# Warehouse Inventory with Distributed Database Replication

Daniel Lucas Thompson
CSC733
December 03, 2015

## Overview

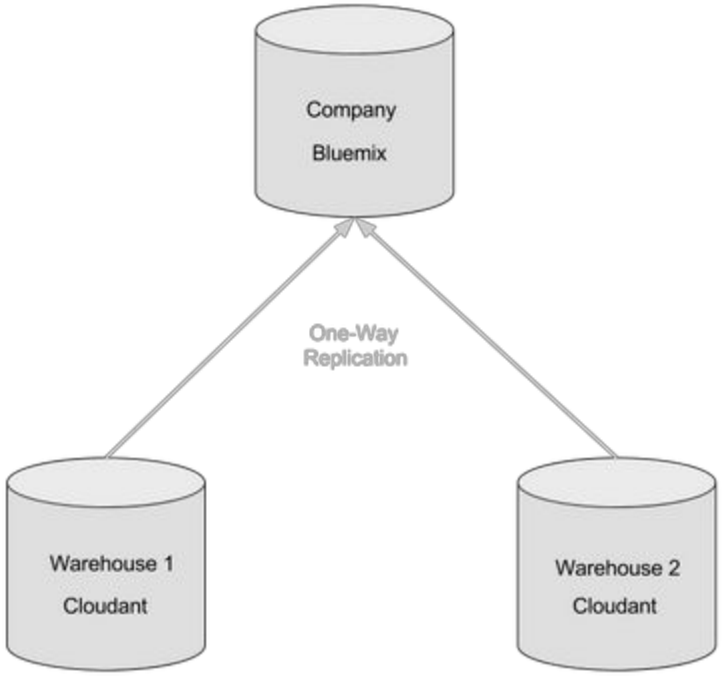
The purpose of this project is to create a distributed database solution for a warehouse inventory system. The project connects two separate inventory databases to a central database. The two inventory databases are set to simulate two separate warehouses having local on-site databases that need to work together. The central database represent a central site to handle all web traffic used in the ordering system.

## Software and Server Setup

1. **Python Flask:** A lightweight web framework for python.
2. **Jinja2:** A web templating library that allows the rendering of dynamic web page content.
3. **Bootstrap:** A CSS framework to get websites looking nice very quickly.
4. **Cloudant CouchDB:** A NO-SQL database based on the CouchDB framework and hosted by Cloudant.
5. **IBM Bluemix:** A platform as a service (PAAS) used to host and run the various frameworks used.

## Database Design

The database design contains a central company database with two warehouse databases replicating to it.




In this design, inventory changes are made in Warehouse 1 and Warehouse 2 located on Cloudant. These changes then replicate up to the Company database that is hosted on Bluemix.

## System Process

When a customer visits the website, they must create an account that includes a username, password, and the district that they live in. Once this account is created, they may browse the website inventories and make orders. When the customer goes to the inventory listing page, the replicated inventories from each warehouse is pulled from the central company database.

When a customer places an order, the inventory is changed directly at each individual warehouse. The district determines the primary warehouse that a customer will receive their ordered items from. When a customer places an order, the warehouse corresponding to their district is checked for the inventory. If the warehouse has sufficient inventory, the order is placed and the customer's order processes. If the warehouse does not have sufficient inventory, the second warehouse is checked to cover the order. If the correct amount of inventory can be obtained from both warehouses, the order is processed. If both warehouses do not contain the correct amount of inventory, the order is reversed and the customer is notified.

Once an order has been successfully processed, a history of that order is created in the company database. This order history is linked to the user's account. The user can then check past orders on the website under their user profile.

## Querying and Views

There are three ways to interact with database data in this system. The first two ways are simple. You can loop over every item in the database or you can use direct access if you know the specific id you are looking for. Direct access is fine to query a customer using their login credentials. However, you would not want to loop over every customer in the database while you were looking for inventory.

To solve this, CouchDB has view functions. Views are precomputed indexes stored in the database. You create a view to index a particular field of an item. For example, in the warehouse databases, all inventory items have a type. The current types are tool and machinery. You can create a view to search by type. You can then query that view with a selector of tool and all the documents in the database of type tool will be returned.

## Problems

During implementation of the project, several problems were encountered and solved.

1. Replication and Document IDs

   In the very first iteration of the project, documents were stored in each warehouse database by item name (hammer, screwdriver, pliers, etc...). This caused problems when attempting to replicate both warehouse databases to the central company database. The first database replicated successfully, but when the second database made an attempt, the document id's it was trying to replicate already existed. To solve this problem, each inventory item id was prefixed with the warehouse number (wh1-hammer, wh2-hammer, etc...).

