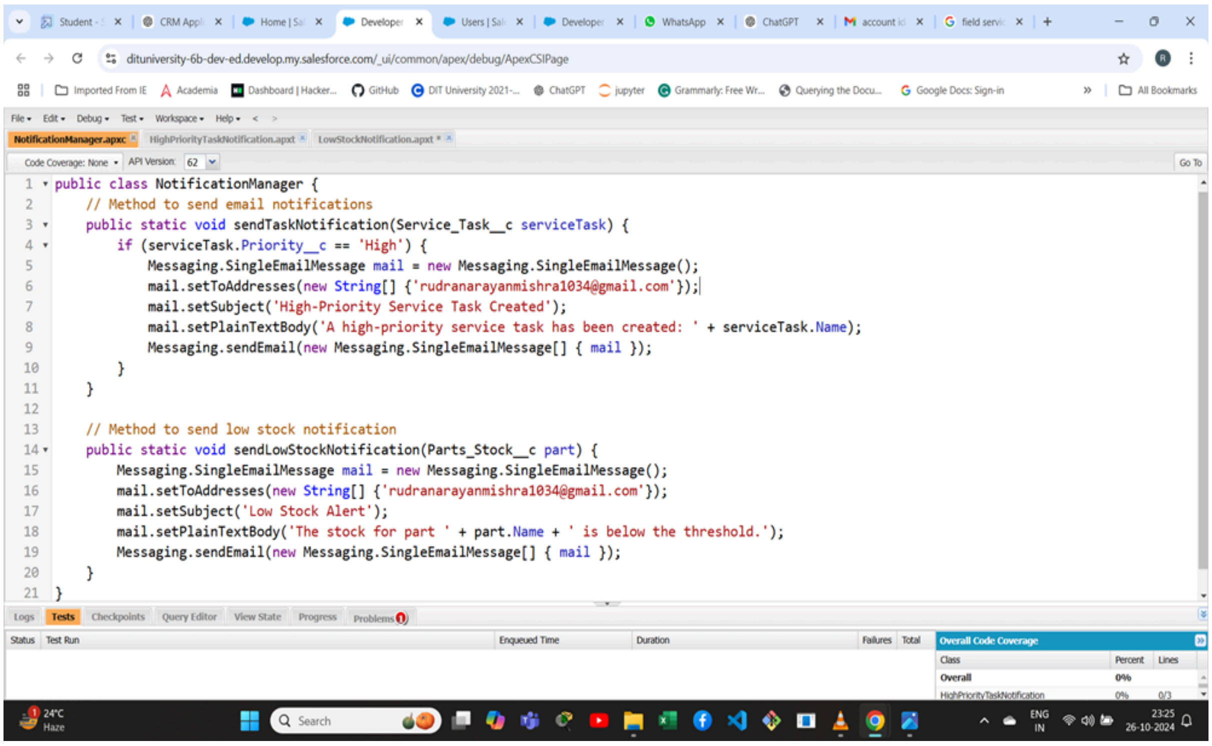
A CRM Application For FieldForce Optimizer

Apex Classes and Triggers:

Apex Classes are server-side scripts written in Salesforce's proprietary programming language, Apex, which encapsulate the logic for business processes and can be used to create reusable components, including custom controllers and services.

