**Student ID:** \_\_\_\_\_987096\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Full Names:** \_\_\_\_\_Abeba Ibrahim Yimer\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Software Engineering

(CS425)

(May 2019)

Professor: O. Kalu

Final Integration Exercise

1. The exam duration is 2 hours.
2. The exam is computer-based; so you may use a computer for both the coding and theory parts.
3. Make sure to switch-off your cell-phones or simply turn the ringer off.
4. **This exam is a copyrighted material and must not be copied or reproduced or transferred**.
5. You are expected to use an IDE or any Code Editor tool of your choice to implement your solutions for the questions in the Coding part. Upon completion, put your project(s), **(source code only)** in a single zip file named **FinalExam.zip**, including your completed/finished exam paper (i.e. this document – **in Microsoft Word or PDF format, only**), and submit to Sakai.

--------------------------------------------------------------------------------------------------------------------

Type your answers to the theory questions in the following pages.

--------------------------------------------------------------------------------------------------------------------

(CS425 - SWE)

(May 2019)

Final Examination (70 points)

**Part I – Theory (True/False, Short answers, Multiple-choice questions):** (13 points)

1. (2 points) **Science of Consciousness**:

Answer the following question; giving 2 sentences in support of your answer:

State two (2) reasons why our daily practice of Transcendental Meditation can be considered to be an Agile technique for improving one’s brain performance?

**Answer:**

Agile technique uses **incremental** and **iterative** work sequences so In TM we do it iteratively and the benefits we get increment every day.

1. (3 points) Answer the following questions with True or False.
   1. (1 point) In the Rational Unified Process model for Software development, the Use-case design activities must take place before Architecture design activities are performed. True or False?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_False\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* 1. (1 point) In a software development project where the Agile methodology is applied, the software requirement features are developed in small, incremental steps which are completed through multiple iterations, resulting in frequent updated releases of the software product. True or False?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_True\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* 1. (1 point) In the RUP process model, during the Architecture Design activity, the class coded below will be globally visible to all other classes across the system. True or False?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_False\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**package** edu.mum.cs.cs425.prodmgmt.global.publicc;

**class** GlobalPublicUtils {

**public** **static** **final** **long** ***MAX\_INVENTORY\_COUNT*** = 10000L;

// **TODO**

}

1. (8 points) Give short answers to the following questions.
   1. (2 points) Explain the difference between the following 2 JPA annotations. (You may give examples to illustrate your answer).
      1. @Column(name="ProductNumber", **nullable=false**) private long productNumber;
      2. **@NotNull** private long productNumber;

**nullable=false🡪** It is to check the nullability of database table column. It will create NOT NULL column

**@NotNull** 🡪 It is java related annotation to check nullability of text box in our UI validation. Java will throw exceptions when this attribute is not set.

* 1. (2 points) With respect to relationship between two classes/entities, what do we mean by the term, Dependency. Give an example using code snippets and/or diagram.

If we use one object uses the other object as a method parameter, return type or local variable, it is dependency (temporary relationship)

Public class A

{

Public void myMethod(B b)

{

b.callMethod();

}

}

The relationship between A and B is dependency

* 1. (2 points) With respect to relationship between classes/entities, what do we mean by the term, Composition? And how does it differ from an Aggregation? Give examples to illustrate your answer.

Composition🡪In composition the object only exists inside the other object (as part of the other)

Eg. **Company** has **department**. If the company doesn’t exist the department can’t exist too.

Aggregation🡪 In aggregation the object can exist outside the other object

Eg. **Company** has **Person** as employees. The employee can exist even if the the company doesn’t exist.

* 1. (2 points) In the Agile methodology, as practiced at Microsoft, what is ATDD?

ATDD is closely related to Test Driven Development (TDD), but stands out because of its highly collaborative approach to effectively write code that is correct and delivers the required business value.

**Part II – Software Engineering Problem-solving, Coding skills:** (57 points)

**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(s) and include these in the FinalExam.zip file, you submit. From your own solution, you are required to take each of the set of 10 evidential sample screenshots, which have been included below.*

1. (57 points) **Implementing an Enterprise Web Application**

A local regional bank, named MidWesTen Bank, has hired you to design and develop a simple banking software solution for them, which they will be using to run part of their banking business; specifically, the system will be used to collect, maintain and manage data about their customers and the bank accounts they operate. They want you to implement a basic web application for this purpose. Especially important to the Bank manager is, the data that provides information about the net liquidity of the bank, on any given day.

The bank’s operational data model is given as follows:

A Customer can own **many** Accounts.

An Account belongs to just **one** Customer.

An Account must be of **one** of many possible AccountTypes.

The Bank currently offers the following 3 AccountTypes:

1. Checking account
2. Loan account
3. Savings account

Your solution model should consist of the following three data entities:

1. Customer
2. Account
3. AccountType

Here are the attributes for the entities:

**Customer**: customerId:long, customerNumber:long, firstName:string, middleName:string, lastName:string, emailAddress:string, contactPhoneNumber:string, dateOfBirth:date

**Account**: accountId:long, accountNumber:long, balance (Note: the account balance represents the amount of money in dollars and cents held in the account)

**AccountType**: accountTypeId:int, accountTypeName:string

Here are excerpts of the Bank’s existing data, which you are expected to input into your new banking software system:

**AccountTypes:**

AT1: { accountTypeId:1, accountTypeName: Checking}

AT2: { accountTypeId:2, accountTypeName: Loan}

AT3: { accountTypeId:3, accountTypeName: Savings}

**Customers:**

C1: {customerId:1, customerNumber:10001, firstName:Anna, middleName:””, lastName:Smith, emailAddress:asmith@gmail.net, contactPhoneNumber:(641) 451-0001, dateOfBirth:1978-5-21}

C2: {customerId:2, customerNumber:10002, firstName:Bob, middleName:Earl, lastName:Jones, emailAddress:bob.e.jones@earthlink.org, contactPhoneNumber:(319) 001-0001, dateOfBirth:1964-12-7}

**Accounts:**

A1: {accountId:1, accountNumber:100001, balance:$190,590.95} – belongs to customer, C1; This account is of the AccountType, Savings.

