

Day1:Activities

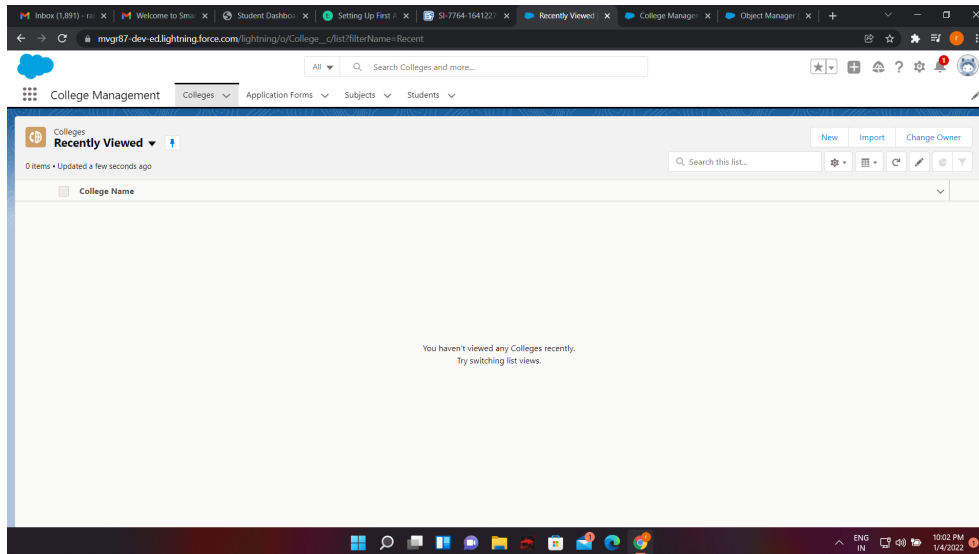
1-->Created my Salesforce Development org

