**Software Requirements Specification**

March 2, 2018

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

**Introduction:**

This report describes Team 2’s software requirements for its project to develop an online ordering system for Davidson College’s Davis Cafe.

1. Introduction
   1. Purpose

Our group aims to develop a web-based ordering platform that serves as an alternative option to the current in-person ordering system.

* 1. Scope

This SRS document covers the basic functionalities of which a full-fledged online ordering platform would include, though it does not cover certain extra features that we would consider adding if we had more time.

* 1. Definitions, Acronyms, and Abbreviations

**SRS** – Software Requirement Specification

**DB** –An organized collection of data, specifically a relational database

**Relation Database** - a database structured to recognize relations among stored items of information

**GUI** – Graphical User Interface; an interface that allows the user to interact with an electronic device through graphical icons and visual indicators

**ID Number** - A Davidson ID number that identifies a student or faculty member; typically begins with “801”

**POS Software** – Point of sale software; a system that controls retail transactions, specifically CatCard and credit/debit transactions

**Davidson Domains** - Davidson Domains allows students, faculty, and staff to register their own domain name and associate it with a hosted web space, free of charge while at Davidson College

* 1. Overview

The online ordering system will serve as a web-based alternative to the current ordering system in Davis Cafe. Our ordering system will start off on a homepage that will take the user to the food options with an “Enter” style button. The ordering system will display all of the food options that Davis cafe currently offers, including the “A-La-Carte” options. The users will be able to choose the quantity of their food option and additionally receive a notification that their order is complete. This modification is intended to alleviate some of the crowding and waiting that currently occurs at the menu cart during peak hours.

In addition, our online ordering system will show the price of each individual food item. Selected items will be added to a cart which will then display the subtotal. Users will be able to remove items from this subtotal or modify the quantity selected. Having the subtotal displayed on the order will ensure that students are more aware of their bill and if they are using the full amount that meal plan offers. When a user wants to put in an order, they will be able to do so in two ways: by inputting an ID number, or by inputting credit/debit card information. For ID numbers, our system will check to see whether inputted numbers are valid by seeing whether they begin with “801” as we do not have access to a database of students and faculty. However, in its current state, our system will simply accept number and letter inputs, but not process any payments or meal-plan swipes as we do not currently have access to Davidson’s POS software.

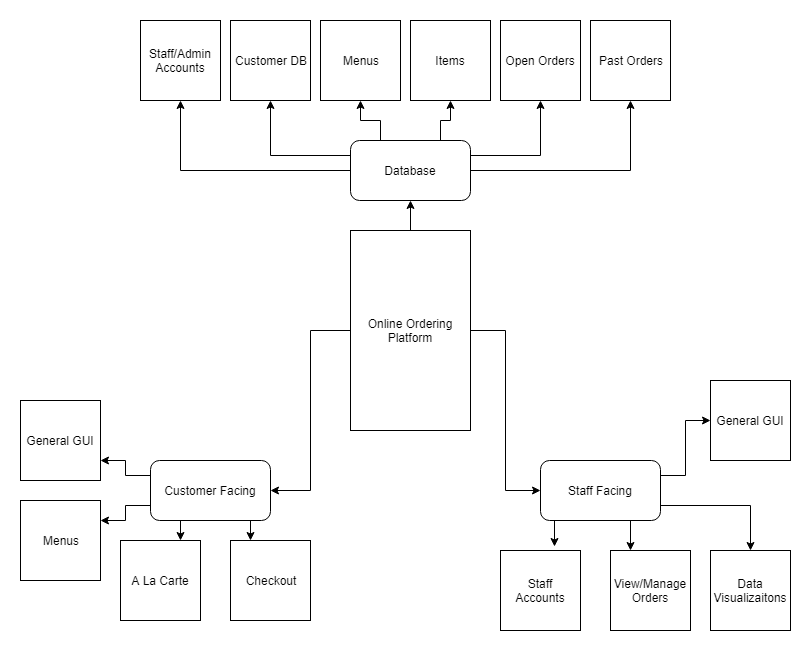
1. Overall Description
   1. Product Perspective

We aim to create a website that can service requests from a variety of users simultaneously while receiving their payment information and maintaining their privacy. Unfortunately, due to the nature of the POS system employed by the Davis Café, we will not be able to collect payments from the CatCards of Davidson students, faculty, and staff. Nonetheless, we still aim to create a product that provides the framework for collecting payments and maintaining privacy, receiving updates in order to accommodate changes in menu, and guiding the user through a simple process for placing orders. Our website must also be able to display the orders so that the workers of the Café can quickly process and complete them. These are the most essential characteristics of our product, and as such are the ones that will guide our decision in terms of the hardware and other infrastructure we will be using.