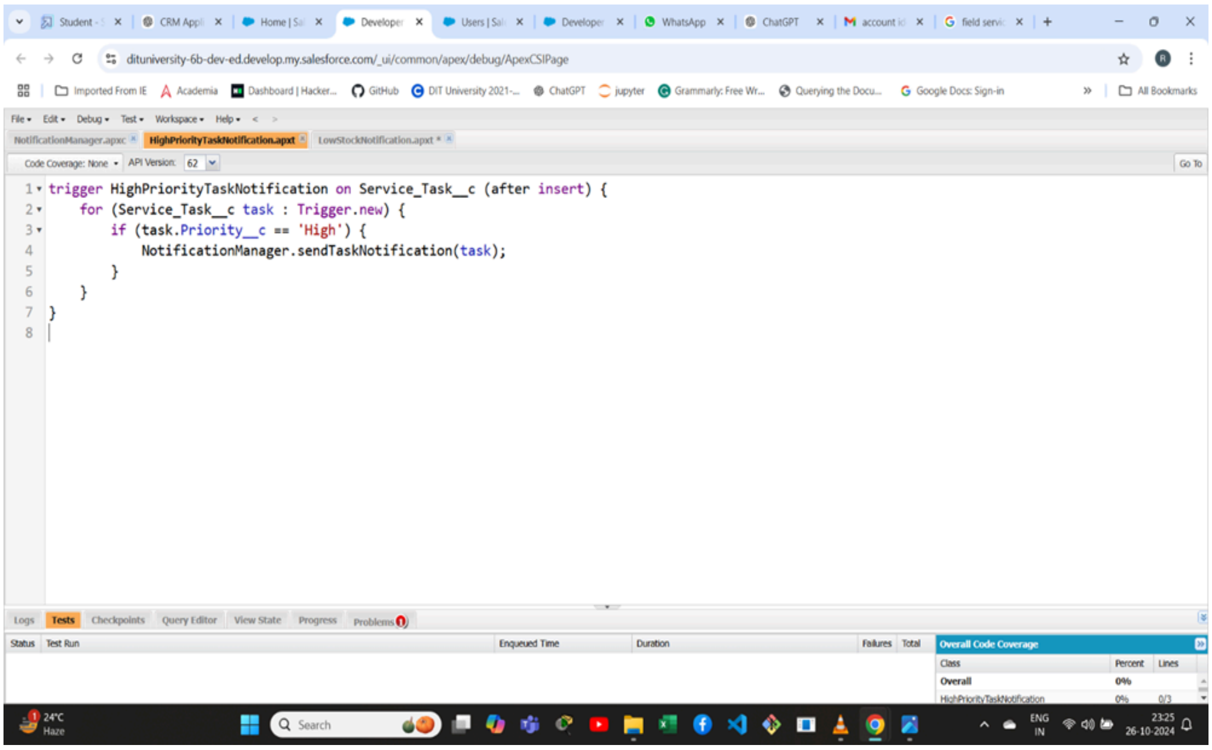
Apex Triggers are pieces of code that automatically execute before or after specific data manipulation language (DML) operations, such as insert, update, delete, or undelete, on Salesforce records.

- **Notification Manager Class** is responsible for handling notifications within the FieldForce Optimizer system. It manages the creation, sending, and tracking of notifications related to task assignments and updates, ensuring that users are informed in a timely manner. This class includes methods for configuring notification settings and for notifying field agents and other stakeholders based on specific triggers or events.



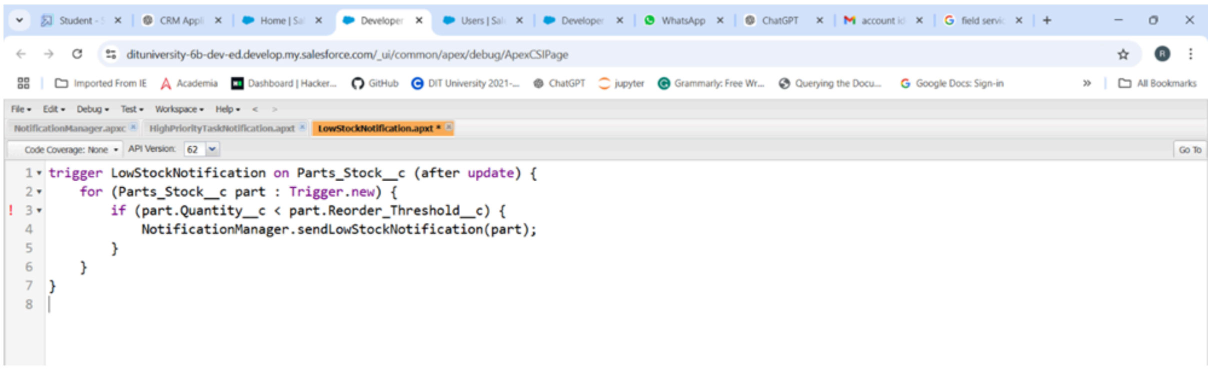
```
public class NotificationManager {
    // Method to send email notifications
    public static void sendTaskNotification(Service_Task__c serviceTask) {
        if (serviceTask.Priority__c == 'High') {
            Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();
            mail.setToAddresses(new String[] {'rudranarayanmishra1034@gmail.com'});
            mail.setSubject('High-Priority Service Task Created');
            mail.setPlainTextBody('A high-priority service task has been created: ' + serviceTask.Name);
            Messaging.sendEmail(new Messaging.SingleEmailMessage[] { mail });
        }
    }

    // Method to send low stock notification
    public static void sendLowStockNotification(Parts_Stock__c part) {
        Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();
        mail.setToAddresses(new String[] {'rudranarayanmishra1034@gmail.com'});
        mail.setSubject('Low Stock Alert');
        mail.setPlainTextBody('The stock for part ' + part.Name + ' is below the threshold.');
        Messaging.sendEmail(new Messaging.SingleEmailMessage[] { mail });
    }
}
```

