Saturday, 12 January 2019

SFC WEB API

System Requirements

1. Design Principles

1.1. Design around **resources**

1.2. Focuses on **business entities**

1.3. Separate URIs for individual resource

1.4. **Stateless** request model

1.5. Use **HATEOAS** approach (navigation links, related resources)

1.6. URIs **should** be based on nouns not verbs

1.7. Design the API as an **abstraction of the data items (DB)**

1.8. A resource **does not have** to be based on a single data item (table)

1.9. Provide through the API the relationship between data items

1.10. Create a consistent naming convention

1.11. Follow the URIs pattern :

- ./resources (get all items for this resource)
- ./resources/item (get one single item)

1.12. Keep the URIs **as simple as possible** :

- ./resources/item/nested_resoucers (recommended nested resources)

1.13. The more requests, the bigger the server load

1.14. Combine related information into a bigger resource (Extraneous Fetching Anti-pattern)

2. Anti-Design Principles

2.1. Do not create API **mirroring the database**

2.2. Do not create **complex nested resources** URIs :

- ./resources/item/resources/item/resources/…

2.3. Do not create APIs that exposes a large number of small resources (**Chatty I/O Anti-pattern**)

2.4. Avoid client to send multiple requests to get all the desired data

2.5. Do not return to the client data that was not required

2.6. Do not return to the client very-large objects (**Extraneous Fetching Anti-pattern**)

2.7. Be careful with pseudo-resources and query-string parameters

3. Scenarios (use-cases)

3.1.Tracking Station

3.1.1. Login into station

- Validate login and password
- Validate the rights to login into the station
- Retrieve the last logged record (station_location)

3.1.2. Change the station

- Validate the rights to login into the station
- Validate login and password
- Update the login record (station_location)

3.1.3. Scan the unit

- Validate the unit route
- Validate the unit state (lock, fail, etc)
- Update the unit state
- Update the unit route
- Update the unit log
- Update the work order

3.2.Fail Station

3.2.1. Login into station

- Validate login and password
- Validate the rights to login into the station
- Retrieve the last logged record (station_location)

3.2.2. Change the station

- Validate the rights to login into the station
- Validate login and password
- Update the login record (station_location)

3.2.3. Scan the unit

- Validate the unit route
- Validate the unit state (lock, fail, etc)
- Update the unit state
- Update the unit route
- Update the unit log
- Update the work order

3.2.4. Fail the unit

- Validate the unit route
- Validate the unit state (lock, fail, etc)
- Update the unit state (fail status)
- Update the unit route
- Update the unit log
- Update the work order

3.3.Assembly Station

3.3.1. Login into station

- Validate login and password
- Validate the rights to login into the station
- Retrieve the last logged record (station_location)
- Retrieve the scan list (wo_parts)

3.3.2. Change the station

- Validate the rights to login into the station
- Validate login and password
- Update the login record (station_location)
- Retrieve the scan list (wo_parts)

3.3.3. Scan the unit

- Validate the unit route
- Validate the unit state (lock, fail, etc)
- Update the unit state (fail status)
- Update the unit route
- Update the unit log
- Update the work order
- Retrieve the scan list (wo_parts)

3.3.4. Scan the parts

- Validate work order parts rules
- Assembly parts into unit
- Update parts status

4. API Resources

4.1. Users

4.2. Stations

4.3. WorkOrders

4.4. Wips

4.5. Parts

5. API Resources Actions

5.1. Users

- Login
- Logout
- Change station
- Check rights
- Change password

5.2. WorkOrders

- Get scan list (by product, process, etc)
- Validate parts (part number, eee-code, etc…)

5.3. Wips

- Scan serial number
- Check state (active, lock, failed, etc…)
- Assembly parts (collection)

5.4. Parts

- Scan serial number
- Assembly part (single part into a unit)
- Get assembled parts

6. API Reference

GET	/users(?pageIndex&pageSize)
Description	Get all users
Query String	(optional) [pageIndex:int] Initial position to get records (Default : 1)
	(optional) [pageSize:int] Number of records to get (Default : 100)
Request Body	none
Response Body	array of user.json