A2: {accountId:2, accountNumber:100002, balance:$60,000.00} – belongs to customer, C1; This account is of the AccountType, Loan.

A3: {accountId:3, accountNumber:100003, balance:$354,005.26} – belongs to customer, C2; This account is of the AccountType, Checking.

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

1. Using a suitable tool, draw a simple UML Static (class) model for the software solution.

2. Using the set of tools, technologies and frameworks which you have learnt about in this CS425 course, including Spring Boot, Spring Web MVC, Spring Data JPA, etc., (or some other Enterprise Web application development platform/tool(s) that you prefer), implement a working web application for MidWesTen Bank. You may use any database of your choice.

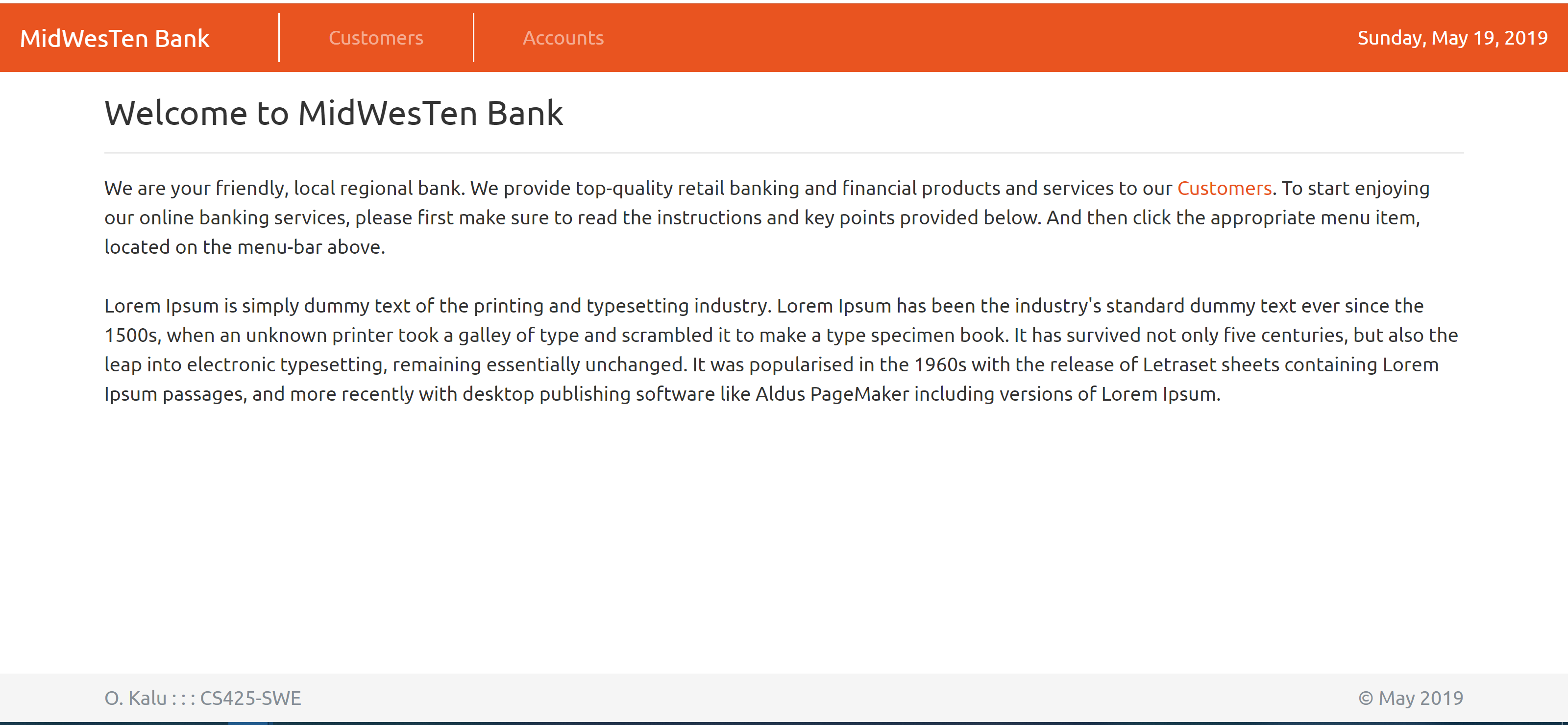
You are expected to implement only the following features and use-cases:

1. Display a homepage which presents a set of menu options.
2. Display list of Customers (Allows the bank manager to view a list of all the Customers registered in the system). The bank requires this list to be displayed sorted in ascending order of the Customers’ last names (see sample screen below).
3. Register a new Customer (Allows the bank manager to add a new Customer into the system).
4. Display list of Accounts (Allows the bank manager to view a list of all the Accounts held in the system). The bank requires this list to be displayed sorted in ascending order of the Account Number (see sample screen below).
5. Open a new Account (Allows the bank manager to add/open a new Account for an existing customer in the system).
6. Display the bank’s net liquidity position (This means the system presents to the bank manager, a computed value that represents the bank’s net liquidity. This value is computed by taking the sum of the Savings and Checking accounts balances minus the sum of the Loan accounts balances. It represents the total amount of cash expected in the bank’s vault). Your solution should display this data at the bottom of the “List of Accounts” table (see sample screen below).
7. Using the JUnit framework, write a Unit Test case for your function/method that computes the bank’s net liquidity. Be sure to execute your Test-case and take a screenshot of your result, as displayed by your IDE.

