



**Smart  
Internz**



## SALESFORCE PROJECT READY

**Entry Level Training & Professional Development  
Program**

**Project Title : College Management  
Application**

### **Day 1**

Creating our Salesforce Developer Org To Get Started

1.Creating Developer AccountCreating a developer org in salesforce.

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

Click on sign up.

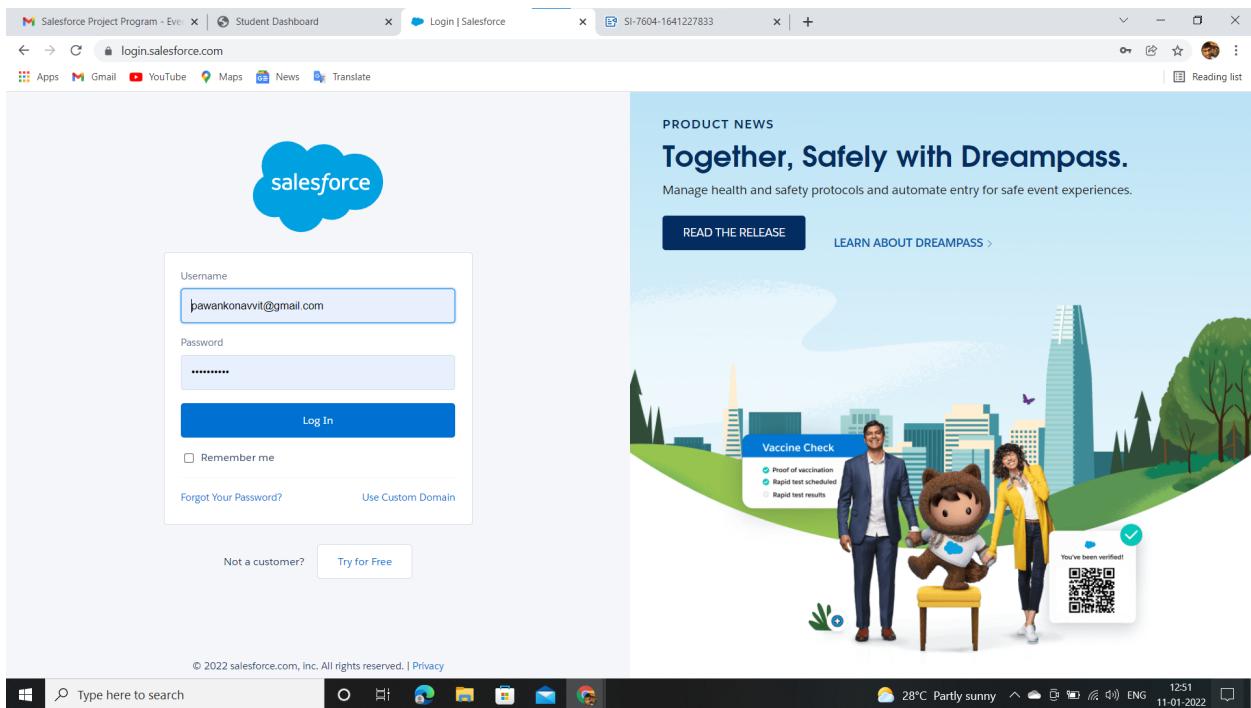
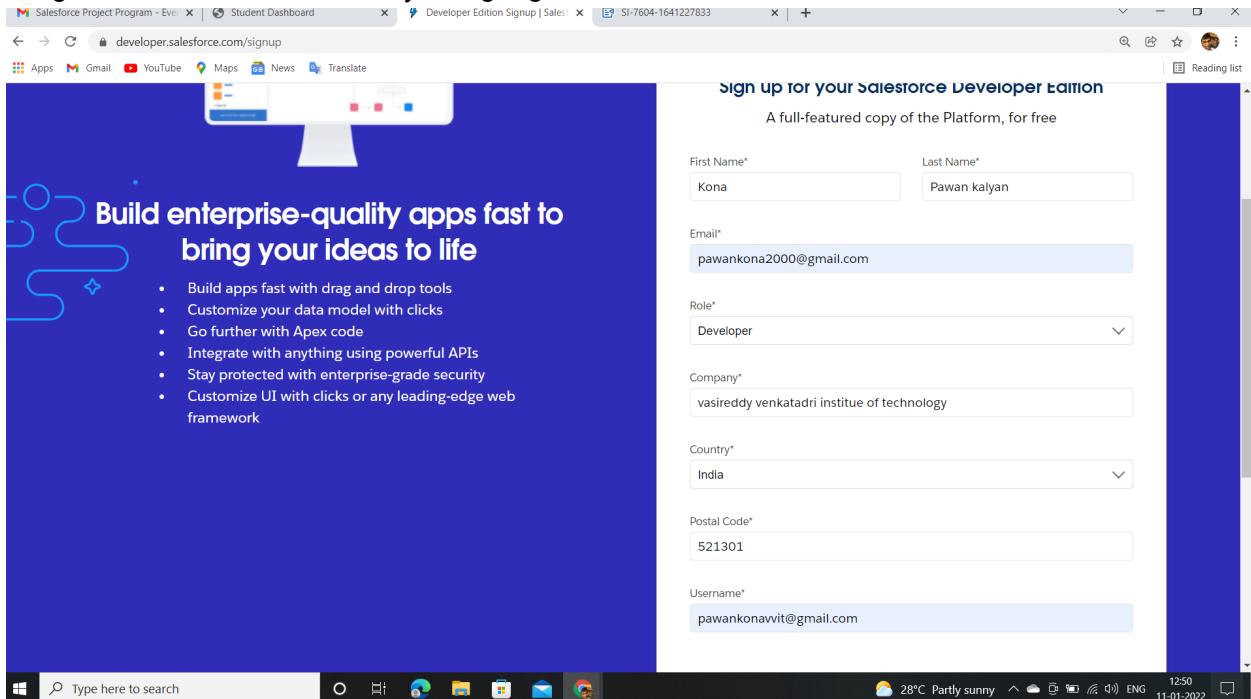
On the sign up form, entering our details

