

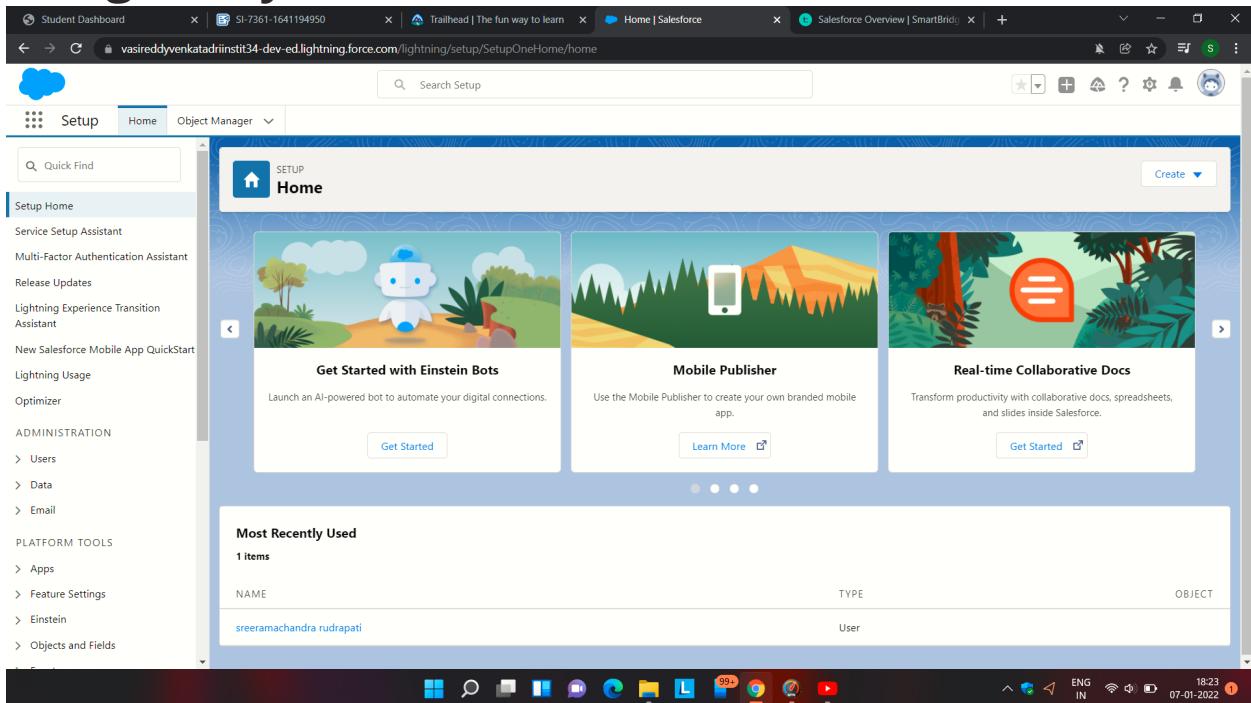
College Management Application

day 1:

1.Created Developer Account

2.Account Activation

3.Login To your Salesforce Account



day 2:

Create Fields On College Object

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

- Page Title: Application Form
- Section: Fields & Relationships
- Table Headers: FIELD LABEL, FIELD NAME, DATA TYPE, CONTROLLING FIELD, INDEXED
- Table Data:

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Application Form Name	Name	Auto Number		✓
college	college__c	Master-Detail(college)		✓
college fee	college_fee__c	Formula (Currency)		
Created By	CreatedById	Lookup(User)		
Date Of Birth	DateOfBirth__c	Date		
Email	Email__c	Email (Unique)		✓
Last Modified By	LastModifiedById	Lookup(User)		
Student Name	Student_Name__c	Text(30)		

Create Fields On Application Form Object

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

- Page Title: Application Form
- Section: Fields & Relationships
- Table Headers: FIELD LABEL, FIELD NAME, DATA TYPE, CONTROLLING FIELD, INDEXED
- Table Data:

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Application Form Name	Name	Auto Number		✓
college	college__c	Master-Detail(college)		✓
college fee	college_fee__c	Formula (Currency)		
Created By	CreatedById	Lookup(User)		
Date Of Birth	DateOfBirth__c	Date		
Email	Email__c	Email (Unique)		✓
Last Modified By	LastModifiedById	Lookup(User)		
Student Name	Student_Name__c	Text(30)		

