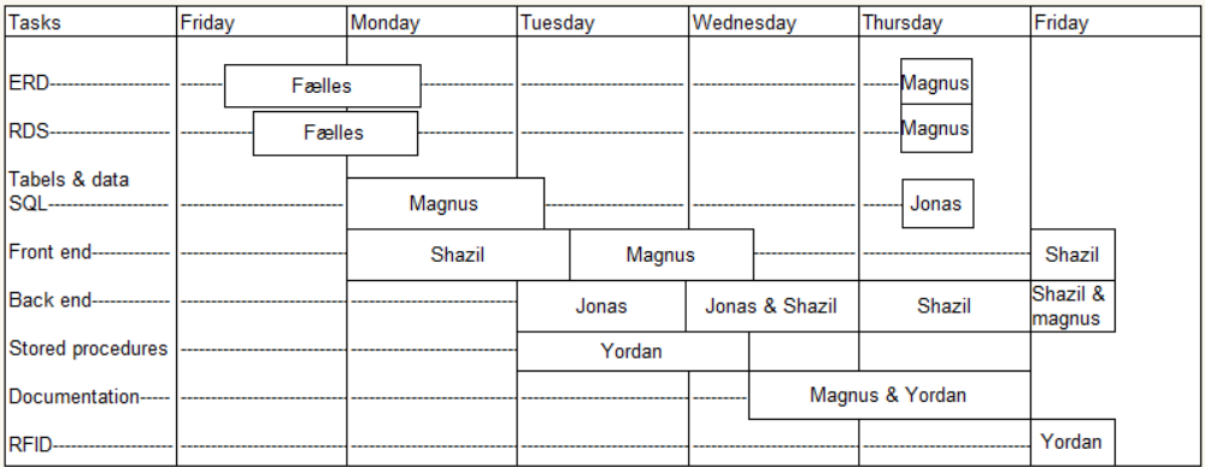
Krølls bank

Table of contents:

Gantt Diagram:



This shows our progress throughout the week.

