**Project Summary:**This project uses node-js, Mysql, MQTT, D3, Express Web Server, and various dependencies to get the cpu load averages from the client computer, store them on the server in a mysql database, graph them with D3 and display them using an Express Web Server in the form of a web page. The client will scan a computer and report cpu load averages and timestamps. This data is then sent to a node.js server using MQTT. The server will then take that data and store it into a mysql database and displays it in a web page. An Express Web Server will host a website that will act as a front end displaying the generated graphs in the user’s web browser.

This assignment is based on a similar project from another class, but the original project was much less in-depth.

**Motivations:**

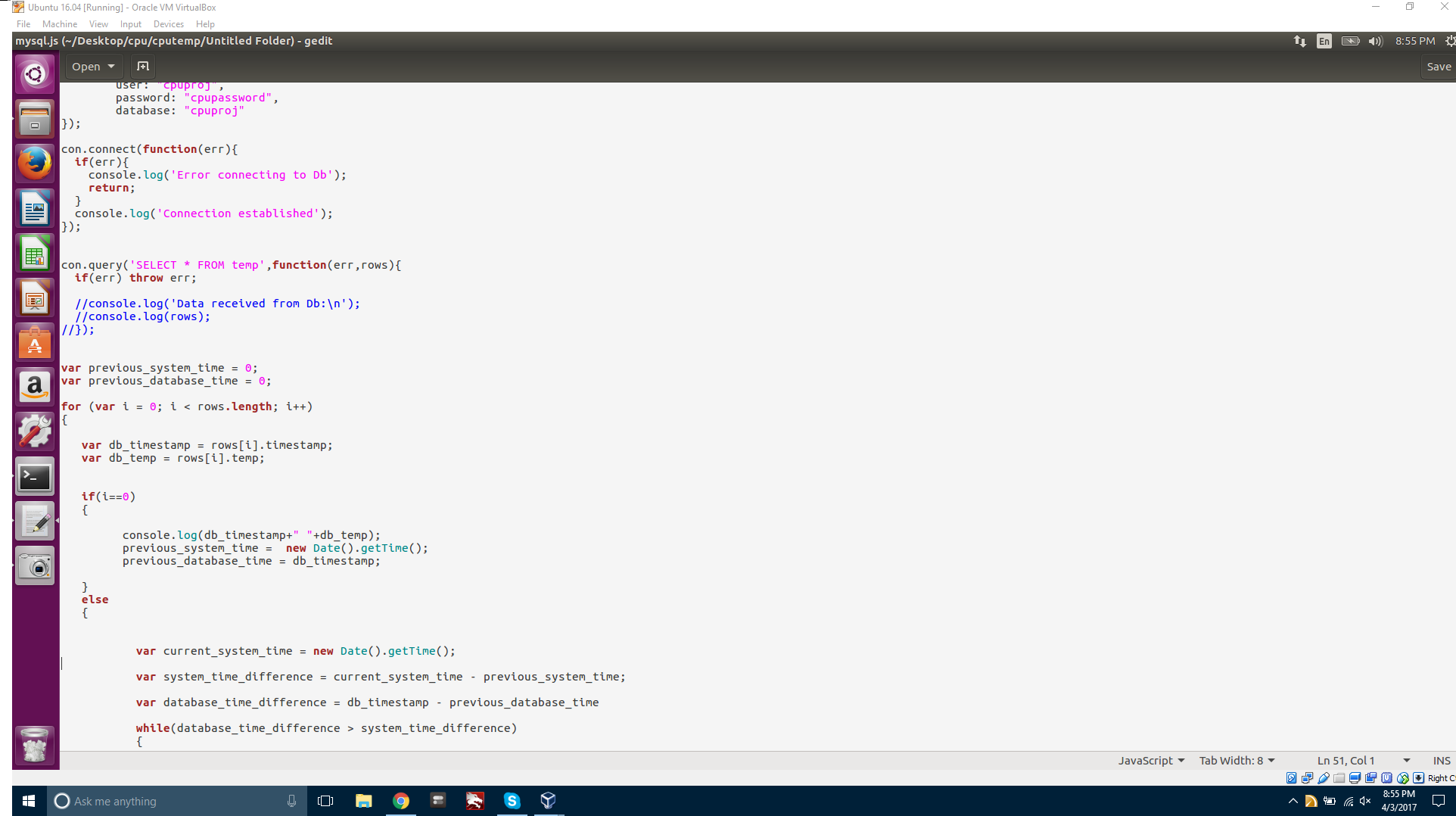
Our primary motivation for this project is to complete all requirements. In order to accomplish this, we originally decided to use an Raspberry Pi for our server, in order to get some experience with using the device. But do to time constraints and conflicts, we decided against this and decided to host our server on Ubuntu running a laptop.

