Kristina Cecura 2495184

**Banking Management System - RESTful Web Service Report**

**Project Overview**

This project implements a Banking Management System using Jakarta RESTful Web Services in a Maven Dynamic Web Project. The system allows banking staff to retrieve and search customer and account data using various REST endpoints. The responses are available in both HTML and JSON formats.

**Business Impact**

The implementation of RESTful Web Services in this Banking Management System brings significant benefits to users:

* **Improved Accessibility**: Bank staff can access account and customer data
* **Extensibility**: New features such as POST or DELETE can be added in the future without changing the existing endpoints.
* **Efficiency**: Services like searching by balance, customer name, or account type enhance the speed and accuracy of managing customer data, reducing manual processing and improving customer satisfaction.

**Design Architecture**

The core architectural components include:

* **@Path**: This annotation is used to define the base path for the resource classes. For example, @Path("/WebCustomer") exposes all customer-related endpoints under the /WebCustomer route.
* **@GET**: Specifies that the method will respond to HTTP GET requests. This is used for retrieving data from the server.
* **@Produces**: Declares the type of media the endpoint returns. For example:
  + @Produces(MediaType.TEXT\_HTML) is used to return data in HTML format.
  + @Produces(MediaType.APPLICATION\_JSON) returns JSON format
* **@QueryParam**: Extracts values from the query string in a URL. For example:
  + @QueryParam("name") String name allows filtering customers by name using /WebCustomer/search?name=John.

Each major business entity (e.g., **Customer**, **Account**) has a dedicated resource class:

* WebCustomerResource: Handles requests related to customers, including listing all customers and searching by name.
* WebAccountResource: Handles account operations, including listing accounts, searching by type, balance range, and customer ID.

**Data Structure Used**

This project uses the **ArrayList** data structure to store and manage Customer and Account objects.

* In the BankData class (Singleton), two ArrayList objects are used:

ArrayList<Customer> customerList

ArrayList<Account> accountList

These lists allow dynamic resizing, which is useful as the number of customers/accounts can grow.

Although ArrayList is not the most efficient for large-scale searches, it is simple and sufficient for small applications like this one.

These data structures are shared across the application via a singleton BankData instance, ensuring consistent access throughout all REST endpoints.

**Project Structure**

The project includes the following key components:

**Classes**

Customer.java

Account.java

**Data Source Singleton**

BankData.java (hardcoded list of customers and accounts)

**REST Resource Classes**

WebCustomerResource.java

WebAccountResource.java

**Implemented RESTful Endpoints**

1. **Customer Endpoints**

WebCustomer HTML response in browser

A screenshot of a computer

AI-generated content may be incorrect.

WebCustomer/json JSON response in Postman

A screenshot of a computer

AI-generated content may be incorrect.

WebCustomer/search?name=Alice result in Postman

A screenshot of a computer

AI-generated content may be incorrect.

1. **Account Endpoints**

WebAccount HTML response in browser

A screenshot of a computer

AI-generated content may be incorrect.

WebAccount/json JSON response in Postman

A screenshot of a computer

AI-generated content may be incorrect.

WebAccount/search?account=Savings result

A screenshot of a computer

AI-generated content may be incorrect.

WebAccount/searchByBalance?min=500&max=3000 result

A screenshot of a computer

AI-generated content may be incorrect.

WebAccount/searchByCustomer?id=C002 result

**A screenshot of a computer

AI-generated content may be incorrect.**

**Tools and Technologies Used**

* **Java 17**
* **Jakarta RESTful Web Services (JAX-RS)**
* **GlassFish Server 7 / Payara**
* **Postman** for testing JSON endpoints
* **Eclipse IDE**
* **Maven** for dependency and project management

**pom.xml Dependencies Summary**

Key dependencies used:

* jersey-container-servlet – JAX-RS servlet container
* jersey-server – Core Jersey REST server
* jersey-media-json-binding – JSON serialization
* javax.servlet-api – For servlet functionality

A screenshot of a computer program

AI-generated content may be incorrect.