

Pension Management System

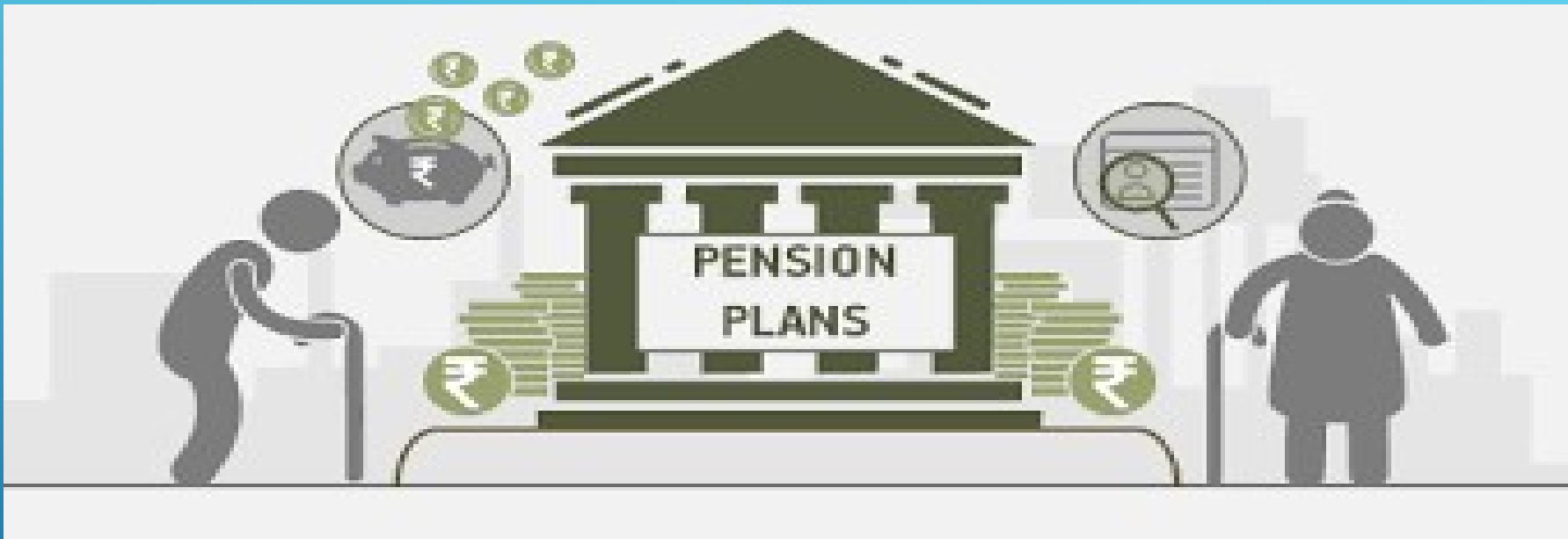
Submitted By : Batch-B4: Group-104

- 1) Sandeep Panda
- 2) Sreenath A G
- 3) Chhaya Vaidhya
- 4) Shreyash Singrupkar
- 5) Pavan Chितtemreddy

Hardware and Software Requirement

- Hardware Requirement:
 - Developer Desktop PC with 8GB RAM
- Software Requirement (Java)
 - Spring Tool Suite (STS) Or any Latest Eclipse
 - Configure Maven in Eclipse
 - Maven
 - Docker
 - Postman
 - Git

Pension Scheme



A pension is a retirement fund for an employee paid into by the employer, employee, or both, with the employer usually covering the largest percentage of contributions.

Spring Framework

- Spring is the most popular application development framework for enterprise Java. Millions of developers around the world use Spring Framework to create high performing, easily testable, and reusable code.
- Java Spring Boot (Spring Boot) is a tool that makes developing web application and microservices with Spring Framework faster and easier through three core capabilities
- Advantages of Spring Framework
 - Fast Development
 - Easy to test
 - Loose Coupling
- Spring Framework also offers built-in support for typical tasks that an application needs to perform, such as data binding, type conversion, validation, exception handling, resource and event management, internationalization, and more.