GET	/users/{userId}
Description	Get single user by ID
Query String	none
Request Body	none
Response Body	user.json

POST	/users
Description	Create new user
Query String	none
Request Body	user.json
Response Body	user.json

POST	/users/login
Description	Log in to the API
Query String	none
Request Body	login.json
Response Body	{ token:string }

POST	/users/{userId}/logout
Description	Log out from the API
Query String	[userId:int] User id
Request Body	none
Response Body	none

PUT	/users/{userId}
Description	Update user by ID
Query String	none
Request Body	user.json
Response Body	user.json

DELETE	/users/{userId}
Description	Delete user by ID
Query String	none
Request Body	none
Response Body	none

GET	/stations(?pageIndex&pageSize)
Description	Get all stations
Query String	(optional) [pageIndex:int] Initial position to get records (Default : 1)
	(optional) [pageSize:int] Number of records to get (Default : 100)
Request Body	none
Response Body	array of station.json

GET	/stations/{stationId}
Description	Get a single station by Id
Query String	(optional) [pageIndex:int] Initial position to get records (Default : 1)
	(optional) [pageSize:int] Number of records to get (Default : 100)
Request Body	none
Response Body	station.json

GET	/workorders(?pageIndex&pageSize)
Description	Get all work orders
Query String	(optional) [pageIndex:int] Initial position to get records (Default : 1)
	(optional) [pageSize:int] Number of records to get (Default : 100)
Request Body	workorder.json
Response Body	array of workorder.json

GET	/workorders/{workorderId}
Description	Get single work order by ID
Query String	none
Request Body	none
Response Body	workorder.json

GET	/workorders/{workorderId}/wips
Description	Get all the wips from the work order
Query String	none
Request Body	none
Response Body	array of wip.json

GET	/workorders/{workorderId}/wips/{wipId}
Description	Get a single wip from the work order
Query String	none
Request Body	none
Response Body	wip.json

GET	/workorders/{workorderId}/parts(?station)
Description	Get all the parts (BOM) from the work order
Query String	(optional) [station:string] Station code to get parts from
Request Body	none
Response Body	array of part.json

GET	/workorders/{workorderId}/parts/{partId}
Description	Get a single the part from the work order
Query String	none
Request Body	none
Response Body	part.json

GET	/workorders/{workorderId}/parts/{partName}
Description	Get all the parts for the given part name
Query String	none
Request Body	none
Response Body	part.json

POST	/workorders
Description	Create new work order
Query String	none
Request Body	workorder.json
Response Body	workorder.json

PUT	/workorders/{workorderId}
Description	Update work order by ID
Query String	none
Request Body	workorder.json
Response Body	workorder.json

DELETE	/workorders/{workorderId}
Description	Delete work order by ID
Query String	none
Request Body	none
Response Body	none

GET	/wips(?pageIndex&pageSize)
Description	Get all the wips
Query String	(optional) [pageIndex:int] Initial position to get records (Default : 1)
	(optional) [pageSize:int] Number of records to get (Default : 100)
Request Body	wip.json
Response Body	array of wip.json

GET	/wips/{wipId}
Description	Get a single wip by Id
Query String	none
Request Body	none
Response Body	wip.json

GET	/wips/{wipId}/parts
Description	Get all assembled parts with the wip
Query String	none
Request Body	none
Response Body	array of part.json

GET	/wips/{wipId}/parts/{partId}
Description	Get a single assembled part with the wip
Query String	none
Request Body	none
Response Body	part.json

POST	/wips
Description	Create new wip
Query String	none
Request Body	wip.json
Response Body	wip.json

POST	/wips/{wipId}/check?station
Description	Only checks the wip status and whether it is ok to pass or not the station
Query String	[station:string] Station code to check the wip
Request Body	none
Response Body	none

POST	/wips/{wipId}/scan?station
Description	Scan the wip to pass or fail into the station
Query String	[station:string] Station code to pass/fail the wip
Request Body	wipScan.json
Response Body	none

POST	/wips/{serialNo}/check?station
Description	Only checks the wip status and whether it is ok to pass the station
Query String	[station:string] Station code to check the wip
Request Body	none
Response Body	none

