SENG2021 Requirements and Design Workshop

Deliverable 2

Boba Me: Technical Conceptualisation

By The Boba Engineers

Navid Bhuiyan (z5260384), Jing Deng (z5213505), Sahibreet Bassi (z5112643), Aaron Shek (z5309113) and Austin Walsh (z5311341)

Contents

Part 1 - Software Architecture

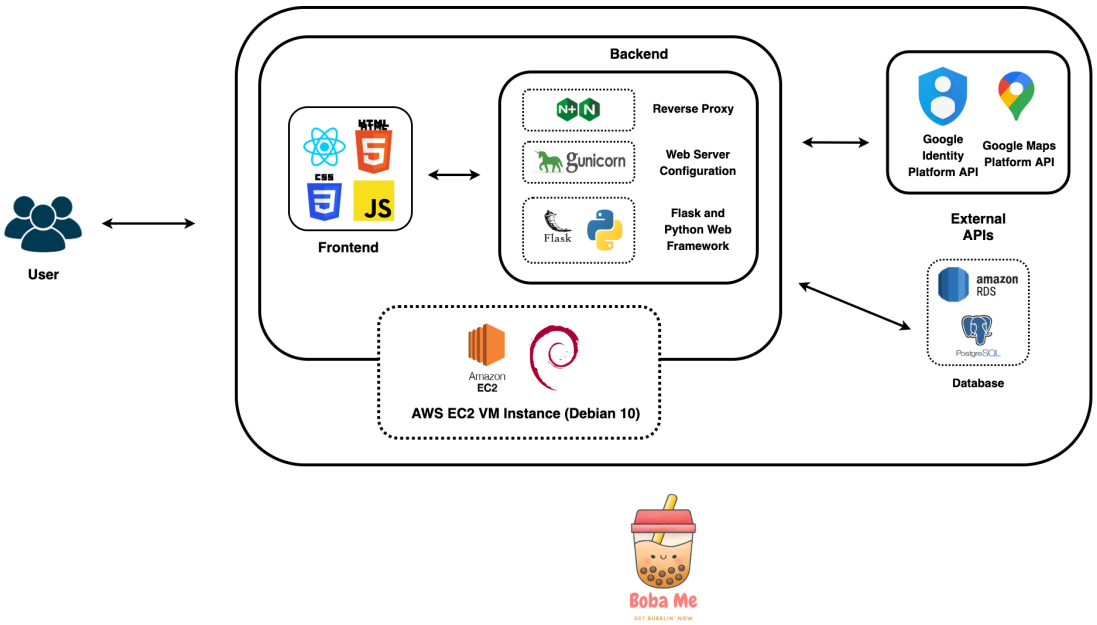


Figure 1.1 - *Software Architecture Diagram (SAD) for Boba Me*

1.1 Summarising the Architecture

The Boba Me web application is based on software and hardware architecture which will allow for a responsive, reactive, small, and yet simple web application for a great user experience. The architecture also allows for future scalability and expansion if the application is set to be launched and expanded.

As observed in Figure 1.1, our software architecture has been optimised around utilising AWS as our cloud hosting solution. Our virtual machine aka the web server, is based on Debian 10, a popular Linux distro. All of the frontend and backend software are compatible with this version of Linux, and listed in the table below, is an extended overview of each component illustrated in our SAD for Boba Me detailing its use and software caveats.

Table 1.1 - Extended Overview of SAD		
Architectural Component	**Software Component**	**Software Details**
Cloud Hosting	AWS EC2	VM service to run our web server on. Will likely use the t2.micro instance. Refer to 1.3 for further details.

	Debian 10 OS	Underlying OS for our web server VM. Popular lightweight linux distro, which is compatible with all popular web development software.
Frontend	Javascript	Javascript is one of our frontend languages to connect to our flask API and backend. We will be using Node.js package manager which runs on ECMAScript 2015 (EC6) which is widely used and compatible with react.
	React	Javascript library to add specific UI functionality to our frontend such as adding animations to our web app.
	HTML	Creates our template web pages which our Javascript will manipulate and manage. We will be utilising HTML5 for our web app.
	CSS	Styles our web pages and will be managed by our Javascript code to ensure our web app compatibility with multiple screen sizes. We will be utilising CSS3.
Backend	Flask and Python	This will lay our underlying API and backend architecture to pass on information valuable for our user to the frontend. We will be utilising Python 3.7.3 and Flask 1.2.1 (Debian OS 10 has Python 3.7.3 installed, and although the latest of Python can be installed, to maintain utmost stability and performance, it is best to run with the pre-installed Python package provided by the OS).
	Nginx	Nginx will be the reverse proxy used to redirect requests to Gunicorn (web server) which hosts flask. Nginx version 10.x will be used as it is compatible with Debian

		10.
	Gunicorn	Gunicorn will be the web server which serves the flask app to Nginx. Gunicorn version 20.1.0 will be utilised (compatible with python3.5+)
	Amazon RDS: PostgreSQL	Amazon RDS will be our database provider as it runs on the cloud, and requires less set-up and toll on our web server. This will also host data we shall scrape about shops. PostgreSQL will be used as it is powerful for post-production servers with in-built functionality we hope to utilise.
External APIs	Google Platform Maps API	One of our features will be to direct people to Bubble Tea shops, so this API can provide an easily injectable map embed. We also will use the API to scrape and build our own database for shop information.
	Google Identity Platform API	This will allow OAuth2 capabilities on our website to generate new account information

There are however other points about the application which services will need to be internally developed as there is no current Bubble Tea database to extract from, which hasn't been addressed by the table. Also, there are caveats for our use of the Google Platform Maps API.

External Data Sources for Inbuilt Applications

To currently note, we have not yet created ER diagrams as they are yet to be further developed. However below, it is conceptually discussed what sort of data we will be collecting, and how these external data sources will be utilised for our web app.
For a basic overview of our relational database it will likely structured in the following ways:

- Ratings and account information will be intertwined
- Discount information and account information will be intertwined

- All other information will be structured separately for each feature/

Application for Bubble Tea Drink Information

Due to the lack of databases or sites which are plausibly scrapable to get Bubble Tea drink information relative to our needs, we will have to likely create an application which harvests some data for us. As for this iteration, the application will not be capable of auto-harvesting information online as we have not outlined this in our features nor will we be approaching this. Currently we will harvest some data on our own externally to enter into the database. As the scope of this project is mainly to present a workable concept piece we shall not add another feature to entirely create a backend application which harvest data from key websites.

For our own data collection, we will be scraping Chatime, Gong Cha, Sharetea and King Tea websites to create a proto-database for our concept piece for Boba Me. If

For future iterations of this product, it would be great to investigate cloud powered web scrapers, to help build the database of the app, and create an API for businesses to enter their own information if they please too.

Application for Bubble Tea Shop Information

The use of the Google Platforms API, will be integrated into our flask application for custom data inputs and outputs. We shall likely use the API directly for map and direction information for our shop information for our shop pages, and our search results page to find shops near our user's location. It will also be used to create a table within our database, which allows for users to rate shops.

Applications Related to Google Account Sign-In

Using the Google Accounts API, will be used to make it easier for people not having to sign up for an entirely new account, as they can use their own. Using this API, we will be storing our own account information, storing their name, email and GoogleID hash to be able to provide them discounts and give them access to rate shops and drinks.

Conclusion

Summarising the technical and development of the software architecture, where the premise is building a light web app on the cloud which can perform well, as well as maximising the utility of external APIs, having room to develop an in-house data

harvesting app. However there is a bigger rationale to why we specifically chose this tech stack.

1.2 The Rationale

Flask

Flask is a micro web application framework. It provides libraries and modules that can be used for making all kinds of web applications or services, and is written in Python. The developers in our team are quite familiar with Python so Flask became a very high valued option. Advantages that come from choosing Flask is that it reduces development time and will allow us to build faster, especially with close deadlines that accompany this smaller project. The framework is also known to support many extensions. It allows for fast debugging, which means we can develop very quickly. Flask is compatible with the Google Login feature, and Google Login will be one of our main features in our system when it comes to signing in when using the website.

We chose Flask because of its lightweight and modular design, and is highly flexible compared to other frameworks, like Django. In comparison to Django, the main difference is Flask is unopinionated, so we can decide how things are implemented, and it provides simplicity and fine grained control. Flask is also a Web Server Gateway Interface (WSGI) framework, a calling convention written in Python language for web servers that forward requests to web apps. Flask is the lightweight client side for this convention, and with a focus for choosing lightweight components, we also chose Nginx which will be the server side software for web serving and reverse proxying.

Nginx

We chose Nginx as our reverse web proxy server because it is one of the two most common open source web servers in the world. It is responsible for half of traffic on the internet, with the other half being served by Apache. It has many selling points. It is often selected because it is efficient with resources and responsive under heavy load. It uses an asynchronous, non-blocking, event-driven connection handling architecture, which scales extremely well on generic server hardware.

Nginx extracts the maximum performance out of both physical and virtual servers, and because we have chosen Amazon ECS, this web server became a good choice.

It can also serve at ten times more requests per server compared to Apache. In the end this means there can be more connected users, bandwidth is better used, CPU and RAM are still readily available, all because Nginx optimizes its resource use.

Gunicorn

Gunicorn is a WSGI HTTP server often used in conjunction with a reverse web proxy server and web application. Gunicorn provides a solution for handling anything happening in between the web server and web application, in this case Nginx and Flask respectively. This includes communicating with multiple web servers, distributing a load of many web requests arriving at once and reacting to them, and keeping multiple processes of the web application running.

Essentially, it works by internally handling the calling of Flask code, and it does this by having ‘workers’ ready to handle requests instead of Flask handling requests one at a time. The reason we chose to use Gunicorn is because, when Flask is not designed to handle heavier volumes of traffic a normal web server receives, despite Flask running well for local development. It is very optimised.

Amazon RDS: PostgreSQL

Amazon Relational Database Service for PostgreSQL (or Amazon RDS for PostgreSQL for short) provided by Amazon Web Services, and is essentially, a database running in a computing ‘cloud’ provided by Amazon.

