

**Introduction (Definition Project Overview ):**

Team 87 Inc. is transitioning from a start up company in StarDew Valley, CA. The company has found itself expanding rapidly in the past 2 years. The amount of inventory at each location has also increased and diversified per customer demand. Finally, the business has started to open across different states in the country.

Note: video did not include the “View Items” function on app.py

**Problem Statement:**

There is a business need to create a database that stores information about the stores that the business is opening. There is also a need to find good locations for a new store and record inventory that is stored for each store. It is also necessary for the store inventory to be accessible for customers who wish to view what is available for purchase. Finally, there is a need to show stakeholders where the stores are located in order to supply the chain of stores in the fast growing business.

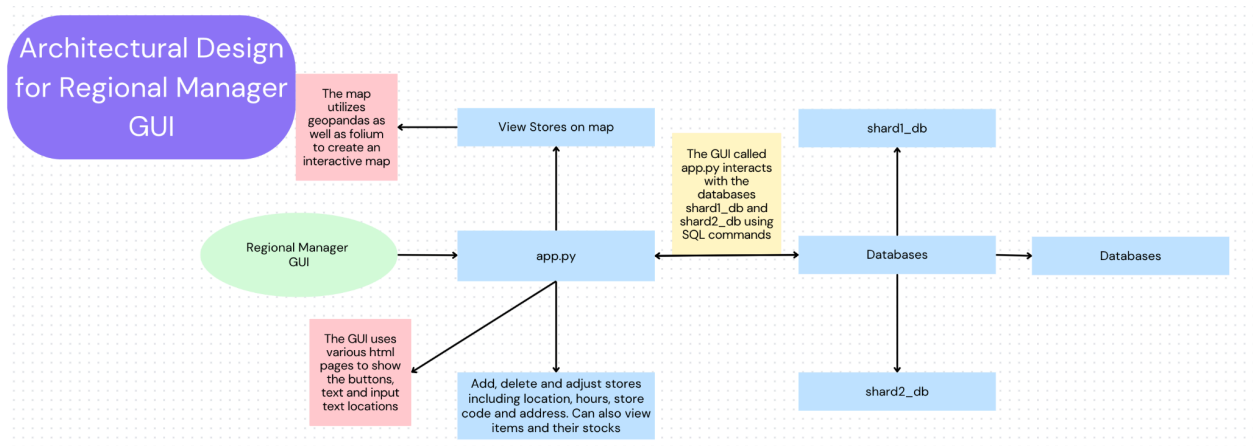
**Planned Implementation (From Project Proposal):**

A database system that mirrors that of major retail chains will be used as inspiration in designing how data will be stored. There are two components involved: the regional manager as well as customer-end interface and the interface for the retailers. Customers will be able to query for items at stores in their surrounding area (using location data); Retailers will be able to modify data regarding inventory of items available at specific stores. Store information will have data distributed in the customer-vendor paradigm. Customer data will include information like availability (y/n), price, and location. Vendor data will include information as items, quantities, price, etc.

The steps in rolling out the database are as follows:

1. Design the format that the retailer data will be stored (schemas)
2. Create a way for retail managers to access and modify this data through an interface
3. Design a database for data customers will be accessing
4. Ensure the customer and vendor databases are in active communication (ACID)
5. Create an interface for customers to make searches over the data (UX)
6. Potentially populate the databases with a Kaggle retail dataset

**Architecture Design (Flow Diagram and its description)**



This is a diagram to help visualize what the Regional Manager can do with the web based GUI. Via the app.py program, the regional manager can add, delete and adjust some of the data about the stores as well as view the items. They can also view the stores on a map to see if there are potential new places to put stores or stores that have overlapping radii, so they could possibly remove one of the stores (if it's in planning, probably not if it is already there).

## Implementation:

### General setup:

In accordance with the read me, mysql and mysql workbench need to be set up. The firewall must not block mysql and mysql should be added to path. Next the user needs to run central\_creation.py, shard\_creation.py, which setup the databases.