2-->Activated my Account

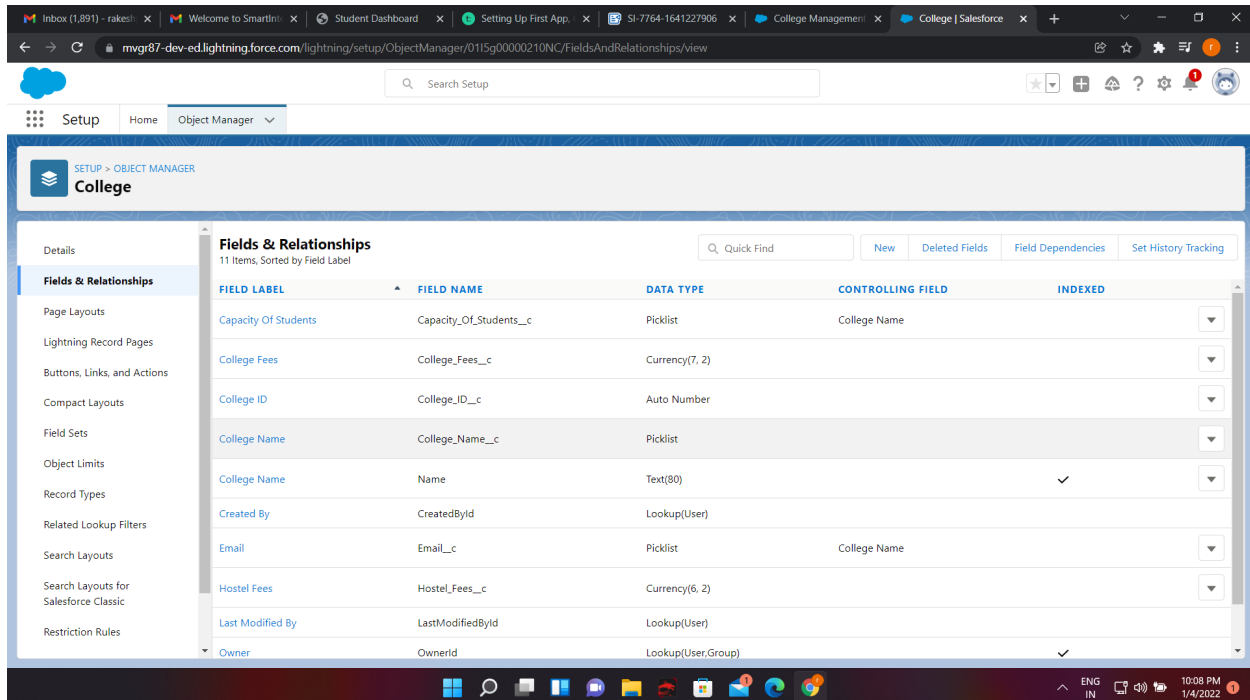
3-->Logged into my account

Day2:Activities

1-->Created a new Application with the name College Management

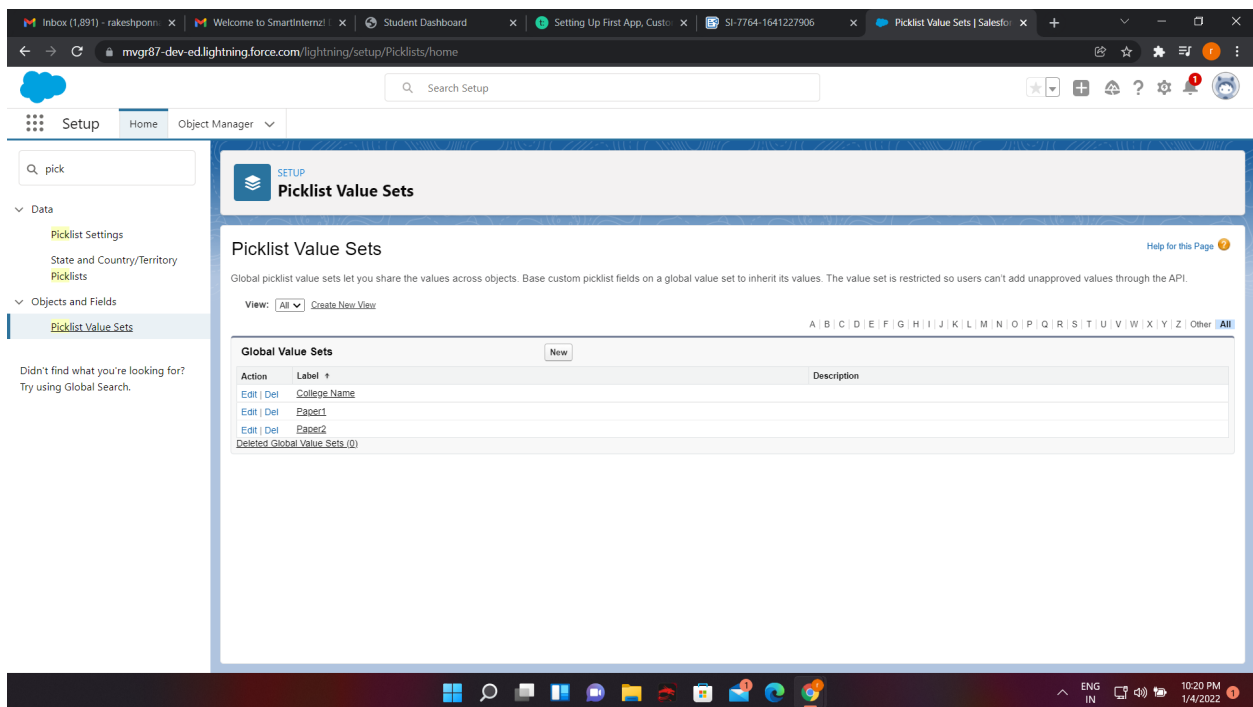


2-->Created fields on College Object



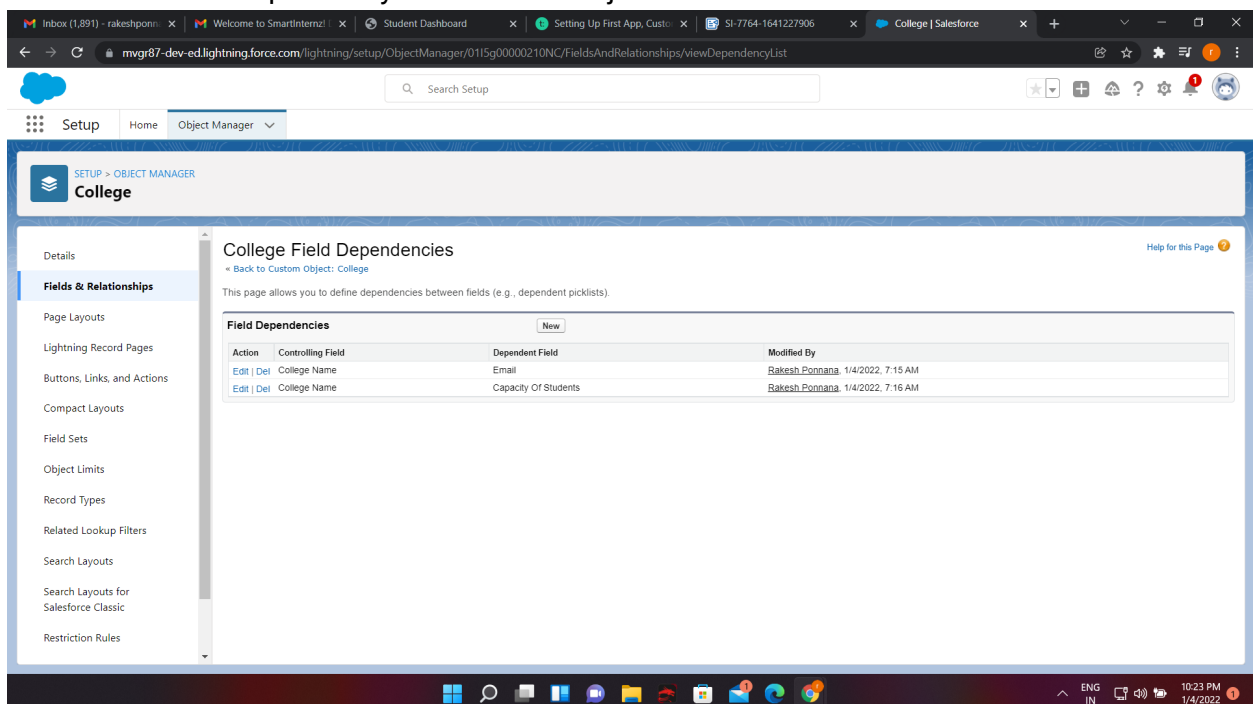
3-->Similarly created fields in Application Form Object,Student Object and in Subject Object

