

Table of Contents

Assignment Deliverable:.....2

Screenshots of Swagger Documentation:.....3

Given Assignment:.....4

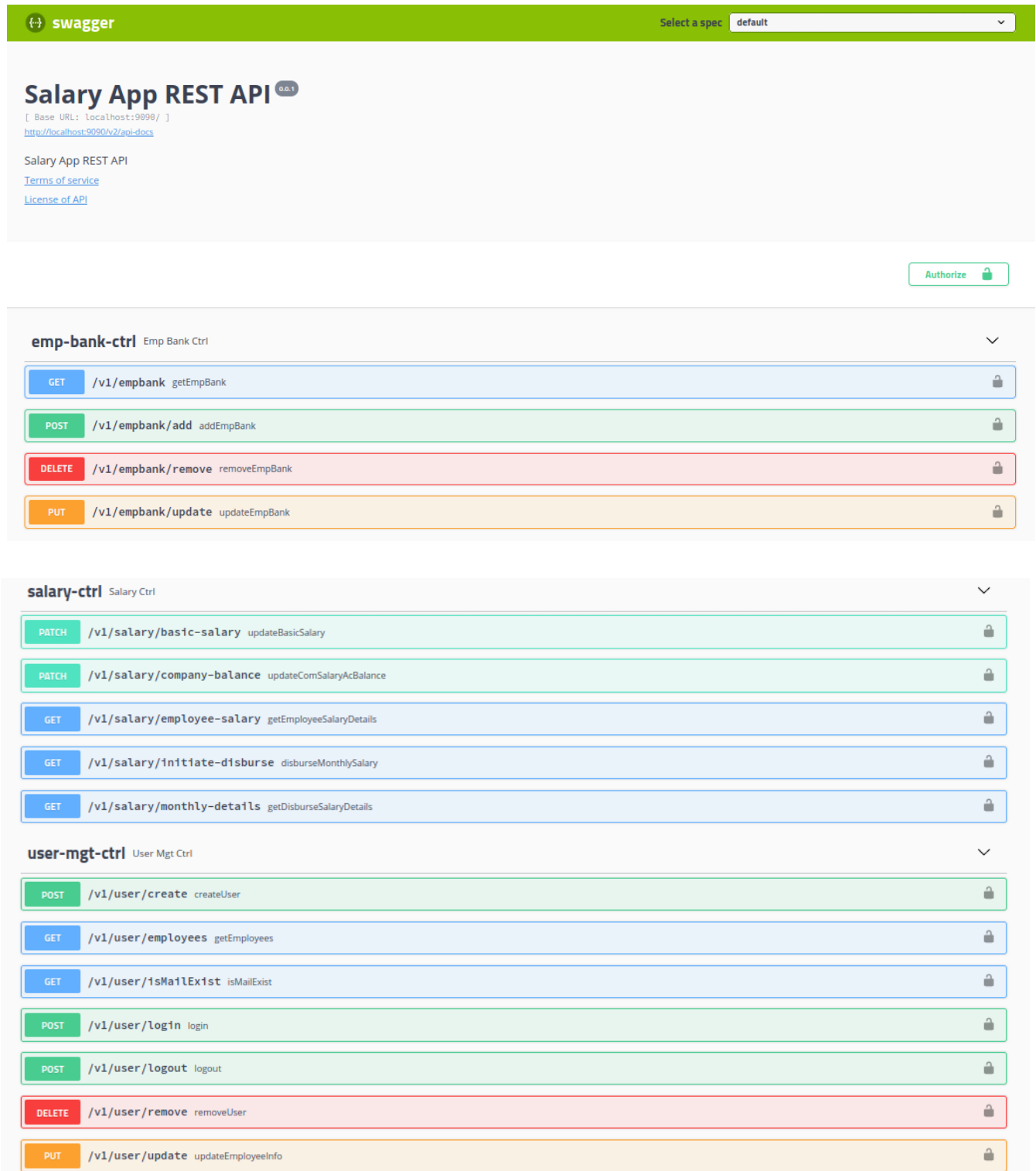
Assignment Deliverable:

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

1. Basic salary of the lowest grade – PATCH [/v1/salary/basic-salary](#) is used to update basic salary of lowest grade.
2. Balance of the company bank account – PATCH [/v1/salary/company-balance](#)
3. Employee Information – GET [/v1/user/employees](#)
4. Prepare CRUD functionality for each entity
 - Employee CRUD
 - Create – POST [/v1/user/create](#)
 - Read – GET [/v1/user/employees](#)
 - Update – PUT [/v1/user/update](#)
 - Delete – DELETE [/v1/user/remove](#)
 - Employee Bank Info CRUD
 - Create – POST [/v1/empbank/add](#)
 - Read – GET [/v1/empbank](#)
 - Update – PUT [/v1/empbank/update](#)
 - Delete – DELETE [/v1/empbank/remove](#)
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](#)
9. Transfer the salary amount from main account to employee's accounts – GET [/v1/salary/initiate-disburse](#)
10. Print / display the name, rank and salary of each employee (A salary sheet) – GET [/v1/salary/monthly-details](#)
11. Print / display the total paid salary and remaining balance of the company account – GET [/v1/salary/initiate-disburse](#)
12. Implement Login/Logout features using JWT – Implemented JWT based authentication-authorization throughout the application using Spring Security.
POST [/v1/user/login](#) POST [/v1/user/logout](#)

13. 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.

Screenshots of Swagger Documentation:



The screenshot displays the Swagger UI for the 'Salary App REST API'. The interface has a green header bar with the 'swagger' logo and a 'Select a spec' dropdown menu set to 'default'. Below the header, the API title 'Salary App REST API' is shown with a version tag '0.0.1'. The base URL is '[Base URL: localhost:9090/]' and a link to 'http://localhost:9090/v2/api-docs' is provided. There are links for 'Terms of service' and 'License of API'. An 'Authorize' button is located on the right. The API is organized into three sections: 'emp-bank-ctrl', 'salary-ctrl', and 'user-mgt-ctrl'. Each section lists its endpoints with their respective HTTP methods, paths, and descriptions. The endpoints are color-coded by method: GET (blue), POST (green), DELETE (red), and PUT (orange). Each endpoint entry includes a lock icon on the right.

emp-bank-ctrl Emp Bank Ctrl

- GET /v1/empbank getEmpBank
- POST /v1/empbank/add addEmpBank
- DELETE /v1/empbank/remove removeEmpBank
- PUT /v1/empbank/update updateEmpBank

salary-ctrl Salary Ctrl

- PATCH /v1/salary/basic-salary updateBasicSalary
- PATCH /v1/salary/company-balance updateComSalaryAcBalance
- GET /v1/salary/employee-salary getEmployeeSalaryDetails
- GET /v1/salary/initiate-disburse disburseMonthlySalary
- GET /v1/salary/monthly-details getDisburseSalaryDetails

user-mgt-ctrl User Mgt Ctrl

- POST /v1/user/create createUser
- GET /v1/user/employees getEmployees
- GET /v1/user/isMailExist isMailExist
- POST /v1/user/login login
- POST /v1/user/logout logout
- DELETE /v1/user/remove removeUser
- PUT /v1/user/update updateEmployeeInfo

Given Assignment:

Assignment

1. Write a web application to calculate and pay the salary of employees?

Details: There are total six grades /ranks; Grade one is the highest and grade 6 is the lowest. There are total 10 employees at the company - grade one: 1, grade two: 1, grade three: 2, grade four: 2, grade five: 2, grade six: 2. Each employee has employee ID, name, grade/rank, address, mobile and a bank account. The bank account has account type – savings/current, account name, account number, current balance, bank and branch name etc. The salary is divided into the following heads - Basic House rent - 20% of the basic Medical allowance - 15% of basic

The basic salary of the lowest grade will be taken as input. The basic of the others grade will be calculated as basic of the previous grade + 5000 taka.

The company will have a main bank account. The initial balance will be taken as input. From the company account the salary will be transferred to the employees account. If during the salary transfer, the company bank account run out of money, there will be an input option to add more money to the account and then continue the salary payment.

Summary: Inputs: 1. Basic salary of the lowest grade 2. Balance of the company bank account 3. Employee Information

Tasks and outputs:

1. Prepare CRUD functionality for each entity
2. Employee ID should be 4 digit and unique
3. Maintain proper relation among entities
4. Provide proper validation to input data
5. Calculate the salary of each employee
6. Transfer the salary amount from main account to employee's accounts
7. Print / display the name, rank and salary of each employee (A salary sheet)
8. Print / display the total paid salary and remaining balance of the company account
9. Implement Login/Logout features using JWT

Tech Stack:

1. Angular/React
2. Spring boot (Rest API) /Express/Node