2. Transactions

   CouchDB does not have a transaction system like traditional SQL databases. You can group updates into transactions, but they will all process even if one fails. This creates a problem in an ordering system. What if a customer order two separate items, and one of them fails? You want to cancel the order and notify the customer. However, with CouchDB, if the second order fails after the first succeeds, you must programmatically update the first item to correct the canceled order.

3. Tables and Foreign Keys

   CouchDB does not use tables or foreign keys. Each CouchDB database is a collection of documents that are hashed by their unique ids. To simulate a foreign key relationship, anytime a document needs to reference another document, it keeps a record of that document's unique primary key. For example, every time a customer places an order, an order history document is created with a unique id. This unique id is then stored under that particular customer's document in the order history section.

4. Race Condition

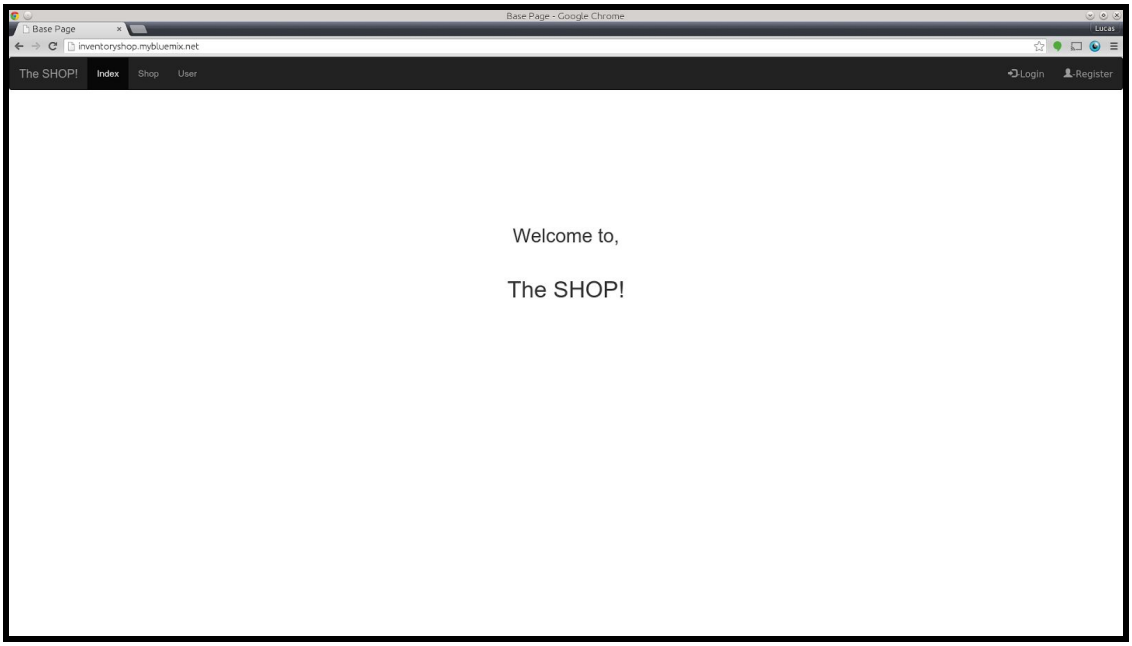
   CouchDB operates with eventually consistent data. This means, when you access data, you may not have the current accurate copy, but the data will eventually be updated with the accurate copy. When running a real time transaction system such as inventory management, this creates a race condition with inventory quantity. Consider the situation where inventory for an item is changed in Warehouse 1 at the exact time

that inventory for an item is changed in Warehouse 2. If these warehouses replicate with each other, both warehouse databases will update their local copy of the inventory. Next, they will try to replicate, and this creates a data conflict because each database is trying to update to a different document. These conflicts can be resolved programmatically, but in a real time system, you would prefer to avoid them all together. To avoid this, the database was designed with two warehouse databases that do one way replication to a central company database as described in the database design section.