2.Account Activation

Go to the inbox of the email and Click on the verify account to activate my account

### 3. Login To Your Salesforce Account

login to salesforce account by using login details

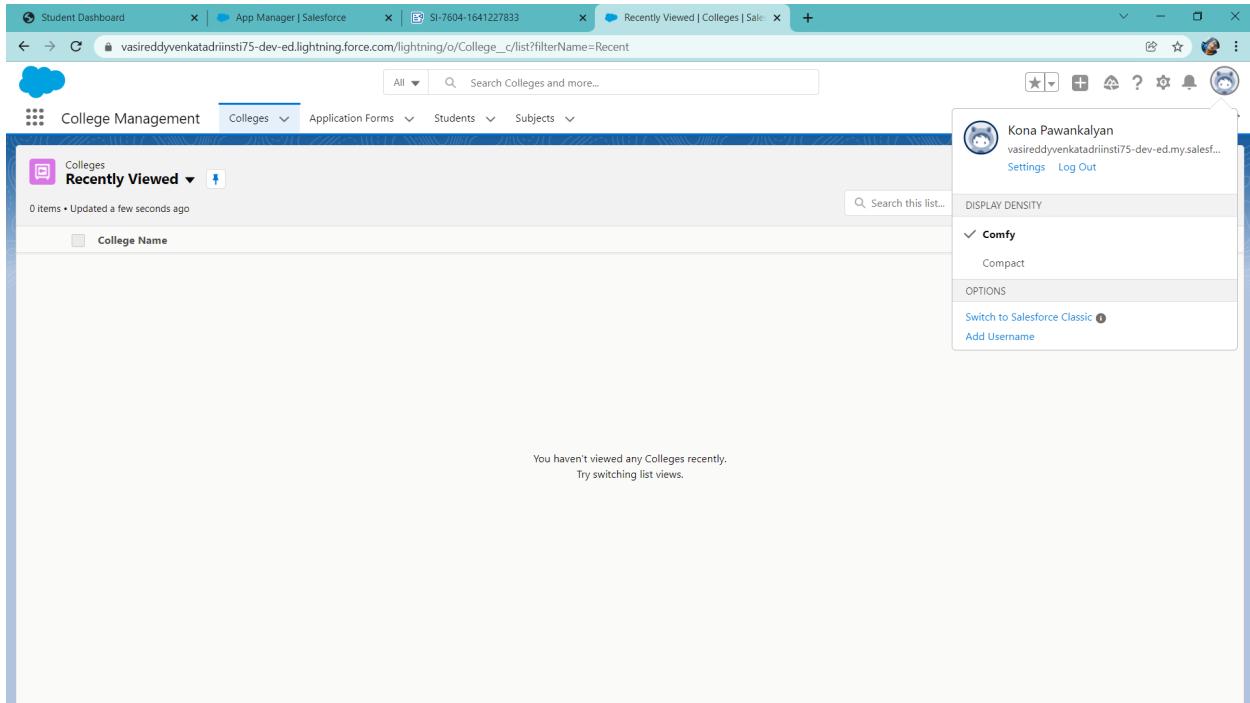


The screenshot shows the Salesforce Setup Home page. At the top right, there is a user profile for "Kona Pawankalyan" with options to "Settings" and "Log Out". A dropdown menu titled "DISPLAY DENSITY" is open, showing "Comfy" (selected), "Compact", and "OPTIONS". Below the dropdown, there are three cards: "Get Started with Einstein Bots", "Mobile Publisher", and "Real-time Collaborative Docs". On the left, a sidebar lists various setup categories like "Setup Home", "Service Setup Assistant", and "ADMINISTRATION". Under "ADMINISTRATION", there are links for "Users", "Data", and "Email". The "PLATFROM TOOLS" section includes "Apps", "Feature Settings", "Einstein", "Objects and Fields", "Events", and "Process Automation". A "Most Recently Used" section shows items like "College" (Custom Object Definition), "Application Form" (Custom Object Definition), and "app\_rule\_two" (Validation Rule). The URL in the browser is [vasireddyvenkatadriinsti75-dev-ed.lightning.force.com/lightning/setup/SetupOneHome/home](https://vasireddyvenkatadriinsti75-dev-ed.lightning.force.com/lightning/setup/SetupOneHome/home).

The above screenshot shows my new Developer Org Account.

## DAY 2:

Firstly, we have to build an college management app with the help of New Lightning App.



The above picture shows college management tab.

Creating an Custom Object :

1.Create The Custom Objects For The Application

First we create the College Management Lightening App

And then we create custom objects

College.

Application Form.

Students.

Subjects.

