c='Chennai';
18 •        insert af;
19
20 //List<Students__c> stdList=[select id, Application_Form__c from Students__c where Application_Form__c =: af.id];
21 //System.debug('List of Students : '+stdList);
22 af.Student_Name__c='Herry porter';
23 af.Ready_To_Join__c=true;
24 update af;
25
26
27 List<Students__c> stdList=[select id, Application_Form__c from Students__c where Application_Form__c =: af.id];
28 System.debug('List of Students : '+stdList);

```

Day5:

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> getapplicationvalues\(id CollegeId\)](#)

 {

[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 =:CollegeId\];](#)

[return formList;](#)

 }

}

```

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

```

Task2:Create college data table component(HTML CLASS)

Create the component with the following files.

HTML:

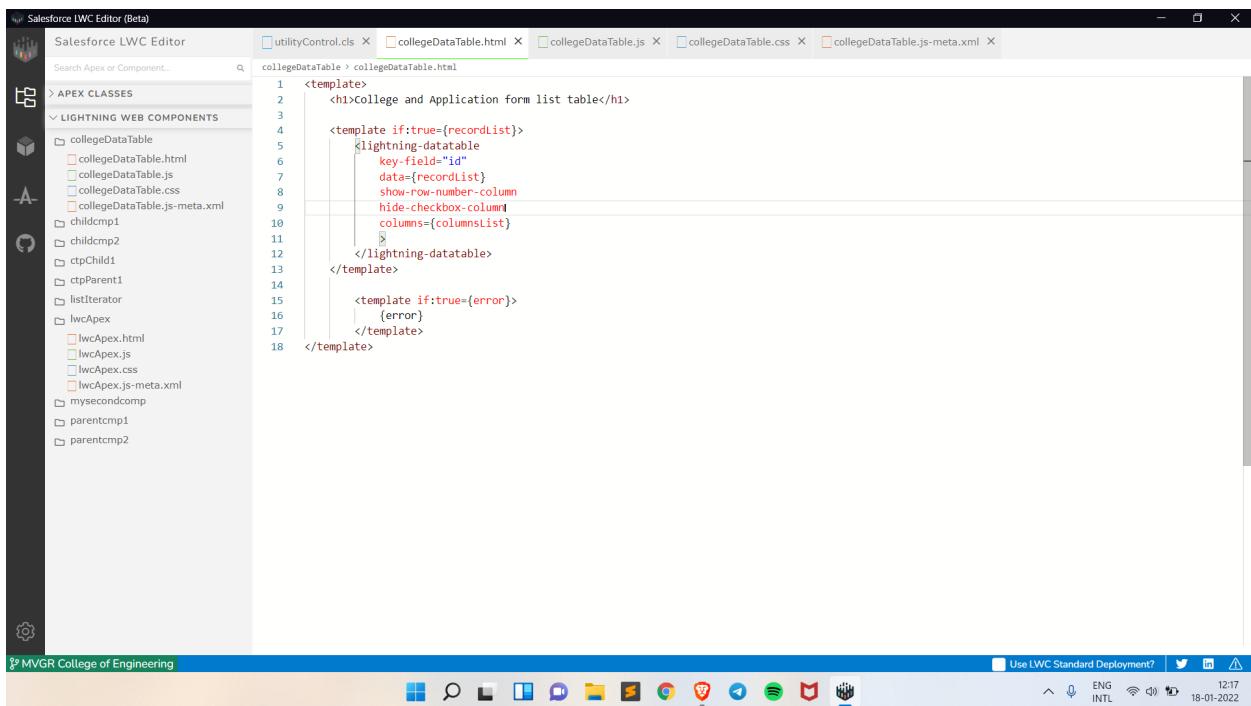
<template>

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

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

</lightning-datatable>
</template>

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



Task3:

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, {Collegelid : '$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 with the title "Salesforce LWC Editor (Beta)". The left sidebar lists various components and files under "APEX CLASSES" and "LIGHTNING WEB COMPONENTS". The main editor area displays the code for "collegeDataTable.js". The code defines a class "Collegedatable" extending "LightningElement". It includes a list of columns with labels and field names, and a @wire block to get form details. The code uses ES6 syntax and includes type annotations for fields.