POST	/wips/{serialNo}/scan?station
Description	Scan the wip to pass or fail into the station
Query String	[station:string] Station code to pass/fail the wip
Request Body	wipScan.json
Response Body	none

POST	/wips/{wipId}/parts/check
Description	Check whether the part can be assembled or not
Query String	none
Request Body	part.json
Response Body	none

POST	/wips/{wipId}/parts
Description	Assembly the part into the WIP
Query String	none
Request Body	part.json
Response Body	none

PUT	/wips/{wipId}
Description	Update wip by ID
Query String	none
Request Body	wip.json
Response Body	wip.json

DELETE	/wips/{wipId}
Description	Delete wip by ID
Query String	none
Request Body	wip.json
Response Body	wip.json

DELETE	/wips/{wipId}/parts
Description	Disassembly the part from the WIP
Query String	none
Request Body	part.json
Response Body	none

7. JSON Data Structures

user.json

```
{
  "id"    : "int",
  "login" : "string",
  "photo" : "string"
}
```

login.json

```
{
  "login"      : "string",
  "password"   : "string",
  "stationId"  : "string",
  "macAddress" : "string"
}
```

station.json

```
{
  "id"   : "int",
  "code" : "string",
  "name" : "string",
  "type" : "string"
}
```

workorder.json

```
{
  "id"         : "int",
  "no"         : "string",
  "mpn"        : "string",
  "sku"        : "string",
  "product"    : "string",
  "closeFlag" : "boolean",
  "input"      : "int",
  "output"     : "int",
  "target"     : "int"
}
```

wip.json

```
{
  "id"              : "int",
  "serial"          : "string",
  "mpn"             : "string",
  "sku"             : "string",
  "product"         : "string",
  "currentStation"  : "string",
  “nextStation”     : "string",
  "finishFlag"      : "string"
}
```

part.json

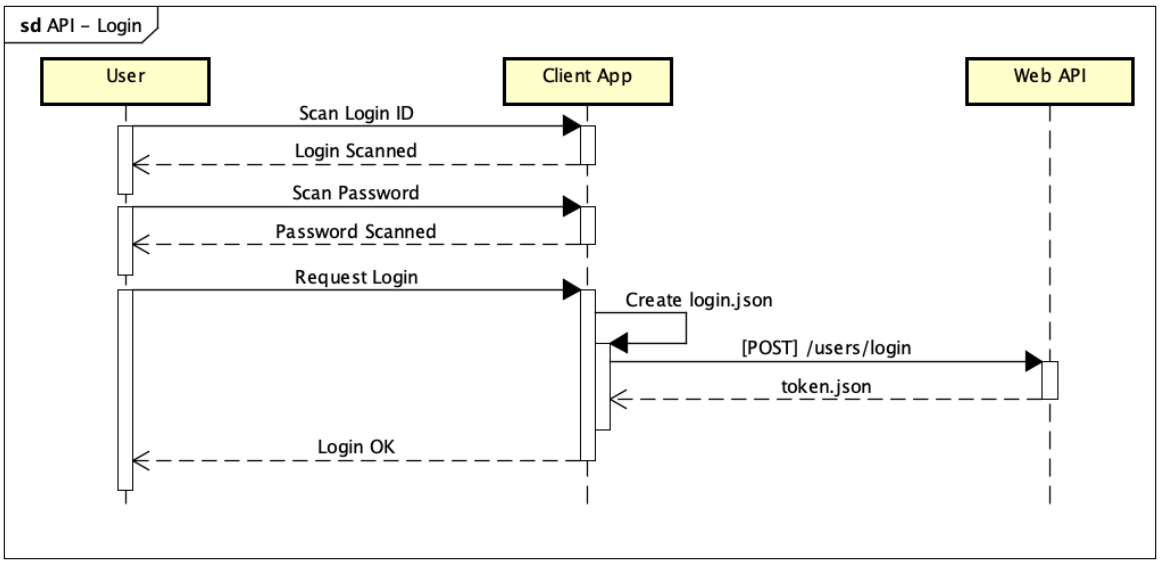
```
{
  "id"              : "int",
  "serial"          : "string",
  "partName"        : "string",
  "custPartNo"      : "string",
  "eeeCode"         : “string"
}
```

