Software Requirements Specification

for

EurekaEats

Version 1.1.7 approved

Prepared by Dana Mendoza, David Tabor, Derek Tan, Devin Chang, Iqra Irfan

CSULA

10/21/2023

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Derek T.	10/22/23	Updated application message codes.	1.1.3
Derek T.	11/10/23	Moved application message docs to SDD.	1.1.4
Derek T.	11/12/23	Updated Definitions of Done and TBD.	1.1.5
Derek T.	11/16/23	Updated Legal & Ethical Considerations.	1.1.6
Devin	11/28/23	Updated 2.2 Product Perspective Updated 2.6 Design and Implementation Constraint Updated 3.3 Software Interfaces Added 4.3 Logical Database Requirement	1.1.7
David T.	11/29/23	Added 4.1 Functional Requirements Added 4.2 External Interface Requirements	1.1.8
Devin	11/29/23	Updated 4.2 External Interface Requirements Updated 4.4 Design Constraints Updated 5.3 Security Requirement	1.1.9
David T.	11/30/23	Finalized formatting for current version	1.2.0

1. Introduction

The software requirement specification (SRS) is a document that describes what the software will do and how it will be expected to perform. The primary audience for the SRS is developers, testers, and project managers.

1.1 Purpose

The purpose of this document is to plan and decide what specifications will be included in the document. The software engineers involved in this project will better understand the requirements, the breakdown of the project, and individual software related to this project.

1.2 Intended Audience and Reading Suggestions

This document is intended for developers and testers to understand the requirements and purpose of the EurekaEats web application. Sections one and two give an overview of the project, sections three to five are specific functional requirements, and section 6 covers other concerns within the scope of this document. The developers should be familiar with the entirety of this document. Software testers should be familiar with the first four sections.

1.3 Product Scope

Software Product: EurekaEats

This software is a website that combines the specific food preferences of a user and finds a local, best-fit restaurant by those factors. Website users can write reviews, rate reviews, and find restaurants. The benefit of the software after release consists of a new foodie base community. Any user's primary use of the EurekaEats website is rigorously selecting the best dining location for themselves. Another primary use of the product by any user is to assist other users in choosing optimal dining locations through reviews.

1.4 Definitions, Acronyms, and Abbreviations

API - Application programming interface

Definition of done for programming work items:

- The code functions as expected with no errors or bugs.
- The code style follows the appropriate requirements, such as PEP8 for Python and the criteria for JavaScript.

Definition of done for non-programming work items:

- If the item covers documentation, the update is necessary and agreed upon by most of the development team.
- If the item covers a spike, most of the development team must agree that the work is finished to a reasonable degree.
  - Example: Learning React would require a team member to understand enough React to implement functional UIs for the website according to the *DoD of Code.*

1.5 References

- Software Requirements Specification document with example - Krazytech
- Server Size Guidelines – Logi Analytics

2. Overall Description

The website will allow users to create a profile, search for restaurants, and leave reviews. The search will be able to find restaurants for users based on their eating preferences.

2.1 System Analysis

The project's most important goals are to offer a more fitting service to find and select the best restaurants for any user and to serve as an informal reference to inform users on which restaurants are most desirable or not.

The technical hurdles of this project include designing an algorithm to find the best restaurant within constraints, creating a database schema, and creating a user-friendly, accessible UI.

Our software development team will research and design a UI and database schema and have multiple coding reviews to fit the requirements.