Such a division between customer-facing and staff-facing features will govern our use cases. The customer use cases are fairly simple as detailed earlier, mainly consisting of viewing and ordering from menus. The staff use cases are more nuanced, being split between normal employees and administrator-level employees. Each access level will have to be able to log in and out of appropriately permissioned accounts and then view, complete or modify orders.

* 1. Product Functions

As detailed in the block diagram below, we break down the major functionalities into three different views: one that faces the customers, one that faces the staff members (both employees and administrators), and one that is server-side. Conceptually, we break the customer-facing side into the front-facing GUI design, the menu view, the a la carte item view, and the checkout view. On the staff-facing side, we break the subcomponent down into staff account operations (employee and administrator), an order view, a visualization view meant to display these results, and the general GUI design. Finally, though both customers and staff will be interacting with our Davidson Domains-based server, we separate our database into a third server-side functionality component consisting of multiple tables in which to group and compare data.



* 1. User Characteristics

We are expecting our users’ technical capabilities to vary widely. As such, we would hope to make the ordering platform as accessible as possible, with clear instructions and descriptions of how to place an order.

* 1. General Constraints

For the users, we do not anticipate there being any particularly cumbersome constraints. It will require access to our website, and the formatting may not be as nice on mobile devices as one might hope, but nonetheless our ordering form will allow users to order anything from the Café so long as they have the means to pay for it. There will be a limit to the number of requests our website will be able to speedily process at one time, but this number will be high enough so that it should not ultimately prove to matter given the limited amount of traffic the Café receives.

* 1. Assumptions and Dependencies

We are having the customer pay for their order upfront, and by doing this we are assuming that they will actually follow up on their order and take the items they placed in their order. The Davis Café also has a limited amount of space to place finished orders, so with this system we assume customers will retrieve their order in a reasonable amount of time. We also assume that the customers will not place any impossibly large orders that the staff could not hope to fulfill in a timely manner. Since the customer must pay upfront, this concern is also somewhat limited by cost. We are also okay with making the first two assumptions, as the Café makes those same assumptions already for their paper forms, and this does not seem to cause many issues. We are also assuming the Davis Café workers will be able to adjust to having an online ordering system, which, while it may take some time to adjust, we do not think is an overly large assumption, as ultimately this is a change that will benefit the workers and make the ordering system more efficient.

1. Detailed Requirements
   1. External Interface Requirements
      1. User Interfaces

Clients on both the customer and staff side shall be able to interact with the software via a web-based browser

* + 1. Hardware Interfaces

Our project shall be accessible via a web-based browser, and thus any computer with access to the internet should be able to access it. Demands on the CPU shall be relatively minimal.

* + 1. Software Interfaces

Our website shall be used on internet and should be accessible for all major modern browsers, (i.e. Safari, Firefox, Chrome).

* + 1. Communication Interfaces

We shall make use of a server that connects the user to the database of food options and sends the users’ orders to the employees so that they may swiftly respond to them.

* 1. Functional Requirements

Here we detail the use cases that our system will face

* + 1. **Mode 1**: Customer Facing
       1. **UC1**: Open/navigate web-page

**Primary Actor**: Customer/Student

**Precondition**: Must be connected to WIFI

**Main Success Scenario**:

* Home page has all possible subpages
* User directed to new page when subpage clicked
  + - 1. **UC2**: View Menu

**Primary Actor**: Customer/Student

**Precondition**: User successfully opened home page

**Main Success Scenario**:

* User chooses a menu and is taken to a new page with selected menu
* User is able to see all food options for each menu
* Systems shows prices of items
* Customer can navigate between menus

3.2.1.3. **UC3**: Place order

**Primary Actor**: Customer/Student

**Precondition**: User successfully opened home page

**Main Success Scenario**:

* System shows prices of items
* User shall be able to select items they want
* User shall be able to deselect items they no longer want
* System shall display subtotal on page
* System displays final receipt
* User submits order

**Exception**:

* User has no food options selected- System will notify user, ask to select items
* User selects item that is unavailable- System will notify user, ask to continue without (showing new balance) and if user would like to select other items

