Student ID: _____

Full Names: _____

Applied Software Development (CS489)

(October 2025)

Instructor: O. Kalu

GSA Selection Exercise (Enterprise Web App – Design and Development)

(First finish: temuujin: 1hour 20 minutes)

(2nd finish: Luiz: 1hour 50 minutes)

(3rd finish: Sai: 1hour 55 minutes)

1. Estimated time for this test is 2.5 hours.
2. You are allowed/expected/encouraged to use computer and any tools, including AI for the coding tasks.
3. **This test is a restricted material and must not be taken away, or copied or photographed or reproduced or transferred or shared or distributed**.
4. You are expected to use an IDE or any Code Editor tool of your choice to implement your solutions for the Coding.
5. Upon completion, put your entire project(s), including screenshot images, into a single zip file named **GSATest.zip**, and send to Prof. Kalu via Microsoft Teams chat.

Make sure to include the screenshots of your results, as required.

(CS489 - ApSD)
(October 2025)
GSA Test - Coding (100 points)

Test of Software Development, Problem-solving, Design and Coding skills

Note: *For the tasks in this question, you are expected to take screenshot(s) of your result(s), save each into a .png or .jpg image file, placed inside a folder named, screenshots and include these in the project folder, which you zip into a GSATest.zip file, you submit. From your own solution, you are required to take each of the set of 5 or more evidential sample screenshots, which have been included at the end of the question.*

1. (100 points) **Implement an end-to-end, full-stack data-driven enterprise web application**

Assume you have been hired by a bank named, The Special Bank of Burlington. And they want you to design and develop a basic web-based software solution, which they call, "Customer-Accounts Management system" (CAMS), which they will be using to manage their customer-accounts data.

Especially important to the Bank Manager is, the current Liquidity Position of the bank, which is calculated by summing the total balance of all the customer-accounts. Also important to the Bank Manager, is the **Prime Accounts. A Prime Account** is account whose balance is greater than $10,000.

For the purpose of this GSA Test, here is a description of the simplified solution model for the system:

A Customer can own one and only one Account.
And, each Account is owned by only one Customer.

Your solution model should consist of only the following two data entity:

1. Customer
2. Account

Here are the attributes for the entities, including some useful descriptions and/or sample data values:

Customer:

customerId: long, (Primary Key field)

firstName, (required field) (e.g. Bob, Anna, Carlos etc.)

lastName, (required field) (e.g. Jones, Smith etc.)

Account:

accountId: long, (Primary Key field)

accountNumber, (required field; unique) (e.g. AC1001, AC1002, etc.)

accountType, (required field) (e.g. Checking, Savings)

dateOpened, (optional field) (e.g. 2021-12-3, 2022-5-21, etc.)

balance, (required field) – This is the amount of money (in dollars and cents) in the account

Data:

Here is the Bank's existing sample data, which you are expected to input into your database:

Customer data:

Customer Id	First Name	Last Name
1	Bob	Jones
2	Anna	Smith
3	Carlos	Jimenez

Account data:

Account Id	Account Number	Account Type	Date Opened	Balance	Customer Id
1	AC1002	Checking	2022-07-10	10900.50	2
2	AC1001	Savings	2021-11-15	125.95	1
3	AC1003	Savings	2022-07-11	15000	3

For this question, you are required to do the following:

1. Draw a Domain model UML Class diagram for the solution.

2. Using your preferred set of tools, technologies and frameworks, such as Java, Spring Boot, Spring Web, Spring Data, etc., (or some other Enterprise Web application development language/platform/tool(s) that you prefer), implement a working web application for the Bank. You may use any database of your choice.

You are expected to implement only the following features and use-cases:

1. Display a homepage which presents a set of menu options. (see sample screenshot below)

2. Display list of all Customer-Accounts (Allows the user to view a list of all the customer-accounts registered in the system). The Bank requires this list to be displayed sorted in descending order of the Account balance amounts.

