Student Performance Tracker

Problem Statement

In many educational institutions, maintaining comprehensive records of student performance—including academic scores, attendance, and extracurricular activities—is often done manually or across multiple disconnected systems.

This leads to delayed reporting, errors in attendance calculation, lack of real-time insights into student performance, and ineffective communication with parents.

The problem:

- Tracking student attendance and academic performance is time-consuming and error-prone.
- Parents often receive delayed information about their child's performance.
- Teachers and administrators lack a unified platform to manage students' overall progress efficiently.

Solution:

Develop a Salesforce-based Student Performance Tracker that integrates student data management, attendance calculation, academic monitoring, and parent communication into a single, easy-to-use application.

This system will automate attendance updates, calculate percentages, notify parents in real-time for failed subjects, and maintain a holistic view of student performance including extracurricular activities.

Phase 1 – Problem Understanding & Industry Analysis

1. Requirement Gathering

Objective: Understand the needs of the school administration, teachers, and parents regarding student tracking and performance management.

Key Requirements Identified:

- Maintain student records (personal, contact, class, section).
- Record and calculate attendance automatically based on daily inputs.
- Record academic scores and calculate percentages and pass/fail status.
- Notify parents when a student fails an exam.
- Track extracurricular activity participation.
- Provide a consolidated dashboard for administrators/teachers.

2. Stakeholder Analysis

Stakeholder	Role	Needs / Concerns
School Administrators	Manage student records & reports	Easy-to-access dashboard, real-time updates, accurate records
Teachers	Record attendance & grades	Efficient data entry, automatic calculations, notifications
Parents	Monitor child performance	Timely alerts, detailed exam & attendance reports
Students	View performance	Transparency and visibility of progress

3. Business Process Mapping

Current Process (Manual / Fragmented):

Attendance recorded manually → Calculations done offline → Academic results recorded separately → Parent communication via calls or emails

Proposed Salesforce Process:

- Teacher records attendance → Student object auto-updates Total_Working_days__c and Number_of_Days_Present__c.
- Teacher records academic scores → Academic object auto-calculates percentage and status.
- Record-triggered flow automatically sends email notifications to parents if percentage < 35%.
- All student data (attendance, academics, extracurricular) consolidated in a Lightning component dashboard.

4. Industry-specific Use Case Analysis

Educational Institutions: Require digital transformation for record-keeping and parent communication.

Use Case: Automate performance tracking and communication in schools to improve accuracy and efficiency.

Competitive Advantage: Real-time updates, automation of repetitive tasks, consolidated view of student data, and easy accessibility via Salesforce platform.

5. AppExchange Exploration

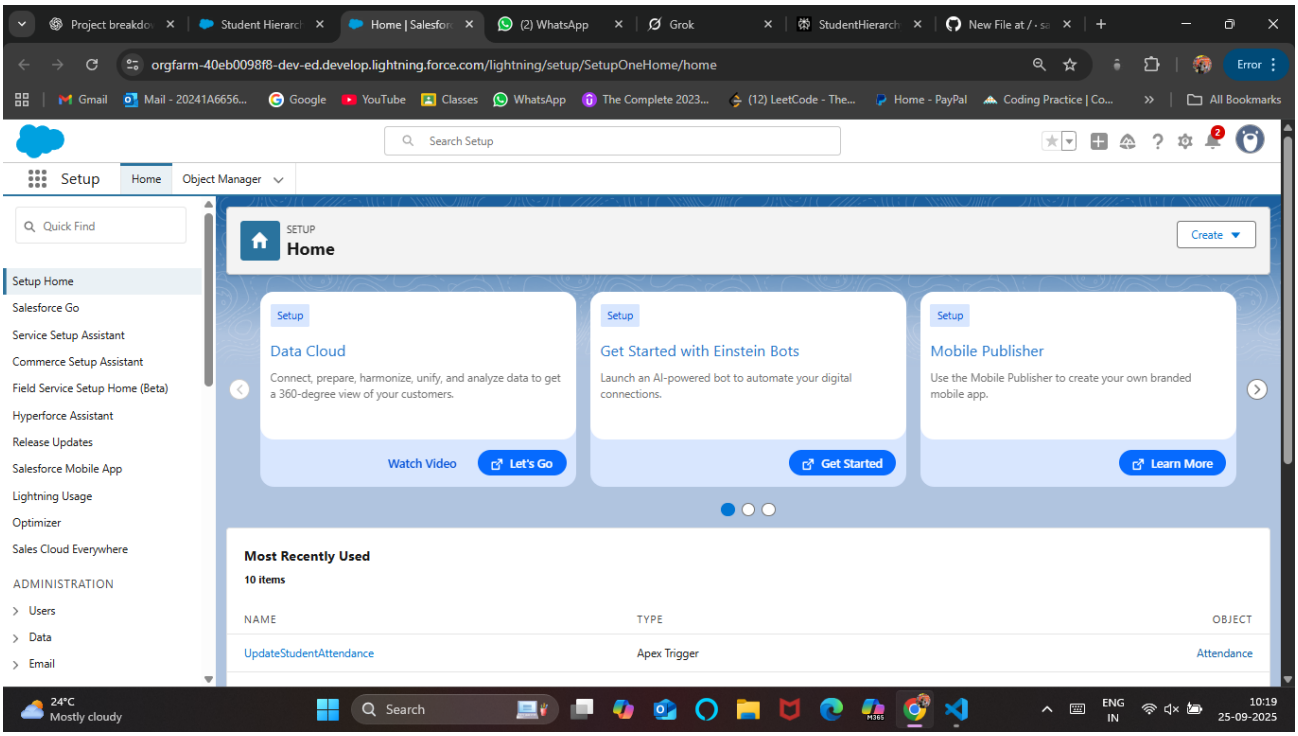
Explored existing Salesforce Education apps such as Education Data Architecture, School Management Apps, and Parent Communication Tools.

Identified gaps: Existing apps often don’t integrate attendance, academic, and extracurricular activities together with automated notifications.

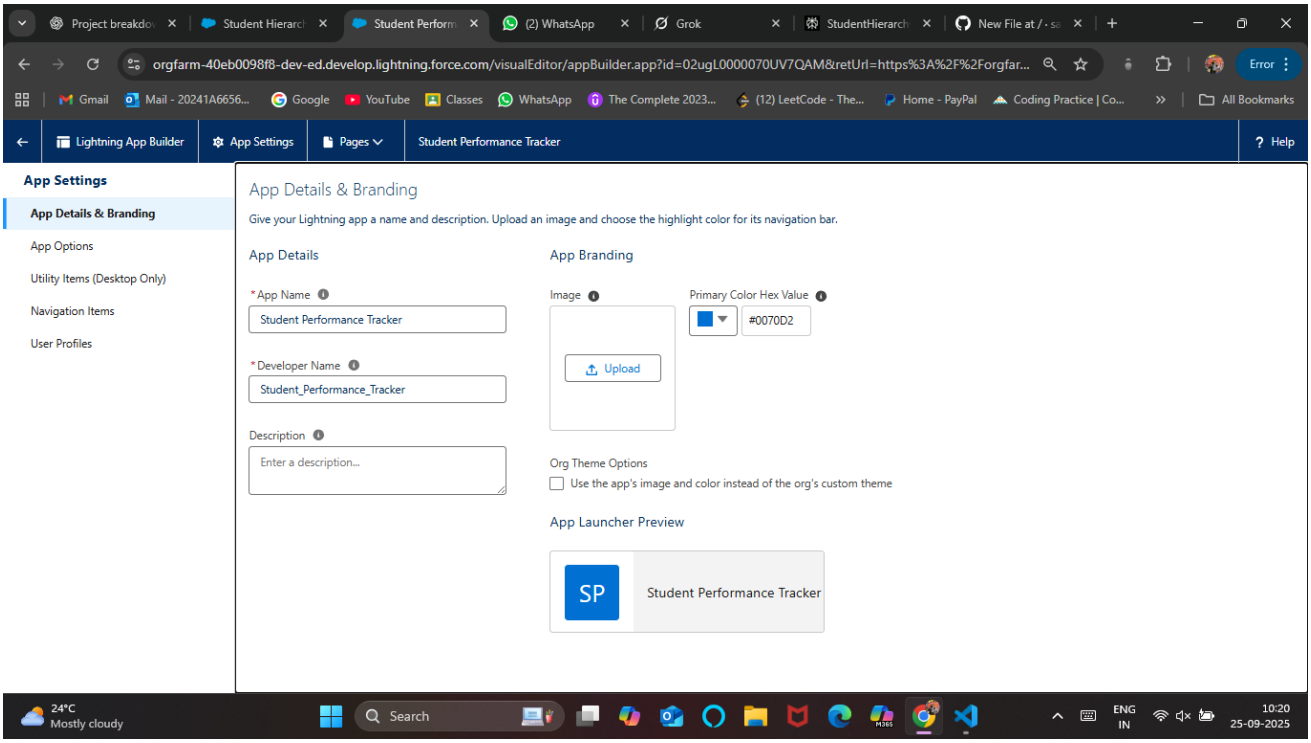
This project fills the gap with a custom-built Salesforce solution tailored for end-to-end student performance tracking.

Phase 2 – Org Setup & Configuration

- Set up a Salesforce Developer Org.
- Configured Lightning App settings, navigation items, and user profiles.
- Prepared the Salesforce environment for development and testing.