- **High Priority Task Notification Trigger** This trigger fires when a service task is created or updated with a high priority status. It automatically sends notifications to the assigned field agent to ensure timely attention to critical tasks.



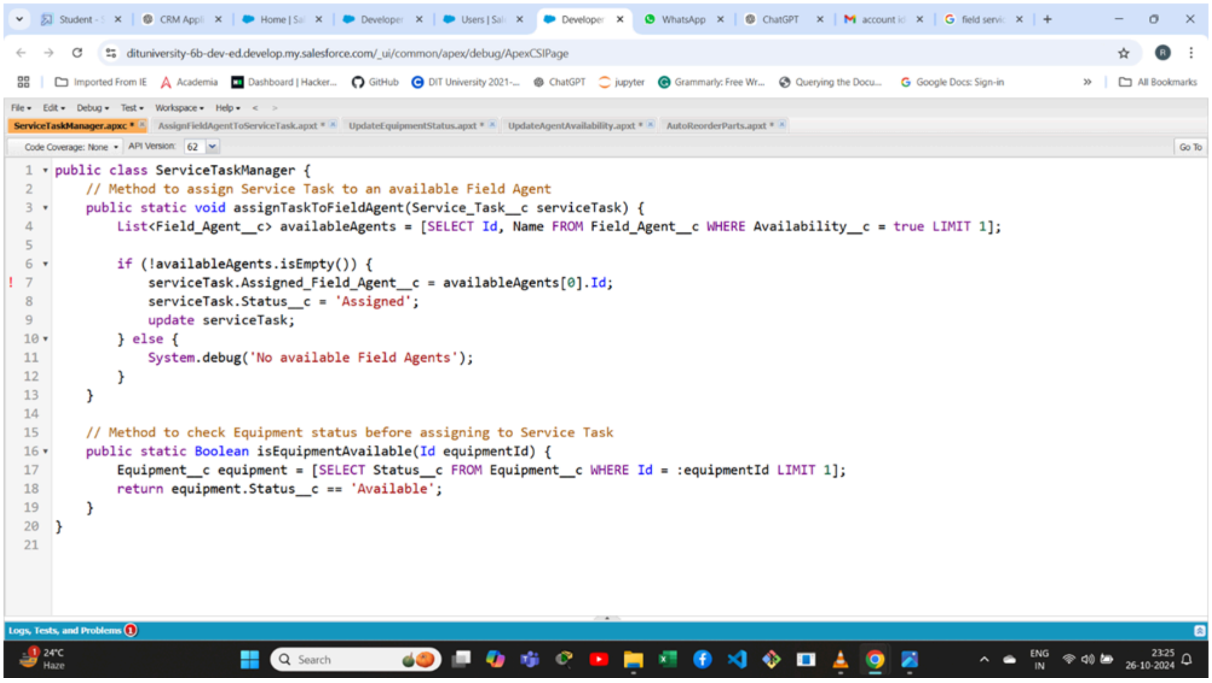
```
trigger HighPriorityTaskNotification on Service_Task__c (after insert) {
    for (Service_Task__c task : Trigger.new) {
        if (task.Priority__c == 'High') {
            NotificationManager.sendTaskNotification(task);
        }
    }
}
```

- **Low Stock Notification Trigger** This trigger activates when the quantity of any part stock falls below the predefined reorder threshold. It generates alerts to inventory managers, enabling timely restocking actions to avoid operational disruptions.



