Penn State Food Locator

Jake Smith, Luke Leiter, Andrew Kyffin

# Software Design Specification Document

**Version: 2.0** **Date: 12/14/2020**

# Table of Contents

OBJ

OBJ

OBJ

OBJ

OBJ

OBJ

OBJ

OBJ

OBJ

OBJ

OBJ

OBJ

OBJ

OBJ

OBJ

OBJ

OBJ

# 1 Introduction

This section provides an overview of the entire design document. This document describes all data, architectural, interface and component-level design for the software.

## 1.1 Goals and objectives

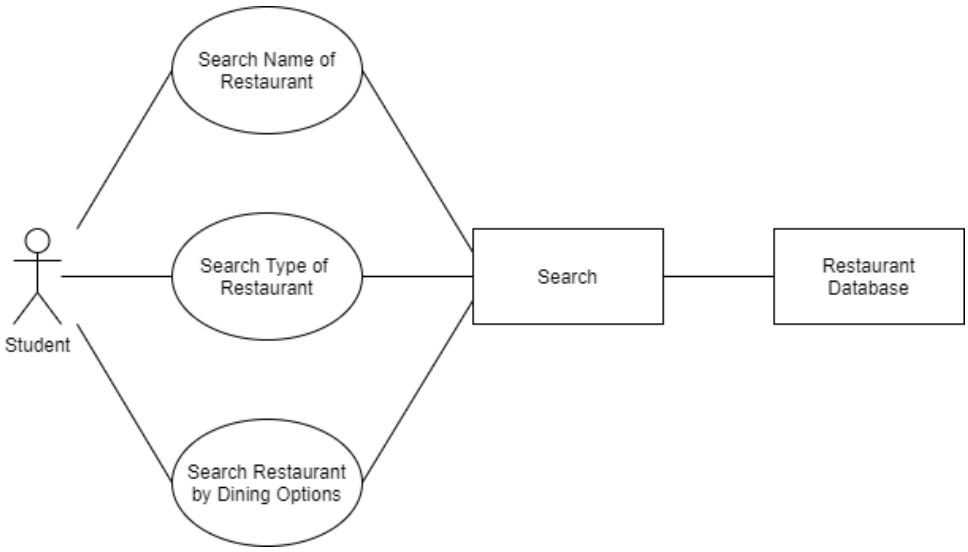
To create an app that extends the current functionality of the Penn State Go app and its food options. This will be implemented in a web browser.

## 1.2 Product context

The product will be distributed as a web application for anyone with access to the internet to use. It will be used to search for dining options in an area. While it is targeted at students to use, the app can be useful for teachers, alumni, and people who live near campus.

## 1.3 Statement of system scope

The software will take in a user defined search for a restaurant. This search will be used to search a SQL database of restaurants in the area. This search will render a webpage that has search results based on the users input. The user will then be able to select a restaurant they want to learn about.



