INTRODUCTION

Through the internship I have got to know the real working environment that was highly particular for my course study. I am really happy to work in the department to know about how a software industry develops the software products as application used by various functional requirements. I am assigned as a software developer in PARKYERI IT Solutions. My responsibilities are qa testing and to develop web applications. I have attend the project discussion meeting and they define my responsibilities for the project. I have also attend and contribute in database design for the project. Those experience increase my knowledge and improve my confidence and prepare myself for the next project. I have faced many problems in my internship period my team members helps me to overcome the problems both technical and non-technical. I also attend some client meeting and collect the requirements for the project. And also go with the sales team to deliver a software. All my experience helps me to get parts like group work and works with a team. I also get the flavor of working in corporate field. Thus, the internship duration period provide me with the opportunity to enlarge my knowledge, acknowledge the strength and weakness that will be helpful for my future career. I really feel proud to be a student of Software Engineering in Altinbas University. Because of my university allow me for the internship within the Bachelor program. I am also thankful to the PARKYERI IT Solutions, they provide me the opportunity to work with them and they make me a part with their developer team, and give me chance to utilize my theoretical knowledge and experience in the real life project.

1.1 Background

I have joined PARKYERI IT Solutions at 8th Augost2023. I am joined as a qa tester and web developer on the organization. At first I have get a training about 2 weeks. Then they defined me with a team. The responsibilities of our team is to develop the website both design and make it dynamic the whole site and another team work on NOAPI CLOUD .

At first I am started with the data base architecture for the "challenge app" and working on backend side then I start to learn cucumber and python to work on automation tests , I mainly use Python, JavaScript, MySQL as a programming language for do my responsibilities.

1.2 Motivation

Internship program provides me an opportunity to enlarge my skills, gather experience through industrial work on advanced production and my goal is to be familiar with real-life corporate environment. Internship helps me to connect with an organization and provides scope to work with them. Through this engagement I can able to sharpen my knowledge and improve my experience. I am able to learn lot of technologies both old and new technologies. I always want to be a valuable asset for a rewound company. And I am determined to give my best service for this. Through this internship program I can able to get the proper knowledge about the software industry.

1.3 Objectives

My objective is to join the internship program for gathering industrial environment experience and gather practical and real life knowledge with the direct interaction with customers. My objective is to know the customer requirements and then go through the development. PARKYERI has strengthen on Quality Management practices since its inception. I have attend the project meeting to gather requirements and then they define my work for the project. They helps me to do my job by following the industrial rules and regulations. I am prepare to do a project with group and team, also get the knowledge how to develop a project properly.

1.4 Scope

The goal is to prepare me as a full stack developer. And i am doing my activities as intern to be a full stack developer. I am also getting and learning the official responsibility to maintain the time and deliver my activities in due time. It helps me to Improving my communication and data exchange.

- Learn how to maintain and follow the official rules.
- To solve the real life problems.
- Know the business pattern & develop the software.
- Deliver product and official presentation
- Training about the new applications.
- Doing automation testing
- Coding is not only the profession.
- It's the passion where should never compromise

COMPANY OVERVIEW

Parkyeri is an Istanbul based company with its scope of business covering software development, operations and consultancy. Since its foundation in 2001; it is considered one of the sector leaders with its solutions on mobile technology, Internet and database systems.

Parkyeri's solutions exceeding client expectations have given us the chance to work together with giant companies such as Turkcell, Ericsson, Etisalat and Türksat. Our unique human resources approach and our applications developed using free software have sealed us a prestigious position in the sector. Parkyeri was also Turkey's first W3C (World Wide Web Consortium) member.

2.1.1 Mission & Vision

Among few Turkish members of MMA (Mobile Marketing Association), Parkyeri has worked as active member between 2009 and 2011. Parkyeri has also established business partnership relations with Turkcell and Ericsson, with whom we have been building projects together since our foundation. Once again in 2010, Parkyeri was chosen one of the seven Gold Partners -of total 238 Turkcell business partners. Besides, Parkyeri was awarded with the Most Successful Infrastructure Provider Prize of the year 2010, with its own developed target marketing system called HaberdarEt.

In Deloitte Fast50 2007 rankings, Parkyeri was chosen the fastest growing Turkish firm and Europe's thirty-sixth. In 2008 Parkyeri managed to be ranked in the same list once again. Those achievements by Parkyeri have received great press coverage

2.1.2 Location

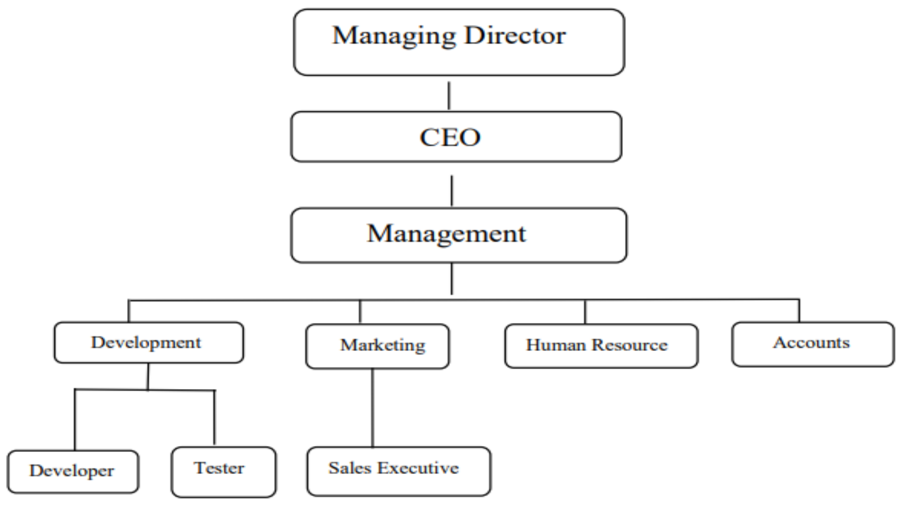
Farmgate Branch:

Parkyeri İletişim İnternet Pazarlama Tanıtım Hizmetleri Tic. ve Sanayi A.Ş.
İnebolu Sok. No:9/4 Setüstü Kabataş İstanbul 34427 Turkiye

Contact: info@parkyeri.com

2.2 Organ-Gram

Organ gram is the key for running a company or organization. PARKYERI is the leading service providing company and it has a strong Organ gram which I illustrated in below



TECHNOLOGY EMPLOYING

4.1 Fundamental Technologies

To working with the team and with the requirements I have to use both Old and New technology. In some projects we use old technology and sometimes we use new technology. We use Python to develop the projects. We never use any framework for the projects.

New Technology I Use:

- Javascript
- Cucumber
- Selenium

We use Cucumber and Selenium for the automation testing, we test the methods that we create in our SDK, also to test the web application (Challenge web site).

Python and Javascript to develop the SDK.

4.1.1 Technology In Use for the Projects

For the running projects we are using :

- Python
- JavaScript
- Cucumber

PROJECT EXERTION

5.1 Training

I have experienced to working on a multiple projects. Now most of the projects are still in progress. I mainly working on the automation testing and writing scenarios to test all the new functions made by developers and check if the give the expected results or not and write reports about that in gitlab inside the issue.

5.2 Project Name: NOAPI CLOUD

5.2.1 Project Plot & Requirements

NoAPI is a Data Framework, that provides software developers a real-time edge/cloud datastore integrated programming languages.

It offers:

- Software Development Kits in several programming languages (such as python, Javascript)
- A cloud datastore accessible through this SDK
- An intuitive data model

- Data appears as objects integrated into the programming language, without a need to develop an API.
- A Developer console, with a visual data browser/editor
- A scalable hosting service to host your application developed using the SDK
- Live data updates by default

and other features.

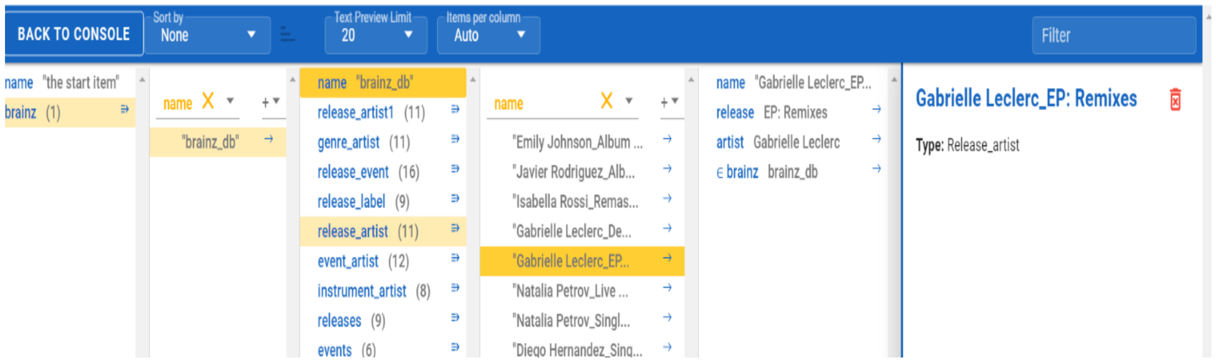
Our motto is: "**simplicity, sanity, speed**":

- **simplicity**: Everything should be simple, without compromising ability.
- **sanity**: You should be able to develop without going insane.
- **speed** Developing should be fast. Running/hosting should be fast.

Here are some simple examples on how to access a datastore:

```python
import noapi

def main():
    start = noapi.datastore("test-school")
    school = start.schools.first()
    name_of_school = school.name
    print(school.name)
    school.name="Ivy College"
    noapi.disconnect()

if __name__ == "__main__":
    noapi.run(main)
```

Example from the data store


BACK TO CONSOLE | Sort by: None | Text Preview Limit: 20 | Items per column: Auto | Filter

name "the start item"
brainz (1)

name
"brainz_db"

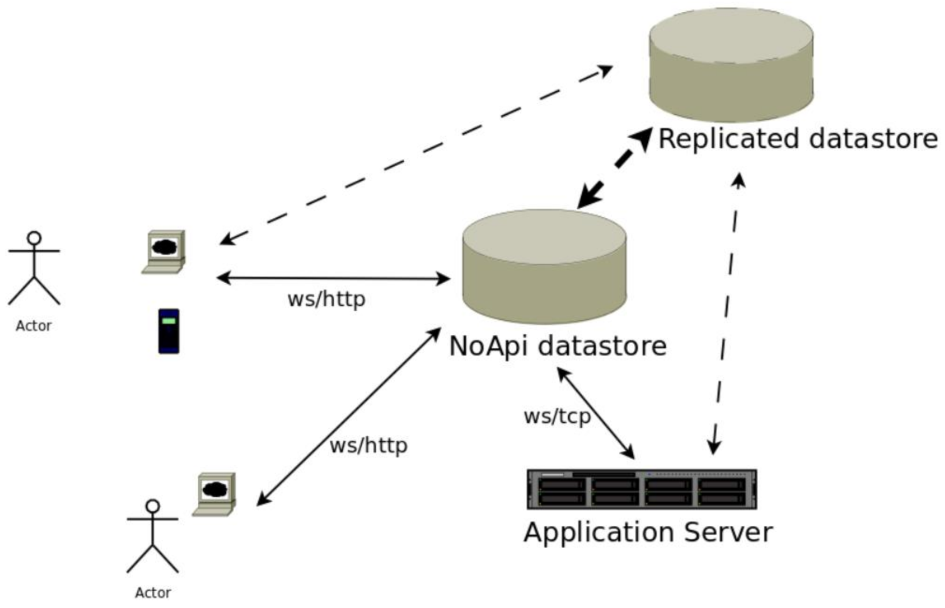
name "brainz_db"
release_artist1 (11)
genre_artist (11)
release_event (16)
release_label (9)
release_artist (11)
event_artist (12)
instrument_artist (8)
releases (9)
events (6)

name
"Emily Johnson_Album ...
"Javier Rodriguez_Alb...
"Isabella Rossi_Remas...
"Gabrielle Leclerc_De...
"Gabrielle Leclerc_EP...
"Natalia Petrov_Live ...
"Natalia Petrov_Singl...
"Diego Hernandez_Sing...

name "Gabrielle Leclerc_EP...
release EP: Remixes
artist Gabrielle Leclerc
∈ brainz brainz_db

Gabrielle Leclerc_EP: Remixes
Type: Release_artist

Architecture Overview



5.2.2 How to use Noapi

Step 1 : Install SDK

We assume that you already have python and pip installed.

Run :

```
% python3 -m pip install no-api-sdk
```

Step 2: Connect to your database

Code :

```
import noapi

async def main():
    start = noapi.datastore("test-school")
    school = start.schools.first()
    name_of_school = school.name
    print(school.name)
    school.name="Ivy College"
    noapi.disconnect()

if __name__ == "__main__":
    noapi.run(main())
```

Step 3 : Create tables in you database:

```
school.define_list_of_items("developement_club")
```

Step 4 : Set properties to the defined list:

```
developement_club.define_property("web site","str")
     developement_club.define_property("logo","str")
```

Step 5 : Update data in the database:

Code :

```
school = start.schools.first()

school.name = "New school name"
# or
school.set("name","New school name")

students = school.students

first_student = students.first()
school.student_president = first_student
# or
school.set("student_president",first_student)

a_new_student = students.create("John Doe")
# or
```

```
a_new_student = students.create()
a_student.name = "John Doe"

noapi.disconnect()
```

Step 6 : Find data in database :

- You can use `find()` to locate something such as a department, student, instructor, courses, etc.

