# CPU DB Data Visualization
# Senior Project Report


Marek Moreno (mmoren14@calpoly.edu)

Ruchita Patel (rpatel31@calpoly.edu)

16 June 2017

# Introduction

### Project Overview/Executive Summary

Given the CPU database from Stanford, we wanted to create something that portrayed the data in a more visually pleasing way. The CPU database website wanted a web page that would allow users to create graphs based on the processor data from the database. The web page would allow users to select different data from the database and create the graphs they wanted to gain insight into the decades of processor data.

### Clients

Our primary client was the team who maintains the CPU database for Stanford, primarily Prof. Andrew Danowitz (who was also our senior project advisor). Their mission is to provide researchers and hobbyists with Stanford data on commercial microprocessor characteristics. The site offers interested parties a repository for this processor data, with the goal of expanding the utility of this data indefinitely. This senior project topic was offered to add features to the site to support this goal.

### Stakeholders

The aforementioned researchers and hobbyists are the primary stakeholders in this project, along with CPU DB itself. By allowing users to visualize processor data in various ways, new perspectives on processor evolution may emerge and contribute to academic discussions or new developments in this field. Removed from these immediate stakeholders, and person who might benefit from advancements in

microprocessor technology would also have a vested interest in the success of this project.

**Framed Insights and Opportunities**

The site already had a Visualize tab that several specific examples of visualized processor data. There is a link to a web page called Interactive CPUDB, which could have resembled our senior project, but unfortunately it links to a page not available for a normal user. These examples were limited in scope, allowing us to build off of the material to design a visualization page that was more open-ended and versatile.

**Project Goals and Objectives**

Below is a list of our project goals and objectives. The goals are the methods and procedures used throughout the project to accomplish the objectives.

Project Goals

- Create the highest quality project possible
- Create a simple UI for the web page
- Create a project that would update to reflect changes in the database

Project Objectives

- Achieve the minimum needs of the client
- Learn the languages, interfaces, and environments of our project
- Distribute work equitably and effectively
- Successfully interface our web page with the client database

## Project Outcomes and Deliverables

The main result of the project is a web page of our data plot. Our team provided this code to our client. We also provided this documentation to detail development and explain the functionality of the project. Our project will hopefully provide a stepping stone to future developments in processor data visualization.

## Relevant Designs

We looked at how other sites tackled data visualization and were able to find some previous examples to base our development off of. We developed our design off of examples from bl.ocks.org in particular, as their examples were clear and provided excellent documentation.

## Resources Needed

We obtained the processor database from the client along with access to the github repository that housed the product code. All of our development was done in the terminal with open source software; no other resources were needed.

# Background

Our project has changed dramatically throughout development, both in terms of scale and focus. From the beginning, we planned to create a data visualization tool that used data from the CPU DB. Originally, we thought we would need to interact with the database a lot more and set up the database ourselves; we spent a large portion of our first quarter focusing on this area of the project. However, we eventually found out that the database was already in place and that our work would exist entirely in the front end instead. From then, we shifted our focus to the front end.

### Competitor Information

There are no competitors for our project. Our project is open source and so are all of the resources we used in our development. Any overlap between the work we were doing and work other developers are doing is entirely beneficial to our efforts.

### Similar Systems

As stated earlier, the closest similar systems are the graphs included on the Visualize tab of the CPU DB site. Other online sites also include information about data visualization, sites that we drew information from when designing our systems.

### Standard Testing and Validation Procedures

There are no preset validation procedures for data visualization, other than comparing the data points that appear on the graph with the ones from the database. This was not

a major concern for our team during development. More important to us was testing all of the inputs of our web page to make sure that no user interaction caused the site to crash or break down in any way.

**Relevant Literature**

We read documentation about databases, HTML, HAML, Ruby, and data plots. All of this documentation was helpful during development.

# Engineering Specifications

The greatest specification is to create a user-friendly web page that uses open-source documents to create a long-lasting product.