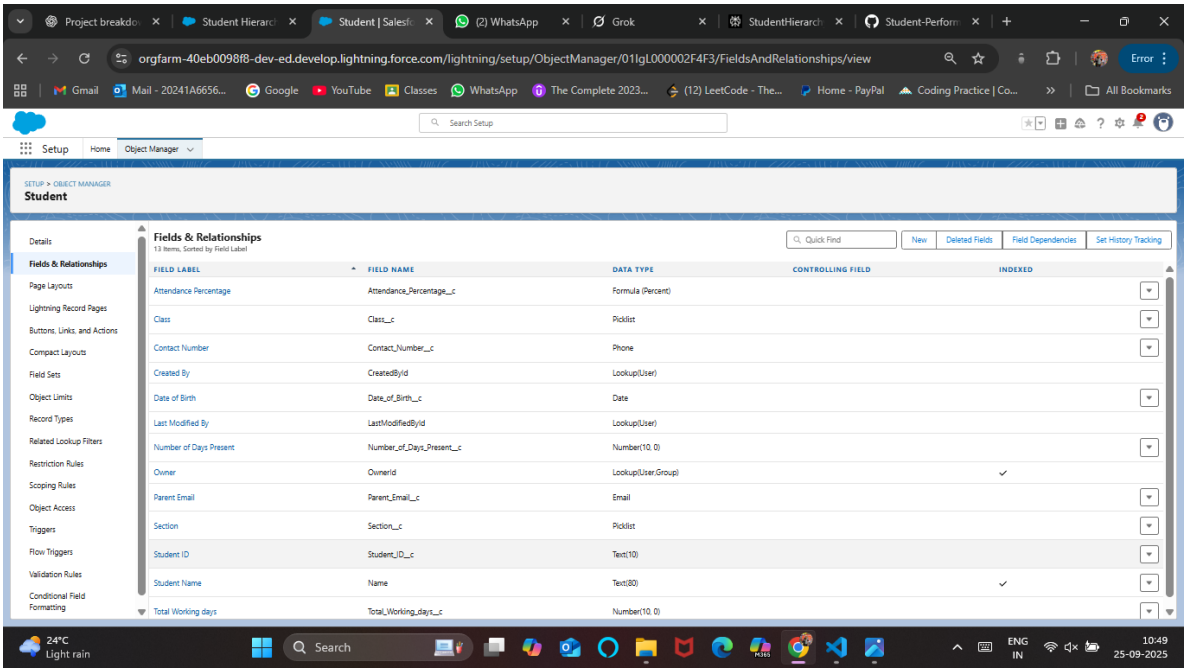
Developer Org Setup



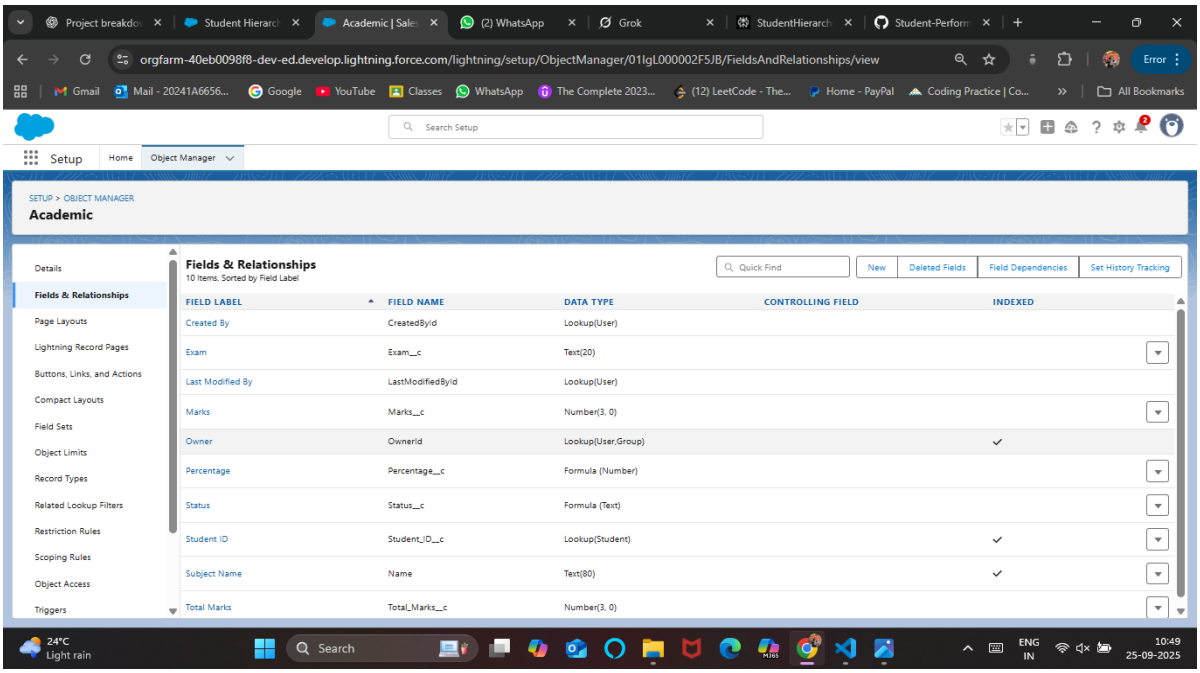
Lightning App Creation

Phase 3 – Data Modeling & Relationships

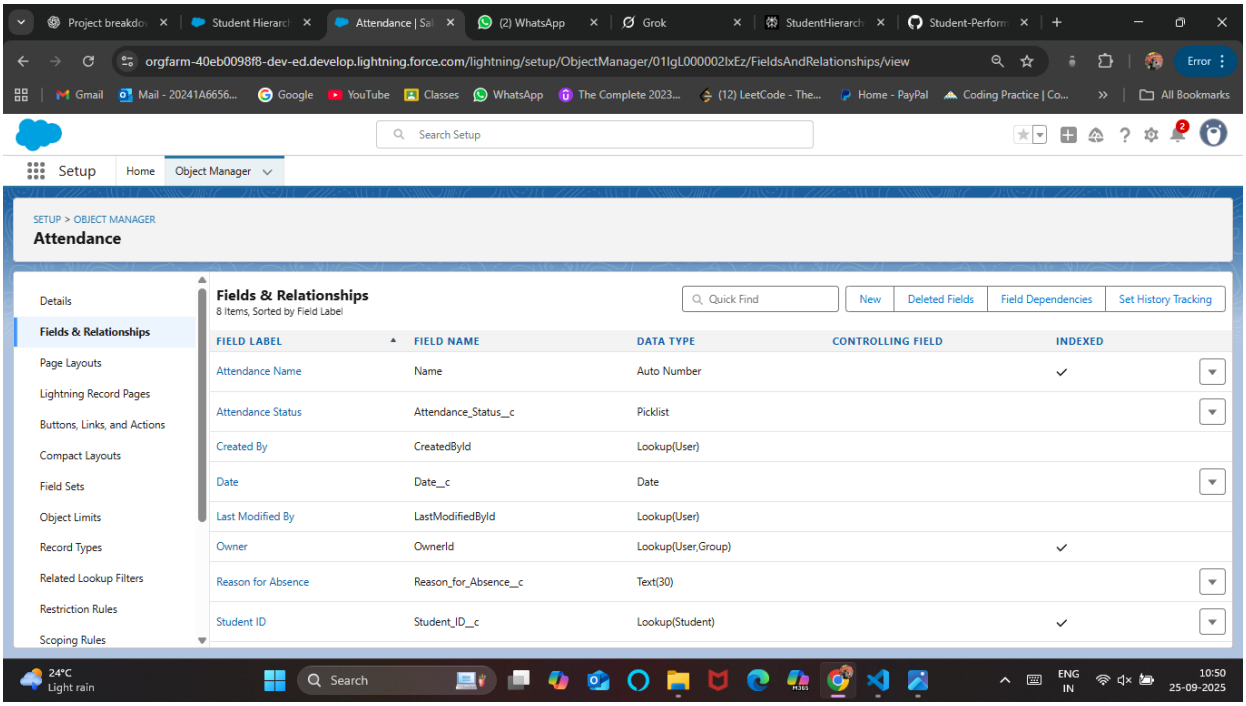
- Created four custom objects: Student, Attendance, Academic, and Extracurricular Activity.
- Defined fields and relationships, including lookup relationships between Student and related objects.
- Used Schema Builder to visualize object relationships and ensure a structured data model.