   Also, display at the bottom of this list, the Liquidity Position of the bank, which is the sum of all Account balances. (see sample screenshot below)

3. Implement a Web Form UI and backend functionality for creating a new Customer-Account.

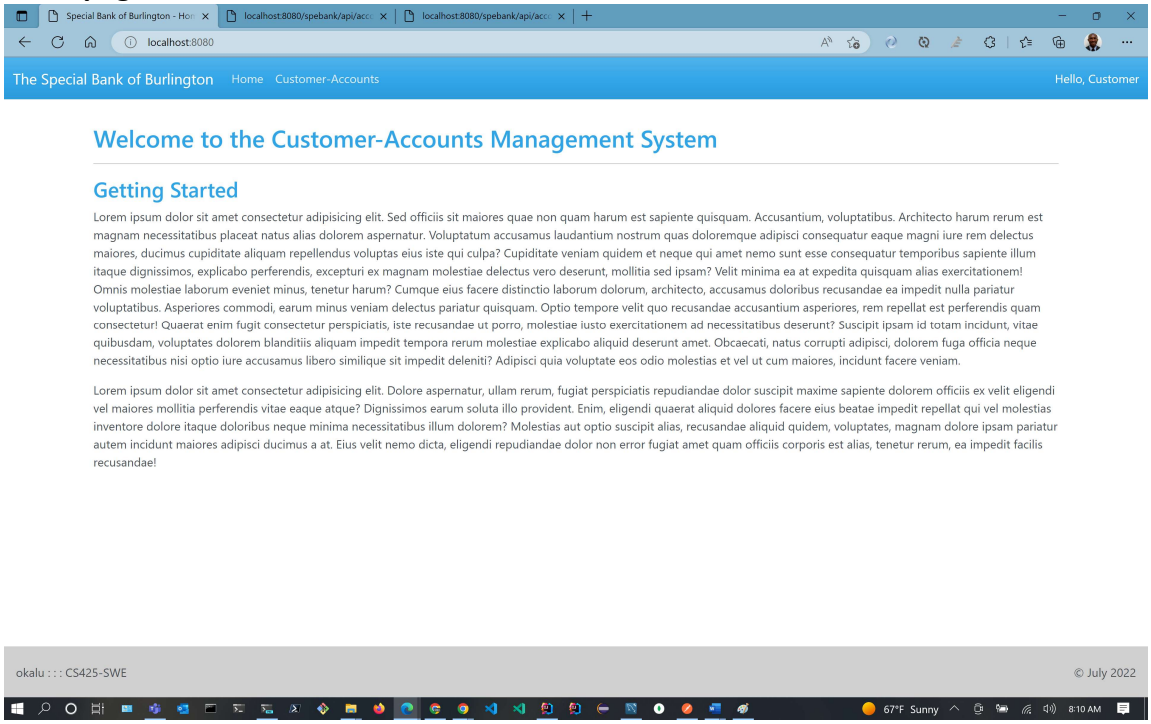
4. Implement a RESTful Web Service Endpoint (i.e. Web API) url which presents the list of all Customer-Accounts in JSON format.

5. Implement a RESTful Web Service Endpoint (i.e. Web API) url which presents the list of only Prime Customer-Accounts in JSON format. Note: Prime Customer-Accounts are accounts that have more than $10,000.00.