Spring Boot

Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications that you can "just run"

Features

- 1) Create stand-alone Spring applications
- 2) Embed Tomcat, Jetty or Undertow directly (no need to deploy JAR files)
- 3) Provide opinionated 'starter' dependencies to simplify your build configuration
- 4) Automatically configure Spring and 3rd party libraries whenever possible
- 5) Provide production-ready features such as metrics, health checks, and externalized configuration
- 6) Absolutely no code generation and no requirement for XML configuration

Maven

Maven is a popular open-source build tool developed by the Apache Group to build, publish, and deploy several projects at once for better project management. The tool provides allows developers to build and document the lifecycle framework.


MAVEN FOCUSES

- A) Builds
- B) Documentation
- C) Dependencies
- D) Reports
- E) SCMs
- F) Distribution
- G) Releases
- H) Mailing list

Database

- A database is an organized collection of data, so that it can be easily accessed and managed.
- You can organize data into tables, rows, columns, and index it to make it easier to find relevant information.
- Database handlers create a database in such a way that only one set of software program provides access of data to all the users.
- The main purpose of the database is to operate a large amount of information by storing, retrieving, and managing data.
- There are many dynamic websites on the World Wide Web nowadays which are handled through databases. For example, a model that checks the availability of rooms in a hotel. It is an example of a dynamic website that uses a database.
- There are many databases available like MySQL, Sybase, Oracle, MongoDB, Informix, PostgreSQL, SQL Server, etc.

Postman

- Postman is one of the most popular software testing tools which is used for API testing. With the help of this tool, developers can easily create, test, share, and document APIs.
 - This tutorial will help in understanding why Postman is so famous and what makes it unique when compared to other API testing tools.
- 
- A series of white diagonal lines of varying lengths and thicknesses are positioned in the bottom right corner of the slide, creating a modern, abstract graphic element.

Microservices In Pension Management System

Microservices - also known as the microservice architecture - is an architectural style that structures an application as a collection of services that are

- Highly maintainable and testable
- Loosely coupled
- Independently deployable
- Organized around business capabilities
- Owned by a small team

Microservices Used In Pension Management System

- PensionGateway Microservice
- Authorization Microservice
- Pension Process Microservice
- Pension Detail Microservice

Pensioner Gateway Microservice

- In a distributed environment, services need to communicate with each other. However, this is interservice communication. We also have use-cases where a client outside our domain wants to hit our services for the API. So, either we can expose the address of all our microservices which can be called by clients OR we can create a Service Gateway which routes the request to various microservices and responds to the clients.
- The Spring Cloud Gateway has three important parts to it. Those are –
- Route – These are the building blocks of the gateway which contain URL to which request is to be forwarded to and the predicates and filters that are applied on the incoming requests.
- Predicate – These are the set of criteria which should match for the incoming requests to be forwarded to internal microservices. For example, a path predicate will forward the request only if the incoming URL contains that path.
- Filters – These act as the place where you can modify the incoming requests before sending the requests to the internal microservices or before responding back to the client.

Pensioner Gateway Microservice

- port = 9000 for pensioner gateway microservice
- By using pensioner gateway microservice we need single port number for calling other microservices the examples are given bellow.
- Examples:-
 - 172.26.80.1:9000/login/authenticate
 - 172.26.80.1:9000/details/addpensioner
 - 172.26.80.1:9000/details/PensionerDetailByAadhaar/101
 - 172.26.80.1:9000/claims/ProcessPension/101

Autherization Microservice

- This service is responsible to provide login access to the application and provide security to it with the help of stateless authentication using JWT Tokens.
- Main Functionality---> To Generate the token and Validate the token.
- Step-1:Open Postman and follow the steps below:
- Step-2:Go to Url section and paste---> 172.26.80.1:9000/login/authenticate
- Method:POST [172.26.80.1:9000/login/authenticate]
- Step-3: Body Section: { "username":“user”, "password":“pass” }
- Step-4: Click Send button,then a token will be generated.Copy the token.