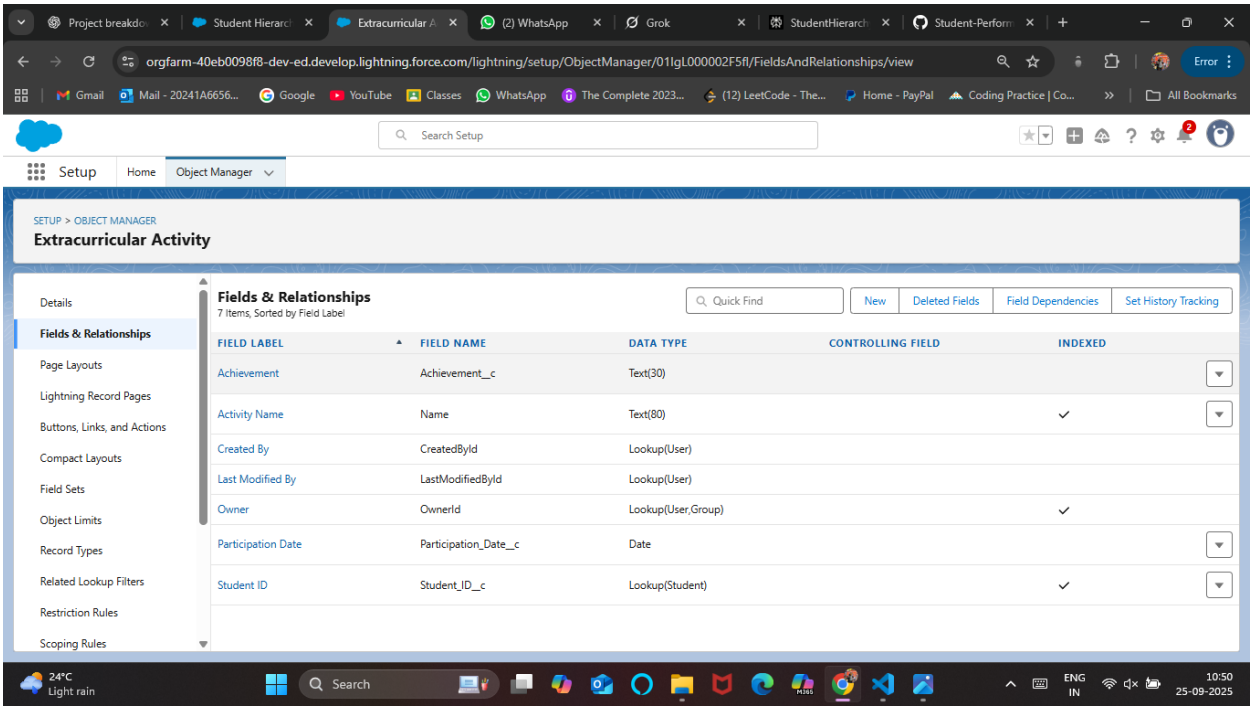
Student Object



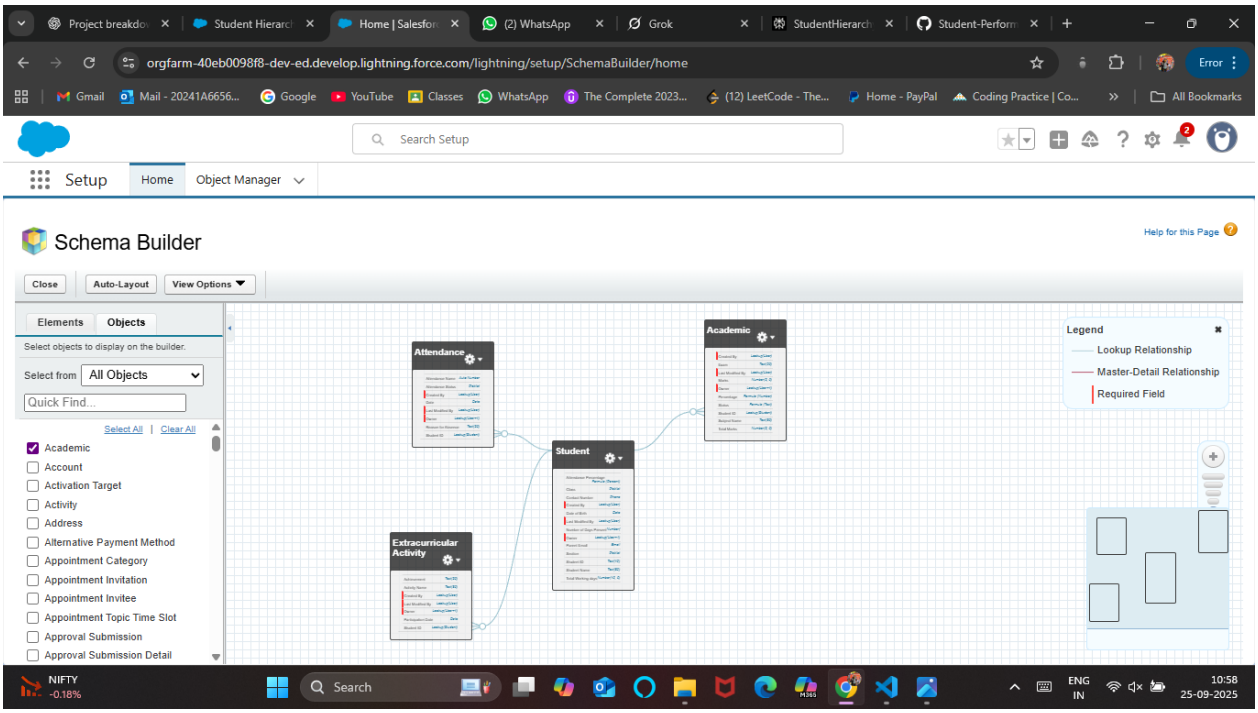
Academic Object



Attendance Object



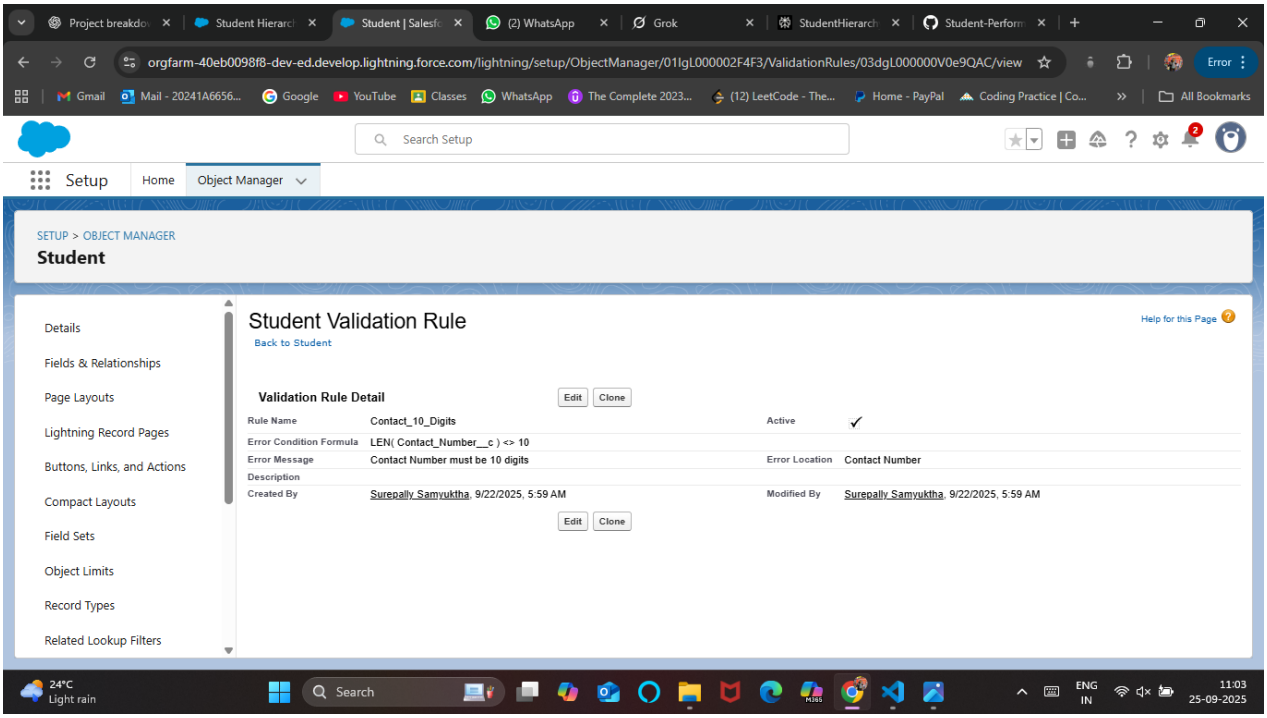
Extracurricular Activity Object



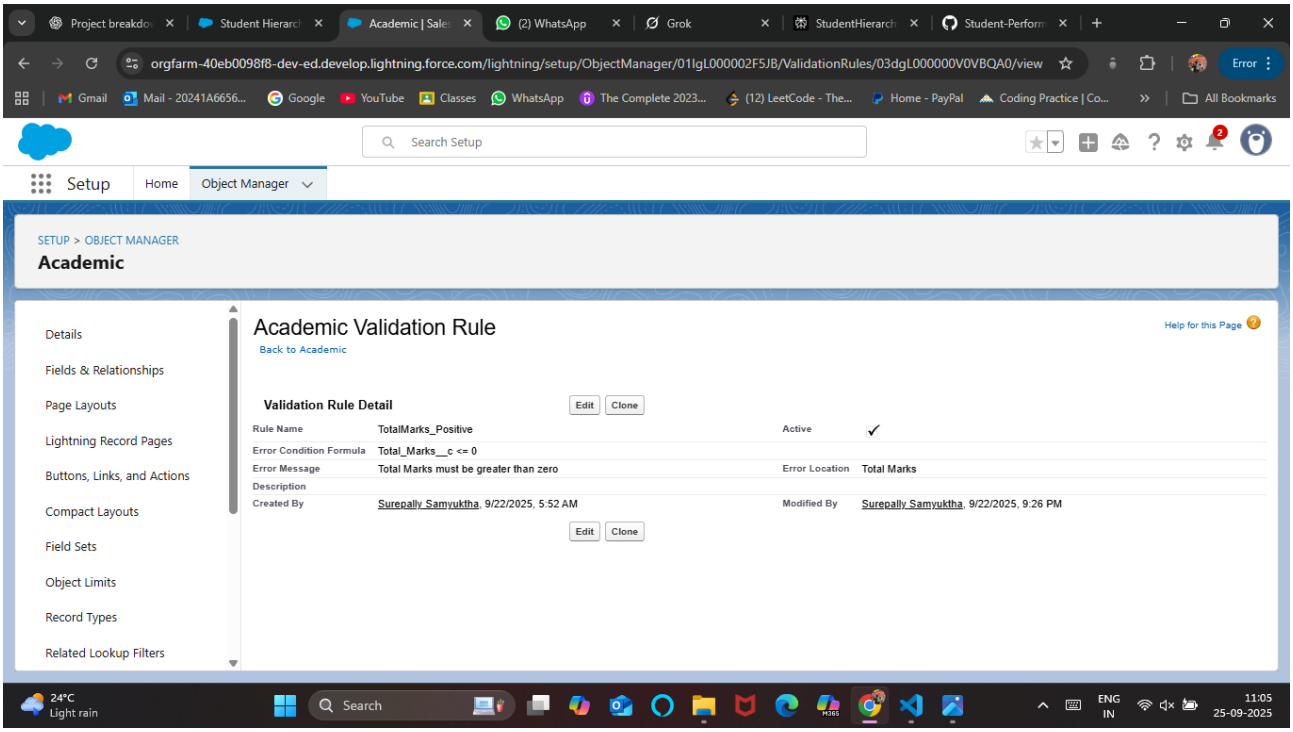
Schema Builder

Phase 4 – Process Automation (Admin)

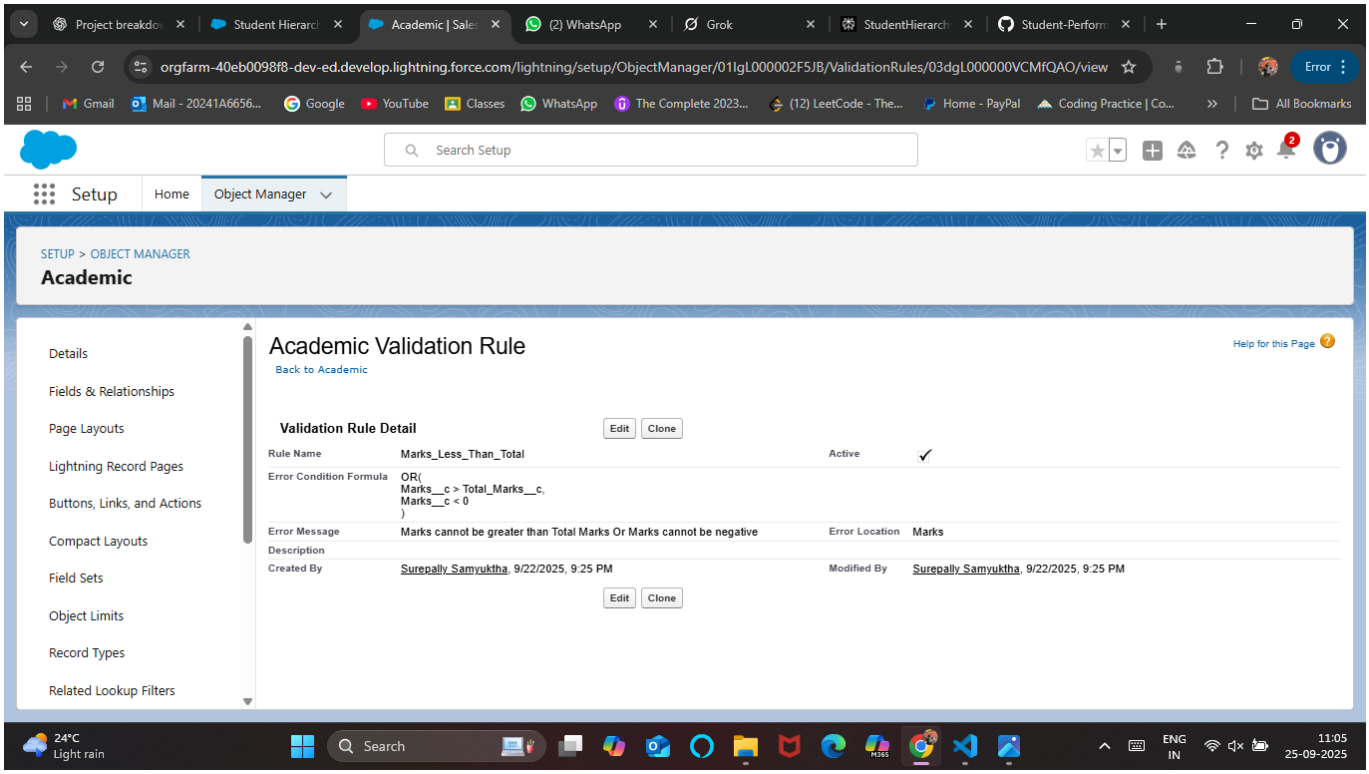
- Implemented Validation Rules to ensure data integrity across all objects.
- Built Record-Triggered Flows to automate calculations and notifications (e.g., sending alerts when academic percentage < 35%).



