COMP 3700 Project 3 Design

Tripp Isbell
cai0004@auburn.edu
github

Design goals:
Since I mostly had multi-user functionality up and running in my previous project, I didn't have much to add for this project except implementing a few of the user types' features. I also cleaned it up a little and added a dark mode, since everyone adds a dark mode these days.

Specific features added in this iteration include:

- User management for admin
- Purchase view for management
- The ability for each user (of any type) to update their username and password
- UI alignment fixes and cleanups
- Dark mode

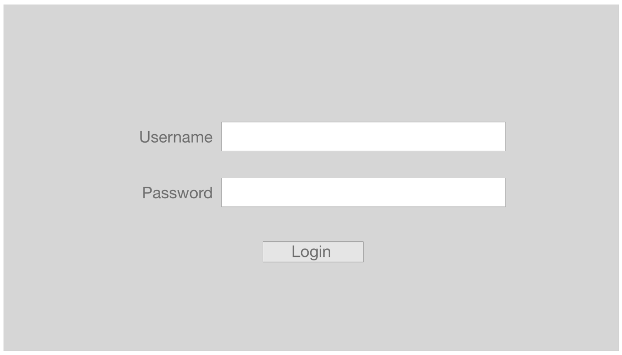
Since the server interaction is implemented through the DataAdapter interface providing the same behavior of a local database, all of these changes were pretty simple and just involved a few UI additions. I did have to change some logic in my table display model to be able to display the user database and update it with changes to the table UI correctly. If I were to have another go at it, I would have used a more generic scheme to implement table displays and edit displays for each of the items stored in the database (users, customers, products, purchases). Since a lot of my views are pretty similar it would have benefitted to subclass some template frames and gone from there.

Below are some use cases from the previous projects. Nothing major has changed about the designs, but each user screen now has a change username/password button.

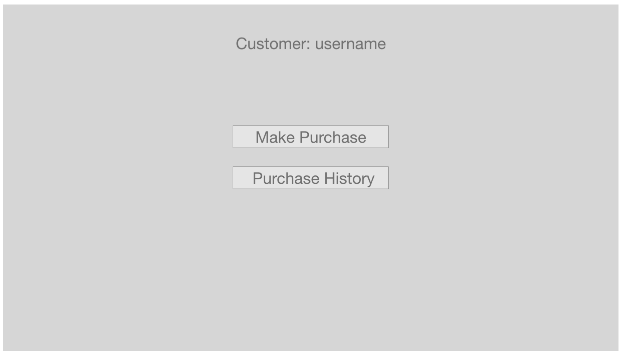
Use Case: Generic login use case
Actors: admin, manager, cashier, customer
Goals: identify the type of user logging in, and display the respective interface
steps:

(1) upon launch, the client application displays a login screen
- the user enters a username/password combo that matches a user in the database
- the system sends that info to server, the server responds with correct user type

(2) the system displays that type's respective interface (customer UI pictured)



(1)



(2)

1 User Story: Add Product

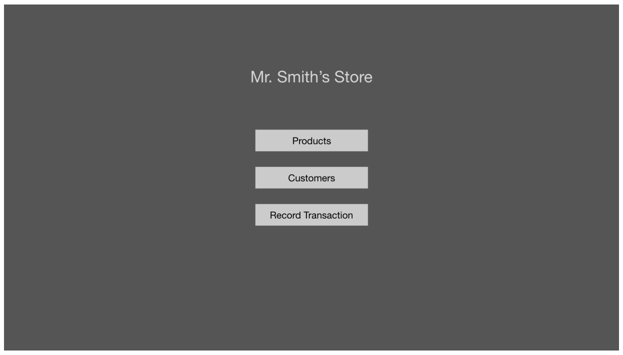
Use Case: add a product into the system
Actors: employees
Goals: update database to include new product
Related use cases: adding a customer to the database or recording a transaction (below) **Preconditions:** interface is functional and ~~connected to underlying database~~ *server is running*
Postconditions: The product database is updated with the item
Steps:

(1) the user clicks a button to display the product database
(2) the system *fetches from server and* displays the database
- the user clicks add product

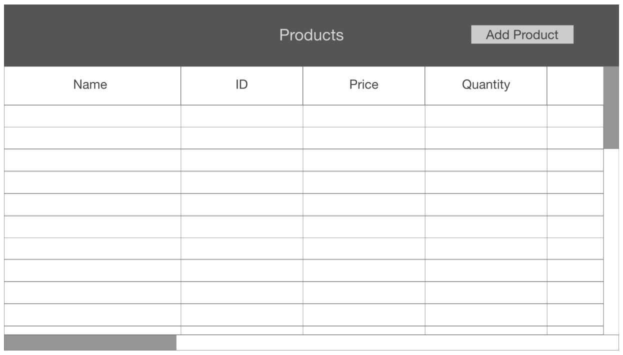
(3) the system displays a screen with text fields for the product info
- the user enters the information and clicks an add button