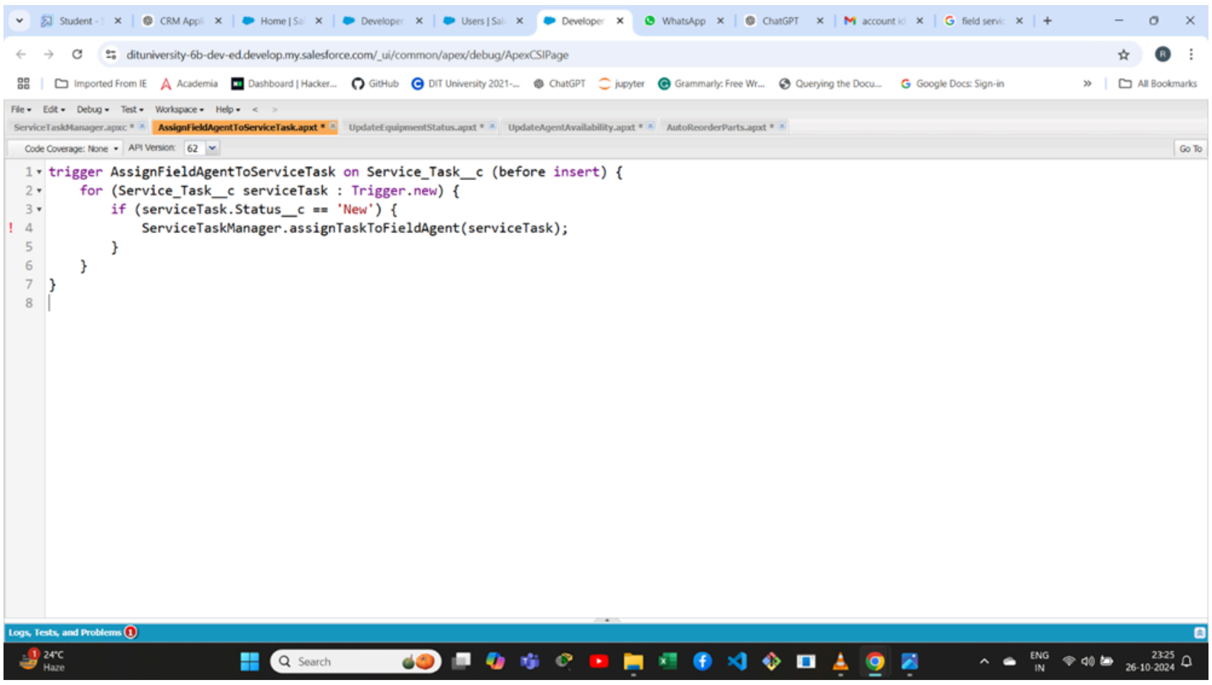
```
trigger LowStockNotification on Parts_Stock__c (after update) {
    for (Parts_Stock__c part : Trigger.new) {
        if (part.Quantity__c < part.Reorder_Threshold__c) {
            NotificationManager.sendLowStockNotification(part);
        }
    }
}
```

- **Service Task Manager Class** oversees the management and lifecycle of service tasks within the FieldForce Optimizer application. It includes functionalities for creating, updating, and retrieving service task records, as well as assigning tasks to field agents based on predefined criteria.



