## INTRODUCTION

There has been a growing popularity in boba, matcha, and milk tea beverages and this project aims to develop a database system in response. This system will comprehensively catalog menu items and prices from different boba shops. The primary goal is to provide users with a powerful tool to help facilitate informed decision-making by enabling price comparison across different establishments. Through intuitive search functionality and filtering, users can easily explore specific beverages or just browse through menu items, empowering them to make selections that align with their budgetary preferences with confidence and ease.

## OBJECTTIVES

**1. Develop a Specialized Database System:** Design and implement a specialized database system specifically for storing menu items and prices from a variety of boba shops.

**2. Facilitate Price Comparison:** Enhance user budgeting by enabling users to compare prices of boba, matcha, and milk tea drinks across different establishments efficiently, aiding in informed purchasing decisions.

**3. Web Scraping & Data Compilation:** Utilize Python to perform web scraping of menu items and prices from selected boba shops, compiling the data into CSVs.

**4. Database Creation:** Upload the compiled data into a PostgreSQL database to establish an organized storage system for easy retrieval and manipulation.

**5. Integration with RStudio:** Integrate the database with RStudio to leverage its capabilities in data visualization and analysis.

**7. Development of Shiny Application:** Transform the database into a user-friendly Shiny application within RStudio, providing an intuitive interface for users to interact with and analyze the collected data effectively.

## METHODOLOGY

### Data Collection

The selection of prominent establishments in Richardson, TX offering boba beverages was established and these included Ding Tea, Fat Straws, Feng Cha, His Tea, Junbi, Tea Daddy, and TP Tea. The target data was the shop name, address, item category, item name, item size, and item cost.

In order to gather the target data efficiently and systematically, Python programming language was employed to conduct web scraping of each boba shop's online menu, using the Beautiful Soup library. It is important to note that each shop's website presents a unique HTML format, requiring the development of custom Python scripts tailored to the specific structure of each site. However, some shop’s websites were so complex that it was faster to manually extract the data rather than creating a customized Python script for it. These complex websites included Feng Cha, His Tea, Junbi, and Tea Daddy. The other shops, Ding Tea, Fat Straws, and TP Tea were web scraped with Python and the codes are shown below. The toppings were extracted manually separately and then merged with the data collected from all the shops.

```python
# Ding Tea

import pandas as pd
import numpy as np
from bs4 import BeautifulSoup

fname = r'./data/DINGTEA_MENU.html'
with open(fname) as f:
    soup = BeautifulSoup(f)

def process_item(item):
    res = {'Shop Name': r'Ding Tea Richardson',
           'Location': r'581 W Campbell Rd, Richardson, TX 75080',
           'Category': r'Tea',
           'Item Name': item.find("div", "name").text
           }
    # res["Item Name"] =
    medium_price, large_price = item.find("div", "cap").text.split()
    res["Size"] = "Medium"
    res["Cost"] = medium_price
    yield res.copy()
    res["Size"] = "Large"
    res["Cost"] = large_price
    yield res.copy()
df = []
menu_items = soup.find_all("div", class_="Txt")
for menu in menu_items:
    category = menu.find("div", class_="itemName")
    if not category: continue
    category = category.find("div", "en").text
    items = menu.find_all('li')
    res = []
    for item in menu.find_all('li'):
        res.extend(process_item(item))
    df.extend(res)
df = pd.DataFrame(df)
df = df[df['Cost'] != '-']
df.head()

df.to_csv('./export_data/DING_TEA_RICHARDSON.csv', index=False)
```

*Figure 1: Ding Tea Web Scraping*

```python
# Fat Straws

import pandas as pd
import numpy as np
from bs4 import BeautifulSoup
from collections import defaultdict

fname = r'./data/FATSTRAWS_MENU.html'
with open(fname) as f:
    soup = BeautifulSoup(f)

menu_items = soup.find_all('div', 'menu paddedContent')
def process_menu_group(grp, main_category):
    subcategory = grp.find('span').text
    for item in grp.find_all('div', 'itemInfo'):
        item_name = item.find('div', 'itemHeader').text
        price = item.find('span', 'price').text.replace('$', '').replace('+', '')
        yield {
            "Shop Name": "Fat Straws Richardson",
            "Location": "1251 W. Campbell Rd, Suite 230",
            "Category": f"{main_category} - {subcategory}",
            "Item Name": item_name,
            "Size": "Regular",
            "Cost": price
        }
df = []
for menu in menu_items:
    main_category = menu.find('div', 'headerWrapper').text
    for menu_group in menu.find_all('div', 'menuGroup'):
        df.extend(process_menu_group(menu_group, main_category))
df = pd.DataFrame(df)
df.head()

df.to_csv('export_data/FAT_STRAWS_RICHARDSON.csv', index=False)
```