**High level design:**



A client accesses the server with the tool protocol which contains the system info such as CPU load, temp, ect. and connects to the server. The server then takes the data from the client, stores it into a SQL database, and makes a chart with the statics from the clients.

**Low level design:**

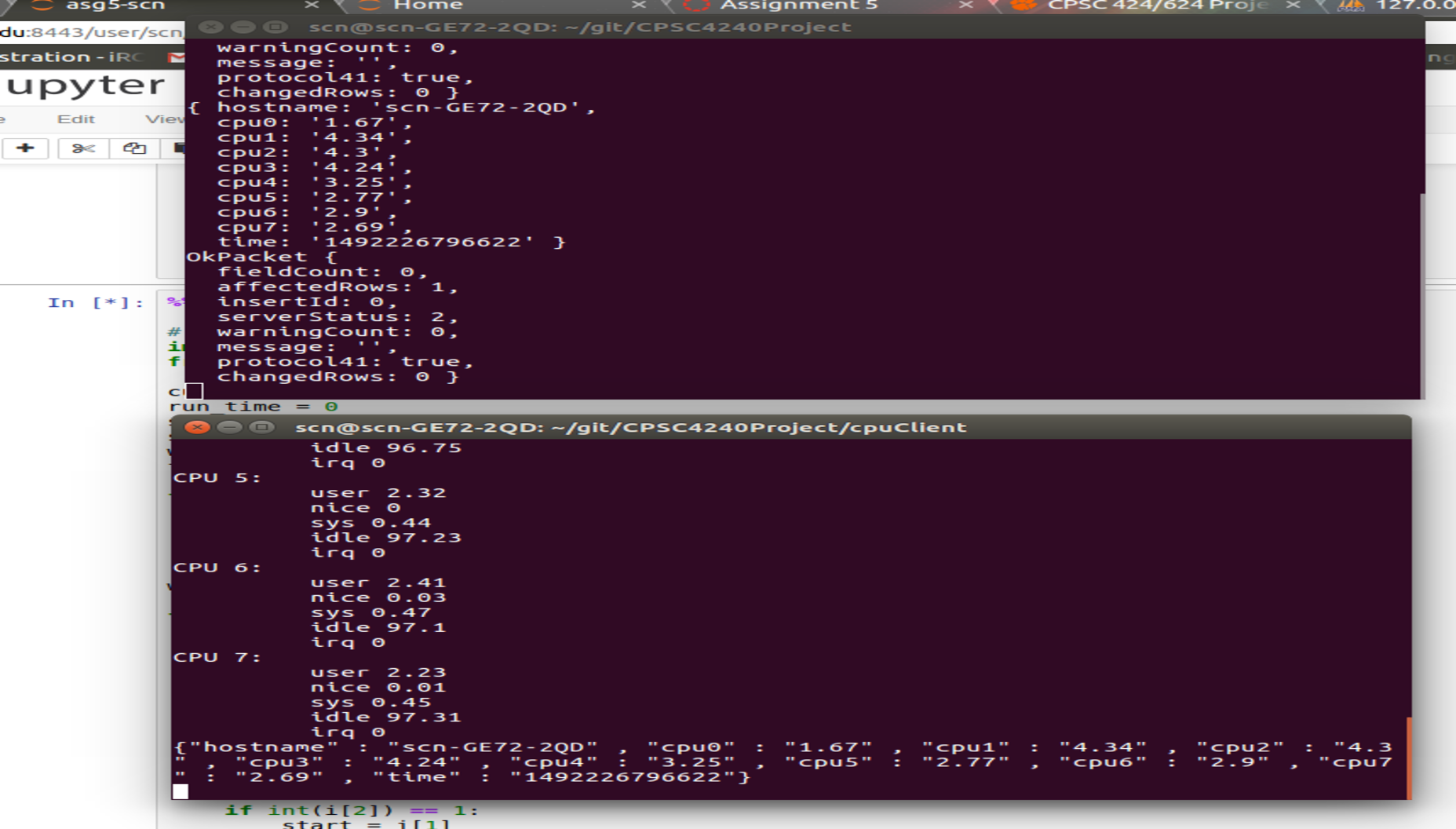
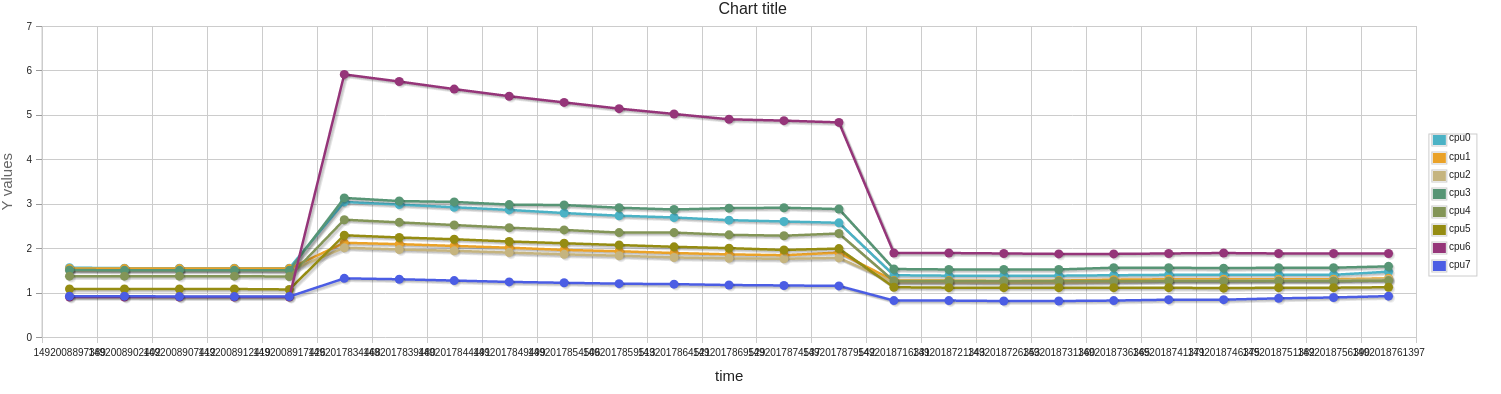


