**Software Architecture Specification**

March 23, 2018

Team 2: John Bost, Julian Casey, Jack Clary, Matthew Days, Eddie Garcia-Alamilla

1. System and Architectural Context
   1. Rationale

For our system, we will be opting to focus on a component-connector view, specifically consider a client-server organization. In choosing the main view with which to consider our system, we promptly eliminated the possibility of an allocation view, as our system relies primarily on software rather than hardware. Additionally, we did not believe that a module view was necessary for our system, since we found that there would be a fair bit of overlap between component-connector and module views, and that such a code-centric view might not capture the interaction between our components properly. The component-connector view that we have taken allows us to touch on the major pieces upon all the major pieces of our project, such as the clients (both customers and employees), the database, and the different servers (including the cache and the authentication server), as well as the way in which these pieces communicate. Our system functions via interaction between the client (either customer or employee) and the server connected to our database of menu options and orders, which involves client making a request on the server and waiting for the server’s response. With the employee-server interaction, we will also implement an authentication server that will require employees to login before they have access to employee-only features. Additionally, we hope to implement a cache server to facilitate a faster response time and reduce server load. Thus, we believe that a client-server view was the most sensible choice.

* 1. Scope

The system will still consist of what amounts to a functional prototype of an online ordering platform for the Davis Café, though it will not actually accept payment. However, our system will still process orders, store relevant product and order information in a database, and retrieve information our database as well. The stakeholders of this system will be made up mainly of the users, though we hope to one day have the Café administration itself as the primary stakeholders. The users will be segmented into two groups—the customers (both students and visitors) and the employees of the Café. Both groups’ relationship with the system will be organized under the client-server view, in which clients submit requests to the server and the server responds to those clients. For the customers, this data will take the form of orders placed, mock payments made, and receipts received. For the employees, this data will take the form of logins, orders completed, and requests for order metadata.

* 1. Definitions, Acronyms, and Abbreviations

**HTTP (Hypertext Transfer Protocol)**: An application protocol for distributed, collaborative, and hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web

**Hypertext**: Structured text that uses logical links (hyperlinks) between nodes containing text

**UML**: A general-purpose, developmental, modeling language in the field of software engineering, that is intended to provide a standard way to visualize the design of a system

**Cache**: A hardware or software component that stores data so future requests for that data can be served faster; the data stored in a cache might be the result of an earlier computation, or the duplicate of data stored elsewhere

**Authentication Server**: a network service that applications use to authenticate the credentials, usually account names and passwords, of their users

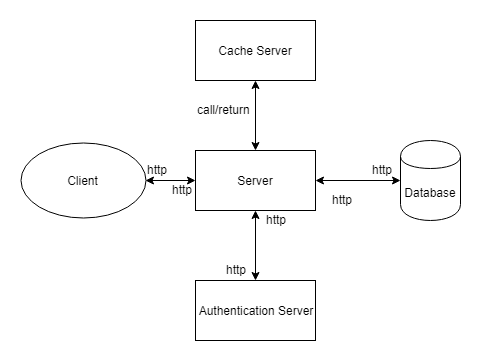
**Order metadata**: Information about multiple orders that can be used to draw a trend and make business decisions by Davis Café employees

**Order ID**- a unique number for each order that helps keep track of all relevant order information (i.e. what was ordered, who ordered it, when it was placed, and what order should it be served in, etc.)

* 1. Behavior

From the customer side, a request for the current menu would initiate interaction between the customer-client and the server. Then, the server will send the client the information from the product database making up the food and drink items available at that day and time. The customer-client will receive this list of items and will select the items and quantities that they want, submitting their choices, along with mock payment information, to the server as an order request. The server will then generate a unique order ID number for that order and store the order and payment information in our database, finally sending an order summary/receipt to the customer-client as confirmation of their order.

On the employee side, an employee-user will initiate interaction by requesting to log in to the web app as an employee to access employee-only features. The server will then direct the employee to login, checking the login information with an authentication server and prompting a retry should it fail. Upon successful login, the server would give display options only available to employees, giving options to complete or close current orders and retrieve order metadata from the order database. Completing or closing order requests would be fairly straightforward in that the employee-client would make a request for the action, the server would execute that request, and then notify both the employee-client and the customer-client. Requesting order metadata would be slightly different, in that the requests would be made to the database, which would then scrape the relevant information as specified by the employee from the database, which would then be processed and turned into visualizations by the server.



1. Architecture Views
   1. Views

As stated earlier and also modelled previously in our UML diagram, we have chosen to view our project as a client-server style of a component-connector view.

* 1. Element Catalog

Looking at the specific primary elements, we will implement a system with a total of 5 tiers.

**Client**(s)

First, we consider the clients, whose population we segment into two distinct groups: the customers and the employees as detailed below. Both groups will access our system the same way in that they would go through our web app (on either phone or computer) to request information from the system.

*Customer*

Utilizes request/reply connector type.

Client is locked until a reply is sent from the server

E.g. client is unable to “Place Order” unless the server responded

*Employee*

Utilizes request/reply connector type

E.g. Client sends confirmation of a valid order to Server

then is sent a reply of a “Place Order”, i.e. order placed from

the server

**Process**/**Business** **Tier** (Server)

Second, we consider the server that processes the data sent by the client. This tier interacts with and manipulates the requests sent by the client to and from the other servers and database. It also checks whether a specific order is valid, for example, or processes an employee’s request to alter an attribute of a menu item.

**Cache** **Tier** (Server)

Next, we consider the server that temporarily stores information and documents frequently requested by the client. This tier would serve to reduce main server lag time and traffic.

**Authentication** **Tier** (Server)

Then, we consider the server that authenticates user credentials (username and password) submitted by the client to the main server. This tier would serve as an employee login to allow appropriate users access to employee-only features.

**Database** **Tier**

Finally, we consider our database tier, which will consist of a relational database made up of tables with product information, unique order log including all information associated with the order, and user account information.