

# Matchmaking Online Food Ordering System

A DISSERTATION SUBMITTED TO MANCHESTER METROPOLITAN UNIVERSITY

FOR THE DEGREE OF MASTER OF SCIENCE

IN THE FACULTY OF SCIENCE AND ENGINEERING



September 2024

By

Deepak Gowda Nilavadi Rajamudi

Department of Computing and Mathematics

# Contents

Abstract.....	vi
Declaration.....	vii
Acknowledgements.....	viii
List of Abbreviations .....	ix
<b>1 Introduction.....</b>	<b>1</b>
1.1 Background.....	1
1.2 Motivation.....	2
1.3 Research Objectives and Questions.....	2
1.4 Methodologies.....	3
1.5 Contribution.....	4
<b>2 Literature Review.....</b>	<b>5</b>
2.1 Introduction.....	5
2.2 Technical details and concepts.....	6
2.2.1 Key features.....	6
2.2.2 Architecture of Qwik.....	8
2.3 Comparison.....	10
2.3.1 Angular - Web application framework.....	10
2.3.2 React - Frontend JavaScript library.....	13
2.3.3 jQuery – Frontend JavaScript library.....	15
2.4 Comparison table for frontend frameworks and libraries (Qwik, React, Angular and jQuery) .....	17
2.5 The table shows the benefits of Qwik over other frameworks.....	18
2.6 Social context.....	19
2.7 Ethical context.....	19
2.8 Legal context.....	20
2.9 Professional context.....	20
<b>3 Matchmaking online food ordering system - Application overview and design.....</b>	<b>21</b>
3.1 Motivation and Objectives.....	21
3.2 User Flow: Matchmaking online food order web application.....	22
3.3 Requirements Analysis.....	25
3.4 Technologies used to build this application.....	26
<b>4 Implementation of Matchmaking online food ordering application.....</b>	<b>27</b>
4.1 Qwik City .....	27
4.2 Vite.....	27
4.3 Implementation of the frontend and code structure – Qwik Framework.....	28
4.4 Root and entry files.....	29

4.5 Routes.....	30
4.6Global.css.....	33
4.7 Backend Implementation.....	36
4.7.1 Config - db.js.....	39
4.7.2 Server.js.....	40
<b>5 Evaluation.....</b>	<b>41</b>
5.1 Experimental results on the performance of the Matchmaking online food ordering system in comparison to other technologies.....	41
5.2 Future work.....	43
<b>6 Conclusion.....</b>	<b>43</b>
 <b>References.....</b>	 <b>44</b>
<b>Appendix A.....</b>	<b>46</b>
<b>Appendix B.....</b>	<b>56</b>
<b>Appendix C.....</b>	

# List of Figures

Figure 1 – Difference between Hydration and Resumable.....	7
Figure 2 – Builder.io landing page and wappalyzer detecting technologies used to develop it.....	10
Figure 3 – DOORDASH landing page and wappalyzer detecting technologies used to develop it.....	11
Figure 4 – Performance result of DOORDASH web application.....	11
Figure 5 - JUSTEAT landing page and wappalyzer detecting technologies used to develop it.....	13
Figure 6 - Performance result of JUSTEAT web application.....	14
Figure 7 - Foodpanda landing page and wappalyzer detecting technologies used to develop it.....	16
Figure 8 - Performance result of Foodpanda web application.....	16
Figure 9 – User flow for Matchmaking online food ordering system.....	23
Figure10 –User flow for Matchmaking online food ordering system with extra features.....	24
Figure 11 – Frontend code structure.....	28
Figure 12 - Matchmaking online food ordering system API server functioning with Wappalyzer tech stack detection.....	34
Figure 13 - Matchmaking online food ordering system admin panel.....	35
Figure 14 - Matchmaking online food ordering system landing page.....	35
Figure 15 - Matchmaking online food ordering system login page.....	36
Figure 16 – Backend code structure.....	37
Figure 17 – Performance result of admin panel using Google lighthouse.....	41

Figure 18 - Performance result of the landing page by using Google lighthouse.....	42
Figure 19 - Performance result of backend API by using Google lighthouse.....	42

## List of Tables

Table 1 - Comparison table for frontend frameworks and libraries (Qwik, React, Angular and jQuery) .....	17
Table 2 - The table shows the benefits of Qwik over other frameworks.....	18
Table 3 - Requirements Analysis.....	25

## List of Code Snippets

1. Code Snippet: vite.config.ts.....	29
2. Code Snippet: root.tsx file.....	30
3. Code Snippet: index.tsx in Cart folder.....	31
4. Code Snippet: index.tsx in Home folder.....	32
5. Code Snippet: assets.ts.....	33
6. Code Snippet: global.css.....	34
7. Code Snippet: LoginPopup.tsx.....	36
8. Code Snippet: package.json.....	38
9. Code Snippet: db.js.....	39
10. Code Snippet: server.js.....	40


# Abstract

This dissertation explores the development of Matchmaking online food ordering system by utilising a novel tech stack such as MongoDB, Express, Qwik, Node.js, Qwik city and Vite to overcome problems faced by modern frameworks regarding performance, glitches, scalability. Hence, Qwik has unique features such as resumability, lazy loading, instant loading which will significantly improve user experience and enhancing resource management. Rigorous performance reports using Google Lighthouse has demonstrated and proved significant improvements especially regarding performance which can handle high traffic situations thus it proves Qwik's superiority in developing efficient and robust web applications, presenting this as a transformative solution for improving e-commerce platforms and contributing for future innovations.

# Declaration

No part of this project has been submitted in support of an application for any other degree or qualification at this or any other institute of learning. Apart from those parts of the project containing citations to the work of others, this project is my own unaided work.

Date: 27/09/2024

Signed: 

EthOS Reference Number: 68943

# Acknowledgements

I would like to express my gratitude to my supervisor, Dr Conor Muldoon for his highly valuable guidance, time, and encouragement throughout the thesis and also during the course in general. Additionally, I would like to thank my family members for being supportive and constant motivation. Furthermore, I appreciate my friends for insightful discussions and suggestions.



# List of Abbreviations

HTML Hypertext Markup Language  
CSS Cascading Style Sheets  
XML Extensible Markup Language  
AJAX Asynchronous JavaScript and XML  
API Application Programming Interface  
REST Representational State Transfer  
HTTP Hypertext Transfer Protocol  
UI/UX User Interface/User Experience  
JSX JavaScript Syntax Extension  
TSX TypeScript Execute  
SSR Server-side Rendering  
CSR Client-side Rendering  
SSG Static Site Generation  
CRUD Create – Read – Update – Delete  
URL Uniform Resource Locator  
MEN MongoDB – Express – Node.js  
EXT Extension  
MVC Model – View – Controller  
QRL Qwik URL  
JSON JavaScript Object Notation

# 1 Introduction

## 1.1 Background

Evolution in the field of web development is happening rapidly, initially it started with static websites with simple HTML but limited interactivity and functionality, once web started to evolve developers began to use server-side technologies like PHP, ASP, and Common Gateway Interface to build more dynamic content. Later the arrival of technologies like JavaScript helped in enabling client-side scripting and offered enhanced interactive user experience hence this was a turning point in web development. AJAX (Asynchronous JavaScript and XML) further revolutionized web development by introducing asynchronous communication between client and server to build more interactive, dynamic web applications. To perform complicated tasks and enable compatibility with multiple browsers early JavaScript libraries were utilized such as Dojo Toolkit, jQuery, Prototype, MooTools, QUnit, Ext JS. However, these technologies started to face issues in performance and scalability due to complex web applications. Furthermore, to develop more sophisticated web applications and the growing demand for interactive applications, libraries, and frameworks such as React, Angular, jQuery, Next.js were introduced due to their ability to perform complex coding tasks, offering robust solutions and enhance user experience (Paakkanen, December 22, 2023). Despite being popular these traditional frameworks have lot of issues, challenges particularly in performance, initial loading. Though React's component-based architecture and Angular's comprehensive toolkit allow developers to use powerful tools they face complexity and potential latency due to heavy dependency on client-side processing. These existing modern frameworks have limitations and negative impact on such food ordering platforms where speed and performance are crucial. Food ordering mobile/web applications have been extremely important tool for the restaurants, hotels as it's a platform to maintain customer relationship and there is a rapid growth of customers relying on these digital platforms for their needs (BALIEVA, July 2023). Therefore, the performance and user experience have a direct impact on customer interaction, and satisfaction and traditional frameworks such as React, Angular, Vue face technical issues like instant loading, poor performance during high traffic, delayed initial load times, high data consumption, unresponsive interface. Hence, for these issues and competitive business Qwik offers an effective solution which can help in developing a high-performance online food ordering web/mobile application that is capable of handling high traffic, provide instant loading, scalability, particularly on slow networks and low-powered devices. The main motive of developing web/mobile applications using Qwik framework is to achieve 100/100 on Google PageSpeed or Google Lighthouse and prove that e-commerce platforms can be fast.

## **1.2 Motivation:**

Online platforms are the primary tools for many businesses to maintain customer engagement, and the motivation for this research arises from the critical need to solve the rising problems faced by online food ordering web/mobile applications in the context of performance, scalability, and user experience and these issues can affect customer satisfaction, revenue, and loss of customers, despite using popular modern frameworks like React, Vue, Angular, many platforms are facing many drawbacks such as complex client-side rendering, longer initial loading times, insufficient efficiency under high traffic circumstances. Due to the rapid evolution of technology especially in web development, there is a rise in demand for technologies that can provide optimal performance on all kinds of devices including those with slower networks.

The Qwik framework offers an effective and robust solution thus acts as a novel approach in the field of web development which has the potential to enhance speed, instant loading and prevents high data consumption, Qwik's goal is to achieve high performance scores like 100/100 on platforms like Google Lighthouse and PageSpeed. Hence, these qualities of Qwik framework make it superior that can perform under challenging conditions and provide great services to build high traffic e-commerce stores such as online food ordering web applications.

Therefore, the motivation of this research and development is to demonstrate and prove that the Qwik framework is unique and offers a novel solution that fulfils high standards and provide uninterrupted and fast user experience. Hence, this project intends to show that it is the best solution to build e-commerce platforms which is highly reliable and powerful, ultimately leads to customer satisfaction and remain competitive in the market.

## **1.3 Research Objectives and Questions**

Due to the rising demand for high performance online food ordering system, this thesis investigates a new approach and provides an effective solution by developing Matchmaking online food ordering system using Qwik framework and comparing its overall performance with modern frameworks.

Qwik is a next-generation JavaScript framework that aims to solve various technical issues which other modern frameworks are not able to solve, especially e-commerce sites where speed and performance are the most important therefore to investigate the performance of Matchmaking online food ordering web application, tools such as Google Lighthouse, PageSpeed, will be used and compared with other online food ordering platforms which are built using modern frameworks like React and Angular.

Therefore, the thesis focuses to address the following questions:

- What are the technical issues faced by e-commerce applications which are built using existing modern frameworks such as Angular, Vue.js, React.
- why do these web applications have poor performance, fail to handle high traffic, slow initial load times and what can be done to fix these issues.
- What is unique about Qwik framework, and its architecture compared to other frontend frameworks, how does it achieve high performance, speed, scalability.
- What are the limitations of Qwik framework and is it reliable compared to other traditional JavaScript frameworks.

All these questions will be discussed and examined in the following chapters to investigate the potential of Qwik framework and evaluate whether it is truly the future of web development.

## **1.4 Methodologies:**

The thesis is a study focused on the research and development of Qwik framework. Each chapter has different sections, the first chapter discusses about background of web development, following section is about research objectives & questions, along with methodologies. The second chapter gives a brief introduction about JavaScript and its modern frameworks, followed by technical details of Qwik frameworks with key features. The thesis includes a comprehensive literature review, which surveys previous publications, journals, articles, official documents, to provide valuable insight. Other section discusses by comparing various modern frameworks with Qwik using real-world case studies. Additionally, the next section discusses about social context, legal context, Ethical context, and Professional context.

The next chapter gives an overview of application's design, it explains the motivation and objectives of Matchmaking online food ordering web application, it is followed by a user flow chart of this application which gives an idea to the users about how it works. The subsequent section discusses about requirement analysis detailing both functional and non-functional requirements. The following section discusses about system infrastructure and technologies used to develop this web application. Another section discusses about application implementation including frontend, backend, database, along with code snippets. The last two chapters evaluate the performance evaluation of the web application and real-world case studies are used to compare it with other web applications built using different technologies and conclude with an explanation regarding possible future work and key findings.

## **1.5 Contribution:**

The thesis contributes to the web development community including Qwik framework community by demonstrating the use of Qwik that can build high performance web application especially e-commerce sites and prove that it has the potential to be the next generation frontend framework and support businesses/organizations to remain competitive. Matchmaking online food ordering web application demonstrates a practical implementation of how the technology works, it opens possibilities for exploring the potential of Qwik framework in different types of applications and collaborating with emerging technologies.

The study provides insights and contributes to the development, research, also acts as foundation for future studies. However, the publications on this topic are insufficient because it's an emerging technology so this thesis contributes to the existing research, communities, and businesses understand the importance and potential of utilizing this technology to stay ahead in the landscape of web development. Qwik can be considered as reliable even though its ecosystem is smaller compared to modern frameworks such as React, Vue, Angular which have range of tools, integrations and components.

Additionally, this thesis promotes further analysis of Qwik's capabilities by integrating with modern and novel technologies such as micro frontends, Server-side rendering (SSR) and progressive web-based applications, the study shows how Qwik framework can handle typical performance bottlenecks by presenting the unique features such as resumability and lazy loading. The researcher not only proves Qwik as an effective alternation compared to other modern established frameworks, but it also proves that developers and organizations can use it to build high performance and scalable web applications hence encouraging innovation in the field of web development.

## 2 Literature Review

### 2.1 Introduction

The purpose of this literature review is to examine the emerging JavaScript framework that solves performance issues in web apps by focusing on quick load times and systematic rendering. Qwik aims to produce interactive and quick experiences on devices that have limited resources. It is still in the growing stage compared to other established frameworks such as React, Angular, Vue.js, Express etc and the Qwik community is contributing to its development even though it's small compared to other communities. Qwik has a unique approach that focuses on building web applications with superior performance, instant loading, unlike other frameworks that depend heavily on client-side rendering, they also have to download and execute large bundles of JavaScript before the application becomes responsive. For several years, web development has been dominated by traditional JavaScript frameworks like React, Vue.js, Angular, because they provide powerful tools and features to build interactive and dynamic web applications. Despite offering such powerful tools and features, they rely severely on client-side rendering which needs to download and execute huge bundles of JavaScript.

With an intention to solve these issues related to performance, the Qwik framework has been established as a highly promising alternative. It has been developed specifically to improve load times and decrease the amount of JavaScript that requires to be executed on the client side. Qwik's advanced architecture allows efficient and precise rendering and enables web applications to swiftly become interactive by sending only the necessary code to the client precisely at the right time. This novel approach is especially valuable for applications that need optimal performance on several devices.

The aim of this literature study is to investigate the fundamental principles, concepts, and perks of using the Qwik framework in comparison with well-developed frameworks. This literature review is intended to show how Qwik, through its novel approach regarding performance optimisation, the literature review also highlight how Qwik overcomes the drawbacks of traditional frameworks and contributes to the development of high-performance web applications that provide excellent user experiences, irrespective of the device or network conditions. This will be accomplished by investigating current literature and analysing Qwik's unique features. This review will also examine the continuous development efforts and support the Qwik community, while reflecting on its present state, potential, and future path in the constantly evolving field of web development.

Limitations of Qwik framework:

- As it is a new and emerging framework, it may not have been recognised by developers or adopted in the industry compared to established frameworks hence less job opportunities (JUHO VEPSÄLÄINEN, 11 January 2024)
- Qwik is designed to provide high performance, but it depends on how good the application is structured and how its features are utilized. Therefore, poor implementation may result in low performance (JUHO VEPSÄLÄINEN, 11 January 2024)
- Even though Qwik has familiar concepts, developers might face problems in adopting and learning it compared to modern frameworks, particularly syntax, code-splitting, new paradigms, and may face challenges in finding resources and community support (JUHO VEPSÄLÄINEN, 11 January 2024)

## **2.2 Technical details and concepts:**

### **Qwik framework**

**Technical details:** Qwik was developed by a team at Builder.io in 2021 also known as Qwik v1.0 it is full-stack web framework, primarily developed by Misko Hevery, one of the creators of Angular. Qwik's architecture provides the quick delivery of responsive pages by loading just the code necessary for the initial view and progressively loading additionally interactivity as required. This procedure significantly reduces the amount of JavaScript that has to be executed on the client, resulting in faster load times and enhanced performance throughout a variety of devices and networks. Qwik is a server-side framework it uses JSX syntax just like React and Solid to define HTML output; by executing JavaScript code on the server side, it generates HTML content of a web page (JuhoPaakkanen, December22,2023).

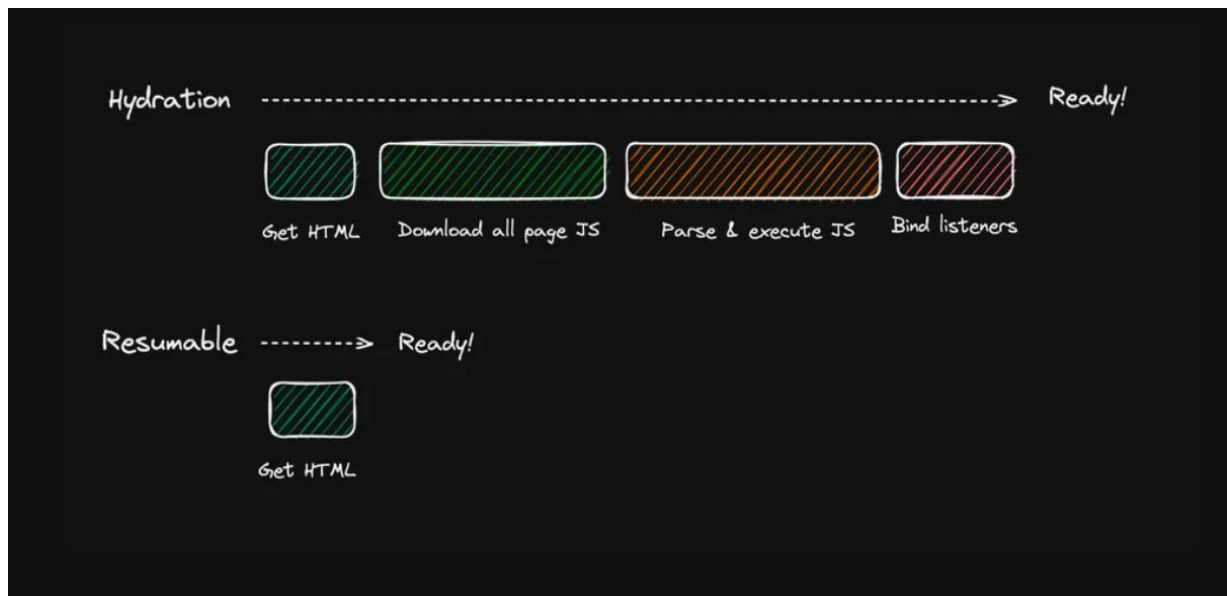
Qwik's architecture provides a rapid delivery of responsive websites by loading the required code which is required for the initial view and gradually enhancing interactivity when it is required. This technique leads to a major reduction in JavaScript execution on the client side, which leads to quicker load times and improved performance on several devices and varying network conditions (Team, May 2023).

#### **2.2.1 Key features**

- a) **Instant loading:** Instant loading is one of the main features that allows web apps to load very quickly by pre-fetching and pre-rendering portions of the app that are likely to be utilised, even before its requested by the user. This leads to rapid transition between pages,

and this significantly improves user engagement (JUHO VEPSÄLÄINEN, 11 January 2024).

- b) Resumability: Resumability is pausing execution on the server and resuming execution on the client without the necessity of downloading and replaying, so the applications have the capability of pausing the operation at some point and continue again from where it was stopped without restarting. At any point of lifecycle Qwik applications are serialized and transferred to different VM instance like server to browser, without hydration the application can easily resume from where the serialization stops so unlike other frameworks Qwik does not require hydration they just resume (JUHO VEPSÄLÄINEN, 11 January 2024).