wipScan.json

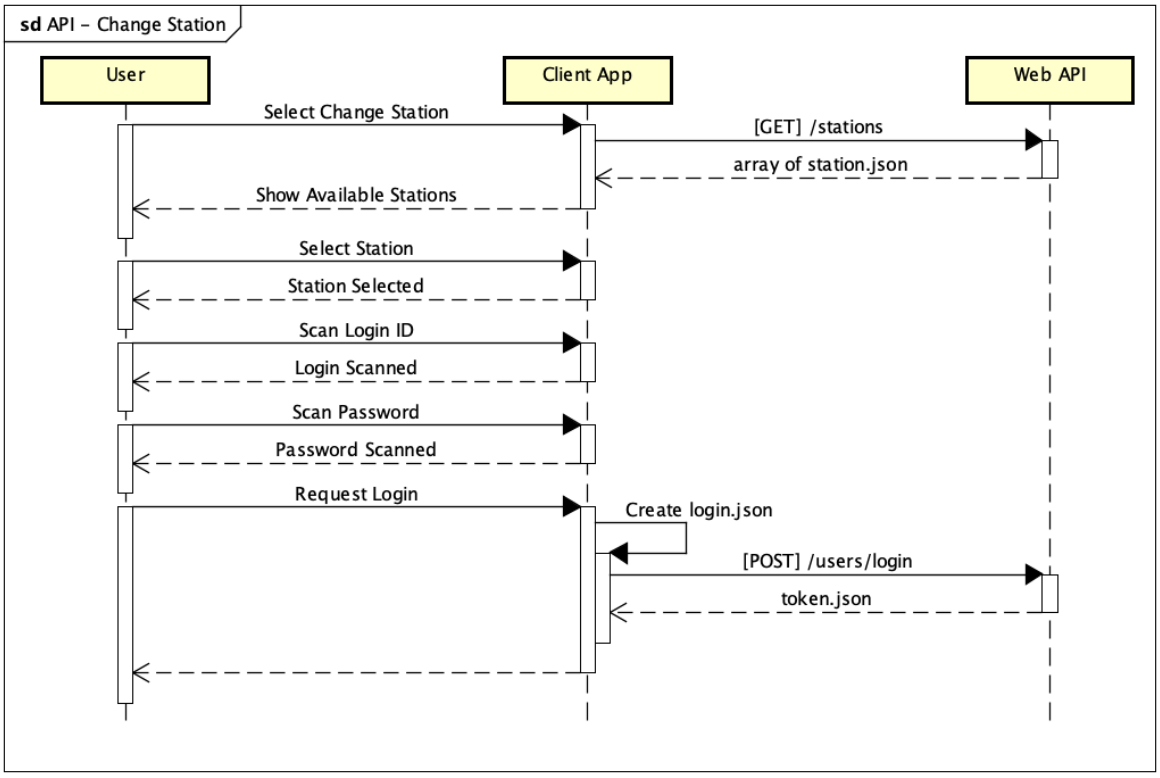
```
{
  "testItem"        : "string",
  "symptom"         : "string"
}
```

