|  |
| --- |
| VISUALIZATION OF THE COMPOSITION OF  CLOUD COMPUTING SERVICES |
| A Thesis  Presented to  The Faculty of the Department of Computer Science  California State University, Los Angeles |
| In Partial Fulfillment  of the Requirements for the Degree  Master of Science  in  Computer Science |
| By  Davis Louie  December 2018 |

© 2018

Davis Louie

ALL RIGHTS RESERVED

The thesis of Davis Louie is approved.

Jiang Guo, Committee Chair

Russell Abbott

Raj Pamula, Department Chair

California State University, Los Angeles

December 2018

# ABSTRACT

Visualization of the Composition of Cloud Computing Services

By

Davis Louie

Visualization of the Composition of Cloud Computing Services also known as VTCCS is a web-based application that provides a way for users to create composite web services out of existing web services. In modern technology, web services are quite common, and many companies use cloud computing web services which are services provided remotely. Those web services are often used together with many others that it would require some programming and effort to make them work. VTCCCS simplifies the process and allows users not only to generate composite web services but also to monitor the performance of the composite web service. VTCCCS lets users benchmark each individual web service and the overall composition.

# ACKNOWLEDGMENTS

I would like to thank my advisor Dr. Jiang Guo for his assistance and guidance on the subject matter. His help has been very valuable for the direction of this project.

Great thanks to the former NASA Direct-Stem team who are: Bill Pham, Andrew McLees, and Patrick Flinner. If it weren’t for them, this project wouldn’t be created and ready.

I would also like to thank the senior design that graduated last spring who aided and assisted me in creating this web application: Zolangi Ramirez, Gonzalo Serrano, Hongsuk Choi, Sudip Baral, and Johnson Truong.

Additional thanks to Dr. Jiang Guo’s current NASA Direct-Stem team who consist of Isaac Cano, David-senpai-San, Ricky Vargas, Adekola Togunloju, and more. I would also acknowledge the assistance of the past Direct Stem graduate students, Shay Nguyen and Adam Berman. Their advice was very helpful.

TABLE OF CONTENTS

Abstract iv

Acknowledgments v

List of Tables viii

List of Figures ix

List of Terms x

Chapter

1. Introduction 1

2. VTCCCS Technical Background and Framework 3

2.1 VTCCCS as a Web Application 3

2.2 Overview of Angular 4

2.2.1 Angular Modules x

2.2.2 Angular Components x

2.2.3 Angular Services x

2.3 HTML Used in Angular x

2.4 CSS Used in Angular x

Level 2 Heading x

Level 2 Heading x

Level 1 Heading x

3. Title of Chapter x

Level 1 Heading x

Level 1 Heading x

4. Title of Chapter x

References x

Appendices

A. Title of Appendix x

B. Title of Appendix x

C. Title of Appendix x

# LIST OF TABLES

Table

1. Title of Table x

2. Title of Table x

3. Title of Table x

4. Title of Table x

5. Title of Table x

6. Title of Table x

7. Title of Table x

8. Title of Table x

9. Title of Table x

10. Title of Table x

11. Title of Table x

12. Title of Table x

13. Title of Table x

# LIST OF FIGURES

Figure

1. Title of Figure x

2. Title of Figure x

3. Title of Figure x

4. Title of Figure x

5. Title of Figure x

6. Title of Figure x

7. Title of Figure x

8. Title of Figure x

9. Title of Figure x

10. Title of Figure x

11. Title of Figure x

12. Title of Figure x

13. Title of Figure x

LIST OF TERMS

VTCCCS Visualization of the Composition of Cloud Computing Services

HTML Hypertext Markup Language

CSS Cascading Style Sheets

CLI Command Line Interface

DI Dependency Injection

IDE Integrated Development Environment

SQL Structured Query Language

SPA Single-Page Application

NPM Node Package Manager

OS Operating System

# CHAPTER 1

# Introduction

VTCCS which is an acronym for Visualization of the Composition of Cloud Computing Services, is a web application created to construct complex composition of web services that would normally be hard to visualize. Our web application creates a nice model for the user to create and link web services to form the composition of their choice. Programmers come across these challenges when they work with multiple web services and each service requires an output from another web service to use as an input. It is a tedious process in which these services need to run in a certain order and in parallel if possible. An example of this sequence is akin to cooking. The cook needs a certain amount of ingredients in and they all need to be prepared at different times. Once the cook gathers everything up and put them together, the cook retrieves the result which is the finished dish which would be the web service composition.