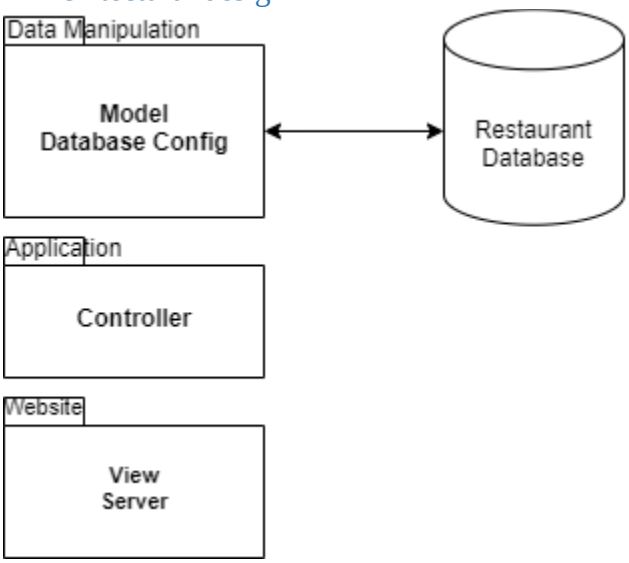

## 1.4 Reference Material

*N/A*

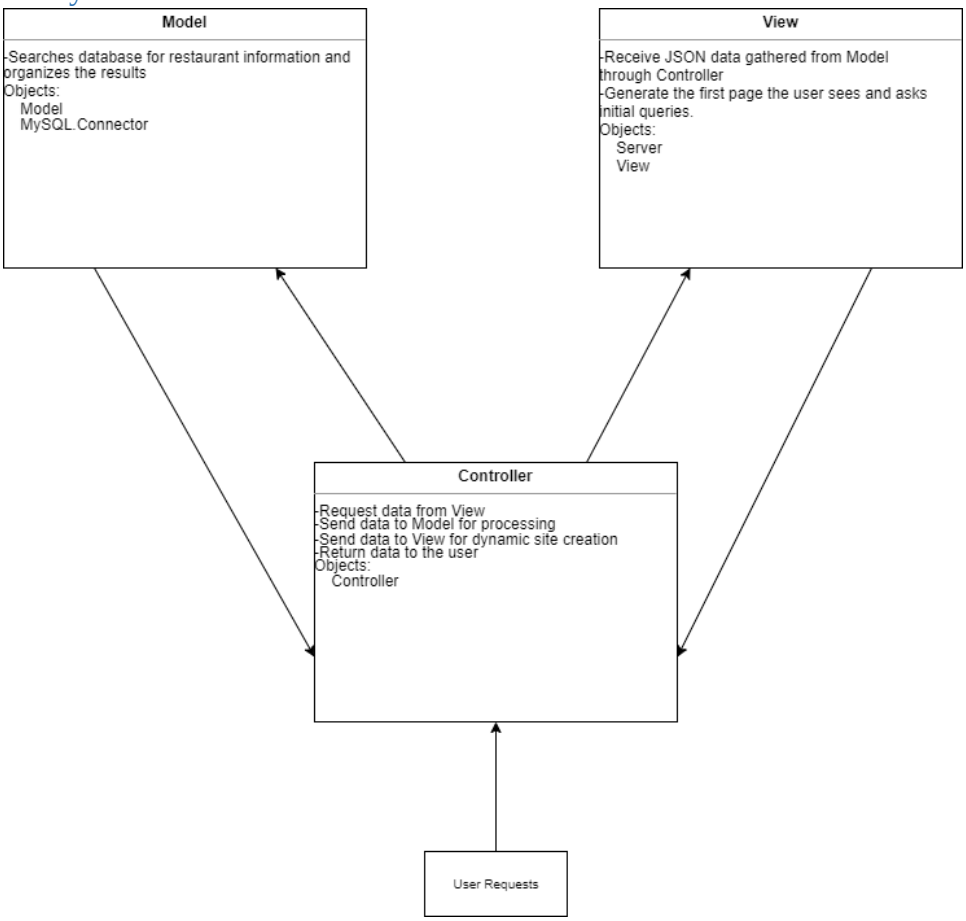
## 1.5 Definitions and Acronyms

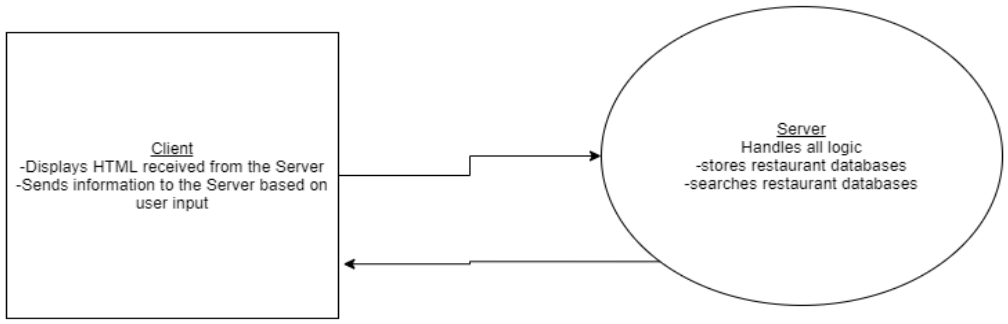
*N/A*

## 2 Architectural design




## 2.1 System Architecture






## 2.2 Design Rational

MVC

We chose to use MVC because it is a commonly used design scheme for web applications. Its usefulness for web applications stems from how MVC compartmentalizes different functionalities of the applications. This makes small and large systems easy to manage, especially when a team is working on the project together. MVC will allow us to identify how the application will communicate between the different systems and the order of our development.

