Rubrik:

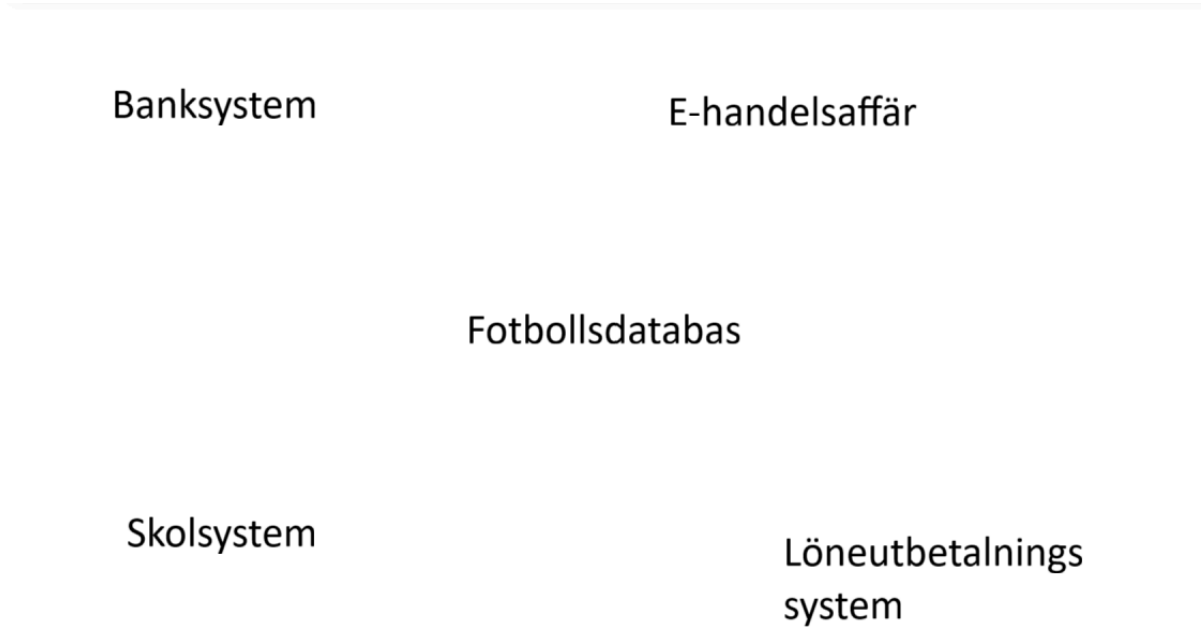
Bank System
Vincent Bejbom

Sammanfattning:

Programmet är ett banksystem, som sparar data i en databas, med en tabell för account_holders, en tabell för accounts, och en tabell för transactions. Account_holders kan alltså äga flera accounts, och varje account kan i sin tur ha flera olika transactions. Användaren startar på inloggningssidan, där den kan välja att antingen logga in som en existerande account_holder, eller skapa en ny account_holder. Sen hamnar den i accountssidan, där man kan se alla account_holderns existerande accounts, samt skapa nya accounts. Om användaren trycker på ett av de existerande accountsen så hamnar den i transaktionssidan, där man kan se alla accountets tidigare transaktioner, samt skapa en ny transaktion. En transaktion kan antingen vara “deposit”, “withdraw”, eller “transfer.”

Framtagande:

Steg 1: Det första som jag gjorde var att försöka komma på vad för sorts program jag skulle skapa. Jag visste att jag ville att programmet skulle använda en databas som jag tog fram i SQL, eftersom att de tidigare databasuppgifterna hade varit de mest intressanta enligt mig. Jag brainstormade ihop lite olika ideér för att komma fram till exakt vad jag ville göra:



Av dessa alternativ som jag lyckades komma på så tyckte jag att ett banksystem verkade mest intressant att skapa, så det valde jag att göra.

Steg 2: Jag bestämde mig för att först skapa alla html-sidor. Den första sidan som användaren hamnar i är loginsidan, så den gjorde jag först. Jag gjorde dock endast login formuläret i det här stadiet, eftersom att jag först hade tänkt att både inloggning och skapande av account_holder skulle ske genom login formuläret, så jag gjorde inte create_account formuläret förens ett senare skede.

