10/28/2016

**School:** University of North Florida

**Course:** Software Engineering (CEN 4010)

**Instructor:** Dr. Douglas Leas

**Team #2**

**Leader Developer:** Matthew Boyette

**Lead Designer:** David Hughes

**Developer:** William Mejia

**Developer:** Demetrius Myers

**Developer:** Slaven Popadic

V-Menu Restaurant Management System

Building the Restaurant of Tomorrow Today

Table of Contents

[Section 1: Overview 2](#_Toc465455639)

[Section 2: Project Team 2](#_Toc465455640)

[Section 3: Design Rationale 3](#_Toc465455641)

[Section 3A: Client-Server - Critical Issues and Tradeoffs 3](#_Toc465455642)

[Section 3B: Web Application - Critical Issues and Tradeoffs 3](#_Toc465455643)

[Section 4: High-Level Design 4](#_Toc465455644)

[Section 4A: High-Level Design - The Short Version 4](#_Toc465455645)

[Section 4B: Architectural Style Breakdown 5](#_Toc465455646)

# Section 1: Overview

V-Menu is a system designed to turn the day-to-day logistics of managing a restaurant into a problem solvable using distributed computing. By turning the individual human elements of the restaurant into computable problems, we hope to push the boundaries and take the next step toward a completely automated restaurant. Although a fully automated restaurant is still relatively infeasible with today’s level of off-the-shelf consumer technology, it won’t be long before advances in robotics make such an occurrence not only practical, but also commonplace. There are three primary factors of functionality that must be addressed before robotic employees become ready for mainstream adoption.

* Precision of movement. Tasks like cooking and waiting tables require precise movements, and the objects being moved are designed to be manipulated by a human hand.
* Collision avoidance. A restaurant is usually a bustling place with a great deal of movement (not only by employees, but also customers). Being able to avoid colliding with obstacles is critical.
* Communication interface. Computerized speech recognition still has some ways to go before a randomly chosen customer can reliably communicate their orders verbally to a robotic employee.

Many restaurants currently have similar systems in place already, but the feature-set and overall design is inconsistent. Further, many of these systems focus solely on patrons who want food delivered to an exterior location, or patrons who wish to pick up their food at the restaurant and then take it with them to dine elsewhere (usually a home or office).

Our goal is two-fold: to serve the needs of those dining inside the restaurant, and to merge the functionality of existing systems for a consistent design and interface which can be easily customized or extended to suit each individual client.

# Section 2: Project Team

* Matthew is the team leader and lead developer. The parts of this deliverable that he contributed are the title page, table of contents, and the overview (this page). He also created most of the diagrams and images presented in this document.
* David was the lead designer. Unfortunately, David had to withdraw from the class for personal reasons; however, we have included him here in recognition of the work that he did prior to his withdrawal. The parts of this deliverable that he contributed are some of the GUI mockups and assisted with creating some of our design diagrams.
* William is a developer. The parts of this deliverable that he contributed are assisting with creating diagrams and integration of the OpenMenu database schema with our Drupal database. He also wrote most of the text used throughout the rest of this document.
* Demetrius is a developer. The parts of this deliverable that he contributed are presenting our design to the rest of the class and the PowerPoint slides for that presentation.
* Slaven was a developer, but has recently taken on the mantle of lead designer due to the loss of David. The parts of this deliverable that he contributed are assisting Demetrius with the design presentation, and contributing GUI mockups.

Additionally, every member of the team participated in generating ideas and discussing implementation related topics during both our regular team meetings and off-hours using our Slack channel.

# Section 3: Design Rationale

There are several critical issues and tradeoffs that we must address due to some of our design decisions. They are listed and described below based on with which facet of our design each is associated.