4-->Created Global value Sets



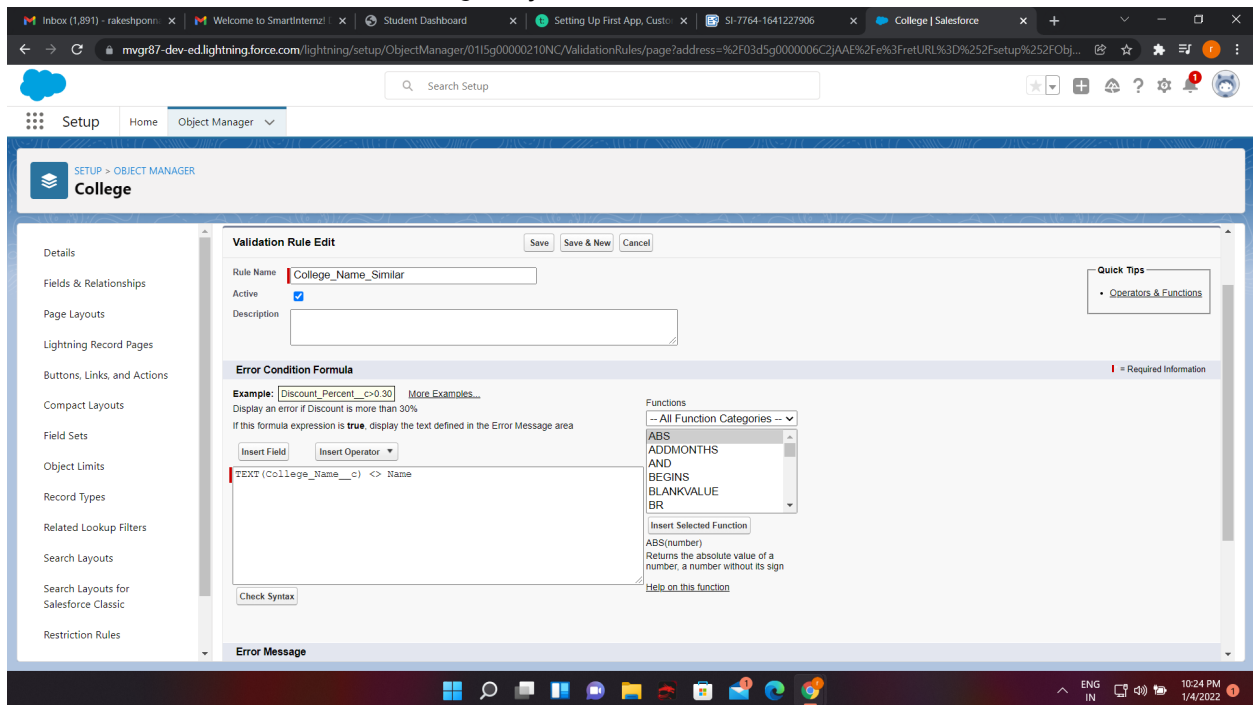
The screenshot shows the Salesforce Setup interface for 'Picklist Value Sets'. The left sidebar contains a search bar with 'pick' and a navigation menu with 'Data', 'Picklist Settings', 'State and Country/Territory', 'Picklists', 'Objects and Fields', and 'Picklist Value Sets'. The main content area is titled 'Picklist Value Sets' and includes a description: '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.' Below this is a 'Global Value Sets' table with columns 'Action', 'Label', and 'Description'. The table lists 'College Name', 'Page1', 'Page2', and 'Deleted Global Value Sets (0)'. A 'New' button is located above the table. The bottom of the screen shows the Windows taskbar with various application icons and the system clock indicating 10:20 PM on 1/4/2022.

5-->Created Field Dependency between two objects



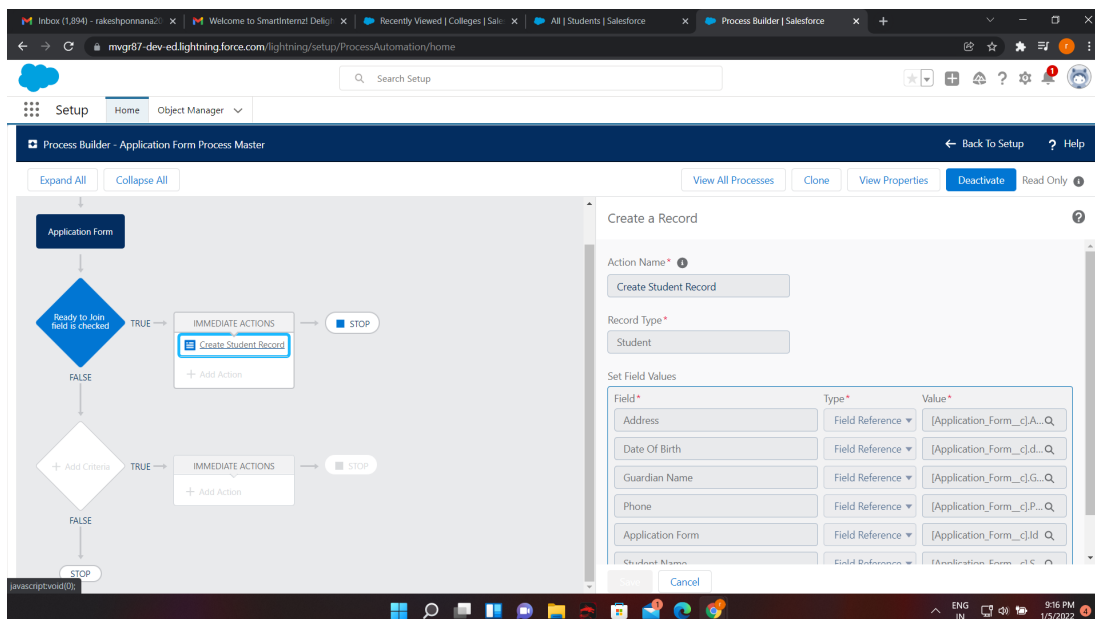
The screenshot shows the Salesforce Setup interface for 'College Field Dependencies'. The left sidebar contains a search bar with 'Search Setup' and a navigation menu with 'Details', 'Fields & Relationships', 'Page Layouts', 'Lightning Record Pages', 'Buttons, Links, and Actions', 'Compact Layouts', 'Field Sets', 'Object Limits', 'Record Types', 'Related Lookup Filters', 'Search Layouts', 'Search Layouts for Salesforce Classic', and 'Restriction Rules'. The main content area is titled 'College Field Dependencies' and includes a description: 'This page allows you to define dependencies between fields (e.g., dependent picklists)'. Below this is a 'Field Dependencies' table with columns 'Action', 'Controlling Field', 'Dependent Field', and 'Modified By'. The table lists two dependencies: 'College Name' (controlling) to 'Email' (dependent) and 'College Name' (controlling) to 'Capacity Of Students' (dependent). The bottom of the screen shows the Windows taskbar with various application icons and the system clock indicating 10:23 PM on 1/4/2022.

