HL DESIGN

Team No Stress
10/06/2021
Daniel Ramirez
Hunter Lewis
Luke Sunaoka
Matthew Quinn (Team Leader)
Michelle Vo
Tyler Thorin

.

1 Project Overview

1.1 Purpose

The High Level Design (HLD) provides an analysis overview of our project, What the Food?, and breaks down its overall functionality by components.

1.2 Audience

The High Level Design's main target audience is the project group members Daniel Ramirez, Hunter Lewis, Luke Sunaoka, Matthew Quinn, Michelle Vo, Tyler Thorin.

1.3 Tech Agnostics

In order to ensure that no errors will arise due to the implementation or change made within our software, we will be applying orthogonality which allows us to make changes to one factor of our code without it affecting other components. Orthogonality allows for independence amongst different components which will allow each member of our group to be assigned to one section of a project without any major holds.It is important to note that any code made by members should not be limited to any specifics and offer flexibility/compatibility given any language or data input. Furthermore, we can ensure if a module was changed which ends up crashing our program, we simply need to change our newly added module without concern for other aspects due to the independence that has been established.

2 Requirements

- To provide users with easy access to a large amount of information about the products they consume.
- Provide users with the information they need to make more educated choices
- Give users tools to discover new products that better fit their needs and lifestyle
- Help users avoid products containing ingredients they believe to be unhealthy, or otherwise wish to not consume.
- Quick load times and easy access to core features
- Intuitive UI that just about anyone can pick up and understand
- Comprehensive SQL Database containing many of the most popular food products that we expect to be scanned

- Centrally located Web Server to minimize latency to the user

2.1 Proposed Solution

- Through the use of HTML, CSS, and React JS, What The Food? Will provide simple transition amongst each feature as well as accessing the back end of our application

2.2 Capacity Planning

- We will separate the web server database and the SQL database server in order to minimize strain on our hardware through over consumption of resources.

3 Architecture

3.1 Web Application Architecture

- What The Food? is a web application and more specifically a progressive web app that will support Google Chrome exclusively. Users will be able to access the app with or without a login but with varying features amongst the two.

Front End

3.2 User Interface Layer

- The user interface layer will be a layer in which the user can access What the Food? and its many features. This layer allows the user to log in or access the application as a guest and from there access the numerous features provided by What the Food? Features provided by the web app include user input and scanning, aggregated news catered specifically towards the user, history of what the user has scanned or searched in the past and reviews on products of choice.

Back End

3.3 Entry/Controller Layer

- The Entry/Controller layer provides security to our application. In this layer we implement any security that will prevent breach of data as well as any malicious attack targeting What the Food? Furthermore, this layer also validates user input and makes sure that any input made by the user is accepted within our standards, which is listed in section 4 under Standards.

3.4 Business Rule Layer

- The Business Rule Layer applies any business rules that our team decides to enforce during the development of our application. This layer should serve as an overall abstraction for any business rule we implement and provide enough abstraction such that these rules can be enforced across all languages and implementations made in the future.

3.5 Service Layer

- The Service layer represents each of our stand alone operations.Features provided by What the Food? include account authorization, history of a user's search, food information about a specific product, scan/search feature of a specific product, the ability to upload reviews and pictures of products to our database if it did not exist previously, and aggregated news catered specifically towards the user.

