1. API Design and API Test Design:

Tasks:

- Draw state-transition diagram,
- Design REST API,
- Test Design: Write Test Cases and Test Scenario(s) for the API.

You are given a System (SUT – System Under Test) Back-end with HTTP API interface.
SUTs states: ”Log in”, “Dashboard”, “Config”, “Item Details”, “Item Control”.
There are two groups of API requests:

- Requests to change the state of the system,
- Requests to operate with the system.

At the beginning the SUT stays in its initial state – “LogIn” and doesn’t allow any operational request. On a successful login SUT changes its state to “Dashboard”,
When the system is in “Dashboard” state the following operations are allowed:

- Return a list of items, following filter parameters can be applied:
    - items can be paginated,
    - sorted by date,
    - filtered by item processing state (run, pause, stop);
- Switch SUT state to “Config”, “Item Details”, “Item Control”, “Log In”.

When SUT is in “Config” state the following operations are allowed:

- Change/Update Password,
- Change/Update Username,
- Allow/Deny “Item Control” state,
- Return to Dashboard state.

  If Username or password is updated or access to “Item Control” is changed the system goes to “LogIn” state automatically.

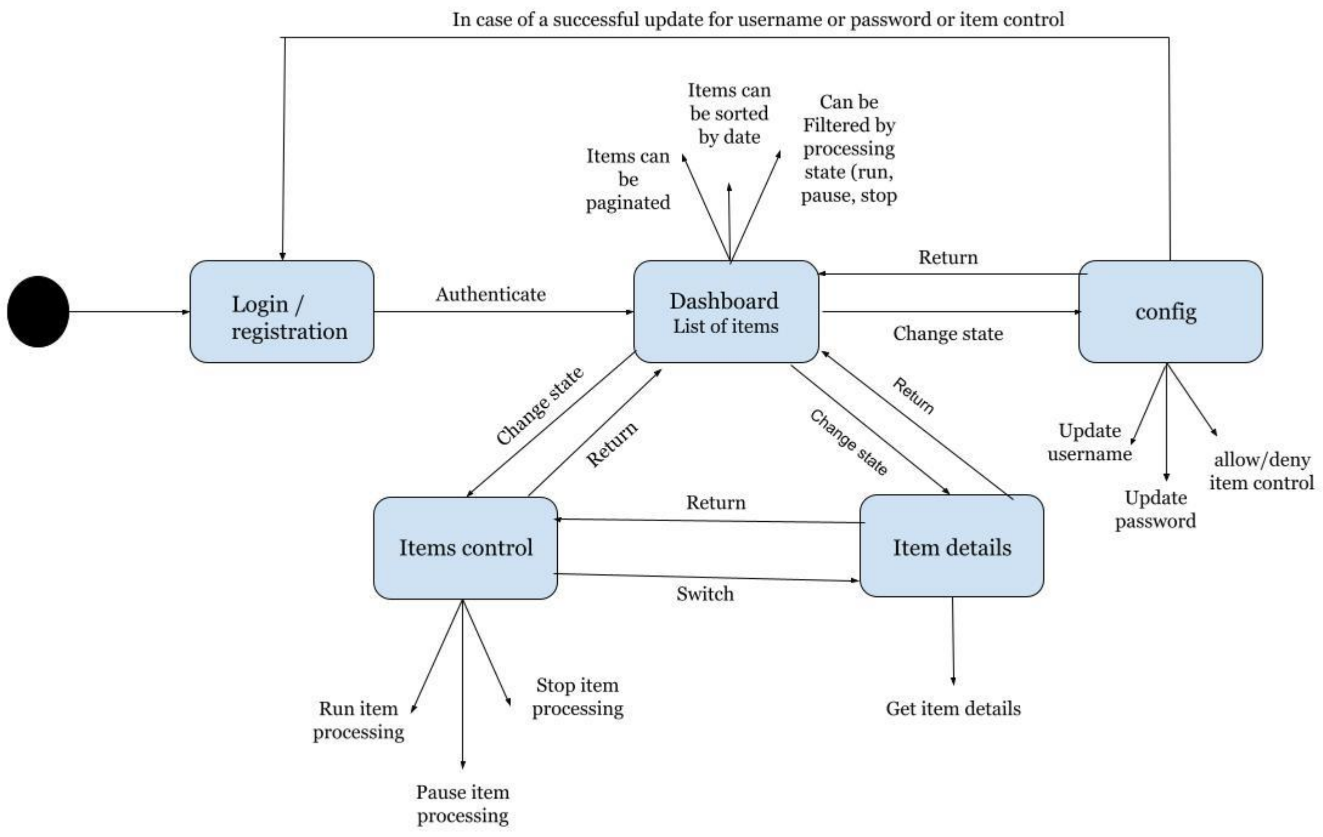
When SUT is in “Item Details” state the following operations are allowed:

- Get Item Details,
- Return to “Dashboard” state,
- Switch to “Item Control” state,

When SUT is in “Item Control” state the following operations are allowed:

- Run Item processing,
- Pause Item processing,
- Stop Item processing,
- Return to “Dashboard” state,
- Switch to “Item Details” state.