The product is attractive for a few reasons related to performance. Amazon Web Services (AWS) does a great job to facilitate the setup, deployment and scaling, for a relational database in the cloud. However the main reason we chose this was it bypasses the process for setup and installation, and we can get into building the product itself.

Amazon RDS also offers different database engines for use, including Amazon Aurora, PostgreSQL, MySQL, MariaDB, Oracle, Microsoft SQL Server. Out of the list, we chose PostgreSQL because it is feature rich. It can handle complex queries and huge databases compared to MySQL, and is usually the preferred solution to complicated, high volume database design. PostgreSQL can additionally define data types, index types, functional languages, and NoSQL. The fact that it is a ‘relational’ database means the data inserted relates to each other, unlike other NoSQL databases (e.g. MongoDB).

Google Platform Maps API

One of our web application's main features includes providing customers with directions to a chosen Bubble Tea shop when requested. Throughout many applications, Google Maps is the go-to API due to the many features it includes. It provides quick directions and estimates for travel times and distances. Compared to other products, Google Maps is an extremely polished service provided by Google, and is often known as an unofficial king of applications for directions. Due to these reasons, it was very obvious that we had to include this in our web application.

Google Identity Platform API

Another main feature of our web application involves logging into the web application to rate drinks and perform other actions. Our options to add this functionality to our application would be to either create our own forms for each user to enter, or to utilise Google's accounts. Comparing the two options, it became clear that the Google Identity API already provides these services with 'frictionless sign in and sign up'. Most people already have a Google Account and are familiar with the process of logging into Google, so users will feel more at home with Account Integration as a result.

1.3 Relation of Software Components and Languages

Frontend Languages

The language we decided on for the front is Javascript. Javascript is the current industry standard of building dynamic web pages alongside HTML and CSS. React is provided through a Javascript library and allows for the creation of simple yet powerful web pages. React adds UI functionality required to provide an effective user experience, adding animations outlined in user stories. React is additionally useful in its ability to create web pages for both desktop and mobile devices with similar ease. It is a fast, simple and scalable language that will provide web pages able to handle high usage from users from a wide variety of devices.

HTML and CSS will provide the base of the web pages in which React will build upon. HTML is used to create the structure of the pages while CSS allows for the styling of such pages. These languages, in combination with React, create user friendly web pages with a high level of functionality and sustainability.

Whilst many other languages are useful in creating frontend websites, such as Bootstrap or Django, React is a current industry standard that is proven and tested to work. It is also hugely used to build a robust user experience and interface, which is a large requirement for Boba Me. React is known to beautifully animate pages and present information in an intuitive way, which will make it easier for our user to navigate our app. React is also used in many large websites and as such has a wide variety of resources and functionalities that allow for the creation of impressive web pages.

Backend Languages

The main language the group decided to use for the backend was Python, specifically the library, Flask. Python is a widely used, simple to work with language that provides many libraries for a wide variety of uses. One such library is Flask which is a popular web framework. Flask is used for the backend of web pages, it provides a connection between the web page and database and sends requests through its backend or receives requests from users. It is compatible with many request formats such as JSON and XML and allows for handling, modification and returning of these requests all through itself. While it is a microframework, meaning it does not include many standard functionality provided from fully-fledged frameworks, there are many external libraries that can be integrated with little issue to provide these missing services.

While Python will be used to create the backend architecture for our web pages API, it will also provide the connection between our server and database, PostgreSQL through the additional library of psycopg2. PostgreSQL was the database chosen for the web page as it has a high level of cohesion with python because of the library psycopg2. PostgreSQL also provides an impressive variety of features that create a database that is fault-tolerant and scalable to large and small datasets. PostgreSQL will fill the backbone of the data used in the web pages drinks and shops features and will communicate with the web page via the Flask implementation.

1.4 Platform and Deployment Choices

Hosting on Web Browsers, iOS and Android

For the running of the applications by users, it has been chosen to develop a web application which aims to be mobile compatible and fit all major screen sizes. This was done, so that we only need to build up one application for anyone to access,

rather than building specific applications for a different assortment of devices. As we are using HTML, CSS and Javascript to run on the frontend, it shall be compatible with all major web browsers, as these languages are standard in the industry.

AWS EC2 - t2.micro Instance

The platform we have chosen is Amazon Elastic Comput-e Cloud (Amazon EC2). It is a part of the cloud computing platform ‘Amazon Web Services’ (AWS), which is a subsidiary for the tech giant ‘Amazon’. Amazon EC2 allows users to rent virtual computers or computing ‘environments’, to run computer applications, and we have chosen this to eliminate the need for investing in hardware up front. These virtual computing environments, or virtual machines / instances as they are more commonly known, can be created, launched, or terminated as needed. Also with AWS’s large scale cloud infrastructure which can manage high network loads and has scaling capabilities, it was the perfect choice to host our web app.

There are numerous types of these instances, and each has their own configuration for CPU, memory, storage and networking capacity. Out of the many choices we had, we chose the t2.micro instance.

The t2.micro instance is a good choice for us to host our web app, because it is free (and low cost after free tier usage), and provides sufficient capabilities for our designed system. It is a burstable performance instance, which means the CPU performance of our virtual machine has the ability to burst above a baseline when needed. It’s ability to burst is influenced by a CPU credits system, where credits are accumulated when the CPU is idle, and consumed when they are active. The t2.micro instance specifically, earns 6 credits per hour, with a maximum of 144 credits, has 1 virtual CPU (vCPU), with a baseline utilisation of 10% per CPU. For our proposed system, this instance type provides value for money, or in this case, for no initial costs.

Debian 10 OS

Amazon EC2 is accessed using a web based user interface, or through an AWS Command Line Interface. Debian is the most logical choice here as it is reputed for stability, lightweight nature and is one of the oldest and most established Linux distributions, and it is free. It can support almost all programming languages, and offers a massive range of applications useful for programming. Many libraries are developed natively for Linux (especially Debian based distributions). There are also a wide range of tutorials and community support when developers run into problems for this distribution.

In relation to the stack, AWS EC2 and Debian are compatible with all other components of the stack, and were selected to optimise the lightweight nature of the

application we are developing, as Debian comes with the minimum software requirements to run, and we mainly install our software on top of it.

1.5 Key Benefits and Takeaways

Summarising the key benefits and takeaways of our tech stack is that it allows us to build and continually assemble a lightweight, fast, and responsive web application quickly with the possibility of cloud optimising our web application in the future, with no to minimal changes.

React, HTML, CSS, JS and Python in conjunction with Flask, Nginx, and Gunicorn on top of Debian, shall allow us to build a powerful application for Boba Me. The combination of these software, allows us to quickly design, template, test and host our application on the cloud without hindrance. As we have also chosen open-source, stable and robust pieces of software throughout our entire web architecture, maintaining the stability and overall performance of our web application will be easier than making our own custom software and services. As all of the software we use is community supported and updated over time, we won't be likely losing any software support nor use any deprecated pieces of software.

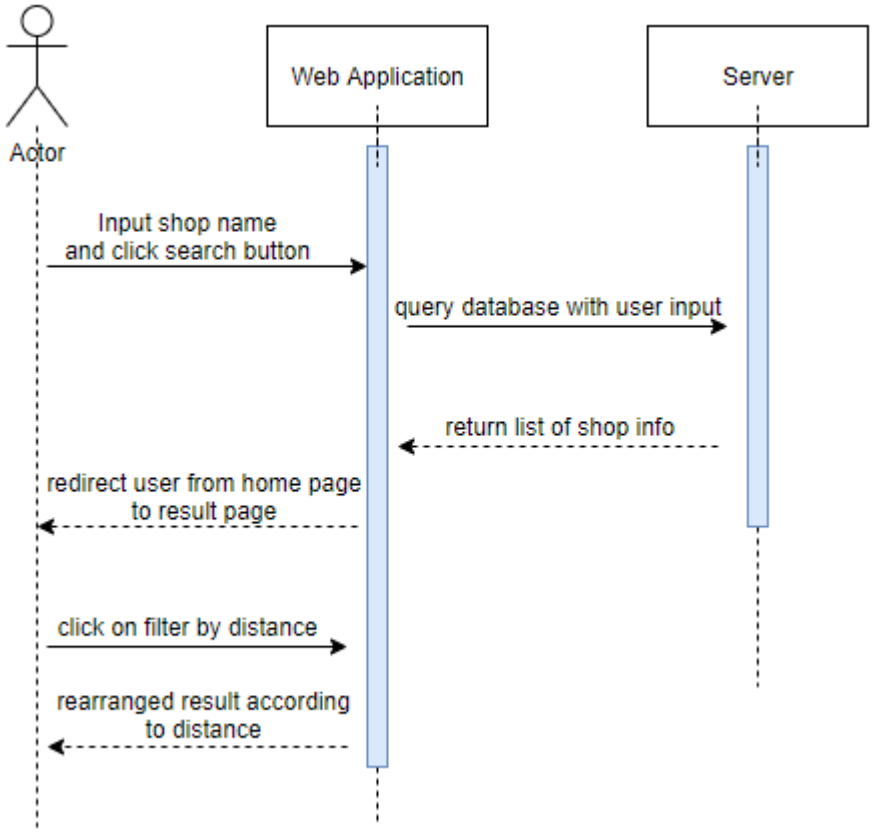
Utilising AWS EC2 to only handle the backend and frontend, we only need to make a container runtime top to bottom, likely using Containerd, and then utilise AWS AKS (Amazon's Kubernetes Service) to automate the distribution of our container loads, disable and create new containers to always meet consumer demand, and minimize overall cloud hosting costs. Also with this design, we do not need to solely rely on AWS, as other cloud providers such as Microsoft Azure and Google Cloud Platform can easily and distribute our app to add an additional layer of runtime security if any of our cloud hosting services gets disrupted.