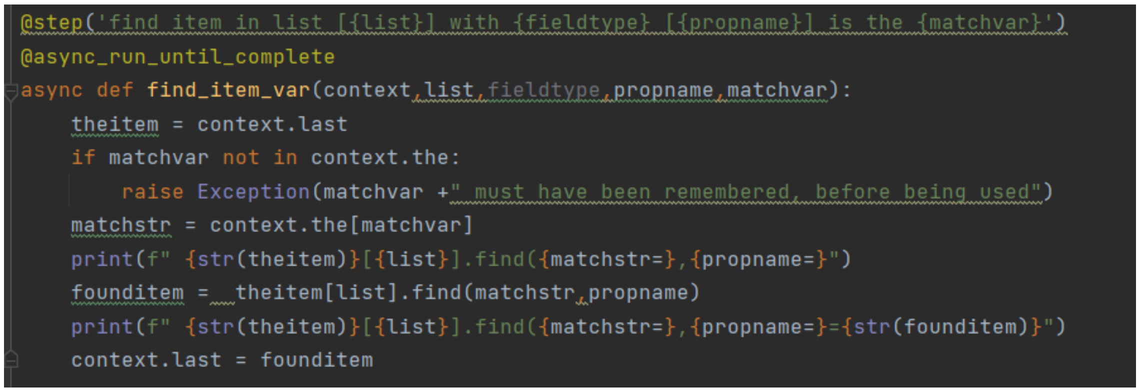
Code :

```
import noapi
def main():
  # connecting to the datastore

  #
  start = noapi.datastore("test-school")

  #getting find students from school and storing it into
student
  school = start.schools.first()
  students = school.students
```

```python

#finding students using the find() method
Brion_stu = students.find('Brion Addison')
Galen_stu  = students.find('Galen Linton')
```

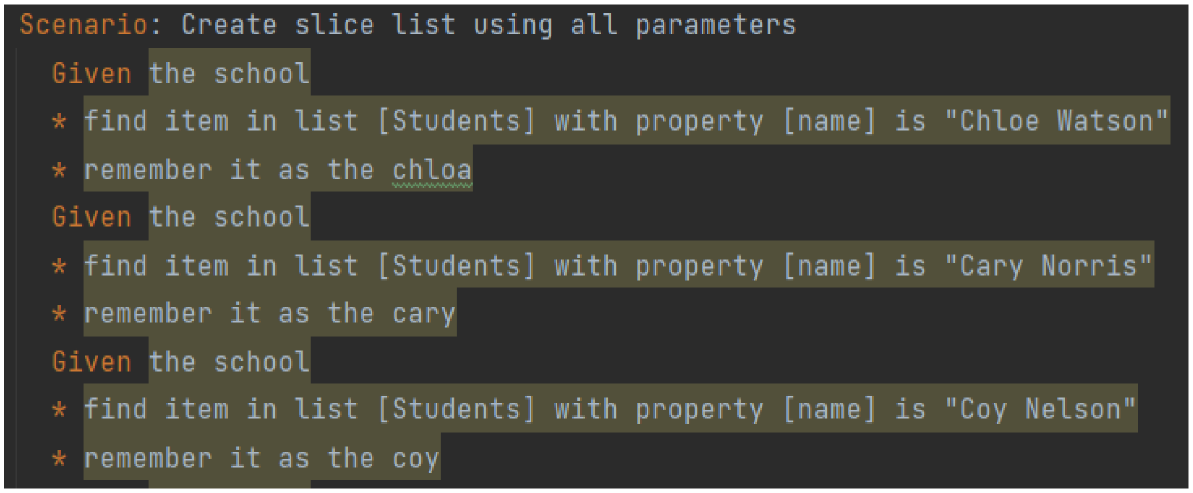
Testing for the Find() :

Step definition :



```python
@step('find item in list [{list}] with {fieldtype} [{propname}] is the {matchvar}')
@async_run_until_complete
async def find_item_var(context,list,fieldtype,propname,matchvar):
    theitem = context.last
    if matchvar not in context.the:
        raise Exception(matchvar +" must have been remembered, before being used")
    matchstr = context.the[matchvar]
    print(f" {str(theitem)}[{list}].find({matchstr=},{propname=}")
    founditem =  theitem[list].find(matchstr,propname)
    print(f" {str(theitem)}[{list}].find({matchstr=},{propname=}={str(founditem)}")
    context.last = founditem
```

Then using the step in the feature file by calling it :


```
Scenario: Create slice list using all parameters
  Given the school
  * find item in list [Students] with property [name] is "Chloe Watson"
  * remember it as the chloa
  Given the school
  * find item in list [Students] with property [name] is "Cary Norris"
  * remember it as the cary
  Given the school
  * find item in list [Students] with property [name] is "Coy Nelson"
  * remember it as the coy
```

Step 7 : add data to database:

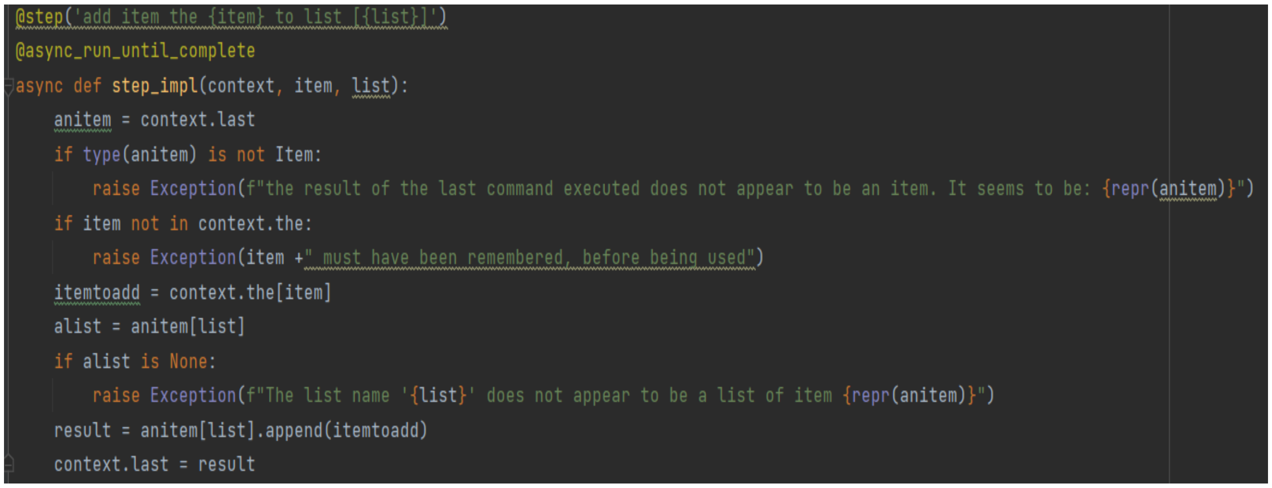
Code :

```
 # storing math department
math_dept = departments.find('Mathematics')

# creating a new list in the math department called
department_students
math_dept.define_list_of_items('department_students')

# getting student
One_student = students.first()

```

```python
# append the student in department
math_dept.department_students.append(One_student)
```

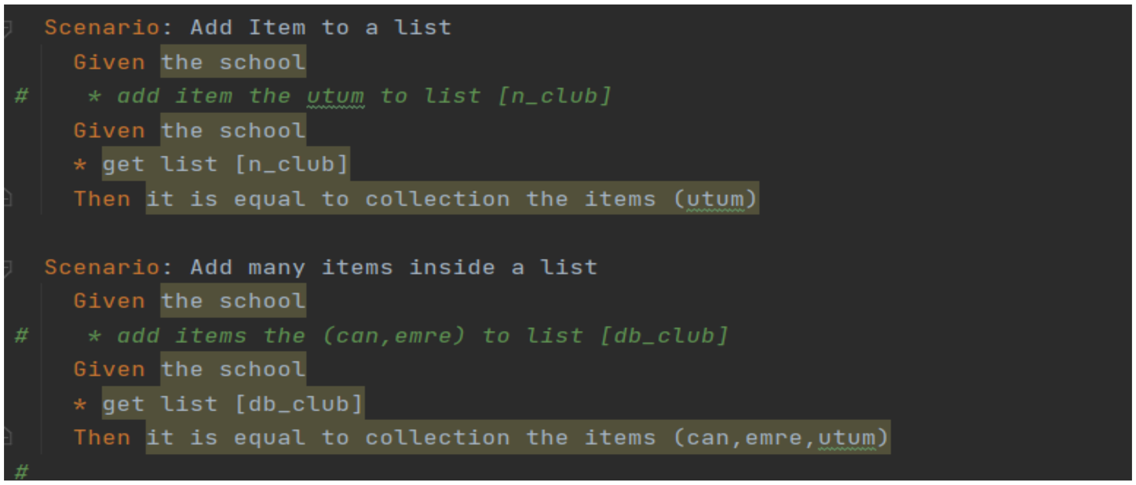
Testing for the append():

Step definition :



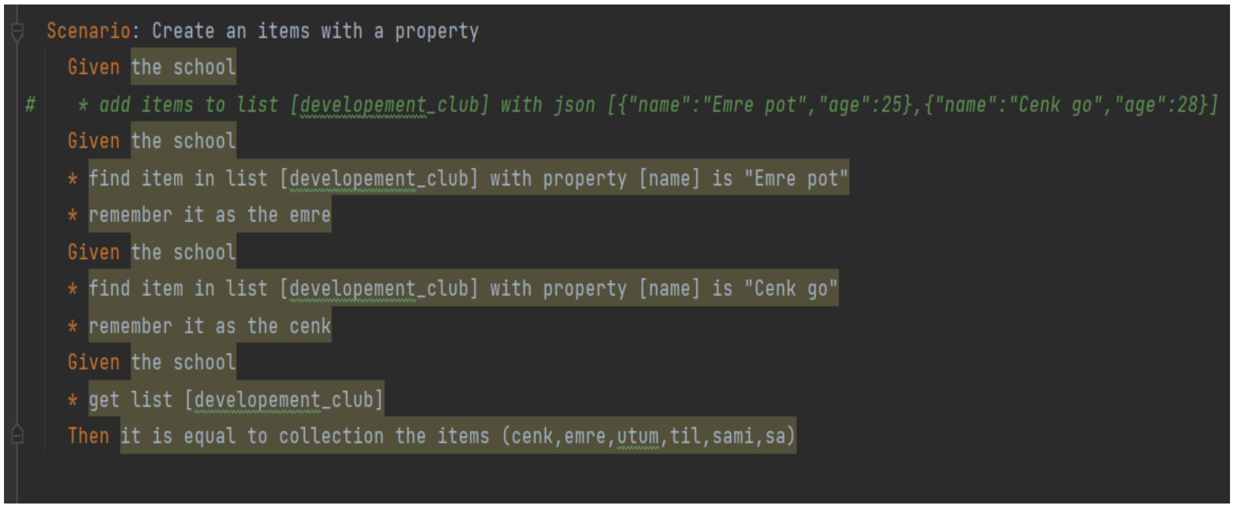
```python
@step('add item the {item} to list [{list}]')
@async_run_until_complete
async def step_impl(context, item, list):
    anitem = context.last
    if type(anitem) is not Item:
        raise Exception(f"the result of the last command executed does not appear to be an item. It seems to be: {repr(anitem)}")
    if item not in context.the:
        raise Exception(item +" must have been remembered, before being used")
    itemtoadd = context.the[item]
    alist = anitem[list]
    if alist is None:
        raise Exception(f"The list name '{list}' does not appear to be a list of item {repr(anitem)}")
    result = anitem[list].append(itemtoadd)
    context.last = result
```

Using the step definition with a scenario :



```
Scenario: Add Item to a list
  Given the school
#   * add item the utum to list [n_club]
  Given the school
  * get list [n_club]
  Then it is equal to collection the items (utum)

Scenario: Add many items inside a list
  Given the school
#   * add items the (can,emre) to list [db_club]
  Given the school
  * get list [db_club]
  Then it is equal to collection the items (can,emre,utum)
#
```

Other form to append Item inside a list (with JSON):



```
Scenario: Create an items with a property
  Given the school
#   * add items to list [developement_club] with json [{"name":"Emre pot","age":25},{"name":"Cenk go","age":28}]
  Given the school
  * find item in list [developement_club] with property [name] is "Emre pot"
  * remember it as the emre
  Given the school
  * find item in list [developement_club] with property [name] is "Cenk go"
  * remember it as the cenk
  Given the school
  * get list [developement_club]
  Then it is equal to collection the items (cenk,emre,utum,til,sami,sa)
```