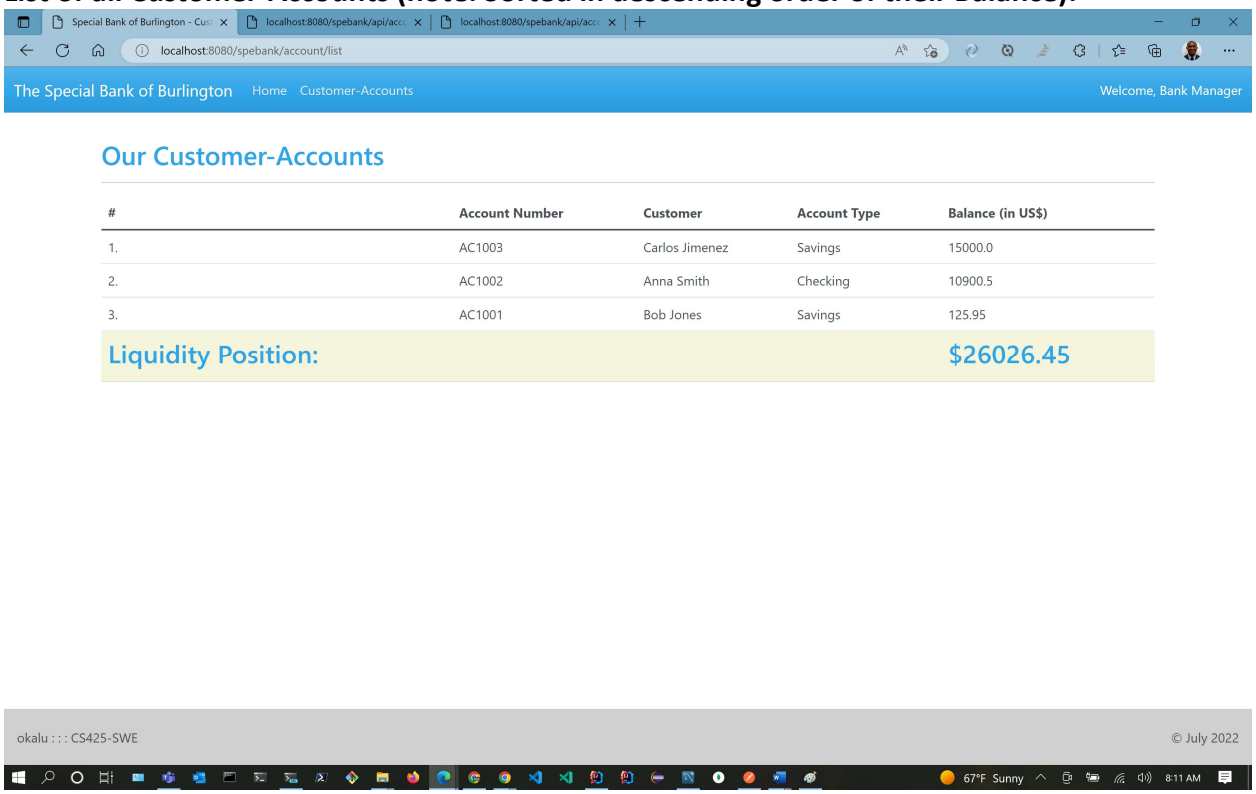
Shown below are sample User Interfaces and data presentation for the above features/use-cases.

Note: Your own UI design does NOT necessarily have to look exactly like these samples. But your UIs should contain/present all the necessary data and data fields, as required.