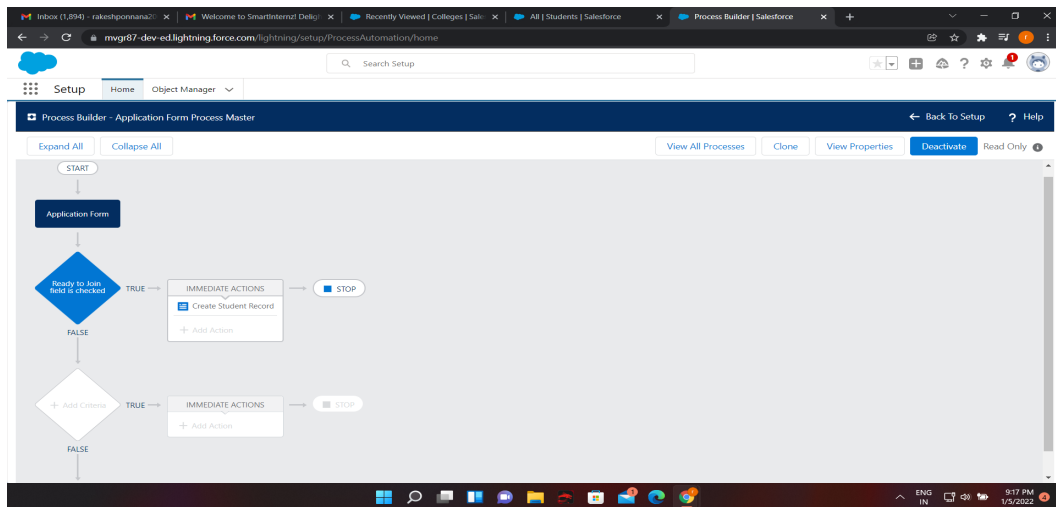
6-->Created validation Rule in College object



Day 3: Activities

1--> Created the connection between two objects in a way such that if there is any change in one object it is reflected in the other object. This is done by using PROCESS BUILDER with the condition of "ready to join" field is checked on the application form object. We need to create the student record automatically with the information specified in the application form record.





College Management

Application Form

College Name

Application Form Name

College Name

Application Form ID

F-000001

Address

1-69,pedha veedhi,cinnakollivalasa,srikakulam,532459

College

MIT-HYD

College Fees

\$30,000.00

Hostel Fees

\$3,000.00

Date of Birth

1/5/2022

Email

rakeshponnana2001@gmail.com

Guardian Name

Tavittinaidu

Looking For Hostel Stay

☐

Ready To Join

☒

Student Name

Rakesh

College Management

Students

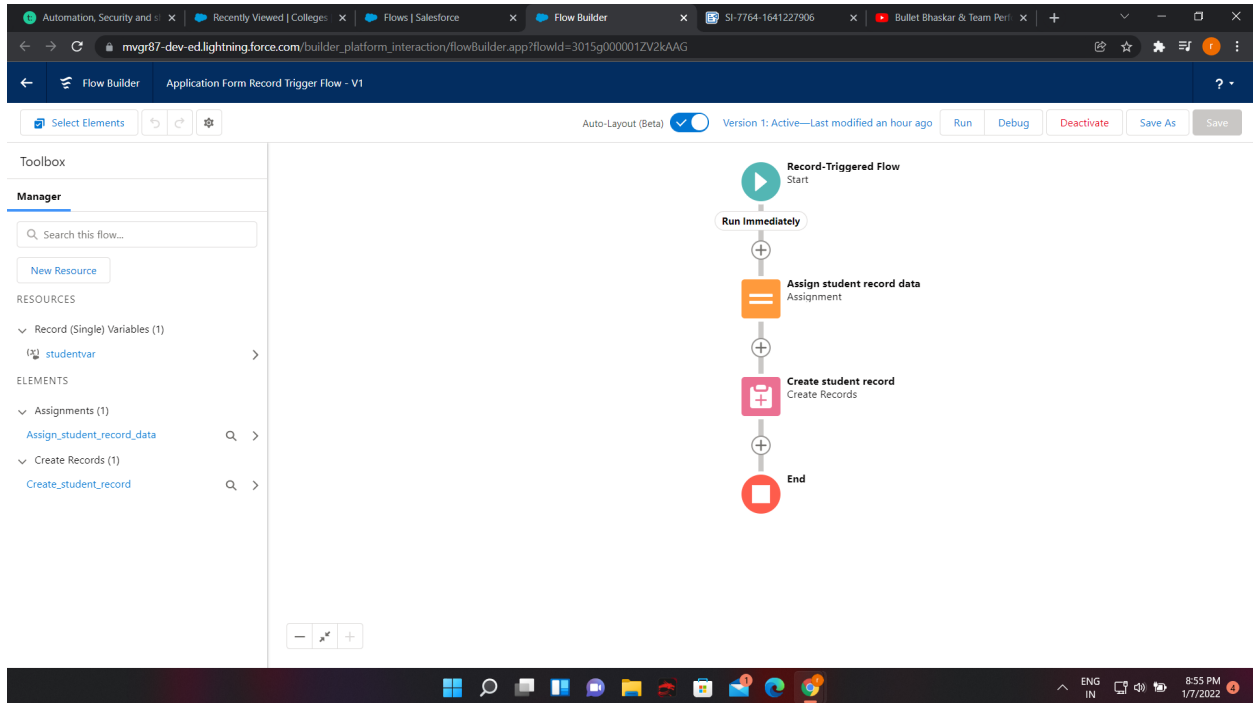
1 item • Sorted by Student Name • Filtered by All students • Updated a few seconds ago

Search this list...