And then we create fields in custom objects according to the requirement

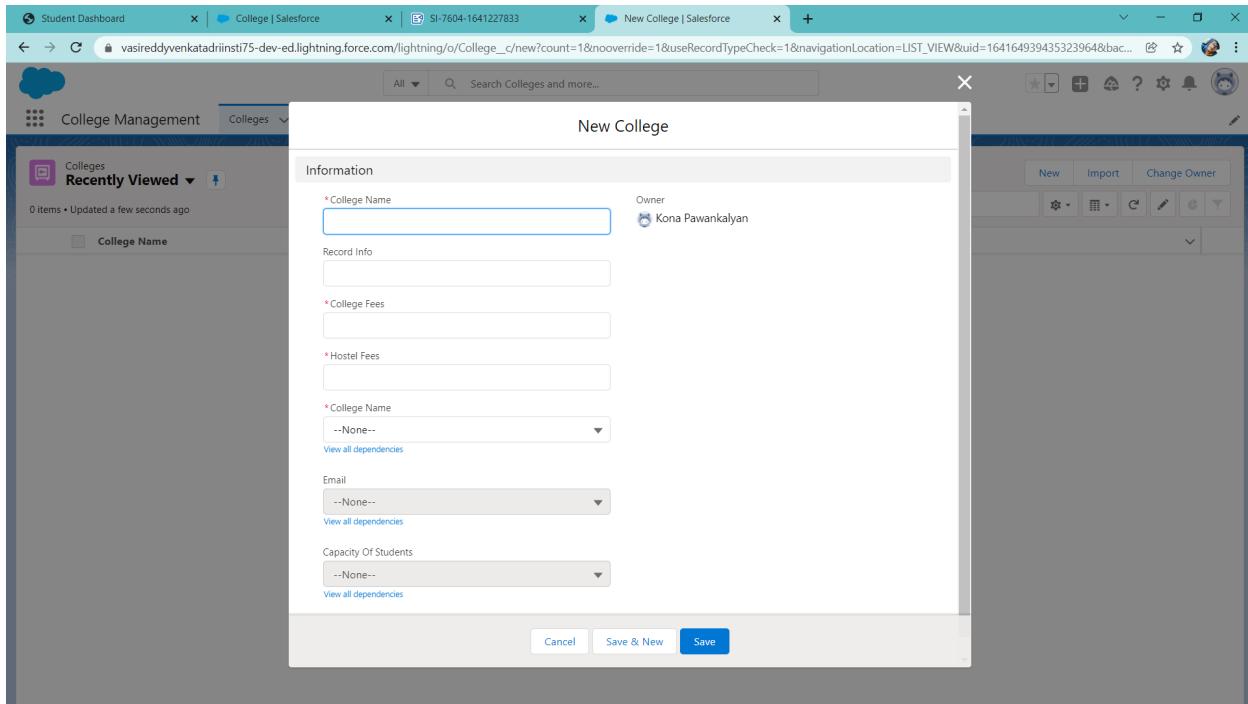
## Creating Fields On College Object

**Creating 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<br>hyd@mit.co.in<br>mum@mit.co.in<br>maa@mit.co.in<br>ccu@mit.co.in<br>del@mit.co.in |
| Capacity Of Students | Picklist                                     |          | 500-1000, 1000-2500, 2500-6000, 6000-10000                                                         |

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

- Setup Home:** The user is in the Object Manager under the College object.
- Fields & Relationships:** This section lists 10 items, sorted by Field Label.
- Table Headers:** FIELD LABEL, FIELD NAME, DATA TYPE, CONTROLLING FIELD, INDEXED.
- Table Data:**
  - Capacity Of Students: Capacity\_of\_Students\_c, Picklist, Controlling Field: 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, Controlling Field: 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(200)



I had created an custom object college. Now by the following requirements, I have created Fields. After that I have created an custom object on Application form and

### ***Fields On Application Form Object***

| Field Name          | Data Type              | Required | Values                          |
|---------------------|------------------------|----------|---------------------------------|
| Application Form ID | Auto number            |          | F-{000000}<br>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  | Checkbox(default=Unch) |          |                                 |

|               |                           |     |  |
|---------------|---------------------------|-----|--|
| Stay          | eck)                      |     |  |
| Ready To Join | Checkbox(default=Uncheck) |     |  |
| Student Name  | Text(30)                  | Yes |  |
| Phone         | Phone                     | Yes |  |

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

- Tab:** Object Manager > Application Form
- Section:** Fields & Relationships
- Table Headers:** FIELD LABEL, FIELD NAME, DATA TYPE, CONTROLLING FLD
- Table Data:**

| FIELD LABEL           | FIELD NAME            | DATA TYPE              | CONTROLLING FLD |
|-----------------------|-----------------------|------------------------|-----------------|
| 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            | CreatedBy             | Lookup(User)           |                 |
| date of Birth         | date_of_Birth__c      | Date                   |                 |
| Email                 | Email_c               | Email                  |                 |
| Guardian Name         | Guardian_Name__c      | Text(30)               |                 |
| Hostel Fees           | Hostel_Fees__c        | Formula (Currency)     |                 |
- Right Panel:**
  - User Profile: Kona Pawankalyan, vasireddyvenkatadriinst75-dev-ed.lightning.force.com
  - DISPLAY DENSITY: Comfy (selected), Compact
  - OPTIONS: Switch to Salesforce Classic, Add Username

The screenshot shows the Salesforce Lightning interface with a modal window titled "New Application Form: student". The modal contains a form with the following fields:

- Application Form Name:** A text input field.
- Address:** A text input field.
- College:** A search bar labeled "Search Colleges..." with a magnifying glass icon.
- Date of Birth:** A date input field with a calendar icon.
- Email:** A text input field.
- Guardian Name:** A text input field.
- Looking For Hostel Stay:** A checkbox.
- Ready To Join:** A checkbox.

At the bottom of the modal are three buttons: "Cancel", "Save & New", and "Save".

This screenshot shows, fields creation in application form.

### Fields On 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 Setup interface for the 'Student' custom object. The 'Fields & Relationships' tab is selected. The page displays a table of fields:

| FIELD LABEL      | FIELD NAME          | DATA TYPE                | CONTROLLING FIELD |
|------------------|---------------------|--------------------------|-------------------|
| 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                    |                   |
| Student Name     | Name                | Text(80)                 | ✓                 |

The above screenshots shows adding of fields on student custom object.

### Fields On Subject Object

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

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

- Tab Bar:** Student Dashboard, SI-7604-1641227833, Subjects | Salesforce, Inbox (2) - pawankona2000@gmail.com
- Header:** Search Setup, Setup, Home, Object Manager
- Left Sidebar:** Details, Fields & Relationships (selected), 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.
- Table:** Fields & Relationships (8 items, Sorted by Field Label)

| FIELD LABEL      | FIELD NAME       | DATA TYPE          | CONTROLLING FIELD |
|------------------|------------------|--------------------|-------------------|
| 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        |                   |
| Subjects Name    | Name             | Text(80)           | ✓                 |
- Right Sidebar:** Kona Pawankalyan, vasireddyvenkatadriinsti75-dev-ed.lightning.force.com/lightning/setup/ObjectManager/0115j000000BDgB/FieldsAndRelationships/view, Settings, Log Out, DISPLAY DENSITY (Comfy selected), OPTIONS (Compact selected), Switch to Salesforce Classic, Add Username.