2.2 Product Perspective

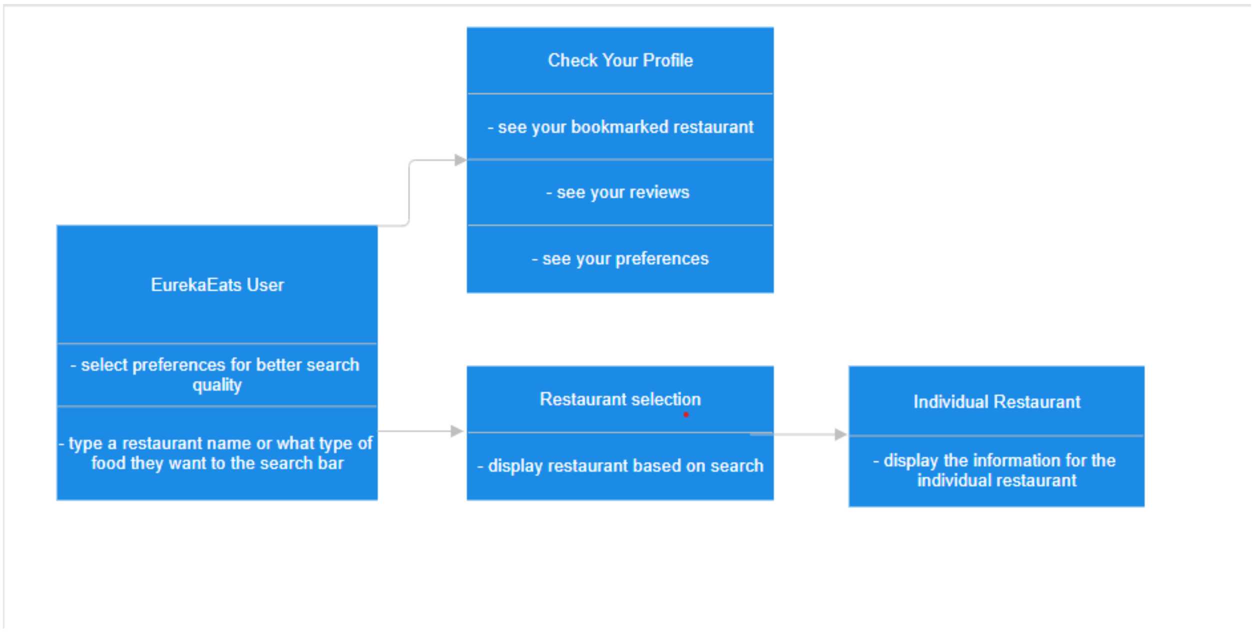
EurekaEats is comparable to Yelp. EurekaEats differs from Yelp because users can define more specific eating requirements when searching for restaurants.

User’s information:

- User information includes the user’s account and eating preferences, such as type of cuisine and type of restaurant. This information can be used to find a restaurant for the user.

Restaurant information:

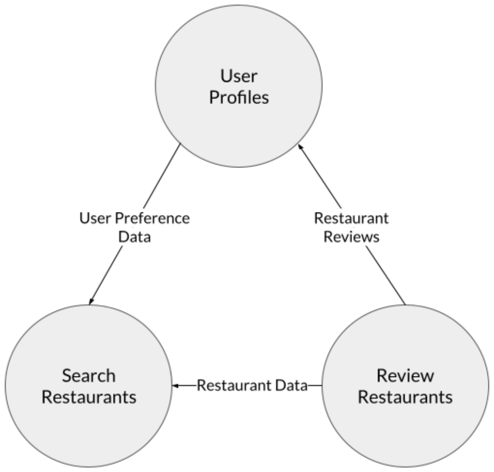
- Restaurant information includes all the restaurant information given from Yelp Fusion API


Check Your Profile
- see your bookmarked restaurant
- see your reviews
- see your preferences
EurekaEats User
- select preferences for better search quality
- type a restaurant name or what type of food they want to the search bar
Restaurant selection
- display restaurant based on search
Individual Restaurant
- display the information for the individual restaurant

2.3 Product Functions

Users should have the ability to use these functions on our website.

- User Profiles - Users can create an account where their eating preferences, restaurant reviews, and favorites will be saved.
- Search Restaurants - Users can search for restaurants based on the preferences voluntarily submitted to their profile.
- Review Restaurants - Users can review a restaurant on the website to let other users be more aware of the foods they serve and the level of service.



2.4 User Classes and Characteristics

- Restaurant Owner - Wants to claim their restaurant and list their food
- Power Reviewer (foodie type/yelp critic) - Reviews restaurants often.
- Casual User - Mostly searches for restaurants and sometimes may review them.

2.5 Operating Environment

The technologies used on this platform will be platform-independent. Specifically, any desktop or server-oriented operating system should be able to run high-level program languages independent of any hardware or operating system.

2.6 Design and Implementation Constraints

Items that can influence the ability of the software developers to implement the product would be system overload due to the number of users using the website.

Give a general description of any items that will influence the ability of the software developers to implement the product. These can include things such as:

- Interfaces with other applications
  - The website will use 3rd party APIs such as the Yelp API for restaurant data and the Mapbox API for location data. We need to comply with their API usage and terms of service.
- Reliability requirements:
  - The software must be reliable in performance and uptime. Page operations should take at most 3 seconds, and the website runs should not crash more than 10% of the time.
- Safety and Security Considerations:
  - Since user-inputted data is unverified, the application must validate it based on the data format, the accepted data values, and the message format. For formatting,

poorly formatted messages and data sent to or from the website could cause errors or crashes if not handled. Also, validating data for appropriate values ensures that the data fits realistic constraints: star ratings cannot be negative, etc.
    - Since passwords are crucial for users to access their profiles, storing them as plain text will allow any database breach to expose sensitive data. If attackers somehow access the database and steal passwords, they could hijack user accounts or expose their personal information. Thus, we will consider using string hashes to protect passwords.
- Memory Constraints
    - Other programs may run on test machines for *localhost* software versions, so the software must not take excessive RAM. A possible constraint can be that the software should not have an out-of-memory error.