Part 2 - Initial Software Design

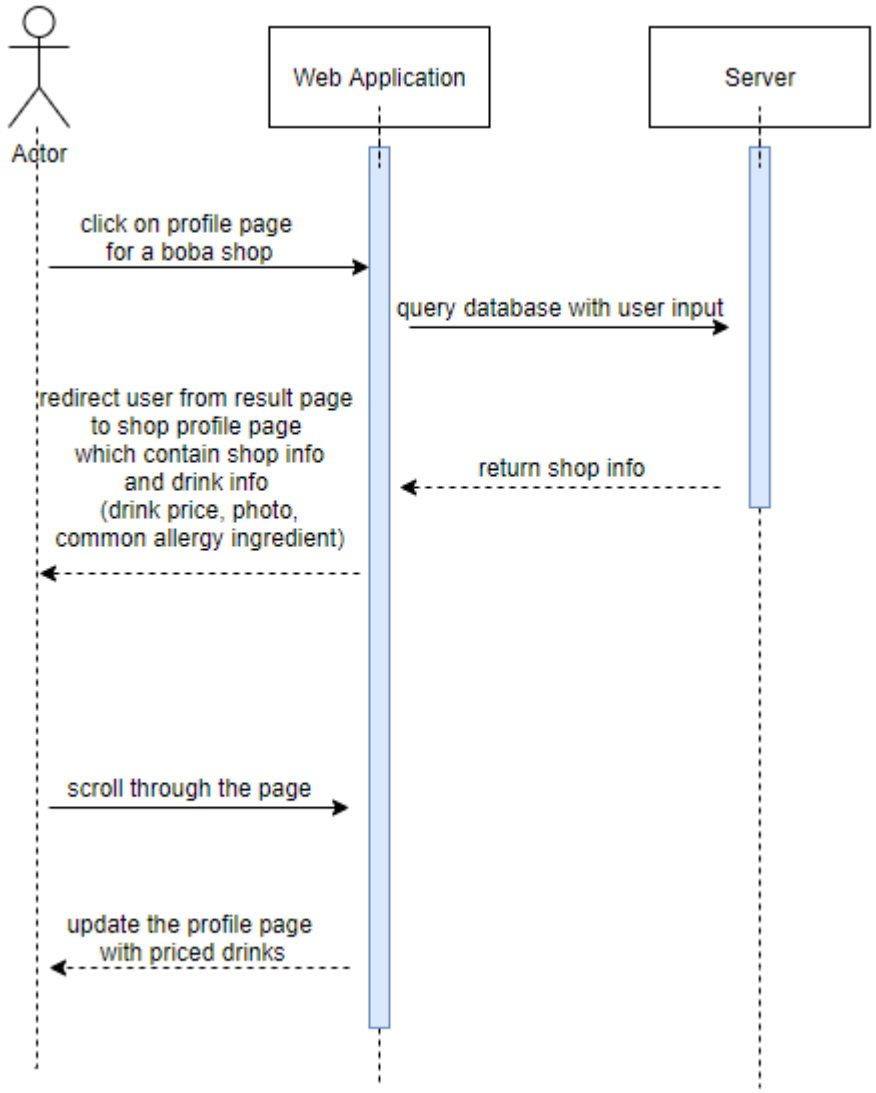
From our initial software design since Deliverable 1, we have updated each use case to be more straightforward. Additionally 2 user stories have been added concerning the filtering and rating for shops. For Deliverable 2, we have presented actors into UML diagrams to explain every different interaction occurring in our features.

2.1 Illustrating Boba Me's Interactions

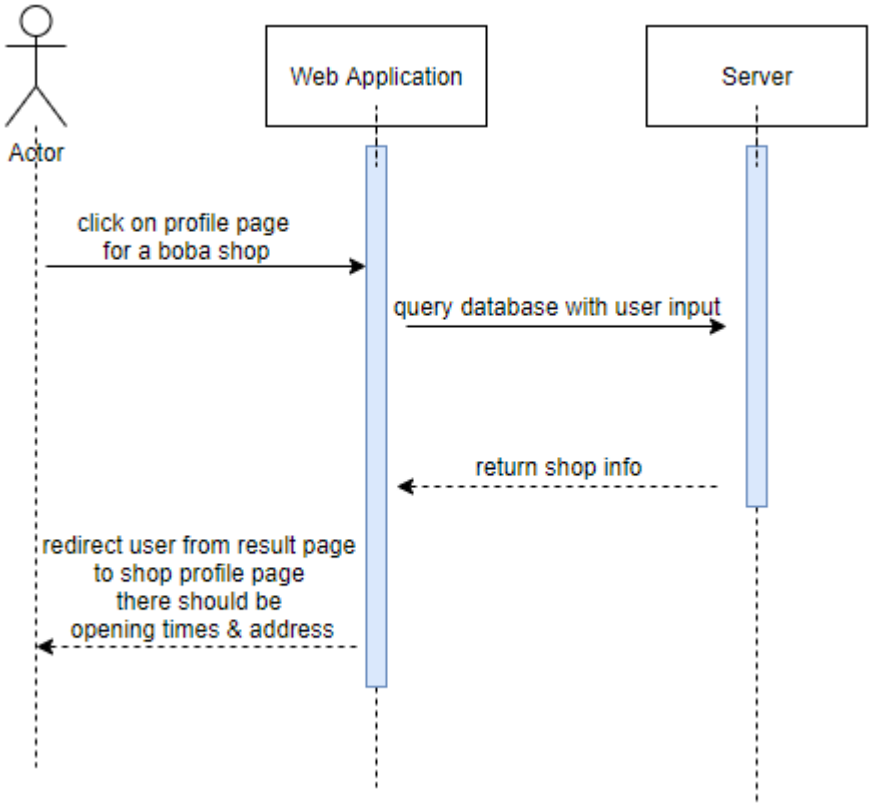
Feature ID	1
Feature	As a client, I should be able to search for boba shops nearby, so I can find boba shops even in places that I am not familiar with.
Scenario	Search nearby boba shops according to distance from the client
	GIVEN I am on the homepage WHEN I type in a shop name AND I press the "Search" button THEN I should be taken to "result" page THEN I should be able to see shops containing the user input THEN I should be able to see the filter tools on the side AND I can choose to filter the result by distance THEN I should be able to see all nearby shops in ascending distance order in grid style.



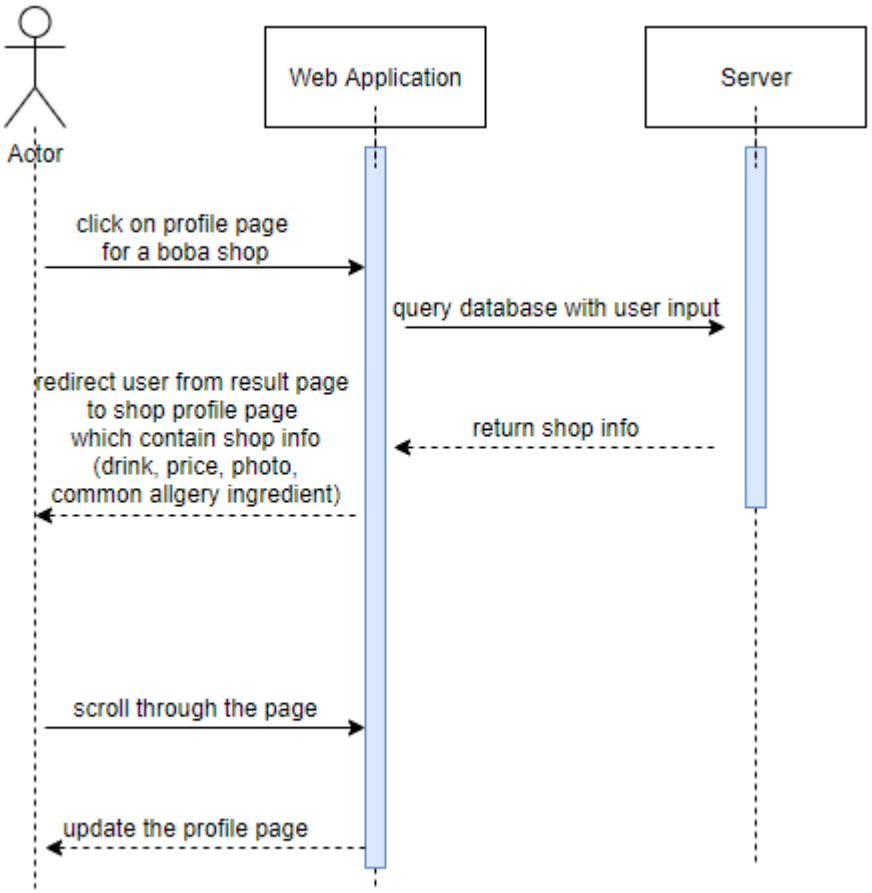
Feature ID	2
Feature	As a client I should be able to view the menu of a particular boba shop, so I can preview what I can get from the shop
Scenario	Viewing the drinks offered by a particular shop
	GIVEN I am on the “profile” page for the particular shop THEN I should be able to see a list of drinks with prices offered by the shop.



