WAFER: Web Application for Evaluating Restaurants

Software Requirement Specification

Team Beep Boop

Software Engineering, CSCE 3444, Fall 2020

Nabin Bhatta

Cayden Chancey

Matthew Curtin

Tyler Parks

Saiman Sigdel

Table of Contents

1. Introduction

1.1 Purpose

This software requirement specification form is written to document Web Application for Evaluating Restaurants (*WAFER's*) system and function requirements. Listed in the form is a generalized overview of the product, user perspective and product characteristics section, interface summary, and all of the requirements needed to produce this product.

1.2 Scope

The web application *WAFER* will be an online reviewing and rating platform where users will be able to search for nearby restaurants. Users will have the ability to write and read reviews of these restaurants, rate them based on a "out of 10" rating system, and leave or answer questions about them.

Our website will assist users by showing them restaurants near them, which can be filtered by distance, rating, and type of food. This functionality is based on the user's location, using the computer's -- or phone's -- location services.

This application will be to the convenience and benefit of the user. Our goal with this application is to decrease the time it takes from when the user decides they're hungry and the time that they are eating. The objectives and goals of this application is to, not only provide this service, but to continue to improve it with user/customer feedback.

1.3 Definitions, Acronyms, Abbreviations

In this section, any acronyms/abbreviations will be defined here as well as other words that need definition while reading.

WAFER - Web Application For Evaluating Restaurants
- The name of the application that is documented in this form.

GUI - Graphical User Interface
- The graphical user interface is a form of user interface that allows a user to interact with a software. This type of user interface is different from text-based ones because it mainly uses graphics and pictures to portray information.

API - Application Programming Interface
- An application programming interface is an interface which defines interactions between multiple softwares and databases.

GPS - Global Positioning System
- Global positioning system is a satellite-based navigation system that can be used by anyone.

TBD - To Be Determined
- Some factors of our project are still unknown or have not been decided on. Finding a TBD acronoym in this document will represent this uncertainty.

1.4 Overview

The structure of this document can be seen in the Table of Contents above; however, a modified, simplified version will be documented below.

This document begins with an introduction section that lays out the structure of it and summarizes the main topics covered including functions, interfacing, and requirements. Following the introduction, the overall description, including product functions and user classes, is documented. The next main topic that is documented is the application's specific requirements. This includes the external interface, functional requirements, non-functional requirements, and design constraints. Finally, the appendices section is used for additional content or references that is not stated in the document.