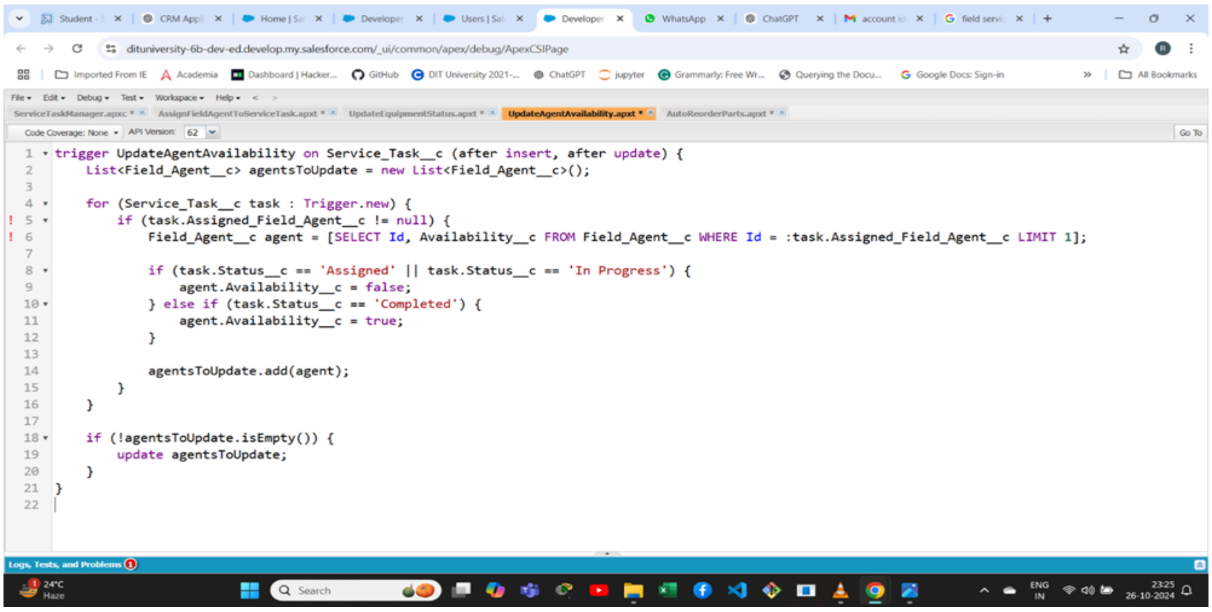
```
public class ServiceTaskManager {
    // Method to assign Service Task to an available Field Agent
    public static void assignTaskToFieldAgent(Service_Task__c serviceTask) {
        List<Field_Agent__c> availableAgents = [SELECT Id, Name FROM Field_Agent__c WHERE Availability__c = true LIMIT 1];

        if (!availableAgents.isEmpty()) {
            serviceTask.Assigned_Field_Agent__c = availableAgents[0].Id;
            serviceTask.Status__c = 'Assigned';
            update serviceTask;
        } else {
            System.debug('No available Field Agents');
        }
    }

    // Method to check Equipment status before assigning to Service Task
    public static Boolean isEquipmentAvailable(Id equipmentId) {
        Equipment__c equipment = [SELECT Status__c FROM Equipment__c WHERE Id = :equipmentId LIMIT 1];
        return equipment.Status__c == 'Available';
    }
}
```

- **Assign Field Agent to Service Task Trigger** This trigger automatically assigns a field agent to a service task upon its creation based on predefined criteria, such as agent availability or skill set. It streamlines the task assignment process, ensuring that tasks are promptly allocated to the appropriate personnel.


dituniversity-6b-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage
File Edit Debug Test Workspace Help
ServiceTaskManager.apxc
AssignFieldAgentToServiceTask.apxt
UpdateEquipmentStatus.apxt
UpdateAgentAvailability.apxt
AutoReorderParts.apxt
Code Coverage: None
API Version: 62
Logs, Tests, and Problems

