Cloud Computing Midterm Write-Ups

Table of Contents

Intro

This file contains **all write-ups** as necessary for the Cloud Computing Midterm assignment of the **Sloppy Cowboys** group (number 4 on Canvas). This put into just one file for the grader's convenience and contains additional, unrequired notes regarding how each requirement was addressed.

This file is organized directly by the requirements given, in the order as they are given on the assignment.

***NOTE:** To access some links, like the engagement pages and data table, you will need to be logged in!*

Requirements

Requirement 1

> Give a **unique team name** to your project team... and **select a lead**.

- **Team name:** Sloppy Cowboys
- **Lead:** Jake Steuver

Requirement 2

> Provide a short writeup on how your project is looking to answer [these two questions]:
>
> 1. **How does customer engagement change over time?**
> 2. **Which demographic factors ... appear to affect customer engagement?**

For each question, we plan to create a dedicated web page with charts and graphs as necessary. Alongside these charts and graphs, we will provide analysis and write-ups to directly answer some part of each question. Given we are not entirely familiar with the data and what areas may or may not yield interesting results, we are keeping our plans loose regarding what factors to look at and which graphs or charts to use, allowing us to explore and flex as necessary.

We then planned which technologies we were going to use. The project will be implemented by hosting a server on Django, a Python-based server framework. Within this, we will use the in-house front-end logic alongside Chart.js to display charts and graphs. This will then of course be deployed to the Azure web service for public access.

Requirement 3

> Launch / configure a web server in Azure... Design an interactive web page to do the following:
>
> - Username
> - Password
> - Email

This is accessible here. You can create an account via the **Sign Up** link at the top, after which you can use the **Login** and **Logout** links as necessary.

Note again that you **need to be logged in** to access some of the remaining requirements.

Requirement 4

> Create a data store / database in Azure of your choise and load Transactions, Households and Products from [a given link]...

We chose to use a **Microsoft SQL** server hosted within Azure, primarily for its easy integration with the Django framework. This was simply loaded as necessary, which is hopefully reflected across the site.

> Create a display page where we can see Sample Data Pull for HSHD_NUM #10, linking the household, transaction, and product tables. Sort by [various fields]...

This was implemented via `django-tables2` and is accessible via the **Data Table** link at the top of the page. Here, HSHD 10 is loaded by default, but any HSHD can be chosen here to replace it. All columns are individually sortable by clicking on them.

Requirement 5

> Create an interactive web page that will allow us to search on DATA PULLs based on Hshd_num. Sort by [various fields]...

We merged this requirement with that of **Requirement 4**, meaning this is also fulfilled by navigating to the **Data Table** link at the top of the page. See Requirement 4 for more details.

Requirement 6

> Web page to answer "How does customer engagement change over time?"

Our result for this can be found at the **Engagement Over Time** link at the top of the page. As specified in the instructions, we simply needed to choose one of the examples and keep the page fairly simple. As such, we decided to do two **line graphs**:

- All purchases over time (separated by month)
- Purchases of a small sample of households over time (separated by month)

This allowed us to look at overall trends as well as a few specific cases, which we analyze directly within the page from the perspective of a data / market analyst for this company.

Requirement 7

> Web page to answer "Which demographic factors appear to affect customer engagement?"

Our result for this can be found at the **Demographic Engagement** link at the top of the page. Again, we were instructed to keep it fairly simple, and our result was again two graphs. This time, they were **bar graphs**:

- Overall purchase amount per household in each commodity category
- Purchase amount per household in each commodity category split based on income range

This again allowed us to look at the overall trends then compare that to further data, this time based on the demographic factor of income range across all households. This is then analyzed within the page, as was done for Requirement 4.

Requirement 8

> Web app, where we can load the most current data sets...

This can be found at the **Upload Data** link at the top of the page. This allows users to upload CSV files to be included in the database. Such data can be viewed by navigating to the **Data Table** page. The columns of the CSV files can be in any order, but they must use the same column names as the given sample data.

Requirement 9

> Use Agile development methodology for development effor and provide a short write-up on:
>
> - What worked
> - What did not
> - What improvements can be made

What Worked

- Django and Chart.js turned out to be great utilities
- Materialize provided clean UI
- Github was a good place for code review and recognition

What Did Not Work

- Attempting to process data manually instead of using the Django ORM
  - Code was originally larger and much slower via loops
  - Django ORM instead puts everything into SQL query to let Azure database figure it out all at once
- Chart.js initial integration
  - Technically functions, but documentation is sparse and the required implementation is complex, two issues we unfortunately can't fix

- Utilizing on-computer environment variables
  - Proved to be too much work for developer
  - Instead used Docker for our project!

Improvements To Be Made

- UI system more robust than Django templates could be used to provide much easier code re-use
- Data processing could be made faster via something like Spark
- We could have more increased testing in our workflow to better ensure code quality
- Django static deployment would be need implemented for a real production software
- Team communication and spread of workflow needs improvement