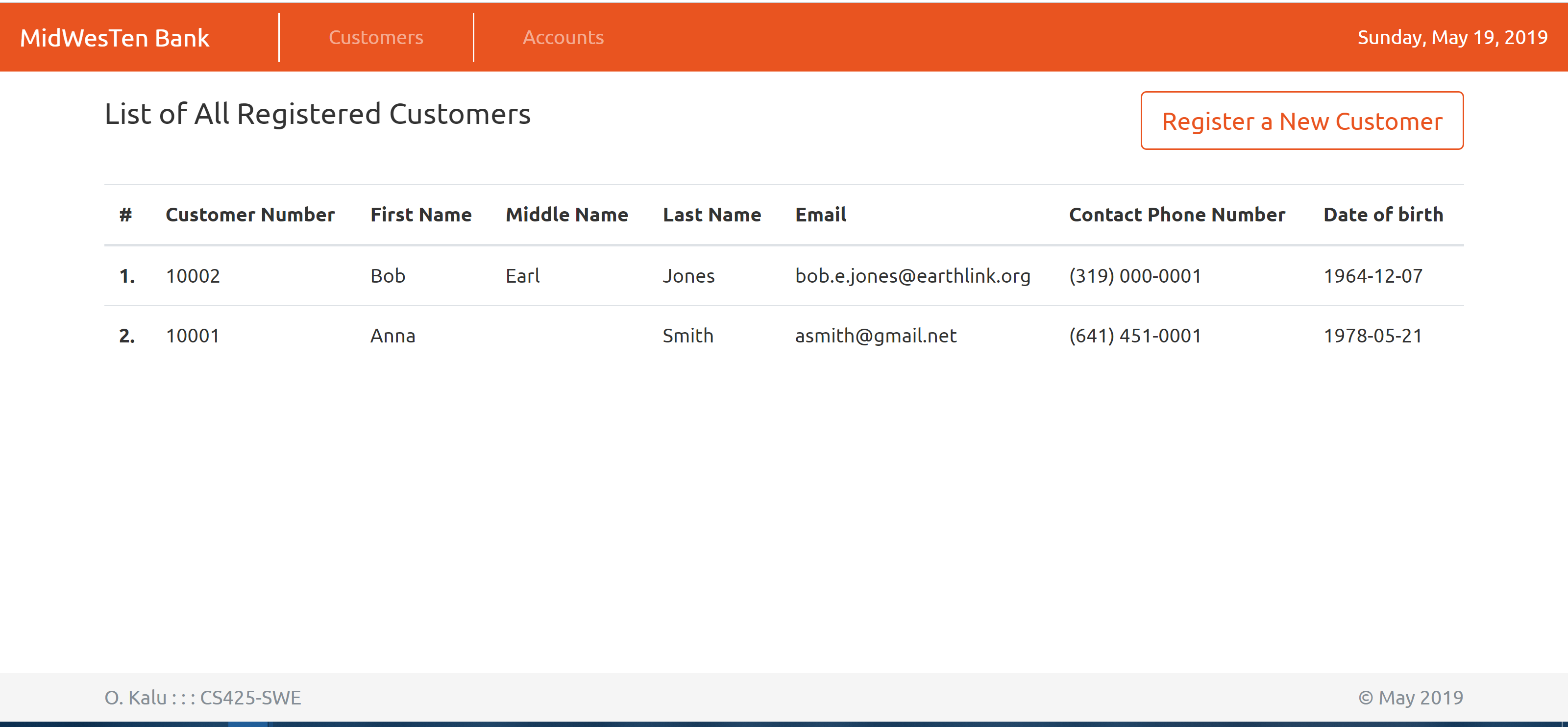
Shown below are sample User Interfaces for the use-cases/tasks.