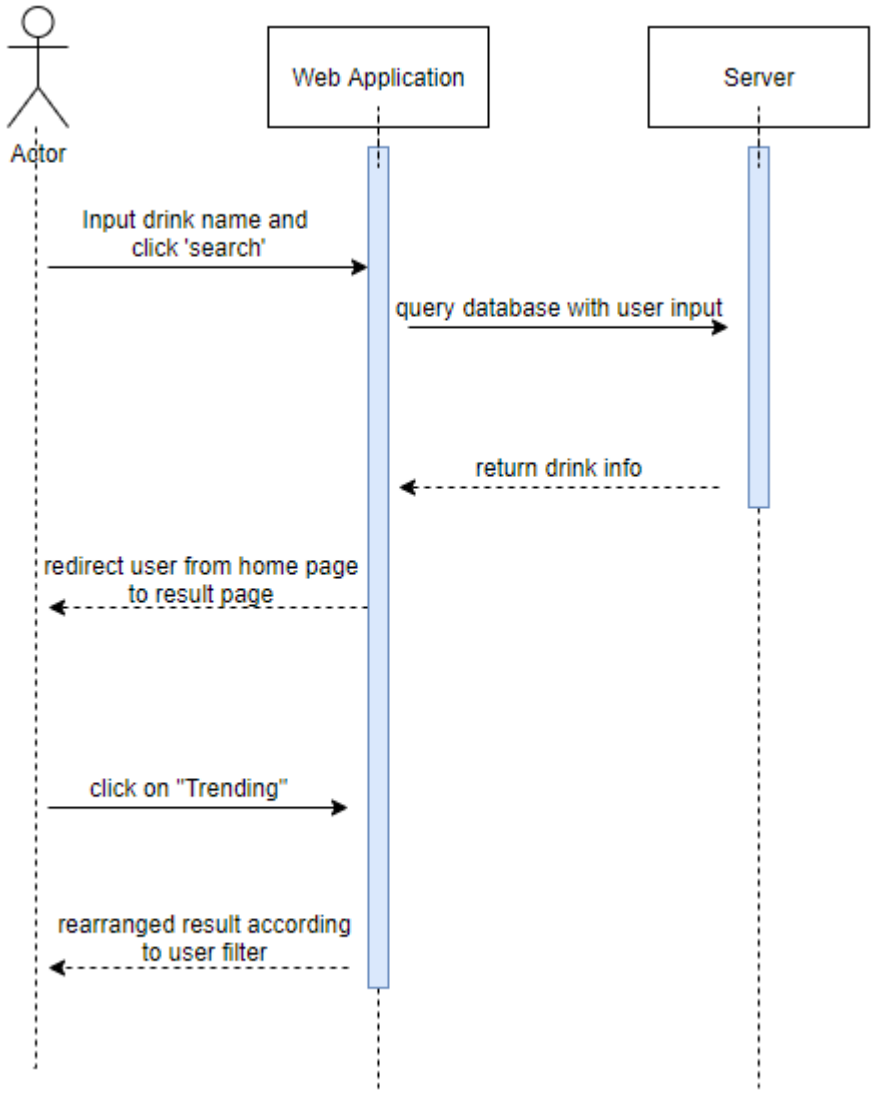
Feature ID	3
Feature	As a Client I should be able view the location and opening information of a specific shop, so that I don't get lost when I want to go to the shop.
Scenario	Viewing the general information by a particular shop
	GIVEN I am on the "profile" page for a particular shop THEN I should be able to see a picture of that store. GIVEN I am on the "profile" page for a particular shop THEN I should be able to see the opening times of that store. GIVEN I am on the "profile" page for a particular shop THEN I should be able to see the address of that store.

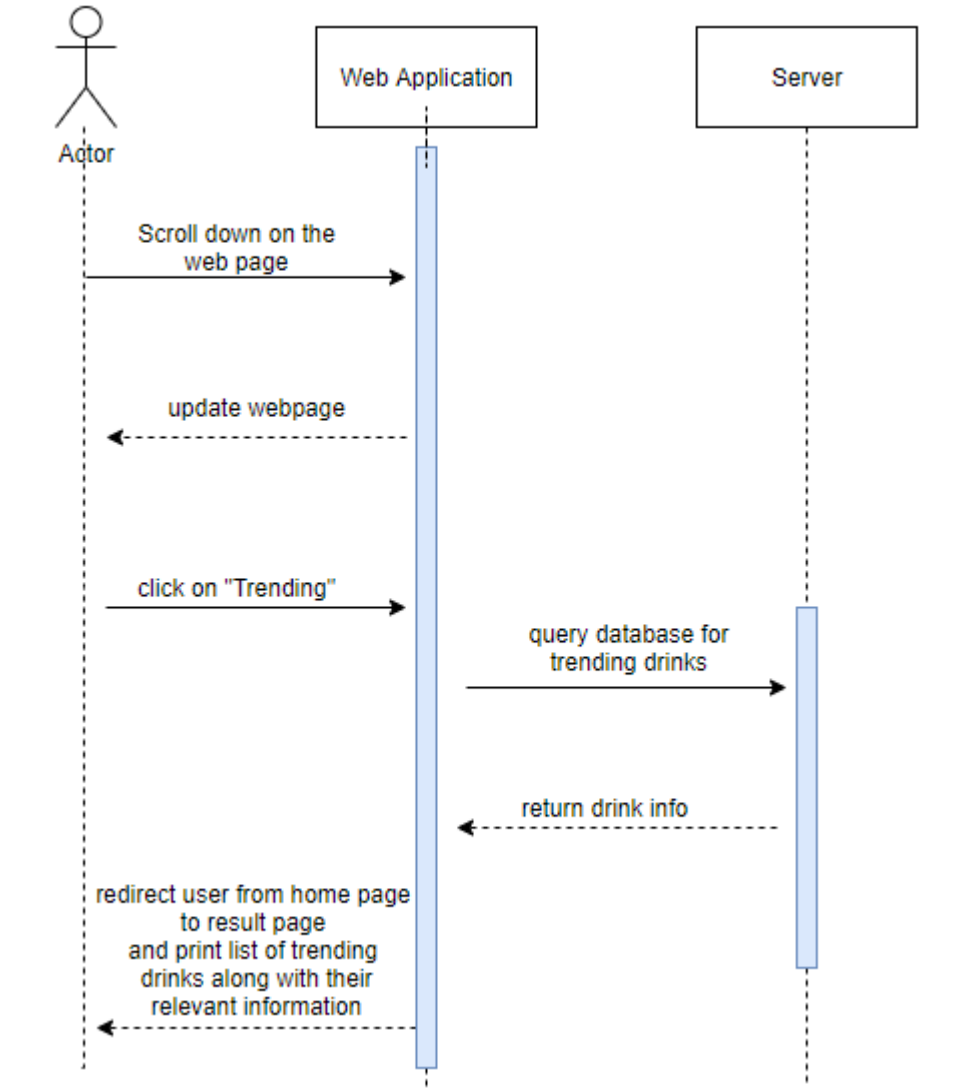


Feature ID	4
Feature	As a Client I should be able view the main ingredients of a particular drink, so I have a brief idea what's inside my drink.
Scenario	Viewing the ingredients contained in a particular drink
	GIVEN I am on the "profile" page for a particular drink THEN I should be able to see list of ingredients contained in the particular drink AND I should be told if common allergy ingredients are used AND I also should be warned to contact shop for detail ingredients

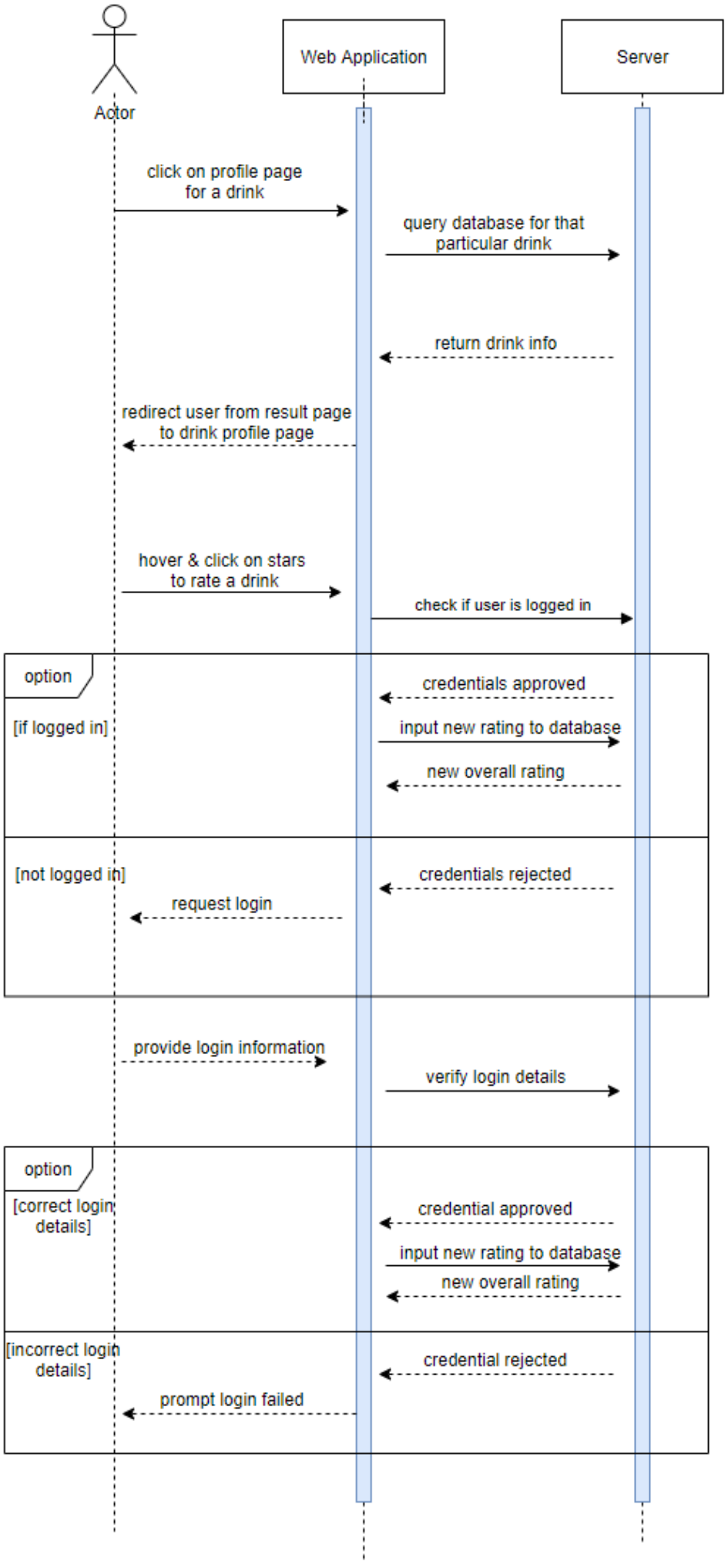


Feature ID	5
Feature	As a client, I should be able to view the trending/popular drinks among other users, so it can help me to make a decision.
Scenario	Viewing the trending drinks among other users when clients can not make a decision
	GIVEN I am on the search result page for drinks WHEN I click on “Trending” button in the filters section THEN I should be able to see a list of trending drinks from my search GIVEN I am on the homepage WHEN I scroll to the bottom of the page THEN I should be able to see a list of all trending drinks