*Figure 1 – Difference between Hydration and Resumable*

- c) Code-splitting: Code-splitting means the application's code is broken down into smaller sections also called as chunks, Qwik enables precise code splitting, allowing developers to define split points inside their components. This indicates that only the appropriate code is loaded when it is needed, it prevents loading the entire codebase because it is inefficient and not necessary, hence the application uses the resources efficiently, this optimises usage of resource and improves load times (JUHO VEPSÄLÄINEN, 11 January 2024).
- d) Lazy loading: Depending on the input from users the framework has the capacity to load resources and components. This signifies that not all JavaScript is needed to be loaded in advance, which helps in minimising the primary payload and boosting the overall performance of the application (JUHO VEPSÄLÄINEN, 11 January 2024).
- e) Scalability: Scalability means when a system can manage bigger projects or extra work without a decrease in performance, so Qwik has been designed to scale as the projects grow and allows developers to handle rising complexity without losing performance. scalability is accomplished by its architectural decisions which delays the task until there is necessity (JUHO VEPSÄLÄINEN, 11 January 2024).



### 2.2.2 Architecture of Qwik:

Applications developed by Qwik are resumable, it means they can be paused on the server and resumed on the client without the need to replay and download all the application logic. Qwik works on principle of lazy loading, at a high level it is similar to other web frameworks and the aim of this framework is to achieve instant-on applications including on mobile devices so to perform this Qwik uses two strategies:

- 1) Slow down the execution and downloading of JavaScript for a long time.
- 2) It saves the current state of the application on the server, then resume executing it on the client from where it was stopped.

Qwik applications perform quickly because of less JavaScript code to run, the application just needs 1KB of JavaScript for the interaction. The unique quality of this framework is that, by deliberately slowing down the download and execution of application it can offer instantaneous startup performance, and this implementation is not matched by any existing frameworks (Cao, 2023). Qwik has fast performance not because of its algorithms, it's because of the way it is designed where most of the JavaScript need not be executed or downloaded and one of the features that makes this framework to stand out is other frameworks such as React, Angular, Vue.js have to do hydration but Qwik does not have to. But to achieve this Qwik has to solve three problems and those are (Team, May 2023),

- 1) **Listeners:** Nowadays all websites require high degree of interaction so all the static websites have event listeners, therefore DOM is also just a static page without listeners and cannot be considered as an application. Therefore, to solve the event listener issue, existing frameworks download components that are necessary and run their templates to get event listeners and then it's connected to DOM, examples are hovers, menus, expanding details and interactive apps. But there are issues with current approach:
  - a) It is required to download the template code rapidly.
  - b) The template code that is downloaded is required to be executed anticipatorily.
  - c) And it is necessary to download event handler code eagerly and has to be attached (Team, May 2023).

But this approach doesn't work practically, this depends on how complex the application is. If it is too complex it is necessary to download more code and execute it and corresponding to the increase in size of the application. Due to this problem, it negatively affects the user experience and the application startup performance and in other existing frameworks the lazy loading sections of the application requires big development which results in consumption of time and effort, but in Qwik the lazy loading is natural because of its resumable architecture so this unique nature of Qwik makes it different from other frameworks.

The problem here is that, when web page is loading it has to set up event listeners, but this is a slow process because browser has to go through whole page to set up listeners individually, so Qwik solves this problem by putting event listeners directly into the HTML instead of JavaScript code that runs in browser. In server-side work, Qwik collects all the listener information on the server when the HTML is produced, then HTML is ready with all the data required for the browser. The role of Qwikloader is to install global event listener for all events instead of having several event listeners and this doesn't require any application codes, but it waits for the events. And this is how it works in the browser, HTML tells the Qwikloader to download the codes and execute it only that is necessary, the necessary information is provided by "on:click" attribute like URL to code chunk and required function to be executed. Qwik can handle asynchronous code by running and executing codes when its required and this makes the application efficient because it loads only that is necessary for the event. Qwik gathers component boundary information that is a part of Server-Side Rendering/Static Site Generation and then that information is serialized into HTML (Cao, 2023).

- a) Qwik is capable of rebuilding component hierarchy information in the absence of component code, so the component code remains lazy.
- b) Instead of re-rendering the whole web page it updates whatever is necessary in the web page, so this makes the web page to update and load quickly.
- c) Qwik keeps track of web page that depends on specific data, and it knows which sections to update when the data changes then this information gets saved in HTML.

2) **Component Trees:** Component trees are used by the frameworks to control how the parts of webpage are updated and constructed, so the frameworks need clear knowledge of the component trees to understand when the components have to be re-rendered. Component boundary information has been lost when you examine the current framework SSR/SSG output, by looking at the generated HTML it's not possible to know where the component boundaries are. Frameworks run the component templates and memoize the component boundary location to replicate the information, Hydration is high cost because it is necessary to download and execute the component templates (Team, May 2023).

3) **Application State:** Just like other frameworks Qwik is capable of saving the web application in its current state into HTML and allows the web app to pick up from where it stopped when the page is reloaded. Other frameworks when they have to load a component its necessary for them to download parent component because they contain important data that is required therefore it's difficult to load components individually because of this dependency chain. In Qwik the component's life cycle and its state is closely linked this means that component's code and state can be loaded separately, to avoid this issue Qwik allows the component to begin its state without the necessary of its parent components' code, this results in rapid loading and updating of web applications (Team, May 2023).

Here is an image of a platform called Builder.io which is developed using the Qwik framework, along with this a tool known as Wappalyzer detects the technologies used to develop the application.

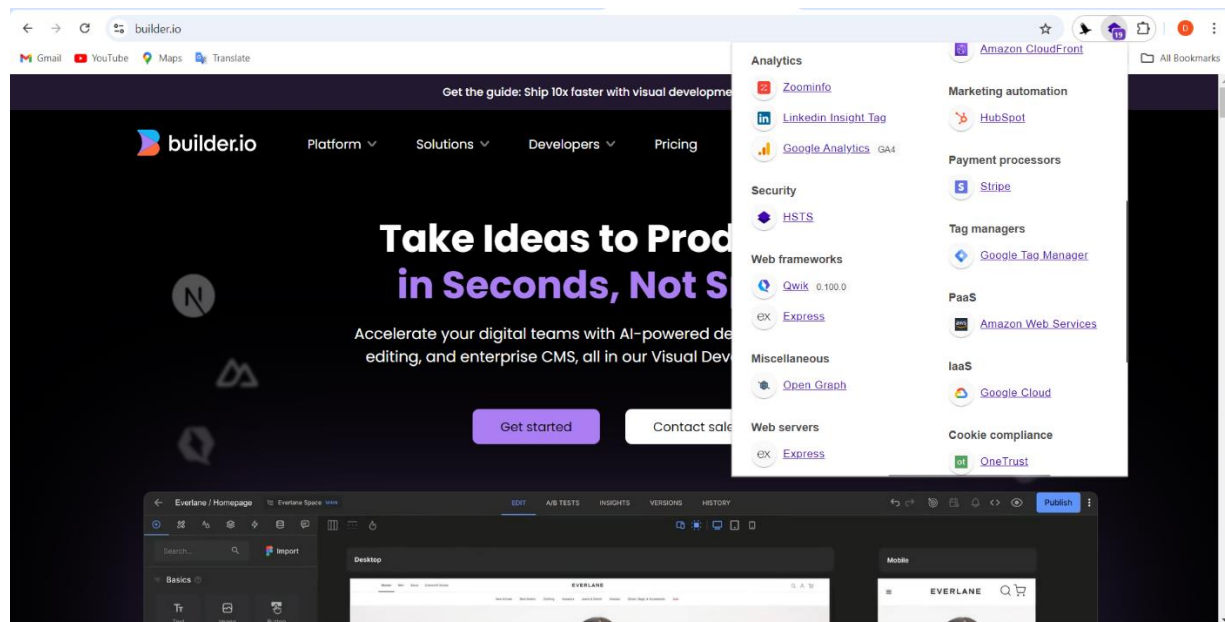


Figure 2 – Builder.io landing page and wappalyzer detecting technologies used to develop it

## 2.3 Comparison:

In this chapter, we'll study a comparison of modern frameworks and libraries by investigating e-commerce stores such as online food ordering platforms alongside exploring the features, functionalities of these frameworks and libraries. To gain more information about technology stack used to develop these online food ordering platforms I have used a tool called Wappalyzer and this tool can be installed as an extension in chrome, it helps in examining the tech stack and other technologies used to develop the application. Furthermore, the images display the Wappalyzer extension in Google Chrome investigating the technologies used to develop the online food ordering application.

### 2.3.1 Angular - Web application framework

The image of an online food ordering platform called DOORDASH is built using Angular and a tool known as Wappalyzer helps in displaying the technologies used to develop this application.

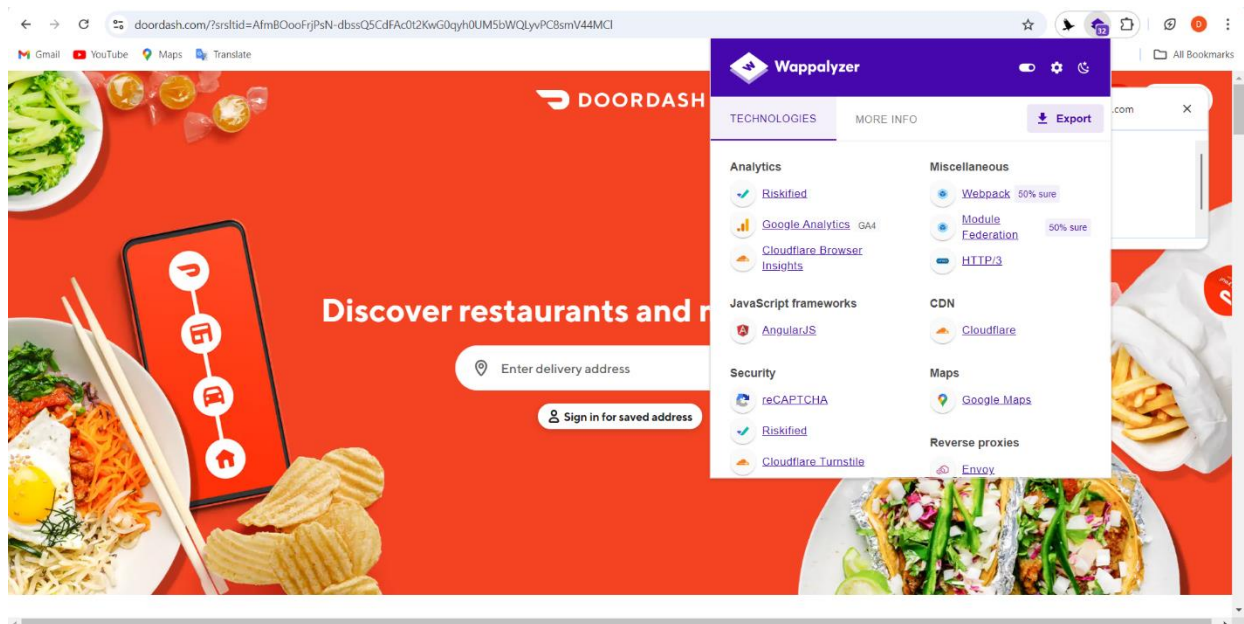


Figure 3 – DOORDASH landing page and wappalyzer detecting technologies used to develop it

The image below shows the performance result of the landing page using Google lighthouse:

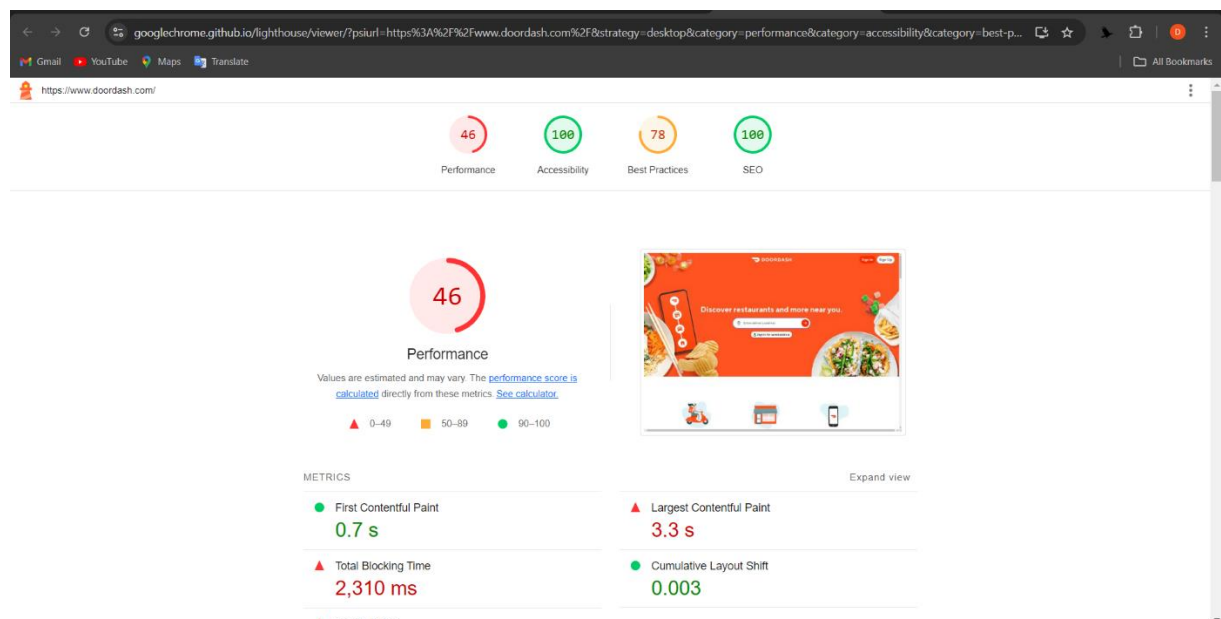


Figure 4 – Performance result of DOORDASH web application

and a complete report can be found in the pdf attached here: [..MSc Project\DoorDash\\_lighthouse report.pdf](#)

Angular framework is the most preferred to build responsive single page applications by focussing on robustness and clarity (Ovidiu Constantin NOVAC, 2021). Comparing traditional frameworks like Angular with Qwik reveals that both frameworks are designed to build high

performance web applications. In this study, it shows that, in a short period of time Angular has gained popularity which helped developers to build dynamic web apps, but some frameworks contain existing technologies this makes the application development very difficult. Misko Hevery, a google engineer invented Angular just for the internal usage of company, initially it was used to build captivating web apps for his other projects later the team began to develop web applications using Angular, then it was agreed to make Angular JS an open-source software that will be accessible to public users, but it has faced several technical issues and are unable to meet the demands of large corporate companies (G.Geetha, 2022).

Angular has a range of technical issues & challenges regarding performance, flexibility, slow load times due to large size of bundles (Sultan, 29-30 November 2017). All these issues cause negative impact on users. To discuss about these issues let's consider an existing online food ordering platform called DOORDASH which is built using Angular it is one of the leading online ordering businesses, but it has faced technical issues and disrupted the customers while placing orders. Therefore, from this case study and comparison, the efficient way to build high traffic web applications such as online ordering platforms is by using Qwik that focuses on performance, instant loading, efficiency compared to Angular that slows down performance, faces challenges with scalability, due to complexity and large bundles. Hence the case of Just Eat shows that its web application which was developed using Angular faced technical challenges and impacted the business, so Qwik is an alternative that can handle high traffic applications. It's important to consider social context so we'll explore how these technologies are used and perceived in different social context. The articles show the report of performance analysis of DOORDASH platform which shows the problems faced by the platform and how it was resolved. This happens because applications built with React are client-side rendered due to the necessity of downloading, parsing and execution on the client and Qwik is built to load instantly by sending the minimum amount of JavaScript which is essential for the website to be responsive because Qwik has a novel approach called resumability and avoids hydration unlike other modern frameworks (Paipo Tang, David Nguyen, August 30, 2022).

Initially DOORDASH platform faced issues such as slow loading time because it relied on client-side rendering (CSR) and due to the complexity of the application users noticed delays up to 10 seconds particularly while loading UI. The challenges faced by the platform are: Unable to manage the JavaScript bundle size, web page used to render all elements at the same time, and Excessive DOM size. Additionally, next section discusses about another application developed with a different framework.

## 2.3.2 React - Frontend JavaScript library

React is characterized as a lightweight application platform, it was released by Facebook in 2013, but this is more classified as a library rather than a framework. In a study it shows the special features of React, which is efficient in rendering User Interface and another feature is its integration with other libraries, React collaborated with Angular JS can build high performance and rich content applications but React requires more lines to achieve functionality than Angular which can do in few lines (Sultan, 29-30 November 2017). JSX is an extension of JavaScript that permits developers to utilise XML like syntax within JavaScript code hence React utilises JSX and this approach makes React components completely isolated and makes it possible to be reused. Regarding documentation, React has a powerful set of documentation which allows developers in learning and using the library properly. It also has a robust community support that provides tools for solving issues and effective practices (Hamed Hussen Ben kora1, 15-06-2024 ). Several React projects are written using ES6 also known as ES2015, it's a version of JavaScript which was approved by ECMA International in 2015. To discuss about these issues let's consider an existing online food ordering platform called JUST EAT which is built using React, though it's one of the leading online food ordering platforms such as extreme delay particularly when communicating with APIs to obtain restaurant data, customer information and real-time tracking of deliveries along with UI UX issues, data syncing etc. The image of an online food ordering platform called JUSTEAT is built using React, Next.js and a tool known as Wappalyzer helps in displaying the technologies used to develop this application.

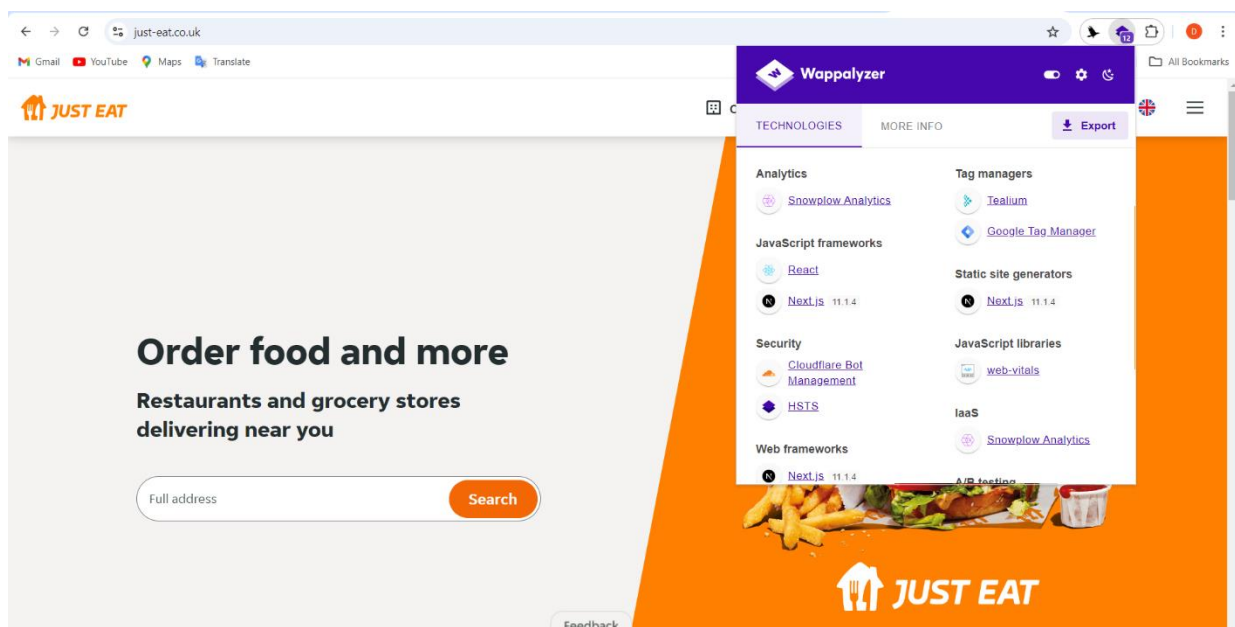


Figure 5 - JUSTEAT landing page and wappalyzer detecting technologies used to develop it

The image below shows the performance result of the landing page using Google lighthouse:

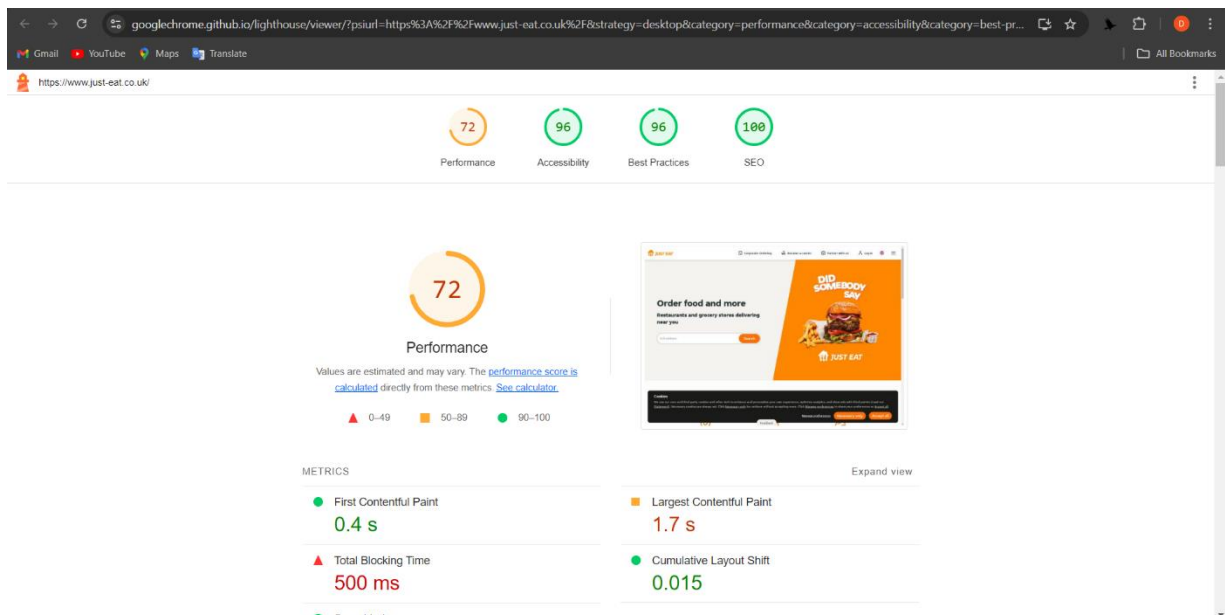


Figure 6 - Performance result of JUST EAT web application

and a complete report can be found in the pdf attached here: [..\MSc Project\JUST EAT lighthouse report \(web app\).pdf](#)

An article says this platform has noticed a major drop in overall sales by 4.9% and revenues reduced by 14% and the main reasons for this incident are crashes and glitches that occurred on the web application, crashes happen due to high traffic, bugs, coding errors, problems with database. Users also experience glitches such as distorted images, buttons not functioning etc all these issues negatively impact the web application also frustrates the users and can lead to loss of customers due to the web app's reliability. The following URLs below provide the articles about the drop in sales and revenue due to technical issues in JUST EAT application (Amber Murray, Wednesday 31 July 2024).