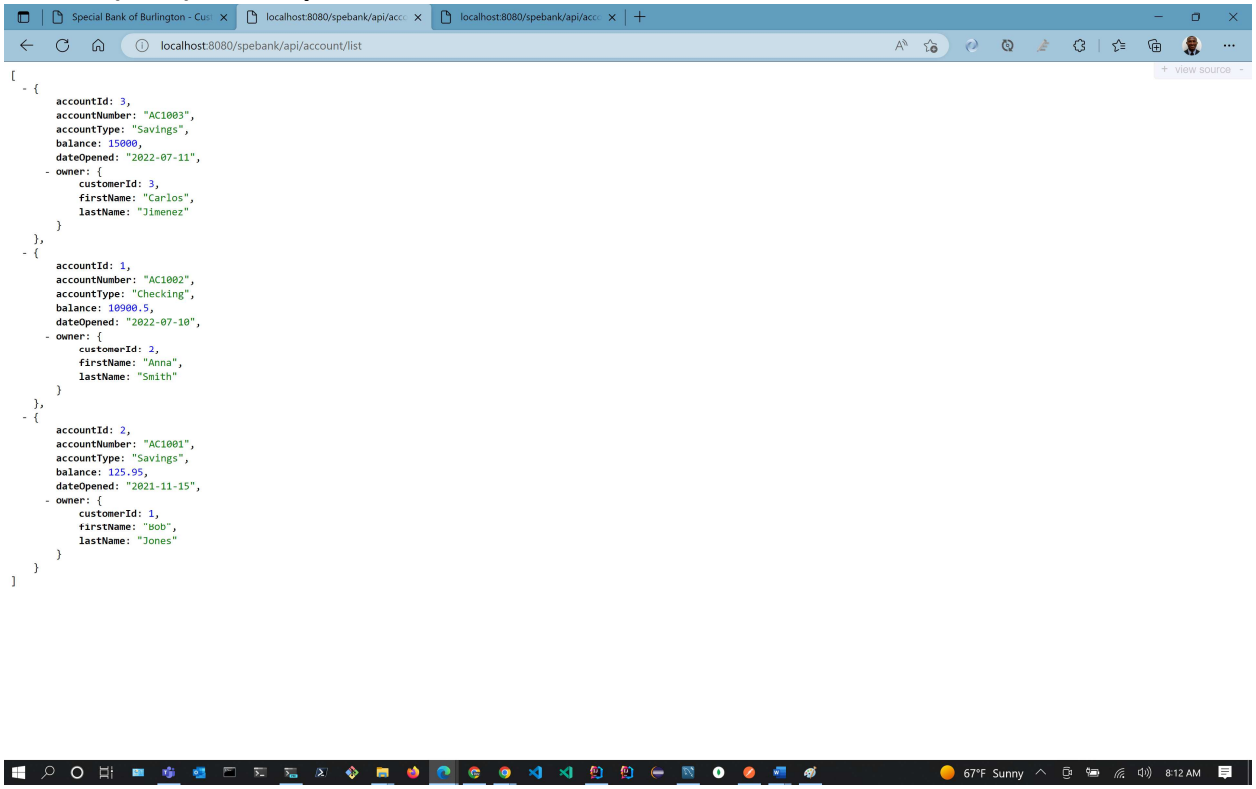
Homepage:



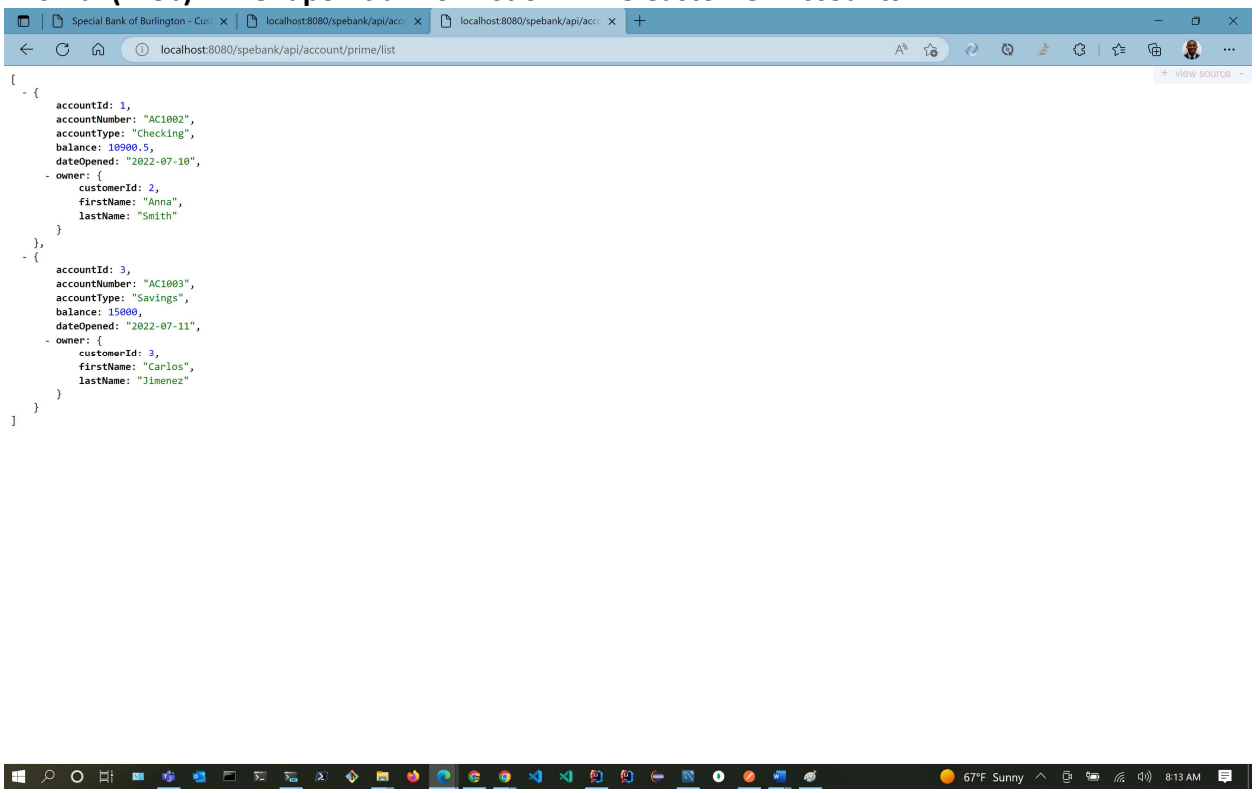
List of all Customer-Accounts (note: Sorted in descending order of their Balance):