Step 8: Move item to first or last position:

Code :

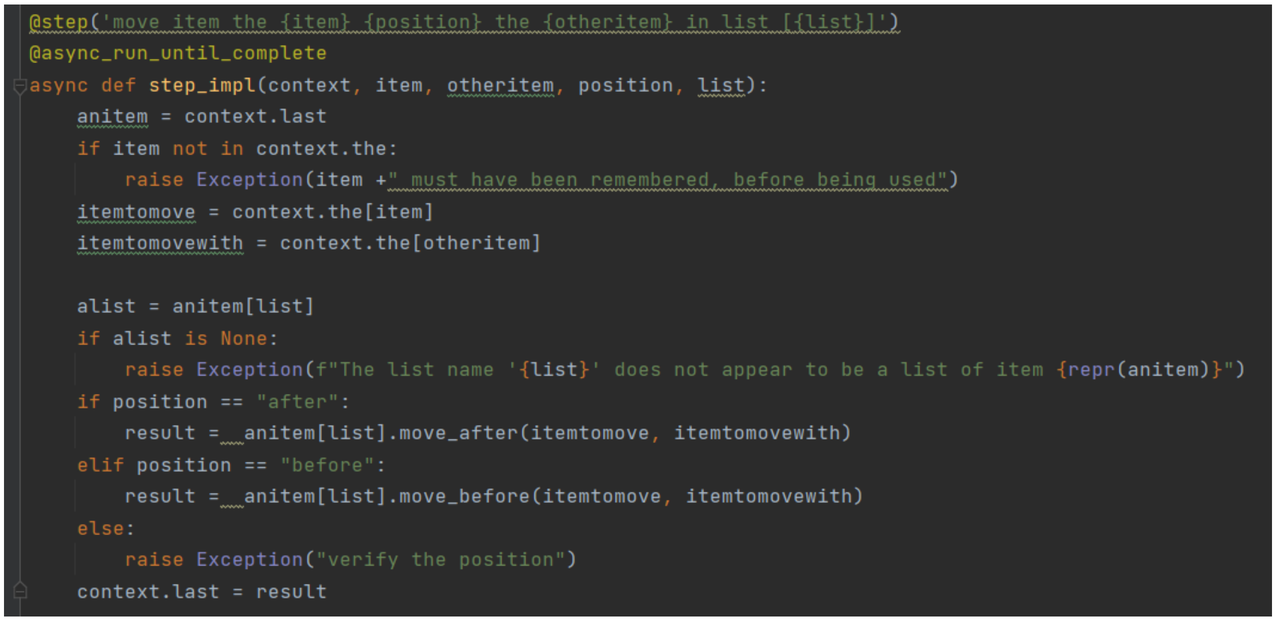
```
school = start.schools.first()

students = school.students
Brion = school.students.find('Brion Addison')

students.move_to_first(Brion)
# OR
students.move_to_last(Brion)
```

Other method to move an Item after or before another item :

Code :

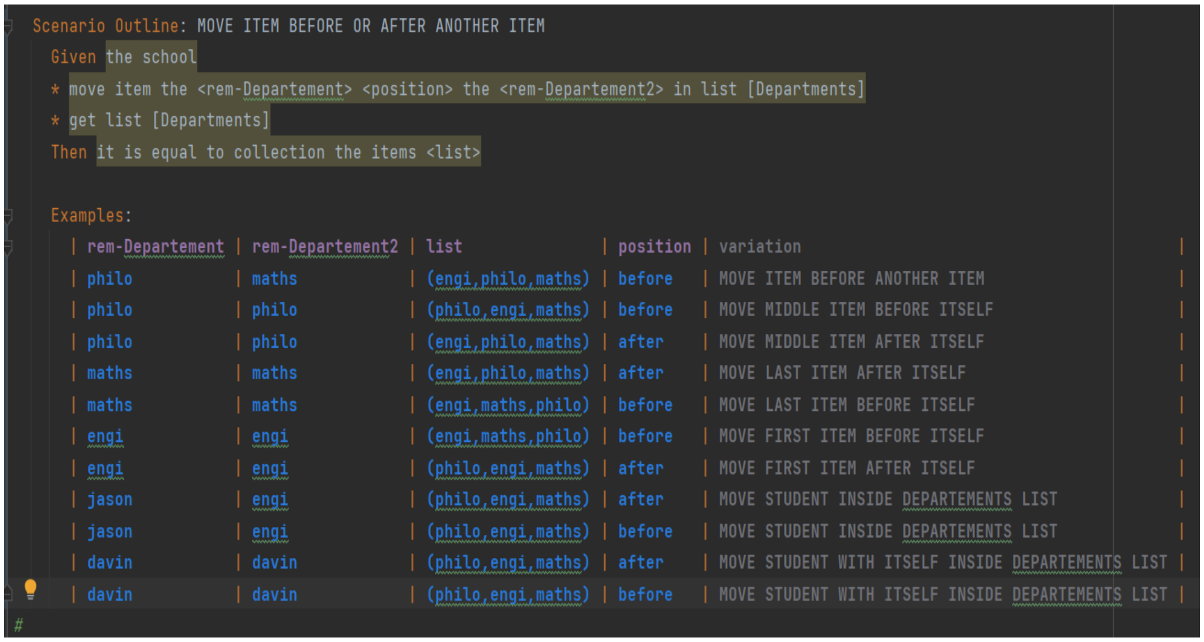
```
school = start.schools.first()

students = school.students
Brion = school.students.find('Brion Addison')
Galen =  school.students.find('Galen Linton')
students.move_after('Galen', 'Brion')

# get 1st student and print to see the changes.
new_student = school.student.first()
print('new student is:'+new_student.name)
```

Step definition to test move function :



```python
@step('move item the {item} {position} the {otheritem} in list [{list}]')
@async_run_until_complete
async def step_impl(context, item, otheritem, position, list):
    anitem = context.last
    if item not in context.the:
        raise Exception(item +" must have been remembered, before being used")
    itemtomove = context.the[item]
    itemtomovewith = context.the[otheritem]

    alist = anitem[list]
    if alist is None:
        raise Exception(f"The list name '{list}' does not appear to be a list of item {repr(anitem)}")
    if position == "after":
        result = anitem[list].move_after(itemtomove, itemtomovewith)
    elif position == "before":
        result = anitem[list].move_before(itemtomove, itemtomovewith)
    else:
        raise Exception("verify the position")
    context.last = result
```

The scenarios for the move function:

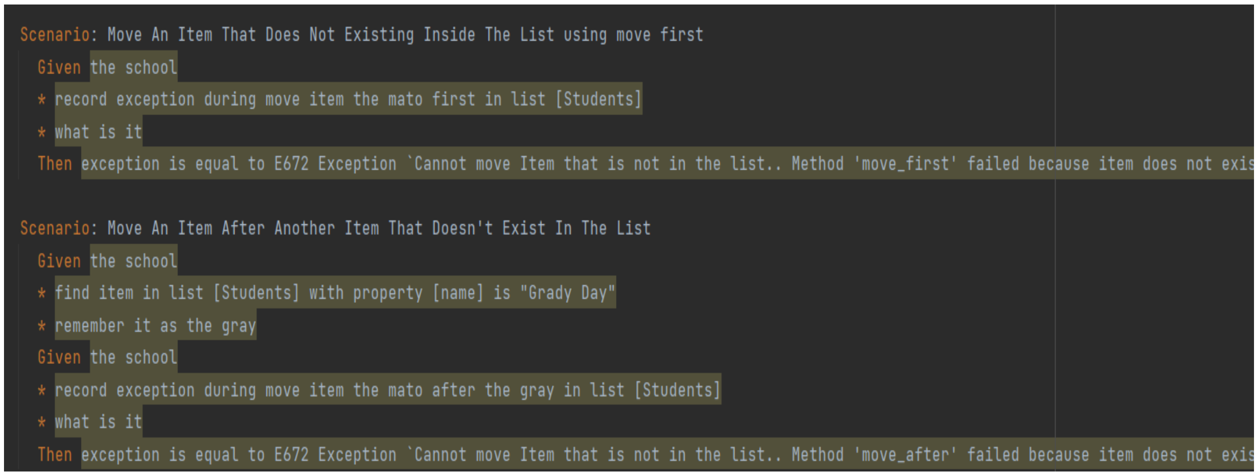


```gherkin
Scenario Outline: MOVE ITEM BEFORE OR AFTER ANOTHER ITEM
  Given the school
  * move item the <rem-Departement> <position> the <rem-Departement2> in list [Departments]
  * get list [Departments]
  Then it is equal to collection the items <list>

  Examples:
    | rem-Departement | rem-Departement2 | list               | position | variation                                      |
    | philo           | maths            | (engi,philo,maths) | before   | MOVE ITEM BEFORE ANOTHER ITEM                  |
    | philo           | philo            | (philo,engi,maths) | before   | MOVE MIDDLE ITEM BEFORE ITSELF                 |
    | philo           | philo            | (engi,philo,maths) | after    | MOVE MIDDLE ITEM AFTER ITSELF                  |
    | maths           | maths            | (engi,philo,maths) | after    | MOVE LAST ITEM AFTER ITSELF                    |
    | maths           | maths            | (engi,maths,philo) | before   | MOVE LAST ITEM BEFORE ITSELF                   |
    | engi            | engi             | (engi,maths,philo) | before   | MOVE FIRST ITEM BEFORE ITSELF                  |
    | engi            | engi             | (philo,engi,maths) | after    | MOVE FIRST ITEM AFTER ITSELF                   |
    | jason           | engi             | (philo,engi,maths) | after    | MOVE STUDENT INSIDE DEPARTEMENTS LIST          |
    | jason           | engi             | (philo,engi,maths) | before   | MOVE STUDENT INSIDE DEPARTEMENTS LIST          |
    | davin           | davin            | (philo,engi,maths) | after    | MOVE STUDENT WITH ITSELF INSIDE DEPARTEMENTS LIST |
    | davin           | davin            | (philo,engi,maths) | before   | MOVE STUDENT WITH ITSELF INSIDE DEPARTEMENTS LIST |
#
```

Recording exceptions for the move function:

Recording exceptions in Cucumber tests hook serves several purposes:

- **Error Reporting:** It allows you to capture and log information about failed scenarios, including the details of any exceptions that occurred during the test execution. This helps in diagnosing and troubleshooting issues.
- **Centralized Handling:** By using a hook, you can centralize the handling of exceptions after each scenario. This promotes code reusability and ensures that error-handling logic is consistently applied across all scenarios.



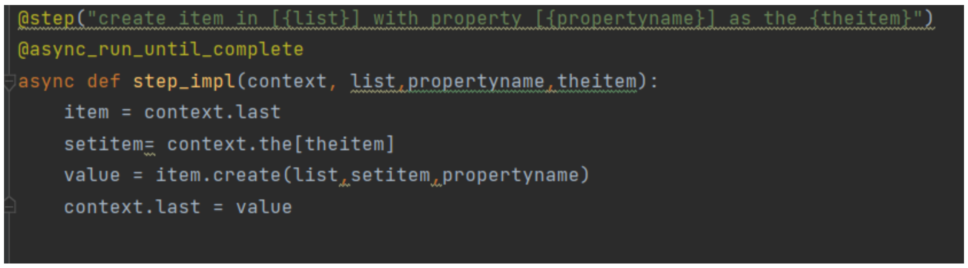
```
Scenario: Move An Item That Does Not Existing Inside The List using move first
  Given the school
  * record exception during move item the mato first in list [Students]
  * what is it
  Then exception is equal to E672 Exception `Cannot move Item that is not in the list.. Method 'move_first' failed because item does not exis

Scenario: Move An Item After Another Item That Doesn't Exist In The List
  Given the school
  * find item in list [Students] with property [name] is "Grady Day"
  * remember it as the gray
  Given the school
  * record exception during move item the mato after the gray in list [Students]
  * what is it
  Then exception is equal to E672 Exception `Cannot move Item that is not in the list.. Method 'move_after' failed because item does not exis
```

Step 9: Create Item directly inside a list:

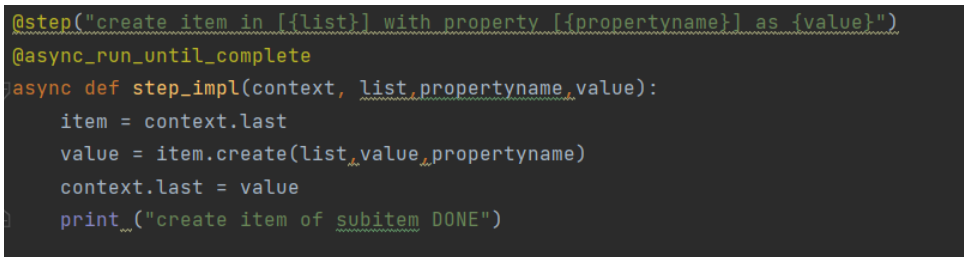
Code :

```
import noapi

     async def main():

    start = noapi.datastore("test-school")
    school = start.schools.first()

    school.define_list_of_items("students")
    students = school.students

    a_student = students.create("Jane Doe")
    students.define_property("age","num")
    a_student.age = 24

    b_student = students.create("Benjamin Castaldi")
    b_student.age = 53

    c_student = students.create("Roger Ferrand")
    c_student.age = 20

    noapi.disconnect()

if __name__ == "__main__":
    noapi.run(main())
```

