CS 4347.003

December 2, 2021

UTD Vending Machines Database

Usaid Malik, Sujay Vadlakonda, Brennan Grady, Andy Tran, Zachary Trundle

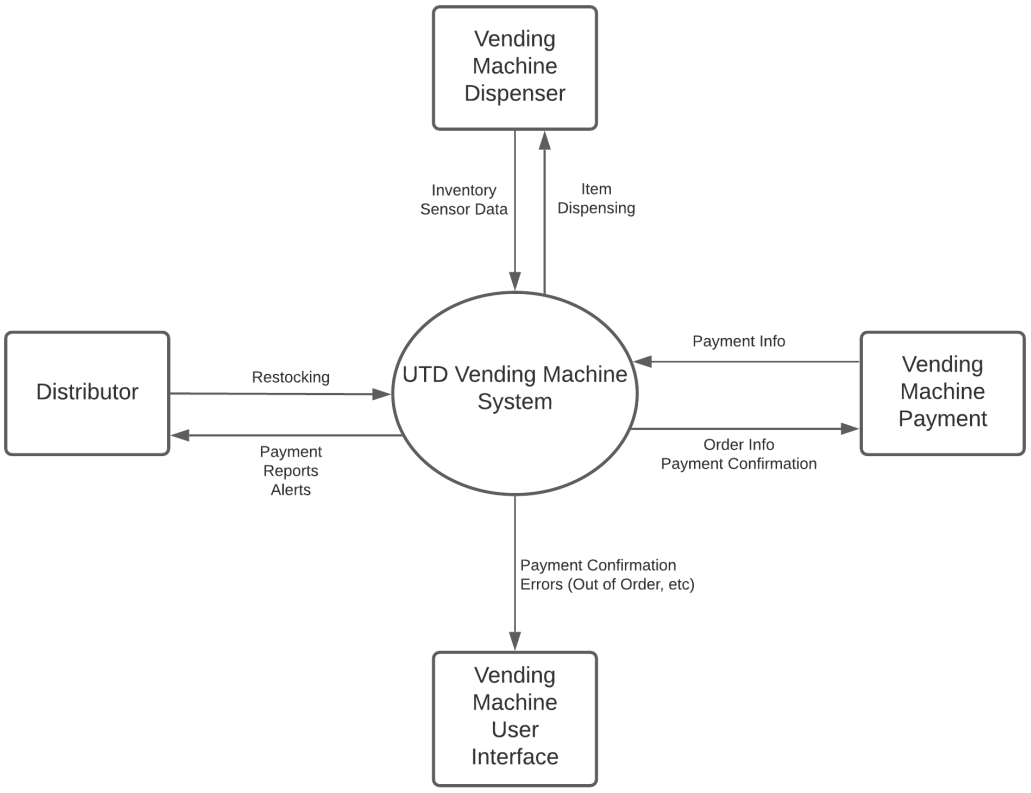
Table of Contents

Introduction

The purpose of this project is to design and implement a database which tracks the status of vending machines around campus by keeping tabs on their contents, the amount of change they hold, type, temperature, etc. It also tracks certains details about the contained items, and the people who supply or pay for the machines. Using this info, the database provides an application interface that lets users submit queries and edit certain information within the database. In the event of a query, the database will quickly retrieve the information and format it for the user view, then once the user is finished the application ends gracefully.

This report is broken up into several sections to better present vital information needed for the operation of the database. It provides several conceptual sections like System Requirements and Logical DB Schema which allow for an in-depth understanding of the backend. Similarly, this report features several sections for the operation and implementation of the database to allow for anyone to quickly utilize it. Lastly, the Appendix contains a link to a zip file with many much needed documents required for the operation of the database.

System Requirements



Functional Requirements

- Takes input from the user, through keypad or touchpad.
- Transaction for payment with the user's bank and the distributor.
- Distributing product.
- Alerting for maintenance.
- Temperature.