RESTful (Web) API endpoint url for List of all Customer-Accounts:

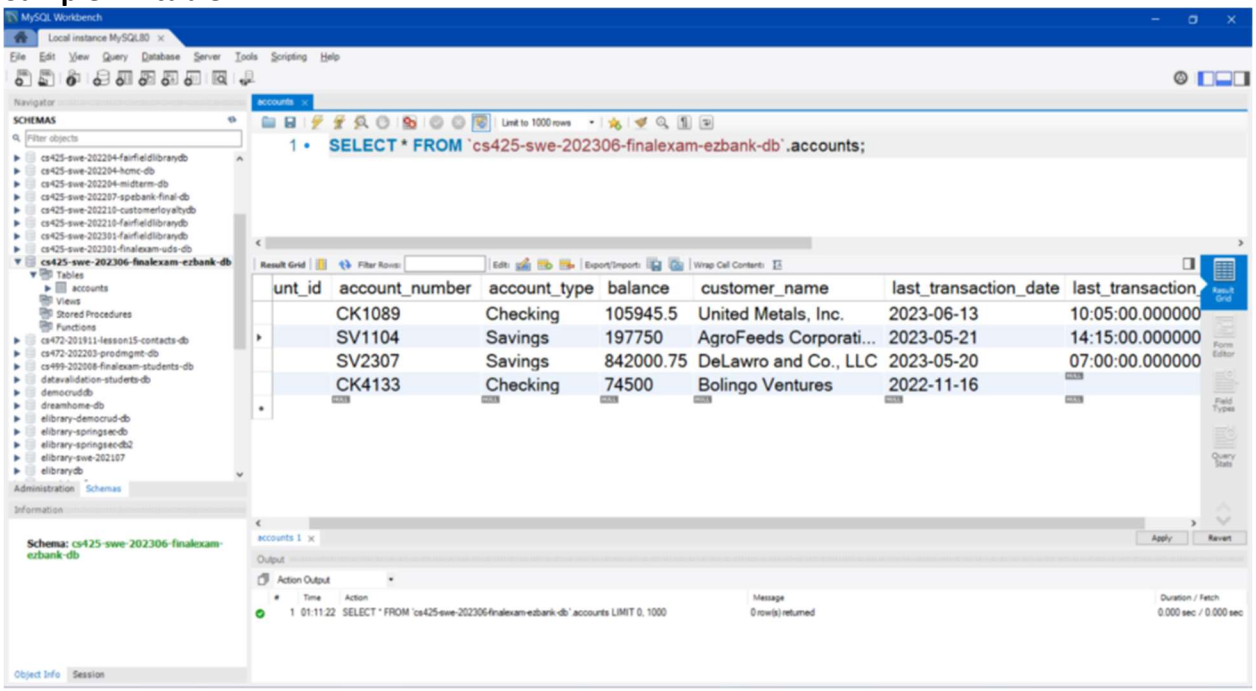


RESTful (Web) API endpoint url for List of Prime Customer-Accounts:



Database Tables screenshot (take a screenshot of your database table(s), showing the data, and similar to the one pasted below):

Sample DB table



//-- The End --//