Step definition to create item:

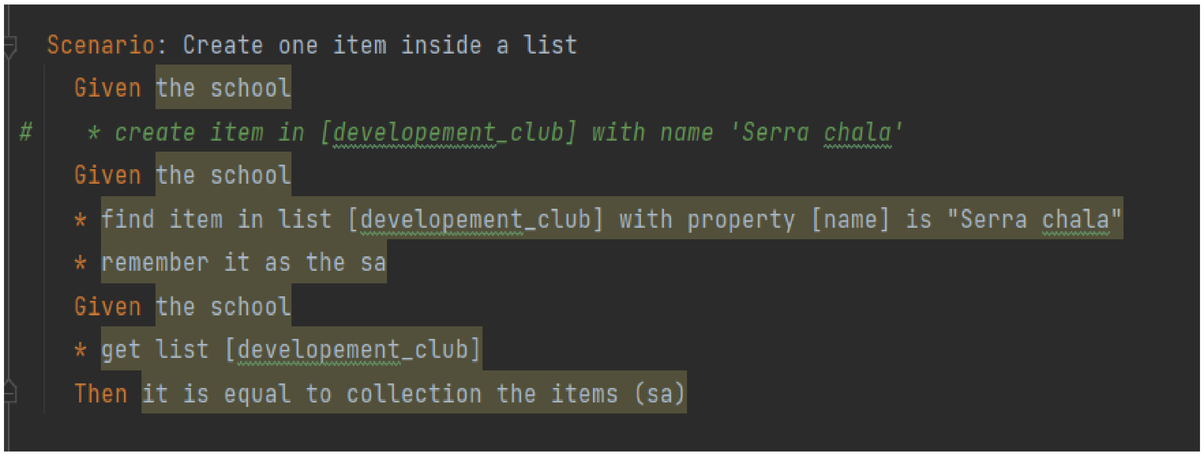


```
@step("create item in [{list}] with property [{propertyname}] as the {theitem}")
@async_run_until_complete
async def step_impl(context, list,propertyname,theitem):
    item = context.last
    setitem= context.the[theitem]
    value = item.create(list,setitem,propertyname)
    context.last = value
```



```
@step("create item in [{list}] with property [{propertyname}] as {value}")
@async_run_until_complete
async def step_impl(context, list,propertyname,value):
    item = context.last
    value = item.create(list,value,propertyname)
    context.last = value
    print ("create item of subitem DONE")
```

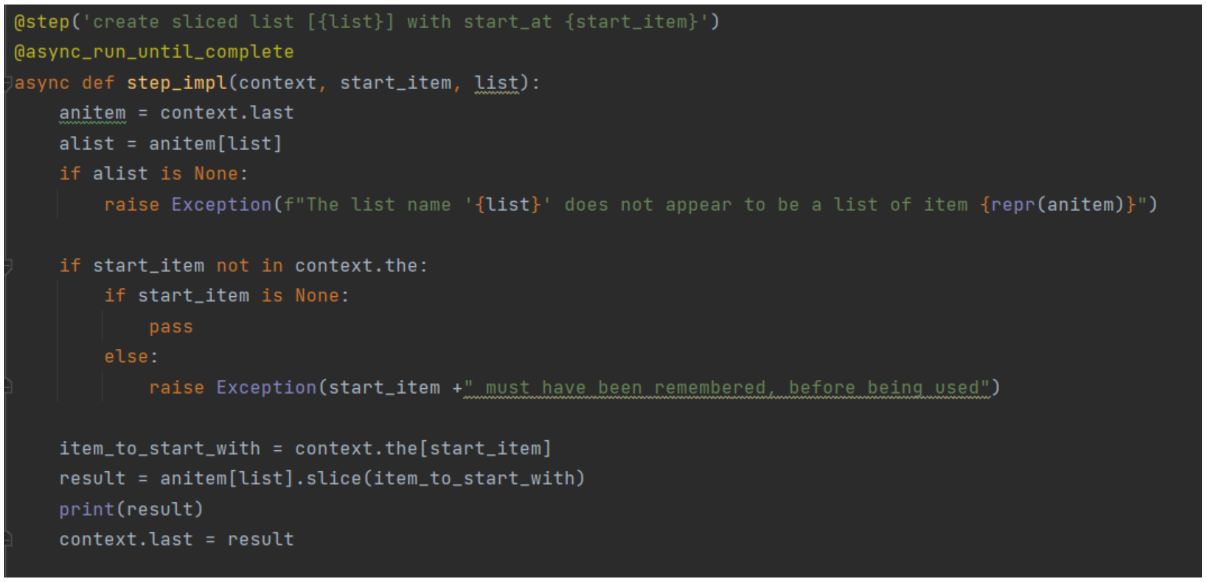
Scenarios to create Items inside lists :



```
  Scenario: Create one item inside a list
    Given the school
#    * create item in [developement_club] with name 'Serra chala'
    Given the school
    * find item in list [developement_club] with property [name] is "Serra chala"
    * remember it as the sa
    Given the school
    * get list [developement_club]
    Then it is equal to collection the items (sa)
```

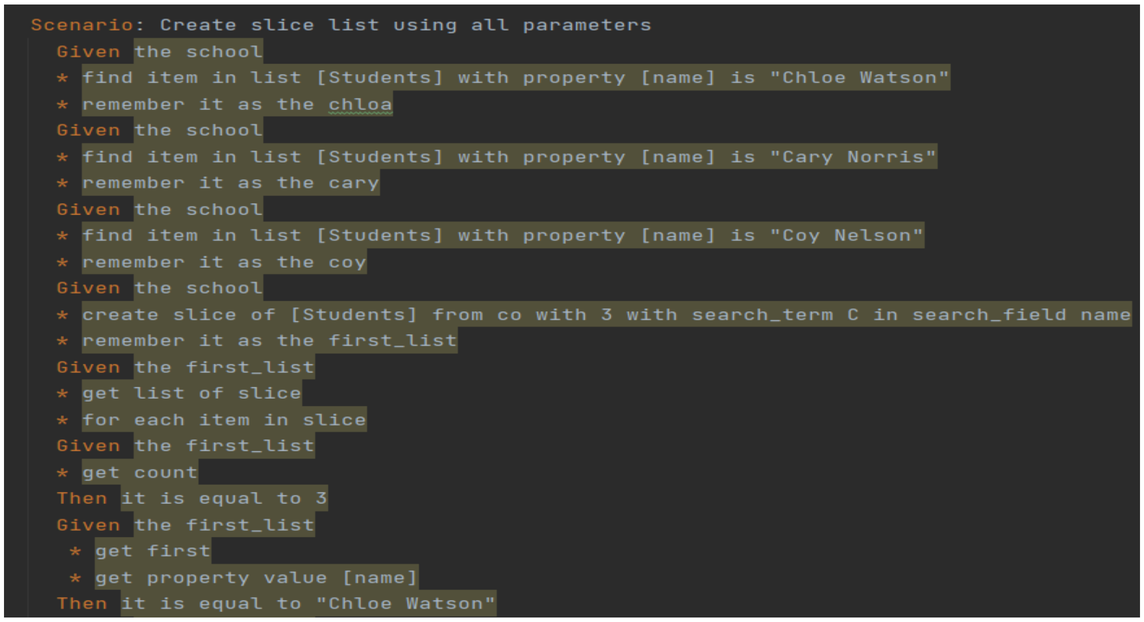
Step 10 : Create sliced list with Slice() :

Step definition to test slice method:



```python
@step('create sliced list [{list}] with start_at {start_item}')
@async_run_until_complete
async def step_impl(context, start_item, list):
    anitem = context.last
    alist = anitem[list]
    if alist is None:
        raise Exception(f"The list name '{list}' does not appear to be a list of item {repr(anitem)}")

    if start_item not in context.the:
        if start_item is None:
            pass
        else:
            raise Exception(start_item +" must have been remembered, before being used")

    item_to_start_with = context.the[start_item]
    result = anitem[list].slice(item_to_start_with)
    print(result)
    context.last = result
```

Scenario for slice method:



```gherkin
  Scenario: Create slice list using all parameters
    Given the school
    * find item in list [Students] with property [name] is "Chloe Watson"
    * remember it as the chloa
    Given the school
    * find item in list [Students] with property [name] is "Cary Norris"
    * remember it as the cary
    Given the school
    * find item in list [Students] with property [name] is "Coy Nelson"
    * remember it as the coy
    Given the school
    * create slice of [Students] from co with 3 with search_term C in search_field name
    * remember it as the first_list
    Given the first_list
    * get list of slice
    * for each item in slice
    Given the first_list
    * get count
    Then it is equal to 3
    Given the first_list
     * get first
     * get property value [name]
    Then it is equal to "Chloe Watson"
```

Step 10: Resmove Item from list:

Removing is the opposite of adding/removing.

When you remove an item from a list of links, it is not in that list anymore.

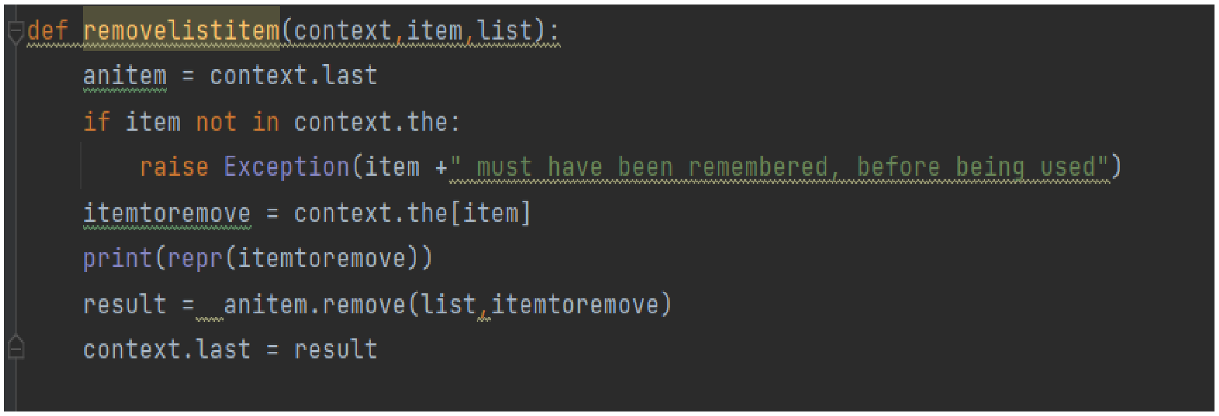
However, the item still exists in the original list of items it was created on.

For example, say you already have some students in the datastore.

Code:

```
school = start.schools.first()

Brion = school.student_council.find("Brion Addison")

# Remove an item
school.student_council.remove(Brion)
```

Step definition to remove an item from list:



```
def removelistitem(context,item,list):
    anitem = context.last
    if item not in context.the:
        raise Exception(item +" must have been remembered, before being used")
    itemtoremove = context.the[item]
    print(repr(itemtoremove))
    result =  anitem.remove(list,itemtoremove)
    context.last = result
```

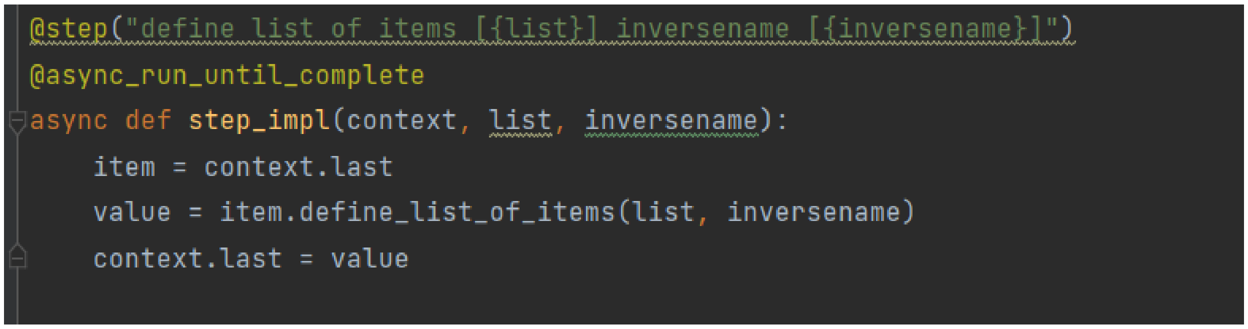
Step 12 : Create a list of items :

The list of students is a list of items, so here is how to create a list of items and adding 3 items to it:

Code :

```
import noapi

async def main():

    start = noapi.datastore("test-school")
    school = start.schools.first()

    school.define_list_of_items("students")
    students = school.students

    a_student = students.create("Jane Doe")
    students.define_property("age","num")
    a_student.age = 24

    b_student = students.create("Benjamin Castaldi")
    b_student.age = 53

    c_student = students.create("Roger Ferrand")
    c_student.age = 20

    noapi.disconnect()

if __name__ == "__main__":
    noapi.run(main())
```

Step definition to create a list of items :



```python
@step("define list of items [{list}] inversename [{inversename}]")
@async_run_until_complete
async def step_impl(context, list, inversename):
    item = context.last
    value = item.define_list_of_items(list, inversename)
    context.last = value
```

Step 13: create a list of link:

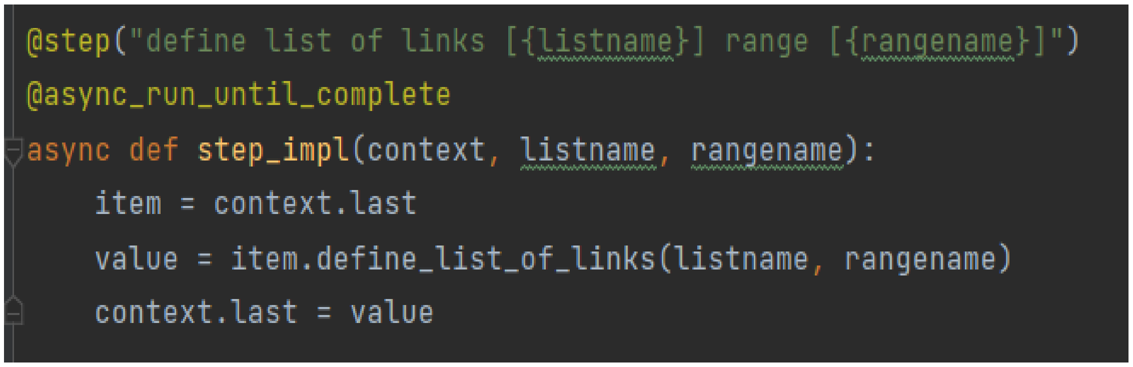
the"`student_council_members`" is a list of links, so here is how to implement it:

Code:

```python
import noapi

async def main():

    start = noapi.datastore("test-school")
    school = start.schools.first()


school.define_list("student_council_members","students","in_school_as_student_council_members")

    student_council_members = school.student_council_members
    jane = students.find("Jane Doe")
    roger = students.find("Roger Ferrand")
```

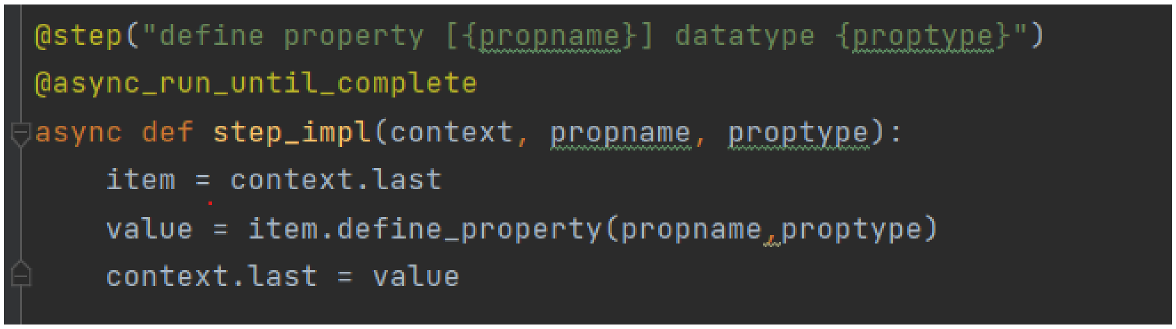
```
    student_council_members.append(jane)
    student_council_members.append(roger)

    noapi.disconnect()

if __name__ == "__main__":
    noapi.run(main())
```

Step definition to create a list of link:

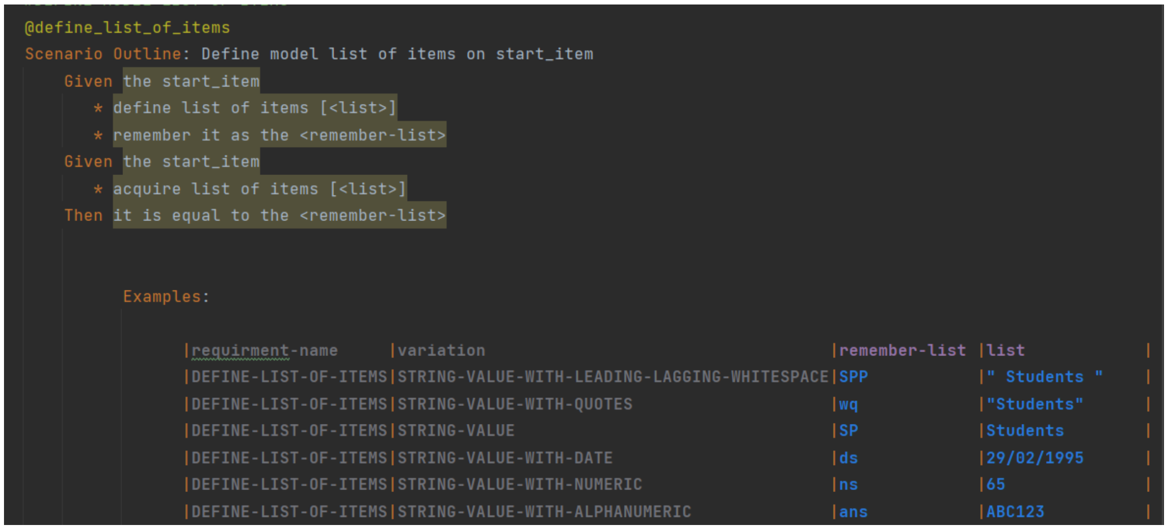


```
@step("define list of links [{listname}] range [{rangename}]")
@async_run_until_complete
async def step_impl(context, listname, rangename):
    item = context.last
    value = item.define_list_of_links(listname, rangename)
    context.last = value
```

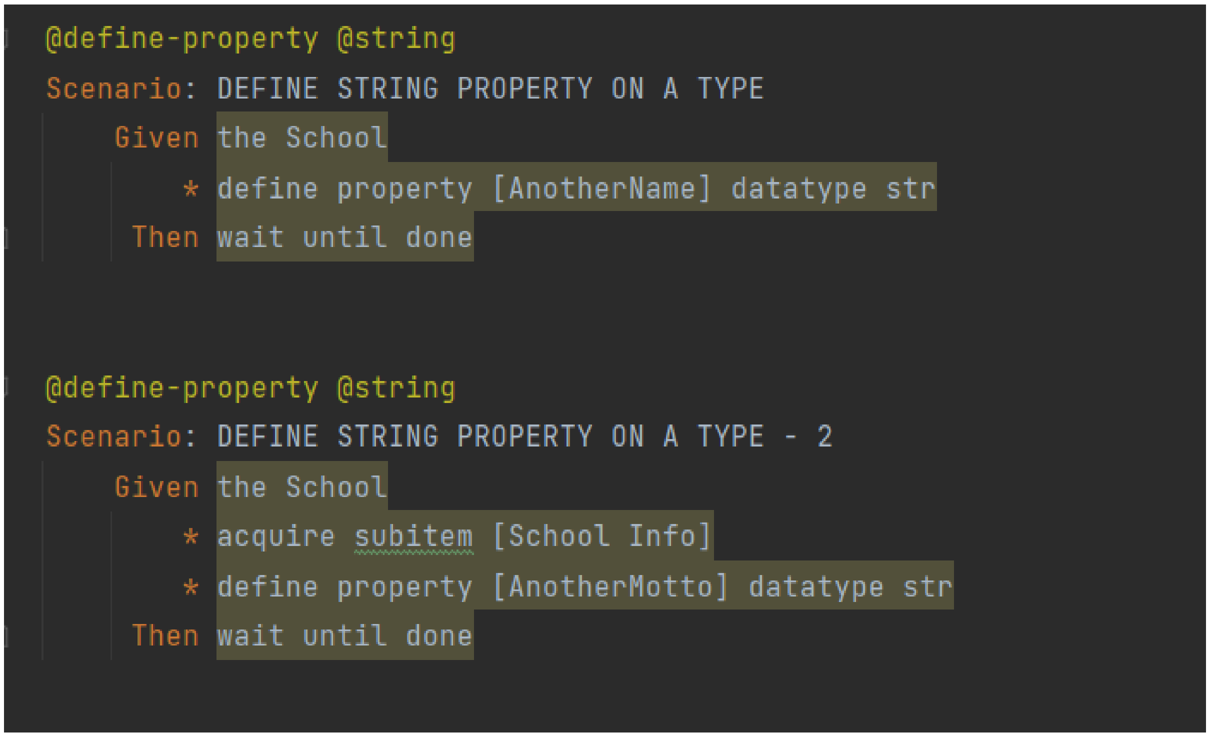
Step definitions to define properties for the lists :


```python
@step("define property [{propname}] datatype {proptype}")
@async_run_until_complete
async def step_impl(context, propname, proptype):
    item = context.last
    value = item.define_property(propname,proptype)
    context.last = value
```

Scenario to define a lists:


```gherkin
@define_list_of_items
Scenario Outline: Define model list of items on start_item
    Given the start_item
       * define list of items [<list>]
       * remember it as the <remember-list>
    Given the start_item
       * acquire list of items [<list>]
    Then it is equal to the <remember-list>


          Examples:

                |requirment-name     |variation                                  |remember-list |list          |
                |DEFINE-LIST-OF-ITEMS|STRING-VALUE-WITH-LEADING-LAGGING-WHITESPACE|SPP           |" Students "  |
                |DEFINE-LIST-OF-ITEMS|STRING-VALUE-WITH-QUOTES                   |wq            |"Students"    |
                |DEFINE-LIST-OF-ITEMS|STRING-VALUE                               |SP            |Students      |
                |DEFINE-LIST-OF-ITEMS|STRING-VALUE-WITH-DATE                     |ds            |29/02/1995    |
                |DEFINE-LIST-OF-ITEMS|STRING-VALUE-WITH-NUMERIC                  |ns            |65            |
                |DEFINE-LIST-OF-ITEMS|STRING-VALUE-WITH-ALPHANUMERIC             |ans           |ABC123        |
```

Scenario to define properties for the lists:



```
@define-property @string
Scenario: DEFINE STRING PROPERTY ON A TYPE
    Given the School
        * define property [AnotherName] datatype str
     Then wait until done


@define-property @string
Scenario: DEFINE STRING PROPERTY ON A TYPE - 2
    Given the School
        * acquire subitem [School Info]
        * define property [AnotherMotto] datatype str
     Then wait until done
```

Calculated Links and Lists (queries) :

Introduction :

These are the lists that are not stored but are calculated and then stored by the use of expressions including system defined and user defined functions.

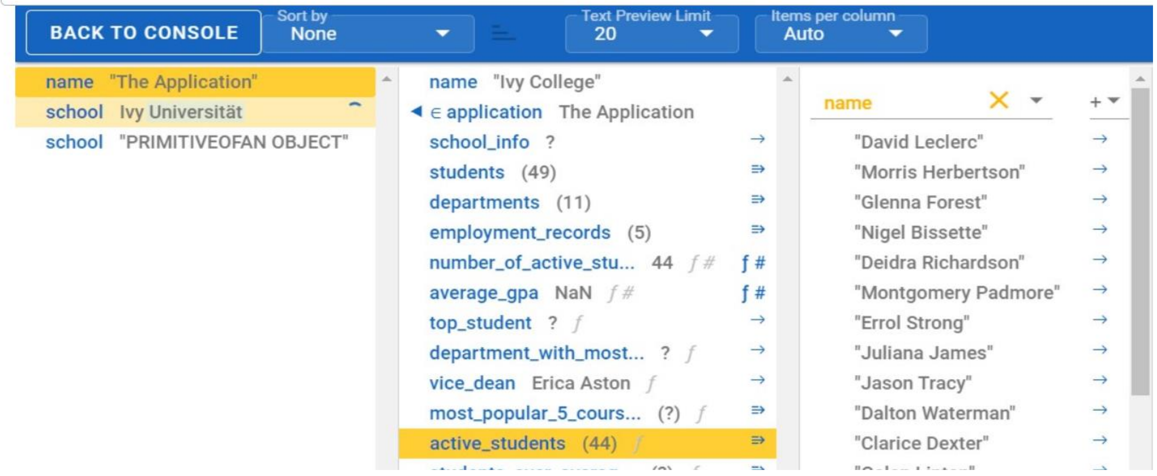
Code structure :

`define_calculated_list(name, range, formula, inversename=None)`

System defined function: ".define_calculated_list()" Blue: calculated list name Yellow: Range of the list Green: function to be performed Red: formula

Example :

```
school = start.schools.first()
school.define_calculated_list('active_students', 'students', 'filter(students, students.active)')
```



Sort by None
Text Preview Limit 20
Items per column Auto

name "The Application"
school Ivy Universität
school "PRIMITIVEOFAN OBJECT"

name "Ivy College"
∈ application The Application
school_info ?
students (49)
departments (11)
employment_records (5)
number_of_active_stu... 44
average_gpa NaN
top_student ?
department_with_most... ?
vice_dean Erica Aston
most_popular_5_cours... (?)
active_students (44)

name
"David Leclerc"
"Morris Herbertson"
"Glenna Forest"
"Nigel Bissette"
"Deidra Richardson"
"Montgomery Padmore"
"Errol Strong"
"Juliana James"
"Jason Tracy"
"Dalton Waterman"
"Clarice Dexter"

if() :

This command will make 'course_hour' list on the range of 'course_modes' which has a property that takes two values 'remote' and 'onsite' for each course. It would check if the 'course_modes' is 'remote', then it will allocate '2' hours for the subject and if it is a 'theory', it will be allocated 3 hours

Example :

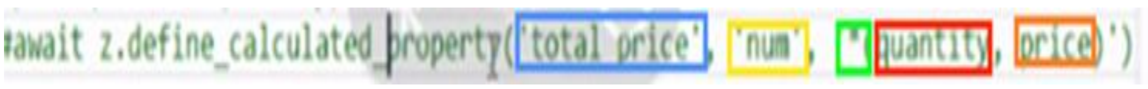
```
school = start.schools.first()
school.define_calculated_list('course_hour','course_modes','if(course_modes.remote, "2", "3")')
```

Define calculated properties :

Introduction:

They are properties that are not stored properties but are calculated and then stored by the use of expressions including system defined and user defined functions. For example Let’s say you have “students.age” defined as a calculated property which is to be calculated from “student.date_of_birth”.