Create Fields On Student Object

The screenshot shows the Salesforce Object Manager interface for the 'Student' object. The left sidebar lists various setup categories like Page Layouts, Lightning Record Pages, Buttons, etc. The main content area displays the 'Fields & Relationships' section with 10 items. The table columns are FIELD LABEL, FIELD NAME, DATA TYPE, CONTROLLING FIELD, and INDEXED. The fields listed are:

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Address	Address__c	Text Area(255)		
Application Form	Application_Form__c	Lookup(Application Form)		
college Name	college_Name__c	Formula (Text)		
Created By	CreatedById	Lookup(User)		
Date of Birth	Date_of_Birth__c	Date		
Guraden Name	Guraden_Name__c	Text(30)		
Last Modified By	LastModifiedById	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		
Phone	Phone__c	Phone		
Student Name	Name	Text(80)		

Create Fields On Subject Object

The screenshot shows the Salesforce Object Manager interface for the 'Subject' object. The left sidebar lists various setup categories. The main content area displays the 'Fields & Relationships' section with 7 items. The table columns are FIELD LABEL, FIELD NAME, DATA TYPE, CONTROLLING FIELD, and INDEXED. The fields listed are:

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedBy	Lookup(User)		
Last Modified By	LastModifiedBy	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		
paper1	paper1__c	Picklist		
paper2	paper2__c	Picklist		
Student	Name__c	Lookup(Student)		
Subject Name	Name	Auto Number		

DAY 3:

Creating Global Picklist Value Sets

a)College

The screenshot shows the Salesforce Setup interface with the 'Picklist Value Sets' page open. The 'Global Value Set Detail' section is displayed, with the 'Information' tab selected. The 'Label' is set to 'College Name' and the 'Name' is 'College_Name'. The 'Description' field is empty. Below this, the 'Picklist Values Used' section shows six active values: MIT - BLR, MIT - MAA, MIT - HYD, MIT - DEL, MIT - MUM, and MIT - CCU. Each value has a 'Default' checkbox and a 'Modified By' field showing 'sreeramachandra.rudrapati' and the date '1/9/2022, 1:16 AM'. The 'Values' table at the bottom lists these values again with their API names: APEX, JAVA, C, and C++. The 'Inactive Values' section is empty.

b)Paper1

The screenshot shows the Salesforce Setup interface with the 'Picklist Value Sets' page open. The 'Global Value Set Detail' section is displayed, with the 'Information' tab selected. The 'Label' is set to 'Paper 1' and the 'Name' is 'Paper_1'. The 'Description' field is empty. Below this, the 'Picklist Values Used' section shows four active values: APEX, JAVA, C, and C++. The 'Values' table at the bottom lists these values again with their API names: APEX, JAVA, C, and C++. The 'Inactive Values' section is empty. A 'Fields Where Used' section is also present at the bottom.

c)Paper2

The screenshot shows the Salesforce Setup interface with two windows open:

- Top Window:** Displays a "Global Value Set" named "Paper_2". The label is "Paper 2".
- Bottom Window:** Shows the "Global Value Set Detail" page for "Paper_2". It lists "Picklist Values Used" with a note: "Active and inactive picklist values 3 (1,000 max)".

Creating Field Dependencies

field dependency between college Name and Email

The screenshot shows the Salesforce Object Manager interface with the "Edit Field Dependency" page for the "college" object:

- Controlling Field:** College Name
- Dependent Field:** College Email
- Instructions:**
 - Double click on a cell to toggle its visibility for the Controlling Field value shown in the column heading.
 - To change multiple cells at once, select multiple cells and then click the Include Values or Exclude Values button to change the visibility of all selected cells at once.
 - Use SHIFT + click to select a range of adjacent cells. Use CTRL + click to select multiple cells that are not adjacent.
 - Use the Preview button to test the results.
- Grid View:** A 5x6 grid where each row represents a college name and each column represents a college email. Cells are color-coded (yellow, green, blue) to indicate visibility status. Buttons for "Include Values" and "Exclude Values" are located at the top of each column.

field dependency between college Name and capacity of students

Screenshot of the Salesforce Setup interface showing the 'Edit Field Dependency' page for the 'college' object.