### Client Requirements

We learned from the very beginning that our client wanted a data visualization tool that would allow the user to select the data that they are interested in and display that data. The web page had to use free resources to match the open-source nature of CPU DB. The product had to have an intuitive design to encourage users to explore and analyze the processor data.

### Personas

Our clients are familiar with technical subjects and fields of study. Therefore, our project could reflect the skill of those main individuals and expect a certain level of technical skill from the user. In a certain sense, skill will be required to create plots with academic significance. We wanted the design of our project to be understood and usable by all persons, however.

# Design Development

Our final product included a webpage with scatter plots that were zoomable and easier to view the data than the original static scatter plots.

### Web Page Schematic

Our ideal web page design had a few main features that we wanted to add to our project. The page would feature two drop-down menus, one per axis. The user would select the processor data that they were interested in from each menu and the resulting points would appear on the graph to the right. The intended design is shown in Figure 1 below.

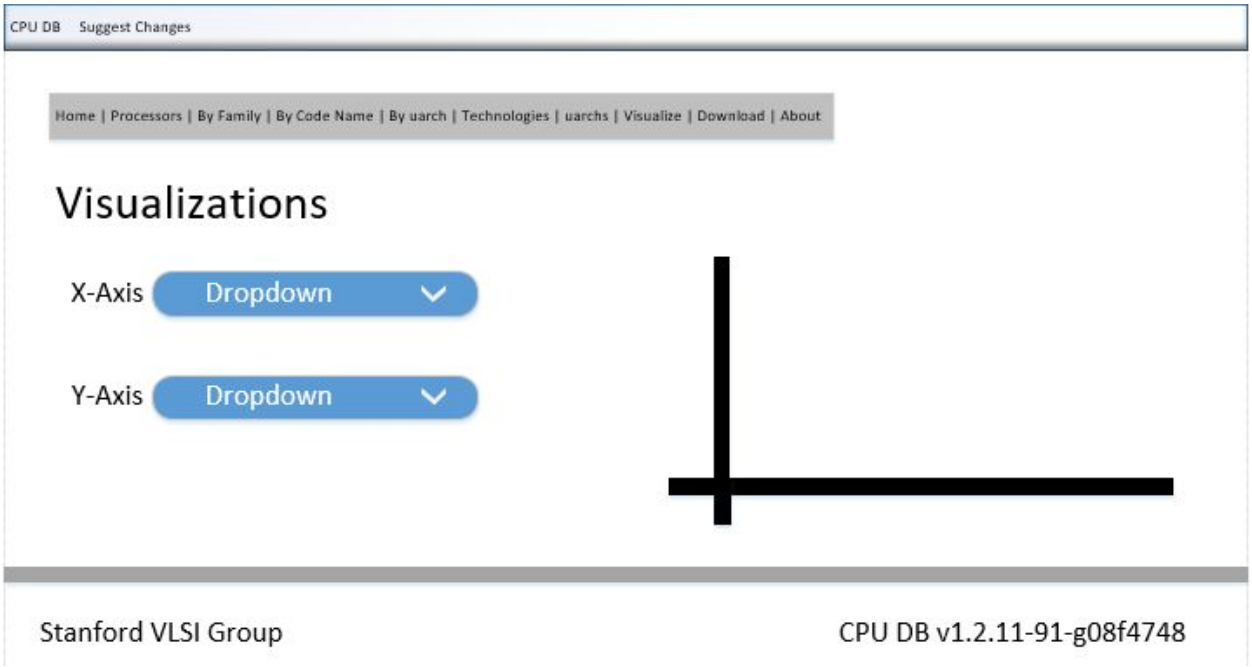


Figure 1. Web Page Schematic

## System-Level Schematics

Our application features data transfer between four major components: the user, the controller, the model, and the view. The interaction between these components is detailed in Figure 2 below.