2.7 User Documentation

User documentation will be provided on the website through tooltips, intuitive design, and likely an FAQ page.

2.8 Assumptions and Dependencies

The cost of API calls can change, and free products can become unavailable. If a service we use (such as Mapbox API) changes pricing, we may need to change our strategy.

3. External Interface Requirements

3.1 User Interfaces

User interfaces will be designed in Figma. Our team will upload the design to match our website once the design is completed. The user interface will be set up as a website, and we will store user inputs into our database. Once inputs are collected into our database, we can use those data to run functions for our website. Website font and size must be clear for the users to see and understand.

3.2 Hardware Interfaces

This software does not have hardware interface requirements.

3.3 Software Interfaces

Browsers & Versions

- Chrome V117.0
    - https://www.google.com/chrome/
- Firefox V118.0
    - https://www.mozilla.org/en-US/firefox/new/

APIs

- Yelp Fusion v3
  - https://fusion.yelp.com/
- Mapbox GL JS `v2.14.1`
  - https://docs.mapbox.com/mapbox-gl-js/guides/

Database:

- MongoDB 7.0
  - https://www.mongodb.com/try/download/community-edition

Programming Languages:

- Python 3.12
  - https://www.python.org/downloads/release/python-3120/?ref=upstract.com
- Javascript: ES6+

3.4 Communications Interfaces

Network protocols:

- Application: HTTP/1.1 (common on local test servers)
- Transport: TCP (almost everywhere) & IP addressing (everywhere)

4. Requirements Specification

4.1 Functional Requirements

User Registration:
- The system shall allow users to create an account with their email address, a unique username, and a password.

User Profile:
- The system shall make a profile page for users to manage their personal information.
- Users can change their personal information, add profile pictures, view/edit their reviews, and view their bookmarked restaurants.

Restaurant Listings:
- The system shall display a list of restaurants with details such as name, location, service hours, cuisine type, average review score, and cost.

Restaurant Pages:
- The system shall have a page for each restaurant displaying detailed information, including photos, menus, and reviews.
- The restaurant page will have a spot for users to submit reviews for the restaurant.

Search and Recommendations:
- The system shall have a search functionality that allows users to find restaurants based on location, cuisine, and price range.
- The search system shall find the most relevant results within the user’s preferred distance range.

User Interaction:
- The system shall allow users to bookmark restaurants they are interested in.

- The website will have a section to show users a list of all the restaurants they have bookmarked.

4.2 External Interface Requirements

User Interface:

- The system shall provide a user-friendly, intuitive, easy-to-navigate graphical user interface (GUI).
- The GUI shall be accessible and comply with the four principles: perceivable, operable, understandable, and robust.
- The system shall support user interactions through input devices like keyboards, mice, and touchscreens.

Application Programming Interfaces(APIs) :

- The system shall integrate with the YelpFusion API to pull restaurant information for the Los Angeles area.
- The system shall integrate with Mapbox GL JS API to create a visual map with a line that acts as a direction to connect EurekaEat's users to the desired restaurant.
- The system shall provide appropriate security measures such as a .env file to hold API keys and MongoDB connection string.

Data Integration:

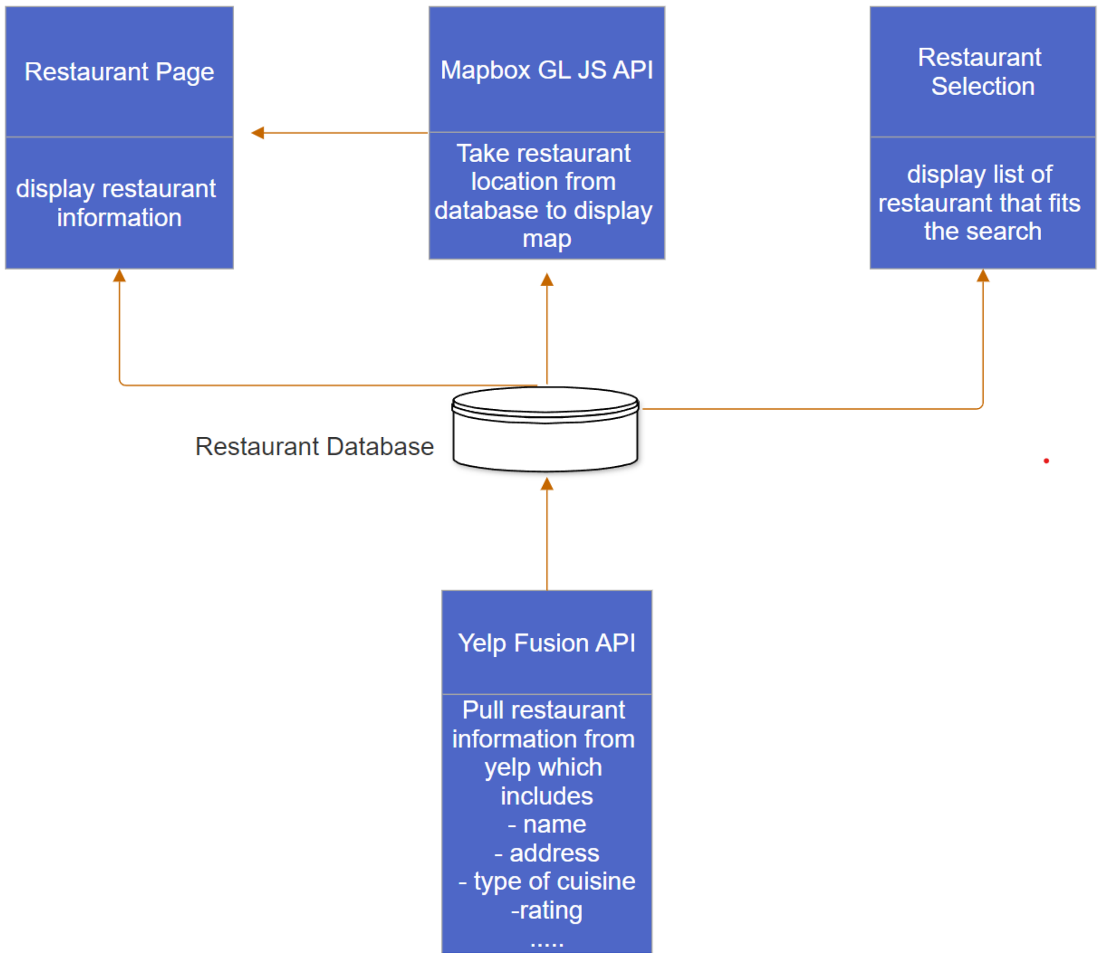
- The system shall have one computer as the local server to ensure that requests made to the website are consistent with the current database, as the database is not on a cloud.
- The system shall be cloned after merging a new pull request to Git Hub.