Another feature our application has is the ability to monitor these web services. Our web application monitors the run time of each cloud computing web service and records them. Our users will be able to determine which web services have the fastest run time speeds. Users might ask why is run time important. Take in account that run time makes a huge difference in the overall composition of the web service. If people have one web service that takes up too much time completing its task, that means another web service that needs the output of the slow web service must wait for the full run time for its service to even start. This is what we call a bottle neck where one web service holds up the whole composition from being completed. Fortunately, users can identify the bottle neck with our web application. Without that feature, users would have to run each web service one by one to see where the bottle neck lies and with higher order compositions, that would be very time consuming.

An augment to the previous monitoring feature is the ability to monitor the web service in real time. It is very common for a web service to be bogged down by many requests from multiple users across the globe. Users need to know when the web service will receive heavy traffic. VTCCCS can monitor and keep track of a web service’s run time speed throughout the day by constantly sending a request. That way, users will be able to determine which hours of the day are best for using a web service and can also be used as a comparison between similar web services that do the same things.

CHAPTER 2

VTCCCS Technical Background and Framework

Since VTCCCS requires real-time execution of web services and the ability to create composite web services, it is important to choose the right technologies and frameworks that allows us to accomplish our objective and give users a friendly experience. With that in mind, VTCCCS is implemented to be a web application.

The VTCCCS web application uses HTML, CSS, Angular 4, Node.js, Angular Command Line Interface also known and Angular CLI, D3.js, and Chart.js for the front end. The back end uses Java, Maven, and MySQL for the back end. Angular uses typescript which is a superset of JavaScript. The project was built by using Maven for the back end and a combination of the Angular CLI with Node.js for the front end. The Integrated Development Environments or IDEs used were Intellij, Visual Studio Code, and the MySQL workbench for the SQL scripts.

**2.1 VTCCCS as a Web Application**

A web application or web app is a client-server computer program which the client runs in a web browser [1]. VTCCCS requires using a modern browser such as Google Chrome, Firefox Mozilla, Microsoft Edge, or Apple Safari to be able to access and execute this application.

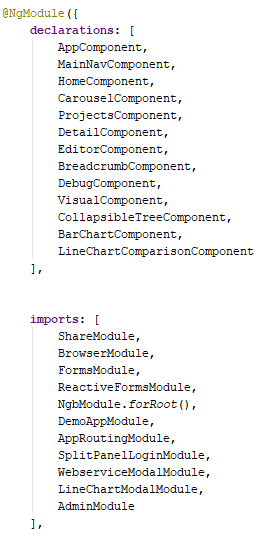
The reason why VTCCCS is a web application is because of accessibility. Users only need a web browser either on their smart phones or their computers to be able to use it which means the barriers of entry is low. Of course, users need internet access in order to utilize a web browser, but that is a trivially thing in this modern age. There are no other requirements.

**2.2 Overview of Angular**

Angular is a TypeScript-based open-source front-end web application platform [2]. Angular is cross platform and has great speed and performance. Angular uses a hierarchy of components and subcomponents for its architecture. The reason why angular is so fast is because it is a single-page application or SPA. Instead of rewriting the page to load a new one, SPAs dynamically change or swap components which requires less processing power and without reloading any redundant data like the header or footer. Angular runs only on the client side so it is responsible for loading the HTML, CSS, and typescript.

**2.2.1 Angular Modules**

Angular web applications are modular, and they are built from individual modules called NgModules. NgModules are like containers that hold a chunk of code that has completes a job or is closely related to the job. They have components, service providers, and other necessary code such as models. Every Angular application has a root module and that root may have child modules that stem from it. As you can see from Figure 1 below that this root module uses 13 components and it has imported 11 modules. The reason for the huge imports is because the components need the classes in order to function.



*Figure 1.* A Portion of Code from the Root Module Called App.module.ts.

**2.2.2 Angular Components**

The way Angular creates a component is first creating the metadata is using the @Component decorator in the typescript file. An example of this can be seen below.

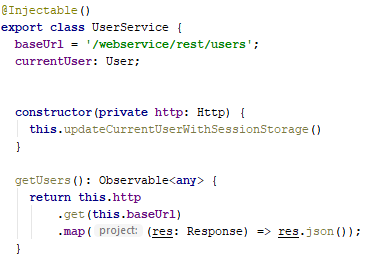


*Figure 2*. Example Code for Component Metadata from Main-nav.component.ts.