	Student Name ↑
1	Rakesh

Day 4: Activities:

1--> Created a Student record work flow in record triggered flow a variable was also created with the name student and add a assignment to it



Day 5: Activities

1--> In Developer console we created a Apex class with a name ApplicationBatchTest

In this class we used three methods namely

- 1) Start method
- 2) Execute method
- 3) Finish method

The screenshot shows the Salesforce Developer Console with the 'ApplicationBatchTest.apex' class open. The class implements the `Database.Batchable<Object>` interface. It includes methods for `start`, `execute`, and `finish`. The `execute` method iterates through a list of `Application_Form__c` records, incrementing `totalForms` and `totalConvertedForms` based on the `Ready_To_Join__c` field. The `finish` method sends an email summary.

```
1 public class ApplicationBatchTest implements Database.Batchable<Object>{
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    }
27 }
```

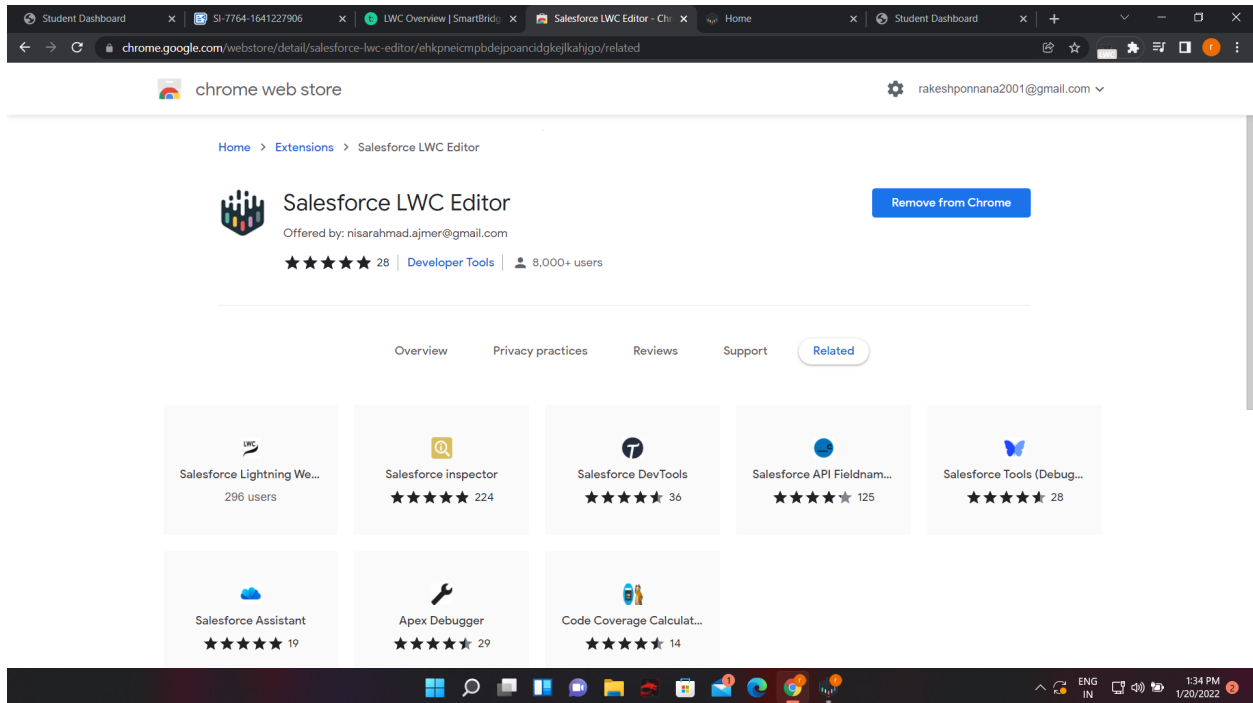
2-->Similarly created another new Apex Class with a name applicationschedule

The screenshot shows the Salesforce Developer Console with the 'applicationschedule.apex' class open. The class implements the `Schedulable` interface. It includes an `execute` method that creates an instance of `ApplicationBatchTest` and executes a batch of records.