### Web GUI:

There is a bit of explanation of the setup for the web GUI as well as a demonstration of some of its use as well as the MySQL databases in the youtube video.

The web GUI uses flask and mysql.connector (should already be installed), so install flask if you do not have it already. To run it navigate to the directory where the app.py file is located and run with python3 app.py. Additionally, the GUI uses folium as well as geopandas, which must be installed. It can be installed using anaconda navigator, as using pip for geopandas was quite troublesome (folium was fine).

In order for the map to have everything, you can have a shapefile of the United States of America which can be downloaded [here](#). Make sure to change the gpd.read\_file path to wherever you downloaded the shapefile.

```
○ (base) john@usc-securewireless-student-new138 Retail_Database % python app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5001
Press CTRL+C to quit
* Restarting with watchdog (fsevents)
* Debugger is active!
* Debugger PIN: 123-264-456
```

This dialogue box should show up, and you can cmd + click on the http link where the initial home page should be up and running in your default browser. Since setting up geopandas can be difficult, I have included a video demonstration. In order for the rest of the code to work, you can simply comment out the import geopandas module (if you were not able to download it properly).

## Functionalities

Web GUI (for regional store manager):

As mentioned earlier, the web GUI is for a regional store manager to view locations and hours of stores to see if there are possible new locations for stores. It takes existing stores from the databases and displays them on the map along with buffers. This buffer is supposed to indicate whether or not the stores are too close too each other. If the manager inserts a new store location they can see if the buffer intersects with other stores.

The first button to use is the “Insert Store Information” button. Here, you can enter the store code, address, opening time, closing time as well as the coordinates which have a suggested minimum and maximum value for America. Based on the store code value the data will be stored in either shard1\_db or shard2\_db. Fields cannot be left blank here, and there are suggested values for the x and y coordinates (suggested puts it somewhere in the United States). The time values must be numerical, if not there will be an error. If they do not adhere to the HH:MM:SS, then the time will most likely be incorrect. The store hours for a store can also be updated by entering the store code and the new store hours for the store (opening and closing). If the store code does not exist, then there is an error and says the store code does not exist. If there is an error in the opening or closing hours (person entered a string) then there is also an error message. The user can also fetch data from a certain store by entering the store ID. If successful, it says data fetched successfully, and then lists the outputs. There is also an option to delete a store, which can be done by entering the store code and clicking delete store. There is also a home button in case the user would like to return to the home page.

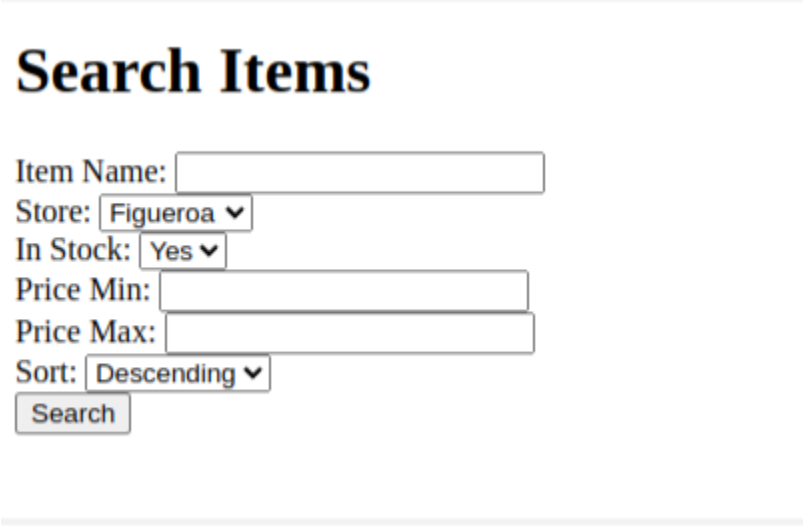
When entering store information, try to enter latitude and longitude values that will be in the United States of America (using degrees), as this is where the map is going to be. The map spatial projection is in WGS (World Geodetic System) 1984 which is a standard reference framing for getting latitude, longitude and elevation. It is also used for google maps which is where you can get coordinates if you would like to select specific stores to see where they are.