Pad

New Import

Overview

POST 172.26.80.1:9000

GET 172.26.80.1:9000

GET 172.26.80.1:9000

POST 172.26.80.1:9000

+

...

No Environment

v

⌂

+

☰

...



You don't have any collections

Collections let you group related requests, making them easier to access and run.

[Create Collection](#)

172.26.80.1:9000/login/authenticate

Save

v



</>

POST

v

172.26.80.1:9000/login/authenticate

Send

v

Params

Authorization ●

Headers (8)

Body ●

Pre-request Script

Tests

Settings

Cookies

● none

● form-data

● x-www-form-urlencoded

● raw

● binary

● GraphQL

JSON v

Beautify

```
1 {
2   "username": "user",
3   "password": "pass"
4 }
```

Body

Cookies

Headers (12)

Test Results



Status: 200 OK

Time: 552 ms

Size: 589 B

Save Response v

Pretty

Raw

Preview

Visualize

JSON v



```
1 {
2   "token": "eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJ1c2VyIiwiaXNjaWoxNjU1MDA5ODY2LCJpYXQiOiJlE2NTUwMDgwNjZ9.
3   i9kW6r1zcqUPI1NGmTisDo-QkjhxcwOKQMCC0oIKQVxuh_1m24ZZMXjK6MLPB20MRQoxPNGG6UPiVgicKQ9o9w",
4   "validity": "1800"
}
```

Pension Process Microservice

- It takes in Aadhaar number and determines the Pension amount and bank service charge Verifies if the pensioner detail is accurate by getting the data from PensionerDetailMicroservice or not. If not, validation message “Invalid pensioner detail provided, please provide valid detail.”. If valid, then pension calculation is done and the pension detail is returned to Postman or UI
- This microservice has 1 REST endpoint
- The GET endpoint calculate the Pension for the person through the Aadhaar number. It invoke the Pensioner detail microservice and get the salary detail.
- Pension amount calculation detail is as follows
 - Self pension:** 80% of the last salary earned + allowances
 - Family pension:** 50% of the last salary earned + allowances
- Bank service charge as follows
 - Public banks – INR 500
 - Private banks – INR 550

Pension Process Microservice

Steps For Process Pension

Step-1: Open Postman and follow the steps below:

Step-2: Go to Url section and paste---> 172.26.80.1:9000/claims/ProcessPension/101

Method: POST

[172.26.80.1:9000/claims/ProcessPension/101]

Step-3: Go to Body Section and add the below Json object.

```
{ "aadharnumber":101 }
```

Pension Process Microservice

Step-4:Go to Authorization Section: select token type as Bearer ,next paste the token there.

Step-5:Result

```
{  
  "pensionamount": {  
    "selfpension": 16400.0,  
    "familypension": 10250.0  
  },  
  "bankservicecharge": 500.0  
}
```




You don't have any collections

Collections let you group related requests, making them easier to access and run.

[Create Collection](#)

New Import

Overview

POST 172.26.80.1:9000

GET 172.26.80.1:9000

GET 172.26.80.1:9000

POST 172.26.80.1:9000

+

...

No Environment

▼

🔍

172.26.80.1:9000/claims/ProcessPension/101

Save

▼

✎

💬

</>

GET

▼

172.26.80.1:9000/claims/ProcessPension/101

Send

▼

Params

Authorization ●

Headers (7)

Body

Pre-request Script

Tests

Settings

Cookies

Type

Bearer To...

▼

The authorization header will be automatically generated when you send the request.

[Learn more about authorization](#)

! Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. [Learn more about variables](#)

✕

Token

eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJ1c2Vyli ...