(4) the system ~~updates the database~~ *sends the product to server* and displays a confirmation message
- the user clicks confirm

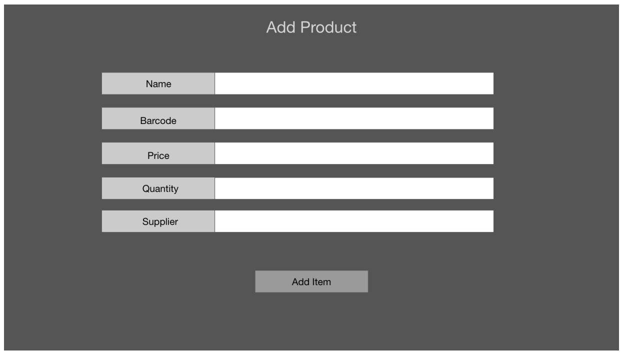
(1) the system returns to the main menu



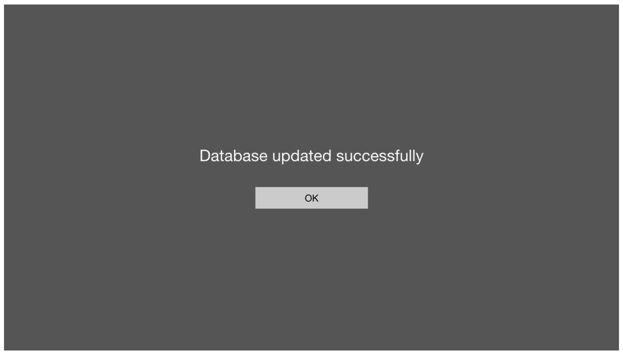
(1)



(2)



(3)


Database updated successfully

(4)

Use Case: update a product in the system
Actors: employees
Goals: update a product in the database
Preconditions: the target product exists in the database
Postconditions: the database is updated with desired changes
steps:

(1) the user clicks a button to display the product database
(2) the system *fetches data from server and* displays the database
- the user (double) clicks on the desired field to edit
- the system responds by making the field editable
- the user enters desired changes and presses enter
- the system ~~updates the underlying database~~ sends the new product to server

the (1) and (2) views here are the same (1) and (2) views on the previous page

2 User Story: Add Customer

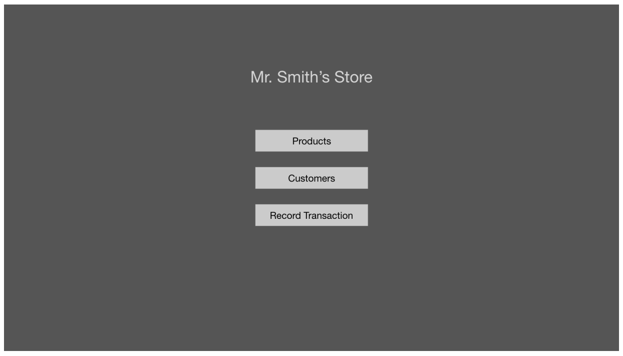
Use case: add a customer into the system
Actors: employees
Goals: update database to include new customer
Related use cases: adding a product or transaction
Preconditions, postconditions: Same as above just replace "product" with "customer"
Steps:

(1) the user clicks a button to display the customer database
(2) the system displays the database
- the user clicks a plus button

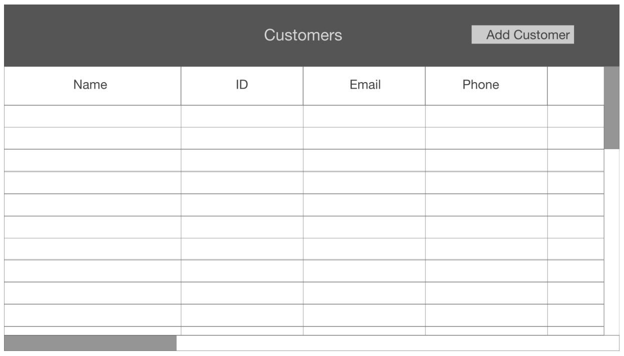
(3) the system displays a screen with text fields for the customer info
- the user enters the information and clicks an add button

(4) the system ~~updates the database~~ *sends data to server* and displays a confirmation message
- the user clicks confirm