Page Layout:

- Controlling Field: College Name
- Dependent Field: Capacity Of Students

Instructions:

- Double click on a cell to toggle its visibility for the Controlling Field value shown in the column heading.
- To change multiple cells at once, select multiple cells and then click the **Include Values** or **Exclude Values** button to change the visibility of all selected cells at once.
- Use SHIFT + click to select a range of adjacent cells. Use CTRL + click to select multiple cells that are not adjacent.
- Use the Preview button to test the results.

Legend:

- Excluded Value (Grey)
- Included Value (Yellow)

Table Data:

College Name:	MIT - BLR	MIT - MAA	MIT - HYD	MIT - DEL	MIT - MUM
Capacity Of Students:	500-1000 1000-2500 2500-6000 6000-10000	500-1000 1000-2500 2500-6000 6000-10000	500-1000 1000-2500 2500-6000 6000-10000	500-1000 1000-2500 2500-6000 6000-10000	500-1000 1000-2500 2500-6000 6000-10000

Buttons:

- Save | Cancel | Preview

Create a validation rules the college object

Screenshot of the Salesforce Setup interface showing the 'college Validation Rule' page.

Validation Rule Detail:

Rule Name	college_name_similar	Active
Error Condition Formula	Name <> TEXT(College_Name__c)	Top of Page
Error Message	please use the college name in record info.	Error Location
Description		
Created By	sreeramachandra.rudrapati	Modified By

Buttons:

- Edit | Clone

the application form object

All | colleges | Salesforce | Object Manager | Salesforce | Student Dashboard | Student Dashboard | SI-7361-1641194950

Application Form Validation Rule

Back to Application Form

Validation Rule Detail

Rule Name	stop_any_modification	Active	✓
Error Condition Formula	AND(Ready_To_Join__c == true, OR(ISCHANGED(Address__c), ISCHANGED(college__c), ISCHANGED(DateOfBirth__c), ISCHANGED>Email__c), ISCHANGED(GuardianName__c), ISCHANGED(Phone__c)))		
Error Message	fill the form.	Error Location	Top of Page
Description		Modified By	sreeramachandra rudrapati 1/18/2022, 6:35 AM
Created By	sreeramachandra rudrapati 1/18/2022, 6:35 AM	Created Date	18-01-2022

Process Automation

Recently Viewed | Application Form | Process Builder | Salesforce | Student Dashboard | Student Dashboard | SI-7361-1641194950

Process Builder - Application Form

Back to Setup ? Help

Expand All Collapse All

```

graph TD
    START([START]) --> ApplicationForm[Application Form]
    ApplicationForm --> ReadyJoin{Ready to Join}
    ReadyJoin -- TRUE --> CreateRecord[Create Record]
    CreateRecord --> STOP1([STOP])
    ReadyJoin -- FALSE --> AddCriteria{+ Add Criteria}
    AddCriteria -- TRUE --> AddAction[+ Add Action]
    AddAction --> STOP2([STOP])
    AddCriteria -- FALSE --> STOP3([STOP])

