PIMSHOE

Authors: April Morrison, Tyler Hiu, Joe Kriviski
Professor: Kreider

Executive Summary:

Virginia Shoe Clinic, a retail shoe store, commissioned this project to help solve their problem of ineffectively tracking retail inventory. This is a pricing and inventory management system that will allow customers, administrators, and sales associates to access the database based on their individual needs. Customers can search for products to see pricing and availability. Administrators can manage discounts and any issues with in-stock items. Sales associates will have tools available to assist the customer with any of their needs and help them check out when they have all their items.

This project will include a website and mobile application to increase accessibility, along with a barcode scanner that makes it easy to find the virtual information of a physical item. The database will take a simplistic approach as it only needs the store to interact with the products through availability of the product and how much the product is discounted. The functionality of the project should be done by midweek 7 leaving the last two weeks to work out any final bugs, issues, and to optimize the entire system.

Table of Contents

Tyler Hiu
Joe Kriviski
This text is intentional due to the Table of Contents.

Introduction
PIMSHOE is a Pricing and Inventory Management system designed for tracking retail inventory at a shoe store with multiple store fronts with shared inventory with no way of tracking inventory effectively. PIMSHOE should let users know whether items are in stock at a store or other stores.

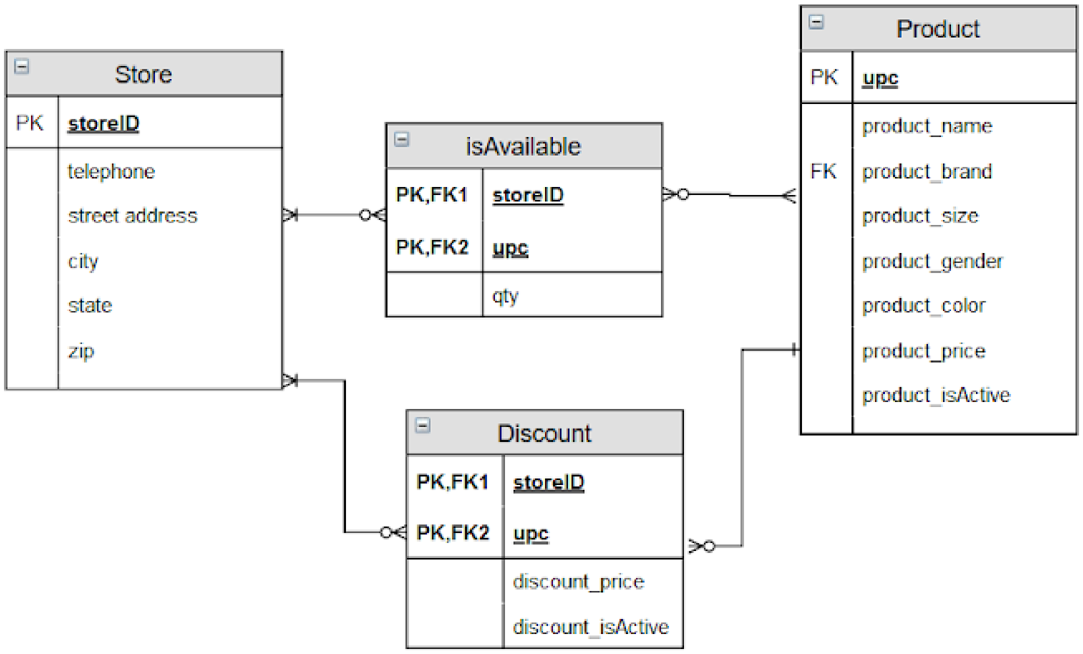
Project Background
This project is important as if it can fully work, it will likely be used by a shoe store chain that April has interned at for the past two years. Working at the company, April found that the most pressing issue the store had was no way to manage their inventory and would constantly have to call other stores to see if they had products available. This, coupled with the fact that vendors would put things on clearance so often, would make sales associates have to search through paperwork to tell customers the prices of products. Unfortunately, the company's primary focus has been repairing shoes, and they have only recently moved towards repairs and retail, so their POS system is based in repairs and does not handle retail products or sales, making it so all retail items must be inputted in manually at checkout. Instead of rebuilding the wheel with a new POS system, April decided it would be lower risk to create a supplemental inventory system to work alongside the current inventory system until and potentially after the developer for the POS system releases a planned retail patch.

Project Description
The project, upon completion, should have an administrator end where items and discounts are created and updated. An employee end should also exist for employees to manage check ins and check outs. Customers and sales associates should be able to search products to see availability and price. I intend for the project to be used by Virginia Shoe Clinic. The project fits into the greater scheme of information systems as a web application.

Proposed Solution
Development Approach
As this is an idea that was planned months ago, we intend to use a waterfall approach. None of the requirements should change at this point, and any changes that would be made would be purely cosmetic or additional to increase user experience.

High Level Plan
We will use AWS for any hosting we'd need. We're going to store code on Github. Languages used should be HTML5 with Bootstrap to supplement CSS, MySQL, and Django (Python). Time permitting, we may try to dabble in Google Authenticate.