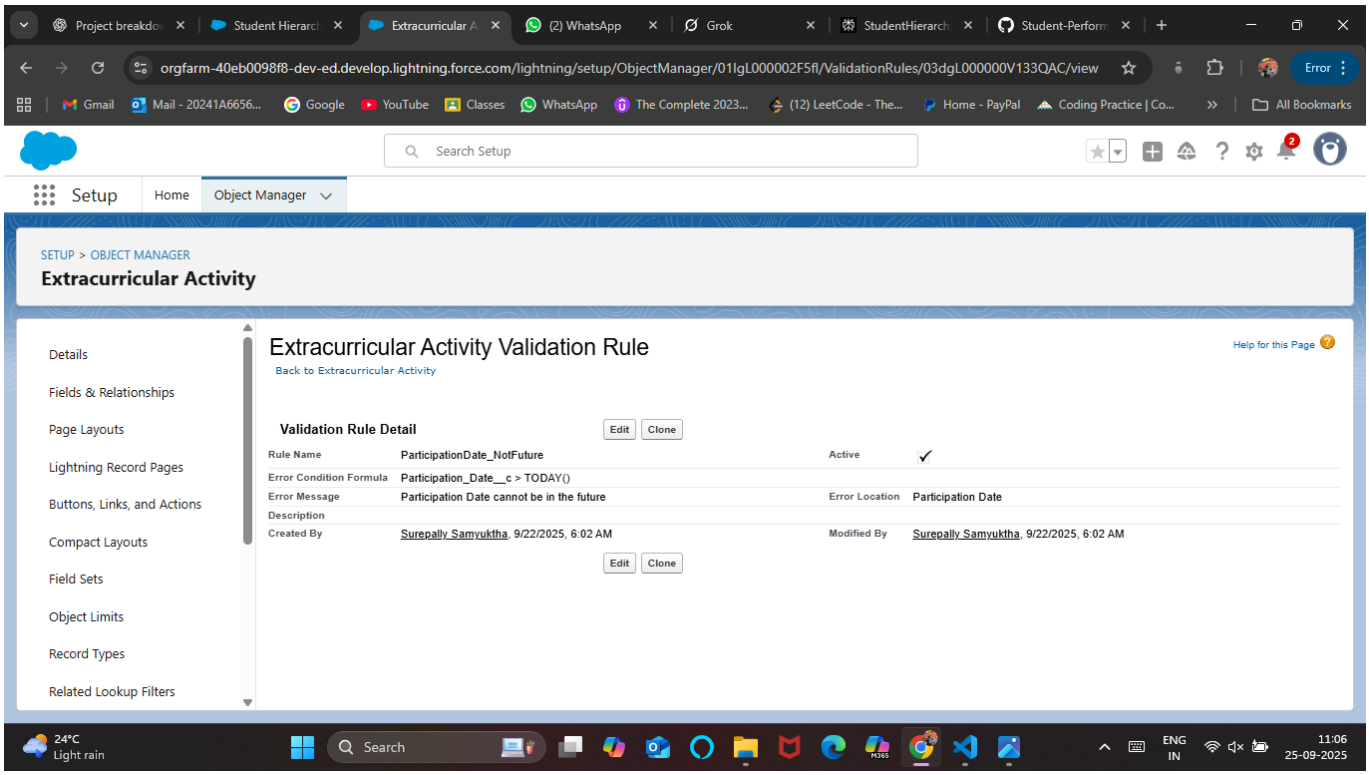
Student Validation Rule



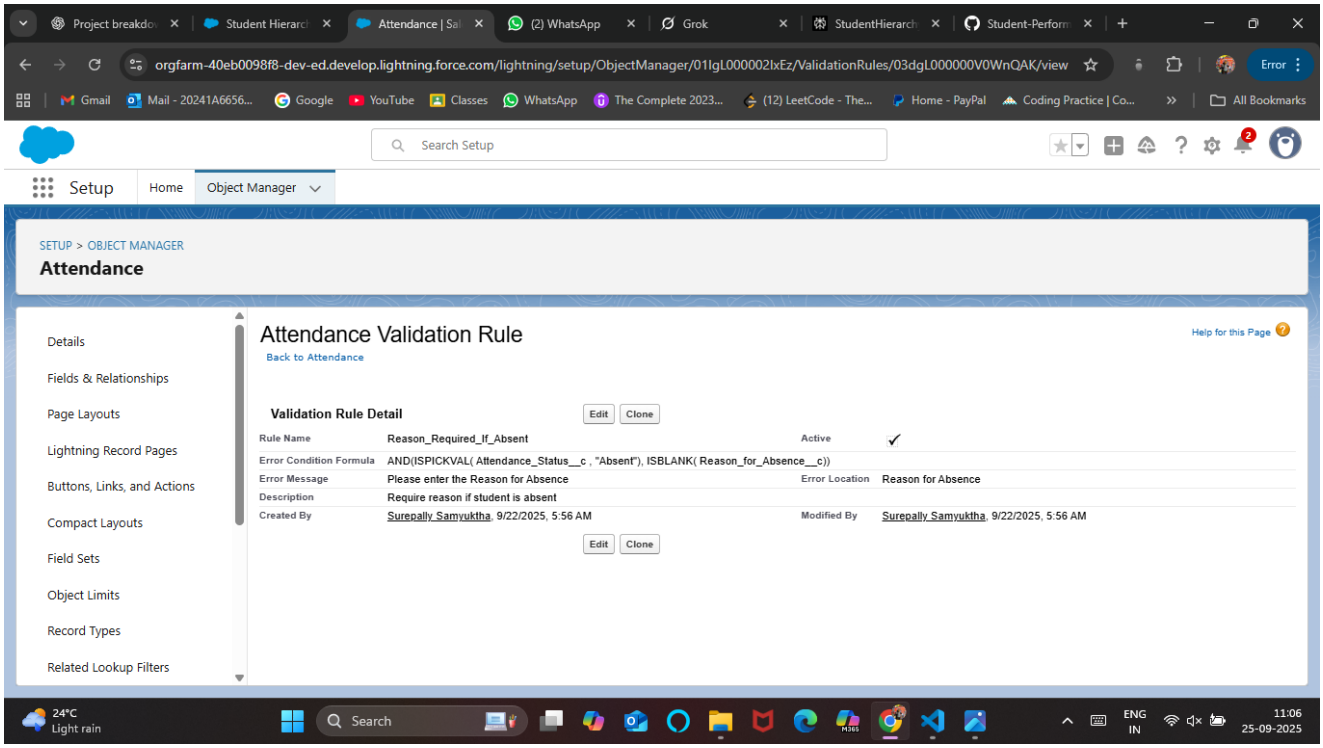
Academic Validation Rule - 1



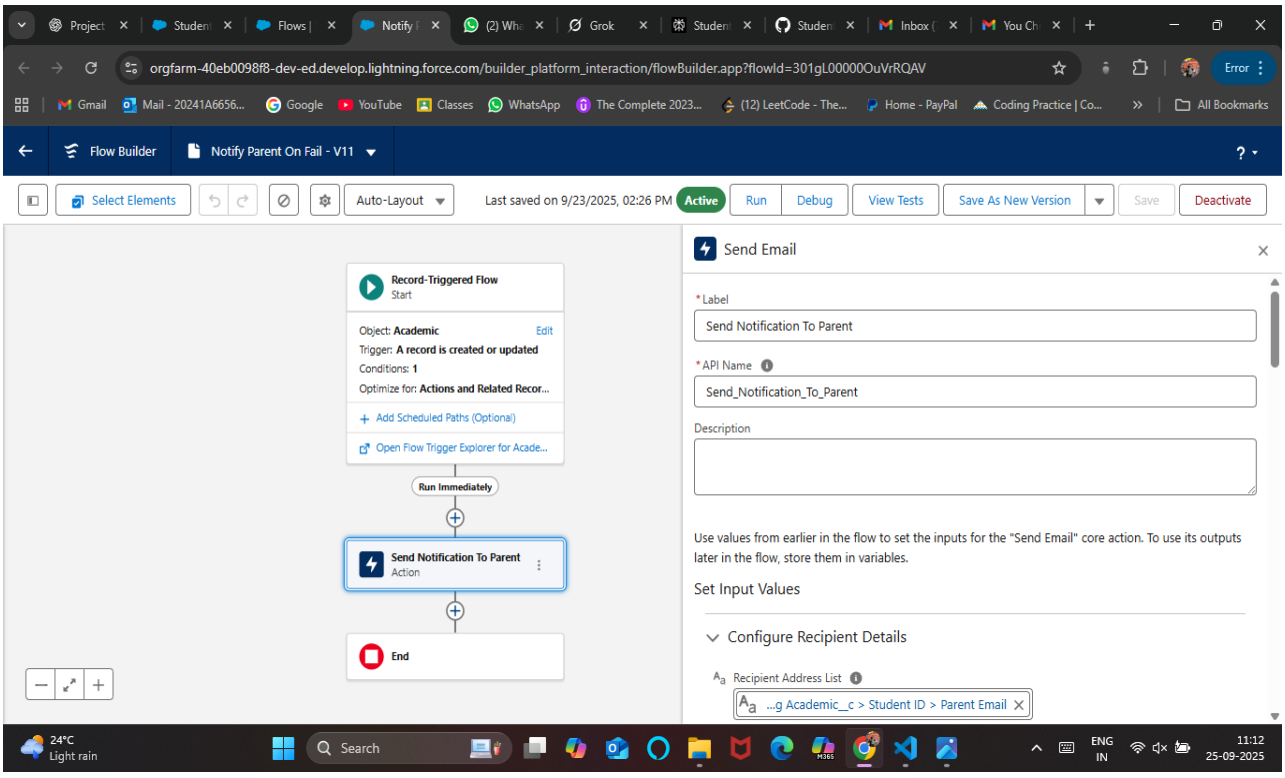
Academic Validation Rule - 2



Extracurricular Activity Validation Rule



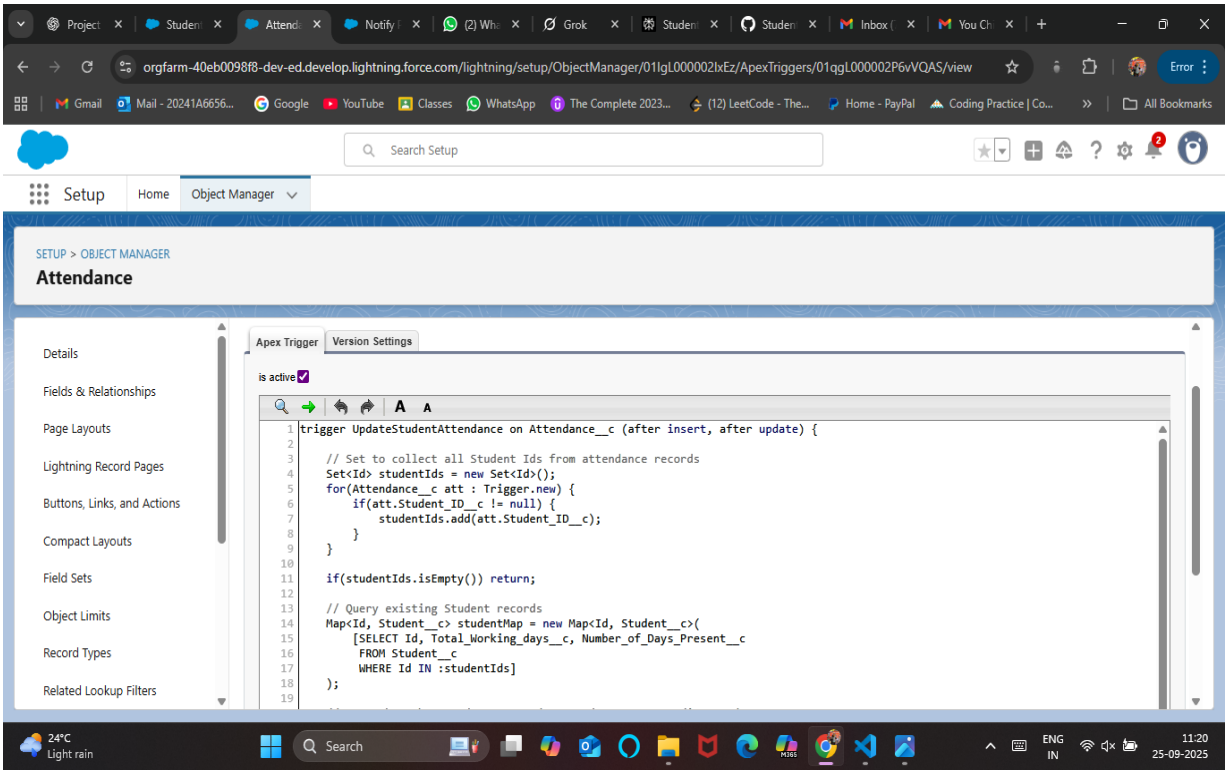
Attendance Validation Rule



Record Triggered Flow to Send Email

Phase 5 – Apex Programming (Developer)

- Developed an Apex Trigger on the Attendance object to update Student records automatically.
- Automated Total Working Days and Number of Days Present calculations.
- Improved data accuracy and reduced manual updates.



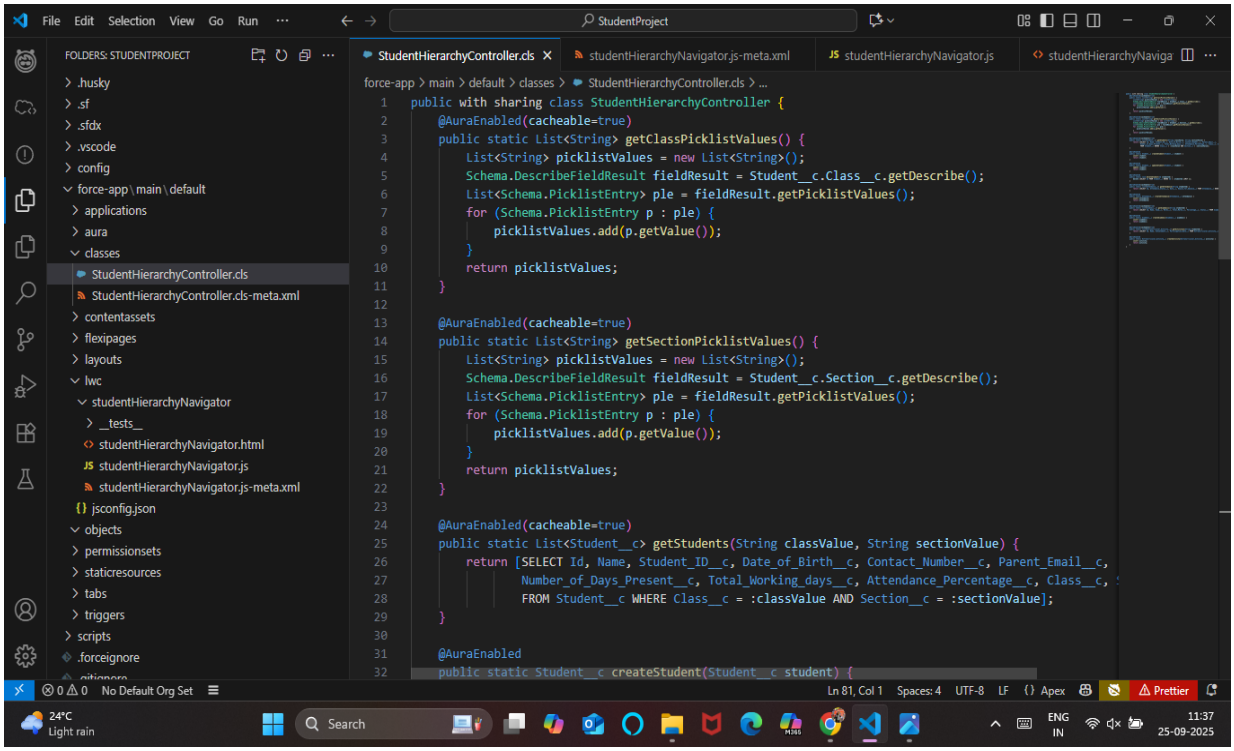
Apex Trigger Code :

```
trigger UpdateStudentAttendance on Attendance__c (after insert, after update) {
    Set<Id> studentIds = new Set<Id>();
    for(Attendance__c att : Trigger.new) {
        if(att.Student_ID__c != null) {
            studentIds.add(att.Student_ID__c);
        }
    }
    if(studentIds.isEmpty()) return;
    Map<Id, Student__c> studentMap = new Map<Id, Student__c>(
        [SELECT Id, Total_Working_days__c, Number_of_Days_Present__c
         FROM Student__c
         WHERE Id IN :studentIds]
```

```
    );
    for(Attendance__c att : Trigger.new) {
        if(att.Student_ID__c != null && studentMap.containsKey(att.Student_ID__c)) {
            Student__c stud = studentMap.get(att.Student_ID__c);
            if(stud.Total_Working_days__c == null) stud.Total_Working_days__c = 0;
            if(stud.Number_of_Days_Present__c == null) stud.Number_of_Days_Present__c = 0
            stud.Total_Working_days__c += 1;
            if(att.Attendance_Status__c == 'Present') {
                stud.Number_of_Days_Present__c += 1;
            }
        }
    }
    if(!studentMap.isEmpty()) {
        update studentMap.values();
    }
}
```