The screenshot shows the College Management application with the following details:

- Tab Bar:** Student Dashboard, College | Salesforce, New Subjects | Salesforce
- Header:** Search Subjects and more..., College Management, Colleges, Application Forms, Students, Subjects
- Left Sidebar:** Subjects, Recently Viewed (0 items, Updated a few seconds ago)
- Modal:** New Subjects (Information tab)

|                      |                    |
|----------------------|--------------------|
| Subjects Name        | Owner              |
| <input type="text"/> | Kona Pawankalyan   |
| Subject ID           |                    |
| Paper 1              | --None--           |
| Paper2               | --None--           |
| Student              | Search Students... |
- Buttons:** New, Import, Change Owner, Cancel, Save & New, Save

The above screenshots shows the creation of fields on Subjects custom object.

## **DAY 3:**

Adding Business Logic To Application:

Creating Global Picklist Value Sets

First we go to the College object.

Create Global picklist value sets

College values are :

- MIT-HYD
- MIT-BLR
- MIT-MUM
- MIT-MAA
- MIT-DEL
- MIT-CCU

Paper 1 :

values are APEX

- JAVA
- C
- C++

Paper2 :

values are MATHEMATICS

- ENGLISH
- STATISTICS

**Picklist Value Sets**

Global picklist value sets let you share the 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.

| Action                        | Label   | Description |
|-------------------------------|---------|-------------|
| Edit   Del                    | College |             |
| Edit   Del                    | Paper1  |             |
| Edit   Del                    | Paper2  |             |
| Deleted Global Value Sets (0) |         |             |

## Creating Field Dependencies

Go the College object and then to Fields and relationship and go Field dependencies we create field dependency between college Name and Email

Create field dependency between college Name and capacity of students

**College Field Dependencies**

This page allows you to define dependencies between fields (e.g., dependent picklists).

| Action     | Controlling Field | Dependent Field      | Modified By                        |
|------------|-------------------|----------------------|------------------------------------|
| Edit   Del | College Name      | Email                | Kona Pawankalyan 1/5/2022, 1:35 AM |
| Edit   Del | College Name      | Capacity Of Students | Kona Pawankalyan 1/5/2022, 1:40 AM |

## Creating Validation Rules

First Create a validation rule on the College object and create a validation rule

The screenshot shows the Salesforce Object Manager interface for the 'College' object. On the left, a sidebar lists various configuration options like Details, Fields & Relationships, Page Layouts, and Validation Rules. The 'Validation Rules' option is selected. In the main content area, a 'College Validation Rule' is displayed with the following details:

- Validation Rule Detail:**
  - Rule Name: First\_Rule
  - Error Condition Formula: TEXT( College\_\_Name\_\_c ) <> Name
  - Error Message: Please Enter the Correct Field
  - Description: (empty)
  - Created By: Kona Pawankalyan 1/5/2022, 1:31 AM
  - Modified By: Kona Pawankalyan 1/5/2022, 1:37 AM
- Buttons:** Edit, Clone

The screenshot shows the 'Validation Rule Edit' screen for the 'College' object. The 'Validation Rules' option is selected in the sidebar. The main form contains the following fields:

- Validation Rule Edit:**
  - Rule Name: First\_Rule
  - Active: checked
  - Description: (empty)
- Quick Tips:** Operators & Functions
- Error Condition Formula:**
  - Example: Discount\_Percent\_\_c>0.30 | 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
  - Formula Editor:
    - Insert Field: College\_\_Name\_\_c
    - Insert Operator: <>
    - Formula: TEXT( College\_\_Name\_\_c ) <> Name
    - Functions dropdown: All Function Categories -->
      - ABS
      - ADDMONTHS
      - AND
      - BEGINS
      - BLANKVALUE
      - BR
    - Check Syntax button
  - Error Message:**
    - Example: Discount percent cannot exceed 30%

After completion of college object validation rule.

We have to Create a validation rule on the application form.

# First we create a Student Report type and then add the validation rule

**Application Form Validation Rule**

**Validation Rule Detail**

**Details**

**Fields & Relationships**

**Page Layouts**

**Lightning Record Pages**

**Buttons, Links, and Actions**

**Compact Layouts**

**Field Sets**

**Object Limits**

**Error Message**: Error: You have entered incorrect Field

**Description**

**Created By**: Kona Pawankalyan, 1/5/2022, 11:06 PM

**Modified By**: Kona Pawankalyan 1/5/2022, 11:08 PM

**Active**: ✓

**Help for this Page**

**Schema Builder**

**Elements Objects**

**Select objects to display on the builder**

**Select from**: All Objects

**Quick Find**

**Elements**

- Account
- Activity
- Alternative Payment Method
- API Anomaly Event Store
- Application Form
- Appointment Topic Time Slot
- Asset
- Asset Action
- Asset Action Source
- Asset Relationship
- Asset State Period
- Assigned Resource
- Associated Location
- Authorization Form
- Authorization Form Consent
- Authorization Form Text
- Authorization Form Use
- Business Brand
- Campaign
- Campaign Member
- Card Payment Method

**Objects**

- Application Form** (Selected)
  - Address
  - Application Form ID
  - Application Form Name
  - College
  - Contact Fee
  - Created By
  - Date of Birth
  - Email
  - Guardian Name
  - Hotel Fee
  - Last Modified By
  - Looking For Hotel Stay
  - Phone
  - Price
  - Ready To Join
  - Record Type
  - Student Name
- Student**
  - Address
  - Application Form
  - Application Form ID
  - Application Form Name
  - College Name
  - Created By
  - Data Of Birth
  - Email
  - Guardian Name
  - Hotel Fee
  - Last Modified By
  - Looking For Hotel Stay
  - Owner
  - Phone
  - Student Name
- Subjects**
  - Created By
  - Last Modified By
  - Paper 1
  - Paper 2
  - Student
  - Subject ID
  - Subject Name
- College**
  - Capacity Of Students
  - Contact Fee
  - Current PZ
  - College Name
  - Created By
  - Email
  - Home Fee