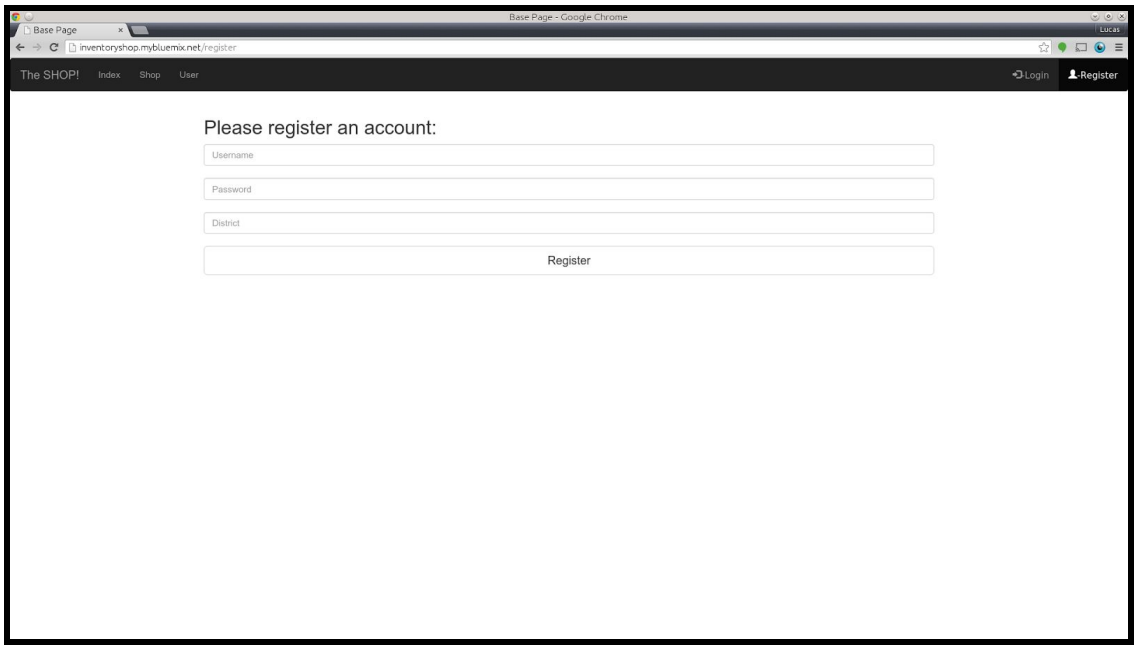
## Website Walkthrough

The following section will be a walkthrough of the current working website.

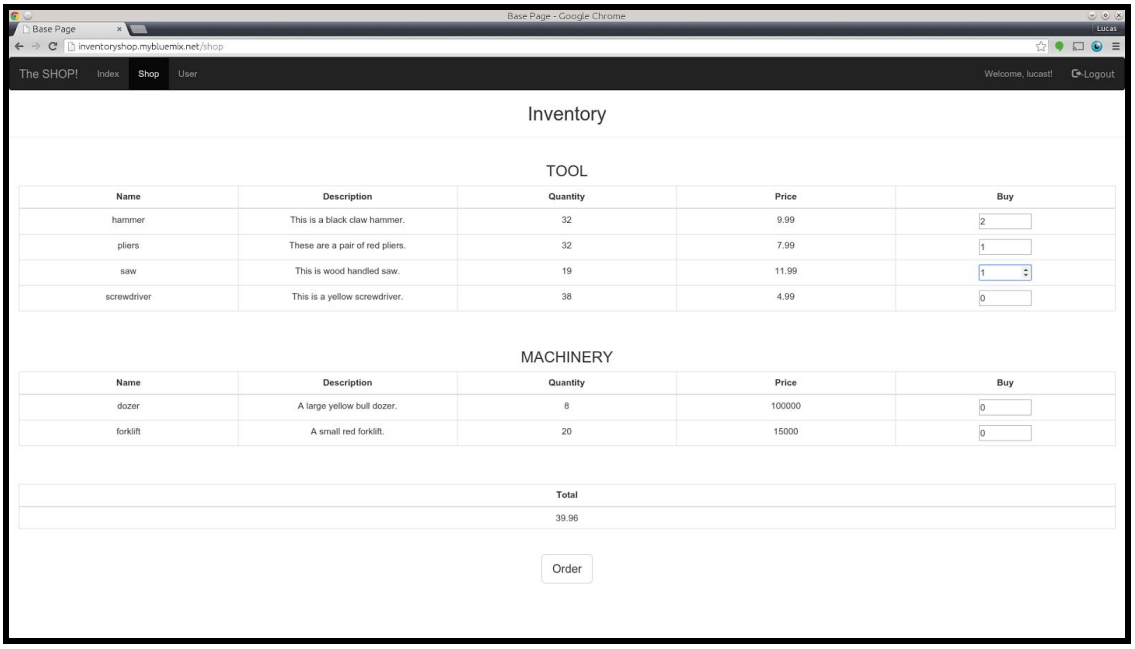
When a user follows the link they land on the basic main page.