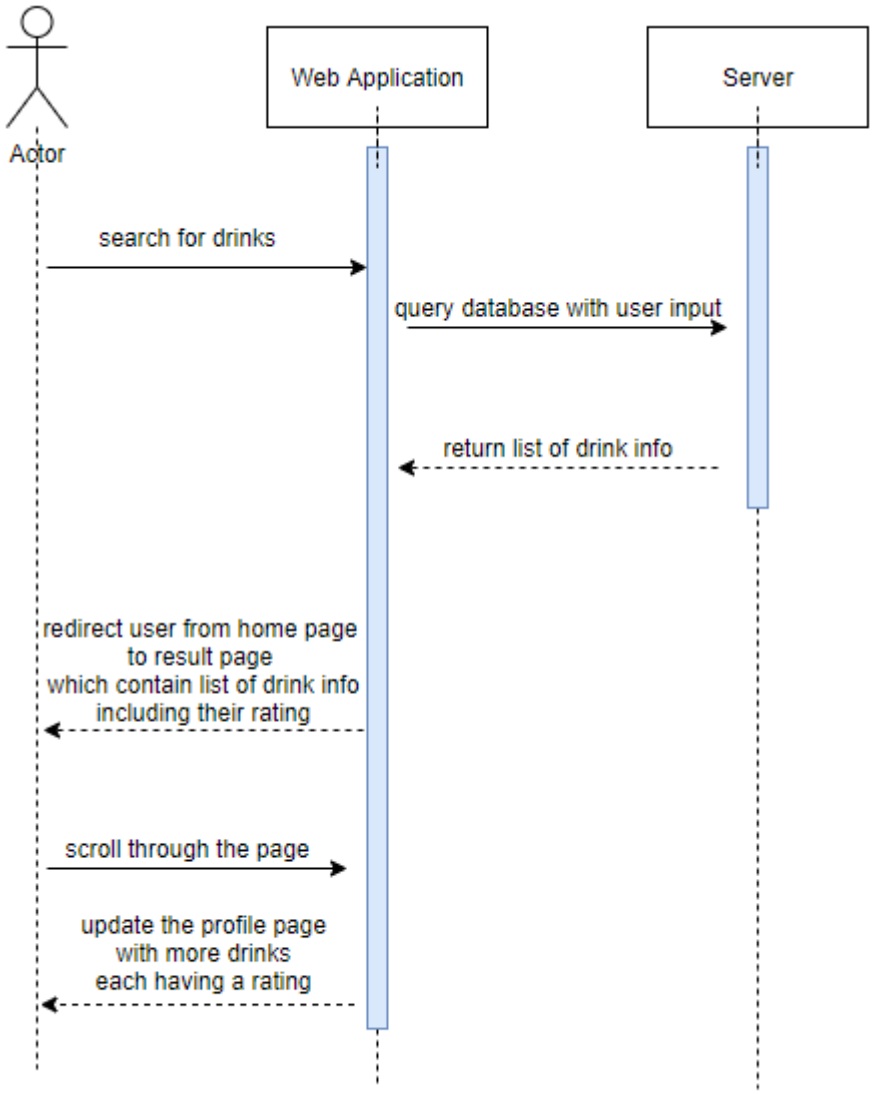

Actor
Web Application
Server
Input drink name and
click 'search'
query database with user input
return drink info
redirect user from home page
to result page
click on "Trending"
rearranged result according
to user filter


Actor
Web Application
Server
Scroll down on the web page
update webpage
click on "Trending"
query database for trending drinks
return drink info
redirect user from home page to result page and print list of trending drinks along with their relevant information

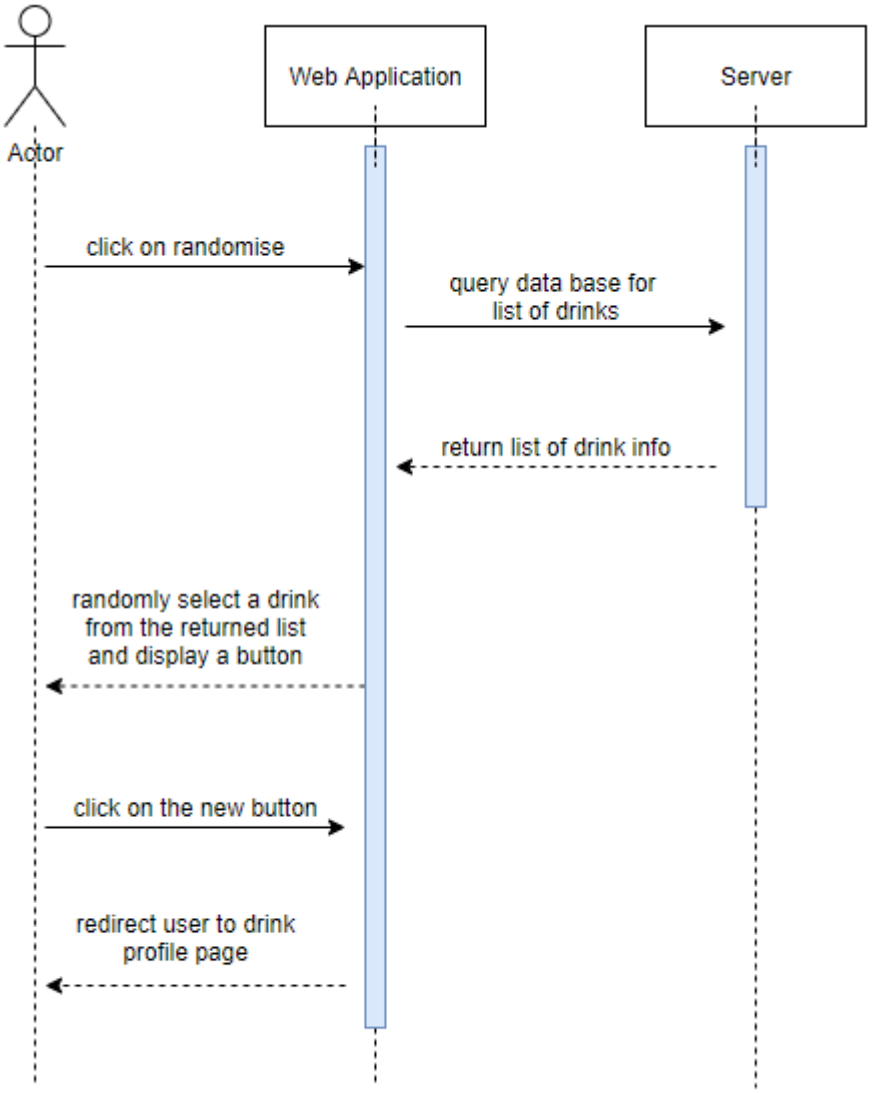
Feature ID	6
Feature	As a client, I should be able to rate a drink out of 5, so I can help other users to make decisions when they are purchasing drinks.
Scenario	Clients can rate a drink after they tried it
	GIVEN I am on the “profile page” for a drink THEN I should be able see the rating of the drink WHEN I hover my mouse over the stars THEN I have the options to click and rate the drink myself IF I am logged in THEN the overall ratings should change according to my ratings OTHERWISE I should be prompted to login before rating IF I provide login information after the login prompt THEN the overall ratings should change according to my ratings OTHERWISE I should receive a prompt error.


Actor
Web Application
Server
click on profile page for a drink
query database for that particular drink
return drink info
redirect user from result page to drink profile page
hover & click on stars to rate a drink
check if user is logged in
option
[if logged in]
credentials approved
input new rating to database
new overall rating
[not logged in]
credentials rejected
request login
provide login information
verify login details
option
[correct login details]
credential approved
input new rating to database
new overall rating
[incorrect login details]
credential rejected
prompt login failed

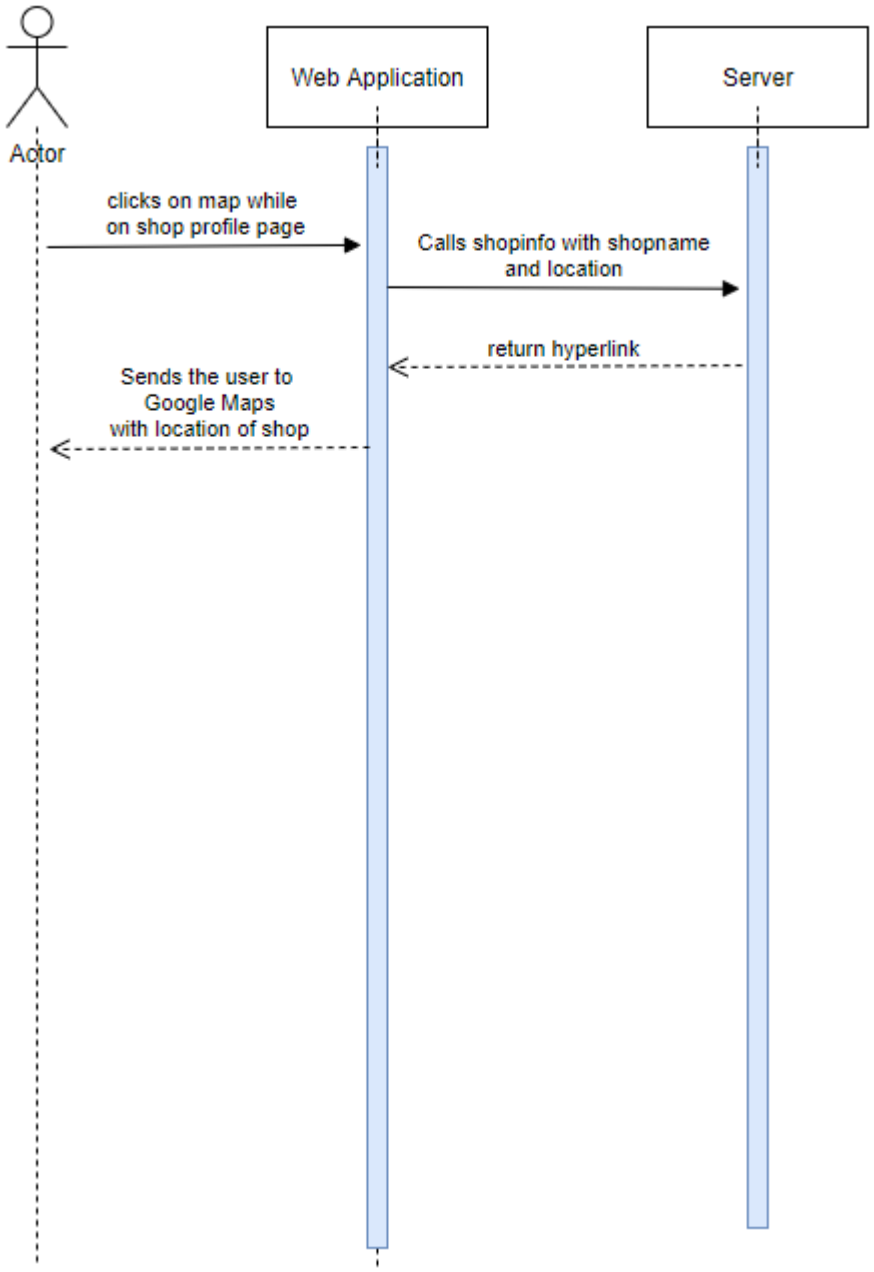
Feature ID	7
Feature	As a client, I should be able to view the rating for a drink, so I can make a decision based on the ratings.
Scenario	Client can view rating of a particular drink in case client is indecisive
	GIVEN I am on the search result page for drinks THEN I should be able to see the corresponding ratings next to the search result WHEN I am on the “profile page” for a drink THEN I can see the overall rating for the particular drink