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

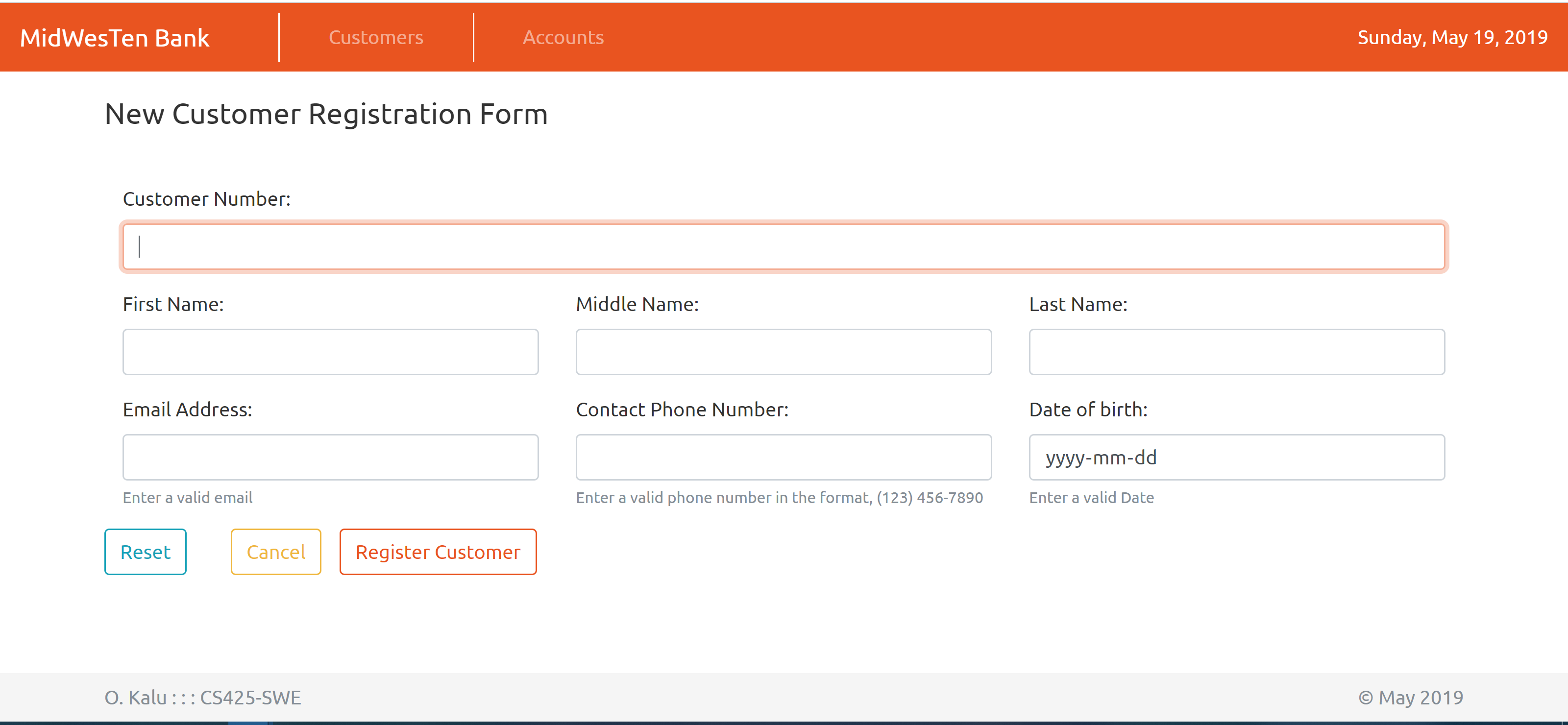
**Homepage:**

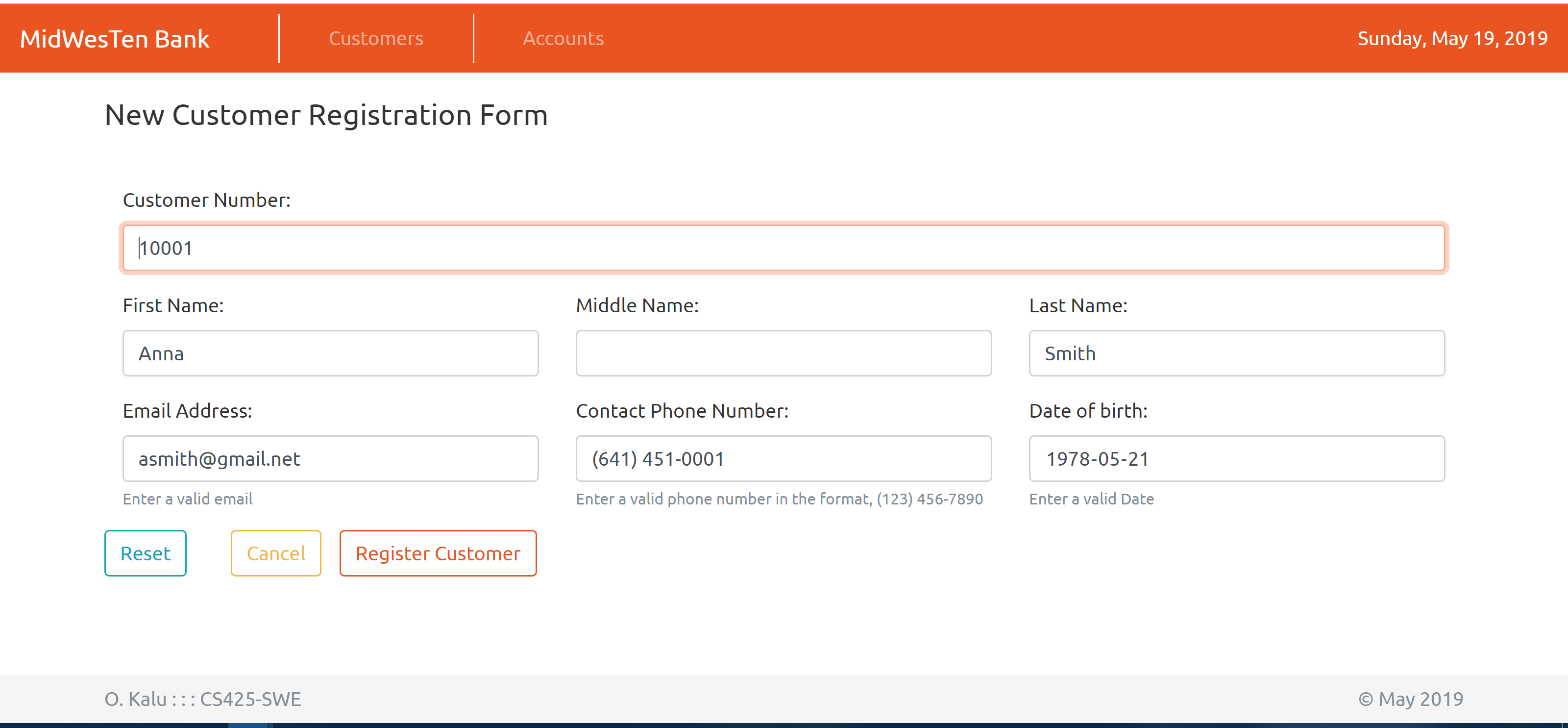