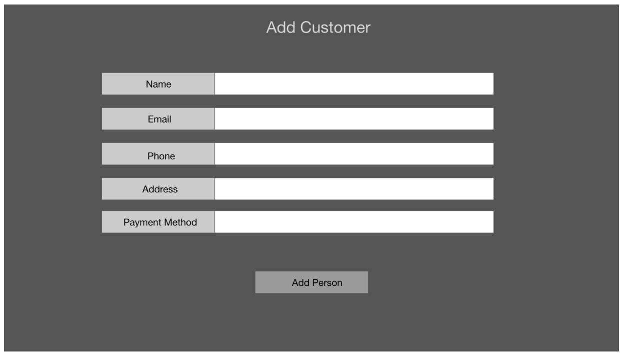
(1) the system returns to the main menu



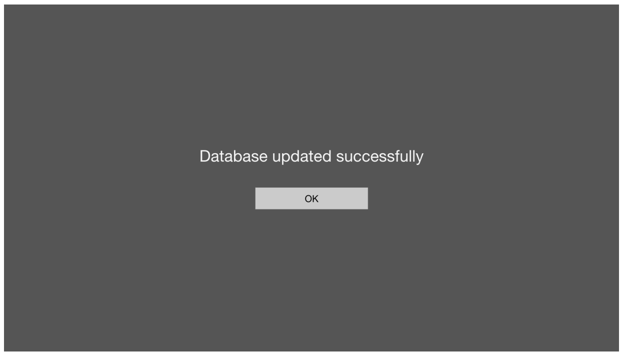
(1)



(2)



(3)



(4)

Use Case: update a customer in the system
Actors: employees
Goals: update a customer entity in the database
Preconditions: the specific customer exists in the database *and server is running*
Postconditions: the database is updated with desired changes
steps:

(1) the user clicks a button to display the customer database
(2) the system displays the database
- the user (double) clicks on the desired field to edit
- the system responds by making the field editable
- the user enters desired changes and presses enter
- the system ~~updates the underlying database~~ *sends info to server*

the (1) and (2) views here are the same (1) and (2) views on the previous page

3 User Story: Add Transaction

Use Case: record a transaction
Actors: employees
Goals: update database to include transaction
steps:

(1) the user clicks a button to record a transaction
(2) the system displays a screen with text fields for transaction
- the user enters in the information and clicks ok

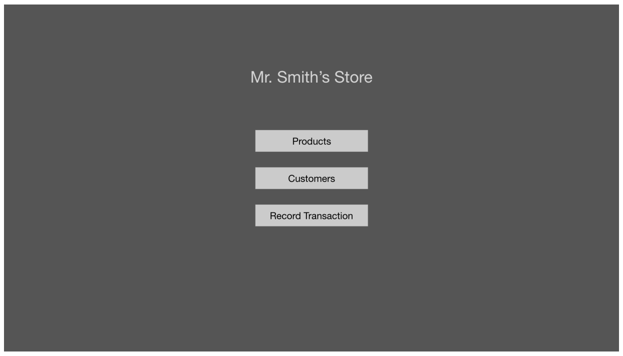
(3) the system displays the receipt with names and prices for confirmation
- the user reviews the information and clicks ok

(4) the system saves the purchase ~~to the database~~ and displays a success message

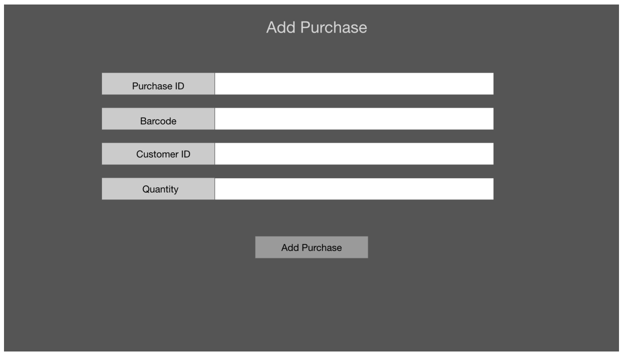
Use Case: refund (delete) a transaction
Actors: employees
Goals: delete purchase from database
steps:

(1) the user clicks a button to view transaction database
(2) the system displays the purchase database
- the user right clicks (or something) on a purchase to delete it

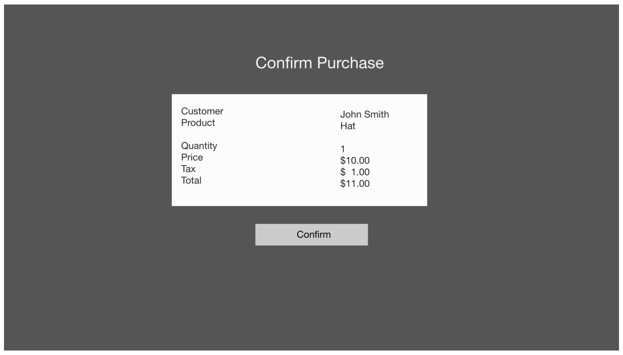
(3) the system displays the purchase receipt with a button to delete
- the user clicks the delete button



(1)

