**Pension Management System**

**MicroServices**

1. Process Pension Service
2. Pension Detail Service
3. Authorization Service
4. Pension Management Portal Service
5. Cloud API Gateway
6. Config-Server

**Hystrix Dashboard is required to do the fault tolerance**

**Eureka Discovery Services is required**

**System Architecture**

Graphical user interface, diagram

Description automatically generated

## Rest API (Products & Frameworks -> Compute & Integration):

* 1. Use Spring Boot to version and implement the REST endpoints.
  2. Implement HTTP methods like GET, POST, PUT, DELETE, PATCH to implement RESTful resources:

|  |  |  |
| --- | --- | --- |
| POST | /api/v1.0/ PMS / ProcessPension/{adhar\_no} | Calculate the pension for the person |
| POST | /api/v1.0/ PMS /auth/signup | Pensioner Registration |
| POST | /api/v1.0/ PMS /auth/signin | Pensioner Login |
| GET | /api/v1.0/ PMS /PensionDetailsByAdhaar/{adhar\_no} | Get the Pensioner Details |
| POST | /api/v1.0/ PMS /PensionManagement | Update the Details in database |

* 1. Use necessary configuration in place for REST API in application.properties or application.yml; whichever is applicable.
  2. Package Structure for Spring Boot Project will be like com.pensionmgmt.\* with proper naming conventions for package and beans.
  3. Use configuration class annotated with @Configuration and @Service for business layer.
  4. Use constructor-based dependency injection in few classes and setter-based dependency injection in few classes.
  5. Follow Spring Bean Naming Conventions

**Beans**

**ProcessPensionInput** -- String adhar\_number

**PensionAmount** -- float pension\_amount, float bankServiceCharge

**PensionerDetail** – String name, Date dob, String pan, float salaryEarned, float allowances, PensionType pensionType, BankDetails bankDetails,

**PensionType – {SELF, FAMILY}**

**BankDetails** –String bankName, String accountNumber, BankType banktype.

**BankType – {PUBLIC, PRIVATE}**