**List of all Registered Customers (note: Sorted by Last Name):**

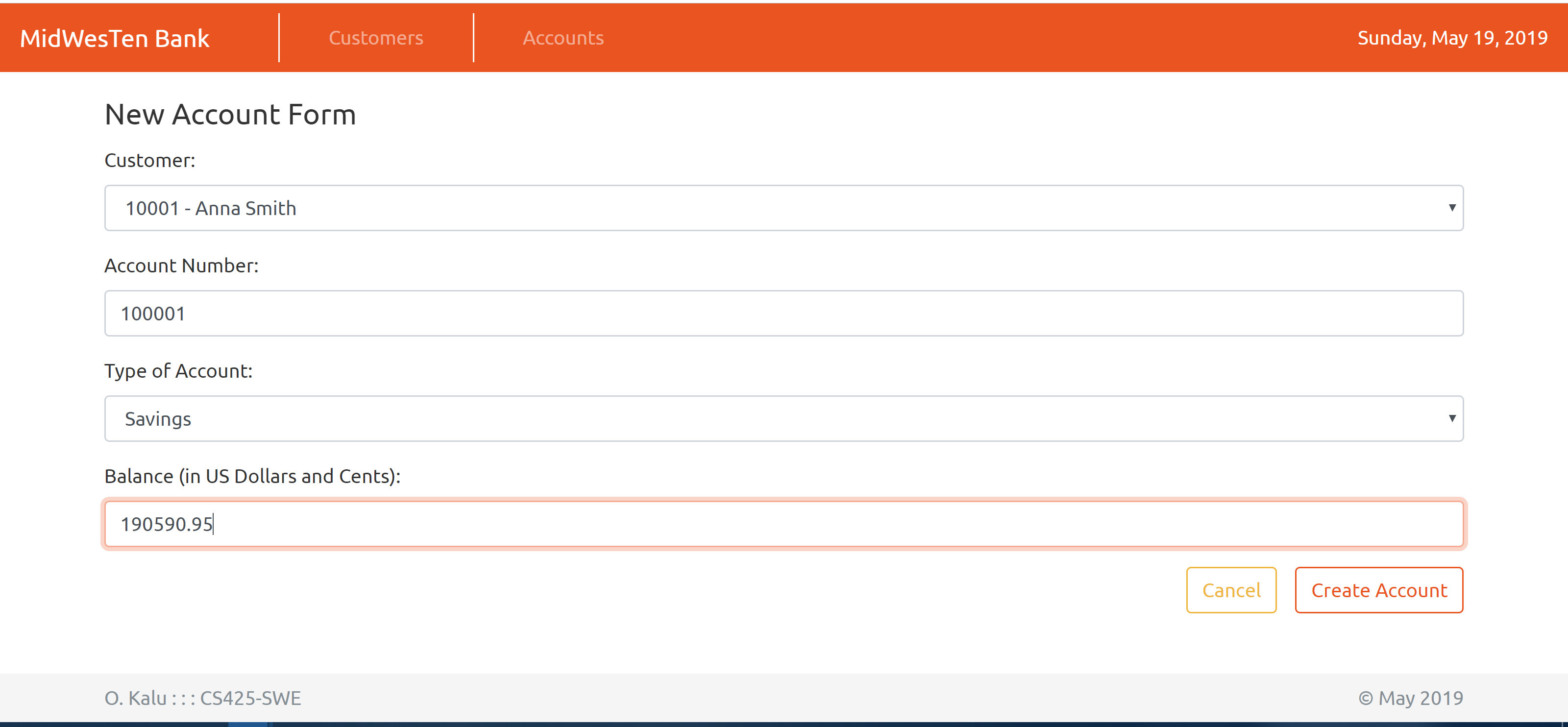