Despite of being a popular frontend framework, React has a lot of technical issues such as slow loading times because of large scripts, ineffective codes, and big pictures these issues are particularly noticeable during high traffic and when the platforms are accessed with weak internet connectivity. Furthermore, this chapter explains about another application developed with a different framework. Here is another example, an article shows a problem faced by an e-commerce store known as Sainsbury's developed using React, faced a technical glitch which affected the business and its customers (Alexander Butler, Saturday 16 March 2024).

### **2.3.3 jQuery – Frontend JavaScript library.**

jQuery is another JavaScript library which was designed to write less code and accomplish tasks, it achieved huge popularity because of its ability to make web development easier in comparison to plain JavaScript. There were other libraries during the launch of jQuery, its popularity was driven by strong community support and responsiveness to user feedback (Tran, October, 2020). However, the rise of modern JavaScript libraries and frameworks jQuery has lost its popularity since 2014 but not completely outdated. jQuery simplifies web development with essential set of features, specific focus on cross-browser compatibility, and plugin architecture, therefore using jQuery is simpler and efficient compared to JavaScript for maintaining events, manipulating DOMs, performing AJAX calls and producing animations (P.Desai, November, 2016). Despite being an efficient library, jQuery has faced technical issues especially in e-commerce stores such as online food ordering platforms (mobile and web applications). There are some limitations of jQuery like, it fails to enforce structure which causes unorganized code and makes it difficult to manage larger codebases, it is not suitable for larger applications and only well-suited for building projects that are small to medium size, it also faces performance issues, there has been a massive decline in usage of jQuery for developing applications due to the rise of modern frameworks, another reason is that jQuery has limited features compared to other advanced frameworks, functionalities such as fetching API for AJAX calls and query selectors can be achieved by using native JavaScript that is why developers consider using native JavaScript because similar result can be obtained without the use of jQuery.

For example, popular food ordering websites such as foodpanda and Delivery Hero are built using jQuery library, but they use old technology stack which causes performance issues like slow response times and scalability issues because jQuery has different architecture, which is imperative and monolithic, it was designed for client-side tasks and does not provide any support for server-side rendering or latest hydration approaches. It starts as an individual JavaScript bundle that performs execution within the browser. The image of an online food ordering platform called Foodpanda is built using jQuery and a tool known as Wappalyzer helps in displaying the technologies used to develop this application.



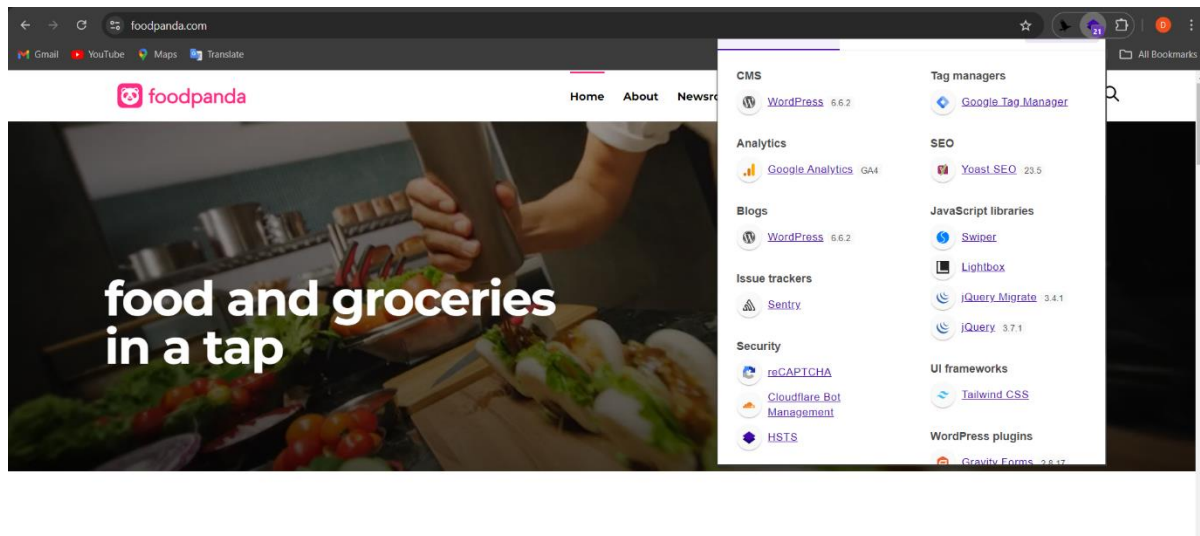


Figure 7 - Foodpanda landing page and wappalyzer detecting technologies used to develop it

The image below shows the performance result of the landing page using Google lighthouse:

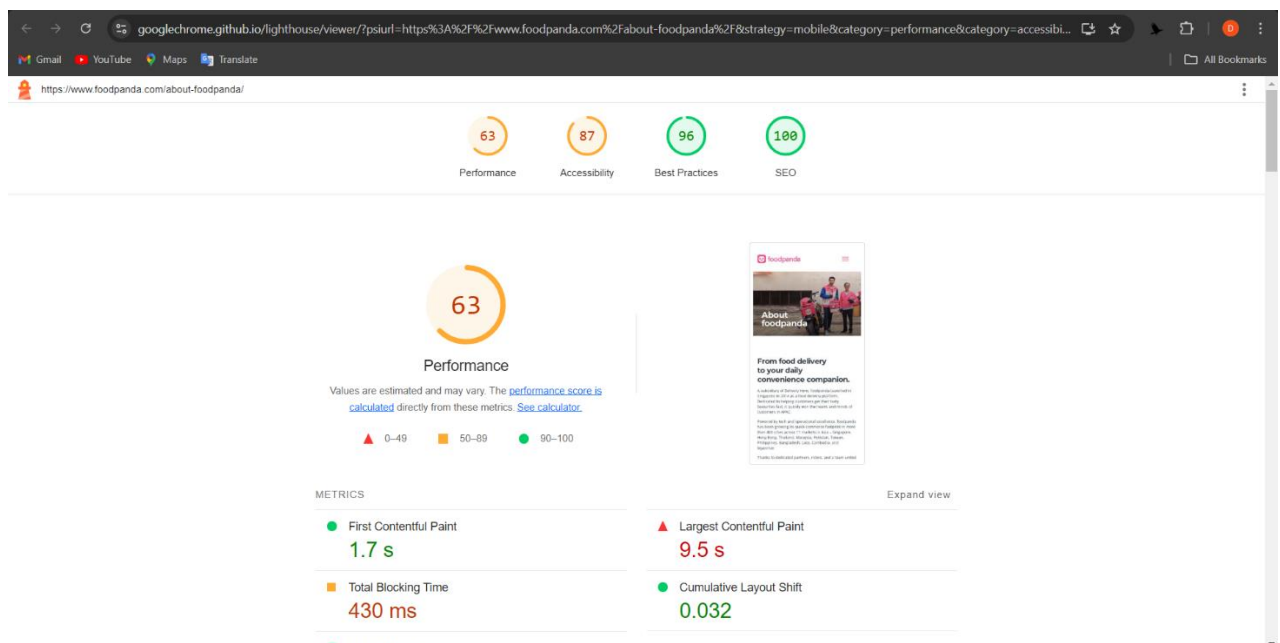


Figure 8 - Performance result of Foodpanda web application

and a complete report can be found in the pdf attached here: [..\MSc Project\Foodpanda\\_lighthouse report.pdf](#)

**2.4 Comparison table for frontend frameworks and libraries (Qwik, React, Angular and jQuery):**

Feature	Qwik	React	Angular	jQuery
Type	Framework	Library, and its ecosystem of tools make it a fully functioning framework.	Framework	Library
Style of architecture	The style of architecture for Qwik is component based and micro-frontend.	React has a component-based architecture.	Angular has a component-based architecture along with MVC.	jQuery has monolithic architecture.
Hydration	Qwik has resumable hydration.	React supports hydration.	Angular supports hydration.	Hydration is not applicable in jQuery.
Lazy loading feature	Qwik has built-in lazy loading feature.	React supports lazy loading and implemented by using React.lazy and suspense.	Lazy loading is supported through modules.	jQuery does not support lazy loading.
Community support	It has a small community which is still growing.	React has a very huge community.	It has strong community of developers, contributors.	jQuery has large community support.

Learning curve	Learning curve is steeper, particularly for beginners.	Learning curve is considered to be moderate.	Learning curve is considered to be high.	Learning curve is considered to be low and easier.
Core capabilities	Focuses on development of performance-based applications.	Focuses on developing user interface and component rendering.	Overall development of an application.	jQuery focuses on making tasks easier such as event handling and DOM manipulation.
Server-side Rendering (SSR)	Excellent support.	React supports SSR but it does not have built-in functionality, therefore frameworks like Next.js enables server-side rendering for applications.	Angular supports SSR using tool known as Angular Universal.	Server-side rendering is not supported in jQuery.

*Table 1 - Comparison table for frontend frameworks and libraries*

## 2.5 The table shows the benefits of Qwik over other frameworks:

Feature	Qwik	Angular	React	jQuery
Resumable hydration	✓			
Minimal JavaScript	✓			
Lazy loading	✓	✓		

SEO friendly	✓	✓		
Micro-frontends	✓			

*Table 2 - Benefits of Qwik over other frameworks*

## 2.6 Social context:

Digital Evolution: Due to rapid digital change that is happening in industries has resulted in development of web applications, industries are looking to enhance their online visibility to offer better user experiences. Therefore, this study addresses the necessity of frameworks that will help the transformation by reducing load times and optimizing overall performance, the aim is to provide strong support that guarantees faster, smoother experience to fulfil demands of technological environments. By allowing smaller restaurants to have an online presence, the concept tackles the issue of the digital divide. This inclusion is necessary for ensuring that all restaurants/shops regardless of their size can take part in the digital economy and this inclusion is essential. By reducing in-person contact during the ordering process, the online ordering system can improve public health and safety. This is especially significant when considering ongoing health issues or pandemic (MARANG, 2018).

Expectations of users: In this generation users have expectations of accessing information/data or any other online services easily and instantly. Whenever users interact with web applications, they expect quick loading and high performance but if its slow it will cause negative impact, users can get frustrated with slow loading web applications and they might stop using such web applications because of its poor performance therefore it is necessary to optimize the performance of web applications to ensure it runs smoothly, load quickly and keep users engaged without getting frustrated hence it is necessary for user satisfaction.

## 2.7 Ethical context:

During the development of web applications, it is necessary to think about ethics and equality it's not just about making the application to run faster but also ensuring it is accessible to everyone including people with disabilities so that they can use it with different devices and internet speed this is because Qwik's architecture is designed in such a way that can be accessible using different types of devices whether it is old or new. This work highlights how developers have an ethical obligation to develop inclusive web applications that serve a wide range of users. Moreover, it's important to keep users' data secured and protecting their privacy and efficient frameworks can help in minimizing vulnerabilities by simplified codebase thereby enhances security measures (F S Grodzinsky, 2003).

Developers have the responsibility to guarantee that the web applications are efficient and don't cause negative impact by consuming excessive data therefore, to enhance user experience and decreasing overall data footprint resumability handles this issue by allowing web applications to download less JavaScript in advance, this aligns with ethical principles of accountability and responsibility when using technology.

## **2.8 Legal context:**

Following data protection and consumer rights rules is necessary for the development and implementation of a web application such as online ordering platform from a legal perspective. Strict data protection measures are required by law such as the CCPA (California Consumer Privacy Act) in USA and the GDPR (General Data Protection Regulation) in Europe, utilising Qwik may allow enhanced data management and security procedures, that can help to ensure compliance. Additionally following the food safety and delivery regulations is important to guarantee that the online ordering system functions within the legal guidelines. Intellectual property rights must be taken into consideration when developing software, branding, and content for a digital platform, and the sustainability of the company depends on these rights being protected (Mafalda Ferreira, 2023).

## **2.9 Professional context:**

From a professional perspective, developing this system using Qwik has the ability to set the new standard for web development strategies. It supports the use of innovative and efficient frameworks that emphasises user experience and performance. Web developers may consider this work as an example and reference that shows the perks of novel frameworks over traditional ones in real life applications. Additionally, it offers to the body of knowledge in web development by giving insights and strategies for upcoming projects in similar domains.

Regarding software development, customer support, and user experience design, the online ordering system should be created keeping with industry best practices and guidelines. Applying these guidelines enhances the reputation and professionalism of restaurant chain. Professional developers and organisations face pressure to develop high-performing applications that have to fulfil user expectations. Resumability adoption can lead to better resource management and lower operating expenses, as applications grow efficiently. Due to users' growing demand for applications that provide quick and excellent experiences, this can enhance a company's reputation and competition in the market (JUHO VEPSÄLÄINEN, 11 January 2024).

For professional career growth in the tech industry, it is essential to understand the nuances of multiple frameworks and how they impact performance. Decision-making process in project management and development roles may benefit from the valuable insights of the research.

Professionals may enhance their skills and marketability in a competitive job market by the analysis of frameworks like as React.js, Qwik and Next.js for a comparative understanding.

### **3.0 Matchmaking online food ordering system - Application overview and design**

This chapter and following sections of this thesis document discusses about the Matchmaking online food ordering application that allows users to order food, buy agricultural products from the local farmers, personalized diets, and food donations by utilizing Qwik framework. The sections of this chapter explain all the steps in the development of this web application that includes database, architecture, frontend and backend implementation the sections will also discuss the objectives, motivation, design, architecture and technical stacks, this acts as the basis for further implementation.

#### **3.1 Motivation and Objectives:**

This thesis focuses on developing the system using unique technologies, addressing and eliminating the technical issues, enhancing performance and scalability. There are limitations that are faced by the existing online food ordering platforms, these existing platforms are dealing with problems such as slow loading, poor performance, poor scalability and poor user experience hence this has direct impact on business and customer satisfaction. Therefore, my web application resolves the technical issues by providing instant loading, best performance, and more scalable so this solution enhances and improves the overall experience.

Matchmaking online food ordering application is inspired by existing online food ordering platforms like Just Eat, Uber Eats, Deliveroo etc and the motivation is to create something new that can handle technical issues, order the products from the web app by eliminating slow loading time, crashes and glitches that leads to customer satisfaction. This concept has been motivated by various factors:

- **Technological Feasibility:** The idea of this web application is clear and novel, this is developed by using the available technology and resources. Furthermore, this concept of web application which faces high traffic is well-suited for utilising the Qwik framework.
- **Wide Audience Appeal:** Psychologically, people are drawn to the comfort of ordering food through online platforms because it is convenient, additionally features like customers that seek personalized meals based on their dietary and nutritional needs. Furthermore, features such as food donations helps to support charities and communities. In addition, features like customers who prefer freshness and environmental sustainability like to purchase farm products from farmers that directly benefits the local economy. Also offering home-cooked meal draws attention of

customers who seek traditional food, comfort, and health benefits. Hence all these features make this concept highly desirable and appealing to a wide range of customers.

The ultimate objective of this project is to investigate the performance, instant loading by comparing this web application with other existing online food ordering platforms, and to also investigate if Qwik framework is reliable to build such high traffic online ordering web app. This project aims to determine the framework's reliability by providing evidence like performance report using google lighthouse to support the advantages of utilising Qwik. Furthermore, this chapter discusses the user flow of the web application and software design.

### **3.2 User Flow: Matchmaking online food order web application**

In this section the user flow of the web application will be discussed, it begins when users visit the homepage of Matchmaking online food order web application here, they'll have an option to log in as an existing customer or must register as a new member by providing their address details, information like email and phone number. Existing members will be able to login directly, once everything is confirmed it allows users to enter the site and search for their preferred restaurants, location, browse menu, adding to the cart, after this customers will proceed to checkout and confirm their order details once again they'll get to see delivery information, in the next step they'll get to choose their payment method for example cash on delivery or credit/debit card whichever is convenient for them, after selecting the payment method orders will be officially placed and the customers will be notified with a order confirmation message.

Additionally, the remaining features such as:

- Food donations, buying organic farm products, ordering personalised diet, and buying home cooked meals will have the same order placement procedure.
- For example, customers can buy their personalised diets, any farm products, home cooked meals, and add it to their cart, they'll also have various options to donate food. Hence these features will follow the same user flow as regular food ordering feature.