Steg 3: Nästa steg var att skapa accountssidan. Det var väldigt simpelt, och sidan bestod av ett formulär för att skapa ett nytt account, och sen en mall för existerande accounts, där —id—, —name—, och —balance— byts ut mot de riktiga värdena. Det var även tänkt att användarna skulle kunna trycka på ett account för att ta sig vidare till transaktionssidan för just det accountet, så därför satte jag all kod som ingick i mallen inuti en href länk, med ett id i URL som byts ut mot det riktiga account id:t.

Steg 4: Här gjorde jag transactionssidan. Den bestod också av ett formulär som gör en ny transaktion och en mall för existerande transactions. I formuläret så skriver användaren in transaction_type, amount, receiver, och description. I transaction_type så kan användaren välja om den vill göra en deposit, withdraw, eller transfer. Därför valde jag att göra transaction_type till en select med options för de olika alternativen. I mallen så ingår id, time, type, from, to, amount, och comment. Dessa byts sedan ut mot de riktiga värdena i transactions. Ovanför mallen så finns det även ett balance värde, vilket är balance för det nuvarande accountet på transactionssidan. —balance— byts senare ut mot det riktiga värdet för account balance.

Steg 5: Efter att jag hade gjort alla html sidor så gjorde jag själva databasen. Jag använde xampp och skapade bank databasen med SQL. Den första tabellen som jag skapade var account_holders, som hade kolumnerna id, name, och password. Id användes som primary key. Nästa tabell var accounts, som hade kolumnerna id, holder_name, holder_id, och balance. Id användes återigen som primary key, och holder_id användes som foreign key för att koppla ihop holder_id med id i account_holders tabellen. Den sista tabellen som jag skapade var transactions, som bestod av kolumnerna id, type, from_account_id, to_account_id, amount, time_stamp, och description. Id användes som primary key och både from_account_id och to_account_id användes som foreign key till id i accounts tabellen. Tanken här var att både transaktioner som har gjorts från ett konto och till ett konto ska visas bland ett accounts transactions. Om typen av transaction är deposit så kommer from_account_id att vara null, och to_account_id att vara account id. Om typen av transaction är withdraw så kommer from_account_id att vara account id, och to_account_id att vara null. Om typen av transaction är transfer så kommer from_account_id att vara account id, och to_account_id kommer att vara id:t på det account som pengarna skickas till.

Steg 6: Efter att jag skapade databasen så var det dags att börja på php filerna. Jag började med show-accounts.php, som skulle hantera både login formuläret och

formuläret för att lägga till ett nytt bankkonto. När jag skulle börja jobba med det så insåg jag att det var smartare att lägga till ännu ett formulär i inloggningssidan för att skapa ett nytt konto, alltså en account_holder, istället för att hantera det tillsammans med login formuläret. Därför lade jag till det i login.html. Efter det så etablerade jag en connection med databasen, samt hämtade accounts.html som skall visas, och delade upp den genom att använda explode på <!--===entries===-->. Sen hanterade jag vad som hände när användaren skickade in login formuläret, samt bytte ut värdena i mallen i accounts.html mot de riktiga värdena genom att göra en SQL query för att få fram alla accounts som tillhör den nuvarande account_holder, och sedan loopa igenom varje account för att byta ut värdena. Efter det så hanterade jag när create_account formuläret blev inskickat, och då gjorde jag först en SQL query där jag tog fram alla account_holders med samma password som användaren skickade in i formuläret. Om det fanns någon account_holder med samma password så skickas användaren tillbaka till inloggningssidan, eftersom att varje account_holder måste ha ett unikt password. Om det å andra sidan inte kom upp någon account_holder med samma lösenord så gör jag en ny SQL query där jag lägger in en ny account_holder med name och password som användaren har skickat in via create_account formuläret. På samtliga SQL querys så använder jag prepare och bind_param, för att förhindra SQL injection i och med att användaren annars skulle kunna skicka med SQL kod i exempelvis name fältet. Det sista jag behövde göra i show-accounts.php var att hantera formuläret för att öppna ett nytt bank account. Där gjorde jag en SQL query för att lägga in en ny account, med värdena som användaren skrev in i formuläret. Efter det så laddas sidan om.

Steg 7: Ett problem som jag stötte på var att holder_name och password inte fick ett värde efter att sidan laddades om när användare öppnade ett nytt bank account. Anledningen till det var för att holder_name och password sattes i hanteringen av login eller create_account formuläret, och den koden kördes då inte. För att hantera detta så använde jag mig av session för att spara värdena på variablerna genom omladdning av sidan.