**DISPLAY DENSITY**

**Comfy**

**OPTIONS**

**Switch to Salesforce Classic**

**Add Username**

## DAY 4 :

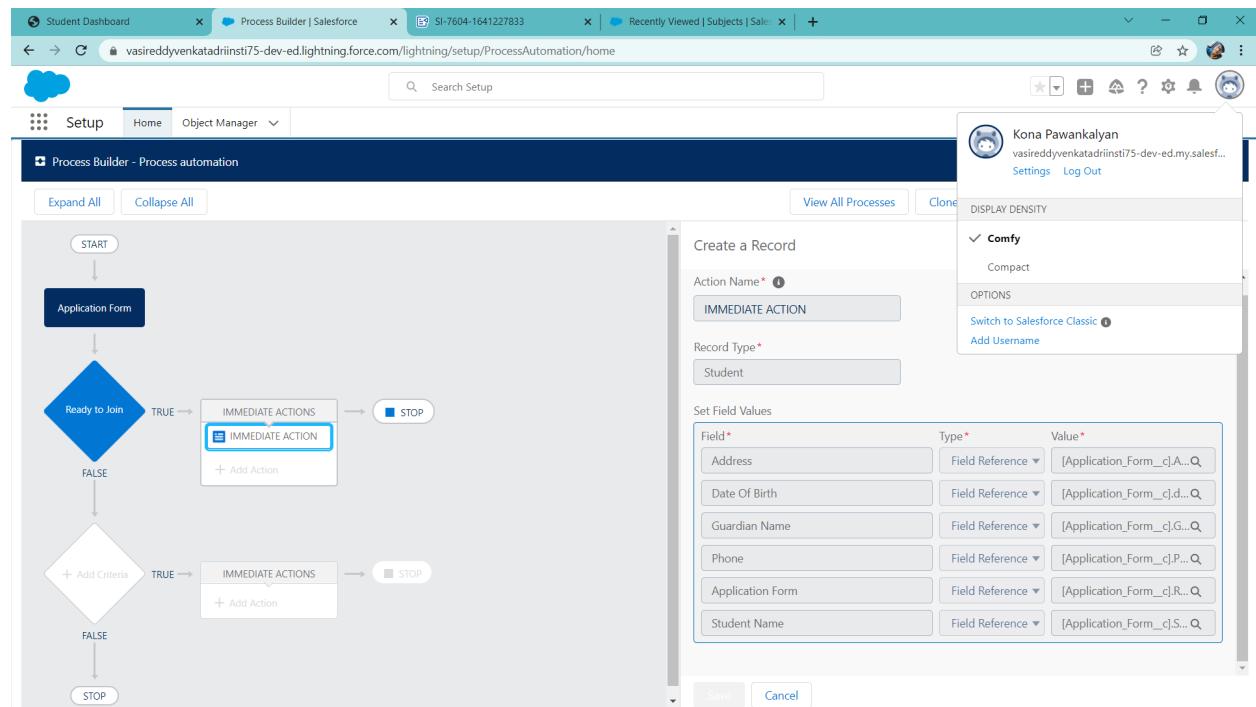
### Process Automation

Go to Setup -> select “Process Builder” from quick find And next click on NEW.

Create a Process Builder on the “Application Form”

Next click on Ready to join

And in adjacent box click on Immediate action and select create a record on the student object and set field values for Student type.

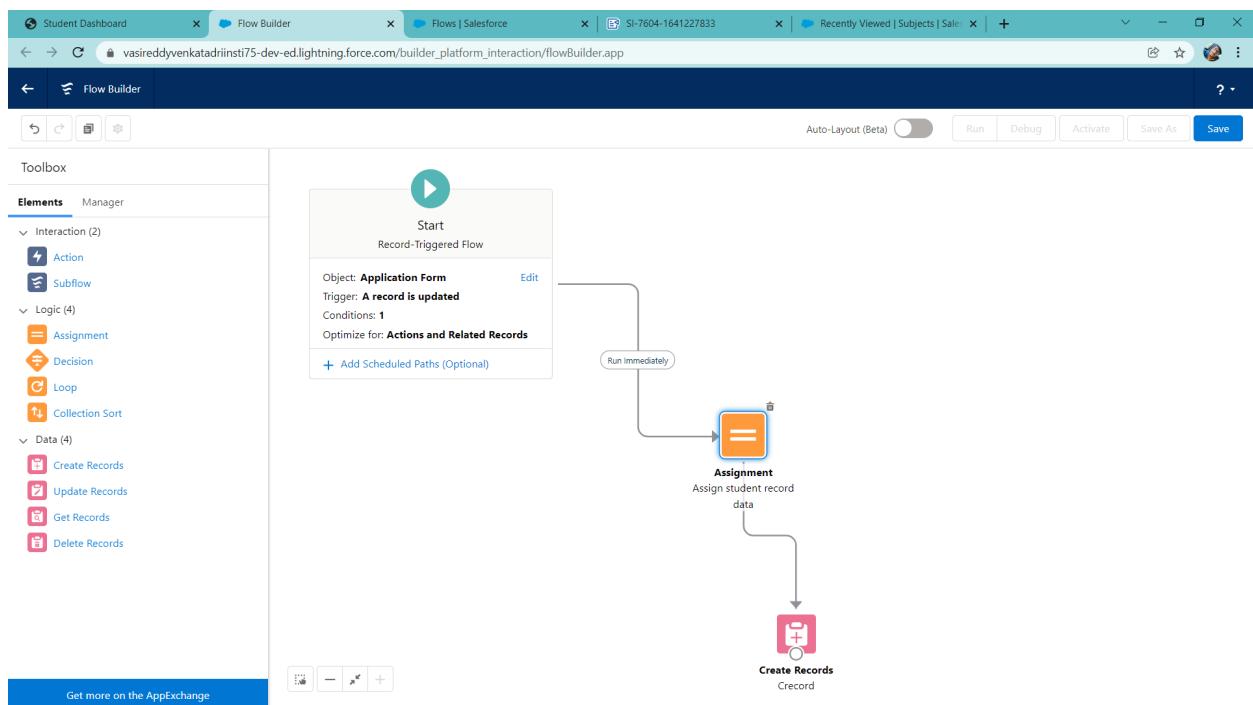


The above flow is an process automation. In which we create flow process and it is also called as Process Builder. Process Builder has more immediate actions as compared with work flows. Process Builder can update related record which work flow cannot.