```

Create a Record

Action Name * Create Record

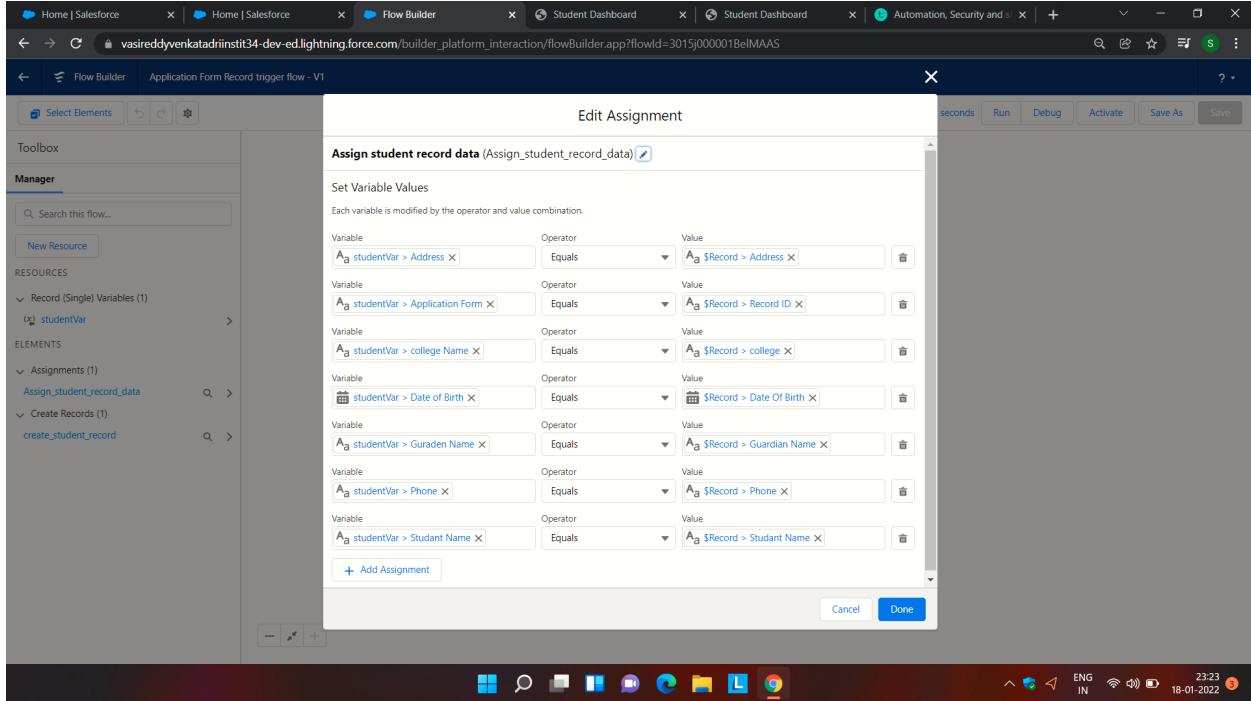
Record Type * Student

Set Field Values

Field *	Type *	Value *
Address	Field Reference	[Application_Form__c]....Q
Application Form	Field Reference	[Application_Form__c].Id Q
Date of Birth	Field Reference	[Application_Form__c]....Q
Guraden Name	Field Reference	[Application_Form__c]....Q
Phone	Field Reference	[Application_Form__c]....Q
Student Name	Field Reference	[Application_Form__c]....Q

Save Cancel Delete

Create The Student Record Using Flow



Create A Batchapex For Application Form

```
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 }
```

The screenshot shows the code for ApplicationBatchTest.apxc in the Developer Console. The code implements Database.Batchable<sobject> and Database.Stateful. It defines start(), execute(), and finish() methods to process Application_Form__c records. The execute() method increments totalForms and totalConvertedForms based on the Ready_To_Join__c field.

A Schedular Class

The screenshot shows the Salesforce Developer Console in Google Chrome. The URL is `vasireddyvenkatadrinstit34-dev-ed.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage`. The tab title is `ApplicationSchedule.apxc`. The code editor contains the following Apex class:

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

Below the code editor is a logs table with columns: User, Application, Operation, Time, Status, Read, and Size. The logs section is currently empty. At the bottom of the developer console window, there is a toolbar with icons for various tools like Logs, Tests, Checkpoints, Query Editor, View State, Progress, and Problems.