8. API Sequence Diagram

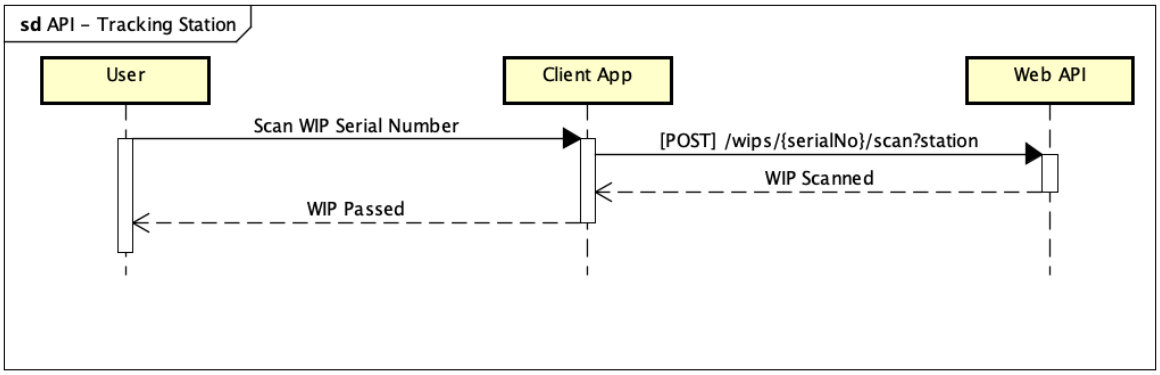
Login Flow



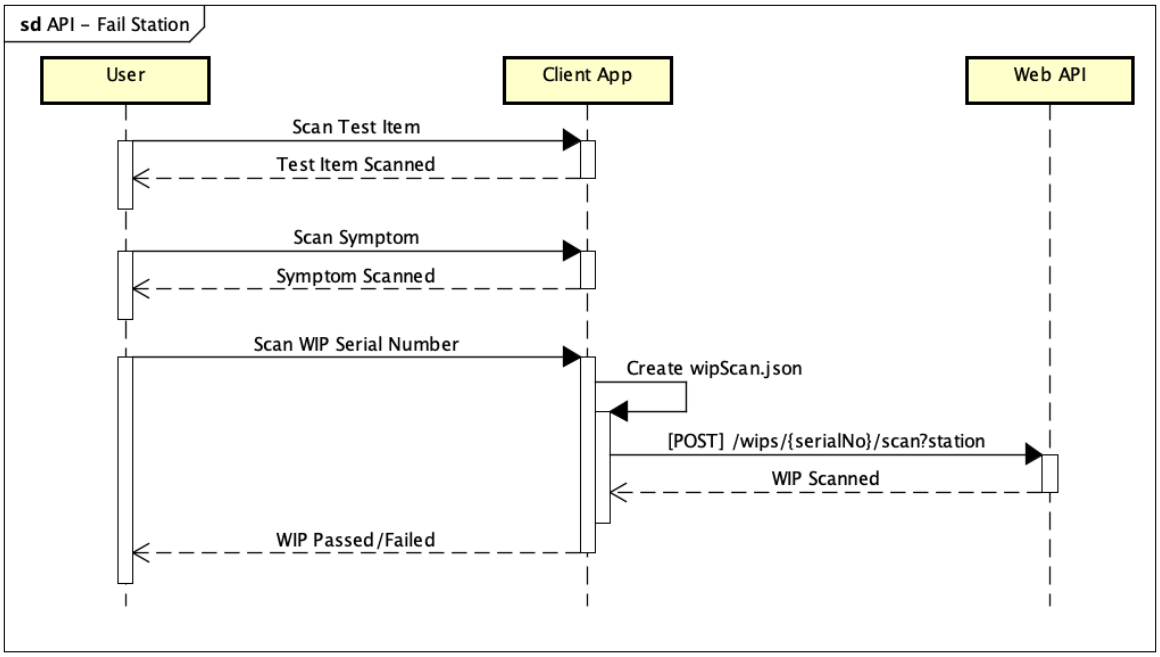
Change Station Flow