1.5 References

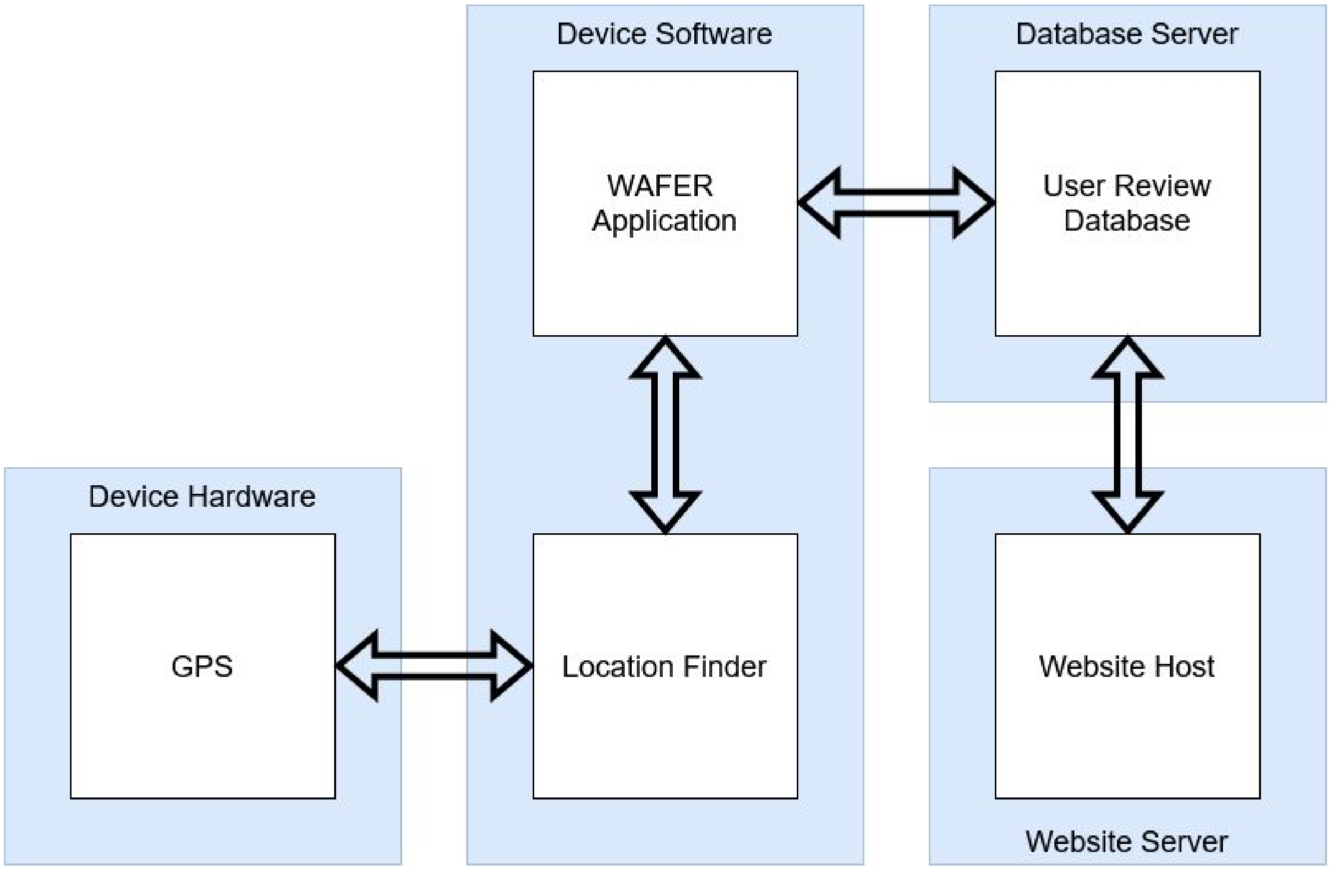
No references were used explicitly to create this document. Outside references that we use to create the program will be cited at a later time.

2. Overall Description

2.1 Product Perspective

WAFER will be made to track reviews and evaluations of restaurants through a web application. The original idea was to have users in a geographical area communicate with each other about restaurants or other public services. It is a brand new, stand-alone product.