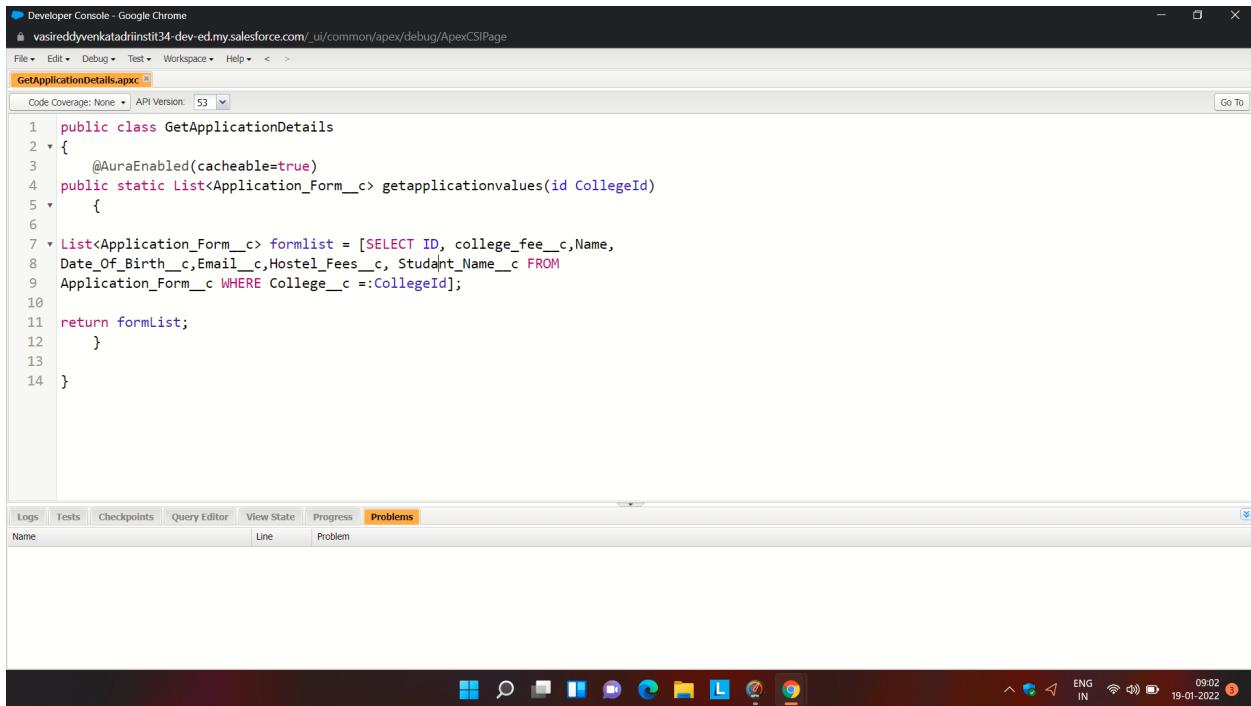
Lightning Web Components

The screenshot shows the Salesforce Lightning Web Components setup page in a browser. The URL is `https://vasireddyvenkatadrinstit34-dev-ed.lightning.force.com/lightning/setup/AsyncApexJobs/home`. The page has a sidebar on the left with sections like Apex Exception Email, Custom Code (Apex Classes, Apex Settings, Apex Test Execution, Apex Test History, Apex Triggers), Environments (Jobs, Apex Flex Queue, Apex Jobs), and a search bar. The main content area is titled "Apex Jobs" and displays a table of running jobs. The table has the following columns:

Action	Submitted Date	Job Type	Status	Status Detail	Total Batches	Batches Processed	Failures	Submitted By	Completion Date	Apex Class	Apex Method	Apex Job ID
	1/18/2022, 7:25 PM	Scheduled Apex	Queued		0	0	0	rudrapati_sreeramachandra		ApplicationSchedule		7075j0000hoTSx