*Figure 2: Fat Straws Web Scraping*

```python
# TP Tea

import pandas as pd
import numpy as np
from bs4 import BeautifulSoup
from collections import defaultdict

fname = './data/TPTEA_MENU.html'
with open(fname) as f:
    soup = BeautifulSoup(f)

with open(fname) as f:
    menus_df = pd.read_html(fname)

records = []
for menu in menus_df:
    columns = menu.columns
    if len(columns) != 4: continue
    old_cat = columns[1]
    menu_category = columns[1].replace('(Caffeine Included)', '').strip()
    menu = menu[columns[1:-1]]
    menu = menu[menu['Ice'] != '22oz']
    menu['Ice'] = menu['Ice'].astype(int)
    menu = menu.rename(columns={old_cat: menu_category, 'Ice': 'Cost'})
    size = '22oz'


    for row in menu.to_dict(orient='records'):
        records.append({
            'Shop Name': 'TP Tea',
            'Location': r'400 N Greenville Ave #14, Richardson, TX 75081',
            'Category': menu_category,
            'Item Name': row[menu_category],
            'Size': 'Regular',
            'Cost': round(row['Cost']*7.75/70, 2)
        })

export_df = pd.DataFrame.from_records(records)
export_df.to_csv('export_data/TPTEA_RICHARDSON.csv', index=False)
export_df
```

*Figure 3: TP Tea Web Scraping*

## Data

The final data from web scraping were the two CSV files below.

| ID | Shop_Name | Item_Category | Item_Name | Item_Size | Item_Cost |
|---|---|---|---|---|---|
| 1 | Ding Tea Richardson | Topping | Egg Pudding | TOPPING | 0.65 |
| 2 | Ding Tea Richardson | Topping | Mango Jelly | TOPPING | 0.65 |
| 3 | Ding Tea Richardson | Topping | Sea Cream | TOPPING | 0.65 |
| 4 | Ding Tea Richardson | Topping | Grass Jelly | TOPPING | 0.65 |
| 5 | Ding Tea Richardson | Topping | Lychee Jelly | TOPPING | 0.65 |
| 6 | Ding Tea Richardson | Topping | Aloe Vera | TOPPING | 0.65 |
| 7 | Ding Tea Richardson | Topping | Starfruit Jelly | TOPPING | 0.65 |
| 8 | Ding Tea Richardson | Topping | Coconut Jelly | TOPPING | 0.65 |
| 9 | Ding Tea Richardson | Topping | Golden Boba | TOPPING | 0.65 |
| 10 | Ding Tea Richardson | Topping | Mango Popping Boba | TOPPING | 0.65 |
| 11 | Ding Tea Richardson | Topping | Coffee Jelly | TOPPING | 0.65 |
| 12 | Ding Tea Richardson | Topping | Rainbow Jelly | TOPPING | 0.65 |
| 13 | Ding Tea Richardson | Topping | Strawberry Popping Boba | TOPPING | 0.65 |
| 14 | Ding Tea Richardson | Topping | Extra Syrup | TOPPING | 0.65 |
| 15 | Ding Tea Richardson | Topping | Crystal Boba | TOPPING | 0.75 |
| 16 | Ding Tea Richardson | Tea | Caffe Americano | Medium | 2.7 |
| 17 | Ding Tea Richardson | Tea | Assam Black Tea | Medium | 3.45 |
| 18 | Ding Tea Richardson | Tea | Jasmine Green Tea | Medium | 3.45 |
| 19 | Ding Tea Richardson | Tea | Oolong Tea | Medium | 3.45 |
| 20 | Ding Tea Richardson | Tea | Caffe Americano | Large | 3.95 |

*Figure 4: CSV File #1*

| Shop_Name | Shop_Address |
|---|---|
| Ding Tea | 581 W Campbell Rd, Richardson, TX 75080 |
| Fat Straws | 1251 W. Campbell Rd, Suite 230 |
| Feng Cha | 2701 Custer Pkwy, Richardson, TX 75080 |
| His Tea | 221 W Polk St. Ste 105. Richardson, TX 75081 |
| Junbi | 326 West Campbell Road, Richardson, TX 75080 |
| TP Tea | 400 N Greenville Ave #14, Richardson, TX 75081 |
| Tea Daddy | 800 East Arapho Road Suite 105 |

*Figure 5: CSV File #2*

**PostgreSQL/pgAdmin**

The next step involved transferring the collected data from the CSV file into a PostgreSQL database. This entailed creating a table and defining column labels to correspond with the data attributes and then importing the CSV file into the newly created table. Because there are two CSV files— "shops" (primary key: Shop Name) and "menu" (primary key: ID) — this process had to be repeated to generate two distinct tables. After queries are executed to verify the accurate importation of data into the tables, pgAdmin was used to export the PostgreSQL database into a .db file.