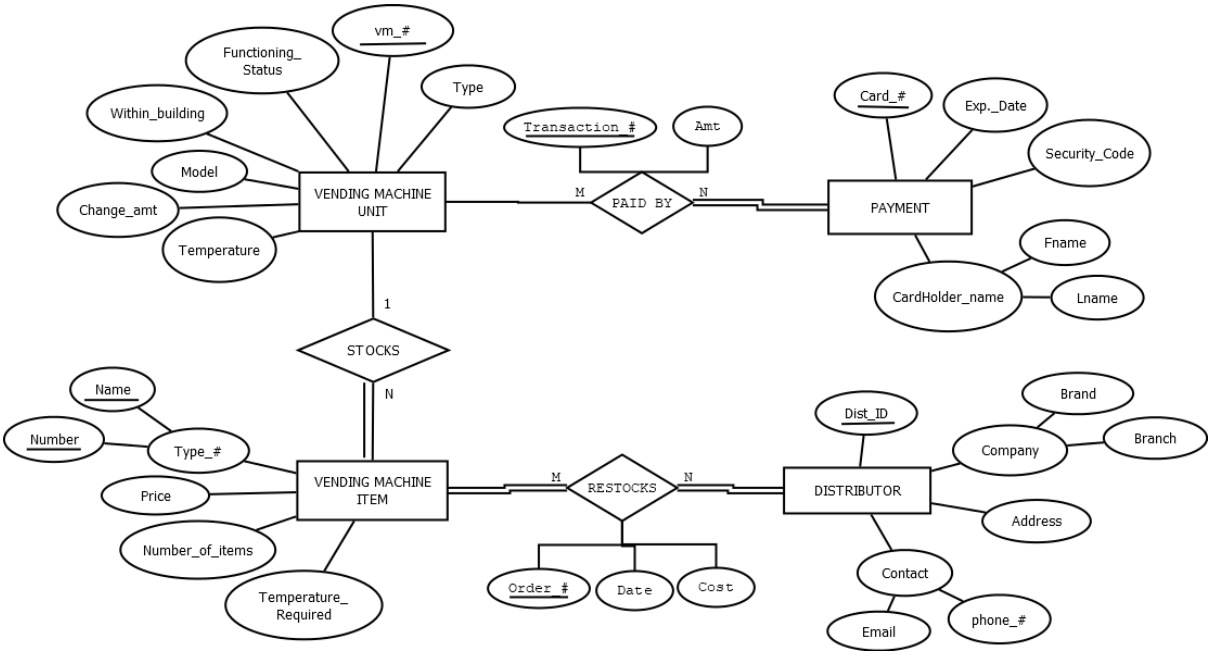
Non-Functional Requirements

- Accessible
- Security
- Time to complete operations
- Reliability

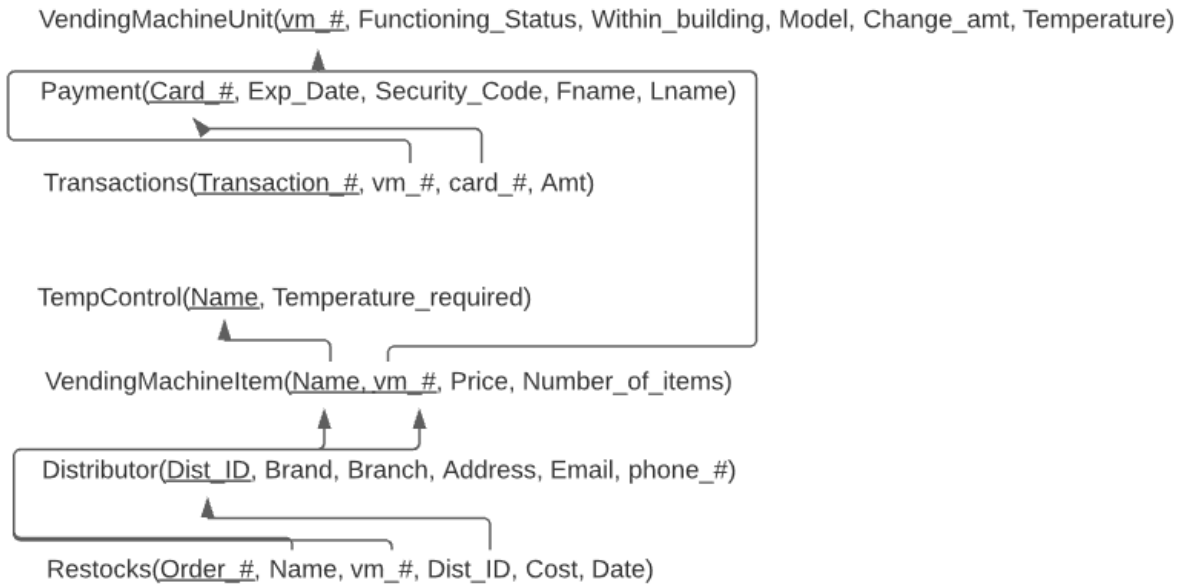
Interface requirements

- Menu that takes a query and responds with the appropriate data.
- Visualization/Listing of vending machine locations and contents.

Conceptual Design of the Database



Logical Database Schema



Creating VendingMachineUnit table

```
CREATE TABLE VendingMachineUnit (
    VM_No              INT     auto_increment      NOT NULL,
    Functioning_Status BOOL                        NOT NULL,
    Within_Building    BOOL                        NOT NULL,
    Model              varchar(100)                NULL,
    Change_Amt         INT                         NOT NULL,
    Temperature        FLOAT                       NULL,
    CONSTRAINT VendingMachineUnit_PK PRIMARY KEY
    (VM_No));
```

Creating Payment table

```
CREATE TABLE DBFinalProj.Payment (
    Card_No            INT           auto_increment    NOT
NULL,
    Exp_Date           DATETIME                        NOT
    NULL,
    Security_Code      SMALLINT      UNSIGNED          NOT
    NULL,
    Fname              varchar(100)                    NOT
    NULL,
    Lname              varchar(100)                    NOT
    NULL,
    CONSTRAINT Payment_PK PRIMARY KEY (Card_No));
```

Creating Transactions table

```
CREATE TABLE DBFinalProj.Transactions (
```

```
Transaction_No    INT    auto_increment    NOT NULL,