The angular metadata tells Angular the location of the building blocks of the component are in the file structure. Figure 2 shows that the blocks are referenced using relative paths and that the HTML and CSS files are in the same directory by templateUrl for HTML and styleUrls for CSS. Selector is the CSS selector that identifies this directive in a template and triggers instantiation of the directive [3]. The component metadata also can register a service for the component to use. A service is generally the way components pass data from one to the other.

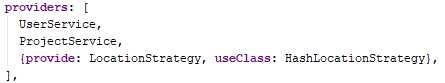
**2.2.3 Angular Services**

Angular Services cover a broad category of what it provides for the web application. It functions as a helper that completes a specific task such as fetching data from the server or validating the input forms. Components can delegate the tasks to the services. Some people may ask why should we even use services? It is mostly to reduce redundant code. If multiple components need to retrieve information from the same database, it is more efficient to create a service rather than type out the same code repeatedly for each component. An example of a service can be seen in Figure 3.



*Figure 3.* Small Snippet of the User.service.ts.

The @Injectable( ) indicates DI or dependency injection. DI is in the Angular framework and it is used everywhere. It basically allows you to inject the service to any of the components that need that service. Also, Angular needs to register a provider for that service in the module. To look at how to register providers, the figure below demonstrates this.



*Figure 4.* Provider Snippet of Code from App.module.ts.

In Figure 4, two service providers have been registered, UserService and ProjectService. Whenever a component needs the UserService, the provider will tell the injector how to create the dependency. An injector creates dependencies on services, so the component can use them whenever they need to. A component may need both the UserService and the ProjectService so the provide tells the injector how to create them and then the injector adds both UserService dependency and the ProjectService dependency. Now the component can use the services it needs.

**2.3 HTML Used in Angular**

HTML is the standard markup language for creating web pages and web applications [4]. HTML uses elements which are building blocks for web pages such as head, body, p for paragraph, img for image, and many more. HTML is the source or text of the web page. Angular uses HTML to create their component or subcomponent. If the component needs text rendered, then HTML will contain all the text and will be solely responsible for that portion of the web page. Angular stores the text in an external HTML file in which angular can access it when use that component. A sample of HTML code is shown in Figure 2.

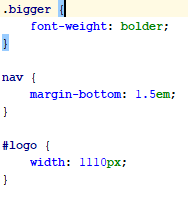


*Figure 5*. Example HTML code from Bar-chart.component.html

In the figure, the table element or tag creates a table for the website. Since there are 2 tr tags which stands for table row, there will be 2 rows. Counting the td which is table data or th which is table header, there will be 3 columns in this table. Within these tags or elements are text that will be placed in the table depending on their position. The first header will have the text Node. After that, it will be Color and so on.

**2.4 CSS Used in Angular**

CSS is the styling of the web page. That means CSS is responsible for the font, text size, alignment, color, etc. CSS is a part of the whole angular component and like HTML, the CSS is kept in an external stylesheet. CSS is like a set of instructions while the text is being rendered telling the computer to make that text color red or make that font to Courier. CSS may not seem important, but it has its benefits. Styling the web page in a more presentation manner would make the web page easier to read. For example, if the web page is single spaced and 50 pages long with size 9 font, that would be too difficult to read. A Simple example can be seen below.



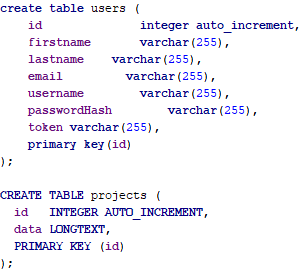
*Figure 6*. CSS Code Example from main-nav.component.html.

From the figure, you can see the CSS selectors which selects the portion of the HTML text to be stylized. The first selector is .bigger, which is a class selector. Basically, it will affect all the HTML tags that has class=”bigger” in them and will make the text bolder. The second selector is directly selecting the HTML element which is the nav element. It is making the bottom margin of the nav 1.5 em. The em unit is the height of the font in nominal points or inches [5]. In simple terms, it will add the bottom margin relative to the font size. The last selector is like the first one. It is an id selector. If the HTML element has id=”logo”, then it will have a width of 1110px. The px is the shorthand for the word pixel.