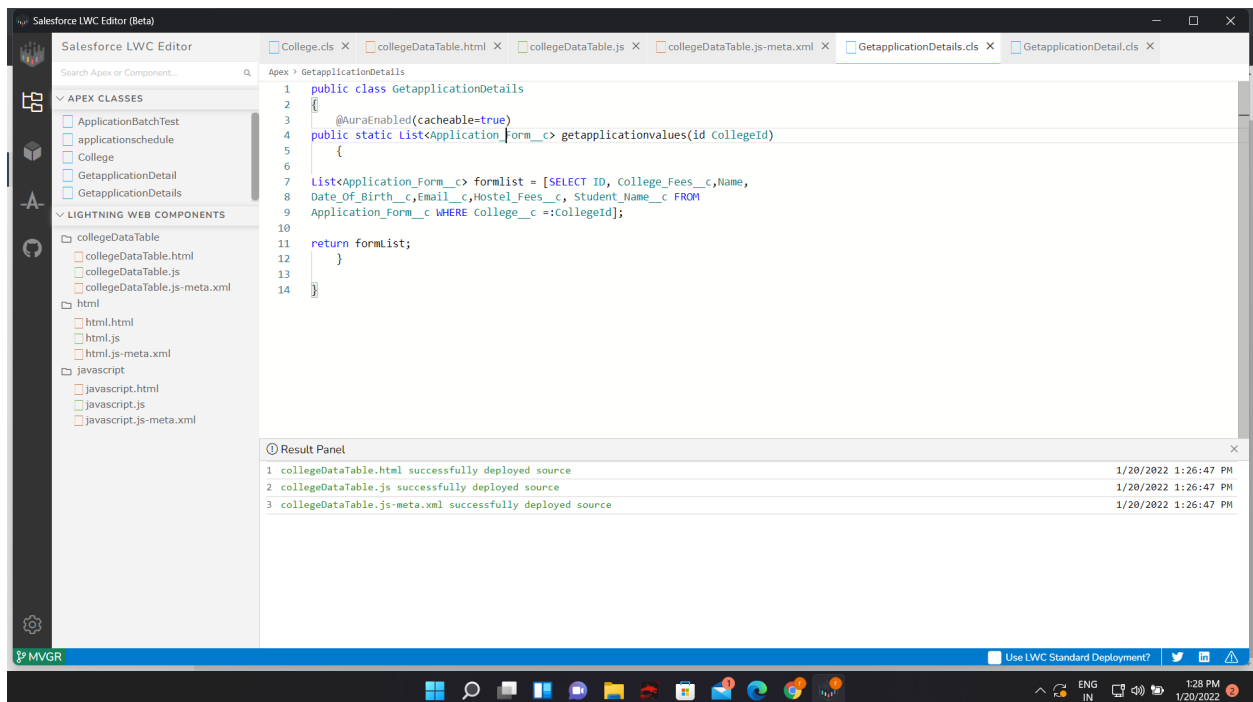
```
1 public class applicationschedule implements Schedulable{
2
3
4
5     public void execute(SchedulableContext sc){
6
7         ApplicationBatchTest abt = new ApplicationBatchTest();
8
9         Database.executeBatch(abt, 400); // 200 to 2000
10
11
12
13    }
14
15 }
```

Last Activity:

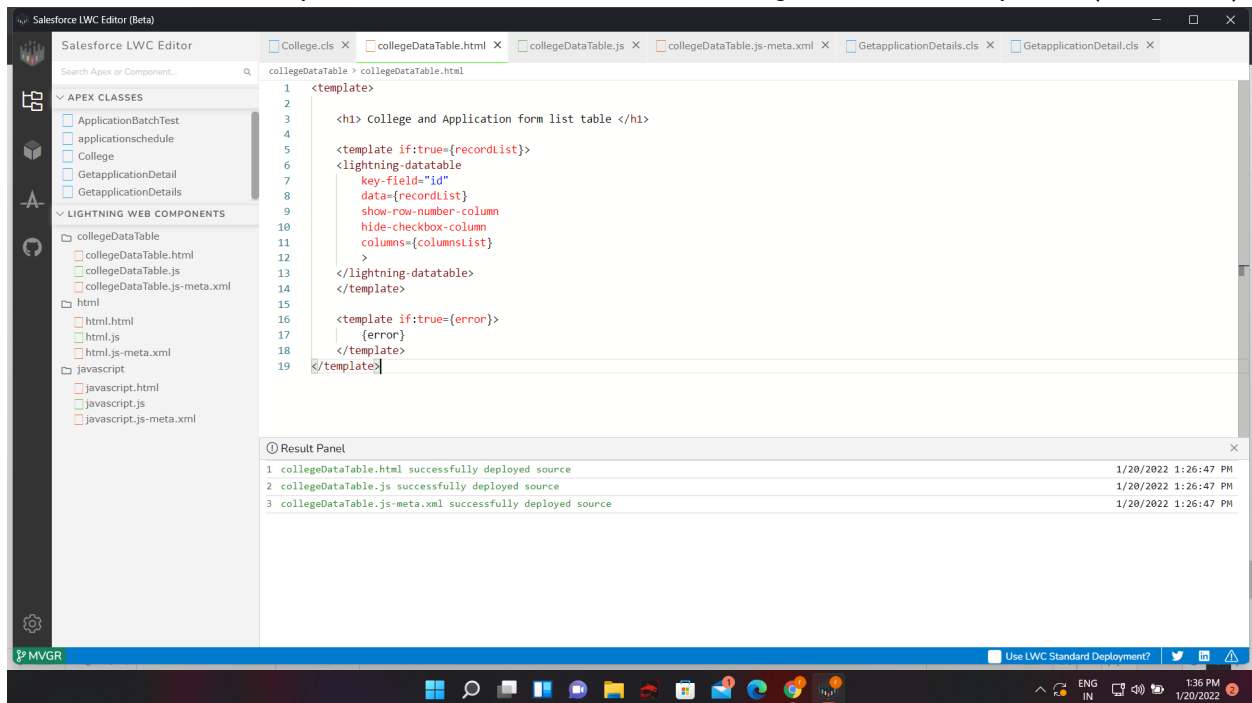
1-->Created a chrome extension



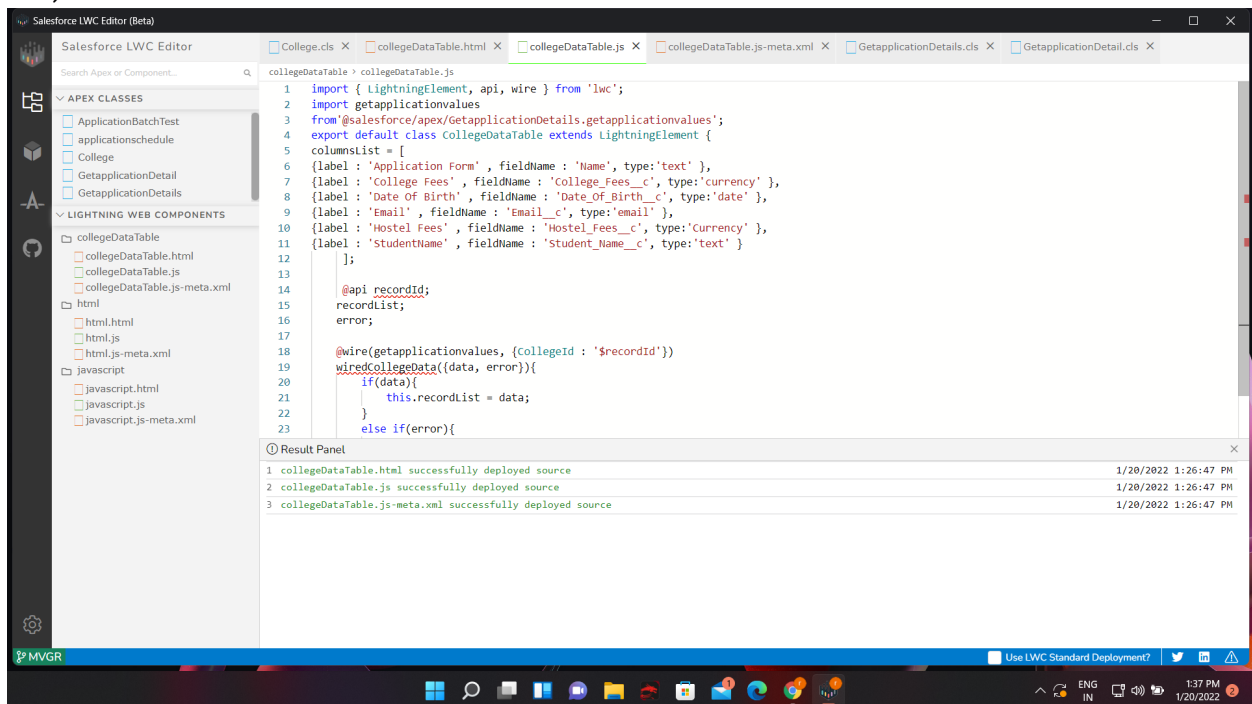
2-->With the help of chrome extension Created CollegeDataBase Component(Apex Class)



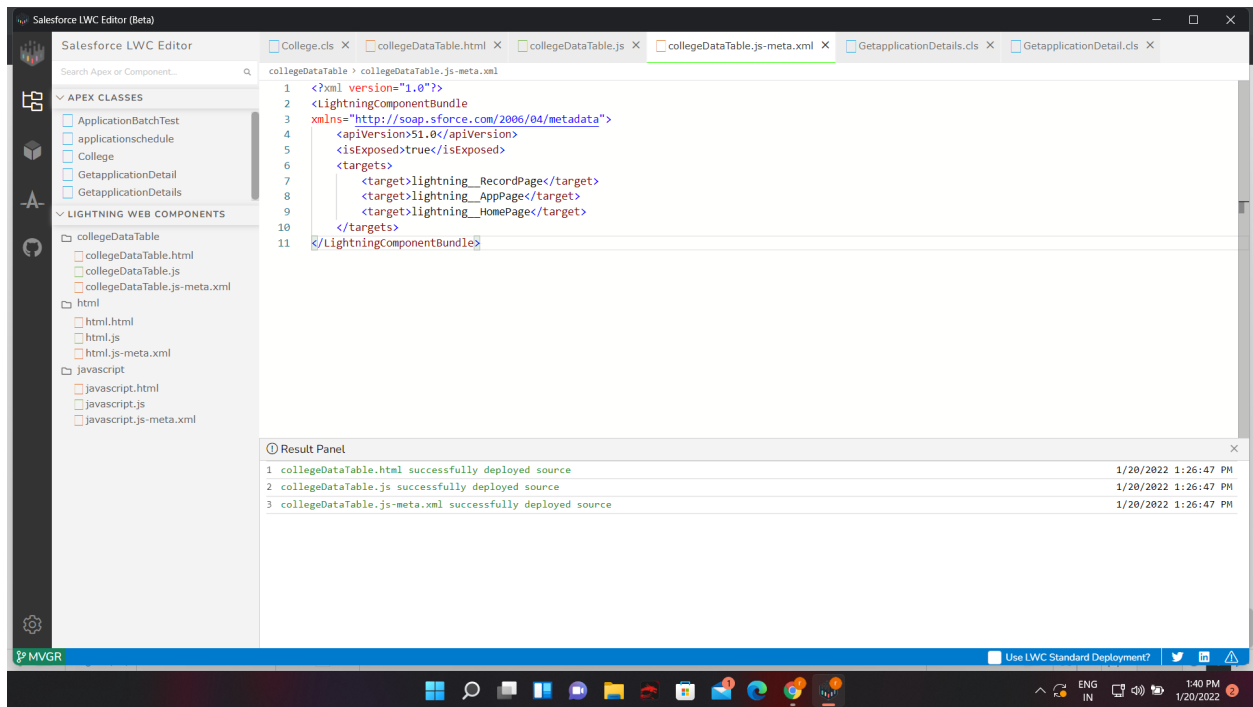
3-->With the help of chrome extension Created CollegeDataBase Component(HTML File)