vm_#              INT                      NOT NULL,
Name              VARCHAR(20)              NOT NULL,
Card_No           INT                      NOT NULL,
Amt               FLOAT                    NOT NULL,
CONSTRAINT Transactions_PK PRIMARY KEY
(Transaction_No),
CONSTRAINT Transactions_FK FOREIGN KEY (Card_No)
REFERENCES DBFinalProj.Payment(Card_No) ON UPDATE
CASCADE,
CONSTRAINT Transactions_FK_1 FOREIGN KEY (vm_#)
REFERENCES DBFinalProj.VendingMachineItem(vm_#) ON
DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT Transactions_FK_2 FOREIGN KEY (Name)
REFERENCES DBFinalProj.VendingMachineItem(Name) ON
DELETE CASCADE ON UPDATE CASCADE);
```

Creating TempControl table

```
CREATE TABLE DBFinalProj.TempControl (
    Name              VARCHAR(20)      NOT NULL,
    Temp_Required     FLOAT            NOT NULL,
    CONSTRAINT TempControl_PK PRIMARY KEY(Name));
```

Creating VendingMachineItem table

```
CREATE TABLE DBFinalProj.VendingMachineItem (
    Name              VARCHAR(20)      NOT NULL,
    vm_#              INT              NOT NULL,
    Price             FLOAT            NOT NULL,
    Number_of_Items   INT              NOT NULL,
    CONSTRAINT VendingMachineItem_PK PRIMARY KEY(Name,
vm_#),
    CONSTRAINT VendingMachineItem_FK FOREIGN KEY(vm_#)
REFERENCES DBFinalProj.VendingMachineUnit(vm_#) ON DELETE
CASCADE ON UPDATE CASCADE,
    CONSTRAINT VendingMachineItem_FK_1 FOREIGN
KEY(Name) ON DELETE CASCADE ON UPDATE CASCADE);
```