**Add/Register a new Customer:**

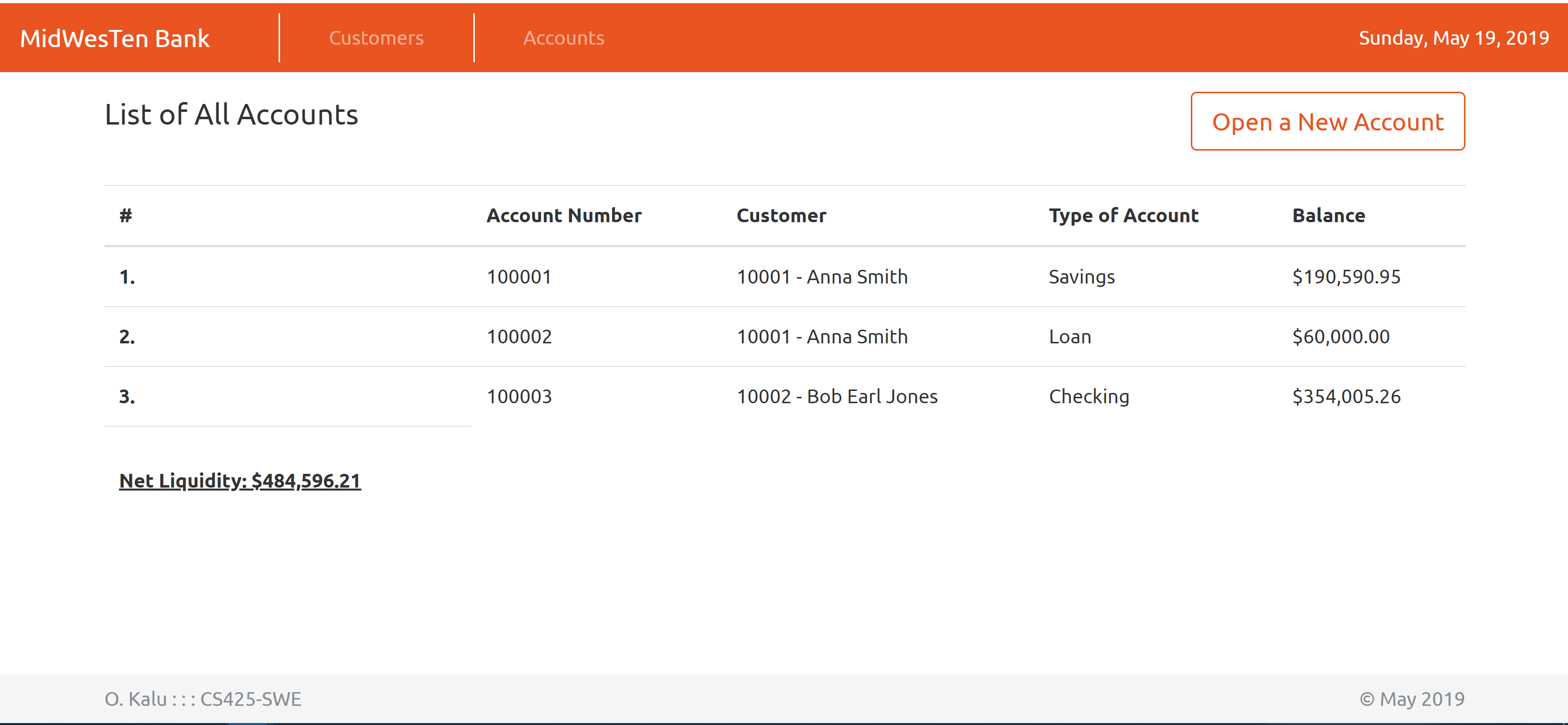




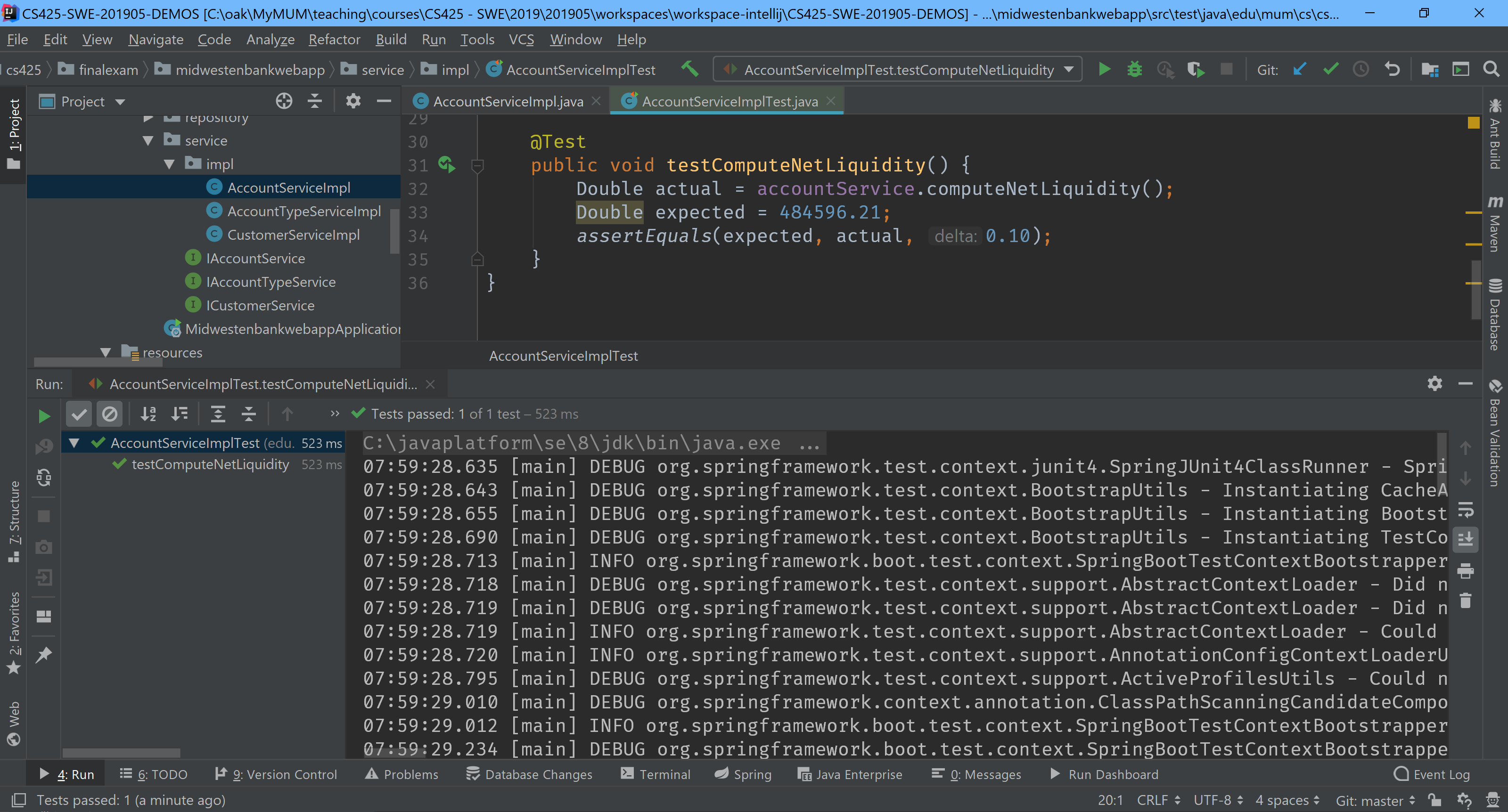
**Add/Open a new Account**