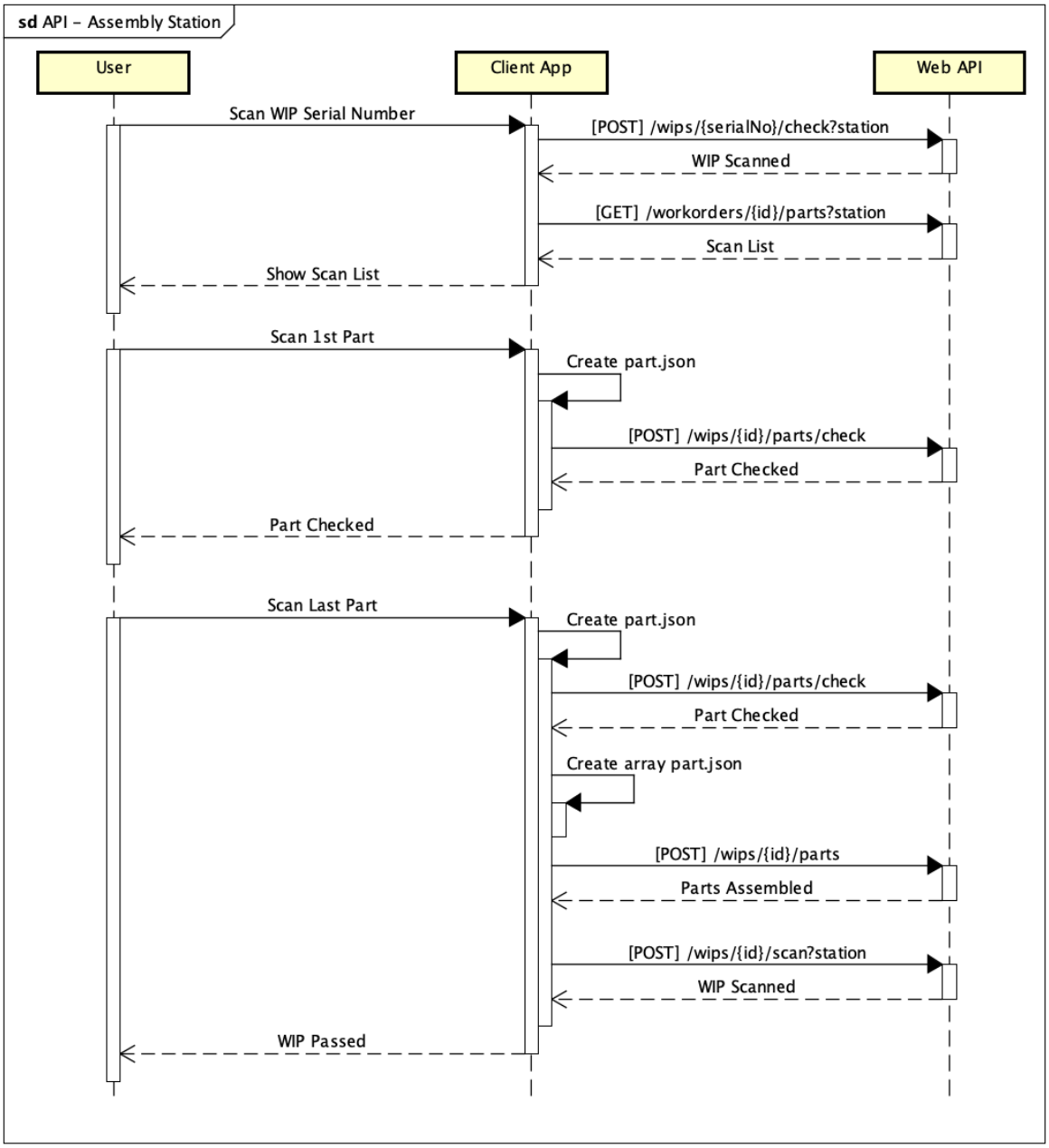
Tracking Station Flow



Fail Station Flow

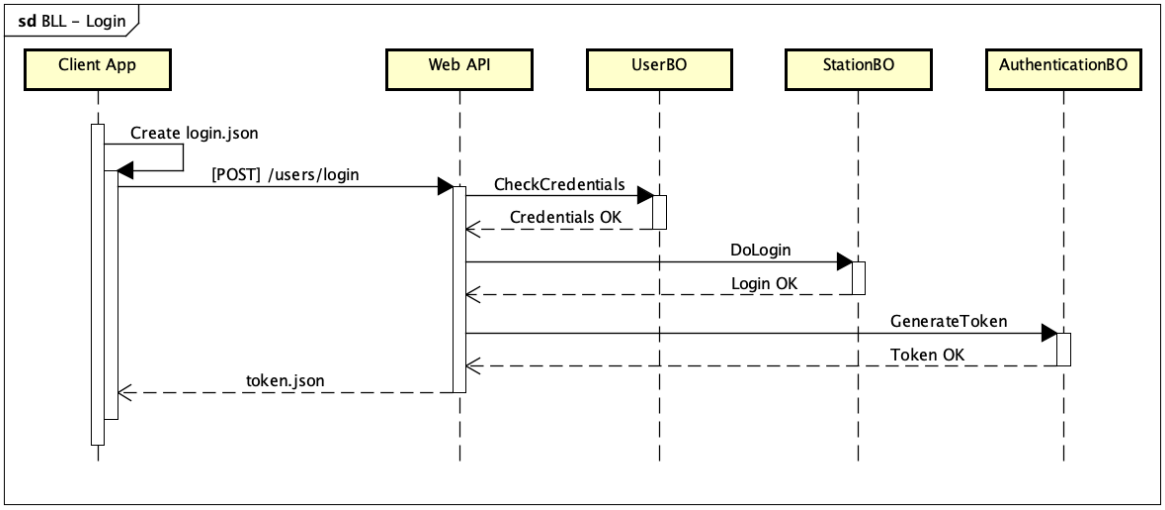