## Section 3A: Client-Server - Critical Issues and Tradeoffs

* User devices will need wireless internet access in some form to reach the server; either Wi-Fi or a data plan. Many low-end consumer tablets do not have SIM card slots and are therefore Wi-Fi only. The restaurant owner cannot assume that all customers will have their own private data plans through a mobile carrier. Thus, they may wish to provide free Wi-Fi access to their customers to mitigate this risk. This is a responsibility our project does not address; providing this service is left to the restaurant owner.
* Performance can be a serious issue if the server is under heavy load. Likewise, there is a single-point of failure; if the server goes down, our application is useless. Restaurant owners using our product may wish to invest in a cloud-computing platform (such as Amazon’s AWS or Microsoft’s Azure, just as examples) to spin up multiple server instances for load balancing and redundancy to mitigate this risk.
* All non-trivial operations must be performed server-side; performing operations on the client-side represents a security risk. Fortunately, in our case the clients are nothing more than internet browsers; their only responsibility is to render content. So, this risk is mitigated by our project through other facets of our overall design, namely that we are building a web application (see below for more details).
* The restaurant owner may wish to invest in at least one dedicated IT professional to perform regular basic maintenance duties on the server. Many of the underlying systems and services powering our application receive regular security updates, from the operating system (Linux) all the way to the content management system (Drupal). This is a responsibility our project does not address; adhering to best practices concerning server maintenance and security is left to the restaurant owner.

## Section 3B: Web Application - Critical Issues and Tradeoffs

* The biggest drawback of this facet of our design is web browser compatibility. In theory, each web browser should render our content the same way. Unfortunately, in practice this is rarely the case. We have selected a subset of the most popular browsers, and will be testing our application with their latest stable versions. You can get more details below in the *Assumptions and Risks* section of this document.
* Different countries have different laws governing the practices of web applications. For example, in the United States, the *Americans with Disabilities Act* (ADA) mandates accessibility requirements to accommodate people with disabilities. Additionally, the European Union mandates not only access requirements, but also data privacy requirements. It is impossible for our team to completely meet these requirements in the time we have available. Instead, we have considered usability and accessibility extensively throughout our design process, and complying with any remaining specific local laws or regulations is left to the restaurant owner; we make no guarantees that we comply with every local law for any individual restaurant.

# Section 4: High-Level Design

## Section 4A: High-Level Design - The Short Version

The high-level design of the V-Menu system is straightforward. We use a standard client-server configuration. The server runs the software that enables the application to function properly, and the clients interact with that server. Clients send requests to the server, and the server responds to those requests. There is nothing particularly special about the underlying hardware used by either the client or the server. Any standard off-the-shelf computing hardware will suffice for both roles. In practice, the server hardware should be tailored for the storage of large data sets and fast access to those data sets. However, going into detail about best practices concerning server hardware is beyond the scope of this document.

Each order is a collection of menu items. Whenever an order is placed, that order gets stored in our database. This process is elaborated on in much more detail further in the document. The clear majority of the application revolves around displaying the menu to the customer, allowing him or her to choose the desired menu items, and then storing the order in our database. Aside from kitchen staff flagging an order as completed (ready for delivery to the customer) and wait staff flagging an order as delivered, the orders are not manipulated once they are placed and confirmed. This constraint will be enforced in our implementation; at no point in time will any user be directly manipulating the database. Our application will do that to maintain the integrity of the data.

// Add more?

## Section 4B: Architectural Style Breakdown

Our application uses a web-based client-server architecture using a centralized standard relational SQL database. For the web-based aspect of the project we are using PHP as our primary server-side programming language, which renders standard HTML documents using CSS for styling. Our content management system, Drupal, provides an API framework that abstracts database operations and simplifies writing code in a consistent manner. We are also employing the OpenMenu specification for our database, so that restaurant owners will be able to create, share, and distribute menus in an open format. This improves interoperability, and allows an owner who owns multiple restaurants to use our system in all his or her establishments, even if they serve different types of food.

Our server architecture takes a layered approach that puts the application layer, that is the layer that directly faces the clients and consists of our application code, at the top of the software stack. Below the application layer is the service layer. This is the layer that represents our implementation of the OpenMenu specification and our data storage schema. Below the service layer is the system layer. This is the layer that represents the content management system we are using (Drupal). Below the system layer is the network layer. This is the layer that represents the LAMP stack. The network layer handles standard network communication protocols (HTTP/HTTPS), contains the PHP interpreter, contains the relational database engine, and handles any necessary interaction with the physical hardware of the computing device that is being used as the server.

//