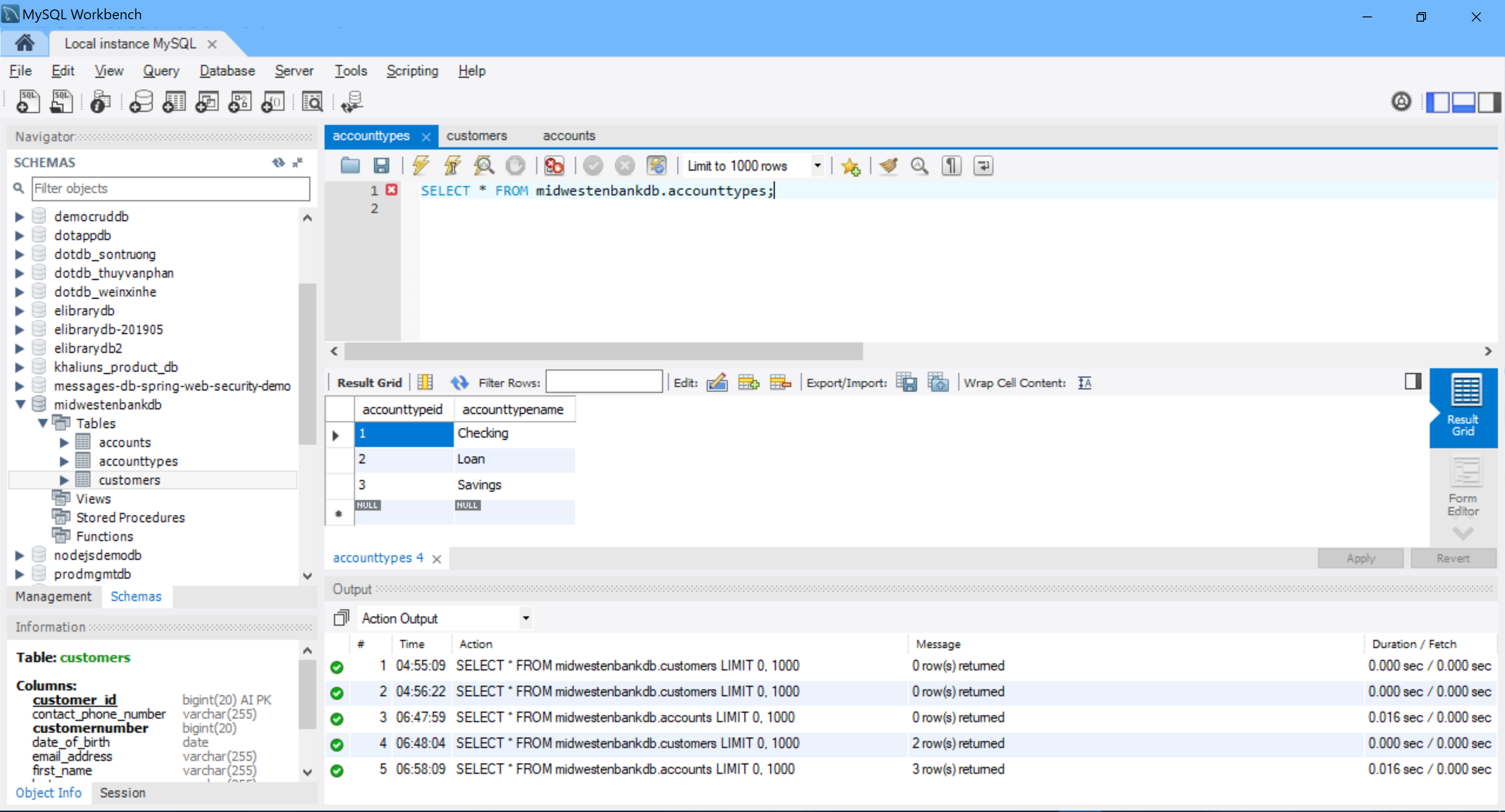
**List of Accounts: (note: Sorted by Account Number and displaying the Net Liquidity):**

