User Stories

Customer:

- **As a customer I want to be able to deposit money into my account, so that I have an overview of my economy.**

Acceptance criteria:

The customer can deposit his/her money, and the account balance will be updated.

- **As a customer I want to be able to withdraw money from my account, so I can spend them.**

Acceptance criteria:

The customer can withdraw his/her money, and the account balance will be updated.

- **As a customer I want to be able to have a general view of my transactions.**

Acceptance criteria:

An overview of the customer's transactions can be displayed.

Bank:

- As a bank I want to be able to transfer money between customers' accounts, so that we have control over the money-flow in our business.

Acceptance criteria:

The bank can transfer funds between the accounts of two customers, and the two customer accounts will be updated.

Use Cases

Use Case 1 – Withdraw funds

Primary actors:

Customer

Bank

Basic flow of events:

1. Customer accesses bank.
2. Bank prompts available actions.
3. Customer chooses: ”Withdraw funds”.
4. Bank prompts for amount.
5. Customer enters amount.
6. Bank checks if the account has sufficient funds.
7. Funds are withdrawn from account.

Alternative flows:

8a. Account has insufficient funds.

8a1. Bank displays error message.

Use Case 2 – Deposit funds

Primary actors:

Customer

Bank

Basic flow of events:

1. Customer accesses bank.
2. Bank prompts available actions.
3. Customer chooses: ”Deposit funds”.
4. Bank prompts for amount.
5. Customer enters amount.
6. Funds are deposited to account.

Use Case 3 – Check transactions

Primary actors:

Customer

Bank

Basic flow of events:

1. Customer accesses bank.
2. Bank prompts available actions.
3. Customer chooses: ”Check transactions”.
4. An overview of transactions from the account is displayed.

Alternative flow of events:

6a. There are no transactions to be shown.

6a1. Bank displays error message.

Use case 4 – Transfer funds

Primary actors:

Bank

Preconditions:

The bank has customers with available accounts.

Basic flow of events:

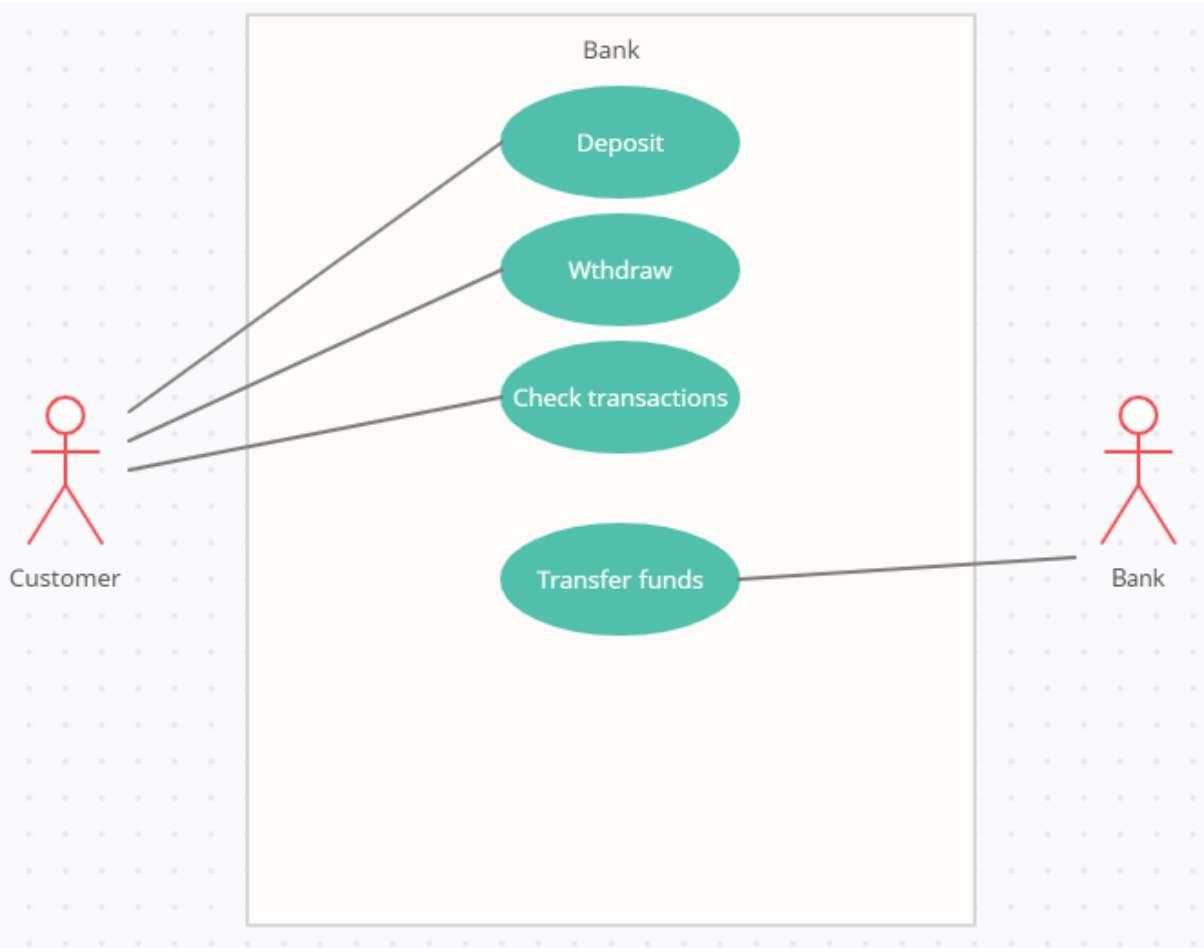
1. Bank prompts available actions.
2. Bank chooses: “Transfer funds”.
3. Bank prompts a list of customers.
4. Bank chooses which customer account the funds will be transferred from.
5. Bank chooses which customer account the fund will be transferred to.
6. Bank prompts for amount to be transferred.
7. Bank checks if the amount is within the limit of the funds available in the account.
8. The funds are transferred.