**2.5 Jersey Framework with RESTful API**

**2.6 MySQL**

MySQL is a relational database management system that uses SQL as the language. Using MySQL may be a daunting task at first, but it is not too difficult. First, a database needs to be created. In order to create a database, the command create database database\_name should be used. After the database is created, the command use database database\_name has to send right after in order to select that database to use it.



*Figure 7.* Sample Code of SQL from Schema.sql.

From the sample code, there are 2 tables being created. The first one is the users table with attributes of id, firstname, lastname, email, username, passwordHash, and token. Generally, the query follows the same command to create a table. The query starts with create table and then the table’s name. Parenthesis is added and then the list of attributes along with the datatype is added and separated with commas. Tables are essential for storing data in MySQL. As the figure shows, most of the attributes of the user name is varchar. Varchar means variable character which means a character length of many different sizes. The number 255 is the maximum number of characters that attribute can hold. This is necessary to control how much data can be stored in the database.

CHAPTER 3

VTCCCS Design and Implementation

CHAPTER 4

Installation Guide

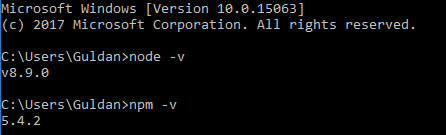
This portion will cover how to get the project running on different platforms. This is mainly to help those who want to use the project and see what it is capable of. This installation guide will be based on using the Windows platform. If you are using a different OS, research how to install these tools onto your machine.

**4.1 Node.js and NPM**

The first thing the project needs is Node.js. Node.js is a platform for JavaScript. It is built on Chrome’s V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model to remain lightweight and efficient. Node.js is needed to use Angular CLI which is one of our main tools for this project.

Installing Node.js also automatically installs NPM. NPM stands for Node Package Manager. NPM lets users install and manage their project’s dependencies. All our project’s front-end dependencies are listed inside our package.json file and running npminstallwill immediately have all the dependencies installed.

1. Download the Windows installer from the Node.js website.
2. Run the installer.
3. Follow the directions of the installer. Choose the default values for the installer if you are not sure of what options to select.
4. Test if Node is properly installed by running the command node -v in the Windows command prompt.
5. Test if NPM is also properly installed by running the command npm -v in the Windows command prompt.



*Figure .* Sample Output of a Successful Installation of Node.js and NPM

**4.2 Angular CLI**

Angular CLI means Angular Command Line Interface. Angular CLI is a tool to initialize, develop, scaffold and maintain Angular applications. Angular is a TypeScript-based open-source front-end web application framework.

1. Open a command prompt and run the command npm install -g @angular/cli.
2. Check if Angular CLI is properly installed by running the command ng –version. The prefix ng stands for Angular and it is how start using the Angular CLI.



*Figure .* Output of Checking if the Angular CLI is installed on.

**4.3 Git**

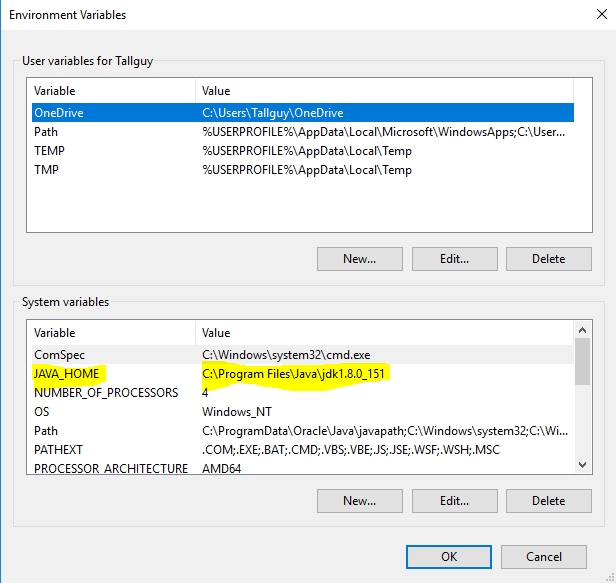
Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. GitHub is where developers store their projects and network. The project is stored on GitHub for easy access.