The database should look somewhat as follows by the ERD:



Product

The Product table will exist to have a primary key as a 12 digit integer (UPC). Product name should contain a product name with the type of varchar(20). Product brand should contain the brand name with the type of varchar(20). Product size should be able to hold the values s, m, l, xl for products that require those values, and American and European sizes from 4-15 and 35-47 with their respective half sizes. Thus, product size should be varchar(4). Product gender should be a varchar(1) with values w, m, and k. Product color should be a varchar(10). Product price should be a decimal(3,2). It is unlikely a pair of shoes at this store would cost more than 400 dollars, so it is unnecessary to have 4 digits of storage. Lastly, it's important to know if the product is active or not. If it is inactive, it should not show up in search results. UPC, name, brand, color, price, and isActive should not be null.

Store

Store's primary key is a storeID with a 1 digit number, as this is for a small retailer. Each store should have a telephone number as a 10 digit integer. Street address should be a varchar(90) type variable, city should be varchar(20), state should be varchar(2), and zip should be a 5 digit integer. No values should be null.

isAvailable

IsAvailable should take in the storeID from the store class and UPC from the product class and assign a quantity integer of size 2. No values should be null, nor should qty be negative. A store can have 0 or more products available, and a product can be available at 0 or more stores.

Discount

Discount also takes in a storeID from the store class and UPC from the product class. Discounts have a discount price of decimal(3,2) and a binary value for whether a discount is active or not. A store can have zero or many discounts, and a product can also have zero or many discounts.

For the web pages, it’s important to have an administrative site and user site.

Employee

Employees can work at any stores and therefore are not included in the ERD. This class deals more with login data, so it will be used for that. Employees have a primary key of a 4 digit employee code, first name, last name, employee email, whether they are active or not, and whether they are admin or not.

Administrative Site

There will be one site for admins and employees. There should be a login page, new item page, update item page, manage discounts page, check out page, check in page, create new store, update store, create new employee, and update employee. All admins are employees, but not all employees are admins, so some pages will only be viewable based on employee permissions. The administrative site needs to be able to work on mobile devices.

Login

The login page should have a dropdown where people logging in select a store based on storeID, a space for employee ID, and password. A “forgot password” link should be at the bottom. If a username/password is incorrect, the system should reject it and come back with “We’re sorry, that username or password was incorrect. Please try again.” If “Forgot password” is clicked, a user should input their username and email. The screen should then change to say “if the username and phone number exist in the system, you will be sent a 5 digit code. If you do not receive one, please contact your system administrator.” With a form to enter and submit a 5 digit code. This part may work better with google authenticate if we cannot get it to work.

Check in

When utilizing check in, each item to check in should be scanned individually, and the system should produce a table featuring the upc and name of each item scanned along with an “x” to delete the item if it was scanned incorrectly. There should also be “complete check in” and “back” buttons. Upon pressing “complete check in”, the system should send a “are you sure?” message with a yes or no response. Each item should go into the database as a +1 to qty in the appropriate isAvailable value.

Check out

Check out should work in a similar way, items are scanned individually with a table of similar value, and the module should have “check-out” and back buttons with a “are you sure” dialogue. Each item should go into the database as a -1 to qty where needed.

Create Product

New items should be created in a form with a space for UPC, product name, product brand, product size, product gender, product color, product price, and whether the product is active or not. Product name, brand, and color should take in a string related to the varchar values in the database. Size and gender should be dropdown menus. Price should be a decimal value, and isActive should be a checkbox. UPC is a 12 digit integer.

Update Product

Items should be updated with a form like the new items form, but the administrator should input the existing UPC to produce a form that is already filled in with values based on the database. The administrator can then edit those fields and resubmit. UPC should not be changeable. Instead of creating a new product in the database, the database should update the current values.

Discount Manager

Discounts should automatically populate in the database when new products are added by taking the UPC of a product and each storeID as primary foreign keys, a null discount price, and the binary discount_isActive as 0/false. To update a discount, an administrator should scan the upc of the product to discount and insert a discount price (same type as product price) and check a checkbox on if the discount is active or not. Time permitting a "activate/deactivate discounts" module may be attempted that will allow the administrator to see a list of all discounts where discount price is not null and a checkbox to toggle discounts as active or inactive.

Create Store

Admins should be able to create new stores through a form that takes in store telephone number and address. This form does not need to take in a primary key number as it is auto incrementing.

Update Store

Admins should update stores by selecting a store number, then editing the fields they need to edit, like done for editing discounts or products.

Create Employee

Admins can create employees through a form with employee ID (typically last 4 digits of their phone number), Employee first name, employee last name, stores they work at, and isAdministrator. isAdministrator should act as a tic as to whether an employee can access a module or not. Employees can work at all stores depending on their schedules, so this creates a many to many link with store.

User Site

The user site needs to have a search where you can search via UPC, or by product name. The UPC should work by just scanning the UPC, while the keyword search takes in a product name given and displays results like that. Filters should exist on the side for brands pulled from the product brands, size, and color to narrow results. The user should be able to select an item from results. This will open a page with the item name, brand, size, color, price, and discount next to price if available. If the product doesn't exist or no items could be found, search should come back stating "sorry, the requested item could not be found."