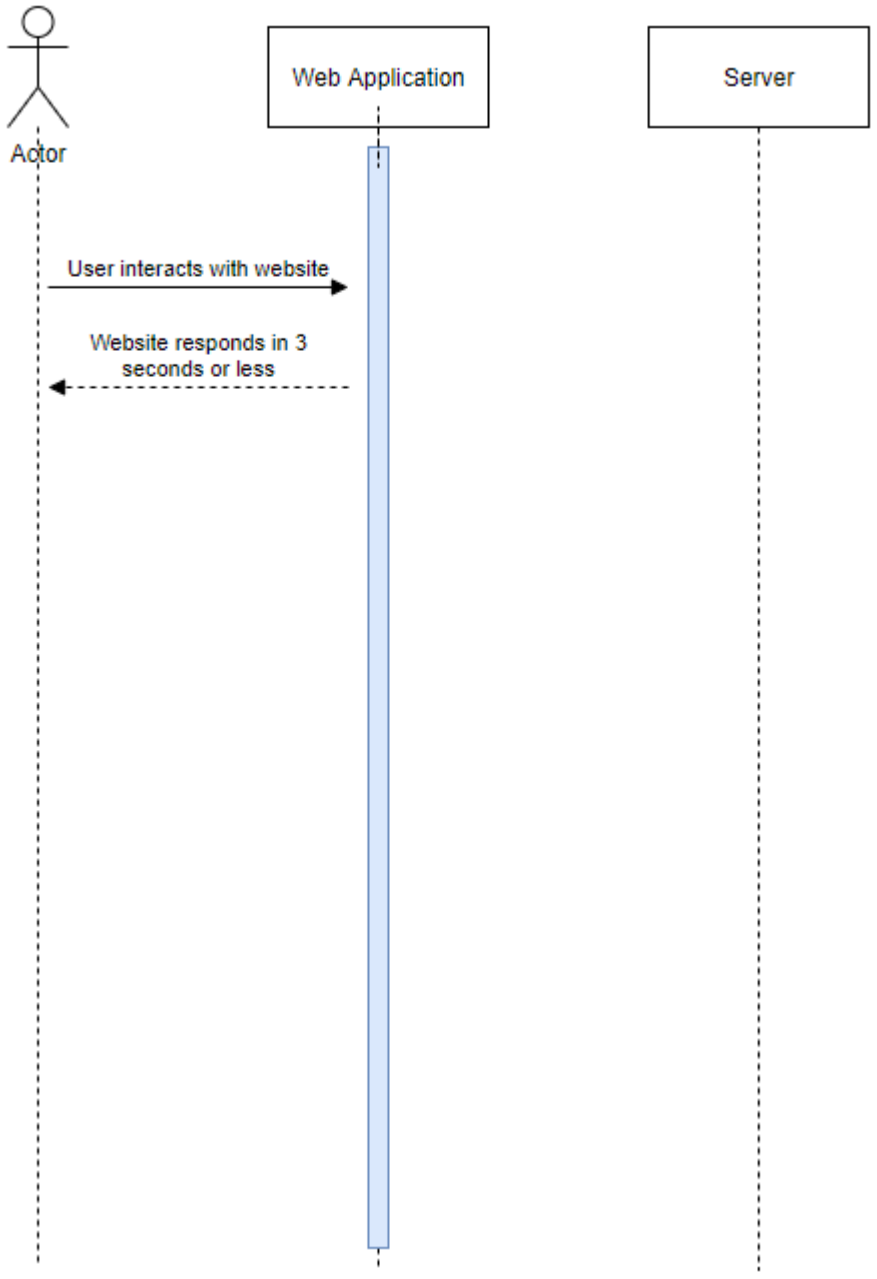
Feature ID	8
Feature	As a client, I should be able to get the website to make a decision for me, so I am not stuck in a dilemma situation.
Scenario	Choose a random drink
	GIVEN I am on the “homepage” WHEN I click on a ‘Randomise…’ button THEN the application will choose a drink from it’s many options at random AND display a clickable button. WHEN I click on the button THEN I should be redirected to that drink profile page



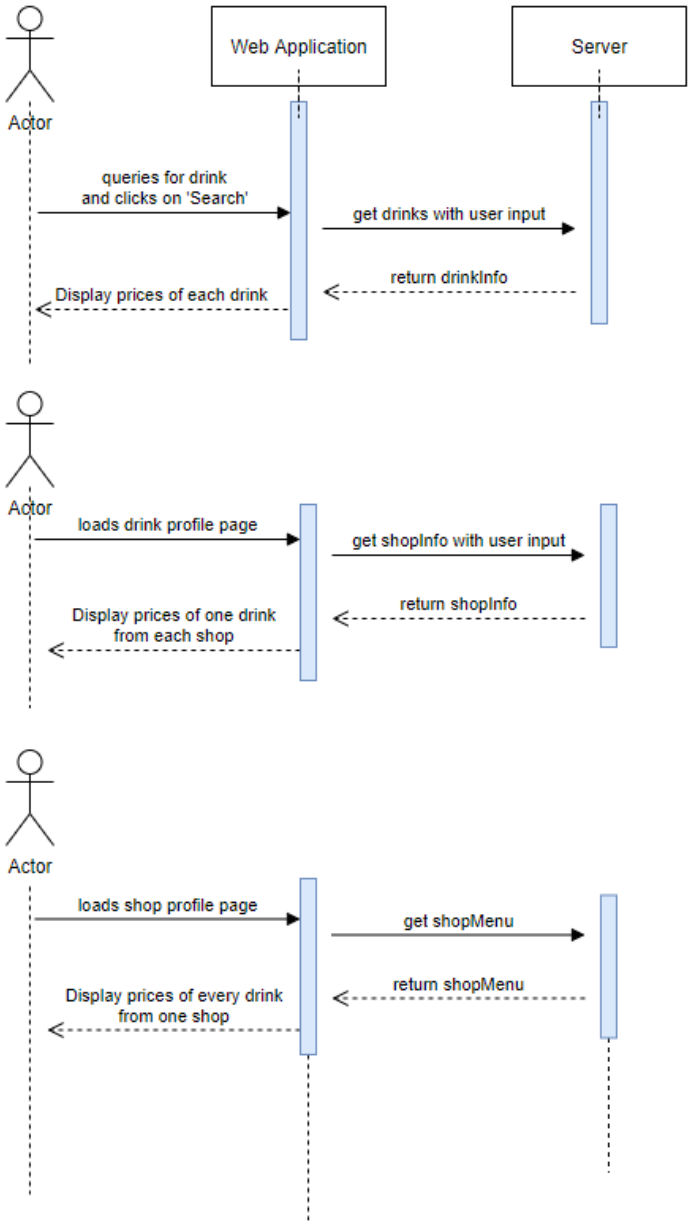
Feature ID	9
Feature	As a client I should be able to get directions to a particular boba shop, so I can get to the shop when I have made up my mind.
Scenario	Find directions to a particular boba shop.
	GIVEN I know what Boba shop i want to go to AND I am on the Boba shop's page WHEN I click on the "directions" button of the Boba shop THEN I should be redirected to google map with destination filled in for me.