Client-Server

The Client-Server software architecture is useful for separation of logic and display code. The client possesses the code to take in display information to show the user. The client is also responsible for taking inputs from the user and sending it to the server. The server takes in information from the client and uses it to search the database. The database's result will determine the next page that needs to be presented to the user. The server is responsible for sending this data. The pros of this architecture is that the client is always one degree away from accessing the databases. A con is that the code for the displaying the webpage is not consolidated, code for the webpage would be spilt across the server-client relationship. This is not ideal for abstraction as some classes would need to be present in the client and server.

# 3 Key Functionality design

## 3.1 The Restaurant Search

### 3.1.1 Restaurant Search Use Cases

**Level:** Primary Task

**Primary Actor:** Student

**Precondition:** None

**Minimal Guarantee:** Student gets a search result

**Success Guarantee:** Student finds a restaurant to order from

**Trigger:** User goes to the web address

**Success Scenario:**

Action 1. Student opens application in browser

Action 2. Student chooses type of restaurant (fast food, sit down, etc)

Action 3. Student browses restaurant options

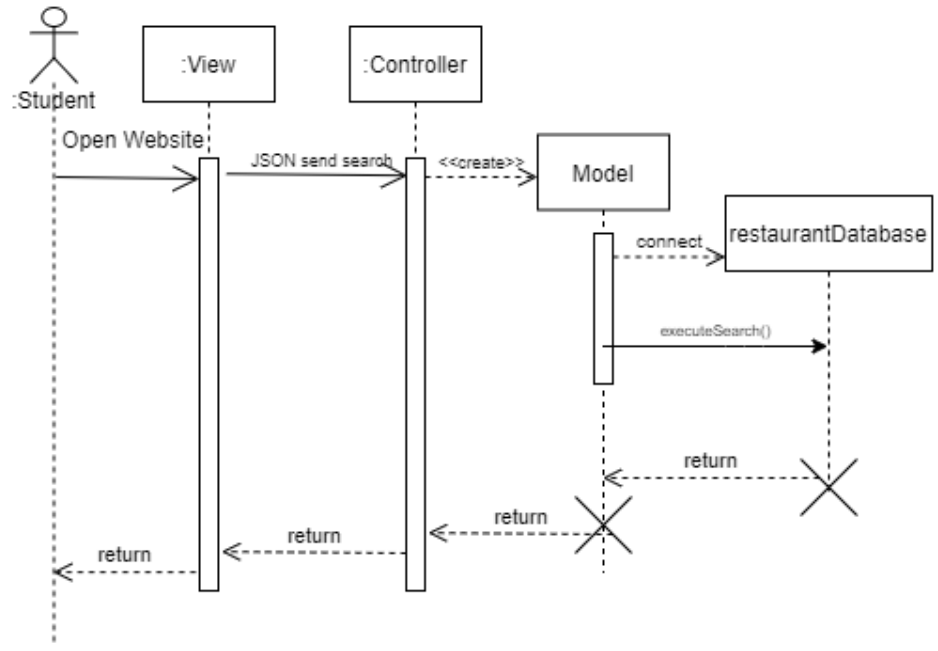
Action 4. Student selects restaurant to read information about