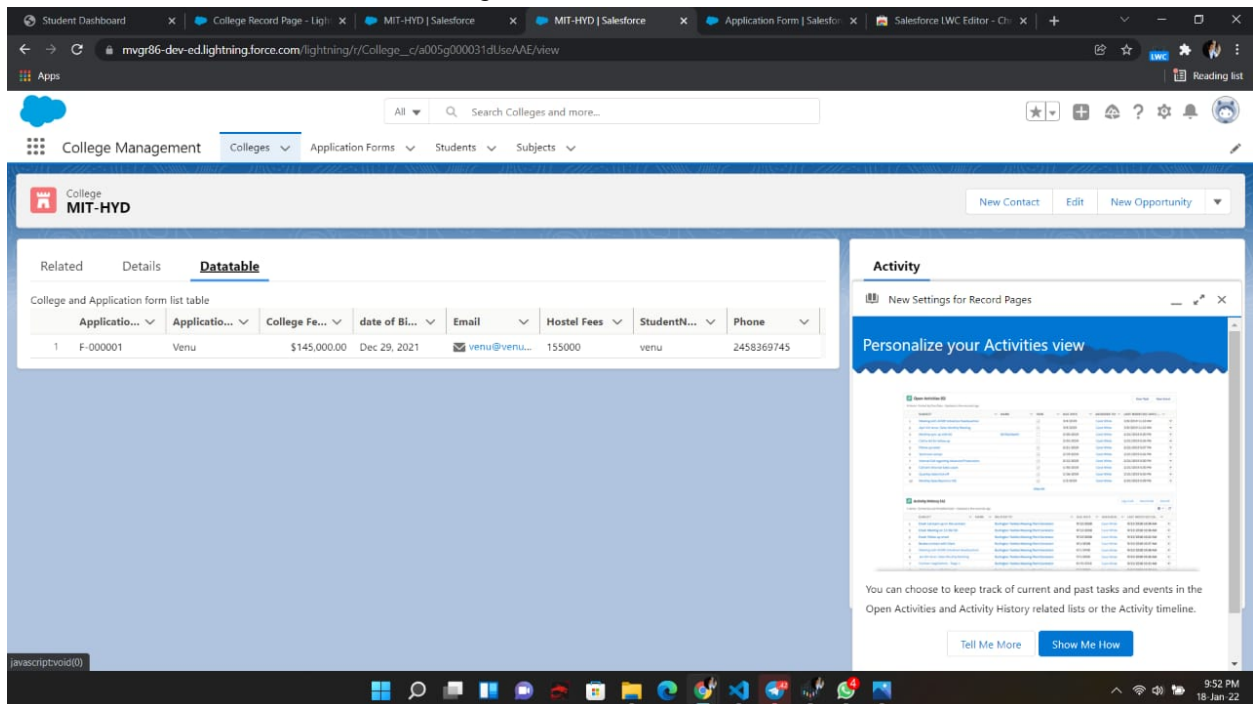
4-->With the help of chrome extension Created CollegeDataBase Component(Java Script File)

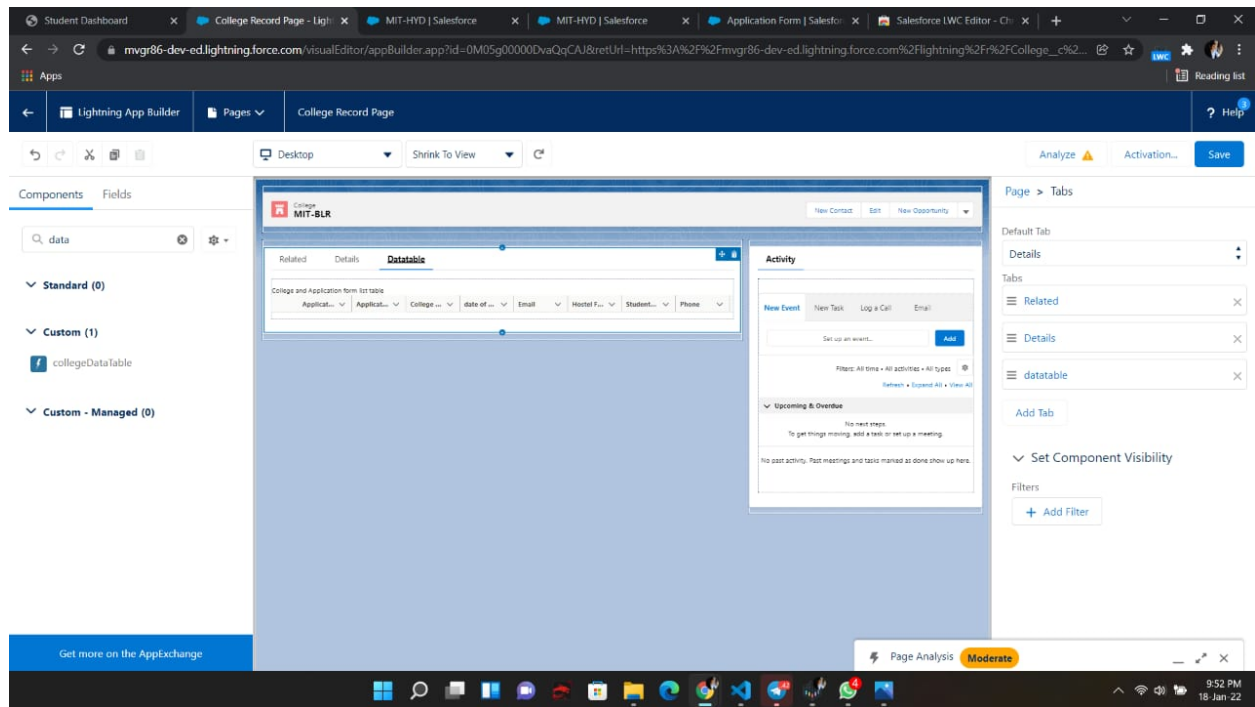


5-->With the help of chrome extension Created CollegeDataBase Component(META File)



As the result of the above codes we get





**ALL THE TASKS HAS BEEN COMPLETED BY ME WHICH ARE
SHOWN IN MY DASHBOARD
THANKING YOU SMARTINTERNZ**