Code source :



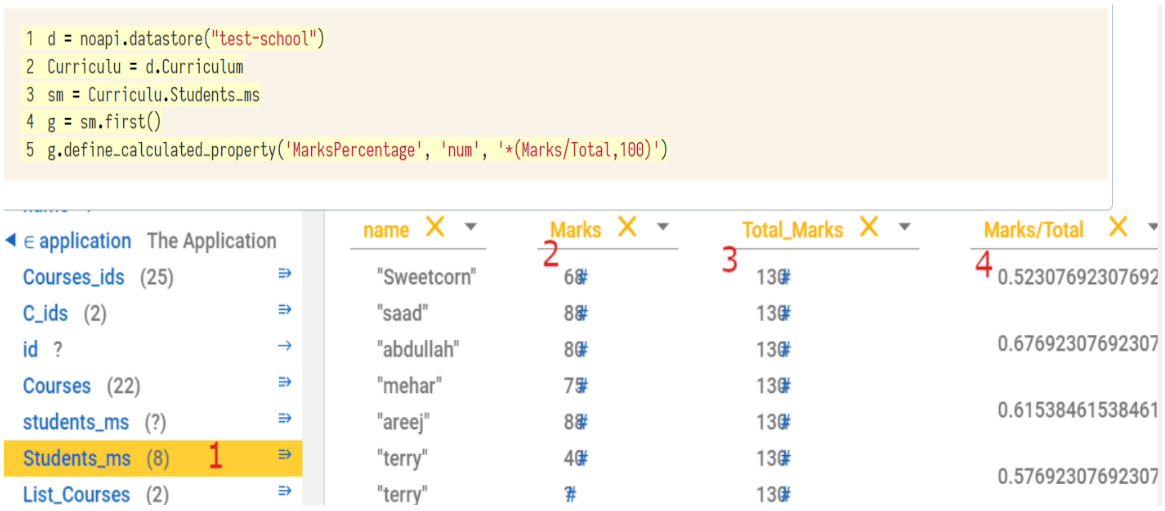
```
#await z.define_calculated_property('total_price', 'num', '*(quantity, price)')
```

- System defined function: `.define_calculated_property()`.
- Blue: calculated property name.
- Yellow: Type of the object that would be stored inside the calculated property list.
- Green: operation to be performed between two properties.

- Red: Property 1.
- Orange: Property 2.

Example :

By using operators like *(+, *, /, -)* multiple operations can be performed to create a calculated property. If you want to create a calculated property from a property of a list, You should execute the following code block. It will take the two properties quantity and price. Multiply them and then pass the result in the third newly calculated property "total_price".



```
d = noapi.datastore("test-school")
Curriculu = d.Curriculum
sm = Curriculu.Students_ms
g = sm.first()
g.define_calculated_property('MarksPercentage', 'num', '*(Marks/Total,100)')
```

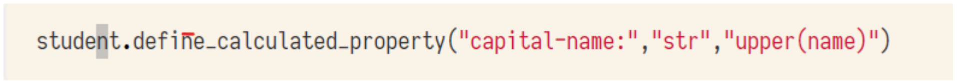
∈ application	The Application
Courses_ids	(25)
C_ids	(2)
id	?
Courses	(22)
students_ms	(?)
Students_ms	(8) 1
List_Courses	(2)

name	2 Marks	3 Total_Marks	4 Marks/Total
"Sweetcorn"	68#	130#	0.52307692307692
"saad"	88#	130#	
"abdullah"	80#	130#	0.67692307692307
"mehar"	75#	130#	
"areej"	88#	130#	0.61538461538461
"terry"	40#	130#	
"terry"	?#	130#	0.57692307692307

Upper ():

- For example, convert a student's name from lowercase to uppercase.

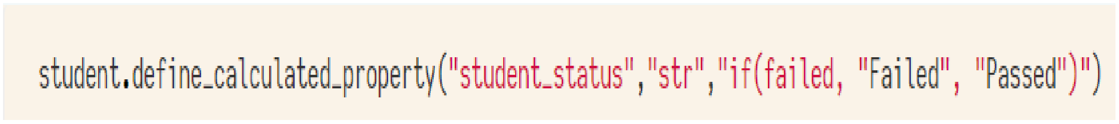
- It is yet to be implemented or is not currently functional.



```
student.define_calculated_property("capital-name:","str","upper(name)")
```

if (bol, str1, str2)

Doing query according to the student status :

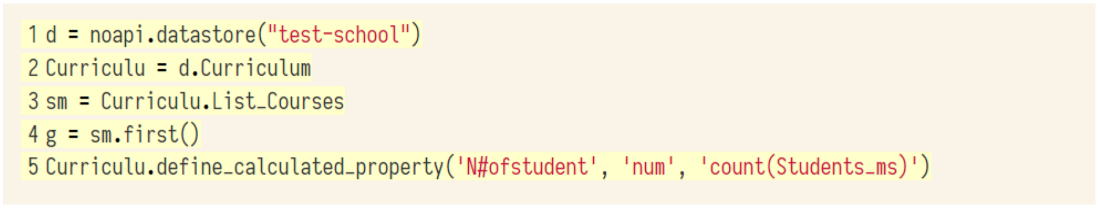


```
student.define_calculated_property("student_status","str","if(failed, "Failed", "Passed")")
```

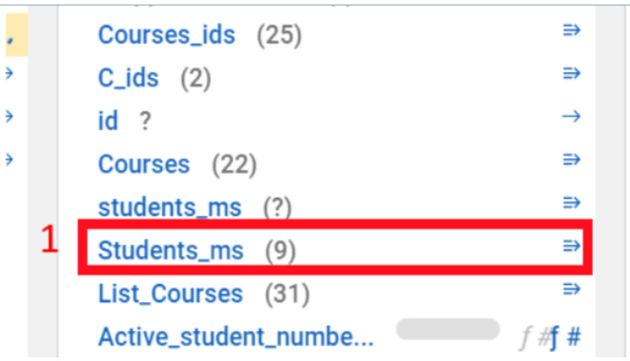
count () :

- to count school students you have to call the list of students from school hence 'school.students'

Example Code :



```
d = noapi.datastore("test-school")
Curriculu = d.Curriculum
sm = Curriculu.List_Courses
g = sm.first()
Curriculu.define_calculated_property('N#ofstudent', 'num', 'count(Students_ms)')
```



```
Courses_ids   (25)
C_ids   (2)
id   ?
Courses   (22)
students_ms   (?)
Students_ms   (9)
List_Courses   (31)
Active_student_numbe...
```

Allure Framwork

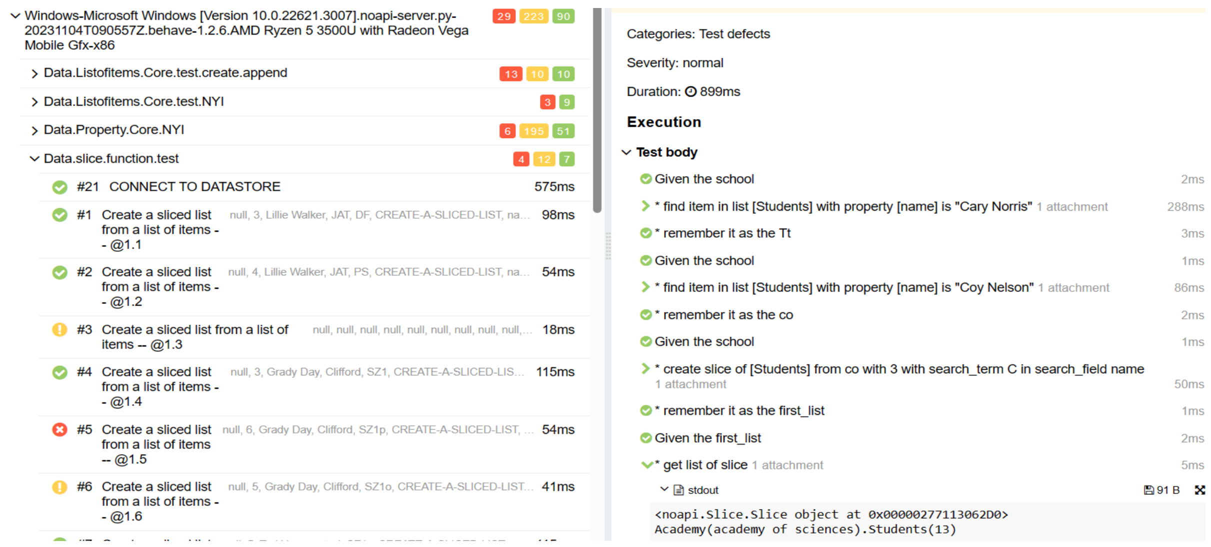
Introduction:

Allure is a test report framework that provides detailed and visually appealing reports for various test frameworks, including Cucumber. When integrating Allure with Cucumber, it generates comprehensive and interactive HTML reports that enhance the visibility of test results.

Key features of Allure results for Cucumber:

- **Rich Visualization:** Allure provides visually appealing reports with charts, graphs, and detailed information, making it easier to understand and interpret test results.
- **Historical Trends:** It allows you to track historical trends in test execution, helping identify patterns or trends in test success or failure over time.
- **Screenshots and Attachments:** Allure supports the embedding of screenshots and other attachments in the reports, providing additional context to aid in debugging and analysis.
- **Categorization and Labeling:** Test results can be categorized and labeled, making it simple to filter and navigate through different aspects of the test suite.

Example of Allure Reports for Noapi Functions:



```
Windows-Microsoft Windows [Version 10.0.22621.3007].noapi-server.py-      29 223 90
20231104T090557Z.behave-1.2.6.AMD Ryzen 5 3500U with Radeon Vega
Mobile Gfx-x86
  Data.Listofitems.Core.test.create.append                                13 10 10
  Data.Listofitems.Core.test.NYI                                          3 9
  Data.Property.Core.NYI                                                  6 195 51
  Data.slice.function.test                                                4 12 7
    #21  CONNECT TO DATASTORE                                             575ms
    #1   Create a sliced list   null, 3, Lillie Walker, JAT, DF, CREATE-A-SLICED-LIST, na...   98ms
         from a list of items -
         - @1.1
    #2   Create a sliced list   null, 4, Lillie Walker, JAT, PS, CREATE-A-SLICED-LIST, na...   54ms
         from a list of items -
         - @1.2
    #3   Create a sliced list from a list of   null, null, null, null, null, null, null, null, null,...   18ms
         items -- @1.3
    #4   Create a sliced list   null, 3, Grady Day, Clifford, SZ1, CREATE-A-SLICED-LIS...   115ms
         from a list of items -
         - @1.4
    #5   Create a sliced list   null, 6, Grady Day, Clifford, SZ1p, CREATE-A-SLICED-LIST, ...   54ms
         from a list of items
         -- @1.5
    #6   Create a sliced list   null, 5, Grady Day, Clifford, SZ1o, CREATE-A-SLICED-LIST...   41ms
         from a list of items -
         - @1.6
```

```
Categories: Test defects
Severity: normal
Duration: 899ms
Execution
Test body
  Given the school                                                                              2ms
  * find item in list [Students] with property [name] is "Cary Norris" 1 attachment          288ms
  * remember it as the Tt                                                                       3ms
  Given the school                                                                              1ms
  * find item in list [Students] with property [name] is "Coy Nelson" 1 attachment            86ms
  * remember it as the co                                                                       2ms
  Given the school                                                                              1ms
  * create slice of [Students] from co with 3 with search_term C in search_field name
    1 attachment                                                                               50ms
  * remember it as the first_list                                                               1ms
  Given the first_list                                                                          2ms
  * get list of slice 1 attachment                                                              5ms
    stdout                                                                               91 B
      <noapi.Slice.Slice object at 0x00000277113062D0>
      Academy(academy of sciences).Students(13)
```

5.3 Project Name: challenge web application

Introduction:

Challenge web application is a platform for software students to upload their projects or applications to participate in challenge.