Alternative flows:

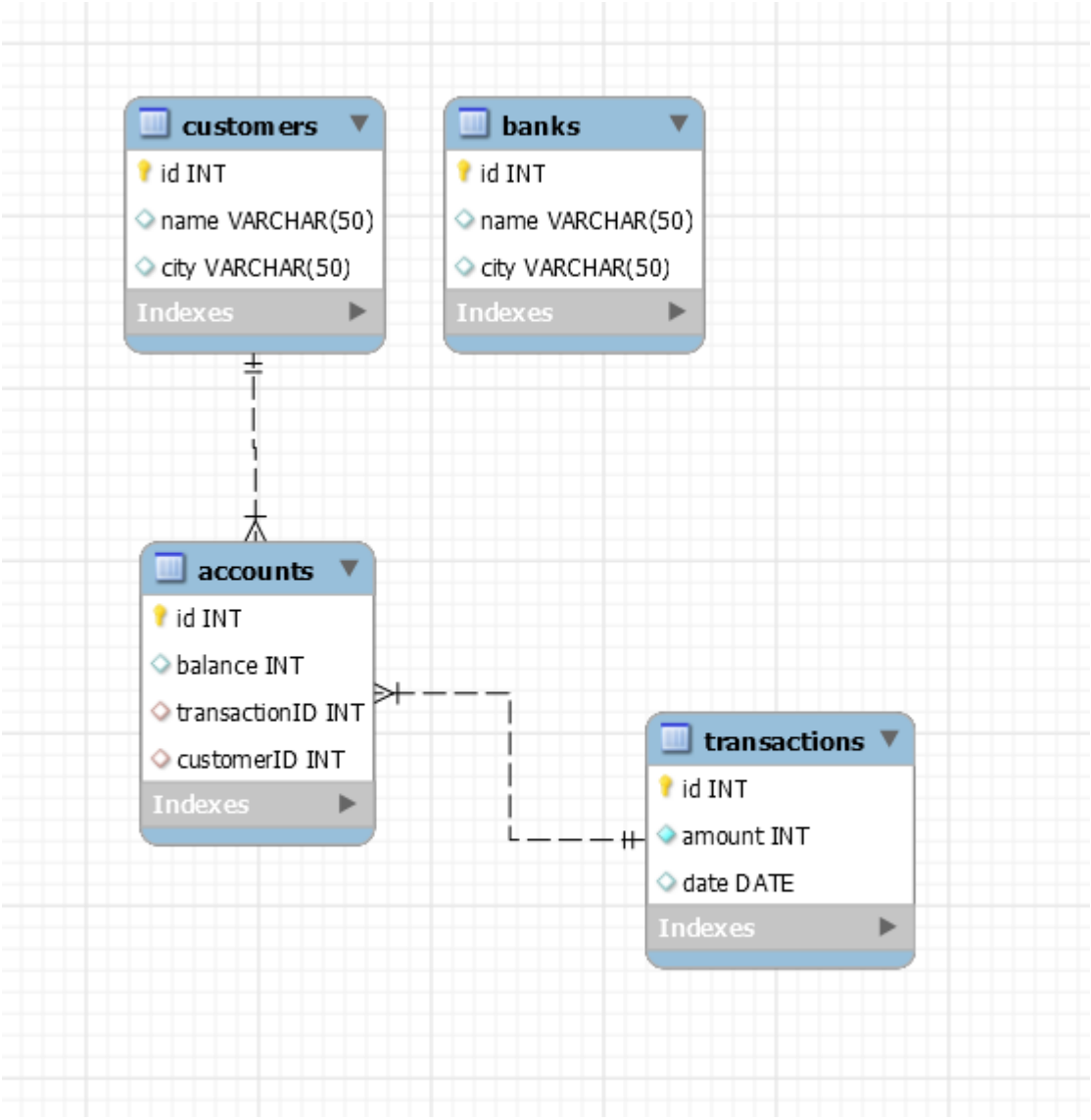
9a. The amount is not within the limit of the funds available in the account.