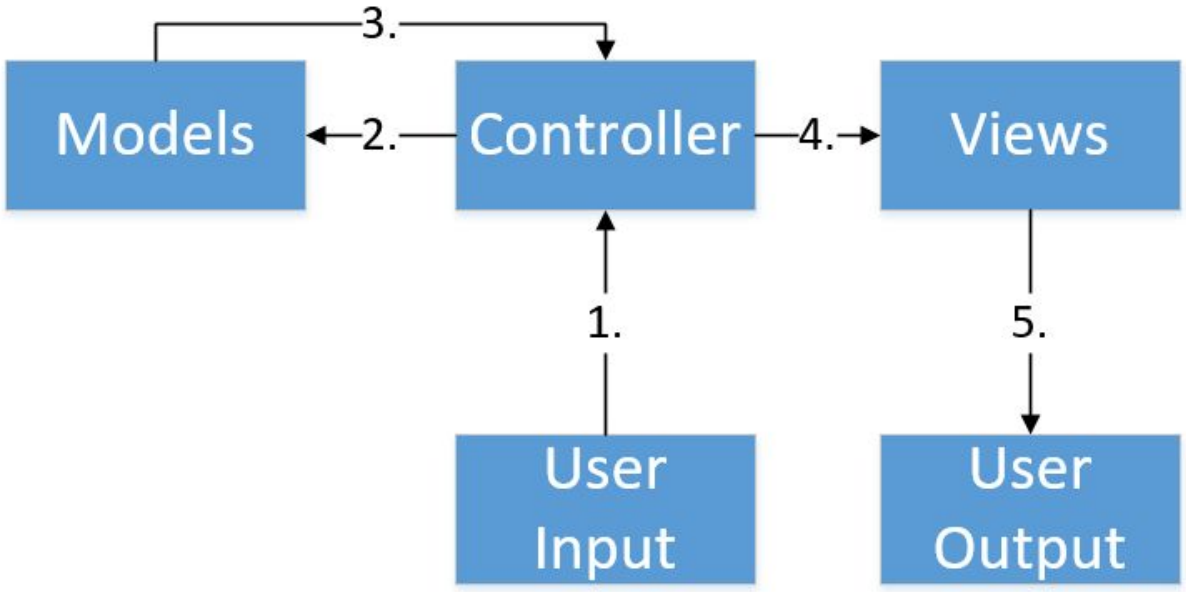


Figure 2. System-Level Schematic

1. The controller takes input from the user.

2. The controller talks to the models to obtain data from the database based on the user request.

3. The models give results back to the user so

4. The data can be passed to the views where, ultimately,

5. Data is displayed for the user

## Project Specification Satisfaction

Our concept satisfies a large proportion of the project requirements. As per the request of our client, we did create a interactive visualizations. We worked with our client throughout the design process and tailored our work to meet their needs. We created design templates and layouts so that our client could see the how the new web application would look. All the designs and graphs were labelled so that our client as well as users are able to understand. Unfortunately, we were unable to create a fully customizable data plot. While we were not able to accomplish our original goals, hopefully with a little more work, our client may be able to use a fully customizable plot.

## Justification of Selected Concept

Our conceptual design comes mostly from the original idea of the client, with some influence of other websites that have used the d3 javascript library. We wanted to create a layout that included all the small details from our client while keeping it as simple and easy to use as possible. Our design includes the data from original plots that were given to us on the CPU database website plotted on zoomable scatter plots. Each plot was labelled, so that all the information of each plot is given to the user.

## Safety Considerations

We designed our product so that there were no safety hazards for our users.

## Implementation and Testing

The implementation of our project involved adding plotting files to the provided CPU DB code base. In app/views/plotting, we created our index.haml to govern how our web page would look when the user clicked on our Plotting menu tab. We used code and libraries from the website bl.ocks.org to plot and display data. We tried working with various online data plotting resources but ultimately found that the tutorials provided on bl.ocks.org were the best fit for our display needs as well as being the most clearly documented.