Phase 6 – User Interface Development

- Used Visual Studio Code (VS Code) for Salesforce development and deployment.
- Developed an Apex Class to fetch and process student-related data for the frontend.
- Built Lightning Web Components (LWC) with custom HTML and JavaScript to display student information interactively.
- Integrated backend logic with frontend to deliver real-time dashboards.



Lightning Web Component

StudentHierarchyController.cls

```
public with sharing class StudentHierarchyController {
    @AuraEnabled(cacheable=true)
    public static List<String> getClassPicklistValues() {
        List<String> picklistValues = new List<String>();
        Schema.DescribeFieldResult fieldResult = Student__c.Class__c.getDescribe();
        List<Schema.PicklistEntry> ple = fieldResult.getPicklistValues();
        for (Schema.PicklistEntry p : ple) {
            picklistValues.add(p.getValue());
        }
        return picklistValues;
    }
    @AuraEnabled(cacheable=true)
    public static List<String> getSectionPicklistValues() {
        List<String> picklistValues = new List<String>();
        Schema.DescribeFieldResult fieldResult = Student__c.Section__c.getDescribe();
        List<Schema.PicklistEntry> ple = fieldResult.getPicklistValues();
```

```
        for (Schema.PicklistEntry p : ple) {
            picklistValues.add(p.getValue());
        }
        return picklistValues;
    }
    @AuraEnabled(cacheable=true)
    public static List<Student__c> getStudents(String classValue, String sectionValue) {
        return [SELECT Id, Name, Student_ID__c, Date_of_Birth__c, Contact_Number__c, Parent_Email__c,
                Number_of_Days_Present__c, Total_Working_days__c, Attendance_Percentage__c, Class__c,
Section__c
                FROM Student__c WHERE Class__c = :classValue AND Section__c = :sectionValue];
    }
    @AuraEnabled
    public static Student__c createStudent(Student__c student) {
        insert student;
        return student;
    }
    @AuraEnabled
    public static Student__c updateStudent(Student__c student) {
        update student;
        return student;
    }
    @AuraEnabled
    public static void deleteStudent(Id studentId) {
        delete [SELECT Id FROM Student__c WHERE Id = :studentId LIMIT 1];
    }
    @AuraEnabled(cacheable=true)
    public static List<Attendance__c> getAttendance(String studentId) {
        return [SELECT Id, Attendance_Status__c, Date__c, Reason_for_Absence__c FROM Attendance__c
WHERE Student_ID__c = :studentId];
    }
    @AuraEnabled
```

```
    public static Attendance__c createAttendance(Attendance__c attendance) {
        insert attendance;
        return attendance;
    }
    @AuraEnabled(cacheable=true)
    public static List<Academic__c> getAcademics(String studentId) {
        return [SELECT Id, Name, Exam__c, Marks__c, Total_Marks__c, Percentage__c, Status__c FROM
Academic__c WHERE Student_ID__c = :studentId];
    }
    @AuraEnabled
    public static Academic__c createAcademic(Academic__c academic) {
        insert academic;
        return academic;
    }
    @AuraEnabled(cacheable=true)
    public static List<ExtraCurricular_Activity__c> getActivities(String studentId) {
        return [SELECT Id, Name, Achievement__c, Participation_Date__c FROM ExtraCurricular_Activity__c
WHERE Student_ID__c = :studentId];
    }
    @AuraEnabled
    public static ExtraCurricular_Activity__c createActivity(ExtraCurricular_Activity__c activity) {
        insert activity;
        return activity;
    }
}
```

StudentHierarchyController.cls-meta.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<ApexClass xmlns="http://soap.sforce.com/2006/04/metadata">
    <apiVersion>60.0</apiVersion>
    <status>Active</status>
</ApexClass>
```

studentHierarchyNavigator.html

```
<template>
    <lightning-card title="Student Hierarchy Navigator">
        <template if:true={showClasses}>
            <lightning-combobox
                name="class"
                label="Select Class"
                value={selectedClass}
                placeholder="Select a Class"
                options={classOptions}
                onchange={handleClassChange}>
            </lightning-combobox>
        </template>
        <template if:true={showSections}>
            <lightning-combobox
                name="section"
                label="Select Section"
                value={selectedSection}
                placeholder="Select a Section"
                options={sectionOptions}
                onchange={handleSectionChange}>
            </lightning-combobox>
        </template>
        <template if:true={showStudents}>
            <lightning-button label="Add Student" onclick={handleAddStudent}></lightning-button>
            <lightning-button label="Refresh Students" onclick={fetchStudents} class="slds-m-
left_small"></lightning-button>
            <lightning-datatable key-field="Id" data={students} columns={studentColumns}
onrowaction={handleStudentSelect} hide-checkbox-column>
            </lightning-datatable>
        </template>
        <template if:true={showStudentDetails}>
            <lightning-tabset variant="scoped">
                <lightning-tab label="Attendance" value="attendance">
```

```
            <lightning-button label="Add Attendance" onclick={handleAddAttendance}></lightning-button>
            <lightning-button label="Refresh Attendance" onclick={fetchAttendance} class="slds-m-left_small"></lightning-button>
            <lightning-datatable key-field="Id" data={attendanceRecords} columns={attendanceColumns} hide-checkbox-column></lightning-datatable>
        </lightning-tab>
        <lightning-tab label="Academics" value="academics">
            <lightning-button label="Add Academic" onclick={handleAddAcademic}></lightning-button>
            <lightning-button label="Refresh Academics" onclick={fetchAcademics} class="slds-m-left_small"></lightning-button>
            <lightning-datatable key-field="Id" data={academicRecords} columns={academicColumns} hide-checkbox-column></lightning-datatable>
        </lightning-tab>
        <lightning-tab label="Activities" value="activities">
            <lightning-button label="Add Activity" onclick={handleAddActivity}></lightning-button>
            <lightning-button label="Refresh Activities" onclick={fetchActivities} class="slds-m-left_small"></lightning-button>
            <lightning-datatable key-field="Id" data={activityRecords} columns={activityColumns} hide-checkbox-column></lightning-datatable>
        </lightning-tab>
    </lightning-tabset>
</template>
<!-- Modal for adding student -->
<template if:true={showAddStudentModal}>
    <section role="dialog" tabindex="-1" aria-modal="true" class="slds-modal slds-fade-in-open">
        <div class="slds-modal__container">
            <header class="slds-modal__header">
                <h2 class="slds-modal__title">Add Student</h2>
            </header>
            <div class="slds-modal__content slds-p-around_medium">
                <lightning-record-edit-form object-api-name="Student__c" onsuccess={handleStudentSave}>
                    <lightning-messages></lightning-messages>
                    <lightning-input-field field-name="Name"></lightning-input-field>
```

```
                    <lightning-input-field field-name="Student_ID__c"></lightning-input-field>
                    <lightning-input-field field-name="Date_of_Birth__c"></lightning-input-field>
                    <lightning-input-field field-name="Contact_Number__c"></lightning-input-field>
                    <lightning-input-field field-name="Parent_Email__c"></lightning-input-field>
                    <lightning-input-field field-name="Number_of_Days_Present__c"></lightning-input-field>
                    <lightning-input-field field-name="Total_Working_days__c"></lightning-input-field>
                    <!-- Class and Section are pre-selected -->
                    <lightning-input type="hidden" name="Class__c" value={selectedClass}></lightning-input>
                    <lightning-input type="hidden" name="Section__c" value={selectedSection}></lightning-input>
                    <lightning-button type="submit" label="Save"></lightning-button>
                </lightning-record-edit-form>
            </div>
            <footer class="slds-modal__footer">
                <lightning-button label="Cancel" onclick={closeModal}></lightning-button>
            </footer>
        </div>
    </section>
    <div class="slds-backdrop slds-backdrop_open"></div>
</template>
<!-- Modal for editing student -->
<template if:true={showEditStudentModal}>
    <section role="dialog" tabindex="-1" aria-modal="true" class="slds-modal slds-fade-in-open">
        <div class="slds-modal__container">
            <header class="slds-modal__header">
                <h2 class="slds-modal__title">Edit Student</h2>
            </header>
            <div class="slds-modal__content slds-p-around_medium">
                <lightning-record-edit-form object-api-name="Student__c" record-id={editStudentData.Id} onsuccess={handleEditStudentSave}>
                    <lightning-messages></lightning-messages>
                    <lightning-input-field field-name="Name"></lightning-input-field>
```

```
                    <lightning-input-field field-name="Student_ID__c"></lightning-input-field>
                    <lightning-input-field field-name="Date_of_Birth__c"></lightning-input-field>
                    <lightning-input-field field-name="Contact_Number__c"></lightning-input-field>
                    <lightning-input-field field-name="Parent_Email__c"></lightning-input-field>
                    <lightning-input-field field-name="Number_of_Days_Present__c"></lightning-input-
field>
                    <lightning-input-field field-name="Total_Working_days__c"></lightning-input-field>
                    <lightning-input-field field-name="Class__c"></lightning-input-field>
                    <lightning-input-field field-name="Section__c"></lightning-input-field>
                    <lightning-button type="submit" label="Save"></lightning-button>
                </lightning-record-edit-form>
            </div>
            <footer class="slds-modal__footer">
                <lightning-button label="Cancel" onclick={closeModal}></lightning-button>
            </footer>
        </div>
    </section>
    <div class="slds-backdrop slds-backdrop_open"></div>
</template>
<!-- Modal for adding attendance -->
<template if:true={showAddAttendanceModal}>
    <section role="dialog" tabindex="-1" aria-modal="true" class="slds-modal slds-fade-in-open">
        <div class="slds-modal__container">
            <header class="slds-modal__header">
                <h2 class="slds-modal__title">Add Attendance</h2>
            </header>
            <div class="slds-modal__content slds-p-around_medium">
                <lightning-record-edit-form object-api-name="Attendance__c"
onsuccess={handleAttendanceSave}>
                    <lightning-messages></lightning-messages>
                    <lightning-input-field field-name="Attendance_Status__c"></lightning-input-field>
                    <lightning-input-field field-name="Date__c"></lightning-input-field>
                    <lightning-input-field field-name="Reason_for_Absence__c"></lightning-input-field>
```