Body

Cookies

Headers (6)

Test Results

🌐

Status: 200 OK

Time: 770 ms

Size: 296 B

Save Response

▼

Pretty

Raw

Preview

Visualize

JSON

▼

🔍

📄

🔍

```
1 {
2   "pensionamount": {
3     "selfpension": 16400.0,
4     "familypension": 10250.0
5   },
6   "bankservicecharge": 500.0
7 }
```

Activate Windows

Pensioner Details Microservice

- The intent of this Microservice is to provide the Pensioner detail based on Aadhaar number. Post Authorization using JWT, pensioner details microservice return data like the name, PAN detail, Bank name and bank account number, Salary Earned, Bank Type etc to the Process Pension Microservice.
- Pensioner Details Microservice be invoked from ProcessPensionmicroservice
- Pensioner Details Microservice is to fetch the pensioner detail by the Aadhaar number. This should be consumed by Process pension microservice.

Pensioner Details Microservice

Main Functionality-->

Step-1:Open Postman and follow the steps below:

Step-2:Go to Url section and paste---> 172.26.80.1:9000/details/addpensioner

Method:POST

[172.26.80.1:9000/details/addpensioner]

Step-3:Go to Body Section and add the below Json object.

```
{  
  "aadharno":"101","name":"kunal","dob":"01-01-1990",  
  "pan":"QWE4523","salaryearned":"20000",  
  "allowances":"500","pensiontype":"self",  
  "bankname":"sbi","accountno":900001,"banktype":"public"  
}
```

Pensioner Details Microservice

Step-4:Go to Url section and paste--

172.26.80.1:9000/details/PensionerDetailByAadhaar/101

Method:POST

[172.26.80.1:9000/details/PensionerDetailByAadhaar/101]

Then we get result

```
{
  "aadharno": 101,"name": "kunal","dob": "01-01-1990","pan": "QWE4523",
  "salaryearned": 20000.0,"allowances": 500.0,"pensiontype": "self","bankname": "
sbi",
  "accountno": 900001,"banktype": "public"
}
```



You don't have any collections

Collections let you group related requests, making them easier to access and run.

[Create Collection](#)

New

Import

Overview

POST 172.26.80.1:9000

GET 172.26.80.1:9000

GET 172.26.80.1:9000

POST 172.26.80.1:9000

+

...

No Environment

▼

172.26.80.1:9000/details/addpensioner

Save

▼

✎

💬

POST

▼

172.26.80.1:9000/details/addpensioner

Send

▼

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

▼

Beautify

```
1 { "aadhar": "101",
2   "name": "kunal",
3   "dob": "01-01-1990",
4   "pan": "QWE4523",
5   "salary": "20000",
6   "allowances": "500",
7   "pension": "self",
8   "bank": "sbi",
```

Body

Cookies

Headers (3)

Test Results

🌐

Status: 200 OK

Time: 2.07 s

Size: 301 B

Save Response

▼

Pretty

Raw

Preview

Visualize

JSON

▼

🔄

📄

🔍

```
1 {
2   "aadhar": 101,
3   "name": "kunal",
4   "dob": "01-01-1990",
5   "pan": "QWE4523",
6   "salary": 20000.0,
7   "allowances": 500.0,
8   "pension": "self",
```

Activate Windows



You don't have any collections

Collections let you group related requests, making them easier to access and run.

[Create Collection](#)

New Import

Overview

POST 172.26.80.1:9000

GET 172.26.80.1:9000

GET 172.26.80.1:9000

POST 172.26.80.1:9000

+ ...

No Environment

▼

👁

172.26.80.1:9000/details/PensionerDetailByAadhaar/101

Save

▼

✎

🗨

</>

GET

▼

172.26.80.1:9000/details/PensionerDetailByAadhaar/101

Send

▼

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
	Key	Value	Description		

Body

Cookies

Headers (3)

Test Results

🌐

Status: 200 OK

Time: 318 ms

Size: 301 B

Save Response

▼

Pretty

Raw

Preview

Visualize

JSON

▼

🔍

🗨

🔍

```
1 {
2   "aadharNo": 101,
3   "name": "kunal",
4   "dob": "01-01-1990",
5   "pan": "QWE4523",
6   "salaryearned": 20000.0,
7   "allowances": 500.0,
8   "pensiontype": "self",
```

Activate Windows

Go to Settings to activate Windows.

Runner

Trash

Replace

Console

Thank You