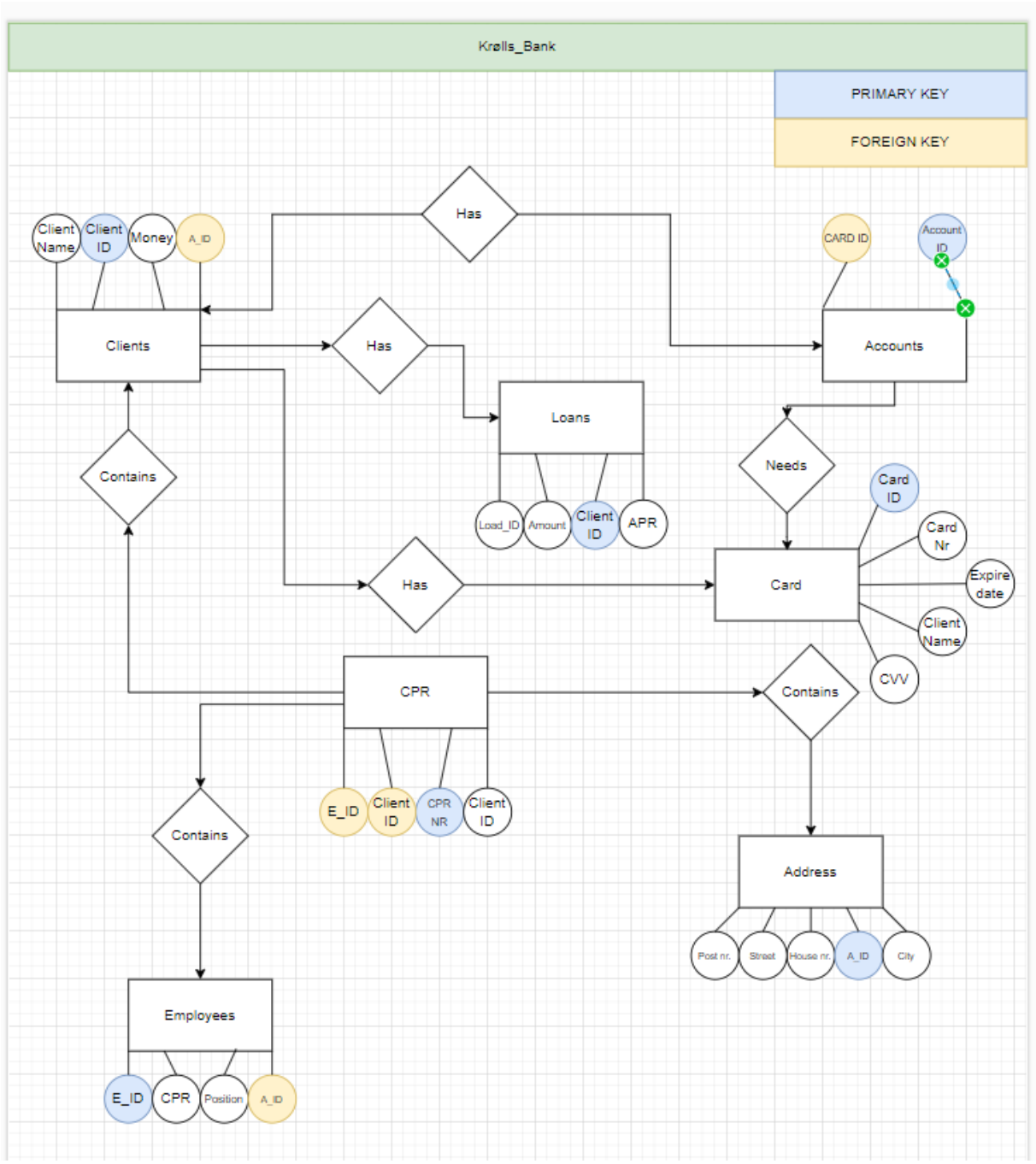
Database and software usage:

We use the software as a user interface for an ATM machine at a bank where you can either withdraw or deposit money into your account balance by "plugging your card into the machine" but because we don't have a physical card, we have skipped that step. We use the database so that we can keep an eye on a lot of bank-customer related information where all customer data is relative and is used to display things on the user interface.

Problem definition:

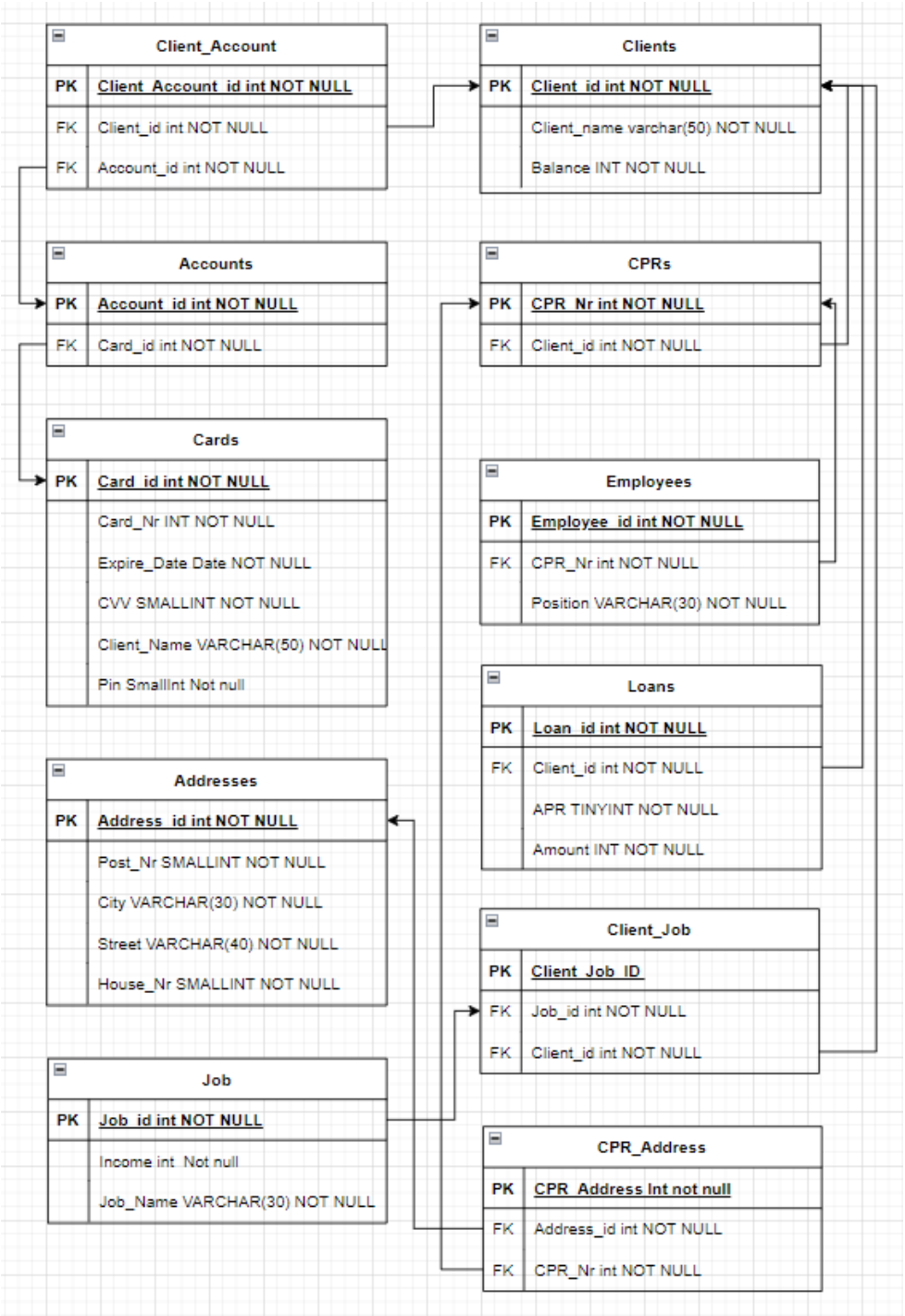
The task at hand is to set up and make technology capable of preserving data of customers that is relative and making a software capable of handling and displaying said data to customers for routine banking. For that to happen a solution for preserving data must be chosen or made and a way for the customer to interact with the data must be developed. Our use of a database solves our first problem of preserving data from customers and making it relational so they are tied to multiple things. Our other solution being the software interface also solves the problem of interacting with the data such as depositing money or withdrawing it as well as displaying customer relative data.

ER-Diagram:



This is our ER-Diagram which describes how our database is structured and which tables and their attributes are related to one another. We use color coding to explain which table attributes are shared and come from a foreign table called Foreign Keys and Primary Key for each individual table which could be referenced as a Foreign Key in a foreign table.

RDS Diagram:



This is our RDS-Diagram which is also used to describe the relation between tables in the database as well as which attributes are used in foreign tables. Here it's explained via arrows which Primary Key is integral to other foreign tables as it's source of relational data. Each Primary Key is an index for its own table.

Normalization of the database:

First normal form:

- No repeated data.
- Each table has a primary key.

Second normal form:

- Attributes are fully dependent on their primary key.

Third normal form:

- Many to many relations have their own middleman table.

Users and Permissions:

Our only user to manage the database is called PinManager and he has Read and Write rights to Krøll's bank database. He is used in our connection string inside the code to connect to the database and write data to it and read data from it.