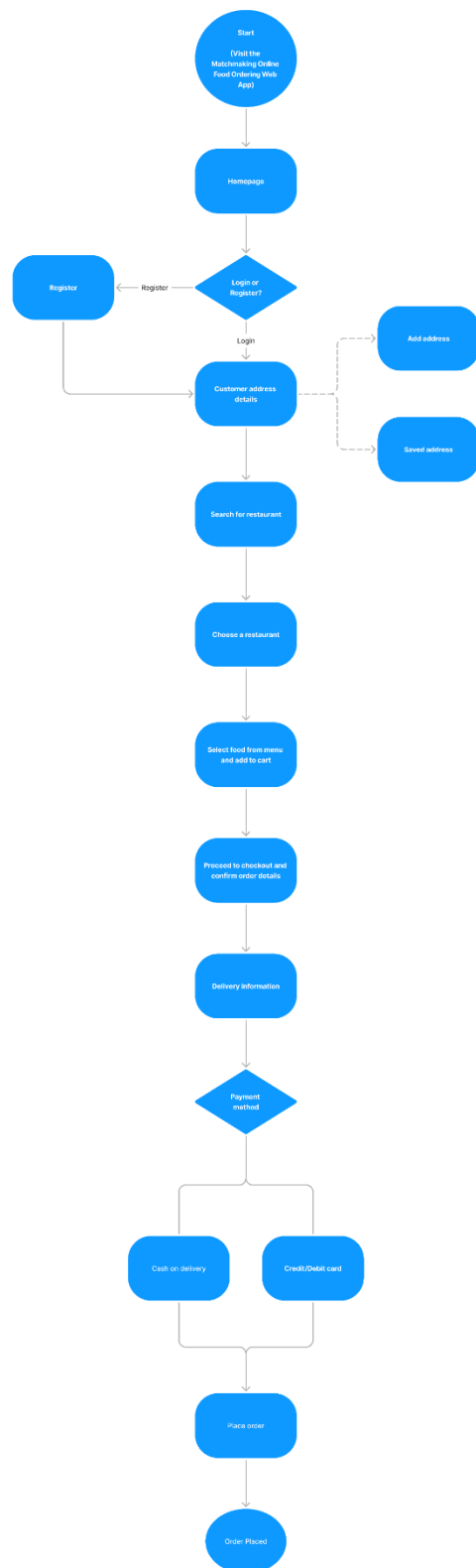


Figure 9 – User flow for Matchmaking online food ordering system



## Matchmaking online food ordering web app

Deepak Gowda

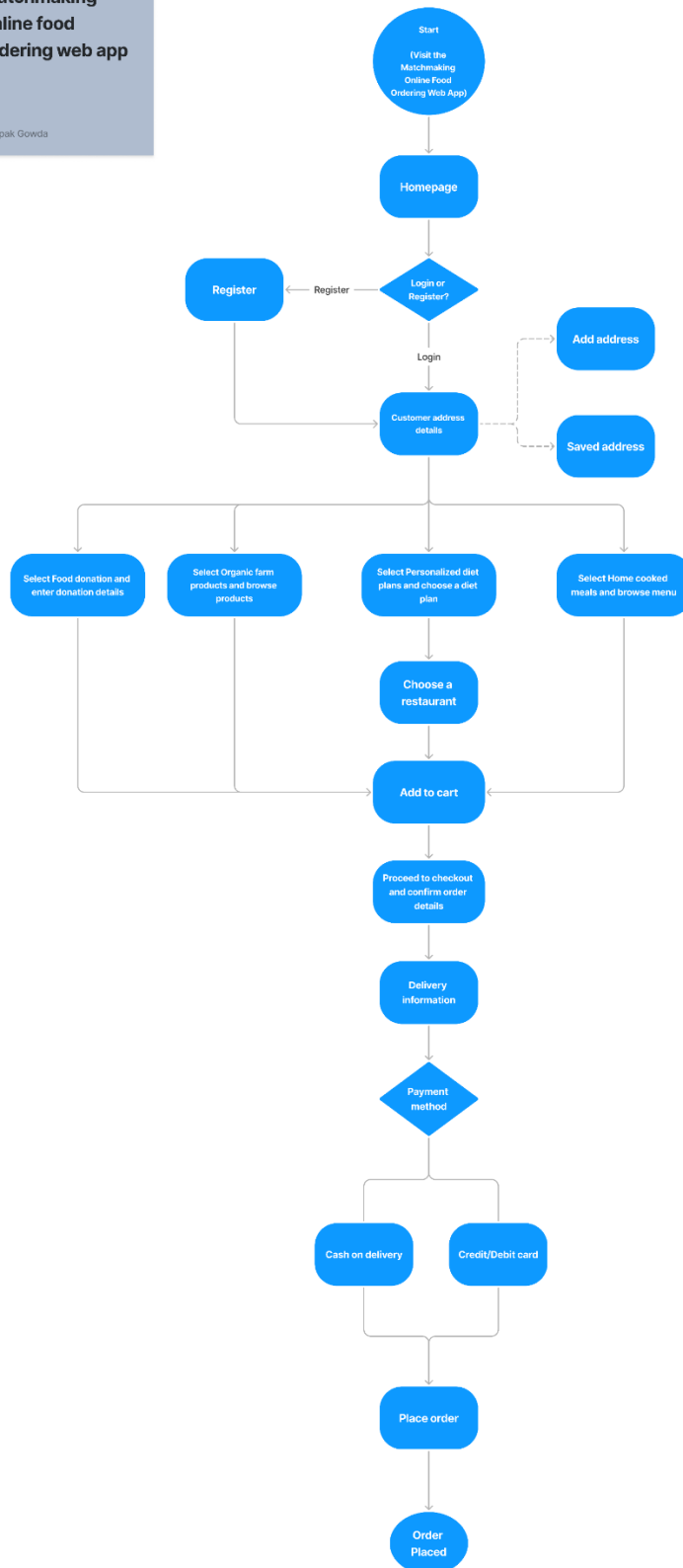


Figure10 –User flow for Matchmaking online food ordering system with extra features

### 3.3 Requirements Analysis:

Functional requirements	Non-Functional requirements
<ul style="list-style-type: none"><li>• <b>Customer authorisation and authentication:</b> Must apply user authentication and authorisation mechanisms that will manage user sessions.</li></ul>	<ul style="list-style-type: none"><li>• <b>Performance:</b> The web application must be able to handle high traffic, provide instant loading and maintain performance during peak usage.</li></ul>
<ul style="list-style-type: none"><li>• <b>Real-Time updates:</b> Its necessary to apply real-time updates to notify the client about their order confirmation, food preparation and delivery. Hence this will keep them updated about each step throughout the process.</li></ul>	<ul style="list-style-type: none"><li>• <b>Security:</b> The web application must protect and secure the customer data, payment details. It must implement solid encryption for data, safeguard authentication and authorisation protocols.</li></ul>
<ul style="list-style-type: none"><li>• <b>API:</b> Using RESTful API to manage CRUD operations for important resources such as customer profile, restaurant information, order information, feedback.</li></ul>	<ul style="list-style-type: none"><li>• <b>User Experience:</b> The design must focus and prioritize on user experience with swift interactions along with reduced load times, by taking advantage of Qwik's capabilities for rapid interactivity.</li></ul>
<ul style="list-style-type: none"><li>• <b>Database:</b> The web application must efficiently handle and store customer data, preferences, menu, history of orders. To retrieve and update this data the system must allow efficient queries and make sure that customers get fast response.</li></ul>	<ul style="list-style-type: none"><li>• <b>Scalability:</b> To handle services like customer management, order processing the web application must be designed utilising a microservices architecture, all services should be individually scalable and deployable, assuring the web application is able to handle high loads by scaling services as required and not impacting the overall performance.</li></ul>

<ul style="list-style-type: none"> <li>• <b>Frontend-Development:</b> Implementing frontend using Qwik framework, and ensuring UI is responsive to make it usable in all kinds of devices.</li> </ul>	<ul style="list-style-type: none"> <li>• <b>Extensibility:</b> The web application should be designed in a way that allows extensibility, also must allow additional features and integrations without interruption to the current functionality.</li> </ul>
---	--

*Table 3 - Requirements Analysis*

### 3.4 Technologies used to build this application:

The Matchmaking online food ordering application is built using several modern technology stack which is designed to achieve high performance, scalability and handle high traffic, the technologies used to develop this system are:

- Frontend technologies: HTML/CSS, TypeScript, Qwik framework, Qwik City, Vite.
- Backend technologies: Node.js, Express.js.
- Database: MongoDB.
- Performance monitoring and testing: Google Lighthouse.
- Technology Profiler: Wappalyzer.

## **4 Implementation of Matchmaking online food ordering application:**

The application has been developed using Qwik framework to show the benefits of this cutting-edge web development approach regarding performance, scalability, and traffic management. Several sections of this chapter explain and investigate implementation procedure, database design, backend implementation, frontend implementation, admin panel, and code snippets, it also provides insights and technical knowledge about the development process of the Matchmaking online food ordering system. All the sections in this chapter aim to provide complete understanding of the application along with necessary details.

### **4.1 Qwik City**

A combination of technologies such as Qwik City and vite have been used in this web application to achieve high performance and manage application's routing along with layout structure. Qwik City is a meta framework which is built on top of Qwik because it provides file-based routing which provides effortless and efficient arrangements of routes for different parts of the system which includes food ordering, customer browsing with other features, hence this removed the absolute need for configuration of complex route thus permitted faster development as the application expanded (Lonka, 30.6.2023).

static site generation (SSG) helped to serve pre-rendered HTML and reducing the necessity for runtime JavaScript. It also provides Server-side rendering (SSR) to improve Search engine operation and help in loading pages faster by pre-rendering them on server. Integration with layouts nested routes help developers to efficiently organize pages and eliminate redundancy as much as possible specifically in bigger applications. The flexible nature of Qwik City has assisted the web app to create shared layouts for many components of the application, which involves customer dashboard, restaurant information, and other results. Nested routing has made easier management of pages with dynamic content which includes restaurant menus and personalised diet preferences thus simplifying the scalability of the web app's structure whenever new features are added. Therefore, this has made the development process more enhanced by providing automatic route handling and also includes built-in optimization that is for both server-side and client-side rendering (Noor, 2024).

### **4.2 Vite**

Vite is a robust tool with a rich plugin ecosystem that has enabled my web application to integrate with TypeScript, JSX and has played a vital role in enhancing overall performance of the matchmaking online food ordering system. Whilst working on the Qwik framework and Qwik City to develop the frontend, Vite supported immediate reflection of code modifications in JavaScript, HTML, CSS in the browser, without the necessity of reloading the full page. Hence this significantly dropped development time and increased productivity by enabling fast iterations along with testing. When the food ordering application was being developed, Vite has provided a quick cold-start time, even though there was an increase in the complexity of the application. Vite's helped in generating faster page speeds by its capability to split and

optimise bundles based on the usage which is essential for a web application for handling real-time food orders and delivery tracking (Evan You & Vite Contributors, n.d.).

### 4.3 Implementation of the frontend and code structure – Qwik Framework

This section shows the code structure and code snippets to display the working of the frontend using Qwik framework. The image shows the frontend structure which includes node modules, src which contains assets, components, context, pages, routes. In this project structure, we can notice ‘vite.config.ts’ in the configuration section it means that Vite has been used which is a developing environment and a new tool basically used to provide fast performance, speed and has the ability to update changes without the necessary of refreshing whole page. Another image shows the key files and directories like components, routes, root.tsx file which will be discussed in the next sections of this chapter.

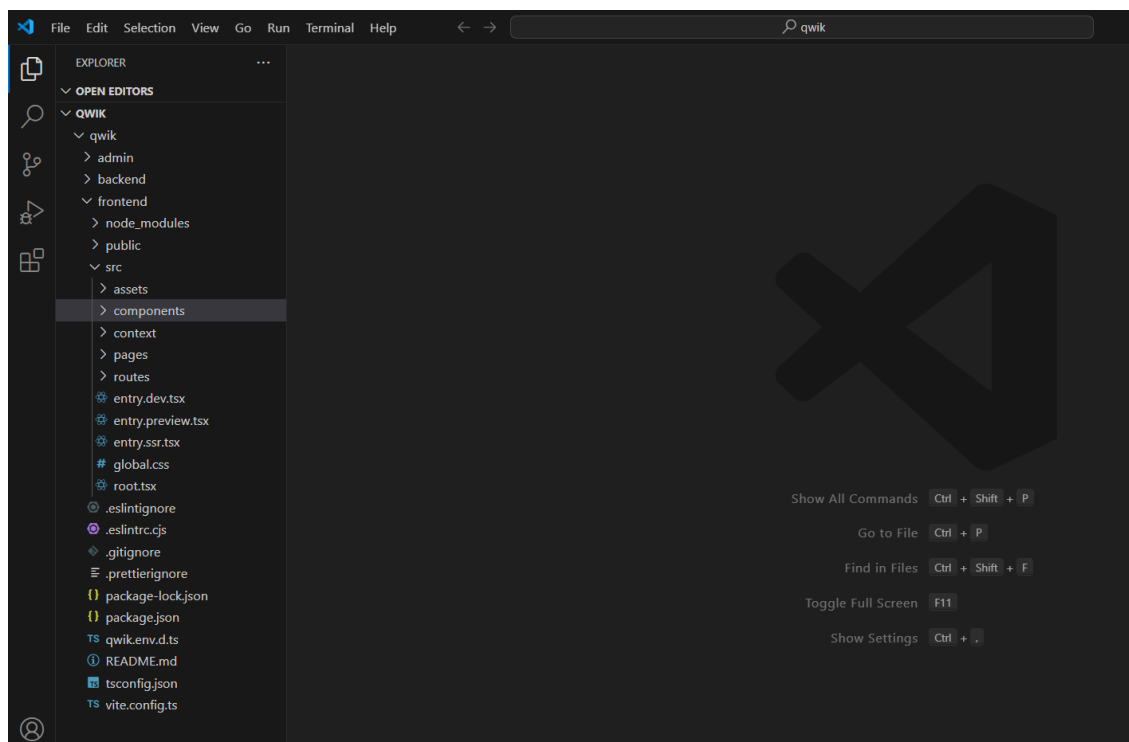
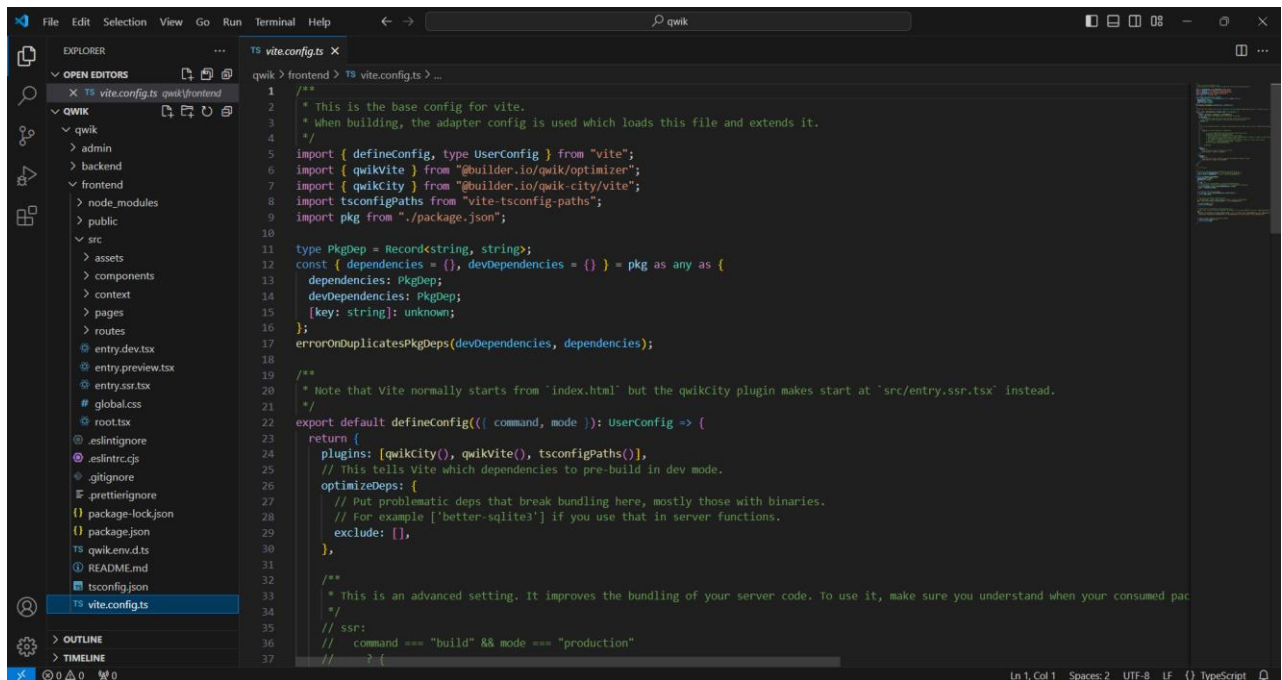


Figure 11 – Frontend code structure



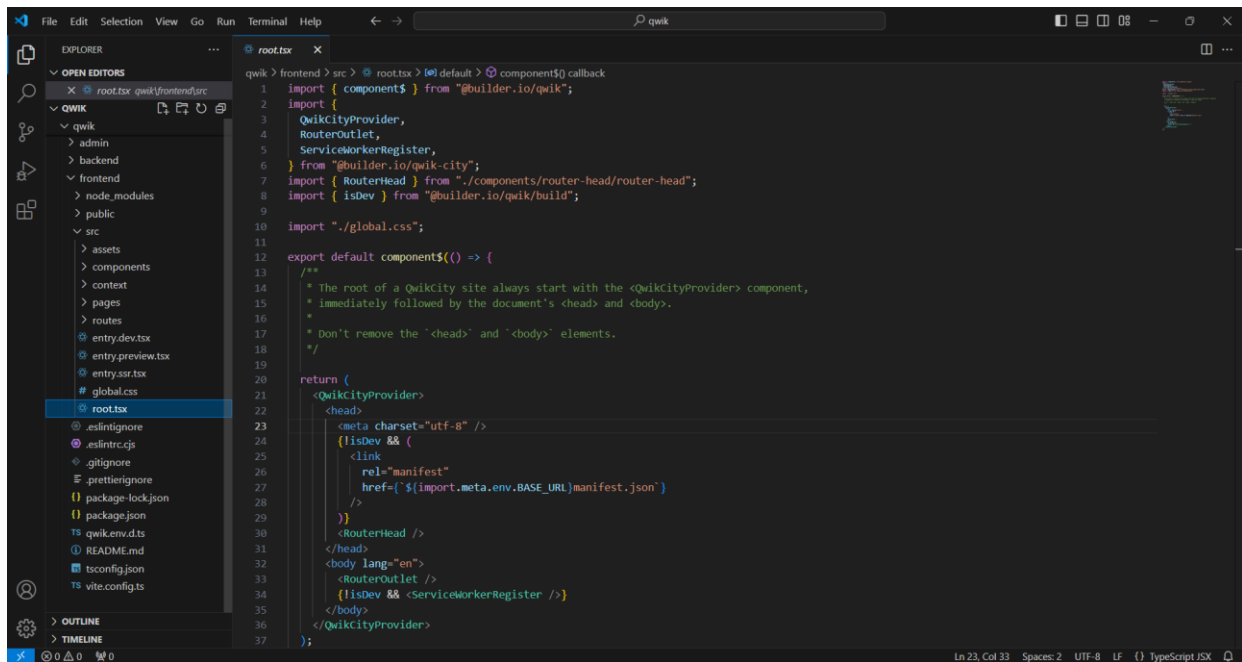
*Code Snippet: vite.config.ts*

The components include several key elements which make up the user interface of this application, such as:

- 1) ExploreMenu: The typescript file includes logic and structure, and this file is responsible for displaying several menu options.
- 2) FoodDisplay: This is responsible for displaying food items.
- 3) FoodItem: Provides numerous dishes with their information.
- 4) Header and Navbar: These elements provide navigation.
- 5) LoginPopup: This element is responsible for managing user authentication.
- 6) Router-head: It helps in boosting overall functionality of the application and provides proper routing including page head configurations.

## 4.4 Root and entry files

The root.tsx file acts as the major entry point for a QwikCity application, creating the core structure and vital components of the application. The root.tsx file uses the component\$ function provided by the Qwik framework to indicate a component that renders the application's layout in general. The application is wrapped within a <QwikCityProvider> component, which is important for supporting routing and extra functionalities being provided by QwikCity.



*Code Snippet: root.tsx file*

The `<QwikCityProvider>` sets up the document's `<head>` and `<body>` elements. The `<head>` element includes meta information like the charset and conditionally involves a link on the app's manifest file if the application is no longer in development mode (`! isDev`). Additionally, the `RouterHead` component has also been added which handles an extra metadata such as meta tags and page title for social sharing and SEO.