Steg 8: Nästa steg var att göra show-transactions.php. Jag började med att hantera formuläret för att göra en ny transaction. I och med att det fanns tre olika alternativ på typen av transaction så var jag tvungen att göra if satser för varje alternativ. Om det var en deposit, så gjorde jag en SQL query där jag lade till deposit amount i account balance, samt satte variabeln för to_account_id till account id. Om det inte var en deposit så gjorde jag en SQL query för att ta fram account balance och kontrollera om amount som användaren hade skrivit i formuläret var mer än account balance. Isåfall laddades sidan om eftersom att accountet hade för lite pengar för att antingen göra en withdraw eller transfer. Om accountet hade tillräckligt mycket pengar så fortsatte programmet till en if sats om typen var en withdraw. Då gjorde jag en SQL query för att minska account balance med amount som skickades med i formuläret, samt satte from_account_id till account id. Om typen var en transfer så gjordes några kontroller för att se att receiver fanns med och inte var felaktig, och

sen så ökade balance med amount på receiver account och minskade balance med amount på account som gjorde transfer. Efter alla if satser för att hantera de olika typerna så gjorde jag själva SQL queryn där jag lade till den nya transaction. Efteråt så hanterade jag mallen där alla nuvarande transactions skulle sättas in, genom att ta fram alla transactions där antingen from_account_id eller to_account_id matchade account id, och loopade sedan igenom alla transactions och lade in värdena i mallen.

Form:

Pseudokod:

Show-accounts.php

```
html = accounts.html
html_pieces = explode(<!--===entries===-->)

db_variables…

account_holder_variables…

session_start()

conn = mysqli_connect(db_variables…)
if (conn->error)
        die(print error)

if (server request == post)
        if (isset(login)
                holder_name = post[name]
                password = post[password]
                sessionname = holder_name
                sessionpassword = password
        else if (isset(create_account)
                holder_name = post[name]
                password = post[password]
                sessionname = holder_name
                sessionpassword = password

                login_statement = SQL query SELECT accounts WHERE password =?
                login_statement-> bind_param(password)
                login_statement…
                if (accounts)
                        return to login.html
```

```
                create_statement = SQL query INSERT account_holders (name,
password)
                create_statement->bind_param(holder_name, password=
                create_statement…

if (isset(session holder_name && password))
        holder_name = session holder_name
        password = session password
else ()
        return to login.html

print html_pieces[0]

holder_statement = SQL query SELECT account_holders WHERE name =
holder_name AND password = password
holder_statement->...

try
        if (!account_holder)
                return to login.html
        holder_id = account_holder.id

        account_statement = SQL query SELECT accounts WHERE holder_id =
holder_id
        account_statement->...

        if (accounts)
                loop
                        accounts_variables…
                        tmp_string = html_pieces[1]
                        tmp_string = str_replace(accounts_variables…, tmp_string)
                        print tmp_string
        else ()
                print html_pieces[1]
catch (error)
        print error

print html_pieces[2]

if (server request == post)
        if (isset(open_bank_account)
                try
                        balance = 0
                        if (isset(balance))
```

```
                    balance = post[balance]
                open_bank_statement = SQL query INSERT accounts
(holder_name, holder_id, balance)
                open_bank_statement->...
                reload page
            catch (error)
                print error
```

Show-transactions.php

```
html = transactions.html

db_variables…

session_start()

if (isset(account_id))
    account_id = get[account_id]
    session account_id = account_id

account_variables…

conn = mysqli_connect(db_variables…)
if (conn->error)
    die(print error)

if (server request == post)
    acount_id = session account_id
    transaction_variables…

    if (transaction_type = deposit)
        transaction_to_account = account_id
        statement = SQL query UPDATE accounts SET balance +=
transaction_balance WHERE id = account_id
        statement->...
    else ()
        transaction_from_account = account_id
        old_balance = 0
        balance_statement = SQL query SELECT balance FROM accounts
WHERE id = account_id
        old balance = balance_statement-> result
        balance_statement…
        if (transaction_amount > old_balance)
            reload page
```

```
                if (transaction_type == withdraw)
                        statement = SQL query UPDATE accounts SET balance -=
transaction_balance WHERE id = account_id
                        statement->...
                else if (transaction_type = transfer)
                        receiver_balance = 0
                        if (empty(receiver) || post[receiver] == account_id)
                                reload page
                        receiver_id = post[receiver]
                        receiver_statement = SQL query SELECT accounts WHERE id
= receiver_id
                        receiver_statement->...
                        if (!receiver)
                                reload page

                        update_receiver_statement = SQL query UPDATE accounts
SET balance += transaction_balance WHERE id = receiver_id
                        update_receiver_statement->...

                        update_account_statement = SQL query UPDATE accounts
SET balance -= transaction_balance WHERE id = account_id
                        update_account_statement->...

        make_transaction_statement = SQL query INSERT transactions
(transactions_variables…
        make_transaction_statement->...
        reload page

account_id = session account_id
account_balance = 0

account_balance_statement = SQL query SELECT balance FROM accounts
WHERE id = account_id
account_balance = account_balance_statement->result
account_balance_statement->...

html = str_replace(---balance—, account_balance, html)

html_pieces = explode(<!--===entries===-->)

print html_pieces[0]

transactions_statement = SQL query SELECT transactions WHERE
from_account_id = account_id OR to_account_id = account_id
```