State Transition Diagram:



REST API Design:

User Details:

Get user details

URL:

/users/:id/

Method:

GET

URL Params:

Required:

"id"=[integer]

Success Response:

Code: 200
Content: { "id" : 1, “name” : "olivia smith”, “email” : "oliviasmith@gmail.com” , “item_control”: false }

Error Response:

Code: 404 NOT FOUND
Content: { error : "User doesn't exist" }

User Details:

Create new user

URL:

/users/

Method:

Post

URL Params:

Required:

Data Params:

“name” = [string]
“email” = [string]
“password” = [string]
“item_control” = [boolean]

Success Response:

Code: 201
Content: { "id" : 3, “name” : "sarah smith”, “email” : “sarahsmith@gmail.com” , “item_control”: true }

Error Response:

Code: 404 NOT FOUND
Content: { error : "User was not created" }

User Details:

Update user details

URL:

/users/:id/

Method:

Put

URL Params:

Required:

“id”=[integer]

Data Params:

“name” = [string]
“email” = [string]
“item_control” = [boolean]

Success Response:

Code: 200
Content: { "id" : 1, “name” : "Goerge smith”, “email” : "georgesmith@gmail.com”, “item_control”: true }

Error Response:

Code: 404 NOT FOUND
Content: { error : "User was not updated successfully " }

User Details:

Delete user details

URL:

/users/:id/

Method:

Delete

URL Params:

Required:

"id"=[integer]

Success Response:

Code: 200
Content: { "user with id”+ id +“was deleted successfully”}

Error Response:

Code: 404 NOT FOUND
Content: { error : "User was not deleted" }

Items Details

Get item details

URL:

/items/id/

Method:

GET

URL Params:

Required:

id=[integer]

Success Response:

Code: 200
Content: { "id" : 1, “name” : “item 1”, “state” :”run”}

Error Response:

Code: 404 NOT FOUND
Content: { error : "Item doesn't exist" }

Items Details

Create new item

URL:

/items/id/

Method:

Post

URL Params:

Required:

Data Params:

“name” = [string]
“state” = [string]

Success Response:

Code: 201
Content: { "id" : 1, “name” : “item 1”, “state” :”run”}

Error Response:

Code: 404 NOT FOUND
Content: { error : "Item was not created" }

Items Details

Update item details

URL:

/items/:id/

Method:

Put

URL Params:

Required:

“id” = [integer]

Data Params:

“name” =[string]
“state” =[string]

Success Response:

Code: 200
Content: { "id" : 1, “name” : “item 1”, “state” :”run”}

Error Response:

Code: 404 NOT FOUND
Content: { error : "Item was not updated successfully" }

Items List

Get item list

URL:

/items

Method:

Get

URL Params:

Required:

Optional:

“id” =[integr]
“name” = [string]
“limit” = [integr]
“state” = [string]
“page” = [integr]
“sort_by” = [string]
“order_by” = [string]

Success Response:

Code: 200
Content:

```
{“metadata”:
{   "page": 1,
      "limit": 4,
      "page_count": 20,
      "total_count": 80,
      "Links": [
        {"self": "/items?page=5&limit=4"},
        {"first": "/items?page=0&limit=4"},
        {"previous": "/items?page=4&limit=4"},
        {"next": "/items?page=6&limit=4"},
        {"last": "/items?page=20&limit=4"},
}
 “items” : [
    { “id” : 1, “name” : “item 1”, “state”  :”run”  },
    { "id" : 2, “name” : “item 2”, “state”  :”pause” },
    { "id" : 3, “name” : “item 3”, “state”  :”stop” },
    { "id" : 4, “name” : “item 4”, “state” :”run”}
  ]
 }
```

Error Response:

Code: 404 NOT FOUND
Content: { error : "Item list doesn't exist" }

Users List

Get users list

URL:

/users

Method:

Get

URL Params:

Required:

Optional:

“id” [integr]
“name” [integr]
“email” [string]
“limit” [integr]
“page” [integr]
“item_control” [string]
“sort_by” [string]
“order_by” [string]