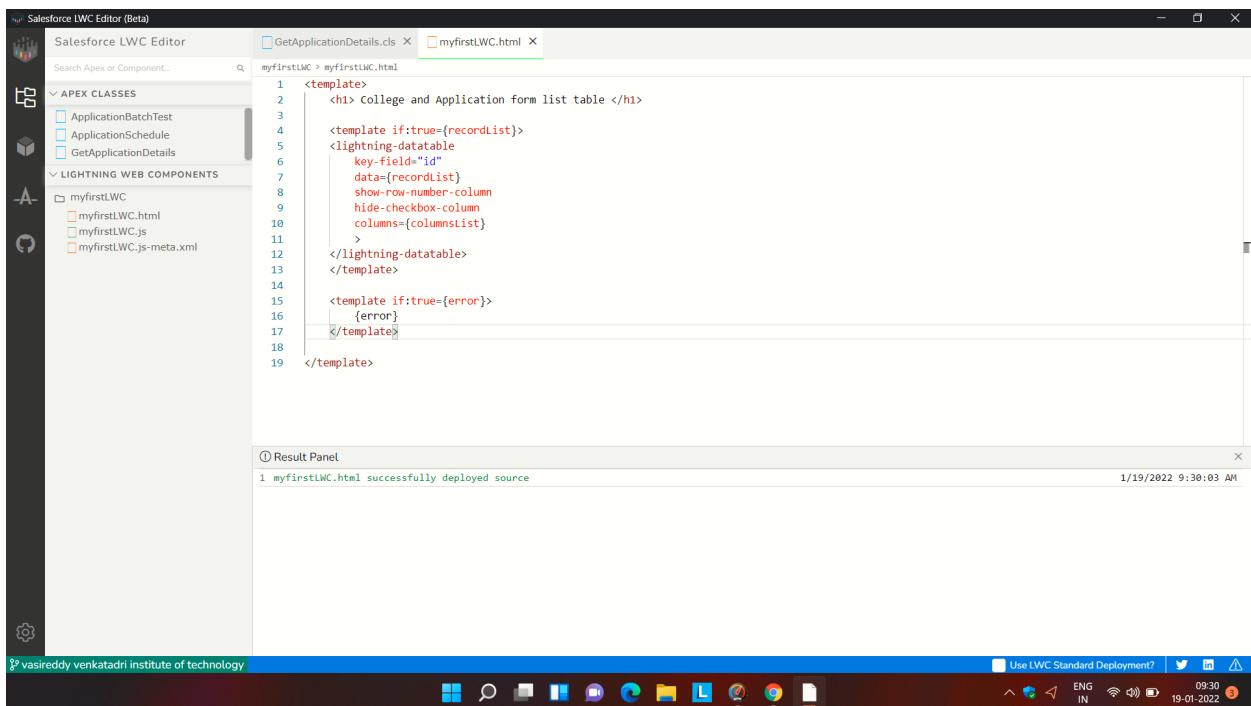
At the bottom of the page, there is a message: "Didn't find what you're looking for? Try using Global Search." The browser's address bar shows the full URL, and the system tray at the bottom right indicates the date and time as 19-01-2022, 08:55.

College DataTable Component(APEX CLASS)



```
1  public class GetApplicationDetails {
2    +
3      @AuraEnabled(cacheable=true)
4    public static List<Application_Form__c> getapplicationvalues(id CollegeId)
5    {
6
7      List<Application_Form__c> formlist = [SELECT ID, college_fee__c,Name,
8      Date_of_Birth__c,Email__c,Hostel_Fees__c, Student_Name__c FROM
9      Application_Form__c WHERE College__c =:CollegeId];
10
11    return formList;
12  }
13}
14
```

College DataTable Component (HTML FILE)



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

The Result Panel shows: myfirstLWC.html successfully deployed source

College DataTable Component(JAVA SCRIPT FILE)

The screenshot shows the Salesforce LWC Editor interface. On the left, there's a sidebar with icons for Apex and LWC components, and a search bar. The main area has tabs for GetApplicationDetails.cls, myfirstLWC.html, and myfirstLWC.js. The code editor displays the following JavaScript code:

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

Below the code editor is a Result Panel showing deployment logs:

Log	Timestamp
1 myfirstLWC.html successfully deployed source	1/19/2022 9:32:58 AM
2 myfirstLWC.js successfully deployed source	1/19/2022 9:32:58 AM

College DataTable Component(META FILE)

The screenshot shows the Salesforce LWC Editor (Beta) interface. The left sidebar displays the project structure under 'myfirstLWC': APEX CLASSES (ApplicationBatchTest, ApplicationSchedule, GetApplicationDetails) and LIGHTNING WEB COMPONENTS (myfirstLWC, myfirstLWC.html, myfirstLWC.js, myfirstLWC.js-meta.xml). The right pane shows the file 'myfirstLWC.js-meta.xml' with the following XML content:

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

The bottom panel, titled 'Result Panel', shows the deployment log:

Deployment Log	Date
1 myfirstLWC.html successfully deployed source	1/19/2022 9:34:29 AM
2 myfirstLWC.js successfully deployed source	1/19/2022 9:34:29 AM
3 myfirstLWC.js-meta.xml successfully deployed source	1/19/2022 9:34:29 AM