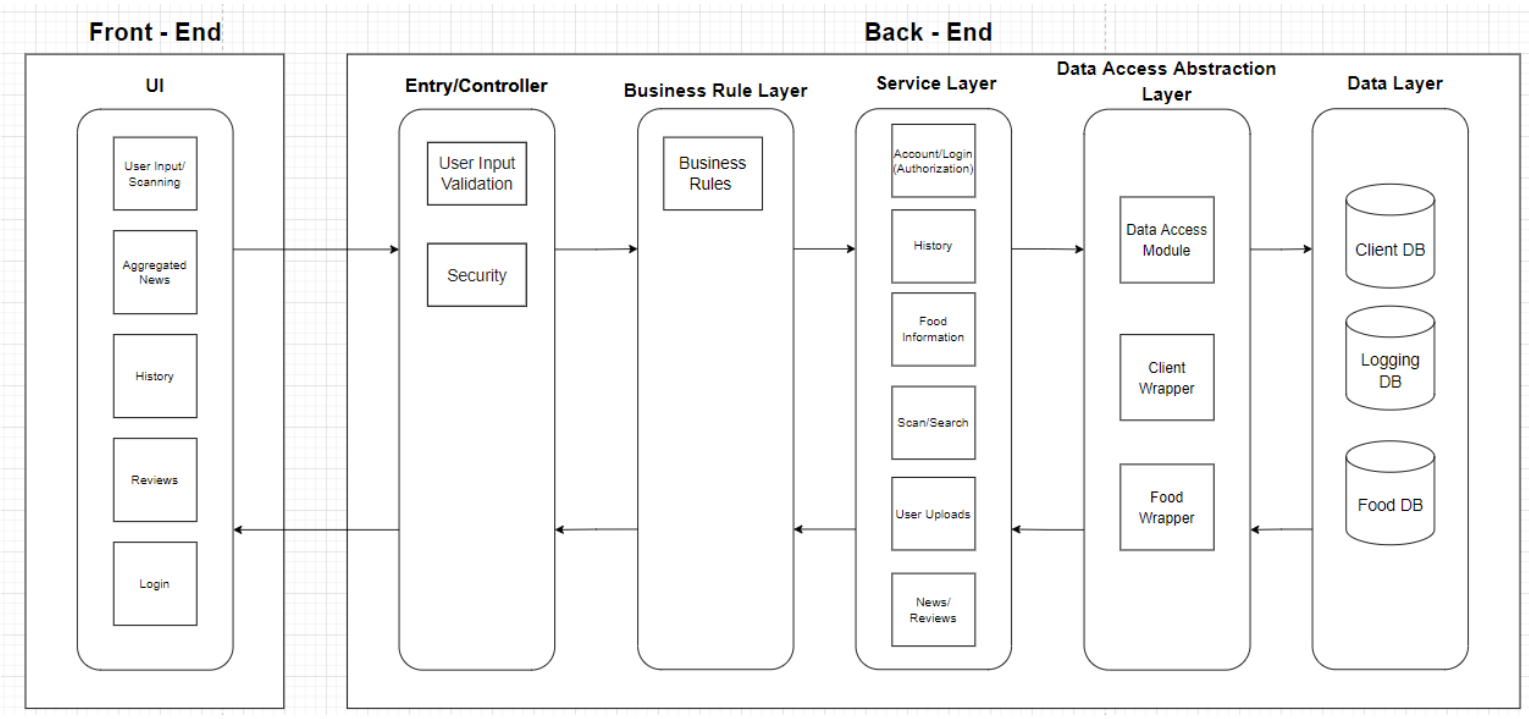
3.6 Data Access Abstraction Layer

- The data abstraction layer creates a layer of abstraction between each microservice and the actual data stored in our database. This layer is necessary in order for a microservice to not directly have access to our database. Furthermore, having this layer provides flexibility towards adaptation if we choose to change the language of how we access our data. This layer will also account for verification on whether certain components exist within a server.

3.7 Data Layer

- The data layer contains all information regarding our clients as well as the products that the product presents. This layer has two separate databases for the client and food products. The client database contains all user information such as account name, password, and preferences. The food

product database on the other hand contains the name of the product, picture of the product, as well as any review related to the product.



3.8 Hardware and Platform Requirements

- Front End
  - Progressive Web App
    - What languages we Use
      - SQL
      - HTML 5
      - CSS
      - Java Script
    - Framework
      - Vue
- Back End
  - Web Server
    - JavaScript
  - Database Servers

        - 10.6.3-mariaDB

3.9 System Connectivity

- The user will be able to connect to the app using only Google Chrome from any device. Compatibility will be ensured for Chrome on Windows, Android version 94.0.4606.61, or iOS with Google Chrome version 94.0.4606.52. Furthermore, if an update occurred that deprecated an essential component, or the user's installed version is too old, the user will be prompted to change to a version of chrome that our application supports.

4 Standards

- In order to ensure that our software works as intended and does not run into any unwanted errors, we will be using Lighthouse which is a PWA analysis tool that rates our application. By using Lighthouse, we will understand what components of our application are weak and apply our corrections accordingly.

4.1 Security Standards

- In order to integrate a proper security standard for our application, we will be applying Secure Scrum as our main secure methodology. Secure Scrum has a, "...special focus on the development of secure software throughout the whole software development process (Pohl & Hof,2015)" which perfectly aligns with our group's work process. During each Sprint, each member will list an identification component which is an indicator of any potential security concerns within the sprint backlog. Each identification component will be ranked by urgency which will identify our team what needs to be fixed. Listing our identification components by urgency within our sprint backlog will also allow for other members to be aware and aid the group member if they are unable to find or fix this specific security concern. By being able to constantly observe any potential flaws or security concerns during our implementation period, we can uphold a standard of security with consistency and frequency while also keeping the team aware of these concerns.

4.2 Error Handling/Input Validation

- In order to account for any errors that will arise within our application, we will implement proper exception handling that will account for all errors

that may arise. Our team will implement exception handling such that we will pick up on the error and notify the team what the error exactly was so that we can fix it accordingly.

- Input Validation will be accounted for to ensure that any user input made is correct and confined within our application's limitations. To ensure this, we will apply whitelist validation which only accepts input that we the programmer deem acceptable. Through the use of whitelist validation, we will ensure that inputs have the content we expect which includes and not limited to, data type, data range, data size, and data content. If an input were to fail validation, we will resort to our disaster recovery mentioned in section 4.3 down below.

4.3 Disaster Recovery

- Upon crashing or any unexpected error, the application will return to its last safe state. As well as present the user with a notification explaining that an error occurred.
- In the event that the application is not able to return to a safe state, the application will instead inform the user that the application is currently unavailable and to try restarting or to come back later.