Assembly Station Diagram

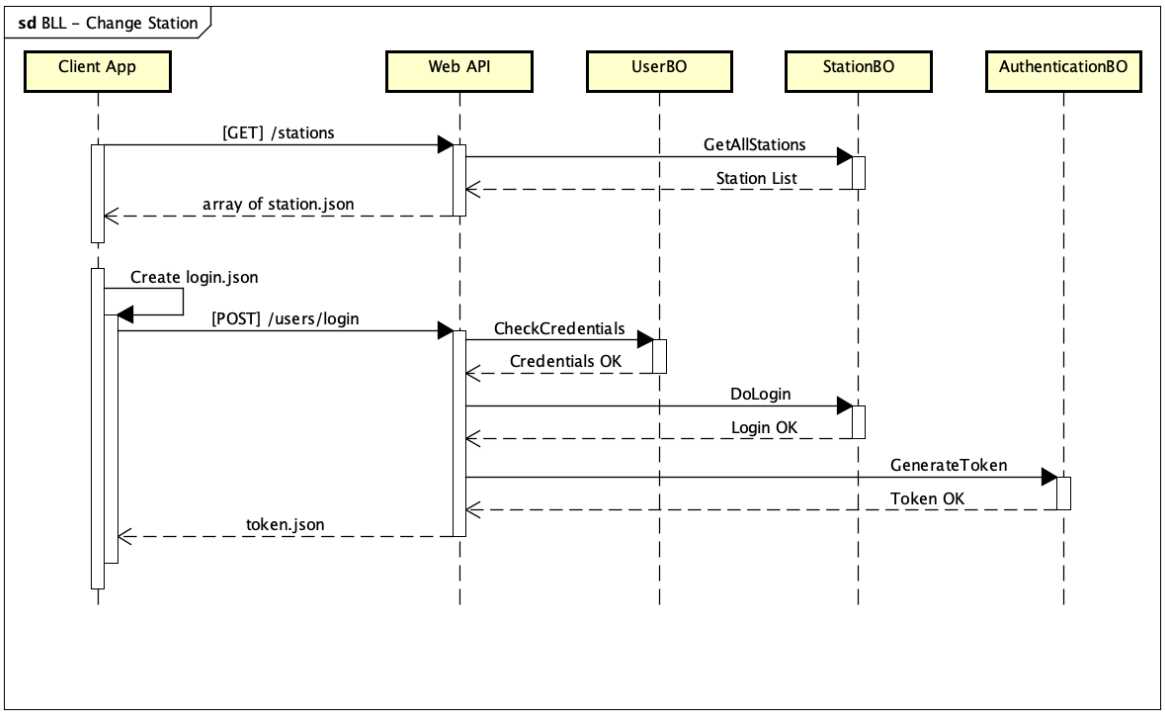


9. Business Logic Sequence Diagram

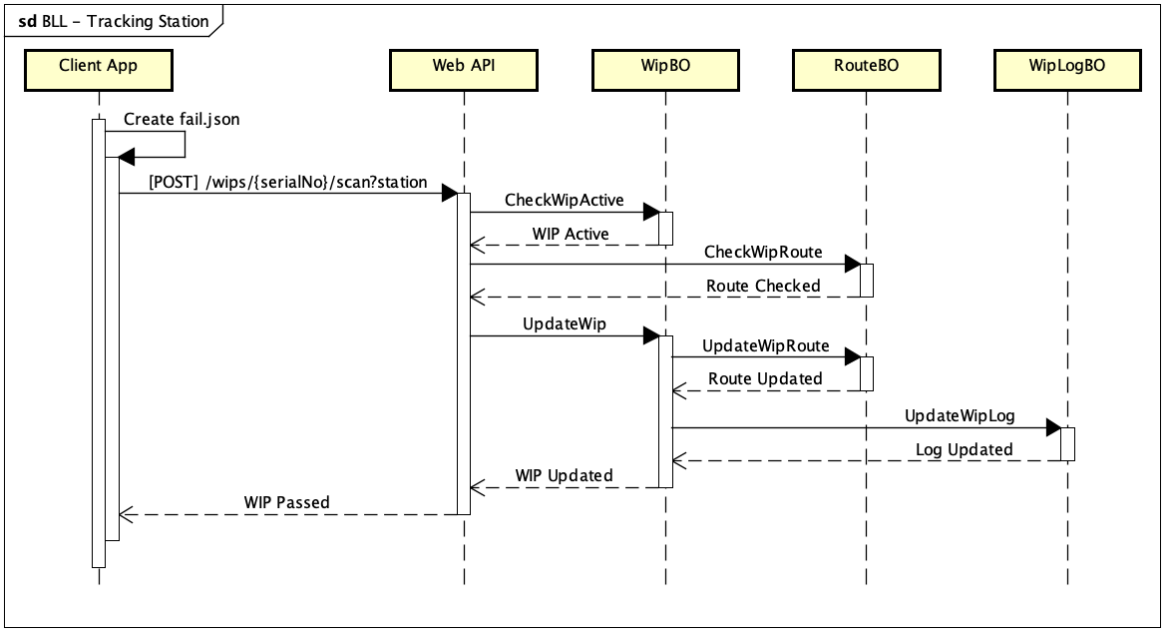