```
                    <lightning-input-field field-name="Student_ID__c"
value={selectedStudentId}></lightning-input-field>
                    <lightning-button type="submit" label="Save"></lightning-button>
                </lightning-record-edit-form>
            </div>
            <footer class="slds-modal__footer">
                <lightning-button label="Cancel" onclick={closeModal}></lightning-button>
            </footer>
        </div>
    </section>
    <div class="slds-backdrop slds-backdrop_open"></div>
</template>
<!-- Modal for adding academic -->
<template if:true={showAddAcademicModal}>
    <section role="dialog" tabindex="-1" aria-modal="true" class="slds-modal slds-fade-in-open">
        <div class="slds-modal__container">
            <header class="slds-modal__header">
                <h2 class="slds-modal__title">Add Academic</h2>
            </header>
            <div class="slds-modal__content slds-p-around_medium">
                <lightning-record-edit-form object-api-name="Academic__c"
onsuccess={handleAcademicSave}>
                    <lightning-messages></lightning-messages>
                    <lightning-input-field field-name="Name"></lightning-input-field>
                    <lightning-input-field field-name="Exam__c"></lightning-input-field>
                    <lightning-input-field field-name="Marks__c"></lightning-input-field>
                    <lightning-input-field field-name="Total_Marks__c"></lightning-input-field>
                    <lightning-input-field field-name="Student_ID__c"
value={selectedStudentId}></lightning-input-field>
                    <lightning-button type="submit" label="Save"></lightning-button>
                </lightning-record-edit-form>
            </div>
            <footer class="slds-modal__footer">
                <lightning-button label="Cancel" onclick={closeModal}></lightning-button>
```

```
            </footer>
          </div>
        </section>
        <div class="slds-backdrop slds-backdrop_open"></div>
      </template>
      <!-- Modal for adding activity -->
      <template if:true={showAddActivityModal}>
        <section role="dialog" tabindex="-1" aria-modal="true" class="slds-modal slds-fade-in-open">
          <div class="slds-modal__container">
            <header class="slds-modal__header">
              <h2 class="slds-modal__title">Add Activity</h2>
            </header>
            <div class="slds-modal__content slds-p-around_medium">
              <lightning-record-edit-form object-api-name="ExtraCurricular_Activity__c"
onsuccess={handleActivitySave}>
                <lightning-messages></lightning-messages>
                <lightning-input-field field-name="Name"></lightning-input-field>
                <lightning-input-field field-name="Achievement__c"></lightning-input-field>
                <lightning-input-field field-name="Participation_Date__c"></lightning-input-field>
                <lightning-input-field field-name="Student_ID__c"
value={selectedStudentId}></lightning-input-field>
                <lightning-button type="submit" label="Save"></lightning-button>
              </lightning-record-edit-form>
            </div>
            <footer class="slds-modal__footer">
              <lightning-button label="Cancel" onclick={closeModal}></lightning-button>
            </footer>
          </div>
        </section>
        <div class="slds-backdrop slds-backdrop_open"></div>
      </template>
    </lightning-card>
</template>
```

studentHierarchyNavigator.js

```
import { LightningElement, wire, track } from 'lwc';
import getClassPicklistValues from
'@salesforce/apex/StudentHierarchyController.getClassPicklistValues';
import getSectionPicklistValues from
'@salesforce/apex/StudentHierarchyController.getSectionPicklistValues';
import getStudents from '@salesforce/apex/StudentHierarchyController.getStudents';
import createStudent from '@salesforce/apex/StudentHierarchyController.createStudent';
import updateStudent from '@salesforce/apex/StudentHierarchyController.updateStudent';
import deleteStudent from '@salesforce/apex/StudentHierarchyController.deleteStudent';
import getAttendance from '@salesforce/apex/StudentHierarchyController.getAttendance';
import getAcademics from '@salesforce/apex/StudentHierarchyController.getAcademics';
import getActivities from '@salesforce/apex/StudentHierarchyController.getActivities';
import { ShowToastEvent } from 'lightning/platformShowToastEvent';
export default class StudentHierarchyNavigator extends LightningElement {
    @track showClasses = true;
    @track showSections = false;
    @track showStudents = false;
    @track showStudentDetails = false;
    @track selectedClass;
    @track selectedSection;
    @track selectedStudentId;
    @track classOptions = [];
    @track sectionOptions = [];
    @track students = [];
    @track attendanceRecords = [];
    @track academicRecords = [];
    @track activityRecords = [];
    @track showAddStudentModal = false;
    @track showEditStudentModal = false;
    @track editStudentData = {};
    @track showAddAttendanceModal = false;
    @track showAddAcademicModal = false;
    @track showAddActivityModal = false;
```

```
studentColumns = [
    { label: 'Student Name', fieldName: 'Name' },
    { label: 'Student ID', fieldName: 'Student_ID__c' },
    { label: 'Date of Birth', fieldName: 'Date_of_Birth__c', type: 'date' },
    { label: 'Contact Number', fieldName: 'Contact_Number__c', type: 'phone' },
    { label: 'Parent Email', fieldName: 'Parent_Email__c', type: 'email' },
    { label: 'Days Present', fieldName: 'Number_of_Days_Present__c', type: 'number' },
    { label: 'Total Working Days', fieldName: 'Total_Working_days__c', type: 'number' },
    { label: 'Attendance Percentage', fieldName: 'Attendance_Percentage__c', type: 'percent' },
    {
        type: 'action',
        typeAttributes: {
            rowActions: [
                { label: 'View Details', name: 'view' },
                { label: 'Edit', name: 'edit' },
                { label: 'Delete', name: 'delete' }
            ]
        }
    }
];

attendanceColumns = [
    { label: 'Attendance Status', fieldName: 'Attendance_Status__c' },
    { label: 'Date', fieldName: 'Date__c', type: 'date' },
    { label: 'Reason for Absence', fieldName: 'Reason_for_Absence__c' }
];
academicColumns = [
    { label: 'Subject Name', fieldName: 'Name' },
    { label: 'Exam', fieldName: 'Exam__c' },
    { label: 'Marks', fieldName: 'Marks__c', type: 'number' },
    { label: 'Total Marks', fieldName: 'Total_Marks__c', type: 'number' },
    { label: 'Percentage', fieldName: 'Percentage__c', type: 'number' },
    { label: 'Status', fieldName: 'Status__c' }
```

```
];
activityColumns = [
    { label: 'Activity Name', fieldName: 'Name' },
    { label: 'Achievement', fieldName: 'Achievement__c' },
    { label: 'Participation Date', fieldName: 'Participation_Date__c', type: 'date' }
];
@wire(getClassPicklistValues)
wiredClasses({ error, data }) {
    if (data) {
        this.classOptions = data.map(value => ({ label: value, value: value }));
    } else if (error) {
        this.showToast('Error', error.body.message, 'error');
    }
}
@wire(getSectionPicklistValues)
wiredSections({ error, data }) {
    if (data) {
        this.sectionOptions = data.map(value => ({ label: value, value: value }));
    } else if (error) {
        this.showToast('Error', error.body.message, 'error');
    }
}
handleClassChange(event) {
    this.selectedClass = event.detail.value;
    this.showSections = true;
    this.showStudents = false;
    this.showStudentDetails = false;
}
handleSectionChange(event) {
    this.selectedSection = event.detail.value;
    this.showStudents = true;
    this.showStudentDetails = false;
    this.fetchStudents();
```

```
}
fetchStudents() {
    getStudents({ classValue: this.selectedClass, sectionValue: this.selectedSection })
        .then(result => {
            this.students = result;
        })
        .catch(error => {
            this.showToast('Error', error.body.message, 'error');
        });
}
handleStudentSelect(event) {
    const actionName = event.detail.action.name;
    const row = event.detail.row;
    if (actionName === 'view') {
        this.selectedStudentId = row.Id;
        this.showStudentDetails = true;
        this.fetchStudentDetails();
    } else if (actionName === 'edit') {
        this.editStudentData = { ...row };
        this.showEditStudentModal = true;
    } else if (actionName === 'delete') {
        this.deleteStudent(row.Id);
    }
}
fetchStudentDetails() {
    this.fetchAttendance();
    this.fetchAcademics();
    this.fetchActivities();
}
fetchAttendance() {
    getAttendance({ studentId: this.selectedStudentId })
        .then(result => {
            this.attendanceRecords = result;
```

```
        })
        .catch(error => {
            this.showToast('Error', error.body.message, 'error');
        });
    }
    fetchAcademics() {
        getAcademics({ studentId: this.selectedStudentId })
        .then(result => {
            this.academicRecords = result;
        })
        .catch(error => {
            this.showToast('Error', error.body.message, 'error');
        });
    }
    fetchActivities() {
        getActivities({ studentId: this.selectedStudentId })
        .then(result => {
            this.activityRecords = result;
        })
        .catch(error => {
            this.showToast('Error', error.body.message, 'error');
        });
    }
    handleAddStudent() {
        this.showAddStudentModal = true;
    }
    handleAddAttendance() {
        this.showAddAttendanceModal = true;
    }
    handleAddAcademic() {
        this.showAddAcademicModal = true;
    }
    handleAddActivity() {
```

```
    this.showAddActivityModal = true;
}
handleStudentSave(event) {
    this.showAddStudentModal = false;
    this.showToast('Success', 'Student added successfully', 'success');
    this.fetchStudents();
}
handleEditStudentSave(event) {
    const updatedStudent = { ...event.detail.fields, sobjectType: 'Student__c' };
    updateStudent({ student: updatedStudent })
        .then(() => {
            this.showToast('Success', 'Student updated successfully', 'success');
            this.fetchStudents();
            this.showEditStudentModal = false;
        })
        .catch(error => {
            this.showToast('Error', error.body.message, 'error');
        });
}
deleteStudent(studentId) {
    deleteStudent({ studentId: studentId })
        .then(() => {
            this.showToast('Success', 'Student deleted successfully', 'success');
            this.fetchStudents();
            if (this.selectedStudentId === studentId) {
                this.showStudentDetails = false;
            }
        })
        .catch(error => {
            this.showToast('Error', error.body.message, 'error');
        });
}
handleAttendanceSave(event) {
```

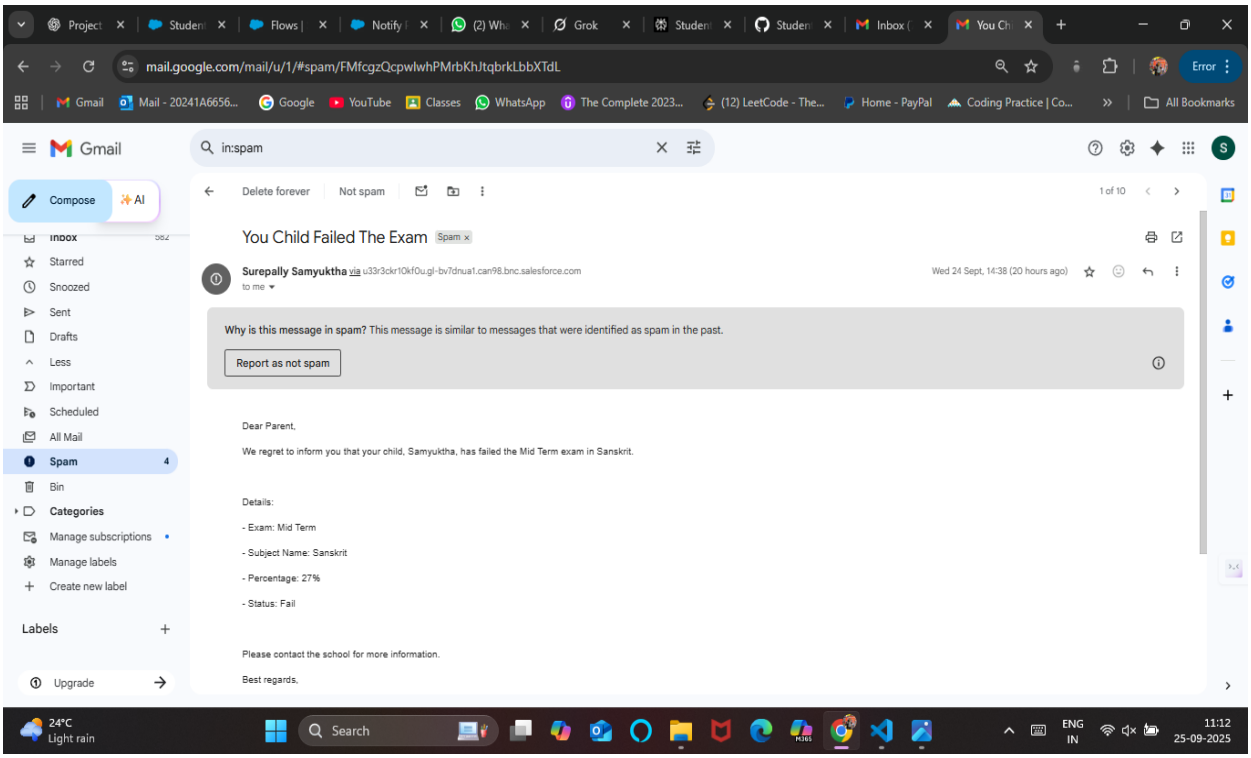
```
        this.showAddAttendanceModal = false;
        this.showToast('Success', 'Attendance added successfully', 'success');
        this.fetchAttendance();
    }
    handleAcademicSave(event) {
        this.showAddAcademicModal = false;
        this.showToast('Success', 'Academic added successfully', 'success');
        this.fetchAcademics();
    }
    handleActivitySave(event) {
        this.showAddActivityModal = false;
        this.showToast('Success', 'Activity added successfully', 'success');
        this.fetchActivities();
    }
    closeModal() {
        this.showAddStudentModal = false;
        this.showEditStudentModal = false;
        this.showAddAttendanceModal = false;
        this.showAddAcademicModal = false;
        this.showAddActivityModal = false;
    }
    showToast(title, message, variant) {
        const event = new ShowToastEvent({
            title: title,
            message: message,
            variant: variant
        });
        this.dispatchEvent(event);
    }
}
```

studentHierarchyNavigator.js-meta.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
    <apiVersion>60.0</apiVersion>
    <isExposed>true</isExposed>
    <targets>
        <target>lightning__AppPage</target>
        <target>lightning__RecordPage</target>
        <target>lightning__HomePage</target>
    </targets>
</LightningComponentBundle>
```