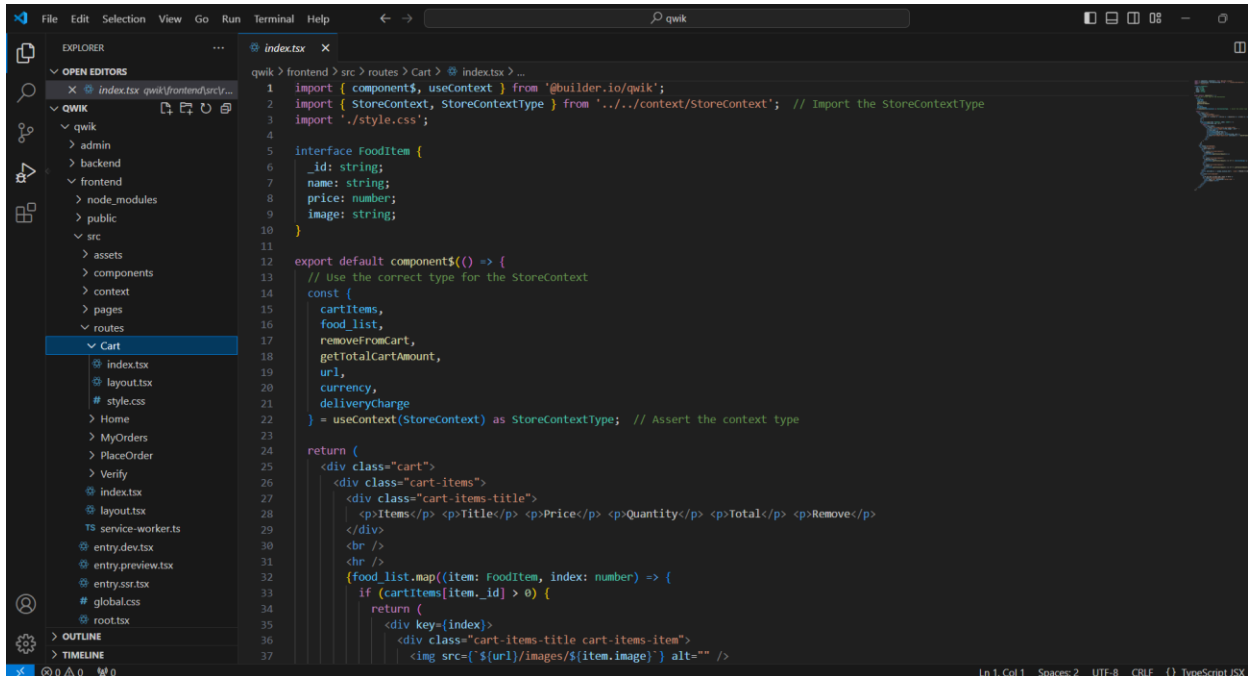
The `<body>` element includes the `RouterOutlet` component, which acts as a placeholder for rendering the appropriate route component based on the current URL. Assuming the app is not in development mode then it registers a service worker through the component called `ServiceWorkerRegister`, which allows features such as caching and offline support. The import `"./global.css"` involves global CSS styles which is required for the application, ensuring uniform styling all throughout components. Hence this file is necessary for creating the app's structure, dynamic web app functionalities and routing.

## 4.5 Routes

Routing in Qwik is necessary to navigate between different pages or views in a single-page app (SPA). Qwik utilises a built-in router for client-side navigation. It allows developers to define routes that align with different components or pages within the application. The router enables navigation without the necessity of reloading an entire page. Therefore, enhancing user experience through seamless and faster transitions. Routes in Qwik tend to be defined in a configuration file or instantaneously within the directory structure. To simplify configuration process, router automatically connects URL pathways to particular components. The routes folder has different components such as cart, home, my orders, place order and verify, furthermore the code snippets will explain how each of them works. The index.tsx files that are

present in every component within routes folder is responsible for delivering the display logic and functionality for the specific pages.

The code snippet below displays the index.tsx file in the Cart folder:



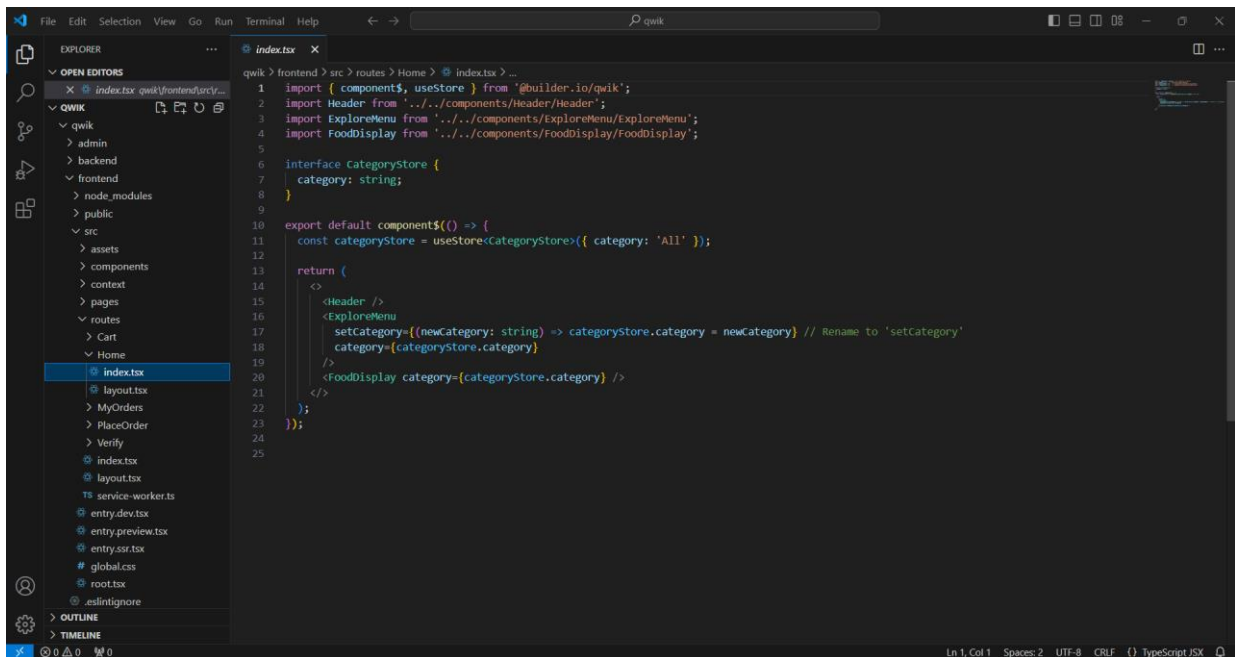
```
1 import { components$, useContext } from '@builder.io/qwik';
2 import { StoreContext, StoreContextType } from '../../context/StoreContext'; // Import the StoreContextType
3 import './style.css';
4
5 interface FoodItem {
6   _id: string;
7   name: string;
8   price: number;
9   image: string;
10 }
11
12 export default components$(() => {
13   // Use the correct type for the StoreContext
14   const {
15     cartItems,
16     food_list,
17     removeFromCart,
18     getTotalCartAmount,
19     url,
20     currency,
21     deliveryCharge
22   } = useContext(StoreContext) as StoreContextType; // Assert the context type
23
24   return (
25     <div class="cart">
26       <div class="cart-items">
27         <div class="cart-items-title">
28           <p>Items</p> <p>Title</p> <p>Price</p> <p>Quantity</p> <p>Total</p> <p>Remove</p>
29         </div>
30         <br />
31         <br />
32         {food_list.map((item: FoodItem, index: number) => {
33           if (cartItems[item._id] > 0) {
34             return (
35               <div key={index}>
36                 <div class="cart-items-title cart-items-item">
37                   <img src={` ${url}/images/${item.image}` } alt="" />
```

*Code Snippet: index.tsx in Cart folder*

This code discusses a Qwik component called Cart which displays a shopping cart using data provided by the StoreContext. It imports required dependencies including the StoreContextType for context type-checking. The component collects cart-related information which includes food list, cart items and payment from the context. It provides a collection of food items in the cart which shows the food name, cost, quantity, and overall cost, and also provides an option to remove products then calculates and shows the overall amount which includes delivery fee. Furthermore, the component utilises an integration of context like TSX and CSS to style and handle the contents of shopping cart.



The code snippet below displays the index.tsx file in the Home folder:

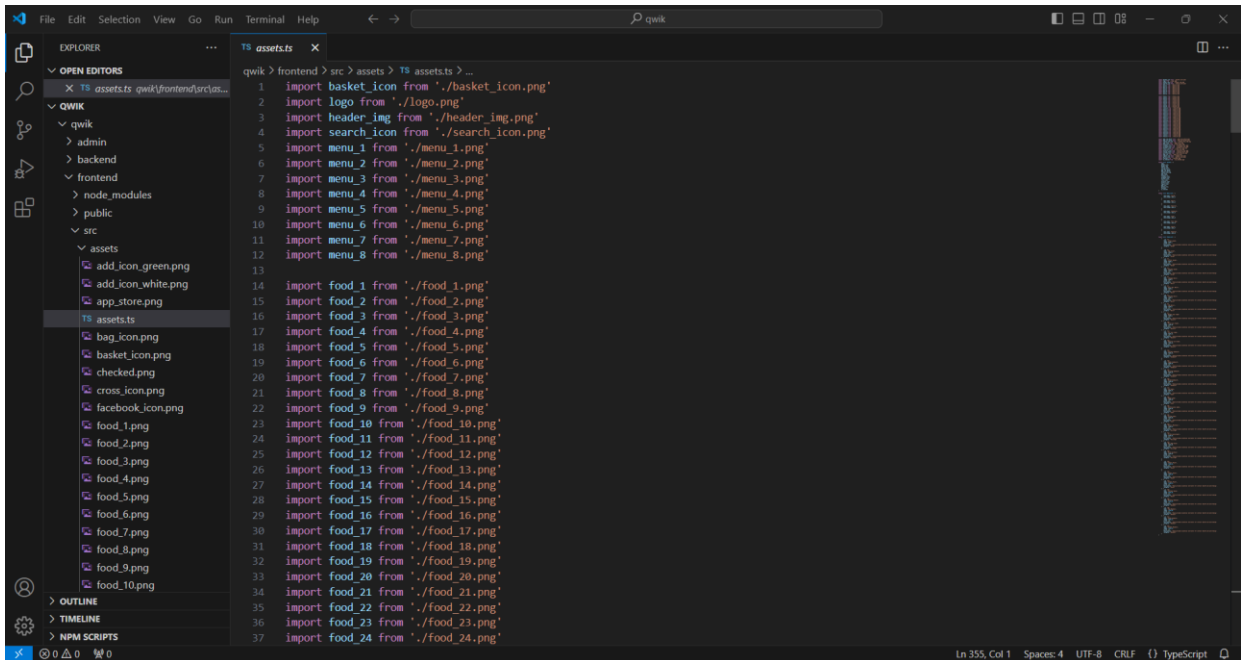


```
1 import { component$, useStore } from '@builder.io/qwik';
2 import Header from '../../components/Header/Header';
3 import ExploreMenu from '../../components/ExploreMenu/ExploreMenu';
4 import FoodDisplay from '../../components/FoodDisplay/FoodDisplay';
5
6 interface CategoryStore {
7   category: string;
8 }
9
10 export default component$(() => {
11   const categoryStore = useStore<CategoryStore>({ category: 'All' });
12
13   return (
14     <>
15       <Header />
16       <ExploreMenu
17         setCategory={(newCategory: string) => categoryStore.category = newCategory} // Rename to 'setCategory'
18         category={categoryStore.category}
19       />
20       <FoodDisplay category={categoryStore.category} />
21     </>
22   );
23 });
```

*Code Snippet: index.tsx in Home folder*

This code describes a Qwik component that handles and shows food items classified by category. It uses Qwik's `component$` and `useStore` APIs to create a dynamic `categoryStore` object. The component gives back a fragment (`<>`) which includes a `Header` component, `FoodDisplay`, and also an `ExploreMenu` component. The `ExploreMenu` component permits the user to alter the category state in `categoryStore` by using the function called `setCategory`, and that will update the category property. So, the category which is updated gets delivered to the `FoodDisplay` component and depending on the selected category it dynamically shows food items. The code helps in organizing the layout, which helps in showing food items in a cohesive and engaging manner.

The code snippet below displays the assets.ts file, which is present in src, and this file is necessary to organize and manage static resources like images:



```
1 import basket_icon from './basket_icon.png'
2 import logo from './logo.png'
3 import header_img from './header_img.png'
4 import search_icon from './search_icon.png'
5 import menu_1 from './menu_1.png'
6 import menu_2 from './menu_2.png'
7 import menu_3 from './menu_3.png'
8 import menu_4 from './menu_4.png'
9 import menu_5 from './menu_5.png'
10 import menu_6 from './menu_6.png'
11 import menu_7 from './menu_7.png'
12 import menu_8 from './menu_8.png'
13
14 import food_1 from './food_1.png'
15 import food_2 from './food_2.png'
16 import food_3 from './food_3.png'
17 import food_4 from './food_4.png'
18 import food_5 from './food_5.png'
19 import food_6 from './food_6.png'
20 import food_7 from './food_7.png'
21 import food_8 from './food_8.png'
22 import food_9 from './food_9.png'
23 import food_10 from './food_10.png'
24 import food_11 from './food_11.png'
25 import food_12 from './food_12.png'
26 import food_13 from './food_13.png'
27 import food_14 from './food_14.png'
28 import food_15 from './food_15.png'
29 import food_16 from './food_16.png'
30 import food_17 from './food_17.png'
31 import food_18 from './food_18.png'
32 import food_19 from './food_19.png'
33 import food_20 from './food_20.png'
34 import food_21 from './food_21.png'
35 import food_22 from './food_22.png'
36 import food_23 from './food_23.png'
37 import food_24 from './food_24.png'
```

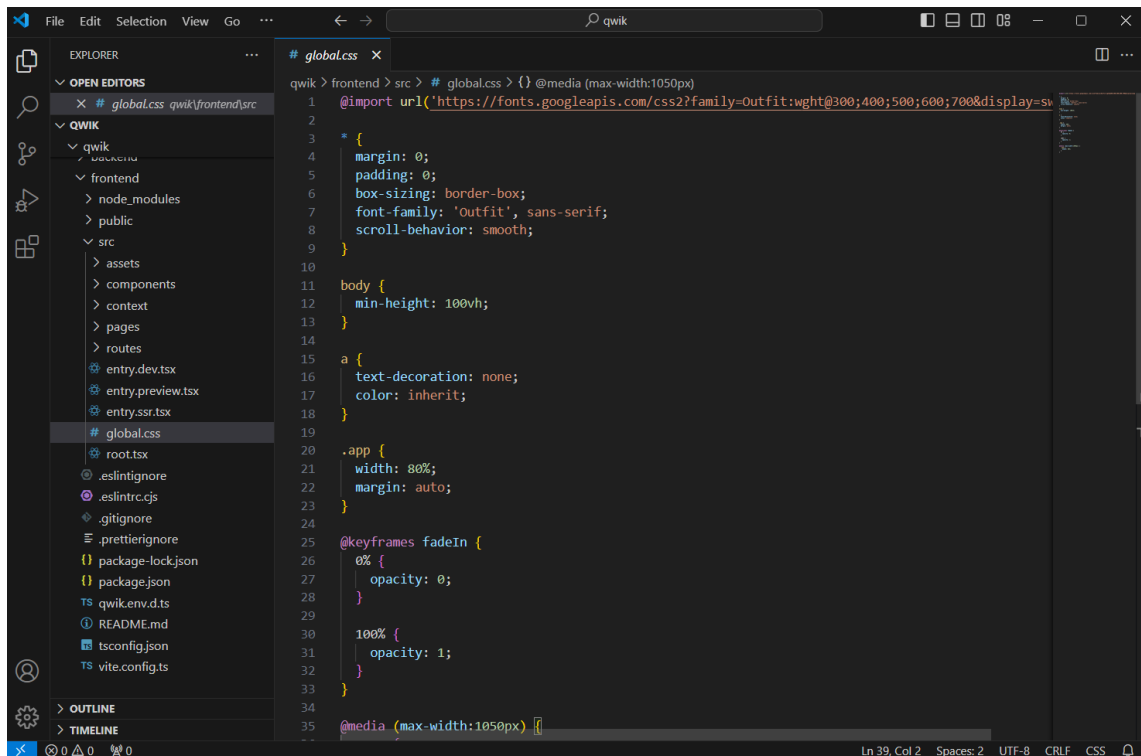
*Code Snippet: assets.ts*

The assets.ts file performs as a centralised module which is used to import and handle several picture assets which are used to build this application, by putting them in an appropriate and user-friendly format. It starts with importing separate images like logo.png, header\_img.png, basket\_icon.png and variety of menu, each connected to a certain graphical element inside the application. The assets file consists of typically used images and visuals, such as the application icon, basket icon, search icon, and several social media images, hence helping them to be more accessible which can be used across various components of the application.

## 4.6 Global.css

This global.css file helps in providing essential global styles for developing a web application. It imports the 'Outfit' typeface from Google Fonts in multiple weights and applies it uniformly throughout the site. 'Outfit' font is downloaded from Google fonts with different font thickness, and it gets applied universally all over the website.

The code snippet below displays the global.css file in src

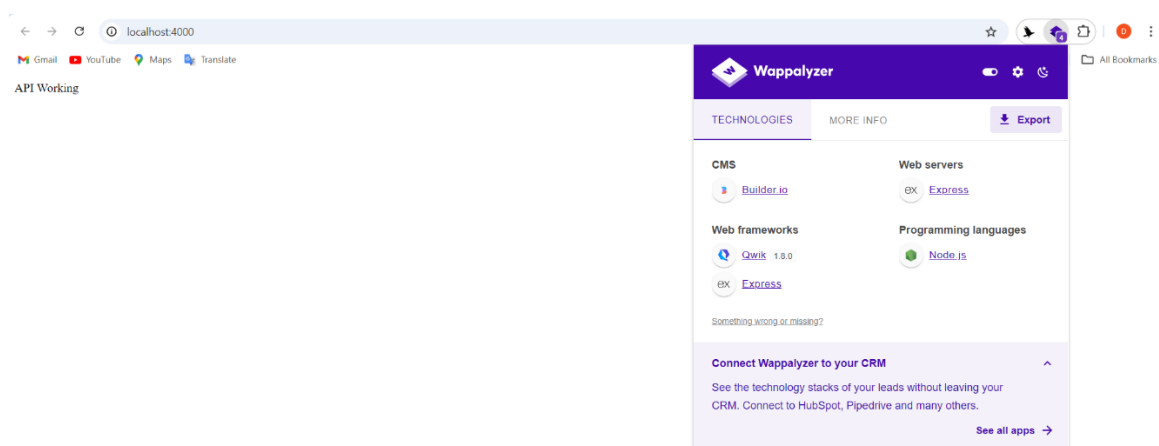


```
1 @import url('https://fonts.googleapis.com/css2?family=Outfit:wght@300;400;500;600;700&display=swap');
2
3 * {
4   margin: 0;
5   padding: 0;
6   box-sizing: border-box;
7   font-family: 'Outfit', sans-serif;
8   scroll-behavior: smooth;
9 }
10
11 body {
12   min-height: 100vh;
13 }
14
15 a {
16   text-decoration: none;
17   color: inherit;
18 }
19
20 .app {
21   width: 80%;
22   margin: auto;
23 }
24
25 @keyframes fadeIn {
26   0% {
27     opacity: 0;
28   }
29
30   100% {
31     opacity: 1;
32   }
33 }
34
35 @media (max-width:1050px) {
```

*Code Snippet: global.css*

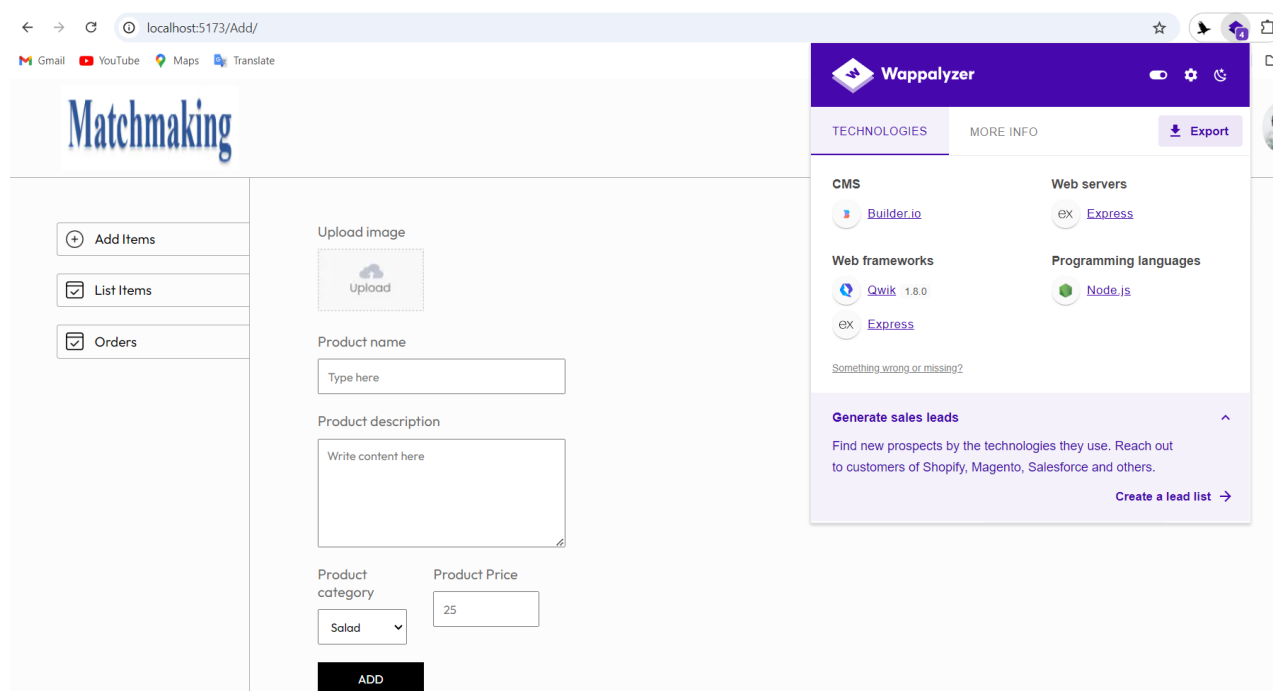
**The visual representation of the Matchmaking online food ordering system which is developed using Qwik framework and other technologies:**

The image below shows the backend of the Matchmaking online food ordering system has been successfully configured and functioning, the API server was tested locally on the localhost:4000 and displays a message “API Working”.