Component Diagram

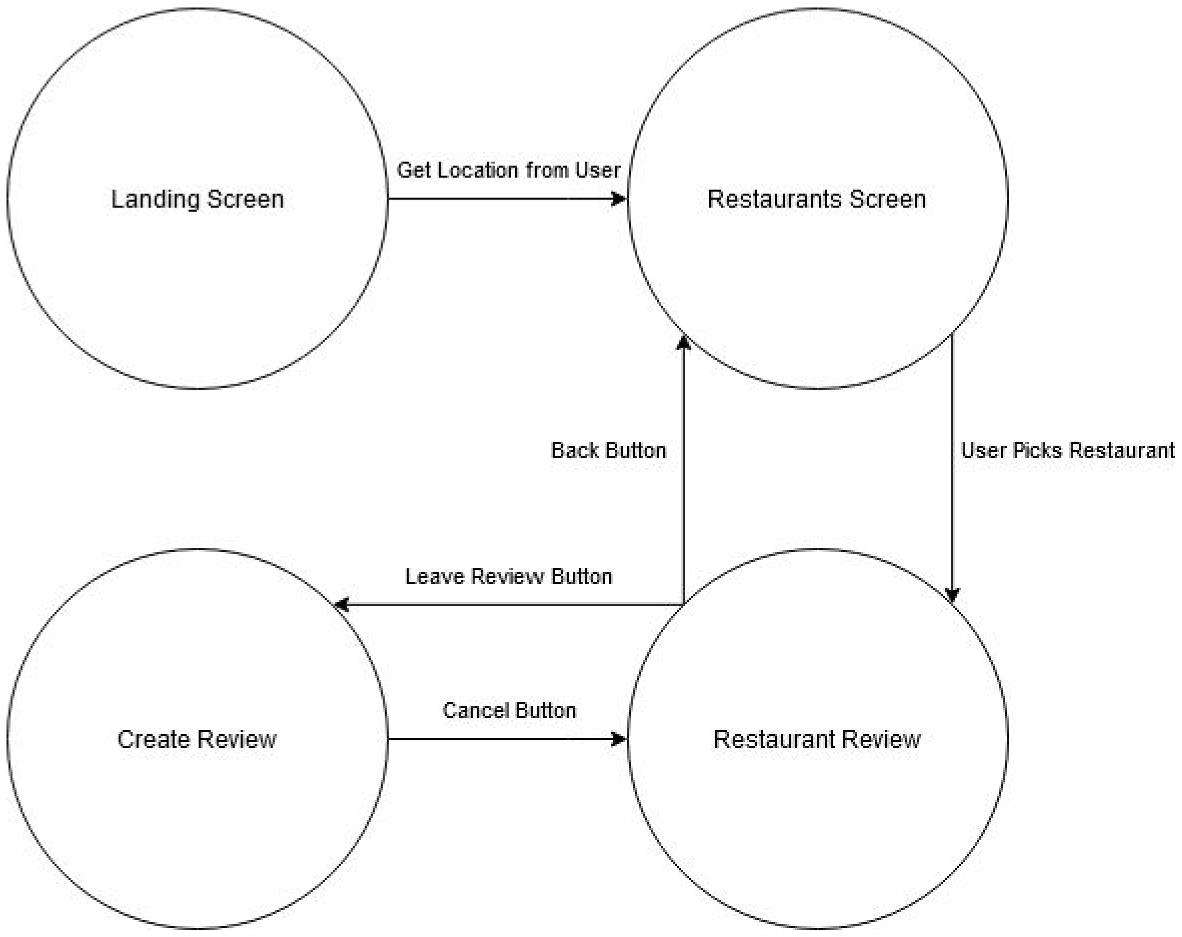


2.2 Product Functions

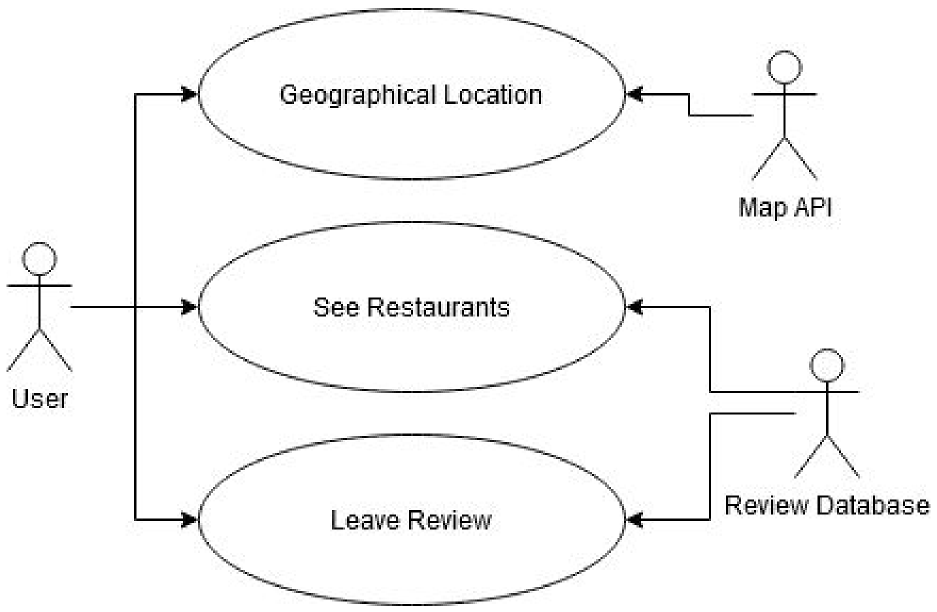
There are several functions the *WAFER* software will need to perform.

1. The application will need to get the user's location.
2. The application will need to find restaurants near that user.
3. The application will display the restaurants near the user.
4. The application will show ratings and reviews about that restaurant.
5. The user can make a new review and rate restaurants.