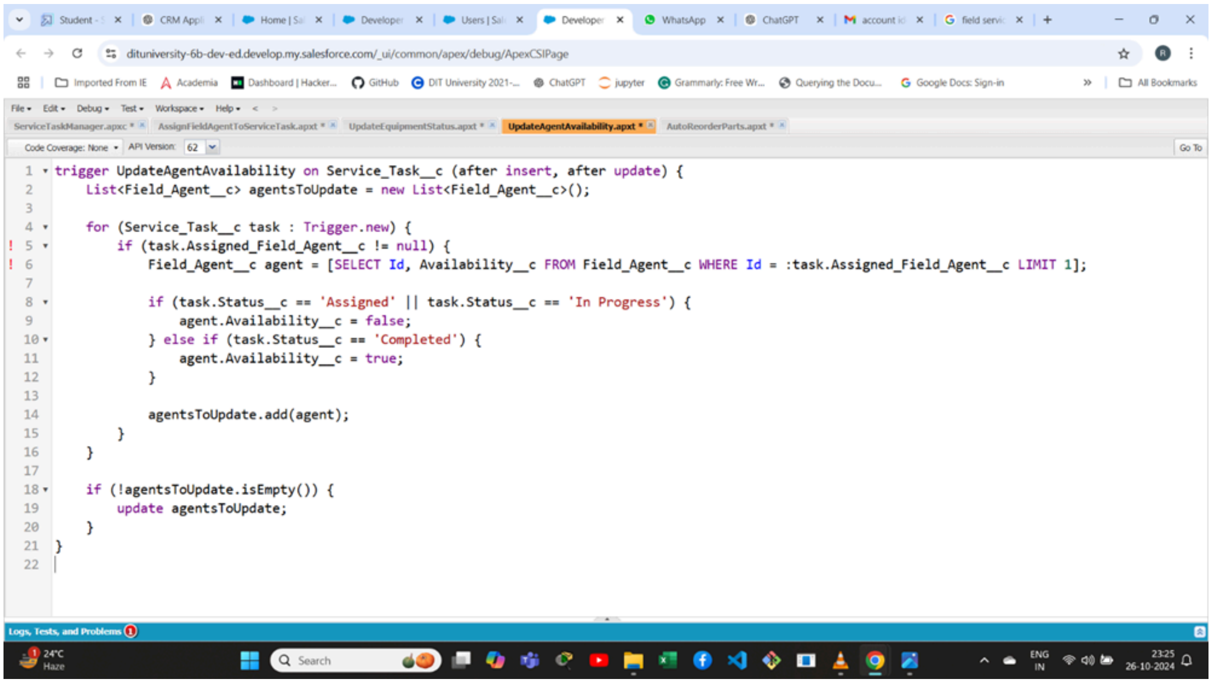
```
trigger AssignFieldAgentToServiceTask on Service_Task__c (before insert) {
    for (Service_Task__c serviceTask : Trigger.new) {
        if (serviceTask.Status__c == 'New') {
            ServiceTaskManager.assignTaskToFieldAgent(serviceTask);
        }
    }
}
```

- **Update Agent Availability Trigger** This trigger updates the availability status of field agents whenever a service task is assigned or completed. It helps maintain accurate records of agent availability, ensuring that only available agents are assigned to new tasks.


dituniversity-6b-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage
ServiceTaskManager.apxc
AssignFieldAgentToServiceTask.apxt
UpdateEquipmentStatus.apxt
UpdateAgentAvailability.apxt
AutoReorderParts.apxt
Logs, Tests, and Problems