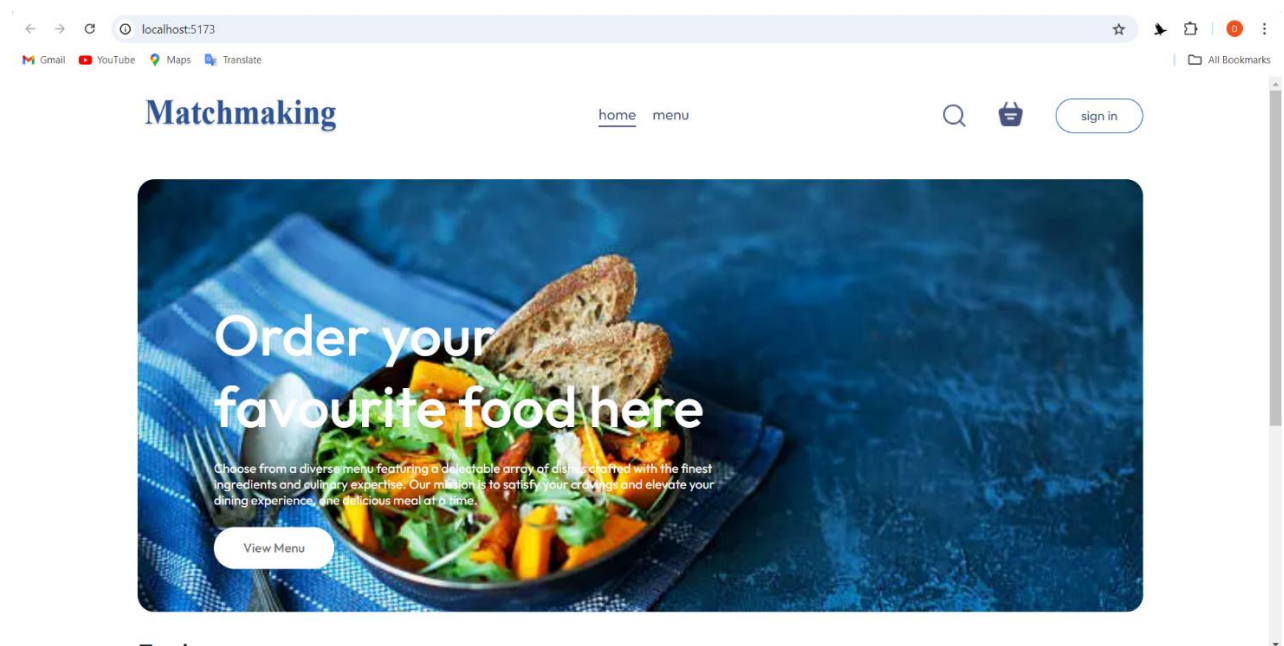
*Figure 12 - Matchmaking online food ordering system API server functioning with Wappalyzer tech stack detection*

**Admin panel:** It displays features such as Add, List items and Orders. (Wappalyzer has been used to identify the technologies which is used to build Matchmaking online food ordering system)



*Figure 13 - Matchmaking online food ordering system admin panel*

The image below shows the web application's landing page, along with initial user interface elements, variety of features and highlights the responsive design.



*Figure 14 - Matchmaking online food ordering system landing page*

The images below display the login and login component code of the application.

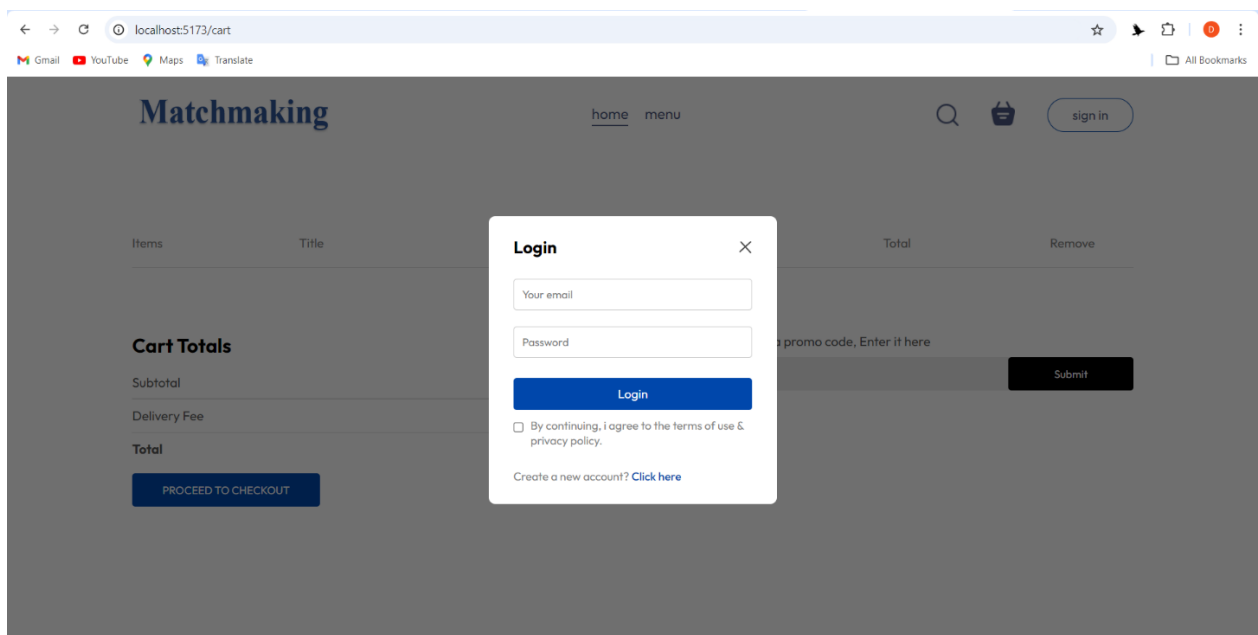
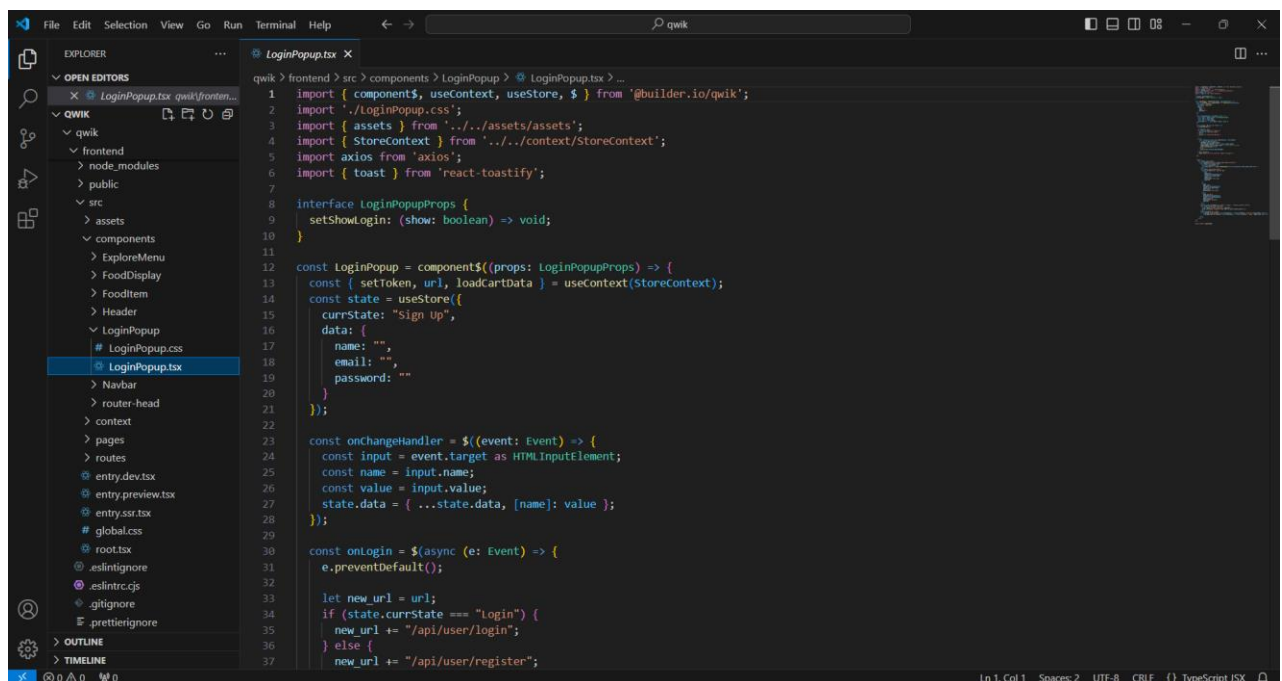


Figure 15 - Matchmaking online food ordering system login page



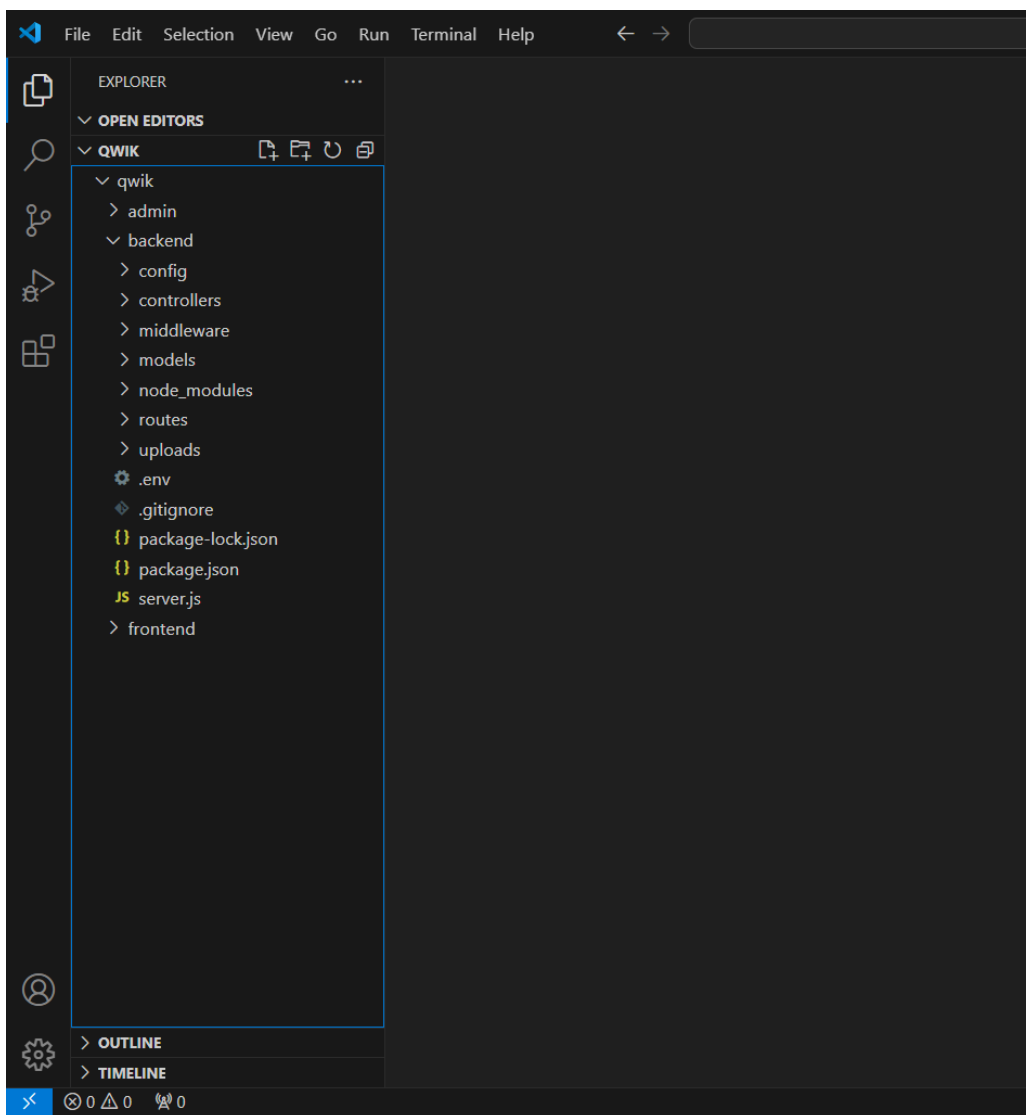
Code Snippet: LoginPopup.tsx

## 4.7 Backend Implementation

The application is developed using Node.js and Express which provides robust server-side architecture and avoids scalability issues. Node.js supports efficient management of operations that are asynchronous meanwhile Express is responsible for managing API and simplifies

routing. Therefore, both the technologies provide reliable and secure communication among backend and frontend of the web application. To create a RESTful API, Node.js and Express has been utilised to handle backend logic for managing user information, order details, matchmaking algorithms and database communication. This API has also enabled integration with Qwik City server-side features and facilitating server-rendered responses. MongoDB has been used for the backend data management in this web application, which is used to store restaurant information, user data, delivery address, order histories. The adaptable schema of MongoDB enabled the efficient management of complex data connections which is required for connecting users to restaurant suggestions and selecting food on their personal choices (Le, 2023).

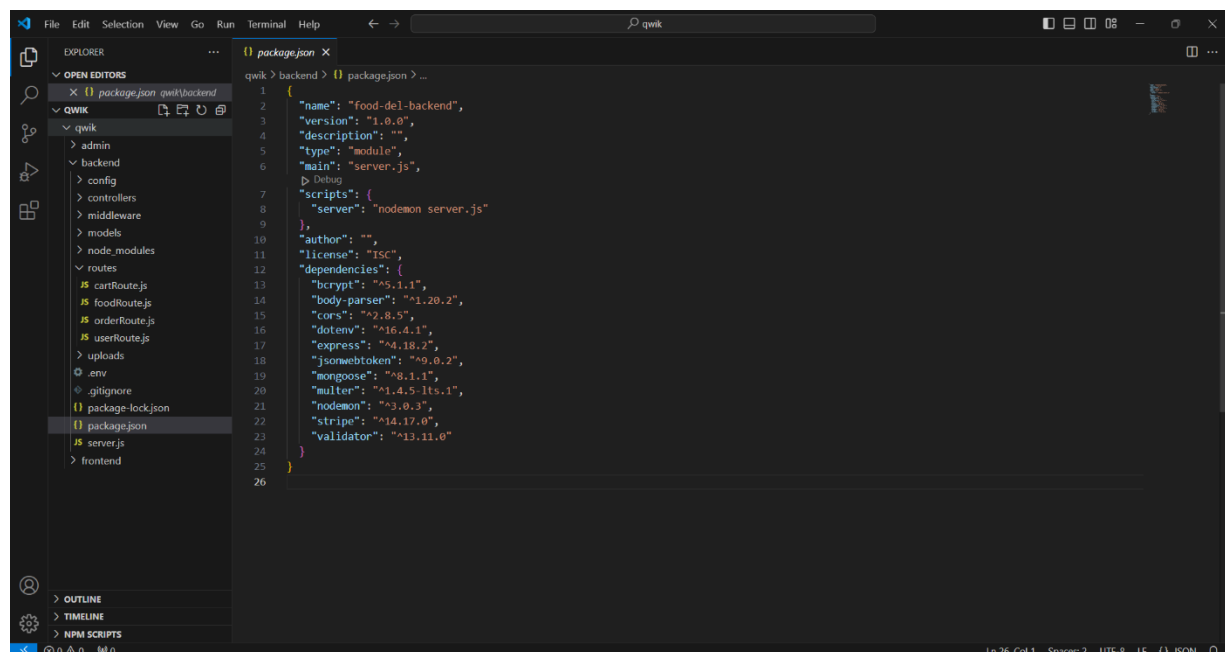
Here is the backend code structure of Matchmaking online food ordering system.



*Figure 16 – Backend code structure*

The backend contains number of folders and files, they provide insights of the backend implementation like libraries, frameworks and integration of database. Since this is a web-based project it has node\_modules installed. This section focuses on examining and understanding the backend implementation. The application is developed by using backend framing called Express.js which is used mainly to set up REST APIs and managing HTTP requests.

The package.json code below describes the key libraries and structure which is necessary for developing and executing the food delivery web application in an effective way.

A screenshot of a code editor interface. On the left, the 'EXPLORER' sidebar shows a file tree with folders like 'admin', 'backend', 'config', 'controllers', 'middleware', 'models', 'node\_modules', 'routes', 'uploads', and files like '.env', '.gitignore', 'package-lock.json', 'package.json', 'server.js', and 'frontend'. The 'package.json' file is selected and its content is displayed in the main editor area. The code is a JSON object with fields for name, version, description, type, main, scripts, author, license, dependencies, and devDependencies. The dependencies include bcrypt, body-parser, cors, dotenv, express, jsonwebtoken, mongoose, multer, nodemon, stripe, and validator. The scripts section shows a 'server' script that runs 'nodemon server.js'.

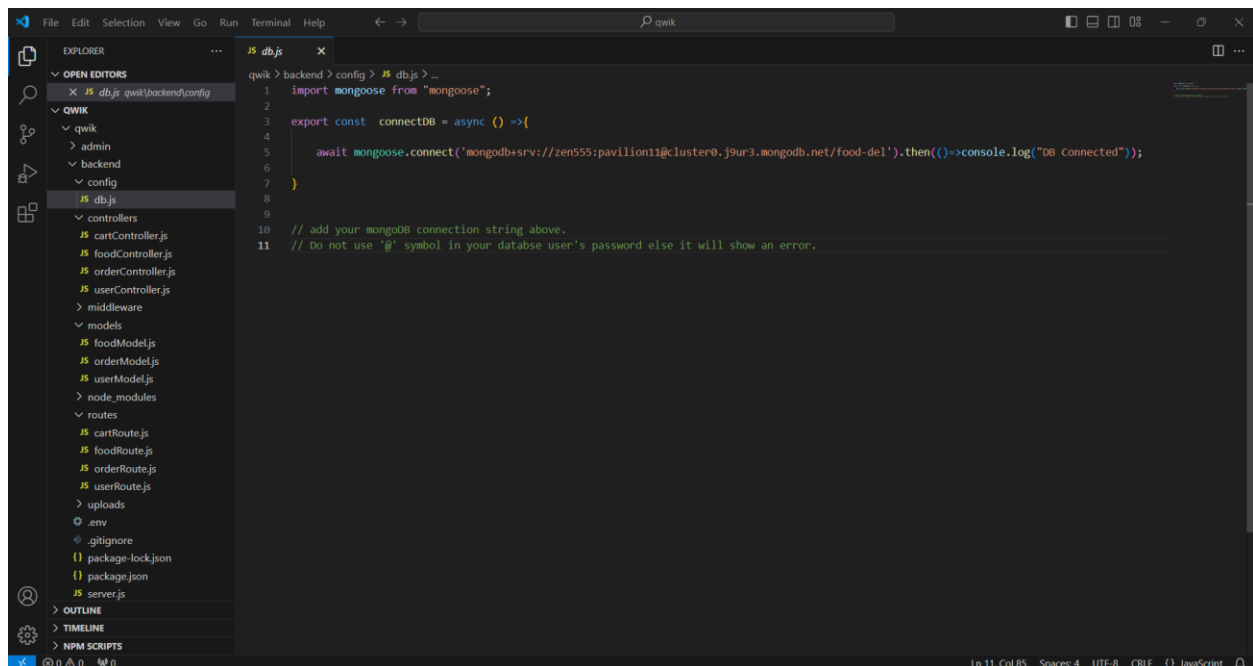
```
1 {
2   "name": "food-del-backend",
3   "version": "1.0.0",
4   "description": "",
5   "type": "module",
6   "main": "server.js",
7   "scripts": {
8     "server": "nodemon server.js"
9   },
10  "author": "",
11  "license": "ISC",
12  "dependencies": {
13    "bcrypt": "^5.1.1",
14    "body-parser": "^1.20.2",
15    "cors": "^2.8.5",
16    "dotenv": "^16.4.1",
17    "express": "^4.18.2",
18    "jsonwebtoken": "^9.0.2",
19    "mongoose": "^8.1.1",
20    "multer": "^1.4.5-lts.1",
21    "nodemon": "^3.0.3",
22    "stripe": "^14.17.0",
23    "validator": "^13.11.0"
24  }
25 }
```

*Code Snippet: package.json*

Package.json defines the most basic metadata and configuration of the project. The title: "module" suggests that the project utilises ECMAScript modules (ESM) instead of CommonJS. server.js is the main access point for the backend server. The scripts portion includes a command which is used to start the server by making the use of a tool called Nodemon, it can instantly restart the application whenever the changes are identified. This web application is dependent on multiple external libraries that are described in the dependencies section. crucial dependencies use express for the setup of server, and mongoose is used to interact with MongoDB database, also bcrypt for having safeguarded passwords. Jsonwebtoken has been used for generating and verification of JSON Web Tokens, because it's important for authentication. Multer handles file uploads, whereas Stripe assists payment processing. The usage of dotenv enables the handling of environment variables, whilst validator provides data validation utilities. The parsing of incoming request bodies is supported by Body-parser, whereas CORS allows Cross-Origin Resource Sharing. Eventually, this web application uses nodemon to enhance development by detecting file alterations and instantly relaunching the server.