Stored Procedures:

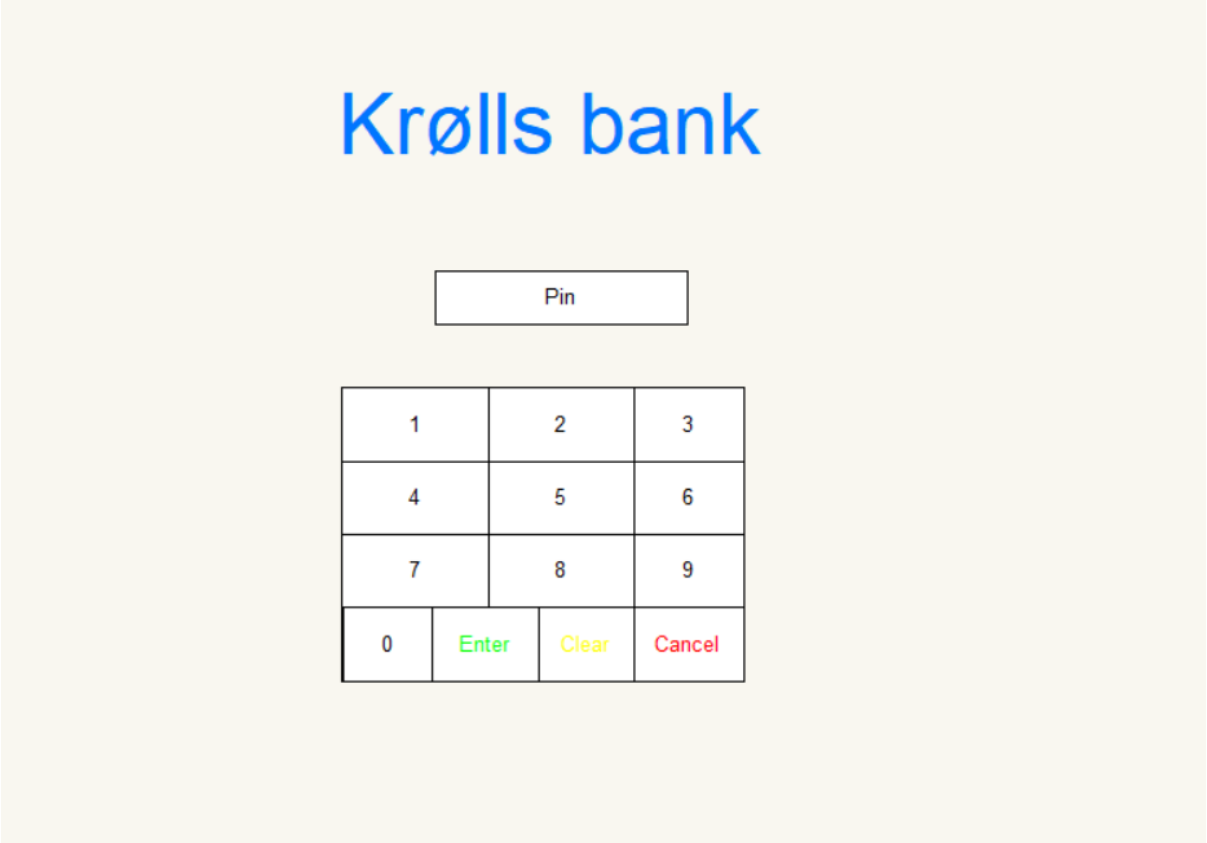
```
CREATE PROCEDURE PinToClientInfo
@Pin SMALLINT
AS
BEGIN
        SELECT Balance, Clients.Client_name FROM Clients
        JOIN Cards
        ON Clients.Client_name = Cards.Client_name
        WHERE Pin = @Pin;
END

CREATE PROCEDURE PinToDepositMoney
@Pin SMALLINT,
@DepositValue DECIMAL
AS
UPDATE Clients
SET Balance = @DepositValue + Balance
WHERE Client_name IN
(
  SELECT Clients.Client_name
  FROM Clients
  INNER JOIN Cards
  ON Clients.Client_name = Cards.Client_name
  WHERE Pin = @Pin
);

CREATE PROCEDURE PinToWithdrawMoney
@Pin SMALLINT,
@WithdrawValue DECIMAL
AS
BEGIN
  -- Check if the user has enough balance
  DECLARE @CurrentBalance DECIMAL;
  SELECT @CurrentBalance = Balance
  FROM Clients
  WHERE Client_name = (SELECT Client_name FROM Cards WHERE Pin = @Pin);

  IF @CurrentBalance IS NOT NULL AND @CurrentBalance >= @WithdrawValue
  BEGIN
    -- Perform the withdrawal
    UPDATE Clients
    SET Balance = Balance - @WithdrawValue
    WHERE Client_name = (SELECT Client_name FROM Cards WHERE Pin = @Pin);
  END
  ELSE
  BEGIN
    -- Insufficient funds, you could set a return code or output variable here
    -- For example:
    RETURN -1;
  END
END;
```

