**Assignment Deliverable**:

Repository URL: <https://github.com/armanJoy/employee-salary-disbursment-poc>

1. Basic salary of the lowest grade – PATCH [/v1/salary/basic-salary](http://localhost:9090/swagger-ui.html" \l "/operations/salary-ctrl/updateBasicSalaryUsingPATCH) is used to update basic salary of lowest grade.
2. Balance of the company bank account – PATCH [/v1/salary/company-balance](http://localhost:9090/swagger-ui.html" \l "/operations/salary-ctrl/updateComSalaryAcBalanceUsingPATCH)
3. Employee Information – GET [/v1/user/employees](http://localhost:9090/swagger-ui.html" \l "/operations/user-mgt-ctrl/getEmployeesUsingGET)
4. Prepare CRUD functionality for each entity
   * Employee CRUD
     + Create – POST [/v1/user/create](http://localhost:9090/swagger-ui.html" \l "/operations/user-mgt-ctrl/createUserUsingPOST)
     + Read – GET [/v1/user/employees](http://localhost:9090/swagger-ui.html" \l "/operations/user-mgt-ctrl/getEmployeesUsingGET)
     + Update – PUT [/v1/user/update](http://localhost:9090/swagger-ui.html" \l "/operations/user-mgt-ctrl/updateEmployeeInfoUsingPUT)
     + Delete – DELETE [/v1/user/remove](http://localhost:9090/swagger-ui.html" \l "/operations/user-mgt-ctrl/removeUserUsingDELETE)
   * Employee Bank Info CRUD
     + Create – POST [/v1/empbank/add](http://localhost:9090/swagger-ui.html" \l "/operations/emp-bank-ctrl/addEmpBankUsingPOST)
     + Read – GET [/v1/empbank](http://localhost:9090/swagger-ui.html" \l "/operations/emp-bank-ctrl/getEmpBankUsingGET)
     + Update – PUT [/v1/empbank/update](http://localhost:9090/swagger-ui.html" \l "/operations/emp-bank-ctrl/updateEmpBankUsingPUT)
     + Delete – DELETE [/v1/empbank/remove](http://localhost:9090/swagger-ui.html" \l "/operations/emp-bank-ctrl/removeEmpBankUsingDELETE)
5. Employee ID should be 4 digit and unique – implement by starting id generation sequence from 1000 using hibernate id generation policy
6. Maintain proper relation among entities – implemented using both hibernate annotations and foreign key constraints.
7. Provide proper validation to input data – validation of @Requestbody and @Requestparam payload ensures using javax.validation framework. Faulty or invalid data cannot be passed to the API. If any invalid data passed to API, the API will return a validation error message with 400 status.
8. Calculate the salary of each employee – GET [/v1/salary/employee-salary](http://localhost:9090/swagger-ui.html" \l "/operations/salary-ctrl/getEmployeeSalaryDetailsUsingGET)
9. Transfer the salary amount from main account to employee’s accounts – GET [/v1/salary/initiate-disburse](http://localhost:9090/swagger-ui.html" \l "/operations/salary-ctrl/disburseMonthlySalaryUsingGET)
10. Print / display the name, rank and salary of each employee (A salary sheet) – GET [/v1/salary/monthly-details](http://localhost:9090/swagger-ui.html" \l "/operations/salary-ctrl/getDisburseSalaryDetailsUsingGET)
11. Print / display the total paid salary and remaining balance of the company account – GET [/v1/salary/initiate-disburse](http://localhost:9090/swagger-ui.html" \l "/operations/salary-ctrl/disburseMonthlySalaryUsingGET)
12. Implement Login/Logout features using JWT – Implemented JWT based authentication-authorization througout the application using Spring Security.

POST [/v1/user/login](http://localhost:9090/swagger-ui.html" \l "/operations/user-mgt-ctrl/loginUsingPOST) POST[/v1/user/logout](http://localhost:9090/swagger-ui.html" \l "/operations/user-mgt-ctrl/logoutUsingPOST)

1. Angular/React frontend - Couldn’t integrate frontend pages due to lack of time. My PC was not configured for React development. I faced some issue which would take more time. So I focused to backend first. In backend code I ensured all the best practices and quality coding.