## 4.7.1 Config - db.js

This code displays the method of connecting a Node.js application to a MongoDB database by making use of the mongoose library which is an Object Data Modelling (ODM) tool for MongoDB.



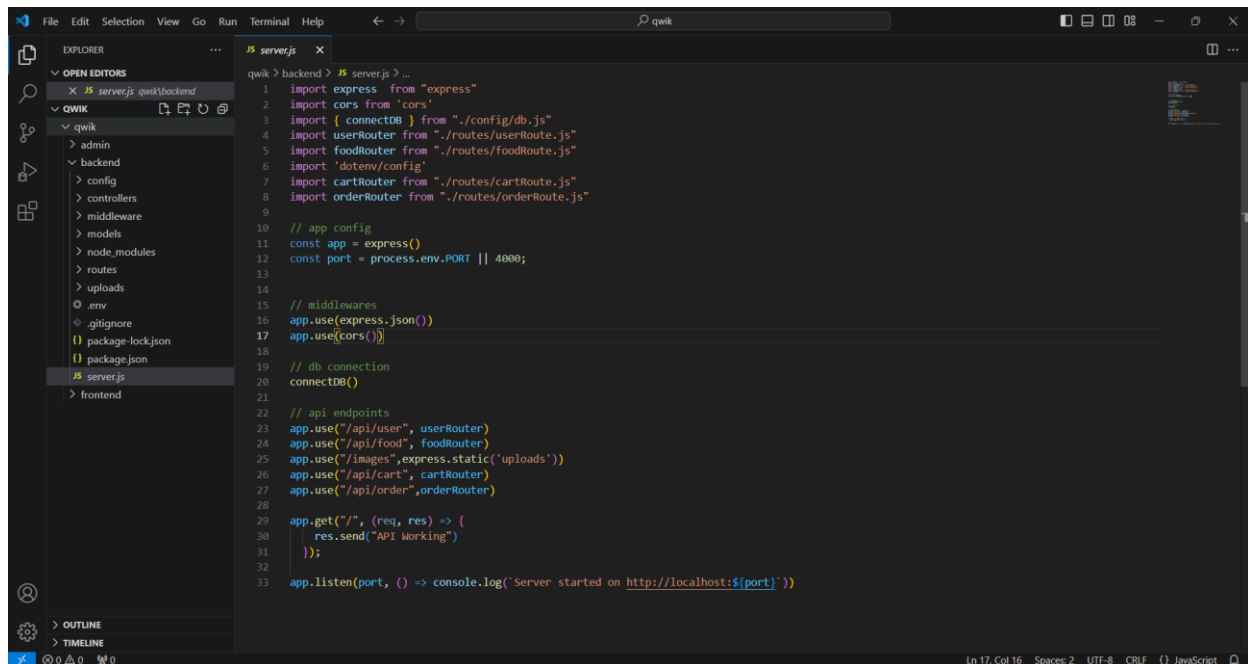
```
1 import mongoose from "mongoose";
2
3 export const connectDB = async () => {
4
5     await mongoose.connect('mongodb+srv://zen555:pavilion11@cluster0.j9ur3.mongodb.net/food-del?');
6
7 }
8
9
10 // add your mongoDB connection string above.
11 // Do not use '@' symbol in your database user's password else it will show an error.
```

*Code Snippet: db.js*

- 1) `import mongoose from "mongoose";`; This statement imports the Mongoose library because of its necessity for interacting with MongoDB from a Node.js application. Mongoose enables to define schemas and models for MongoDB collections; hence this helps to structure data interactions.
- 2) `export const connectDB = async () => {...};`; This statement is an asynchronous function whereas `connectDB` is assigned to build a connection to the MongoDB database. By making the use of `async` and `await` it indicates that the connection attempt has been solved before moving on to the further process in the application.
- 3) `mongoose.connect(...)`: This explains that within the function the `mongoose.connect()` method is utilised for making a connection to the MongoDB instance. (`mongodb+srv://...`) is the connection string that is provided it contains the credentials (`zen555:pavilion11`) and links to the MongoDB cluster which is hosted on MongoDB Atlas.
- 4) `.then(()=>console.log("DB Connected"));`; This explains that when the connection is successful, it displays a message that shows in the terminal as "DB Connected".
- 5) It is recommended to not use `@` symbol in the database user's password, because it causes errors in the connection string.



## 4.7.2 Server.js



```
1 import express from "express"
2 import cors from "cors"
3 import { connectDB } from "../config/db.js"
4 import userRouter from "../routes/userRoute.js"
5 import foodRouter from "../routes/foodRoute.js"
6 import 'dotenv/config'
7 import cartRouter from "../routes/cartRoute.js"
8 import orderRouter from "../routes/orderRoute.js"
9
10 // app config
11 const app = express()
12 const port = process.env.PORT || 4000;
13
14 // middlewares
15 app.use(express.json())
16 app.use(cors())
17
18 // db connection
19 connectDB()
20
21 // api endpoints
22 app.use("/api/user", userRouter)
23 app.use("/api/food", foodRouter)
24 app.use("/images", express.static('uploads'))
25 app.use("/api/cart", cartRouter)
26 app.use("/api/order", orderRouter)
27
28 app.get("/", (req, res) => {
29   res.send("API Working")
30 });
31
32 app.listen(port, () => console.log(`Server started on http://localhost:${port}`))
```

*Code Snippet: server.js*

This is the code snippet of server.js which acts as the setup for the web application by using Express, it also includes many crucial features such as routing, database connectivity and middleware. In the beginning, many modules are imported: for creating server Express is used, and then cors is used for handling cross-origin requests, and dotenv to handle environment variable management. The connectDB function is imported from the db.js file, enables the setting up of a connection to the MongoDB database. On top of that, multiple routers such as foodRouter, cartRouter, userRouter, orderRouter are imported to handle various API routes for food items, cart, users, and orders. The application configuration starts by performing initialisation of an Express instance and the description of the server's port and this can be obtained from environment variables (process.env.PORT) or default of 4000. Middleware functions are then established. To parse JSON data, expressjson() is being used and cors() permits for managing requests from different origins so that the API can be accessed beyond domains. Then the database connection is set up by using the connectDB() function. The endpoints of API are set up which includes routes for users, orders, cart, food items and also includes a static route for displaying images obtained from the uploads folder. Furthermore, the server starts running and listens for incoming requests on the specified port, a simple test endpoint ("/") confirms the functionality of API, later a console log message displays "server started on http://localhost:\${port}" which means the server has successfully started and can be accessed.

## 5 Evaluation:

The Matchmaking online food ordering system has been developed using Qwik which is an emerging framework, it has the potential to provide high performance, handle scalability issues, improve user interfaces, this framework can solve the issues and challenges that is faced by other modern frameworks such as React, Angular, jQuery and Vue.js. Therefore, this section evaluates the research and development of the application based on the objectives set regarding the performance, comparison with other frameworks and technologies used.

### 5.1 Experimental results on the performance of the Matchmaking online food ordering system in comparison to other technologies.

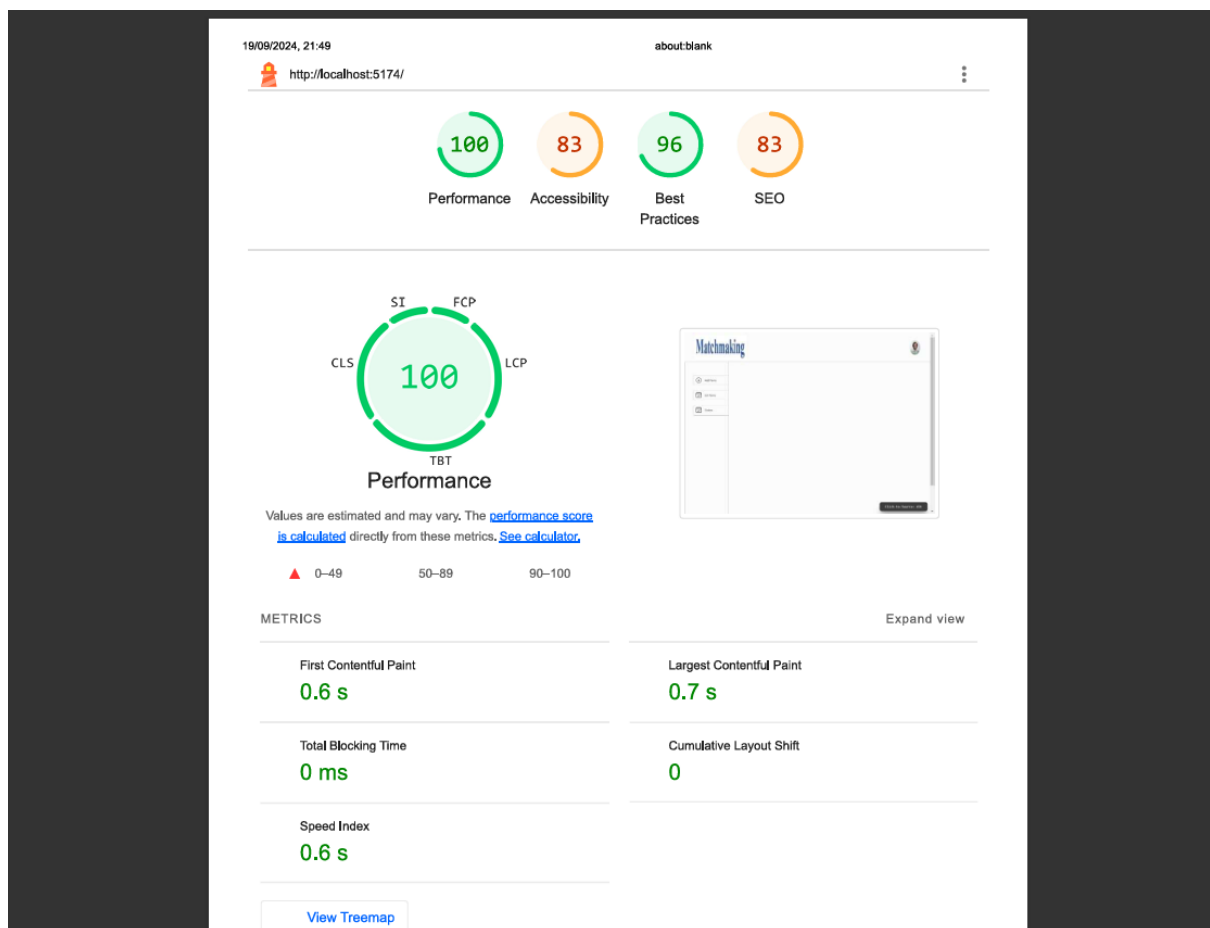


Figure 17 – Performance result of admin panel using Google lighthouse

And a complete report can be found in the pdf attached here: [..\MSc Project\Admin Lighthouse report.pdf](#)

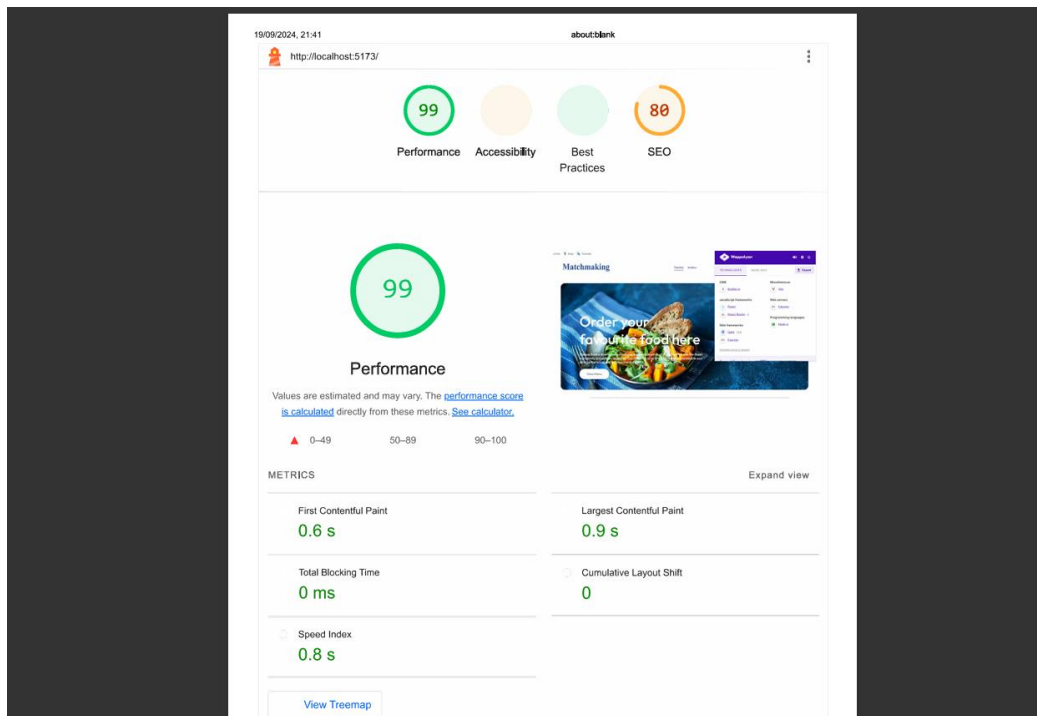


Figure 18 - Performance result of the landing page by using Google lighthouse and a complete report can be found in the pdf attached here: [..\..\Downloads\Landingpage\\_lighthouse report \(Qwik framework\).pdf.pdf](..\..\Downloads\Landingpage_lighthouse report (Qwik framework).pdf.pdf)

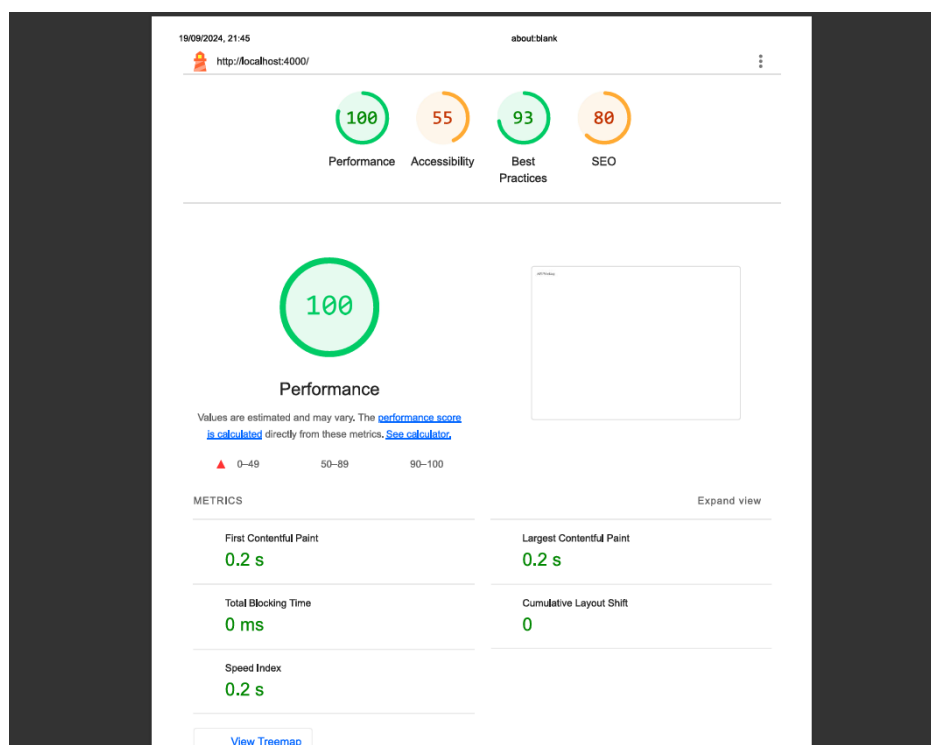


Figure 19 - Performance result of backend API by using Google lighthouse and a complete report can be found in the pdf attached here: [..\MSc Project\Backend\\_Lighthouse report.pdf](..\MSc Project\Backend_Lighthouse report.pdf)

## 5.2 Future work:

This thesis meets most of its objectives but there are many areas which requires attention, importance and development such as:

- **Recommendation AI:** Utilising AI recommendation can boost the user experience by recommending dishes, restaurants, and dietary requirements depending on the customer's previous preferences and orders. Machine learning models inspect large amounts of datasets of feedback, customer interactions and food/restaurant choices thus offering personalised user experience. The function of the AI is to suggest specific meals, products or any other services to the customers without human interaction along with this Natural Language Processing (NLP) is a machine learning technology which can be used to understand and help to resolve customer issues in a human-like manner thus it can improve customer satisfaction and engagement.
- **Additional features for the Matchmaking online food ordering web application:** Currently the application has basic food ordering feature like choosing restaurants, browse menu, placing order. However, the area of future improvements involves personalized diet plan, food donation and organic farm products, all these functionalities can be added to the application. Food donation feature can use geolocation services

## 6 Conclusion

In conclusion, the development of Matchmaking online food ordering system has produced significant improvements in terms of performance and scalability by using Qwik framework along with other tech stack, this research and development has proved that Qwik is an efficient and reliable solution in the field of web development especially comparing it to traditional frameworks such as React, Angular, jQuery, etc. due to their limitations in terms of handling high traffic, scalability issues, and performance particularly e-commerce platforms such as food ordering applications. Special features such as lazy loading, instant loading, and resumability enhanced user experience through decreased load times which is necessary for user satisfaction. The thesis draws attention to the positive impacts of Qwik's ability in decreasing JavaScript payloads and boosting the responsiveness of web-based applications. The performance results obtained through technologies such as Google Lighthouse proves Qwik framework's ability to excellent performance especially compared to other food ordering platforms developed with modern frameworks. This research and development offer resources and possibilities for future work which involves integration with AI recommendation systems, SEO, food donations, organic farm products thus enhance the platform's contribution.