9a1. Bank displays error message.

Use Case Diagram



EER-Diagram


customers
id INT
name VARCHAR(50)
city VARCHAR(50)
Indexes
banks
id INT
name VARCHAR(50)
city VARCHAR(50)
Indexes
accounts
id INT
balance INT
transactionID INT
customerID INT
Indexes
transactions
id INT
amount INT
date DATE
Indexes

Status & Reflektion

Status:

Vi har ikke opfyldt alle de ønskede funktionalitetskriterier i programmet, men har i stedet fokuseret på design-strukturen i vores kode. Vi har benyttet MVC design-pattern og har kodet op mod interfaces med henblik på at skabe lav kobling.

Selvom vi ikke er nået helt i mål med programmets funktionalitet, så har vi alligevel nået at implementere withdrawFunds() og depositFunds(), da vi ser disse metoder, som de mest essentielle i programmet. Derudover har vi fået lavet en simpel menu, som skifter baseret på user-input.

Refleksion:

Gruppearbejdet er gået godt på trods af vi alle skulle lære at bruge nye værktøjer og fungere i en større gruppe end vi er vant til.

Vores fokus har været på at samarbejde og lære hinanden at kende, samt at bruge Git og Kanbas til at strukturere vores arbejde. Dog har dette også forhindret os i at arbejde lige så effektivt, som vi havde ønsket, fordi vi alle har brugt tid på at blive familiære med disse nye værktøjer. Det blev derfor sværere at fordele arbejdet imellem os, hvilket resulterede i at det overordnet set blev en fælles arbejdsproces, hvor et enkelt gruppemedlem tog styringen og de resterende medlemmer kom med input. På den måde kunne vi dele vores erfaringer og inspirere hinanden.

Vi var alle positivt overraskede over integrationen af Kanbas i GitHub, og kan godt se potentialet i at bruge det igen i fremtidige projekter, hvor vi har fået mere erfaring med det. Det samme gælder for Git, som til tider skabte nogle bump på vejen, men ultimativt gavnede effektiviteten.

Vores største problemer opstod, da vi skulle forbinde vores database til vores projekt, og prøve at implementerer metoderne, så de korrelerer med databasens struktur. Vi kunne alle godt tænke os at få en slags skabelon eller gennemgang af, hvordan man kan strukturere sit projekt og oprette klasser og metoder så det passer til databasens.

Næste gang vil vi bruge mere tid på at sætte vores databasestruktur og projektstruktur bedre op, så implementeringen er mere ligetil.

Vi prøvede at udfordre os selv med dette projekt, hvilket resulterede i mangel på funktionalitet i programmet, men på den anden side fik vi meget mere erfaring med nye værktøjer, mens vi samarbejdede og lærte hinanden bedre at kende.