## DAY 5:

### Create The Student Record Using Flow

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 to the Variable and Values
- save the Flow as Student flow.



## Flow Builders

1. It is more powerful when compared with workflow, process builders.
2. It can do any tasks where work flow and process builder cannot.
3. It is also called as LIGHTNING FLOW.
4. In previous days there is no outbound messages in flows. Now, we can access the outbound messages in flows also.

## Day 6:

# 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.

## Creating A Batch apex For Application Form

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

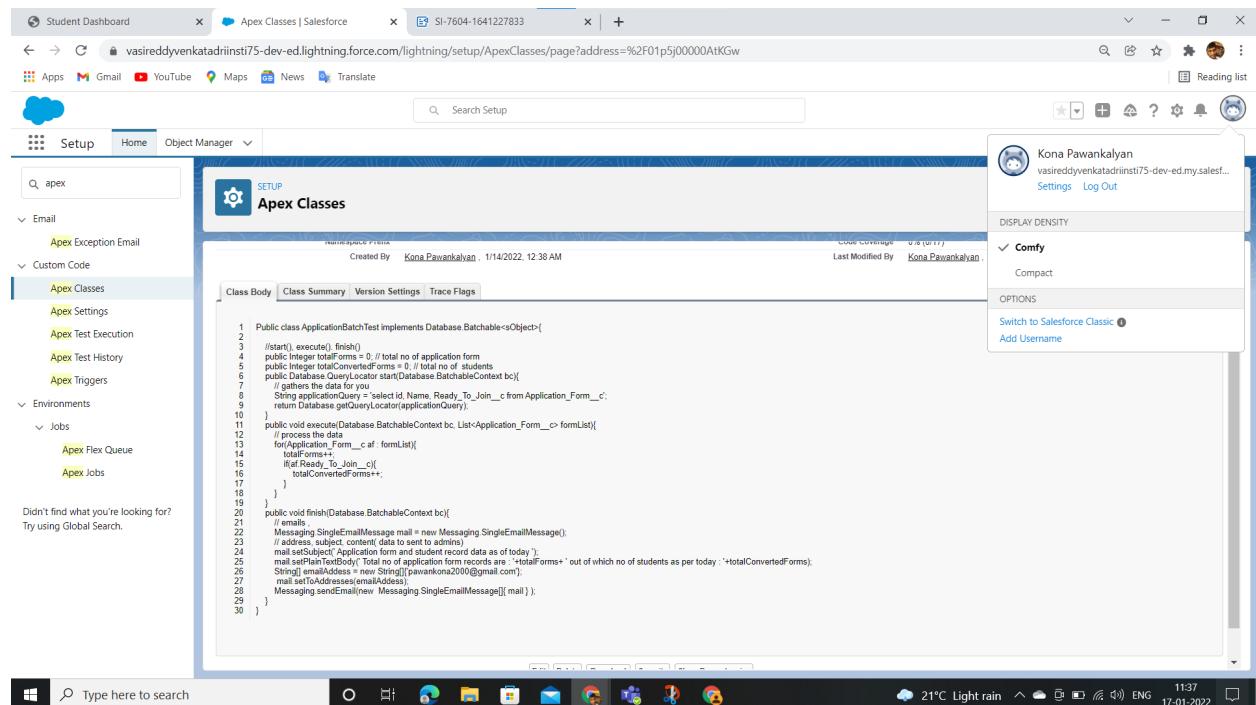
```

```

which no of students as per today : '+totalConvertedForms);
String[] emailAddess = new String[]{'your email address'};
mail.setToAddresses(emailAddess);
Messaging.sendEmail(new Messaging.SingleEmailMessage[]{ mail } );
}
}
}

```

Now, I am creating a scheduler class



## Creating 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){

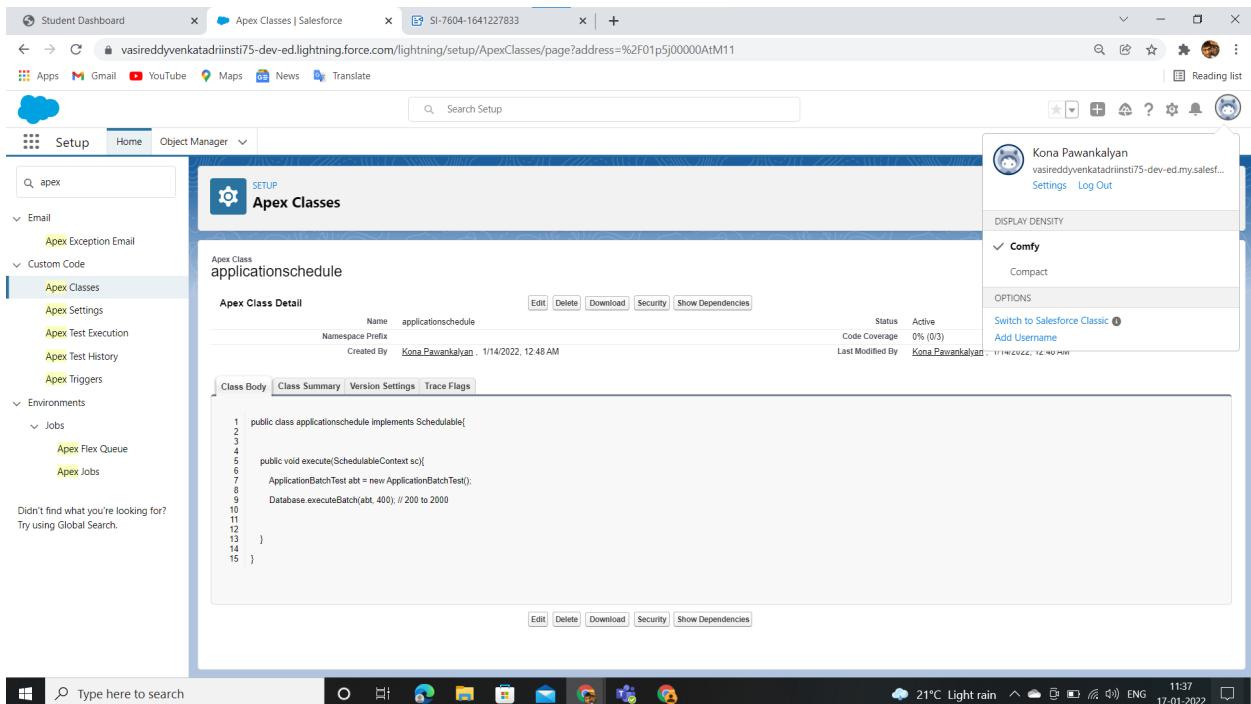
    ApplicationBatchTest abt = new ApplicationBatchTest();

    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 require.



# Day 7

On the day 7, we learnt Apex Trigger.

***There are different types of triggers:***

- before insert.
- before update.
- before delete.
- after insert.
- after update.
- after delete.
- after undelete

There are two types of triggers BEFORE and AFTER

Operations are insert, update, delete, undelete.

Trigger context variable

trigger.old, trigger.new, trigger.new-map, trigger.old map.

# Day 8

## Lightning Web Components

Lightning Web Components (LWC) is a stack of modern lightweight frameworks built on the latest web standards. It is a DOM (Document Object Model), element created through reusable code and is used to generate a dynamic interface without using JavaScript or building a Library. This feasibility makes it quick and seamless, saving the developers a ton of time and effort on the Web Stack. Let's look at some of its remarkable features:

- Improved performance of the component as most of the code is recognized by the native web browser engine and web stack
- Ability to compose applications using smaller chunks of code since the

crucial elements that are required to create a component is part of the native web browser engine and web stack

- Increase in the robustness of the applications built using LWCs as they are inclusive of the said modern web standards.
- Parallel interoperability and feasibility to use both Lightning Web Components and Aura components together in the applications with no visible differentiation to the end-users

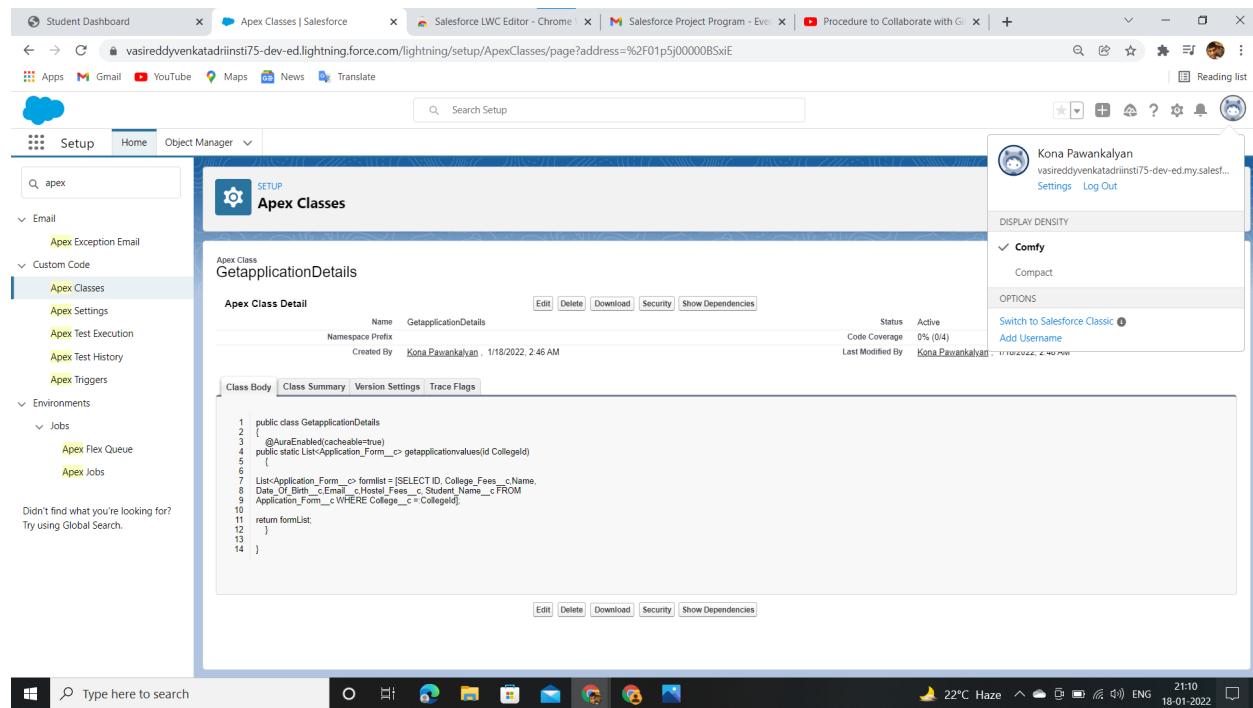
## Lightning Web Components

Create College DataTable Component( APEX CLASS)

First Go to setup and search Apex classes and next click onDeveloper console and create a apex class with name GetapplicationDetails and enter the code and click on save to save the apex class

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 .

Create an Apex Class With The Required Functionality



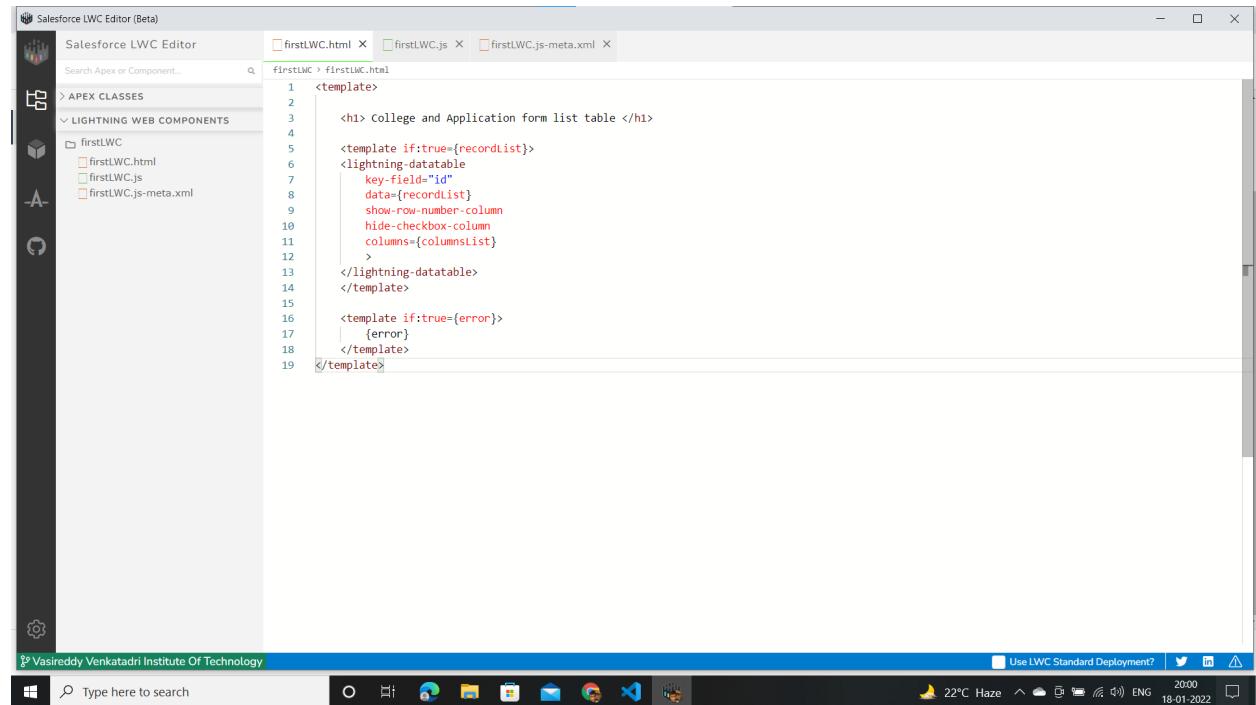
# Day 9

## Create College DataTable Component (HTML FILE)

Open visual studio app and search for sf:create a new lightning component.

after clicking that I named it as "FirstLWC". It creates a three types of extension files, .html, .js, .js.meta.xml.

After clicking that copy the code into the firstLWC.html.



The screenshot shows the Salesforce LWC Editor (Beta) interface. On the left, there's a sidebar with icons for Apex Classes, Lightning Web Components, and a search bar. Under 'LIGHTNING WEB COMPONENTS', a 'firstLWC' component is selected, showing its three files: firstLWC.html, firstLWC.js, and firstLWC.js-meta.xml. The main area displays the code for firstLWC.html:

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

# Day 10

## Create College DataTable Component(JAVA SCRIPT FILE)

After the creation of the Html file create the java script file name it as college date table