Once you open the map the default zoom is at a country wide scale. The highlighted region is the United States of America (including territories, not just the contiguous US). The store locations are displayed with green and red markers with green showing stores from shard 1 and red showing stores from shard 2. If you click onto the marker, it brings you a pop up that displays the address, opening hours and closing hours. You can pan across the map by clicking anywhere and dragging and you can zoom by clicking the plus and minus buttons or scrolling. For visual analysis purposes WGS 1984 works. However, if you wanted to look for a new retail location in just one city, then a projection would need to be applied for proper analysis.

In addition to the functionality revolving store information, the GUI also allows store managers to edit and view inventory relating to the stores. Through the “View Items” tab, admin can access inventory of a selected store. Then, by filling in fields related to items, they can either delete, insert, or update more information into the inventory table. This allows for complete functionality and control over the inventory of items that will later be queried in the customer interface.

#### Customer Interface:

The customer interface has limited functionality and limits the user to look up data. The manner in which customers are able to look up data is in the following way: through item name, whether the item is in stock, through price range (min and max can be set by the customer). Finally the data can be sorted in descending or ascending order. The image below illustrates the user interface:



The image shows a web form titled "Search Items" in a large, bold, black serif font. Below the title are several input fields and dropdown menus. The first row is "Item Name:" followed by a text input box. The second row is "Store:" followed by a dropdown menu showing "Figueroa" with a downward arrow. The third row is "In Stock:" followed by a dropdown menu showing "Yes" with a downward arrow. The fourth row is "Price Min:" followed by a text input box. The fifth row is "Price Max:" followed by a text input box. The sixth row is "Sort:" followed by a dropdown menu showing "Descending" with a downward arrow. At the bottom left of the form is a "Search" button with a light gray background and a thin black border.

#### Tech Stack

Python was the main programming language with bits of SQL to communicate with the mysql databases. The web GUI uses flask as well as folium and geopandas to create the map.