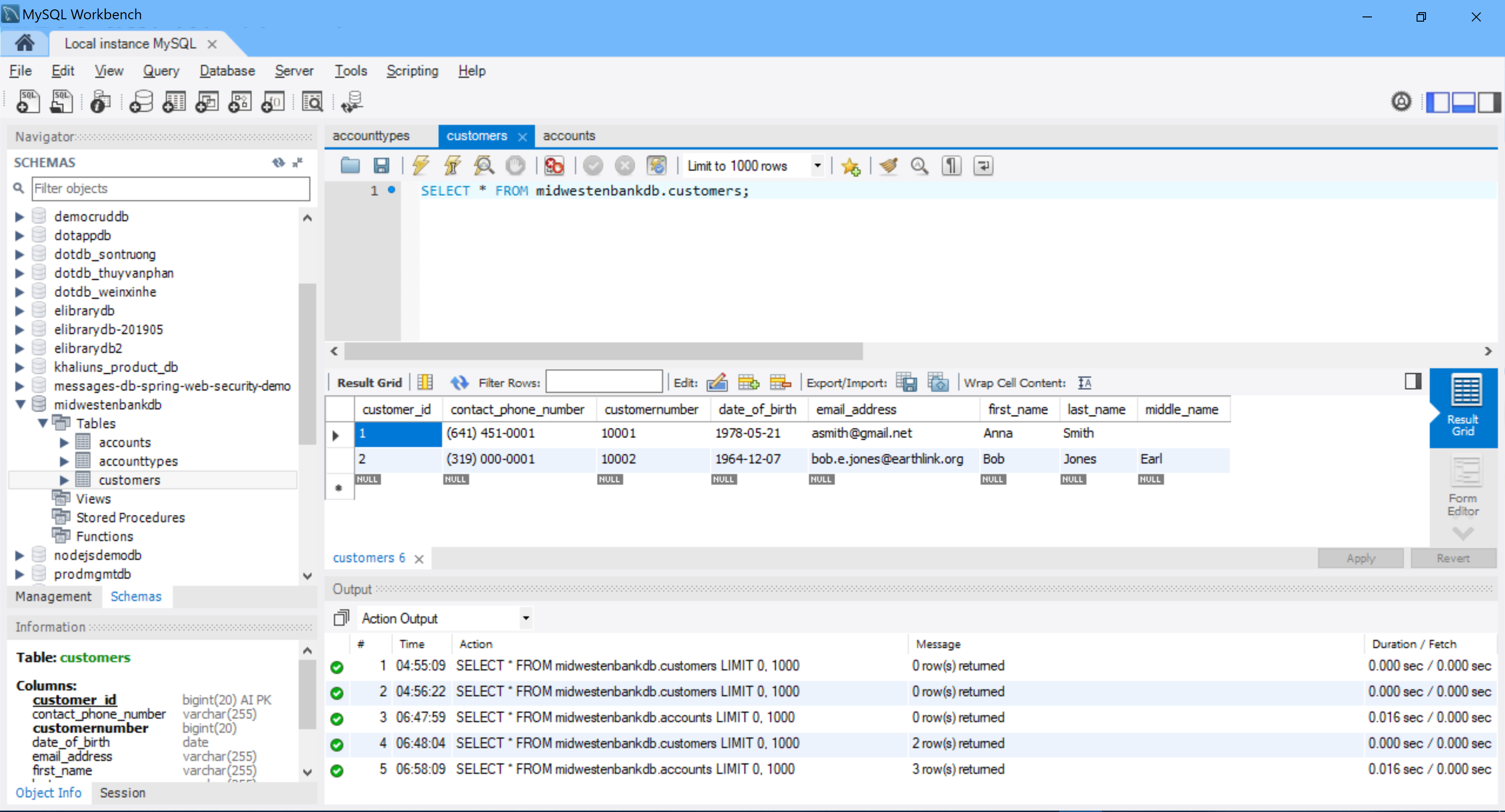


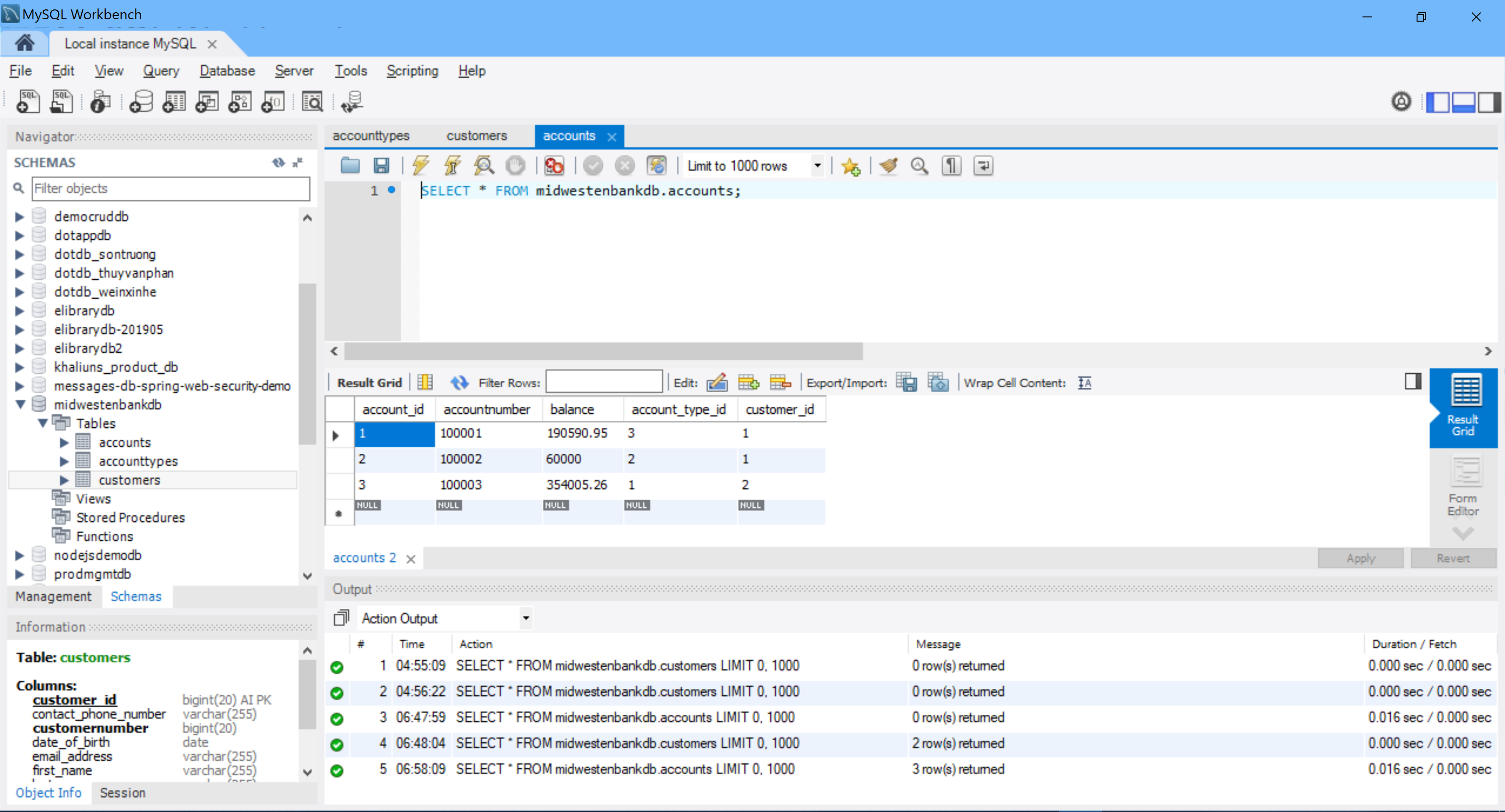
**JUnit Test of computeNetLiquidity method**

****

**Database Tables screenshots:**

****

****

****

**//-- The End --//**