```
import { LightningElement, api, wire } from 'lwc';
import getForm from '@salesforce/apex/utilityControl.getApplicationDetails';
export default class Collegedatable 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;
        }
    }
}
```

The screenshot shows the Salesforce LWC Editor (Beta) interface. The left sidebar lists components and files under 'APEX CLASSES' and 'LIGHTNING WEB COMPONENTS'. The main editor area displays the 'collegeDataTable.css' file with the following code:

```
1 .dummy{  
2   color : black;  
3 }
```

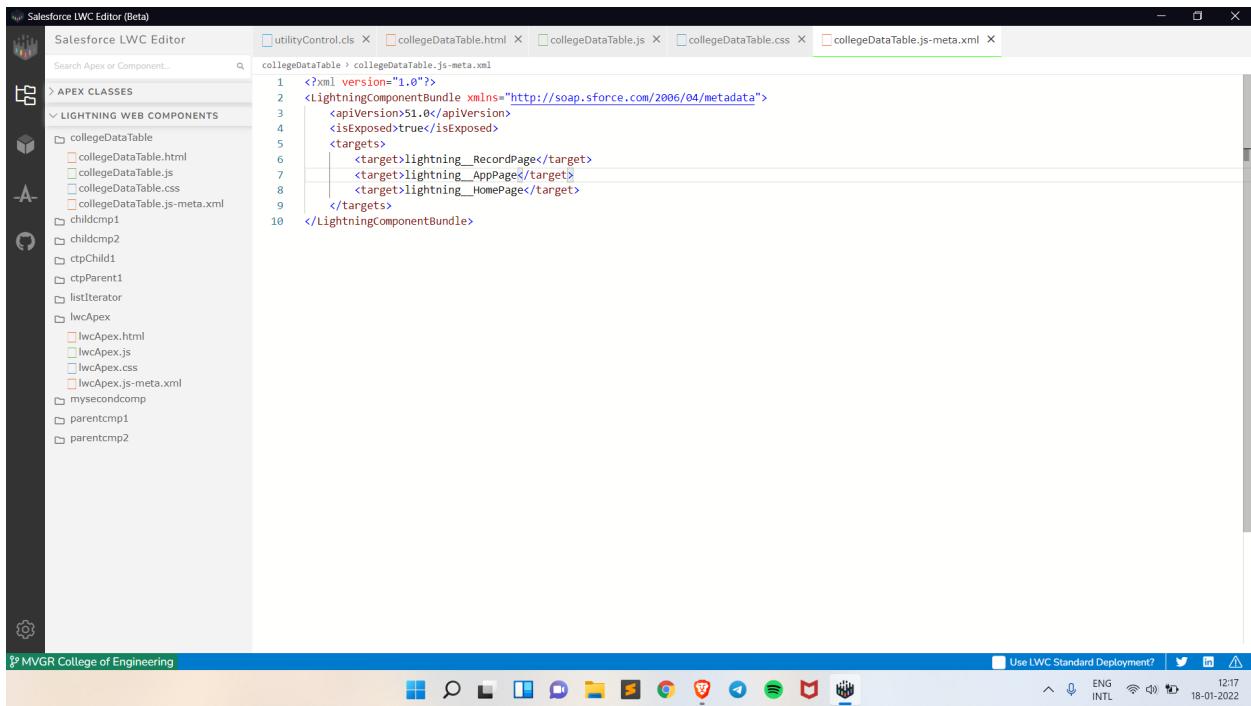
The top navigation bar includes icons for search, refresh, and various browser tabs. The system tray at the bottom right shows battery level, signal strength, and the date/time (18-01-2022). The bottom status bar also indicates 'Use LWC Standard Deployment?'.

Task4:

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>
```



CONCLUSION:

In this project we got to learn the various steps of a college management application development with the help of various milestones like creating objects creating fields on 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 web components.