In this project i worked only on the backend side :

- Database design
- Creating the Entities and fixing the relations between them

- Doing authentication
- Email verification for the users
- Phone verification for users
- Controllers and Apis
- Testing with Postman

5.3.1 Database design:

Database design is a critical aspect of building a robust web application, influencing its performance, scalability, and data integrity. It involves organizing and structuring data to efficiently store, manage, and retrieve information. A well-designed database ensures data consistency, minimizes redundancy, and supports the application's functionality.

5.3.2 Creating the Entities and fixing the relations between them

In web development, creating entities and establishing relationships between them is a crucial step in the database design process. This step is particularly relevant when using an Object-Relational Mapping (ORM) framework like Hibernate (Java), Entity Framework (.NET), or Sequelize (Node.js), where entities map to database tables.

- **Database Schema:**
    - **Table Creation:** Each entity typically corresponds to a table in the database. Define the structure of these tables, specifying the data types and constraints for each attribute.
    - **Primary Keys:** Identify the primary key for each table, a unique identifier for each record.
    - **Foreign Keys:** If entities have relationships, define foreign keys to establish connections between tables.
- **Relationships:**

- **One-to-One:** One record in one table is related to one record in another table. For example, a **User** might have one **Profile**.
- **One-to-Many:** One record in one table is related to multiple records in another table. For instance, one **User** can have multiple **Orders**.
- **Many-to-Many:** Many records in one table are related to many records in another table, often requiring an intermediate table to represent the relationship.

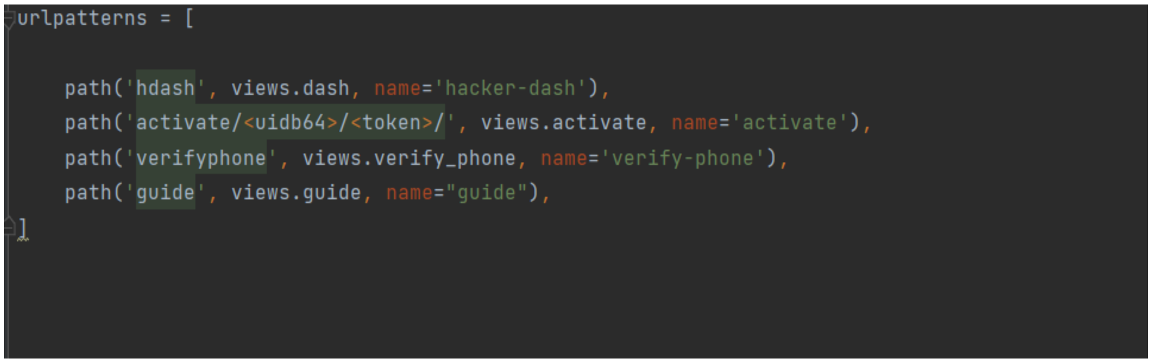
Code Example with django:

```
from django.db import models

class Author(models.Model):

    name = models.CharField(max_length=100)

    email = models.EmailField(unique=True)

    def __str__(self):

        return self.name

class Post(models.Model):

    title = models.CharField(max_length=200)

    content = models.TextField()

    pub_date = models.DateTimeField(auto_now_add=True)

    author = models.ForeignKey(Author, on_delete=models.CASCADE, related_name='posts')

    def __str__(self):

        return self.title
```

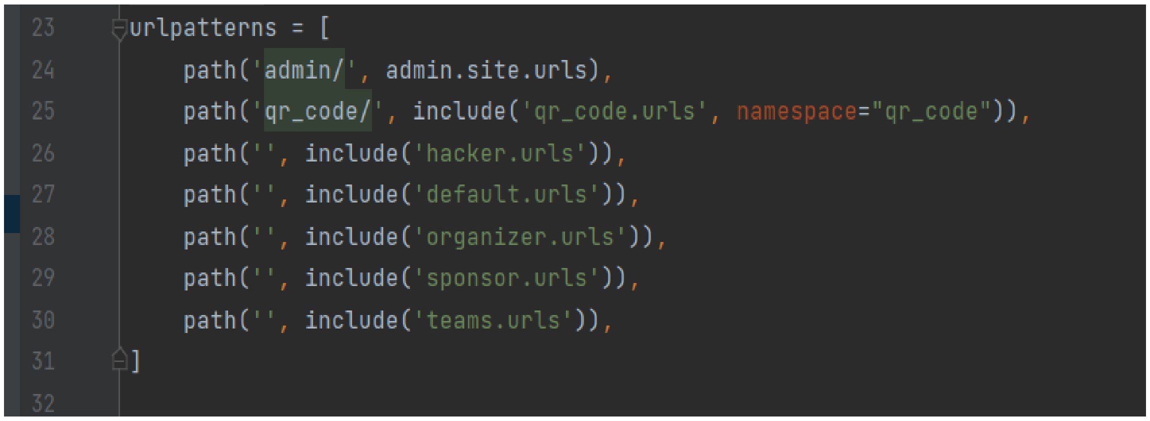
5.3.3 Authentication:

Django provides built-in authentication with features like user registration, login, and password reset. To use it, include 'django.contrib.auth' in your INSTALLED_APPS and configure urls.py with path('accounts/', include('django.contrib.auth.urls')). Leverage the @login_required decorator for view authentication.

Example of Url.py file:



```
urlpatterns = [

    path('hdash', views.dash, name='hacker-dash'),
    path('activate/<uidb64>/<token>/', views.activate, name='activate'),
    path('verifyphone', views.verify_phone, name='verify-phone'),
    path('guide', views.guide, name="guide"),
]
```

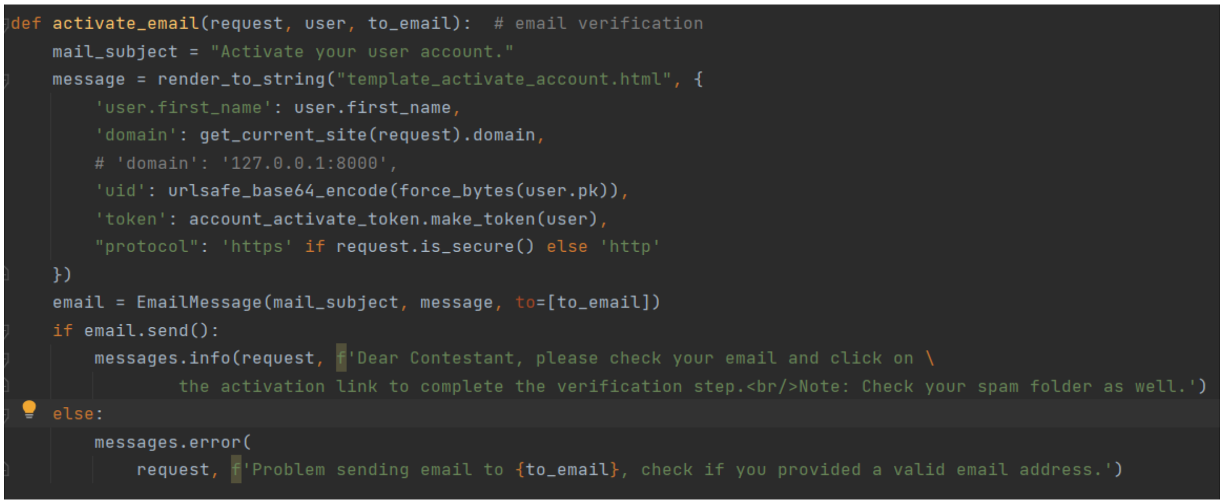


```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('qr_code/', include('qr_code.urls', namespace="qr_code")),
    path('', include('hacker.urls')),
    path('', include('default.urls')),
    path('', include('organizer.urls')),
    path('', include('sponsor.urls')),
    path('', include('teams.urls')),
]
```

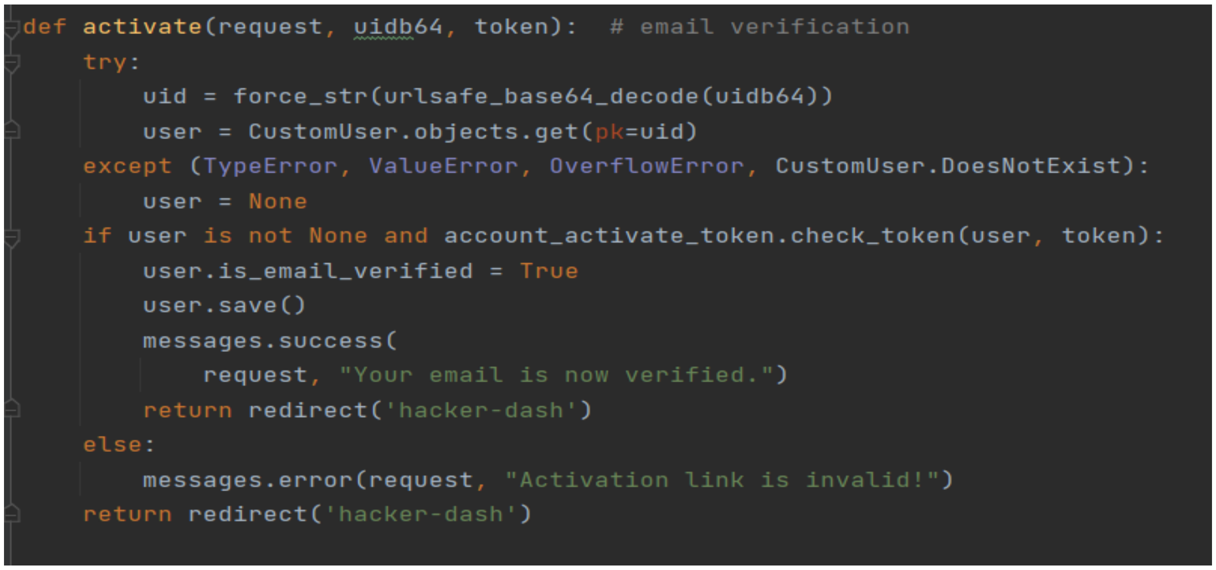
5.3.4 Email verification for the users:

Email verification in Django is a process that ensures users provide a valid email address and confirms ownership before gaining access to certain functionalities. By implementing email verification, developers enhance security and reduce the likelihood of fake or unauthorized user accounts. The popular Django package **django-allauth** simplifies this process, offering comprehensive features for user authentication, registration, and mandatory email verification, contributing to a more secure and reliable web application.

Code Example:



```
def activate_email(request, user, to_email):  # email verification
    mail_subject = "Activate your user account."
    message = render_to_string("template_activate_account.html", {
        'user.first_name': user.first_name,
        'domain': get_current_site(request).domain,
        # 'domain': '127.0.0.1:8000',
        'uid': urlsafe_base64_encode(force_bytes(user.pk)),
        'token': account_activate_token.make_token(user),
        "protocol": 'https' if request.is_secure() else 'http'
    })
    email = EmailMessage(mail_subject, message, to=[to_email])
    if email.send():
        messages.info(request, f'Dear Contestant, please check your email and click on \
                the activation link to complete the verification step.<br/>Note: Check your spam folder as well.')
    else:
        messages.error(
            request, f'Problem sending email to {to_email}, check if you provided a valid email address.')
```