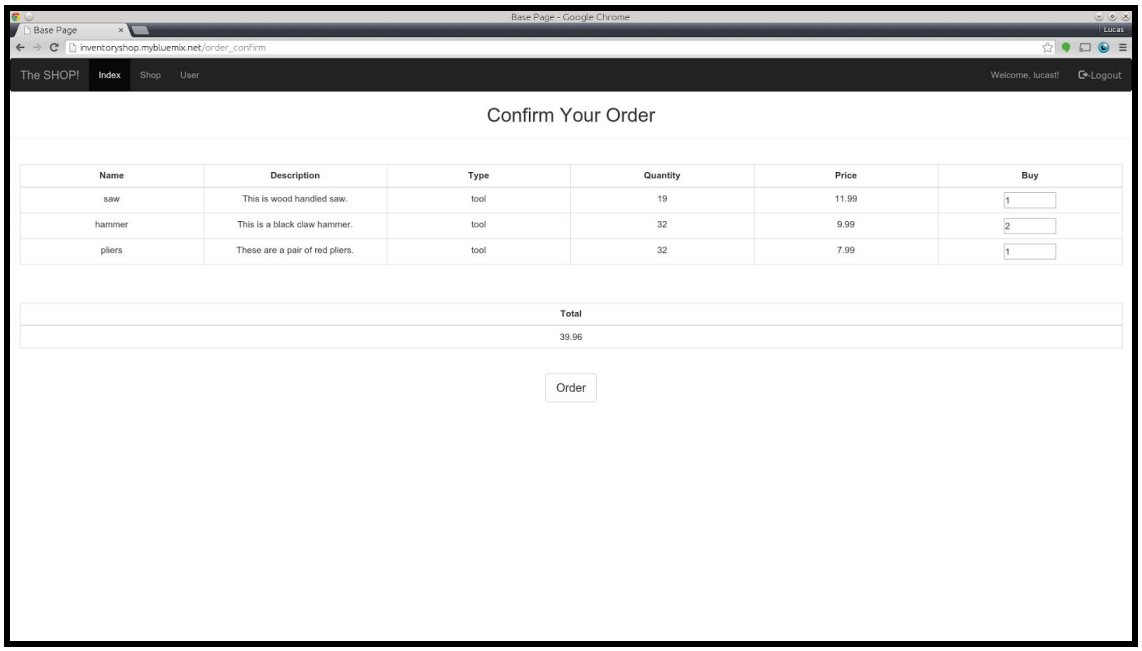
If this is the first time visiting the site, they will need to register an account and login.



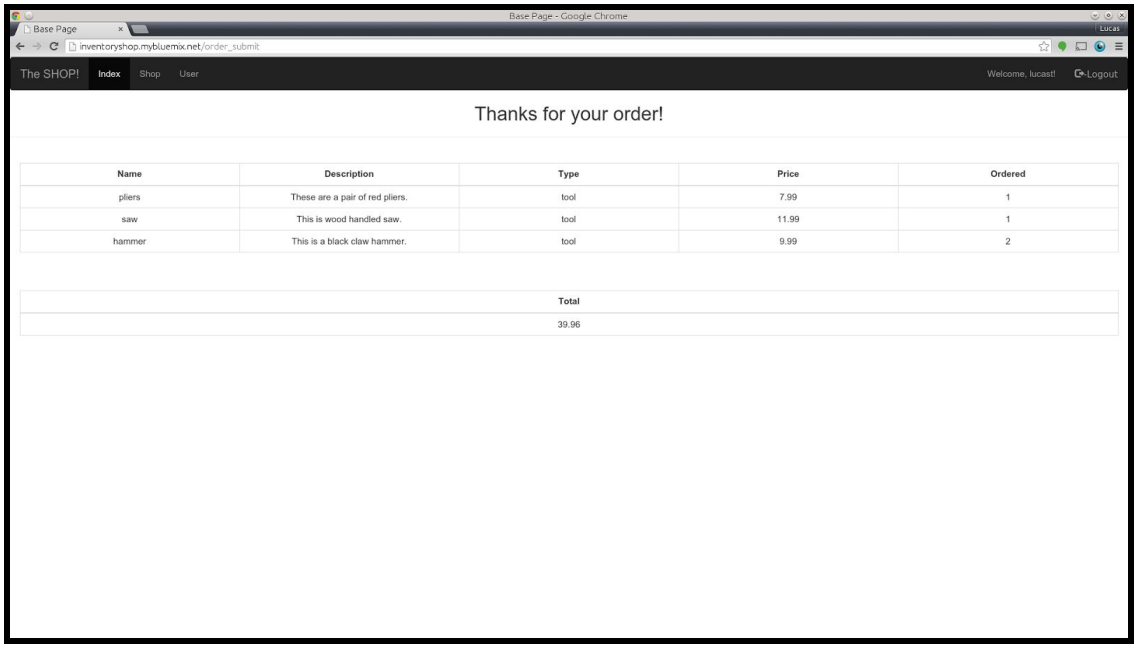

Once they have registered and logged in, they may browse the inventory and place an order.



After selecting items to order, the customer is directed to a page to confirm their order.




After confirming the order, the order is placed.



A customer can view their order history by visiting the user profile link.