**RStudio**

The next step is establishing the integration between the database and RStudio. This requires a few installations within RStudio such as DBI, RSQLite, shiny, tidyr, and dplyr. Additionally, details such as the database name, host, port, user, and password are needed. RSQLite is the package that is used to access the .db file.

```
install.packages("RPostgreSQL")
library(RPostgreSQL)
library(RSQLite)
library(DBI)
postgre_con <- dbConnect(RPostgres::Postgres(),
                         dbname = 'postgres',
                         host = '127.0.0.1',
                         port = 5432,
                         user = 'postgres',
                         password = 'Dance!5678')

# create object to represent all tables in the database
tables <- dbGetQuery(postgre_con, "SELECT tablename FROM pg_catalog.pg_tables WHERE schemaname='public';")

# use rsqlite package to prepare a .db file
sqlite_conn <- dbConnect(RSQLite::SQLite(), dbname = "boba.db")

for (table_name in tables$tablename) {
  message(sprintf("Copying table: %s", table_name))
  # read data from postgresql
  query <- sprintf("SELECT * FROM %s", table_name)
  data <- dbGetQuery(postgre_con, query)

  # write data to sqlite
  dbWriteTable(sqlite_conn, table_name, data, overwrite = TRUE, row.names = FALSE)
}

dbDisconnect(postgre_con)
dbDisconnect(sqlite_conn)
```

*Figure 6: Export Database to .db File*

```
ui <- fluidPage(
  titlePanel("Boba Shops"),

  # create new row in ui for select inputs
  fluidRow(
    column(4,
           selectInput("Shop_Name",
                       "Shop Name:",
                       c("All",
                         unique(as.character(menu.df$Shop_Name))))
    ),
    column(4,
           selectInput("Item_Category",
                       "Item Category:",
                       c("All",
                         unique(as.character(menu.df$Item_Category))))
    ),
    column(4,
           selectInput("Item_Name",
                       "Item Name:",
                       c("All",
                         unique(as.character(menu.df$Item_Name))))
    ),
    column(4,
           selectInput("Item_Size",
                       "Item Size:",
                       c("All",
                         unique(as.character(menu.df$Item_Size))))
    ),
    column(4,
           selectInput("Item_Cost",
                       "Item Cost:",
                       c("All",
                         unique(as.character(sort(menu.df$Item_Cost)))))
    )
  ),
  # create new row for table
  DT::dataTableOutput("table")
)
```

*Figure 7: UI Code*

```r
server <- function(input, output) {
  data <- reactive({
    # connect to database
    conn <- dbConnect(RSQLite::SQLite(), dbname ='boba.db')
    dbListTables(conn)

    menu.df <- dbGetQuery(conn, 'SELECT * FROM "Menu"')
    shops.df <- dbGetQuery(conn, 'SELECT * FROM "Shops"')

    menu.df <- merge(menu.df, shops.df) %>%
      select(Shop_Name, Item_Category, Item_Name,
             Item_Size, Item_Cost, Shop_Address)

    dbDisconnect(conn)
    data.frame(menu.df)
  })

  # filter data based on selections
  output$table <- DT::renderDataTable(DT::datatable({
    data <- data()

    if (input$Shop_Name != "All") {
      data <- filter(data, Shop_Name == input$Shop_Name)
    }
    if (input$Item_Category != "All") {
      data <- filter(data, Item_Category == input$Item_Category)
    }
    if (input$Item_Name != "All") {
      data <- filter(data, Item_Name == input$Item_Name)
    }
    if (input$Item_Size != "All") {
      data <- filter(data, Item_Size == input$Item_Size)
    }
    if (input$Item_Cost != "All") {
      data <- filter(data, Item_Cost == input$Item_Cost)
    }

    data
  }))
```

*Figure 8: Server Code*

**Shiny**

With all connections in place, the development of the application using Shiny can now be started. The Shiny application's code resides in a single "app.R" file, containing code for the UI, responsible for defining the layout and appearance of the application, and the server which contains the instructions necessary for constructing the application's functionality.

## CONCLUSION

**Future Studies**

To enhance the database comprehensively, my primary goal is to gather additional data. In my ongoing commitment to this project, I intend to incorporate all boba shops across Richardson, TX. The reason for not doing so previously was the considerable time required for web scraping and manual data entry. Additionally, as I am relocating to Berkeley, CA for school at the end of the summer, I think it would be useful to me to integrate all boba shops there into the database. This inclusion would facilitate my exploration of new shops while managing my budget.

Another enhancement I seek to implement is to the category field. Currently, I have compiled the categories as assigned by their respective shops. However, there are instances where similar items, such as "ice blended" and "smoothies," are categorized differently due to varying shop terminology. Despite the anticipated time investment, manually reorganizing these items into more cohesive categories will significantly enhance the database's usability.

**Link to Shiny App:** christinalam.shinyapps.io/bobashiny/