Success Response:

Code: 200
Content:

```
{“metadata”:
{   "page": 1,
      "limit": 6,
      "page_count": 100,
      "total_count": 600,
      "Links": [
        {"self": "/items?page=10&limit=6"},
        {"first": "/items?page=0&limit=6"},
        {"previous": "/items?page=9&limit=6"},
        {"next": "/items?page=11&limit=6"},
        {"last": "/items?page=600&limit=6"},
}
 “users” : [
    { “id” : 1, “name” : “Mariam Ibrahim”, “item_control”  :”true”  },
    { "id" : 2, “name” : “Olivia George”, “item_control”  :”false” },
    { "id" : 3, “name” : “John Smith”, “item_control”  :”false” },
    { "id" : 4, “name” : “Ben Smith”, “item_control” :”true”}
 ]
}
```

Error Response:

Code: 404 NOT FOUND
Content: { error : "Item list doesn't exist" }

Authentication

URL:

/login

Method:

Post

Data Params:

username [string]
password [string]

Success Response:

Code: 200
Content: - An encoded string in the response body referred as TOKEN -
{ “eyJhbGciOiJIUzI1NiJ9.
eyJleHAiOjE0MTc2NTM1ODgsInN1YiI6IkFsbGVuIiwibmJmIjoxNDE3NjQ5ODY4LCJpc3MiOi
JXLUNTRUlFUk9FLTI5LmFkcHJvZC5ibWMuY29tIiwianRpIjoiSURHQUFCRFVDMllHSUFONkJGUTJBQUFFUEZBNV
FXIiwiX2NhY2hlSWQiOjQ3LCJpYXQiOjE0MTc2NDk5ODh9.
V4LGLcEdwD8V_l4rzoWYYSZmEMA82LBB_lEfz4Xnz9Y” }

Error Response:

Code: 404 NOT FOUND
Content: { error : "wrong username and password combination" }

Test Design:

Test scenarios:

1. User signs in with valid credentials.
    - Expected Result: success authentication/ login
2. User signs in with invalid credentials
    - Expected Result: invalid authentication
3. User update/change username successfully.
    - Expected Result: go back to login state
4. User update/change password successfully.
    - Expected Result: go back to login state
5. User update/change items control successfully.
    - Expected Result: go back to login state
6. After successful authentication, the user is navigated to the dashboard.
    - Expected Result: successful navigation.
7. Items in the dashboard can be paginated.
    - Expected Result: when item number exceeds the limit of a page, a page is created and the items are paginated.
8. Items in the dashboard can be sorted by date.
    - Expected Result:
9. Items in the dashboard can be filtered by process.
    - Expected Result: when filter by process “run” only item with this state is availabe
10. Items in the dashboard cannot be sorted by id.
    - Expected Result: id column cannot be sorted by.
11. items control has 3 states: run, pause, stop for every item.
    - Expected Result: 3 states are shown to choose from for every item.
12. an item can only have one state.
    - Expected Result: user can only choose one state for every item.
13. The user cannot choose 2 states for one item.
    - Expected Result: user can only choose one state for every item.
14. from items control the user can navigate to item details.
    - Expected Result: successful navigation.
15. from items control the user can navigate to the dashboard.
    - Expected Result: successful navigation.
16. from item details the user can navigate to the dashboard.
    - Expected Result: successful navigation.
17. from item details the user can navigate to the items control.
    - Expected Result: successful navigation.
18. from the dashboard the user can navigate to the config.
    - Expected Result: successful navigation.
19. from the dashboard the user can navigate to the login.
    - Expected Result: successful navigation.
20. from the dashboard the user can navigate to the items control.
    - Expected Result: successful navigation.
21. from the dashboard the user can navigate to the item details.
    - Expected Result: successful navigation.
22. from the config the user can navigate to the dashboard.
    - Expected Result: successful navigation.

Test scenarios: (form fields)

Test cases for Radio Button

1. Check if the radio control button gets selected using mouse cursor action or TAB keyboard key selection.
2. Check the alignment of the radio button control on the form.
3. Check if the multiple radio buttons on the form get selected or not.
4. Check the CSS style of the radio button is as per the specification.
5. Check the CSS size of the radio button is as per the specification.
6. Check the CSS color of the radio button is as per the specification.
7. Click the radio control button and see if the page gets redirected to the next page.
8. Don't click on the radio buttons and see if clicking on the submit button generates a warning “to make a choice” or not.
9. Click on ‘Yes’ radio button control and hit submit to see if it redirects to the specific page.
10. Click on ‘No’ radio button control and hit submit to see if it redirects to the specific page.
11. Check if the user response of Yes is saved in the right database column.
12. Check if the user response of No is saved in the right database column.
13. Check if the database gets updated with either radio control choice being made.