transactions_statement->...

```
if (transactions)
        loop
            transactions_variables…
            tmp_string = html_pieces[1]
            tmp_string = str_replace(transactions_variables…, tmp_string)
            print tmp_string
    else ()
        print html_pieces[1]

print html_pieces[2]
```

Centrala kod-avsnitt:

```php
$conn = mysqli_connect($host, $user, $pass, $db);
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}
```

```php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
   // Sets the name and password when the user has logged in
to their account
   if (isset($_POST["login"])) {
       $holder_name = strip_tags($_POST["name"]);
       $password = strip_tags($_POST["password"]);
       $_SESSION['holder_name'] = $holder_name;
       $_SESSION['password'] = $password;
```

```php
} else if (isset($_POST["create_account"])) {
   $holder_name = strip_tags($_POST["name"]);
   $password = strip_tags($_POST["password"]);
   $_SESSION['holder_name'] = $holder_name;
   $_SESSION['password'] = $password;
   // Gets existing account holders that already have the
password the user put in
   $login_statement = $conn->prepare("SELECT * FROM
account_holders WHERE password = ?");
```

```php
$create_statement = $conn->prepare("INSERT INTO
account_holders (name, password) VALUES (?, ?)");
```

```
if ($account_rows && $account_rows->num_rows > 0) {
   while ($account_row = $account_rows->fetch_assoc()) {
       $id = $account_row["id"];
       $name = $account_row["holder_name"];
       $account_balance = $account_row["balance"];
       $tmp_string = $html_pieces[1];
       $tmp_string = str_replace(["---id---", "---name---",
"---balance---"],
           [$id, $name, $account_balance], $tmp_string);
       echo $tmp_string;
   }
} else {
   echo $html_pieces[1];
}
```

```
if ($_SERVER["REQUEST_METHOD"] == "POST") {
   // Creates new account for account holder
   if (isset($_POST["open_bank_account"])) {
       try {
           $balance = 0;
           // Checks if user made initial deposit
           if (isset($_POST["deposit"])) {
               $balance = strip_tags($_POST["deposit"]);
           }

           // Inserts the new account to the database and
reloads the page
           $open_bank_statement = $conn->prepare("INSERT INTO
accounts (holder_name, holder_id, balance) VALUES (?, ?, ?)");
```

```
if ($transaction_type == "deposit") {
   $transaction_to_account = $account_id;
   // Adds the deposit amount to account
   $statement = $conn->prepare("UPDATE accounts SET balance =
balance + ? WHERE id = ?");
```

```
if ($transaction_type == "withdraw") {
   // Subtracts the withdrawal amount from the account balance
   $statement = $conn->prepare("UPDATE accounts SET balance =
balance - ? WHERE id = ?");
```

```
} else if ($transaction_type == "transfer") {
    $receiver_balance = 0;
    // Cancels the transfer and reloads the page if there is no
receiver or
    // if the receiver is the account that is making the
transfer
    if (empty($_POST["receiver"]) || $_POST["receiver"] ==
$account_id) {
        header("Location:
show-transactions.php?account_id=$account_id");
        exit;
    }
    $receiver_id = $_POST["receiver"];
    $transaction_to_account = $receiver_id;
    // Gets the receiver account
    $receiver_statement = $conn->prepare("SELECT * FROM
accounts WHERE id = ?");
```