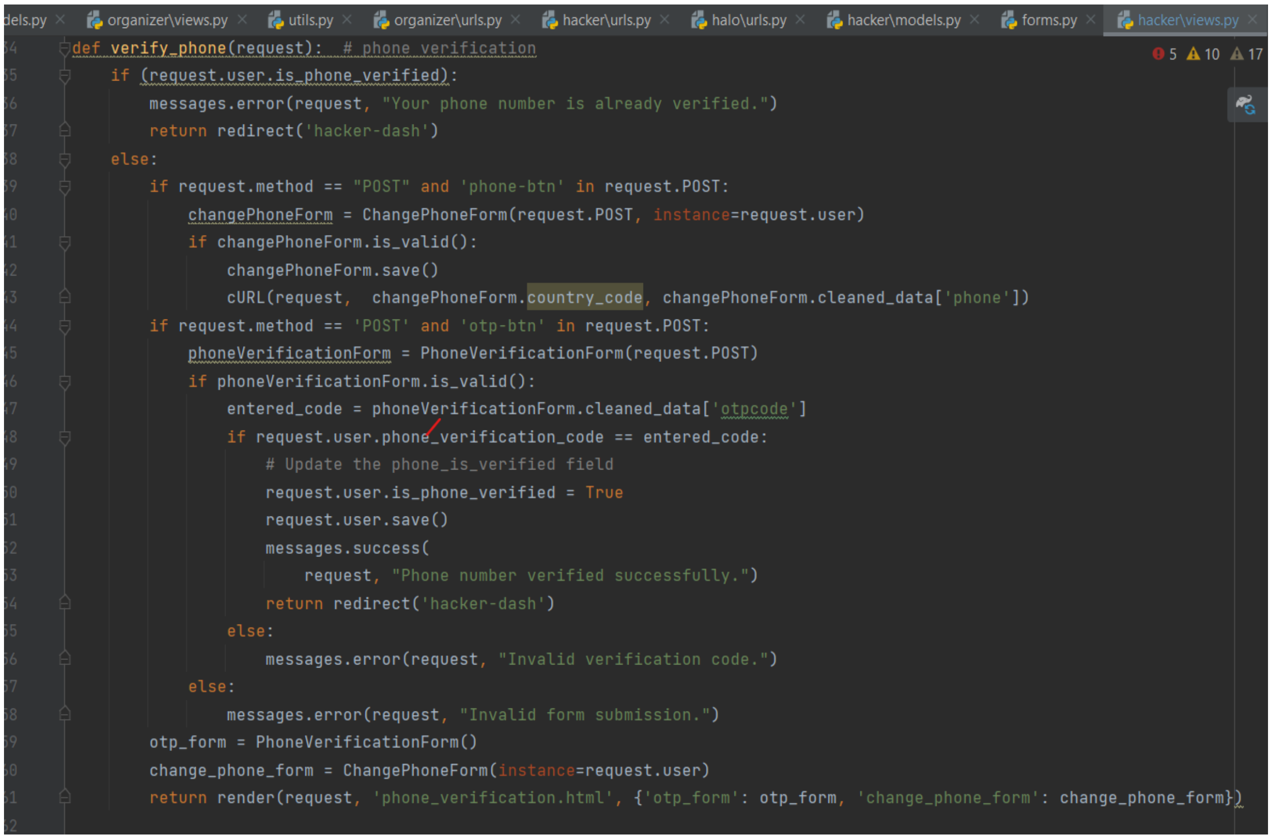


```
def activate(request, uidb64, token):  # email verification
    try:
        uid = force_str(urlsafe_base64_decode(uidb64))
        user = CustomUser.objects.get(pk=uid)
    except (TypeError, ValueError, OverflowError, CustomUser.DoesNotExist):
        user = None
    if user is not None and account_activate_token.check_token(user, token):
        user.is_email_verified = True
        user.save()
        messages.success(
            request, "Your email is now verified.")
        return redirect('hacker-dash')
    else:
        messages.error(request, "Activation link is invalid!")
    return redirect('hacker-dash')
```

5.3.5 Phone verification for users:

Phone verification for users is a security measure employed in web applications to confirm the authenticity of users' phone numbers. This process involves sending a unique code to the user's provided phone number, which they must input to verify their identity. By implementing phone verification, web developers enhance account security, reduce the risk of fraudulent activities, and ensure that users have valid and reachable contact information. This authentication method is commonly used to bolster user registration, password recovery, and other critical workflows, fostering trust and reliability in the application.

Code source :



```python
def verify_phone(request):  # phone verification
    if (request.user.is_phone_verified):
        messages.error(request, "Your phone number is already verified.")
        return redirect('hacker-dash')
    else:
        if request.method == "POST" and 'phone-btn' in request.POST:
            changePhoneForm = ChangePhoneForm(request.POST, instance=request.user)
            if changePhoneForm.is_valid():
                changePhoneForm.save()
                cURL(request,  changePhoneForm.country_code, changePhoneForm.cleaned_data['phone'])
        if request.method == 'POST' and 'otp-btn' in request.POST:
            phoneVerificationForm = PhoneVerificationForm(request.POST)
            if phoneVerificationForm.is_valid():
                entered_code = phoneVerificationForm.cleaned_data['otpcode']
                if request.user.phone_verification_code == entered_code:
                    # Update the phone_is_verified field
                    request.user.is_phone_verified = True
                    request.user.save()
                    messages.success(
                        request, "Phone number verified successfully.")
                    return redirect('hacker-dash')
                else:
                    messages.error(request, "Invalid verification code.")
            else:
                messages.error(request, "Invalid form submission.")
        otp_form = PhoneVerificationForm()
        change_phone_form = ChangePhoneForm(instance=request.user)
        return render(request, 'phone_verification.html', {'otp_form': otp_form, 'change_phone_form': change_phone_form})
```

5.3.6 Controllers and apis:

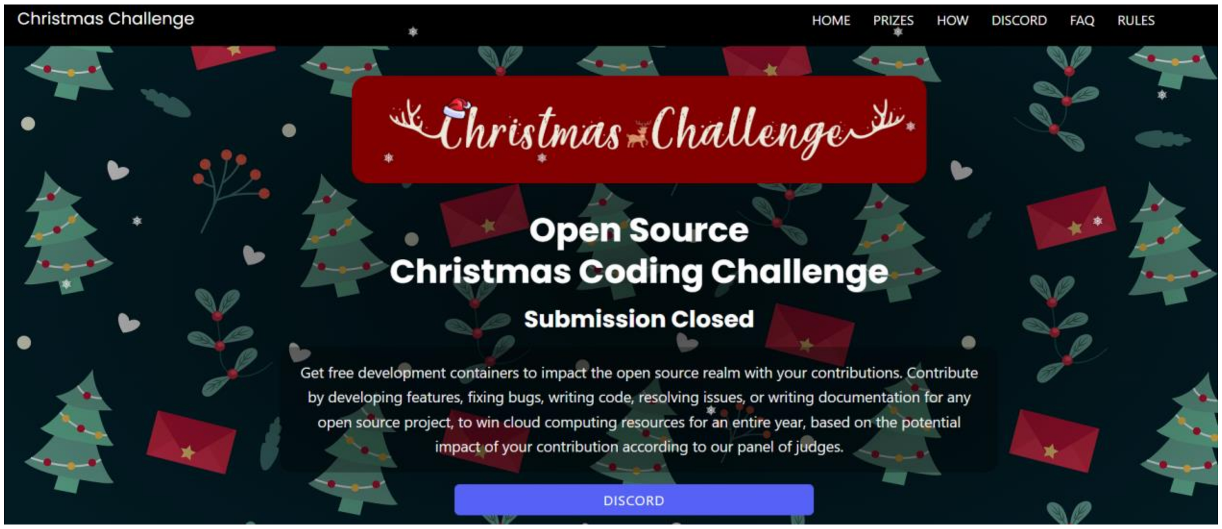
In web development, controllers and APIs play pivotal roles in handling and managing application logic, data flow, and communication between the frontend and backend.

Controllers: Controllers are components within the backend of a web application responsible for processing user inputs, managing data, and determining the appropriate response. They bridge the gap between the user interface and the underlying business logic, handling requests from the client and ensuring the proper execution of corresponding actions. In the Model-View-Controller (MVC) architecture, controllers serve as the central point for directing and coordinating the flow of data.

APIs (Application Programming Interfaces): APIs are interfaces that enable different software components to communicate with each other. In the context of web development, APIs often refer to sets of rules and protocols that allow the frontend (client-side) and backend (server-side) of an application to exchange data. Web APIs, commonly implemented using REST or GraphQL, provide a standardized way for developers to request and manipulate data. They are instrumental in building scalable and modular applications, allowing different parts of a system to interact seamlessly.

Together, controllers and APIs facilitate the development of dynamic and interactive web applications by managing the flow of data, responding to user actions, and enabling communication between frontend and backend components. Controllers handle the application's internal logic, while APIs define how different software components can interact, promoting modularity and interoperability in web development.

The user interface of the challenge website:



Open Source
Christmas Coding Challenge
Submission Closed

Get free development containers to impact the open source realm with your contributions. Contribute by developing features, fixing bugs, writing code, resolving issues, or writing documentation for any open source project, to win cloud computing resources for an entire year, based on the potential impact of your contribution according to our panel of judges.

Conclusion :

During my internship in web development, I successfully navigated through crucial aspects of the development lifecycle. I started by meticulously designing the database, creating entities, and establishing relationships to ensure efficient data storage and retrieval. Implementation of authentication mechanisms, including both email and phone verification, fortified the application's security and user validation processes.

Delving into the realm of controllers and APIs, I honed my skills in crafting backend logic, orchestrating data flow, and creating interfaces for seamless communication between frontend and backend components. This experience not only solidified my understanding of the Model-View-Controller (MVC) architecture but also exposed me to the significance of well-designed Application Programming Interfaces (APIs) in promoting modular and scalable development.

Furthermore, my exploration extended to testing the application functionalities using Postman, allowing me to verify the reliability and efficiency of the implemented features. This hands-on experience with testing tools enriched my understanding of quality assurance practices in web development.

Overall, this internship provided me with a holistic view of the web development process, from conceptualizing and designing databases to implementing robust authentication, verification mechanisms, and developing backend logic through controllers and APIs. The practical exposure gained during this period has significantly contributed to my proficiency in building secure, scalable, and well-tested web applications.

Conclusion :

During my internship, I immersed myself in the realm of automation testing, focusing on the Noapi SDK using Cucumber with Python. My primary responsibilities included thoroughly testing essential functions such as move, slice, find, append, prepend, remove, create, and destroy, meticulously crafted by the development team. This process involved writing comprehensive step definitions and scenarios to ensure the functionality, correctness, and robustness of the SDK.

By delving into automation testing, I contributed to enhancing the reliability and efficiency of the Noapi SDK, providing valuable insights into potential bugs and ensuring the seamless integration of new features. The utilization of Cucumber with Python allowed me to create clear, expressive scenarios and step definitions, fostering collaboration between the development and testing phases.

The incorporation of Allure reports further enriched my testing practices, providing a visually appealing and detailed representation of test results. This not only facilitated effective communication with the development team but also offered a comprehensive overview of the SDK's testing landscape.

In conclusion, my internship in automation testing for the Noapi SDK equipped me with practical skills in crafting robust test scenarios, step definitions, and utilizing reporting tools like Allure. This experience significantly deepened my understanding of the importance of meticulous testing in the software development lifecycle and my ability to contribute to the creation of reliable and high-quality SDKs.

EXPERIENCE AND ACHIEVEMENTS:

My experience in development of complex solutions following customers-implementation of concepts for data security, corporate and virtual private networks, databases and software development are improving.

6.1 Overcome Problems and Difficulties :

I have faced many problems and errors while developing those projects. I got the support and help from the team and they helps me much to solve the problems. I have faced many problems while I working with database, I makes many mistakes in the naming convention. By following proper guideline I overcame the problems.

6.2Working Practices :

I excelled in web development, demonstrating proficiency in database design, authentication, and verification. In automation testing for the Noapi SDK, I successfully tested key functions, wrote clear step definitions, and utilized Allure reports for comprehensive result analysis, contributing to the SDK's reliability.

6.2.2Testing :

I had a good experience with testing the web applications before, in my internship I learnt more a bout automation test and become good to find test cases and use cucumber to write step definitions and scenarios to test Noapi SDK.

6.3Technological Enhancement :

At this developing period I have familiar with lot of new tools and have to use them. I never do any project into a cross platform before. It’s a new experience for me to work in a industry with new tools. I also learn to use many tools and learn many new technology.

6.4Non-Technical Growth :

Beyond technical skills, my internships fostered significant non-technical growth by enhancing my communication, teamwork, and problem-solving abilities. Engaging with diverse teams, writing comprehensive documentation, and presenting findings improved my capacity to contribute effectively to collaborative projects.

I have also do my responsibility into client's interaction and project requirement. I also go with the sales team to deliver and sitting up a software. I have learning the official behaviour and how to abide them.

6.5Achievement :

I choose internship for the learning purpose and to get the flavour of industry for my career. For the permission of my university and helps of some people I am able to manage a internship and started work as a developer. As a developer internee I have learn the official rule and the pattern of developing a project following by the professional strategy and rules. I do testing, developing and go with the sales team to deliver a software. Those experience are my achievement and I determined that those experience helps me for the future career.

CONCLUSIONS

My responsibility as a intern is to develop web sites and web applications. I am going to complete my internship program successfully. As a intern I have got opportunity to work with experienced team and I have learn a lot from the team. At first I have get the knowledge about how to work with a team and how to divide the task and completed the responsibility. I also learn many important point to be a full stack developer. I have lot of limitations and very little knowledge about developing. Now I am able to fix my limitations and I have learn how to work on a cross platform.

Throughout my web development internship, I acquired a solid foundation in database design, authentication, and verification processes, showcasing my ability to build secure and efficient web applications. Subsequently, my automation testing internship for the Noapi SDK equipped me with advanced skills in crafting robust test scenarios, step definitions, and utilizing reporting tools like Allure, enhancing the reliability and functionality of the SDK. Collectively, these experiences have not only expanded my technical prowess but also honed my collaborative and problem-solving skills, laying a strong foundation for my future contributions in the field