GrocLog is a web application meant to help users keep track of their grocery spending, enabling them to view data analytics based off what items they spend, for what amounts, at which locations, etc. Using the program, users are able to see their past spending to properly plan for future spending or analyze grocery store prices in their area by relying on their own spending, rather than an economic report that they may not trust.

GrocLog uses MongoDB to store login-data and user-data, and NPMCharts to display the purchasing analytics to the user. Login details are stored in a MongoDB table, with passwords being hashed using BCrypt (initially SHA-256, then changed). Users login by sending a POST request to the server, which will respond with a JWT token for the client. Users that are marked as not being administrators are taken to the itemlog page, where they can see items they’ve previously added while also being able to add more. This page contains a hyperlink-button to allow users to access the analytics page, which shows them charts based off their purchases.

With this project, I learned a lot. Some examples of concepts we used in this project I had not encountered previously are AJAX requests, SQL format, long-term VM use, subdomain/public ip routing, SHA-256 Hashing, NodeJS, npm, and more. All of these concepts were very interesting to learn and getting to use them on a large project in addition to the smaller assignments was really neat.

I spent a lot of time with the project with the user-login handling, and learning how to securely handle logins from the backend without allowing client command-requests to infiltrate our system was challenging and somewhat complicated, but really cool to see come to fruition.

I wish I knew that SQL was going to give us problems long-term, our server kept crashing if left unused for a long time so the system could not be used without being restarted. Once we changed to MongoDB, we no longer had this issue so it would’ve been nice if we had initially created the system to use this, it would’ve saved a lot of time.

Over the summer, I will be attempting to convert this project into React, Angular, or Vue (I haven’t decided yet). I would also like to implement an image-viewer, like ImageGlass or PicView, to read receipts and add items for users automatically.