### Implementation Screenshots (Few not all)

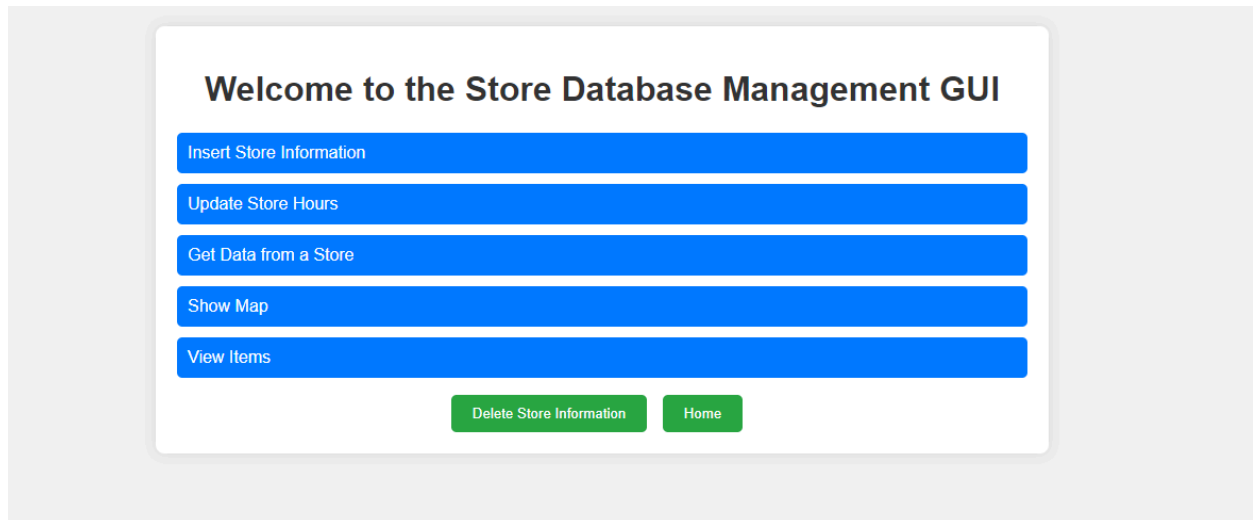


Figure: Web GUI for retail manager

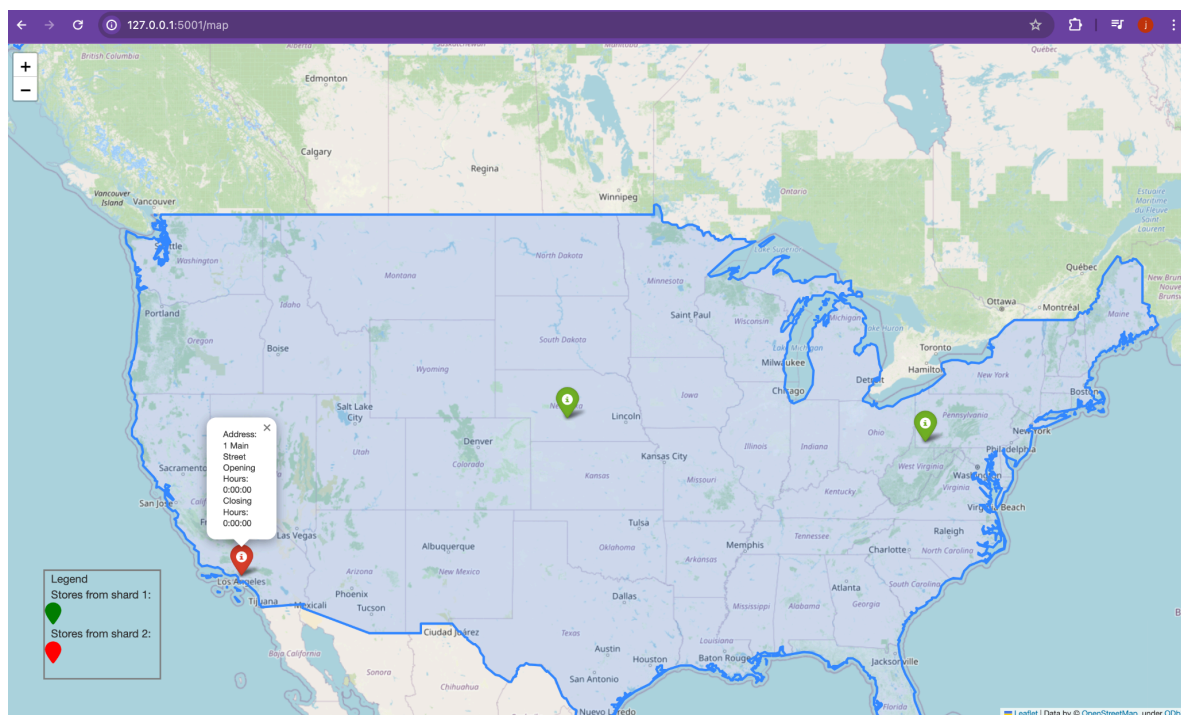


Figure: Map with a few sample locations and the popup activated on one of them

← → ↻ 127.0.0.1:5001/insert\_store ☆ 📄 🛡️ 📱 🔴 ⋮

### Insert Store Information

Store Code:

Address:

Opening Time (HH:MM:SS):

Closing Time (HH:MM:SS):

X Coordinate(-120 to -50):

Y Coordinate(25 to 80):

[Insert Store Information](#)

Figure: HTML page for inserting store information

## Item Management

Store ID:

Item Code:

Item Name:

Quantity:

Price:

[Filter Items](#) [Stock New Item](#) [Restock Item](#) [Change Price](#) [Remove Item](#)

Item Code	Item Name	Quantity	Price
2	Item	10	700.00
3	Paper	10	3.99
4	Toilet Paper	10	8.99
6	Toilet Paper	10	6.99
7	Bike	10	100.00

Figure: HTML page for modifying and viewing the items belonging to stores

```
mysql> use shard1_db;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
```

```
Database changed
```

```
mysql> select * from Stores;
```

store_code	address	opening_time	closing_time	x	y
2	2029 Sawtelle Blvd, Los Angeles, CA 90025	08:00:00	20:00:00	-118.439	34.0444
6	123 Main Street	06:00:00	17:00:00	-118.225	34.1425

```
2 rows in set (0.00 sec)
```

Figure: One of the shards in the mysql database showing some of the entered data

## Learning Outcomes

John:

I learned the importance of good documentation and learning how to set up environments. I also learned the difficulties of connecting a database to a GUI, and that for tutorials to work you need to have the same applications running in the background (never got budibase to work despite various tutorials and help pages). I relearned how important adding a spatial aspect to data is as the visualizations of a map are unparalleled. Being able to see where stores are and where they could be brings a valuable dimension to analysis or just looking for fun.

Christian:

Likewise to John, the importance of clearly documenting and learning how to set up environments was quite clear to me after going through this project. In general, it was illuminating that the challenge of working on collaborative, technical development primarily lies in learning how to communicate and build off of each other effectively. Going forward, I am going to put more thought and time into learning on best practices regarding dividing work, collaborating with others on development projects such as this one.

Jesse:

There was a steep learning curve involved in learning new frameworks (geopandas, folium, flask, etc). Collaborating with team mates helped me learn new frameworks and understand the expectations from the assignment (at times I found myself at a loss on what to do). Another learning experience from the project is the importance of project management; that is, to ensure that everyone has a task to do and to ask questions when they are at a loss on what to do. Finally, working with different machines and different codebases highlighted the importance of designating a code manager who chooses the coding environment (dependency management), reviews and merges code to the main codebase.

## Challenges Faced

John:

I was tasked with the web and regular GUI and I first used streamlit and budibase, both of which I could not get the database to connect. I then used flask, and was able to get it connected to the database almost instantly. Making the buttons on the GUI connect to the sql database and making sure the backend worked properly was quite a challenge because I had to rewrite certain parts of the backend to make it work for what I had. Additionally, I don't have a lot of html experience and I have never used flask so this was quite new and challenging. However, the hardest part was definitely getting everything set up.

Within the GUI, I was not able to use functions created by my teammates due to some issues, so I had to rewrite a lot of the code within my own program. This felt quite redundant and time wasted as my teammate had already spent time doing this. In the map, I had some trouble just getting it to appear at first as I am usually not using a web GUI to create maps. The majority of my experience was in python notebooks or on a GIS software where things are much simpler and have far more documentation. There was unfortunately an error in projecting the coordinate system as the projection does not change when you run the `to_crs` function. It just changes the metadata for it without actually changing the projection of the coordinates. When I changed to a more appropriate projection, the buffers were not showing up. The issue is most likely due to the unit changing, so instead of angular units it was instead units in feet. This is problematic as -118, 35 in degrees brings you to somewhere in California or western US. However, -118, 35 feet brings you to the Atlantic Ocean off the coast of Africa. I was not able to resolve the issue with geopandas through the web GUI. Since building a map through html and a mysql database on a web GUI is not super common, I was not able to find a reasonable solution. The implications of this are that the buffers will not be super accurate. This gives a good visual solution to "is this store close too close to another?". However, there are more accurate ways of depicting this such as site suitability which will take into account many more factors and will of course use a proper projection. The web GUI still has great visual functionality though, it is just not the one thing you would look for in a site suitability.