Creating Distributor table

```
CREATE TABLE DBFinalProj.Distributor (
    Dist_ID       INT              NOT NULL,
    Brand         VARCHAR(10)      NOT NULL,
    Branch        VARCHAR(10)      NOT NULL,
    Address       VARCHAR(15)      NOT NULL,
```

```
    Email       VARCHAR(25)     NOT NULL,
    Phone_#     INT             NOT NULL.
    CONSTRAINT Distributor_PK PRIMARY KEY(Dist_ID));
```

Creating Restocks table

```
CREATE TABLE DBFinalProj.Restocks (
    Order_#     INT             NOT NULL,
    Name        VARCHAR(10)     NOT NULL,
    vm_#        INT             NOT NULL,
    Dist_ID     INT             NOT NULL,
    Cost        FLOAT           NOT NULL,
    Date        DATE            NOT NULL,
    CONSTRAINT Restocks_PK PRIMARY KEY(Order_#)
    CONSTRAINT Restocks_FK FOREIGN KEY(Name, vm_#)
REFERENCES DBFinalProject.VendingMachineItem(Name, vm_#) ON
DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT Restock_FK_1 FOREIGN KEY(Dist_ID)
REFERENCES DBFinalProject.Distributor(Dist_ID) ON DELETE
CASCADE ON UPDATE CASCADE);
```

Expected Database Operations:

- Change Item Price.
- Get All Vending Machines with an Item.
- Get Amount of Change in the Vending Machine.
- Get Last Restock From a Distributor.
- Get Number of Transactions for each Vending Machine

Estimated Data Volume:

Estimated data volume depends on the table. In general, we expect in order from greatest to least - transactions, unique payments, restocks, vending machines, and distributors. Using this sort of “relative” ordering, we can get an idea of the estimated data volume.

- On average, an American buys $27/yr of items from vending machines. Given the average vending machine transaction is $1.75, that equates to ~15 transactions per frequent vending machine user per year [1].
    - Using this, we can assume that there would be around 15 Transaction data tuples for every Payment, since a single Payment tuple indicates a unique card (likely unique user).
- According to some vendors, snack vending machines restocked every 1-2 weeks [2].
    - Therefore, every month the Restocks table would have added an additional estimated 0.5N tuples corresponding to restocks from N active Distributors.
- The average number of unique items per vending machine is 40-60 [3].

Overall, the estimated data volumes are as follows:

- The estimated size of Transactions would be around 15x the size of Payments

- The estimated size of Restocks would be Y * 0.5N * M, where Y is the number of weeks on-average the vending machines have been active, N is the number of active distributors per vending machine, and M is the number of vending machines.
- The estimated size of VendingMachineUnit would be around 2 to 3 times larger than Distributor given a distributor would likely be in charge of several vending machines.
- The estimated size of TempControl would at minimum match the number of VendingMachineItem and at maximum be the size of VendingMachineUnit multiplied by the size of VendingMachineItem.
- The estimated size of VendingMachineItem would be around 100, since on average vending machines stock 40-60 unique items and most likely vending machines would share many items in the overall “catalog”.

Functional Dependencies and Database Normalization

RESTOCKS
Order # → { Cost, Date, Dist_ID, Type_# }

Because Dist_ID is the primary key of the entity DISTRIBUTOR, it has a functional dependency to all the other attributes of DISTRIBUTOR.

Because vm_# is the primary key of the entity VENDING MACHINE UNIT, it has a functional dependency to all the other attributes of VENDING MACHINE UNIT.

Essentially, Order # has a functional dependency to all the attributes of DISTRIBUTOR, VENDING MACHINE UNIT, and RESTOCKS.Cost and RESTOCKS.Date.

TRANSACTIONS
Transaction # → { Amt, Card_#, vm_# }

Because Card_# is the primary key of the entity PAYMENT, it has a functional dependency to all the other attributes of PAYMENT.

Because vm_# is the primary key of the entity VENDING MACHINE UNIT, it has a functional dependency to all the other attributes of VENDING MACHINE UNIT.

Essentially, Transaction # has a functional dependency to all the attributes of PAYMENT, VENDING MACHINE UNIT, and PAID_BY.Amt.

STOCKS
VENDING_MACHINE_ITEM.vm_# → { VENDING_MACHINE_UNIT.vm_# }

Because VENDING_MACHINE_ITEM.vm_# is the primary key of the table VENDING_MACHINE_ITEM, it has a functional dependency to all the other attributes of VENDING_MACHINE_ITEM.