Login Flow



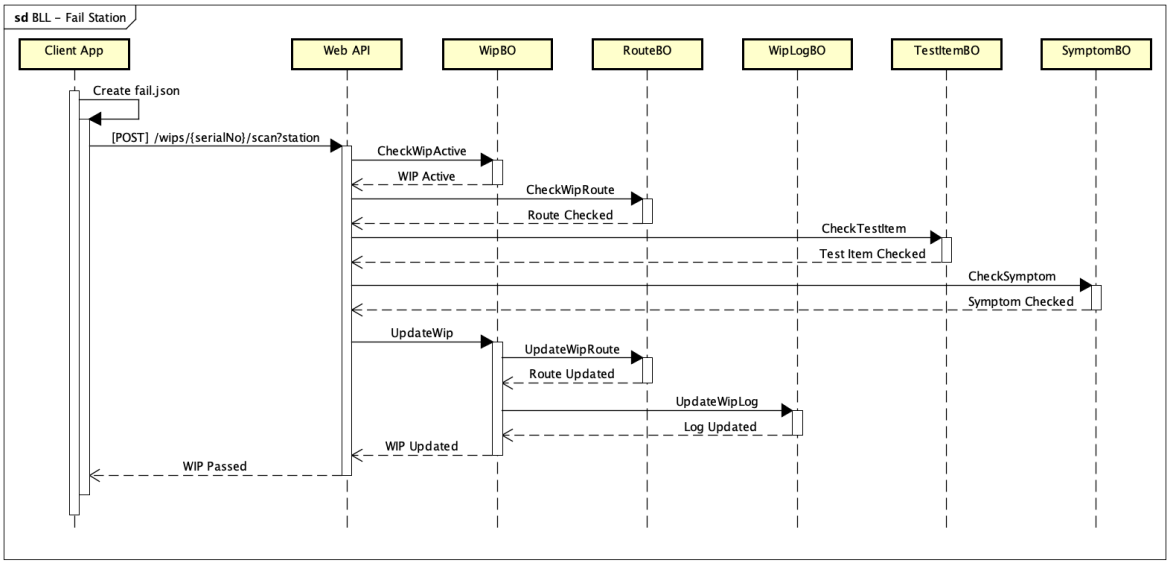
Change Station Flow



Tracking Station Flow



Fail Station Flow



Assembly Station Diagram