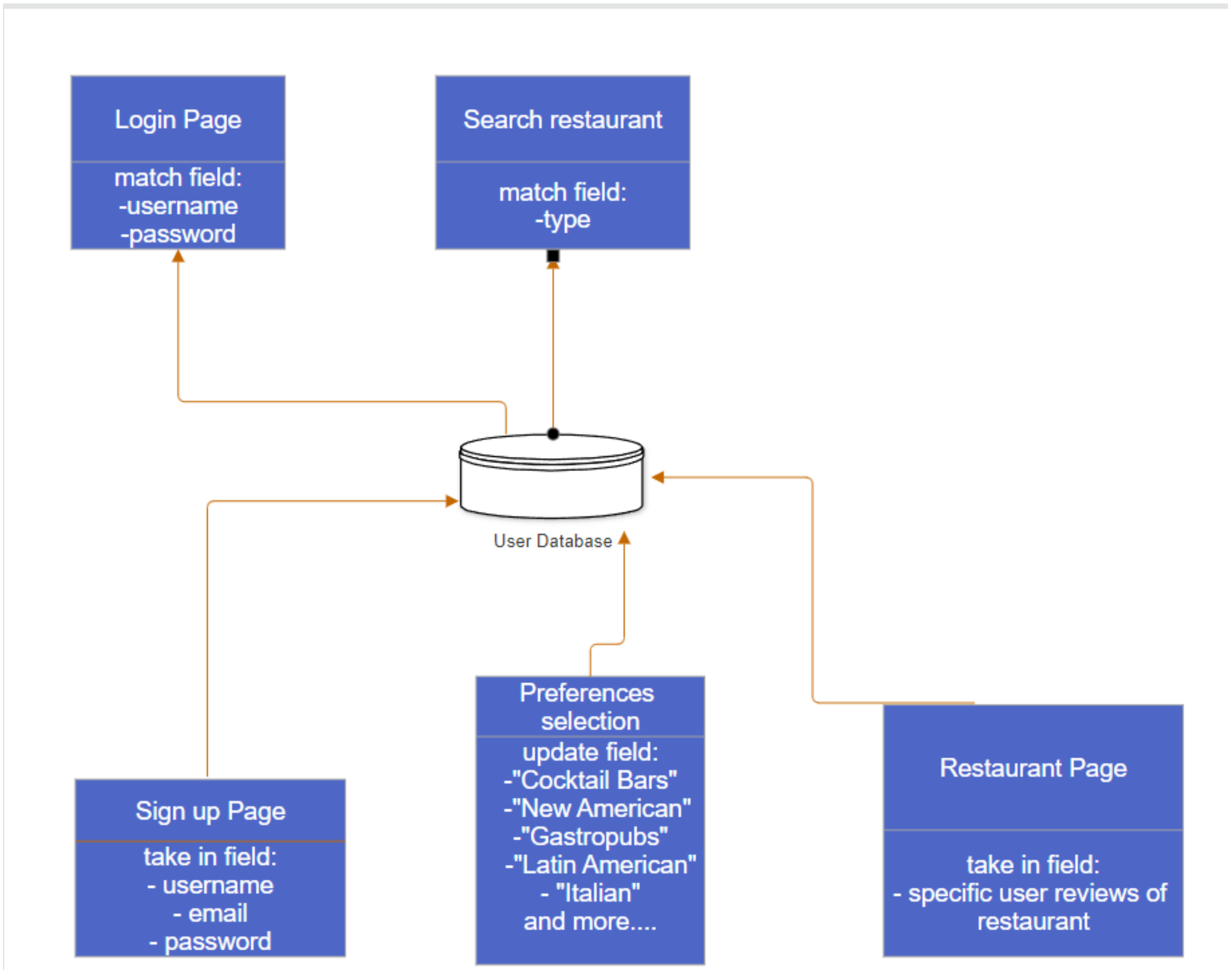
Communication Protocols:

- The system shall support standard communication protocols, such as HTTP, HTTPS, and TCP/IP, for exchanging data with other systems or devices.

4.3 Logical Database Requirements

The MongoDB database shall consist of two primary collections, "users" and "restaurants." The relations between the database and the functionalities of our website are displayed below. Check the database design from the SDD for more reference.


Restaurant Page
display restaurant information
Mapbox GL JS API
Take restaurant location from database to display map
Restaurant Selection
display list of restaurant that fits the search
Restaurant Database
Yelp Fusion API
Pull restaurant information from yelp which includes
- name
- address
- type of cuisine
-rating
.....



4.4 Design Constraints

APIs Restraints:

- The system shall only allow 50,000 requests made to Mapbox GL JS
- The system shall only allow 500 requests a day to the Yelp Fusion API

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The number of simultaneous users will be up to 1000, which will require the following hardware specifications:

- Memory: 16 GB
- CPU: 16 core
- Disk: 100 GB

5.2 Safety Requirements

Possible loss, damage, or harm: EurekaEats will include safeguards to prevent data/security leaks using tools already available via MongoDB; this is to ensure personal data remains confidential. Mechanisms to prevent data loss include regular scheduled data backups in the case of a system failure.

5.3 Security Requirements

Use a .env file to store API keys and MongoDB connection strings.

User identity authentication requirements: Strong password policy (must be over ten characters with one special character), captcha, not a robot check.

5.4 Software Quality Attributes

Straightforward UI for usability prevents users from re-learning how to navigate the product. Emphasis on ease of use over ease of learning. This will be done by having basic functionalities/buttons the user has seen before in other applications. In addition, the layout of the screens will be pleasing to the eye so that users can find what they are looking for.

Well-maintained code: EurekaEats code will be documented thoroughly so that any updates to the software will not result in system failure, in addition to having easily readable code (variable, function, and class names are clearly named). Having well-maintained code leads to a more reliable product.

5.5 Business Rules

Main engineers (group members) are considered admins of the product. They will have access to the database (CRUD functions) and code for the product.

Communication between users of the product and the admins can be done with a report button through the product.

6. Legal and Ethical Considerations

We will reference data or code used within this project that we did not develop. Credits will be in documentation, package manager lists, or code comments. Also, we should put disclaimers for data taken from 3rd party interfaces, e.g., APIs.

Appendix A: Glossary

See section 1.4 for all abbreviations or special terms.