Because VENDING_MACHINE_UNIT.vm_# is the primary key of the table VENDING_MACHINE_UNIT, it has a functional dependency to all the other attributes of VENDING_MACHINE_UNIT.

For the STOCKS relation, VENDING_MACHINE_ITEM.vm_# determines the values of all the attributes of the participating VENDING_MACHINE_ITEM and VENDING_MACHINE_UNIT.

<u>Normalization:</u>

VendingMachineUnit FDs:

VM_No → { Functioning_Status, Within_building, Model, Change_Amt, Temperature }

Payment FDs:
Card_No → { Exp_Date, Security_Code, Fname, Lname }

Transactions FDs:
Transaction_No → { vm_#, Name, Card_No, Amt }

VendingMachineItem FDs:
Name, VM_No → { Price, Number_Of_Items }

TempControl FDs:
Name → { Temperature_Required }

Distributor FDs:
Dist_ID → { Brand, Branch, Address, Email, Phone_No }

Restocks FDs:
Order_No → { Name, Number, Dist_ID, Cost, Date }

These Functional Dependencies are already in 3rd Normal Form, so no normalization is required or applicable to the above FDs. Similarly, the SQL code for constructing normalized tables is not-applicable, so those tables are the same as the previously shown SQL initialization tables.

The Database System

We used SQLite as our database system. We chose SQLite because it had a relatively small size while still maintaining all the DBMS functionality we needed to create the project. On MacOS, the operating system our application runs on, SQLite comes preinstalled. SQLite can be invoked using SQL in the terminal or using Python with the ORM provided by SQLAlchemy.

```
sujayvadlakonda@Sujays-MacBook-Pro CS-4347-Project % sqlite3 project.sqlite3
SQLite version 3.32.3 2020-06-18 14:16:19
Enter ".help" for usage hints.
sqlite> select * from item_vm;
Water|3
sqlite> insert into item_vm
   ...> values('Soda', 3)
   ...> ;
sqlite> select * from item_vm;
Water|3
Soda|3
```

Additional Queries and Views

This creates a view that holds the vending machine numbers, ordered by the amount of change, and displays the average price of all the items in the machine.

```
CREATE VIEW view1 AS
SELECT VendingMachineUnit.vm_#, AVG(VendingMachineItem.price)
FROM VendingMachineItem, VendingMachineUnit
WHERE VendingMachineItem.vm_# == VendingMachineUnit.vm_#
GROUP BY VendingMachineUnit.vm_#
ORDER BY Change_Amt;
```

```
sujayvadlakonda@Sujays-MacBook-Pro CS-4347-Project % sqlite3 project.sqlite3
SQLite version 3.32.3 2020-06-18 14:16:19
Enter ".help" for usage hints.
sqlite> create view view1 as
   ...> select vm.vm_no, AVG(price)
   ...> from vm, item_vm
   ...> where vm.vm_no = item_vm.vm_no
   ...> group by vm.vm_no
   ...> order by change_amt
   ...> ;
sqlite> select * from view1
   ...> ;
1|6.16666666666667
2|6.16666666666667
3|6.16666666666667
4|6.16666666666667
5|6.16666666666667
```