State Transition Diagram



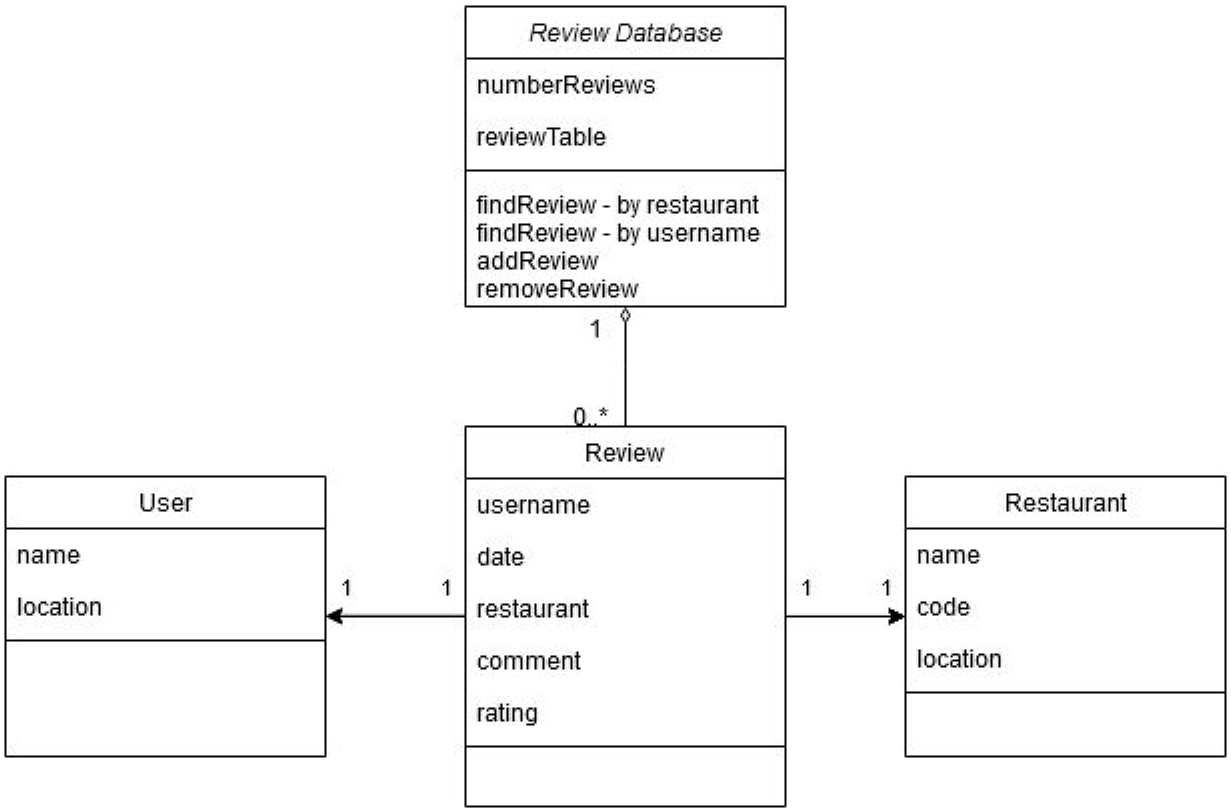
Use Case Diagram



2.3 User Classes and Characteristics

The intended user of this system is a person that lives in an area looking for restaurants to go to. The user is never stored in the database for security reasons. The user can leave a review for a restaurant that is stored in a Review class. The Review class keeps track of the username that was entered and a restaurant. The Review contains a comment, a rating, and a date of the review. All the Review objects are stored in the Review Database collection class. This class keeps a table of all the reviews and keeps track of the number of reviews added. You can add and remove removes, as well as search by restaurant or username.

Class Diagram



2.4 Constraints

The *WAFER* project has a few constraints that we will need to be mindful of.

1. User Security
    a. None of our group members has experience with web security. We do not feel comfortable with asking users to input private information such as passwords, their addresses, their pictures, or their legal names.
    b. Our solution around this will be to implement a system that does not rely on user information besides their general area (zip code) and a username of their choice.

2. Geographical Information
    a. *WAFER* depends on geographical information from the user being able to find restaurants near them. We are constrained by if we can find free or open source software that lets us connect a user's geographical position to restaurants around them.
    b. If the user does not have geographical location found by their device, we can have them enter a zip code to get their location.
3. Database
    a. The project needs a way to store review information by restaurant. This means that we will need to implement a database. We are constrained by the size of this database, the security of it, and how easy it is to implement the database into our application.

2.5 Assumptions and Dependencies

We made a few assumptions that dictated what our requirements are.