3.2.1.4. **UC3**: Submit Payment Option

**Primary Actor**: Customer/Student

**Precondition**: User submitted an order

**Main Success Scenario**:

* System offers two options of payment: ID or credit card
* User enters form of payment
* System accepts payment

**Exception**:

* User enters invalid ID or card number- System will notify user and ask to reenter
  + 1. **Mode 2**: Staff Facing
       1. **UC1**: Open/navigate webpage

**Primary Actor**: Employee

**Precondition**: Must be connected to WIFI

**Main Success Scenario**:

* Home page has all possible subpages
* User directed to new page when subpage clicked

**Exception Scenarios**:

* Link clicked is broken
* User loses connection with internet
  + - 1. **UC2**: Login as employee

**Primary Actor**: Employee

**Precondition**: User must be within web app domain

**Main Success Scenario**:

* Page displays employee-only pages
* Employee can see and access options for viewing and managing orders

**Exception Scenarios**:

* Login fails (username and/or password is incorrect)
  + - 1. **UC3**: View/manage orders

**Primary Actor**: Employee

**Precondition**: Must be logged into employee account

**Main Success Scenario**:

* Can view all orders currently open
* Can mark orders as complete (removing them from “Open” view)

**Exception Scenarios**:

* Cannot view open orders
* Cannot mark orders as complete
* Completed orders are not removed from list of open orders
  + - 1. **UC4**: Logout of employee account

**Primary Actor**: Employee

**Precondition**: Employee must be logged into web app domain

**Main Success Scenario**:

* Page displays public facing view
* User unable to access employee-only features

**Exception Scenarios**:

* Logout fails
* User has continued access to employee-only features
  + - 1. **UC5**: Login as administrator

**Primary Actor**: Administrator

**Precondition**: User must be within web app domain

**Main Success Scenario**:

* Page displays admin-only pages (order metadata)
* Admin can see and access options for viewing and managing order metadata

**Exception Scenarios**:

* Login fails
  + - 1. **UC6**: View/manage order metadata as administrator

**Primary Actor**: Administrator

**Precondition**: User must be logged in as admin

**Main Success Scenario**:

* Can view open and completed orders over specified range of time
* Can delete/modify past orders

**Exception Scenarios**:

* Cannot view current and/or past orders
* Cannot modify current these orders
  + - 1. **UC7**: Logout of administrator account

**Primary Actor**: Administrator

**Precondition**: Administrator must be logged into web app domain

**Main Success Scenario**:

* Page displays admin-only pages (order metadata)
* Admin can see and access options for viewing and managing order metadata

**Exception Scenarios**:

* Logout fails
* User has continued access to admin-only features



* 1. Performance Requirements

As our ordering platform will be hosted on a Davidson Domains server, our platform will be limited to the specific constraints of that server. That being said, we expect that our system exhibits not more than a 5-second response time for normal (view-only pages) and a not more than 8-second response time for order submission. Such wait times will also be mitigated by a notification that tells the user that acknowledges the submission and states that it may take a few moments to be processed. Our system shall handle an average user count of about 25 users at any given time, and should be able to handle a peak expected sustained usage of 100 users without service interruption. Should exceptionally high volume occur, the system will display a message to inform users of unexpected high traffic volume and requests they try again in a few minutes or order in-person.

* 1. Design Constraints

The choice of internet browser is one particular constraint that we must address in the development of our online ordering system. We plan on our ordering system to be compatible with Google Chrome, Safari, and Mozilla Firefox. However, since we will be working with HTML5, which is aimed at being compatible with major internet browsers, this should not be a large issue. Additionally, as discussed above in the performance requirements, our platform will be limited to the specific constraints of the Davidson Domain Server.

Another constraint that we may possibly face lies in our choice of programming languages. The languages we choose may be limited in functionality and thus might restrict what we can or cannot do.

* 1. Attributes

Since users will have to enter payment information into the system to place their orders, protecting their privacy is of the utmost importance. Thus, our website should require users to enter a password in order to log onto their account, and their payment information will be protected. Furthermore, we would like to make it so that users can order any type of item available at the Café, but not all of these items can be prepared by the staff. Ideally, our software would provide for some sort of verification system, like a digital receipt, that would assure the user took the items he/she paid for. However, as far as our initial release is concerned we will likely not implement this feature and instead rely upon the “honor system” for these purposes.