## References

- Adam Lipiński, B.P. (2023) 'Performance optimization of web applications using Qwik', *Journal of Computer Sciences Institute* pp.1-3.
- Alexander Butler (Saturday 16 March 2024) *Sainsbury's and Tesco technical issues: Supermarket CEO apologises after glitch hits online food deliveries*. Available at: <https://www.independent.co.uk/news/uk/home-news/sainsburys-tesco-technical-problems-online-deliveries-payments-b2513702.html> (Accessed: 11 September 2024).
- Amber Murray (Wednesday 31 July 2024) *Just Eat: Shares spike, but is there trouble ahead?* Available at: <https://www.cityam.com/just-eat-shares-spike-but-is-there-trouble-ahead/> (Accessed: 11 September 2024).
- BALIEVA, G. (July 2023) 'Online Food Purchasing During COVID-19 Pandemic', *Scientific Papers Series Management, Economic Engineering in Agriculture and Rural Development*, Vol. 23( Issue 2), pp.51-52.
- Cao, X.-A. (2023) 'Headless CMS and Qwik Framework ' pp.43-47.
- Evan You & Vite Contributors (n.d.) *Vite*. Available at: <https://vitejs.dev/guide/why.html> (Accessed: 11 September 2024).
- F S Grodzinsky, K.M.M.J.W. (2003) 'Ethical Issues in Open Source Software', *Info, Comm & Ethics in Society* , 1(193–205), pp.195-97.
- G.Geetha, M.M.D.K.M.P.D.J.G.P. (2022) 'Interpretation and Analysis of Angular Framework ', *International Conference on Power, Energy, Control and Transmission Systems (ICPECTS)* ( 978-1-6654-6275-4/22), pp.1-6.
- Hamed Hussen Ben kora1, M.S.M. ( 15-06-2024 ) 'Modern Front-End Web Architecture Using React.js and Next.js', *Univ Zawia J Eng Sci Technol* pp.1-4.
- JUHO VEPSÄLÄINEN, M.A.P.V. ( 11 January 2024) ' Resumability—A New Primitive for Developing', VOLUME 12, pp. 9039-9043.
- JUHO VEPSÄLÄINEN, M.A.P.V. ( 11 January 2024) 'Resumability—A New Primitive for Developing Web Applications', VOLUME 12, pp.9042-45.
- JUHO VEPSÄLÄINEN, M.A.P.V. (11 January 2024) 'Resumability—A New Primitive for Developing Web Applications', VOLUME 12, pp.9042-45.
- JUHO VEPSÄLÄINEN, M.A.P.V. (11 January 2024) 'Resumability—A New Primitive for Developing Web Applications', VOLUME 12, pp.9042-45.
- JuhoPaakkanen ( December22,2023) ' Upcoming JavaScript web frameworks and their techniques', *Bachelor's Programme in Science and Technology* pp.19-21.
- Le, D.A. (2023) 'E-Commercial Full Stack Web Application Development ' pp.10-25.

- Lonka, T. (30.6.2023) 'Improving the Initial Rendering Performance of React Applications Through Contemporary Rendering Approaches', *Computer, Communication and Information Sciences* pp.30-31.
- Mafalda Ferreira, T.B.J.e.F.S.N.S. (2023) 'RuleKeeper: GDPR-Aware Personal Data Compliance for Web Frameworks', *IEEE Symposium on Security and Privacy (SP)* pp.2817-19.
- MARANG, A.Z. (2018) 'Analysis of web performance optimization and its impact on user experience' pp.3-8.
- Noor, J.H. (2024) 'The Effects of Architectural Design Decisions on Framework Adoption: A Comparative Evaluation of Meta-Frameworks in Modern Web Development', *Computer, Communication and Information Sciences (CCIS)* pp.39-46.
- Ovidiu Constantin NOVAC, D.E.M.C.M.N.G.B.M.O.T.G. (2021) 'Comparative study of some applications made in the Angular and Vue.js frameworks ', *16th International Conference on Engineering of Modern Electric Systems (EMES)* pp.1-2.
- P.Desai, M.V. ( November, 2016) 'Jquery- An Interactive Web Designing Tool in Web Domain', *International Journal of Academic Research* pp.112-15.
- Paakkanen, J. ( December 22, 2023) ' Upcoming JavaScript web frameworks and their techniques', *Bachelor's Programme in Science and Technology* pp.8-10.
- Paipo Tang, David Nguyen (August 30, 2022) *DoorDash's Lessons on Improving Performance on High-Traffic Web Pages*. Available at: <https://careers.doordash.com/blog/doordashes-lessons-on-improving-performance-on-high-traffic-web-pages/> (Accessed: 11 September 2024).
- Rahul Sonwane, A.D.S.C. (2023) ' Designing Web Application of Online Food Ordering for Restaurant Chain using Web Technologies', *3rd International Conference on Pervasive Computing and Social Networking (ICPCSN)* pp.1056-58.
- Sultan, M. (29-30 November 2017) ' Angular and the Trending Frameworks of Mobile', *Future Technologies Conference (FTC)* pp. 929-930.
- Team, T.Q. (May 2023) *Qwik Documentation page*. Available at: <https://qwik.dev/docs/>.
- Tran, L. (October, 2020) 'Javascript and jQuery versus React in web development ', *Laurea University of Applied Sciences* pp.11-12.

# Appendix A

## EthOS

### START HERE - Basic Information

This form must be completed for all student projects.

#### Before you proceed

Some activities inherently involve increased risks or approval by external regulatory bodies, so a proportional ethics review is not recommended and a full ethical review may be required.

These may include:

- i. Approval from an external regulatory body (including, but not limited to: NHS (HRA), HMPPS etc.);
- ii. Misleading participants;
- iii. Research without the participants' consent;
- iv. Clinical procedures with participants;
- v. The ingestion or administration of any substance to participants by any means of delivery;
- vi. The use of novel techniques, even where apparently non-invasive, whose safety may be open to question;
- vii. The use of ionising radiation or exposure to radioactive materials;
- viii. Engaging in, witnessing, or monitoring criminal activity;
- ix. Engaging with, or accessing terrorism related materials;
- x. A requirement for security clearance to access participants, data or materials;
- xi. Physical or psychological risk to the participants or researcher;
- xii. The project activity takes place in a country outside of the UK for which there is currently an active travel warning issued by the authorities (see info button);
- xiii. Animals, animal tissue, new or existing human tissue, or biological toxins and agents;
- xiv. The sharing of participant personal data with a third party, regardless of the form under which the data is presented.

**If any of these activities are fundamental to your project, please contact your supervisor to determine if a full application is required.**

This form must be completed for each research project which you undertake at the University. It must be approved by your supervisor (where relevant) PRIOR to the start of any data collection.

In completing this form, please consult the University's [Research Ethics and Governance standards](#).

A1a Please confirm that you will abide by the University's Research Ethics and Governance standards in relation to this project.

☐ Yes

☐ No

#### A1b Data Protection

The University is responsible for complying with the UK General Data Protection Regulation whenever personal data is processed. Under the Data Protection Policy, all staff and students have a responsibility to comply with the regulation in their day-to-day activities. The first step you can take to understand these responsibilities is to review the [Data Protection in Research guidance pages](#) and complete the University's Mandatory Data Protection Training. Student training is available through Moodle (in the 'Skills Online' section – [please follow this link](#)). To make sure your knowledge is up to date, all staff and students must complete the training every two years. If you have any issues in accessing the data protection training or have any questions about the training, please contact [dataprotection@mmu.ac.uk](mailto:dataprotection@mmu.ac.uk).

Have you reviewed the Data Protection guidance pages and completed the Data Protection Training in the last two years?

☒ Yes

☐ No

A2 Are you submitting this application as a learning experience, for a unit which already has ethical approval? (please confirm with your supervisor)

☒ Yes

☐ No

A2.1 Approval reference (supplied by your supervisor)

#### A3 Student details

Title	First Name	Surname
<input type="text"/>	<input type="text" value="Deepak Gowda"/>	<input type="text" value="Nilavadi Rajamudi"/>
Email <input type="text" value="DEEPAK.G.NILAVADI-RAJAMUDI@stu.mmu.ac.uk"/>		

A3.1 Manchester Metropolitan University ID number

#### A4 Supervisor

Title	First Name	Surname
<input type="text" value="Dr"/>	<input type="text" value="Conor"/>	<input type="text" value="Muldoon"/>
Faculty <input type="text" value="Science and Engineering"/>		
Telephone <input type="text" value="+44 0161 247 1406"/>		
Email <input type="text" value="c.muldoon@mmu.ac.uk"/>		



A5 Which Faculty is responsible for the project?

Science and Engineering

A6 Course title

Masters Project (6G7V0007)

A7 Project title

Matchmaking online food ordering system

A8 What is the proposed start date of your project?

07/06/2024

A9 When do you expect to complete your project?

27/09/2024

A10 Please describe the overall aims of your project (3-4 sentences). Research questions should also be included here.

The aim of the project is to develop a unique online ordering platform for restaurants/takeaway business, which is user-friendly and robust. By developing this online ordering system, it enables customers to personalize their diet, donate food, support home chefs and local farmers, view menu, place orders, and make secure payments. It is not just about digital transformation of online ordering process but it's about enhancing customer experience, customer satisfaction, efficient operation and flexible business environment.

A11 Please describe the research activity

A lot of small and medium chain restaurants have been affected and faced challenges from large chain restaurants, therefore this project aims to develop an unique solution to remain in the challenging market, next-generation online food ordering system will help these small and medium chain restaurants to improve their service by providing customers an easier and efficient way of ordering food via website. In this project, the first task is to conduct research and analysis to find out the problems faced by customers in traditional food ordering systems, therefore it's necessary to conduct surveys to understand their requirements and expectations. The next step of the project is to design and develop an online ordering platform using that can be accessible via web, smartphones, tablets etc so it'll allow customers to view menu, add food items to cart, delivery options and choose payment methods along with these features like order tracking and real-time updates and the innovative features that will make this web application stand out from the existing ones are: 1) A feature that will allow customers who prefer to buy home-cooked meals from home chefs, a platform for those people who can't start their own restaurant business because of financial problems. 2) Another functionality is to let customers donate food to those in need and support the community, this also helps to reduce food waste. 3) Personalizing diet plans, this functionality in this web application will allow customers to choose meal based on their health goals and diet plans, it lets customers to choose vegetarian, non-vegetarian or vegan meal options. 4) The next functionality that will be added to the system is, supporting local farmers to sell their products and ingredients to customers, this feature will help to support the local economy

A12 Please provide details of the participants you intend to involve (please include information relating to the number involved and their demographics; the inclusion and exclusion criteria)

There would be no other participants involved.

A13 Please upload your project protocol

Type	Document Name	Documents			
		File Name	Version Date	Version	Size
Project Protocol	ToR	ToR.pdf	21/06/2024	1	127.4 KB

### Project Activity

B1 Are there any Health and Safety risks to the researcher and/or participants?

- ☐ Yes  
☒ No

B2 Please select any of the following which apply to your project

- ☐ Aspects involving human participants (including, but not limited to interviews, questionnaires, images, artefacts and social media data)  
☐ Aspects that the researcher or participants could find embarrassing or emotionally upsetting  
☐ Aspects that include culturally sensitive issues (e.g. age, gender, ethnicity etc.)  
☐ Aspects involving vulnerable groups (e.g. prisoners, pregnant women, children, elderly or disabled people, people experiencing mental health problems, victims of crime etc.), but does not require special approval from external bodies (NHS, security clearance, etc.)  
☐ Project activity which will take place in a country outside of the UK  
☒ None of the above

B2.4 Is this project being undertaken as part of a larger research study for which a Manchester Metropolitan application for ethical approval has already been granted or submitted?

- ☐ Yes  
☒ No

### Data

F1 How and where will data and documentation be stored?

As the data being collected is freely available, there is no need for secure storage. The documents would be simply stored on my laptop for ease of access.

F2 Will you be using personal data? Personal data is anything that can be used to identify a living individual, directly or indirectly. Pseudonymised data is still personal data.

- ☐ Yes
- ☒ No

#### Insurance

F3 Does your project involve:

- ☐ Pregnant persons as participants with procedures other than blood samples being taken from them? (see info button)
- ☐ Children aged five or under with procedures other than blood samples being taken from them? (see info button)
- ☐ Activities being undertaken by the lead investigator or any other member of the study team in a country outside of the UK as indicated in the info button? If 'Yes', please refer to the 'Travel Insurance' guidance on the info button
- ☐ Working with Hepatitis, Human T-Cell Lymphotropic Virus Type iii (HTLV iii), or Lymphadenopathy Associated Virus (LAV) or the mutants, derivatives or variations thereof or Acquired Immune Deficiency Syndrome (AIDS) or any syndrome or condition of a similar kind?
- ☐ Working with Transmissible Spongiform Encephalopathy (TSE), Creutzfeldt-Jakob Disease (CJD), variant Creutzfeldt-Jakob Disease (vCJD) or new variant Creutzfeldt-Jakob Disease (nvCJD)?
- ☐ Working in hazardous areas or high risk countries? (see info button)
- ☐ Working with hazardous substances outside of a controlled environment?
- ☐ Working with persons with a history of violence, substance abuse or a criminal record?
- ☒ None of the above

#### Additional Information

G1 Do you have any additional information or comments which have not been covered in this form?

- ☐ Yes
- ☒ No

G2 Do you have any additional documentation which you want to upload?

- ☐ Yes
- ☒ No

#### Signatures

H1 I confirm that all information in this application is accurate and true. I will not start this project until I have received Ethical Approval.

- ☒ I confirm

H2 Please notify your supervisor that this application is complete and ready to be submitted by clicking "Request" below. Do not begin your project until you have received confirmation from your supervisor - it is your responsibility to ensure that they do this.

**Signed:** This form was signed by Conor Muldoon (C.Muldoon@mmu.ac.uk) on 26/06/2024 17:41

H3 Have you been instructed by your supervisor to request a second signature for this application?

☐ Yes

☒ No

H4 By signing this application you are confirming that all details included in the form have been completed accurately and truthfully. You are also confirming that you will comply with all relevant UK data protection laws, and that that research data generated by the project will be securely archived in line with requirements specified by the University, unless specific legal, contractual, ethical or regulatory requirements apply.

**Signed:** This form was signed by Deepak Gowda Nilavadi Rajamudi (DEEPAK.G.NILAVADI-RAJAMUDI@stu.mmu.ac.uk) on 26/06/2024 13:11

# Project ToR

Project Topic: Matchmaking online food ordering system.

Course specific outcome:

This project allows the development of the following knowledge and skills

- The working practice of the software industry, and the typical tools and techniques employed in the software development process.
- The role of computer science in the development of the wider technological community and the relevant ethical and societal issues.
- Analyse, design, and implement algorithms using a range of appropriate languages and/or methodologies.
- Critical thinking and ability to communicate ideas and solutions both in writing and orally to specialist and non-specialist audiences.

Project description:

During COVID-19 pandemic, a lot of small and medium chain restaurants have been affected and faced challenges from large chain restaurants, therefore this project aims to develop a unique solution to remain in the challenging market, next-generation online food ordering system will help these small and medium chain restaurants to improve their service by providing customers an easier and efficient way of ordering food via website[1]. Food industry is relying on technology to enhance their business and interact with customers, this online platform is more efficient and aims to replace phone call, walk-in ordering system because they might cause wrong orders, miscommunications and are not reliable. [2]

In this project, the first task is to conduct research and analysis to find out the problems faced by customers in traditional food ordering systems, therefore it's necessary to conduct surveys to understand their requirements and expectations. The next step of the project is to design and develop an online ordering platform using that can be accessible via web, smartphones, tablets etc so it'll allow customers to view menu, add food items to cart, delivery options and choose payment methods along with these features like order tracking and real-time updates and the innovative features that will make this web application stand out from the existing ones are: 1) A feature that will allow customers who prefer to buy home-cooked meals from home chefs, a platform for those people who can't start their own restaurant business because of financial problems. 2) Another functionality is to let customers donate food to those in need and support the community, this also helps to reduce food waste. 3) Personalizing diet plans, this functionality in this web application will allow customers to choose meal based on their health goals and diet plans, it lets customers to choose vegetarian, non-vegetarian or vegan meal options. 4) The next functionality that will be added to the system is, supporting local farmers to sell their products and ingredients to customers, this feature will help to support the local economy.

Later, usability test will be conducted to ensure it meets customers' needs. Considering legal issues, users' data such as personal information, payment, should be secured safely and it's

important to follow data protection rules, food safety standards and online payment laws to ensure quality of food being delivered. Regarding social issues, have to design platform in such a way that customers who are disabled can access it and have to consider different cultural food in menu. Ethical issues such as obtaining customer data only for improving platform and getting their consent, also have to gain customer trust by providing best service. Considering professional issues, have to give customers best service by answering their questions, solving problems quickly, follow user experience design standards to achieve customer satisfaction, customer engagement and expectations. [3]

#### Project Aim:

The aim of the project is to develop a unique online ordering platform for restaurants/takeaway business by using novel technologies, which is user-friendly and robust (Rahul Sonwane, 2023). By developing this online ordering system, it enables customers to personalize their diet, donate food, support home chefs and local farmers, view menu, place orders, and make secure payments. It is not just about digital transformation of online ordering process but it's about enhancing customer experience, customer satisfaction, efficient operation and flexible business environment.

#### Objectives:

To achieve the aim, the objectives of this project are:

1. To develop a user-friendly online ordering platform that can be accessible through web and mobile devices.
  - Admin interface: Create an admin dashboard so that managers can receive orders, view customer opinion, and manage inventory, admin dashboard should provide instantaneous updates and help operations.
  - Customer interface: Create a customer interface so that it helps customers to place orders, book reservations, look at menu, select & add food items to cart, along with innovative features like food donations, home chefs, personalized diet plans, supporting local farmers and recommending customers their favourite cuisines, diet preferences by allowing users to create profiles where they can specify their food preferences.
2. By implementing safe verification: To secure user data and prevent unauthorized access it's necessary to execute security measures like encrypted passwords. Also have to make sure that admins and customers can log in securely to their interfaces.
3. Secure online payment: It should allow multiple payment methods so that the payment process will be easy, it should support payment options like debit cards, credit cards, online banking, and cash on delivery.
4. Creating user-friendly menu with high-quality visuals and item description. Also have to generate insights through order analytics and improve features like menu offerings, pricing.
5. Implementing location-based service, this allows customers to find and place orders from the nearest restaurant and this helps in upgrading service efficiency.

### Evaluation Plan:

In order to evaluate this project, first have to check how it works and how customers are satisfied with it, then get feedback from stakeholders who are involved in the project just to ensure the system is working well and easy to use while it's being developed. After the system is completely developed, it must be tested with customers and staff to see how it performs. Later, secure payment, customer satisfaction, accuracy of order will be evaluated through surveys, will also see how customers interact with the system through this data will be analysed, including customer feedback, time saved in processing orders, all these will help to improve the system and ensure objectives are achieved.

### Anticipated problems:

- Expectations of customers keep changing.
- Cybersecurity threats and customer data theft.
- Resolving customer issues and providing timely support, if it's not done customers might lose trust.

### Hardware Resources Required:

- Internet access
- Router
- Laptop or desktop
- Smartphone, tablet

### Software Resources Required:

- HTML
- CSS
- TypeScript, Qwik framework, Qwik City, Vite, Node.js, Express, MongoDB
- Performance monitoring and testing: Google Lighthouse, PageSpeed Insight.
- Tools: Wappalyzer.

### Schedule:

Task	Start date	End date	Duration (Days)
TOR & Ethics	07/06/2024	21/06/2024	14
Literature Review	10/06/2024	09/08/2024	60
Data Collection	10/06/2024	20/07/2024	40
Data Analysis	24/07/2024	23/08/2024	30
Report Writing	29/07/2024	17/09/2024	45

### References:

1. Sonwane, R., A. Deshmukh, and S. Choudhary, *Designing Web Application of Online Food Ordering for Restaurant Chain using Web Technologies*, in *2023 3rd International Conference on Pervasive Computing and Social Networking (ICPCSN)*. 2023. p. 1055-1058.

2. Ferdianto, F., et al., *Development of Mobile Application for Pre Order Food and Beverage*, in *2021 International Conference on Information Management and Technology (ICIMTech)*. 2021. p. 177-182.
3. Sam, Y.H., P.H. Leong, and C.F. Ku, *The Implementation of Mobile Application Ordering System to Optimize The User Experience of Food and Beverage Industry*, in *2023 IEEE 14th Control and System Graduate Research Colloquium (ICSGRC)*. 2023. p. 22-26.



## Appendix B

The link provided here is a Google drive link that has the source code for my Qwik project (Matchmaking online food ordering system)

[https://drive.google.com/drive/folders/1TPWr6wI4jlJJEL68RFphCJOAk6\\_Ylbqn?usp=sharing](https://drive.google.com/drive/folders/1TPWr6wI4jlJJEL68RFphCJOAk6_Ylbqn?usp=sharing)

To run the project in Visual Studio Code, select open in integrated terminal:

Backend - npm run server

Admin - npm run dev

Frontend - npm run dev