**Extension Step:** Branching action

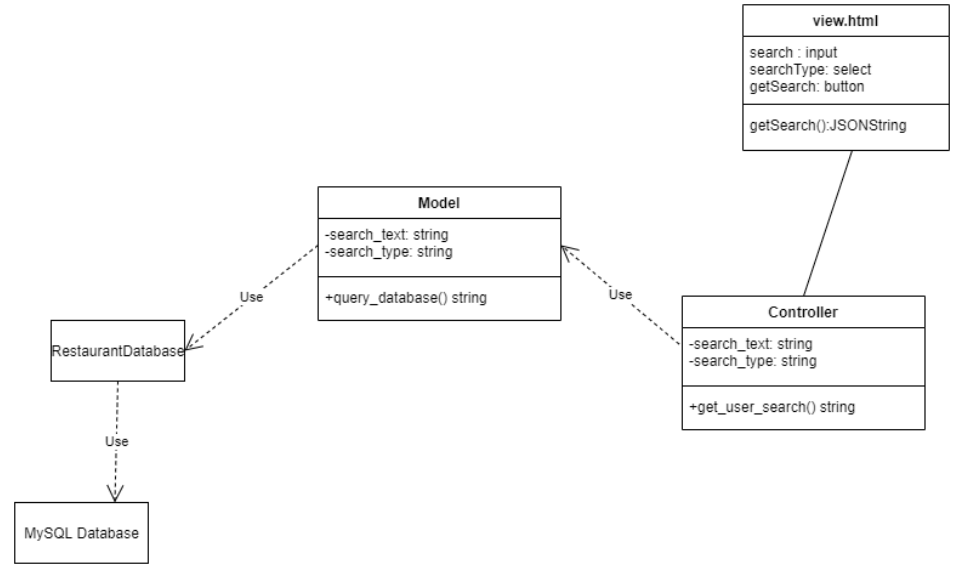
2a. No suitable restaurant is found

a1. Student searches again with different criteria

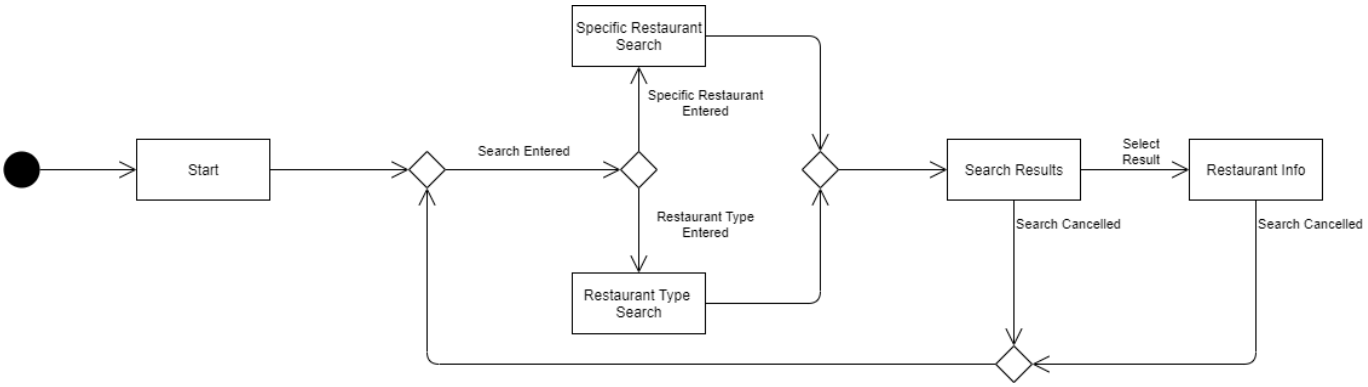
### 3.1.2 Processing sequence for Restaurant Search




### 3.1.3 Structural Design for Restaurant Search




### 3.1.4 Key Activities




### 3.1.5 Software Interface to other components

Software queries a database to find information on the restaurant. This is based on what they are searching, type of food, a specific restaurant, or different dining options.

## 4 User interface design

### 4.1 Interface design rules

Simple, understandable layout that adheres to the Penn State color palette.

### 4.2 Description of the user interface

The user interface (UI) is web-based and provides a visual front-end to our client's searches.

### 4.2.1 Search Page

The Search page will display the search results when the user searches the database.