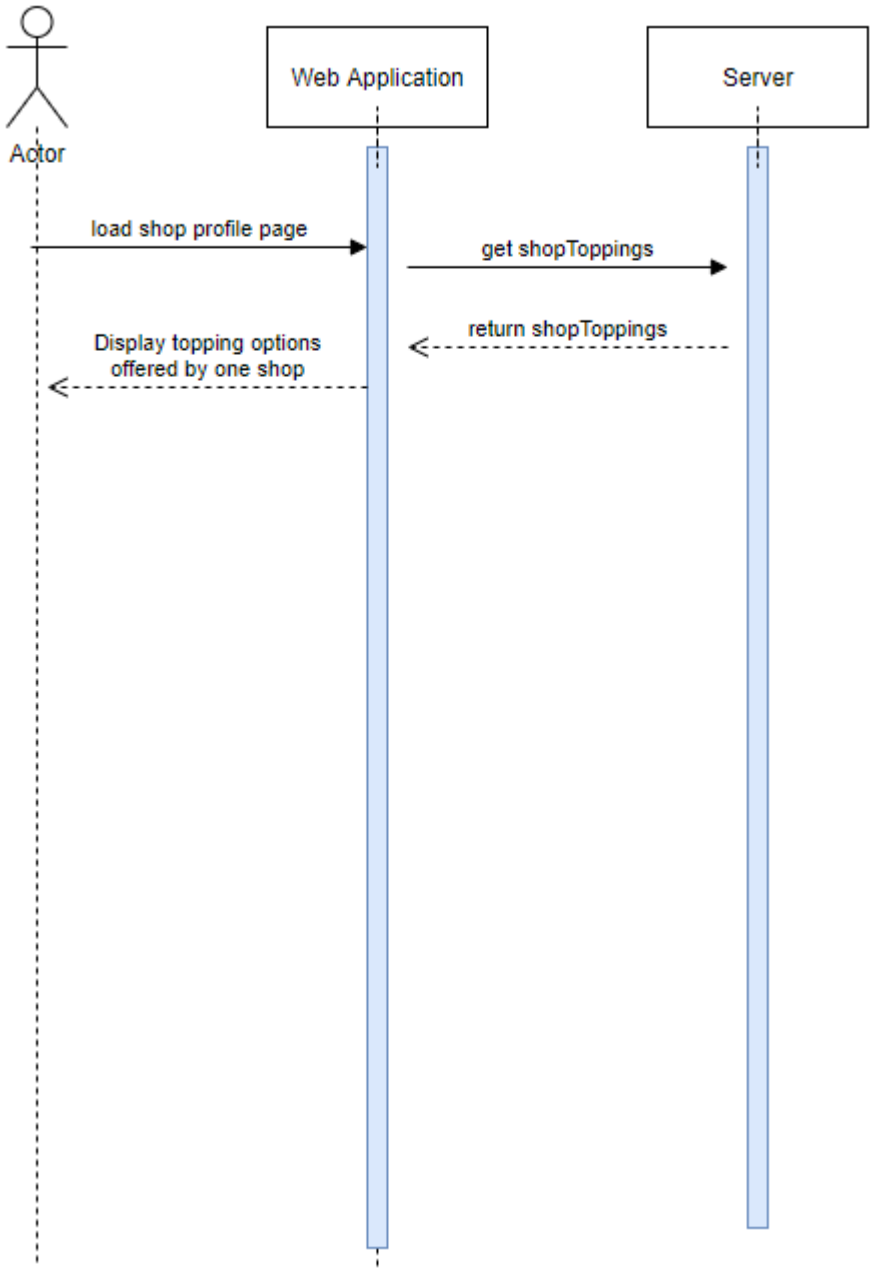
Feature ID	10
Feature	As a client, I should be able to see the response from the website within 3 seconds after I addressed a command, so I can carry out my tasks more efficiently.
Scenario	Navigating through the website as a general user
	GIVEN I am on ANY page WHEN I click on ANY button/link THEN I should see the effect of the button/link within 3 seconds



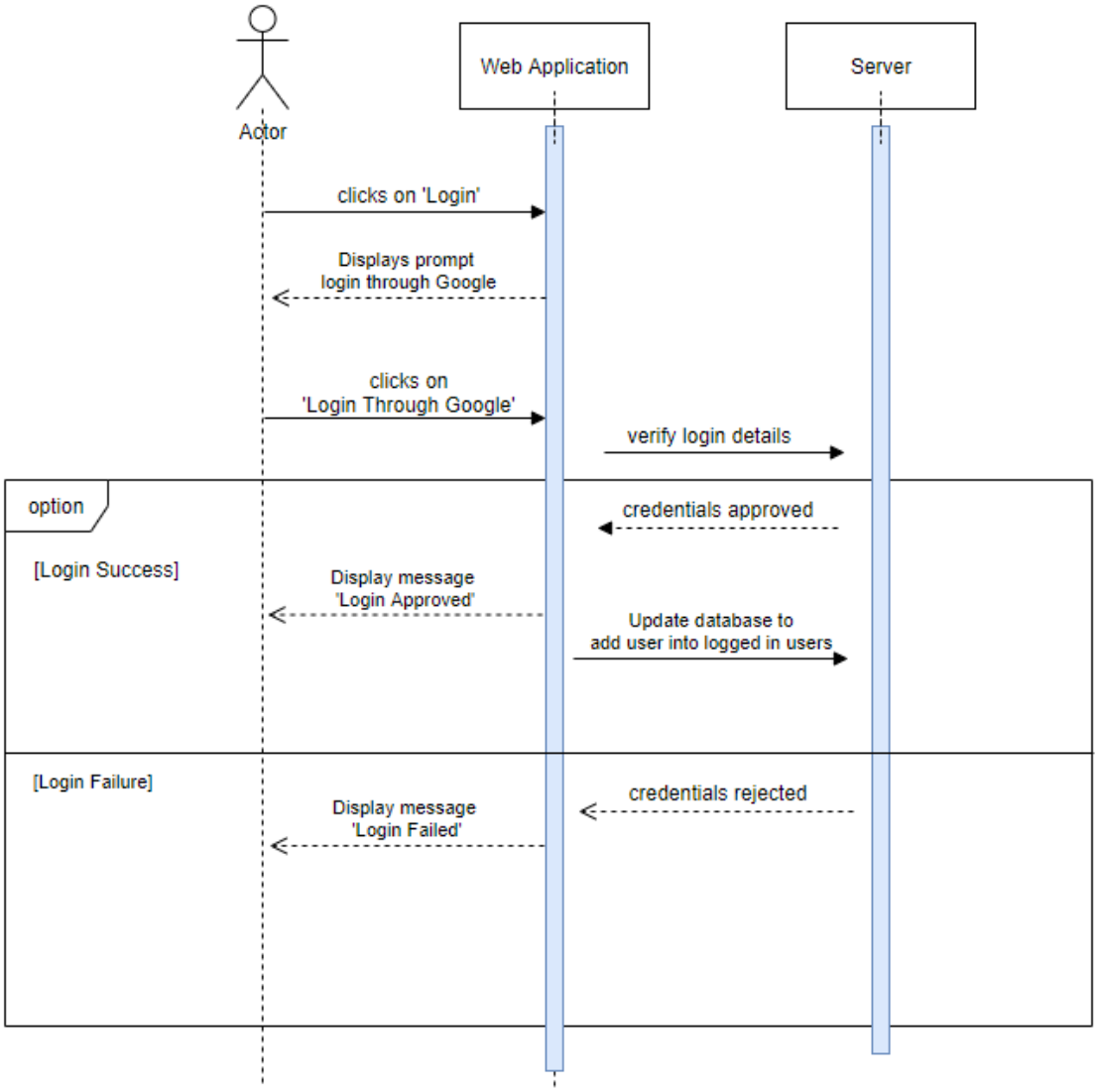
Feature ID	11
Feature	As a client, I should be able to see the price for any given drinks, so I know how much I am paying.
Scenario	Client can check for price before they commit to it
	GIVEN I am on the search result page for drinks THEN I should be able to see the corresponding price for each drinks GIVEN I am on the profile page for a drink THEN I should be able to see the price of the drink for each shop GIVEN I am on the profile page for a shop THEN I should be able to see the corresponding price for each drinks



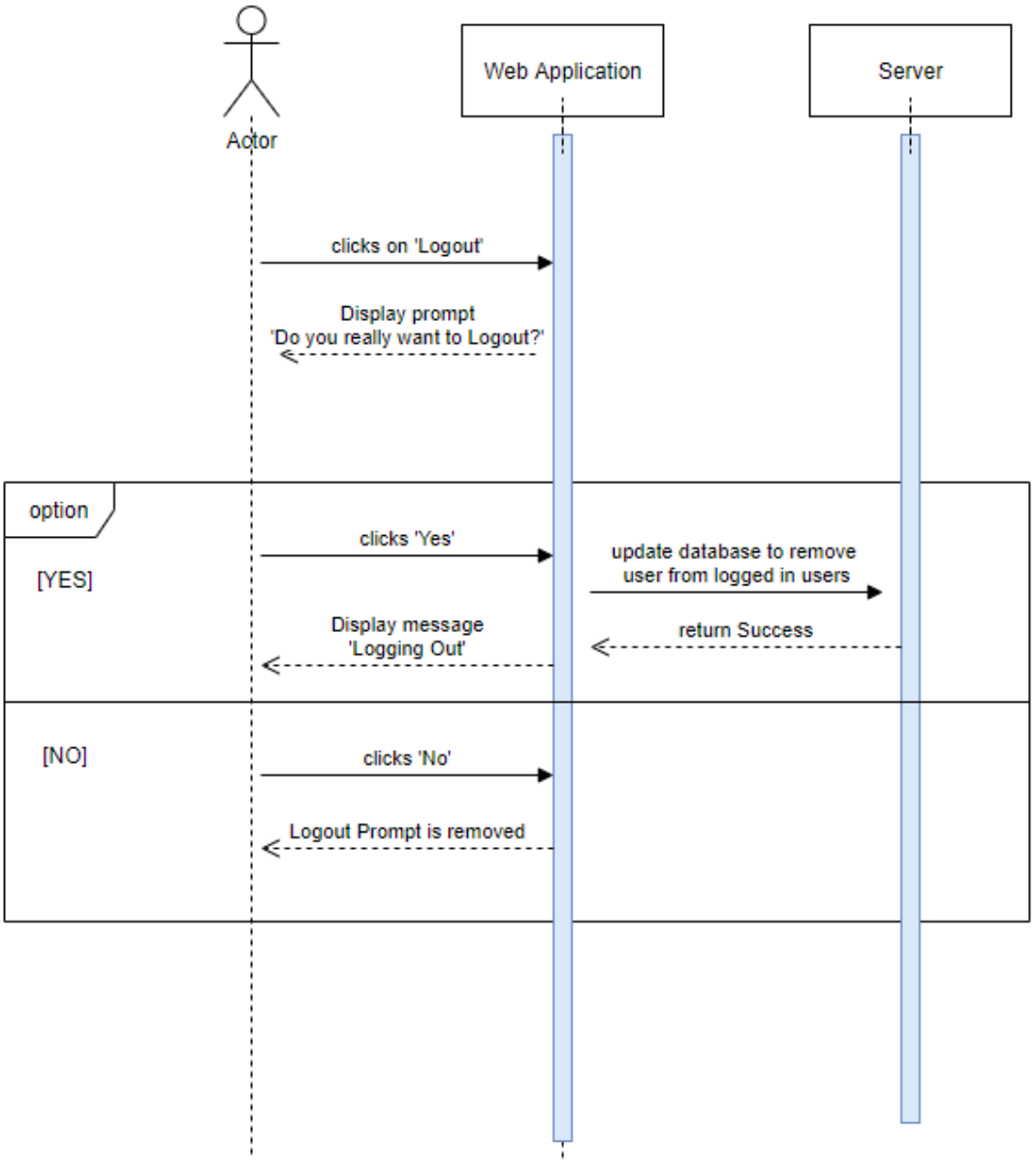
Feature ID	12
Feature	As a client, I should be able to see the range of toppings offered by the shop, so I can make my unique drink to enrich my drinking experience.
Scenario	Client can view the list of toppings offer by the shops to make their own drinks
	GIVEN I am on profile page for a shop THEN I should be able to see all the topping options offered by the particular shop.