Phase 7 – Integration & External Access

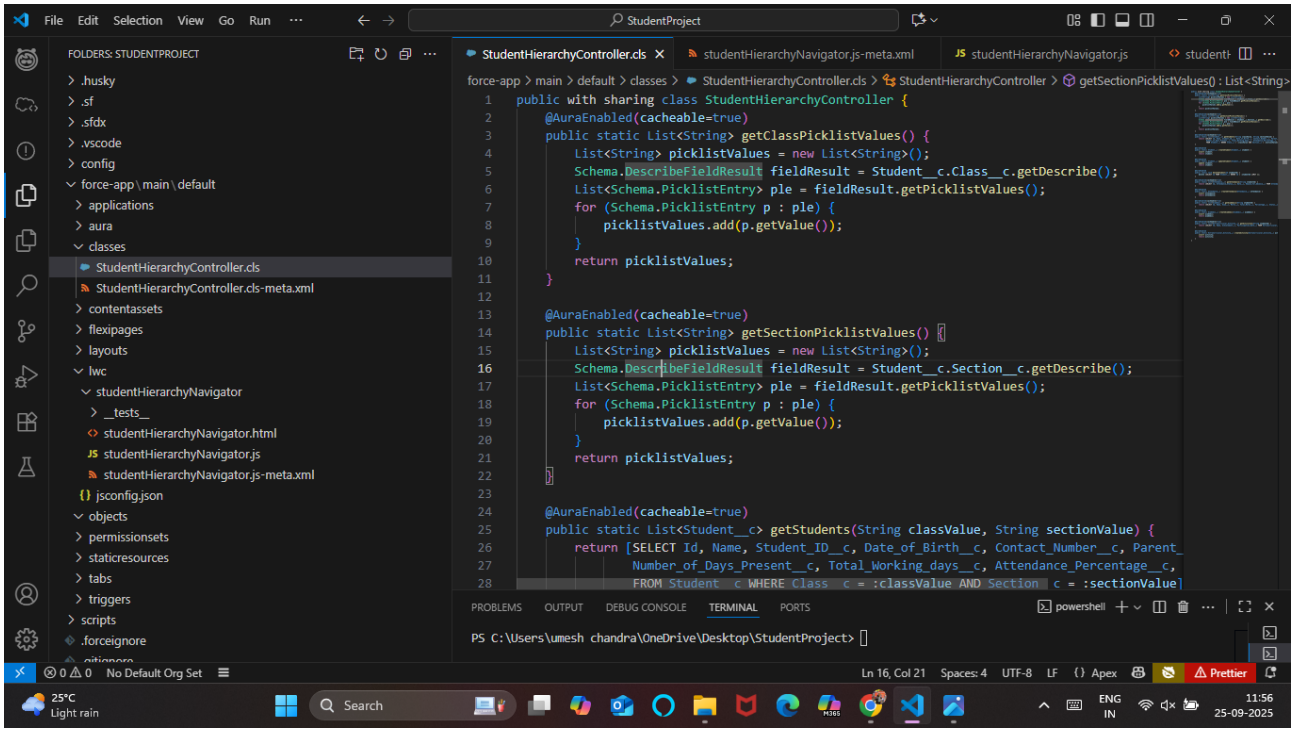
- Created a Record-Triggered Flow to send emails to parents when a student's exam percentage falls below 35%.
- Configured the flow to fetch Parent Email and include exam details in the email content.
- Verified functionality with screenshots of received emails.



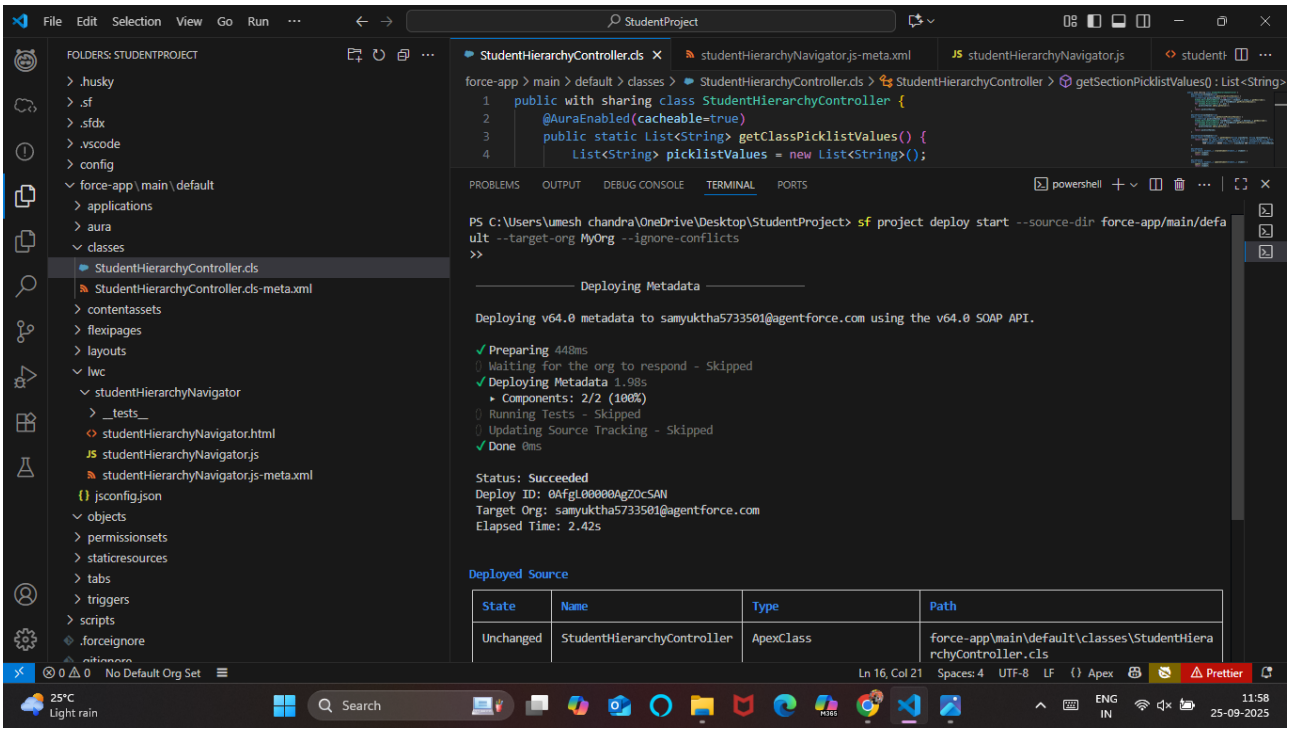
Email Received to Parent

Phase 8 – Data Management & Deployment

- Showcased the Salesforce DX (SFDX) project structure in VS Code.
- Deployed Apex classes and LWCs to the Salesforce Developer Org using SFDX commands.
- Ensured all custom logic and UI components were successfully integrated.