#### *4.2.1.1 Screen Images*


**Search Page**

## 5 Restrictions, limitations, and constraints

- The information will be stored in a MySQL database.
- The final test product will be used only on Google Chrome, while the final product will only be accessible through the Penn State Go mobile application

## /* 6 Testing Issues

N/A

## 6.1 Classes of tests

## 6.2 Expected software response

## 6.3 Performance bounds*/

# 7 Appendices

## 7.1 Packaging and installation issues

Type the URL into your choice of internet browser.

## 7.2 User Manual

Give step-by-step description of using the key features of the system.

Type the URL into your choice of internet browser. Use the sign in feature in the top-right corner to create/access your account. Using the query options in the center of the page, decide what type(s) of restaurants you are interested in looking for in your area. Once search results have appeared, select one you are interested in to view its information.

## 7.3 Open Issues

Features considered but not finished:

Account System

IP to Location system

## 7.4 Lessons Learned

### 7.4.1 Design Patterns

What are the design patterns used and why?

The Singleton Design pattern can be useful for our project. We used this in our webpages and databases. After these objects are created, they need their information changed but they are reused throughout the program. MVC is also useful; it is commonly used with web applications. MVC separates different processes of the program so that they are more abstract. This allows us to create less code and subsequently more maintainable code.

### 7.4.3 Team Communications

How team communications were conducted and where could you improve in the future?

The team would consistently use the Discord app to communicate about and work on the project. One way that we could have improved our communications was to create a set schedule for when the team would convene to discuss the project. We also used GitHub to store the code and some documentation files so that they were accessible to everyone.

### 7.4.4 Task Allocations

How your team allocate tasks and responsibilities and where could you improve in the future?

The team would allocate tasks and responsibilities on a first-come-first-serve basis, where team members would call what they wanted to work on as opportunities to do so showed themselves.

To improve this system, the team could have simply discussed who would do what prior to anyone starting the project work.

### 7.4.5 Desirable Changes

Assume that you have another month to work on the project, what aspects of the system you would like to improve? What are the additional features you want to add to the system? [Each student should use a separate paragraph to respond to the questions]

Andrew:

I would have liked to improve my knowledge of Python. Prior to this project, I had no Python experience whatsoever, thus this project benefitted me greatly in that regard. Using that extra month, I could improve how I wrote some of the code for this project as well as improve my own knowledge of the programming language for future projects.

Jake:

I would like to develop a system that allows the user to adjust the range of their “nearby” search. By the end of the project, they will be able to filter their results within a limited distance, however I would like to adjust the maximum distance, allowing users to have more options. I would also like to generally improve the quality of the web app and improve the visuals.

Luke:

I would like to make quality of life improvements; improve the website layout and look and build account functionality. The website layout and look aren't functionally important, but it would make the user experience better. Building more account functionality will allow the user to save information, such as a favorite restaurant. Those are just some of the quality of life improvements I would have liked to make for our app.

### 7.4.6 Challenges Faced

Among requirements specification, system design, and system implementation, which one you think is the hardest task? Why? [Each student should use a separate paragraph to respond to the questions]

Andrew:

I believe that requirements specification is the hardest task for this project. I feel that way because we do not have a strong method of getting requirements other than considering what we believe to be important for this project. Thus, the project may not fully reflect what is required in an actual space.

Luke:

I believe system design is the most difficult task. It is difficult because this is when the system is taken from requirements, which is relatively abstract, to implementable classes and systems. Especially if one is following the waterfall method, the system design is relatively unchangeable and contributes greatly to the quality of a project. Deciding how to organize system objects to each other and other systems, some we may not have control over, is incredibly difficult to do well. Requirements specification is simple and just requires the right questions to be asked. Implementation is simply because the design has already been decided. This only leaves the way to code something in question; there is often a best way to code certain logic. That is why system design is the most difficult task.

Jake:

System design was easily the hardest task, because none of us had any practical experience with this kind of system before, so decided what we needed before we implemented things became very difficult for us.