Design - User Device Mapping

# **Background**

After successful login by the Admin person, user can tag the list of devices already configured in the admin portal with respect to the center id should get mapped to the logged in station id based on the device category.

The **target users** are

* Super Admin

The key **requirements** are

Device Mapping:-

On successful login of the user, will show off “New Registration” screen. There exists the “Menu”, which lists the options/ features. On clicking the “Device Mapping” link, should land up in displaying the list of devices along with the respective categories. Based on the user selected category [GPS, BIO, and Scanner] the respective devices specific to the enrollment center should be displayed in the left side list.

User can select the required devices [with specific serial no] and associate or de-associate with the current machine [Station].

There is no device connectivity validation required.

The key **non-functional requirements** are

* Security:
  + Should allow the user to map only the devices tagged to their registration center id.
  + Should not store any sensitive information as plain text information.
  + The data which resides in the data-base should be in encrypted format.

# **Solution**

The key solution considerations are –

* On clicking the “Device Mapping” list in menu tab on Registration screen, it should hit the controller [**DeviceMappingController**] to fetch the master device and already mapped device list from database.
* **DeviceMappingController:**
  + Invoke the interface **MachineMappingService** and call the **getDeviceMappingList(String centerId, String machineId, String deviceType)** to get the master device list and already mapped device list from db.
  + Display the fetched detail in the UI screen.
  + **updateMappedDevice() –** Once the device mapping is done in the ui and while submitting the page, invoke this method and pass the respective DTO list. It invokes the **MachineMappingService.updateMappedDevice()** method to complete the mapping process at db.
* Create **MachineMappingServiceImpl** and create DTO for the same.
  + When this service is triggered for **getDeviceMappingList**() functionality - it gets the device [master list and mapped list] details.
  + It hits DAO and the respective repository to fetch **list<master device> and list<mapped device>.**
  + Compare the whole device list with the already mapped device list and remove the mapped device from the master device list and render the result to the controller.
  + **updateMappedDevice**( List< DeviceMappingDTO>) – it does the device update to the db by calling the DAO class.
* **MachineMappingDAOImpl**
  + getAllDeviceBasedOnCenterId(String centerId, String deviceType) – return all the active device. If deviceid provided then filter the data based on the deviceType and render the result.
  + getAllMappedDevice(String machineId, String deviceType) - return all the device specific to the machine which are in active state. If deviceid provided then filter the data based on the deviceId and render the result.
  + updateMappedDevice( List< DeviceMappingDTO>) – this method updates the existing list in the database with the new list provided by the user.
* Send an Alert message (say) “Device mapped successfully” or an error message.
* Handle exceptions in using custom Exception handler and send correct response to client.

**UI:**

* Design UI using FXML and map the UI individual components in DeviceMappingController class.
* DeviceMappingController – it should communicate between UI screen and Service ‘MachineMappingServiceImpl’ class to render the data to screen and capture the data from screen.
* Create the proper alert success/error to intimate the user.

**Classes**:

**UI**: DeviceMachineMapping.fxml

**Controller**: DeviceMappingController.java

**Service**: MachineMappingServiceImpl.java

**DAO**: MachineMappingDAOImpl.java

**Repository:** DeviceMappingRepository.java

**DTO**: DeviceMappingDTO.java

Sequence and Class Diagram:

