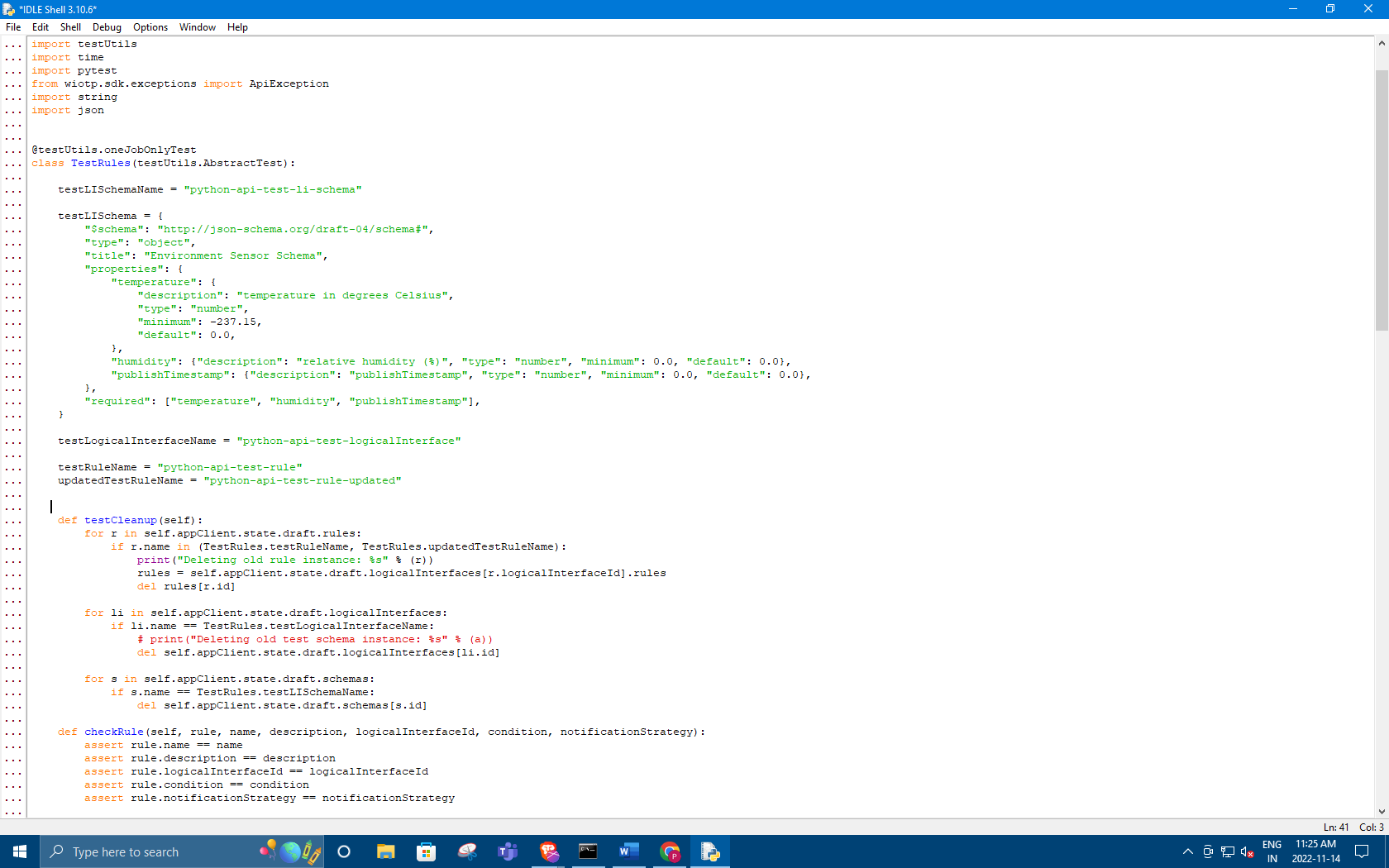
**DEVELOP THE PYTHON CODE**

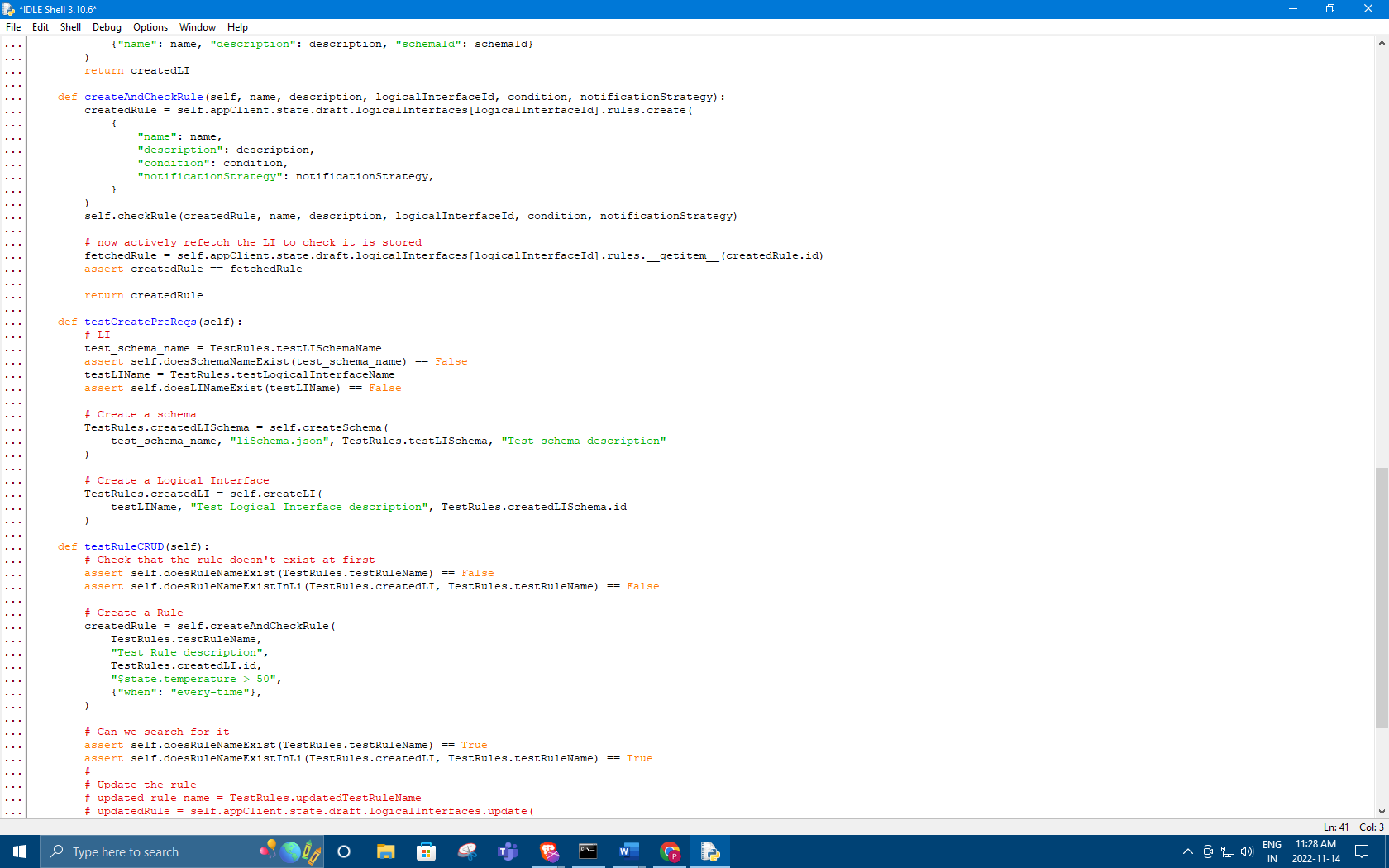
**Arun prakash V (7376201EC502)**

**Dinakar S (7376201EC504)**

**Paarthiban K (7376201EC514)**

**Santhose P (7376201EC518**





Python code:

import uuid

from datetime import datetime

import testUtils

import time

import pytest

from wiotp.sdk.exceptions import ApiException

import string

import json

@testUtils.oneJobOnlyTest

class TestRules(testUtils.AbstractTest):

testLISchemaName = "python-api-test-li-schema"

testLISchema = {

"$schema": "http://json-schema.org/draft-04/schema#",

"type": "object",

"title": "Environment Sensor Schema",

"properties": {

"temperature": {

"description": "temperature in degrees Celsius",

"type": "number",

"minimum": -237.15,

"default": 0.0,

},

"humidity": {"description": "relative humidity (%)", "type": "number", "minimum": 0.0, "default": 0.0},

"publishTimestamp": {"description": "publishTimestamp", "type": "number", "minimum": 0.0, "default": 0.0},

},

"required": ["temperature", "humidity", "publishTimestamp"],

}

testLogicalInterfaceName = "python-api-test-logicalInterface"

testRuleName = "python-api-test-rule"

updatedTestRuleName = "python-api-test-rule-updated"

def testCleanup(self):

for r in self.appClient.state.draft.rules:

if r.name in (TestRules.testRuleName, TestRules.updatedTestRuleName):

print("Deleting old rule instance: %s" % (r))

rules = self.appClient.state.draft.logicalInterfaces[r.logicalInterfaceId].rules

del rules[r.id]

for li in self.appClient.state.draft.logicalInterfaces:

if li.name == TestRules.testLogicalInterfaceName:

# print("Deleting old test schema instance: %s" % (a))

del self.appClient.state.draft.logicalInterfaces[li.id]

for s in self.appClient.state.draft.schemas:

if s.name == TestRules.testLISchemaName:

del self.appClient.state.draft.schemas[s.id]

def checkRule(self, rule, name, description, logicalInterfaceId, condition, notificationStrategy):

assert rule.name == name

assert rule.description == description

assert rule.logicalInterfaceId == logicalInterfaceId

assert rule.condition == condition

assert rule.notificationStrategy == notificationStrategy

assert isinstance(rule.created, datetime)

assert isinstance(rule.createdBy, str)

assert isinstance(rule.updated, datetime)

assert isinstance(rule.updatedBy, str)

def doesSchemaNameExist(self, name):

for a in self.appClient.state.draft.schemas.find({"name": name}):

if a.name == name:

return True

return False

def doesLINameExist(self, name):

for li in self.appClient.state.draft.logicalInterfaces.find({"name": name}):

if li.name == name:

return True

return False

def doesRuleNameExist(self, name):

for r in self.appClient.state.draft.rules.find({"name": name}):

if r.name == name:

return True

return False

def doesRuleNameExistInLi(self, li, name):

for r in li.rules:

if r.name == name:

return True

return False

def createSchema(self, name, schemaFileName, schemaContents, description):

jsonSchemaContents = json.dumps(schemaContents)

createdSchema = self.appClient.state.draft.schemas.create(name, schemaFileName, jsonSchemaContents, description)

return createdSchema

def createLI(self, name, description, schemaId):

createdLI = self.appClient.state.draft.logicalInterfaces.create(

{"name": name, "description": description, "schemaId": schemaId}

)

return createdLI

def createAndCheckRule(self, name, description, logicalInterfaceId, condition, notificationStrategy):

createdRule = self.appClient.state.draft.logicalInterfaces[logicalInterfaceId].rules.create(

{

"name": name,

"description": description,

"condition": condition,

"notificationStrategy": notificationStrategy,

}

)

self.checkRule(createdRule, name, description, logicalInterfaceId, condition, notificationStrategy)

# now actively refetch the LI to check it is stored

fetchedRule = self.appClient.state.draft.logicalInterfaces[logicalInterfaceId].rules.\_\_getitem\_\_(createdRule.id)

assert createdRule == fetchedRule

return createdRule

def testCreatePreReqs(self):

# LI

test\_schema\_name = TestRules.testLISchemaName

assert self.doesSchemaNameExist(test\_schema\_name) == False

testLIName = TestRules.testLogicalInterfaceName

assert self.doesLINameExist(testLIName) == False

# Create a schema

TestRules.createdLISchema = self.createSchema(

test\_schema\_name, "liSchema.json", TestRules.testLISchema, "Test schema description"

)

# Create a Logical Interface

TestRules.createdLI = self.createLI(

testLIName, "Test Logical Interface description", TestRules.createdLISchema.id

)

def testRuleCRUD(self):

# Check that the rule doesn't exist at first

assert self.doesRuleNameExist(TestRules.testRuleName) == False

assert self.doesRuleNameExistInLi(TestRules.createdLI, TestRules.testRuleName) == False

# Create a Rule

createdRule = self.createAndCheckRule(

TestRules.testRuleName,

"Test Rule description",

TestRules.createdLI.id,

"$state.temperature > 50",

{"when": "every-time"},

)

# Can we search for it

assert self.doesRuleNameExist(TestRules.testRuleName) == True

assert self.doesRuleNameExistInLi(TestRules.createdLI, TestRules.testRuleName) == True

#

# Update the rule

# updated\_rule\_name = TestRules.updatedTestRuleName

# updatedRule = self.appClient.state.draft.logicalInterfaces.update(

# createdLI.id, {'id': createdLI.id, 'name': updated\_li\_name, 'description': "Test LI updated description", "schemaId": createdSchema.id})

# self.checkLI(updatedLI, updated\_li\_name, "Test LI updated description", createdSchema.id)

# Delete the Rule

del TestRules.createdLI.rules[createdRule.id]

# It should be gone

assert self.doesRuleNameExist(TestRules.testRuleName) == False

assert self.doesRuleNameExistInLi(TestRules.createdLI, TestRules.testRuleName) == False

def testDeletePreReqs(self):

# Delete the LI

del self.appClient.state.draft.logicalInterfaces[TestRules.createdLI.id]

assert self.doesLINameExist(TestRules.testLogicalInterfaceName) == False

# Delete the schema

del self.appClient.state.draft.schemas[TestRules.createdLISchema.id]

assert self.doesSchemaNameExist(TestRules.testLISchemaName) == False