```
//import { LightningElement } from 'lwc';
export default class FirstLWC 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 : 'Student Name', fieldName : 'Student_Name__c', type:'text' }
    ];
    @api recordId;
    recordList;
    error;
    @wire(getapplicationvalues, {collegeId : '$recordId'})
    wiredCollegeData({data, error}){
```

The screenshot shows the Salesforce LWC Editor interface. On the left, there's a sidebar with icons for Apex Classes, Lightning Web Components, and a search bar. The main area displays the code for `firstLWC.js`:

```

1 //import { LightningElement } from 'lwc';
2
3 //export default class FirstLWC extends LightningElement {}
4
5
6 import { LightningElement, api, wire } from 'lwc';
7 import getapplicationvalues
8 from '@salesforce/apex/GetapplicationDetails.getapplicationvalues';
9 export default class collegedatatable extends lightningElement {
10
11    columnsList = [
12      {label : 'Application Form' , fieldName : 'Name', type:'text' },
13      {label : 'College Fees' , fieldName : 'College_Fees__c', type:'currency' },
14      {label : 'Date of Birth' , fieldName : 'Date_of_Birth__c', type:'date' },
15      {label : 'Email' , fieldName : 'Email__c', type:'email' },
16      {label : 'Hostel Fees' , fieldName : 'Hostel_Fees__c', type:'Currency' },
17      {label : 'Student Name' , fieldName : 'Student_Name__c', type:'text' }
18    ];
19
20    @api recordid;
21    recordList;
22    error;
23
24    @wire(getapplicationvalues, {CollegeId : '$recordId'})
25    wiredcollegeData({data, error}) {
26      if(data){
27        this.recordList = data;
28      }
29      else if(error){
30        this.error = error;
31        this.recordList = undefined;
32      }
33    }
}

```

The status bar at the bottom indicates "Vasireddy Venkata Institute Of Technology".

At last we created a meta file in visual studio code.

## Create College DataTable Component(META FILE)

The screenshot shows Visual Studio Code with the workspace titled "firstLWCjs-meta.xml - collegemanagement - Visual Studio Code". The Explorer panel on the left shows a project structure with files like `firstLWC.html`, `firstLWC.js`, and `firstLWCjs-meta.xml`. The `firstLWCjs-meta.xml` file is selected and its content is displayed in the center:

```

<?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 status bar at the bottom indicates "26°C Haze" and the date "18-01-2022".

Salesforce LWC Editor (Beta)

Salesforce LWC Editor

Search Apex or Component...

APEX CLASSES

LIGHTNING WEB COMPONENTS

firstLWC

- firstLWC.html
- firstLWC.js
- firstLWC.js-meta.xml

firstLWC > firstLWC.js-meta.xml

```
1 <?xml version="1.0"?>
2 <lightningComponentBundle
3 xmlns="http://soap.sforce.com/2006/04/metadata">
4   <apiVersion>51.0</apiVersion>
5   <i18nExcluded>false</i18nExcluded>
6   <targets>
7     <target>lightning__RecordPage</target>
8     <target>lightning__AppPage</target>
9     <target>lightning__HomePage</target>
10   </targets>
11 </LightningComponentBundle>
```

Vasireddy Venkatadri Institute Of Technology  Use LWC Standard Deployment?   

Type here to search        22°C Haze 2000 ENG 18-01-2022