The client/”tool” sends cpu load averages to server using MQTT. The server receives data from the client and stores the data into a mySQL database. The Apache Web Server will host a website that will act as the front end and display the graphs being generated by the server. MySQL will store the data and PHPmyadmin is a user interface for the mySQL database.

**Validation:**

For the cpu load averages the only way to validate right now is to compare the server console log and the sql database. Both the Server and the Client are validated and the MySQL database is storing and querying as it should. Unfortunately we have not been able to get our php code working with our web page so we have not been able to successfully generate a graph. We hope to have the graph in phase 2. That being said we did generate a line graph in phpmyadmin to demonstrate it.

**Mysql Demo**

**Server and Client Demo**

**Security:**

**Mysql:** We hardened the mysql server using mysql\_secure\_installation. This includes setting a strong randomly generated password for root, removing anonymous users, Disallowing root login remotely, removing test databases, and installing the validate\_password plug-in which allows you to change password requirements. Then we set them to high.

**Client and Server:** The biggest thing here is to have the client encrypt data and then have the server decrypt it and store it on the local mysql database. However given the nature of our client and server we can encrypt to prevent from man in the middle attacks, but our code is exposed since it is javascript which means there isn’t much we can do to hide our encryption methods to our users. I would like to do this using ssl/tls but it will be more practical to use the crypto module and make our own. MQTT doesn’t have encryption built in so the data needs to be encrypted before it leaves the client and decrypted on arrival at the server.

**Server Expressjs Webserver:** We Installed the helmet module which provides protection against well known web vulnerabilities by setting the HTTP headers appropriately. Helmet does this by installing nine smaller middleware functions that set security-related HTTP headers. This includes csp, hidePoweredBy, hpkp, hsts, ieNoOpen, noCache, noSniff, frameguard, xssFilter.

**Dependencies:** Dependencies give nodejs a lot of power however they can also be a source of insecurity so I installed two vulnerability scanners nsp check and snyk. Nsp check will only check for vulnerabilities and give suggestions. Snyk will scan for vulnerabilities and then patch or update for you.