These are our stored procedures.

PinToClient is used for the app to associate the Client with the right Card via the Pin number.
PinToDeposit is where we use the Card table to add to the balance of the right Client by using the Pin number.
PinToWithdraw is almost the same as PinToDeposit where we subtract from the balance instead and check if the client has insufficient funds and return an error code.

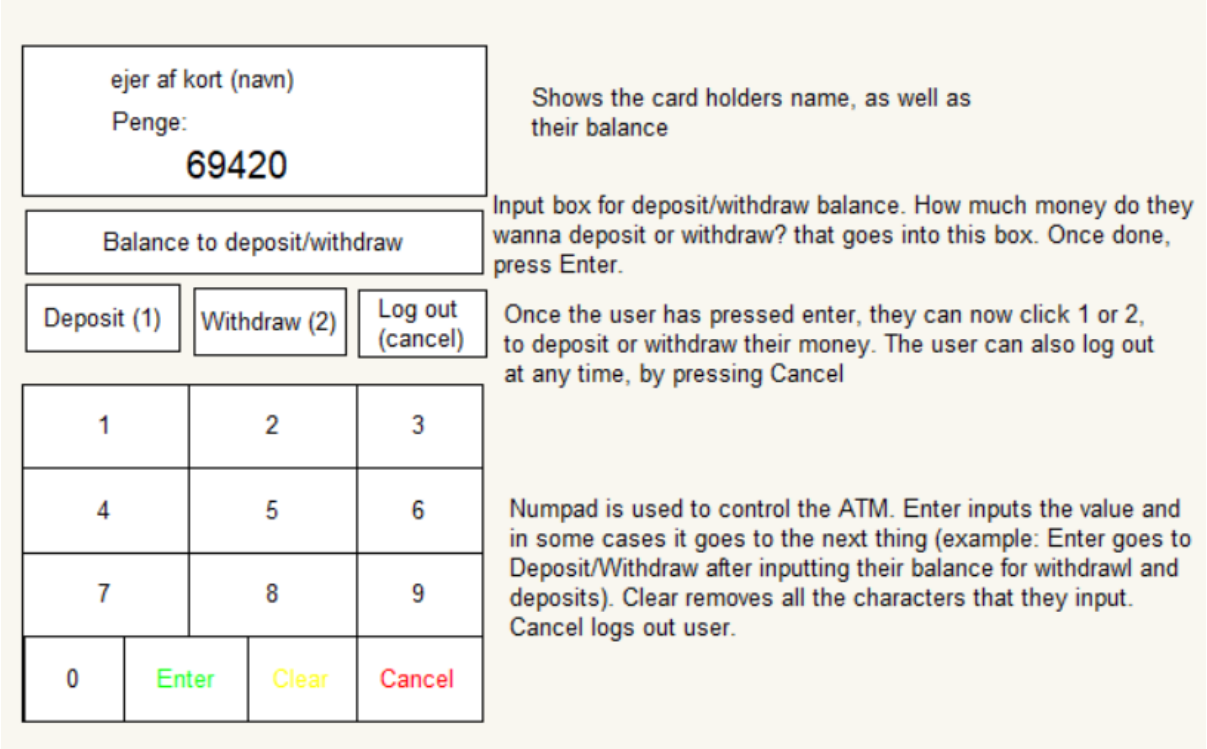
The security of the SQL is a bit questionable in this case because the user is capable of depositing as much money as they want, but this is due to the fact that we aren't capable of checking whether or not the customer has physically inserted cash. So we assume the customer has inserted the appropriate amount in cash before typing out the amount they've inserted to transact.

