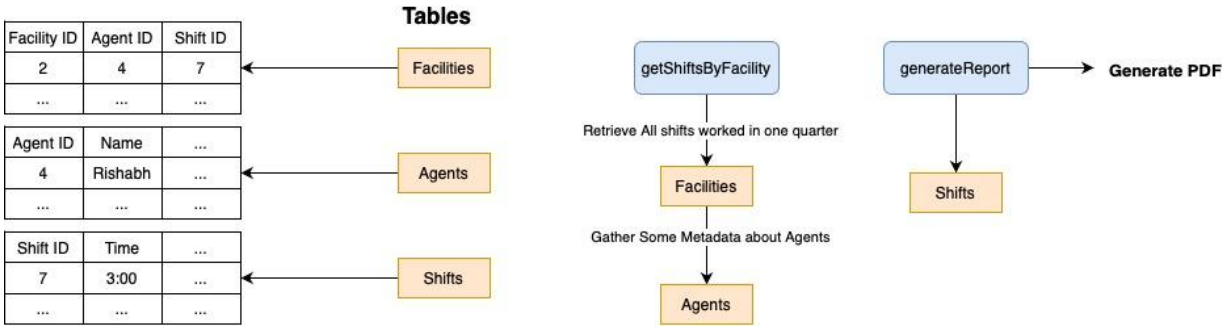
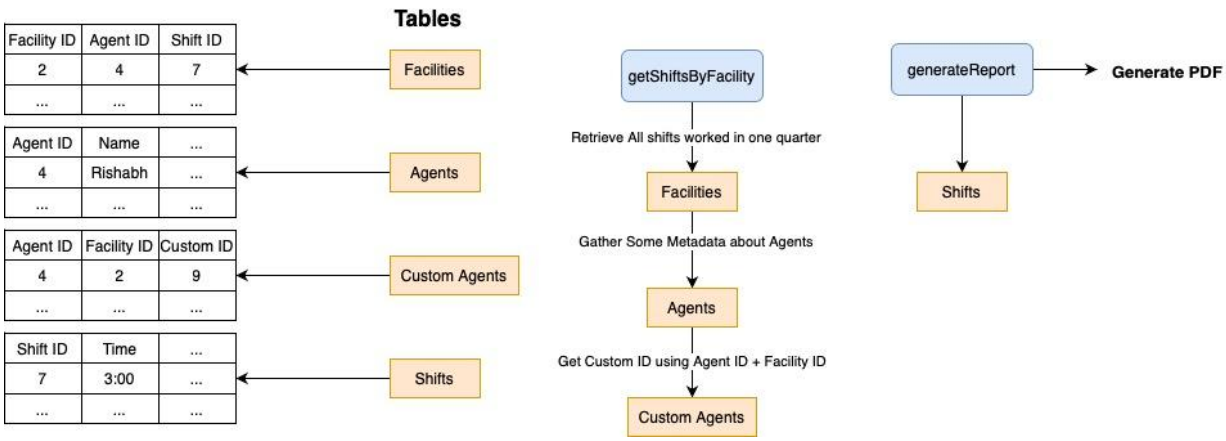
CBH Ticket Breakdown

Problem:



According to my understanding, There are 3 tables: Facilities, Agents and Shifts. The data that is stored in each of them is:
Facilities: Stores Information about a Facility, and the Agent and Shift that took place
Agents: Stores basic data about Agent
Shifts: Stores basic data about a Shift

Solution:



There are two solutions that we can implement. The one I went with is one where we create a new Table called Custom Agents. This will allow us to store the Custom ID of the Agent.

When calling the getShiftsByFacility function, we will make another call to this table and check if a custom ID is present. If yes, we will display the custom ID. If not, we will display the ID as before that we store in our database. The custom ID will be retrieved using Agent ID and Facility ID.

The other solution that we can implement is one of storing the custom ID along with the Agent table. But there are two drawbacks:

1. If there are Agents who work in multiple facilities and have different custom IDs in each of them, we would not be able to handle those cases
2. If there are Facilities which do not require a custom ID, we would be wasting space in our database by allocating unnecessary space for a field that is not required for all entries.

Ticket Breakdown:

1. **Creating the Table (0.5 hour):** Firstly, we will create a new table. If working with MySQL it would be a simple query that would be run.
   'CREATE TABLE `CustomAgents` (`agent_id` int(11) NOT NULL, `facility_id` int(11) NOT NULL, `custom_id` varchar(255) DEFAULT NULL, PRIMARY KEY (`agent_id`, `facility_id`))'
   This should be simple enough and would require about 0.5-1 hour in creating and testing
2. **Modifying getShiftsByFacility (2 hours):** Second, we need to modify the getShiftsByFacility function and retrieve the custom ID for the Agent if present.
   This would involve creating a Data Manipulation Layer where we keep the functions that write, delete, and update entries in the new table. Of Course, it also depends on the Architecture
   This is also an easy enough task, however, would require a lot of testing as this is mission critical. Should take 2-3 hours with testing.
3. **Modifying getShiftsByFacility return object (2 hours):** Third, we must return a standard format from this function so that other functions remain unaffected.
   We may choose to overwrite the agent_id field that we are returning. However, that would not be a real representation of the data. Hence we will create a new field as custom_id and keep it null if there is no custom_id for that agent and facility combination. Most of the time would be required for testing so should take an addition 2 hours
4. **Modifying generatePDF (1 hours):** Lastly, we will modify the generatePDF function to display the custom_id if present over agent_id. If not present then we shall display the Agent_id as before.
   Again, most of the time would be spent testing the individual function which should take 1 hour
5. **Integration Testing (1 hours):** Final testing or Integration Testing would take about an hour and we may need to Debug some code but an additional 1 hour would be spent.

I would say 6 hours would be spent on the task. However, there may be some situations where we may get ahead of schedule. This is a liberal estimate.