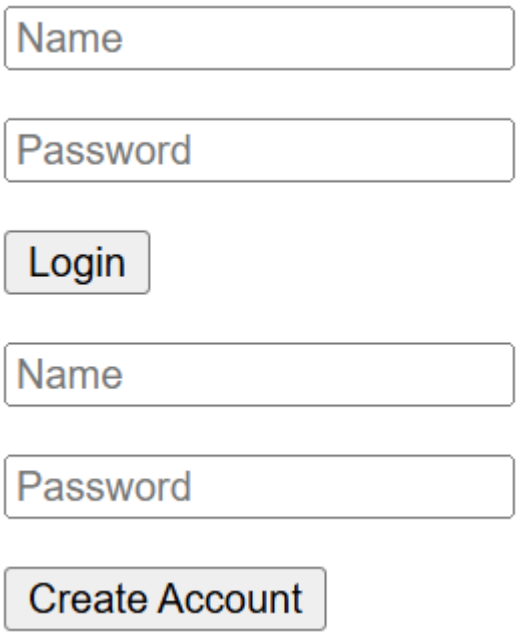
```
$update_receiver_statement = $conn->prepare("UPDATE accounts
SET balance = balance + ? WHERE id = ?");
$update_receiver_statement->bind_param("ii",
$transaction_amount, $receiver_id);
$update_receiver_statement->execute();
$update_receiver_statement->close();

// Subtracts the money sent from the account making the
transfer
$update_account_statement = $conn->prepare("UPDATE accounts
SET balance = balance - ? WHERE id = ?");
$update_account_statement->bind_param("ii",
$transaction_amount, $account_id);
$update_account_statement->execute();
$update_account_statement->close();
```

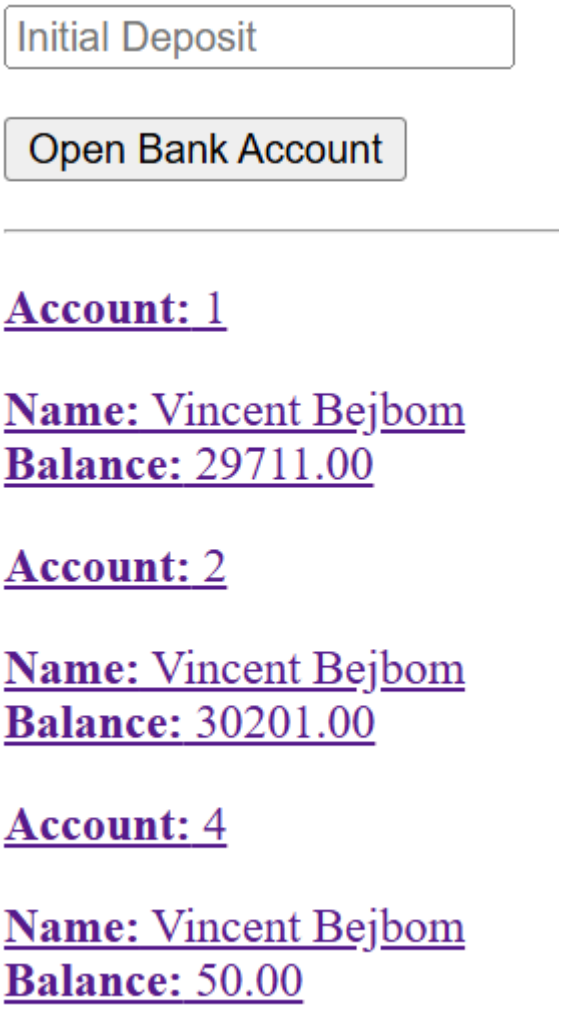
```
$make_transaction_statement = $conn->prepare("
    INSERT INTO transactions (type, from_account_id,
to_account_id, amount, time_stamp, description)
    VALUES (?, ?, ?, ?, ?, ?)
");
```

```
if ($transactions_rows && $transactions_rows->num_rows > 0) {
   while ($transactions_row =
$transactions_rows->fetch_assoc()) {
       $id = $transactions_row["id"];
       $type = $transactions_row["type"];
       if (!is_null($transactions_row["from_account_id"])) {
           $from_account =
$transactions_row["from_account_id"];
       } else {
           $from_account = "";
       }
       if (!is_null($transactions_row["to_account_id"])) {
           $to_account = $transactions_row["to_account_id"];
       } else {
           $to_account = "";
       }
       if (!is_null($transactions_row["description"])) {
           $description = $transactions_row["description"];
       }
       $amount = $transactions_row["amount"];
       $time = $transactions_row["time_stamp"];
       $tmp_string = $html_pieces[1];
       $tmp_string = str_replace(["---id---", "---time---",
"---type---", "---from---", "---to---", "---amount---",
"---comment---"],
           [$id, $time, $type, $from_account, $to_account,
$amount, $description], $tmp_string);
       echo $tmp_string;
   }
} else {
   echo $html_pieces[1];
```

Funktion:



Först hamnar användaren i inloggningssidan. Den kan då välja att fylla i och skicka in Login formuläret eller Create Account formuläret. Om användaren skriver en ogiltig login eller skriver ett password som redan finns i create account så kommer sidan att laddas om. Om ett giltiga namn och lösenord för login eller create account skrivs in så kommer användaren att flyttas vidare till show-accounts sidan.



På show-accounts sidan så finns det ett formulär högst upp för att öppna ett nytt bankkonto. När användaren skickar in formuläret så kommer ett nytt account skapas och visas längst ner bland listan av existerande accounts, med balance värdet som skickas in på initial deposit fältet. Om inget initial deposit skrivs in så kommer balance att vara 0. De olika accountsen som visas under formuläret är alltså accounts som tillhör account_holder som användaren loggade in som. Id, namn, och balance visas för varje account, och varje account har även en länk till show-transactionssidan. Om man trycker på exempelvis account 2 så kommer användaren att hamna i show-transactionssidan för account med id 2.

Transaction Type: ---Select one---

Amount

Receiver

Description

Make Transaction

Balance: 30201.00

Transaction: 15

Time: 2025-10-27 20:23:45
Type: transfer
From: 1
To: 2
Amount: 50.00

Comment: Tesst1231

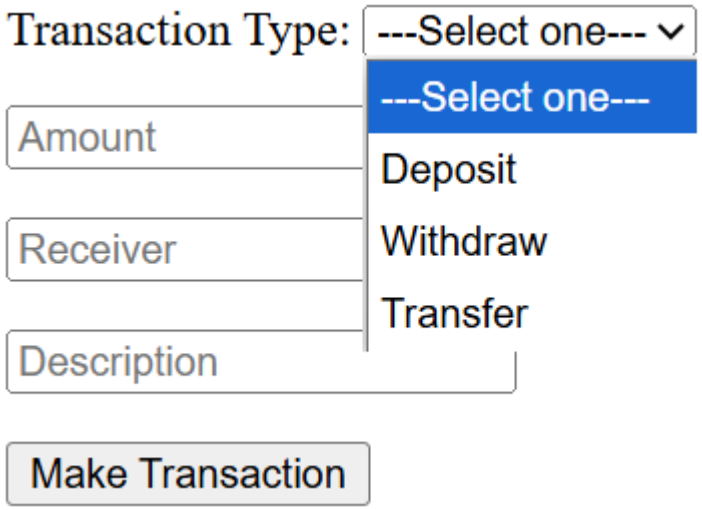
Transaction: 22

Time: 2025-10-28 17:22:35
Type: deposit
From:
To: 2
Amount: 40000.00

Comment:

På show-transactionssidan så visas alla tidigare transactions för accountet. Det som visas är id, time, type, from_account_id, to_account_id, amount, och comment.

Ovanför alla transactions och under formuläret för att göra en ny transaction så visas även account balance.



I formuläret för att göra en ny transaction så väljer användaren först transaction type, vilket är en dropdown meny med deposit, withdraw, och transfer. Om användaren väljer deposit så behöver den endast skriva in amount, och så kommer en ny transaction göras och visas längst ner i listan av transactions efter att användaren har tryckt på Make Transaction knappen. From kommer då att vara tomt och to kommer att vara account id:t. Om användaren väljer withdraw så måste amount vara mindre än balance för att Make Transaction ska gå igenom. From kommer då att vara account id:t och to kommer att vara tomt. Om användaren väljer transfer så måste dels amount vara mindre än balance, men receiver måste även var ett existerande account id som inte är samma som account id:t som användaren gör transaction från. From kommer då att vara account id:t och to kommer att vara receiver id:t. Description är alltid frivilligt att fylla i och kommer isåfall att visas som comment. Efter varje transaction så uppdateras balance till det nya account balance.