1. Restaurant reviews will be made anonymously
    a. Because of the small scope of our project, we did not want to implement a user database. We did not feel that we could securely manage private user information. We are going forward with the assumption that reviews will be placed anonymously.
2. Users will not abuse review power
    a. Because we are not monitoring user logins or how many times a user posts, there is no safeguard around one user reviewing with spam. We decided that the scope of the project is small enough where we can assume the users will use the website responsibly.
3. Geographical information access
    a. We assume that we will be able to get geographical information from the user. Specifically their zip-code or location provided by their GPS. We want the project to sort reviews by location, so we are assuming we will get geographical information from the user.
4. Database is big enough
    a. We are assuming that the database we plan on using will be large enough to store reviews about the restaurants. If needed, we can narrow our scope to just restaurants in Denton.

3. Specific Requirements

3.1 External Interface

3.1.1 User Interface

The user interface is GUI based. The users will click on a button labeled as find restaurants near me which will show the users a list of restaurants near them within a specific radius (TBD), which can be filtered by distance, rating, and type of food, with the review. The users will rate the restaurants “out of 10” by clicking a number between 1 and 10, which will be listed horizontally. Users will use a text box to leave a question, comment or answer a question for a specific restaurant of user’s choosing.

3.1.2 Hardware Interface

Users will be able to access the program through a gps enabled personal computer or a mobile phone. The UI will be almost similar for both devices ept for the dimensions of the UI. Users will be able to perform the same function on both devices.

3.1.3 Software Interface

A web-based API will be used to build the application. The web application will use a geolocation API (TBD) to get the location of the User and a geocoding API will be used to find restaurants near the user’s location. A database (TBD) will be used to store and manage data for the application. It will be used to store the user created content like ratings, information entered in the text box and so on.

3.1.4 Communications Interface

The users will use a web browser and internet connection to interact with the application. The application uses a web-based API (TBD) to communicate between user and server.

3.2 Functional Requirements

The basic system behaviour on the WAFER defines various processes listed below:

a. The user should have the ability to choose a username.
b. The users provide the information about the restaurant with the ratings and review.
c. The GUI should have a connection with the API that interacts with the user when the certain button is pressed.
d. There should also be a maps geocoding API that should interpret the users text or a location.
e. There should also be the rating options to give a rating and the overall average rating is calculated.
f. All the user information and data should be saved in a table in a database.
g. The user should be kept anonymous after the review is posted.
h. The netlify should be used to host the web interface.

3.3 Non-functional Requirements

3.3.1 Usability

The WAFER project will be a relatively simple and easy-to-use website which should take no more than 10-15 minutes to learn how to use. The UI will be the most user friendly design we can create to make it as quick to use as possible so users can get the information they need and then go to the restaurant of their choosing.

3.3.2 Reliability

The project should have perfect reliability assuming that both our web hosting service and the geocoding API do not go down as a whole, in turn bringing down the WAFER project with it. The website would only be down outside of these two situations if we need to do site maintenance or need to update the website to a newer version.

3.3.3 Performance Requirements

The WAFER project will be using the Netlify web hosting service and the Google maps geocoding API to create a seamless website experience. The response time of the WAFER project will be near instantaneous as we only rely on small files to be loaded during the viewing of reviews for a specified restaurant.

3.3.4 Safety Requirements

To reduce the risks of our website we will be choosing the most stable versions and products for our web hosting service and geocoding API. This is because, like listed above, these are the two ways that our website will be unavailable to any users. Another way we will be reducing risk is by using a version control system such as GitHub to keep a stable backup of each version of the website in case of a failure while using the website.

3.3.5 Security Requirements

The security system will be our web hosting service's firewall, which should fend off most attackers from the WAFER project.

3.3.6 Legal

To avoid legal issues such as copyright infringement on the restaurant's logos and any other intellectual property belonging to the associated companies we will provide a warning both when you open the website and at the bottom of the web page stating that we do not own any of the pictures used for each website.

3.4 Design Constraints

Front End:

The system is built on a windows operating system using mainly HTML, CSS and Javascript. The main interface for the user will be GUI where the user interacts in the website. This interface will be built using HTML, CSS and Javascript. We will be using Atom as a text editor which is very simple to use and supports many languages. There will be two main functionality for the user: name and location. The system is hosted with Netlify for the website that provides free SSL. Since, the user will not be providing their personal information, the security of the website is not the priority in our application. However, we will be adding the certificate from Lets' Encrypt for the user to feel more secure.

Maps Geocoding:
We will be using a geocoding API that accepts a location name or a name of a restaurant or a zip-code and returns the geographic coordinates that are plotted in the map. The Google Maps Geocoding API or similar application will be used to locate location and save the corresponding rate and review of the user in the database.

4. Appendices

No extra information was included within this appendix.