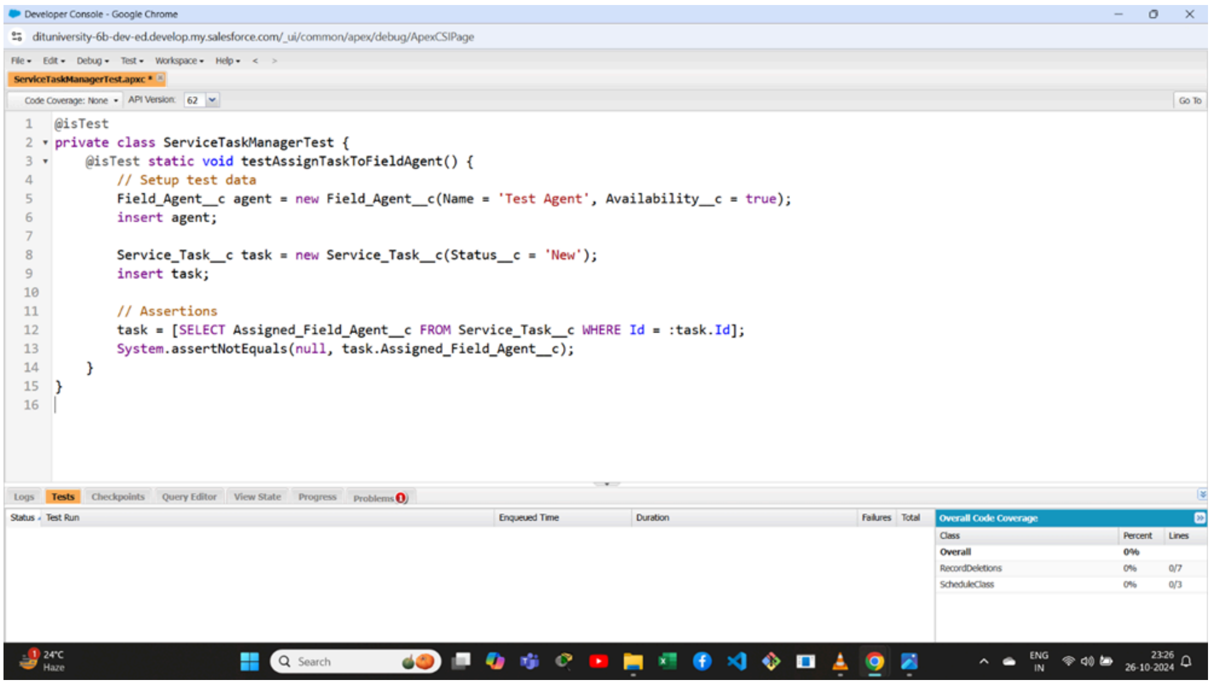
```
trigger UpdateAgentAvailability on Service_Task__c (after insert, after update) {
    List<Field_Agent__c> agentsToUpdate = new List<Field_Agent__c>();

    for (Service_Task__c task : Trigger.new) {
        if (task.Assigned_Field_Agent__c != null) {
            Field_Agent__c agent = [SELECT Id, Availability__c FROM Field_Agent__c WHERE Id = :task.Assigned_Field_Agent__c LIMIT 1];

            if (task.Status__c == 'Assigned' || task.Status__c == 'In Progress') {
                agent.Availability__c = false;
            } else if (task.Status__c == 'Completed') {
                agent.Availability__c = true;
            }

            agentsToUpdate.add(agent);
        }
    }

    if (!agentsToUpdate.isEmpty()) {
        update agentsToUpdate;
    }
}
```

- **Auto Reorder Parts Trigger** This trigger initiates an automatic reorder process when parts required for a service task are below a certain stock level. It helps maintain optimal inventory levels, ensuring that necessary parts are available for upcoming service tasks without manual intervention.



```
trigger UpdateAgentAvailability on Service_Task__c (after insert, after update) {
    List<Field_Agent__c> agentsToUpdate = new List<Field_Agent__c>();

    for (Service_Task__c task : Trigger.new) {
        if (task.Assigned_Field_Agent__c != null) {
            Field_Agent__c agent = [SELECT Id, Availability__c FROM Field_Agent__c WHERE Id = :task.Assigned_Field_Agent__c LIMIT 1];

            if (task.Status__c == 'Assigned' || task.Status__c == 'In Progress') {
                agent.Availability__c = false;
            } else if (task.Status__c == 'Completed') {
                agent.Availability__c = true;
            }

            agentsToUpdate.add(agent);
        }
    }

    if (!agentsToUpdate.isEmpty()) {
        update agentsToUpdate;
    }
}
```

Trigger Test: Verify triggers for creating restock requests and ensure they meet the criteria (e.g., Service Task Manager test).


Developer Console - Google Chrome
dituniversity-6b-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage
File Edit Debug Test Workspace Help
ServiceTaskManagerTest.apxc *
Code Coverage: None API Version: 62
Go To

```
@isTest
private class ServiceTaskManagerTest {
    @isTest static void testAssignTaskToFieldAgent() {
        // Setup test data
        Field_Agent__c agent = new Field_Agent__c(Name = 'Test Agent', Availability__c = true);
        insert agent;

        Service_Task__c task = new Service_Task__c(Status__c = 'New');
        insert task;

        // Assertions
        task = [SELECT Assigned_Field_Agent__c FROM Service_Task__c WHERE Id = :task.Id];
        System.assertNotEquals(null, task.Assigned_Field_Agent__c);
    }
}
```

Logs Tests Checkpoints Query Editor View State Progress Problems 1
Status Test Run Enqueued Time Duration Failures Total
Overall Code Coverage

Class	Percent	Lines
Overall	**0%**	
RecordDeletions	0%	0/7
ScheduleClass	0%	0/3