I had also not used Github much beforehand and committed to a forked repository from the original one by Christian Becker. This meant that Christian did not see some of John's work that could have been useful for implementing other functions.

Jesse:

A challenge faced was communication and project management. New ideas/features to the project led to import of libraries with dependencies. This led to a fragmented ecosystem causing stakeholders to be unable to run coded effectively on their machines (one solution going into the future is to implement or create a `*.yaml` file with all required dependencies) Another challenge in code management was redundancy in coding; several shard functions were defined



and caused time losses due to debugging. A solution to this is designating a code manager to merge contributions on Git to the main codebase.

Christian:

The main difficulties for myself in this project involved setting up the environments necessary for the code, as well as integrating the front-end with the back-end. I faced substantial challenges making sure that the functions that had been defined in the back-end were working properly on the GUIs for both the admin and customers. Oftentimes, the interface would not operate as expected, causing confusion about where along the pipeline the error would take place. This meant a lot of iterative, trial-and-error efforts to pinpoint problems and get the programs to work correctly.

### **Individual Contribution (for multi-person team)**

Christian Becker:

Primary contributor and group leader. Presented our project to the professor and peers. Showcased all aspects of codebase realtime, that is, Create, Retrieve, Update and Delete. Developed the Admin and Customer interface for the database. I set up the original tables, design of the database system in MySQL. Then, I added to the work done by both John & Hemmer with the GUIs, namely in getting the inventory information working on both the admin and customer sides of the project.

John Hemmer:

Handled a lot of the communication. Created the youtube video showcasing the web GUI for regional manager (app.py, but missing the “view items” function) and showed a little bit of the setup necessary. Created most of the html pages for the web GUI (mine are under template folder, copied over to templates folder as well), created initial GUI (GUI.py) using tkinter and the entirety of the web GUI for retail regional manager (app.py) using flask (show map, insert store, delete store, update hours, fetch data functions, home, return). Set up the web GUI backend as well (connecting to the databases, making the buttons connect to different functions). Added the x and y coordinates into the database creation for both shard dbs as well as the central\_db. Implemented geopandas and folium to create gpd object of the store locations. Added locations to html file with legend as well as a buffer to visualize stores better. Created video to showcase app.py and some implementation. Made flow diagram for the web GUI.

NOTE ABOUT SUBMISSION: John Hemmer (submission by John Hemmer) forked from Christian Becker’s github. Not sure if this matters, but it may only list John Hemmer and Jesse Gallegos as contributors even though Christian Becker made significant contributions earlier on prior to the fork.

Jesse Gallegos:

Note: Deletion methods was overhauled and migrated (original codebase is no longer used). Reviewed the backend working of code. Added and revised aspects of the report adding a narrative to the business need and the technology implemented. Assisted and reviewed code to ensure that the CRUD paradigm was maintained in the code. Added Deletion code base on the implementation merged into the main code base (portions of code were merged to the app.py file). Reviewed and specified the distinction between the interfaces for the Database Manager and the End-User. Jesse's contribution shifted focus to testing code remotely and verifying that code works locally on their machine (this is important in producing a product that maintains quality for the User Experience).

### **Conclusion:**

Two solutions were developed with different users in mind (the Database Manager & the End-User). The Database Manager has a simple interface for Creation, Retrieval, Updates and deletions. The End-User has a more user friendly interface (eye candy). Both mediums offer a varied experience to the user. For example, a Database Manager will want to insert information into the database without needing to see where it is on a map. An End-User will want to see a store location in order to travel to the store and purchase merchandise.

### **Future Scope:**

With the visualization of the stores on the map there is future potential for spatial data analysis. This could include a site suitability for a new store, network analysis and/or 2 step floating catchment. Although the initial scope was for retail stores, this could be expanded or modified to be hospitals with attributes such as beds, amenities and number of doctors or schools with teachers, class sizes, square footage, etc. Being able to add a spatial component may complicate things, but it also opens up a huge amount of potential for analysis and various possibilities.