Mock up of ATM PIN Login:



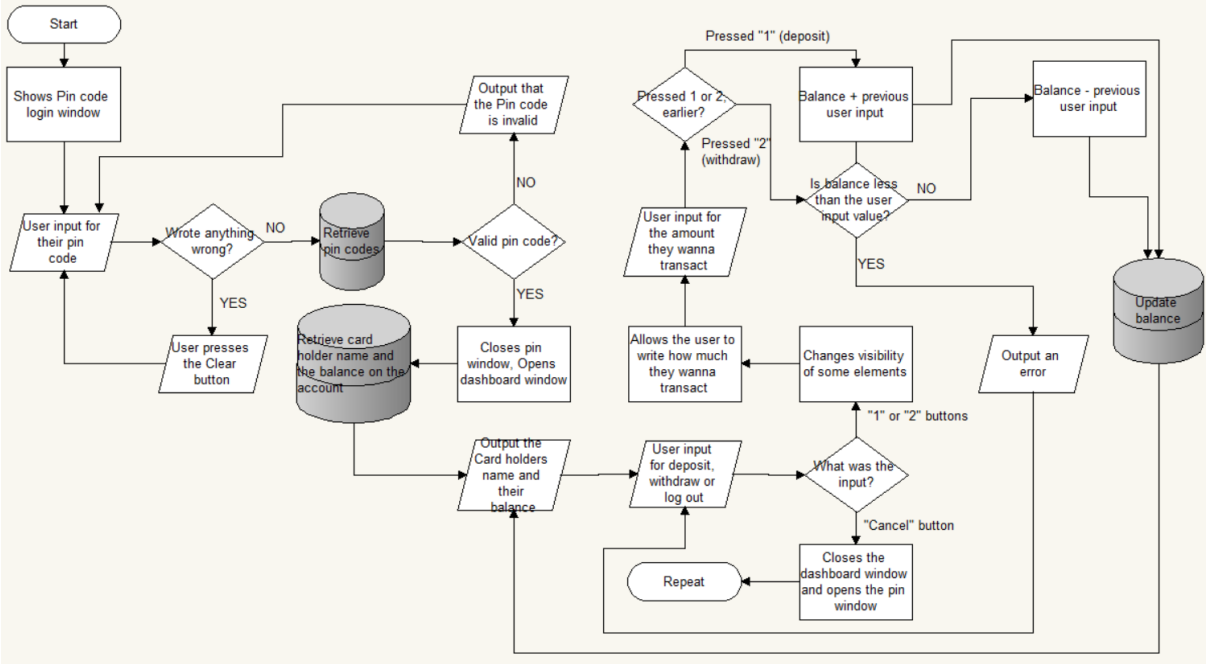
This is our mock up of how we imagined the ATM app's user interface should look when we design it. Our design philosophy is that you use the numpad to both navigate the menu and enter numbers in the PIN field, and use Enter, Clear and Cancel for general functionality.