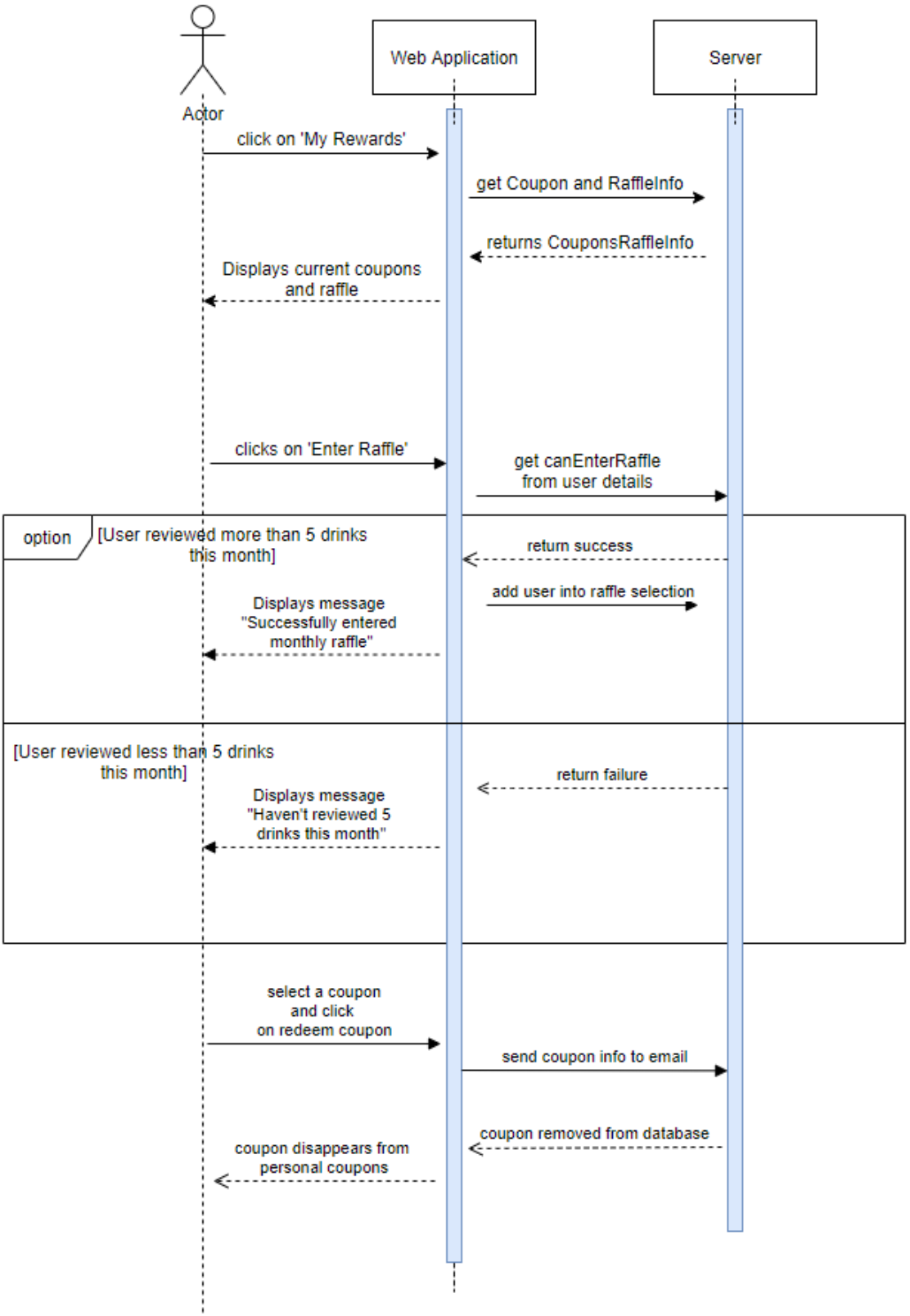
Feature ID	13
Feature	As a client, I should be able to login using google accounts so I can utilise the full potential of the website.
Scenario	Client can login using google accounts to rate drinks and perform other actions
	GIVEN I am on ANY page on the web WHEN I click on the “login” button THEN I should be able to see a “login” prompt WHEN I click the “login through google” button THEN I should be logged in IF I fail to login THEN I should receive an error prompt



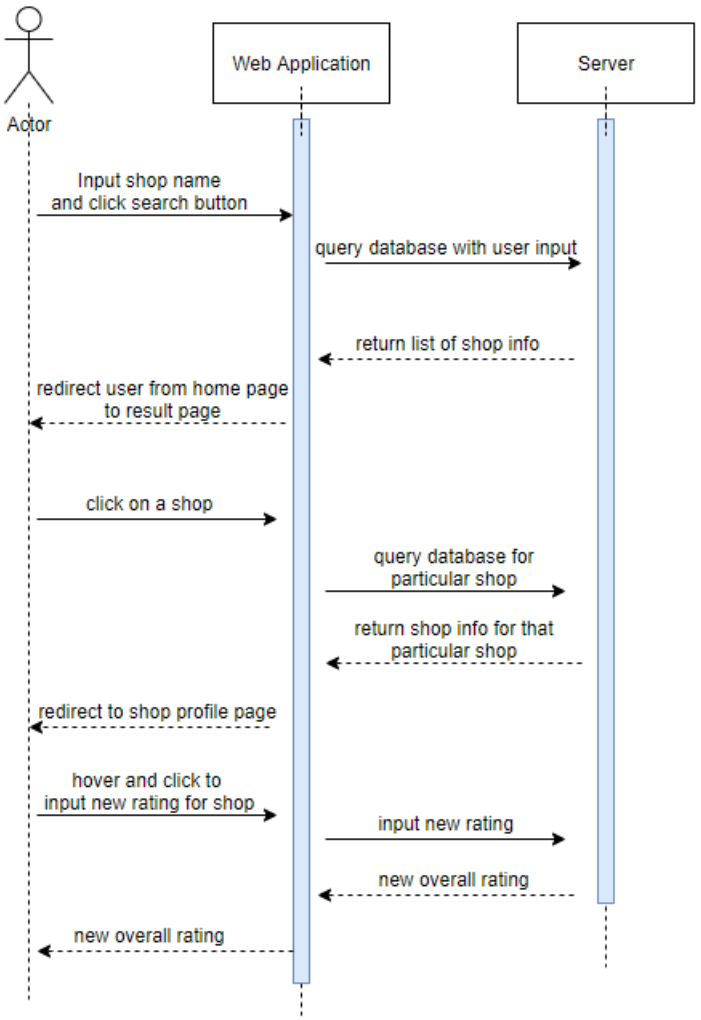
Feature ID	14
Feature	As a client, I should be able to logout from an account so I can login as other users.
Scenario	Clients can logout from the current account if they wish to.
	GIVEN I am on ANY page on the web AND I am logged in THEN I should be able to see the “logout” button in the top right corner WHEN I click on the “logout” button THEN I should be see a logout prompt WHEN I click “YES” THEN I should be logged out IF instead I click “NO” THEN I should remain logged in AND the prompt should disappear


Actor
Web Application
Server
clicks on 'Logout'
Display prompt
'Do you really want to Logout?'
option
[YES]
clicks 'Yes'
update database to remove
user from logged in users
return Success
Display message
'Logging Out'
[NO]
clicks 'No'
Logout Prompt is removed

Feature ID	15
Feature	As a client, I should be rewarded for using the website, so I can be convinced to be a long term user for the website
Scenario	Logged in users can be rewarded with free drinks or discount vouchers after they reviewed a certain amount of drink for each month.
	GIVEN I am on “home” page AND I am logged in THEN I should be able to see a ‘My Rewards’ button at the top of the page WHEN I click on the “My Rewards“ button THEN I should see a “Personal coupons” box AND I should see “Redeem Coupon” button AND I should see an “Enter Raffle” button WHEN I click on Enter Raffle button AND I have reviewed 5 drinks this month THEN I should be in the draw pool for a chance to get free drinks or a discount voucher AND I should receive a message that I did enter the raffle. IF I have not reviewed 5 drinks this month THEN I should receive a message that I did not enter the raffle. WHEN I select a coupon inside the “Personal Coupons” box AND I also click on the “Redeem Coupon” button THEN I should be sent a code to my google email for my coupon AND the coupon should disappear from “Personal Coupons”


Actor
Web Application
Server
click on 'My Rewards'
get Coupon and RaffleInfo
returns CouponsRaffleInfo
Displays current coupons and raffle
clicks on 'Enter Raffle'
get canEnterRaffle from user details
option
[User reviewed more than 5 drinks this month]
return success
add user into raffle selection
Displays message "Successfully entered monthly raffle"
[User reviewed less than 5 drinks this month]
return failure
Displays message "Haven't reviewed 5 drinks this month"
select a coupon and click on redeem coupon
send coupon info to email
coupon removed from database
coupon disappears from personal coupons

US id	16
Feature	As a client, I should be able to rate Boba shops for their services.
Scenario	Rate a shop using a five-star system for their services
	GIVEN I am on the homepage WHEN I type in a shop name AND I press the “Search” button THEN I should be taken to “result” page THEN I should be able to see shops containing the user input THEN I should be able to click on the shop AND I can choose to rate the shop



US id	17
Feature	As a client, I should be able to search for boba shops and sort them by rating..
Scenario	Search nearby boba shops according to distance from the client
	GIVEN I am on the homepage WHEN I type in a shop name AND I press the “Search” button THEN I should be taken to “result” page THEN I should be able to see shops containing the user input THEN I should be able to see the filter tools on the side AND I can choose to filter the result by top rated THEN I should be able to see all nearby shops in ascending distance order in grid style. OR I can choose to filter the result by most rated THEN I should be able to see all nearby shops in ascending distance order in grid style.