1. Download the Windows installer from the git website.
2. Run the installer.
3. Follow the directions of the installer. Choose the default values of the installer if you are unsure of what options to select.
4. Open a command prompt or use git bash if you chose to add it in during the installation process.
5. Run two commands to configure your username and email for git: git config -global user.name “John Doe” and git config -global user.name johndoe@gmail.com.
6. You need to generate a new SSH key for authentication so follow the instructions in <https://help.github.com/articles/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent/>.

**4.4 Java Development Kit**

Java Development Kit or JDK is a software development environment used for developing Java applications. This project requires at least JDK 1.7.

1. Download the JDK installer for Windows from the Oracle website.
2. Run the installer.
3. Add JAVA\_HOME to the system environment variables in the control panel.



*Figure .* Modifying the Environment Variables in Windows 10.

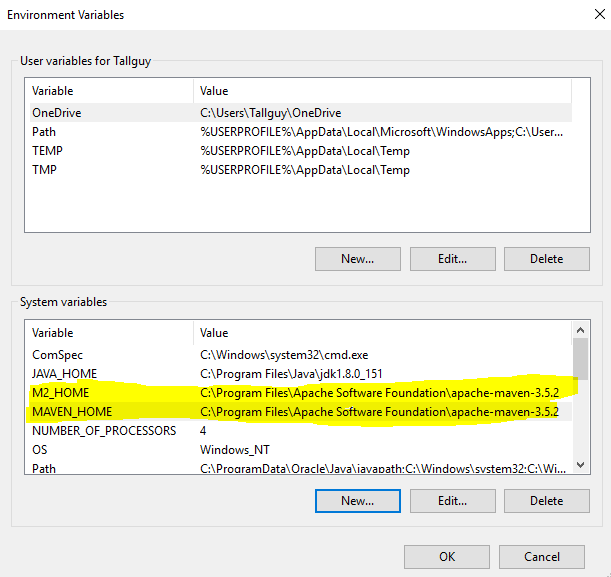
**4.5 Installing MySQL**

1. Download the Windows installer for MySQL community edition.
2. Go through the installer to get MySQL installed. All the default options are fine, but you only need the server installed.
3. Either start MySQL workbench or access MySQL through the MySQL command line so you can run the script located at CSULA-DIRECTSTEM-Webservices\java-server\sql.

**4.6 Apache Maven**

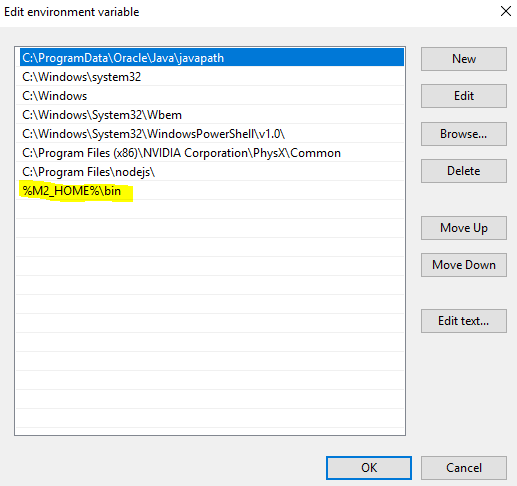
Apache Maven is a software project management and comprehension tool. Maven is needed to build the REST api in order to access the database.

1. Download the apache-maven-3.5.2-bin.zip file from the Apache Maven website.
2. Extract the zip file to your desired location.
3. Add M2\_HOME and MAVEN\_HOME to the system variables in the control panel.



*Figure .* Adding the Environment Variables for Maven.

1. Update the path variable to include %M2\_HOME%\bin so you can run the maven command.



*Figure .* Showing the Path Variable for Maven Directory.

1. Finally, to verify if maven is installed, run the command mvn -version in the command prompt.



*Figure .* Displaying the Sample Output of Mvn -version that is Installed.

**4.7 Project Deployment**

# REFERENCES

[1] “Web application.” *Wikipedia*, <https://en.wikipedia.org/wiki/Web_application>.

[2] “Angular (application platform).” *Wikipedia*, <https://en.wikipedia.org/wiki/Angular_(application_platform)>.

[3] “Component.” *Angular*, <https://angular.io/api/core/Component.>

[4] “HTML.” *Wikipedia*, <https://en.wikipedia.org/wiki/HTML>.

[5] “Em (typography).” *Wikipedia,* <https://en.wikipedia.org/wiki/Em_(typography)>.

# APPENDIX

# Title of Appendix

The appendix is optional. Delete this page if you do not need it.