When viewing a particular product searched for, a web page should display the product name, brand, size, color, availability per store, and price. If a discount exists, a red line should go through the price and the discount should appear in red to the left of it. Availability should work as follows: each store should be listed by city name. Next to the store name, the webpage should say "in stock" in green if qty for that store's inventory is greater than 0, and "out of stock" if that store's inventory is 0. Quantity should never be negative.

Security

Whether PIMSHOE is used by Virginia Shoe clinic or any other retailer in the future, PIMSHOE must be secure. The user site should be accessible from anywhere. Meanwhile, the PIMSHOE admin site should only be accessible from IP addresses found in retail stores that use PIMSHOE. It is important for us, considering this, to whitelist the admin site so that it can only be accessible from a retailer's IP.

Project Schedule

Work Breakdown Structure

To complete the project, we must set up a server, design and implement the database, create an admin site, create a user site, link the database to the sites, enter starting data into the database, and test the project. We should also ensure the user site and at least check in/check out can be accessed from mobile. We expect the project to take around 11 weeks so that we can comfortably finish the project and make sure it is designed well.

Project Calendar

Intermediate Milestones

Intermediate Milestone #1

- By intermediate milestone 1 we should have our database completed and populated with some data. The server should be running and the login should be function. The forms for the user site should be completed. Admin site home should be completed, as should the user search query forms.

Intermediate Milestone #2

- The user site should return query results that can be clicked on to view details. Check-in, check-out, and the create modules should be completed for admin and read to the database. At lease one update module should be completed. Should have a home page to show for mobile.

Weekly Updates

Weekly Update #1

- ERDs completed (1 hour)
- Mockups setup and revision (6 hours)
- Github setup (30 minutes)

Weekly Update #2

- Server setup (1 hour)
- Mockups for mobile (2 hours)
- Start setting up database for products (1 hour)
- Preliminary user account setup (1 hour)

Weekly Update #3

- Database setup is complete (3 hours)
- Finish user account setup (2 hours)
- Admin site – home page (3 hours)
- Admin site – check in (5 hours)
- Admin site – check out (2 hours)
- User site – home page (3 hours)
- User site – search via UPC (5 hours)
- User site – search via terms (1 day)
- Admin site login – (2 hours – 1 day)

Weekly Update #4

- Database is populated (5 days)
- Admin site – create new product (1 day)

Weekly Update #5

- Any leftover data entry from week 4 (0-2 days)
- Database testing (3 Hours – 3 days)
- Admin site – create employee (1/2 day)
- Admin site – create store (1/2 day)
- Admin site – update product (1 day)
- Admin site – update discount (1-2 days)
- Admin site – update employee (1-2 days)

Weekly Update #6

- Link admin sites to database (1 day)
- Mobile sites development (weeklong)
- Website testing (1-3 days)
- Data entry validation through website (1-2 days)

Weekly Update #7

- If anything from prior weeks could not be completed, complete it now (0-5 days)

- Add in IP whitelisting to the admin site (1-2 days)
- Website must be functional and linked completely. Use this time for testing and Functionality finalization (1 week)

Weekly Update #8

- Refinement (2 days)
- Forgot password (1 day)
- UI and UX optimization (2 days)
- Readme.txt is made (1 day)

Weekly Update #9

- Testing (weeklong)
- Any extra complexity we have time for like google authenticate (2 days)

Project Deliverables

Documentation

- System Vision Document: 1 page brief description of the vision for this project/system being built
- Stakeholder Identification: Discussion of Virginia Shoe Clinic, needs and requirements. Future user accessibility if needed.
- Analysis Results/User Stories: Results of analysis phased in the form of user stories
- Database Entity Relationship Diagrams: ERDs we made
- GUI Mockups: our website mockups

Technical Deliverables

- Zip Archive with completed code for project and scripts for creating databases
- Link to URL with actively working project for instructor review
- Link to GIT repo with completed code for project including scripts for creating databases
- Instructions/readme document explaining how the instructor can install/test the system that was demonstrated, both from a URL of the active site, as well as if the instructor desires to set it up themselves.

Final Presentation Deliverables

- Poster going over documentation
- Setup of how the webapp would function in store featuring a monitor and keyboard with mouse, upc scanner. Phone connected to the system to show mobile.

Conclusion

Virginia Shoe Clinic will find that at completion of this project they have commissioned, they will be able to more effectively manage their internal pricing and inventory data. By centralizing all the data in a database and use a singular connected system, Virginia Shoe Clinic will be able to more efficiently run their stores, simplifying their employee's workload in regards to inventory and pricing. Customers will be able to search for products from home using desktops or mobile devices in a more refined manner than is currently implemented.

The biggest issues we plan on having to deal with is making sure a synchronization of all systems so that all levels of users will not have conflict and making sure all the information we plan to use synchronizes properly as well. Customer elements of the system must be easily understood while employee elements of the system must require little training and must not be overly complex and difficult to learn. The project should have all functional elements completed by midweek 7 with refinement and debugging reserved for the final two weeks.