Mock up of ATM dashboard:



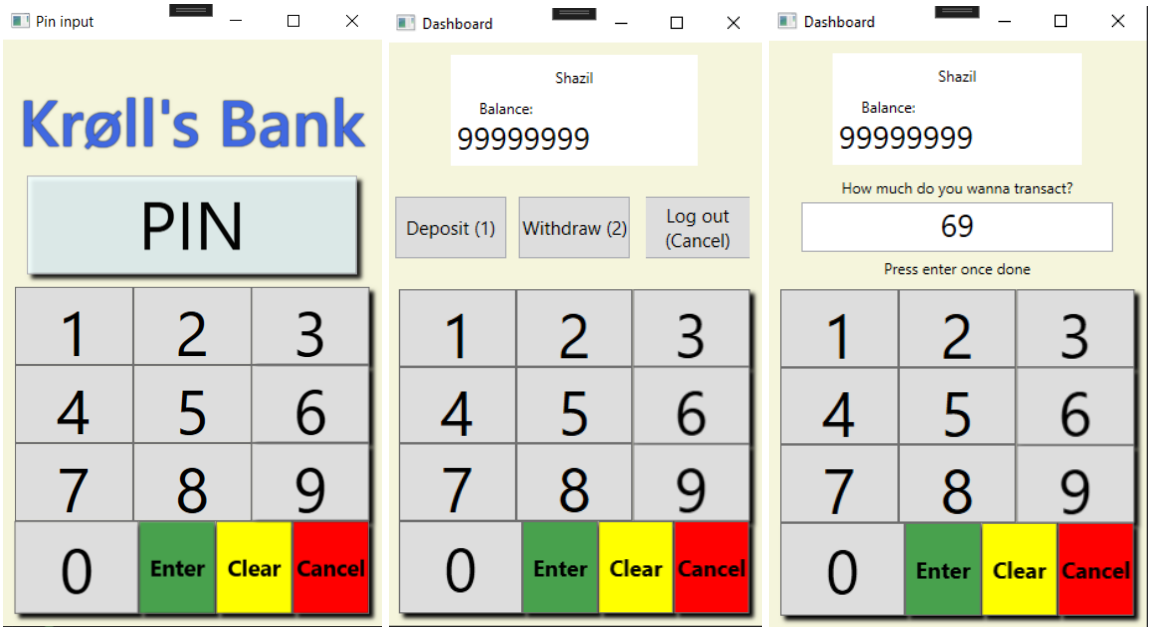
This is our second mock up of our ATM app where you already are logged in via your PIN. The design philosophy with it is that it is displayed after you have entered your PIN and pressed enter and the first window gets closed. Then, via the numpad, you can choose between depositing money into your account or withdrawing money from it.

Flowchart:



This is the flowchart that we have made.
We start on the pin login page. On this page, the user writes his pin code. If the user writes incorrectly, the user can press the Clear button. When the user is satisfied with his input, the user can press enter, after which it checks whether the pin code matches in the database. If the pin code is not in the database, the user gets an error, after which the user must again enter a pin code. If the Pin code is in the database, it retrieves the cardholder's name and money. The user must now write 1, 2 or "cancel". If the user presses cancel, the user logs out and returns to the start. If the user writes 1, the user can deposit money into the account. If the user writes 2, the user must withdraw money.
If the user does not have enough money to withdraw, then the user gets an error and comes back to having to choose 1, 2 or "cancel". Otherwise, the data in the database is updated and then the sum of the money is updated on the dashboard.

C# Program:



Our C# app is made as a WPF project. You use a numpad to control the program.
The first thing you do is enter a pin code to log in to your account.
You will then be greeted by a new window called "Dashboard". On the dashboard you can deposit or withdraw money. The user must then press 1 on the numpad to deposit or press 2 to withdraw. The user can also press the "Cancel" button to log out.
If the user presses either 1 or 2, a box is displayed where the user's input is.
The user must write the amount they want to deposit/withdraw and then the user's money is updated in the database and the new value is retrieved from the database to the label in the WPF application.