Test Cases for Text Field

1. Depending on the text field requirement in the form, the input needs to be processed accordingly.
2. Name field needs to only accept the alphabet values.
3. Name field should not accept the numeric content.
4. Name field should not accept the symbols.
5. Card Number is a numeric field then it should only accept numbers.
6. The Card Number field should not accept characters and symbols in the input field.
7. Forms with the Credit Card number field only accept the specific numbers.
8. Card number field should not accept more input than the field needs.
9. The Credit and Debit card number field detects the card type based on the number.
10. Card number field should also detect the debit or credit card type.
11. In case it is a numeric field or a field that can accept all we can add:
12. Mobile number field should accept only the alphanumeric input from the users.
13. Mobile number field should
14. Address fields should allow characters, numeric and symbol input.
15. Address field could be a single or multi-line input box.
16. Address may or may not be a mandatory field.
17. How much is the maximum length of a text field?
18. What is the minimum length of a text field?
19. How much input is expected in the text field?
20. Does the text field allow input more than the textbox?
21. Does the text field allow input less than the specified textbox?
22. Does the textbox accept numbers only?
23. Does the textbox accept decimal numbers?
24. Does the textbox accept formatted numbers?
25. Does the text field accept alphabets?
26. Does the text field accept uppercase alphabets?
27. Does the text field accept lowercase alphabets?
28. Does the text field accept a mix of upper and lowercase alphabets?
29. Is there any specific character that field allows?

30. Is there any specific character that field doesn’t allow?
31. Does the field accept HTML characters?
32. Does the text field accept javascript?
33. Is the text field immune to SQL injection?
34. Does the text field allow copy paste?
35. Does the text field allow drag and drop of text content?
36. Does the cursor appear while typing the content?
37. Does the text field allow spaces?
38. Does the text field process content with only spaces?
39. Does the text field allow blank input?
40. How does the text field manage trailing spaces?
41. How does the text field manage leading spaces?

2. Unit testing:

Task:

- You are given a function $f(x),\ x \in R,\ \Delta x = \frac{1}{128}$. Please write a code (a function in C or Python) which takes x as an argument and returns f(x). Write unit tests with cunit in case of C or pytest if Python.

$$f(x) = \{-x^2,\ -10 \leq x \leq -2\ \frac{1-x}{1+x},\ -2 < x < 8, x \neq -1\ |x-12|,\ 8 < x \leq 35$$

The solution:

Please find the solution to this in this git repository:
https://github.com/MariamIbrahim127/Python-UnitTest-Task-Irdeto.git

3. DASH Parsing and Validation:

Task:

- You are given a piece of DASH manifest which represents one Period, please write a test that confirms/disproves validity of the period. Explain why this period is valid or invalid.

```xml
<Period duration="PT59.2S" id="2215" start="PT1614853010.04S">
    <BaseURL>dash/</BaseURL>
    <AdaptationSet audioSamplingRate="48000" codecs="mp4a.40.5" contentType="audio" group="1" id="1" lang="en"
mimeType="audio/mp4" segmentAlignment="true" startWithSAP="1">
        <AudioChannelConfiguration schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"
/>
        <Role schemeIdUri="urn:mpeg:dash:role:2011" value="main" />
        <SegmentTemplate initialization="YOYO-$RepresentationID$.dash" media="YOYO-$RepresentationID$-$Time$.dash"
presentationTimeOffset="77512944481920" timescale="48000">
            <SegmentTimeline>
                <S d="237568" t="77512944245119" />
                <S d="147457" />
                <S d="190464" />
                <S d="192511" r="2"/>
                <S d="192512" r="2"/>
                <S d="192513" r="2"/>
                <S d="190464" />
                <S d="192511" />
                <S d="192512" />
                <S d="192513" />
                <S d="190464" />
                <S d="190464" />
                <S d="199679" />
            </SegmentTimeline>
        </SegmentTemplate>
        <Representation bandwidth="64000" id="audio_1=64000">
      </Representation>
</AdaptationSet>
</Period>
```

The solution:

Test the dash manifest period:

1. Check that every period should have an ID and a start.
2. Check that every period is divided into adaptation sets.
3. Check that every adaptation set is divided into representations.
4. Check that every representation is divided into segments.
5. Check that every segment has segment info.

and that is why this xml dash manifest is not valid, because the representation is written in the end and is not holding the segment template inside of it.