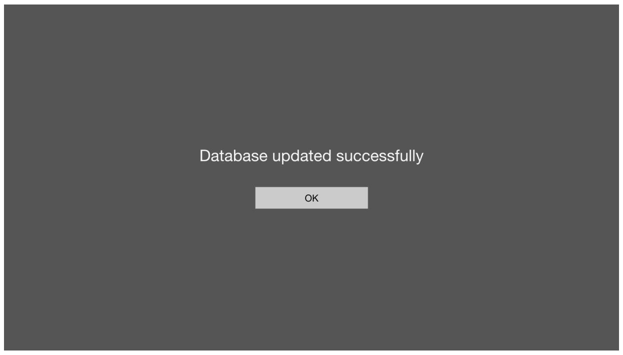


(2)



Customer	John Smith
Product	Hat
Quantity	1
Price	$10.00
Tax	$ 1.00
Total	$11.00

(3)



(4)

(2) the system removes the purchase ~~from the database~~ and updates the view

Same views as above except the (1) view (main menu) includes a "view transaction" button.

SQL code

```sql
CREATE TABLE Customers (
   "CustomerID" INTEGER NOT NULL UNIQUE,
   "Name" TEXT,
   "Email" TEXT,
   "Phone" TEXT,
   "Address" TEXT,
   "PaymentInfo" TEXT,
   PRIMARY KEY("CustomerID")
);

CREATE TABLE Products (
   "Barcode" INTEGER NOT NULL UNIQUE,
   "Name" TEXT,
   "Price" REAL,
   "Supplier" TEXT,
   "Quantity" INTEGER,
   PRIMARY KEY("Barcode")
);

CREATE TABLE Purchases (
   PurchaseID INTEGER NOT NULL UNIQUE,
   CustomerID INTEGER,
   ProductID INTEGER,
   Price REAL,
   Tax REAL,
   Cost REAL,
   Date TEXT,
   Quantity INTEGER,
   PRIMARY KEY("PurchaseID"),
   FOREIGN KEY("CustomerID") REFERENCES Customers(CustomerID),
   FOREIGN KEY("Barcode") REFERENCES Products(Barcode)
);

CREATE TABLE IF NOT EXISTS User (
   Username text,
   Password text,
   UserType integer,
   CustomerID integer
);

INSERT INTO Customers (CustomerID, Name, Email, Phone, Address, PaymentInfo)
VALUES   (1, 'Alice Appleton', 'aappleton@yahoo.com', '334-555-6123',
      '345 Apple St', 'credit card'),
         (2, 'Bob Baker', 'bbaker@gmail.com', '123-456-7891',
      '123 Abbey Rd', 'paypal'),
         (3, 'Charlie Wilson', 'cwilson@outlook.com', '453-674-1235',
      '7351 Ross St', 'cash'),
         (4, 'Dan Glover', 'dglover@gmail.com', '954-102-6423',
      '123 Main St', 'credit card'),
         (5, 'Eve Mcgee', 'emcgee@icloud.com', '121-644-1345',
      '101 College St', 'credit card');

INSERT INTO Products (Barcode, Name, Price, Supplier, Quantity)
VALUES   (1, 'jacket', 59.99, 'Jackets Inc.', 10),
         (2, 'pants', 21.95, 'Nike', 25),
         (3, 't-shirt', 14.99, 'Old Navy', 100),
         (4, 'dress', 44.95, 'Dress store', 30),
         (5, 'shoes', 49.99, 'Shoemart', 15);

INSERT INTO Purchases (PurchaseID, CustomerID, Barcode, Price, Tax, Cost,
   Date, Quantity)
VALUES   (1, 3, 2, 21.95, 0.05, 22.00, '10/4/2019', 1),
         (2, 5, 5, 41.55, 1.00, 42.55, '10/5/2019', 1),
         (3, 4, 5, 49.99, 0.00, 99.98, '10/6/2019', 2),
         (4, 2, 1, 59.99, 0.01, 60.00, '10/7/2019', 1),
         (5, 5, 2, 24.99, 5.00, 29.99, '10/7/2019', 1),
         (6, 3, 4, 44.95, 3.00, 47.95, '10/8/2019', 1),
         (7, 3, 3, 10.83, 1.00, 11.83, '10/9/2019', 1),
         (8, 1, 2, 21.95, 3.00, 24.95, '10/9/2019', 1),
         (9, 4, 5, 49.99, 0.01, 50.00, '10/10/2019', 1),
         (10, 1, 1, 51.12, 3.00, 54.12, '10/11/2019', 1);

INSERT INTO User (Username, Password, UserType, CustomerID)
VALUES ('admin', 'admin', 0, 0)
      ('manager', 'manager', 1, 0)
      ('cashier', 'cashier', 2, 0)
      ('alice', 'alice', 3, 1)
      ('bob', 'password', 3, 2);
```