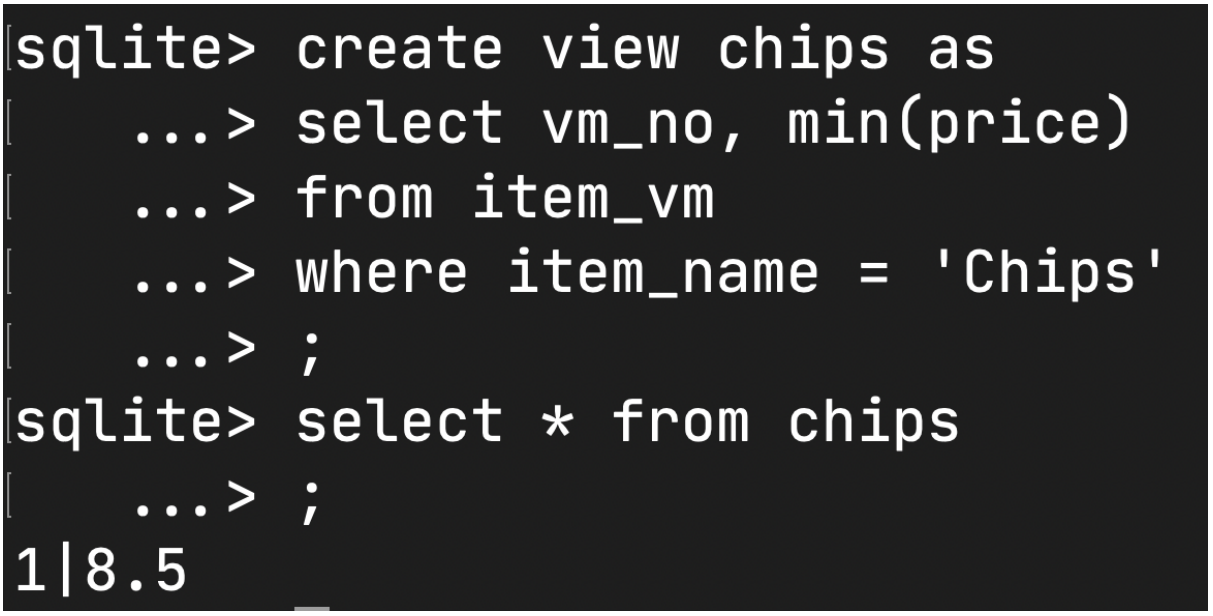
This creates a view that displays the vending machine with the cheapest chips price

CREATE VIEW chips AS

SELECT vm_#, MIN(price)

FROM VendingMachineItem

WHERE item_name = ‘Chips’;



User Application Interface

The UI is built using HTML and CSS, because it is a web application. The workflow for our users is as follows. First, they select the way they want to modify the database from a list of five options. Second, they input any necessary information in a set of input fields. Third, they are directed to a screen that confirms that the change was made, and shows any relevant data in an easy-to-understand manner. The only exception to this is one of the functions which does not take input, and instead gives an overview of the state of the vending machines. The functions provided by our application would be used by the managers of the relevant vending machines. The first function provided is changing the price of an item at a vending machine. It takes an item name, vending machine number, and new price as input. It runs an SQL update on the VendingMachineItem table on the records where item name and vending machine number match. The second function provided is getting all vending machines with a specific item. The SQL will return all vending machine numbers that have a VendingMachineItem record where the item name matches the item name the user is looking for. The third function gives the amount of change in a vending machine and the SQL works by selecting the change attribute of the specific vending machine in question. The fourth function gets the last restock from a distributor. This works by using SQL aggregate functions to find the most recent restock record from the restocks table that has the correct distributor id. The fifth function gets the number of transactions for each vending machine, which is useful for analyzing which vending machine locations are working the best. The SQL works by taking the natural join of the transactions table and vending machine table and using the count aggregate function.

Conclusions and Future Work

Ultimately, we have found this project valuable for the opportunity to implement and test out the ideas we covered in class. The phases of this database project allowed us to have the chance to work with the conceptual and logical elements that are a bit harder to understand from a lecture due to their abstract nature, so our work with them helps us to understand the majority of the concepts better. As for work in the future, we plan to touch up this system slightly, but after that we just plan on moving forward towards newer, more complex projects.

References

[1] "Are Vending Machines Profitable? (2020) | Naturals2Go", *Naturals2Go*, 2021. [Online]. Available: https://www.naturals2go.com/are-vending-machine-profitable/. [Accessed: 04- Dec- 2021].

[2] "FAQ", *Mid Coast Vending*, 2021. [Online]. Available: https://www.midcoastvending.com/faq.html. [Accessed: 04- Dec- 2021].

[3] "Vending Machine Size Guide", *Vendinggroup.com*, 2021. [Online]. Available: https://www.vendinggroup.com/coca-cola-vending-machine-size-guide. [Accessed: 04- Dec- 2021].