We have not developed formal testing programs for our application. When building and composing up the docker image, if there is any issue with the code it will not compile. As a result, in our experience if the docker image builds it works without error.

# Final Design Concept

Our final design concept unfortunately falls short of our goals for this project. For various reasons which I will explain below, our development was much more arduous and slow-paced than had anticipated at the beginning of our project. As a result, our final product does not resemble our proposed concept for this project. Our final project implements two data sets from our repository of Excel files in one index.haml file, and allows the user to change data sets with the push of a button in one index.haml file. Unfortunately we were able to integrate the two features, but we included both index.haml files in our github project repository.

One of the most exciting aspects of choosing this project was neither team member had ever worked on a project like this before. While the task seemed daunting at first, it was a wonderful opportunity to dive into a new subject and learn so many different concepts associated with the project. Everything, from the development environment, to the languages, to the program structure, was a new concept for us. And while this remains the most fun and exciting aspect of our project, it has lead to some missteps during development.

As mentioned earlier in the document, we spent a lot of time our first quarter learning about the back end systems of our project. We spent a lot of time familiarizing ourselves with the Docker process and getting that set up on our respective computers. Then we spent a few weeks studying Ruby, JavaScript, and HTML from Codecademy courses online. However, after meeting with our senior project advisor and discussing the goals and development plans of our project, we realized that we had misplaced our focus on the backend aspects of the project.

From that point on, we abandoned our work up until that point and began developing our UI for the front end.

For many weeks we read online documentation about data plotting, trying various code samples online and learning how plot information is spread throughout the code base. Once we discovered a plotting system that worked well for our goals, we spent our efforts trying to incorporate our data sets with those plots. But the biggest hinderance to our development is the time it takes to build and compose-up a Docker project. Ruchita averaged a Docker project build time of ten minutes throughout development; Marek averaged twenty minutes. Eventually, this was beneficial in sharpening our design skills as there were very real consequences for writing bad code. However, it was difficult and time-consuming to test iteratively throughout development. Our biggest regret or annoyance during the project was not being able to avoid these large compile times.

However, we do not want to give the impression that we are just making excuses and that we are not responsible for the state of our project. While our final design does not meet our original requirements, we have both learned a lot about the subject matter and have enjoyed working on this problem for the last two quarters. We enjoyed researching the back end topics early in development, for while we did not end up using that knowledge directly for our final project, it contributed to the depth of our knowledge. Even though it took us a long time to find a plotting that both worked for our goals and was implementable, we were able to study different methods for solving the same problem and how to pull useful information from disparate sources to solve the problem facing at hand.

# Management Plan

## Team Mission and Team Objectives

Our team mission was to improve the website visualizations so that our users and client can see the trends in the processors. We put together this mission statement by listing out some brief team goals and objectives. Both our team goals and objectives together create our overall mission statement.

Team Goals

- Create a user-friendly web application that supports client requests
- Pursue the learning opportunity given to us by taking on this senior project
- Create the highest quality product for our client as we can

Team Objectives

- Communicate well with our team members
- Resolve any conflicts internally
- Distribute work effectively and equitably
- Obtain the skills needed without much client guidance

## Team Membership and Roles

Our project was all software based. We decided to share the work evenly since there was only two of us. We used version control such as git to better communicate our findings and keep up to date with each other. Since this project is all software based, we both took on roles as the software designer, architect, and product verification. Within these we both contributed to the high-level design of the product. This includes the data structures, functions, and features of the

web application. Acting as a software designer, we both contributed to the creation of the web application. We split this part into front end and back end. We were both able to dabble in both sides to help each other out and figure out a solution for our client. We tested and debugged the code to make sure it was working as expected. Testing included error checking as well as making sure the backend and frontend worked well together. We used an agile methodology during development. We designed 2-3 week long sprints to achieve smaller more specific goals for the project. We had meetings twice a week to work on the project and to evaluate the current status and adjust our goals as needed.