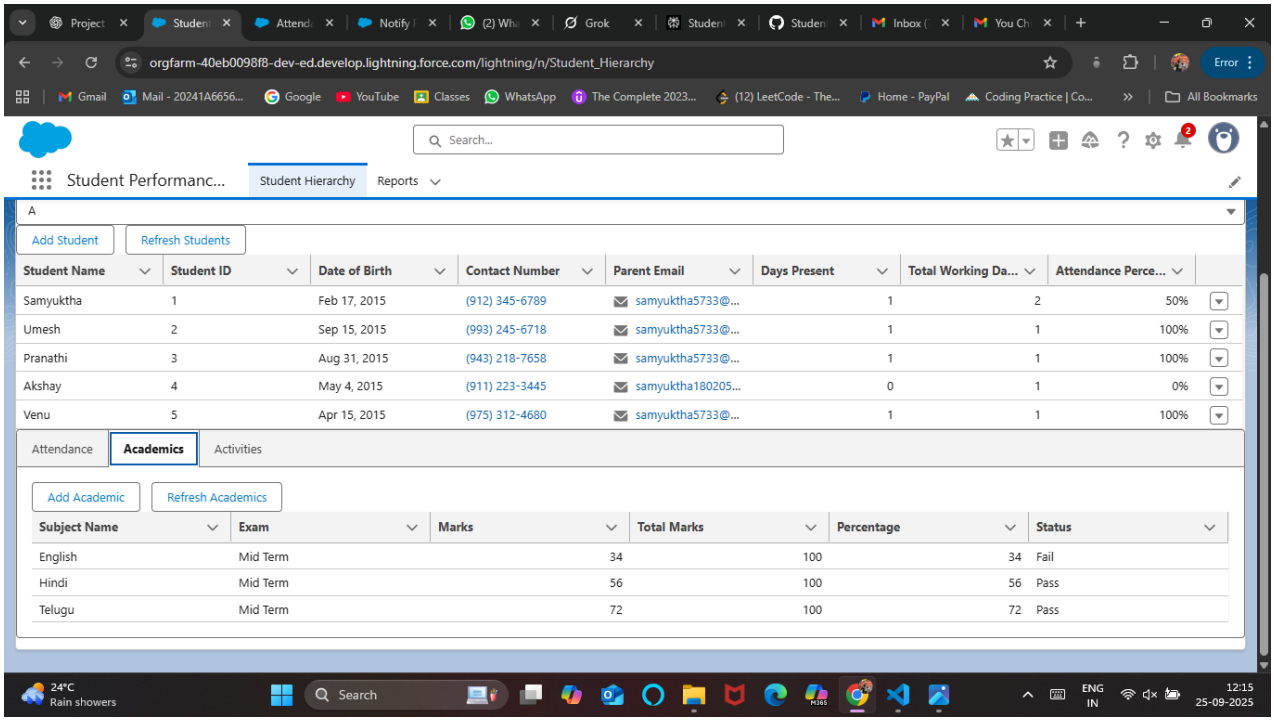
VSCode SFDX Project Structure



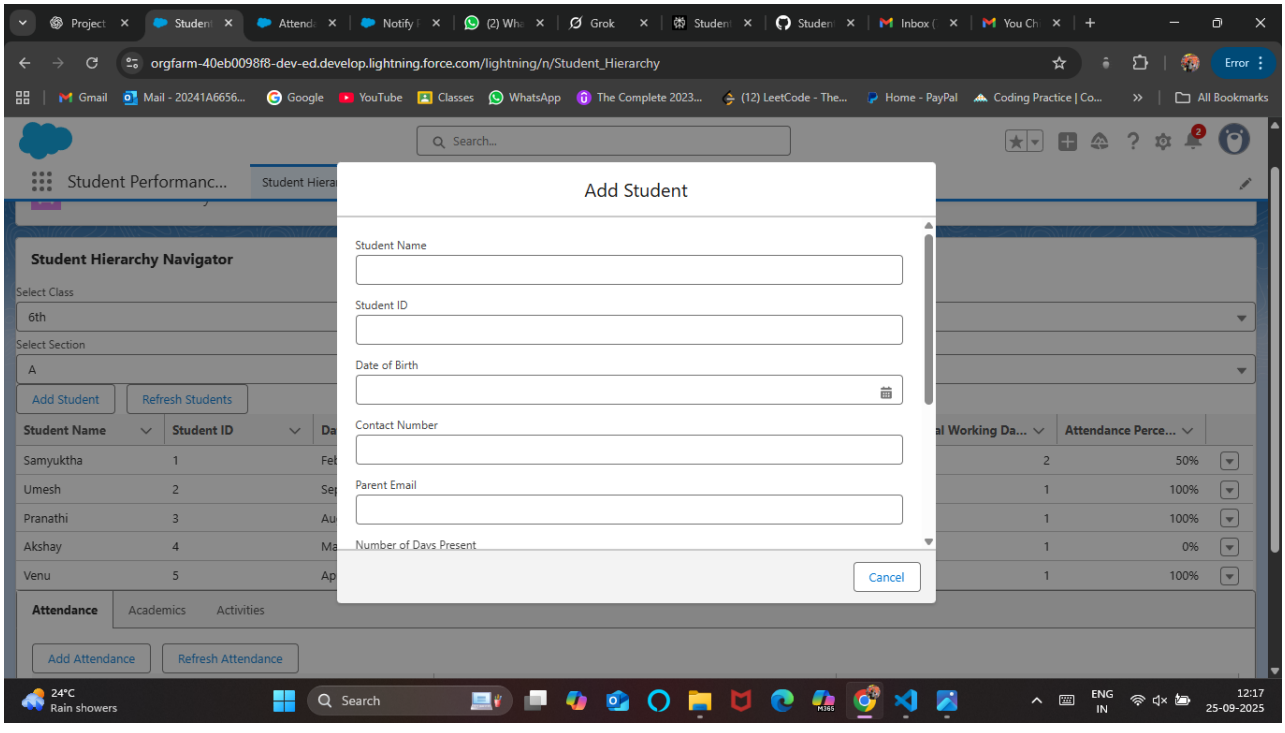
Apex Class & LWC Deployment

Phase 9 & 10 – Reporting, Dashboards & Security Review

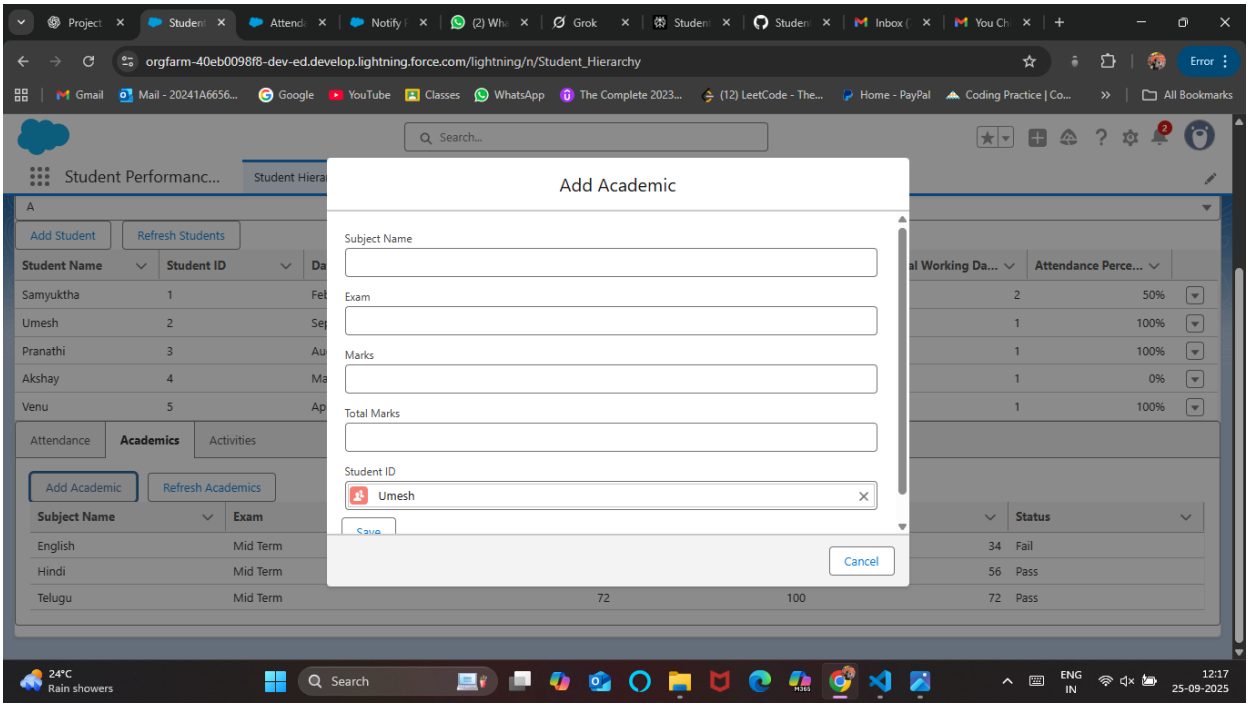
- Displayed the final Lightning App interface with a complete student management dashboard.
- Showcased student academic, attendance, and activity details within the app.
- Provided forms for adding new records: Student, Attendance, Academic, and Activity.
- Highlighted the interactive, user-friendly interface and seamless navigation across sections.



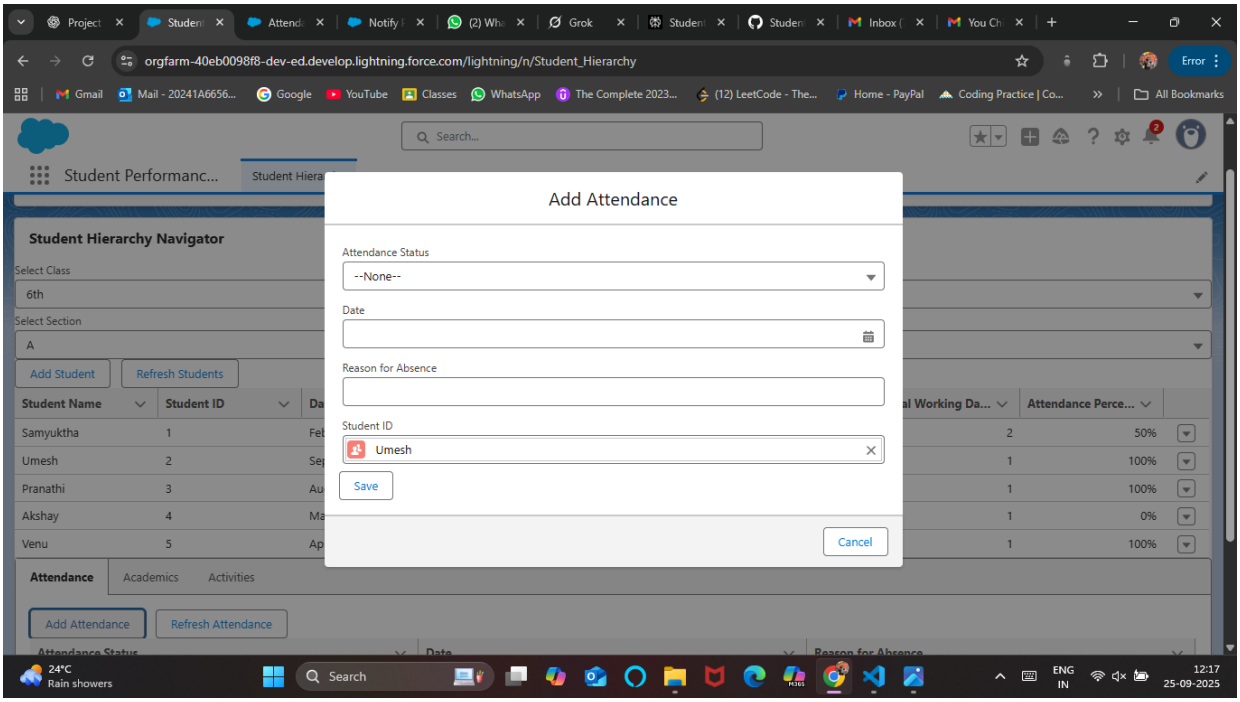
App Outlook



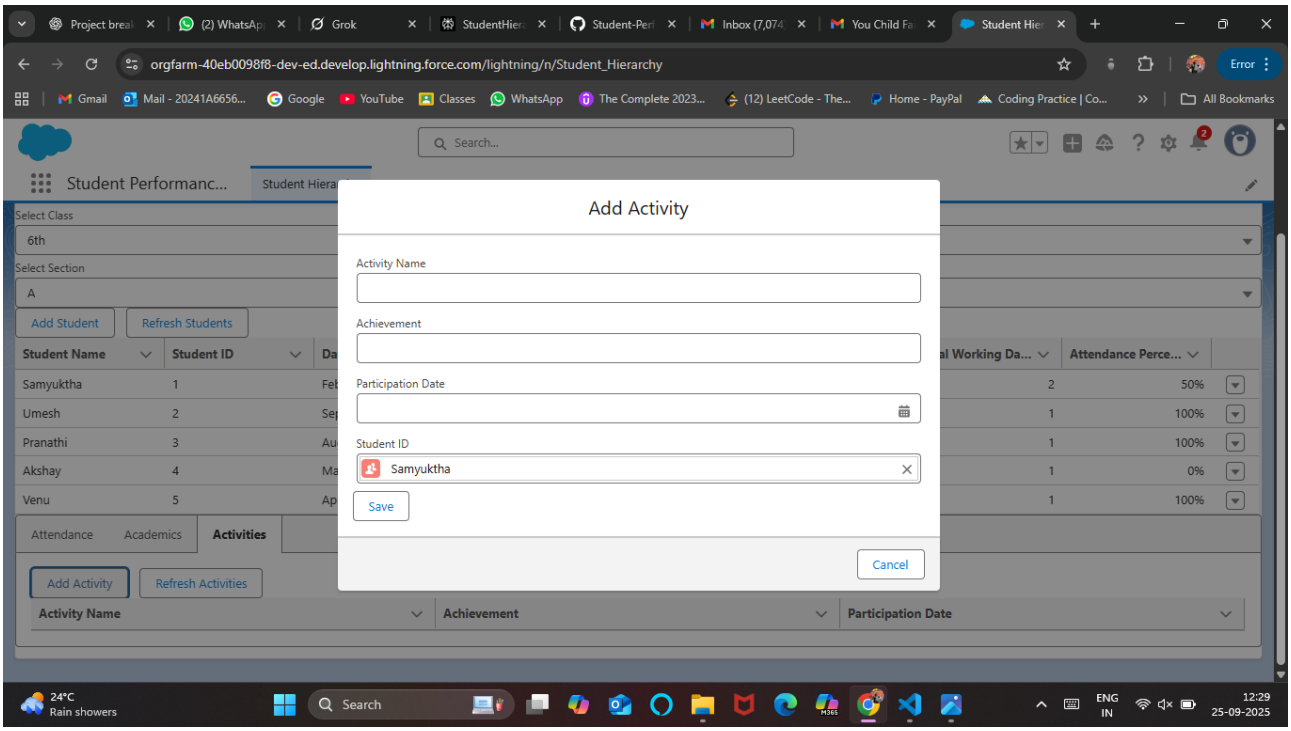
Add Student Form



Add Academic Form



Add Attendance Form



Add Activity Form